DATA 624 Homework 6

Fomba Kassoh

2024-10-17

Contents

{r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE)

```r
# Install required packages if they are not already installed
if (!require("fpp3")) install.packages("fpp3", dependencies=TRUE)
```

```
## Loading required package: fpp3
```

```
## Registered S3 method overwritten by 'tsibble':
##   method               from
##   as_tibble.grouped_df dplyr
```

```
## -- Attaching packages --------------------------------------- fpp3 1.0.0 --
```

```
## v tibble      3.2.1     v tsibble     1.1.5
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.3.2
## v lubridate   1.9.3     v fable       0.3.4
## v ggplot2     3.5.1     v fabletools  0.4.2
```

```
## -- Conflicts ------------------------------------------ fpp3_conflicts --
## x lubridate::date()    masks base::date()
## x dplyr::filter()      masks stats::filter()
```

```
## x tsibble::intersect() masks base::intersect()
## x tsibble::interval()  masks lubridate::interval()
## x dplyr::lag()         masks stats::lag()
## x tsibble::setdiff()   masks base::setdiff()
## x tsibble::union()     masks base::union()
```

```r
# Load required libraries
library(fpp3)
```

0.1 Exercise 9.1

0.1.1 1. Do they all indicate that the data are white noise?

Yes, all three plots indicate that the data are likely **white noise**. In a white noise process:

- **White noise** means that on average, the true autocorrelations for all lags should be zero.
- In all three plots, the ACF values fluctuate around zero.
- None of the individual lags crossed the significance bounds (blue dashed lines).

Therefore, all the ACF plots support the conditions that the data are white noise.

0.1.2 2. Why are the critical values at different distances from the mean of zero?

The **critical values** (the blue dashed lines) represent the confidence interval for the autocorrelation coefficients. These lines typically indicate a 95% confidence interval for testing whether the autocorrelations are significantly different from zero. The distance of the critical values from zero depends on the **sample size** (n). Specifically:

- For a white noise process, the confidence bounds are approximately **2/sqrt(n)**, where **n** is the number of observations.
- This formula indicates that as **n** increases, the confidence intervals shrink because larger sample sizes provide more precise estimates.

For the three plots:

- **36 random numbers**: The critical bounds are farther from zero because the sample size is small.
- **360 random numbers**: The critical bounds are closer to zero because of the larger sample size.
- **1,000 random numbers**: The critical bounds are even closer to zero because the sample size is much larger.

0.1.3 3. Why are the autocorrelations different in each figure when they each refer to white noise?

Although all three datasets represent white noise, the autocorrelations differ due to **random sampling variation** and the **finite sample size**.

0.2 Exercise 9.2 Plot the daily closing prices for Amazon stock (contained in gafa_stock), along with the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.

The three plots below—the **closing price plot**, **ACF plot**, and **PACF plot**—together strongly indicate the need for **differencing** to make the series stationary.

1. **Upward Trend in the Daily Closing Prices**: The time series plot of Amazon's daily closing prices (Figure 1) shows a clear upward trend, indicating that the series is non-stationary. A stationary series should fluctuate around a constant mean without any long-term trend, but this is not the case here.
2. **High Persistence in ACF**: The autocorrelation function (ACF) plot (Figure 2) shows slow decay, with high autocorrelation values at many lags. This is a characteristic sign of non-stationarity, as a stationary series would typically have much lower autocorrelations after a few lags.
3. **Significant Spike at Lag 1 in PACF**: The partial autocorrelation function (PACF) plot (Figure 3) shows a significant spike at lag 1 and smaller spikes at later lags, which often suggests that differencing is needed to remove the trend and make the series stationary.
4. **Need for Differencing**: Both the time series plot and the behavior of the ACF and PACF plots strongly suggest that the series should be differenced to remove the trend and autocorrelation, making the series stationary and more suitable for modeling with ARIMA or other techniques.

```
# Install required packages if not already installed
if (!require("patchwork")) install.packages("patchwork")
```

```
## Loading required package: patchwork
```

```
# Load necessary libraries
library(fable)
library(dplyr)
library(ggplot2)
library(tsibble)
library(patchwork)

# Extract Amazon stock data from the gafa_stock dataset
amazon_stock <- gafa_stock |>
  filter(Symbol == "AMZN")

# Plot the daily closing prices for Amazon
plot1 <- amazon_stock |>
  autoplot(Close) +
  labs(title = "Figure 1: Amazon Daily Closing Prices", y = "Price", x = "Date")

# ACF and PACF plots for the daily closing prices
plot2 <- amazon_stock |>
  ACF(Close) |>
  autoplot() +
  labs(title = "Figure 2: Amazon Daily Closing Prices")
```

```
## Warning: Provided data has an irregular interval, results should be treated
## with caution. Computing ACF by observation.
```

```
plot3 <- amazon_stock |>
  PACF(Close) |>
  autoplot() +
  labs(title = "Figure 3: Amazon Daily Closing Prices")
```

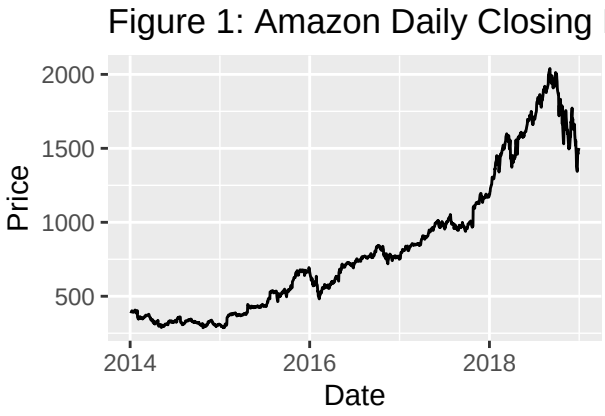
```
## Warning: Provided data has an irregular interval, results should be treated
## with caution. Computing ACF by observation.
```

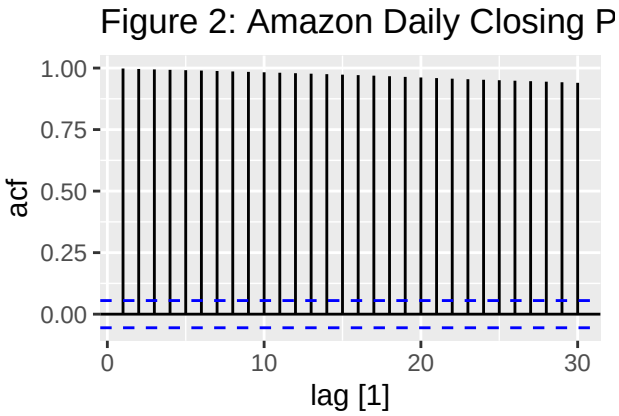
```
# Display the two plots
library(gridExtra)
```

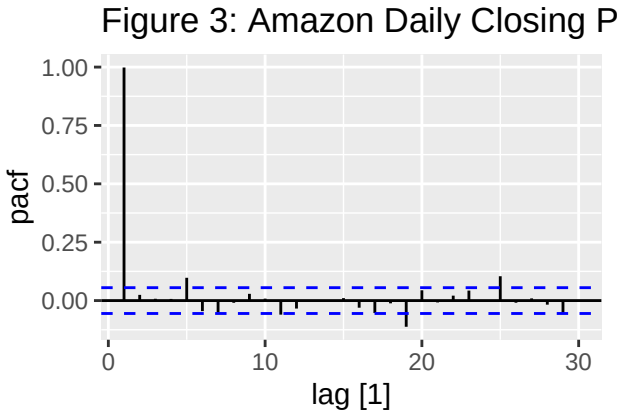
```
##
## Attaching package: 'gridExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
grid.arrange(plot1, plot2, plot3, ncol = 2)
```







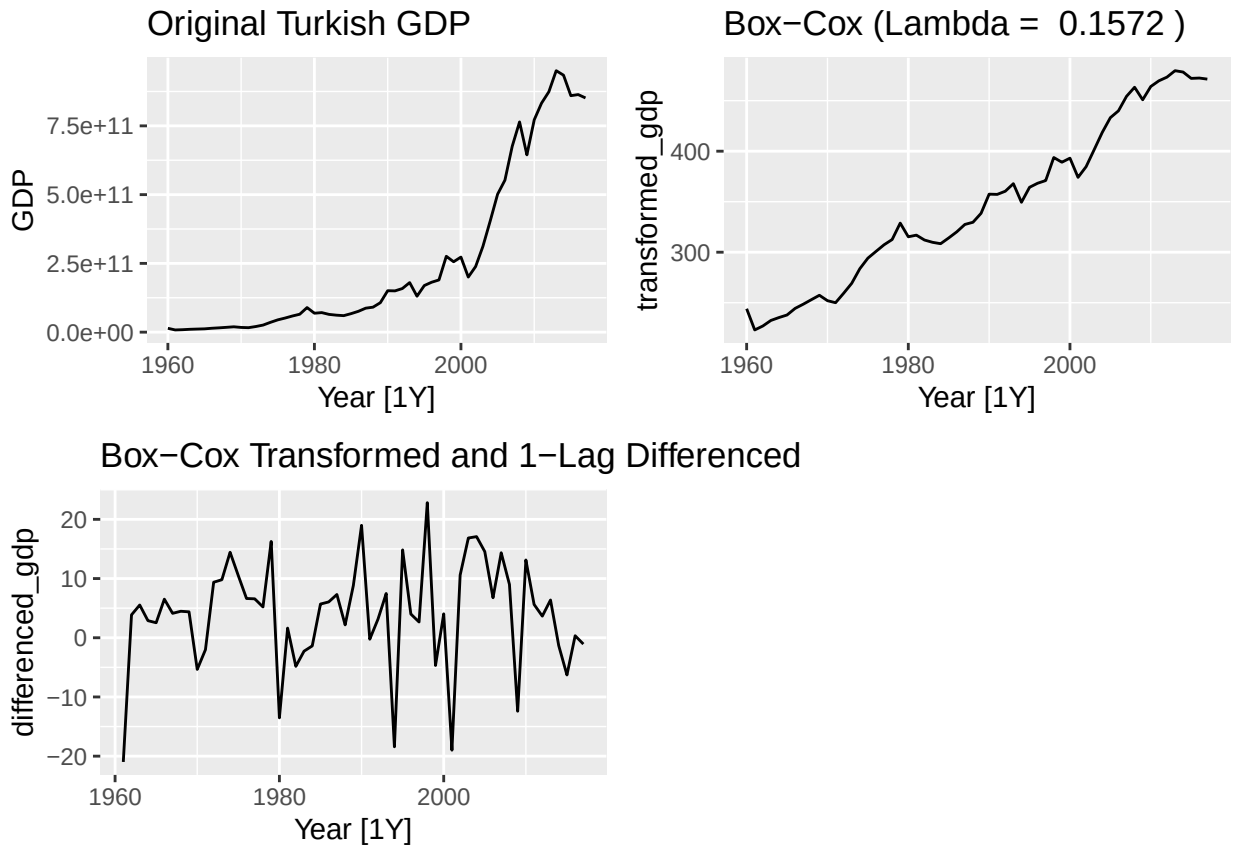
0.3 Exercise 9.3: For the following series, find an appropriate Box-Cox transformation and order of differencing in order to obtain stationary data.

0.3.1 a. Turkish GDP from `global_economy`.

0.3.1.0.1 i. Calculate the optimal Box-Cox lambda and plot the original and transformed series

- Box-Cox transformation (with $\lambda = 0.1572$) and a 1-Lag differencing (d = 1) is required to make the series stationary.
- The **Original Turkish GDP** shows a strong upward trend, indicating non-stationarity, which suggests that transformations or differencing would be required for effective modeling or forecasting.
- The **Box-Cox Transformed Turkish GDP** (with $\lambda = 0.1572$) reduces some of the variance and smooths the data, but it still retains a visible trend, indicating that further differencing might be needed to achieve stationarity.
- The **Transformed and Differenced Turkish GDP** shows fluctuations around a constant mean and no apparent trend, suggesting that differencing has successfully removed the trend and the series is likely stationary.

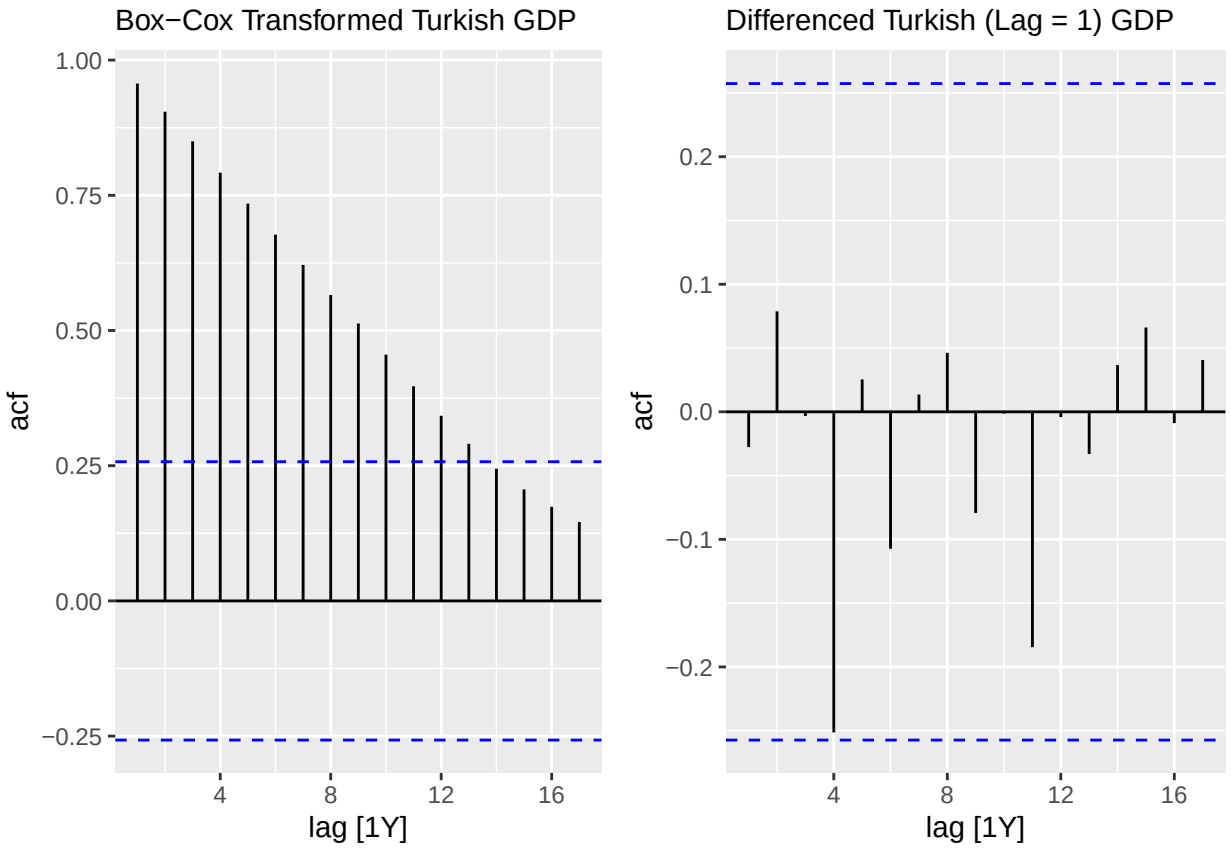
```
# Calculate the optimal Box-Cox lambda using fable's features function
gdp_lambda <- global_economy |>
  filter(Country == "Turkey") |>
  features(GDP, features = guerrero) |>
  pull(lambda_guerrero)

# Apply the Box-Cox transformation
turkish_gdp <- global_economy |>
  filter(Country == "Turkey") |>
  mutate(transformed_gdp = box_cox(GDP, gdp_lambda)) |>
  mutate(differenced_gdp = difference(transformed_gdp))

# Plot the original series
p1 <- turkish_gdp |>
  autoplot(GDP) + labs(title = "Original Turkish GDP")

# Plot the Box-Cox transformed series
p2 <- turkish_gdp |>
  autoplot(transformed_gdp) + labs(title = paste("Box-Cox (Lambda = ",round(gdp_lambda, 4), ")"))

# Plot the differenced series, dropping NA values
p3 <- turkish_gdp |>
  drop_na(differenced_gdp) |>
  autoplot(differenced_gdp) + labs(title = paste("Box-Cox Transformed and 1-Lag Differenced"))


# Display the two plots
library(gridExtra)
grid.arrange(p1, p2, p3, ncol = 2)
```



0.3.1.1 ii. ACF of Box-Cox transformation and differencing analysis to verify stationarity

- The ACF of the original series shows slow decay, indicating non-stationarity and ruling out white noise.
- After differencing, most autocorrelations are close to zero and within the 95% confidence interval, suggesting the series has become stationary.
- The Ljung-Box test with a p-value of 0.8289 confirms no significant autocorrelation, indicating the differenced series resembles white noise.

```
plot1 <- turkish_gdp |> ACF(transformed_gdp) |>
  autoplot() + labs(subtitle = "Box-Cox Transformed Turkish GDP")

plot2 <- turkish_gdp |> ACF(difference(transformed_gdp)) |>
  autoplot() + labs(subtitle = "Differenced Turkish (Lag = 1) GDP")


# Display the two plots
library(gridExtra)
grid.arrange(plot1, plot2, ncol = 2)
```




```
turkish_gdp |>
  mutate(diff_gdp = difference(transformed_gdp)) |>
  features(diff_gdp, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 3
##   Country lb_stat lb_pvalue
##   <fct>     <dbl>     <dbl>
## 1 Turkey     5.84     0.829
```

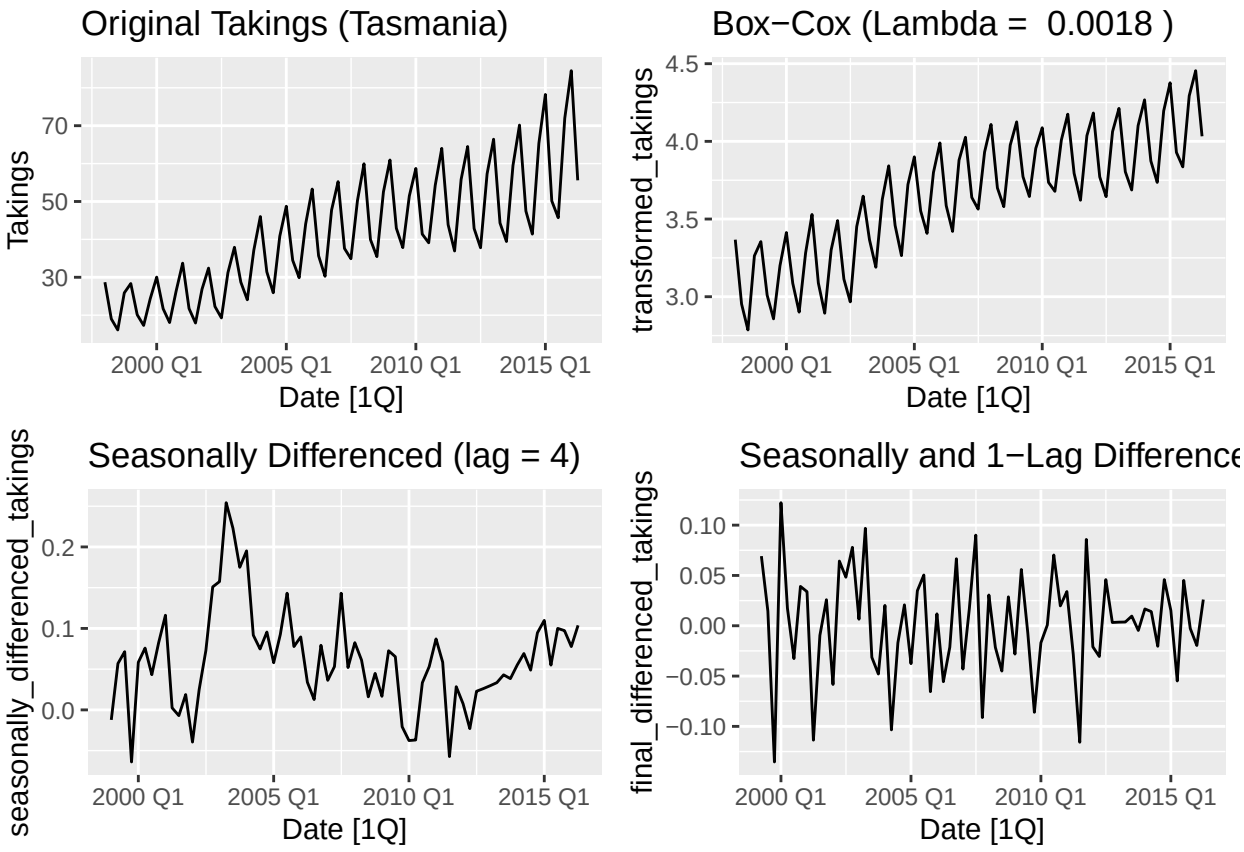
0.3.2 b. Accommodation takings in the state of Tasmania from `aus_accommodation`.

0.3.2.1 i. Calculate the optimal Box-Cox lambda and plot the original and transformed series

- Box-Cox transformation (with $\lambda = 0.0018$) and a seasonal (Lag =4) and non-seasonal (Lag=1) differencing (d = 2) is required to make the series stationary.
- The **Original Accommodation Takings (Tasmania)** shows a strong upward trend, clear seasonality, and increasing variance over time, indicating non-stationarity that may require transformation or differencing for effective modeling.
- The **Box-Cox Transformed Takings** (with $\lambda = 0.0018$) stabilizes the variance to some extent, but the series still retains a visible trend and seasonality, suggesting further differencing is needed to achieve stationarity.
- The **Seasonally Differenced Takings** removes the repeating seasonal pattern, but fluctuations still remain, indicating that additional differencing may be necessary to fully achieve stationarity.

- The **Seasonally and 1-Lag Differenced Takings** shows fluctuations around a constant mean with reduced variance and no obvious trend, suggesting that the series has now likely achieved stationarity.

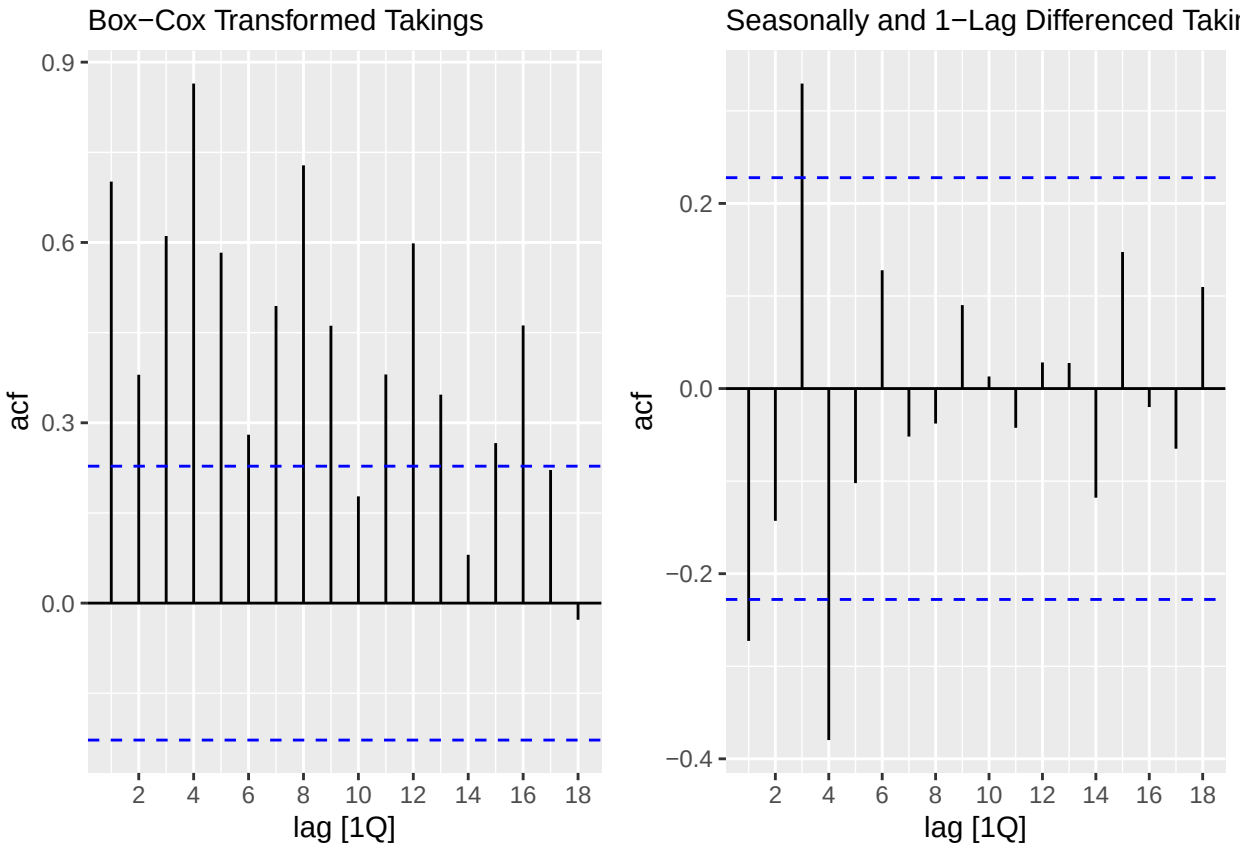
```
# Calculate the optimal Box-Cox lambda using fable's features function
takings_lambda <- aus_accommodation |>
  filter(State == "Tasmania") |>
  features(Takings, features = guerrero) |>
  pull(lambda_guerrero)

# Apply the Box-Cox transformation, seasonal differencing (lag = 4 for quarterly data), and then apply
tas_accommodation <- aus_accommodation |>
  filter(State == "Tasmania") |>
  mutate(transformed_takings = box_cox(Takings, takings_lambda)) |>
  mutate(seasonally_differenced_takings = difference(transformed_takings, lag = 4)) |>
  mutate(final_differenced_takings = difference(seasonally_differenced_takings, lag = 1))

# Plot the original series
p1 <- tas_accommodation |>
  autoplot(Takings) + labs(title = "Original Takings (Tasmania)")

# Plot the Box-Cox transformed series
p2 <- tas_accommodation |>
  autoplot(transformed_takings) + labs(title = paste("Box-Cox (Lambda = ",round(takings_lambda, 4), ")")

# Plot the seasonally differenced series
p3 <- tas_accommodation |>
  drop_na(seasonally_differenced_takings) |>
  autoplot(seasonally_differenced_takings) + labs(title = "Seasonally Differenced (lag = 4)")

# Plot the final series with both seasonal and 1-lag differencing
p4 <- tas_accommodation |>
  drop_na(final_differenced_takings) |>
  autoplot(final_differenced_takings) + labs(title = "Seasonally and 1-Lag Differenced")

# Display the two plots
library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol = 2)
```



0.3.2.2 ii. ACF of Box-Cox transformation and differencing analysis to verify stationarity

- The **ACF of the Box-Cox Transformed series** shows a slow decay with significant autocorrelations at several lags, indicating non-stationarity and ruling out white noise.
- After applying **seasonal and 1-lag differencing**, most autocorrelations are close to zero and within the 95% confidence interval, suggesting that the series has likely become stationary.
- The **Ljung-Box test** with a p-value of 4.63e-07 indicates significant autocorrelation remains in the series, suggesting that the differencing has not completely removed all dependencies. This suggests that any model needs a seasonal component as well as potentially autoregressive (AR) and moving average (MA) terms to capture the remaining structure.

```
# Plot ACF of Box-Cox transformed series
p1 <- tas_accommodation |>
  ACF(transformed_takings) |>
  autoplot() + labs(subtitle = "Box-Cox Transformed Takings")

# Plot ACF of seasonally differenced series
p2 <- tas_accommodation |>
  ACF(final_differenced_takings) |>
  autoplot() + labs(subtitle = "Seasonally and 1-Lag Differenced Takings")

# Display the two plots
library(gridExtra)
grid.arrange(p1, p2, ncol = 2)
```




```
# Calculate and apply Ljung-Box test to the final differenced series
tas_accommodation |>
  mutate(diff_takings = difference(final_differenced_takings, lag = 1)) |>
  features(diff_takings, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 3
##   State    lb_stat   lb_pvalue
##   <chr>      <dbl>       <dbl>
## 1 Tasmania    48.7 0.000000463
```

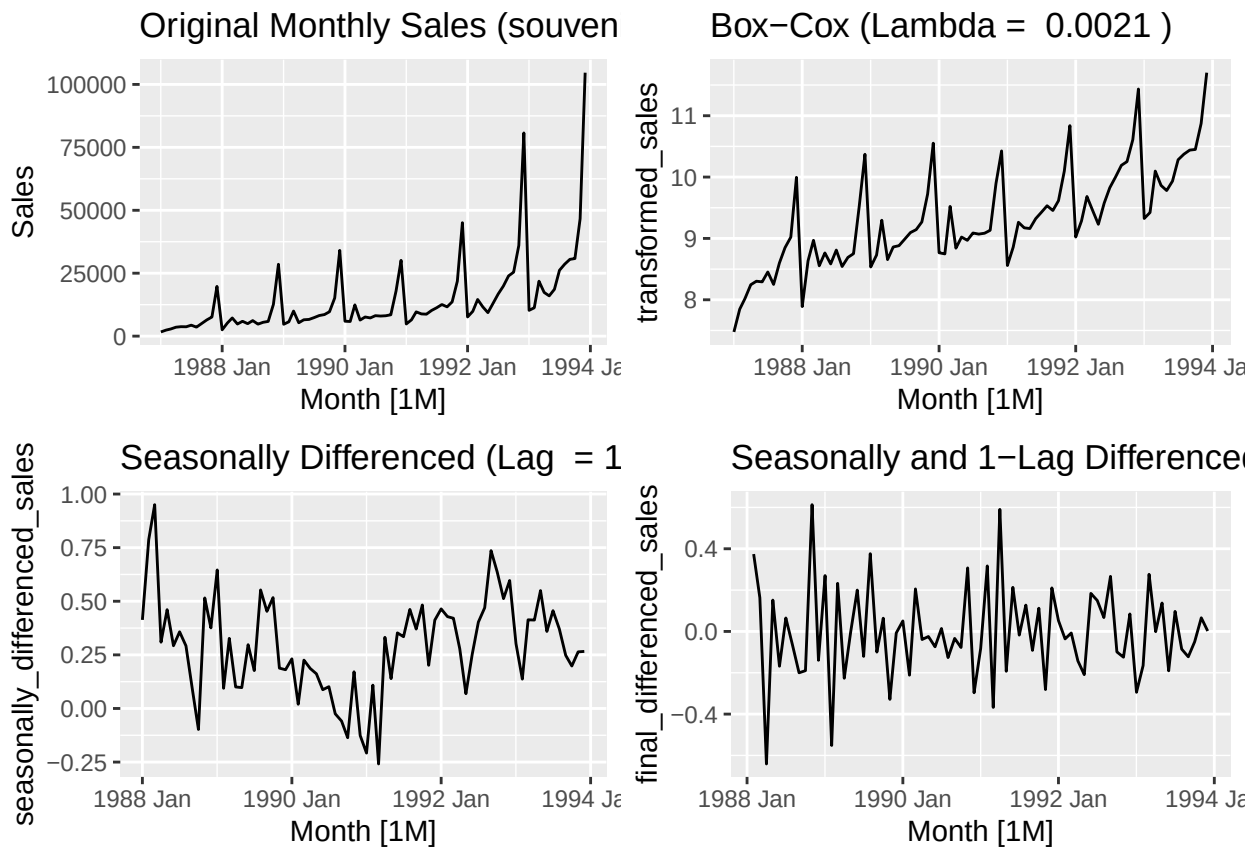
0.3.3 c. Monthly sales from `souvenirs`.

0.3.3.1 i. Calculate the optimal Box-Cox lambda and plot the original and transformed series

- Box-Cox transformation (with $\lambda = 0.0021$) and a seasonal (Lag = 12) and non-seasonal (Lag = 1) differencing (d = 2) is required to make the series stationary.
- The **Original Monthly Sales (Souvenirs)** shows a strong upward trend, clear seasonality, and increasing variance over time, indicating that transformation and differencing are required to achieve stationarity.
- The **Box-Cox Transformed Sales** (with $\lambda = 0.0021$) reduces the variance but retains the trend and seasonality, suggesting that differencing is still necessary to make the series stationary.

- The **Seasonally Differenced Sales** removes much of the seasonal pattern but still shows fluctuations, indicating that additional lag-1 differencing may be required to fully address stationarity.
- The **Seasonally and 1-Lag Differenced Sales** series fluctuates around a constant mean with reduced trend and variance, indicating that the series has likely achieved stationarity and is ready for ARIMA modeling.

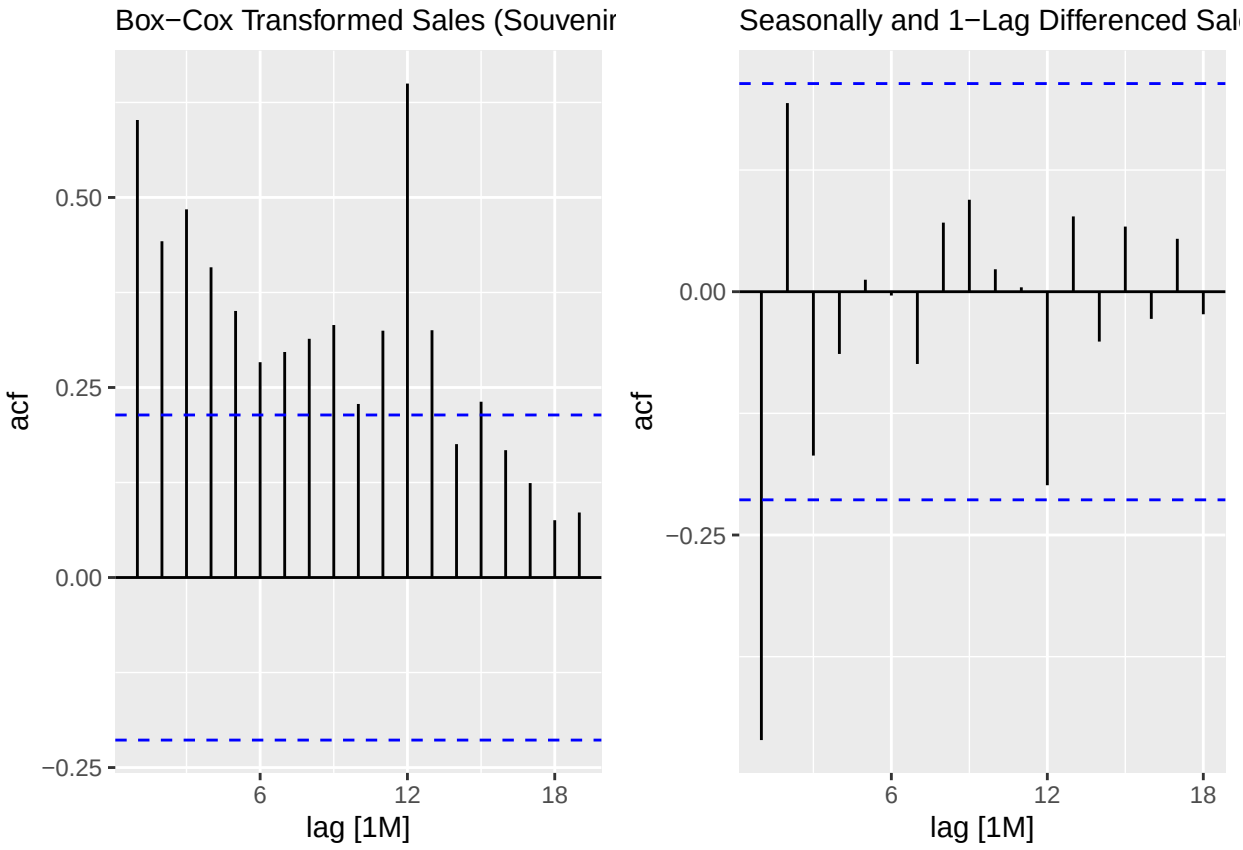
```
# Calculate the optimal Box-Cox lambda using fable's features function
sales_lambda <- souvenirs |>
  features(Sales, features = guerrero) |>
  pull(lambda_guerrero)

# Apply the Box-Cox transformation, seasonal differencing (lag = 12 for monthly data), and then 1-lag d
souvenir_sales <- souvenirs |>
  mutate(transformed_sales = box_cox(Sales, sales_lambda)) |>
  mutate(seasonally_differenced_sales = difference(transformed_sales, lag = 12)) |>
  mutate(final_differenced_sales = difference(seasonally_differenced_sales, lag = 1))

# Plot the original series
p1 <- souvenir_sales |>
  autoplot(Sales) + labs(title = "Original Monthly Sales (souvenirs)")

# Plot the Box-Cox transformed series with lambda rounded to 4 decimal places
p2 <- souvenir_sales |>
  autoplot(transformed_sales) +
  labs(title = paste("Box-Cox (Lambda = ",round(sales_lambda, 4), ")"))


# Plot the seasonally differenced series (dropping NAs)
p3 <- souvenir_sales |>
  drop_na(seasonally_differenced_sales) |>
  autoplot(seasonally_differenced_sales) + labs(title = "Seasonally Differenced (Lag  = 12)")

# Plot the final series with both seasonal and 1-lag differencing (dropping NAs)
p4 <- souvenir_sales |>
  drop_na(final_differenced_sales) |>
  autoplot(final_differenced_sales) + labs(title = "Seasonally and 1-Lag Differenced")

# Display the four plots using gridExtra
library(gridExtra)
grid.arrange(p1, p2, p3, p4, ncol = 2)
```



0.3.3.2 ii. ACF of Box-Cox transformation and differencing analysis to verify stationarity

- The **ACF of the Box-Cox Transformed Series** shows strong seasonal autocorrelation at lag 12, indicating the presence of seasonality in the data.
- The **ACF of the Seasonally and 1-Lag Differenced Series** shows most autocorrelations near zero, suggesting that the series has become more stationary, although some autocorrelation remains.
- The **Ljung-Box test** p-value of 0.012 indicates that some autocorrelation persists, implying that a **Seasonal ARIMA (SARIMA)** model could be a suitable option to better capture the seasonal and residual patterns in the data.

```
# Plot ACF of Box-Cox transformed series
p1 <- souvenir_sales |>
  ACF(transformed_sales) |>
  autoplot() + labs(subtitle = "Box-Cox Transformed Sales (Souvenirs)")

# Plot ACF of seasonally differenced series
p2 <- souvenir_sales |>
  ACF(final_differenced_sales) |>
  autoplot() + labs(subtitle = "Seasonally and 1-Lag Differenced Sales")

# Display the four plots using gridExtra
library(gridExtra)
grid.arrange(p1, p2, ncol = 2)
```




```
# Apply Ljung-Box test to the final differenced series
souvenir_sales |>
  features(final_differenced_sales, ljung_box, lag = 10)
```

```
## # A tibble: 1 x 2
##   lb_stat lb_pvalue
##     <dbl>     <dbl>
## 1    22.7    0.0120
```

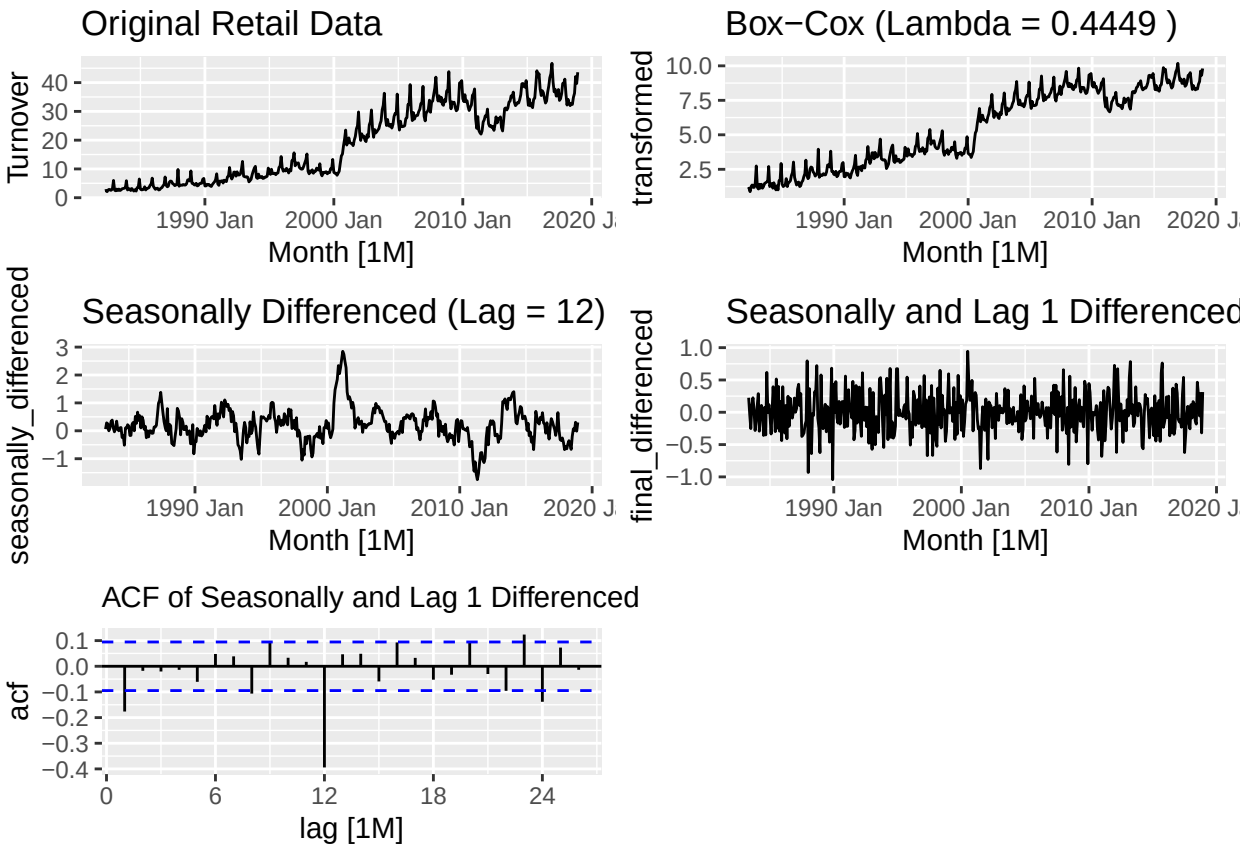
0.4 Excercise 9.5: For your retail data (from Exercise 7 in Section 2.10), find the appropriate order of differencing (after transformation if necessary) to obtain stationary data.

- Box-Cox transformation (with $\lambda = 0.4449$) and a seasonal (Lag = 12) and non-seasonal (Lag = 1) differencing (d = 2) is required to make the series stationary.
- The **Original Australian Retail Data** shows a strong upward trend and increasing variance over time, indicating non-stationarity, requiring transformation or differencing for effective modeling.
- The **Box-Cox Transformed Turnover** (with $\lambda = 0.4449$) stabilizes the variance somewhat, but the upward trend persists, suggesting that further differencing may be necessary to achieve stationarity.
- The **Seasonally Differenced Turnover** (with a lag of 12 for monthly data) removes much of the seasonality but still shows some irregular patterns that might indicate non-stationarity.

- The **Seasonally and Non-Seasonally Differenced Turnover** series fluctuates around a constant mean and shows no apparent trend, indicating that differencing successfully removed the trend and seasonality.
- The **ACF of Seasonally and Non-Seasonally Differenced Turnover** shows most autocorrelations within the 95% confidence intervals, implying that the series is likely stationary and ready for ARIMA modeling.
- The Ljung-Box test with a p-value of 0.003 indicates some remaining autocorrelation, suggesting that a seasonal ARIMA model might be needed to account for any remaining structure in the residuals.

```
# Set a seed for reproducibility
set.seed(678)

# Select a random series from the aus_retail dataset
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))

# Plot the original series to visually inspect trends and seasonality
p1 <- myseries |>
  autoplot(Turnover) + labs(title = "Original Retail Data")

# Step 1: Calculate the optimal Box-Cox lambda (if transformation is needed to stabilize variance)
retail_lambda <- myseries |>
  features(Turnover, features = guerrero) |>
  pull(lambda_guerrero)

# Step 2: Apply the Box-Cox transformation
myseries <- myseries |>
  mutate(transformed = box_cox(Turnover, retail_lambda))

# Plot the transformed series
p2 <- myseries |>
  autoplot(transformed) + labs(title = paste("Box-Cox (Lambda =", round(retail_lambda, 4), ")"))

# Step 3: Perform seasonal differencing (lag = 12 for monthly data) to remove seasonality
myseries <- myseries |>
  mutate(seasonally_differenced = difference(transformed, lag = 12)) |>
  drop_na(seasonally_differenced)

# Plot the seasonally differenced series
p3 <- myseries |>
  autoplot(seasonally_differenced) + labs(title = "Seasonally Differenced (Lag = 12)")

# Step 4: Perform non-seasonal differencing (lag = 1) to remove remaining trends
myseries <- myseries |>
  mutate(final_differenced = difference(seasonally_differenced, lag = 1)) |>
  drop_na(final_differenced)

# Plot the final differenced series
p4 <- myseries |>
  autoplot(final_differenced) + labs(title = "Seasonally and Lag 1 Differenced")

# Step 5: Check stationarity using ACF
p5 <- myseries |>
```

```
  ACF(final_differenced) |>
  autoplot() + labs(subtitle = "ACF of Seasonally and Lag 1 Differenced")

# Display the four plots using gridExtra
library(gridExtra)
grid.arrange(p1, p2, p3, p4, p5, nrow = 3)
```




```
# Step 6: Perform the Ljung-Box test to check for autocorrelation in residuals
myseries |>
  features(final_differenced, ljung_box, lag = 10)
```

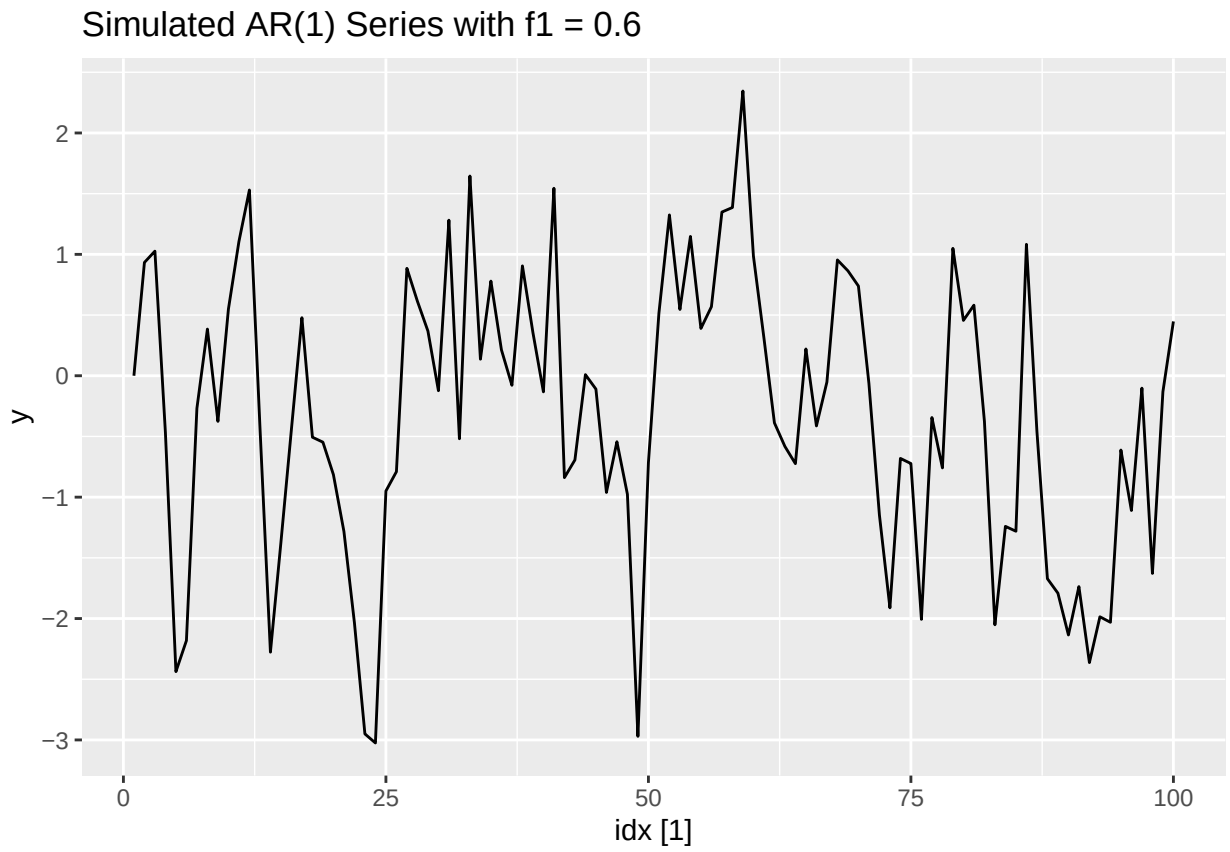
```
## # A tibble: 1 x 4
##   State    Industry                                        lb_stat lb_pvalue
##   <chr>    <chr>                                             <dbl>     <dbl>
## 1 Tasmania Hardware, building and garden supplies retailing    26.6   0.00303
```

0.5 Exercise 9.6: Simulate and plot some data from simple ARIMA models.

0.5.1 a. Use the following R code to generate data from an AR(1) model with 1 = 0.6 and ^2 = 1. The process starts with y1 = 0.

The simulated AR(1) series with $\phi_1 = 0.6$ exhibits a smoother pattern with gradual fluctuations over time, reflecting the autoregressive nature where each value depends on the previous value in the series.

```r
# Set a seed for reproducibility
set.seed(678)

y <- numeric(100)
e <- rnorm(100)
for(i in 2:100) {
  y[i] <- 0.6 * y[i-1] + e[i]
}
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

# Plot the series
autoplot(sim, y) + labs(title = "Simulated AR(1) Series with  1 = 0.6")
```



0.5.2 b. Produce a time plot for the series.

How does the plot change as you change ϕ_1?

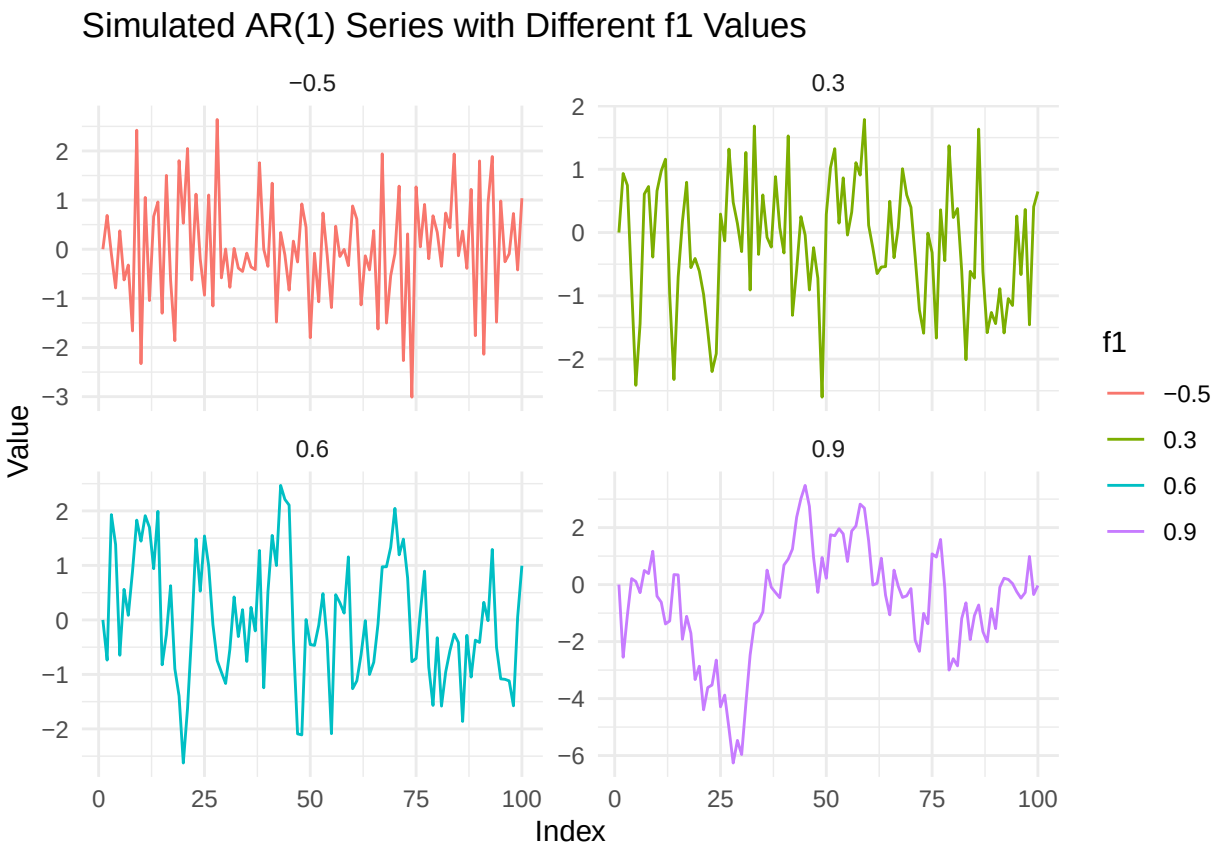
As the value of ϕ_1 changes, the behavior of the AR(1) series varies significantly:

- For $\phi_1 = -0.5$, the series exhibits more random fluctuations around zero with higher frequency reversals, producing a noisy and oscillating pattern.
- For $\phi_1 = 0.3$, the series shows weaker autocorrelation, with moderate oscillations around the mean but less persistence in the direction of movement.

- For $\phi_1 = 0.6$, the series starts to show stronger persistence, meaning that values depend more on previous ones, creating longer-lasting trends in the same direction before changing.
- For $\phi_1 = 0.9$, the autocorrelation is very strong, leading to a smooth and highly persistent series, where changes in direction are slower and the series displays more pronounced trends and less random fluctuation.

Thus, as ϕ_1 increases, the series shows greater dependence on its past values, leading to smoother trends and slower changes.

```r
library(tsibble)
library(ggplot2)
library(dplyr)

# Set a seed for reproducibility
set.seed(678)

# Define the function to simulate AR(1) process for a given phi1
simulate_ar1 <- function(phi1, n = 100) {
  y <- numeric(n)
  e <- rnorm(n)
  for (i in 2:n) {
    y[i] <- phi1 * y[i-1] + e[i]
  }
  return(as_tibble(tsibble(idx = seq_len(n), y = y, index = idx)) %>% mutate(phi = phi1))
}

# Simulate AR(1) processes for different values of  1
phi_values <- c(0.3, 0.6, 0.9, -0.5)
simulations <- lapply(phi_values, simulate_ar1)

# Combine all simulations into one data frame
combined_sim <- bind_rows(simulations)

# Plot all AR(1) simulations for comparison
ggplot(combined_sim, aes(x = idx, y = y, color = factor(phi))) +
  geom_line() +
  facet_wrap(~ phi, scales = "free_y") +
  labs(title = "Simulated AR(1) Series with Different  1 Values",
       x = "Index",
       y = "Value",
       color = " 1") +
  theme_minimal()
```

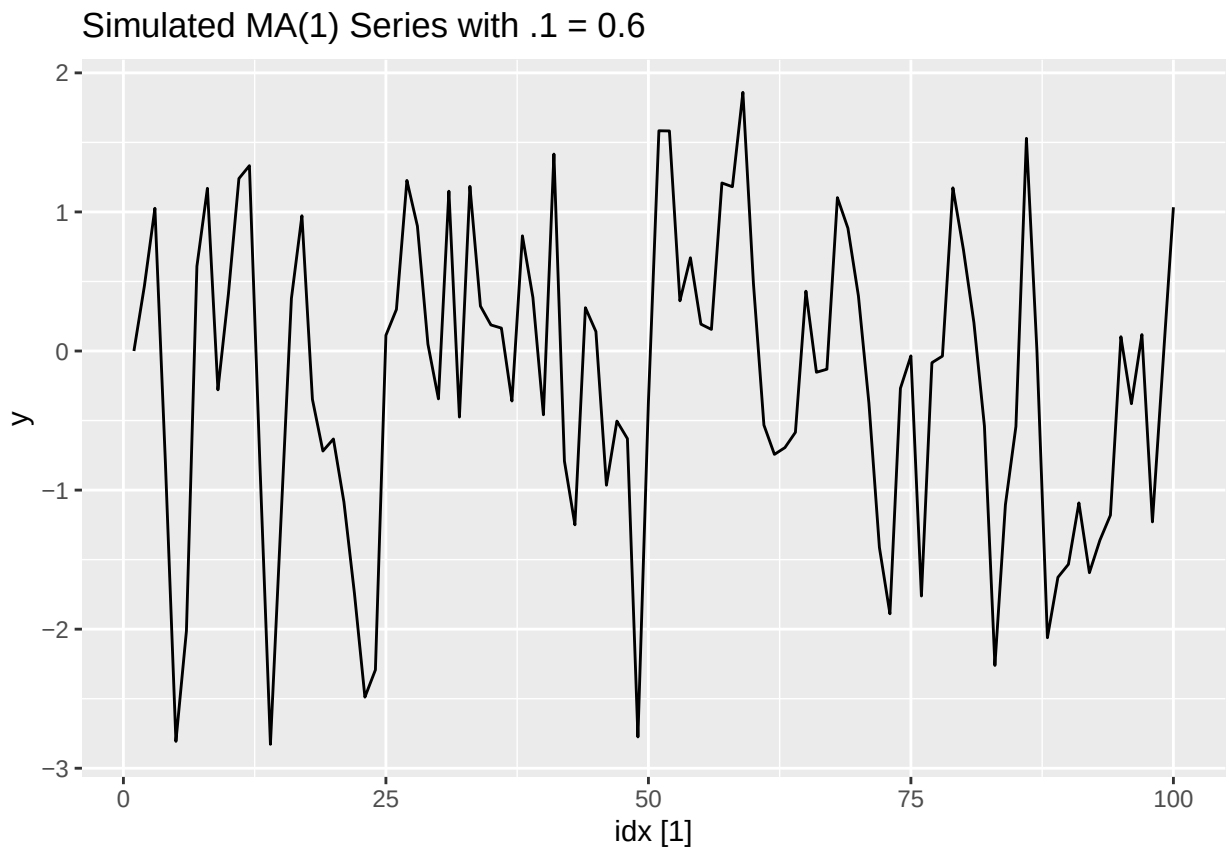


0.5.3 c. Write code to simulate data from an MA(1) model:

The following code simulates data from an MA(1) process with $\theta_1 = 0.6$ and $\sigma^2 = 1$:

The simulated MA(1) series with $\theta_1 = 0.6$ shows random fluctuations around a mean of zero, with short-term dependencies between adjacent values, as expected in a moving average process.

```
# Set a seed for reproducibility
set.seed(678)

y <- numeric(100)
e <- rnorm(100)
for(i in 2:100) {
  y[i] <- e[i] + 0.6 * e[i-1]
}
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

# Plot the series
autoplot(sim, y) + labs(title = "Simulated MA(1) Series with  1 = 0.6")
```



0.5.4 d. Produce a time plot for the MA(1) model. How does the plot change as you change θ_1

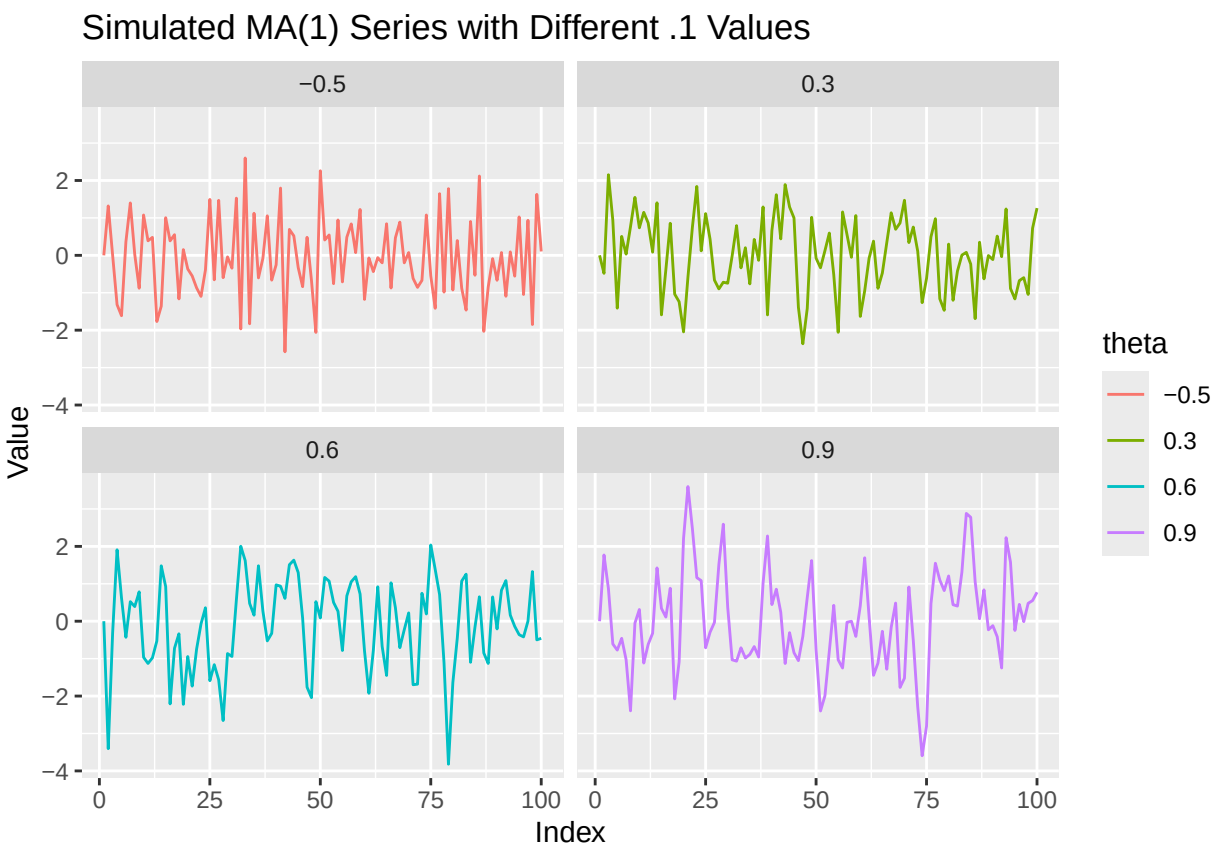
As θ_1 changes in the MA(1) model:

- **For $\theta_1 = -0.5$:** The series appears more erratic and volatile, with rapid changes between values. There is less smoothing and short-term correlations are weaker.
- **For $\theta_1 = 0.3$:** The series shows more smoothness and less volatility. There are moderate correlations between adjacent observations, and the noise level is reduced slightly.
- **For $\theta_1 = 0.6$:** The smoothing effect becomes stronger, making the series appear more persistent. Observations are more dependent on previous shocks, reducing volatility.
- **For $\theta_1 = 0.9$:** The series is significantly smoothed, and fluctuations are more gradual. Shocks have a longer-lasting effect, causing strong short-term correlations between adjacent values. The series appears less noisy with more pronounced trends.

This illustrates how increasing θ_1 in an MA(1) process leads to greater smoothing and a stronger impact of past shocks on the current observation.

```
# Set a seed for reproducibility
set.seed(678)

# Define different theta values
```

```r
theta_vals <- c(-0.5, 0.3, 0.6, 0.9)

# Create an empty list to store simulations
simulations <- list()

# Loop through different theta values and generate MA(1) process for each
for (theta in theta_vals) {
  y <- numeric(100)
  e <- rnorm(100)
  for (i in 2:100) {
    y[i] <- e[i] + theta * e[i - 1]
  }
  sim <- tibble(Index = seq_len(100), Value = y, theta = as.factor(theta))
  simulations[[as.character(theta)]] <- sim
}

# Combine all simulations into one data frame
simulations_df <- bind_rows(simulations)

# Plot the series for different theta values
ggplot(simulations_df, aes(x = Index, y = Value, color = theta)) +
  geom_line() +
  facet_wrap(~theta, nrow = 2) +
  labs(title = "Simulated MA(1) Series with Different  1 Values", y = "Value", x = "Index")
```

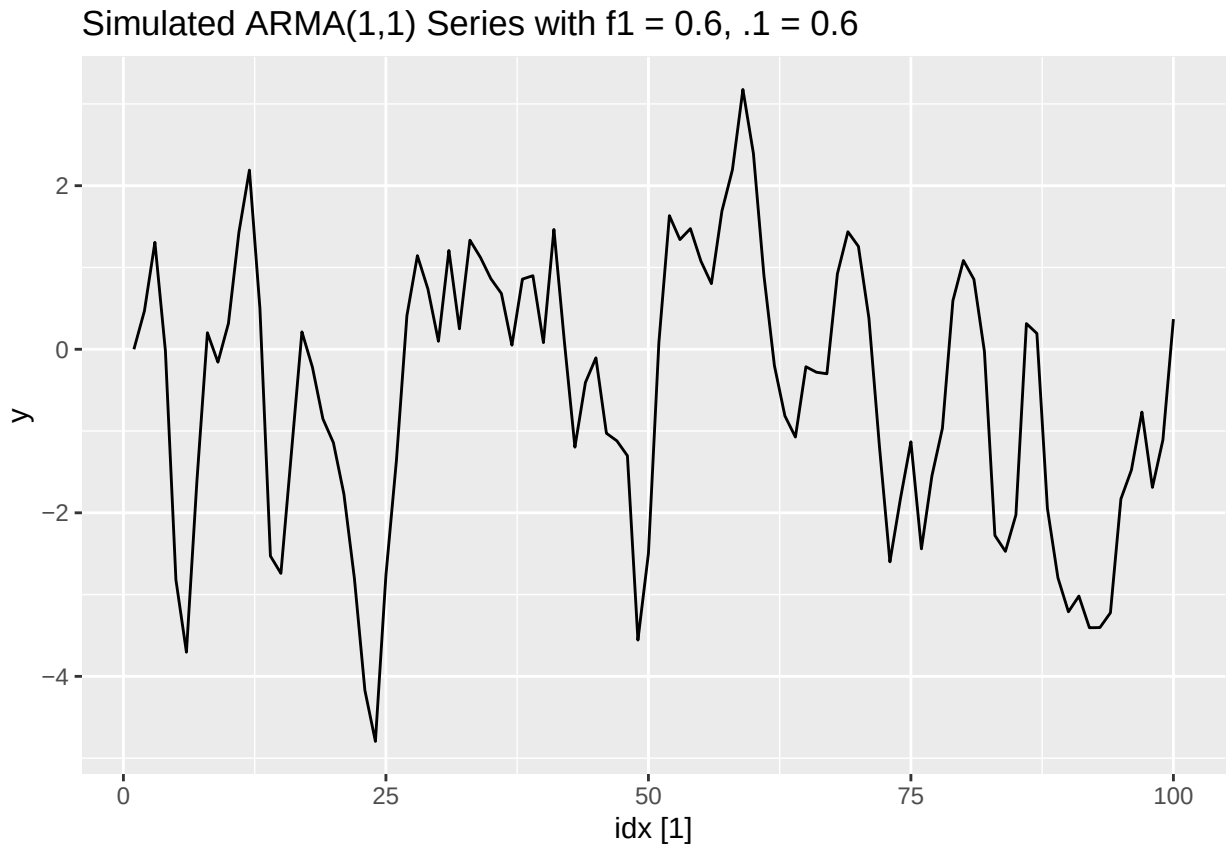


0.5.5 e. Generate data from an ARMA(1,1) model:

The following code simulates an ARMA(1,1) process with $\phi_1 = 0.6$ and $\theta_1 = 0.6$:

The ARMA(1,1) series with $\phi_1 = 0.6$ and $\theta_1 = 0.6$ shows fluctuating behavior, with no apparent trend, indicating stationarity. The series combines characteristics of both autoregressive and moving average processes.

```
# Set a seed for reproducibility
set.seed(678)

y <- numeric(100)
e <- rnorm(100)
for(i in 2:100) {
  y[i] <- 0.6 * y[i-1] + e[i] + 0.6 * e[i-1]
}
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

# Plot the series
autoplot(sim, y) + labs(title = "Simulated ARMA(1,1) Series with  1 = 0.6,  1 = 0.6")
```



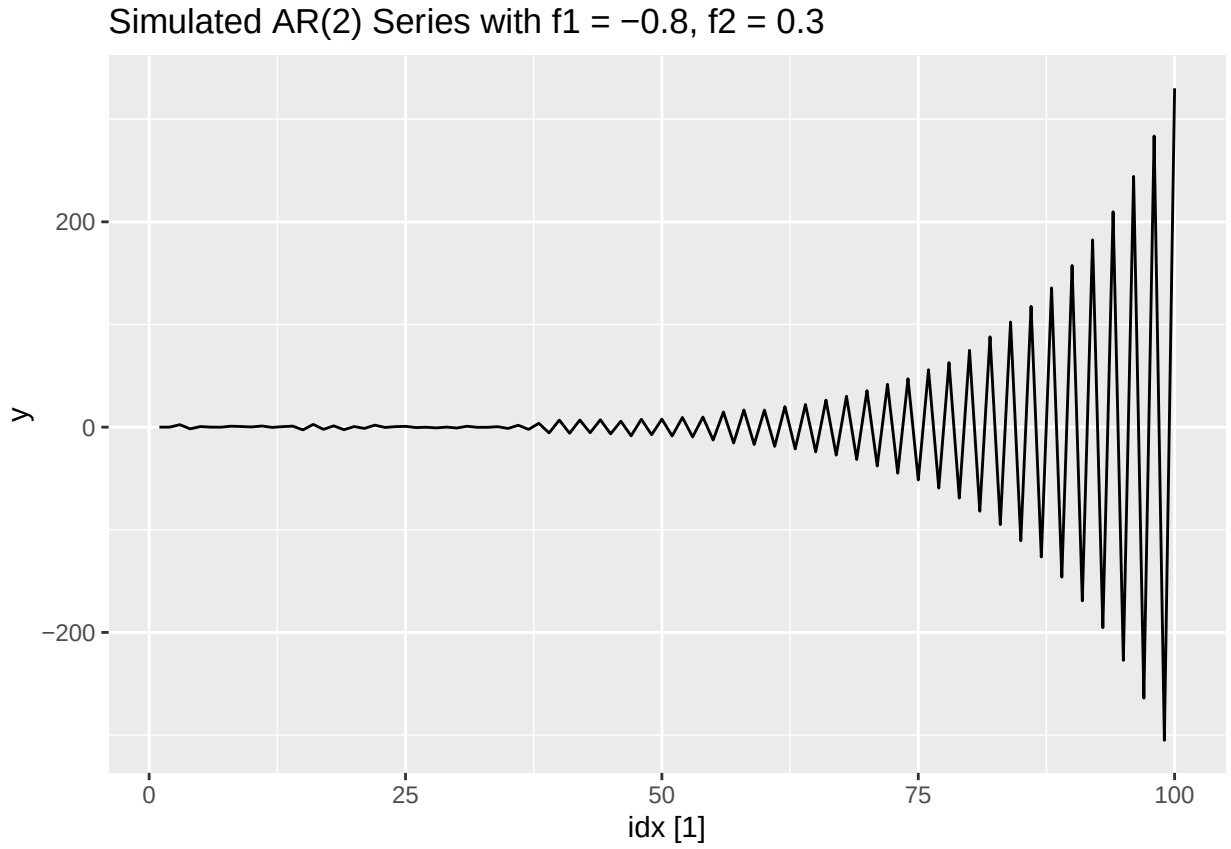
0.5.6 f. Generate data from an AR(2) model:

The following code simulates an AR(2) process with $\phi_1 = -0.8$, $\phi_2 = 0.3$:

The simulated AR(2) series with $\phi_1 = -0.8$ and $\phi_2 = 0.3$ exhibits an explosive behavior, where the magnitude of the fluctuations grows rapidly over time. This indicates that the series is non-stationary, as the values do

not stabilize around a constant mean and instead continue to increase in amplitude, showing the need for adjustments to the model parameters to ensure stationarity.

```
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]
}
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

# Plot the series
autoplot(sim, y) + labs(title = "Simulated AR(2) Series with  1 = -0.8,  2 = 0.3")
```



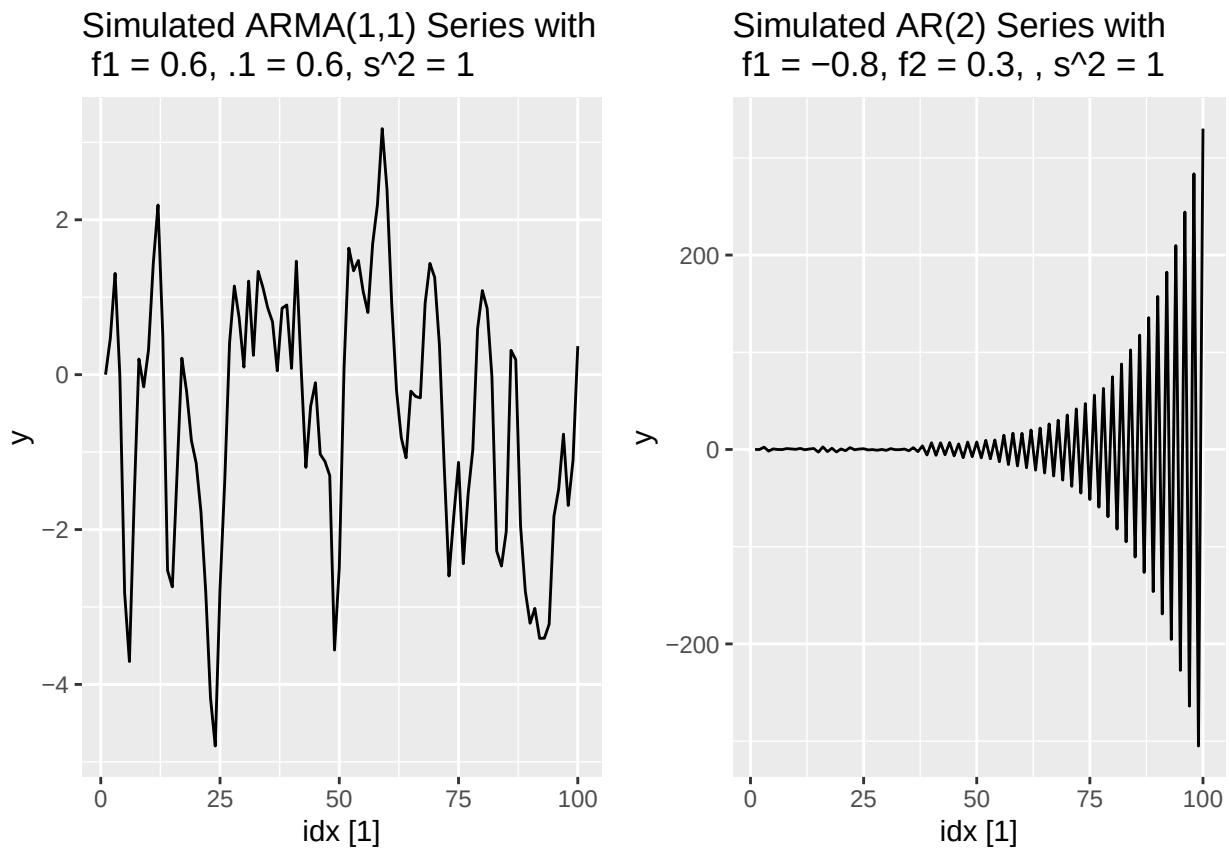
0.5.7 g. Graph and compare the last two series

The two series exhibit distinct behavior:

- The **ARMA(1,1)** series with $\phi_1 = 0.6$ and $\theta_1 = 0.6$ appears more stable, fluctuating around the mean with some variability, showing moderate oscillations.
- The **AR(2)** series with $\phi_1 = -0.8$ and $\phi_2 = 0.3$ displays a growing oscillation pattern, indicating non-stationary explosive behavior as the series diverges over time.

These differences highlight how the combination of AR and MA terms can stabilize a time series compared to a pure AR(2) model with unstable parameter values.

```r
# Load necessary libraries
library(ggplot2)
library(tsibble)

# Simulating ARMA(1,1) series with  1 = 0.6 and  1 = 0.6
set.seed(678)
y_arma <- numeric(100)
e_arma <- rnorm(100)
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)

# Plot ARMA(1,1) series
p1 <- autoplot(sim_arma, y) + labs(title = "Simulated ARMA(1,1) Series with\n  1 = 0.6,  1 = 0.6,  ^2 = 1

# Simulating AR(2) series with  1 = -0.8 and  2 = 0.3
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)

# Plot AR(2) series
p2 <- autoplot(sim_ar2, y) + labs(title = "Simulated AR(2) Series with\n  1 = -0.8,  2 = 0.3, ,  ^2 = 1")

# Display the two plots
library(gridExtra)
grid.arrange(p1, p2, ncol = 2)
```



0.6 Exercise 9.7: Consider aus_airpassengers, the total number of passengers (in millions) from Australian air carriers for the period 1970-2011.

0.6.1 a. Use ARIMA() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

To analyze the `aus_airpassengers` dataset using the ARIMA modeling procedure, I will follow the general approach:

0.6.1.1 Step 1: Plot the original, Box-Cox Transformed, and Differenced series.

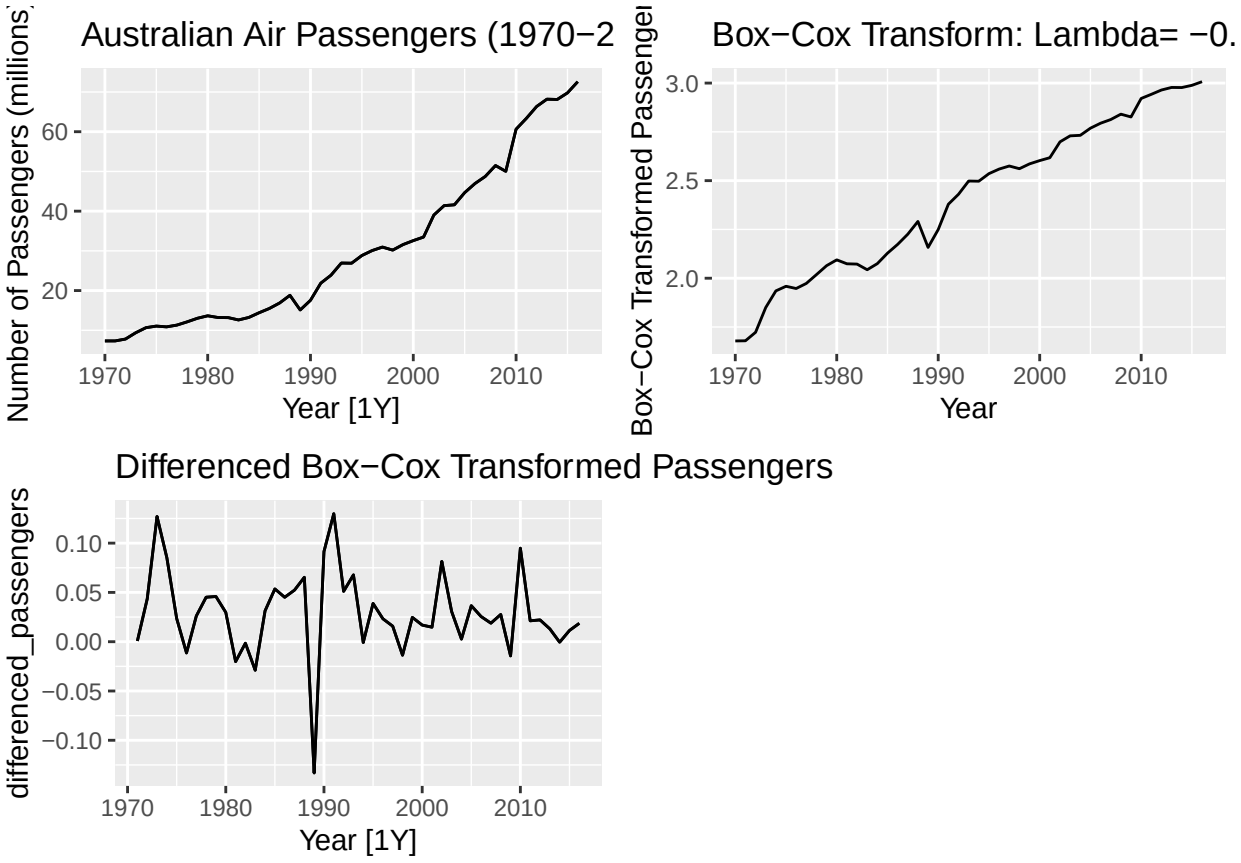
1. **Original Series (Top Left)**: The original Australian Air Passengers series from 1970 to 2011 shows an upward trend, indicating a consistent increase in the number of passengers over time.
2. **Box-Cox Transformed Series (Top Right)**: The Box-Cox transformed series smooths out the variance, making the series more stable and linear, which helps in stabilizing the variance and making it easier to model.
3. **Differenced Box-Cox Transformed Series (Bottom)**: The differenced Box-Cox series appears to fluctuate around zero, indicating that the differencing has removed most of the trend, making the series closer to being stationary and ready for time series modeling.

```
library(fpp3)
library(ggplot2)
```

```r
library(dplyr)
library(tsibble)
library(fabletools)

# Step 1: Plot the data and identify unusual observations.
p1 <- aus_airpassengers |>
  autoplot(Passengers) +
    geom_line(color = "black") +
  labs(title="Australian Air Passengers (1970-2011)", y="Number of Passengers (millions)")

# Step 2: Apply Box-Cox transformation to stabilize variance
lambda_passengers <- aus_airpassengers |>
  features(Passengers, features = guerrero) |>
  pull(lambda_guerrero)

# Apply the Box-Cox transformation
aus_airpassengers <- aus_airpassengers |>
  mutate(box_cox_passengers = box_cox(Passengers, lambda_passengers))

# Plot the Box-Cox transformed series
p2 <- ggplot(aus_airpassengers, aes(x = Year, y = box_cox_passengers)) +
  geom_line(color = "black") +
  labs(title = paste("Box-Cox Transform: Lambda=", round(lambda_passengers, 4)),
       y = "Box-Cox Transformed Passengers")

# Step 3: Take first differences if the data is non-stationary.
# First, check if there are missing values in the transformed data
aus_airpassengers |> filter(is.na(box_cox_passengers))
```

```
## # A tsibble: 0 x 3 [?]
## # i 3 variables: Year <dbl>, Passengers <dbl>, box_cox_passengers <dbl>
```

```r
# Now, proceed with differencing and remove any NA values
aus_airpassengers <- aus_airpassengers |>
  mutate(differenced_passengers = difference(box_cox_passengers)) |>
  drop_na(differenced_passengers)

# Plot the differenced series
p3 <- aus_airpassengers |>
  autoplot(differenced_passengers) +
    geom_line(color = "black") +
  labs(title = "Differenced Box-Cox Transformed Passengers")

# Display the two plots
library(gridExtra)
grid.arrange(p1, p2, p3, ncol = 2)
```



0.6.1.2 Step 2: Examine the ACF/PACF plots.

Based on the ACF and PACF plots:

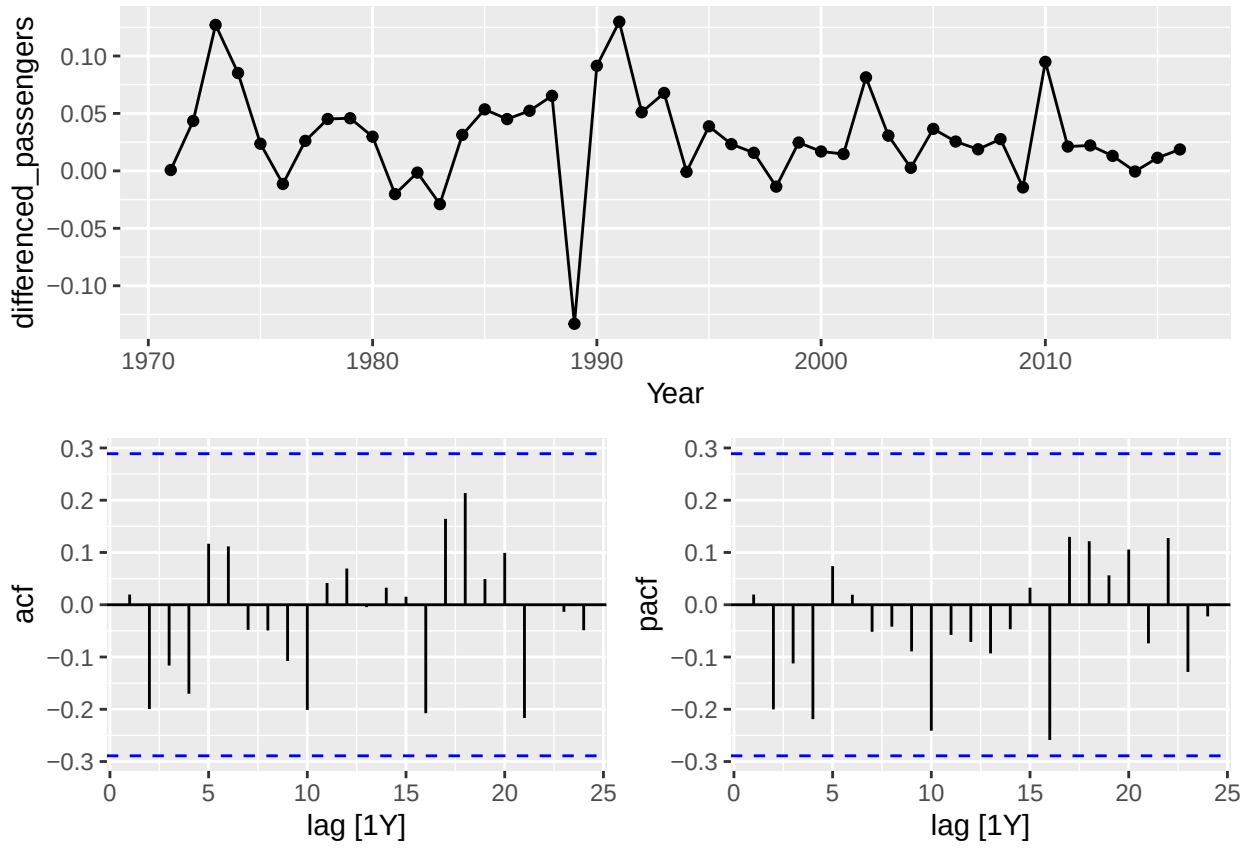
1. **ARIMA (p, d, 0)** model:
   - **PACF** plot shows significant spikes at lag 1 and possibly lag 2, which taper off after these lags. This suggests the presence of **autoregressive (AR) components** in the model.
   - Hence, an AR model with $p = 2$ can be a good starting point, where the autoregressive order accounts for the significant spikes in the PACF.
   - As we have already differenced the series, $d = 1$ (to account for stationarity).

   Recommended ARIMA model: ARIMA$(2, 1, 0)$

2. **ARIMA (0, d, q)** model:
   - **ACF** plot shows significant spikes at lags 1 and 3, indicating that there may be a **moving average (MA)** component in the model.
   - Thus, an MA model with $q = 2$ seems appropriate since there are early significant spikes in the ACF that indicate the presence of short-term dependencies.
   - Again, since the data was already differenced, $d = 1$ is used.

   Recommended ARIMA model: ARIMA$(0, 1, 2)$

```
# Plot ACF/PACF
aus_airpassengers |>
  gg_tsdisplay(differenced_passengers, plot_type = 'partial', lag_max = 24)
```



0.6.1.3 Step 3: Fit ARIMA models based on AICc. The **stepwise model** and **search model** both have the lowest AICc values, which means they are likely the best models to use. Their BIC values are also the lowest compared to the others. I will select the **Stepwise model** as the best model since it came on top when sorted by AICc

```
library(knitr)
library(kableExtra)
```

```
##
## Attaching package: 'kableExtra'
```

```
## The following object is masked from 'package:dplyr':
##
##     group_rows
```

```
# Fit ARIMA models
passengers_fit <- aus_airpassengers |>
  model(
    arima210 = ARIMA(Passengers ~ pdq(2,1,0)),
    arima012 = ARIMA(Passengers ~ pdq(0,1,2)),
    stepwise = ARIMA(Passengers),
    search = ARIMA(Passengers, stepwise = FALSE)
  )
```

```r
# Extract the model summary with AICc and BIC values
model_summary <- glance(passengers_fit) |>
  arrange(AICc) |>
  select(.model, AICc, BIC)

# Extract the ARIMA orders (p,d,q) for each model
model_orders <- passengers_fit |>
  pivot_longer(everything(), names_to = "Model name", values_to = "Orders") |>
  select(`Model name`, Orders)

# Combine the AICc, BIC and ARIMA orders into a single data frame
combined_results <- model_summary |>
  left_join(model_orders, by = c(".model" = "Model name"))

# Display the combined data frame using kable
combined_results %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

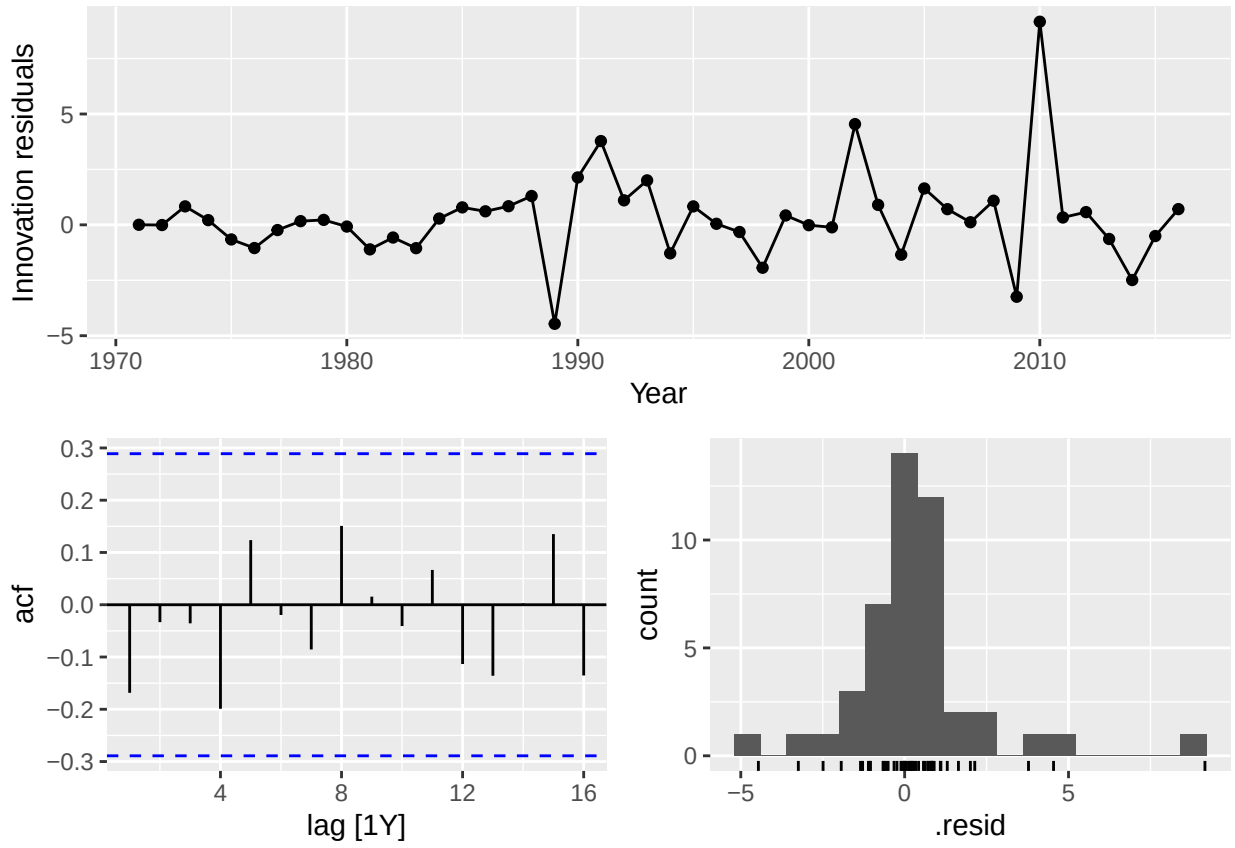
.model	AICc	BIC	Orders
stepwise	194.9021	198.1778	<ARIMA(0,2,1)>
search	194.9021	198.1778	<ARIMA(0,2,1)>
arima012	201.1053	207.3319	<ARIMA(0,1,2) w/ drift>
arima210	201.1883	207.4149	<ARIMA(2,1,0) w/ drift>

0.6.1.4 Step 4: Check the residuals for white noise. Here are three statements regarding the residuals:

1. **Residuals over time**: The residuals appear to fluctuate around zero with no obvious patterns, indicating that the model has captured most of the trend and seasonality in the data.
2. **ACF of residuals**: The autocorrelation function (ACF) of the residuals shows that most of the autocorrelations are within the 95% confidence interval, which suggests that there is no significant autocorrelation left in the residuals. This is a good sign that the residuals resemble white noise.
3. **Histogram of residuals**: The residuals are approximately normally distributed with a mean around zero, although there may be a few outliers. This suggests that the model residuals are fairly well-behaved.

These results indicate that the model is likely appropriate and the residuals resemble white noise.

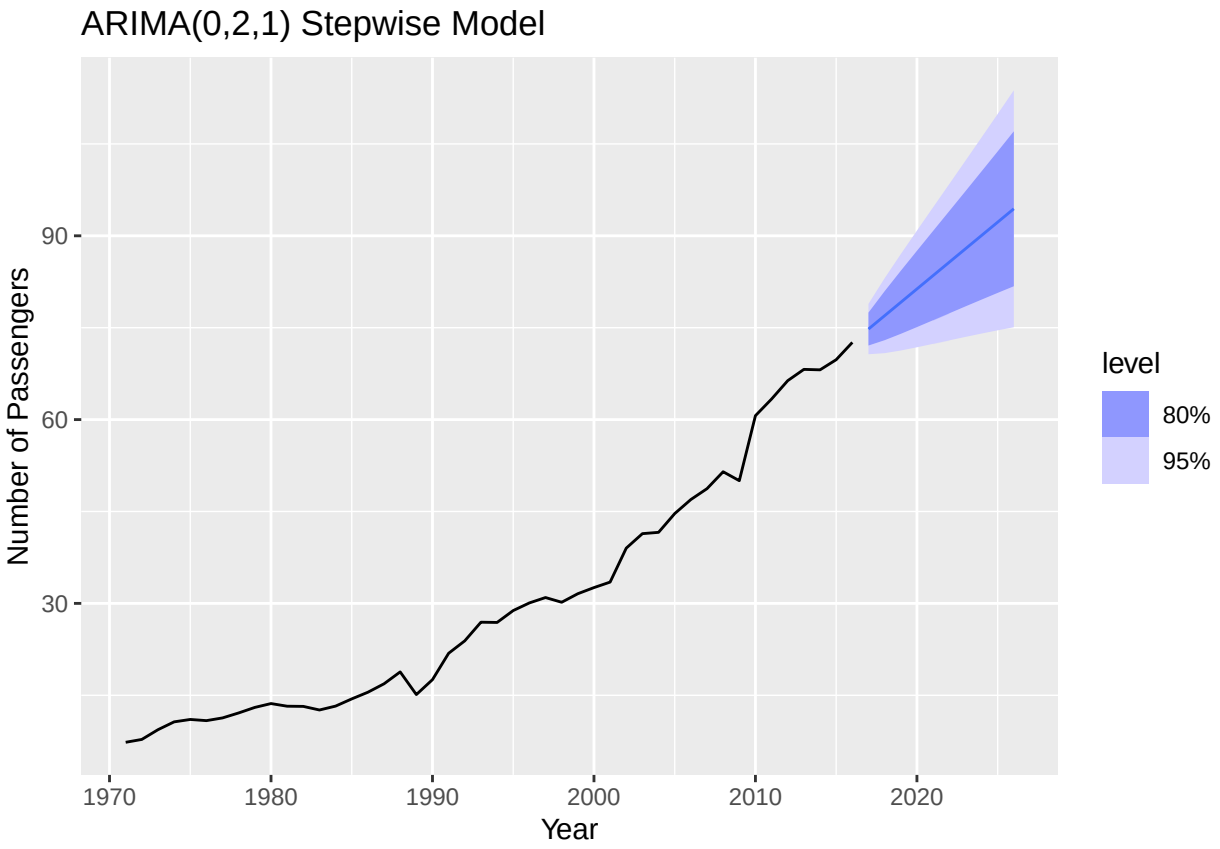
```r
# Check residuals for white noise
passengers_fit |> select(stepwise) |>
  gg_tsresiduals()
```



0.6.1.5 Step 5: Forecast once residuals look like white noise. The forecast plot below shows the following:

1. **Steady Growth**: The forecast for the next 10 periods predicts a steady increase in the number of passengers, continuing the long-term upward trend observed in the historical data.
2. **Prediction Interval**: The forecast includes an 80% and 95% confidence interval, represented by the shaded regions. As expected, the further into the future, the wider the intervals, indicating increased uncertainty in the predictions over time.
3. **Model Fit**: The model appears to capture the general trend well, and the residuals resembled white noise, indicating that the model is likely appropriate for forecasting future values of passenger numbers.

```
# Forecast for the next 10 periods
passengers_forecast <- passengers_fit |>
  forecast(h=10) |>
  filter(.model == 'stepwise')

# Plot the forecast
passengers_forecast |>
  autoplot(aus_airpassengers)  +
  labs(title = "ARIMA(0,2,1) Stepwise Model", y = "Number of Passengers")
```



0.6.2 b. Write the stepwise model in terms of the backshift operator.

The ARIMA(0,2,1) model in terms of the backshift operator is:

$$(1 - B)^2 y_t = (1 - \theta_1 B)\epsilon_t$$

This expands to:

$$y_t - 2y_{t-1} + y_{t-2} = \epsilon_t - \theta_1 \epsilon_{t-1}$$

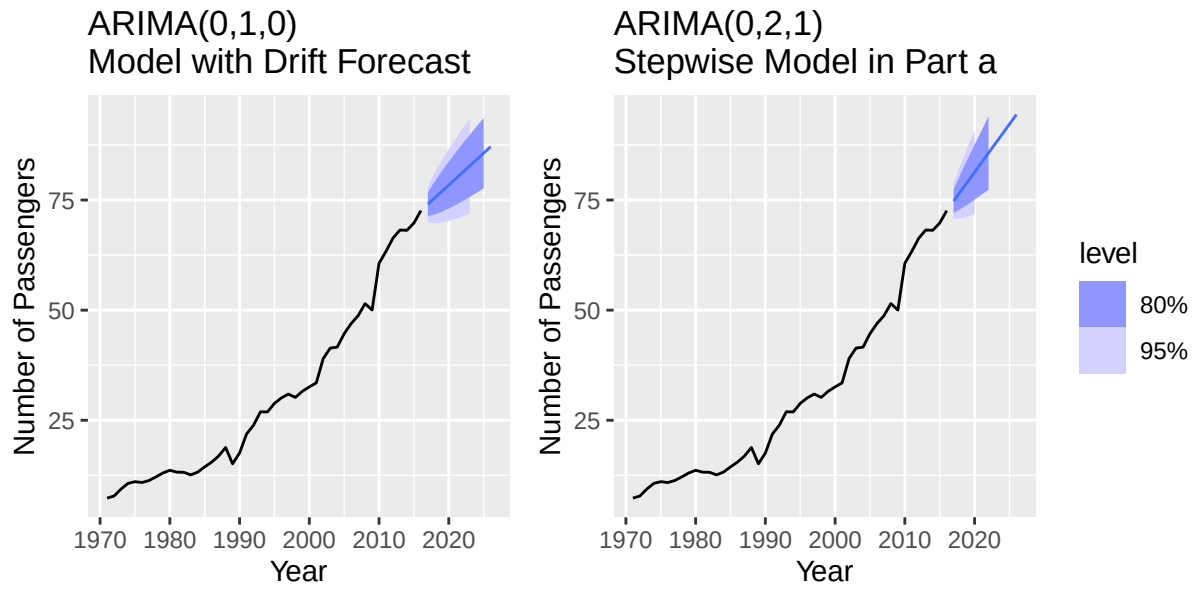
Where: - B is the backshift operator. - θ_1 is the MA(1) coefficient. - ϵ_t is the white noise error term.

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

The **ARIMA(0,1,0) with drift** model has greater uncertainty because it relies primarily on the drift term to model the trend without capturing short-term fluctuations as effectively. This leads to wider confidence intervals as the model does not incorporate autoregressive (AR) or moving average (MA) components, resulting in more uncertainty about future values compared to the **ARIMA(0,2,1)** model, which accounts for short-term dependencies with its MA term. The stepwise model better captures the underlying trend.

```
# Install required packages if not already installed
if (!require("patchwork")) install.packages("patchwork")
```

```r
# Load necessary libraries
library(fable)
library(dplyr)
library(ggplot2)
library(tsibble)
library(patchwork)

# Forecast for the next 10 periods for the stepwise model
passengers_forecast <- passengers_fit |>
  forecast(h=10) |>
  filter(.model == 'stepwise')

# Fit the ARIMA(0,1,0) model with drift
arima010_drift_fit <- aus_airpassengers |>
  model(ARIMA(Passengers ~ pdq(0,1,0) + 1))

# Forecast for the next 10 periods using ARIMA(0,1,0) with drift
arima010_drift_forecast <- arima010_drift_fit |>
  forecast(h=10)

# Find the range of y-axis limits (common for both plots)
y_limits <- range(aus_airpassengers$Passengers,
                  arima010_drift_forecast$.mean,
                  passengers_forecast$.mean)

# Plot 1: ARIMA(0,1,0) with drift forecast
plot1 <- autoplot(aus_airpassengers, Passengers) +
  autolayer(arima010_drift_forecast) +
  labs(title = "ARIMA(0,1,0) \nModel with Drift Forecast",
       y = "Number of Passengers",
       x = "Year") +
  guides(colour = guide_legend(title = "Forecast")) +
  ylim(y_limits)

# Plot 2: ARIMA(0,2,1) stepwise forecast
plot2 <- passengers_forecast |>
  autoplot(aus_airpassengers)  +
  labs(title = "ARIMA(0,2,1) \nStepwise Model in Part a", y = "Number of Passengers") +
  ylim(y_limits)


# Combine the two plots side by side using patchwork
plot1 + plot2 + plot_layout(guides = 'collect') +
 theme(aspect.ratio = 1/1) # Set aspect ratio: width twice the height (2:1)
```



0.6.4 d. Plot forecasts from an ARIMA(2,1,2) model with drift and compare these to parts a and c. Remove the constant and see what happens.

1. The **ARIMA(0,1,0) with Drift** model shows a steady upward trend in the forecast, and the prediction intervals are narrow, indicating less uncertainty in short-term predictions.
2. The **ARIMA(0,2,1) Stepwise** model also shows a consistent trend, with broader prediction intervals compared to the ARIMA(0,1,0), suggesting higher uncertainty in long-term predictions.
3. The **ARIMA(2,1,2) with Drift** model forecasts similar growth but with the widest prediction intervals, indicating substantial uncertainty in the long-term forecast.
4. The **ARIMA(2,1,2) without Drift** model exhibits more conservative forecasts, and although the growth is still evident, the removal of drift leads to a flatter forecast compared to the drift-included model.
5. A warning was triggered by the **ARIMA(2,1,2) model without drift** due to a non-stationary AR part, which implies that the autoregressive components of the model might not be suitable for this data without additional differencing or adjustments.

```
# Set a higher recursion limit
options(expressions = 5000)

# Install required packages if not already installed
if (!require("patchwork")) install.packages("patchwork")
```

```r
# Load necessary libraries
library(fable)
library(dplyr)
library(ggplot2)
library(tsibble)
library(patchwork)

# Function to fit models with error handling
fit_arima_model <- function(data, formula) {
  tryCatch({
    model_fit <- data |> model(ARIMA(formula))
    return(model_fit)
  }, error = function(e) {
    message("Error: ", e)
    return(NULL)
  })
}

# Fit ARIMA(0,1,0) with drift
arima010_drift_fit <- fit_arima_model(aus_airpassengers, Passengers ~ pdq(0,1,0) + 1)

# Fit ARIMA(2,1,2) with drift
arima212_drift_fit <- fit_arima_model(aus_airpassengers, Passengers ~ pdq(2,1,2) + 1)
```

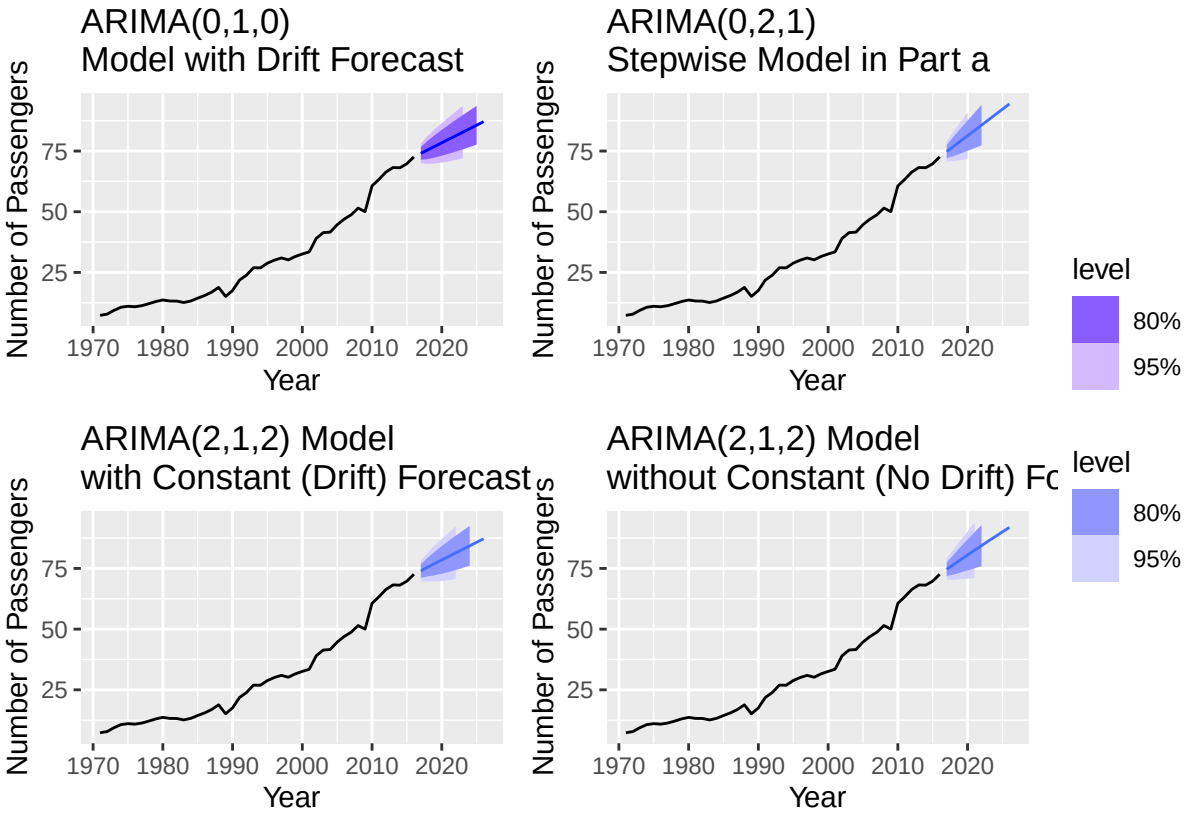
```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

```r
# Fit ARIMA(2,1,2) without drift
arima212_no_drift_fit <- fit_arima_model(aus_airpassengers, Passengers ~ pdq(2,1,2) + 0)
```

```
## Warning in wrap_arima(y, order = c(p, d, q), seasonal = list(order = c(P, :
## possible convergence problem: optim gave code = 1
## Warning in wrap_arima(y, order = c(p, d, q), seasonal = list(order = c(P, :
## NaNs produced
```

```r
# Check if models were fitted successfully
if (is.null(arima010_drift_fit) || is.null(arima212_drift_fit) || is.null(arima212_no_drift_fit)) {
  stop("One or more models could not be fitted. Check model parameters.")
}

# Forecast for the next 10 periods using ARIMA(0,1,0) with drift
arima010_drift_forecast <- arima010_drift_fit |> forecast(h = 10)

# Forecast for the next 10 periods using ARIMA(2,1,2) with and without drift
arima212_drift_forecast <- arima212_drift_fit |> forecast(h = 10)
arima212_no_drift_forecast <- arima212_no_drift_fit |> forecast(h = 10)

# Forecast for the stepwise model
passengers_forecast <- passengers_fit |>
  forecast(h=10) |>
  filter(.model == 'stepwise')

# Handle missing values in the forecast data (if any)
```

```r
arima010_drift_forecast <- arima010_drift_forecast |> na.omit()
arima212_drift_forecast <- arima212_drift_forecast |> na.omit()
arima212_no_drift_forecast <- arima212_no_drift_forecast |> na.omit()
passengers_forecast <- passengers_forecast |> na.omit()

# Find the range of y-axis limits (common for both plots)
y_limits <- range(c(aus_airpassengers$Passengers,
                    arima010_drift_forecast$.mean,
                    passengers_forecast$.mean), na.rm = TRUE)

# Plot 1: ARIMA(0,1,0) with drift forecast
plot1 <- autoplot(aus_airpassengers, Passengers) +
  autolayer(arima010_drift_forecast, .mean, colour = "blue") +
  labs(title = "ARIMA(0,1,0) \nModel with Drift Forecast",
       y = "Number of Passengers",
       x = "Year") +
  guides(colour = guide_legend(title = "Forecast")) +
  ylim(y_limits)

# Plot 2: ARIMA(0,2,1) stepwise forecast
plot2 <- passengers_forecast |>
  autoplot(aus_airpassengers)  +
  labs(title = "ARIMA(0,2,1) \nStepwise Model in Part a", y = "Number of Passengers") +
  ylim(y_limits)

# Plot 3: ARIMA(2,1,2) with drift forecast
plot3 <- arima212_drift_forecast |>
  autoplot(aus_airpassengers) +
  labs(title = "ARIMA(2,1,2) Model \nwith Constant (Drift) Forecast",
       y = "Number of Passengers") +
  ylim(y_limits)

# Plot 4: ARIMA(2,1,2) without drift forecast
plot4 <- arima212_no_drift_forecast |>
  autoplot(aus_airpassengers) +
  labs(title = "ARIMA(2,1,2) Model \nwithout Constant (No Drift) Forecast",
       y = "Number of Passengers") +
  ylim(y_limits)

# Combine the four plots into a 2x2 layout using patchwork
(plot1 + plot2) / (plot3 + plot4) +
  plot_layout(guides = 'collect') +  # Arrange plots in 2x2 grid
  theme(aspect.ratio = 1/1) # Set aspect ratio: width twice the height (2:1)
```




```
# Model diagnostics for ARIMA(2,1,2) without drift
report(arima212_no_drift_fit)
```

```
## Series: Passengers
## Model: ARIMA(2,1,2)
##
## Coefficients:
##          ar1     ar2     ma1      ma2
##       0.019  0.9638  0.0817  -0.8762
## s.e.    NaN     NaN     NaN      NaN
##
## sigma^2 estimated as 4.683:  log likelihood=-97.15
## AIC=204.3   AICc=205.84   BIC=213.33
```

0.6.5 e. Plot forecasts from an ARIMA(0,2,1) model with a constant. What happens?

1. The stepwise ARIMA(0,2,1) model forecast (left) follows a linear trend based on historical data, with a smooth prediction that aligns with the past growth pattern.

2. The ARIMA(0,2,1) model with a constant (right) introduces a quadratic or higher-order trend, as indicated by the warning, causing an exaggerated increase in the forecast compared to the stepwise model.

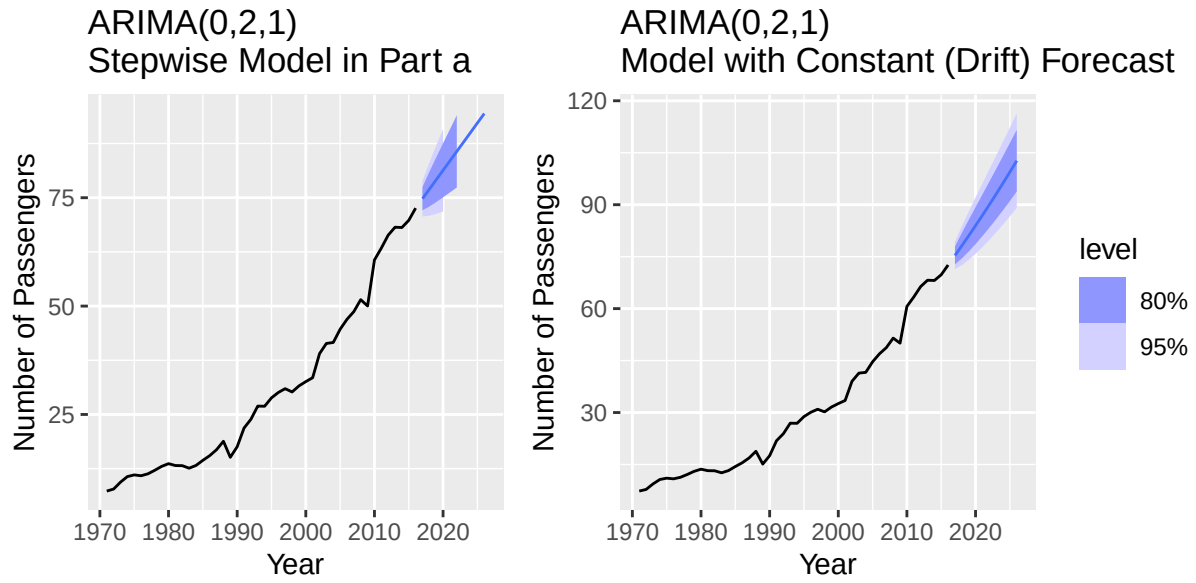
3. The inclusion of a constant (drift) in a model with second differencing leads to a more aggressive growth forecast, as the drift accumulates over time.

4. The warning suggests that the model with a constant should be reconsidered, as including both a constant and a second difference induces a polynomial trend, which is not recommended for time series that exhibit linear growth.

```
# Fit the ARIMA(0,2,1) model with a constant (drift)
arima021_with_constant <- aus_airpassengers |>
  model(ARIMA(Passengers ~ pdq(0,2,1) + 1))  # +1 adds the constant (drift)
```

```
## Warning: Model specification induces a quadratic or higher order polynomial trend.
## This is generally discouraged, consider removing the constant or reducing the number of differences.
```

```
# Forecast from the ARIMA(0,2,1) model with constant
forecast_arima021_with_constant <- arima021_with_constant |>
  forecast(h = 10)

# Plot 2: ARIMA(0,2,1) stepwise forecast
plot1 <- passengers_forecast |>
  autoplot(aus_airpassengers)  +
  labs(title = "ARIMA(0,2,1) \nStepwise Model in Part a", y = "Number of Passengers") +
  ylim(y_limits)

# Plot the forecast with constant
plot2 <- forecast_arima021_with_constant |>
  autoplot(aus_airpassengers) +
  labs(title = "ARIMA(0,2,1) \nModel with Constant (Drift) Forecast",
       y = "Number of Passengers")

# Combine the two plots side by side using patchwork
plot1 + plot2 + plot_layout(guides = 'collect') +
 theme(aspect.ratio = 1/1) # Set aspect ratio: width twice the height (2:1)
```



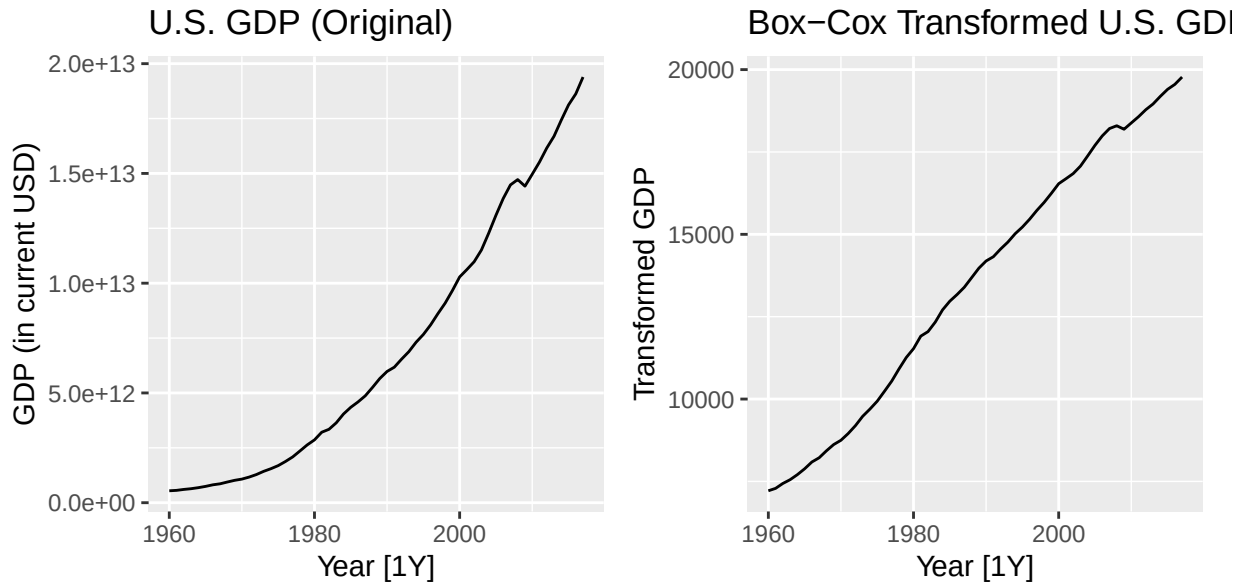
0.7 Exercise 9.8: For the United States GDP series (from global_economy):

0.7.1 a. if necessary, find a suitable Box-Cox transformation for the data;

The original U.S. GDP series shows an exponential growth pattern, while the Box-Cox transformation reduces this trend, making the data more linear and stabilizing the variance for easier modeling.

```
library(fpp3)

# Filter for the United States GDP data
us_gdp <- global_economy |>
  filter(Code == "USA") |>
  select(Year, GDP)

# Plot the original U.S. GDP series (plot1)
plot1 <- us_gdp |>
  autoplot(GDP) +
  labs(title = "U.S. GDP (Original)",
       y = "GDP (in current USD)")

# Apply the Guerrero method to find a suitable Box-Cox transformation
lambda_gdp <- us_gdp |>
  features(GDP, features = guerrero) |>
  pull(lambda_guerrero)
```

```r
# Apply the Box-Cox transformation
us_gdp <- us_gdp |>
  mutate(transformed_gdp = box_cox(GDP, lambda_gdp))

# Plot the transformed U.S. GDP series (plot2)
plot2 <- us_gdp |>
  autoplot(transformed_gdp) +
  labs(title = paste("Box-Cox Transformed U.S. GDP (Lambda =", round(lambda_gdp, 4), ")"),
       y = "Transformed GDP")

# Combine the two plots side by side using patchwork
plot1 + plot2 + plot_layout(guides = 'collect') +
 theme(aspect.ratio = 1/1) # Set aspect ratio: width twice the height (2:1)
```



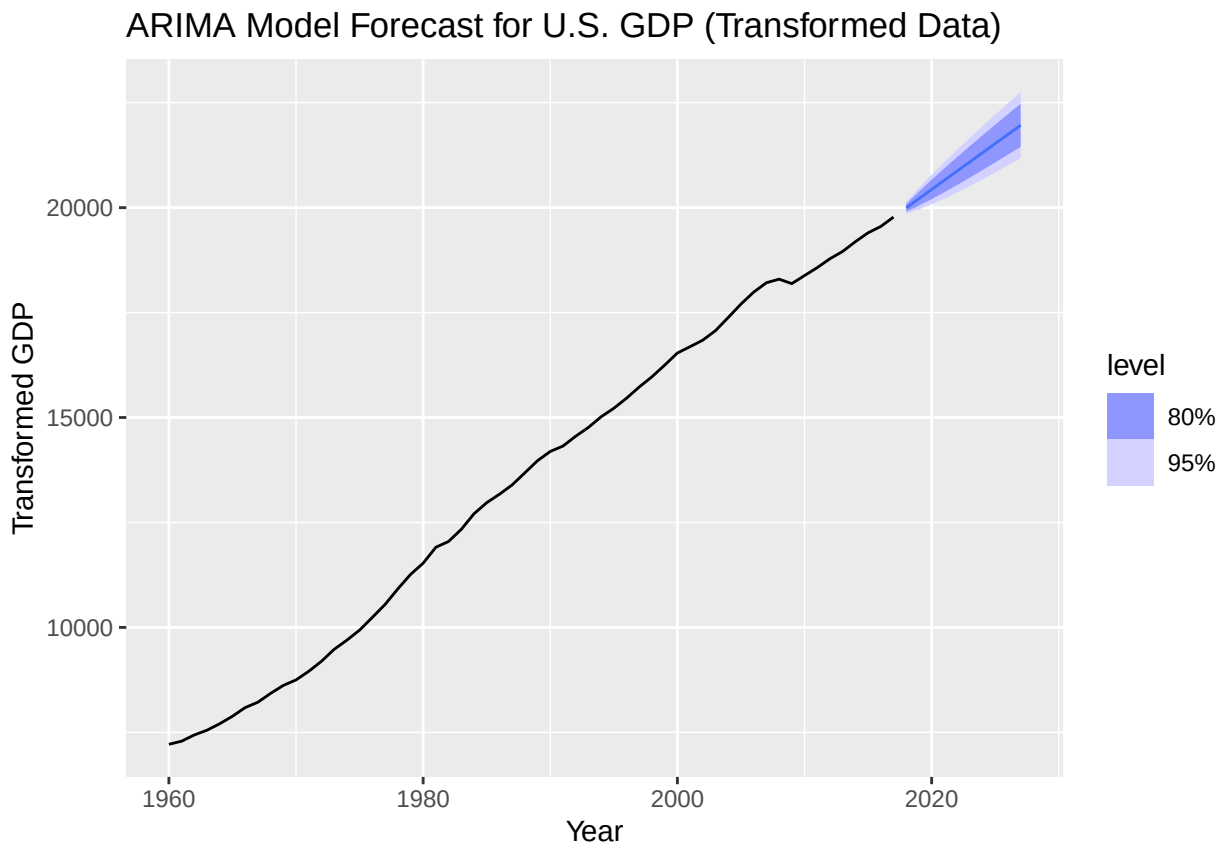
0.7.2 b. Fit a suitable ARIMA model to the transformed data using ARIMA();

1. The ARIMA(1,1,0) model with drift shows an autoregressive coefficient (AR1) of 0.4586, indicating moderate short-term autocorrelation in the transformed GDP series.
2. The forecast for U.S. GDP using the ARIMA(1,1,0) model projects continued growth in GDP, with 80% and 95% confidence intervals becoming wider as the forecast horizon extends, reflecting greater uncertainty.
3. The AIC and BIC values (656.65 and 662.78, respectively) suggest that the model fit is reasonable, but further tuning could potentially improve the model's performance.

```r
# Fit ARIMA model to Box-Cox transformed data
us_gdp_transformed_fit <- us_gdp |>
  model(ARIMA(transformed_gdp))

# Display the ARIMA model summary
report(us_gdp_transformed_fit)
```

```
## Series: transformed_gdp
## Model: ARIMA(1,1,0) w/ drift
##
## Coefficients:
##          ar1  constant
##       0.4586  118.1822
## s.e.  0.1198    9.5047
##
## sigma^2 estimated as 5479:  log likelihood=-325.32
## AIC=656.65   AICc=657.1   BIC=662.78
```

```r
# Forecast for the next 10 periods
us_gdp_forecast <- us_gdp_transformed_fit |>
  forecast(h = 10)

# Plot the forecast
us_gdp_forecast |>
  autoplot(us_gdp) +
  labs(title = "ARIMA Model Forecast for U.S. GDP (Transformed Data)",
       y = "Transformed GDP")
```



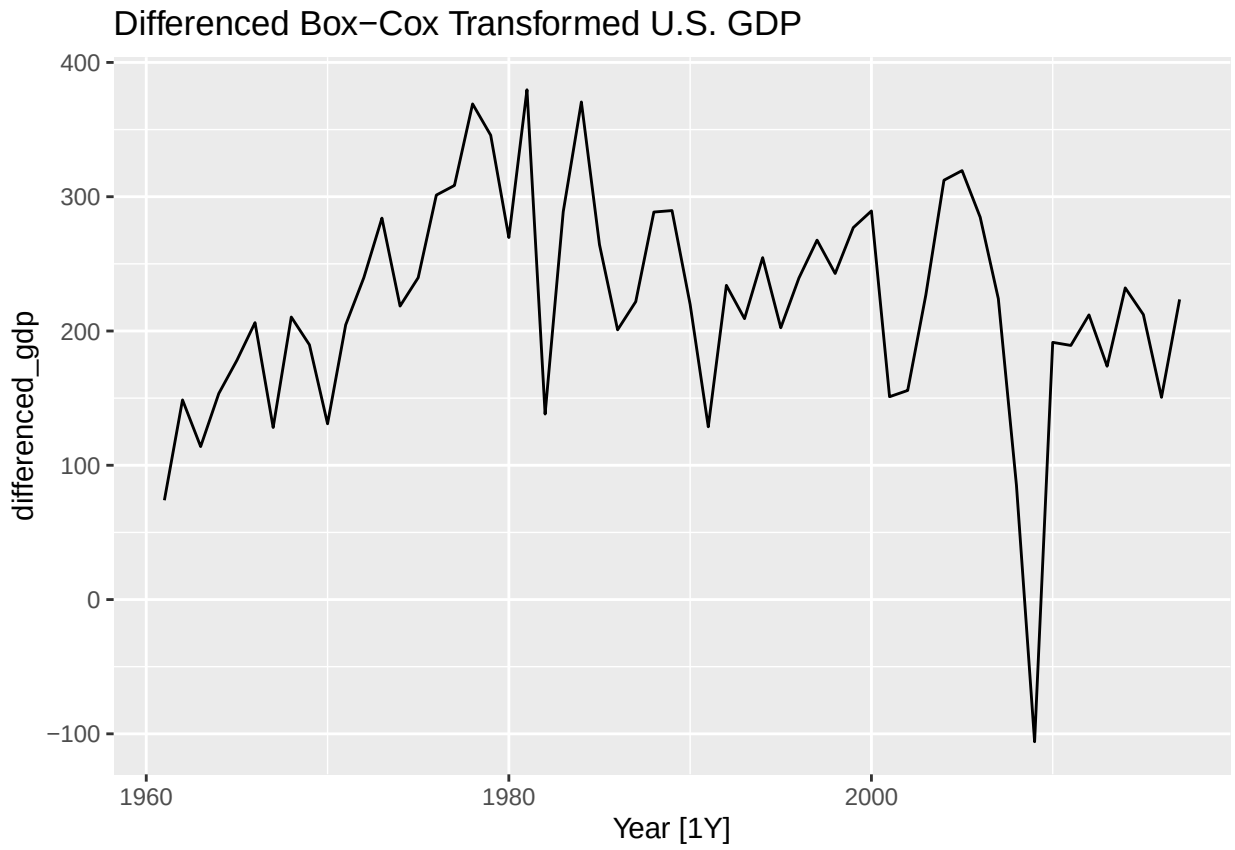
0.7.3 c. Try some other plausible models by experimenting with the orders chosen;

1. **Differenced Series**: The differenced U.S. GDP series appears to fluctuate around a mean close to zero, which indicates that differencing was effective in stabilizing the trend. There is no clear trend left in the differenced series, which is expected after applying first differencing.
2. **Autocorrelation Function (ACF)**: The ACF plot shows significant spikes at lags 1, 3, and possibly lag 4. This suggests the presence of short-term dependencies, indicating that a moving average (MA) component might be necessary in the model. An MA model with $q = 3$ could be appropriate.
3. **Partial Autocorrelation Function (PACF)**: The PACF plot shows a significant spike at lag 1, followed by a tapering off. This suggests the presence of an autoregressive (AR) component. An AR model with $p = 1$ could be suitable, as the PACF cuts off sharply after lag 1.
4. **ARIMA(1,1,0)**: Based on the PACF plot, we can select an autoregressive model with $p = 1$, where the series is differenced once ($d = 1$), and no moving average component ($q = 0$).
5. **ARIMA(0,1,1) or ARIMA(0,1,3)**: Based on the ACF plot, we can select a moving average model with $q = 1$ or $q = 3$, where the series is differenced once ($d = 1$), and no autoregressive component ($p = 0$).

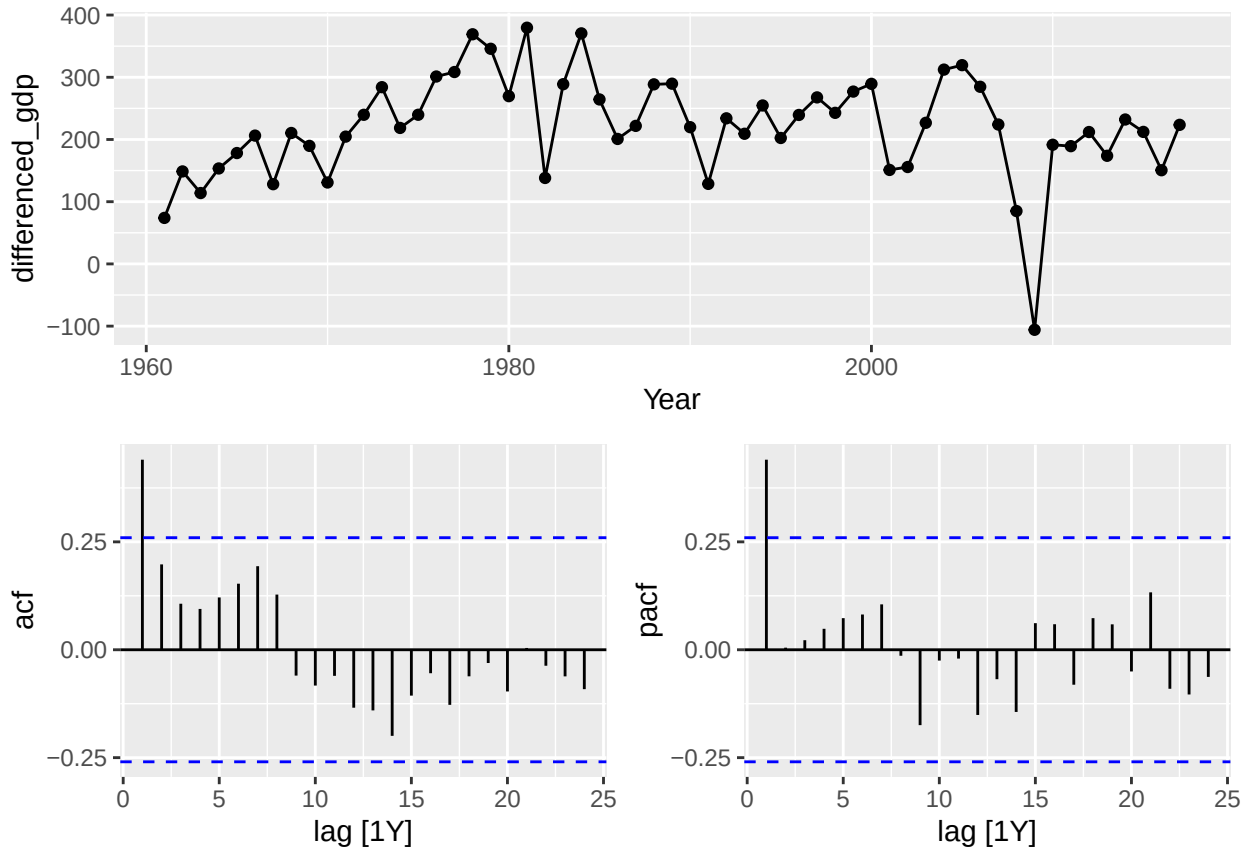
These models will capture the patterns indicated by the ACF and PACF plots.

```
# First difference the transformed data to make it stationary and remove NA values
us_gdp <- us_gdp |>
  mutate(differenced_gdp = difference(transformed_gdp)) |>
```

```
  drop_na(differenced_gdp)

# Plot the differenced series
us_gdp |>
  autoplot(differenced_gdp) +
  labs(title = "Differenced Box-Cox Transformed U.S. GDP")
```




```
# Plot ACF/PACF for the differenced U.S. GDP series
us_gdp |>
  gg_tsdisplay(differenced_gdp, plot_type = "partial", lag_max = 24)
```



0.7.4 d. choose what you think is the best model and check the residual diagnostics;

Model Selection and Interpretation:

1. **ARIMA(1,1,0)**:
   - **AICc = 644.2101**, **BIC = 649.8246**
   - This model was selected based on the PACF plot, which showed a significant spike at lag 1, suggesting an autoregressive (AR) component with $p = 1$. The series was differenced once ($d = 1$) to achieve stationarity. This model includes no moving average ($q = 0$) and performed well according to the AICc and BIC values.
2. **ARIMA(0,1,3)**:
   - **AICc = 648.9743**, **BIC = 657.9010**
   - This model was selected based on the ACF plot, which showed significant spikes at lags 1 and 3, indicating the presence of a moving average (MA) component with $q = 3$. The series was differenced once ($d = 1$) to achieve stationarity. The AICc and BIC values for this model were slightly higher, indicating that it is not as strong as the ARIMA(1,1,0) model.
3. **Stepwise Model (ARIMA(1,1,0))**:
   - **AICc = 644.2101**, **BIC = 649.8246**
   - The stepwise model selection process chose the ARIMA(1,1,0) model, which matches the model chosen manually based on the PACF plot. This confirms that the stepwise method effectively identified the same model as the best option according to AICc and BIC values.
4. **Search Model (ARIMA(1,1,0))**:

- **AICc = 644.2101, BIC = 649.8246**
- The search model, which systematically explores potential ARIMA models, also selected the ARIMA(1,1,0) model. This further supports the validity of the selected ARIMA(1,1,0) model.

```
library(knitr)
library(kableExtra)

# Fit ARIMA models to the transformed U.S. GDP data
us_gdp_fit <- us_gdp |>
  model(
    arima110 = ARIMA(transformed_gdp ~ pdq(1,1,0)),
    arima011 = ARIMA(transformed_gdp ~ pdq(0,1,1)),
    arima013 = ARIMA(transformed_gdp ~ pdq(0,1,3)),
    stepwise = ARIMA(transformed_gdp),
    search = ARIMA(transformed_gdp, stepwise = FALSE)
  )


# Extract the model summary with AICc and BIC values
model_summary <- glance(us_gdp_fit) |>
  arrange(AICc) |>
  select(.model, AICc, BIC)

# Extract the ARIMA orders (p,d,q) for each model
model_orders <- us_gdp_fit |>
  pivot_longer(everything(), names_to = "Model name", values_to = "Orders") |>
  select(`Model name`, Orders)

# Combine the AICc, BIC and ARIMA orders into a single data frame
combined_results <- model_summary |>
  left_join(model_orders, by = c(".model" = "Model name"))

# Display the combined data frame using kable
combined_results %>%
  kable() %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed"))
```

.model	AICc	BIC	Orders
arima110	644.2101	649.8246	<ARIMA(1,1,0) w/ drift>
stepwise	644.2101	649.8246	<ARIMA(1,1,0) w/ drift>
search	644.2101	649.8246	<ARIMA(1,1,0) w/ drift>
arima011	645.5071	651.1217	<ARIMA(0,1,1) w/ drift>
arima013	648.9743	657.9010	<ARIMA(0,1,3) w/ drift>

0.7.4.1 i. Best Model Selected

Based on both manual inspection of the ACF and PACF plots and the automated stepwise and search processes, the **ARIMA(1,1,0)** model emerges as the most appropriate for forecasting the U.S. GDP data, as it consistently shows the best AICc and BIC values. The ARIMA(0,1,1) and ARIMA(0,1,3) models also performed reasonably well but is less competitive in terms of these metrics.

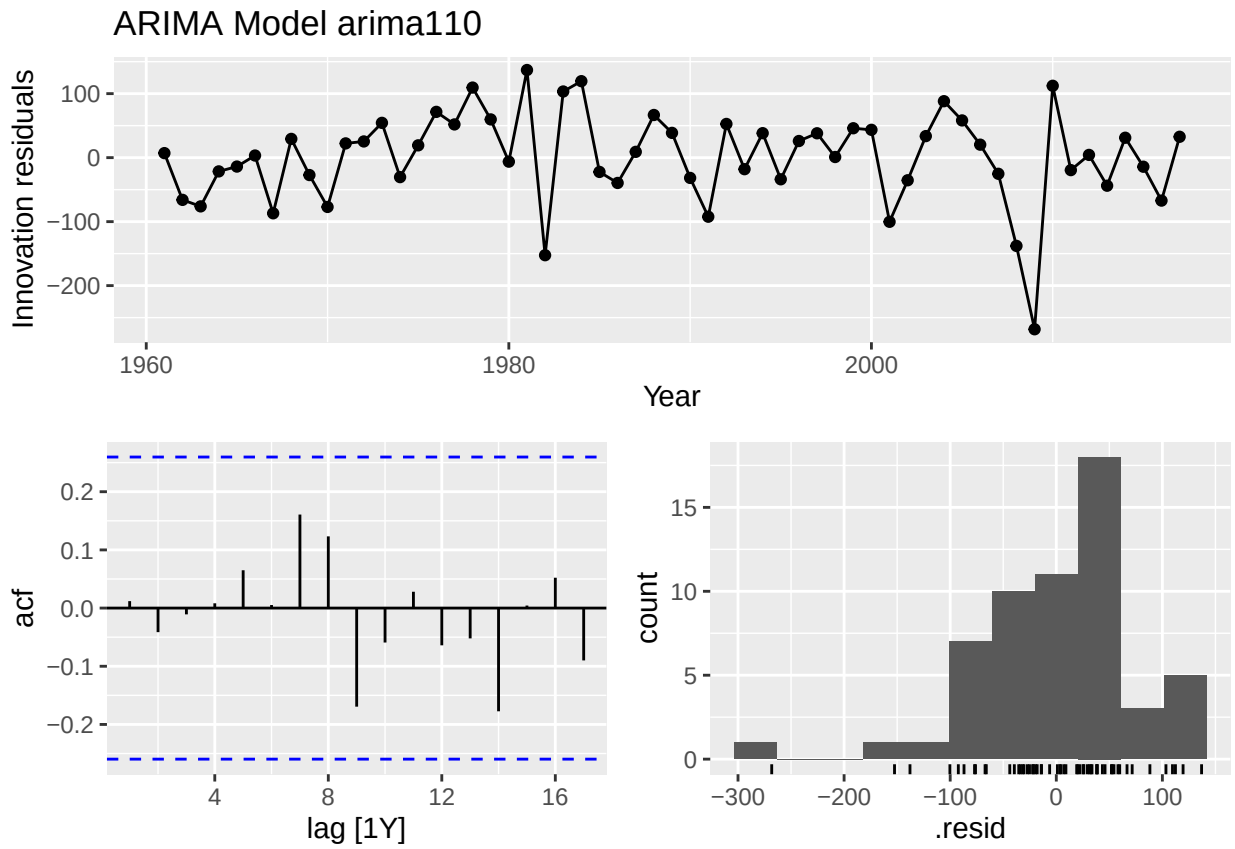
0.7.4.2 ii. Residual Diagnostics of Bst Model

1. **Innovation Residuals**: The residuals oscillate around zero without any significant patterns, indicating that the model has captured the underlying trend and seasonality in the data well. However, there appear to be a few larger residuals, especially in more recent periods, suggesting potential outliers.

2. **Autocorrelation Function (ACF)**: Most of the autocorrelations lie within the 95% confidence bands, implying that there is little autocorrelation remaining in the residuals. This suggests that the residuals resemble white noise, which is a good indication of model fit.

3. **Residual Distribution**: The histogram of residuals shows a reasonably normal distribution, although there are a few larger residuals on both sides. This suggests that the model's residuals are fairly well-behaved, but the presence of a few outliers indicates that there might still be room for improvement in the model.

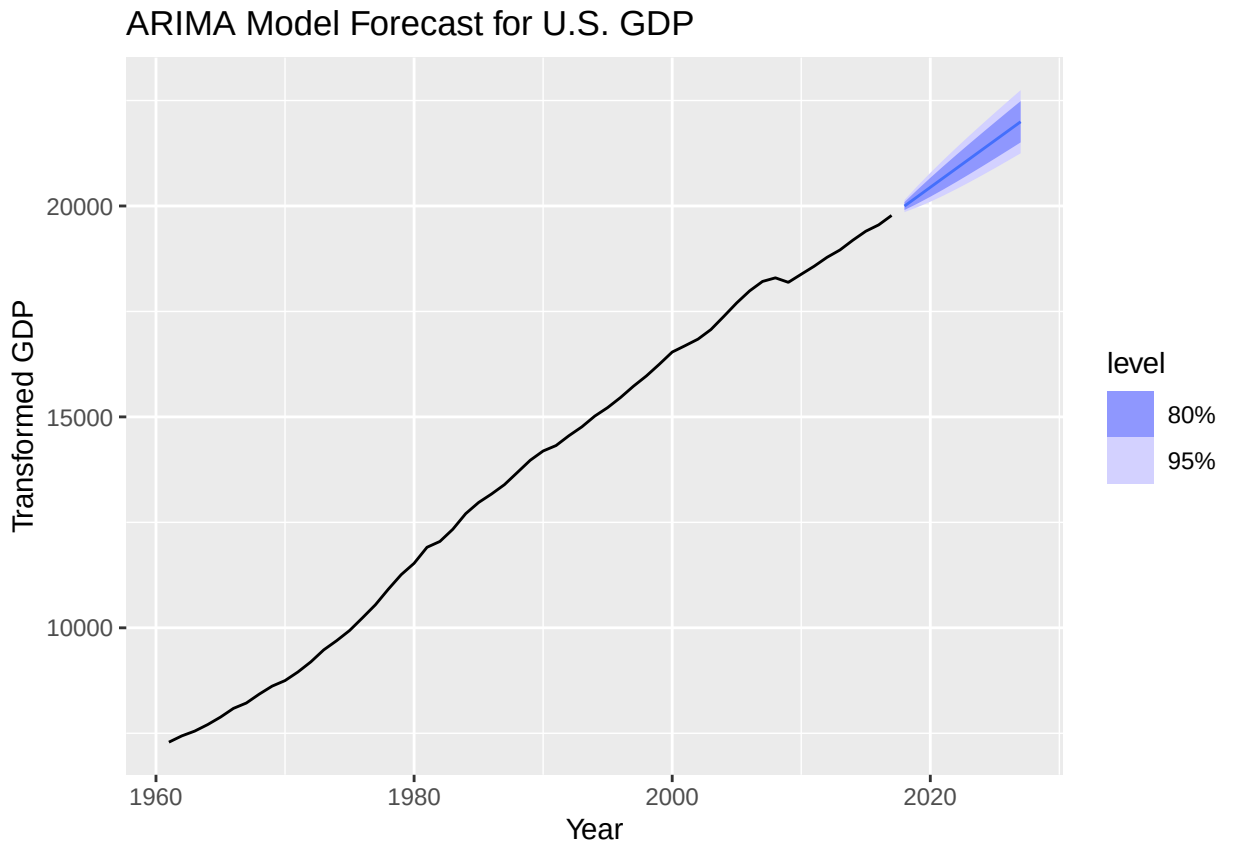
```
# Check residuals for white noise
us_gdp_fit |> select(arima110) |>
  gg_tsresiduals() +
  labs(title = "ARIMA Model arima110")
```



0.7.5 e. produce forecasts of your fitted model. Do the forecasts look reasonable?

```
# Forecast U.S. GDP for the next 10 periods using the arima110 model
us_gdp_forecast <- us_gdp_fit |>
  forecast(h = 10) |>
  filter(.model == 'arima110')
```

```r
# Plot the forecast
us_gdp_forecast |>
  autoplot(us_gdp) +
  labs(title = "ARIMA Model Forecast for U.S. GDP",
       y = "Transformed GDP")
```



0.7.5.1 i. Fit the ETS Model and Compare with ARIMA

0.7.5.1.1 Step 1. Data Preparation for Comparison We will apply **Min-Max scaling** to the data for both models and retain the forecast intervals for comparison.

- **ETS Model**: This model is fit to the original U.S. GDP series without any transformation.
- **ARIMA(1,1,0) Model**: This model is fit to the Box-Cox transformed U.S. GDP data.
- **Min-Max Scaling**: We apply min-max scaling to both the original and transformed GDP series for a better comparison.

0.7.5.1.2 Step 2. ETS vs ARIMA Model with Transformation

- The **ETS model** (without transformation) shows wider confidence intervals compared to the **ARIMA(1,1,0) model with Box-Cox transformation**, indicating more uncertainty in the ETS forecasts. The transformation in the ARIMA model seems to have provided tighter intervals, suggesting more confidence in the future predictions.

- **Impact of Transformation**: The **Box-Cox transformation** applied to the ARIMA model has effectively stabilized the variance, resulting in smoother growth predictions for U.S. GDP compared to the untransformed ETS model, which exhibits more volatility in its forecast.

```
# Install required packages if not already installed
if (!require("scales")) install.packages("scales")
```

```
## Loading required package: scales
```

```
# Load necessary libraries
library(fable)
library(dplyr)
library(ggplot2)
library(tsibble)
library(scales)

# Fit ETS model for U.S. GDP
us_gdp_fit_ets <- us_gdp |>
  model(ETS(GDP ~ error("A") + trend("A") + season("N")))

# Fit ARIMA(1,1,0) model for transformed GDP
us_gdp_fit_arima110 <- us_gdp |>
  model(ARIMA(transformed_gdp ~ pdq(1,1,0)))

# Function to apply Min-Max scaling
min_max_scale <- function(x) {
  (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) - min(x, na.rm = TRUE))
}

# Apply scaling to the original GDP and transformed GDP for comparison
us_gdp <- us_gdp |>
  mutate(scaled_gdp = min_max_scale(GDP),
         scaled_transformed_gdp = min_max_scale(transformed_gdp))

# Forecast using ETS model and ARIMA(1,1,0) model
ets_forecast <- us_gdp_fit_ets |>
  forecast(h = 10)

arima110_forecast <- us_gdp_fit_arima110 |>
  forecast(h = 10)

# Plot ETS forecast with original forecast intervals, but scaled GDP for comparison
plot1 <- autoplot(ets_forecast, us_gdp, level = c(80, 95)) +
  aes(y = scaled_gdp) +  # Keep the forecast intervals, scale only the data
  labs(title = "ETS (no Transformation) \nForecast for U.S. GDP",
       y = "Scaled GDP") +
  theme_minimal()

# Plot ARIMA(1,1,0) forecast with original forecast intervals, but scaled transformed GDP for comparis
plot2 <- autoplot(arima110_forecast, us_gdp, level = c(80, 95)) +
  aes(y = scaled_transformed_gdp) +  # Keep the forecast intervals, scale only the data
  labs(title = "ARIMA(1,1,0) (Box-Cox Transformed) \nForecast for U.S. GDP",
       y = "Scaled Transformed GDP") +
```

```
  theme_minimal()

# Combine the two plots side by side using patchwork
plot1 + plot2 + plot_layout(guides = 'collect') +
 theme(aspect.ratio = 1/1) # Set aspect ratio: equal width and height (1:1)
```