Data 624 - HW6 (Fall 2024)

Khyati Naik

9.1 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 primary difference among these figures lies in the number of random numbers each graph represents. As we move from the ACF for 36 random numbers to the ACF for 1,000 random numbers, we observe a general trend: the peaks and valleys of the ACFs diminish in height.

This observation can be attributed to the bounded area defined by the blue lines, which represent the significance threshold calculated by the formula:

$$\pm 1.96 \times \frac{1}{\sqrt{N}}$$

where N is the length of the time series. Notably, in all three graphs, the autocorrelations remain within this bounded area, suggesting that all data sets exhibit characteristics of white noise.

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?

The critical values differ from the mean of zero due to the calculation method:

$$\pm 1.96 \times \frac{1}{\sqrt{N}}$$

As the length of the time series N increases, these critical values approach zero because the denominator increases, reducing the distance from the mean. This pattern also corresponds to the observed differences in autocorrelation among the figures. Specifically, as the sample size of random numbers increases, the autocorrelations tend to decrease, reinforcing the idea that the data in each figure still reflect white noise properties, despite variations in their statistical characteristics.

9.2 A classic example of a non-stationary series are stock prices. 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.

```
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 4.3.3
```

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

```
## v tibble      3.2.1     v tsibble     1.1.3
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.0     v feasts      0.3.2
## v lubridate   1.9.2     v fable       0.3.4
## v ggplot2     3.5.1     v fabletools  0.4.2
```

```
## Warning: package 'dplyr' was built under R version 4.3.2
```

```
## Warning: package 'tidyr' was built under R version 4.3.2
```

```
## Warning: package 'ggplot2' was built under R version 4.3.3
```

```
## Warning: package 'tsibble' was built under R version 4.3.2
```

```
## Warning: package 'tsibbledata' was built under R version 4.3.3
```

```
## Warning: package 'feasts' was built under R version 4.3.3
```

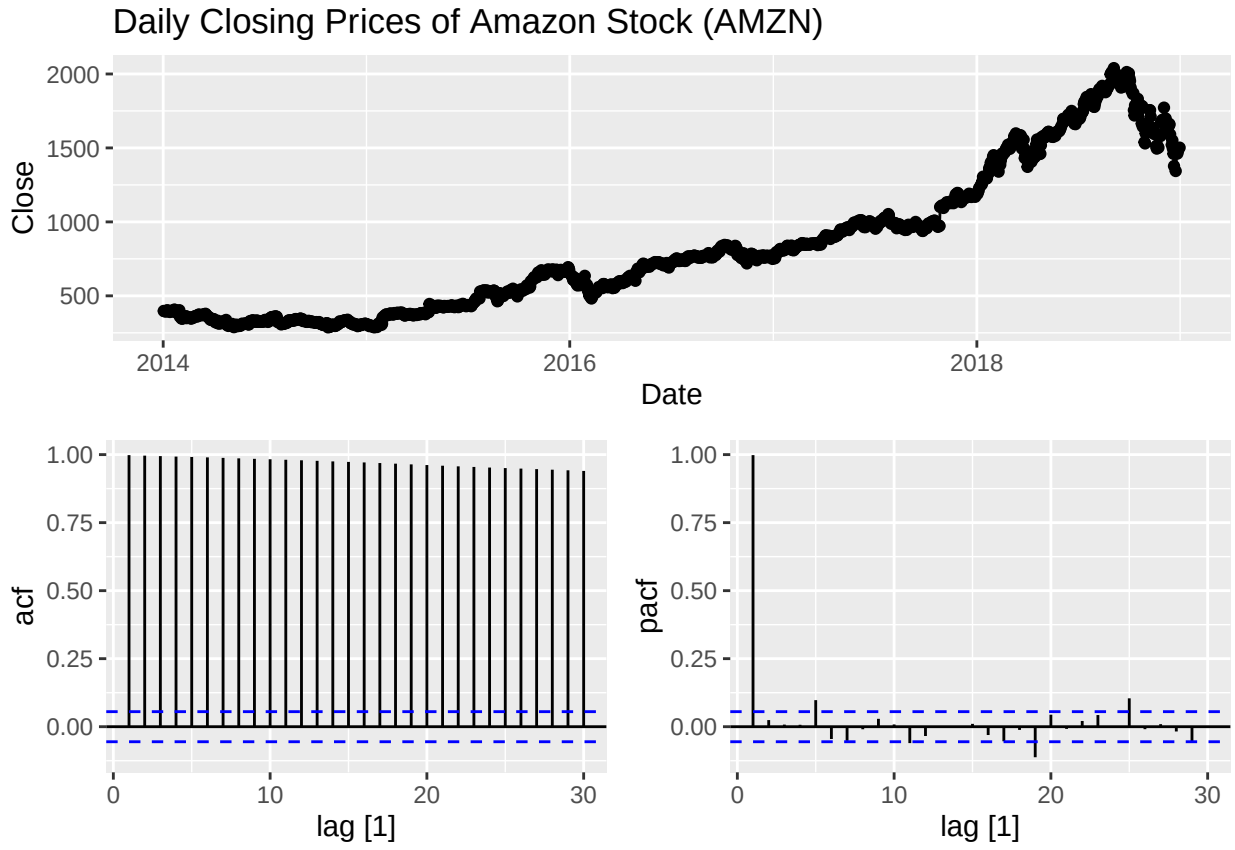
```
## Warning: package 'fabletools' was built under R version 4.3.3
```

```
## Warning: package 'fable' was built under R version 4.3.3
```

```
## -- 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
# Filter the GAFA stock data for Amazon's daily closing prices
# Plot the time series, ACF, and PACF for Amazon stock prices
amazon_stock <- gafa_stock %>%
  filter(Symbol == "AMZN")

# Plot the closing prices along with ACF and PACF
amazon_stock %>%
  gg_tsdisplay(Close, plot_type = 'partial') +
  labs(title = "Daily Closing Prices of Amazon Stock (AMZN)")
```

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




```r
# Determine how many differences are needed for stationarity (ndiffs)
diff_count <- amazon_stock %>%
  features(Close, unitroot_ndiffs)

# Check stationarity using the KPSS test
kpss_amzn <- amazon_stock %>%
  features(Close, unitroot_kpss)
```

Upon analyzing the daily closing prices of Amazon stock, we observe an upward trend that suggests the series is non-stationary. Here's how each plot reinforces that:

ACF and PACF Analysis:

Autocorrelation Function (ACF):
The ACF plot shows that the autocorrelation starts high and decreases slowly across lags. This is a typical sign of non-stationarity, as a stationary series would see the autocorrelations drop off quickly after a few lags. The slow decay indicates the presence of a trend in the data.

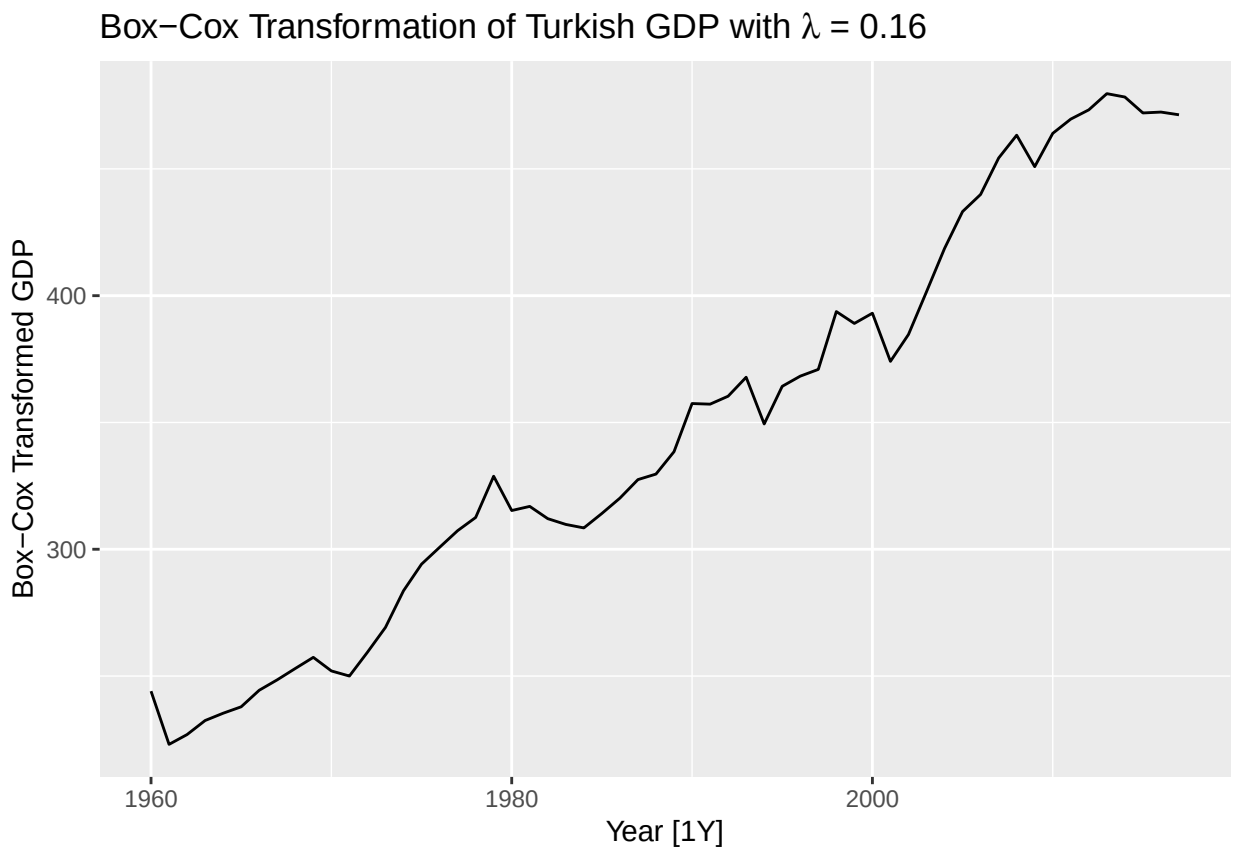
Partial Autocorrelation Function (PACF):
In the PACF plot, the first lag shows a strong positive correlation, while subsequent lags have smaller values. This pattern is common in non-stationary series because, unlike stationary series, the PACF doesn't drop off abruptly after the first lag, suggesting the data needs to be differenced.

Differencing for Stationarity:
Based on the KPSS test, one differencing step is sufficient to transform the non-stationary Amazon stock data into a stationary series. This is confirmed by the unit root test, indicating that the first difference will eliminate the trend, making the series more suitable for modeling.

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

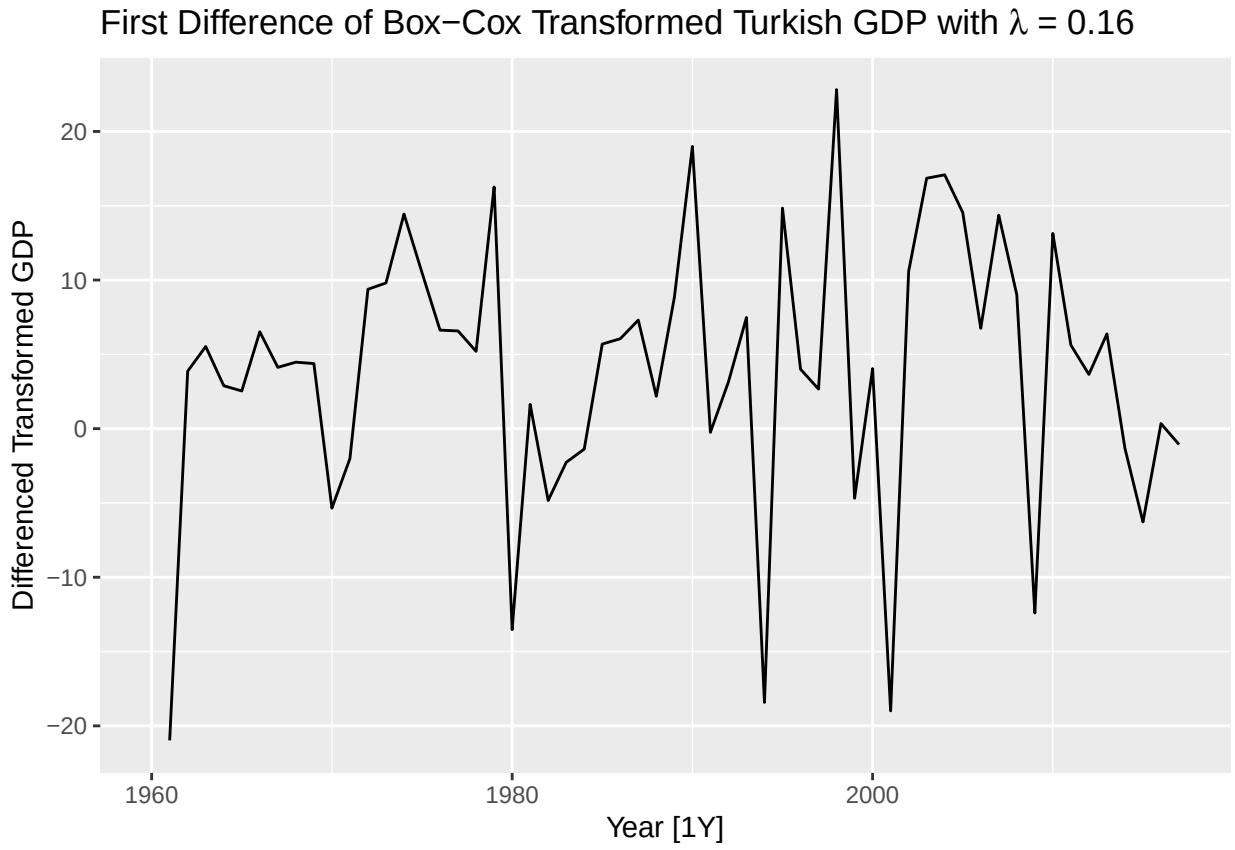
a. Turkish GDP from global_economy.

```
# Load the global_economy dataset and filter for Turkey's GDP data
gdp_turkey <- global_economy %>%
  filter(Country == "Turkey")

# Use the Guerrero method to determine the best lambda for Box-Cox transformation
lambda_gdp <- gdp_turkey %>%
  features(GDP, features = guerrero) %>%
  pull(lambda_guerrero)

# Plot the transformed Turkish GDP using the calculated lambda
gdp_turkey %>%
  autoplot(box_cox(GDP, lambda_gdp)) +
  labs(y = "Box-Cox Transformed GDP",
       title = latex2exp::TeX(paste0(
         "Box-Cox Transformation of Turkish GDP with $\\lambda$ = ",
         round(lambda_gdp, 2))))
```




```
# Apply first-order differencing to the Box-Cox transformed GDP series
gdp_turkey <- gdp_turkey %>%
  mutate(diff_transformed_gdp = difference(box_cox(GDP, lambda_gdp)))

# Plot the differenced Box-Cox transformed series to check for stationarity
gdp_turkey %>%
  autoplot(diff_transformed_gdp) +
  labs(y = "Differenced Transformed GDP",
       title = latex2exp::TeX(paste0(
         "First Difference of Box-Cox Transformed Turkish GDP with $\\lambda$ = ",
         round(lambda_gdp, 2))))
```

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




```
# Perform the KPSS test to assess stationarity of the differenced series
gdp_kpss_test <- gdp_turkey %>%
  features(diff_transformed_gdp, unitroot_kpss)
```

Box-Cox Transformation:
The Guerrero method was applied to determine the appropriate value of lambda, ensuring the GDP data is stabilized with respect to variance. The Box-Cox transformation helps deal with non-linearity in variance, common in economic series like GDP.

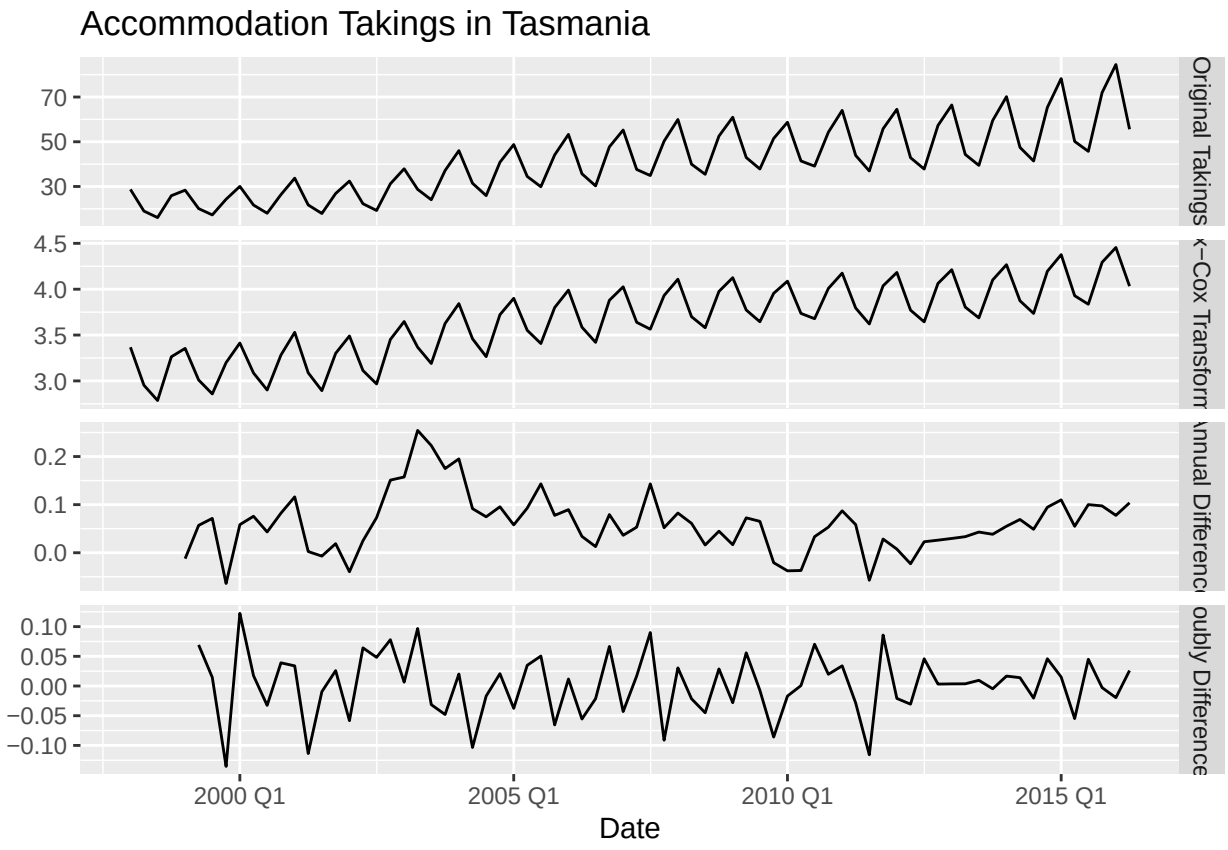
Differencing for Stationarity:
After applying the Box-Cox transformation, the series is differenced once to remove any trends. The first difference of the transformed series is plotted to check for stationarity visually.

KPSS Test:
Finally, the KPSS test is applied to the differenced data. A low test statistic (p-value > 0.1) suggests that the differenced series is stationary, indicating that the transformation and differencing have successfully stabilized the data.

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

```
# Load accommodation data for Tasmania
tas_accom <- aus_accommodation %>%
  filter(State == "Tasmania")
```

```r
# Use the Guerrero method to determine the appropriate lambda for the Box-Cox transformation
lambda_accom <- tas_accom %>%
  features(Takings, features = guerrero) %>%
  pull(lambda_guerrero)

# Transform the data using Box-Cox and apply differencing to explore stationarity
tas_accom %>%
  transmute(
    `Original Takings` = Takings,
    `Box-Cox Transformed` = box_cox(Takings, lambda_accom),
    `Annual Difference` = difference(box_cox(Takings, lambda_accom), 4),
    `Doubly Differenced` = difference(difference(box_cox(Takings, lambda_accom), 4), 1)
  ) %>%
  pivot_longer(-Date, names_to = "Type", values_to = "Takings") %>%
  mutate(
    Type = factor(Type, levels = c(
      "Original Takings",
      "Box-Cox Transformed",
      "Annual Difference",
      "Doubly Differenced"))
  ) %>%
  ggplot(aes(x = Date, y = Takings)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
  labs(title = "Accommodation Takings in Tasmania", y = NULL)
```



```
# Test for stationarity using the KPSS test after annual differencing
tas_accom %>%
  mutate(diff_bc_takings = difference(box_cox(Takings, lambda_accom), 4)) %>%
  features(diff_bc_takings, unitroot_kpss)
```

```
## # A tibble: 1 x 3
##   State    kpss_stat kpss_pvalue
##   <chr>        <dbl>       <dbl>
## 1 Tasmania     0.198         0.1
```

```
# Test for stationarity after applying double differencing
tas_accom %>%
  mutate(double_diff_bc_takings = difference(difference(box_cox(Takings, lambda_accom), 4), 1)) %>%
  features(double_diff_bc_takings, unitroot_kpss)
```

```
## # A tibble: 1 x 3
##   State    kpss_stat kpss_pvalue
##   <chr>        <dbl>       <dbl>
## 1 Tasmania    0.0474         0.1
```

```
# Test for seasonal differencing required (NSDIFF)
tas_accom %>%
  mutate(box_cox_takings = box_cox(Takings, lambda_accom)) %>%
  features(box_cox_takings, unitroot_nsdiffs)
```

```
## # A tibble: 1 x 2
##   State    nsdiffs
##   <chr>      <int>
## 1 Tasmania       1
```

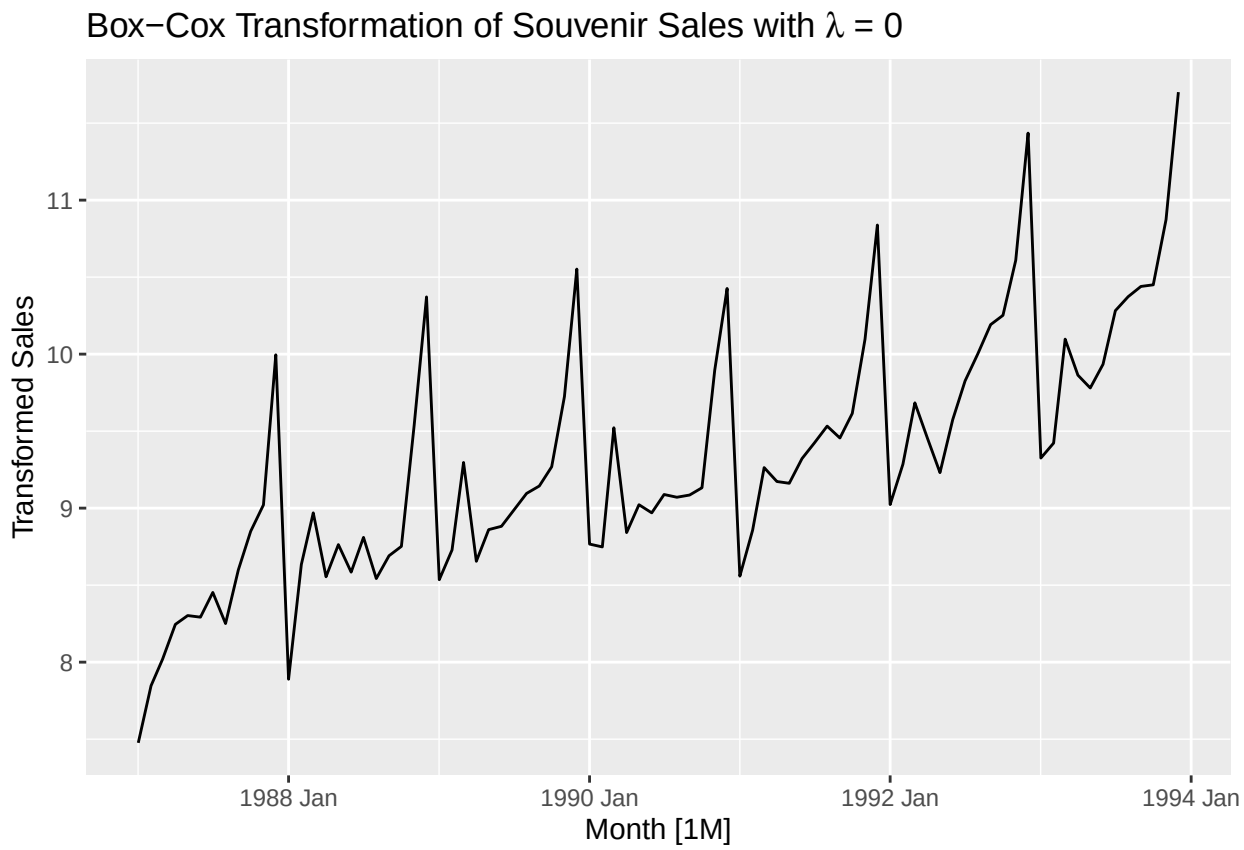
```
# Test for additional differencing on the seasonal differenced data (NDIFF)
tas_accom %>%
  mutate(seasonal_diff_bc_takings = difference(box_cox(Takings, lambda_accom), 4)) %>%
  features(seasonal_diff_bc_takings, unitroot_ndiffs)
```

```
## # A tibble: 1 x 2
##   State    ndiffs
##   <chr>     <int>
## 1 Tasmania      0
```

c. Monthly sales from souvenirs.

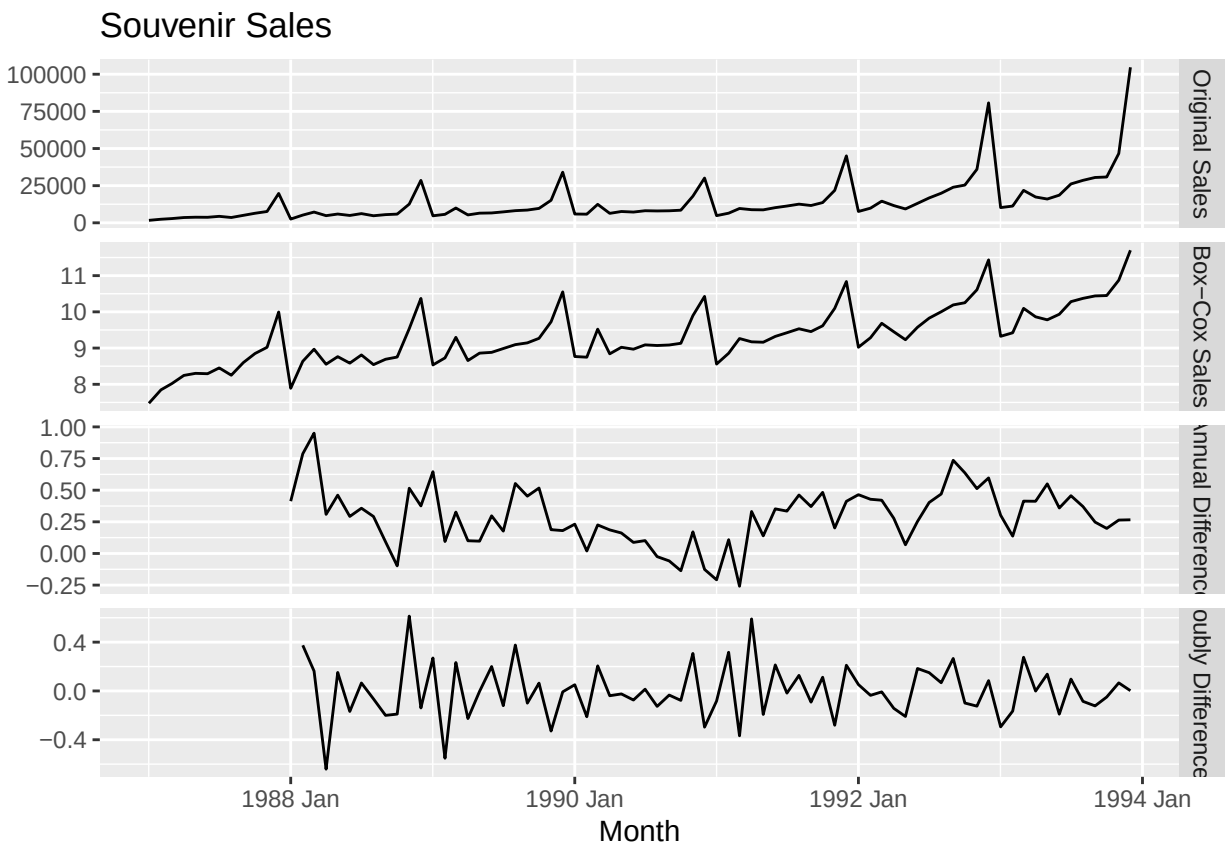
```
# Calculate the best lambda for Box-Cox transformation for souvenirs sales
lambda_sales <- souvenirs %>%
  features(Sales, features = guerrero) %>%
  pull(lambda_guerrero)

# Plot the Box-Cox transformed sales
souvenirs %>%
```

```
  autoplot(box_cox(Sales, lambda_sales)) +
  labs(y = "Transformed Sales",
       title = latex2exp::TeX(paste0(
         "Box-Cox Transformation of Souvenir Sales with $\\lambda$ = ",
         round(lambda_sales, 2))))
```




```
# Apply transformations and plot different differenced series for comparison
souvenirs %>%
  transmute(
    `Original Sales` = Sales,
    `Box-Cox Sales` = box_cox(Sales, lambda_sales),
    `Annual Difference` = difference(box_cox(Sales, lambda_sales), 12),
    `Doubly Differenced` = difference(difference(box_cox(Sales, lambda_sales), 12), 1)
  ) %>%
  pivot_longer(-Month, names_to = "Type", values_to = "Sales") %>%
  mutate(
    Type = factor(Type, levels = c(
      "Original Sales",
      "Box-Cox Sales",
      "Annual Difference",
      "Doubly Differenced"))
  ) %>%
  ggplot(aes(x = Month, y = Sales)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
```

```
  labs(title = "Souvenir Sales", y = NULL)
```




```
# KPSS test on first difference of Box-Cox transformed data
souvenirs %>%
  mutate(diff_bc_sales = difference(box_cox(Sales, lambda_sales), 12)) %>%
  features(diff_bc_sales, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.240         0.1
```

```
# KPSS test on doubly differenced Box-Cox transformed data
souvenirs %>%
  mutate(double_diff_bc_sales = difference(difference(box_cox(Sales, lambda_sales), 12), 1)) %>%
  features(double_diff_bc_sales, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0381         0.1
```

```r
# Test if seasonal differencing is required using NSDIFF
souvenirs %>%
  mutate(box_cox_sales = box_cox(Sales, lambda_sales)) %>%
  features(box_cox_sales, unitroot_nsdiffs)
```

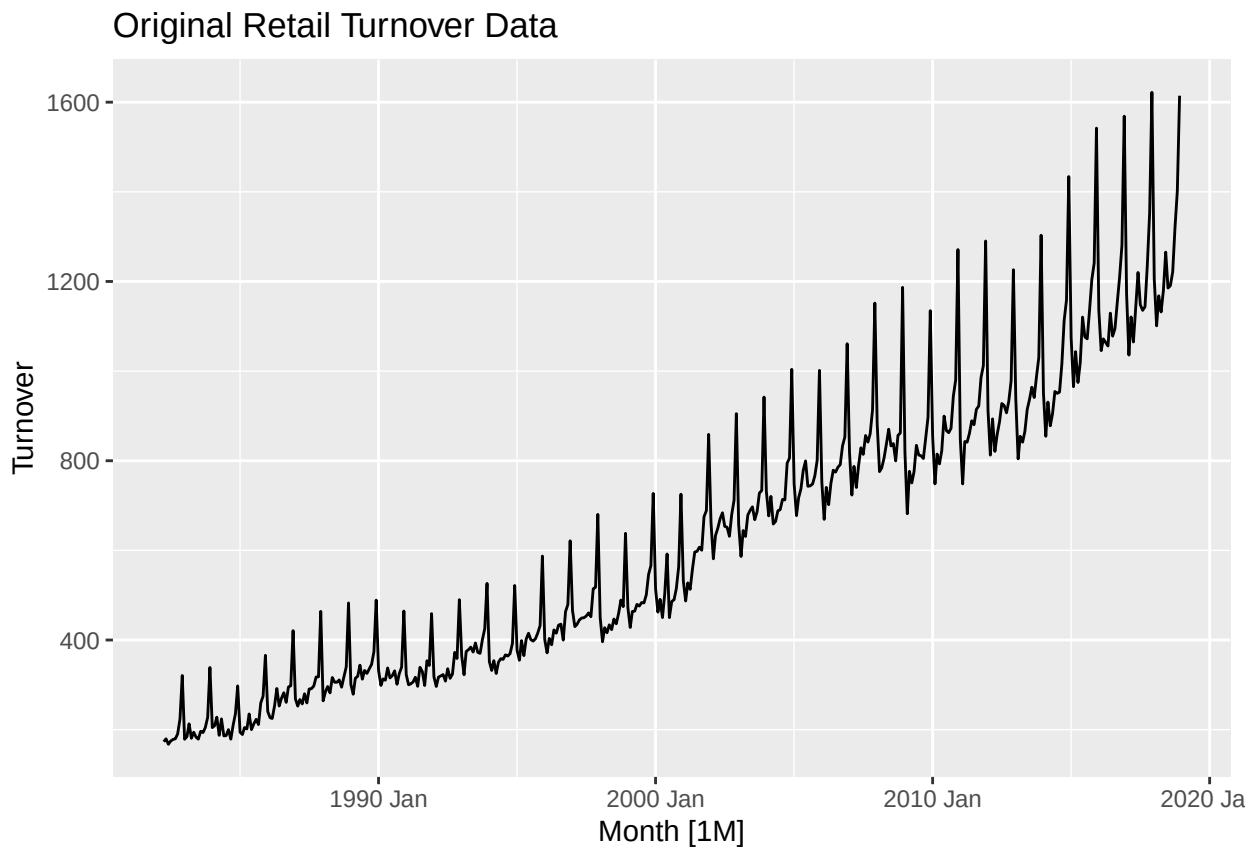
```
## # A tibble: 1 x 1
##   nsdiffs
##     <int>
## 1       1
```

```r
# Additional differencing (NDIFF) test on seasonally differenced series
souvenirs %>%
  mutate(seasonal_diff_bc_sales = difference(box_cox(Sales, lambda_sales), 12)) %>%
  features(seasonal_diff_bc_sales, unitroot_ndiffs)
```

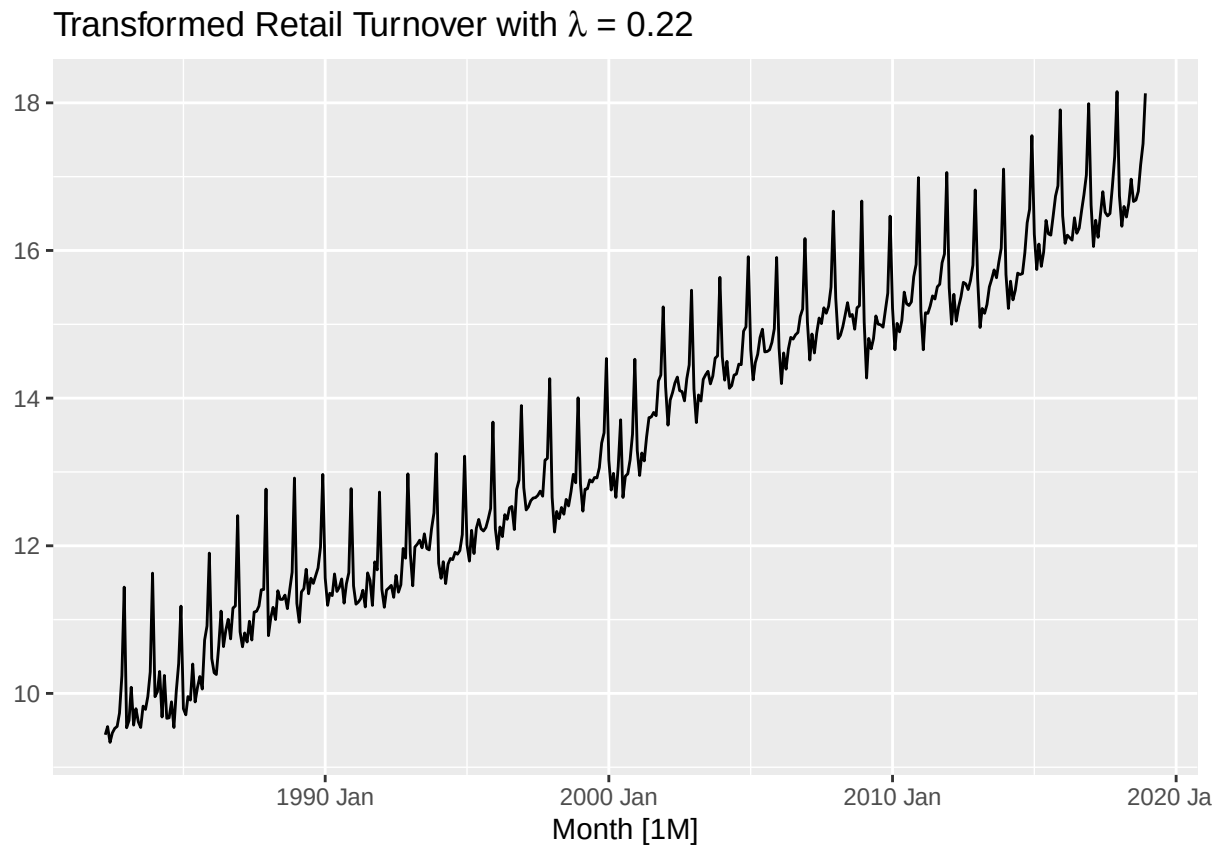
```
## # A tibble: 1 x 1
##   ndiffs
##    <int>
## 1      0
```

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.

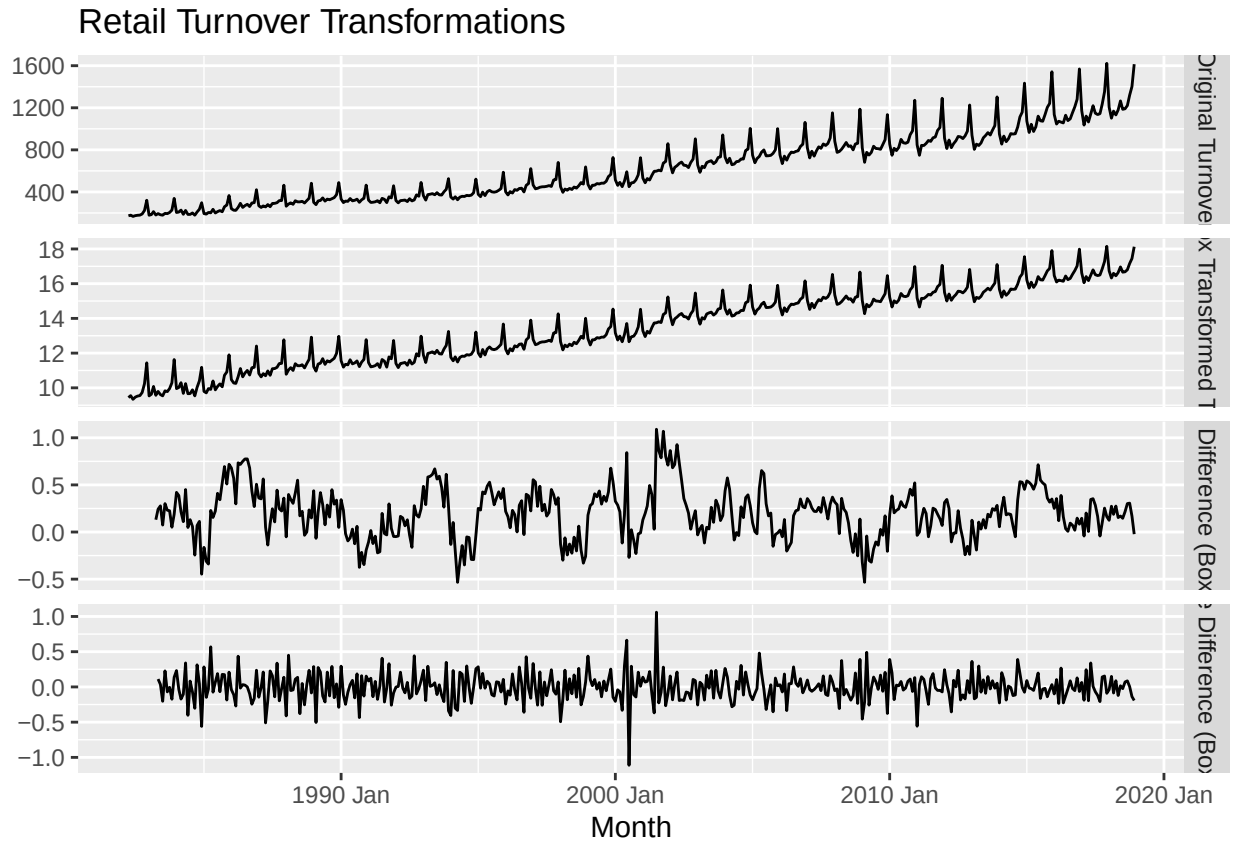
```r
# Set seed for reproducibility
set.seed(123)

# Filter monthly data by selecting a random 'Series ID'
retail_data <- aus_retail %>%
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))

# Plot the original turnover data
retail_data %>%
  autoplot(Turnover) +
  labs(title = "Original Retail Turnover Data", y = "Turnover")
```




```
# Calculate the optimal lambda for Box-Cox transformation using guerrero method
lambda_trans <- retail_data %>%
  features(Turnover, features = guerrero) %>%
  pull(lambda_guerrero)

# Plot the Box-Cox transformed turnover data
retail_data %>%
  autoplot(box_cox(Turnover, lambda_trans)) +
  labs(y = "",
       title = latex2exp::TeX(paste0(
         "Transformed Retail Turnover with $\\lambda$ = ",
         round(lambda_trans, 2))))
```




```r
# Create new columns for the original, Box-Cox transformed, and differenced data
retail_data %>%
  transmute(
    `Original Turnover` = Turnover,
    `Box-Cox Transformed Turnover` = box_cox(Turnover, lambda_trans),
    `Yearly Difference (Box-Cox)` = difference(box_cox(Turnover, lambda_trans), 12),
    `Double Difference (Box-Cox)` = difference(difference(box_cox(Turnover, lambda_trans), 12), 1)
  ) %>%
  pivot_longer(-Month, names_to = "Transformation", values_to = "Turnover") %>%
  mutate(
    Transformation = factor(Transformation, levels = c(
      "Original Turnover",
      "Box-Cox Transformed Turnover",
      "Yearly Difference (Box-Cox)",
      "Double Difference (Box-Cox)"))
  ) %>%
  ggplot(aes(x = Month, y = Turnover)) +
  geom_line() +
  facet_grid(vars(Transformation), scales = "free_y") +
  labs(title = "Retail Turnover Transformations", y = NULL)
```




```
# Check for seasonal differencing requirements using nsdiffs (non-seasonal differencing)
retail_data %>%
  mutate(box_cox_turnover = box_cox(Turnover, lambda_trans)) %>%
  features(box_cox_turnover, unitroot_nsdiffs)
```

```
## # A tibble: 1 x 3
##   State    Industry                  nsdiffs
##   <chr>    <chr>                       <int>
## 1 Victoria Household goods retailing       1
```

```
# Check for first-order differencing requirements after seasonal differencing
retail_data %>%
  mutate(box_cox_turnover = difference(box_cox(Turnover, lambda_trans), 12)) %>%
  features(box_cox_turnover, unitroot_ndiffs)
```

```
## # A tibble: 1 x 3
##   State    Industry                  ndiffs
##   <chr>    <chr>                      <int>
## 1 Victoria Household goods retailing      0
```

9.6 Simulate and plot some data from simple ARIMA models.

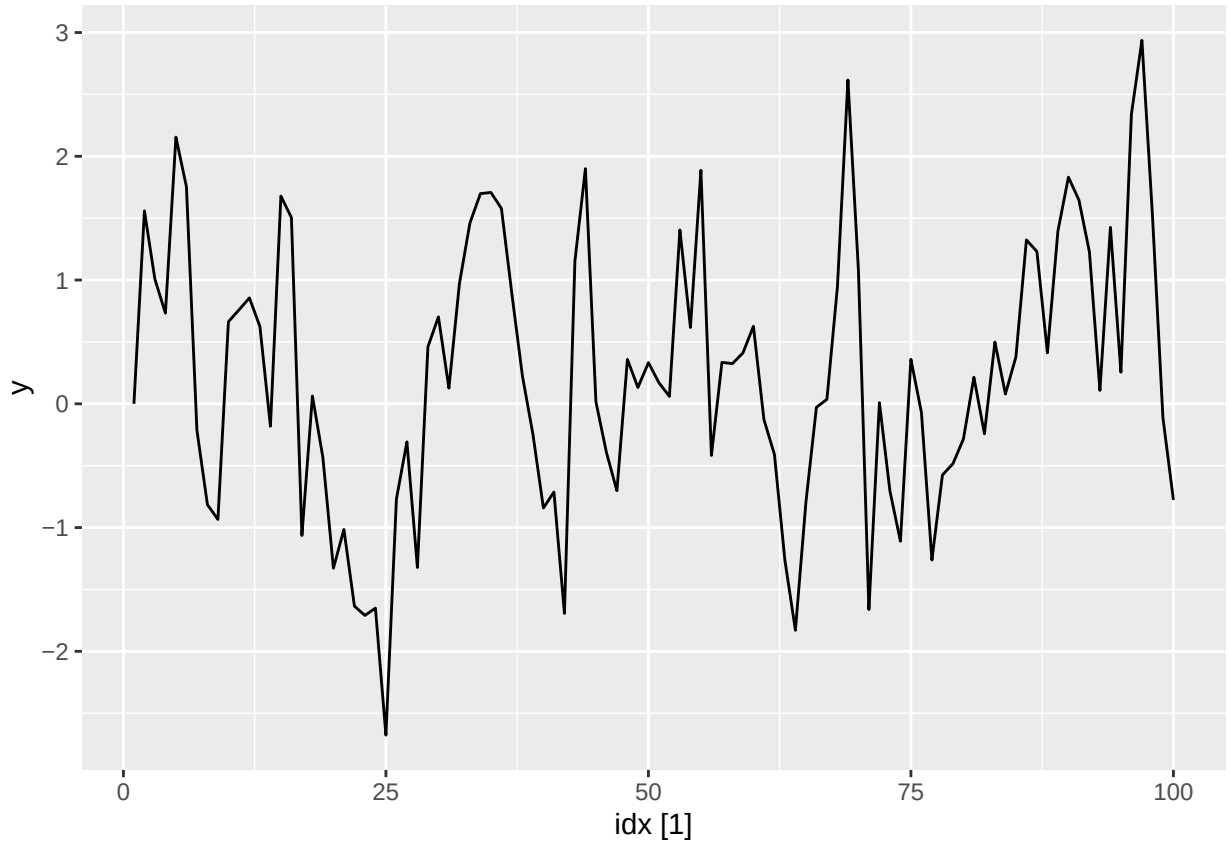
a. Use the following R code to generate data from an AR(1) model with

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)

```
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)
```

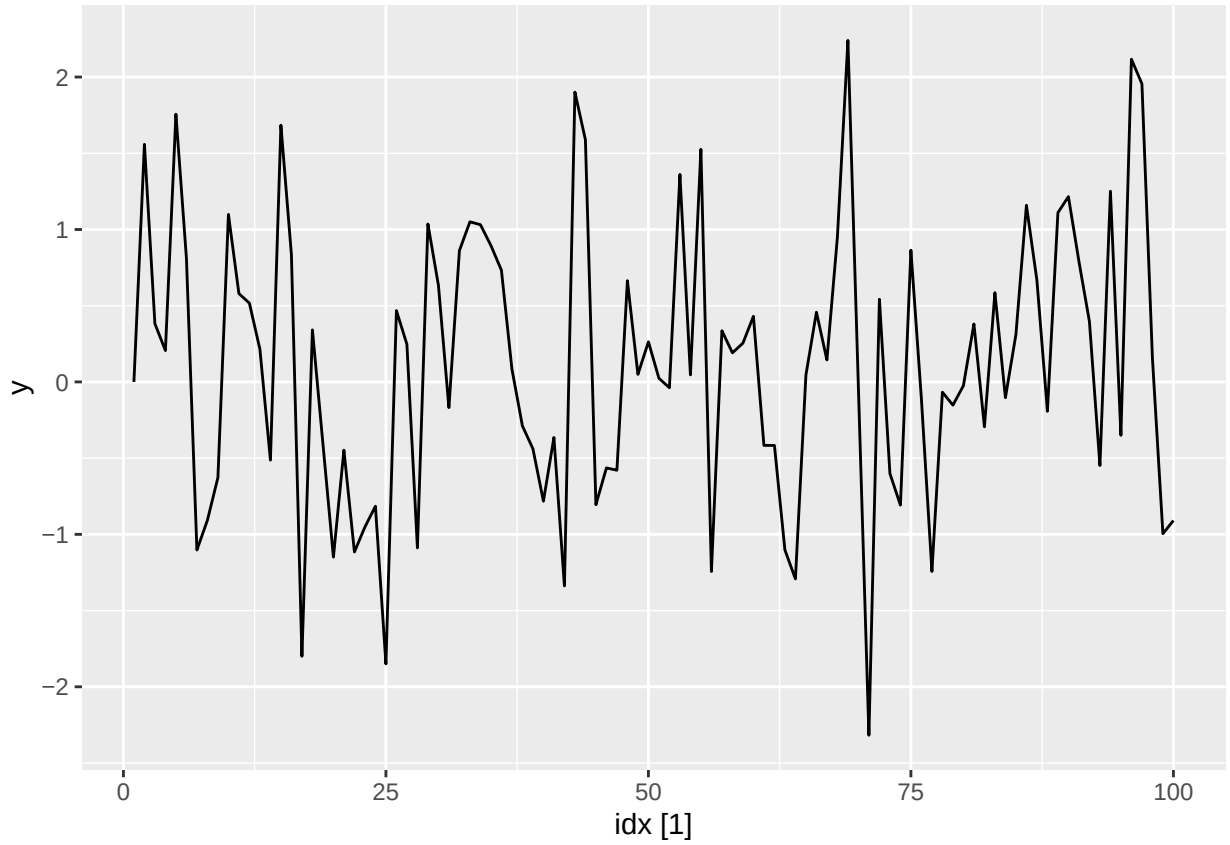
b. Produce a time plot for the series. How does the plot change as you change

```
sim %>% autoplot(y)
```

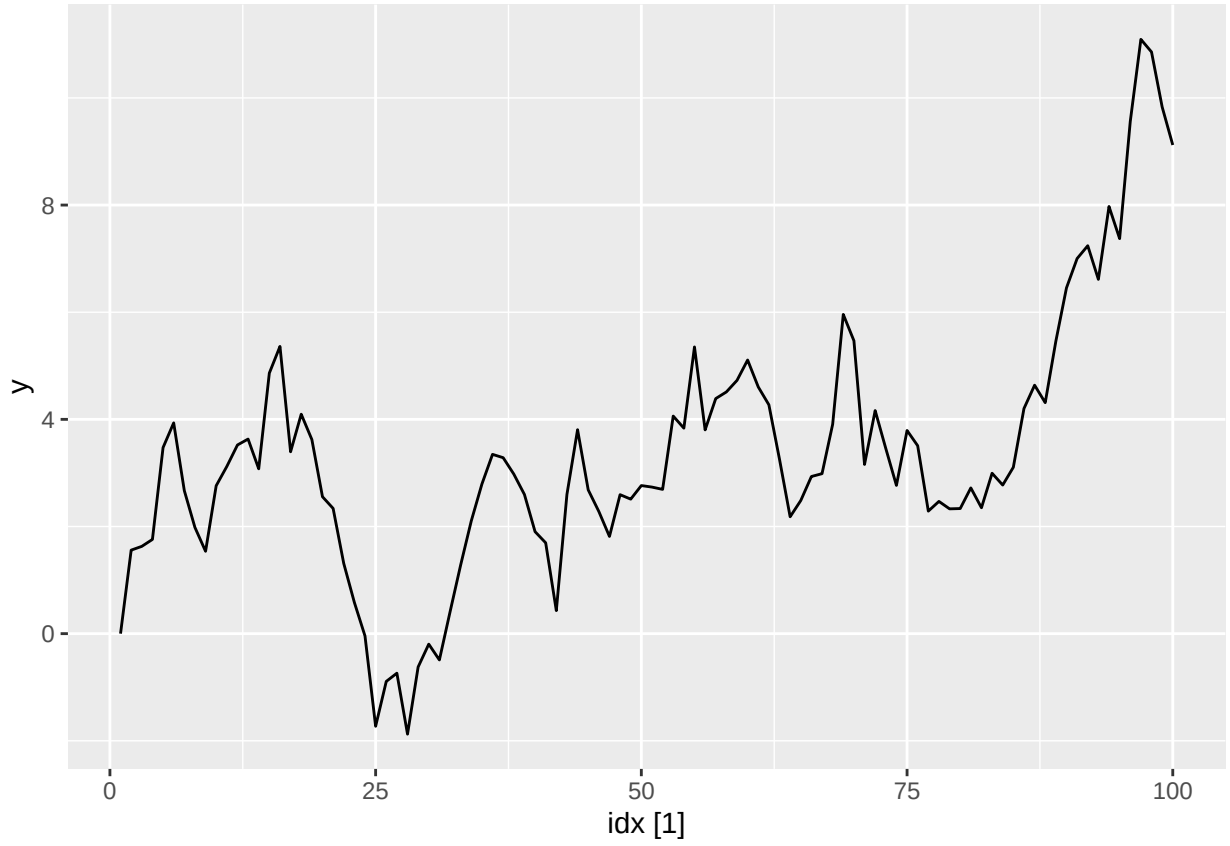



```
# phi=0.2
for(i in 2:100)
  y[i] <- 0.2*y[i-1] + e[i]
```

```r
sim02 <- tsibble(idx = seq_len(100), y = y, index = idx)

# phi=1.0
for(i in 2:100)
  y[i] <- 1.0*y[i-1] + e[i]
sim10 <- tsibble(idx = seq_len(100), y = y, index = idx)

sim02 %>% autoplot(y)
```




```r
sim10 %>% autoplot(y)
```

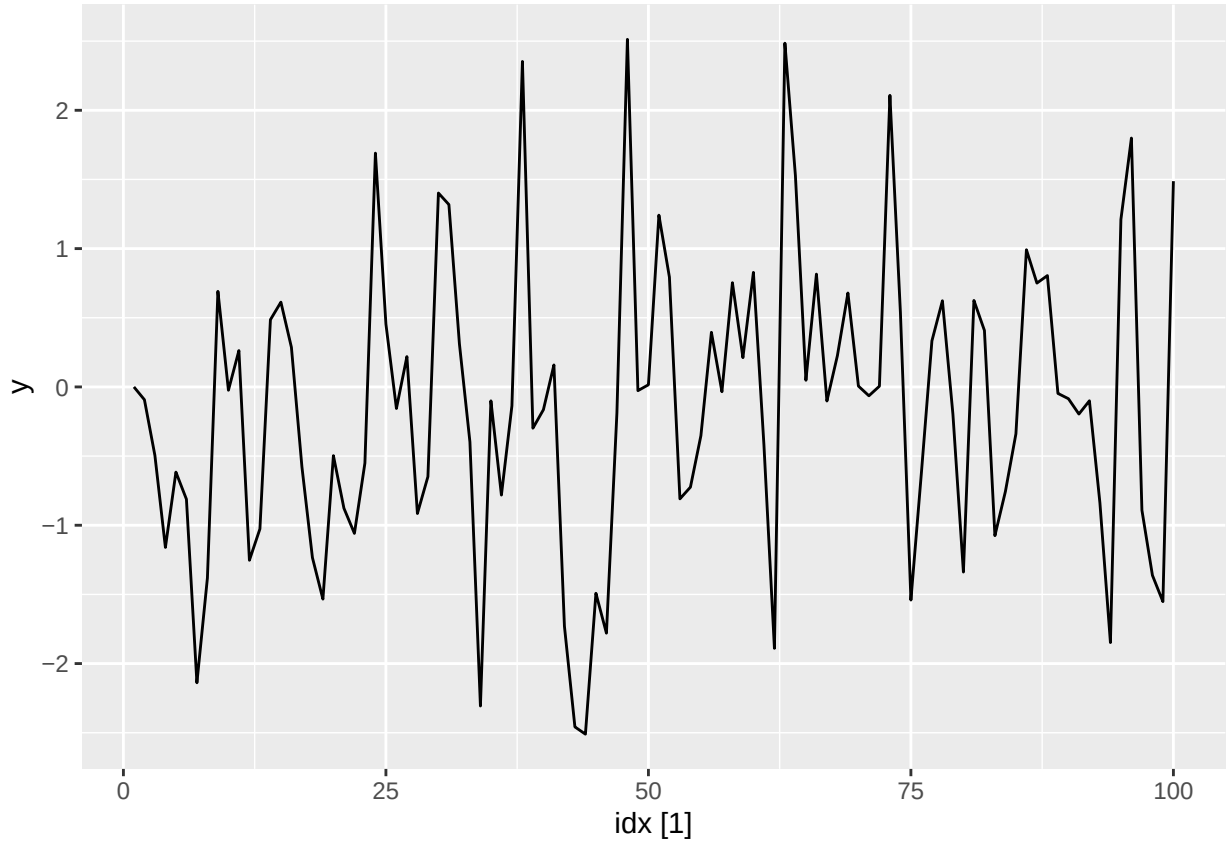


c. Write your own code to generate data from an MA(1) model with

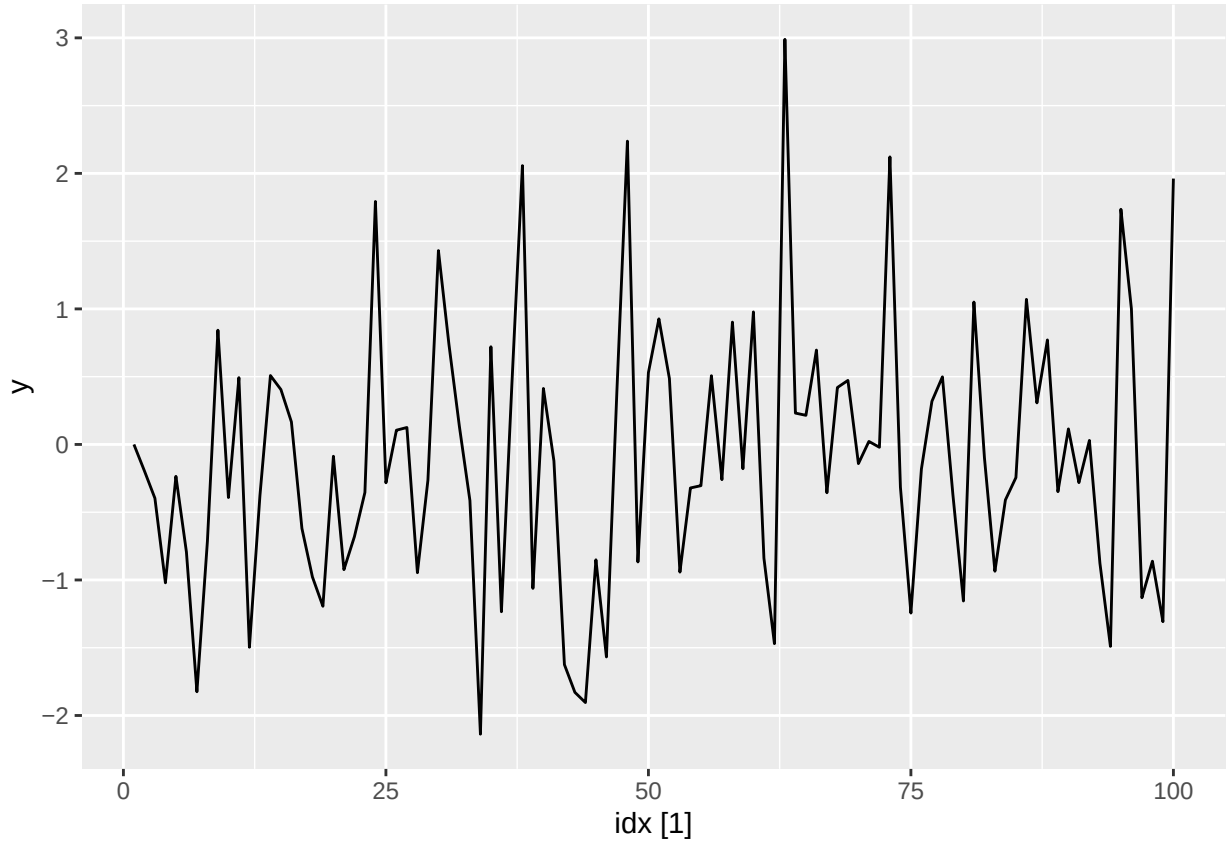
```
y <- numeric(100)
e <- rnorm(100)

for(i in 2:100)
  y[i] <- e[i] + 0.6*e[i-1]

sim_ma1 <- tsibble(idx = seq_len(100), y = y, index = idx)
```

d. Produce a time plot for the series. How does the plot change as you change

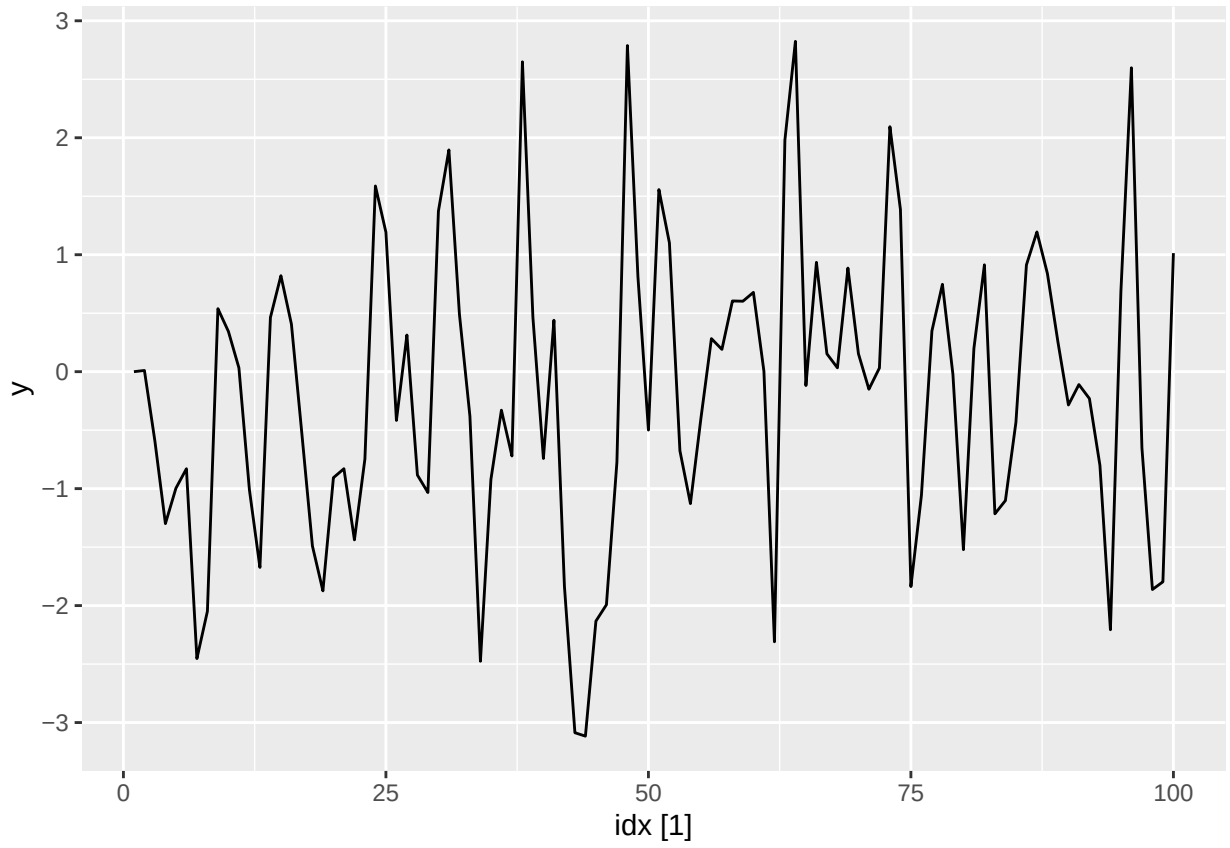
```
sim_ma1 %>% autoplot(y)
```




```
# theta is 0.2
for(i in 2:100)
  y[i] <- e[i] + 0.2*e[i-1]
sim_ma02 <- tsibble(idx = seq_len(100), y = y, index = idx)

# theta is 1.0
for(i in 2:100)
  y[i] <- e[i] + 1.0*e[i-1]
sim_ma10 <- tsibble(idx = seq_len(100), y = y, index = idx)

sim_ma02 %>% autoplot(y)
```




```
sim_ma10 %>% autoplot(y)
```



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

```
y <- numeric(100)
e <- rnorm(100)

phi <- 0.6
theta <- 0.6

for(i in 2:100)
  y[i] <- phi*y[i-1] + theta*e[i-1] + e[i]

sim_arma11 <- tsibble(idx = seq_len(100), y = y, index = idx)
```

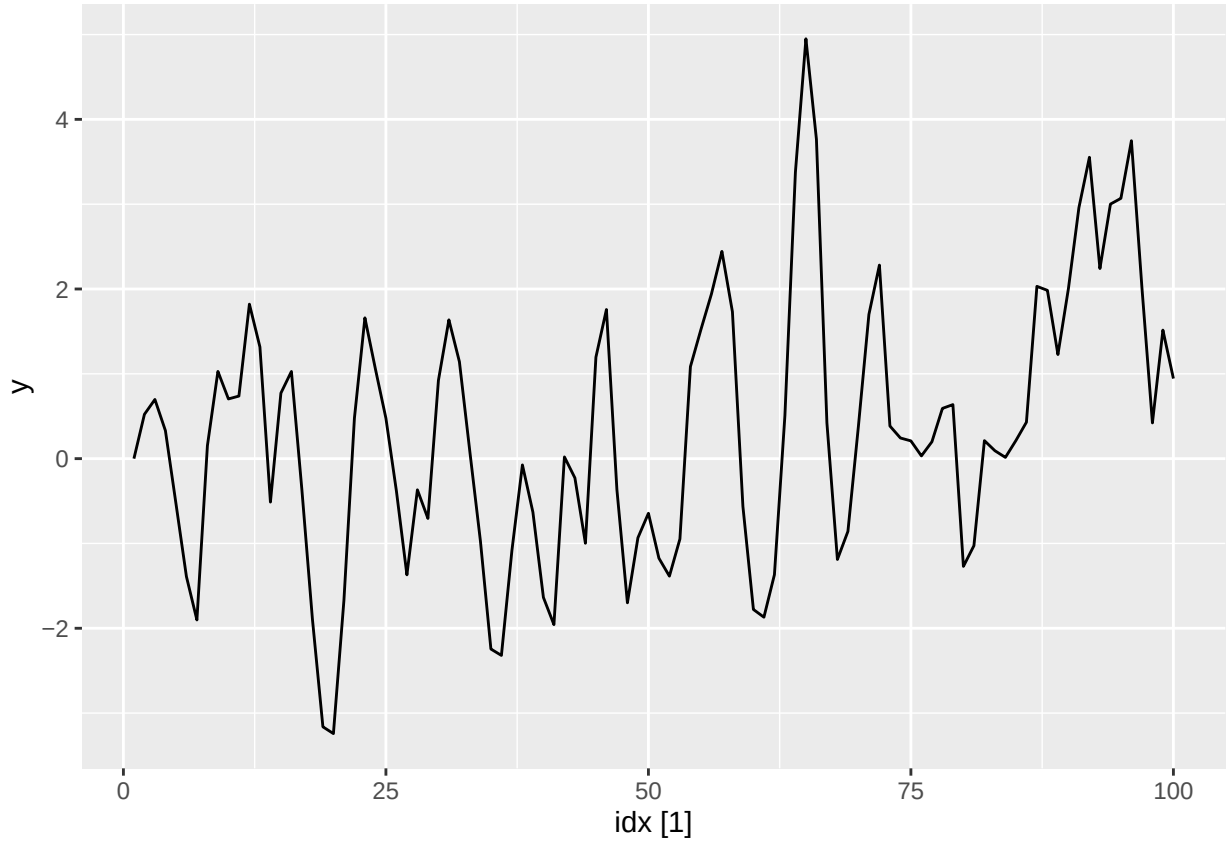
f. Generate data from an AR(2) model (Note that these parameters will give a non-stationary series.)

```
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]
```

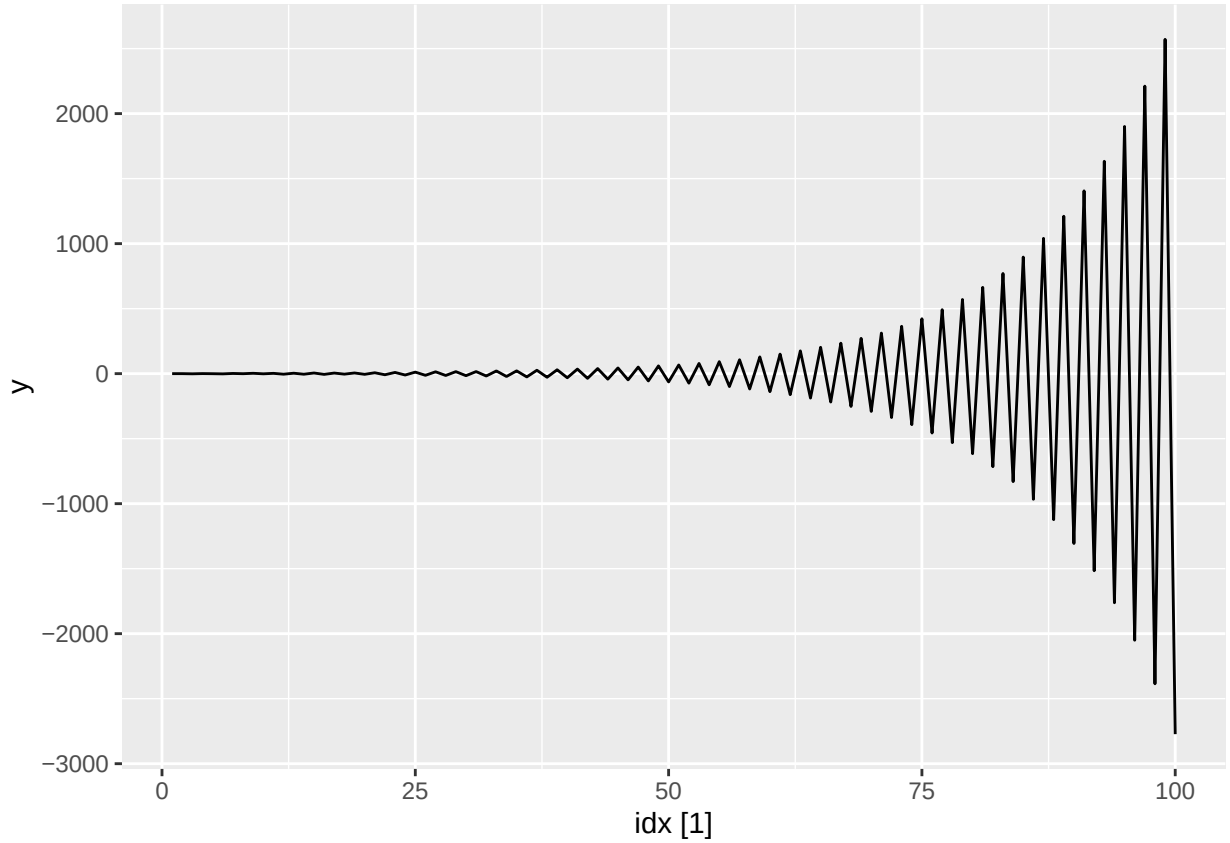
```
sim_ar2 <- tsibble(idx = seq_len(100), y = y, index = idx)
```

g. Graph the latter two series and compare them.

```
sim_arma11 %>% autoplot(y)
```




```
sim_ar2 %>% autoplot(y)
```



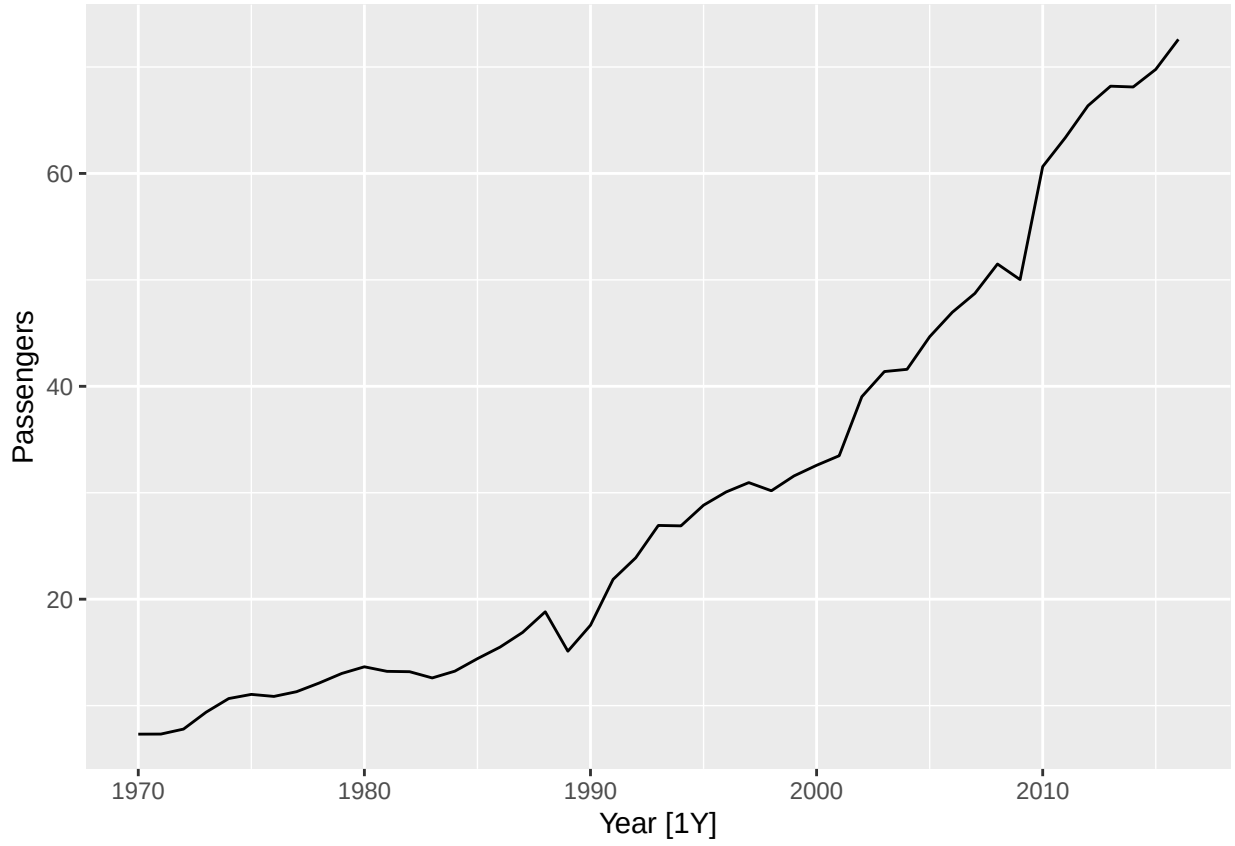
9.7 Consider aus_airpassengers, the total number of passengers (in millions) from Australian air carriers for the period 1970-2011.

```
# Explore the first few rows and plot the data
aus_airpassengers %>% head()
```

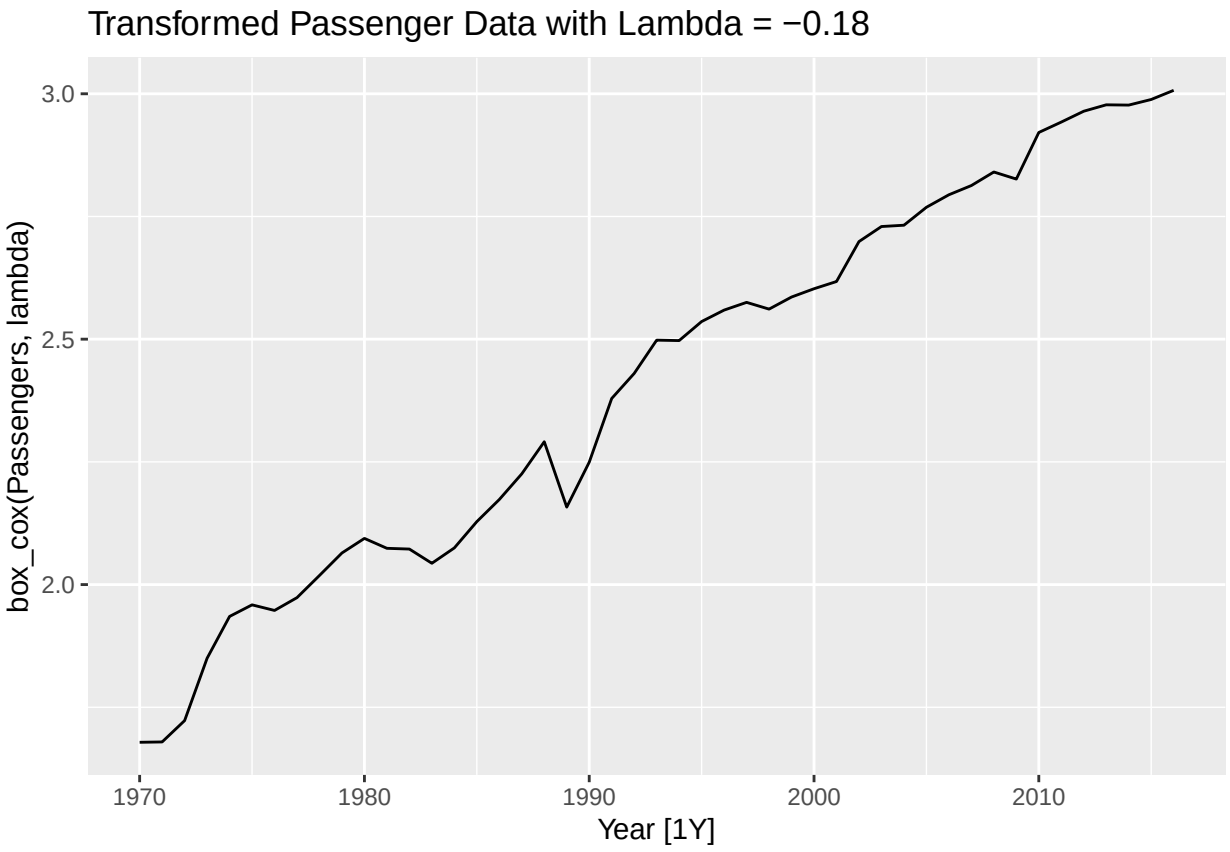
```
## # A tsibble: 6 x 2 [1Y]
##    Year Passengers
##   <dbl>      <dbl>
## 1  1970       7.32
## 2  1971       7.33
## 3  1972       7.80
## 4  1973       9.38
## 5  1974      10.7
## 6  1975      11.1
```

```
aus_airpassengers %>% autoplot()
```

```
## Plot variable not specified, automatically selected `.vars = Passengers`
```




```
# Calculate lambda for Box-Cox transformation using the Guerrero method
lambda <- aus_airpassengers %>%
  features(Passengers, features = guerrero) %>%
  pull(lambda_guerrero)

# Plot Box-Cox transformed data
aus_airpassengers %>%
  autoplot(box_cox(Passengers, lambda)) +
  labs(title = paste0("Transformed Passenger Data with Lambda = ", round(lambda, 2)))
```

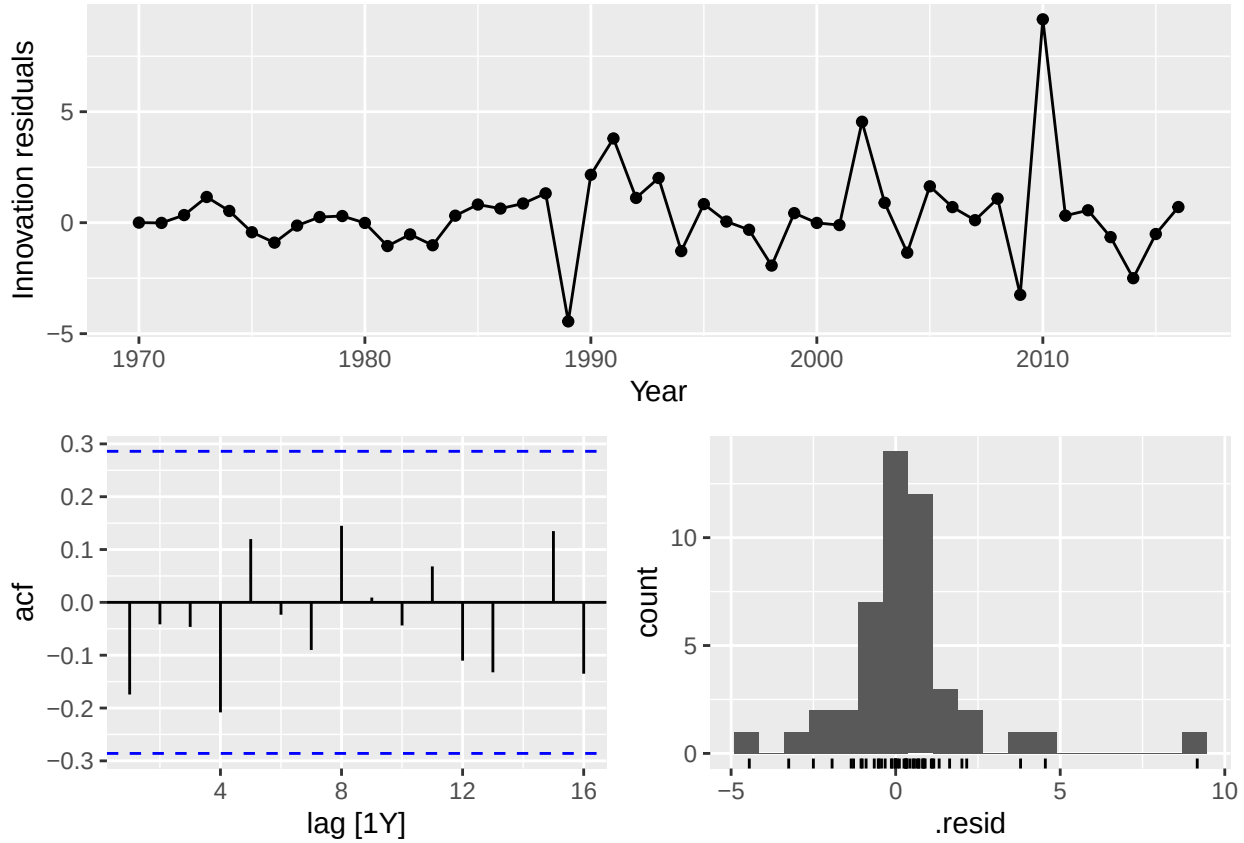


a. Use ARIMA() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

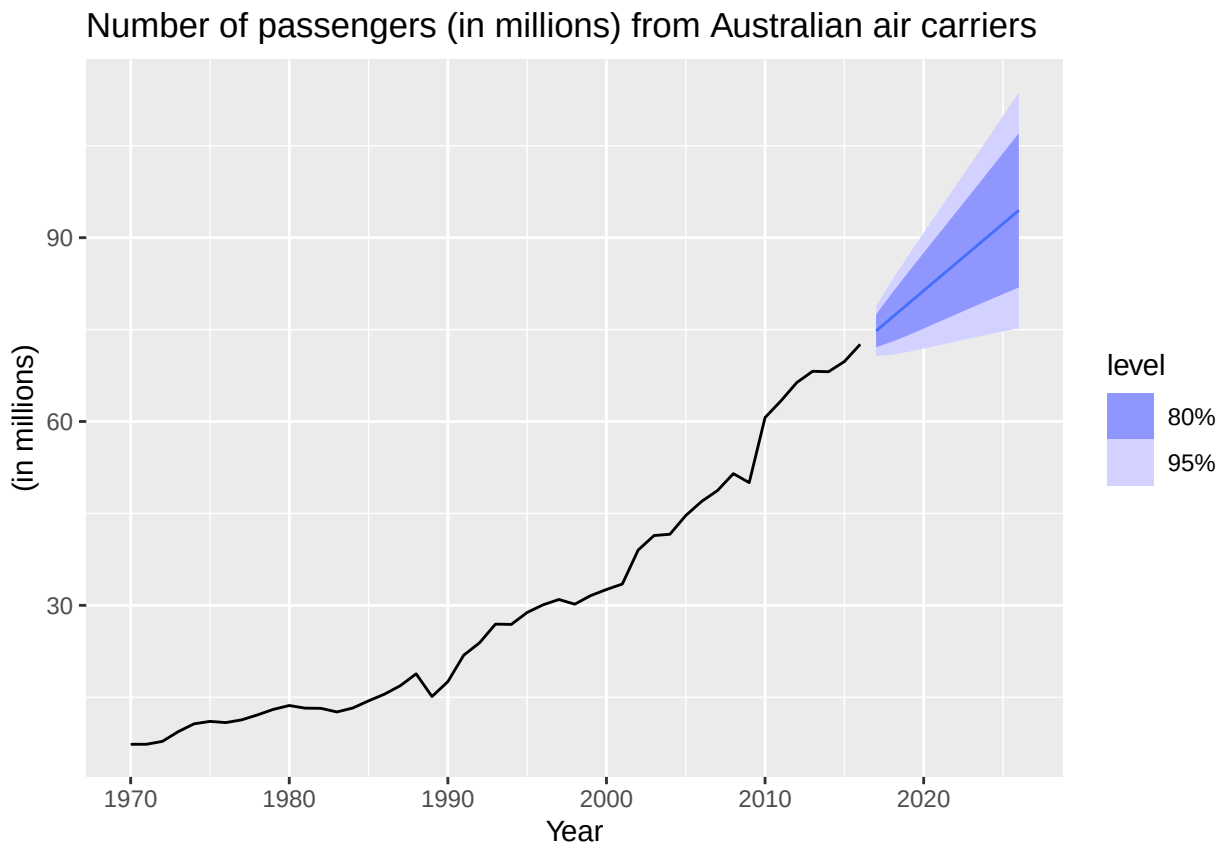
```
# Fit the ARIMA model
aus_air_mod <- aus_airpassengers %>% model(ARIMA(Passengers, stepwise = F))

# Report the selected model
report(aus_air_mod)
```

```
## 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
```

```r
# Plot residual diagnostics to check for white noise
aus_air_mod %>% gg_tsresiduals()
```




```r
# Forecast for the next 10 periods
aus_air_fc <- aus_air_mod %>% forecast(h=10)

# Plot the forecast
autoplot(aus_air_fc, aus_airpassengers) +
  labs(title="Number of passengers (in millions) from Australian air carriers", y="(in millions)")
```



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

Model: ARIMA(0,2,1). As the model has no p term, thus AR(0) and no constant, the model in terms of backshift operator is: (1-B)^2y(t)=(1+theta(1)B)epsilon(t)

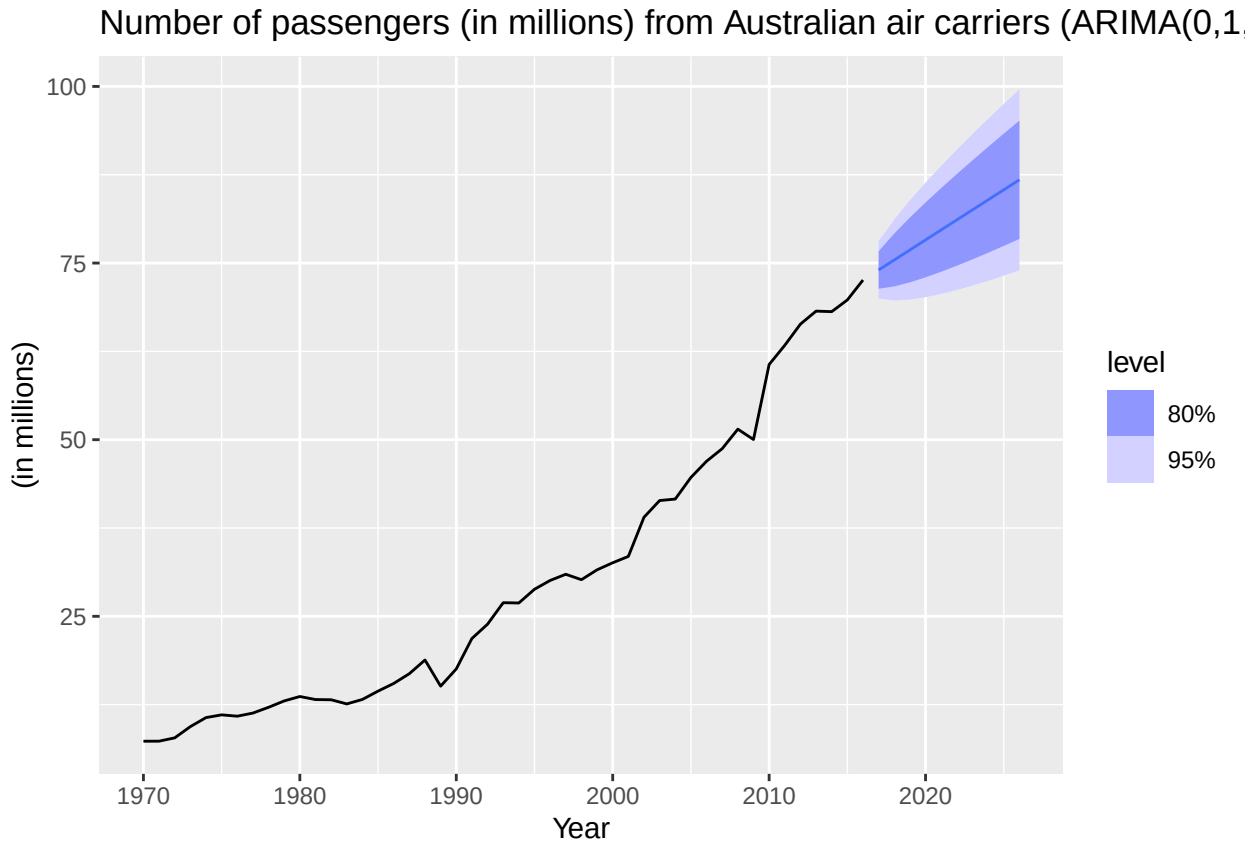
c. Plot forecasts from an ARIMA(0,1,0) model with drift and compare these to part a.

```
# Fit ARIMA(0,1,0) model with drift
aus_air_arima010 <- aus_airpassengers %>%
  model(ARIMA(Passengers ~ pdq(0,1,0)))

# Report the model
report(aus_air_arima010)
```

```
## Series: Passengers
## Model: ARIMA(0,1,0) w/ drift
##
## Coefficients:
##       constant
##         1.4191
## s.e.    0.3014
```

```
##
## sigma^2 estimated as 4.271:  log likelihood=-98.16
## AIC=200.31   AICc=200.59   BIC=203.97
```

```
# Forecast for the next 10 periods
aus_air_fc_010 <- aus_air_arima010 %>% forecast(h=10)

# Plot the forecast
autoplot(aus_air_fc_010, aus_airpassengers) +
  labs(title="Number of passengers (in millions) from Australian air carriers (ARIMA(0,1,0))", y="(in mi
```



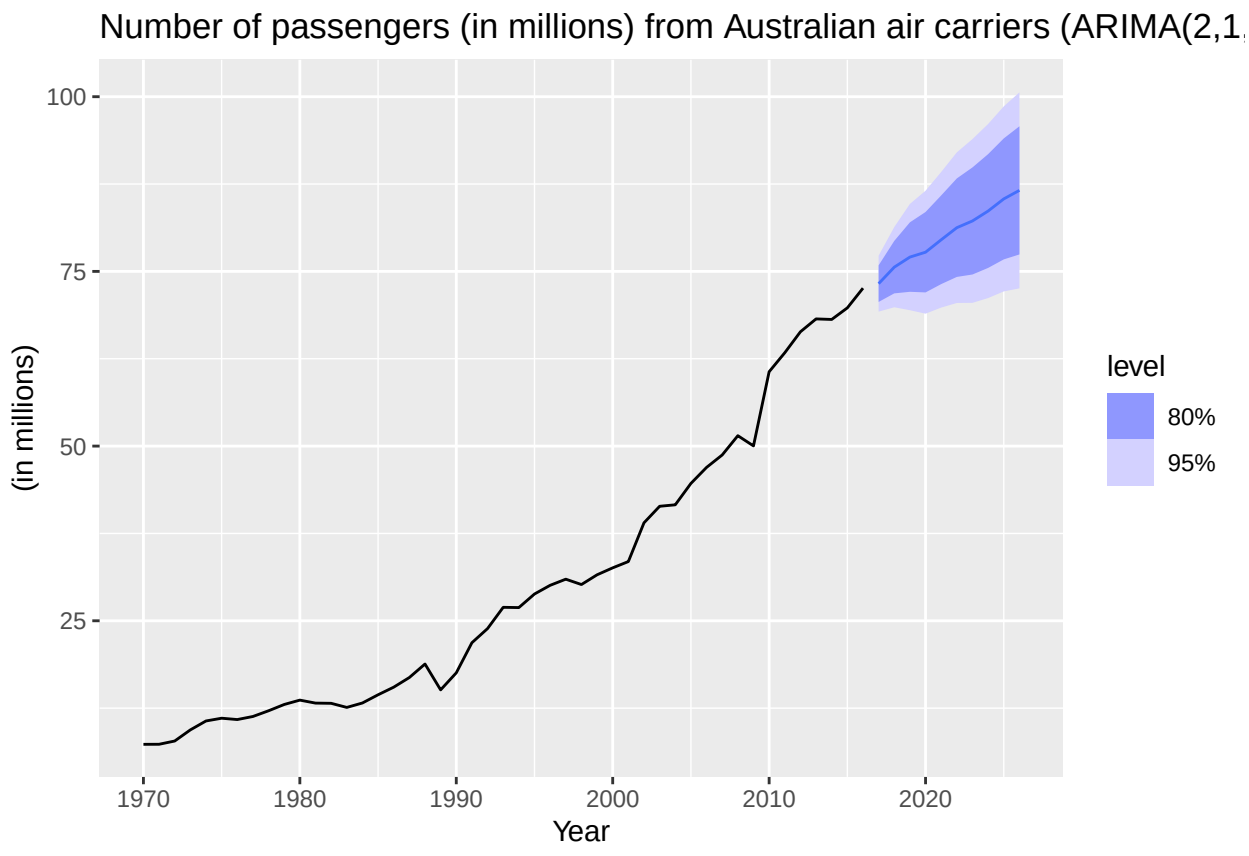
d. Plot forecasts from an ARIMA(2,1,2) model with drift and compare these to parts a and c. Remove the constant and see what happens.

```
# Fit ARIMA(2,1,2) with drift
aus_air_arima212 <- aus_airpassengers %>%
  model(ARIMA(Passengers ~ 1 + pdq(2,1,2)))

# Report the model
report(aus_air_arima212)
```

```
## Series: Passengers
## Model: ARIMA(2,1,2) w/ drift
```

```
##
## Coefficients:
##           ar1      ar2     ma1     ma2  constant
##       -0.5518  -0.7327  0.5895  1.0000    3.2216
## s.e.   0.1684   0.1224  0.0916  0.1008    0.7224
##
## sigma^2 estimated as 4.031:  log likelihood=-96.23
## AIC=204.46   AICc=206.61   BIC=215.43
```

```r
# Forecast for the next 10 periods
aus_air_fc_212 <- aus_air_arima212 %>% forecast(h=10)

# Plot the forecast
autoplot(aus_air_fc_212, aus_airpassengers) +
  labs(title="Number of passengers (in millions) from Australian air carriers (ARIMA(2,1,2))", y="(in mi
```




```r
# Fit ARIMA(2,1,2) with no constant (exclude constant)
aus_air_arima212_noCon <- aus_airpassengers %>%
  model(ARIMA(Passengers ~ 0 + pdq(2,1,2) + PDQ(0,0,0)))
```

```
## Warning: 1 error encountered for ARIMA(Passengers ~ 0 + pdq(2, 1, 2) + PDQ(0, 0, 0))
## [1] non-stationary AR part from CSS
```

```r
# Report the model
report(aus_air_arima212_noCon)
```

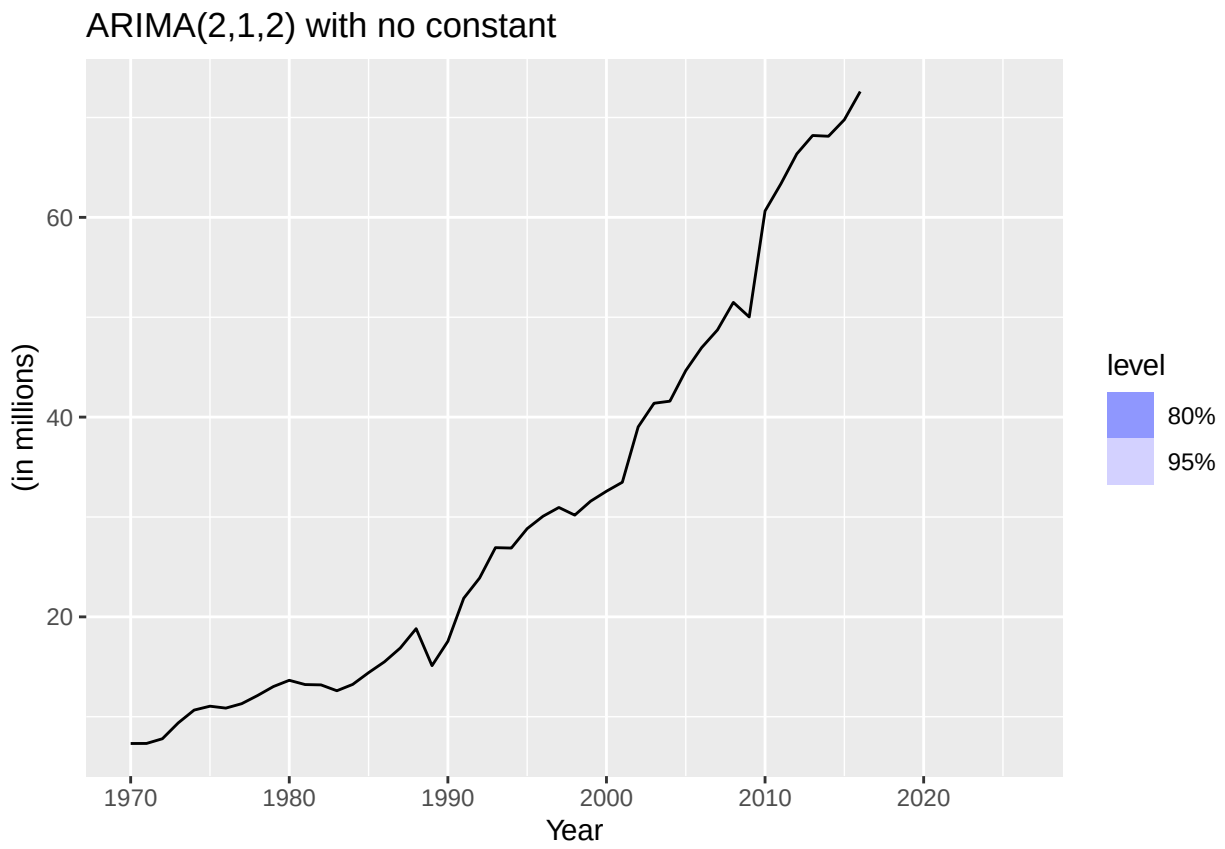
```
## Series: Passengers
## Model: NULL model
## NULL model
```

```r
# Forecast for the next 10 periods
aus_air_fc_212_noCon <- aus_air_arima212_noCon %>% forecast(h=10)

# Plot the forecast
autoplot(aus_air_fc_212_noCon, aus_airpassengers) +
  labs(title="ARIMA(2,1,2) with no constant", y="(in millions)")
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

```
## Warning in max(ids, na.rm = TRUE): no non-missing arguments to max; returning
## -Inf
```

```
## Warning: Removed 10 rows containing missing values or values outside the scale range
## (`geom_line()`).
```



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

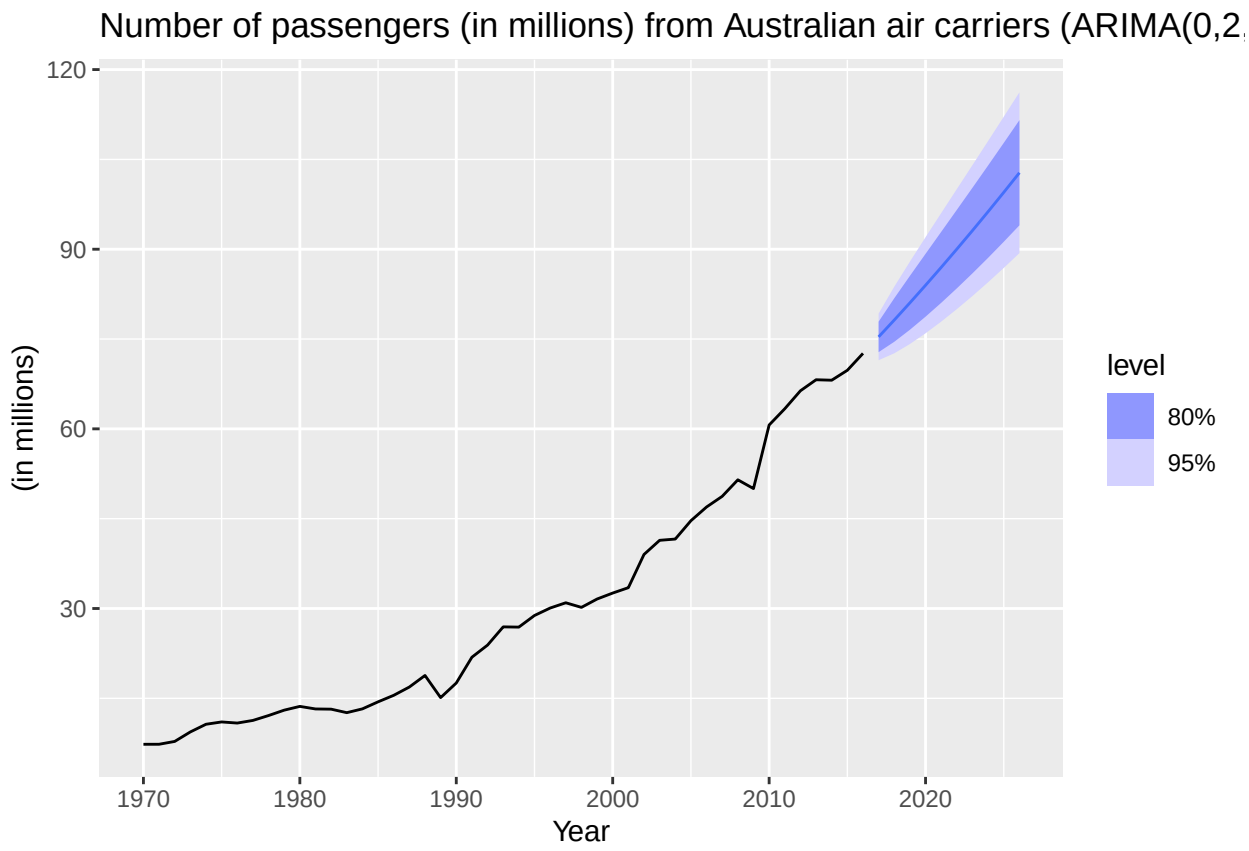
```
# Fit ARIMA(0,2,1) with a constant
aus_air_arima021 <- aus_airpassengers %>%
  model(ARIMA(Passengers ~ 1 + pdq(0,2,1)))
```

```
## 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.
```

```
# Report the model
report(aus_air_arima021)
```

```
## Series: Passengers
## Model: ARIMA(0,2,1) w/ poly
##
## Coefficients:
##           ma1  constant
##       -1.0000    0.0571
## s.e.   0.0585    0.0213
##
## sigma^2 estimated as 3.855:  log likelihood=-95.1
## AIC=196.21   AICc=196.79   BIC=201.63
```

```
# Forecast for the next 10 periods
aus_air_fc_021 <- aus_air_arima021 %>% forecast(h=10)

# Plot the forecast
autoplot(aus_air_fc_021, aus_airpassengers) +
  labs(title="Number of passengers (in millions) from Australian air carriers (ARIMA(0,2,1))", y="(in mi
```

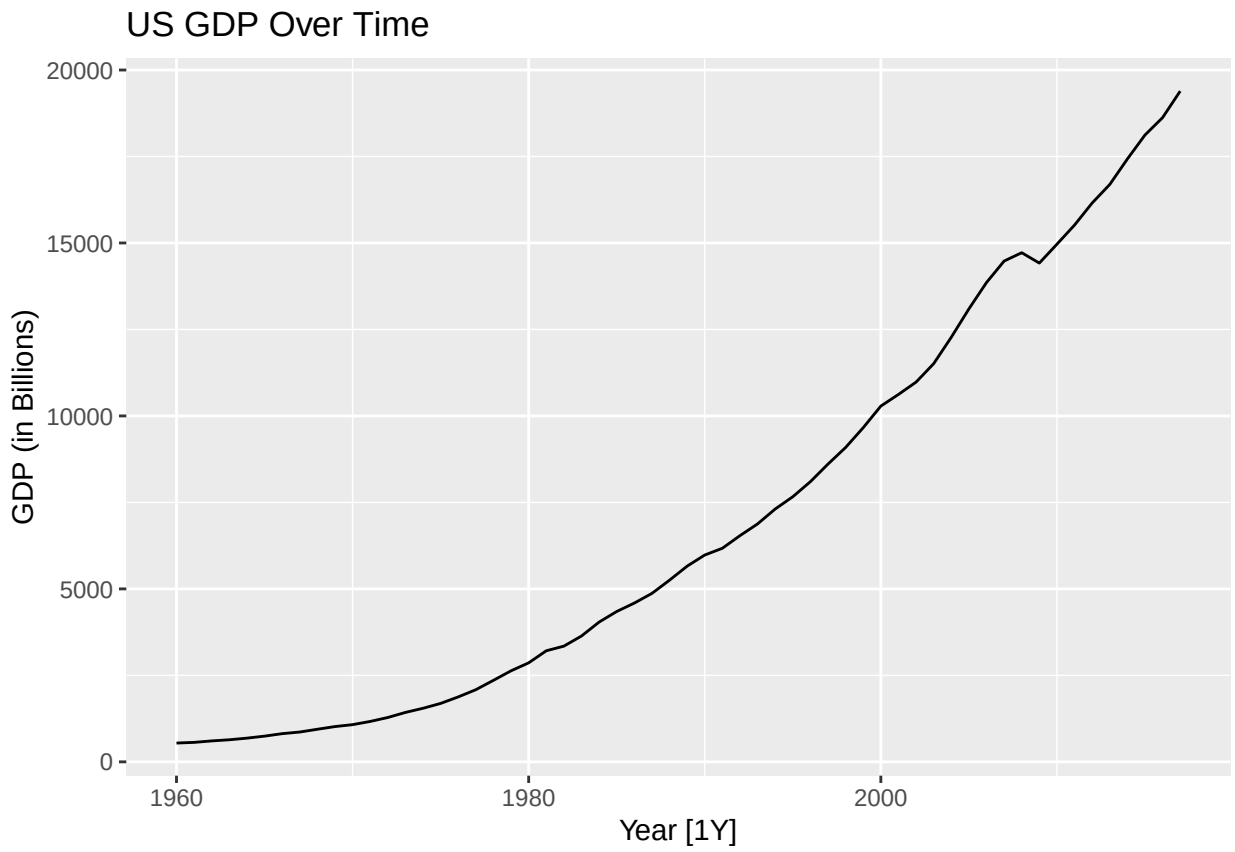


9.8 For the United States GDP series (from global_economy):

```
# Extract US GDP data and convert it to billions
us_gdp_data <- global_economy %>%
  filter(Country == "United States") %>%
  mutate(GDP_billions = GDP / 1e9)

# Display the first few rows of the data
head(us_gdp_data)
```

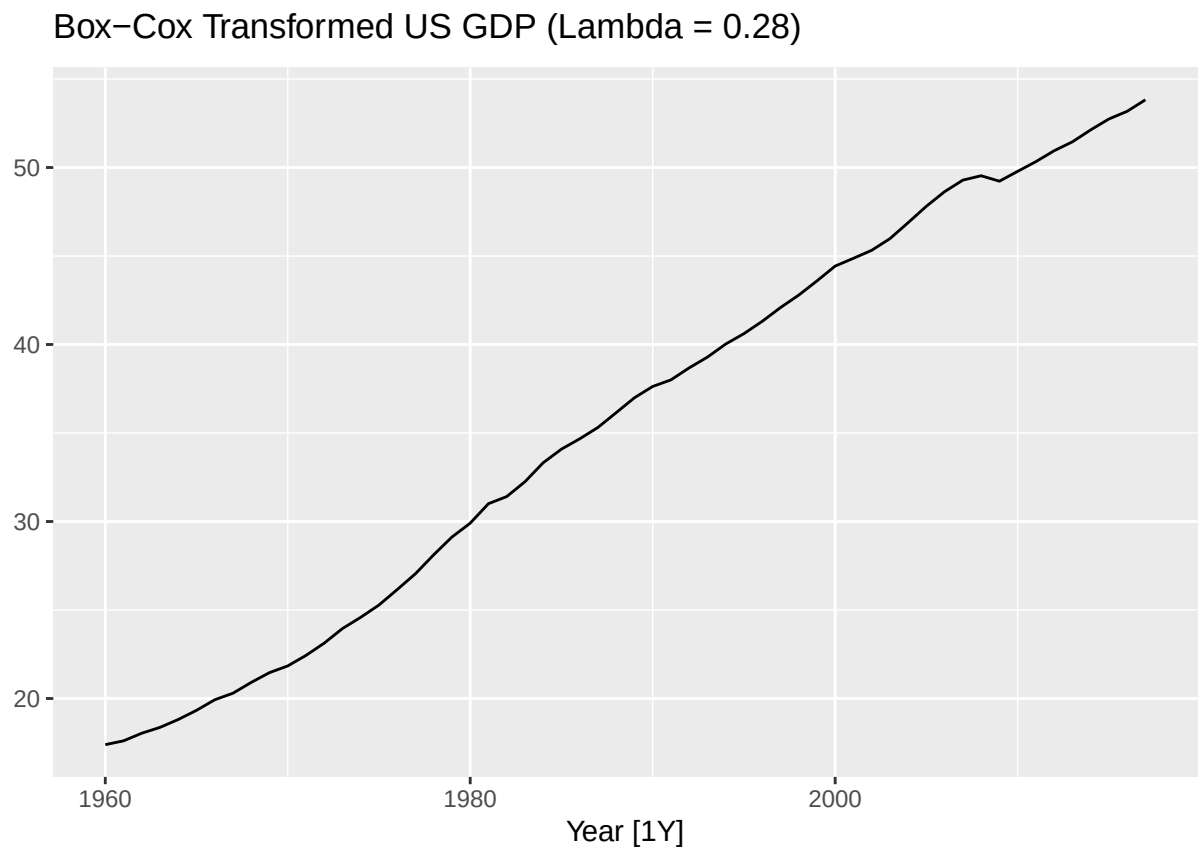
```
## # A tsibble: 6 x 10 [1Y]
## # Key:       Country [1]
##   Country       Code   Year          GDP Growth   CPI Imports Exports Population
##   <fct>         <fct> <dbl>        <dbl>  <dbl> <dbl>   <dbl>   <dbl>      <dbl>
## 1 United States USA    1960 543300000000  NA     13.6    4.20    4.97  180671000
## 2 United States USA    1961 563300000000   2.30  13.7    4.03    4.90  183691000
## 3 United States USA    1962 605100000000   6.10  13.9    4.13    4.81  186538000
## 4 United States USA    1963 638600000000   4.40  14.0    4.09    4.87  189242000
## 5 United States USA    1964 685800000000   5.80  14.2    4.10    5.10  191889000
## 6 United States USA    1965 743700000000   6.40  14.4    4.24    4.99  194303000
## # i 1 more variable: GDP_billions <dbl>
```

```r
# Plot the original GDP data
us_gdp_data %>% autoplot(GDP_billions) +
  labs(title = "US GDP Over Time", y = "GDP (in Billions)")
```

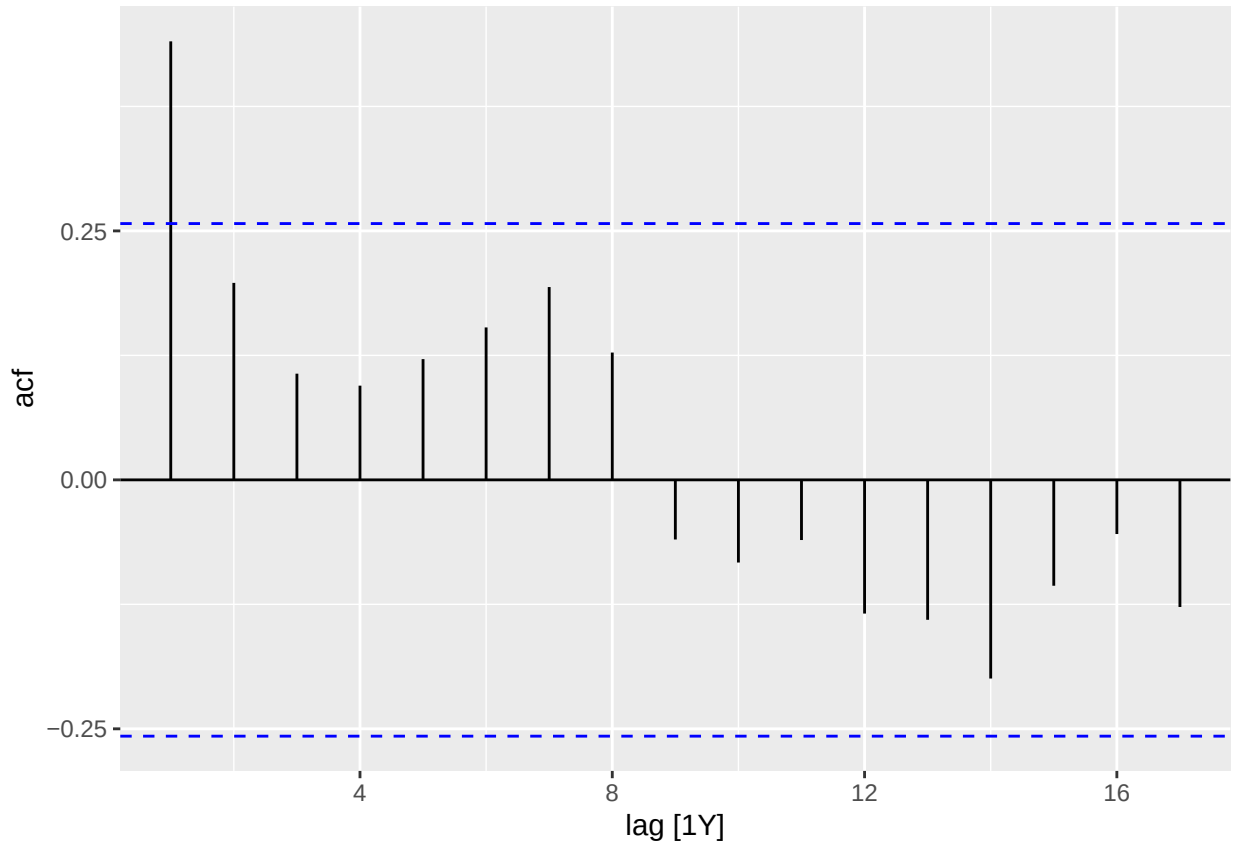


a. if necessary, find a suitable Box-Cox transformation for the data;

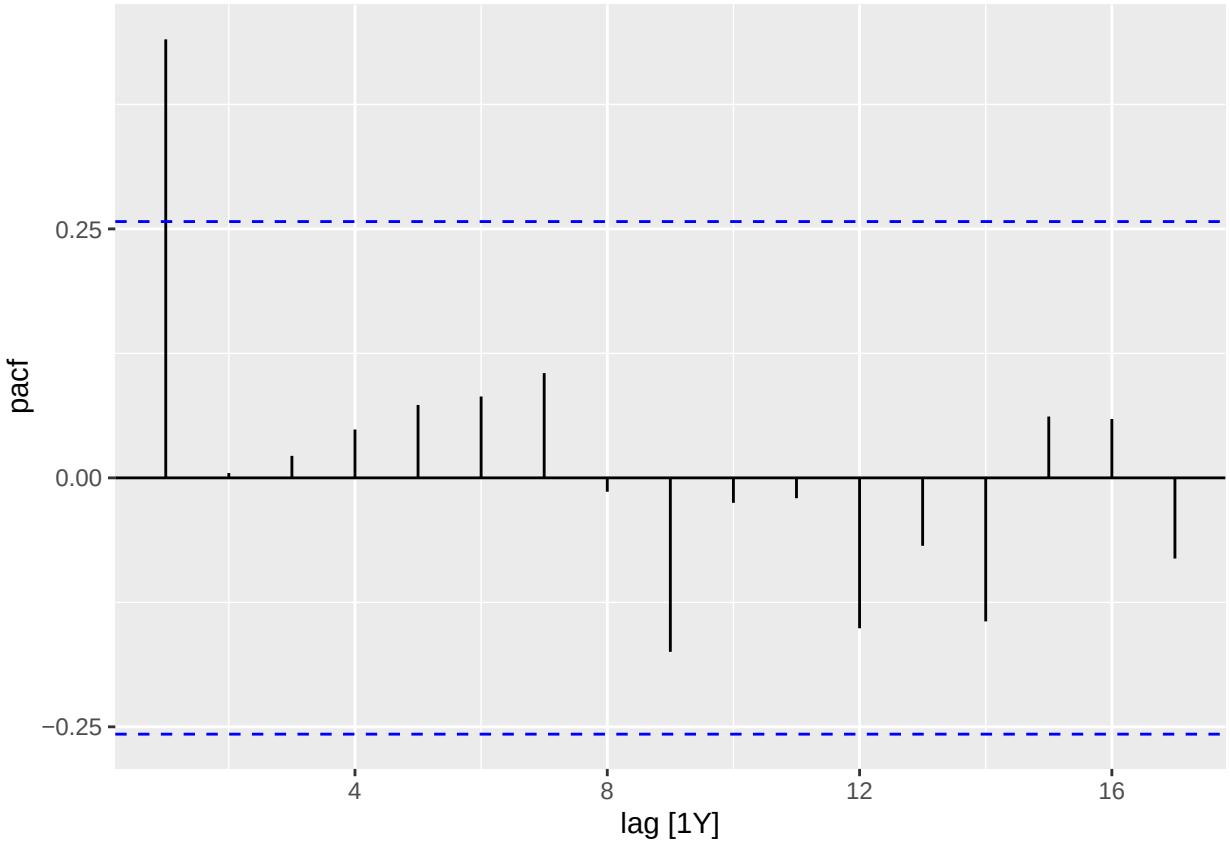
```r
# Find the optimal Box-Cox lambda using the guerrero method
lambda_bc <- us_gdp_data %>%
  features(GDP_billions, features = guerrero) %>%
  pull(lambda_guerrero)

# Apply the Box-Cox transformation to the GDP data
us_gdp_data %>%
  autoplot(box_cox(GDP_billions, lambda_bc)) +
  labs(y = "", title = latex2exp::TeX(paste0(
    "Box-Cox Transformed US GDP (Lambda = ", round(lambda_bc, 2), ")")))
```




```
# Store the transformed GDP data in a new column
us_gdp_data <- us_gdp_data %>%
  mutate(GDP_transformed = box_cox(GDP_billions, lambda_bc))

# Calculate the first difference of the transformed GDP
us_gdp_data <- us_gdp_data %>%
  mutate(GDP_diff = difference(GDP_transformed))

# Plot ACF and PACF for differenced GDP
us_gdp_data %>%
  ACF(GDP_diff) %>%
  autoplot()
```




```
us_gdp_data %>%
  PACF(GDP_diff) %>%
  autoplot()
```



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

```
# Fit an automatic ARIMA model to the transformed GDP data
gdp_arima_model <- us_gdp_data %>%
  model(ARIMA(GDP_transformed))

# Display the model summary
report(gdp_arima_model)
```

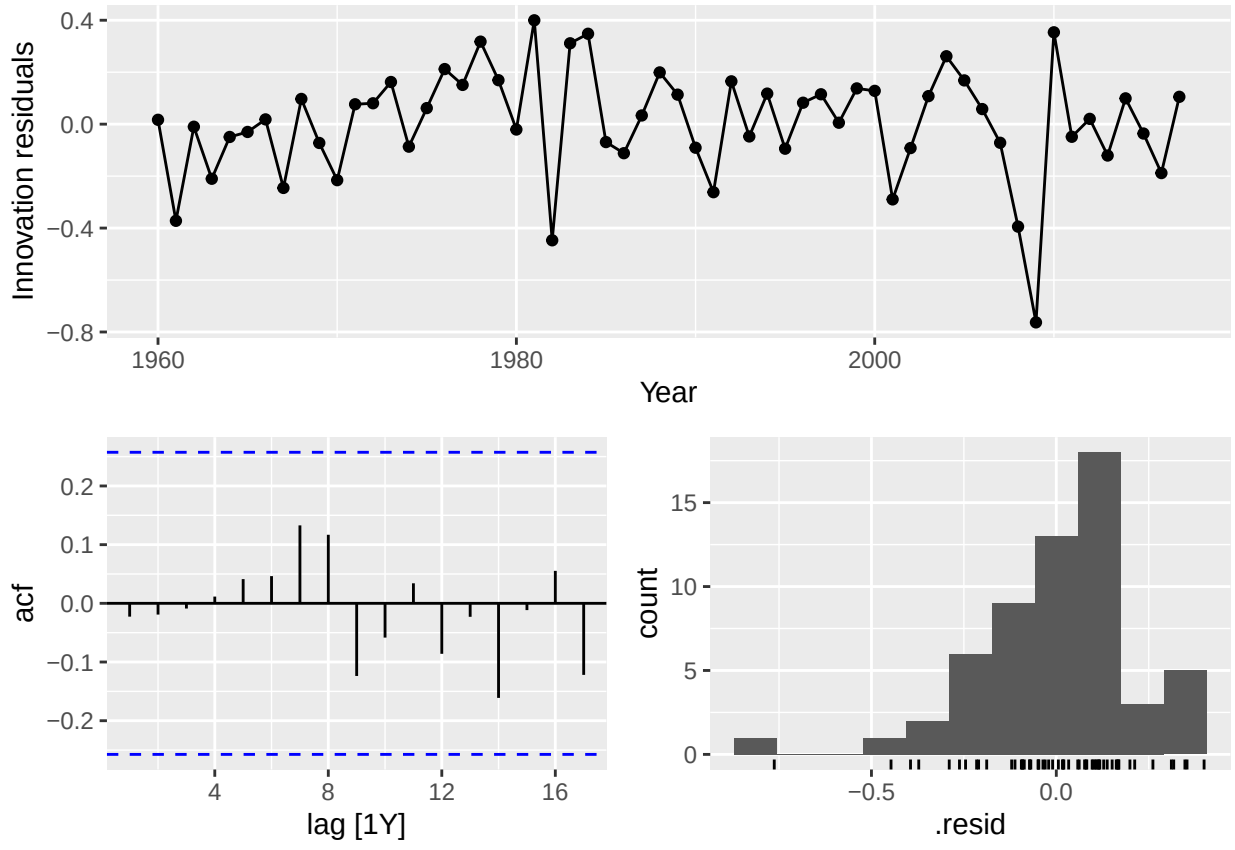
```
## Series: GDP_transformed
## Model: ARIMA(1,1,0) w/ drift
##
## Coefficients:
##          ar1  constant
##       0.4586    0.3428
## s.e.  0.1198    0.0276
##
## sigma^2 estimated as 0.0461:  log likelihood=7.72
## AIC=-9.43   AICc=-8.98   BIC=-3.3
```

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

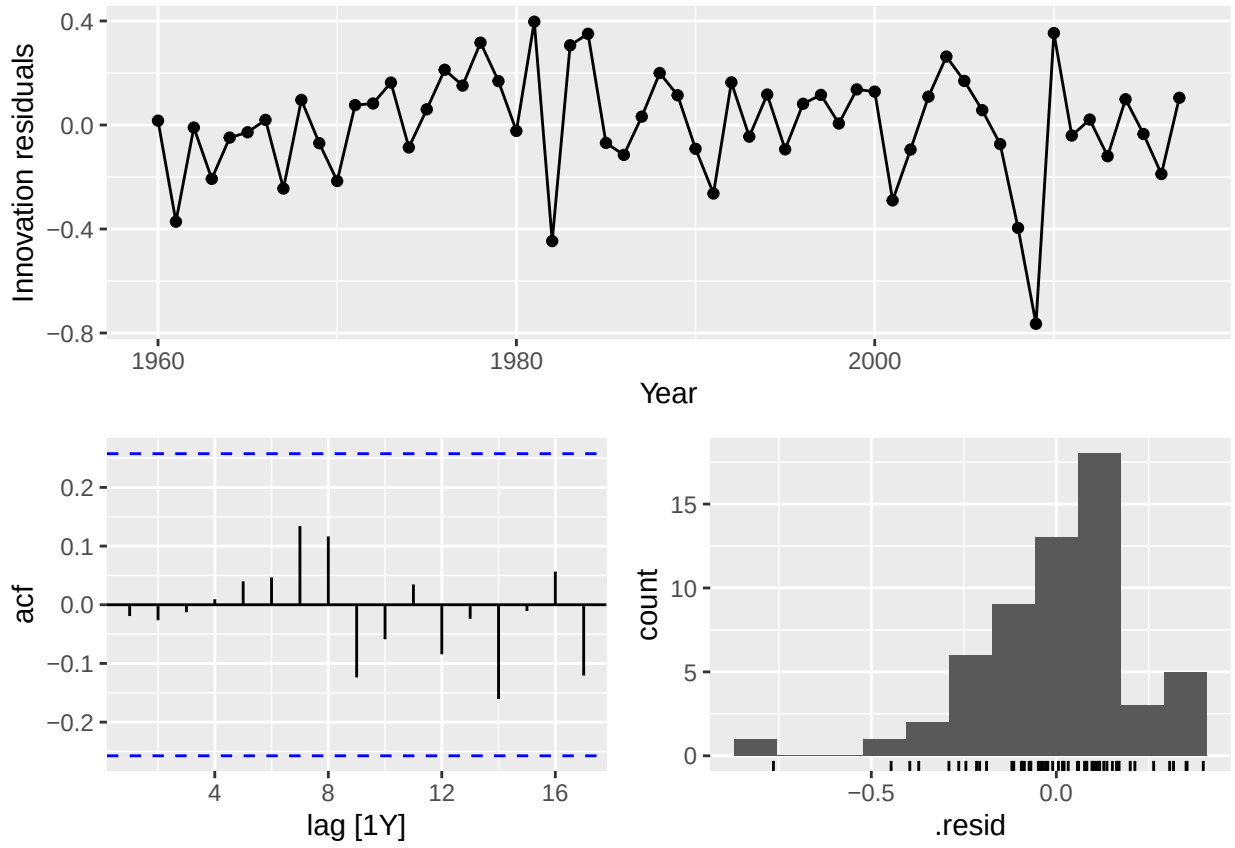
```
# Fit various ARIMA models with different pdq combinations
arima_111 <- us_gdp_data %>%
  model(ARIMA(GDP_transformed ~ pdq(1,1,1)))

arima_211 <- us_gdp_data %>%
  model(ARIMA(GDP_transformed ~ pdq(2,1,1)))

arima_112 <- us_gdp_data %>%
  model(ARIMA(GDP_transformed ~ pdq(1,1,2)))

arima_210 <- us_gdp_data %>%
  model(ARIMA(GDP_transformed ~ pdq(2,1,0)))
```

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

```
# Check residuals of the initial ARIMA model
gdp_arima_model %>% gg_tsresiduals()
```



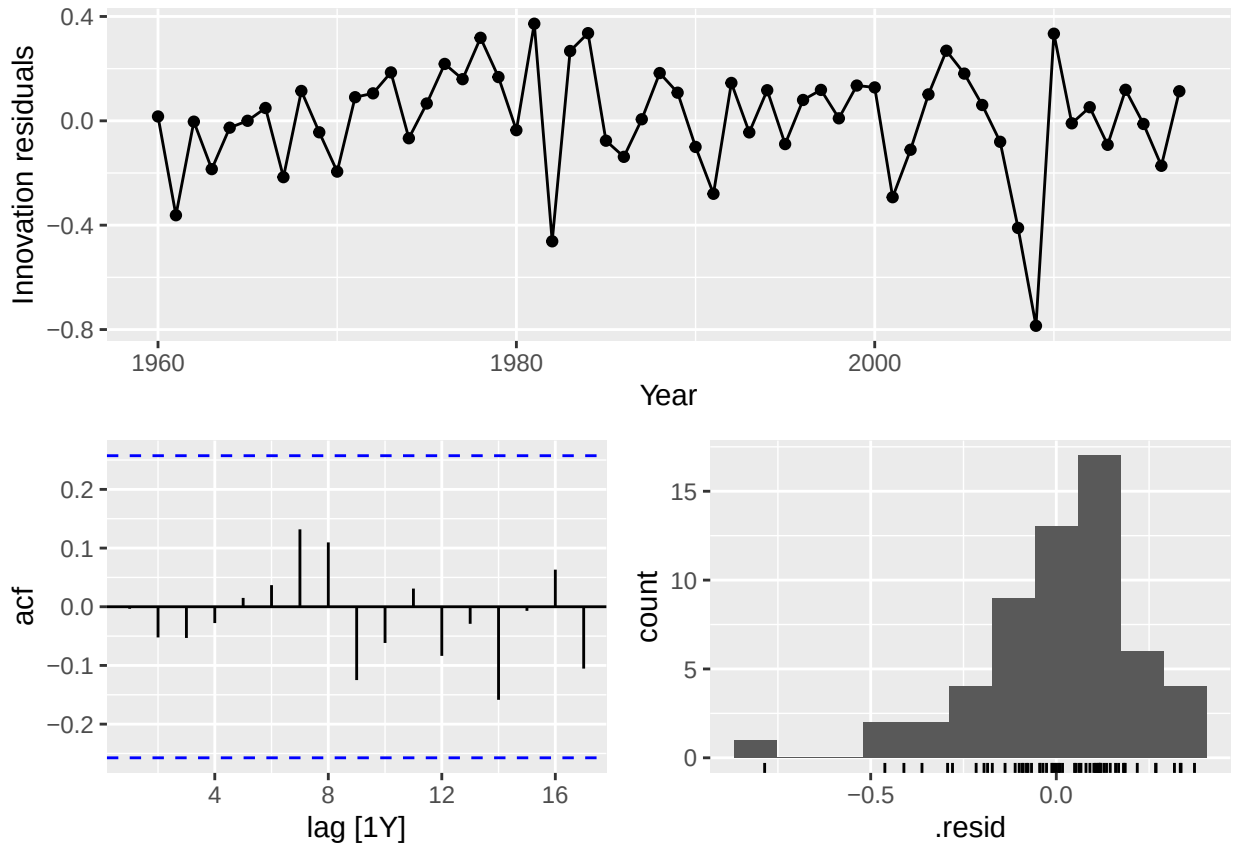

```
# Check residuals of alternative models
arima_111 %>% gg_tsresiduals()
```




```
report(arima_111)
```

```
## Series: GDP_transformed
## Model: ARIMA(1,1,1) w/ drift
##
## Coefficients:
##          ar1      ma1  constant
##       0.4736  -0.0189    0.3332
## s.e.  0.2851   0.3286    0.0271
##
## sigma^2 estimated as 0.04695:  log likelihood=7.72
## AIC=-7.44   AICc=-6.67   BIC=0.74
```

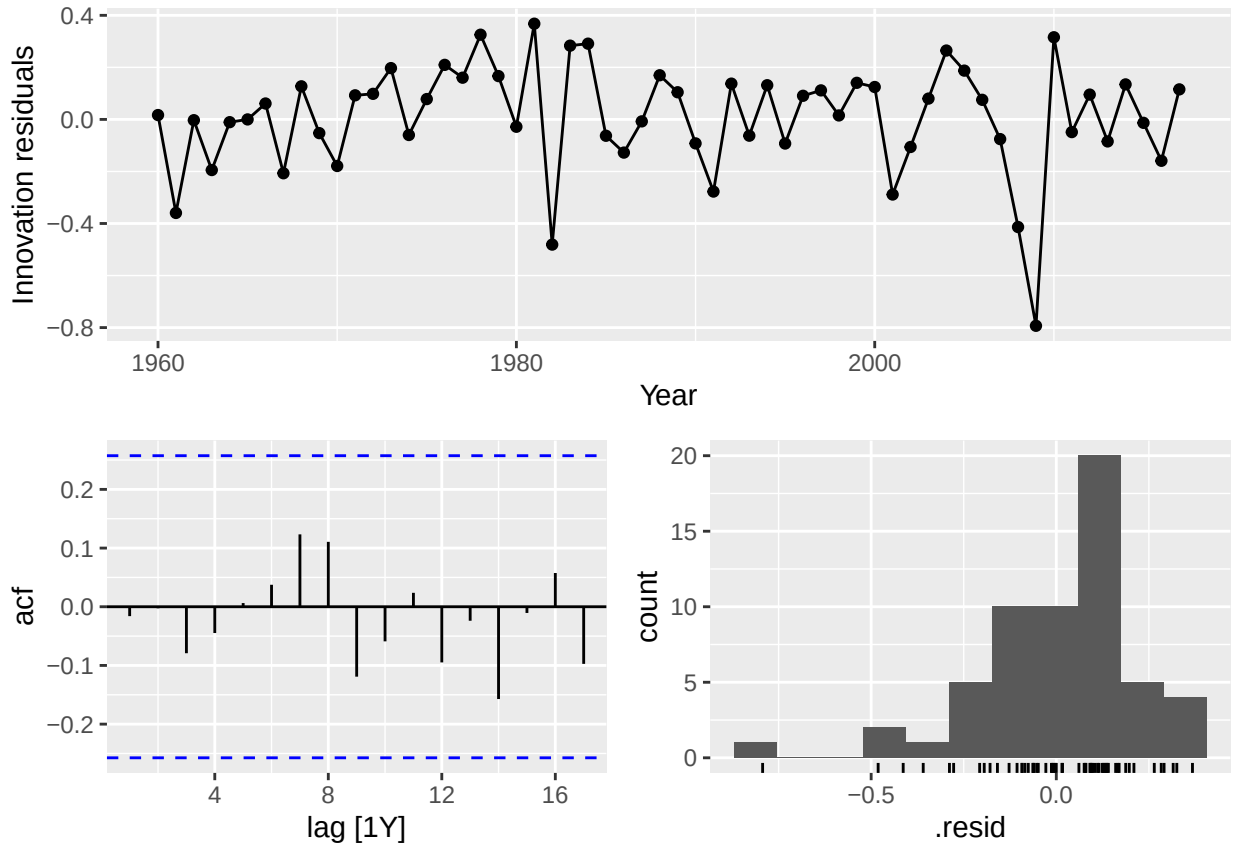
```
arima_211 %>% gg_tsresiduals()
```




```
report(arima_211)
```

```
## Series: GDP_transformed
## Model: ARIMA(2,1,1) w/ drift
##
## Coefficients:
##          ar1      ar2      ma1  constant
##       1.1662  -0.2792  -0.7357    0.0706
## s.e.  0.3418   0.2108   0.3077    0.0074
##
## sigma^2 estimated as 0.04751:  log likelihood=7.9
## AIC=-5.79   AICc=-4.62   BIC=4.42
```

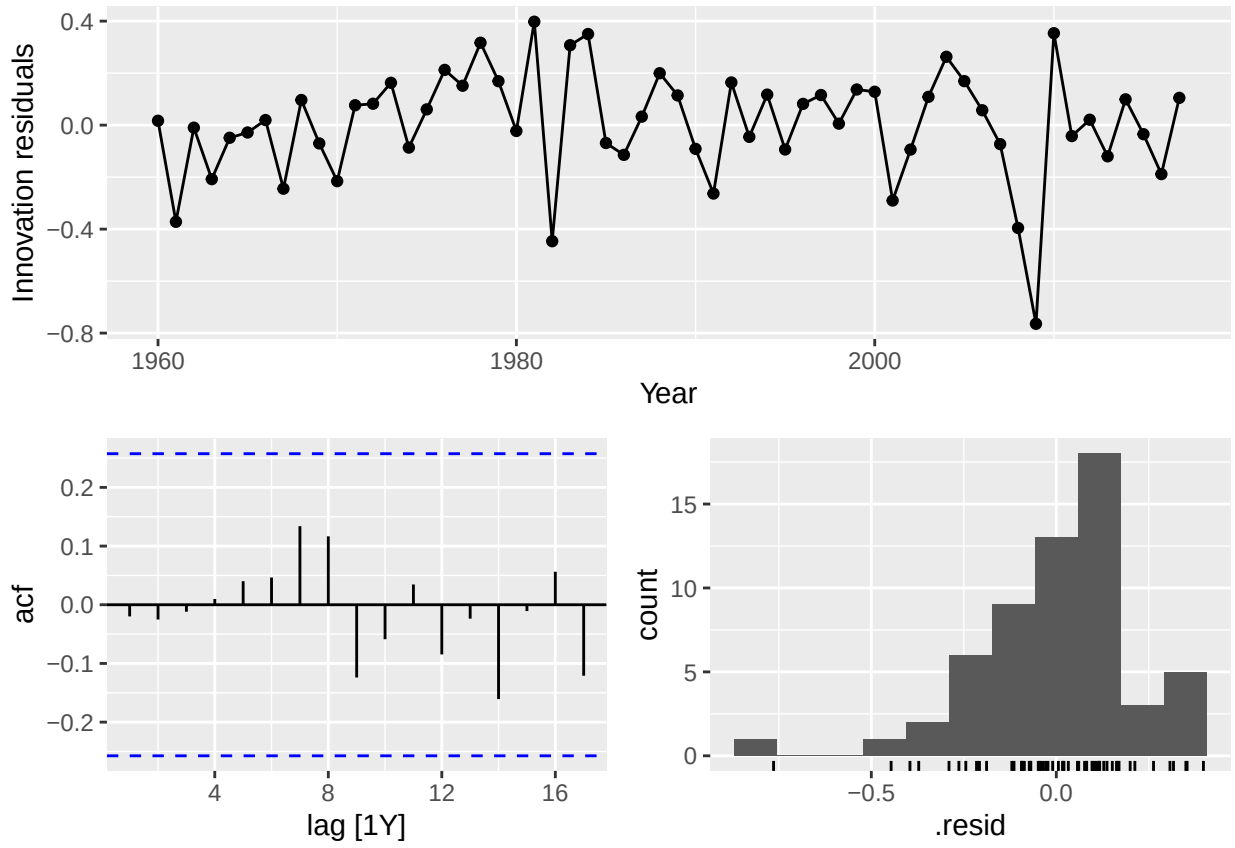
```
arima_112 %>% gg_tsresiduals()
```




```
report(arima_112)
```

```
## Series: GDP_transformed
## Model: ARIMA(1,1,2) w/ drift
##
## Coefficients:
##          ar1      ma1      ma2  constant
##       0.8284  -0.3879  -0.1931    0.1068
## s.e.  0.2060   0.2501   0.1592    0.0117
##
## sigma^2 estimated as 0.04737:  log likelihood=7.97
## AIC=-5.94   AICc=-4.77   BIC=4.27
```

```
arima_210 %>% gg_tsresiduals()
```

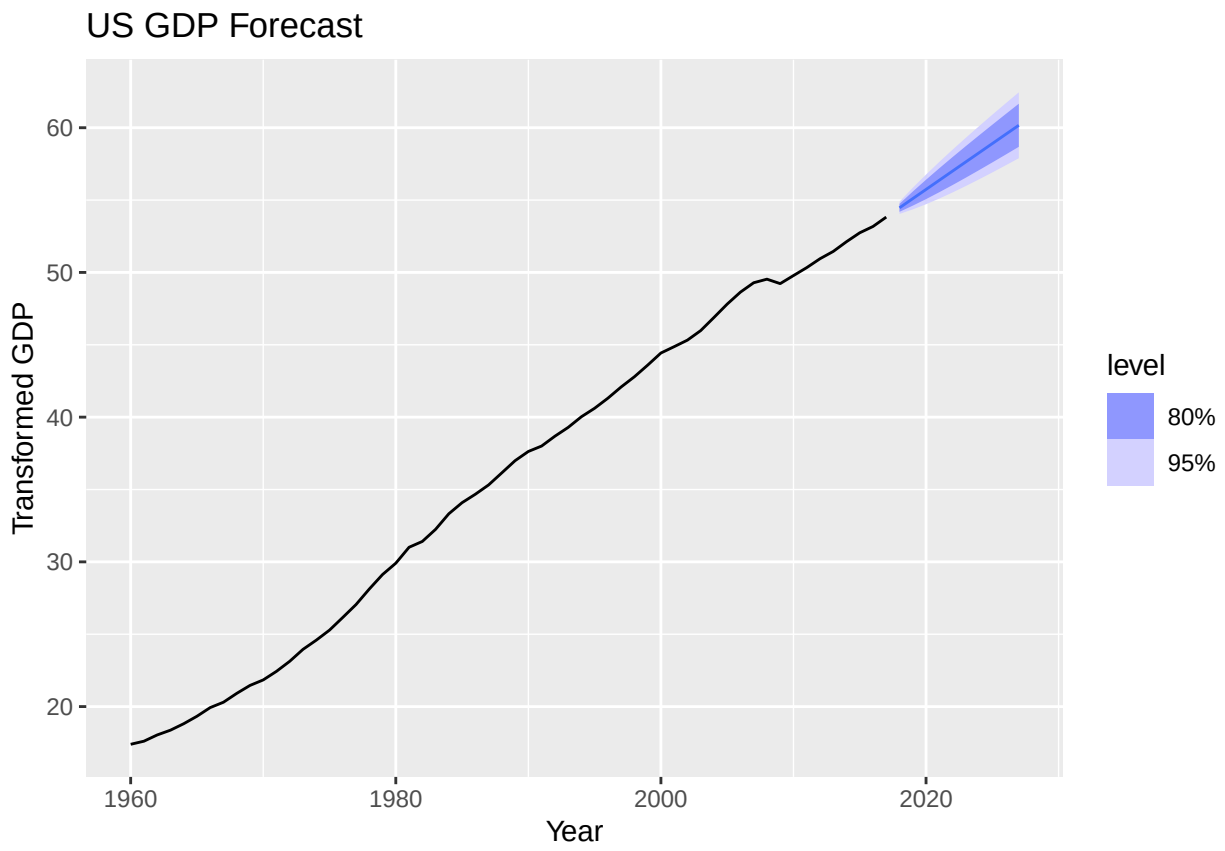



```
report(arima_210)
```

```
## Series: GDP_transformed
## Model: ARIMA(2,1,0) w/ drift
##
## Coefficients:
##          ar1     ar2  constant
##       0.4554  0.0071    0.3402
## s.e.  0.1341  0.1352    0.0276
##
## sigma^2 estimated as 0.04695:  log likelihood=7.72
## AIC=-7.44   AICc=-6.67   BIC=0.74
```

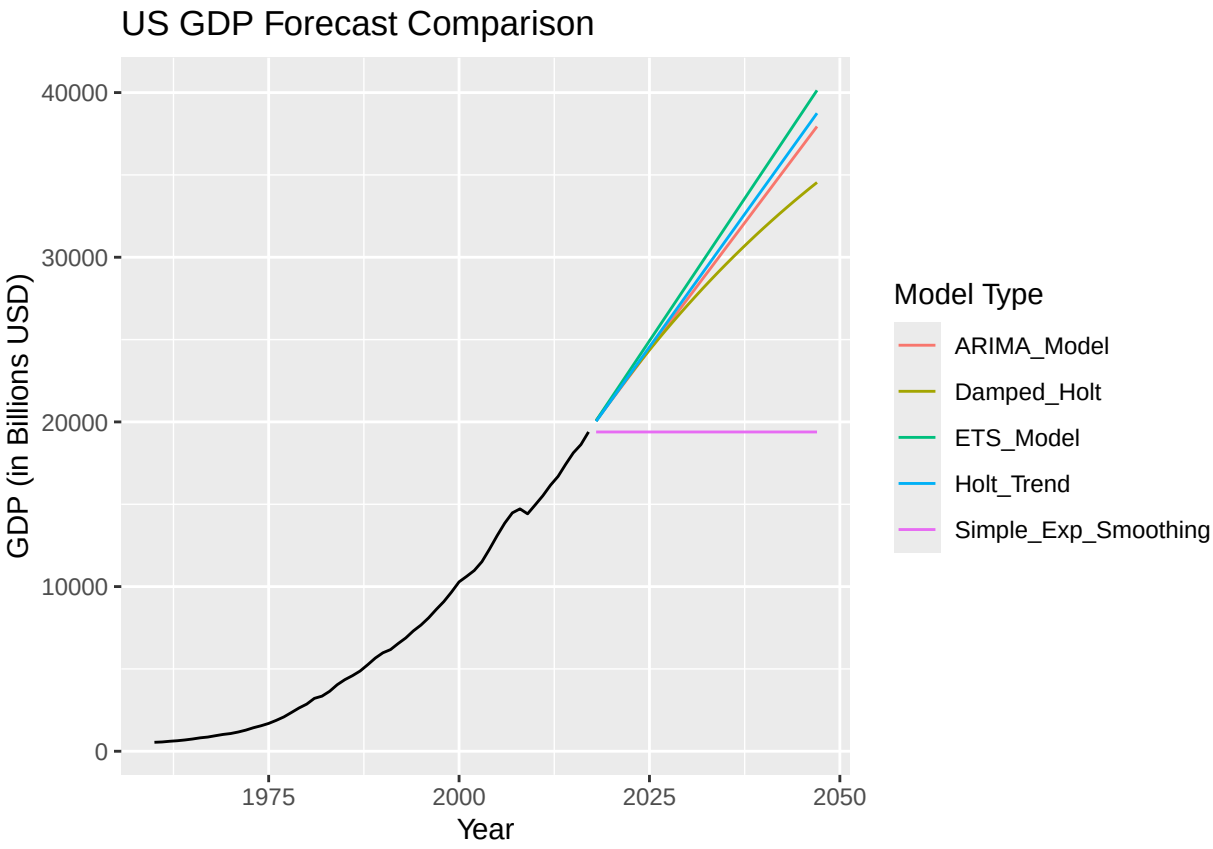
e. produce forecasts of your fitted model. Do the forecasts look reasonable?

```
# Forecast the next 10 periods using the selected model
gdp_forecast <- gdp_arima_model %>% forecast(h = 10)

# Plot the forecast with the original data
gdp_forecast %>%
  autoplot(us_gdp_data) +
  labs(title = "US GDP Forecast", y = "Transformed GDP")
```



f. compare the results with what you would obtain using ETS() (with no transformation).

```
# Fit multiple ETS models, including Simple Exponential Smoothing (SES), Holt, and Damped models
gdp_ets_models <- us_gdp_data %>%
  model(
    ETS_Model = ETS(GDP_billions),
    Simple_Exp_Smoothing = ETS(GDP_billions ~ error("A") + trend("N") + season("N")),
    Holt_Trend = ETS(GDP_billions ~ error("A") + trend("A") + season("N")),
    Damped_Holt = ETS(GDP_billions ~ error("A") + trend("Ad") + season("N")),
    ARIMA_Model = ARIMA(GDP_billions)
  )

# Forecast for the next 30 periods
gdp_forecast_ets <- gdp_ets_models %>% forecast(h = 30)

# Plot all the forecasts for comparison
gdp_forecast_ets %>%
  autoplot(us_gdp_data, level = NULL) +
  labs(y = "GDP (in Billions USD)", title = "US GDP Forecast Comparison") +
  guides(colour = guide_legend(title = "Model Type"))
```


US GDP Forecast Comparison
GDP (in Billions USD)
40000
30000
20000
10000
0
1975
2000
2025
2050
Year
Model Type
ARIMA_Model
Damped_Holt
ETS_Model
Holt_Trend
Simple_Exp_Smoothing