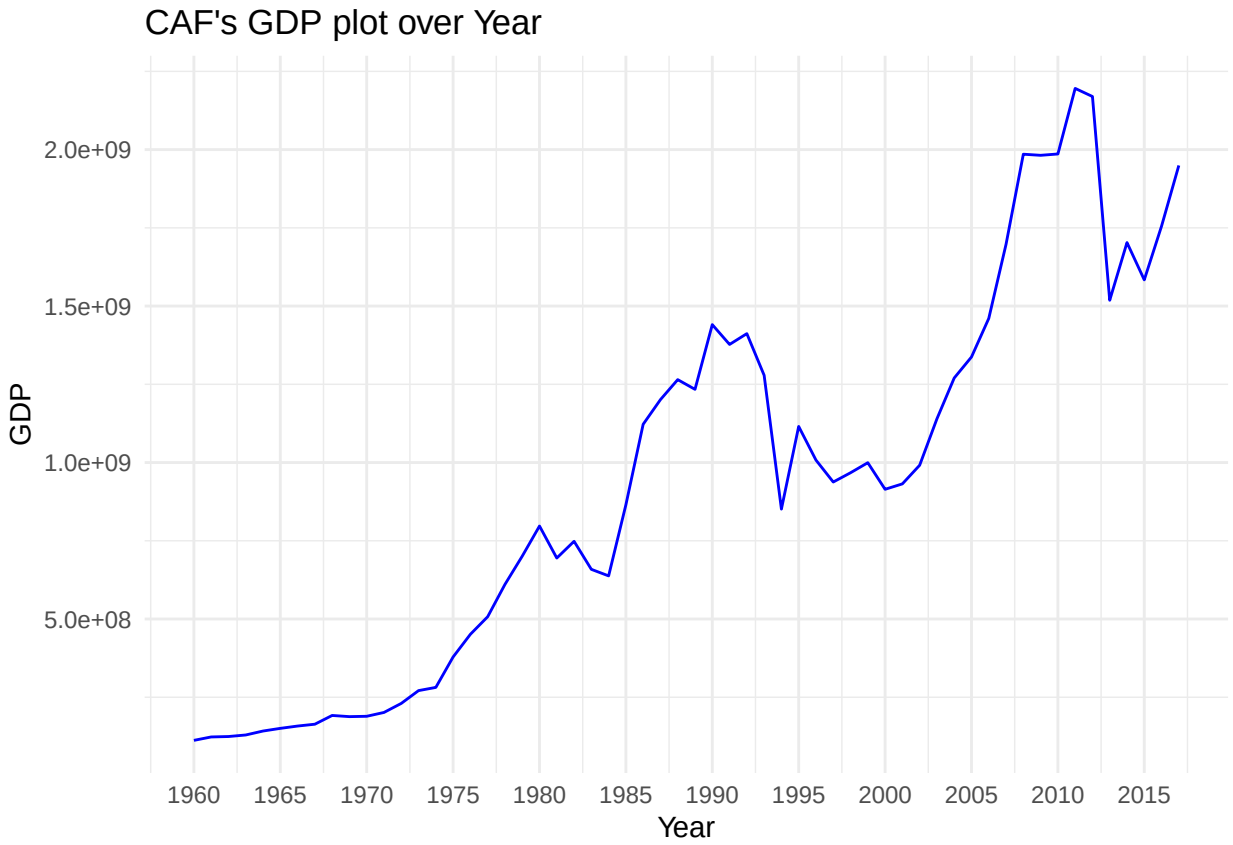
STA137-Final Project

Johnson Tian

2024-06-6

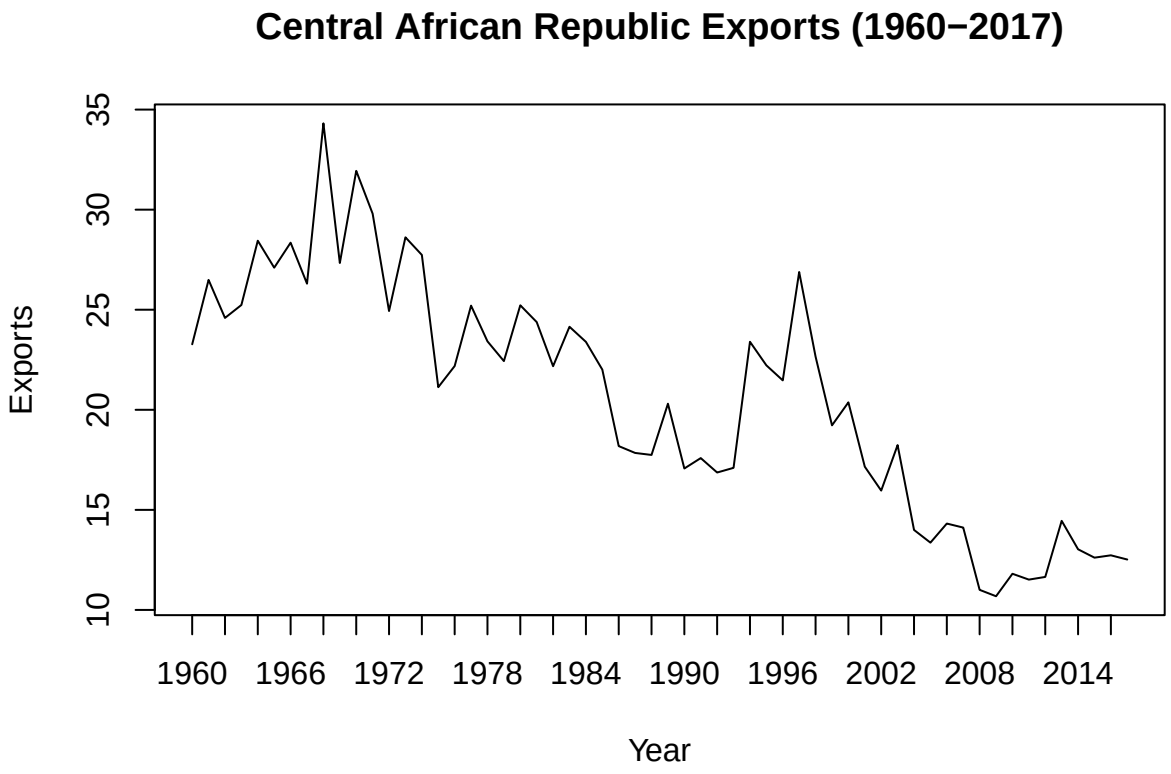
```
# GDP v.s. Year Plot
ggplot(finalPro_data, aes(x = Year, y = GDP)) +
  geom_line(color = "blue") +
  labs(title = "CAF's GDP plot over Year", x = "Year", y = "GDP") +
  scale_x_continuous(breaks = seq(min(finalPro_data$Year), max(finalPro_data$Year), by = 5)) +
  theme_minimal()
```



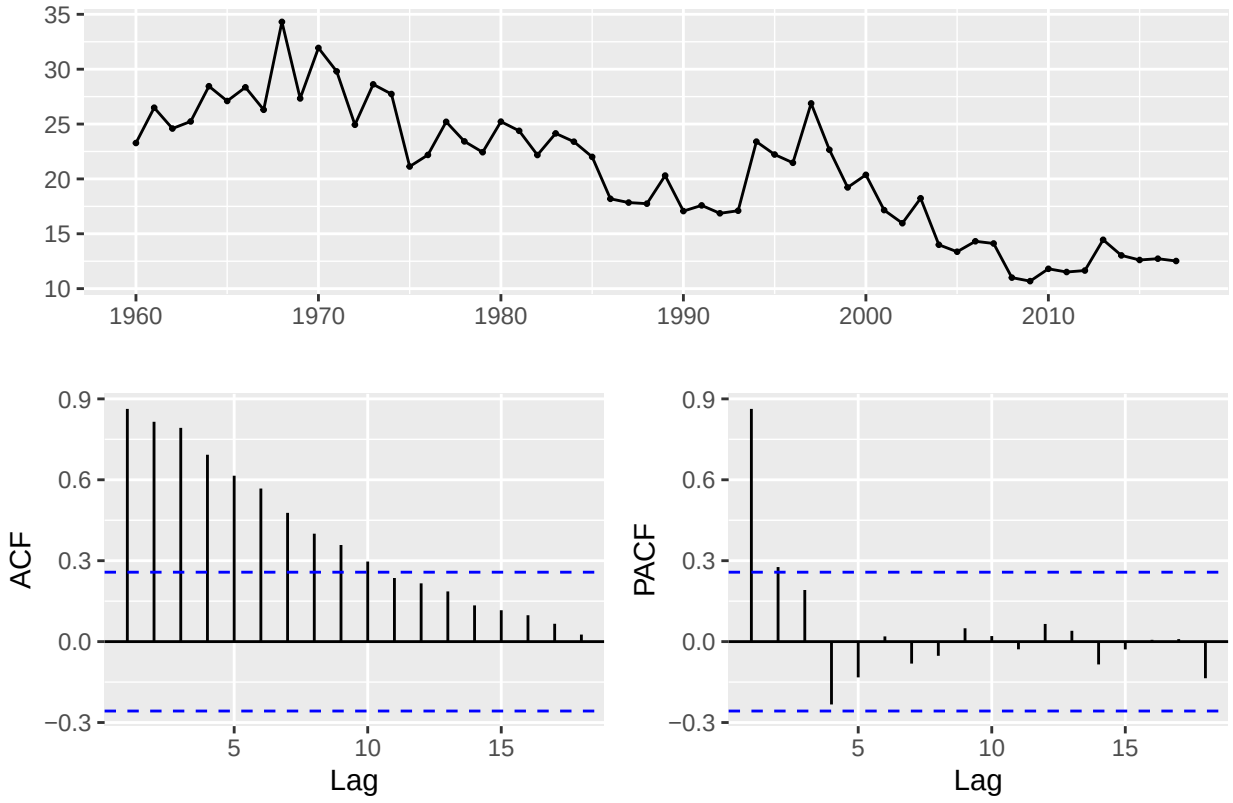

```
# Economic Growth v.s Year Plot
ggplot(finalPro_data, aes(x = Year, y = Growth)) +
  geom_line(color = "green") +
  labs(title = "CAF's Economic Growth plot over Year", x = "Year", y = "Growth Rate") +
  scale_x_continuous(breaks = seq(min(finalPro_data$Year), max(finalPro_data$Year), by = 5)) +
  theme_minimal()
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```


```r
# Create the time series object
tts_fp <- ts(finalPro_data$Exports, start = min(finalPro_data$Year), end = max(finalPro_data$Year), fre

# Plot the time series
plot(tts_fp, main = "Central African Republic Exports (1960-2017)", xlab = "Year", ylab = "Exports", xa
years <- seq(start(tts_fp)[1], end(tts_fp)[1], by = 2)
axis(1, at = years, labels = years)
```




```
#ACF&PACF
ggtsdisplay(tts_fp, plot.type = "partial")
```



```r
#ADF test
adf_test <- adf.test(tts_fp)
print(adf_test)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  tts_fp
## Dickey-Fuller = -3.1744, Lag order = 3, p-value = 0.1006
## alternative hypothesis: stationary
```

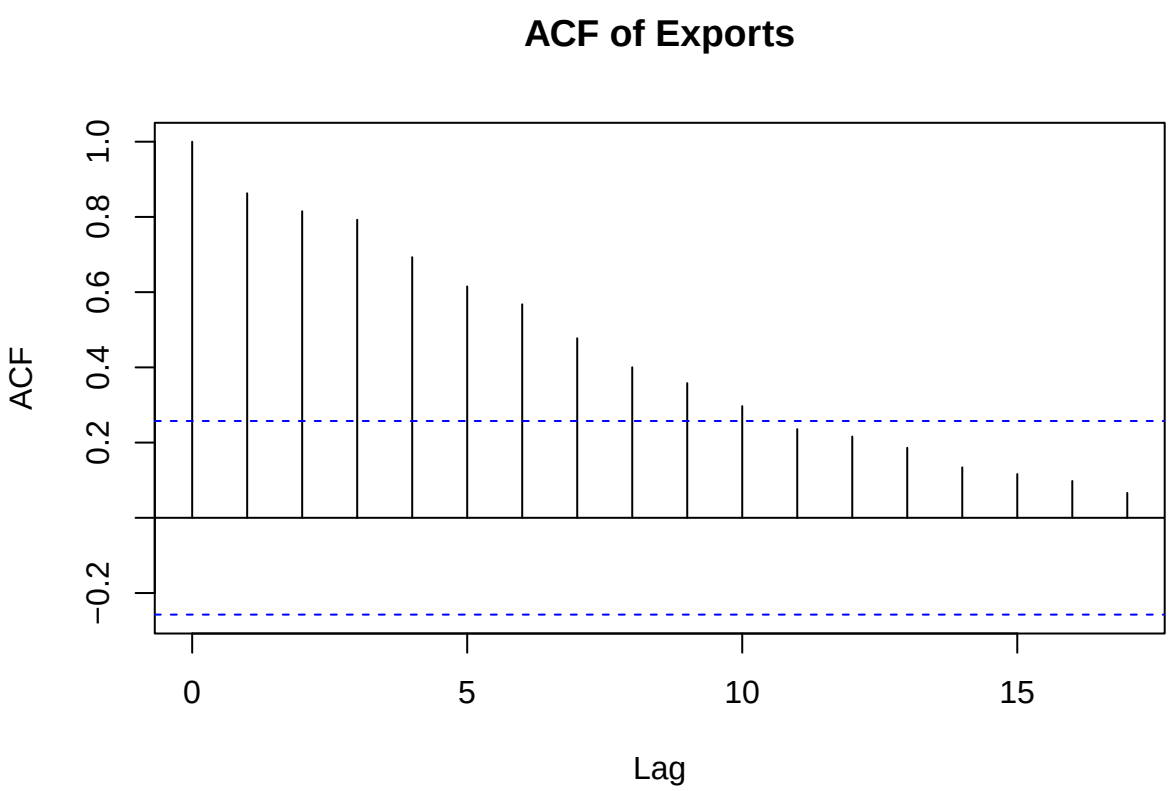
```r
#KPSS test
kpss_test <- kpss.test(tts_fp)
```

```
## Warning in kpss.test(tts_fp): p-value smaller than printed p-value
```

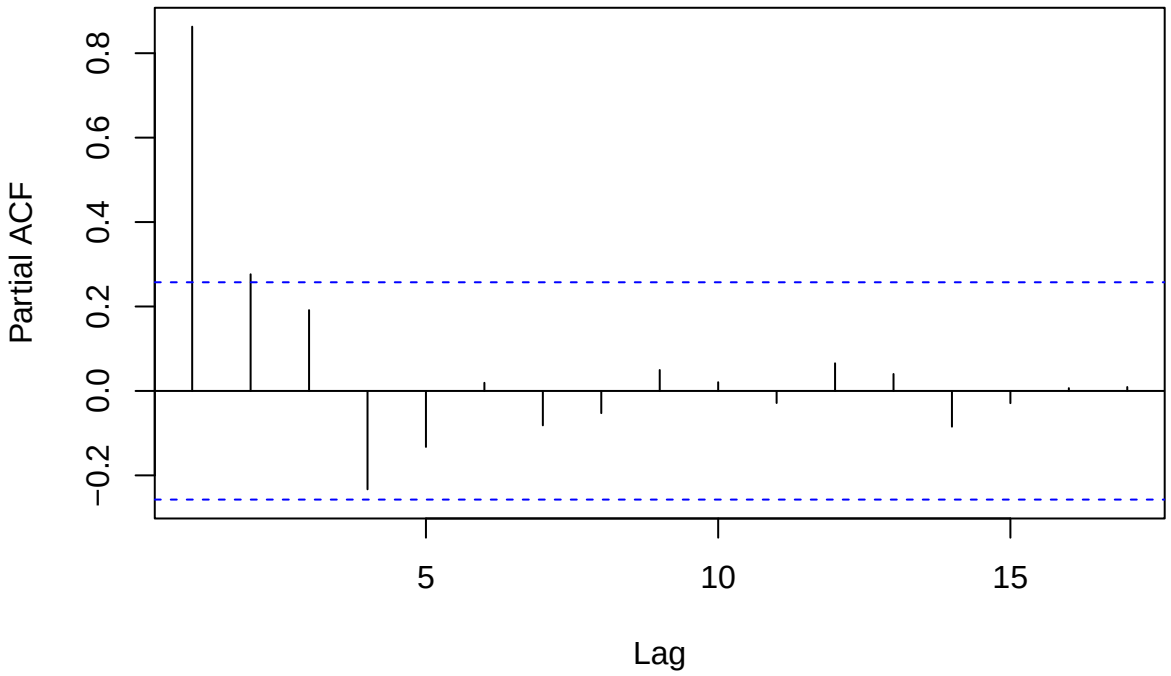
```r
print(kpss_test)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tts_fp
## KPSS Level = 1.2824, Truncation lag parameter = 3, p-value = 0.01
```

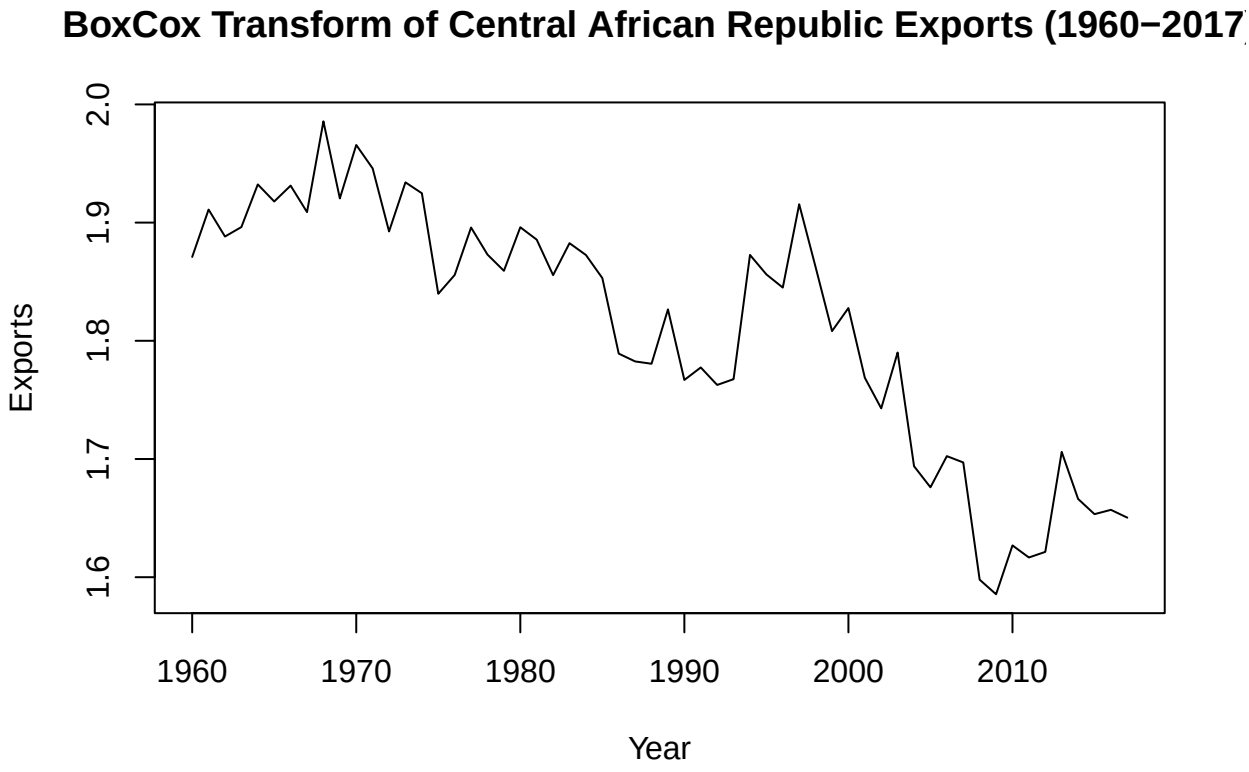
```r
acf(tts_fp, main = "ACF of Exports")
```



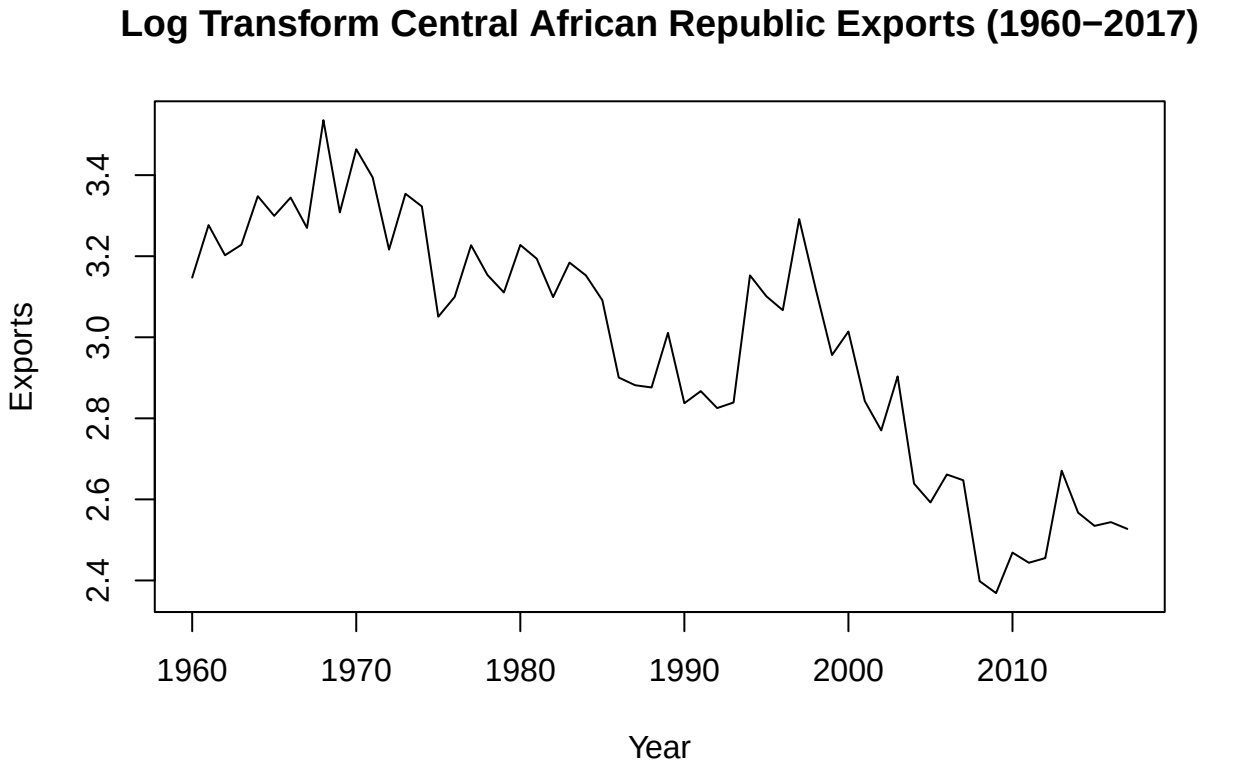

```r
pacf(tts_fp, main = "PACF of Exports")
```




```
# best lamda
lambda <- BoxCox.lambda(tts_fp)

# Box-Cox Transform
tts_fp_boxcox <- BoxCox(tts_fp, lambda)
# log Transform
tts_fp_log <- log(tts_fp)
plot(tts_fp_boxcox, main = "BoxCox Transform of Central African Republic Exports (1960-2017)", xlab = "
```




```
plot(tts_fp_log, main = "Log Transform Central African Republic Exports (1960-2017)", xlab = "Year", yla
```




```
adf_test_tts_fp_boxcox <- adf.test(tts_fp_boxcox)
print(adf_test_tts_fp_boxcox)
```

```
##
```

```
##  Augmented Dickey-Fuller Test
##
## data:  tts_fp_boxcox
## Dickey-Fuller = -2.8586, Lag order = 3, p-value = 0.2281
## alternative hypothesis: stationary
```

```
kpss_test_tts_fp_boxcox <- kpss.test(tts_fp_boxcox)
```

```
## Warning in kpss.test(tts_fp_boxcox): p-value smaller than printed p-value
```

```
print(kpss_test_tts_fp_boxcox)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tts_fp_boxcox
## KPSS Level = 1.2639, Truncation lag parameter = 3, p-value = 0.01
```

```
adf_test_tts_fp_log <- adf.test(tts_fp_log)
print(adf_test_tts_fp_log)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  tts_fp_log
## Dickey-Fuller = -2.9724, Lag order = 3, p-value = 0.1821
## alternative hypothesis: stationary
```

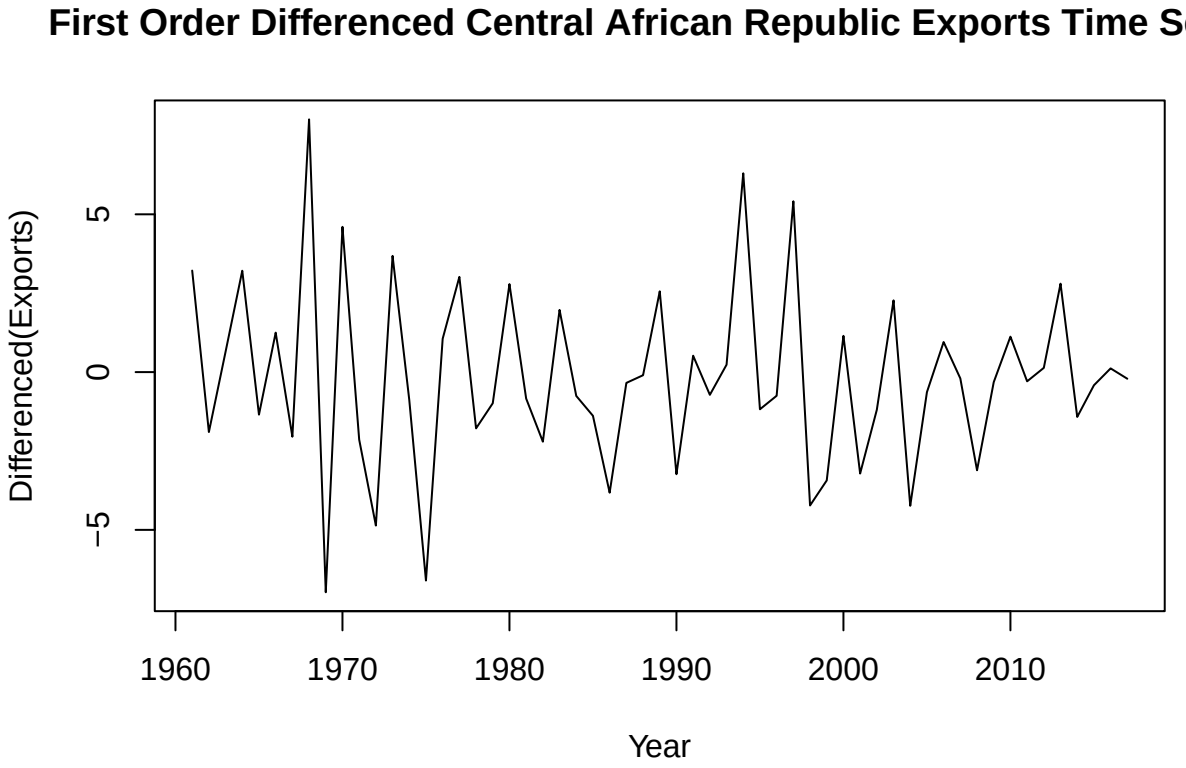
```
kpss_test_tts_fp_log <- kpss.test(tts_fp_log)
```

```
## Warning in kpss.test(tts_fp_log): p-value smaller than printed p-value
```

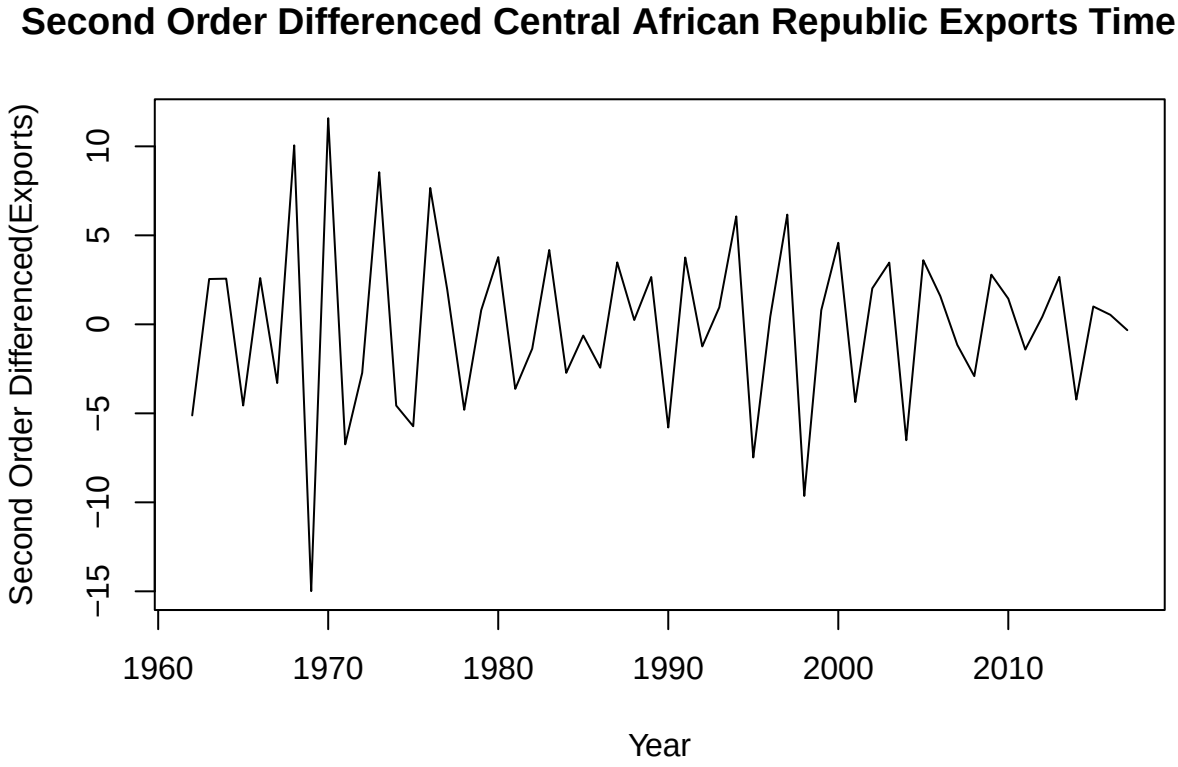
```
print(kpss_test_tts_fp_log)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  tts_fp_log
## KPSS Level = 1.276, Truncation lag parameter = 3, p-value = 0.01
```

```
# First Order Differencing
diff_tts_fp <- diff(tts_fp)

# Plot First Order Differenced Time Series
plot(diff_tts_fp, main = "First Order Differenced Central African Republic Exports Time Series", ylab =
```




```
# If higher order differencing is needed, such as second order differencing
diff2_tts_fp <- diff(diff_tts_fp)

# Plot Second Order Differenced Time Series
plot(diff2_tts_fp, main = "Second Order Differenced Central African Republic Exports Time Series", ylab
```



```r
acf_plot_diff <- ggAcf(diff_tts_fp, main = "ACF of First Differenced Series")
```

```
## Warning in ggplot2::geom_segment(lineend = "butt", ...): Ignoring unknown
## parameters: `main`
```

```r
pacf_plot_diff <- ggPacf(diff_tts_fp, main = "PACF of First Differenced Series")
```

```
## Warning in ggplot2::geom_segment(lineend = "butt", ...): Ignoring unknown
## parameters: `main`
```

```r
adf_test_diff_tts_fpx <- adf.test(diff_tts_fp)
print(adf_test_diff_tts_fpx)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff_tts_fp
## Dickey-Fuller = -3.2518, Lag order = 3, p-value = 0.08812
## alternative hypothesis: stationary
```

```r
kpss_test_diff_tts_fp <- kpss.test(diff_tts_fp)
```

```
## Warning in kpss.test(diff_tts_fp): p-value greater than printed p-value
```

```r
print(kpss_test_diff_tts_fp)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff_tts_fp
## KPSS Level = 0.092232, Truncation lag parameter = 3, p-value = 0.1
```

```r
diff2_tts_fp <- diff(tts_fp, differences = 2)


adf_test_diff2_tts_fpx <- adf.test(diff2_tts_fp)
```

```
## Warning in adf.test(diff2_tts_fp): p-value smaller than printed p-value
```

```r
print(adf_test_diff2_tts_fpx)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff2_tts_fp
## Dickey-Fuller = -5.883, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

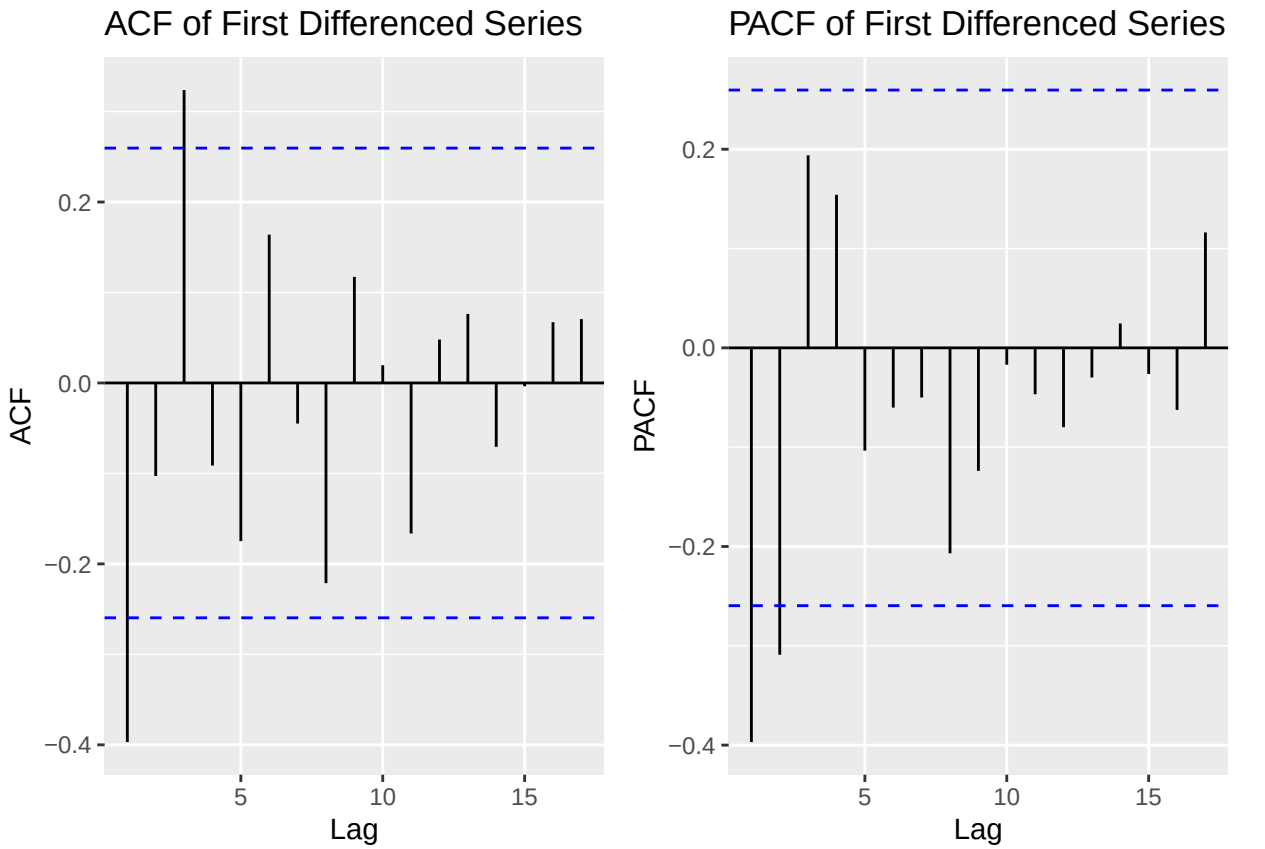
```r
kpss_test_diff2_tts_fp <- kpss.test(diff2_tts_fp)
```

```
## Warning in kpss.test(diff2_tts_fp): p-value greater than printed p-value
```

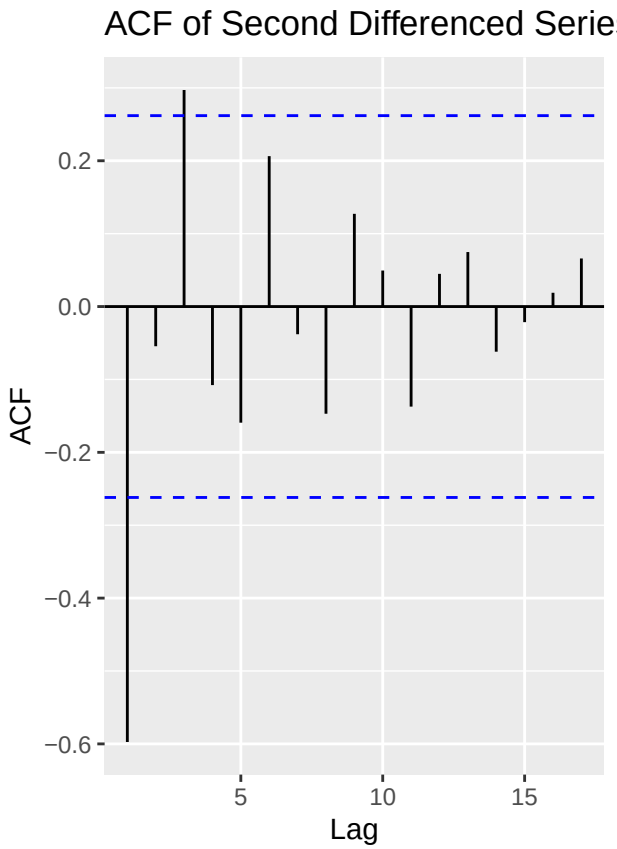
```r
print(kpss_test_diff2_tts_fp)
```

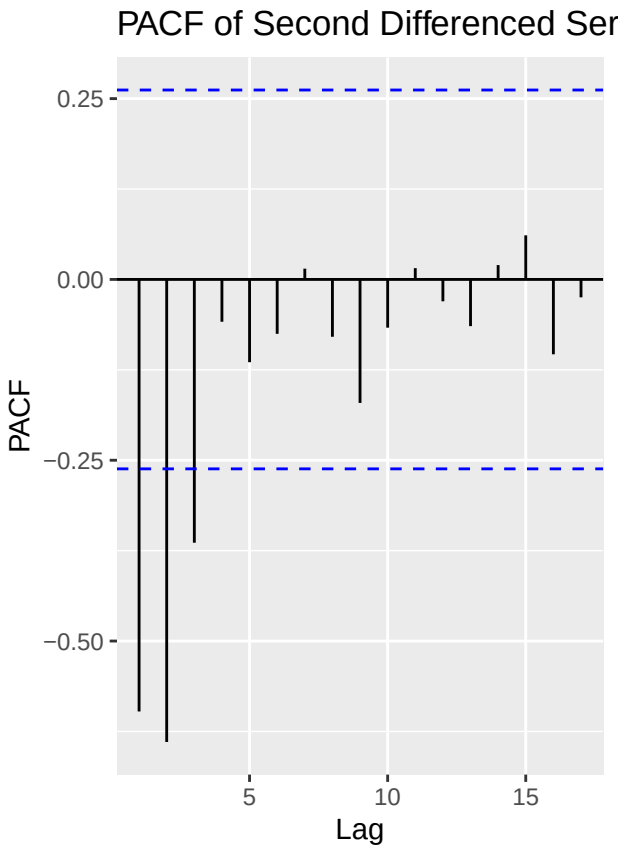
```
##
##  KPSS Test for Level Stationarity
##
## data:  diff2_tts_fp
## KPSS Level = 0.04633, Truncation lag parameter = 3, p-value = 0.1
```

```
acf_plot_diff <- ggAcf(diff_tts_fp) + ggtitle("ACF of First Differenced Series")
pacf_plot_diff <- ggPacf(diff_tts_fp) + ggtitle("PACF of First Differenced Series")


grid.arrange(acf_plot_diff, pacf_plot_diff, ncol = 2)
```




```
diff2_tts_fp <- diff(tts_fp, differences = 2)

acf_plot_diff2 <- ggAcf(diff2_tts_fp) + ggtitle("ACF of Second Differenced Series")
pacf_plot_diff2 <- ggPacf(diff2_tts_fp) + ggtitle("PACF of Second Differenced Series")

grid.arrange(acf_plot_diff2, pacf_plot_diff2, ncol = 2)
```






```r
# Select ARIMA parameter
arima_2_2_0 <- Arima(tts_fp, order = c(2, 2, 0))
arima_0_2_2 <- Arima(tts_fp, order = c(0, 2, 2))
arima_2_2_2 <- Arima(tts_fp, order = c(2, 2, 2))
arima_2_1_0 <- Arima(tts_fp, order = c(2, 1, 0))
arima_0_1_3 <- Arima(tts_fp, order = c(0, 1, 3))
arima_1_1_1 <- Arima(tts_fp, order = c(1, 1, 1))

# auto select
auto_arima_model <- auto.arima(tts_fp)
auto_arima_model_stepwise <- auto.arima(tts_fp, stepwise = TRUE, trace = TRUE)
```

```
##
##  ARIMA(2,1,2) with drift         : 277.0579
##  ARIMA(0,1,0) with drift         : 287.0319
##  ARIMA(1,1,0) with drift         : 279.397
##  ARIMA(0,1,1) with drift         : 277.2738
##  ARIMA(0,1,0)                    : 285.1246
##  ARIMA(1,1,2) with drift         : 279.1568
##  ARIMA(2,1,1) with drift         : 277.4614
##  ARIMA(3,1,2) with drift         : Inf
##  ARIMA(2,1,3) with drift         : Inf
##  ARIMA(1,1,1) with drift         : 279.1303
##  ARIMA(1,1,3) with drift         : 278.8484
##  ARIMA(3,1,1) with drift         : 278.39
##  ARIMA(3,1,3) with drift         : Inf
##  ARIMA(2,1,2)                    : 275.3732
```

```
##  ARIMA(1,1,2)                    : 277.2802
##  ARIMA(2,1,1)                    : 276.2026
##  ARIMA(3,1,2)                    : Inf
##  ARIMA(2,1,3)                    : Inf
##  ARIMA(1,1,1)                    : 277.88
##  ARIMA(1,1,3)                    : 276.8856
##  ARIMA(3,1,1)                    : 276.5309
##  ARIMA(3,1,3)                    : Inf
##
##  Best model: ARIMA(2,1,2)
```

```
models <- list(arima_2_2_0, arima_0_2_2, arima_2_2_2, arima_2_1_0, arima_0_1_3, arima_1_1_1, auto_arima
model_names <- c("ARIMA(2,2,0)", "ARIMA(0,2,2)", "ARIMA(2,2,2)", "ARIMA(2,1,0)", "ARIMA(0,1,3)", "ARIMA

results <- data.frame(
  Model = model_names,
  AICc = sapply(models, AICc)
)

head(results)
```

```
##          Model     AICc
## 1 ARIMA(2,2,0) 287.1248
## 2 ARIMA(0,2,2) 278.6766
## 3 ARIMA(2,2,2) 278.9076
## 4 ARIMA(2,1,0) 274.9897
## 5 ARIMA(0,1,3) 275.0169
## 6 ARIMA(1,1,1) 277.8800
```

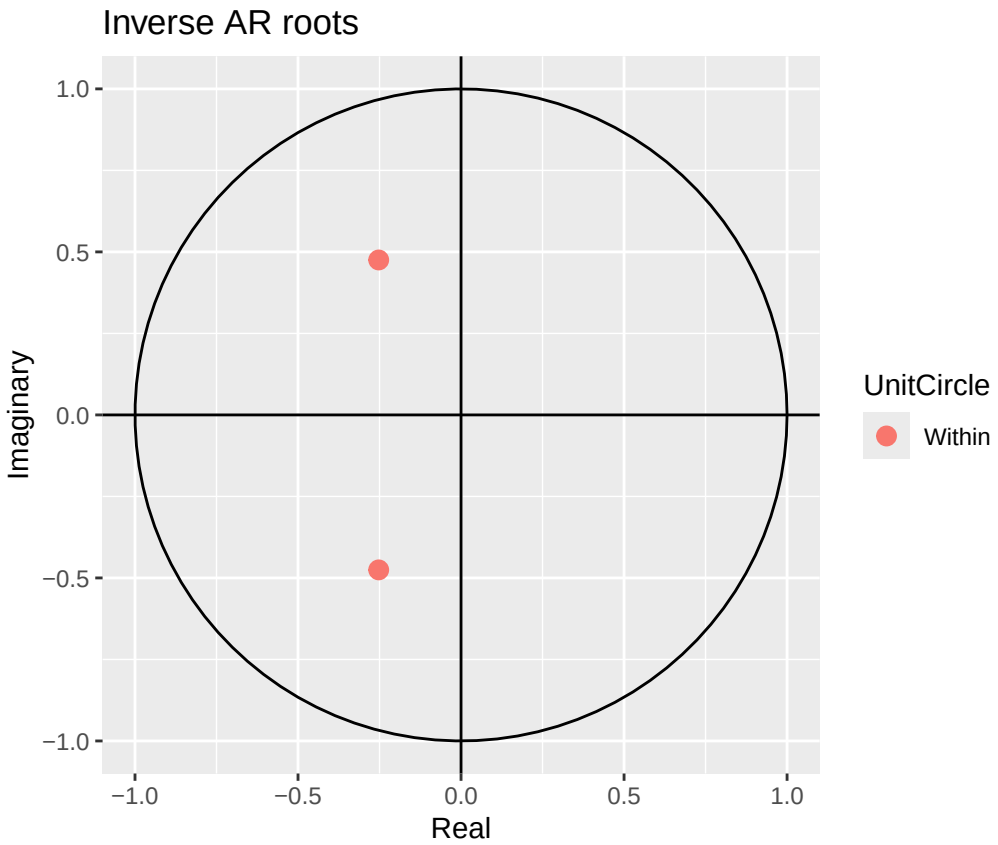
```
write.csv(results, file = "results_final.csv", row.names = FALSE)

# select top less 4 aicc models
top_models <- results[order(results$AICc)[1:3], "Model"]
top_models_list <- models[match(top_models, model_names)]

#check roots available
print (top_models_list)
```

```
## [[1]]
## Series: tts_fp
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1      ar2
##       -0.5050  -0.2897
## s.e.   0.1266   0.1254
##
## sigma^2 = 6.706:  log likelihood = -134.27
## AIC=274.54   AICc=274.99   BIC=280.67
##
## [[2]]
## Series: tts_fp
## ARIMA(0,1,3)
##
## Coefficients:
```

```
##           ma1     ma2     ma3
##       -0.4459  0.0932  0.2748
## s.e.   0.1309  0.1509  0.1333
##
## sigma^2 = 6.539:  log likelihood = -133.12
## AIC=274.25   AICc=275.02   BIC=282.42
##
## [[3]]
## Series: tts_fp
## ARIMA(2,1,2)
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -0.6741  -0.7142  0.2468  0.4831
## s.e.   0.1821   0.2037  0.2531  0.2576
##
## sigma^2 = 6.416:  log likelihood = -132.1
## AIC=274.2   AICc=275.37   BIC=284.41
```

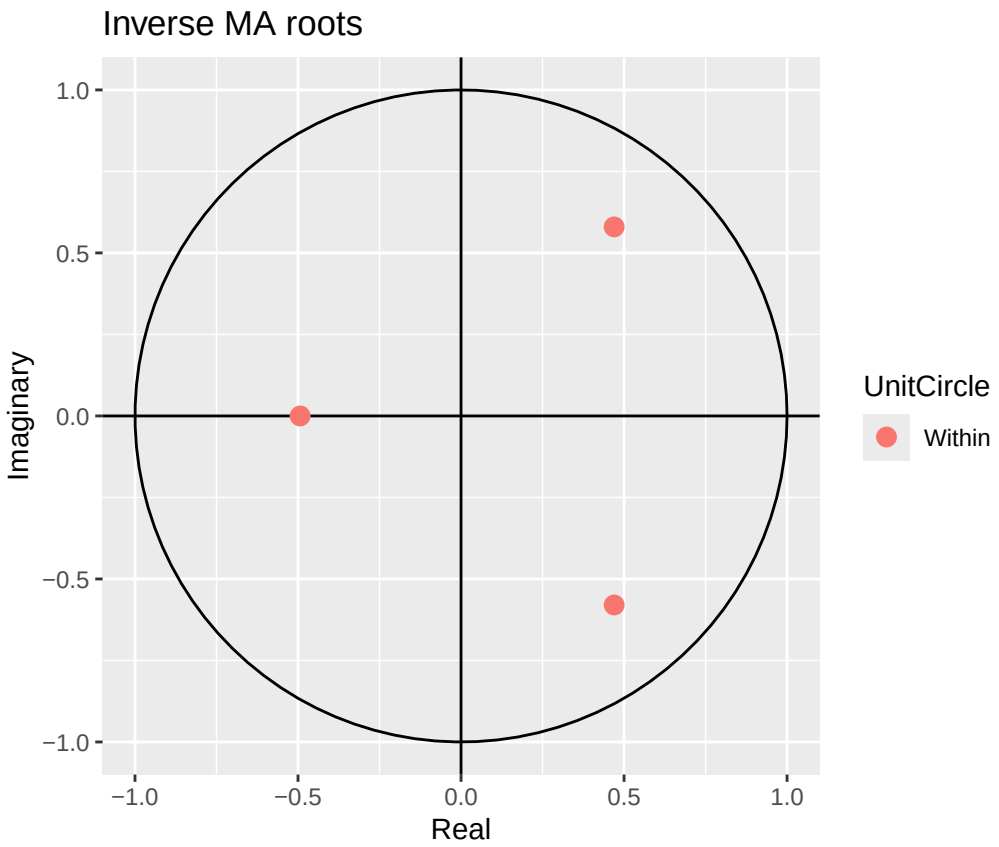
```
for (i in 1:3) {
  cat("Roots of characteristic polynomial for model:", top_models[i], "\n")
  print(autoplot(top_models_list[[i]]))
  cat("\n\n")
}
```

```
## Roots of characteristic polynomial for model: ARIMA(2,1,0)
```

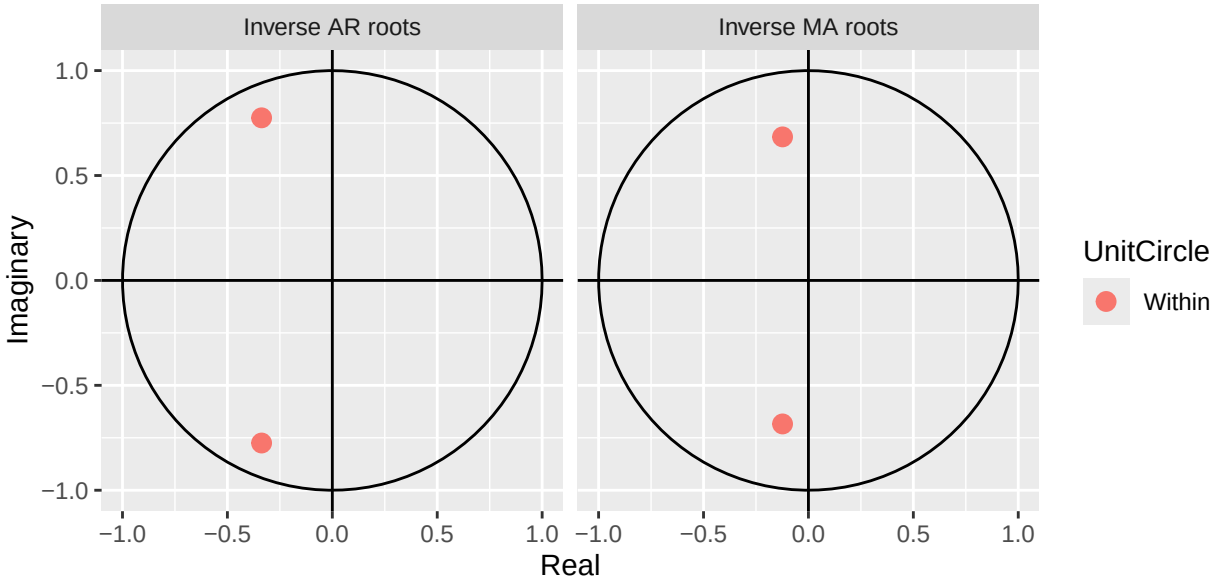



```
##
```

```
##
## Roots of characteristic polynomial for model: ARIMA(0,1,3)
```



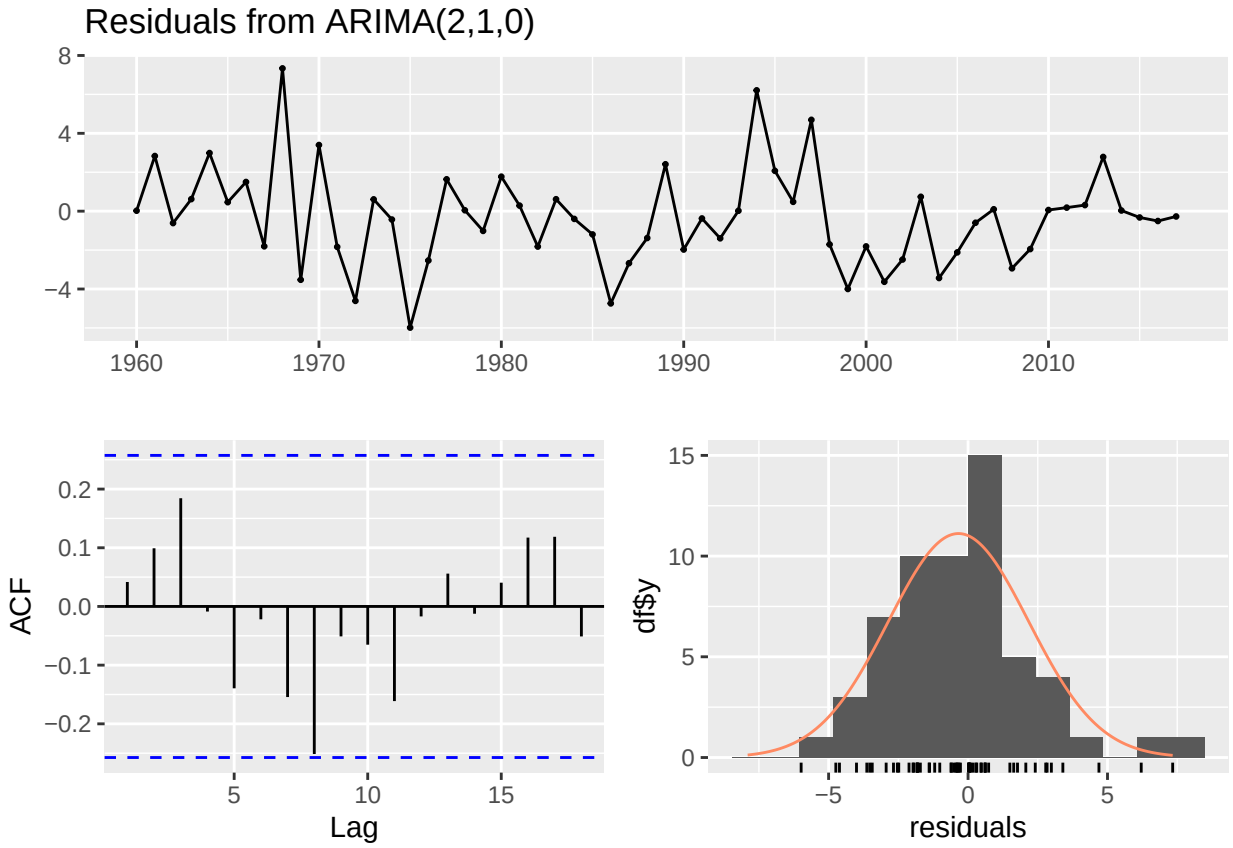

```
##
##
## Roots of characteristic polynomial for model: AUTO ARIMA
```



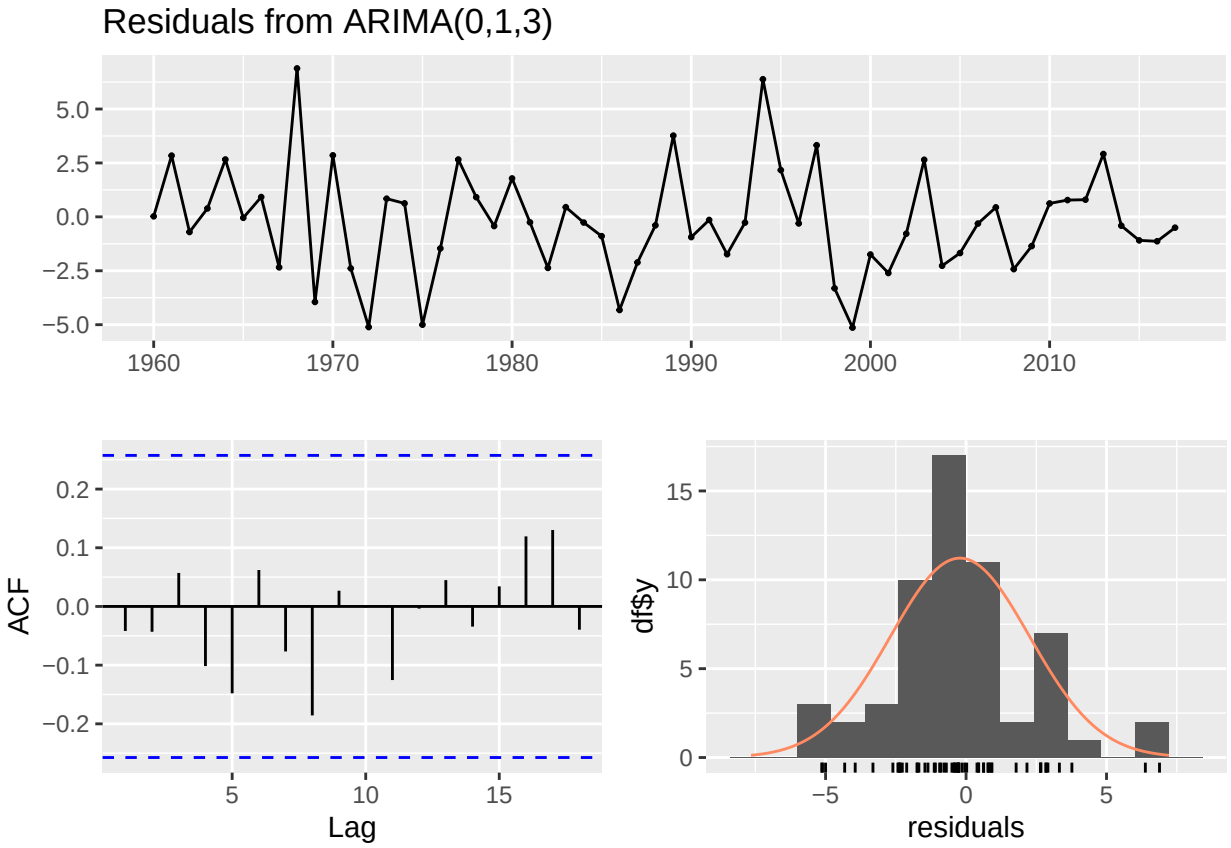

```r
top_models <- results[order(results$AICc)[1:4], "Model"]
top_models_list <- models[match(top_models, model_names)]
```

```r
for (i in 1:3) {
  cat("Summary of model:", top_models[i], "\n")
  print(summary(top_models_list[[i]]))

  checkresiduals(top_models_list[[i]])

  autoplot(top_models_list[[i]])
  cat("\n\n")
}
```

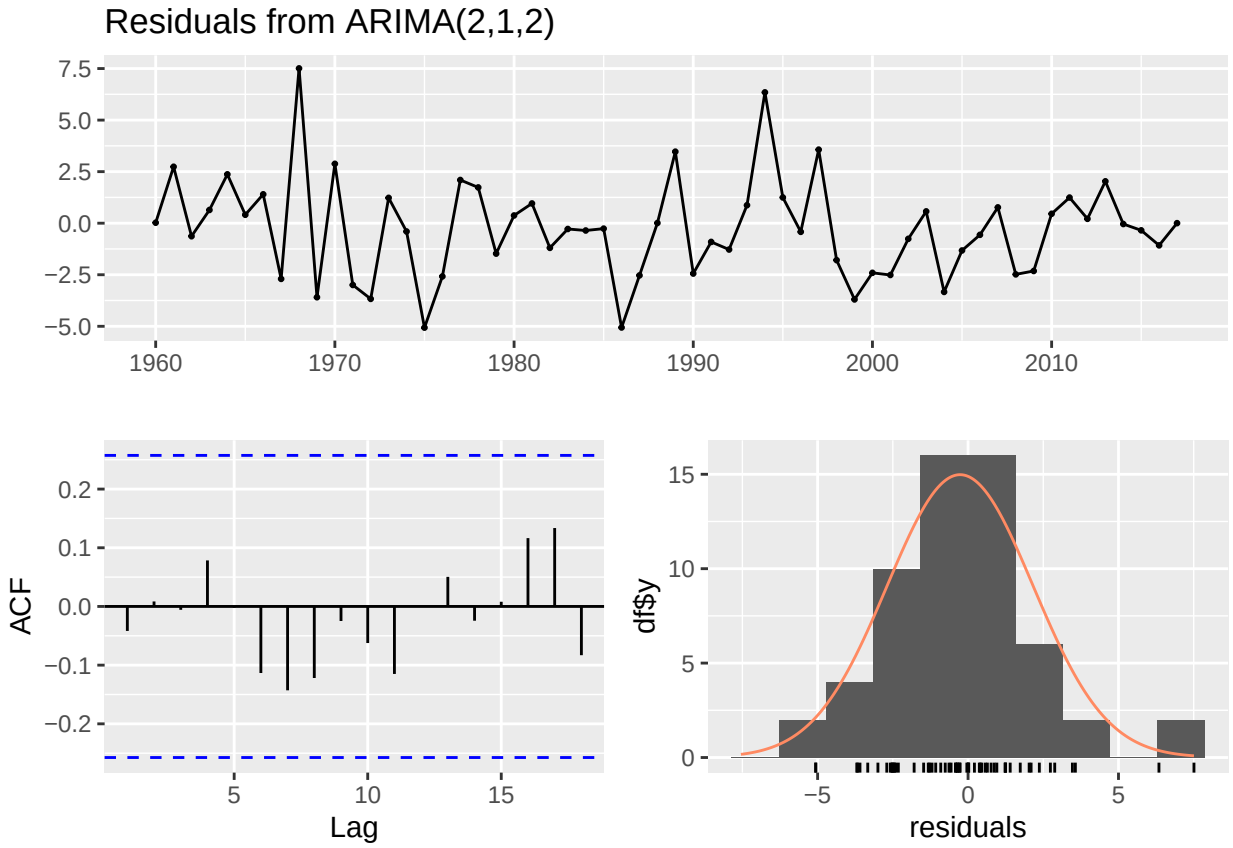
```
## Summary of model: ARIMA(2,1,0)
## Series: tts_fp
## ARIMA(2,1,0)
##
## Coefficients:
##           ar1      ar2
##       -0.5050  -0.2897
## s.e.   0.1266   0.1254
##
## sigma^2 = 6.706:  log likelihood = -134.27
## AIC=274.54   AICc=274.99   BIC=280.67
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.3425155 2.521754 1.867171 -2.968058 9.136747 0.8526358
##                     ACF1
## Training set 0.04169984
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,0)
## Q* = 10.698, df = 8, p-value = 0.2194
##
## Model df: 2.   Total lags used: 10
##
##
##
## Summary of model: ARIMA(0,1,3)
## Series: tts_fp
## ARIMA(0,1,3)
##
## Coefficients:
##           ma1     ma2     ma3
##       -0.4459  0.0932  0.2748
## s.e.   0.1309  0.1509  0.1333
##
## sigma^2 = 6.539:  log likelihood = -133.12
## AIC=274.25   AICc=275.02   BIC=282.42
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.2153469 2.467434 1.859567 -1.974757 9.117778 0.8491637
##                      ACF1
## Training set -0.04174063
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,3)
## Q* = 5.6397, df = 7, p-value = 0.5824
##
## Model df: 3.   Total lags used: 10
##
##
##
## Summary of model: AUTO ARIMA
## Series: tts_fp
## ARIMA(2,1,2)
##
## Coefficients:
##           ar1      ar2     ma1     ma2
##       -0.6741  -0.7142  0.2468  0.4831
## s.e.   0.1821   0.2037  0.2531  0.2576
##
## sigma^2 = 6.416:  log likelihood = -132.1
## AIC=274.2   AICc=275.37   BIC=284.41
##
## Training set error measures:
##                       ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -0.2645835 2.421275 1.821659 -2.327502 8.929045 0.8318532
##                      ACF1
## Training set -0.04168949
```



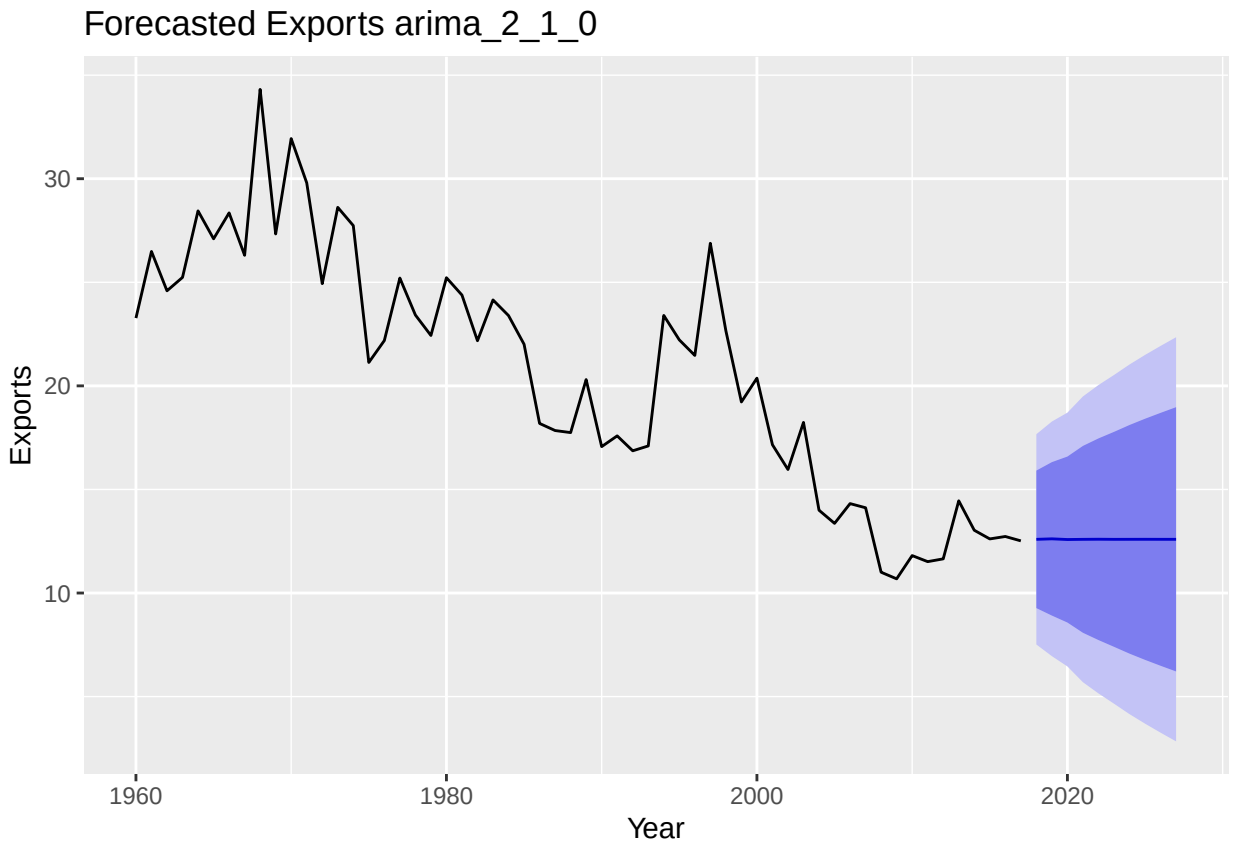

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)
## Q* = 4.122, df = 6, p-value = 0.6602
##
## Model df: 4.   Total lags used: 10
```

```r
#forecasting for next 10 years
forecast_horizon <- 10
forecast_result <- forecast(arima_2_1_0, h = forecast_horizon)

print(forecast_result)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2018       12.59070 9.271976 15.90943 7.515148 17.66626
## 2019       12.61514 8.912140 16.31813 6.951892 18.27838
## 2020       12.58176 8.575977 16.58755 6.455443 18.70808
## 2021       12.59154 8.081925 17.10115 5.694679 19.48840
## 2022       12.59627 7.729464 17.46307 5.153133 20.03941
## 2023       12.59105 7.403960 17.77814 4.658081 20.52402
## 2024       12.59232 7.074250 18.11038 4.153163 21.03147
## 2025       12.59319 6.773812 18.41256 3.693221 21.49315
## 2026       12.59238 6.489037 18.69572 3.258123 21.92664
## 2027       12.59254 6.213798 18.97127 2.837098 22.34797
```

```r
autoplot(forecast_result) +
  ggtitle("Forecasted Exports arima_2_1_0") +
```

```
  xlab("Year") +
  ylab("Exports")
```




```
tts_fp <- ts(finalPro_data$Exports, start = min(finalPro_data$Year), frequency = 2)
print(tts_fp)
```

```
## Time Series:
## Start = c(1960, 1)
## End = c(1988, 2)
## Frequency = 2
##  [1] 23.27272 26.49007 24.59017 25.23659 28.44827 27.10027 28.35052 26.30273
##  [9] 34.31230 27.33466 31.93850 29.79923 24.93480 28.61804 27.73682 21.13035
## [17] 22.18687 25.20440 23.42032 22.43257 25.22155 24.38345 22.17775 24.14900
## [25] 23.39399 22.00852 18.18643 17.84414 17.74475 20.30389 17.06747 17.58667
## [33] 16.86570 17.09955 23.39941 22.22094 21.47039 26.88428 22.65689 19.22365
## [41] 20.37221 17.15725 15.96270 18.23600 13.99792 13.36275 14.31601 14.11533
## [49] 11.00366 10.68442 11.80725 11.51483 11.64916 14.45149 13.03009 12.61192
## [57] 12.72904 12.51809
```

```
decomposed_ts <- decompose(tts_fp)
print(decomposed_ts)
```

```
## $x
## Time Series:
## Start = c(1960, 1)
## End = c(1988, 2)
## Frequency = 2
##  [1] 23.27272 26.49007 24.59017 25.23659 28.44827 27.10027 28.35052 26.30273
```

```
##  [9] 34.31230 27.33466 31.93850 29.79923 24.93480 28.61804 27.73682 21.13035
## [17] 22.18687 25.20440 23.42032 22.43257 25.22155 24.38345 22.17775 24.14900
## [25] 23.39399 22.00852 18.18643 17.84414 17.74475 20.30389 17.06747 17.58667
## [33] 16.86570 17.09955 23.39941 22.22094 21.47039 26.88428 22.65689 19.22365
## [41] 20.37221 17.15725 15.96270 18.23600 13.99792 13.36275 14.31601 14.11533
## [49] 11.00366 10.68442 11.80725 11.51483 11.64916 14.45149 13.03009 12.61192
## [57] 12.72904 12.51809
##
## $seasonal
## Time Series:
## Start = c(1960, 1)
## End = c(1988, 2)
## Frequency = 2
##  [1]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
##  [7]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [13]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [19]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [25]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [31]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [37]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [43]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [49]  0.02880065 -0.02880065  0.02880065 -0.02880065  0.02880065 -0.02880065
## [55]  0.02880065 -0.02880065  0.02880065 -0.02880065
##
## $trend
## Time Series:
## Start = c(1960, 1)
## End = c(1988, 2)
## Frequency = 2
##  [1]       NA 25.21076 25.22675 25.87790 27.30835 27.74983 27.52601 28.81707
##  [9] 30.56550 30.23003 30.25272 29.11794 27.07172 27.47693 26.30551 23.04610
## [17] 22.67712 24.00399 23.61940 23.37675 24.31478 24.04155 23.22199 23.46744
## [25] 23.23638 21.39936 19.05638 17.90487 18.40938 18.85500 18.00637 17.27663
## [33] 17.10441 18.61605 21.52983 22.32792 23.01150 24.47396 22.85543 20.36910
## [41] 19.28133 17.66236 16.82967 16.60816 14.89865 13.75986 14.02752 13.38758
## [49] 11.70177 11.04494 11.45344 11.62152 12.31616 13.39556 13.28090 12.74574
## [57] 12.64702       NA
##
## $random
## Time Series:
## Start = c(1960, 1)
## End = c(1988, 2)
## Frequency = 2
##  [1]          NA  1.30811114 -0.66538264 -0.61251177  1.11111886 -0.62076141
##  [7]  0.79570762 -2.48553788  3.71799981 -2.86656619  1.65697403  0.71009248
## [13] -2.16571901  1.16991544  1.40251418 -1.88694859 -0.51905359  1.22920353
## [19] -0.22788310 -0.91538138  0.87796762  0.37070131 -1.07303691  0.71036371
## [25]  0.12881691  0.63795501 -0.89875256 -0.03192344 -0.69343130  1.47768850
## [31] -0.96770557  0.33884395 -0.26750705 -1.48769979  1.84077991 -0.07817616
## [37] -1.56991382  2.43912257 -0.22734000 -1.11664925  1.06208011 -0.47630206
## [43] -0.89576260  1.65664696 -0.92953119 -0.36830335  0.25968173  0.75654690
## [49] -0.72690445 -0.33172042  0.32501398 -0.07788572 -0.69580301  1.08473392
## [55] -0.27960908 -0.10502096  0.05321819          NA
##
```

```
## $figure
## [1]  0.02880065 -0.02880065
##
## $type
## [1] "additive"
##
## attr(,"class")
## [1] "decomposed.ts"
```

```r
autoplot(decomposed_ts) +
  ggtitle("Decomposition of Time Series") +
  xlab("Year") +
  ylab("Exports")
```