FPP Chapter 9 HW: ARIMA

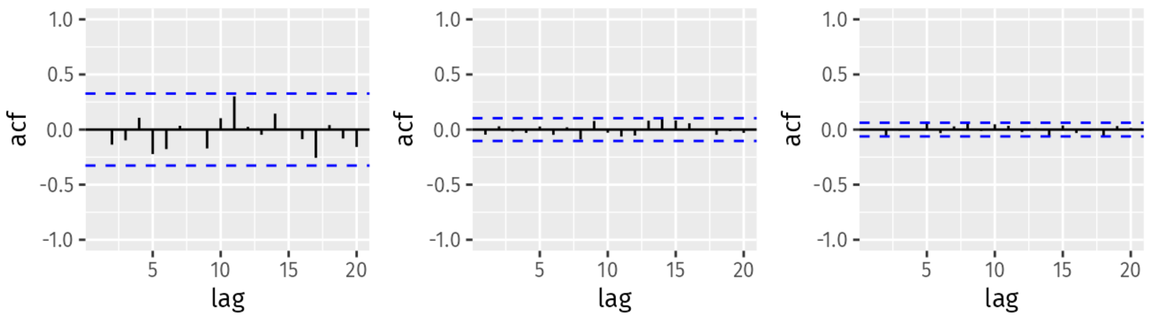
Alex Ptacek

```r
library(tidyverse)
library(fpp3)
```

Question 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?

Answer: The confidence interval decreases in width from left to right. In the same order, the autocorrelation stats also decrease in size. All data looks like white noise, because all of the autocorrelations fall inside the confidence interval.



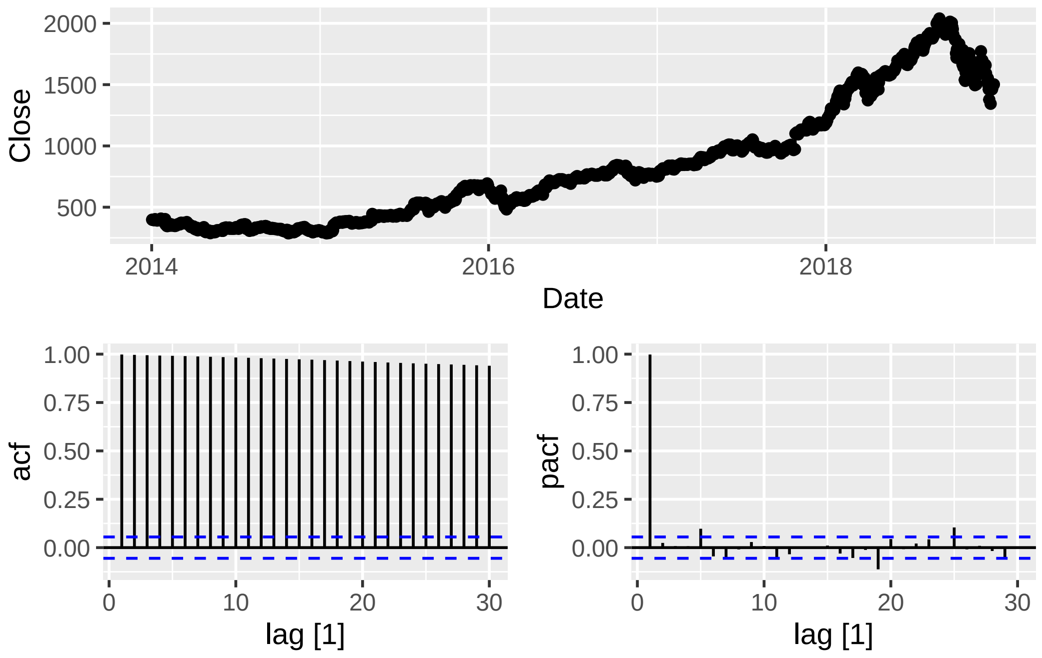
b. Why are the critical values at different distances from the mean of zero? Why are the autocorrelations different in each figure when they each refer to white noise?

Answer: The confidence interval varies based on the size of the sample, because it is included in the denometer of part of the formula - the larger the sample size, the smaller the interval. It can also change depending on the specified significance level - the higher the significance level, the smaller the interval. Individual autocorrelations can vary with sample size as well, because larger samples tend to have less random variation, and less random variation tends to result in lower autocorrelation.

Question 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.

Answer: The time series shows trend, the ACF plot shows significant autocorrelation that is decreasing very slowly, and the PACF shows significant partial autocorrelation at later lags. These are all clear signs that the data is non-stationary and, therefore, should be differenced.

```r
gafa_stock |>
  filter(Symbol == "AMZN") |>
  gg_tsdisplay(Close, plot_type = "partial")
```



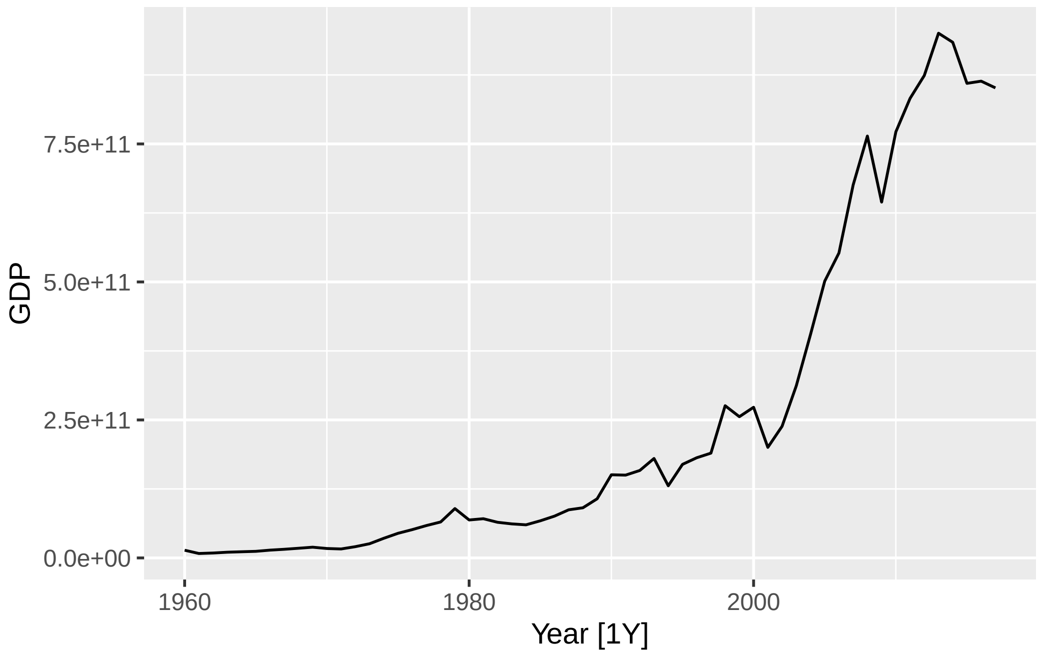
Question 9.3: For the following series, find an appropriate Box-Cox transformation and order of differencing in order to obtain stationary data.

a. Turkish GDP from global_economy.

Answer: Based on the below, we can use a Box-Cox transformation with lambda 0.157 (or probably get away with using log transform for simplicity). Then, we can make the data stationary with order 1 differencing. Yearly data does not have seasonality, so I did not test for seasonal differencing.

```r
turkish_econ <- global_economy |>
  filter(Country == "Turkey")

turkish_econ |>
  autoplot(GDP)
```



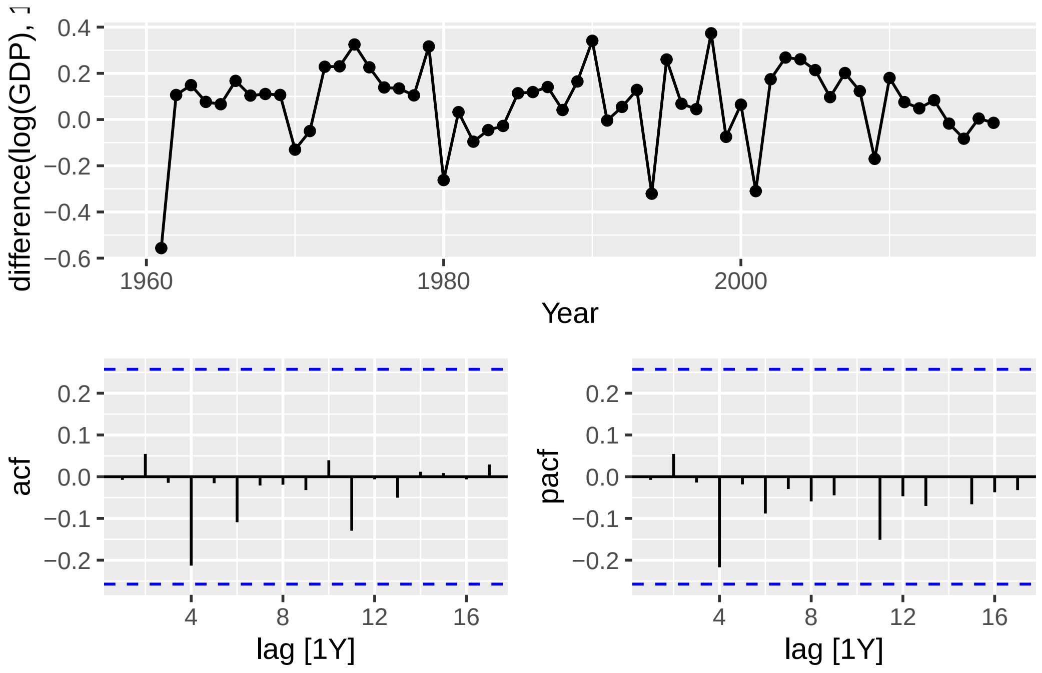
```r
turkish_econ |>
  features(GDP, guerrero)
```

```
# A tibble: 1 x 2
  Country lambda_guerrero
  <fct>              <dbl>
1 Turkey             0.157
```

```
turkish_econ |>
  features(log(GDP), unitroot_ndiffs)
```

```
# A tibble: 1 x 2
  Country ndiffs
  <fct>    <int>
1 Turkey       1
```

```
turkish_econ |>
  gg_tsdisplay(difference(log(GDP), 1), plot_type = "partial")
```

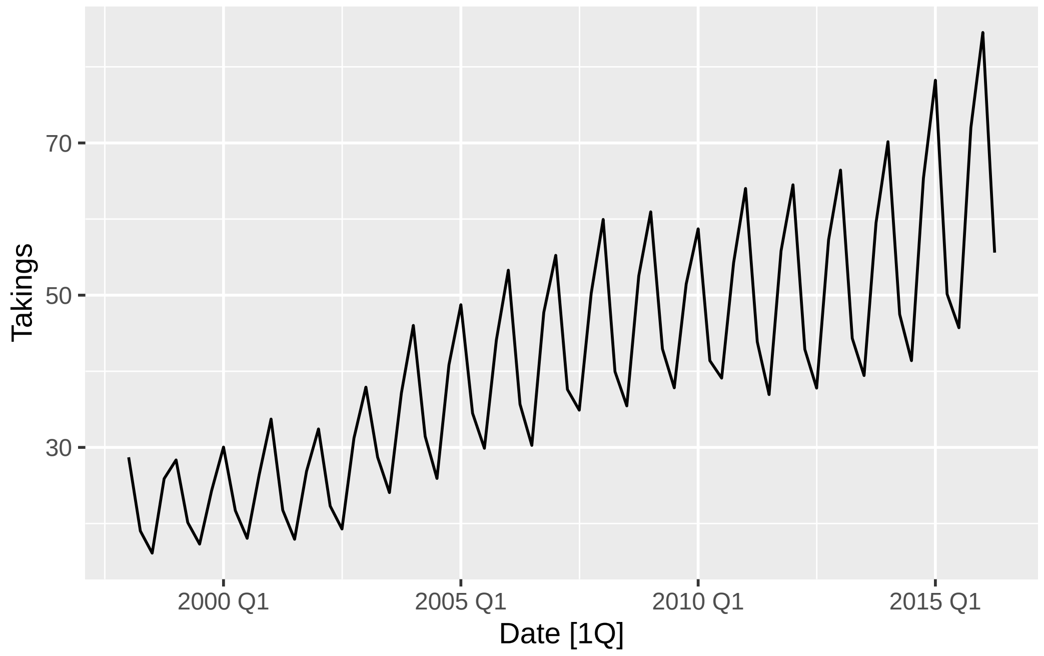


b. Accommodation takings in the state of Tasmania from aus_accommodation.

Answer: Here, the lambda found is so close to zero we can almost surely be comfortable using a log transform. Since the data is quarterly, I first tested for seasonal differencing and found that the data needs 1 seasonal differencing. After transforming the Takings variable, I then tested if additional differencing was needed and found that none was needed.

```r
tas_takings <- aus_accommodation |>
  filter(State == "Tasmania")

tas_takings |>
  autoplot(Takings)
```



```r
tas_takings |>
  features(Takings, guerrero)
```

```
# A tibble: 1 x 2
  State    lambda_guerrero
  <chr>              <dbl>
1 Tasmania         0.00182
```

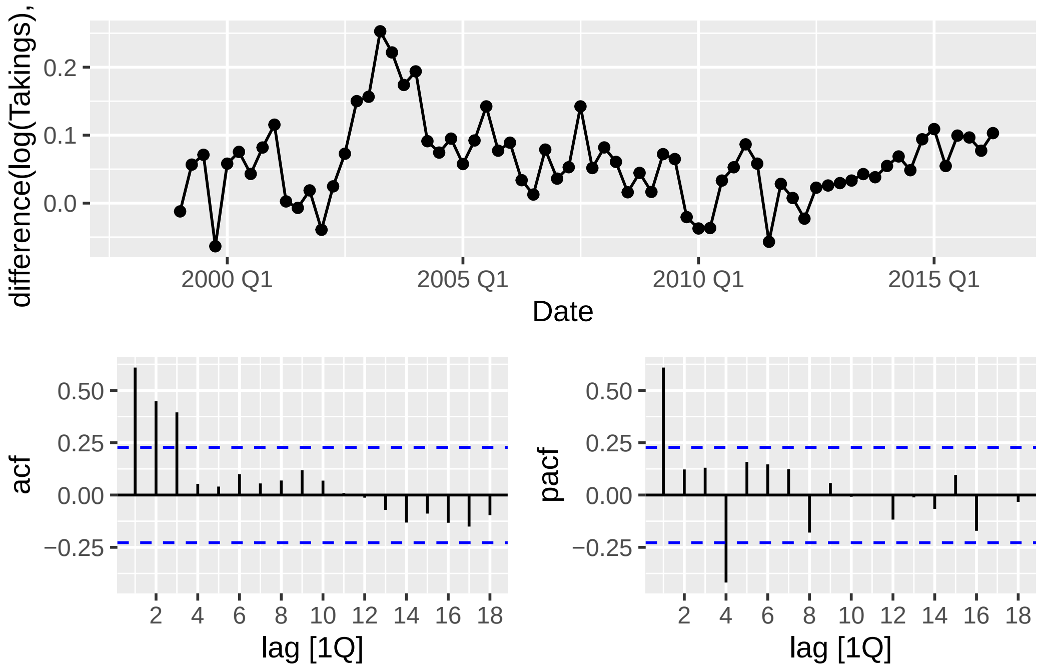
```r
tas_takings |>
  features(log(Takings), unitroot_nsdiffs)
```

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

```r
tas_takings |>
  mutate(adj_takings = difference(log(Takings), 4)) |>
  features(adj_takings, unitroot_ndiffs)
```

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

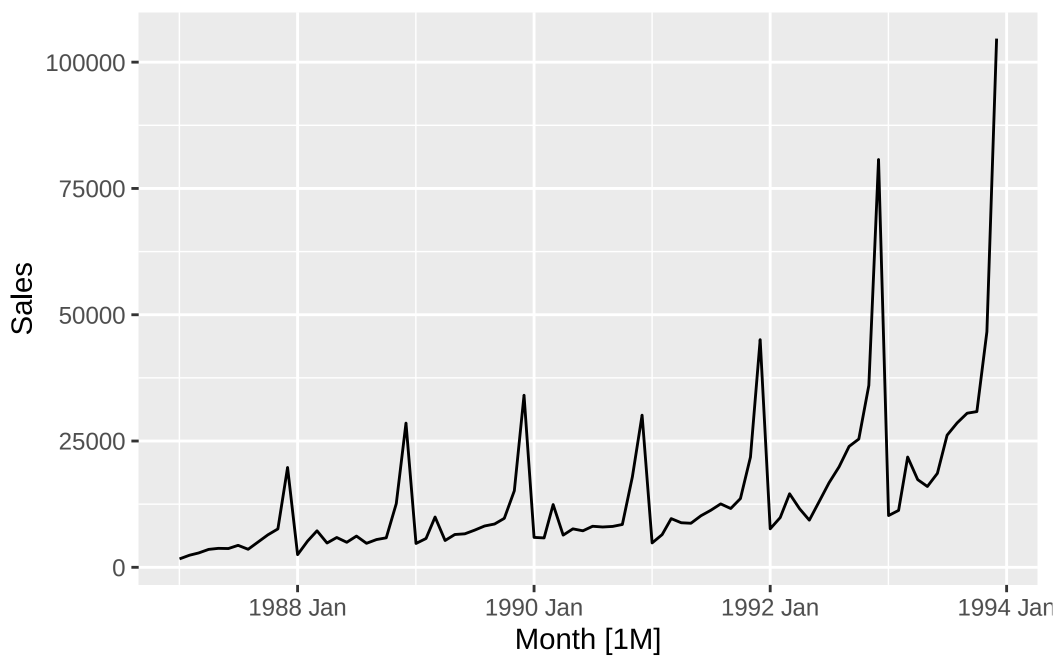
```r
tas_takings |>
  gg_tsdisplay(difference(log(Takings), 4), plot_type = "partial")
```



c. Monthly sales from souvenirs.

Answer: Same as above, the lambda for this time series is close enough to 0 to confidently use log transform, and the data needs 1 seasonal difference to become stationary.

```r
souvenirs |>
  autoplot(Sales)
```



```
souvenirs |>
  features(Sales, guerrero)
```

```
# A tibble: 1 x 1
  lambda_guerrero
            <dbl>
1         0.00212
```

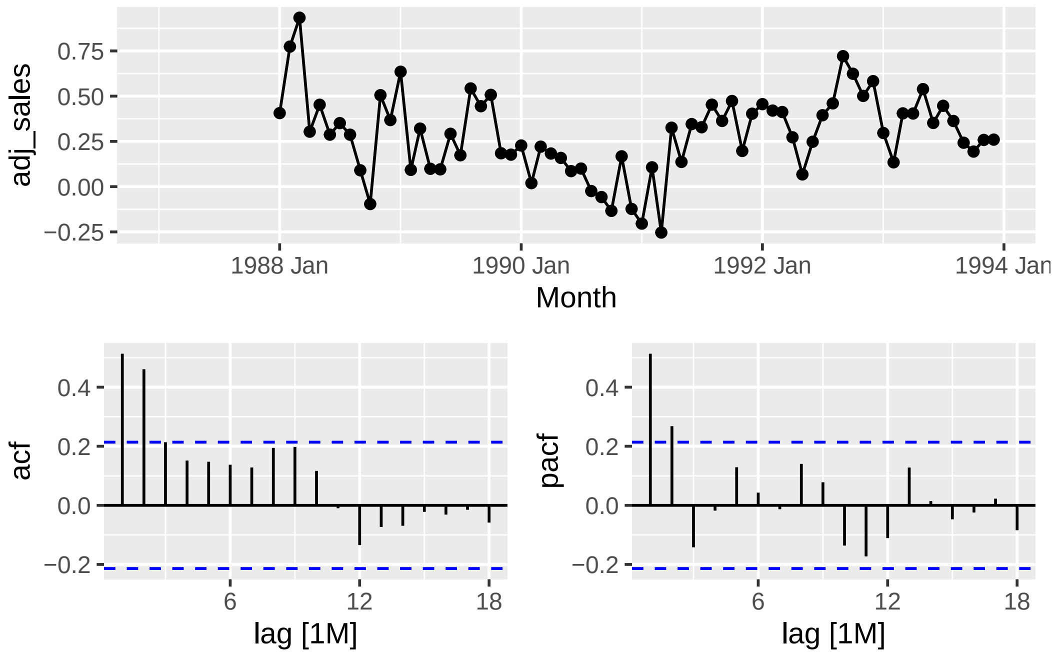
```
souvenirs |>
  features(log(Sales), unitroot_nsdiffs)
```

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

```
souvenirs |>
  mutate(adj_sales = difference(log(Sales), 12)) |>
  features(adj_sales, unitroot_ndiffs)
```

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

```
souvenirs |>
  mutate(adj_sales = difference(log(Sales), 12)) |>
  gg_tsdisplay(adj_sales, plot_type = "partial")
```

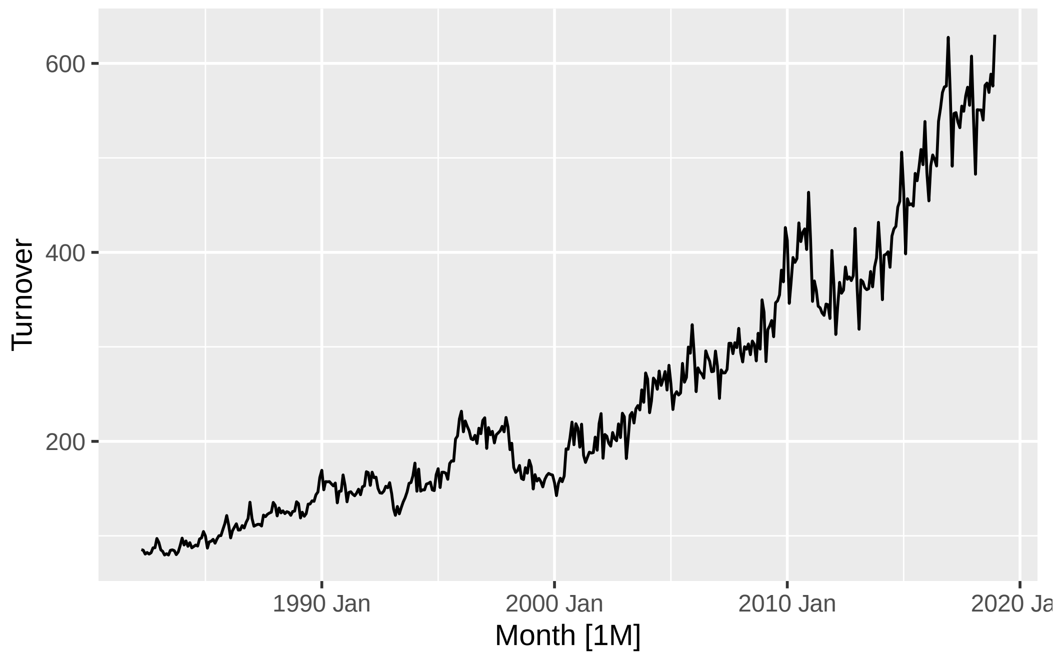


Question 9.5: For your retail data (from Exercise 7 in Section 2.10), find the appropriate order of differencing (after transformation if necessary) to obtain stationary data.

Answer: Based on the guerrero function and unit root tests, `Turnover` in `myseries` needs a log transform and order 1 seasonal difference to become stationary.

```
set.seed(624)
myseries <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`, 1))
```

```
myseries |>
  autoplot(Turnover)
```



```
myseries |>
  features(Turnover, guerrero)
```

```
# A tibble: 1 x 3
  State           Industry                 lambda_guerrero
  <chr>           <chr>                              <dbl>
1 New South Wales Takeaway food services           0.00214
```

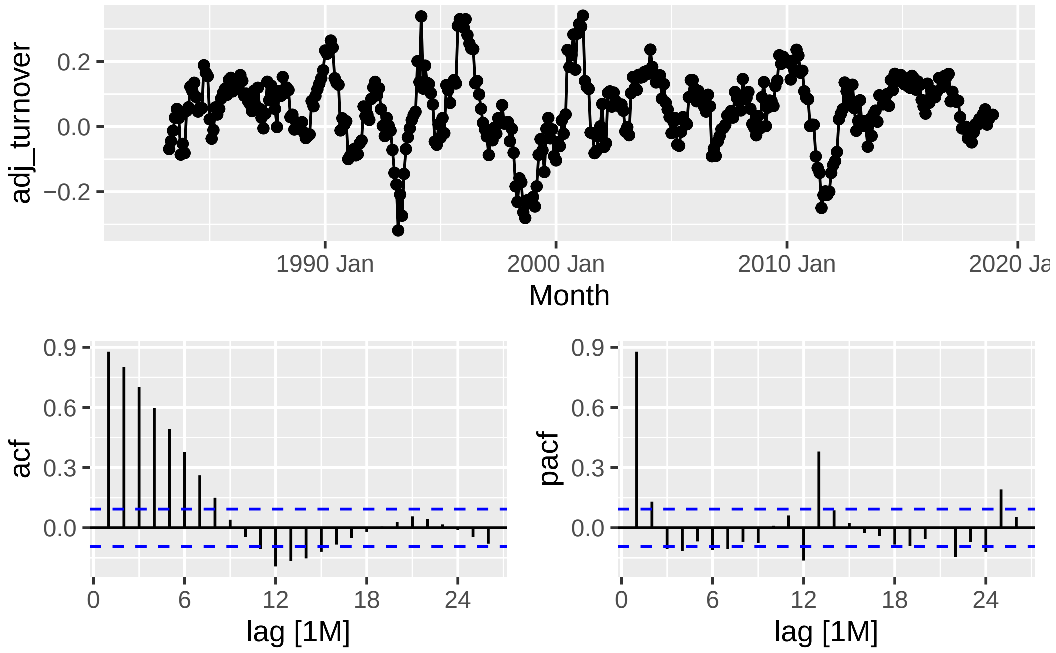
```
myseries |>
  features(log(Turnover), unitroot_nsdiffs)
```

```
# A tibble: 1 x 3
  State           Industry                 nsdiffs
  <chr>           <chr>                      <int>
1 New South Wales Takeaway food services         1
```

```
myseries |>
  mutate(adj_turnover = difference(log(Turnover), 12)) |>
  features(adj_turnover, unitroot_ndiffs)
```

```
# A tibble: 1 x 3
  State           Industry                 ndiffs
  <chr>           <chr>                     <int>
1 New South Wales Takeaway food services        0
```

```
myseries |>
  mutate(adj_turnover = difference(log(Turnover), 12)) |>
  gg_tsdisplay(adj_turnover, plot_type = "partial")
```



Question 9.6 Simulate and plot some data from simple ARIMA models.

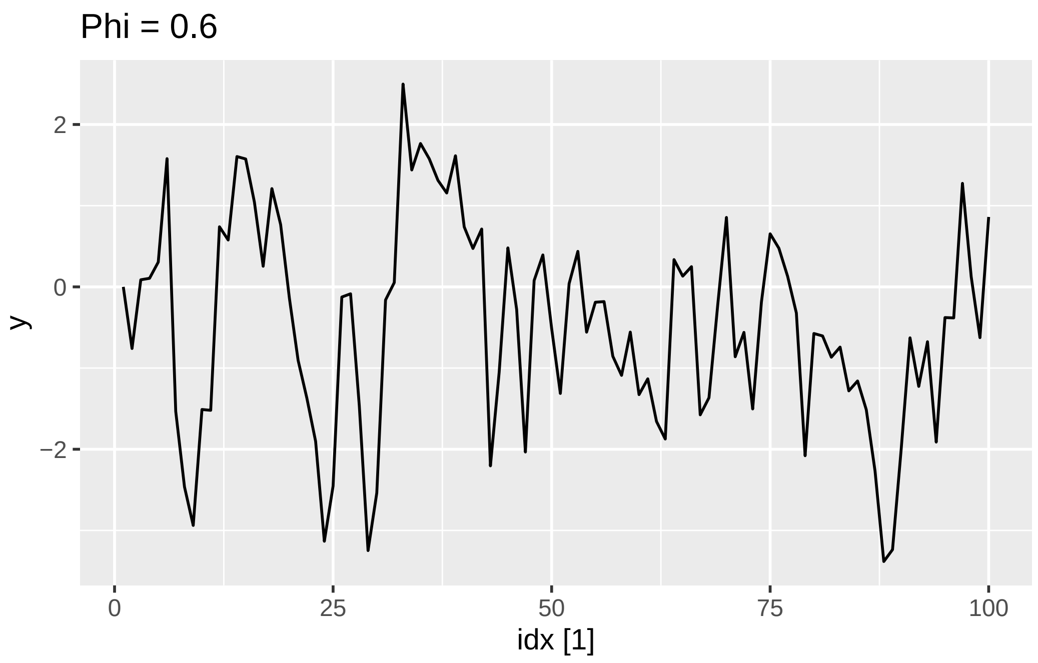
a. Use the following R code to generate data from an AR(1) model with 1 = 0.6 and 2 = 1. The process starts with y1 = 0.

```
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)
```

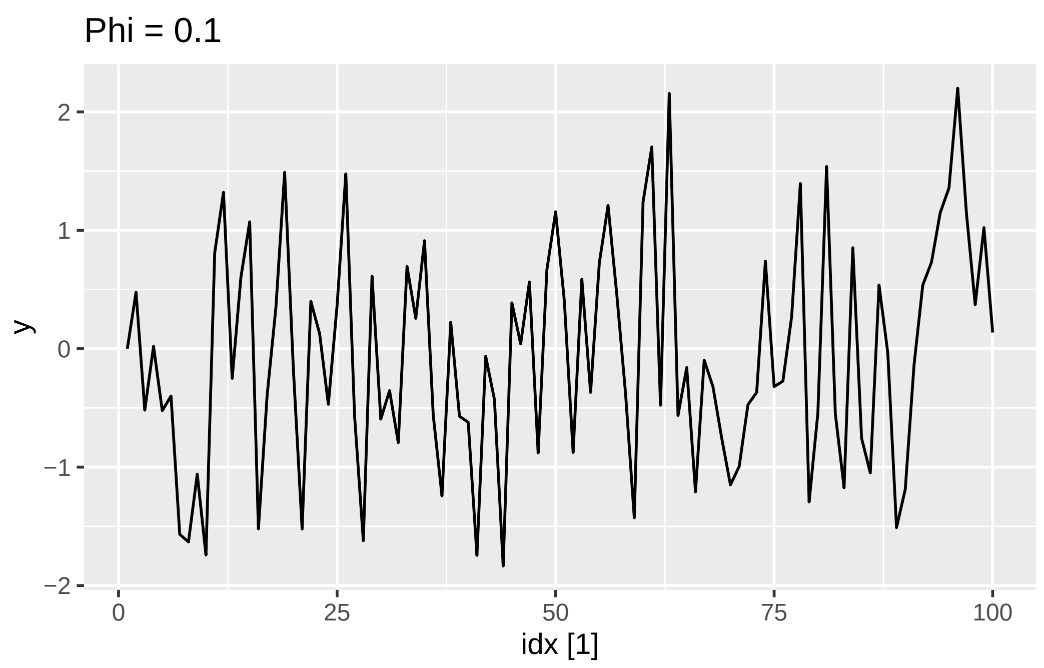
b. Produce a time plot for the series. How does the plot change as you change 1?

Answer: As phi decreases, random variation increases. This is because when phi decreases, less weight is given to the predictor, therefore giving more weight to the error term.

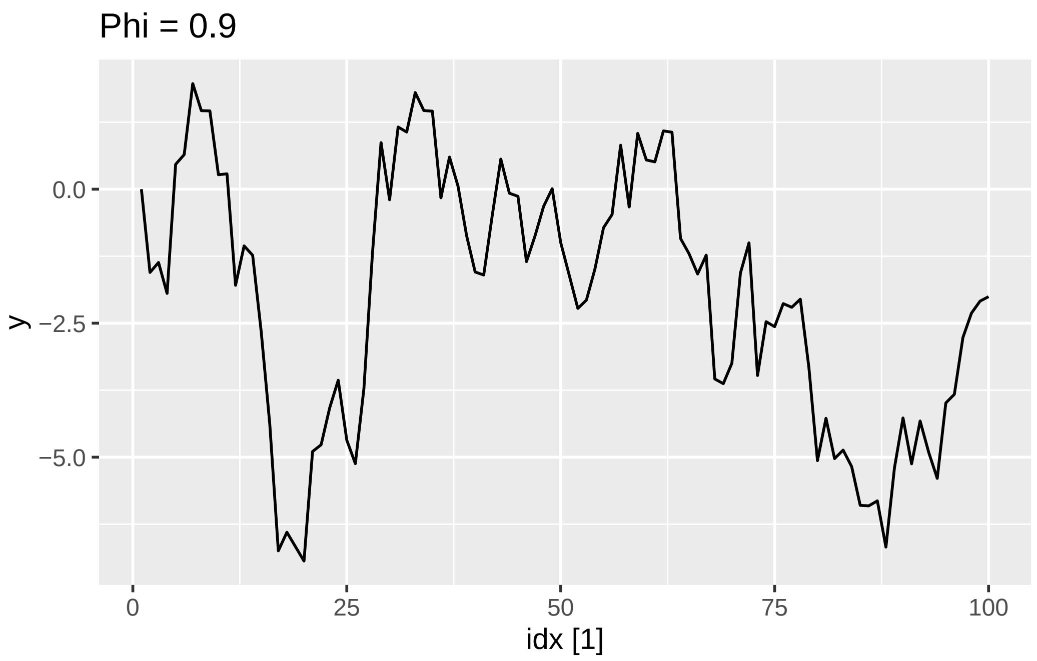
```
sim |>
  autoplot(y) + ggtitle("Phi = 0.6")
```



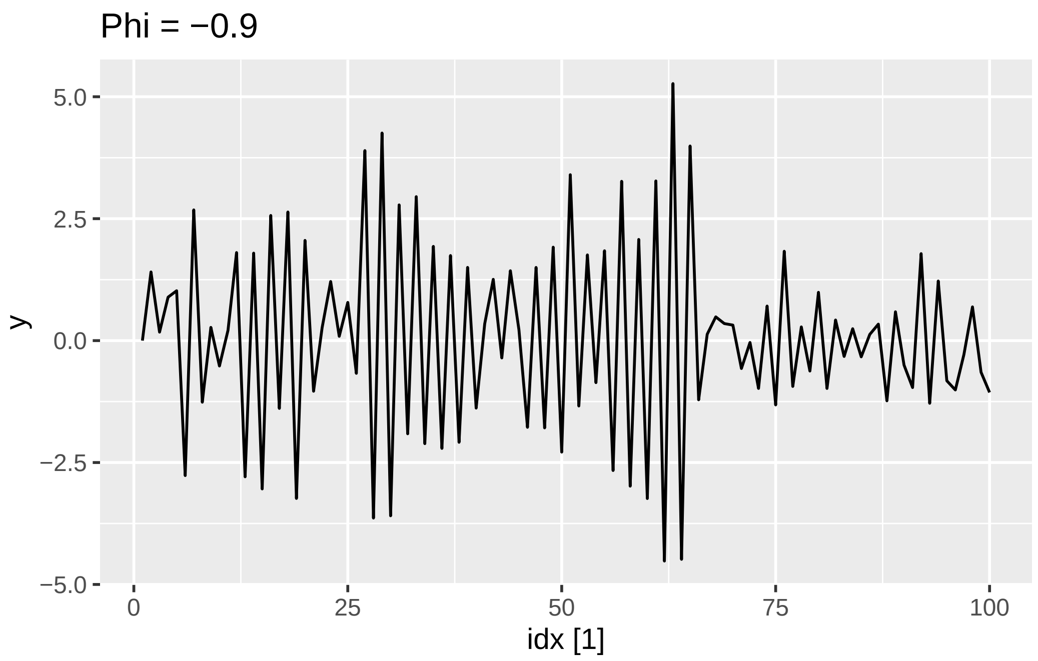
```
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- 0.1*y[i-1] + e[i]
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
sim |> autoplot(y) + ggtitle("Phi = 0.1")
```



```
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- 0.9*y[i-1] + e[i]
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
sim |> autoplot(y) + ggtitle("Phi = 0.9")
```



```
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- -0.9*y[i-1] + e[i]
sim <- tsibble(idx = seq_len(100), y = y, index = idx)
sim |> autoplot(y) + ggtitle("Phi = -0.9")
```



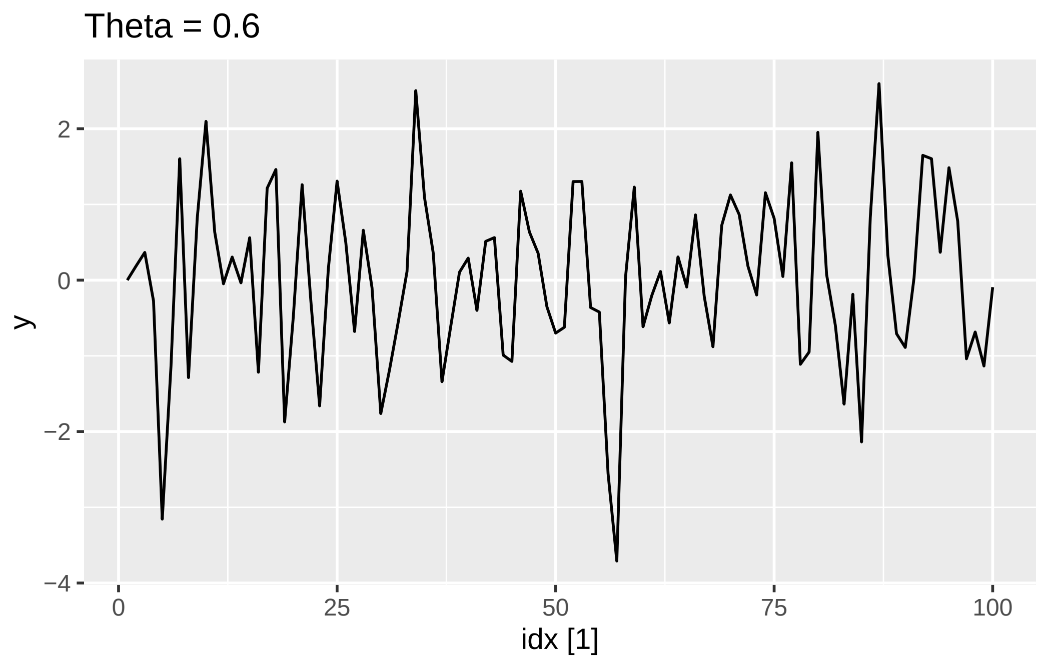
c. Write your own code to generate data from an MA(1) model with 1 = 0.6 and 2 = 1.

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

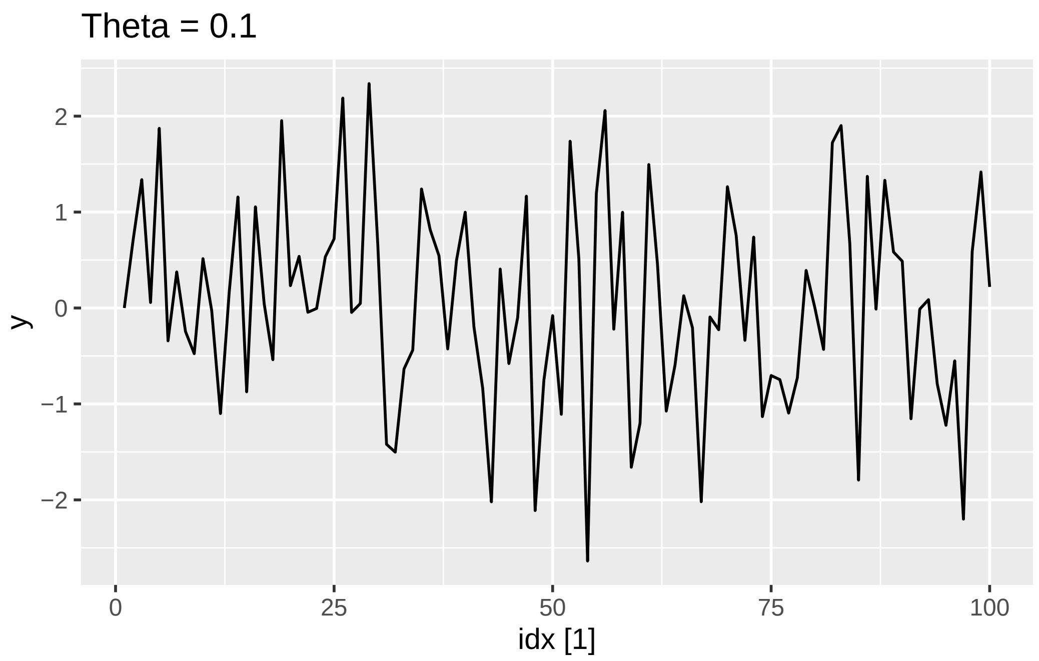
d. Produce a time plot for the series. How does the plot change as you change 1?

Answer: As theta increases, random variation decreases.

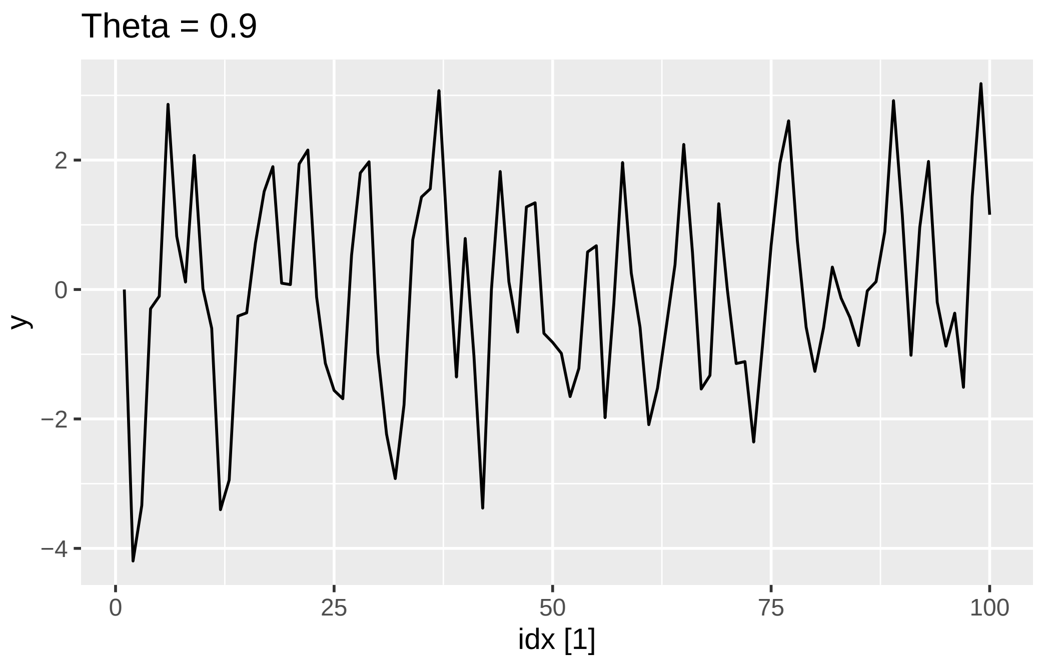
```
ma_sim |> autoplot(y) + ggtitle("Theta = 0.6")
```



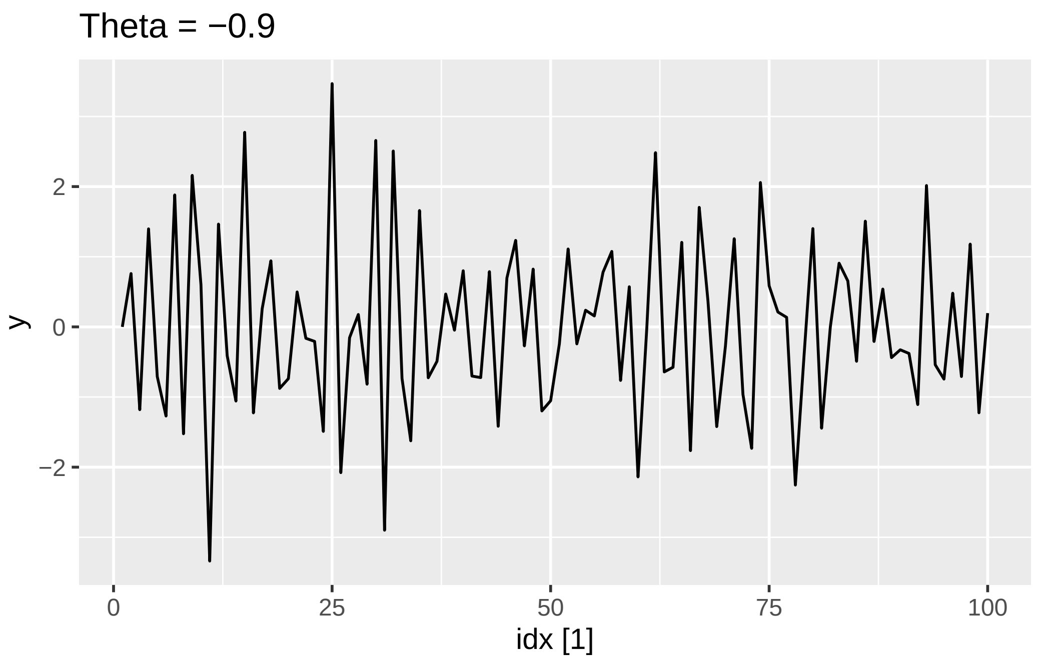
```r
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- 0.1*e[i-1] + e[i]
ma_sim <- tsibble(idx = seq_len(100), y = y, index = idx)
ma_sim |> autoplot(y) + ggtitle("Theta = 0.1")
```



```
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- 0.9*e[i-1] + e[i]
ma_sim <- tsibble(idx = seq_len(100), y = y, index = idx)
ma_sim |> autoplot(y) + ggtitle("Theta = 0.9")
```



```r
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- -0.9*e[i-1] + e[i]
ma_sim <- tsibble(idx = seq_len(100), y = y, index = idx)
ma_sim |> autoplot(y) + ggtitle("Theta = -0.9")
```



e. Generate data from an ARMA(1,1) model with 1 = 0.6, 1 = 0.6 and 2 = 1.

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

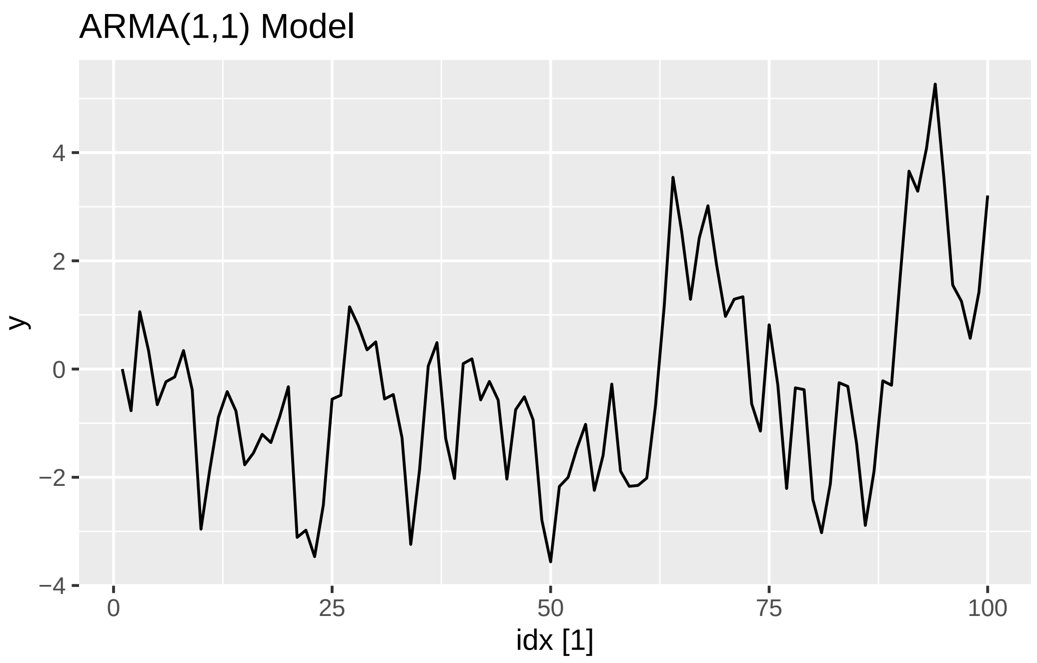
f. Generate data from an AR(2) model with 1 = −0.8, 2 = 0.3 and 2 =1. (Note that these parameters will give a non-stationary series.)

```
y <- numeric(100)
e <- rnorm(100)
y[2] <- -0.8*y[1] + e[1]
for(i in 3:100)
  y[i] <- -0.8*y[i-1] + 0.3*y[i-2] + e[i]
ar2_sim <- tsibble(idx = seq_len(100), y = y, index = idx)
```

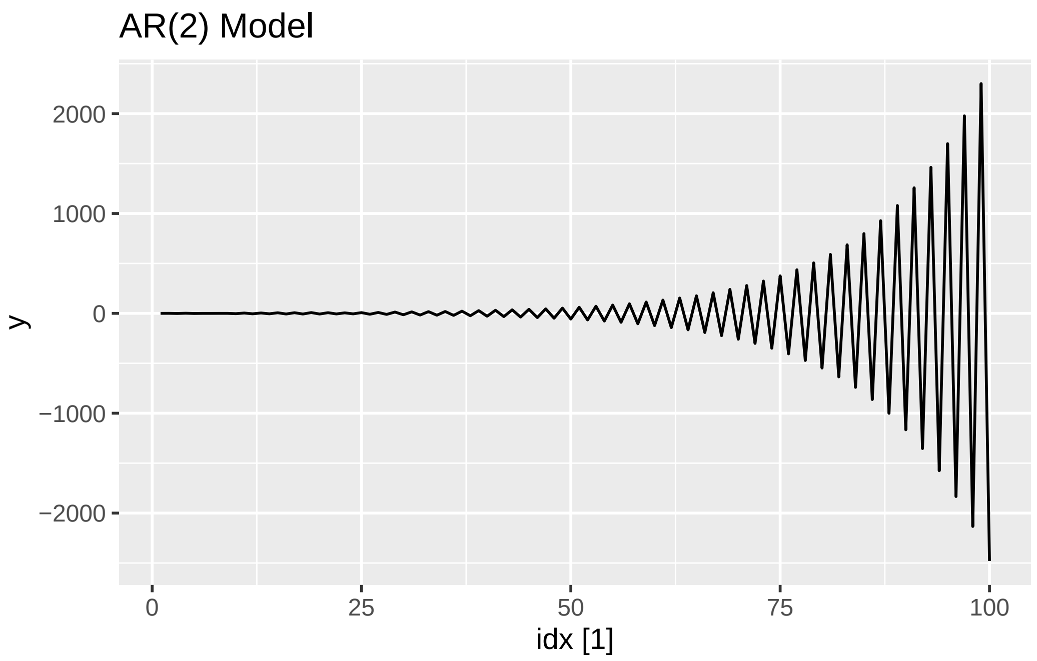
g. Graph the latter two series and compare them.

Answer: Both models fluctuate around 0, but the ARMA(1,1) model has constant variation, while the AR(2) model does not.

```r
arima_sim |> autoplot(y) + ggtitle("ARMA(1,1) Model")
```



```r
ar2_sim |> autoplot(y) + ggtitle("AR(2) Model")
```



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

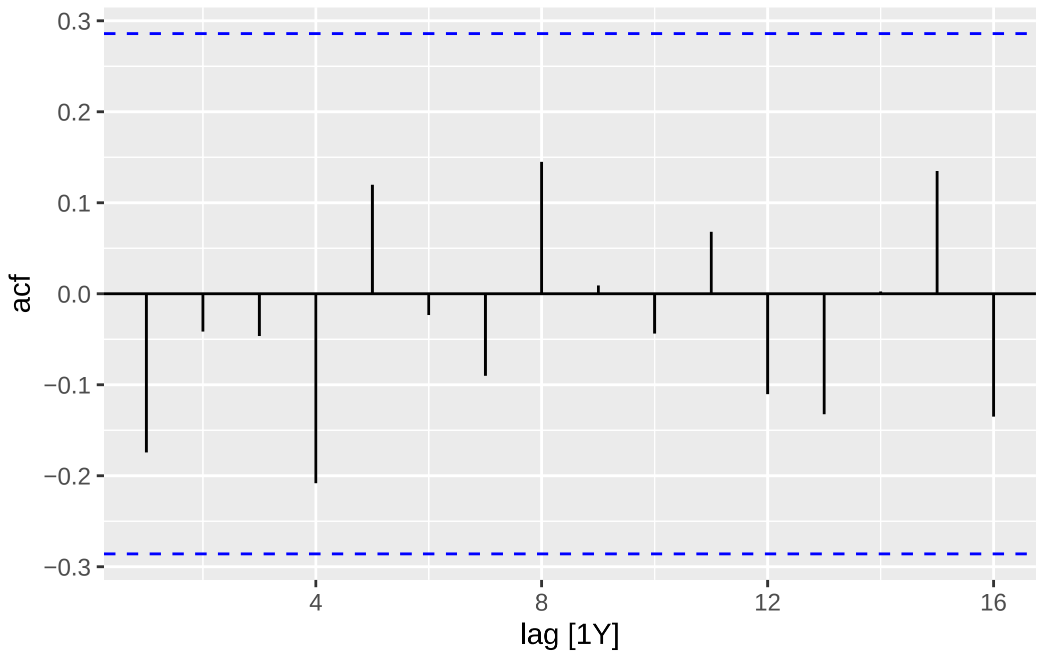
a. Use ARIMA() to find an appropriate ARIMA model. What model was selected. Check that the residuals look like white noise. Plot forecasts for the next 10 periods.

Answer: The auto-selected model is an ARIMA(0,2,1). The residuals look like white noise.

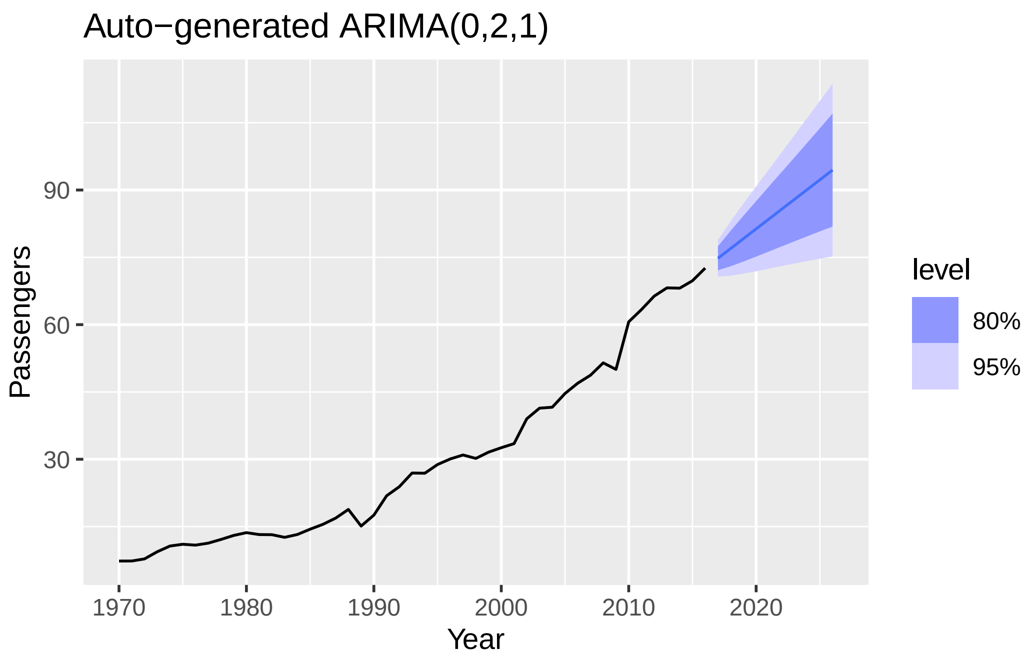
```
pass_auto_arima <- aus_airpassengers |>
  model(ARIMA(Passengers))
pass_auto_arima
```

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

```
#Check residuals
augment(pass_auto_arima) |>
  ACF(.innov) |> autoplot()
```



```r
#Plot forecast
pass_auto_arima |>
  forecast(h = 10) |>
  autoplot(aus_airpassengers) +
  ggtitle("Auto-generated ARIMA(0,2,1)")
```



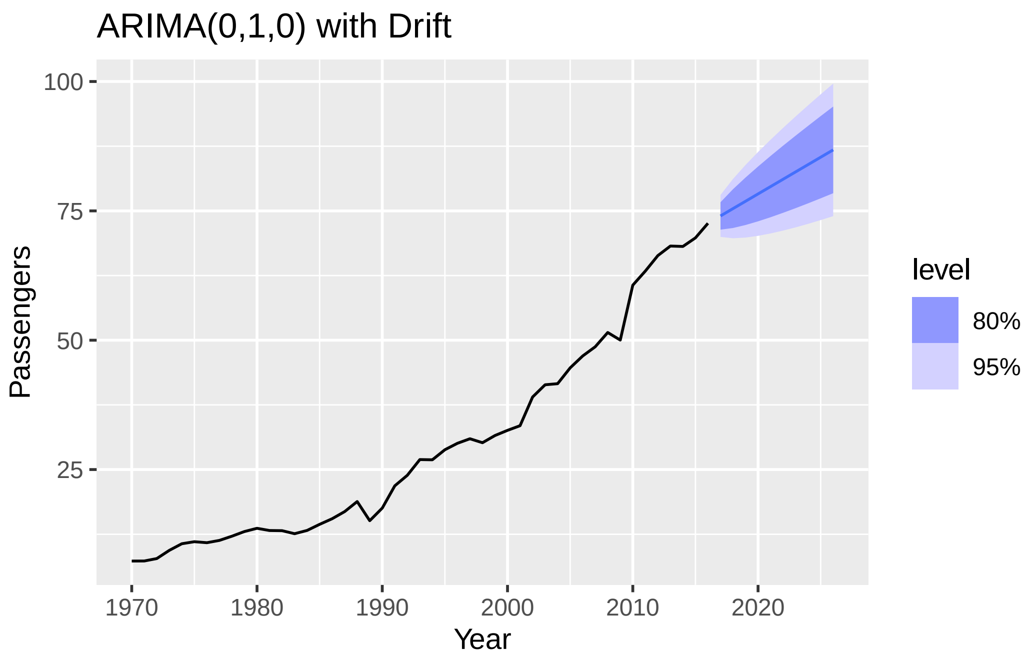
b. Write the model in terms of the backshift operator.

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

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

Answer: The (0,2,1) model has a steeper slope for its point forecasts compared to the (0,1,0) model. The (0,1,1) model also has curved prediction intervals compared to the straight lines of the (0,2,1) model.

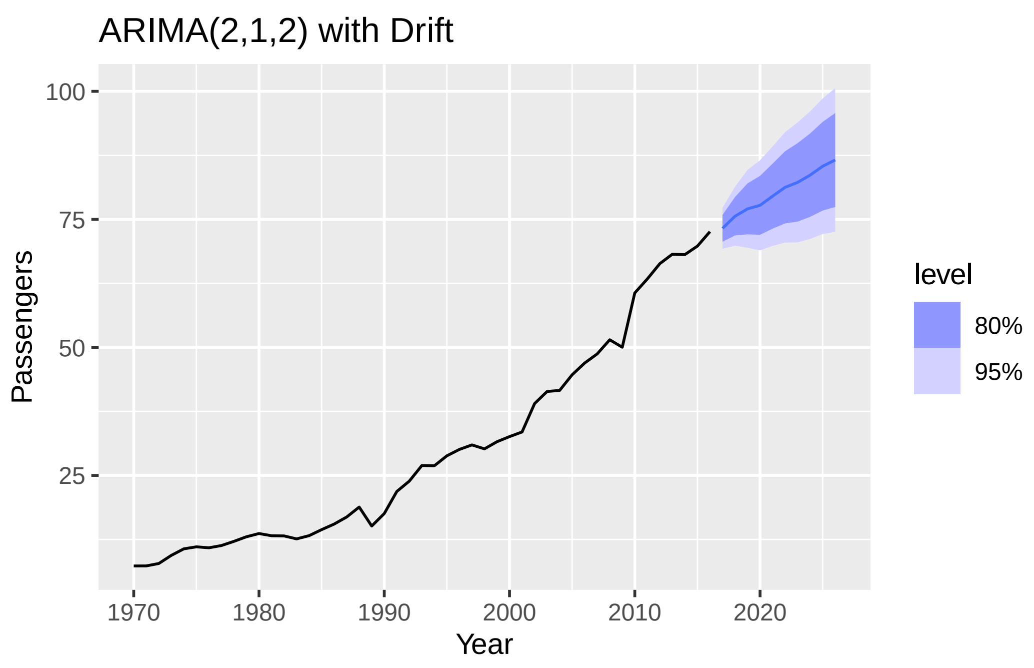
```
aus_airpassengers |>
  model(ARIMA(Passengers ~ 1 + pdq(0,1,0))) |>
  forecast(h = 10) |>
  autoplot(aus_airpassengers) +
  ggtitle("ARIMA(0,1,0) with Drift")
```



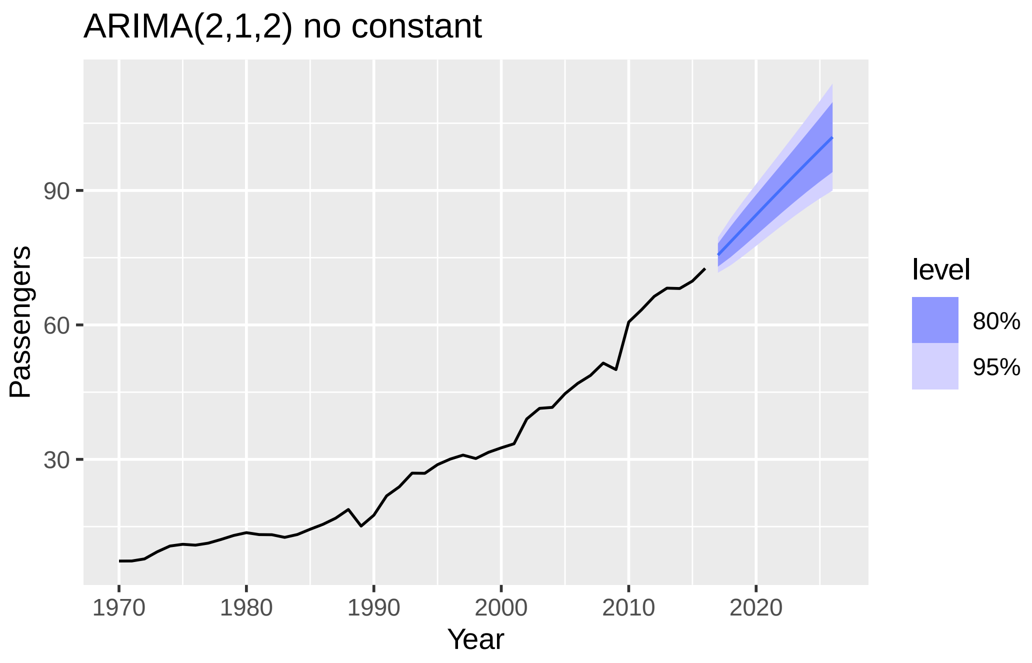
d. Plot forecasts from an ARIMA(2,1,2) model with drift and compare these to parts a and c. Remove the constant and see what happens.

Answer: The (2,1,2) model with Drift looks very similar to the (0,1,0) model with drift, but the point forecasts and prediction intervals are much more squiggly and the prediction intervals are wider. When we remove the constant from the (2,1,2) model, the point forecasts and prediction intervals straighten out, the slope is the steepest we've seen so far, and the prediction intervals are the narrowest we've seen so far.

```r
aus_airpassengers |>
  model(ARIMA(Passengers ~ 1 + pdq(2,1,2))) |>
  forecast(h = 10) |>
  autoplot(aus_airpassengers) +
  ggtitle("ARIMA(2,1,2) with Drift")
```



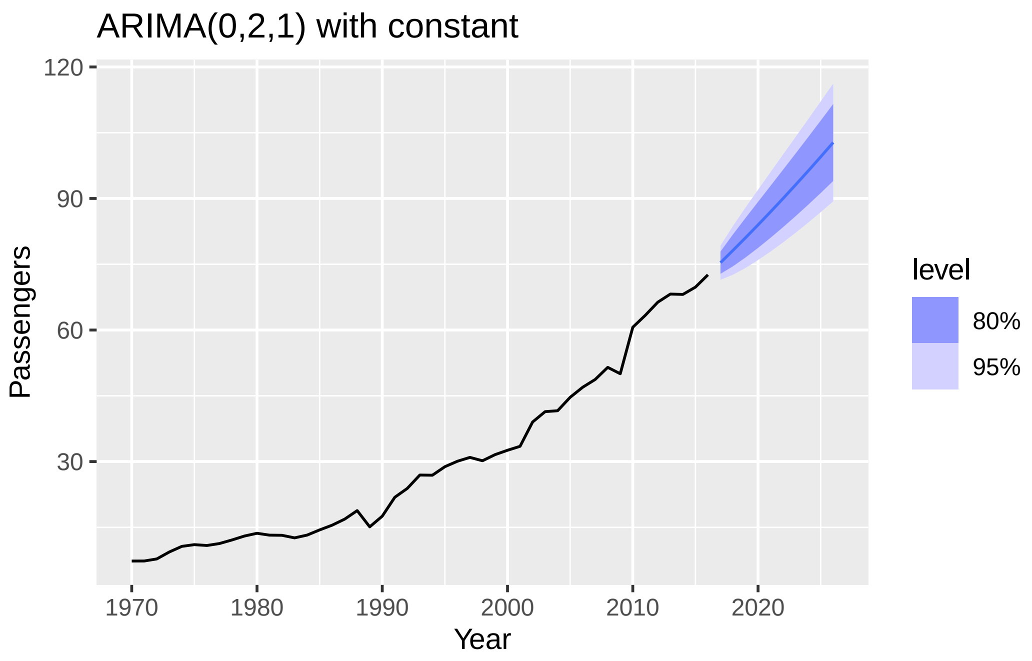
```r
aus_airpassengers |>
  model(ARIMA(Passengers ~ 0 + pdq(2,1,2), method = "ML")) |>
  forecast(h = 10) |>
  autoplot(aus_airpassengers) +
  ggtitle("ARIMA(2,1,2) no constant")
```



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

Answer: Compared to the auto-generated (0,2,1) model, this model has steeper point forecast slope, narrower prediction intervals, and a quadratic trend.

```
aus_airpassengers |>
  model(ARIMA(Passengers ~ 1 + pdq(0,2,1))) |>
  forecast(h = 10) |>
  autoplot(aus_airpassengers) +
  ggtitle("ARIMA(0,2,1) with constant")
```



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

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

```
us_gdp <- global_economy |>
  filter(Country == "United States")

us_gdp |>
  features(GDP, guerrero)
```

```
# A tibble: 1 x 2
  Country       lambda_guerrero
  <fct>                   <dbl>
1 United States           0.282
```

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

Answer: fable has automatically chosen an ARIMA(1,1,0) model with Drift.

```
gdp_auto_arima <- us_gdp |>
  model(ARIMA(box_cox(GDP, 0.282)))
gdp_auto_arima
```

```
# A mable: 1 x 2
# Key:     Country [1]
  Country       `ARIMA(box_cox(GDP, 0.282))`
  <fct>                              <model>
1 United States      <ARIMA(1,1,0) w/ drift>
```

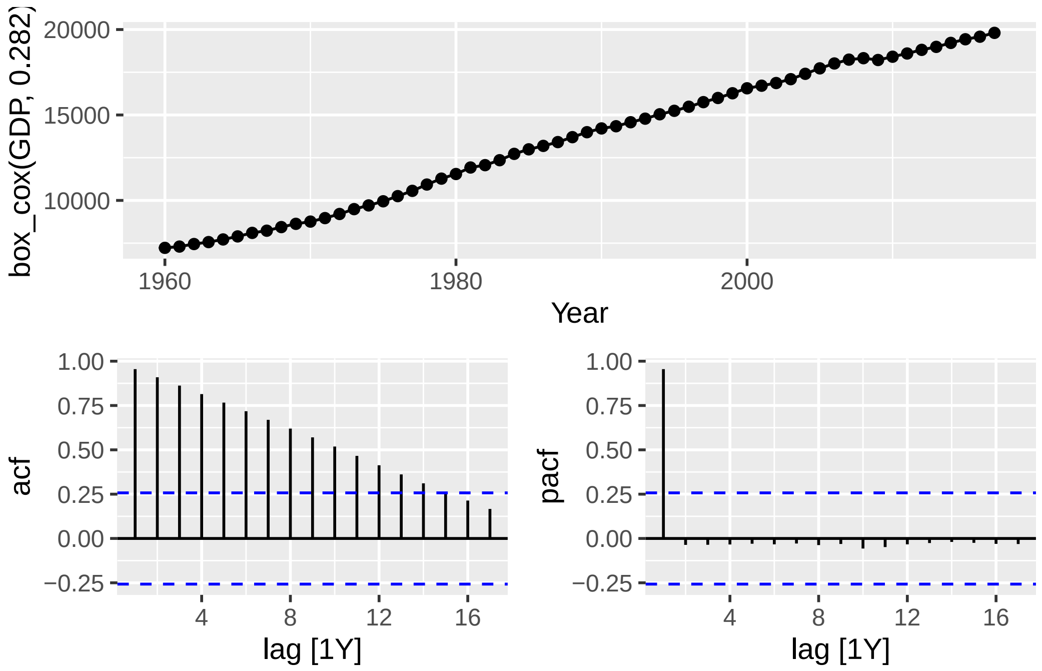
c. try some other plausible models by experimenting with the orders chosen;

Answer: I started by using a unit root test and examining the ACF and PACF plots, and I came to same conclusion that fable did automatically, so I just experimented with changing all of parameters by 1 (except d). Lastly, I tried increasing the computational power of `ARIMA()` with some additional arguments and it still recommended the same model as the original auto-generated one.

```r
# Check for differencing
us_gdp |>
  features(box_cox(GDP, 0.282), unitroot_ndiffs)
```

```
# A tibble: 1 x 2
  Country       ndiffs
  <fct>          <int>
1 United States      1
```

```r
# Check ACF and PACF for plausible p/q orders
us_gdp |>
  gg_tsdisplay(box_cox(GDP, 0.282), plot_type = "partial")
```



```r
# Experiment with p + 1
plus_p <- us_gdp |>
  model(ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(2,1,0)))
plus_p
```

```
# A mable: 1 x 2
# Key:     Country [1]
  Country       `ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(2, 1, 0))`
  <fct>                                                 <model>
1 United States                            <ARIMA(2,1,0) w/ drift>
```

```r
# q + 1
plus_q <- us_gdp |>
  model(ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(1,1,1)))
plus_q
```

```
# A mable: 1 x 2
# Key:     Country [1]
  Country       `ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(1, 1, 1))`
  <fct>                                                 <model>
1 United States                            <ARIMA(1,1,1) w/ drift>
```

```
# remove drift
no_drift <- us_gdp |>
  model(ARIMA(box_cox(GDP, 0.282) ~ 0 + pdq(1,1,0)))
no_drift
```

```
# A mable: 1 x 2
# Key:     Country [1]
  Country       `ARIMA(box_cox(GDP, 0.282) ~ 0 + pdq(1, 1, 0))`
  <fct>                                                 <model>
1 United States                                   <ARIMA(1,1,0)>
```

```
# raise compute power
max_compute <- us_gdp |>
  model(ARIMA(box_cox(GDP, 0.282), stepwise = FALSE, approx = FALSE,
              order_constraint = p + q <= 9))
max_compute
```

```
# A mable: 1 x 2
# Key:     Country [1]
  Country       ARIMA(box_cox(GDP, 0.282), stepwise = FALSE, approx = FALSE, \~1
  <fct>                                                                  <model>
1 United States                                          <ARIMA(1,1,0) w/ drift>
# i abbreviated name:
#   1: `ARIMA(box_cox(GDP, 0.282), stepwise = FALSE, approx = FALSE, \n    order_constraint =
```

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

Answer: I believe the best performing model is the ARIMA(1,1,0) with drift model, which was the auto-generated model by fable. I based this selection on AICc and the MASE from test forecasts, for which the auto-generated model had the lowest of both. Lastly, based on the residual diagnostics, we can see that the residuals are white noise and not significantly correlated, which satisfies the conditions.

```
# Compare AIC
glance(gdp_auto_arima) |> select(.model, AICc)
```

```
# A tibble: 1 x 2
  .model                      AICc
  <chr>                      <dbl>
1 ARIMA(box_cox(GDP, 0.282))  657.
```

```r
glance(plus_p) |> select(.model, AICc)
```

```
# A tibble: 1 x 2
  .model                                          AICc
  <chr>                                          <dbl>
1 ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(2, 1, 0))  660.
```

```r
glance(plus_q) |> select(.model, AICc)
```

```
# A tibble: 1 x 2
  .model                                          AICc
  <chr>                                          <dbl>
1 ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(1, 1, 1))  660.
```

```r
glance(no_drift) |> select(.model, AICc)
```

```
# A tibble: 1 x 2
  .model                                          AICc
  <chr>                                          <dbl>
1 ARIMA(box_cox(GDP, 0.282) ~ 0 + pdq(1, 1, 0))  672.
```

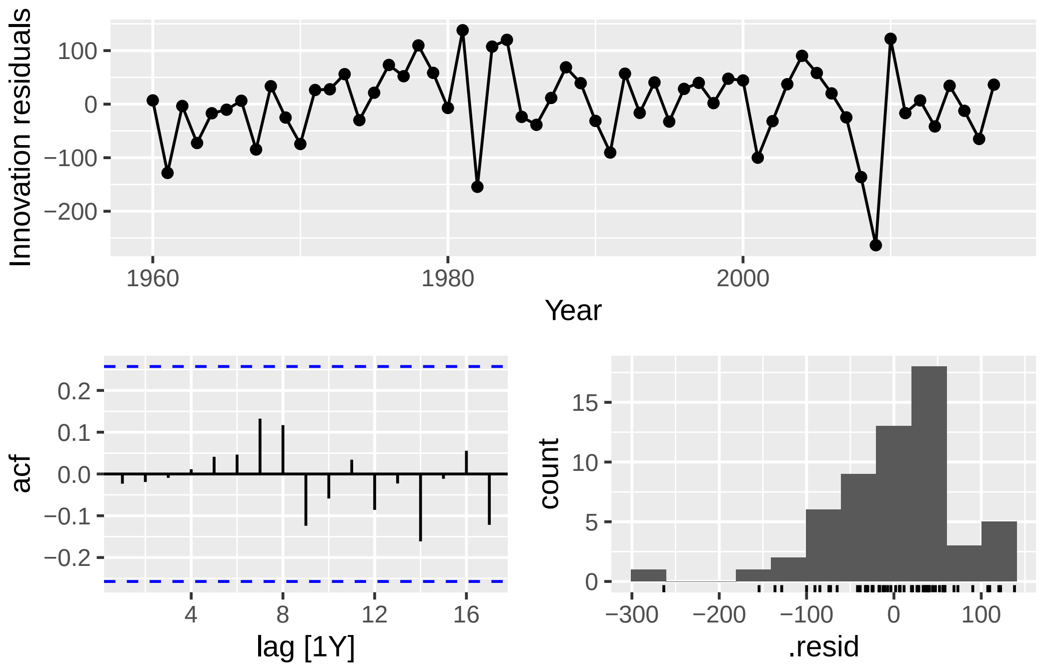
```r
# Train models on all but 7 years of data
all_models_fit <- us_gdp |>
  filter(Year <= 2010) |>
  model(auto = ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(1,1,0)),
        plus_p = ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(2,1,0)),
        plus_q = ARIMA(box_cox(GDP, 0.282) ~ 1 + pdq(1,1,1)),
        no_drift = ARIMA(box_cox(GDP, 0.282) ~ 0 + pdq(1,1,0)))

# Compare accuracy measures of our models on test set
all_models_fit |>
  forecast(h = 7) |>
  accuracy(us_gdp) |>
  select(.model, MASE)
```

```
# A tibble: 4 x 2
  .model    MASE
  <chr>    <dbl>
1 auto     0.770
```

```
2 no_drift 1.84
3 plus_p   0.986
4 plus_q   1.03
```

```
# Examine the residual diagnostics for selected model
gg_tsresiduals(gdp_auto_arima)
```



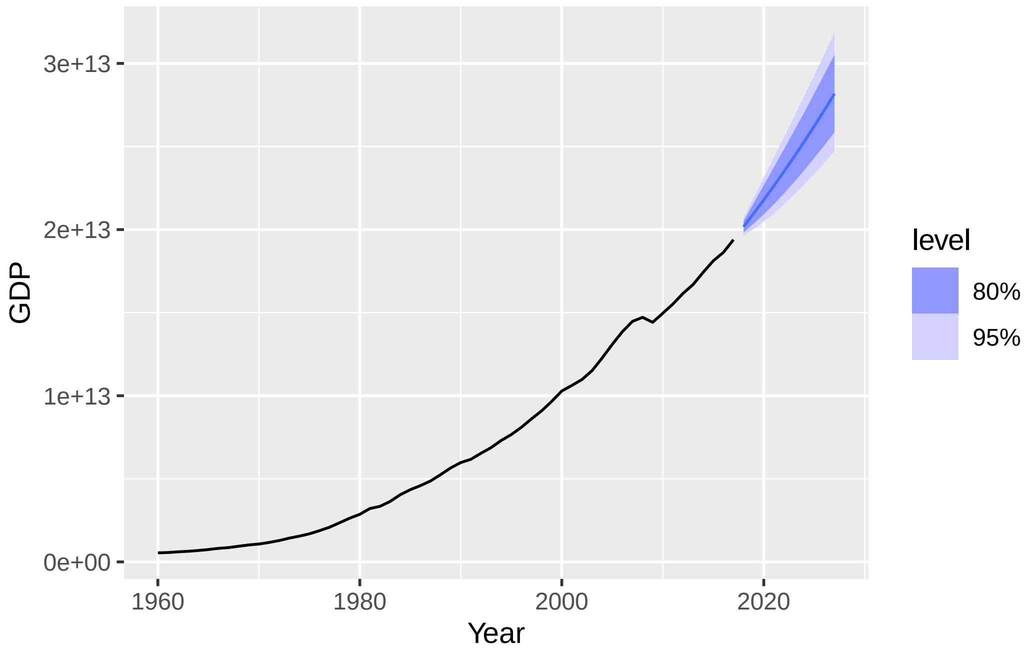
```
augment(gdp_auto_arima) |>
  features(.innov, ljung_box, lag = 10, dof = 1)
```

```
# A tibble: 1 x 4
  Country       .model                          lb_stat lb_pvalue
  <fct>         <chr>                             <dbl>     <dbl>
1 United States ARIMA(box_cox(GDP, 0.282))         3.81     0.923
```

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

Answer: I believe the forecasts look reasonable because it follows a clear trend in our y variable and the prediction intervals are quite narrow.

```
gdp_auto_arima |>
  forecast(h = 10) |>
  autoplot(us_gdp)
```



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

Answer: Compared to the auto-generated ARIMA(1,1,0) with drift model, the auto-generated ETS model has higher AICc and MASE, and the predictiion intervals are wider. I would choose the ARIMA model.

```
ets_gdp <- us_gdp |>
  model(ETS(GDP))

glance(ets_gdp) |> select(.model, AICc)
```

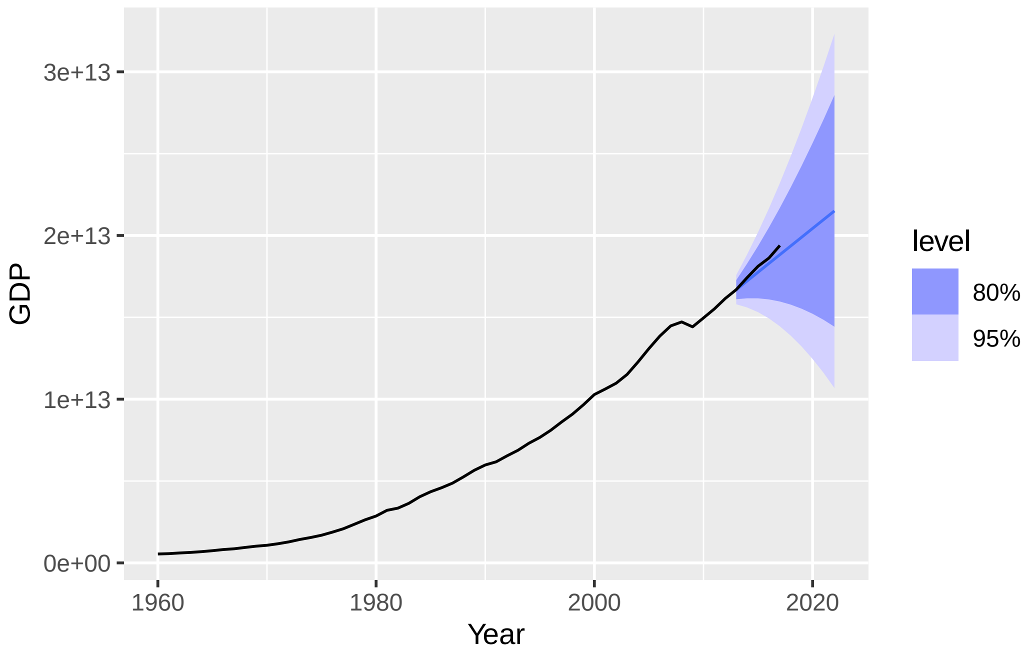
```
# A tibble: 1 x 2
  .model    AICc
  <chr>    <dbl>
1 ETS(GDP) 3192.
```

```r
final_test <- us_gdp |>
  filter(Year <= 2012) |>
  model(ets = ETS(GDP))

final_test |>
  forecast(h = 5) |>
  accuracy(us_gdp)
```

```
# A tibble: 1 x 11
  .model Country      .type       ME     RMSE      MAE   MPE  MAPE  MASE RMSSE  ACF1
  <chr>  <fct>        <chr>    <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 ets    United Sta~ Test   2.91e11 3.45e11 2.91e11  1.57  1.57 0.935 0.920 0.191
```

```r
final_test |>
  forecast(h = 10) |>
  autoplot(us_gdp)
```


3e+13
2e+13
1e+13
0e+00
GDP
Year
level