ARIMA models

Souleymane Doumbia

2024-10-22

Exercise 1: Analyzing ACF for Random Numbers

Figure 9.32 shows the ACFs for 36 random numbers, 360 random numbers, and 1,000 random numbers.

a. Explain the differences among these figures. Do they all indicate that the data are white noise?

The three ACF plots correspond to different sample sizes (36, 360, and 1,000 random numbers). The differences among these figures are mainly due to the sample size:

- **Left Plot (36 numbers)**: The ACF has more variation, and several spikes appear to be outside the blue dashed confidence bounds. This is because, with a small sample size, random fluctuations can result in larger apparent autocorrelations, even for white noise.

- **Middle Plot (360 numbers)**: The ACF shows fewer significant spikes compared to the smaller sample, and most of the values fall within the confidence bounds, indicating white noise behavior. The larger sample size reduces the random variability in the ACF.

- **Right Plot (1,000 numbers)**: The ACF is even more stable, with almost all spikes within the confidence bounds. This suggests that as the sample size increases, the behavior of the ACF for white noise becomes more consistent, with fewer apparent correlations.

All three plots indicate that the data are white noise, but the smaller sample sizes show more variability, making it harder to distinguish true white noise from random fluctuations.

b. Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?

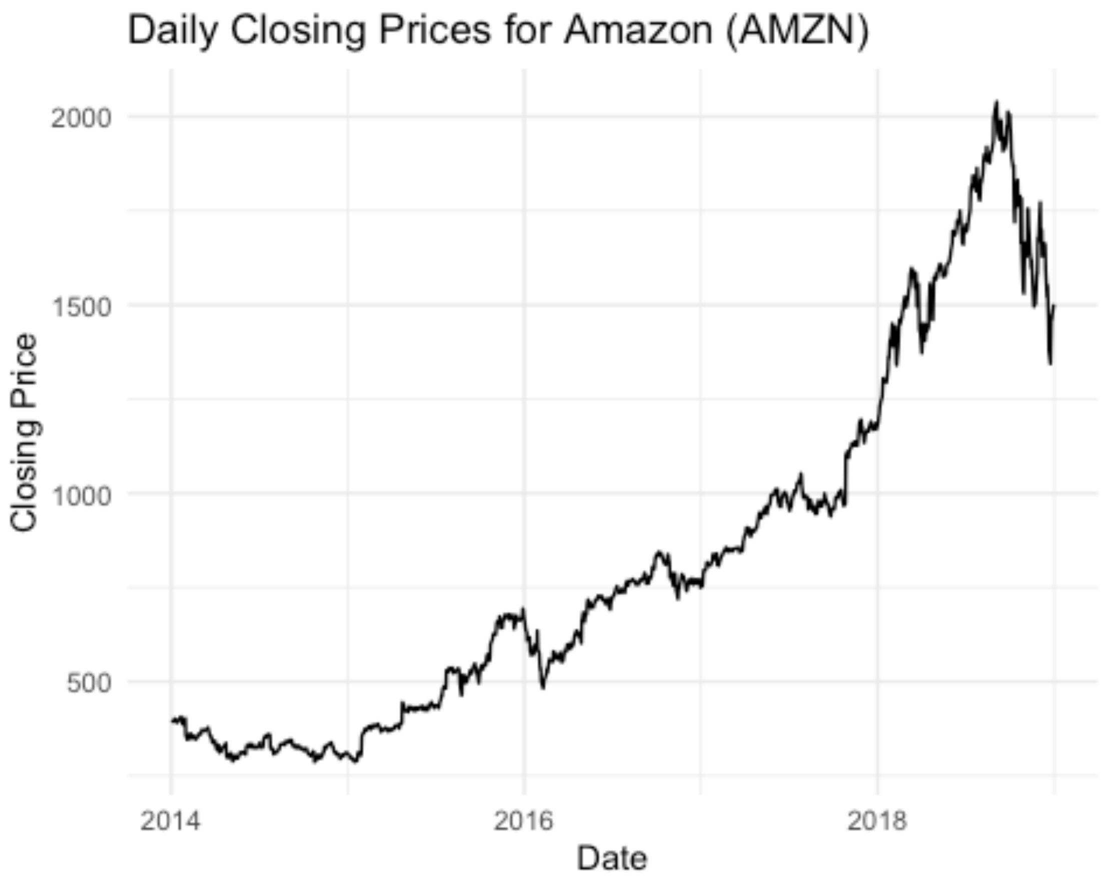
- **Critical Values**: The critical values (blue dashed lines) represent the confidence bounds for testing whether the autocorrelations are significantly different from zero. These bounds depend on the sample size. As the sample size increases, the confidence interval becomes narrower, leading to smaller critical values. This is why the confidence bounds are wider for the plot with 36 numbers compared to the plots with 360 and 1,000 numbers.

- **Autocorrelations**: The apparent autocorrelations vary in each plot due to the randomness inherent in smaller samples. With fewer data points (e.g., 36), random fluctuations can cause more extreme autocorrelation values, making it harder to conclude that the data is truly white noise. As the sample size increases, these fluctuations average out, resulting in more stable ACF values that are closer to zero, which is what we expect for white noise.

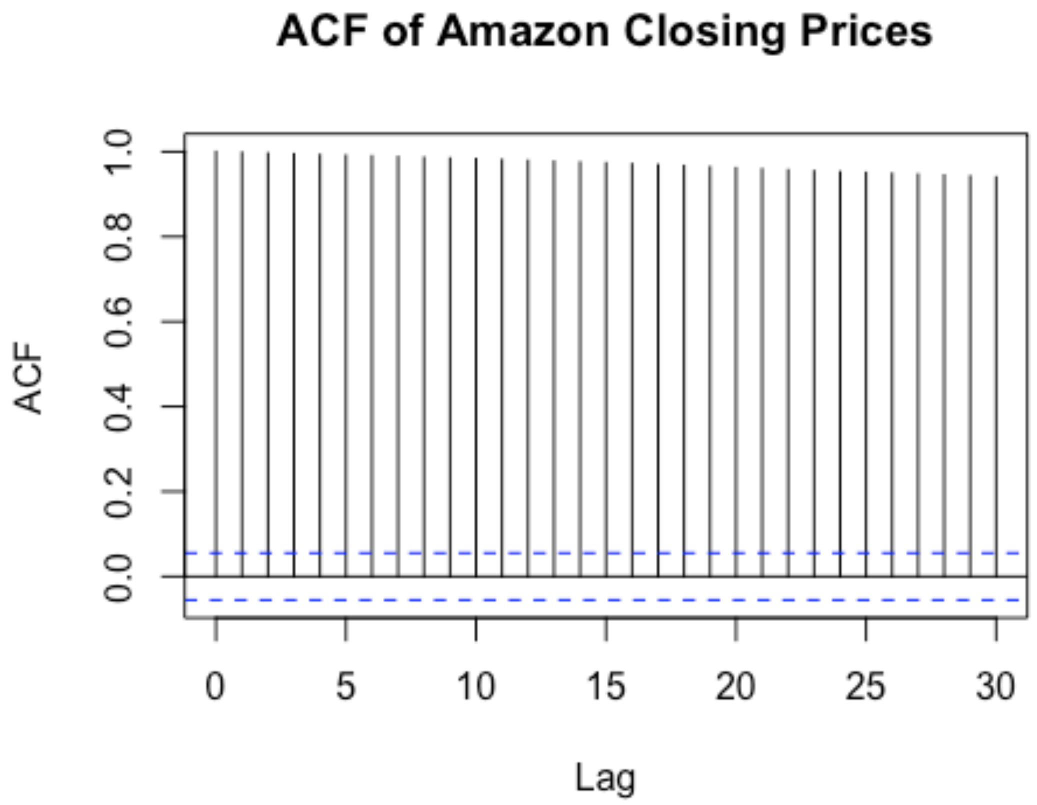
Exercise 2: Non-Stationary Series in Stock Prices

Below are the daily closing prices for Amazon stock (contained in `gafa_stock`), along with the ACF and PACF.

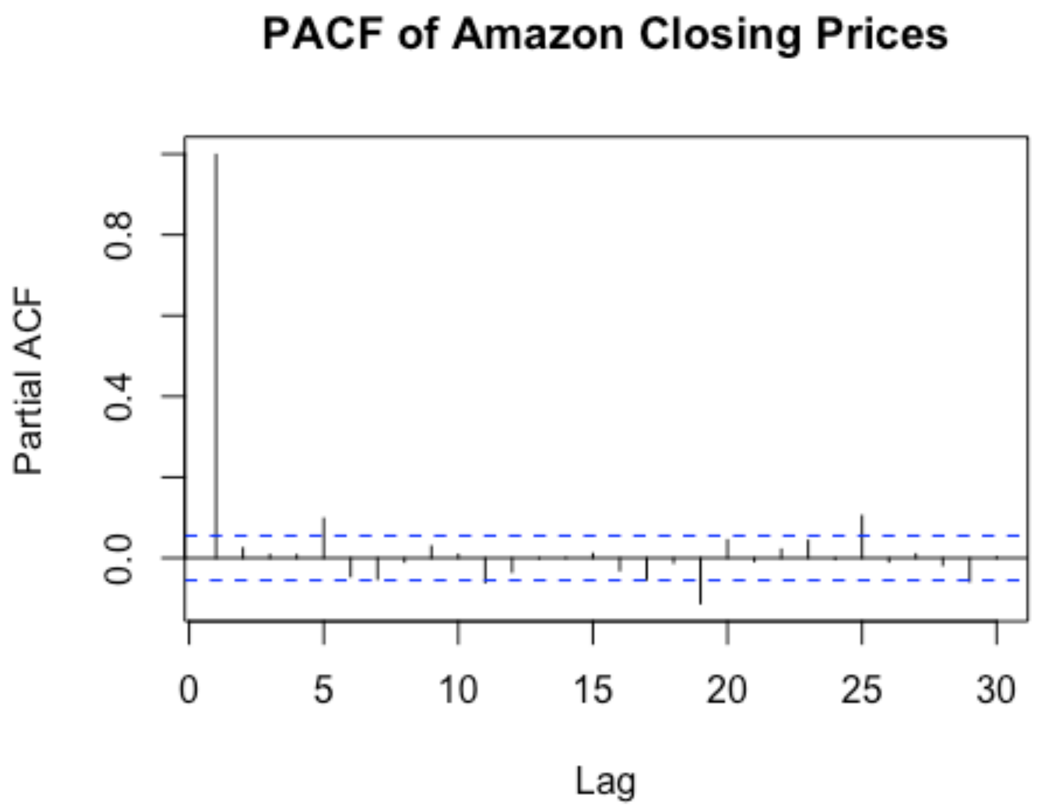
```r
# Filtering for Amazon (AMZN) stock
amazon_stock <- gafa_stock %>% filter(Symbol == "AMZN")

# Plotting the daily closing prices for Amazon
ggplot(amazon_stock, aes(x = Date, y = Close)) +
  geom_line() +
  labs(title = "Daily Closing Prices for Amazon (AMZN)",
       x = "Date", y = "Closing Price") +
  theme_minimal()
```



```r
# Plotting the ACF of the closing prices
acf(amazon_stock$Close, main = "ACF of Amazon Closing Prices")
```



```r
# Plotting the PACF of the closing prices
pacf(amazon_stock$Close, main = "PACF of Amazon Closing Prices")
```

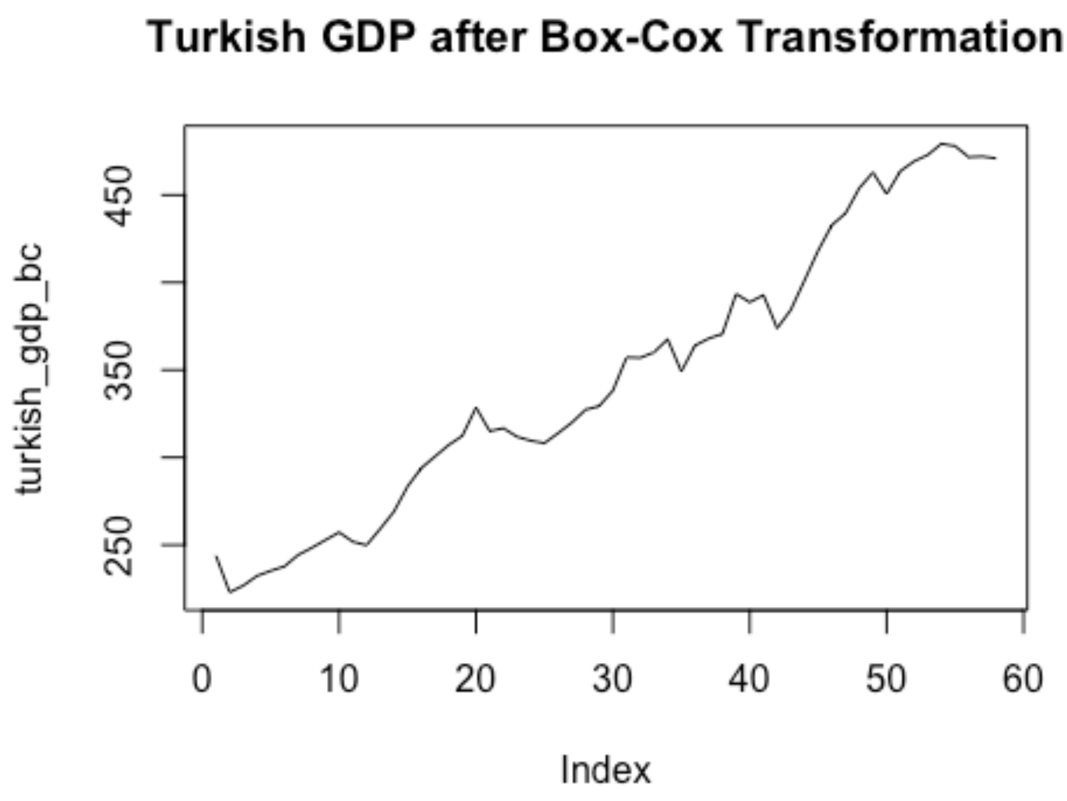


- **Closing Price Plot:** The plot of Amazon's closing prices shows a clear upward trend over time, particularly from 2014 onwards. This indicates that the series is **non-stationary**, as stationary series typically fluctuate around a constant mean without such a noticeable trend.
- **ACF Plot:** The ACF plot exhibits **very high autocorrelations** at all lags, with no quick decay. This slow decay suggests that past values are highly correlated with future values, a common characteristic of non-stationary series. In a stationary series, the ACF would decline sharply and approach zero after a few lags.
- **PACF Plot:** The PACF plot shows a **significant spike at lag 1**, followed by much smaller spikes. This pattern indicates that most of the variation is explained by the first lag, another sign of a non-stationary series. Such behavior implies that differencing can be used to reduce this high autocorrelation at lag 1 and potentially make the series stationary.

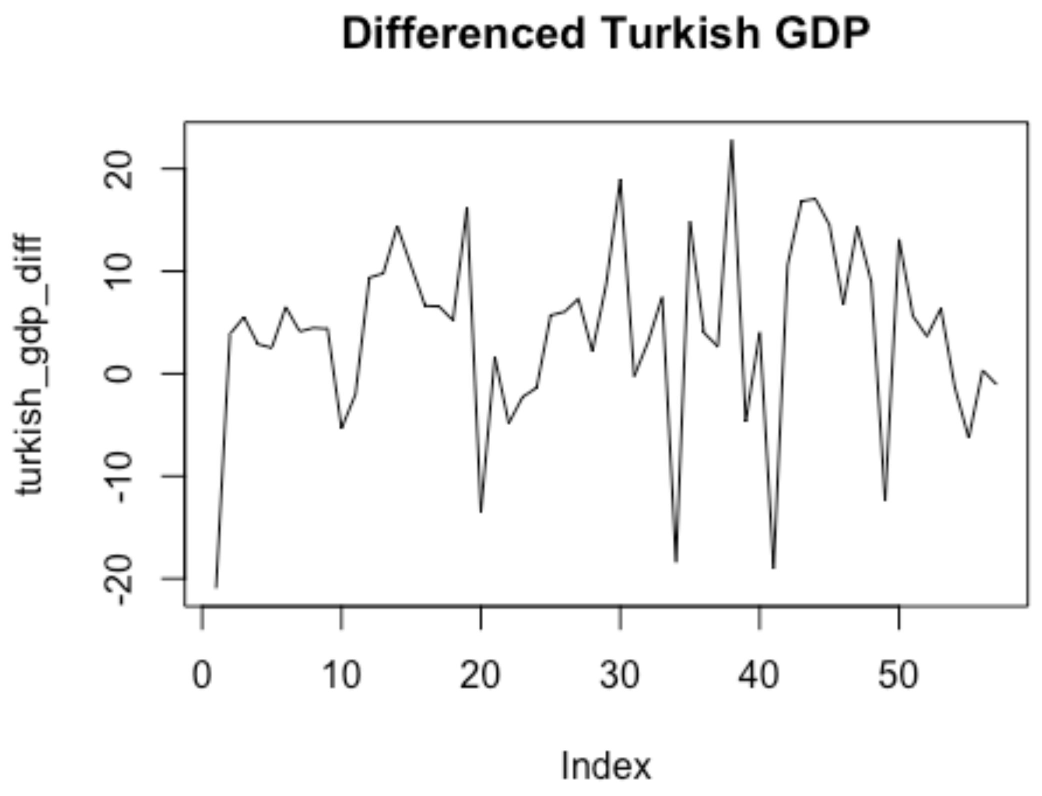
Exercise 3: Finding the Box-Cox transformation and determining the order of differencing to make each of the time series stationary.

a. Turkish GDP from global_economy

```
# Filtering for Turkish GDP
turkish_gdp <- global_economy %>% filter(Country == "Turkey")

# Applying Box-Cox transformation and plotting the series
turkish_gdp_bc <- BoxCox(turkish_gdp$GDP, BoxCox.lambda(turkish_gdp$GDP))
plot(turkish_gdp_bc, type = "l", main = "Turkish GDP after Box-Cox
Transformation")
```



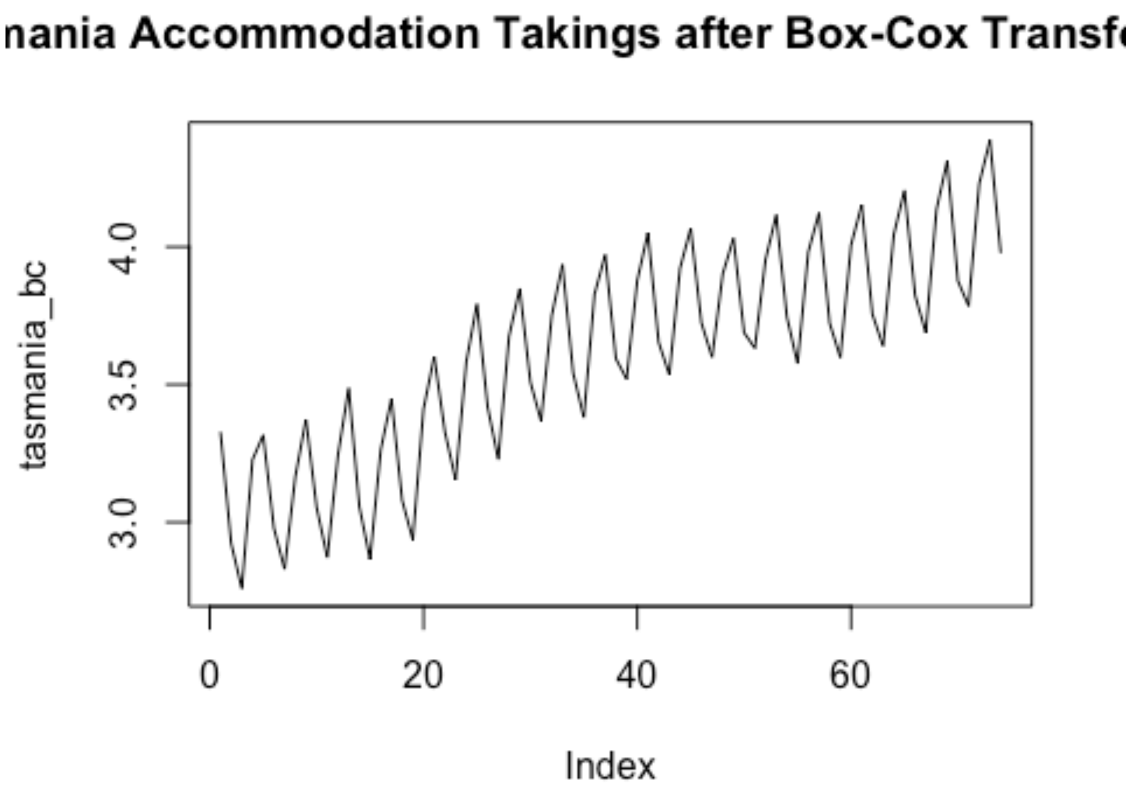
```
# Determining the order of differencing (using ndiffs)
diff_order_gdp <- ndiffs(turkish_gdp_bc)
print(diff_order_gdp)

## [1] 1

# Differencing the series
turkish_gdp_diff <- diff(turkish_gdp_bc, differences = diff_order_gdp)
plot(turkish_gdp_diff, type = "l", main = "Differenced Turkish GDP")
```



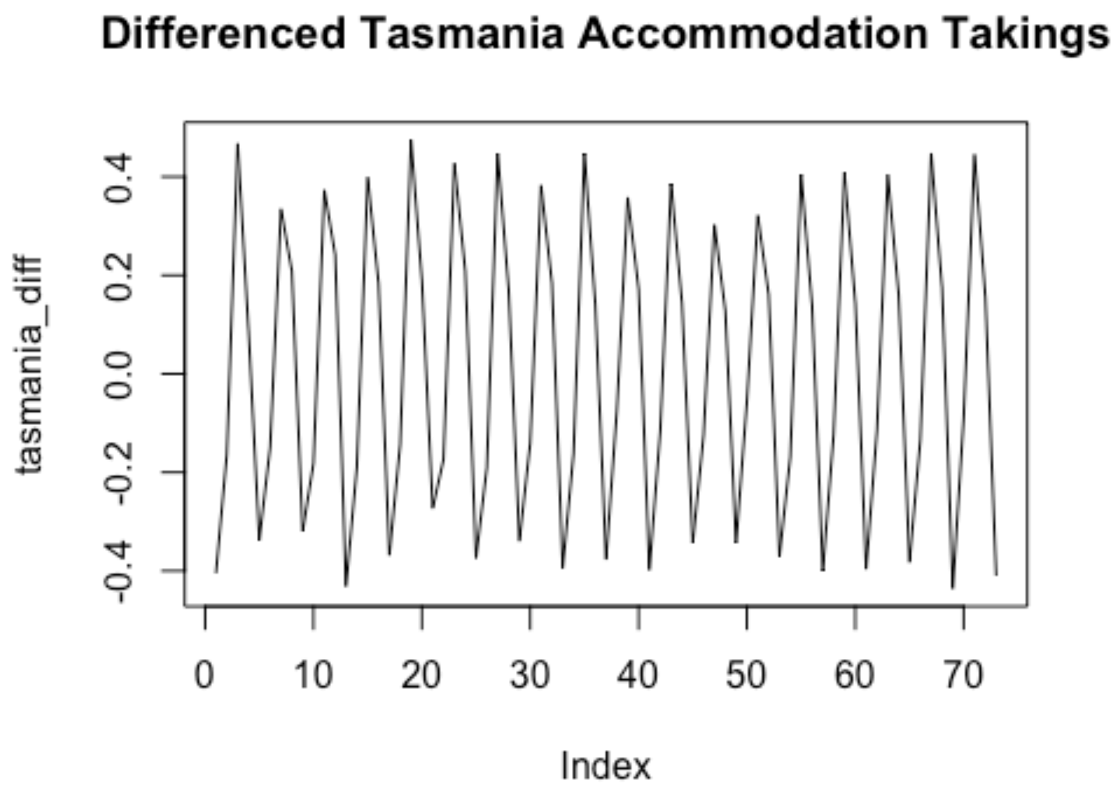
- The **Turkish GDP series** was first transformed using the Box-Cox method to stabilize the variance. However, the data still exhibited an upward trend, indicating non-stationarity. The `ndiffs()` function suggested that one differencing was needed to achieve stationarity. After applying the first difference, the series appeared stationary, with fluctuations around a constant mean and no noticeable trend remaining.

b. Accommodation Takings from aus_accommodation

```
# Filtering for Tasmania Accommodation Takings
tasmania_takings <- aus_accommodation %>% filter(State == "Tasmania")

# Applying Box-Cox transformation and plotting the series
tasmania_bc <- BoxCox(tasmania_takings$Takings,
BoxCox.lambda(tasmania_takings$Takings))
plot(tasmania_bc, type = "l", main = "Tasmania Accommodation Takings after
Box-Cox Transformation")
```



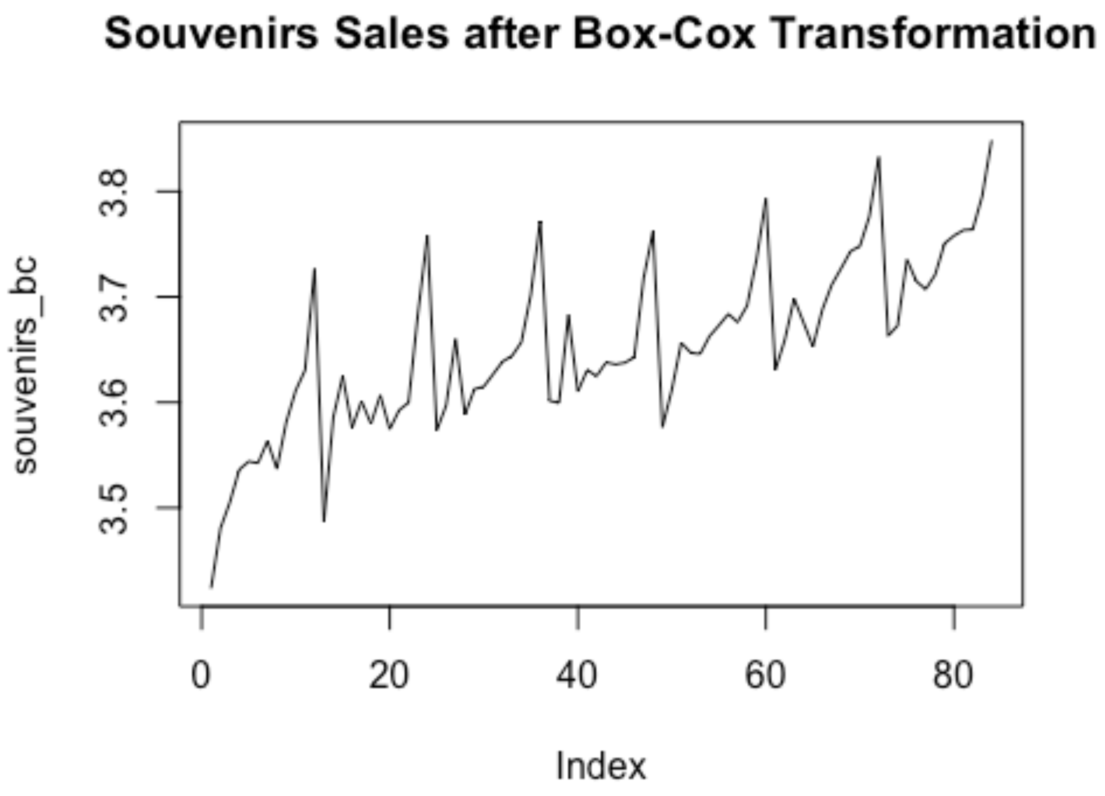
```r
# Determine=ing the order of differencing
diff_order_tas <- ndiffs(tasmania_bc)
print(diff_order_tas)

## [1] 1

# Differencing the series
tasmania_diff <- diff(tasmania_bc, differences = diff_order_tas)
plot(tasmania_diff, type = "l", main = "Differenced Tasmania Accommodation
Takings")
```



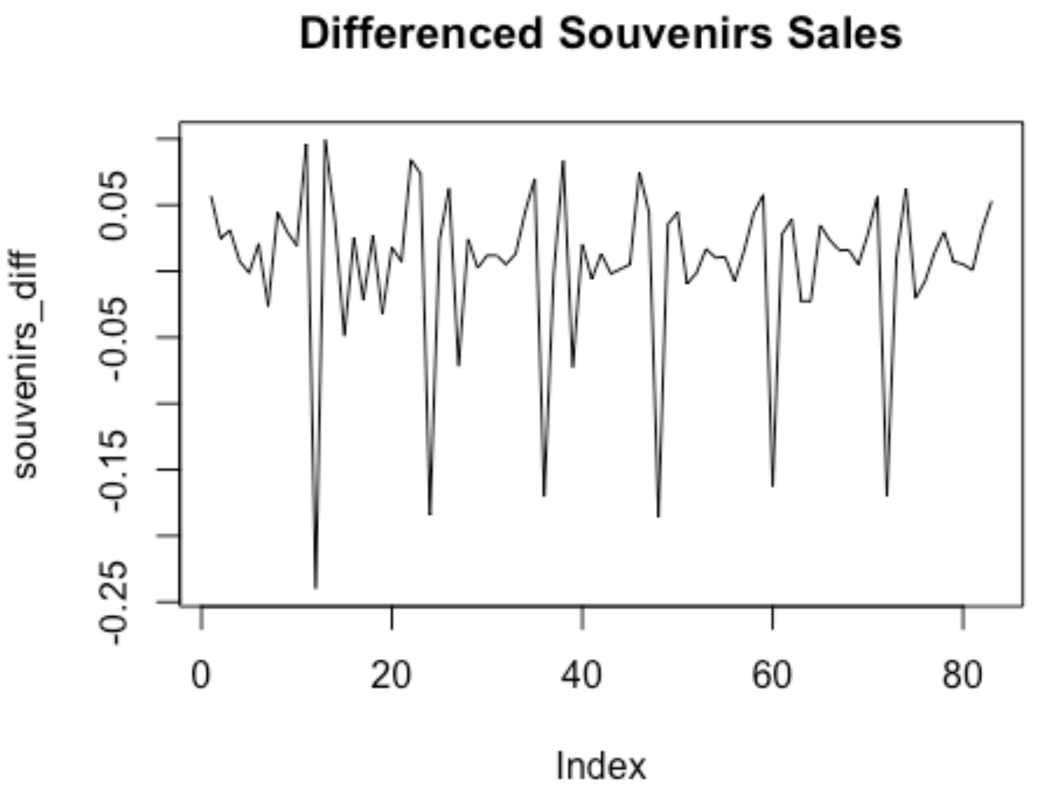
- The Tasmania accommodation takings series was first transformed using the Box-Cox method to stabilize the variance. However, a seasonal pattern and upward trend remained in the transformed data, indicating non-stationarity. The `ndiffs()` function suggested that one difference was necessary to achieve stationarity. After applying the first difference, the series still exhibited a strong seasonal pattern, but the trend was removed which might suggest that additional seasonal differencing or modeling may be required to address the remaining seasonality.

c. Monthly_Sales_from_souvenirs

```
# Applying Box-Cox transformation and plot the series
souvenirs_bc <- BoxCox(souvenirs$Sales, BoxCox.lambda(souvenirs$Sales))
plot(souvenirs_bc, type = "l", main = "Souvenirs Sales after Box-Cox
Transformation")
```



```
# Determining the order of differencing
diff_order_souvenir <- ndiffs(souvenirs_bc)
print(diff_order_souvenir)

## [1] 1

# Differencing the series
souvenirs_diff <- diff(souvenirs_bc, differences = diff_order_souvenir)
plot(souvenirs_diff, type = "l", main = "Differenced Souvenirs Sales")
```

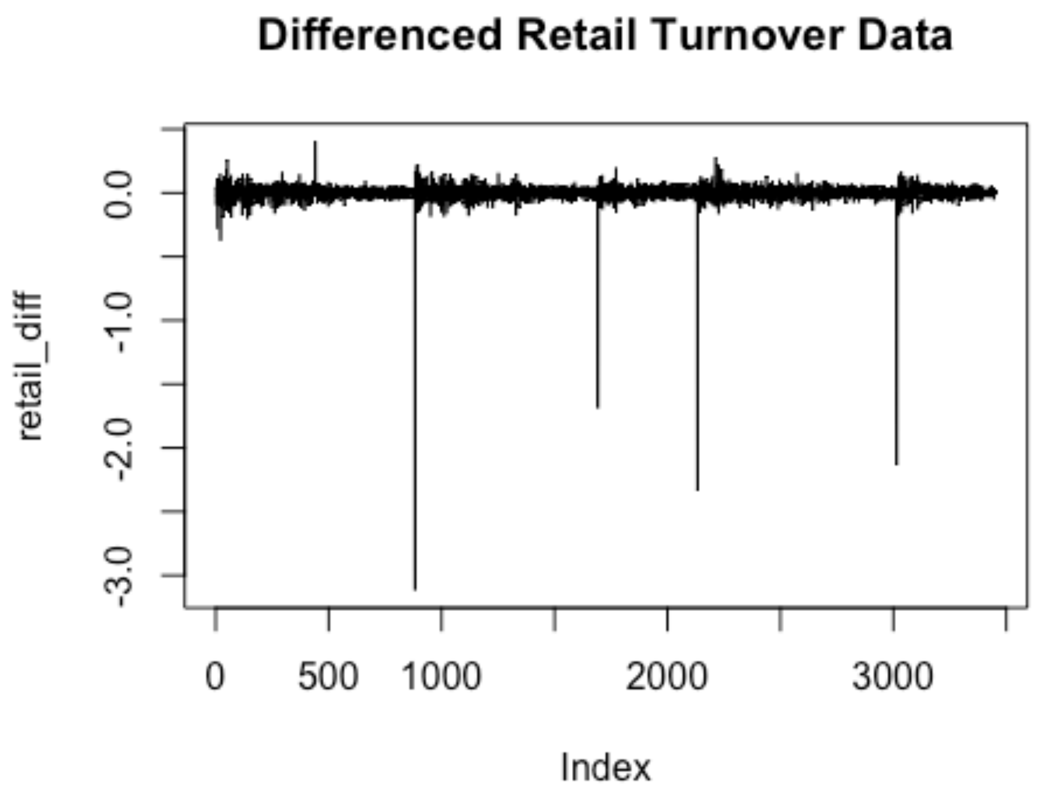


- The souvenir sales series was first transformed using the Box-Cox method, which helped stabilize the variance, although the data still showed a trend and seasonal fluctuations. The `ndiffs()` function recommended one differencing to achieve stationarity. After applying the first difference, the series appeared more stationary, with no significant trend, but seasonal variations still remained. These results suggest that further seasonal differencing or adjustments might be necessary to fully address the seasonality in the data.

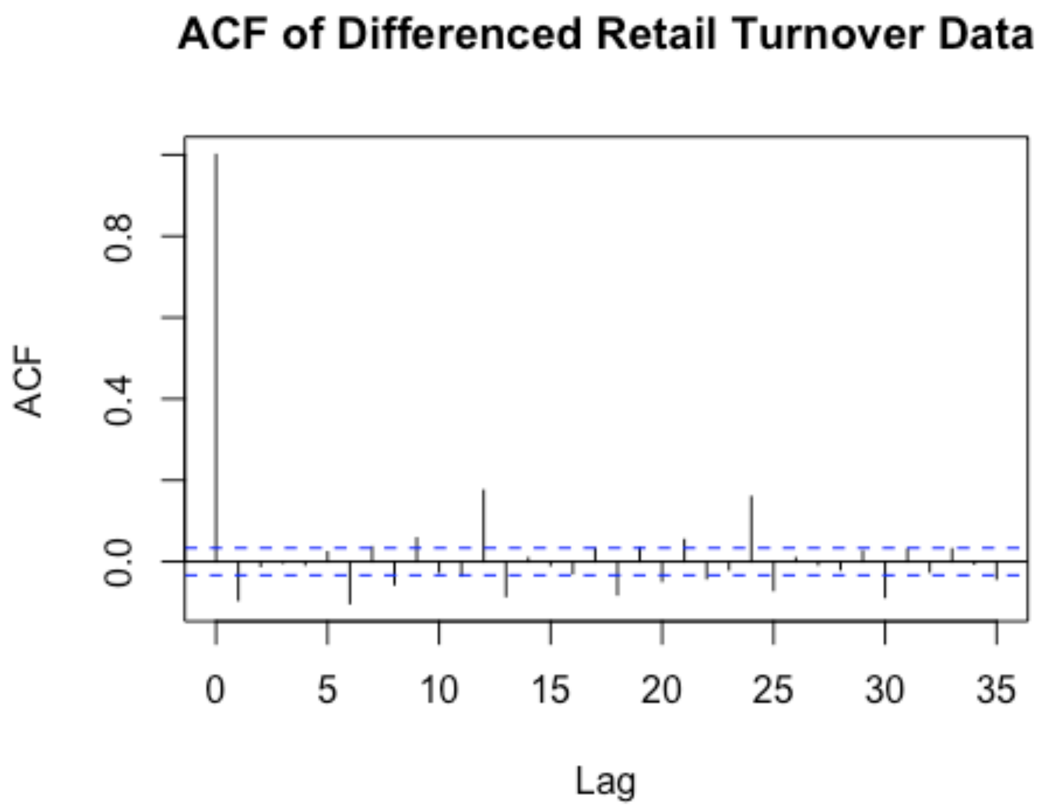
Exercise 5: Determining the Appropriate Order of Differencing for Retail Data to obtain stationary data

```
# Filtering for a specific retail category ('Cafes, restaurants and takeaway
food services')
retail_data <- aus_retail %>% filter(Industry == "Cafes, restaurants and
takeaway food services")

# Applying Box-Cox Transformation
lambda <- BoxCox.lambda(retail_data$Turnover)
retail_bc <- BoxCox(retail_data$Turnover, lambda)

# Determining the order of differencing
```

```r
diff_order_retail <- ndiffs(retail_bc)
print(diff_order_retail)

## [1] 1

# Applying differencing
retail_diff <- diff(retail_bc, differences = diff_order_retail)

# Plotting the differenced series to confirm stationarity
plot(retail_diff, type = "l", main = "Differenced Retail Turnover Data")
```



```r
# Checking ACF to confirm stationarity
acf(retail_diff, main = "ACF of Differenced Retail Turnover Data")
```



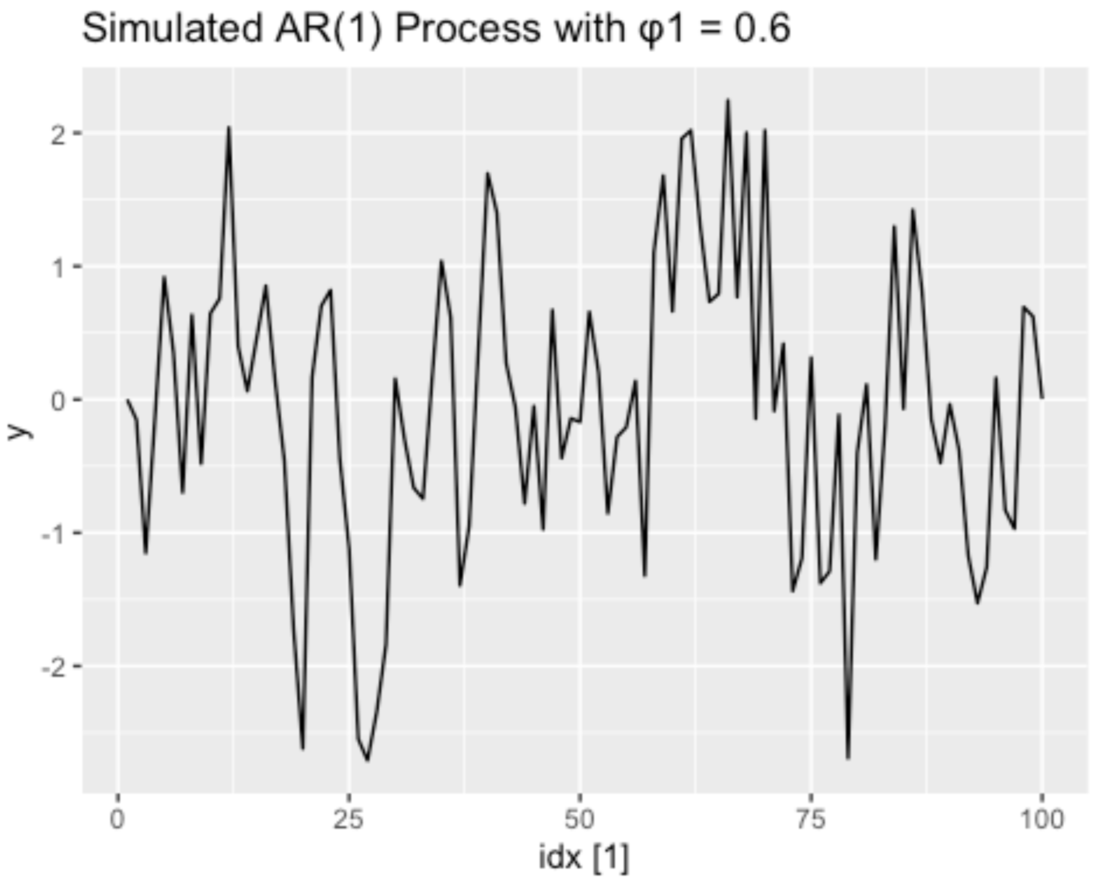
- The retail turnover data required to achieve stationarity, as indicated by the function. After differencing, the ACF plot shows that the autocorrelations drop off quickly, confirming that the data is stationary with no evident trend remaining. The differenced series fluctuates around a constant mean, further supporting the conclusion of stationarity.

Exercise 6: Simulating and plotting some data from simple ARIMA models.

a. Simulate data from an AR(1) model with $\phi_1 = 0.6$ and $\sigma^2 = 1$

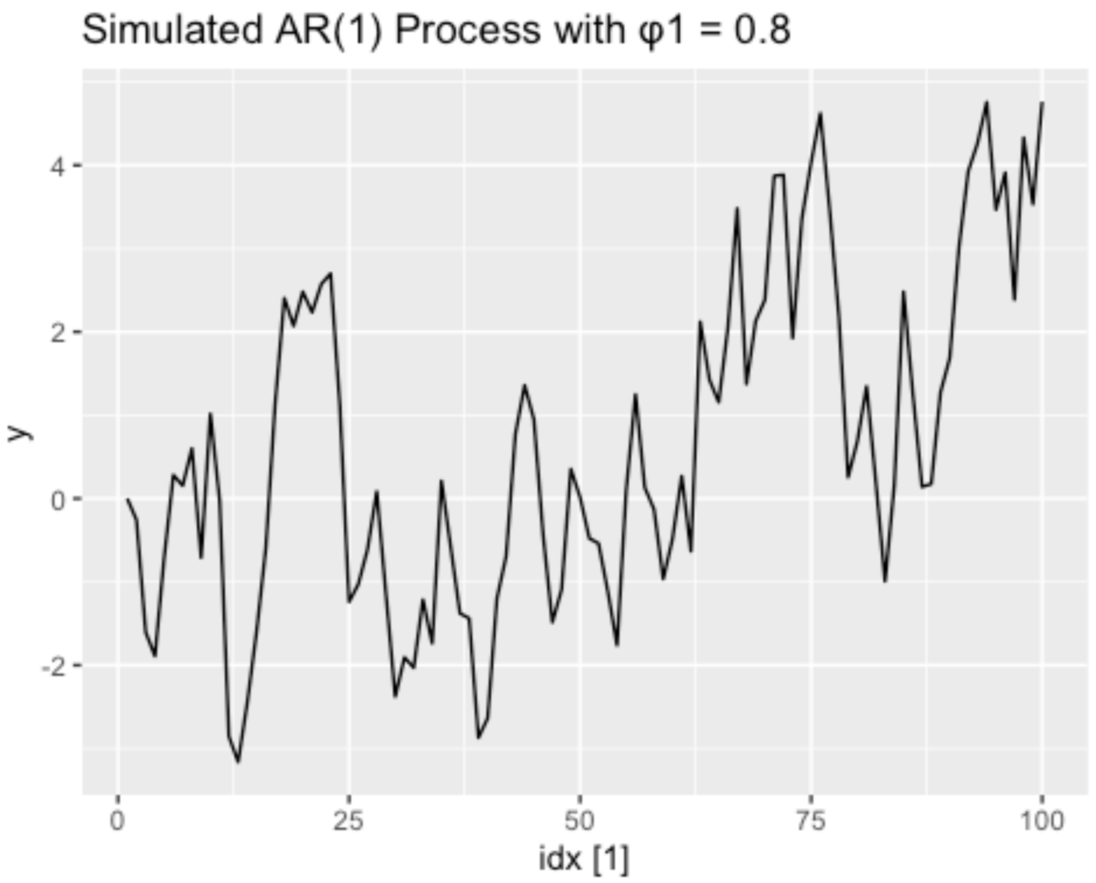
```r
# AR(1) process with phi_1 = 0.6 and sigma^2 = 1
y <- numeric(100)        # Initializing y
e <- rnorm(100)          # Error term (white noise)
for(i in 2:100) {        # Generating AR(1) process
  y[i] <- 0.6*y[i-1] + e[i]
}

# Converting to a tsibble for time series plot
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
```

```r
# Time plot for the series
autoplot(sim, y) + ggtitle("Simulated AR(1) Process with ϕ1 = 0.6")
```



b. Time plot of the AR(1) process with $\phi_1 = 0.8$

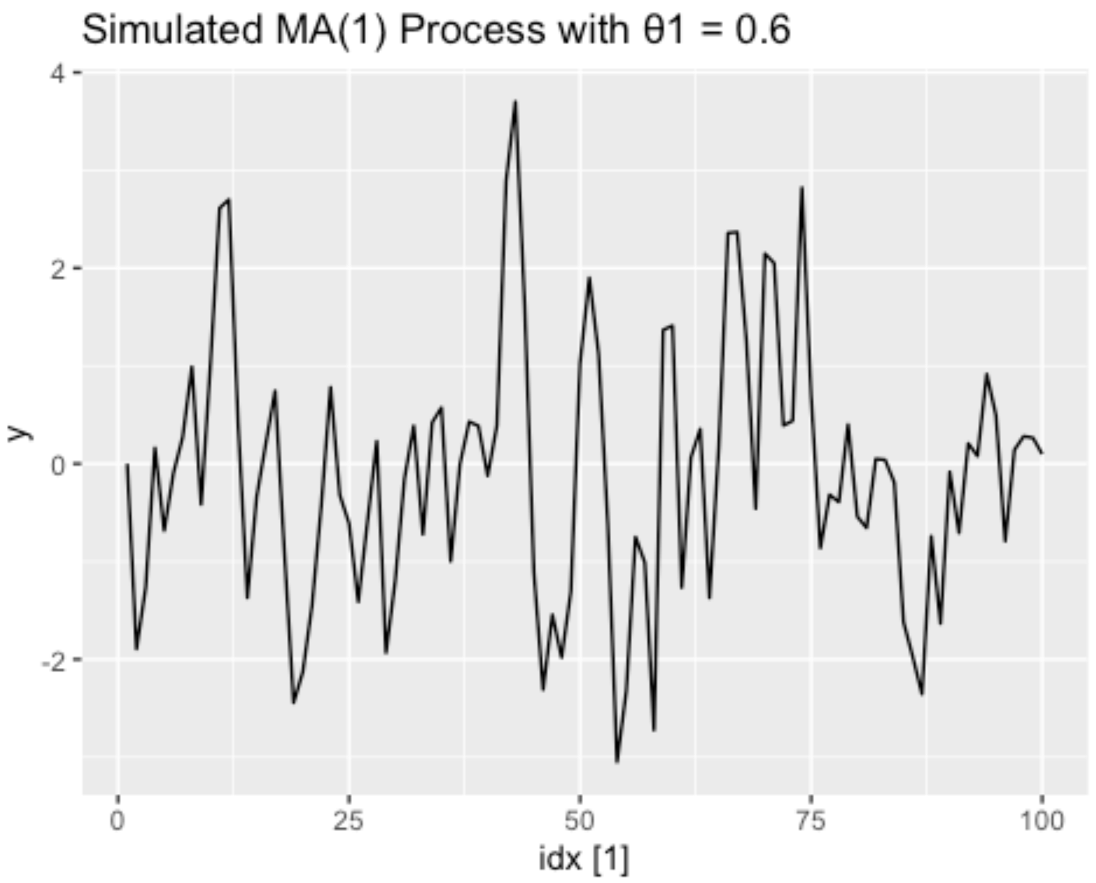
```r
# AR(1) process with phi_1 = 0.8 and sigma^2 = 1
y <- numeric(100)        # Initializing y
e <- rnorm(100)          # Error term (white noise)
for(i in 2:100) {        # Generating AR(1) process
  y[i] <- 0.8*y[i-1] + e[i]
}

# Converting to tsibble for time series plot
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

# Plotting the time series
autoplot(sim, y) + ggtitle("Simulated AR(1) Process with ϕ1 = 0.8")
```



- As ϕ_1 increases from 0.6 to 0.8, the time plot shows smoother, more persistent deviations from zero. This indicates stronger autocorrelation, where the values depend more heavily on their past values, resulting in slower changes in direction and longer-lasting trends.

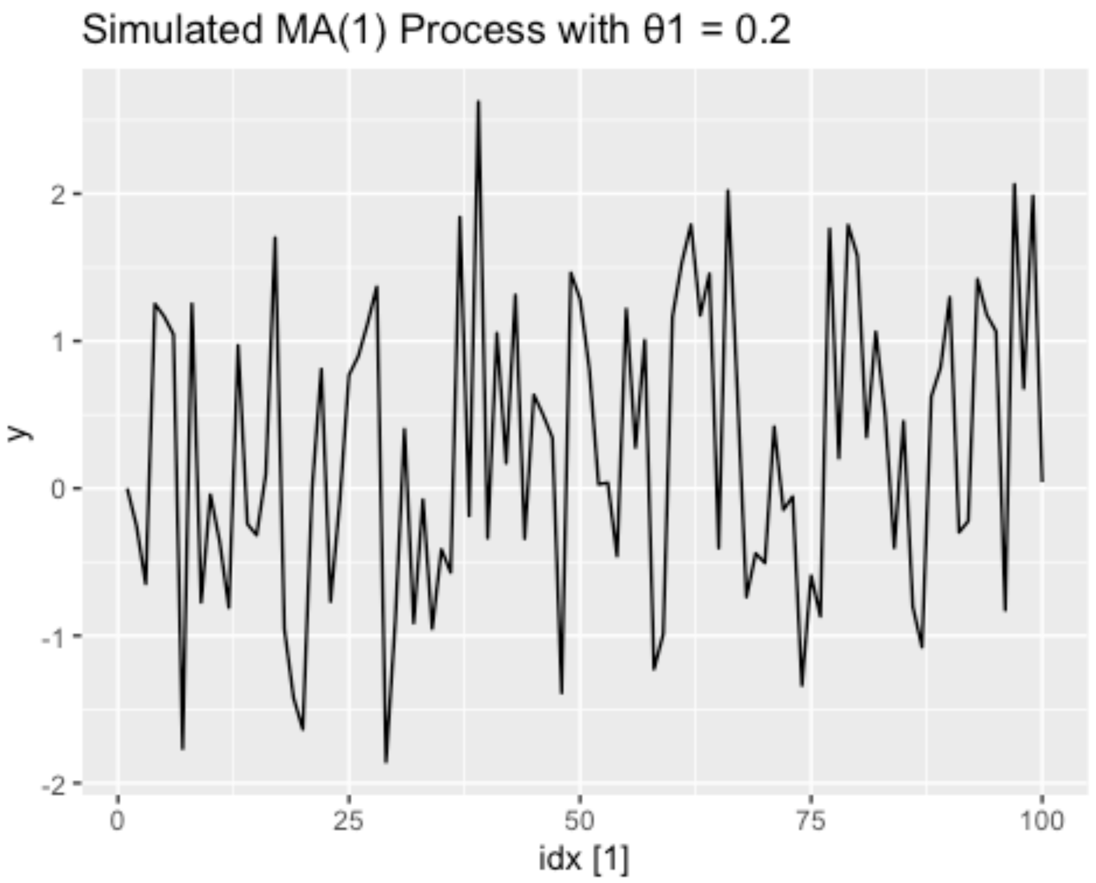
c. Simulate data from an MA(1) model with $\theta_1 = 0.6$ and $\sigma^2 = 1$

```r
# MA(1) process with theta_1 = 0.6 and sigma^2 = 1
y <- numeric(100)
e <- rnorm(101)
for(i in 2:100) {
  y[i] <- e[i] + 0.6*e[i-1]
}

# Converting to tsibble and plot
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
autoplot(sim, y) + ggtitle("Simulated MA(1) Process with θ1 = 0.6")
```



d. Time plot of the MA(1) process with $\theta_1 = 0.2$

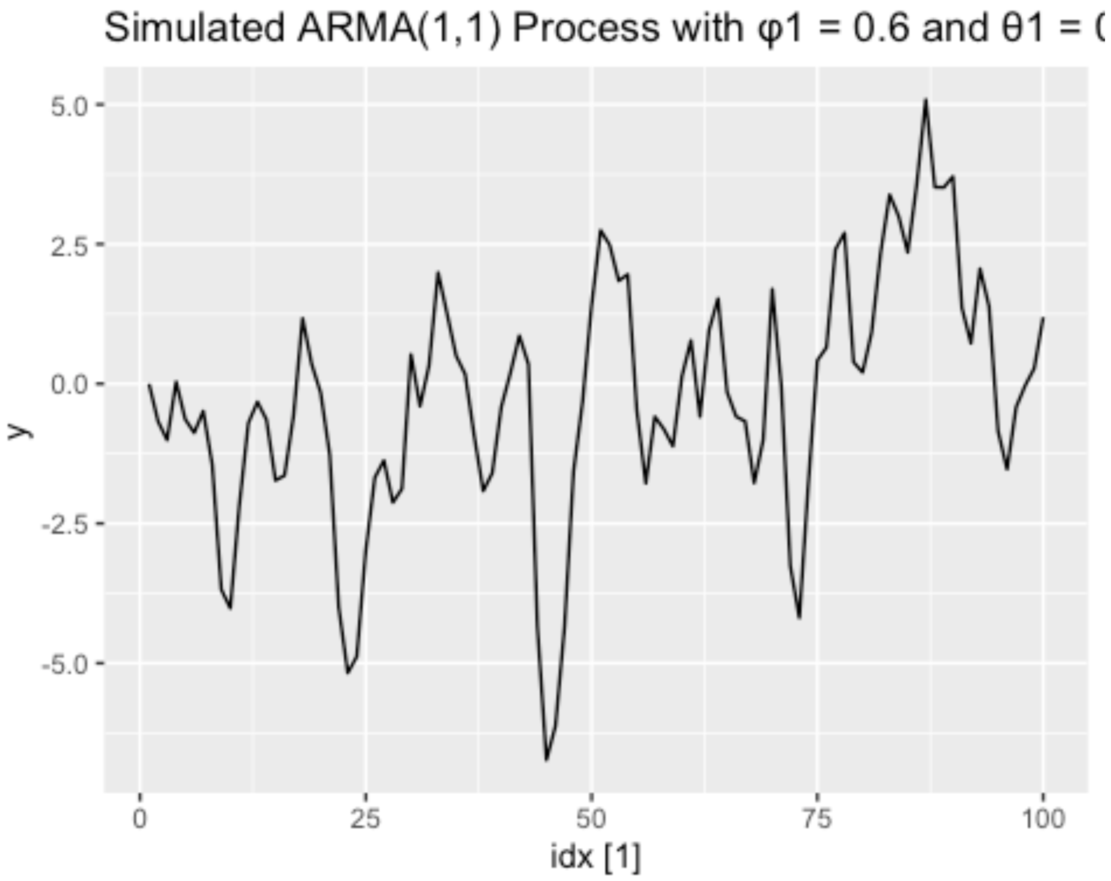
```
# MA(1) process with theta_1 = 0.2 and sigma^2 = 1
y <- numeric(100)
e <- rnorm(101)  # Error term (one extra for MA process)
for(i in 2:100) {
  y[i] <- e[i] + 0.2 * e[i-1]
}

# Converting to tsibble and plot
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

# Plotting the time series
autoplot(sim, y) + ggtitle("Simulated MA(1) Process with θ1 = 0.2")
```



- As we change the value of θ_1 from 0.6 to 0.2, the time series shows a reduction in the magnitude of the fluctuations. When $\theta_1 = 0.6$, the series exhibits more pronounced peaks and valleys, indicating stronger moving average effects. On the other hand, when $\theta_1 = 0.2$, the variations are more subdued, leading to a series with less extreme movements.

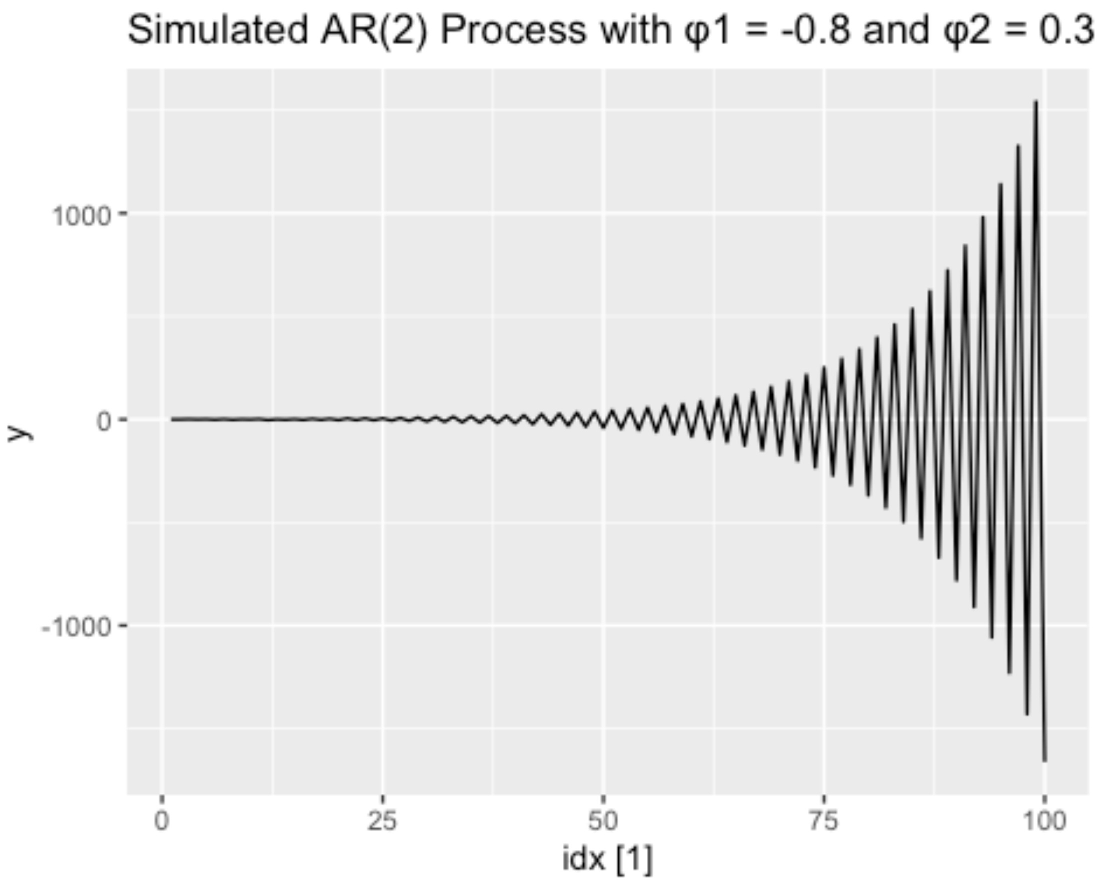
e. Simulate data from an ARMA(1,1) model with $\phi_1 = 0.6$, $\theta_1 = 0.6$, and $\sigma^2 = 1$

```
# ARMA(1,1) process with phi_1 = 0.6, theta_1 = 0.6, and sigma^2 = 1
y <- numeric(100)
e <- rnorm(101)
for(i in 2:100) {
  y[i] <- 0.6*y[i-1] + e[i] + 0.6*e[i-1]
}

# Converting to tsibble and plot
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
autoplot(sim, y) + ggtitle("Simulated ARMA(1,1) Process with ϕ1 = 0.6 and θ1
= 0.6")
```



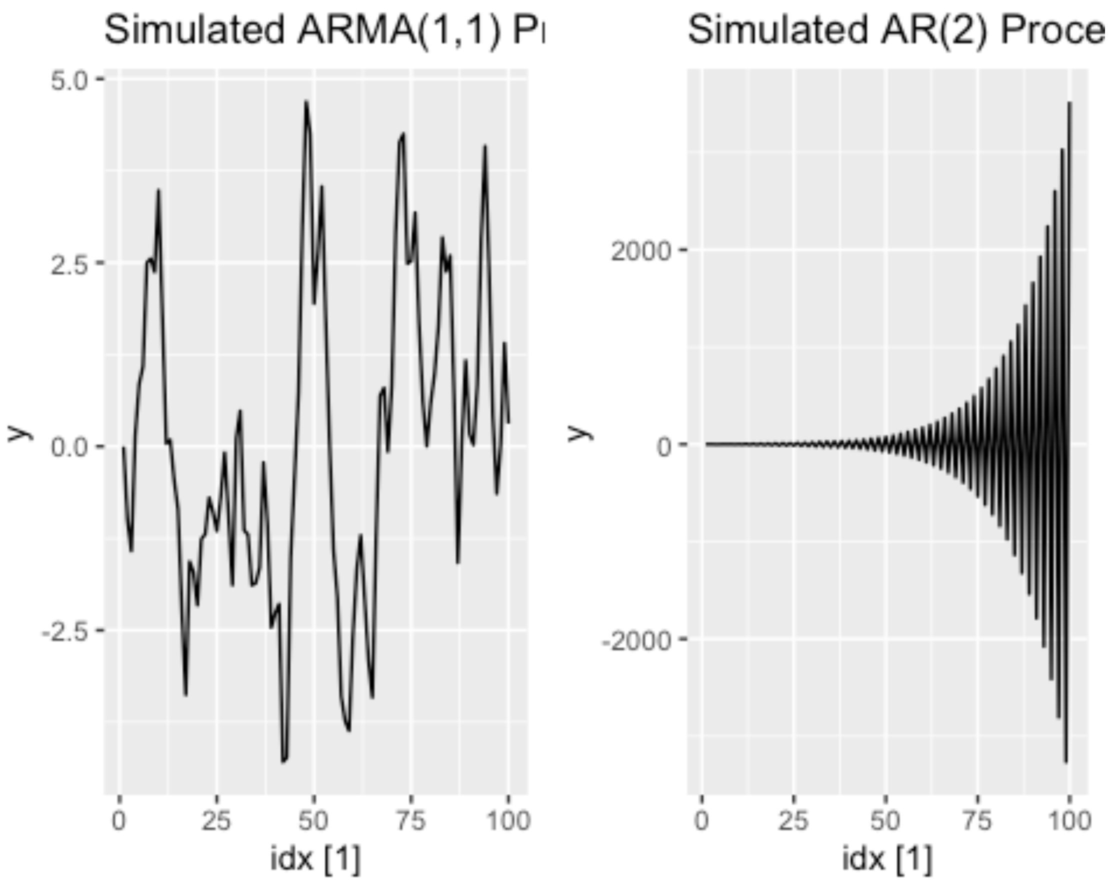
f. Simulate data from an AR(2) model with $\phi_1 = -0.8$, $\phi_2 = 0.3$, and $\sigma^2 = 1$

```
# AR(2) process with phi_1 = -0.8, phi_2 = 0.3, and sigma^2 = 1
y <- numeric(100)
e <- rnorm(100)
for(i in 3:100) {
  y[i] <- -0.8*y[i-1] + 0.3*y[i-2] + e[i]
}

# Converting to tsibble and plot
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
autoplot(sim, y) + ggtitle("Simulated AR(2) Process with φ1 = -0.8 and φ2 =
0.3")
```



g. Compare the ARMA(1,1) and AR(2) time series plots

```r
# ARMA(1,1) data
y_arma <- numeric(100)
e_arma <- rnorm(101)
for(i in 2:100) {
  y_arma[i] <- 0.6*y_arma[i-1] + e_arma[i] + 0.6*e_arma[i-1]
}
sim_arma <- tsibble(idx = seq_len(100), y = y_arma, index = idx)

# AR(2) data
y_ar2 <- numeric(100)
e_ar2 <- rnorm(100)
for(i in 3:100) {
  y_ar2[i] <- -0.8*y_ar2[i-1] + 0.3*y_ar2[i-2] + e_ar2[i]
}
sim_ar2 <- tsibble(idx = seq_len(100), y = y_ar2, index = idx)

# Plotting both series side by side
p1 <- autoplot(sim_arma, y) + ggtitle("Simulated ARMA(1,1) Process")
p2 <- autoplot(sim_ar2, y) + ggtitle("Simulated AR(2) Process")
grid.arrange(p1, p2, ncol = 2)
```



The two series generated from ARMA(1,1) and AR(2) models show clear differences:

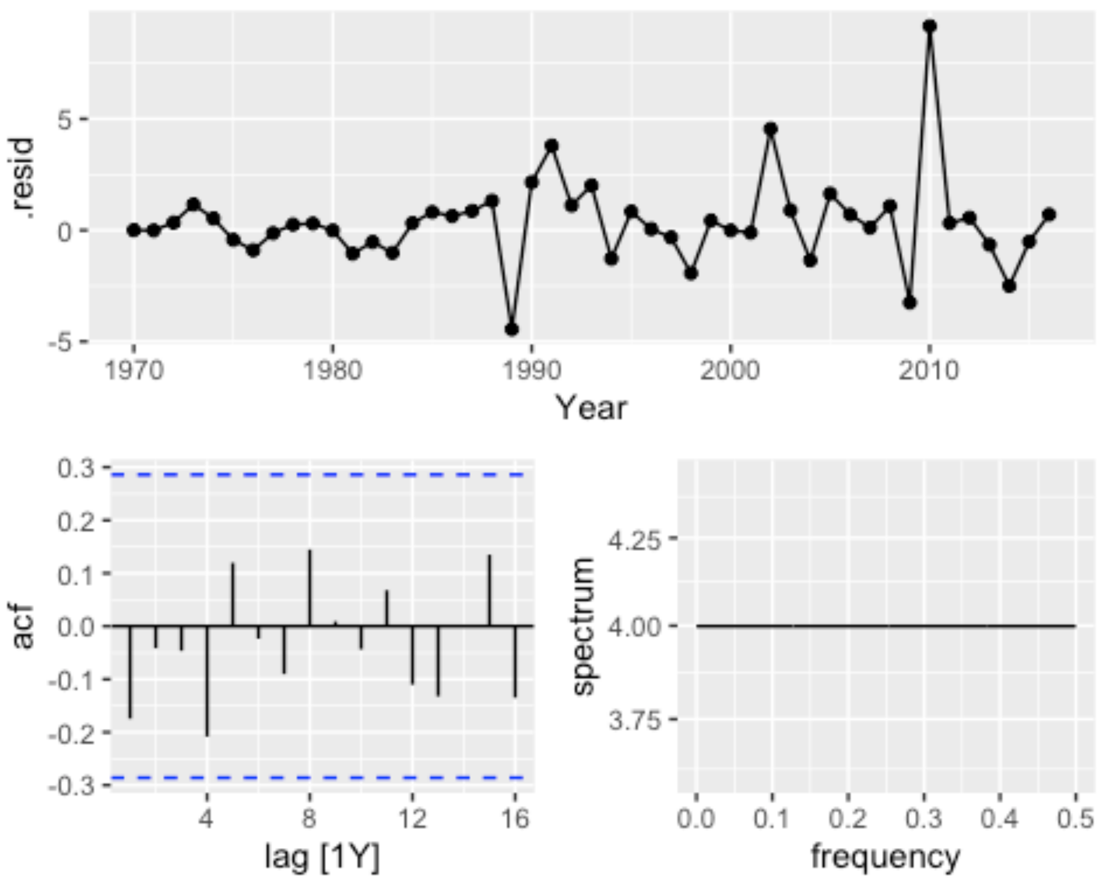
- The with parameters $\phi_1 = 0.6$ and $\theta_1 = 0.6$ displays a stationary process. The plot fluctuates around a constant mean and exhibits variability that appears stable over time, which is characteristic of a stationary time series.
- The with $\phi_1 = -0.8$ and $\phi_2 = 0.3$ shows a non-stationary behavior, as expected. The plot starts with small values but gradually diverges, increasing in magnitude over time. This growing oscillation confirms that the chosen parameters result in a non-stationary series, where the values do not settle around a constant mean.

Exercise 7: Using aus_airpassengers dataset, the total number of passengers (in millions) from Australian air carriers for the period 1970-2011.

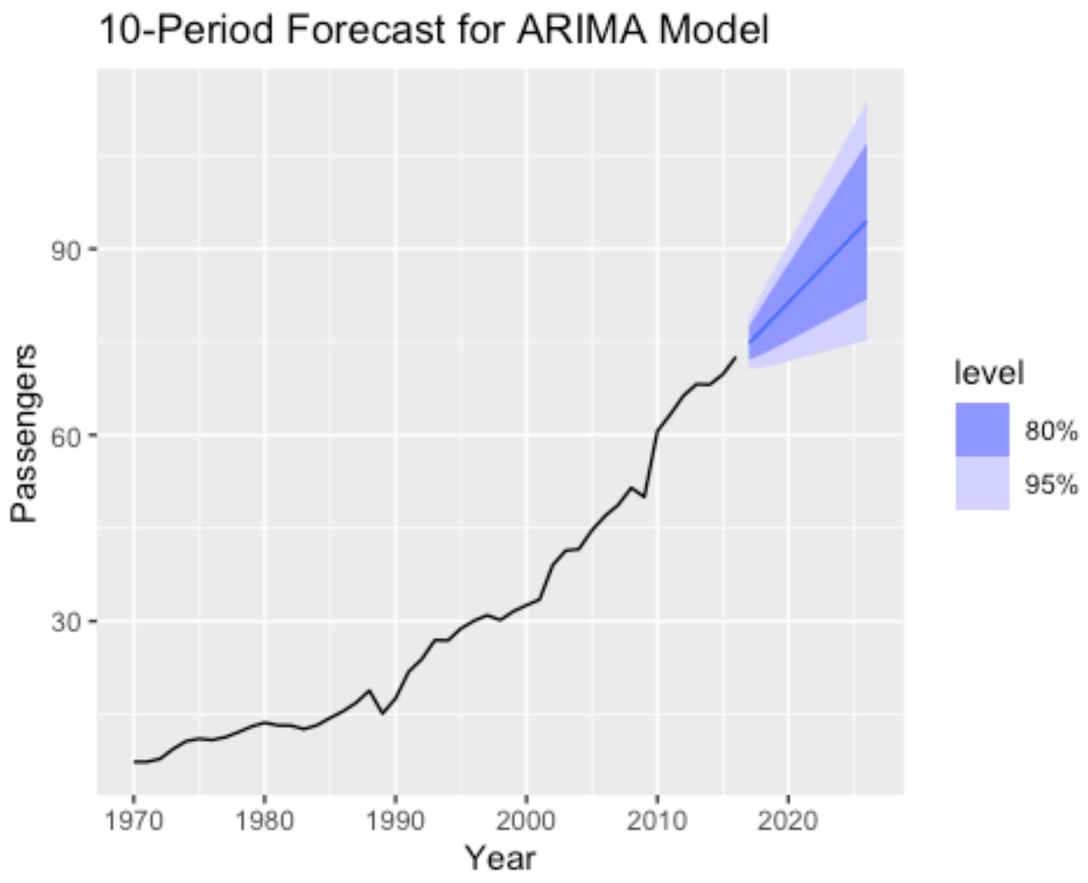
a. Fit ARIMA model and plot forecasts for the next 10 periods

```
fit_a <- aus_airpassengers %>%
  model(ARIMA(Passengers))

# Checking the fitted ARIMA model
fit_a
```

```
## # A mable: 1 x 1
##   `ARIMA(Passengers)`
##               <model>
## 1      <ARIMA(0,2,1)>

# Extracting and plotring the residuals to check if they resemble white noise
fit_a %>%
  residuals() %>%
  gg_tsdisplay()

## Plot variable not specified, automatically selected `y = .resid`
```



```
# Forecasting the next 10 periods
fc_a <- fit_a %>%
  forecast(h = 10)

# Plotting the forecasts
fc_a %>%
  autoplot(aus_airpassengers) + ggtitle("10-Period Forecast for ARIMA Model")
```



- **Selected Model:** The ARIMA(0,2,1) model was selected as the best fit for the `aus_airpassengers` dataset.
- **Residuals Check:** The residual plot shows no significant patterns, and the ACF of the residuals falls within the confidence intervals, indicating that the residuals resemble white noise. This suggests that the model is adequately capturing the structure of the data.
- **Forecast:** The forecast for the next 10 periods shows an upward trend in the number of passengers, with the prediction intervals (80% and 95%) widening over time, indicating increasing uncertainty in the forecasts for future periods.
- Overall, the ARIMA(0,2,1) model is an appropriate fit for the data, and the residuals confirm that the model has accounted for the underlying data trends.

b. Write the ARIMA model in terms of the backshift operator

```r
# Fitting the ARIMA model
fit_b <- aus_airpassengers %>%
  model(ARIMA(Passengers))
```

```
# Extracting ARIMA model components (AR, MA, differencing)
glance(fit_b)
```

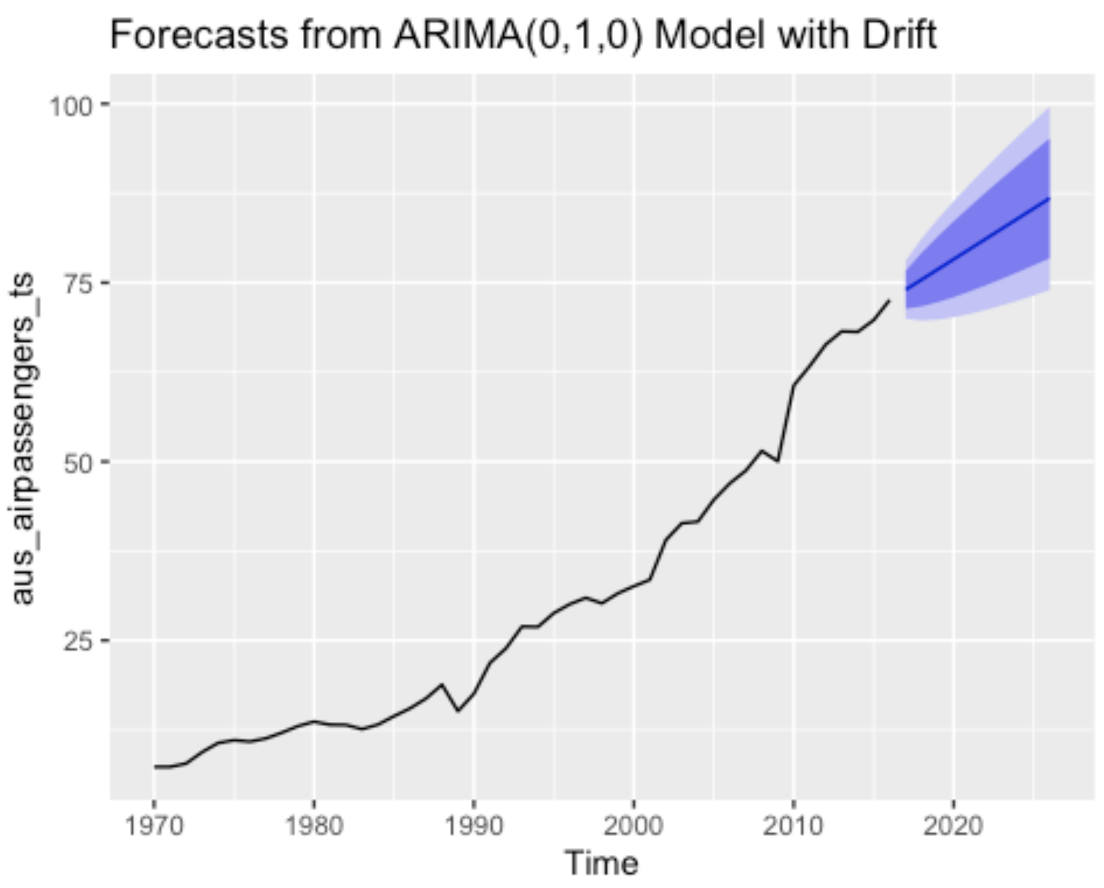
```
## # A tibble: 1 × 8
##   .model            sigma2 log_lik   AIC  AICc   BIC ar_roots   ma_roots
##   <chr>              <dbl>   <dbl> <dbl> <dbl> <dbl> <list>     <list>
## 1 ARIMA(Passengers)   4.31   -97.0  198.  198.  202. <cpl [0]> <cpl [1]>
```

```
# Printing the model with its full details (order and coefficients)
fit_b %>% report()
```

```
## Series: Passengers
## Model: ARIMA(0,2,1)
##
## Coefficients:
##           ma1
##       -0.8963
## s.e.   0.0594
##
## sigma^2 estimated as 4.308:  log likelihood=-97.02
## AIC=198.04   AICc=198.32   BIC=201.65
```

c. Plot forecasts from ARIMA(0,1,0) model with drift and compare these to part a.

```
# Converting aus_airpassengers to a time series object
aus_airpassengers_ts <- ts(aus_airpassengers$Passengers, start = 1970,
frequency = 1)

# Fitting ARIMA(0,1,0) model with drift
fit_c <- Arima(aus_airpassengers_ts, order = c(0,1,0), include.drift = TRUE)

# Forecasting next 10 periods
fc_c <- forecast(fit_c, h = 10)

# Plotting the forecasts
autoplot(fc_c) + ggtitle("Forecasts from ARIMA(0,1,0) Model with Drift")
```

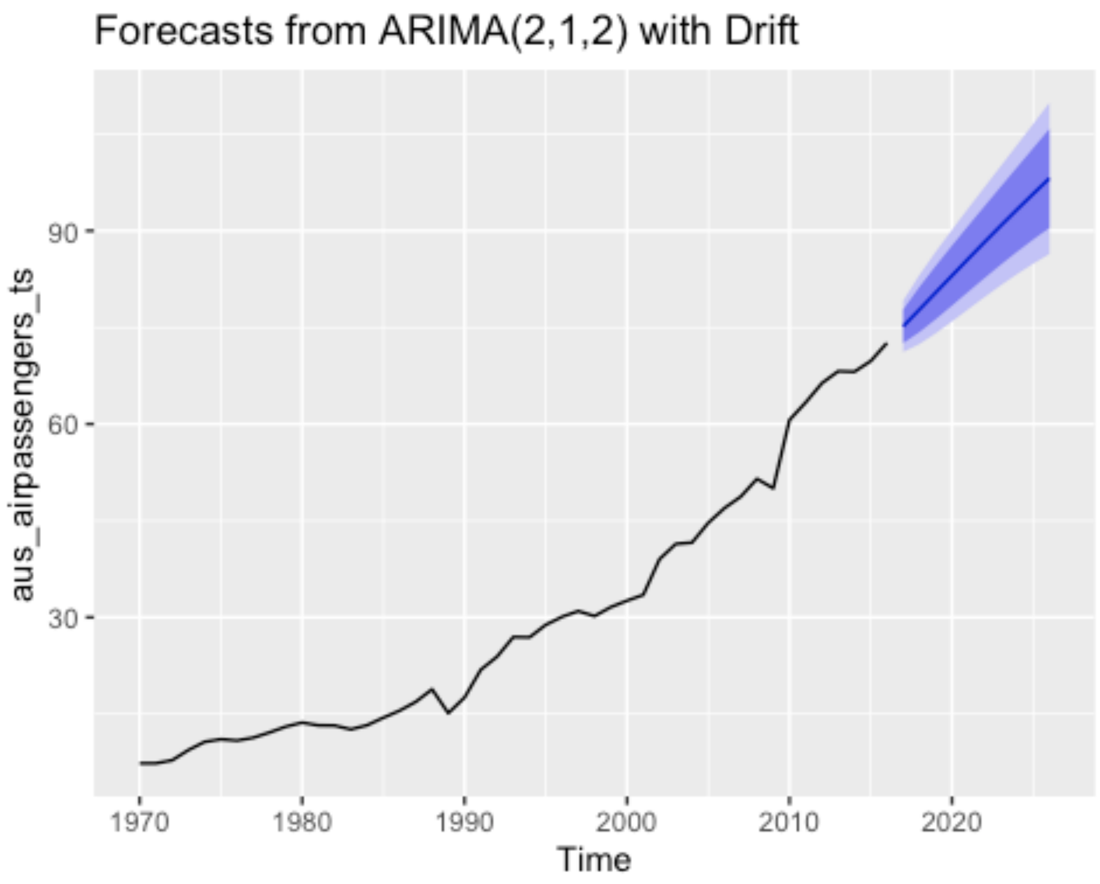


- The forecast from the ARIMA(0,1,0) model with drift shows a similar upward trend as in part **a**, but with slightly more gradual growth.
- The drift component introduces a linear trend in the forecast, which is more pronounced when compared to the ARIMA(0,2,1) model in part **a**, where the non-stationary aspect led to stronger exponential growth in the forecasts.
- Both models show increasing forecasts for the next 10 periods, but the ARIMA(0,1,0) with drift is more conservative in its projections.

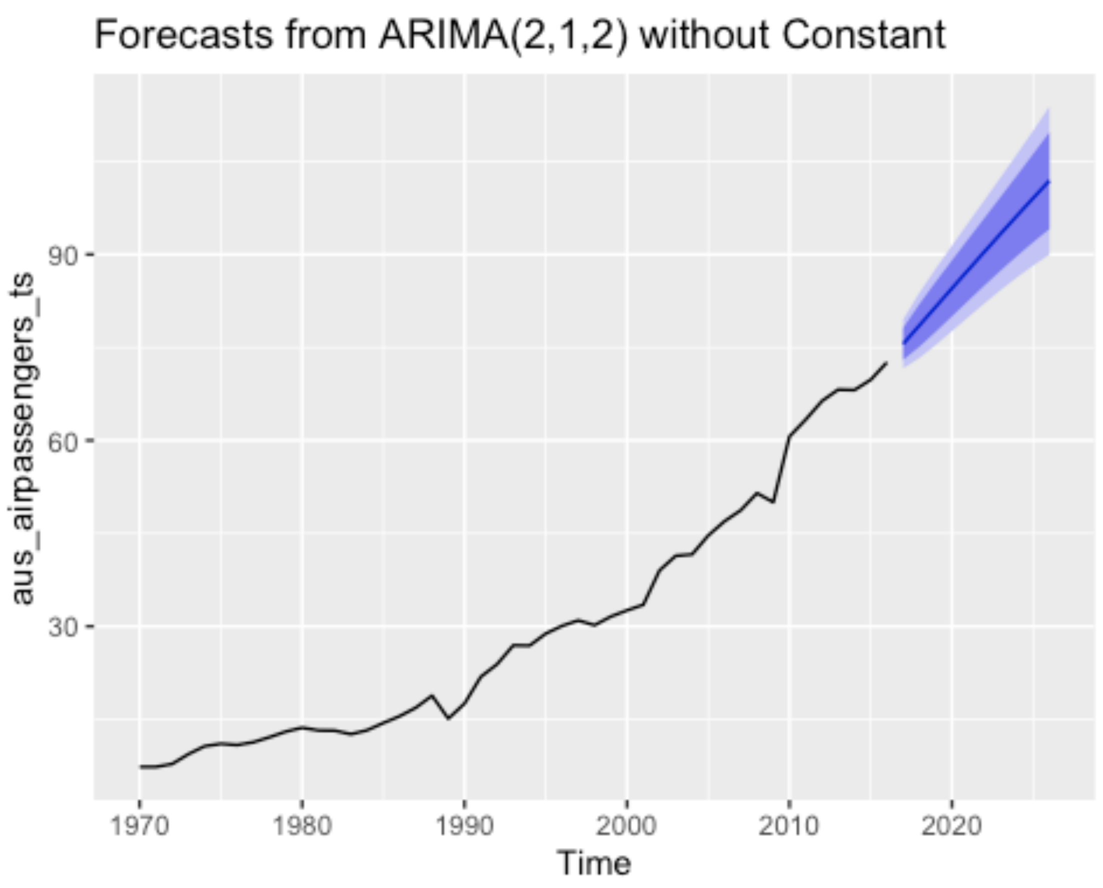
d. Plot forecasts from ARIMA(2,1,2) model with drift and without constant

```
# Converting aus_airpassengers to a time series object
aus_airpassengers_ts <- ts(aus_airpassengers$Passengers, start = 1970,
frequency = 1)

# Fitting ARIMA(2,1,2) model with drift using ML (Maximum Likelihood)
fit_d_with_drift <- Arima(aus_airpassengers_ts, order = c(2,1,2),
include.drift = TRUE, method = "ML")

# Fitting ARIMA(2,1,2) model without constant using ML
fit_d_without_constant <- Arima(aus_airpassengers_ts, order = c(2,1,2),
include.constant = FALSE, method = "ML")
```

```r
# Forecasting for both models
fc_d_with_drift <- forecast(fit_d_with_drift, h = 10)
fc_d_without_constant <- forecast(fit_d_without_constant, h = 10)

# Plotting forecasts with drift
autoplot(fc_d_with_drift) + ggtitle("Forecasts from ARIMA(2,1,2) with Drift")
```



```r
# Plotting forecasts without constant
autoplot(fc_d_without_constant) + ggtitle("Forecasts from ARIMA(2,1,2)
without Constant")
```



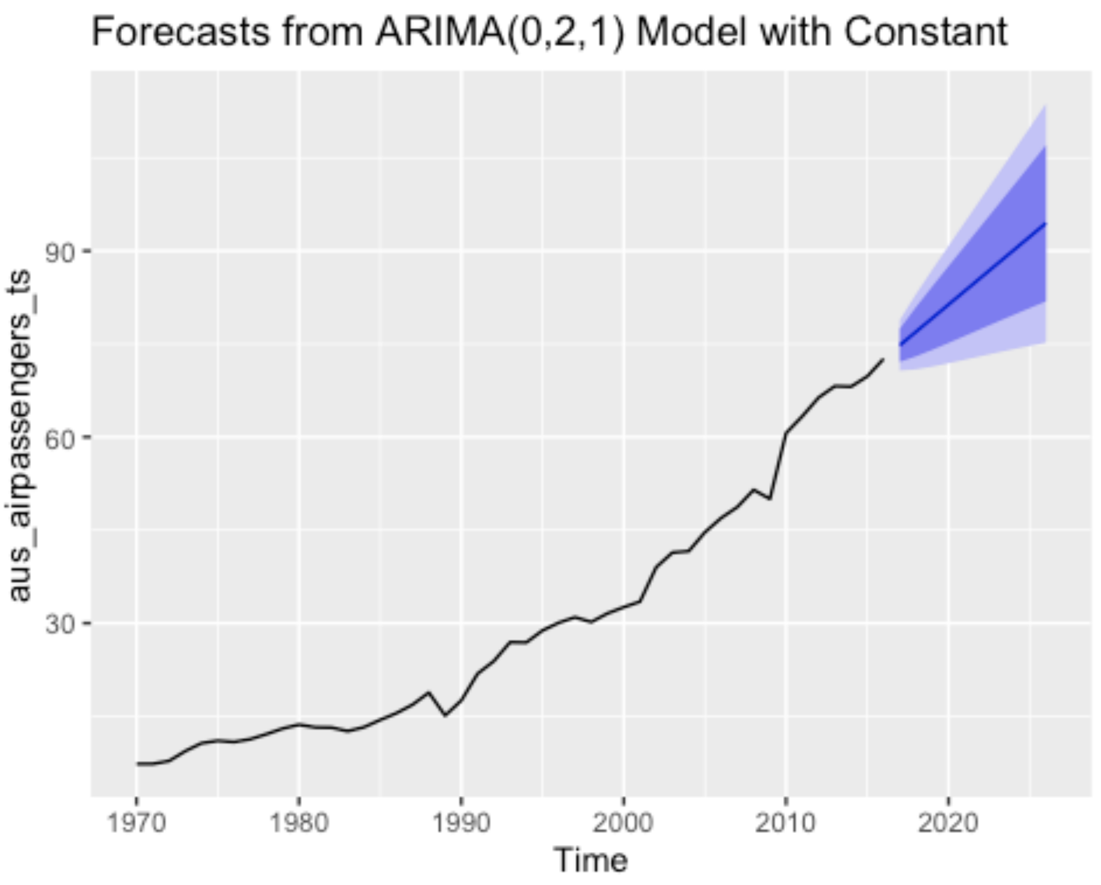
- : The forecast from this model shows a steeper increase in the number of passengers for the next 10 periods compared to parts (a) and (c). The inclusion of drift (representing a trend) leads to a more pronounced upward trajectory.
- : Removing the constant results in a forecast that still trends upward, but with a more modest increase compared to the model with drift. The predictions are less optimistic without the inclusion of drift.
- : The ARIMA(0,1,0) model in part (c) produced a more linear and moderate forecast. The ARIMA(2,1,2) with drift, however, shows a sharper increase in future periods. The model in part (a) (ARIMA(0,2,1)) is somewhat comparable, but the ARIMA(2,1,2) model affects the predictions more strongly due to its higher order.

In summary, the ARIMA(2,1,2) model with drift provides the most aggressive forecast, while removing the constant leads to a less steep increase. Both are more optimistic than the models in parts (a) and (c).

e. Plot forecasts from ARIMA(0,2,1) model with constant

```r
# Converting aus_airpassengers to a time series object
aus_airpassengers_ts <- ts(aus_airpassengers$Passengers, start = 1970,
frequency = 1)
```

```r
# Fitting ARIMA(0,2,1) model with constant using ML
fit_e <- Arima(aus_airpassengers_ts, order = c(0,2,1), include.constant =
TRUE, method = "ML")

# Forecasting next 10 periods
fc_e <- forecast(fit_e, h = 10)

# Plotting the forecasts
autoplot(fc_e) + ggtitle("Forecasts from ARIMA(0,2,1) Model with Constant")
```



- **ARIMA(0,2,1) Model with Constant**: The forecast from this model continues the upward trend in the number of passengers. Including a constant in the model helps maintain the increasing pattern, but the slope of the forecast is moderate compared to models with higher-order terms. The inclusion of the constant smooths the forecast while capturing the long-term upward trajectory observed in the historical data.

- **What happens?**: The forecast produces a steady increase, but the rise is not as sharp as in some of the higher-order ARIMA models (such as ARIMA(2,1,2)). The constant term helps the model capture the upward trend in the data, leading to a smoother forecast, maintaining consistency with past patterns.

Exercise 8: Using United States GDP series (from global_economy)

a. Box-Cox transformation

```
# Filtering for United States GDP data from the global_economy dataset
us_gdp <- global_economy %>% filter(Country == "United States")

# Checking for a suitable Box-Cox transformation
lambda <- BoxCox.lambda(us_gdp$GDP)

# Applying Box-Cox transformation if needed
us_gdp_boxcox <- BoxCox(us_gdp$GDP, lambda)
```

b. Fit a suitable ARIMA model to the transformed data

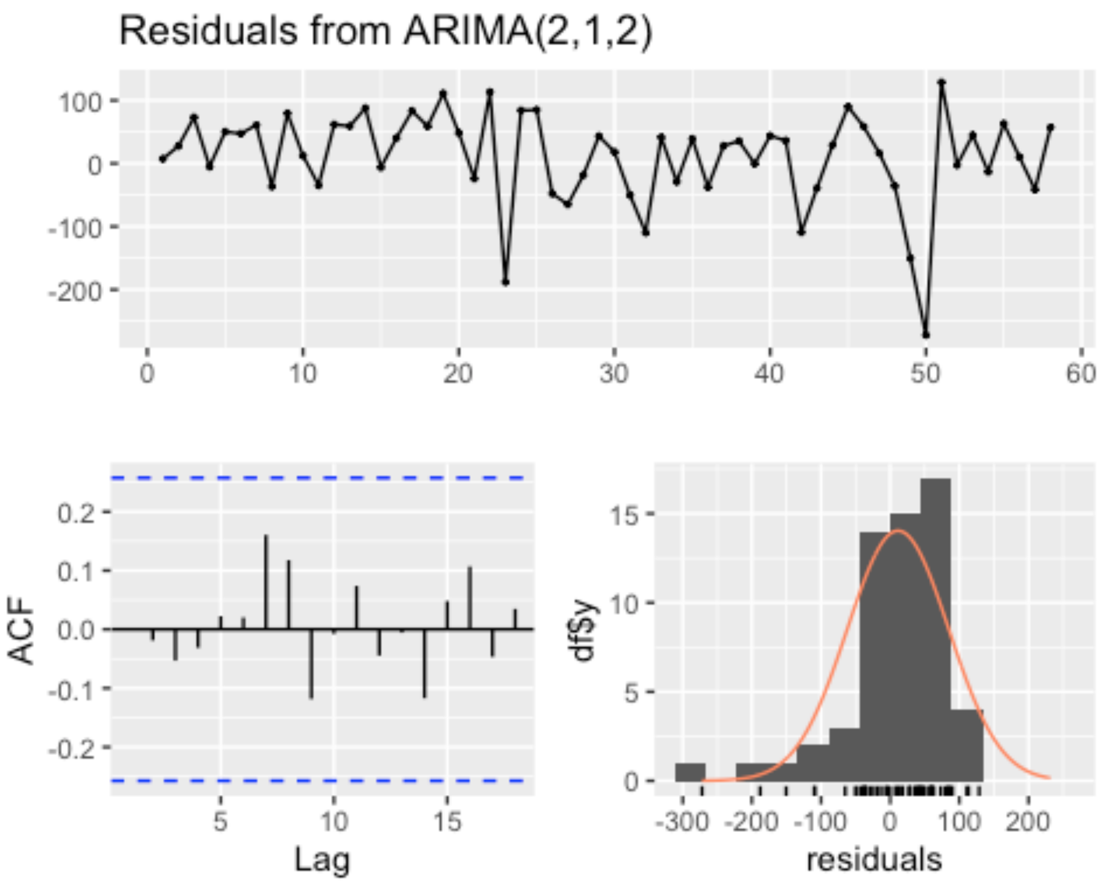
```
# Fitting ARIMA model on the transformed data (or raw data if no
transformation is needed)
fit_arima <- auto.arima(us_gdp_boxcox)

# Summary of the ARIMA model
summary(fit_arima)

## Series: us_gdp_boxcox
## ARIMA(1,1,0) with drift
##
## Coefficients:
##          ar1     drift
##       0.4586  218.4542
## s.e.  0.1198   17.5690
##
## sigma^2 = 5488:  log likelihood = -325.37
## AIC=656.74   AICc=657.19   BIC=662.87
##
## Training set error measures:
##                    ME     RMSE      MAE          MPE      MAPE      MASE
## Training set 1.540306 72.14009 54.34684 0.0007634341 0.4284988 0.2423587
##                     ACF1
## Training set -0.02316701
```

c. Try other plausible models

```
# Manually fitting ARIMA models with different orders
fit_arima_alt1 <- Arima(us_gdp_boxcox, order = c(2,1,2))
fit_arima_alt2 <- Arima(us_gdp_boxcox, order = c(0,1,1))
fit_arima_alt3 <- Arima(us_gdp_boxcox, order = c(1,1,1))

# Comparing models
AIC(fit_arima, fit_arima_alt1, fit_arima_alt2, fit_arima_alt3)

##                df      AIC
## fit_arima       3 656.7387
## fit_arima_alt1  5 665.4224
## fit_arima_alt2  2 739.0234
## fit_arima_alt3  3 664.8958
```

d. Choose the best model and check residuals

```
# Checking residuals for the selected model
checkresiduals(fit_arima_alt1) # Replacing with the best model
```



```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)
## Q* = 4.0474, df = 6, p-value = 0.6703
##
## Model df: 4.   Total lags used: 10
```

Based on the outputs from part d, we chose the ARIMA(2,1,2) model for the following reasons:

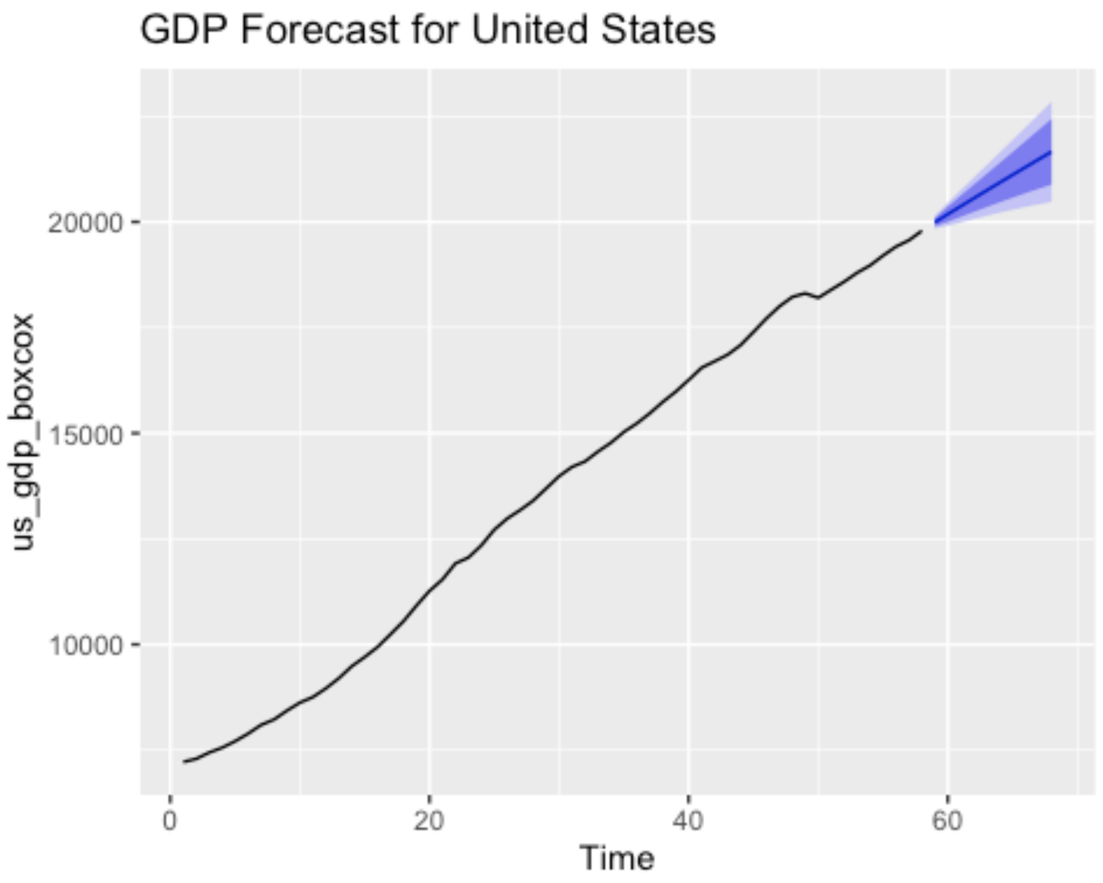
- The residuals appear to be randomly scattered around zero with no significant pattern, as shown in the residual plot, indicating that the model has captured most of the structure in the data.

- The ACF plot of the residuals shows no significant autocorrelations, suggesting that the residuals are uncorrelated and behave like white noise.
- The Ljung-Box test yields a p-value of 0.6703, which is greater than 0.05, meaning that we fail to reject the null hypothesis that the residuals are independent. This further confirms that the model's residuals are not autocorrelated.

These diagnostics indicate that the ARIMA(2,1,2) model provides a good fit for the data compared to other models considered.

e. Produce forecasts from your fitted model

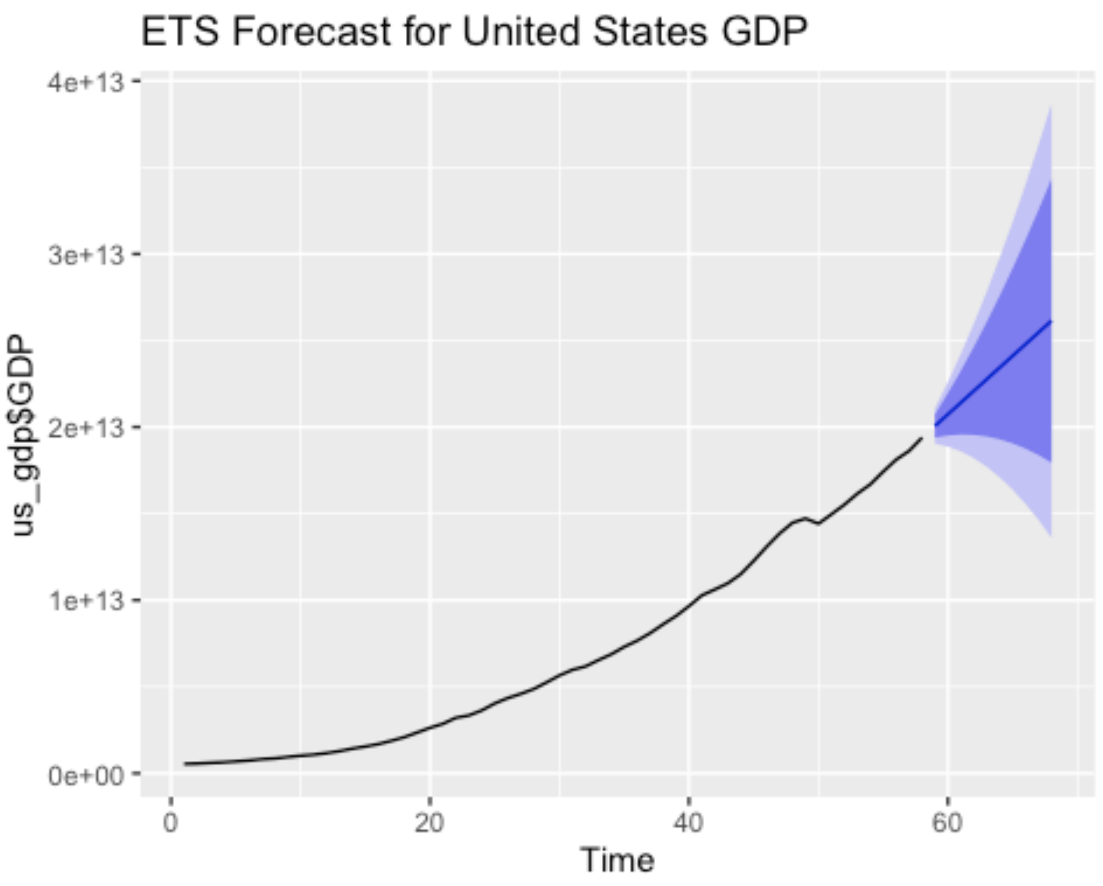
```r
# Forecasting the next 10 periods using the chosen model
forecast_arima <- forecast(fit_arima_alt1, h = 10)

# Plotting the forecast
autoplot(forecast_arima) + ggtitle("GDP Forecast for United States")
```



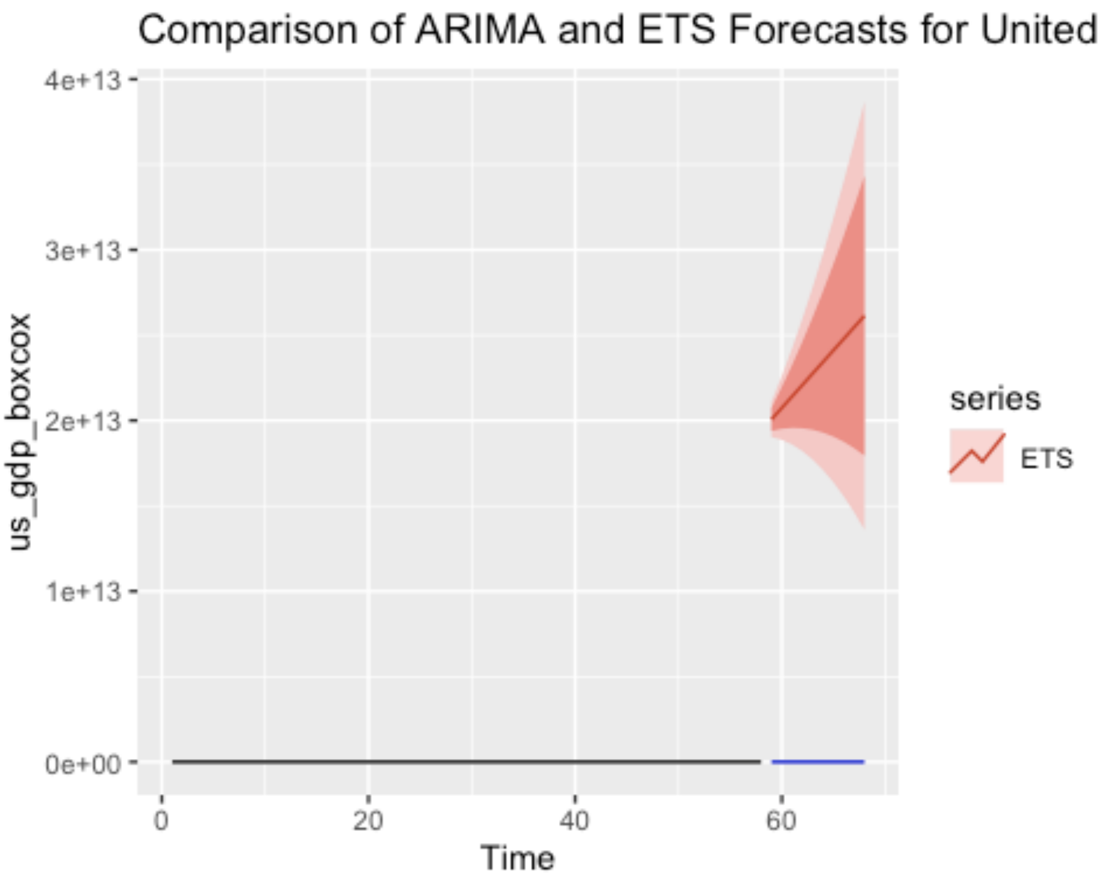
- The forecast for the next 10 periods produced by the fitted ARIMA(2,1,2) model shows a reasonable continuation of the upward trend in the GDP series. The confidence intervals (shaded in blue) indicate uncertainty around the forecast, and they widen as the forecast horizon increases, which is expected in time series forecasting. Given the historical pattern of steady growth in the data, the forecast follows a plausible trajectory, suggesting that the model captures the underlying trend in the United States GDP well.

f. Compare the results with ETS

```r
# Fitting an ETS model and compare with ARIMA
fit_ets <- ets(us_gdp$GDP)

# Forecasting the next 10 periods using the ETS model
forecast_ets <- forecast(fit_ets, h = 10)

# Plotting the ETS forecast
autoplot(forecast_ets) + ggtitle("ETS Forecast for United States GDP")
```



```r
# Comparing ARIMA and ETS forecasts
autoplot(forecast_arima, series = "ARIMA") +
  autolayer(forecast_ets, series = "ETS") +
  ggtitle("Comparison of ARIMA and ETS Forecasts for United States GDP")
```



- The forecasts from both the ARIMA and ETS models predict an upward trend in the United States GDP over the next 10 periods. However, the ETS model displays wider confidence intervals, indicating greater uncertainty compared to the ARIMA model, particularly at longer time horizons. The ARIMA model provides tighter bounds, suggesting more confidence in its forecasted values. Despite this difference in uncertainty, both models are in general agreement on the direction of GDP growth.