Week 8 ARIMA

Chi Hang(Philip) Cheung

2025-03-16

```
library(fpp3)
```

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

```
## -- Attaching packages --------------------------------------------- fpp3 1.0.1 --
```

```
## v tibble      3.2.1     v tsibble     1.1.6
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.4.1
## v lubridate   1.9.4     v fable       0.4.1
## v ggplot2     3.5.1
```

```
## -- 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()
```

```
library(gridExtra)
```

```
## Warning: package 'gridExtra' was built under R version 4.4.3
```

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

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

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?

Ans: These three plots differ in their number of lag days, meaning the numbers of days away from the present date. All of the plots are showing spikes within the bluelines, indicating that residuals do not have significant autocorrelations and therefore white-noises.

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?

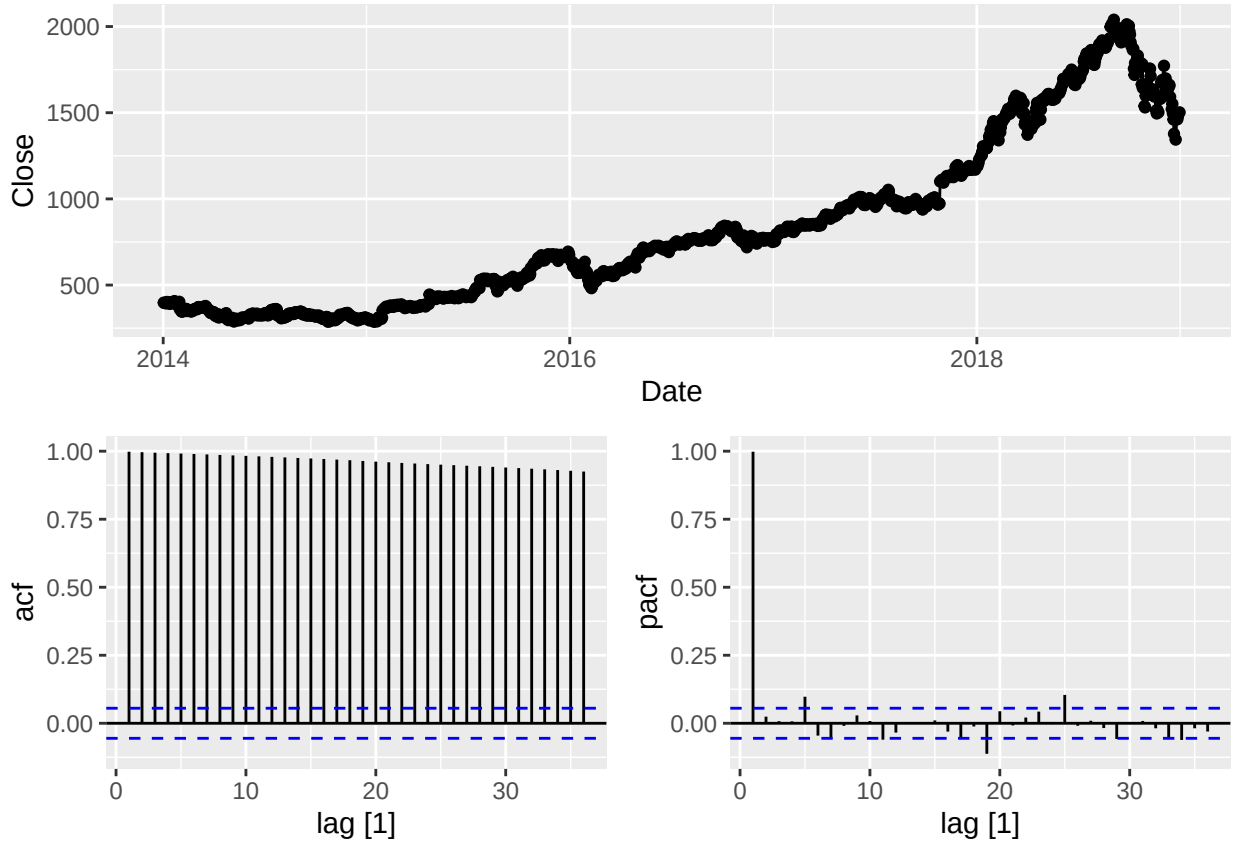
Ans: The critical values are at different distances from the mean of zero due to sample sizes. The critical values are typically narrower, or more precise, as sample size increases. The same is true for autocorrelations in the figures. As sample size increases, the autocorrelation flunctions stablizes or becomes more evened out and therefore a better precision in measuring the autocorrelations with large data.

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.

Ans: As seen in the ACF plot, the spikes show an exponential decay as lag increases, indicating a high autocorrelation in the 'Close' stock price. The data is not stationary at all. KPSS ndiff function was used to check the number of differencing needed to convert the data into stationary. The result showed 1 differencing and it passed the KPSS test.

```
amzn<- gafa_stock %>%
  filter(Symbol == 'AMZN')

amzn %>%
  gg_tsdisplay(Close, lag = 36, plot_type = 'partial')
```

```
## 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
```



amzn differenced ACF and PACF:

```
#use KPSS to determine the number of differencing needed:
amzn %>% features(Close, unitroot_ndiffs) #1 differencing is needed
```

```
## # A tibble: 1 x 2
##   Symbol ndiffs
##   <chr>   <int>
## 1 AMZN        1
```

```
#check with KPSS after 1 differencing:
amzn %>% features(difference(Close), unitroot_kpss) #p-value ~0.1, data is stationary
```

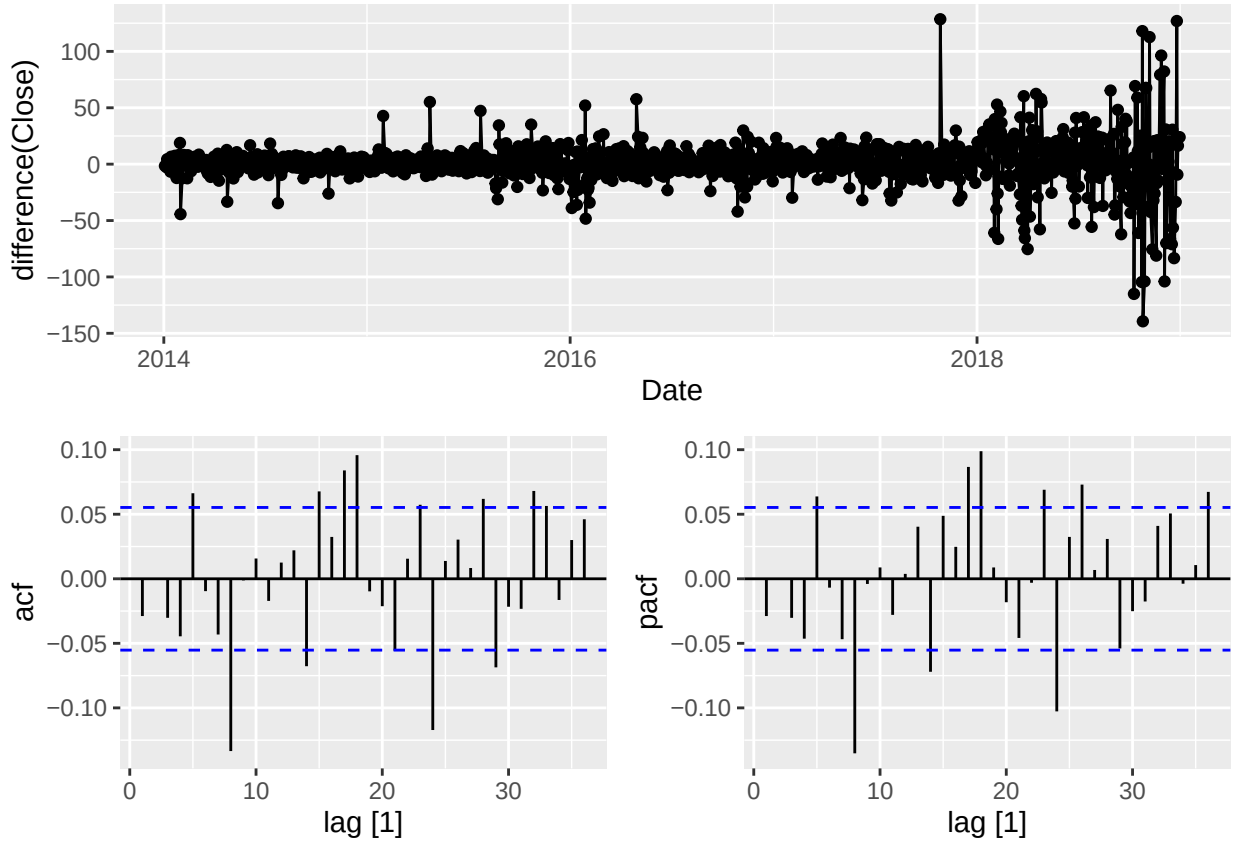
```
## # A tibble: 1 x 3
##   Symbol kpss_stat kpss_pvalue
##   <chr>      <dbl>       <dbl>
## 1 AMZN       0.149         0.1
```

```
amzn %>% gg_tsdisplay(difference(Close),lag = 36, plot_type = 'partial')
```

```
## 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
```

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

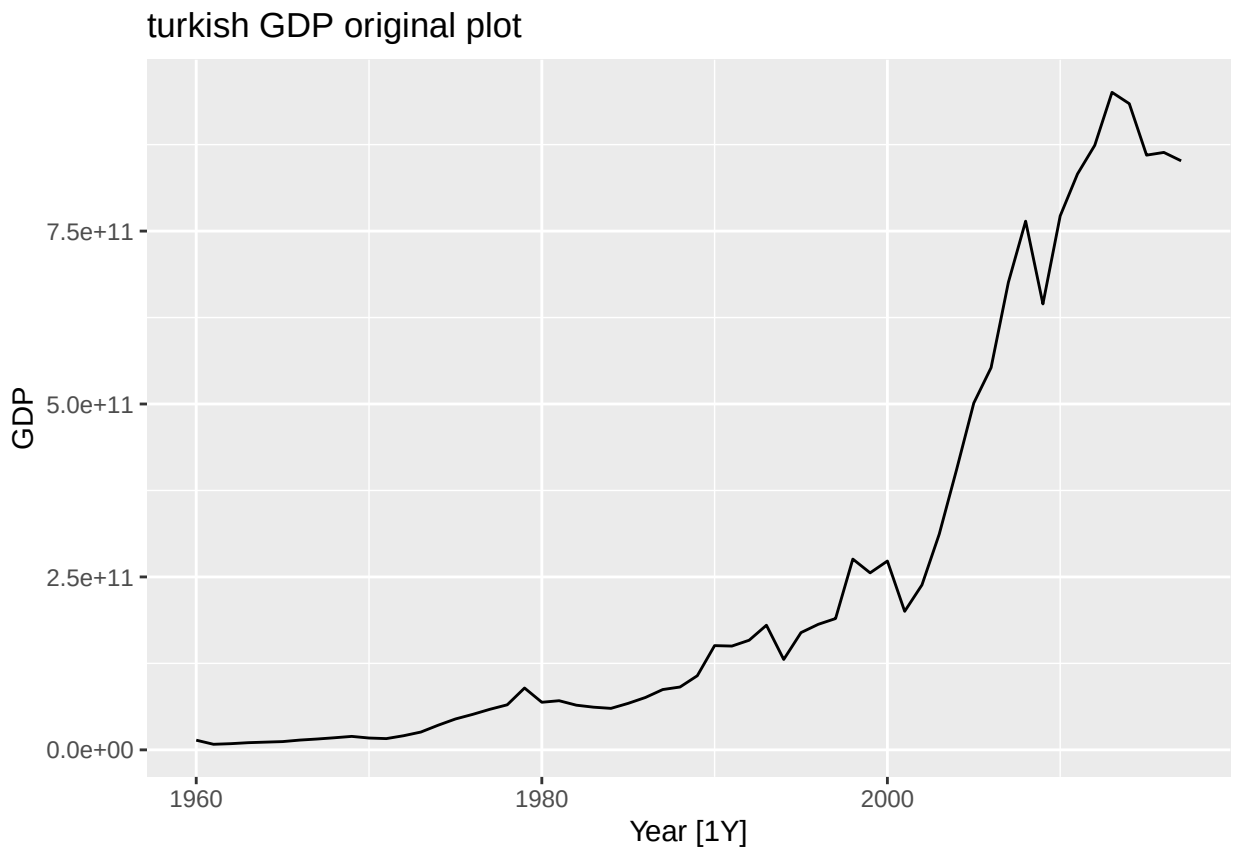
```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



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

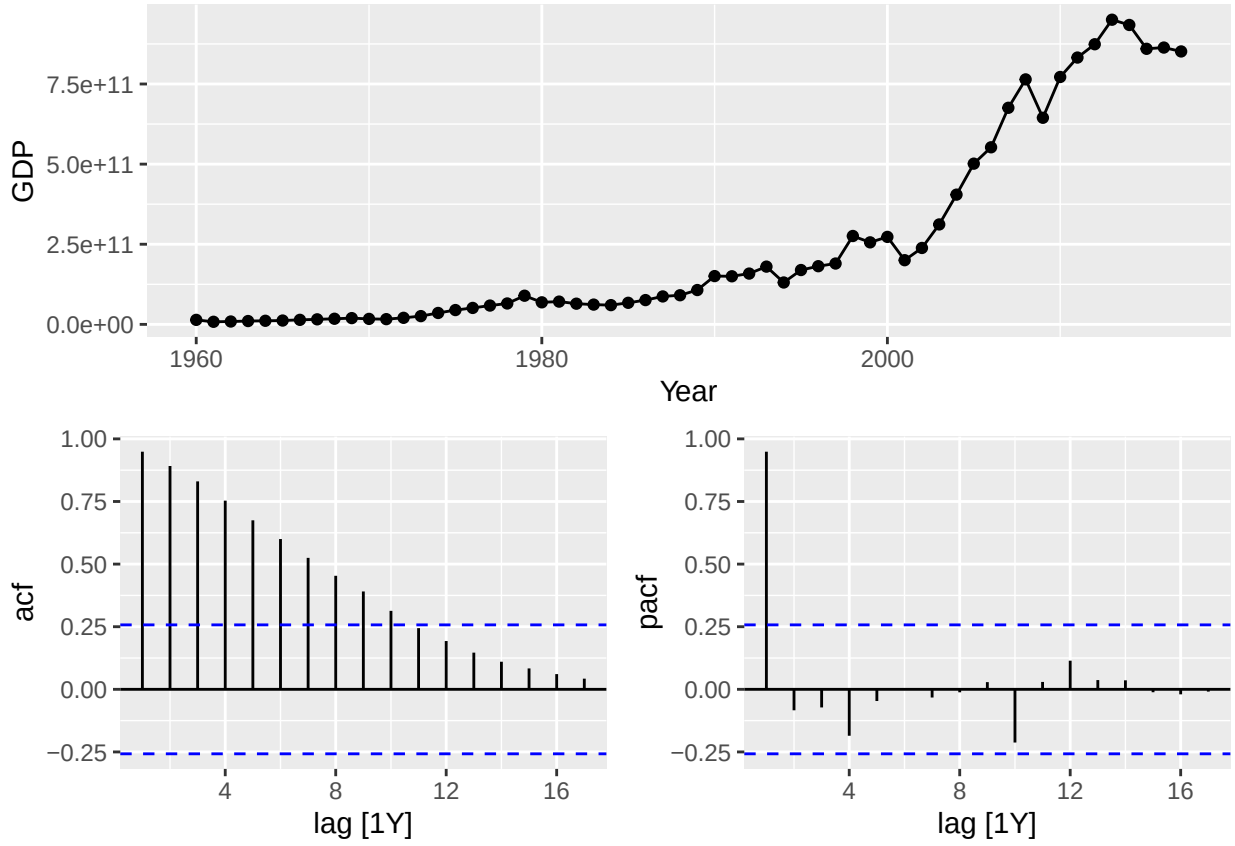
a) Turkish GDP from global_economy.

```
tur_gdp<- global_economy %>%
  filter(Country == 'Turkey') %>%
  select(Year, GDP)

tur_gdp %>% autoplot(GDP)+labs(title='turkish GDP original plot')
```



box cox transformation:

```
#auto-select lambda for transformation:
lambda<- tur_gdp %>%
  features(GDP, features = guerrero) %>%
  pull(lambda_guerrero)

#Plotting the PACF and ACF plots:
tur_gdp %>% gg_tsdisplay(GDP, plot_type = 'partial')
```



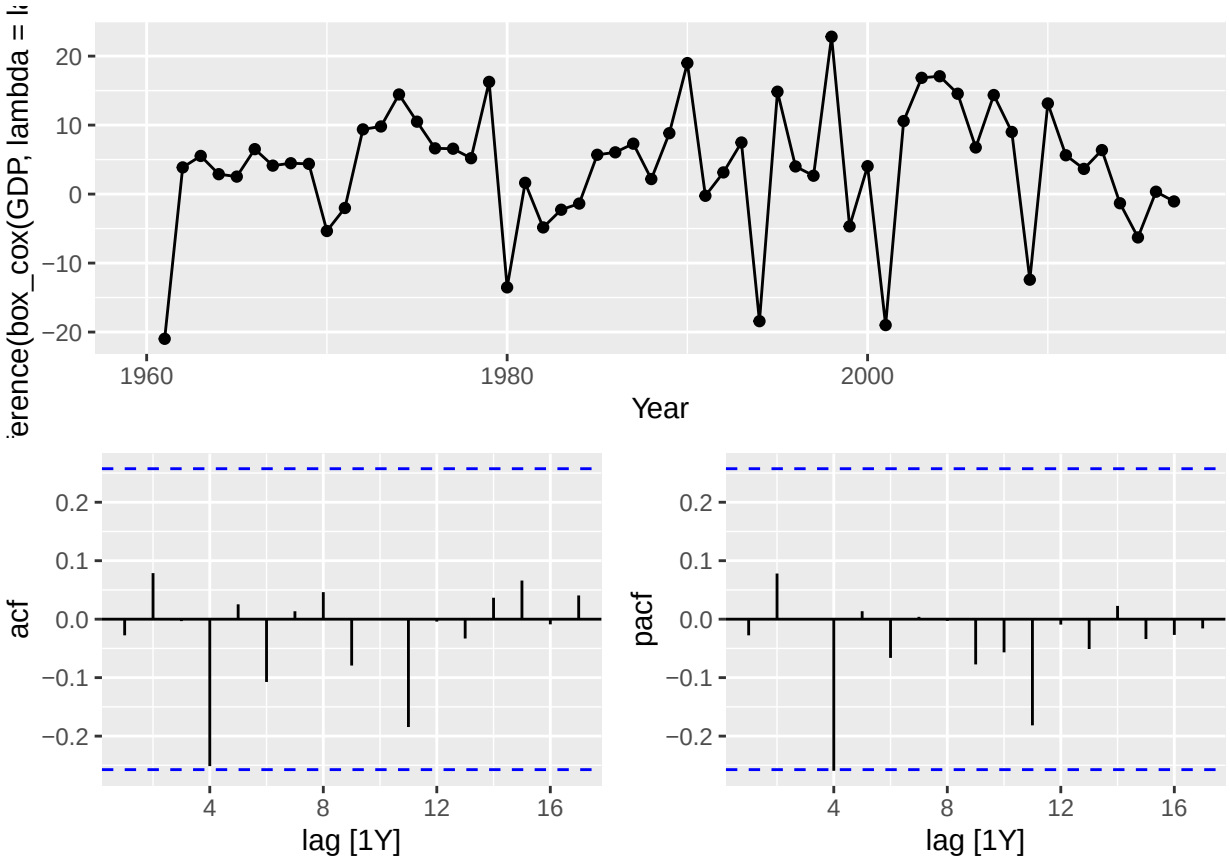

```
#compute the level of differencing
tur_gdp %>% features(GDP, unitroot_ndiffs) # 1 differencing
```

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

```
#Applying both box cox transformation and 1 level of differencing:
tur_gdp %>%
  gg_tsdisplay(difference(box_cox(GDP, lambda = lambda)),plot_type = 'partial')
```

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

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



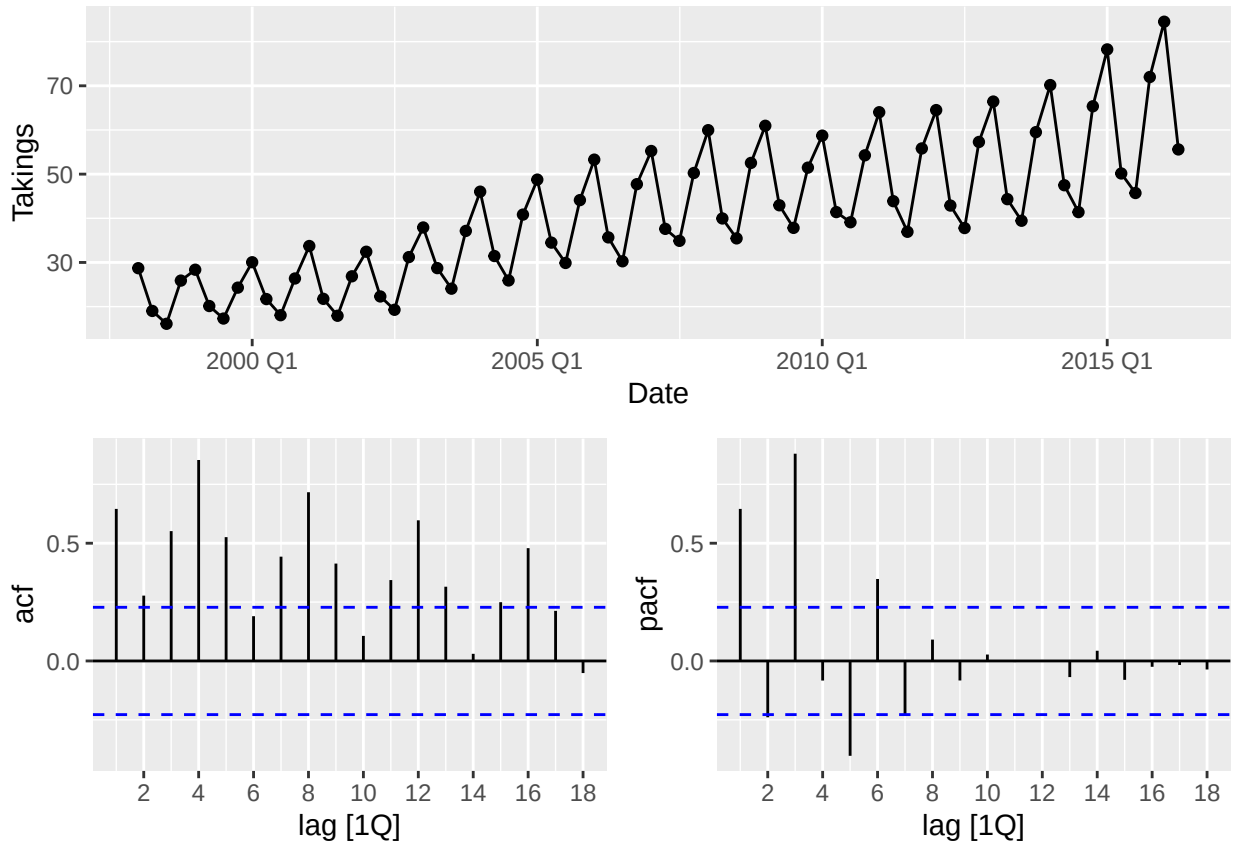

```
#Use KPSS test to confirm whether data is stationary:
tur_gdp %>%
  mutate(bc_diff = difference(box_cox(GDP, lambda = lambda))) %>%
  features(bc_diff, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0889         0.1
```

```
#P_values >= 0.1 the data is stationary
```

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

```
#loading data
tas<- aus_accommodation %>%
  filter(State == 'Tasmania') %>%
  select(Date, Takings)
#checking ACF and PACF plots:
tas %>%
  gg_tsdisplay(Takings, plot_type = 'partial')
```




```
#find the best lambda:
tas_lambda<- tas %>%
  features(Takings, features = guerrero) %>%
  pull(lambda_guerrero)
#find the best number of differencing:
tas %>% features(Takings, unitroot_ndiffs) # 1 level of differencing
```

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

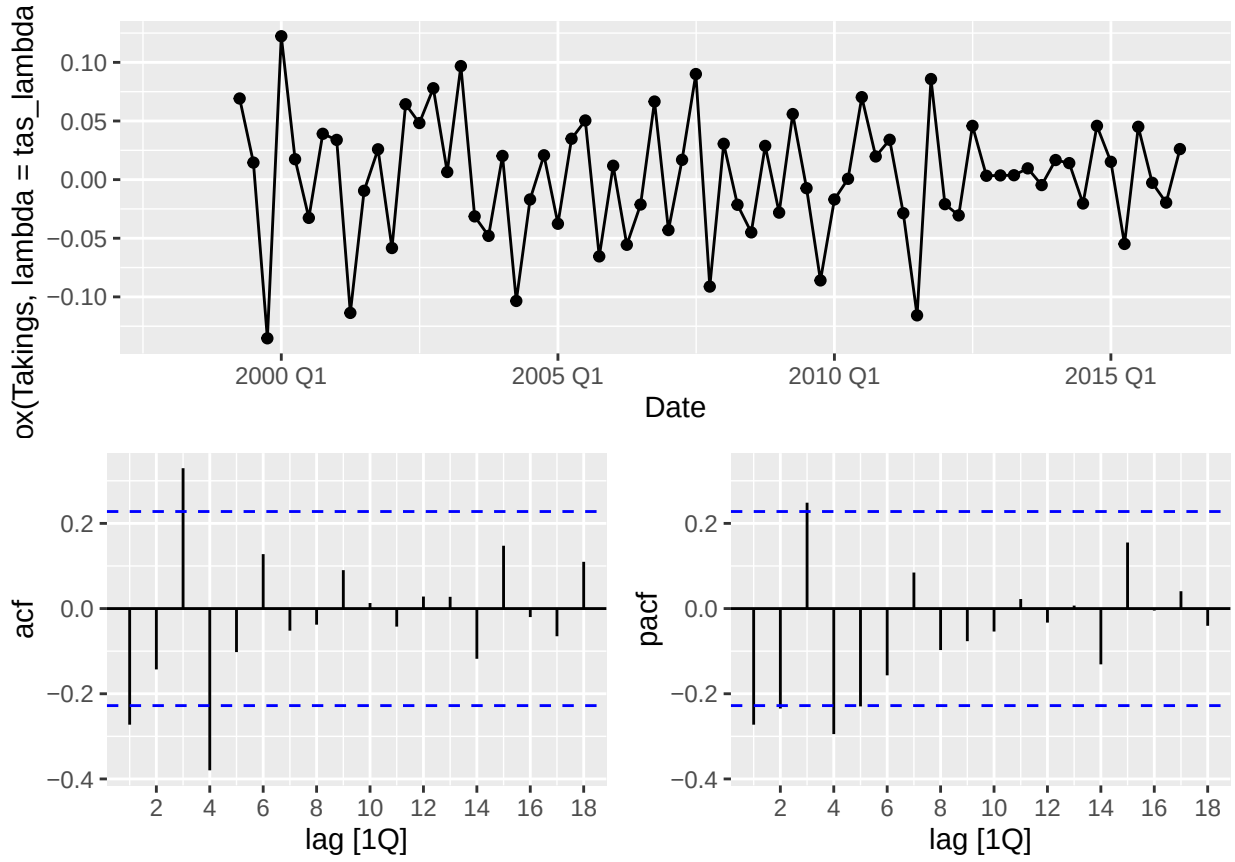
```
tas %>% features(Takings, unitroot_nsdiffs) # 1 level of seasonal differencing
```

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

```
#checking the new data:
tas %>%
  gg_tsdisplay(difference(box_cox(Takings, lambda = tas_lambda), 4) %>% difference(), plot_type = 'parti
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 5 rows containing missing values or values outside the scale range
## (`geom_point()`).
```

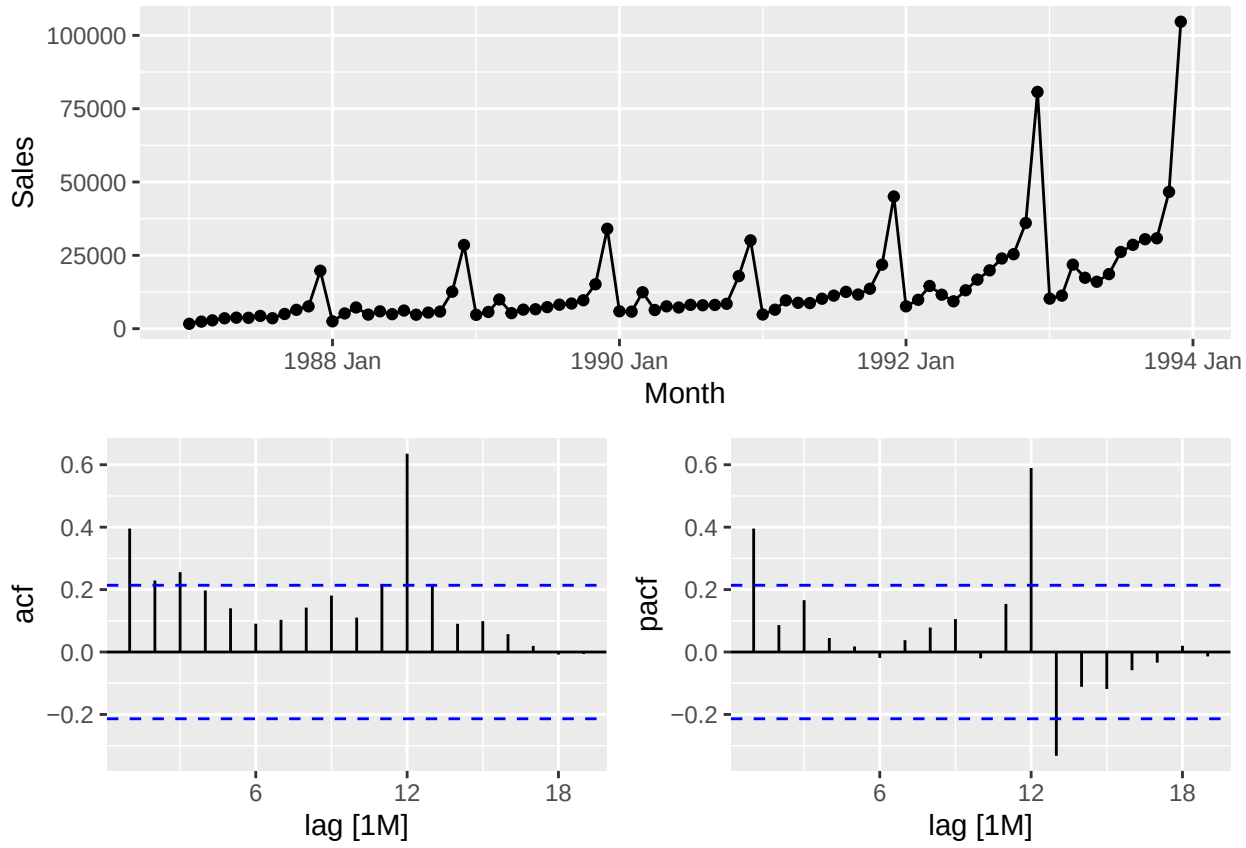



```
#confirming with KPSS test:
tas %>%
  mutate(new = difference(difference(box_cox(Takings, lambda = tas_lambda), 4))) %>%
  features(new, unitroot_kpss) #P value >= 0.1
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0474         0.1
```

c) Monthly sales from souvenirs.

```
#checking ACF and PACF plots:
souvenirs %>%
  gg_tsdisplay(Sales, plot_type = 'partial')
```




```
#find the best lambda:
ss_lambda<- souvenirs %>%
  features(Sales, features = guerrero) %>%
  pull(lambda_guerrero)
#find the best number of differencing:
souvenirs %>% features(Sales, unitroot_ndiffs) # 1 level of differencing
```

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

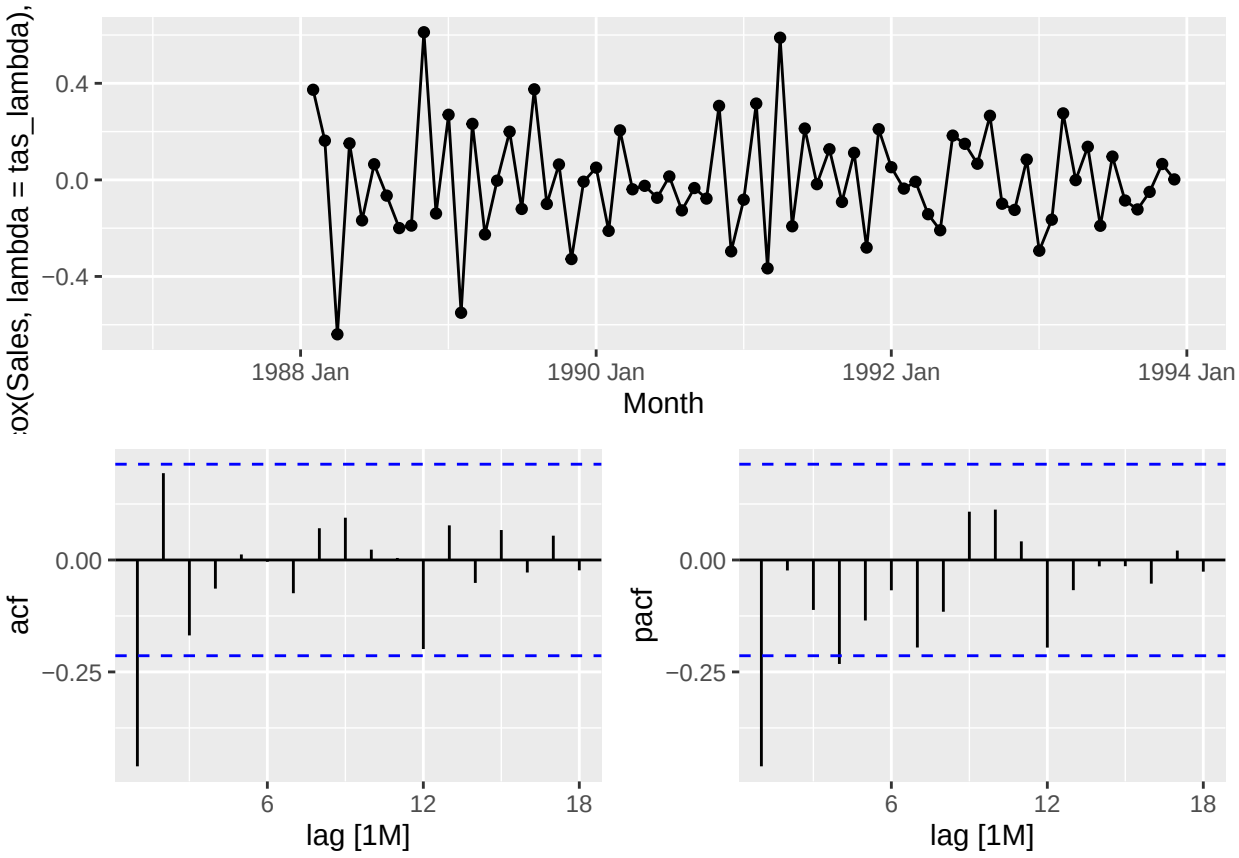
```
souvenirs %>% features(Sales, unitroot_nsdiffs) # 1 level of seasonal differencing
```

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

```
#checking the new data:
souvenirs %>%
  gg_tsdisplay(difference(box_cox(Sales, lambda = tas_lambda), 12) %>% difference(), plot_type = 'parti
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#confirming with KPSS test:
souvenirs %>%
  mutate(new = difference(difference(box_cox(Sales, lambda = tas_lambda), 12))) %>%
  features(new, unitroot_kpss) #P value >= 0.1
```

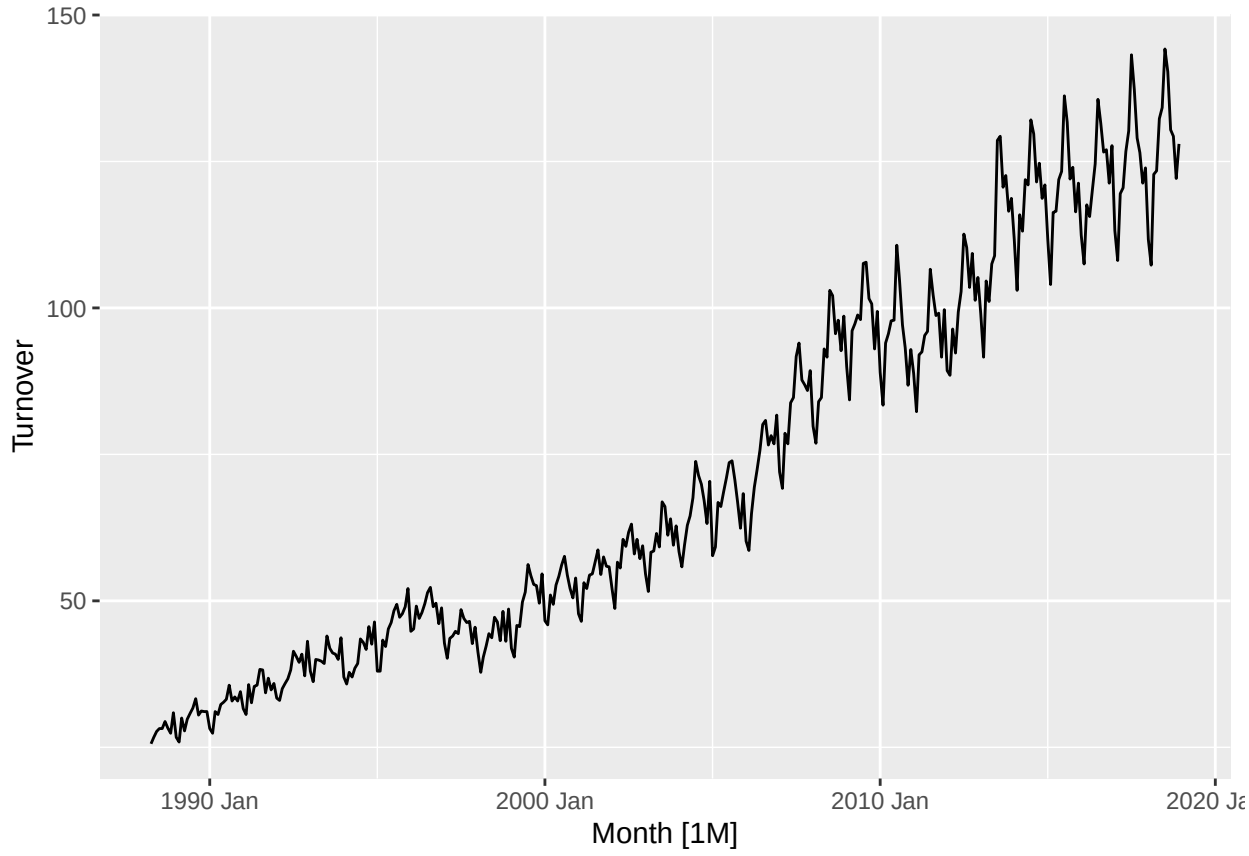
```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0381         0.1
```

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.

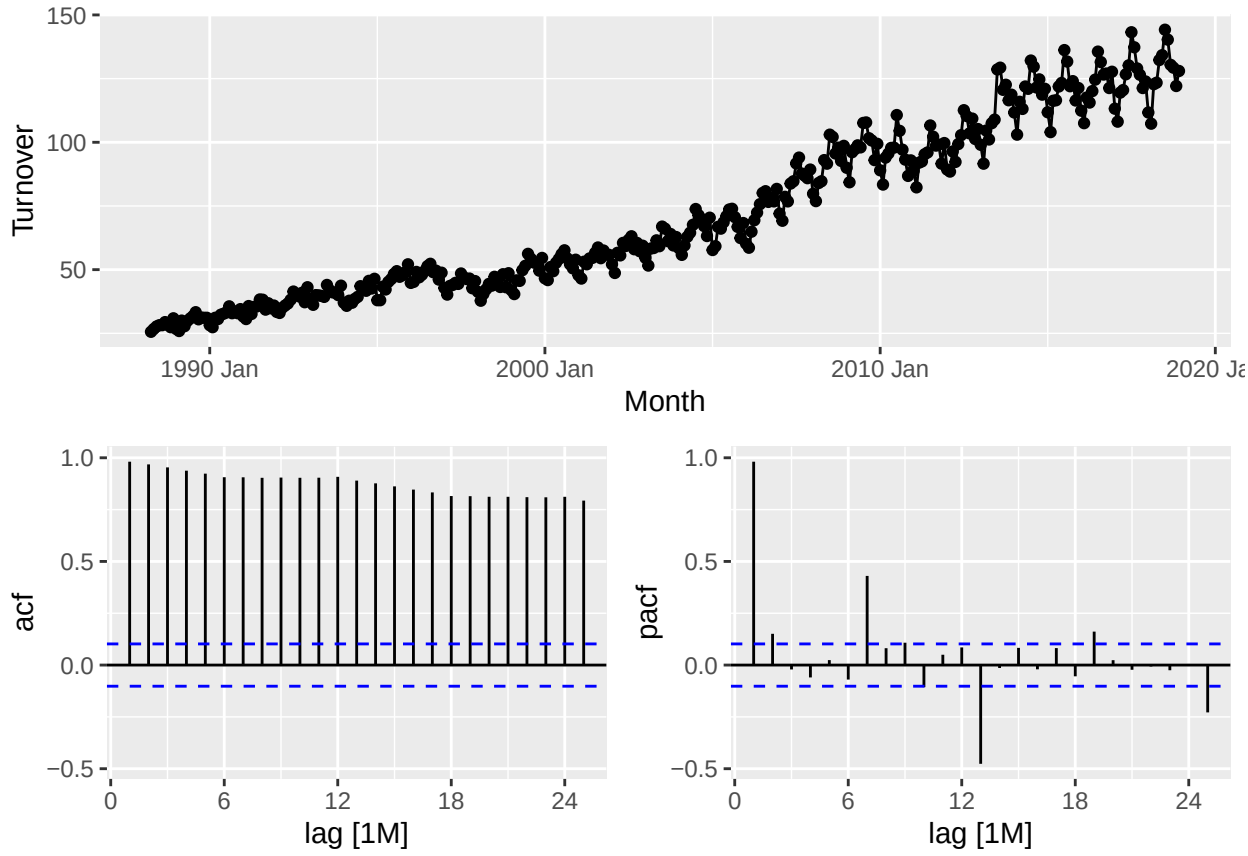
```
#load data:
set.seed(123458)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`,1)) %>%
  select(Month, Turnover)
```

```
#checking the data
myseries %>% autoplot()
```

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



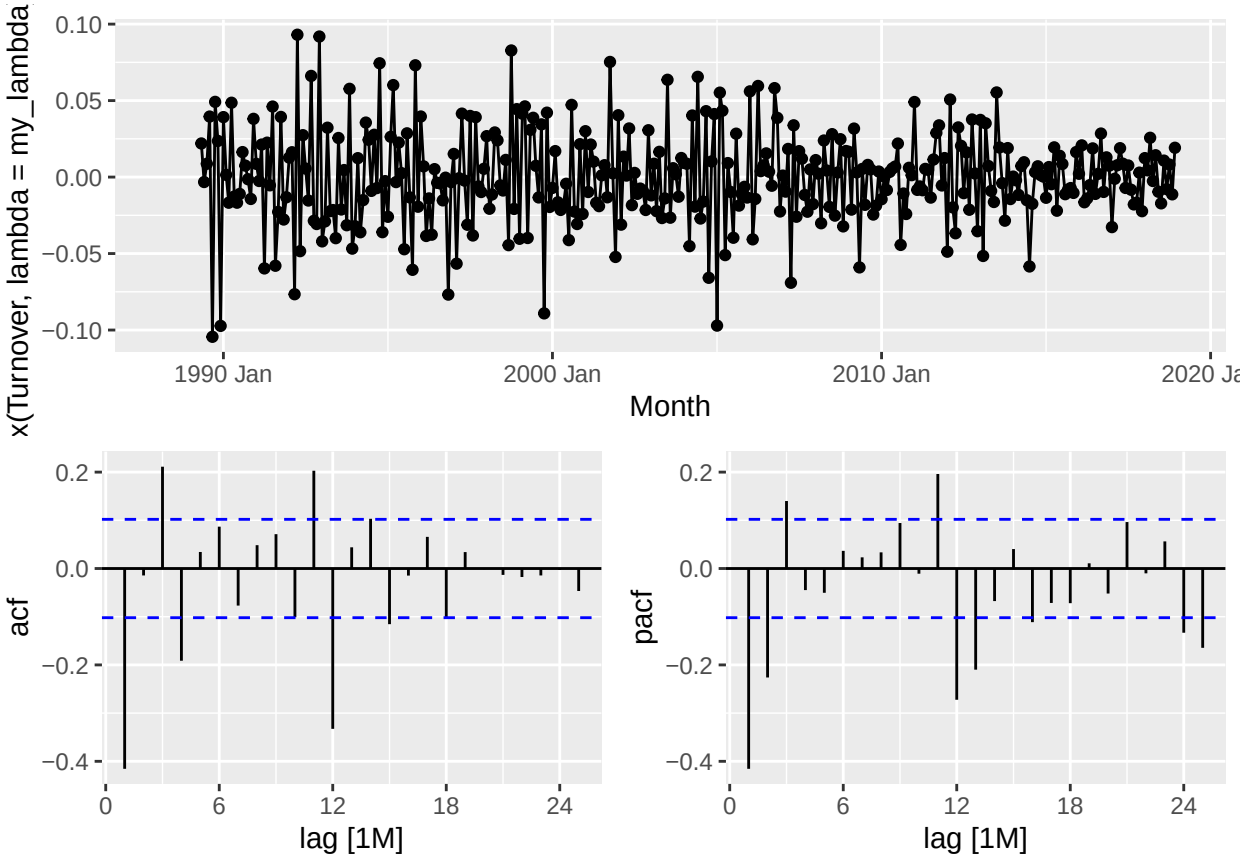

```
#Find the best lambda for box cox:
my_lambda<- myseries %>%
  features(Turnover, features = guerrero) %>%
  pull(lambda_guerrero)
#checking the ACF and PACF plot:
myseries %>% gg_tsdisplay(Turnover, plot_type = 'partial')
```




```
#checking the ACF and PACF after the seasonal differencing and first differencing:
myseries %>%
  gg_tsdisplay(difference(box_cox(Turnover, lambda = my_lambda), 12) %>% difference(), plot_type = 'par
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 13 rows containing missing values or values outside the scale range
## (`geom_point()`).
```




```
#checking the p_value from KSPP test:
myseries %>%
  mutate(new = difference(difference(box_cox(Turnover, lambda = my_lambda), 12))) %>%
           features(new, unitroot_kpss) #P-value >= 0.1 the data is stationary
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0179         0.1
```

9.6) Simulate and plot some data from simple ARIMA models

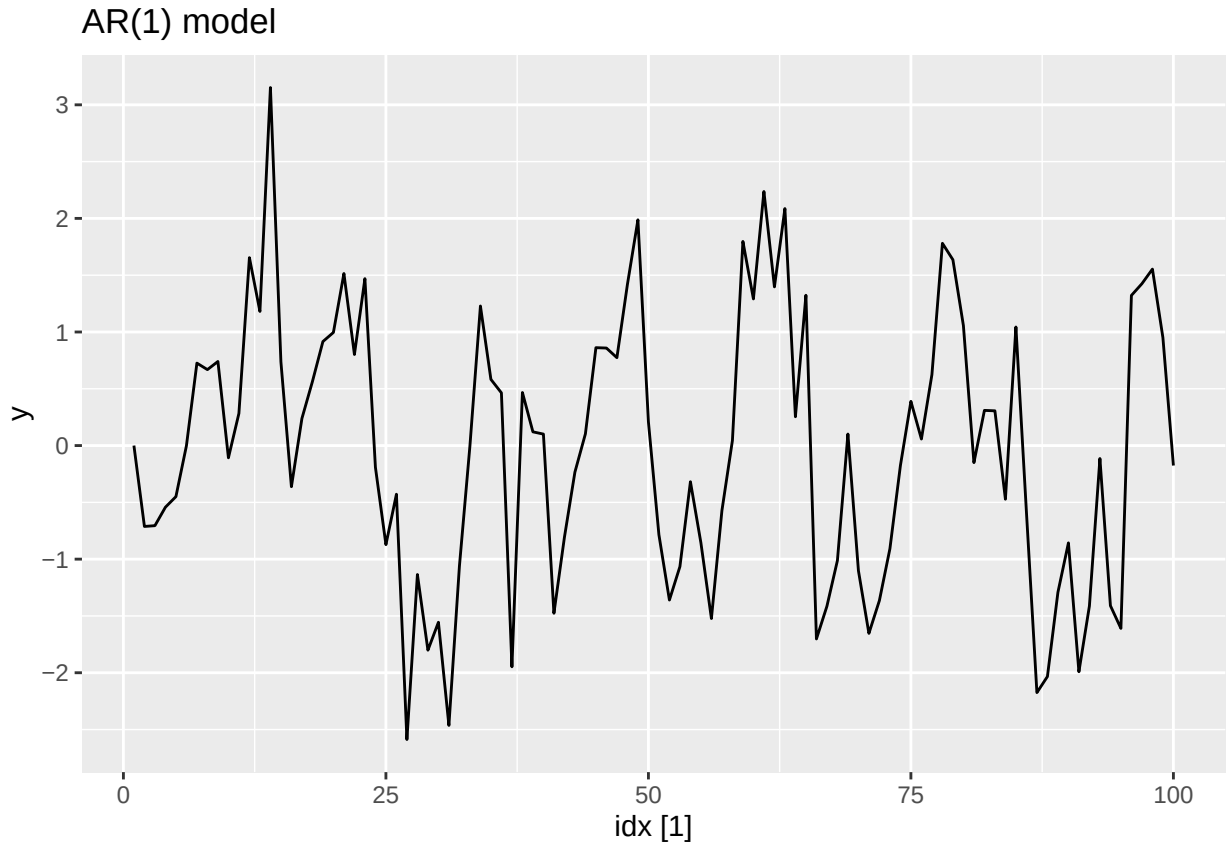
a)

```
set.seed(321)
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 phi1? Ans: As phi increases, the plot becomes 'smoother' and the pattern of the time series becomes clearer.

```
sim %>% autoplot()+ggtitle('AR(1) model')
```

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



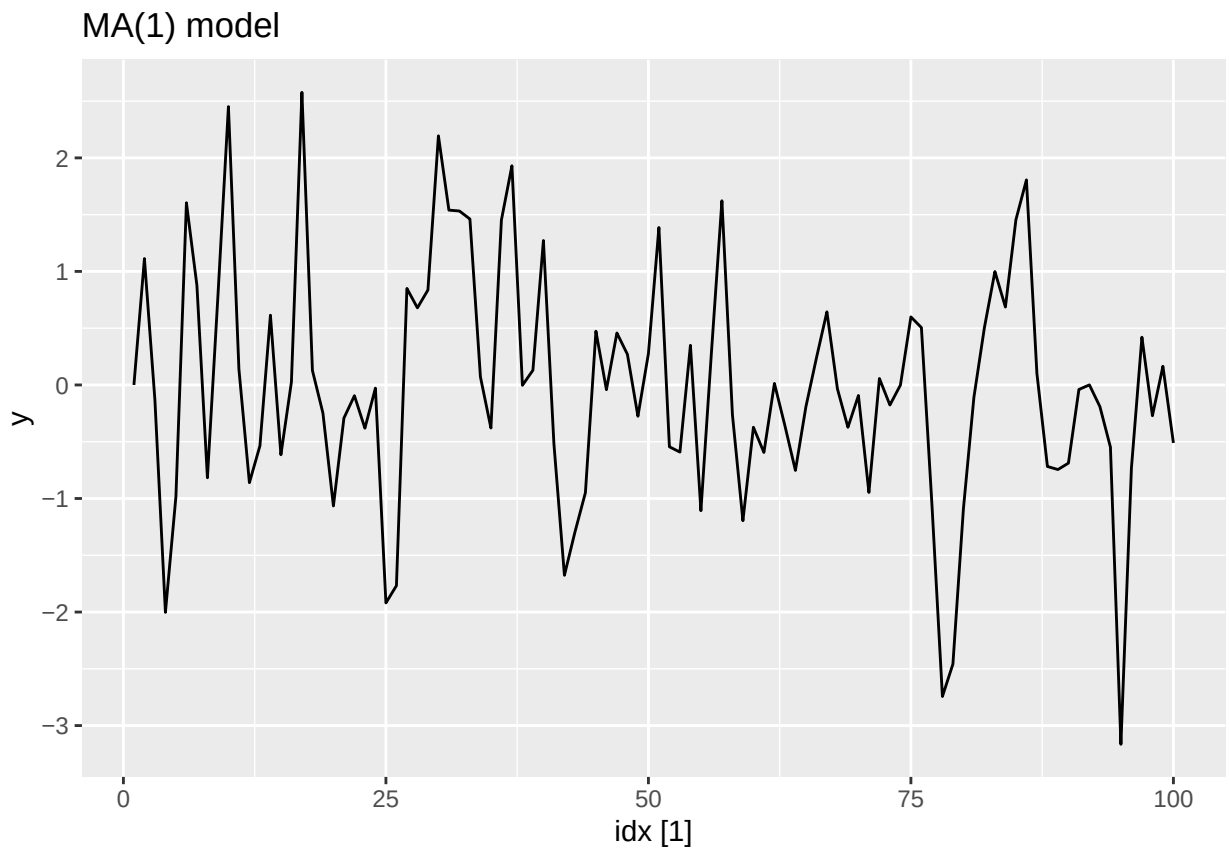
c) Write your own code to generate data from an MA(1) model with theta1 = 0.6 and sigma^2 = 1:

```
set.seed(311)
theta_1 <- 0.6
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- theta_1*e[i-1] + e[i]
sim_ma <- 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 theta1?

Ans: The idea is same as increasing phi value in AR(1) model. Increasing Theta in MA(1) model, the plot will become more smooth and weighting the present values more heavily. The data becomes more persistent to the present values.

```
sim_ma %>% autoplot(y) + ggtitle('MA(1) model')
```



e) Generate data from an ARMA(1,1) model with phi1=0.6, theta = 0.6, sigmaˆ2 = 1

```
set.seed(311)
theta_1 <- 0.6
phi <- 0.6
y <- numeric(100)
e <- rnorm(100)
for(t in 2:100) {
  y[t] <- theta_1*e[t-1] + e[t] + phi*y[t-1]}
sim_arma11 <- tsibble(idx = seq_len(100), y = y, index = idx)
p1<- sim_arma11 %>% autoplot(y) +ggtitle('arma11')
```

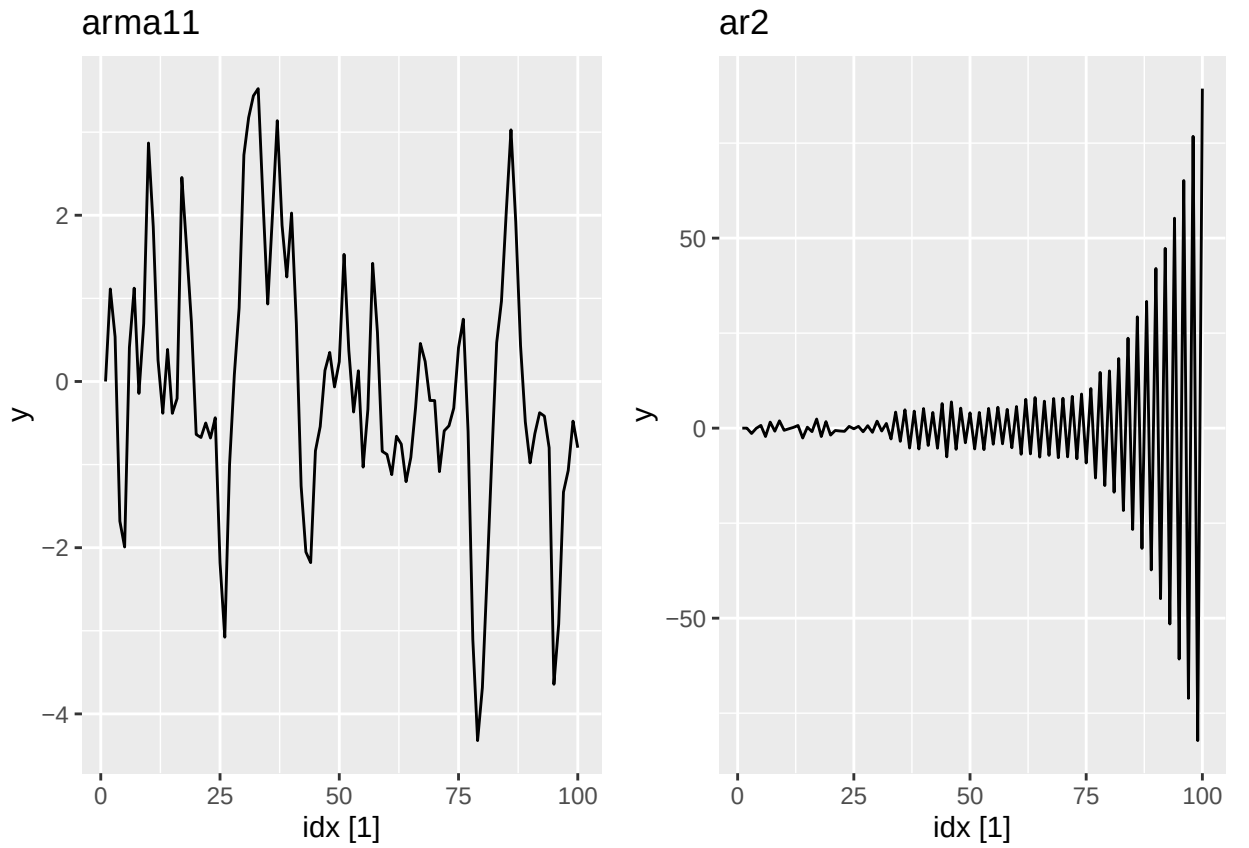
f) Generate data from an AR(2) model with phi1 = -0.8, phi2=0.3 and sigmaˆ2 = 1

```
phi1 <- -0.8
phi2<- 0.3
y <- numeric(100)
e <- rnorm(100)

for(t in 3:100)
  y[t] <- e[t] + phi1*y[t-1] + phi2*y[t-2]
sim_ar2 <- tsibble(idx = seq_len(100), y = y, index = idx)
p2<- sim_ar2 %>% autoplot(y) + ggtitle('ar2')
```

g) Graph the latter two series and compare them.

Ans: ar2 with phi1 = -0.8 and phi2 = 0.3 causes the plot to oscillate and increases over time. The charactistics of a non-stationary data.

```
grid.arrange(p1, p2, ncol = 2)
```



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

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. Ans: The stepwise ARIMA selected (0,2,1)

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

#checking for model selection:
report(ps_fit)
```

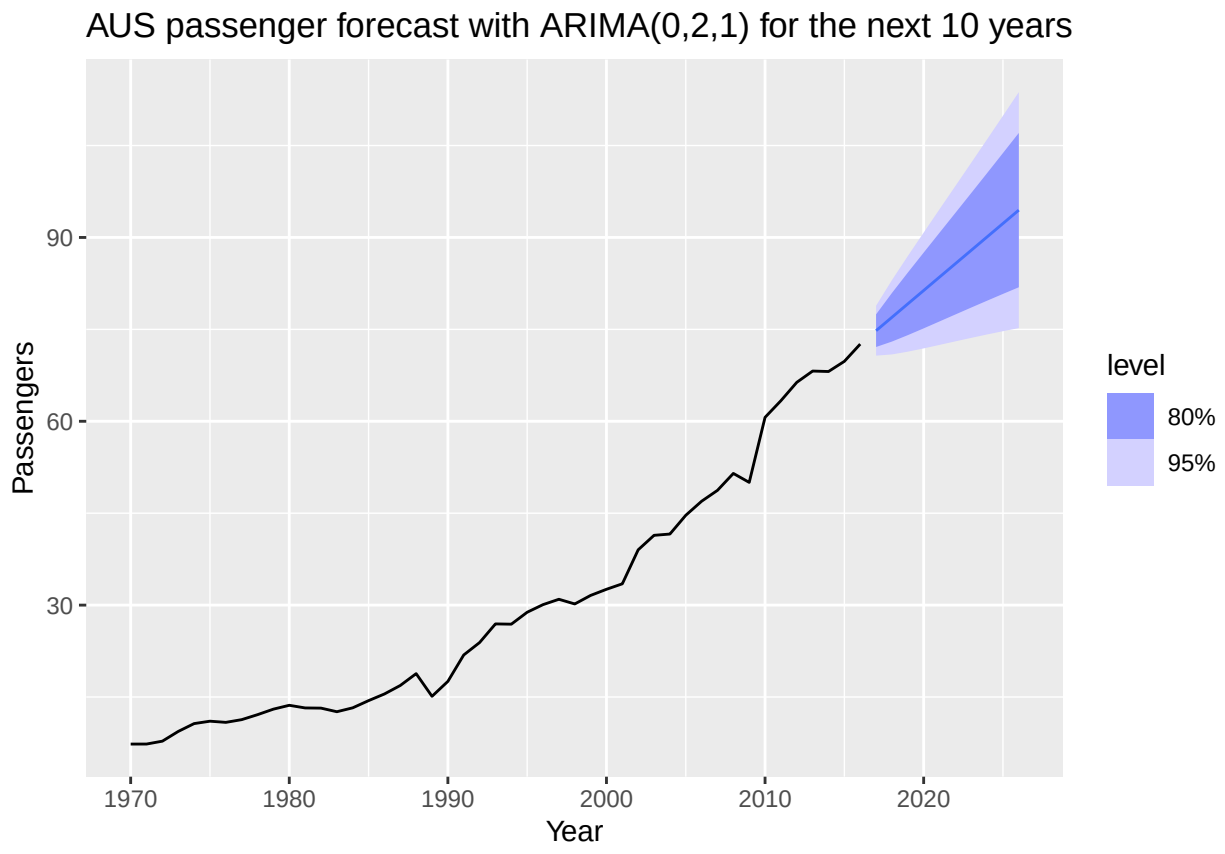
```
## Series: Passengers
## Model: ARIMA(0,2,1)
##
## Coefficients:
##          ma1
```

```
##       -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
```

```
#checking residuals
augment(ps_fit) %>%
  features(.innov, ljung_box, lag=24, dof = 1) #p_vale = 0.4, meaning residuals are white-noises
```

```
## # A tibble: 1 x 3
##   .model            lb_stat lb_pvalue
##   <chr>               <dbl>     <dbl>
## 1 ARIMA(Passengers)    24.0     0.405
```

```
#plotting forecasts:
ps_fit %>%
  forecast(h = '10 year') %>%
  autoplot(aus_airpassengers)+
  ggtitle('AUS passenger forecast with ARIMA(0,2,1) for the next 10 years')
```



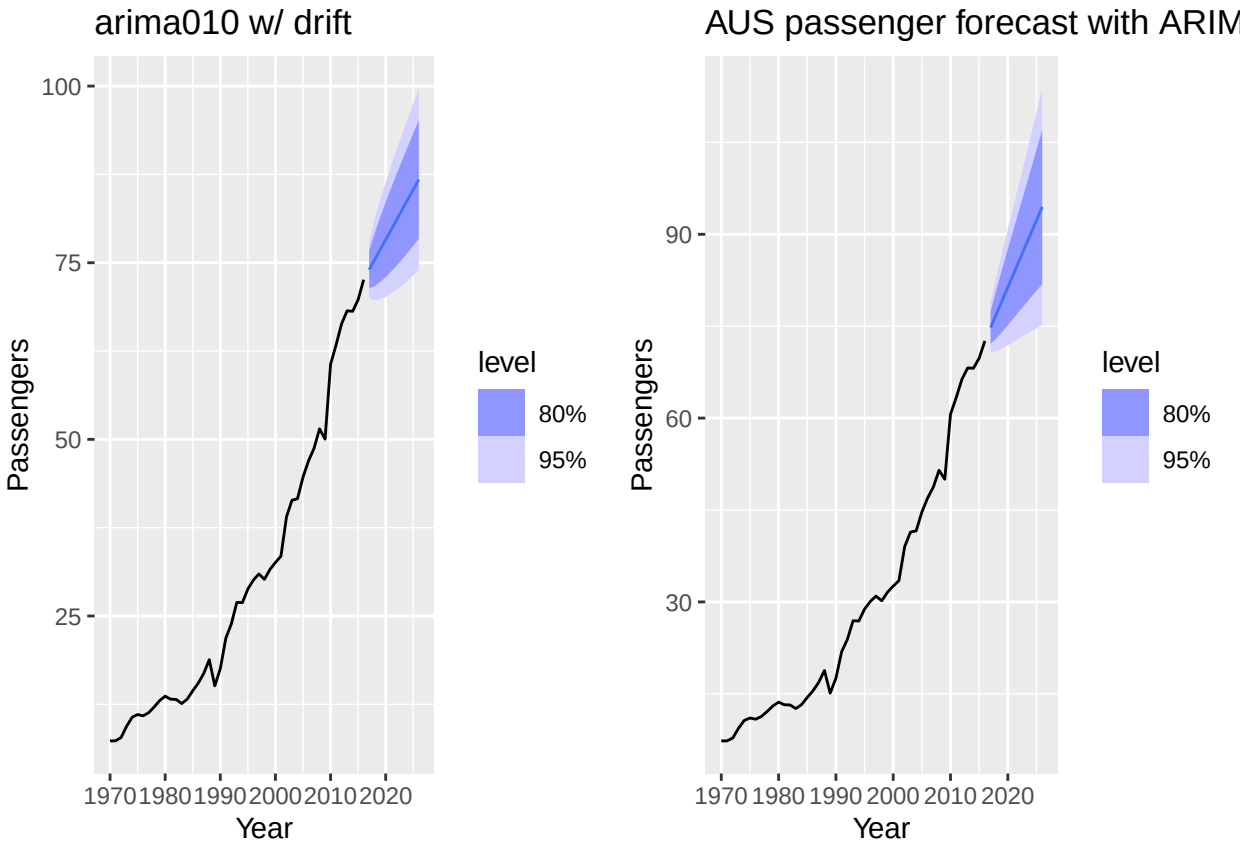
b) Write the model in terms of the backshift operator.

Ans:For Arima (0,2,1): (1-B)^2*y[t] = e[t] + 1*e[t-1]

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

Ans: The two forecasts are very similar to each other with an increasing forecast over time.

```
#added a constant for drift (mu)
ps_fit_drift<- aus_airpassengers %>%
  model(
    ARIMA(Passengers  ~ 1 + pdq(0,1,0))
  )
p1<- ps_fit_drift %>% forecast(h = '10 year') %>% autoplot(aus_airpassengers) + ggtitle('arima010 w/ dr
p2<- ps_fit %>%
  forecast(h = '10 year') %>%
  autoplot(aus_airpassengers)+
  ggtitle('AUS passenger forecast with ARIMA(0,2,1) for the next 10 years')

grid.arrange(p1,p2, ncol=2)
```

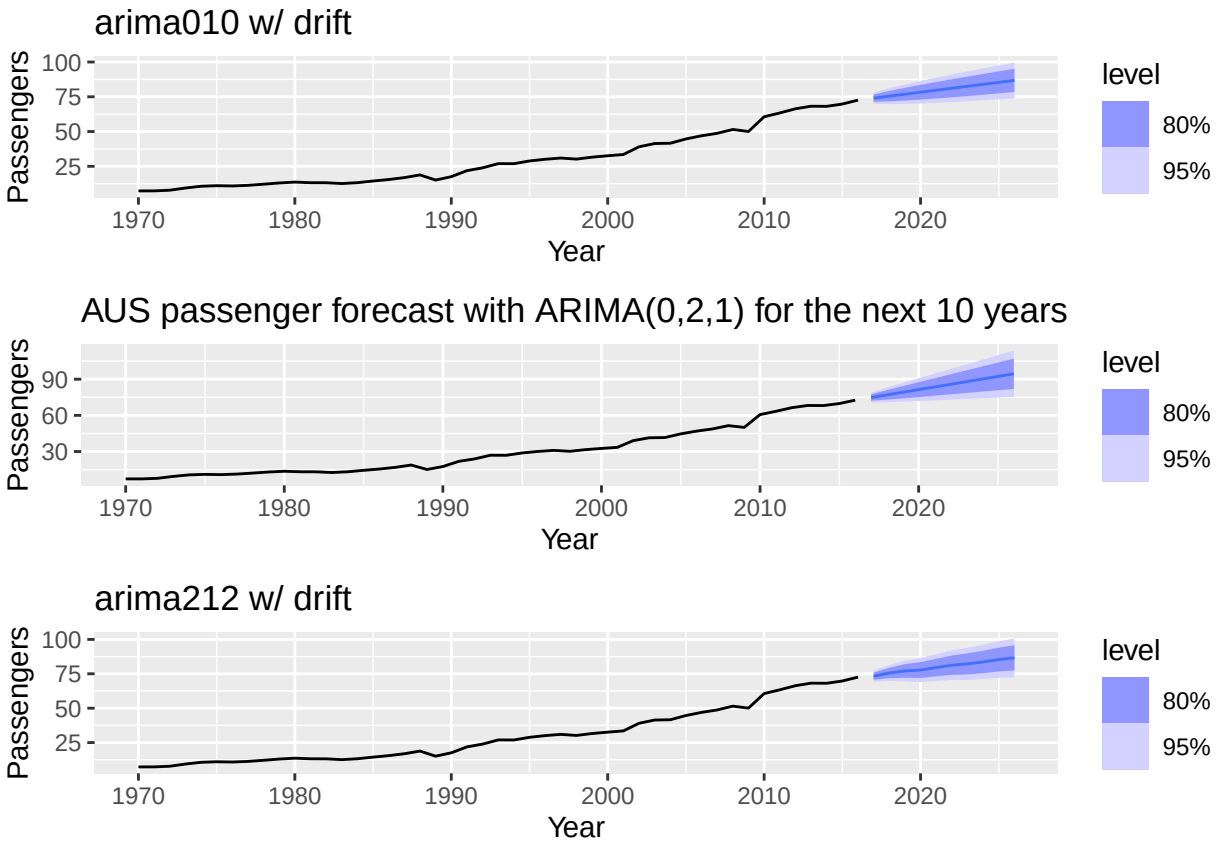


d) Plot forecasts from an ARIMA(2,1,2) model with drift and compare these to parts a and c. Remove the constant and see what happens.

Ans: with drift, the results are similar. However, removing the drift constant, the ARIMA is unable to produce a forecast due to the non-stationary nature of the data. The drifting constant is needed to capture the upward trend unless another degree of differencing is applied.

```
ps_fit_drift212<- aus_airpassengers %>%
  model(
```

```
    ARIMA(Passengers  ~ 1 + pdq(2,1,2))
  )
p3<- ps_fit_drift212 %>% forecast(h = '10 year') %>% autoplot(aus_airpassengers) + ggtitle('arima212 w/
grid.arrange(p1,p2,p3)
```



Removing the drift constant:

```
ps_fit_212<- aus_airpassengers %>%
  model(
    ARIMA(Passengers ~ 0 + pdq(2,1,2))
  )
```

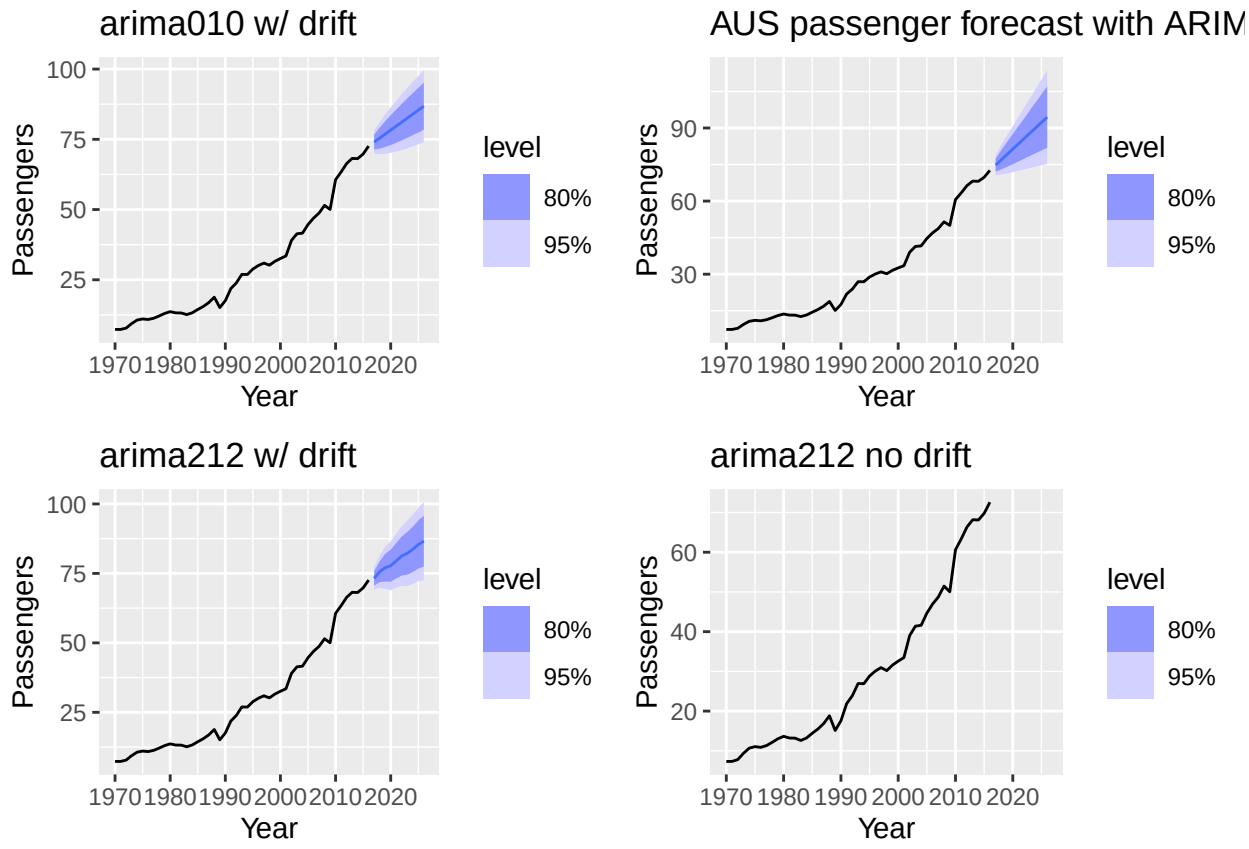
```
## Warning: 1 error encountered for ARIMA(Passengers ~ 0 + pdq(2, 1, 2))
## [1] non-stationary AR part from CSS
```

```
p4<- ps_fit_212 %>% forecast(h = '10 year') %>% autoplot(aus_airpassengers) + ggtitle('arima212 no drif
grid.arrange(p1,p2,p3, p4)
```

```
## 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()`).
```

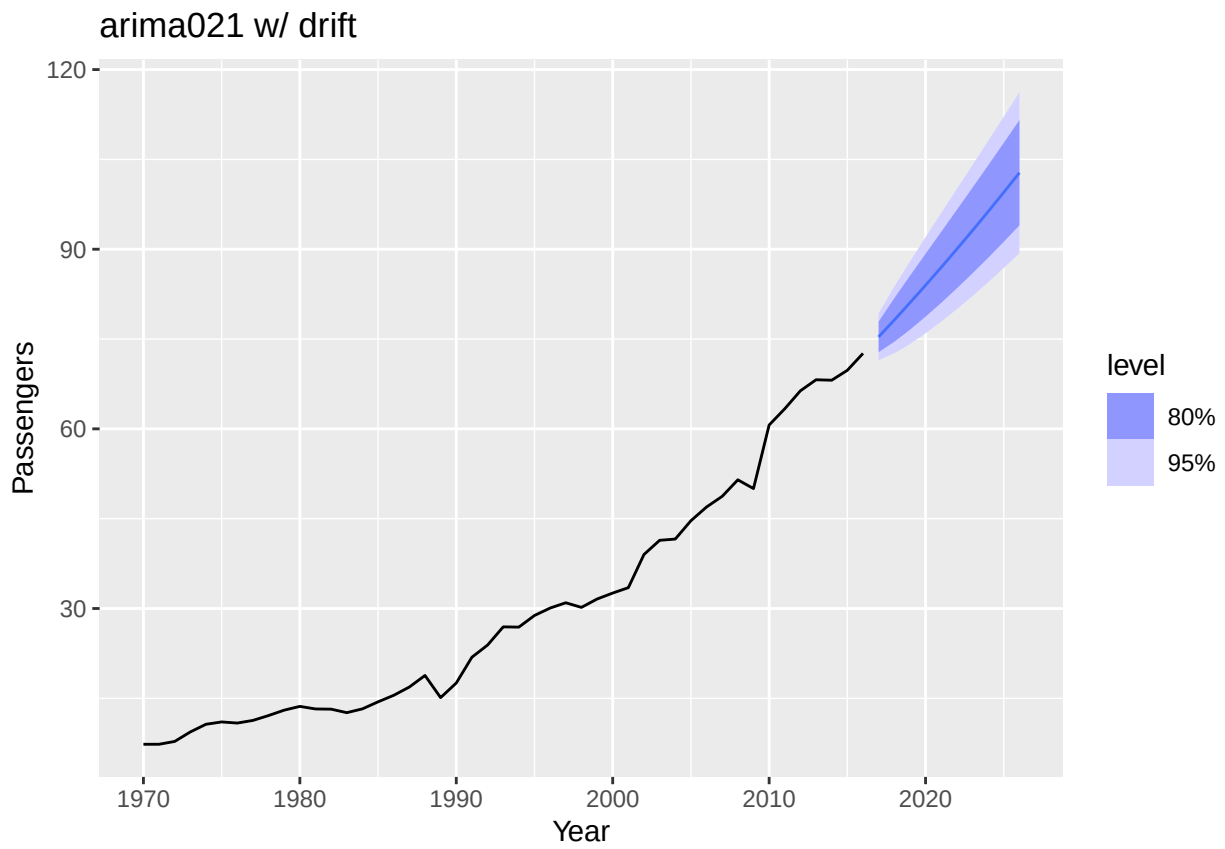


e) Plot forecasts from an ARIMA(0,2,1) model with a constant. What happens? Ans: There is an linear upward trend forecast in the data

```
ps_fit_drift021<- 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.
```

```
ps_fit_drift021 %>% forecast(h = '10 year') %>% autoplot(aus_airpassengers) + ggtitle('arima021 w/ drif
```



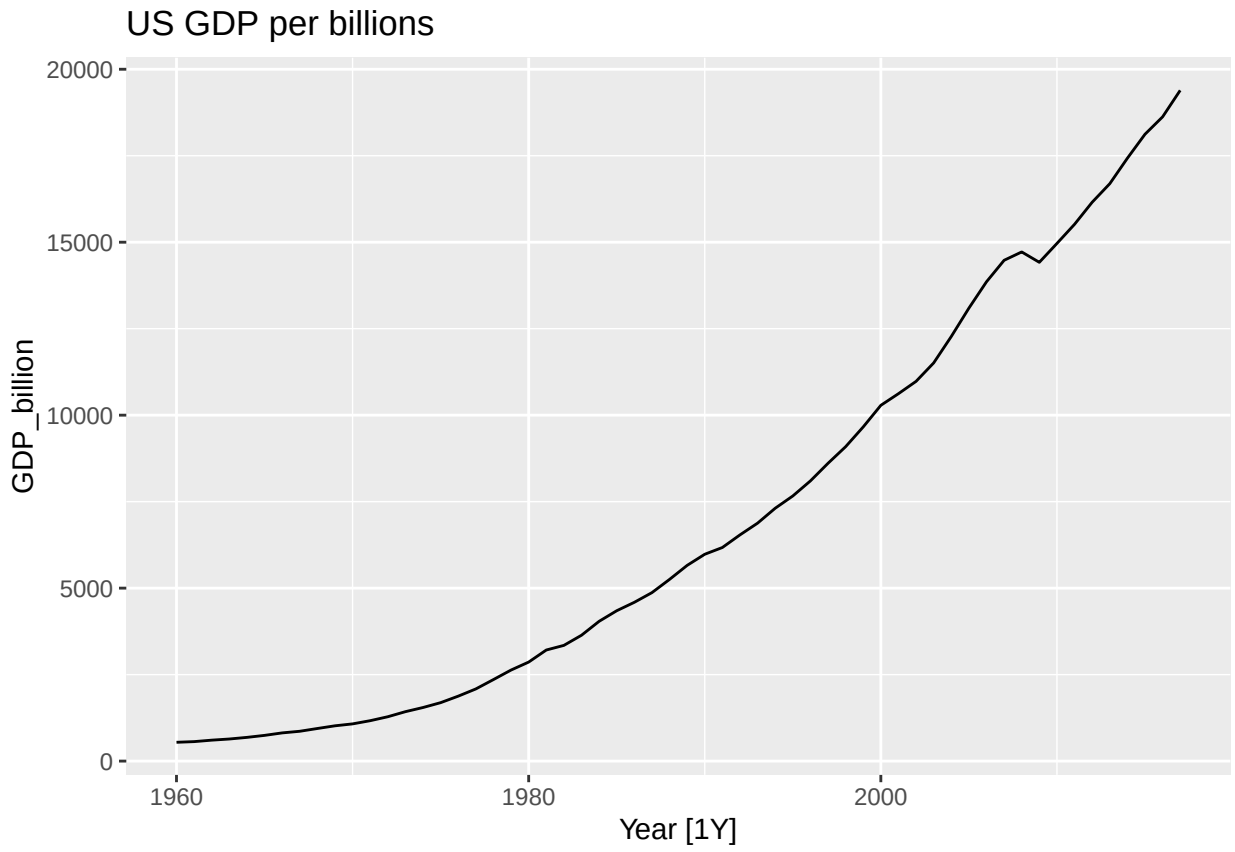
9.8 For the United States GDP series (from global_economy):

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

Ans: Not necessary to transform since the data does not show high variance. However, a box cox is done to see compare the results.

```
us<- global_economy %>%
  filter(Country == 'United States') %>%
  select(Year, GDP) %>%
  summarise(GDP_billion= sum(GDP)/1e9)
us %>% autoplot() + ggtitle('US GDP per billions')
```

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




```
#box cox transformation
us_lambda<- us %>%
  features(GDP_billion, features = guerrero) %>%
  pull(lambda_guerrero)

us<- us %>%
  mutate(bc = box_cox(GDP_billion, lambda = us_lambda))
```

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

c) try some other plausible models by experimenting with the orders chosen;
Ans: ARIMA (0,2,2) and box cox transformed = ARIMA (1,1,0) w/ drift

```
us_fit<- us %>%
  model(
    original = ARIMA(GDP_billion),
    original_max_search = ARIMA(GDP_billion, stepwise = FALSE, approximation = FALSE),
    box_cox = ARIMA(box_cox(GDP_billion, us_lambda)),
    box_cox_max_search = ARIMA(box_cox(GDP_billion, us_lambda), stepwise = FALSE, approximation = FALSE)
  )
#arrange the data
us_fit_long<- us_fit %>%
  pivot_longer(everything(),
               names_to = 'models',
               values_to = 'orders')
```

```
#compare models only with the transformed data or the untransformed data:
glance(us_fit_long %>% filter((models == 'original' | models == 'original_max_search')))
```

```
## # A tibble: 2 x 9
##   models              .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>               <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 original            orders 26150.   -364.  733.  734.  739. <cpl [0]> <cpl>
## 2 original_max_search orders 26150.   -364.  733.  734.  739. <cpl [0]> <cpl>
```

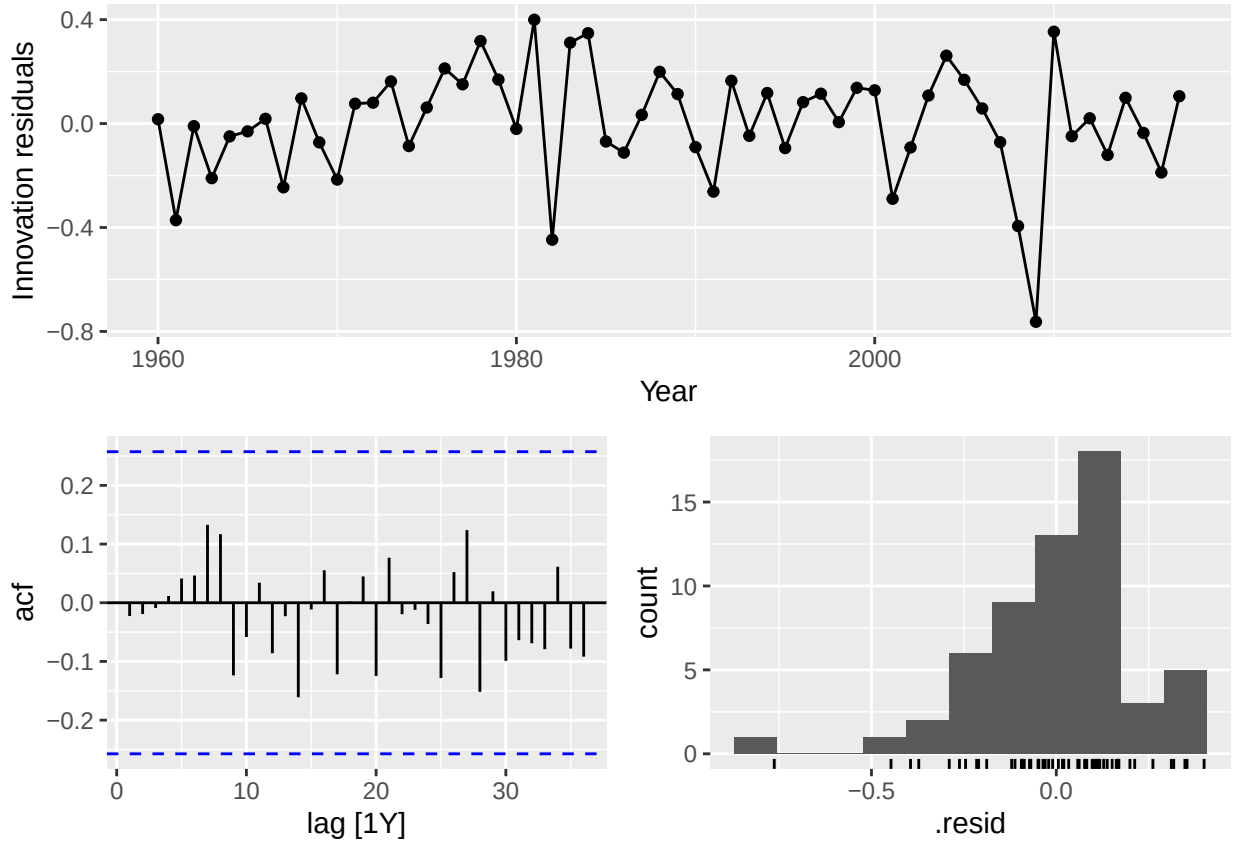
```
glance(us_fit_long %>% filter((models == 'box_cox' | models == 'box_cox_max_search')))
```

```
## # A tibble: 2 x 9
##   models             .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>              <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 box_cox            orders 0.0461    7.72 -9.43 -8.98 -3.30 <cpl [1]> <cpl [0]>
## 2 box_cox_max_search orders 0.0461    7.72 -9.43 -8.98 -3.30 <cpl [1]> <cpl [0]>
```

```
#both are same with the most exhaustive search
```

d) choose what you think is the best model and check the residual diagnostics;
Ans: i choose the box cox transformed data

```
us_fit_long %>%
  filter(models == 'box_cox') %>%
  gg_tsresiduals(lag=36) # all ACF spikes are not significant. reisduals are mostly distributed normall
```



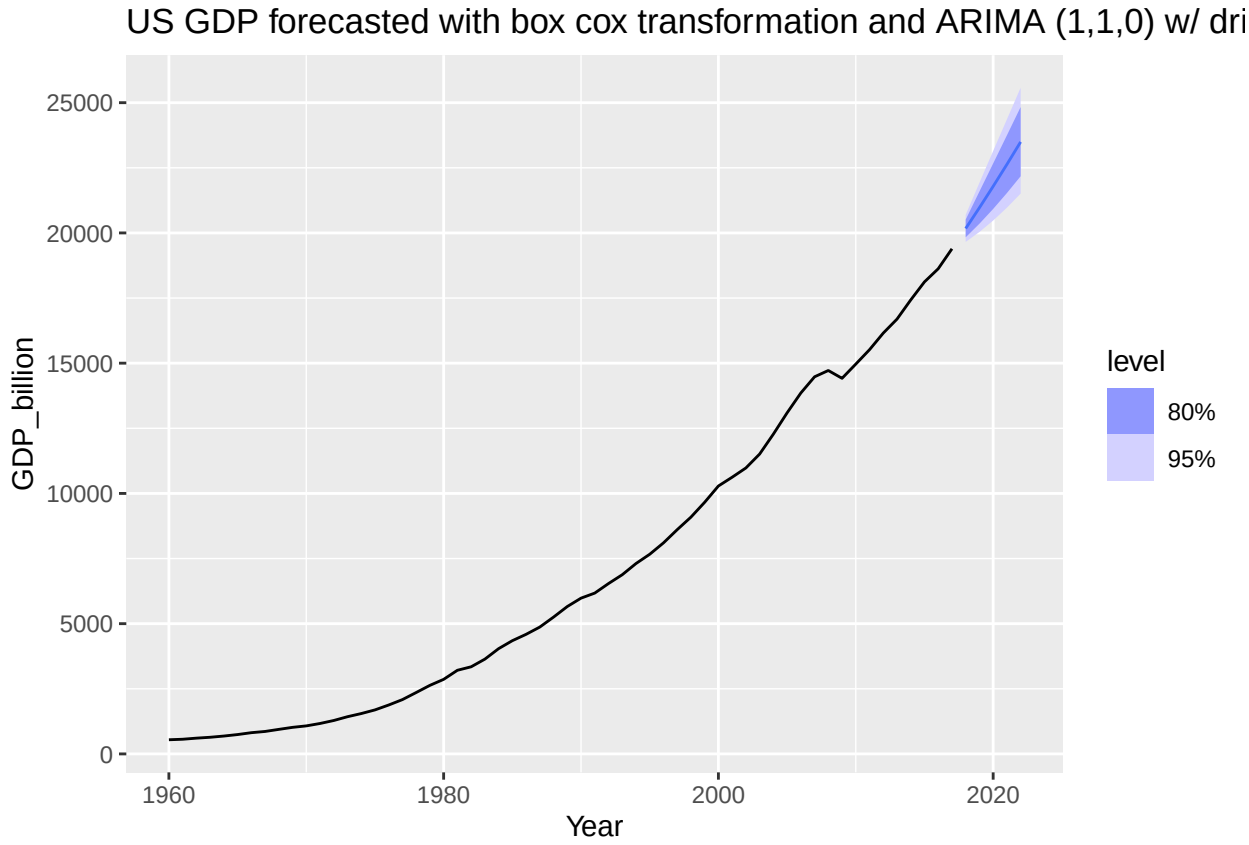
```
#checking if residuals are white noise:
us_fit_long %>%
  filter(models =='box_cox') %>%
  augment() %>%
  features(.innov, ljung_box, lag = 24, dof= 2) #p-value > 0.05, the residuals are of white-noise
```

```
## # A tibble: 1 x 4
##   models  .model lb_stat lb_pvalue
##   <chr>   <chr>    <dbl>     <dbl>
## 1 box_cox orders    10.4     0.982
```

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

Ans: it does look reasonable since it captures the upward trend nicely.

```
us_fc_bc<- us_fit_long %>%
  filter(models == 'box_cox') %>%
  forecast(h = 5)

#create box cox transformed data column:
us_fc_bc %>%
  select(Year, GDP_billion) %>%
  autoplot(us) + ggtitle('US GDP forecasted with box cox transformation and ARIMA (1,1,0) w/ drift')
```



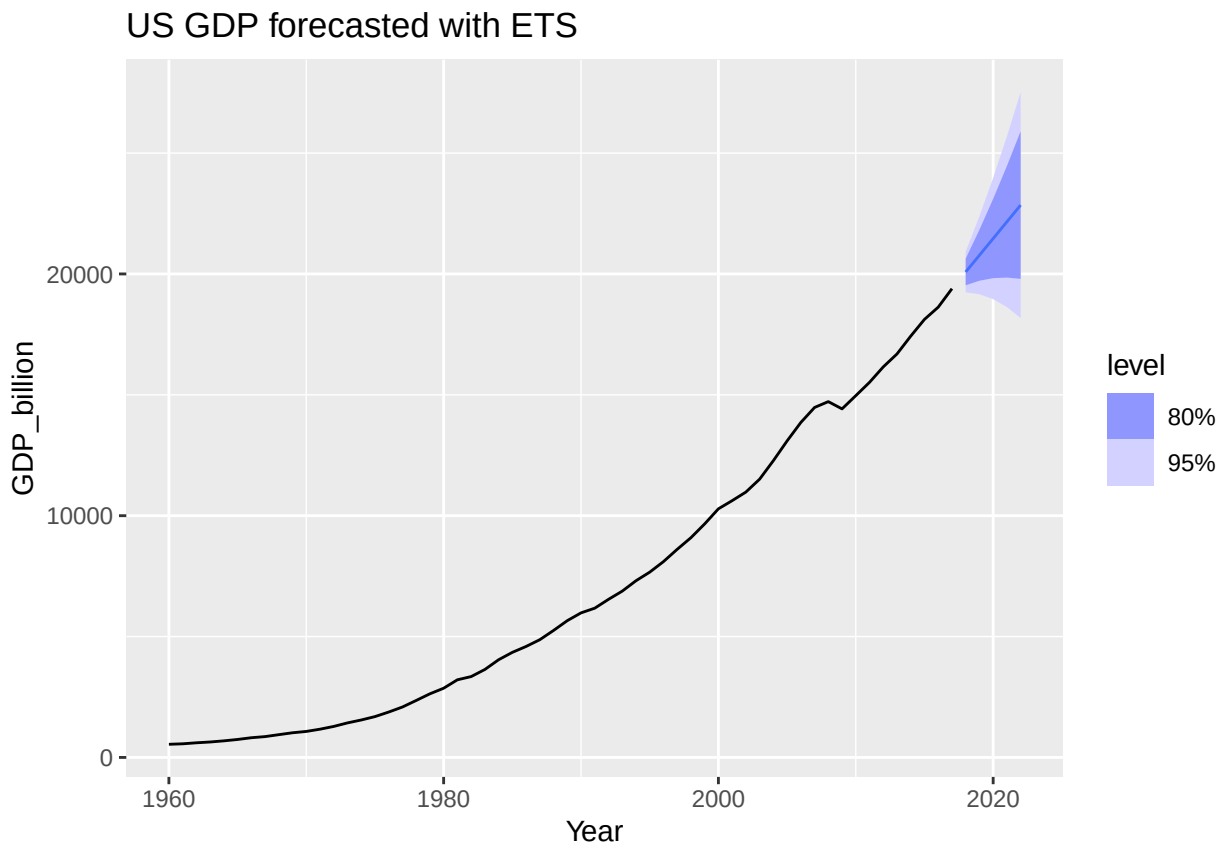
f) compare the results with what you would obtain using ETS() (with no transformation).

Ans: The ETS model selected with the trend and multiplicity included. The forecast interval is wider than the ARIMA model with possibility of a downward in the 95% interval. The ARIMA with drift constant captures the upward trend of the data and forecasted a narrower interval.

```
#auto search for ETS models:
us_ets<- us %>%
  model(
    ETS(GDP_billion)
  )
#checking for ETS model selected: ETS(M,A,N)
report(us_ets)
```

```
## Series: GDP_billion
## Model: ETS(M,A,N)
##   Smoothing parameters:
##     alpha = 0.999899
##     beta  = 0.6151203
##
##   Initial states:
##      l[0]     b[0]
##  516.8849 26.39527
##
##   sigma^2:  5e-04
##
##      AIC     AICc      BIC
## 763.6422 764.7960 773.9444
```

```
#forecasting to 5 years
us_etc_fc<- us_ets %>% forecast(h=5)

#plotting
us_etc_fc %>%  autoplot(us)+ggtitle('US GDP forecasted with ETS')
```



To compare the RSME between ETS and ARIMA model:

Ans: The ARIMA model with box_cox transformation produces a lower RMSE value and therefore is a better model to use. The un-transformed ARIMA model is also compared and the RMSE is still lower than the ETS model.

```
bind_rows(
us_fit_long %>% filter(models =='box_cox') %>%  accuracy(),
us_fit_long %>% filter(models =='original') %>%  accuracy(),
us_ets %>% accuracy())
```

```
## # A tibble: 3 x 11
##   models   .model      .type    ME  RMSE   MAE     MPE  MAPE  MASE RMSSE    ACF1
##   <chr>    <chr>       <chr> <dbl> <dbl> <dbl>   <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 box_cox  orders      Trai~ -2.95  147.  87.5 -0.0131  1.52 0.257 0.361  0.0232
## 2 original orders      Trai~ 36.8   156.  98.7  1.04    1.74 0.289 0.383 -0.0240
## 3 <NA>     ETS(GDP_bi~ Trai~ 18.6   167. 103.   0.585   1.67 0.302 0.410  0.0843
```