Homework Six for Fall 2024 DATA 624: ARIMA

Kevin Kirby

2024-10-08

Overview

The below are answers to exercises 9.1, 9.2, 9.3, 9.5, 9.6, 9.7, 9.8 from the exercise section of chapter 9 of [Forecasting: Principles and Practice (3rd ed)] (https://otexts.com/fpp3/graphics-exercises.html). This is homework six of the DATA 624 class "Predictive Analytics & Forecasting." Unless otherwise noted, all datasets used below are from the fpp3 package that's owned and maintained by the book's authors.

First, I'll load the required libraries:

9.1: Questions about ACFs from provided images

The exercise states: "Figure 9.32 shows the ACFs for 36 random numbers, 360 random numbers and 1,000 random numbers."

There are two component questions:

A. Explain the differences among these figures. Do they all indicate that the data are 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?

I've answered each component below the picture.

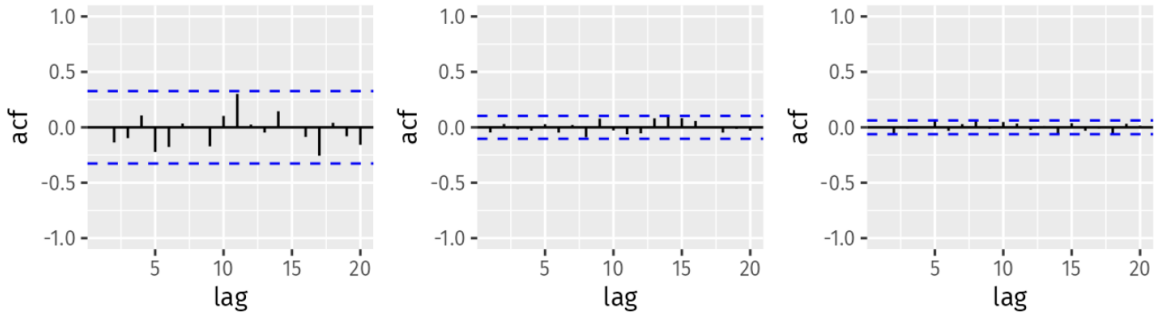


Figure 9.32: Left: ACF for a white noise series of 36 numbers. Middle: ACF for a white noise series of 360 numbers. Right: ACF for a white noise series of 1,000 numbers.

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

36 Numbers: The ACF shows a decent amount of variability, with several spikes that fall outside the significance bounds. This is most likjely caused by the small sample size, which leads to more random fluctuations. However, the series still likely represents white noise because the autocorrelations are random instead of concerted.

360 Numbers: This ACF is much smoother compared to the 36 numbers one and there's less popping out of

the significance bounds, with the majority of values clustered around zero. This means that the dataset is behaving much more closely to the expeted behavior of white noise. The larger sample size has produced a more reliable datset.

1,000 Numbers: This ACF is the. most consistent, with very little deviation from the zero line. This is how I would expect a white noise pattern to lookpattern for a white noise series. The large sample size gives a tighter view of the lack of autocorrelation that characterizes white noise.

All three charts do represent white noise but the larger sample size charts provides cleaner indicators of what makes white noise behave like white noise.

9.1.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 are at different distances from the mean of zero because a larger or smaller sample size impacting the estimation variability. Less samples means much weight given to deviated data points that could, by random chance, populate a smaller sample size. Their wewight is reduced when more samples are brought in and any given point has less impact.

The autocorrelations appear different in each plot because each white noise series is independently generated, with smaller sample sizes having more varability due to the same reason above. A larger sample size give more opportunity for the ACF to represent the independence of the data.

9.2 Working with Amazon stock from gafa_stuck

This exercise states: "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)
```

```
## 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     1.1.5
## v dplyr       1.1.4     v tsibbledata 0.4.1
## v tidyr       1.3.1     v feasts      0.4.0
## v lubridate   1.9.3     v fable       0.4.0
## 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(tidyverse)
```

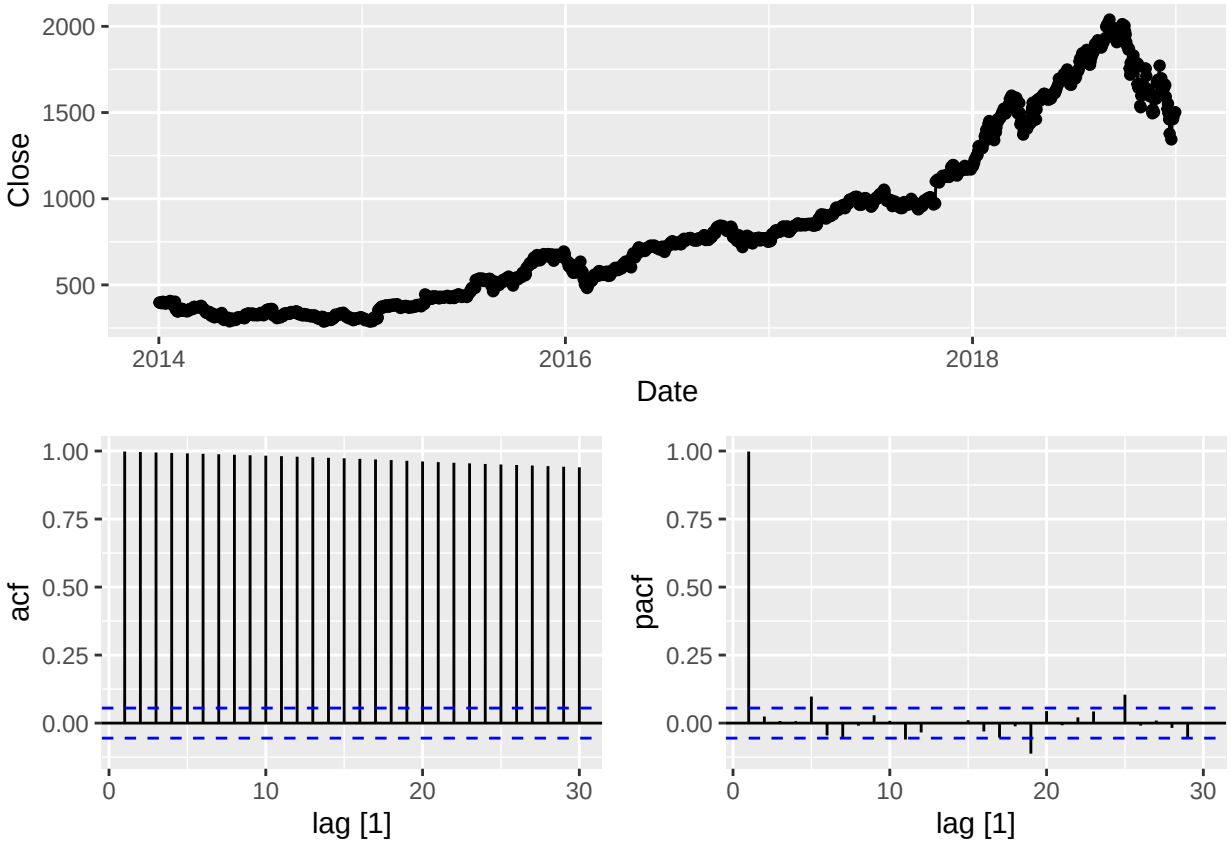
```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v forcats 1.0.0     v readr   2.1.5
## v purrr   1.0.2     v stringr 1.5.1
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
```

```
## x dplyr::filter()     masks stats::filter()
## x tsibble::interval() masks lubridate::interval()
## x dplyr::lag()        masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
library(ggplot2)

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



The time series plot shows a strong trend, which is indicative on a non-stationary dataset.The ACF plot shows signs of slow decay, which leads me to believe that the data has persistent autocorrelation. This type of autocorrelation also indicates a non-stationary series.

The PACF plot has an intense spike at lag 1, which means that the current value needs to be not be dependent on the previous value. This another evidence point for non-stationary. suggesting the need to remove the dependence of the current value on the previous value, further implying non-stationarity. I would reccommend differencing this dataset by removing the value at the time point one year ago from the current value. This would help make the data stationary.

9.3 Box-Cox transformations and differencing on selected data

This exercise states: 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 B. Accommodation takings in the state of Tasmania from aus_accommodation C. Monthly sales from souvenirs

9.3.A Turkish GDP from global_economy

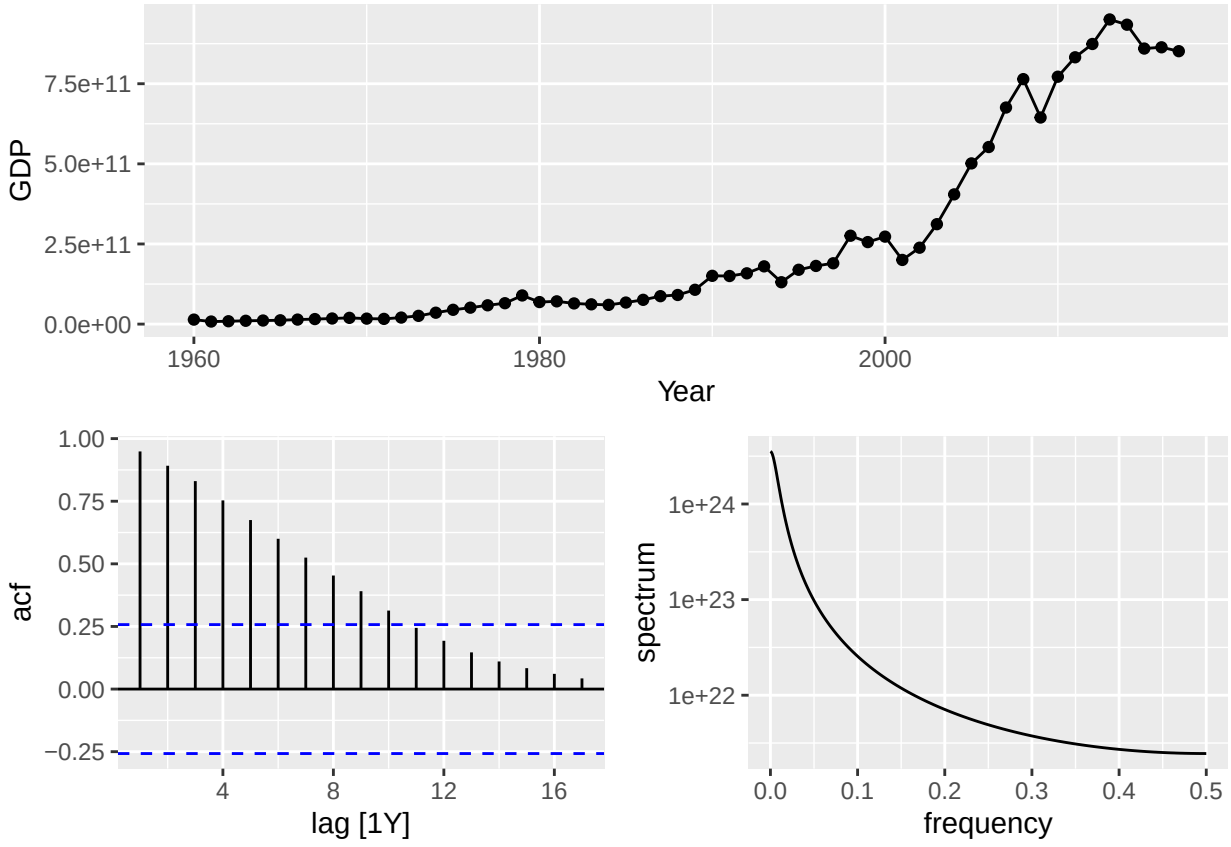
```
library(dplyr)
library(fabletools)
library(feasts)
library(tsibble)
library(scales)
```

```
##
## Attaching package: 'scales'
```

```
## The following object is masked from 'package:purrr':
##
##     discard
```

```
## The following object is masked from 'package:readr':
##
##     col_factor
```

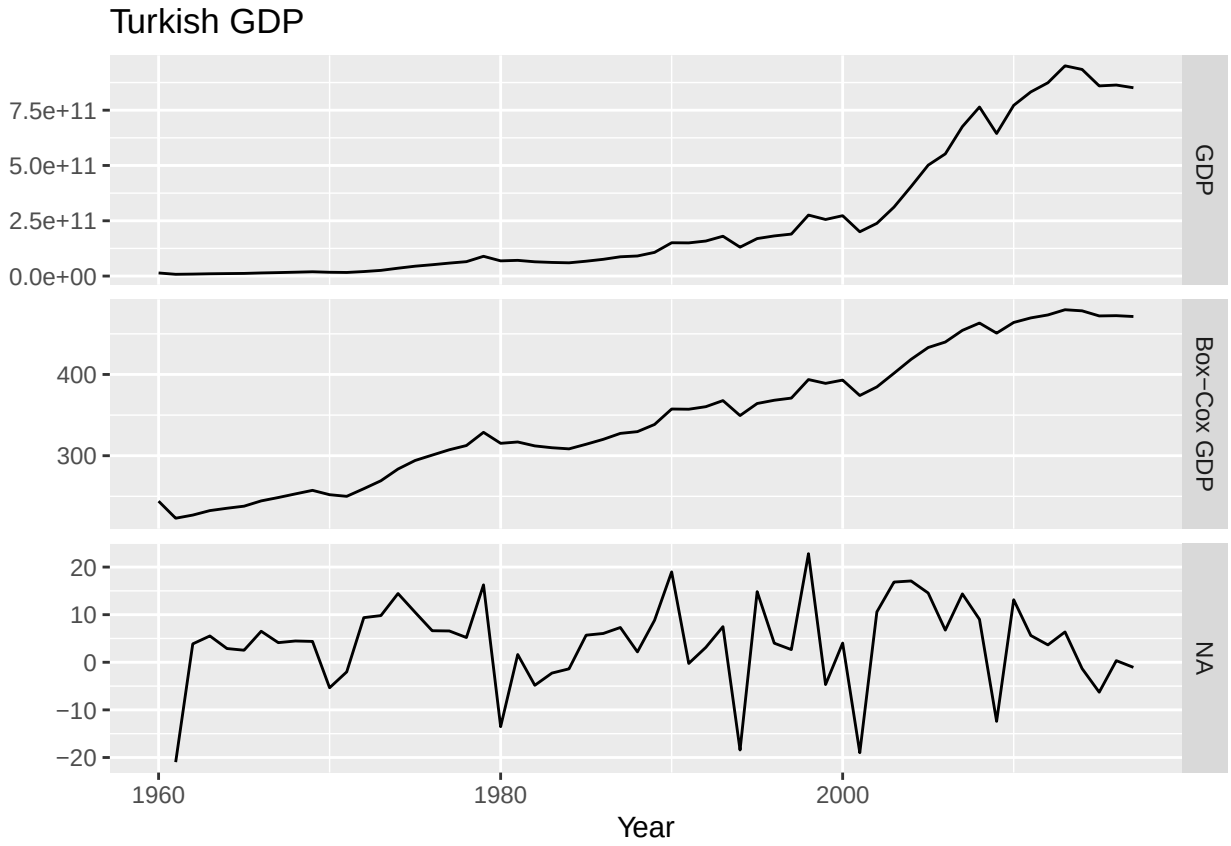
```
turk_economy <- global_economy |>
  filter(Country == "Turkey")

turk_economy |>
  gg_tsdisplay(GDP)
```



```r
turk_lambda <- turk_economy |>
  features(GDP, features = guerrero) |>
  pull(lambda_guerrero)

turk_nsdiffs <- turk_economy |>
  features(box_cox(GDP, turk_lambda), unitroot_nsdiffs) |>
  pull(nsdiffs)

turk_ndiffs <- turk_economy |>
  features(box_cox(GDP, turk_lambda), unitroot_ndiffs) |>
  pull(ndiffs)

cat("Optimal lambda: ", turk_lambda,
    "\nRequired seasonal differences: ", turk_nsdiffs,
    "\nRequired first differences: ", turk_ndiffs)
```

```
## Optimal lambda:  0.1572187
## Required seasonal differences:  0
## Required first differences:  1
```

```r
turk_economy |>
  transmute(
    `GDP` = GDP,
    `Box-Cox GDP` = box_cox(GDP, turk_lambda),
    `First Differenced Box-Cox GDP` = difference(box_cox(GDP, turk_lambda), turk_ndiffs)
    ) |>
  pivot_longer(-Year, names_to="Type", values_to="GDP") |>
  mutate(
    Type = factor(Type, levels = c(
      "GDP",
      "Box-Cox GDP",
      "Box-Cox GDP for Turkey - First difference applied"
      ))
  ) |>
  ggplot(aes(x = Year, y = GDP)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
  labs(title = "Turkish GDP", y = NULL)
```

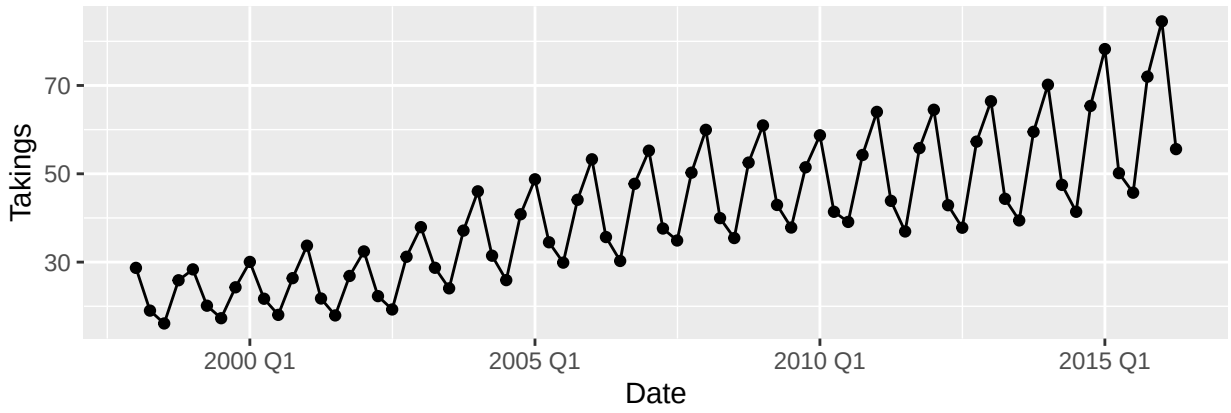


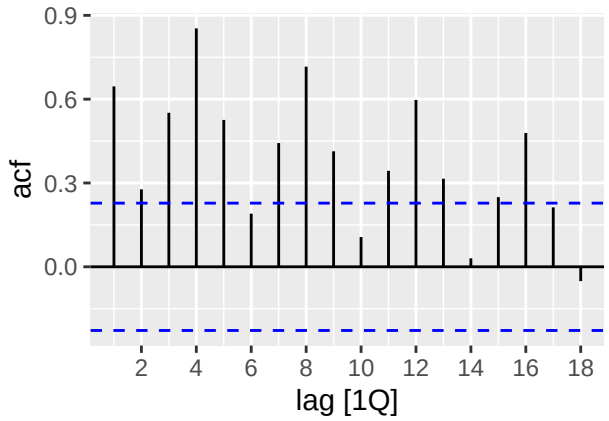
Optimal lambda: 0.1572187 Required seasonal differences: 0 Required first differences: 1

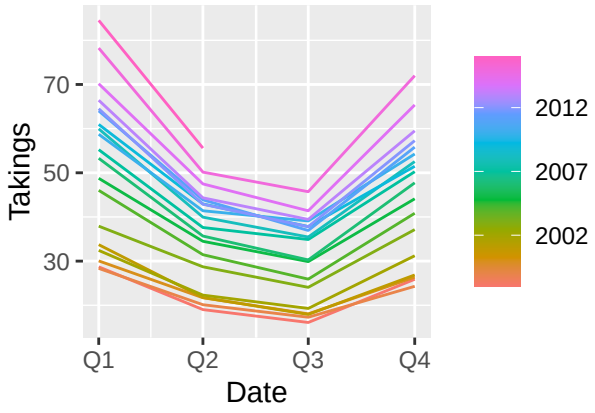
Stationary data is achieved with no seasonal difference and one first difference. This means the GDP of Turkey has non-stationary behavior that can be minimized and/or removed by subtracking the previous year value from the current value.

9.3.B - Accommodation takings in the state of Tasmania from aus_accommodation

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

tas_takings |>
  gg_tsdisplay(Takings)
```





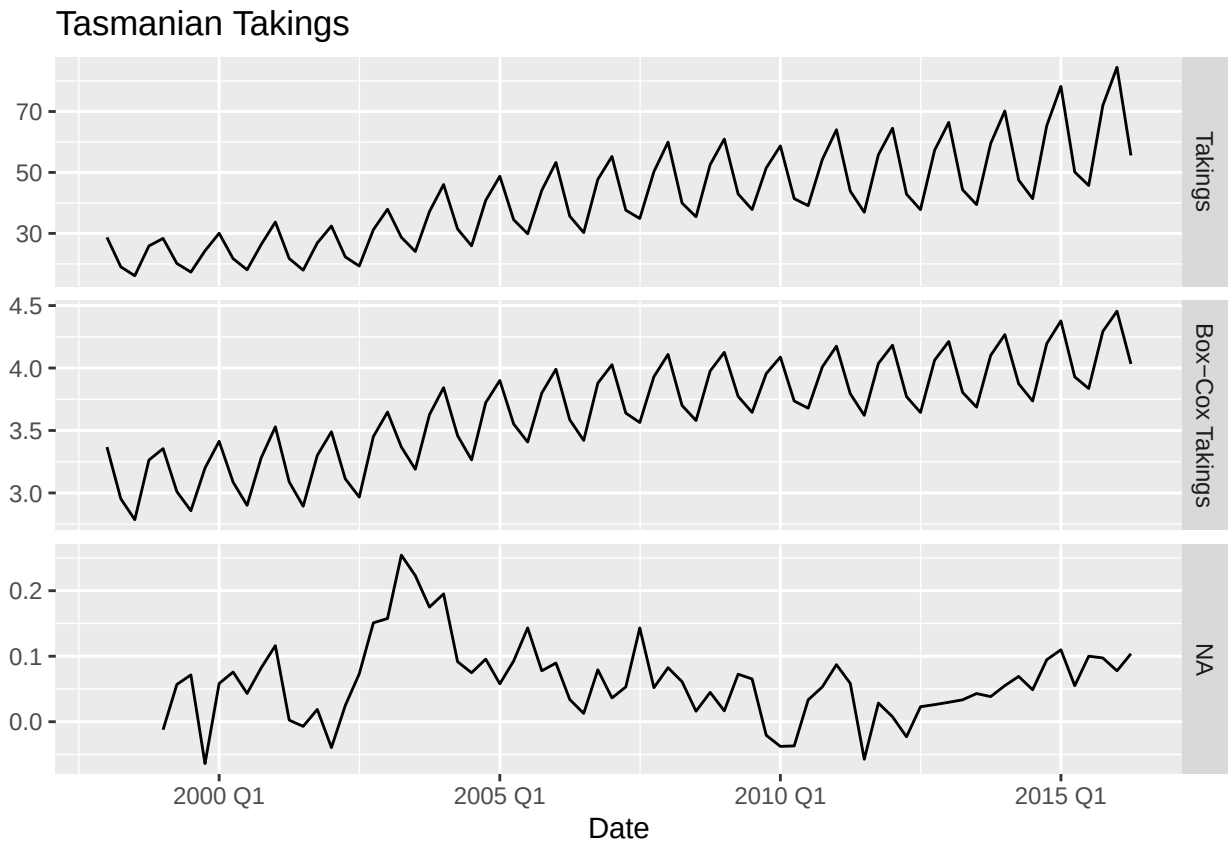



```
tas_lambda <- tas_takings |>
  features(Takings, features = guerrero) |>
  pull(lambda_guerrero)

tas_nsdiffs <- tas_takings |>
  features(box_cox(Takings, tas_lambda), unitroot_nsdiffs) |>
  pull(nsdiffs)

tas_ndiffs <- tas_takings |>
  features(difference(box_cox(Takings, tas_lambda),lag = 4, differences = tas_nsdiffs), features = unit
  pull(ndiffs)

cat("Optimal lambda: ", tas_lambda,
    "\nRequired seasonal differences: ", tas_nsdiffs,
    "\nRequired first differences: ", tas_ndiffs)
```

```
## Optimal lambda:  0.001819643
## Required seasonal differences:  1
## Required first differences:  0
```

```
tas_takings |>
  transmute(
    Date,
    `Takings` = Takings,
    `Box-Cox Takings` = box_cox(Takings, tas_lambda),
    `Tasmania Takings with season diff` = difference(box_cox(Takings, tas_lambda), lag = 4, differences
  pivot_longer(-Date, names_to = "Type", values_to = "Takings") |>
  mutate(
```

```
    Type = factor(Type, levels = c(
      "Takings",
      "Box-Cox Takings",
      "Box-Cox Takings - First Differenced"
    ))
  ) |>
  ggplot(aes(x = Date, y = Takings)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
  labs(title = "Tasmanian Takings", y = NULL)
```

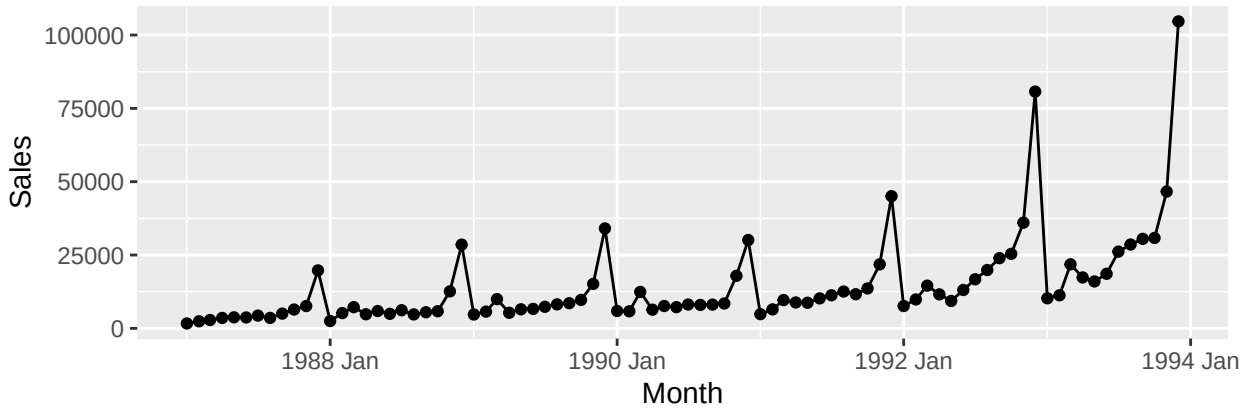


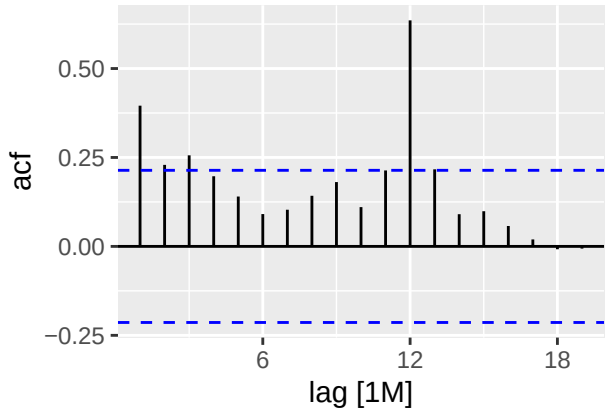
Optimal lambda: 0.001819643 Required seasonal differences: 1 Required first differences: 1

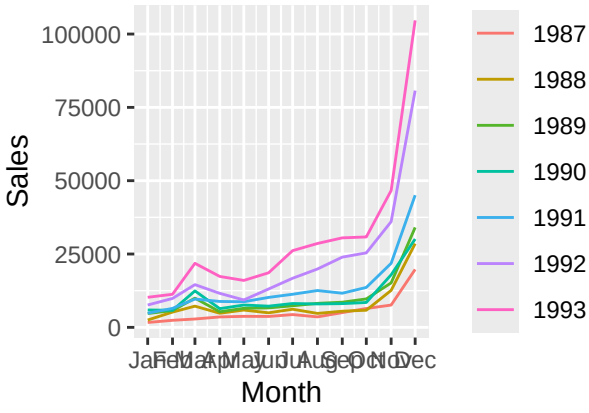
Applying the optimal lambda value of 0.0018 to a Box-Cox transformation stabilized the data and created more uniformity. The required seasonal differencing of 1 removed the repeating seasonal pattern and the required first differencing of 1 removed any linear trend. Thus a stationary dataset was achieved, as shown in the bottom graph above.

9.3.C Monthly sales from souvenirs

```
souvenirs |>
  gg_tsdisplay(Sales)
```





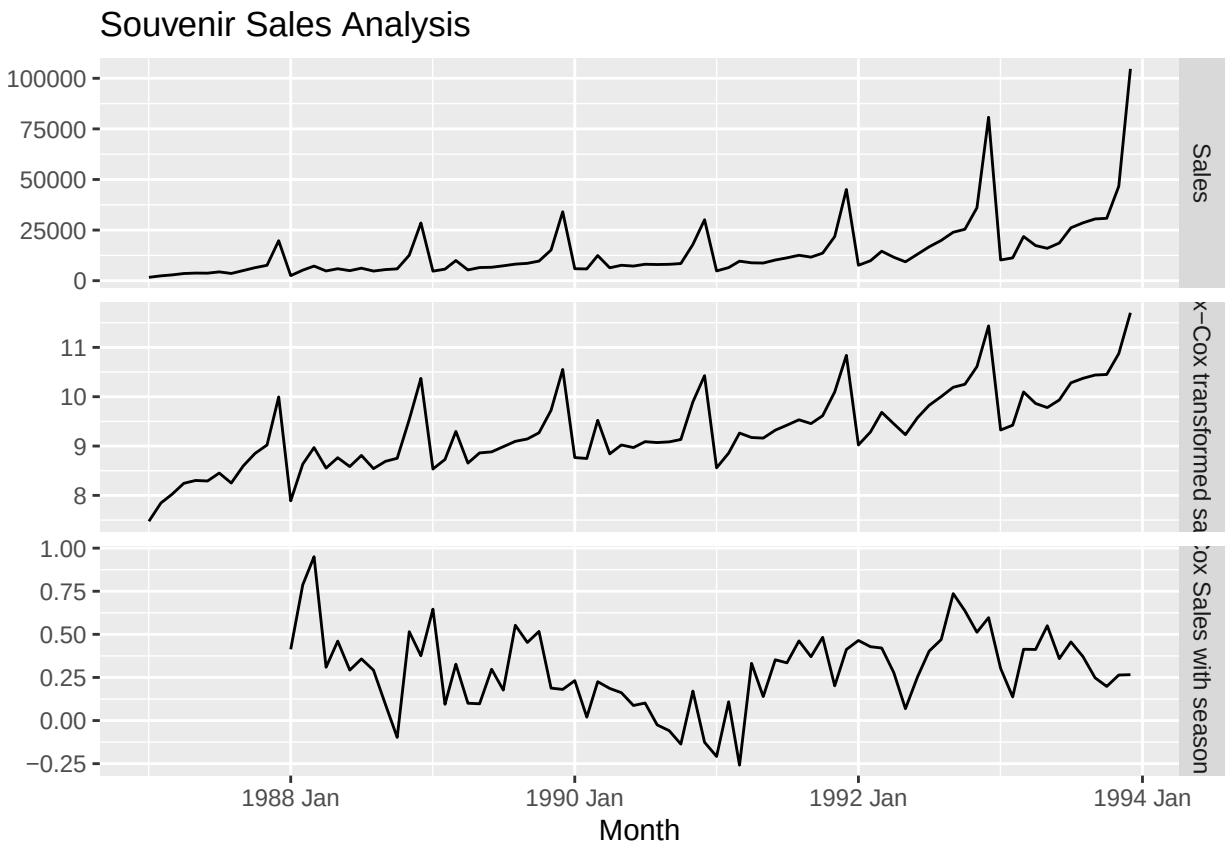



```
souv_lambda <- souvenirs |>
  features(Sales, features = guerrero) |>
  pull(lambda_guerrero)

souv_nsdiffs <- souvenirs |>
  features(box_cox(Sales, souv_lambda), features = unitroot_nsdiffs) |>
  pull(nsdiffs)

souv_ndiffs <- souvenirs |>
  features(difference(box_cox(Sales, souv_lambda),lag = 12, differences = souv_nsdiffs), features = uni

cat("Optimal lambda: ", souv_lambda,
    "\nRequired seasonal differences: ", souv_nsdiffs,
    "\nRequired first differences: ", souv_ndiffs)
```

```
## Optimal lambda:  0.002118221
## Required seasonal differences:  1
## Required first differences:  0
```

```
souvenirs |>
  transmute(
    `Sales` = Sales,
    `Box-Cox transformed sales` = box_cox(Sales, souv_lambda),
    `Box-Cox Sales with seasonal diffs` = difference(box_cox(Sales, souv_lambda), lag = 12, differences
    ) |>
  pivot_longer(-Month, names_to="Type", values_to="Sales") |>
  mutate(
    Type = factor(Type, levels = c(
```

```
      "Sales",
      "Box-Cox transformed sales",
      "Box-Cox Sales with seasonal diffs"
      ))
  ) |>
  ggplot(aes(x = Month, y = Sales)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
  labs(title = "Souvenir Sales Analysis", y = NULL)
```



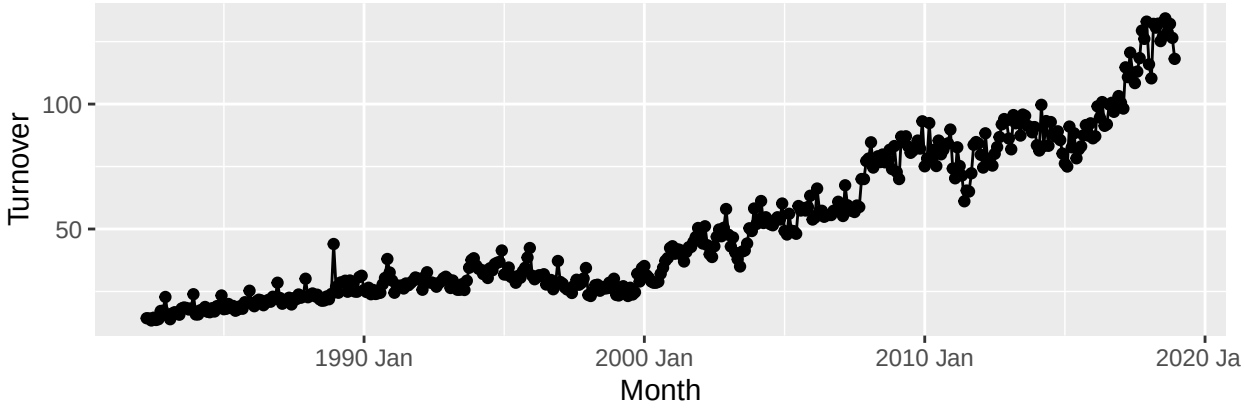
Optimal lambda: 0.002118221 Required seasonal differences: 1 Required first differences: 0

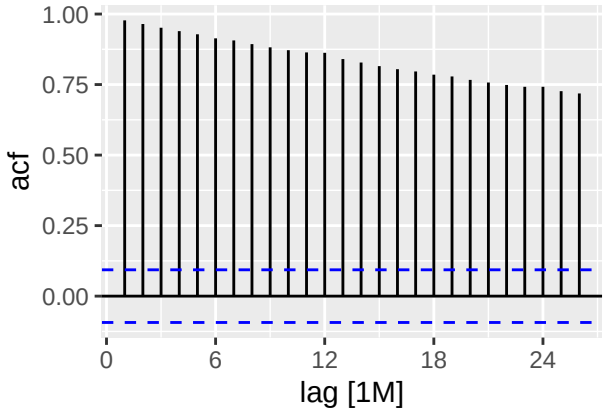
Non-seasonality is achieved mainly by removing last year's data from this year's value.The low lambda had the intended effect of softening up the trend and creating something a bit more uniform.

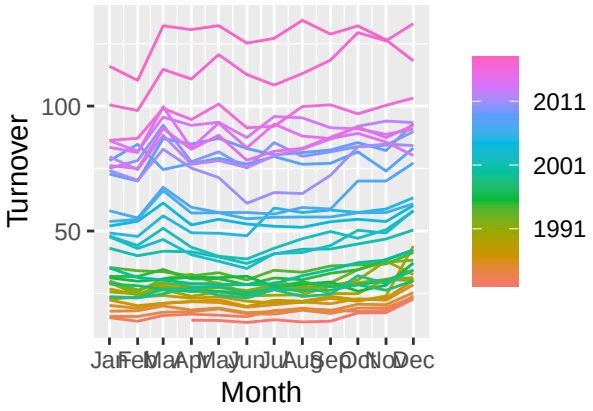
9.5 aus_retail order of differencing

The exercise states: "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."

```
aus_sample <- aus_retail |>
  filter(`Series ID` == sample(aus_retail$`Series ID`,1))

aus_sample |>
  gg_tsdisplay(Turnover)
```








```
aus_lambda <- aus_sample |>
  features(Turnover, features = guerrero) |>
  pull(lambda_guerrero)

aus_nsdiffs <- aus_sample |>
  features(box_cox(Turnover, aus_lambda), features = unitroot_nsdiffs) |>
  pull(nsdiffs)

aus_nsdiffs <- ifelse(aus_nsdiffs > 0, as.integer(aus_nsdiffs), 1)

aus_ndiffs <- aus_sample |>
  features(difference(box_cox(Turnover, aus_lambda), lag = 12, differences = aus_nsdiffs),
           features = unitroot_ndiffs) |>
  pull(ndiffs)

cat("Optimal lambda: ", aus_lambda,
    "\nRequired seasonal differences: ", aus_nsdiffs,
    "\nRequired first differences: ", aus_ndiffs)
```

```
## Optimal lambda:  0.441087
## Required seasonal differences:  1
## Required first differences:  0
```

Optimal lambda: 0.07636928 Required seasonal differences: 1 Required first differences: 1

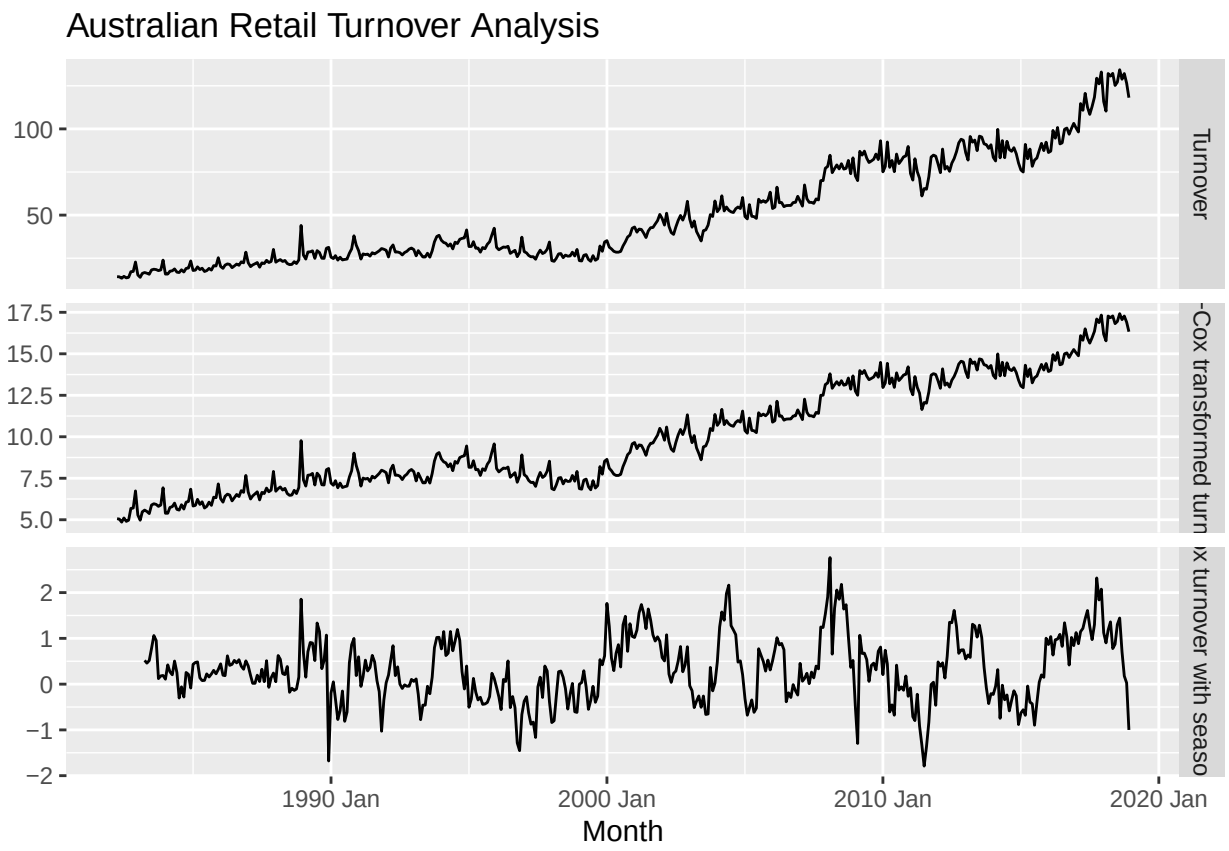
Based on these values, the appropriate order of differencing is:

- Apply a Box-Cox transformation with lambda set as 0.076. This will stabilize variance
- Apply one seasonal difference to remove the seasonal component

- Apply one first difference to remove the remaining trend

Here's where I implement those steps:

```
aus_sample |>
  transmute(
    `Turnover` = Turnover,
    `Box-Cox transformed turnover` = box_cox(Turnover, aus_lambda),
    `Box-Cox turnover with seasonal diffs` = difference(box_cox(Turnover, aus_lambda), lag = 12, differe
    ) |>
  pivot_longer(-Month, names_to="Type", values_to="Turnover") |>
  mutate(
    Type = factor(Type, levels = c(
      "Turnover",
      "Box-Cox transformed turnover",
      "Box-Cox turnover with seasonal diffs"
      ))
  ) |>
  ggplot(aes(x = Month, y = Turnover)) +
  geom_line() +
  facet_grid(vars(Type), scales = "free_y") +
  labs(title = "Australian Retail Turnover Analysis", y = NULL)
```



The bottom plot is the result of applying the lambda and difference values. You can see that all the seasonal noise and trends have been removed, leading to a more stationary series when compared to the original data in the top row. Applying the seasonal and first difference after the Box-Cox transformation has stabilized the variance and mean as well. Reducing the autocorrelations allows the data to be better used in time series modeling as seasonal and other trends tend to get in the way of effective forecasting. This data is now much

closer to being stationary, which is a key requirement for ARIMA models.

9.6 Simulate and plot some data from simple ARIMA models

This exercise consists of components A through G

A. Use providerd R code to generate data from an AR(1)

This component states: "Use the following R code to generate data from an AR(1) model with $\phi_1 = 0.6$ and $\sigma^2 = 1$. The process starts with $y_1 = 0$."

Answer: I've done this for two different values, with phi1_base being what was provided and then phi1_adjust being a second value that supports plotting in component B.

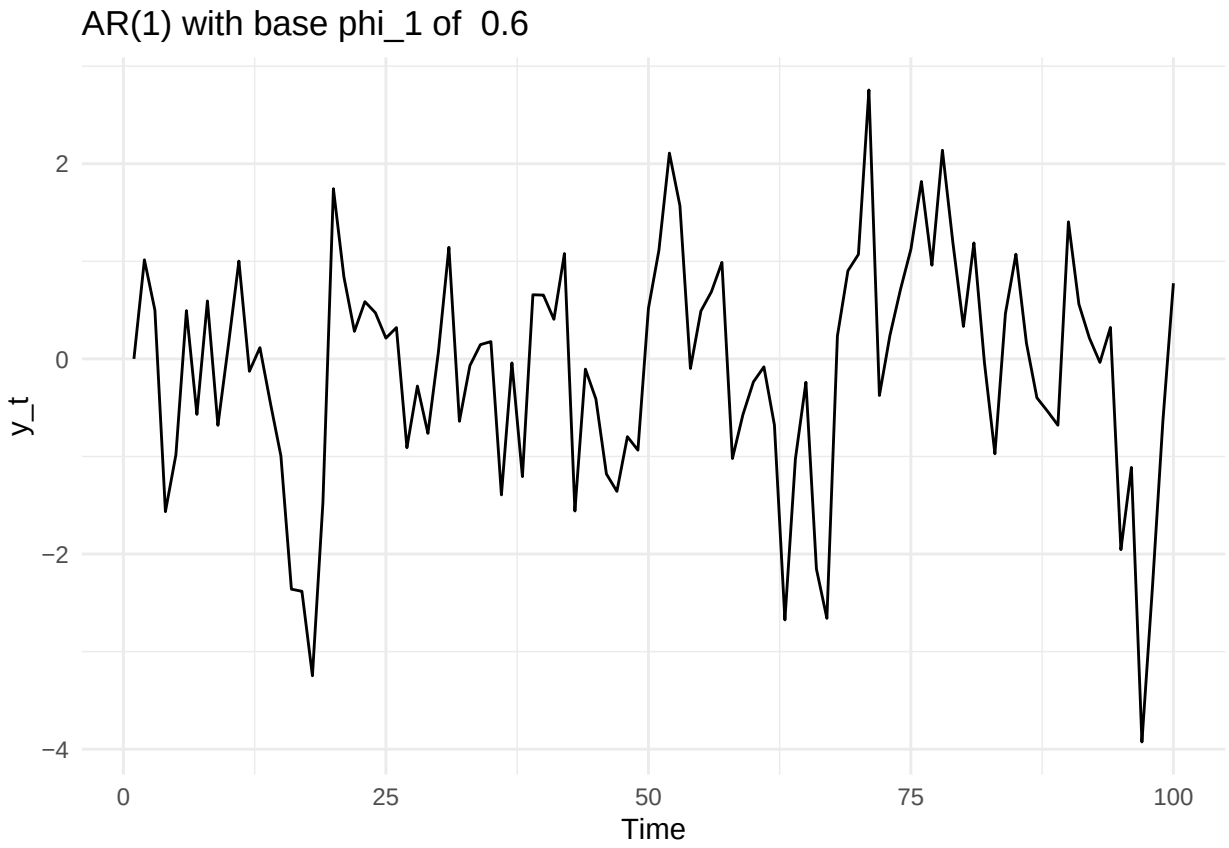
```
phi1_base <- 0.6
y <- numeric(100)
e <- rnorm(100)
for(i in 2:100)
  y[i] <- phi1_base*y[i-1] + e[i]
sim <- tsibble(idx = seq_len(100), y = y, index = idx)

phi1_adjust <- 0.9
y_a <- numeric(100)
e_a <- rnorm(100)
for(i in 2:100)
  y_a[i] <- phi1_adjust*y_a[i-1] + e_a[i]
sim_a <- tsibble(idx_a = seq_len(100), y = y_a, index = idx_a)
```

B. Produce a time plot for the series

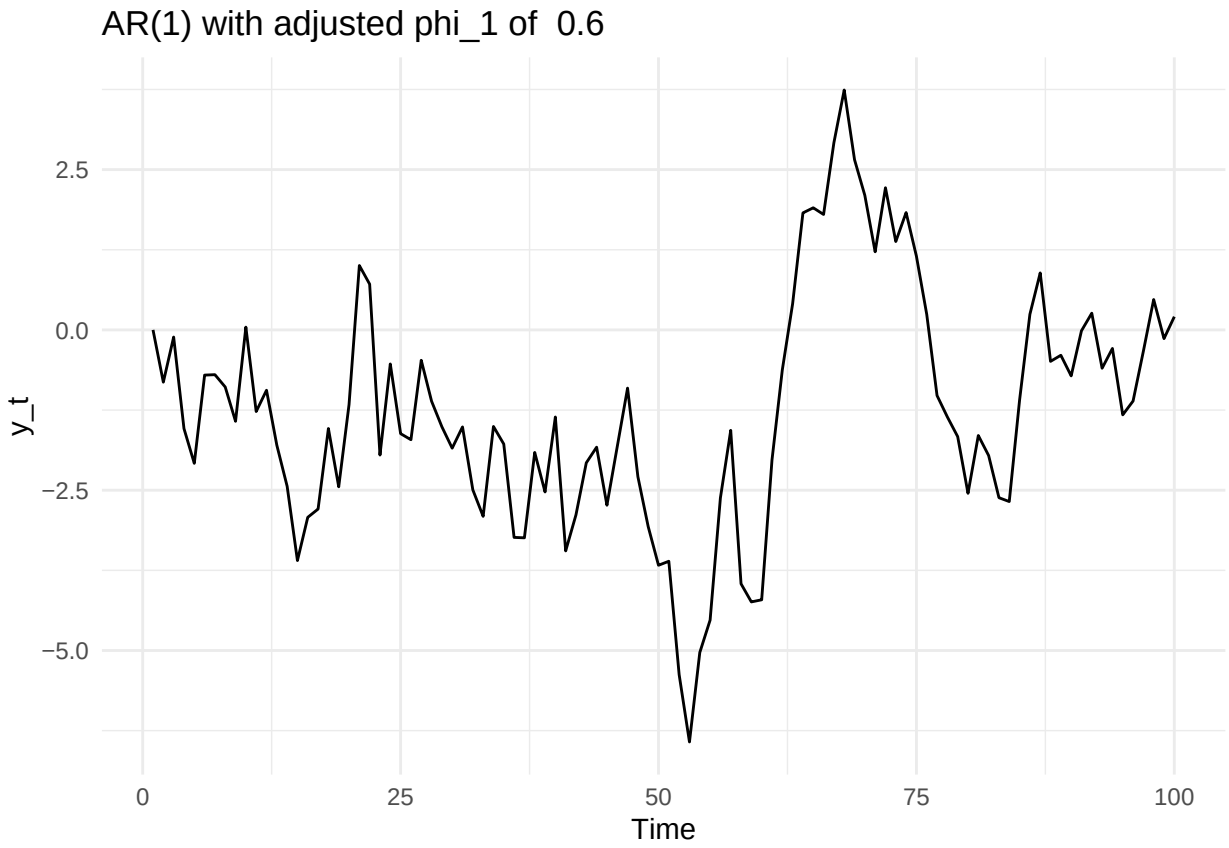
"How does the plot change as you change ϕ_1?"

Answer: Changing ϕ_1 from 0.6 to 0.9 led to the adjusted chart having a smoother and more persistent structure when compared to the baseline chart. In the adjusted plot, the spikes up and low points are more defined and there are longer lasting trends. This is good evidence of strong autocorrelation. On the other hand, the base plot is more eratic while also having no real meaningful trends, other than to essentially be flat when looking start to the end. This indicates weaker autocorrelation.

```
ggplot(sim, aes(x = idx, y = y)) +
  geom_line() +
  labs(title = paste("AR(1) with base phi_1 of ", phi1_base),
       x = "Time",
       y = "y_t") +
  theme_minimal()
```




```
ggplot(sim_a, aes(x = idx_a, y = y_a)) +
  geom_line() +
  labs(title = paste("AR(1) with adjusted phi_1 of ", phi1_base),
       x = "Time",
       y = "y_t") +
  theme_minimal()
```



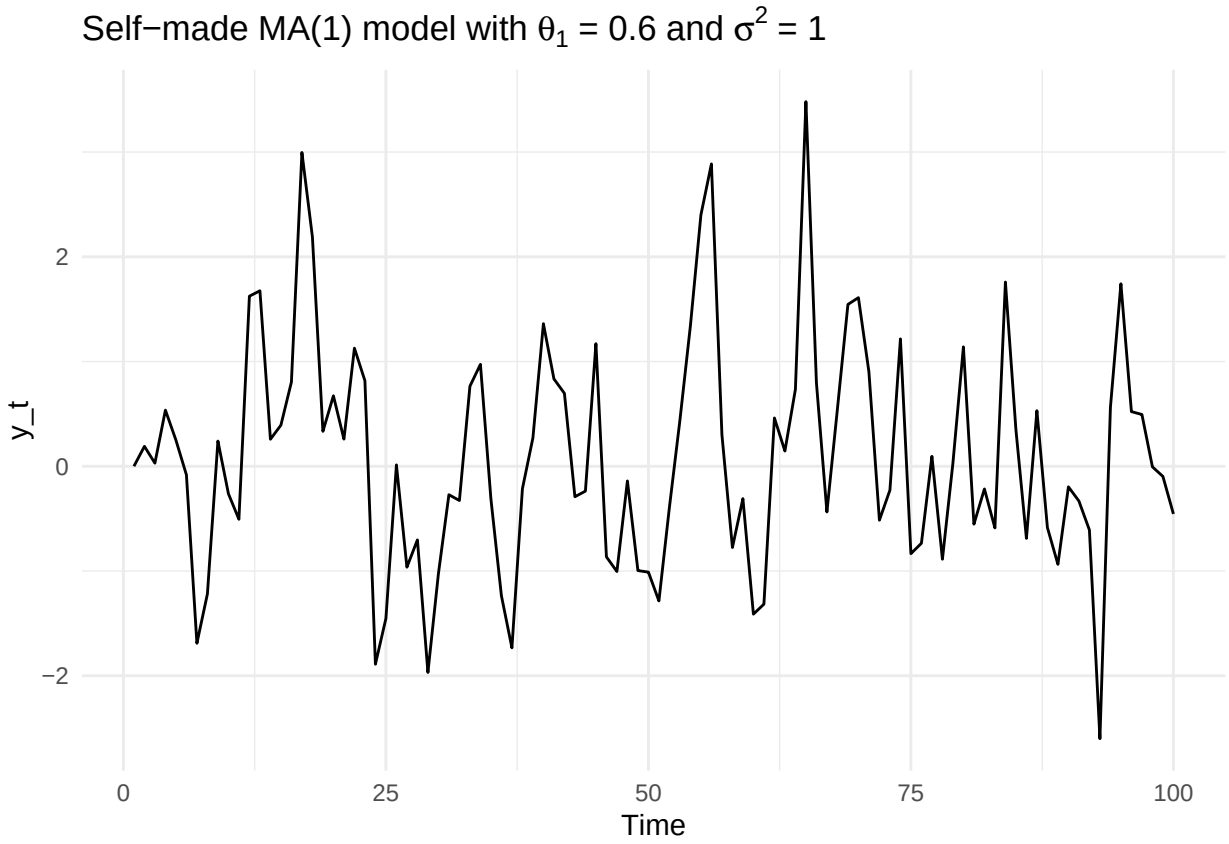
C. Write your own code to generate data from an MA(1) model

"..with $\theta_1 = 0.6$ and $\sigma^2 = 1$."

Answer:

This generates the model with the required parameters and then plots the results. Overallm a relatively unremarkable chart that shows consistenly seasonality and no strong trends.

```
theta_one <- 0.6
sigma_two <- 1
sample_size <- 100
e <- rnorm(sample_size, mean = 0, sd = sqrt(sigma_two))

y <- numeric(sample_size)
for (i in 2:sample_size) {
  y[i] <- e[i] + theta_one * e[i-1]
}

sim_maone_moi <- tsibble(idx = seq_len(sample_size), y = y, index = idx)

ggplot(sim_maone_moi, aes(x = idx, y = y)) +
  geom_line() +
  labs(title = expression("Self-made MA(1) model with " * theta[1] * " = 0.6 and " * sigma^2 * " = 1"),
       x = "Time",
       y = "y_t") +
  theme_minimal()
```



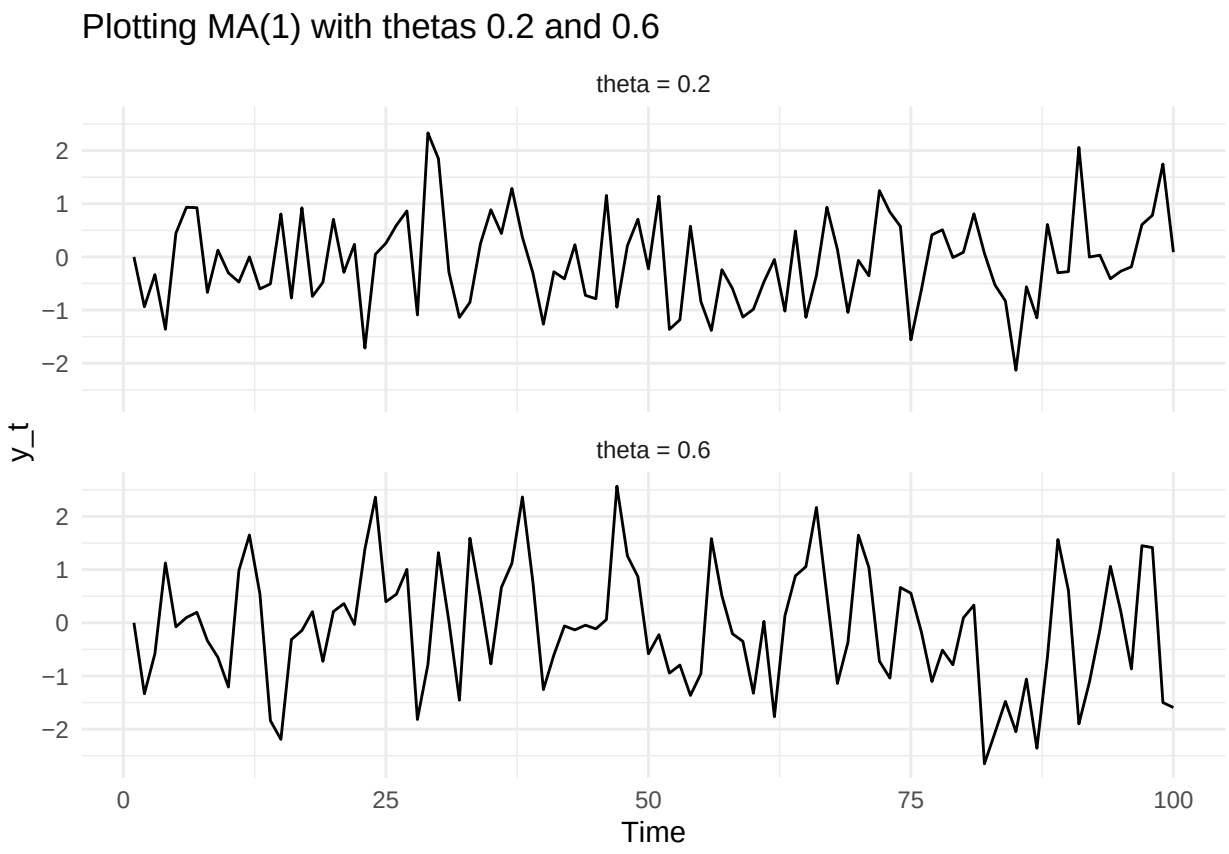
D. Produce a time plot for the series

"...How does the plot change as you change θ_1 ?

Answer: When θ_1 increases from 0.2 to 0.6, the plots show largerfluctuations and more pronounced highs and lows. With $\theta_1 = 0.6$, previous noise terms are having a larger impact, causing greater volatility. 0.2 is much smoother in this respect.

```
thetas <- c(0.2, 0.6)
sigmas <- 1
samples <- 100

ma1_constructor <- function(theta, sigma, samplers) {
  e <- rnorm(samplers, mean = 0, sd = sqrt(sigma))
  y <- numeric(samplers)
  for (i in 2:samplers) {
    y[i] <- e[i] + theta * e[i - 1]
  }
  return(y)
}

data_holder <- lapply(thetas, function(theta) {
  y <- ma1_constructor(theta, sigmas, samples)
  data.frame(idx = seq_len(samples), y = y, theta = paste("theta =", theta))
})

thetas_power <- do.call(rbind, data_holder)
```

```r
ggplot(thetas_power, aes(x = idx, y = y)) +
  geom_line() +
  facet_wrap(~ theta, ncol = 1) +
  labs(title = "Plotting MA(1) with thetas 0.2 and 0.6",
       x = "Time",
       y = "y_t") +
  theme_minimal()
```



E. Generate data from an ARMA(1,1)

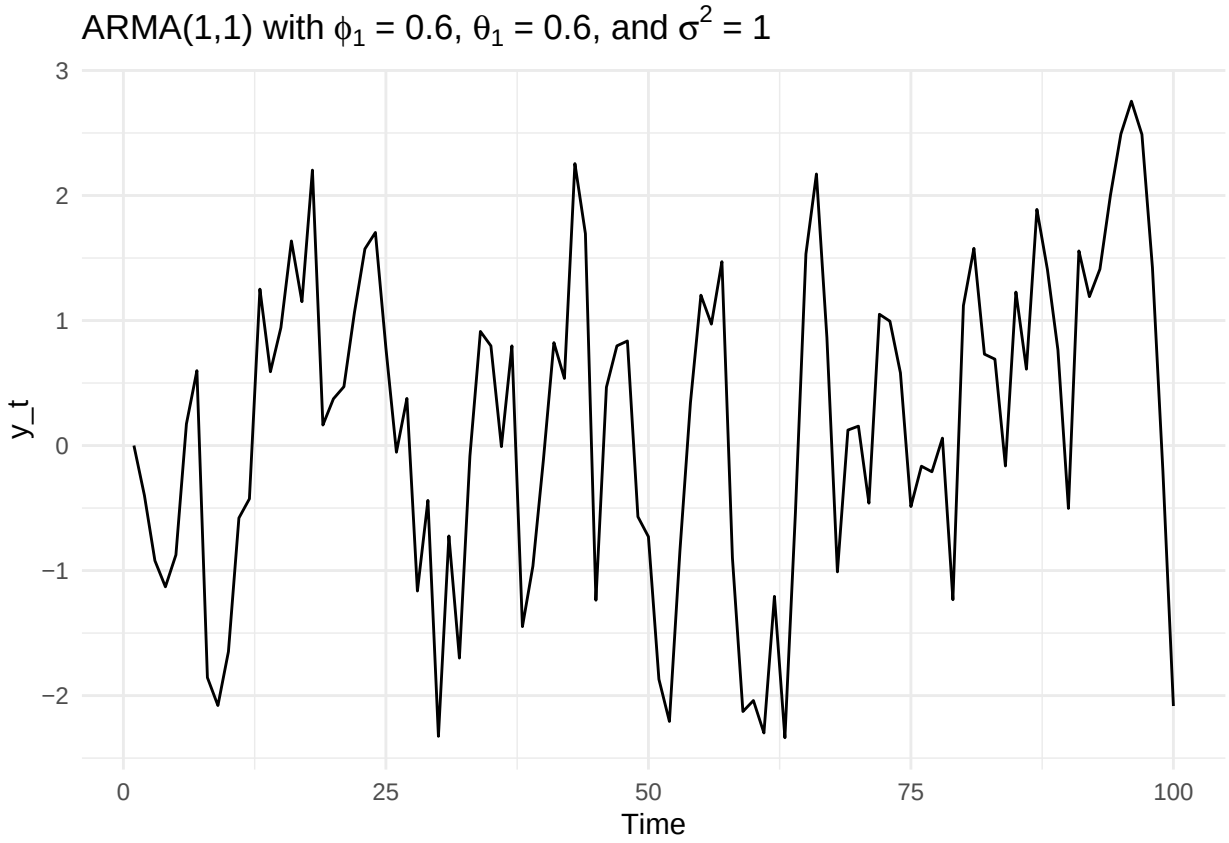
"...with $\phi_1 = 0.6$, $\theta_1 = 0.6$, and $\sigma^2 = 1$."

Answer:

I have generated the data and graphed it here. I will provide commentary on both graphs as part of component G below.

```r
phi_one <- 0.6
theta_one <- 0.6
sigma_two <- 1
gimme_more <- 100
e_power <- rnorm(gimme_more, mean = 0, sd = sqrt(sigma_two))

y_power <- numeric(gimme_more)
for (i in 2:gimme_more) {
  y_power[i] <- phi_one * y_power[i - 1] + e_power[i] + theta_one * e[i - 1]
}
```

```
armada <- tsibble(idx = seq_len(gimme_more), y = y_power, index = idx)

ggplot(armada, aes(x = idx, y = y_power)) +
  geom_line() +
  labs(title = expression("ARMA(1,1) with " * phi[1] * " = 0.6, " * theta[1] * " = 0.6, and " * sigma^2
       x = "Time",
       y = "y_t") +
  theme_minimal()
```



F. Generate data from an AR(2) model

"...with $\phi_1 = -0.8$, $\phi_2 = 0.3$, and $\sigma^2 = 1$. (Note that these parameters will give a non-stationary series.)"

Answer:

Same as above, I have generated the data and chart here and will provide commentary as part of component G below.

"'{r-ar2-plot} phi_uno <- -0.8 phi_dos <- 0.3 sigma_dos <- 1 n_noway <- 100 e_yesway <- rnorm(n_noway, mean = 0, sd = sqrt(sigma_dos))

y_not <- numeric(n_noway) y_not[1] <- e_yesway[1] y_not[2] <- e_yesway[2]

for (i in 3:n_noway) { y_not[i] <- phi_uno * y_not[i - 1] + phi_dos * y_not[i - 2] + e_yesway[i] }

artwo_dtwo <- tsibble(idx = seq_len(n_noway), y = y_not, index = idx)

ggplot(artwo_dtwo, aes(x = idx, y = y_not)) + geom_line() + labs(title = expression("AR(2) with" * phi[1] * " = -0.8, " * phi[2] * " = 0.3, and " * sigma^2 * " = 1"), x = "Time", y = "y_t") + theme_minimal()

G. Graph the latter two series and compare them

Answer:
The plots are above and I will now compare them.

The ARMA(1,1) chart has both autoregressive (\(\phi_1 = 0.6\)) and moving average (\(\theta_1 = 0.6\)) c

Meanwhile, the AR(2) chart with \(\phi_1 = -0.8\) and \(\phi_2 = 0.3\) is quite different in...every way

9.7 Reviewing aus_passengers data

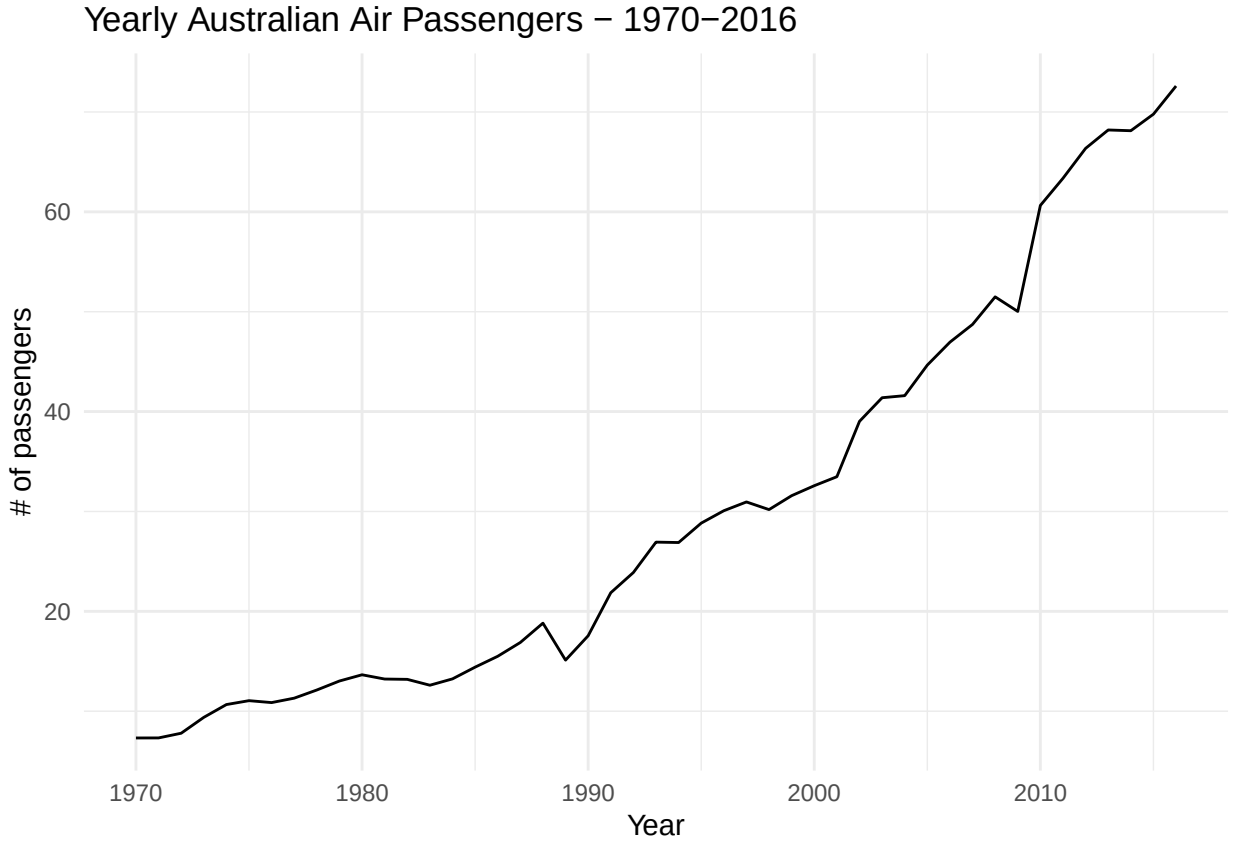
This exercise consistents of component questions A through E, with D having a subcomponent within it as

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

```r
  autoplot(aus_airpassengers) +
    labs(title = "Yearly Australian Air Passengers - 1970-2016",
       x = "Year",
       y = "# of passengers") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```

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

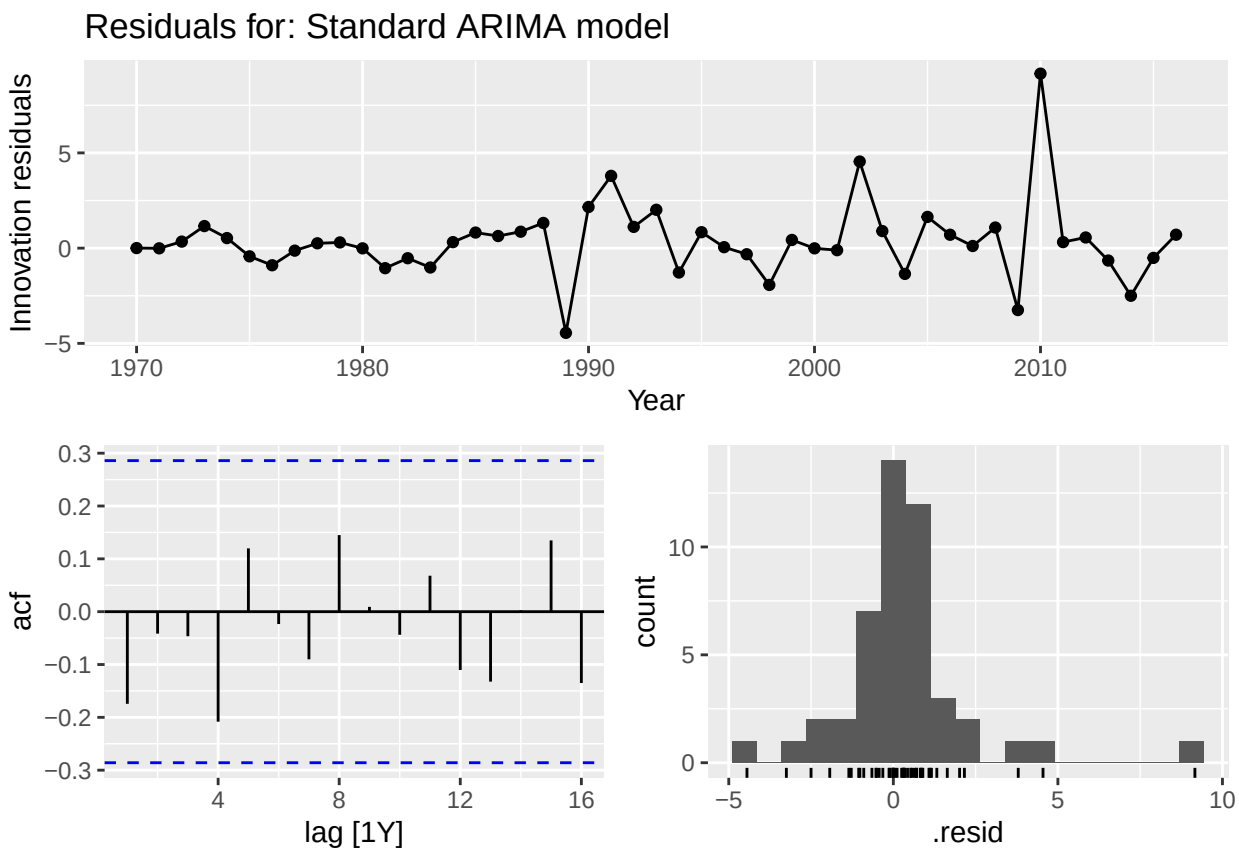


```
(modeler_airside <- aus_airpassengers |>
  model(
    arima_air = ARIMA(Passengers),
    `arima_up` = ARIMA(Passengers ~ 1 + pdq(0,2,0)),
    `arima_down` = ARIMA(Passengers ~ 1 + pdq(1,2,2)),
    `constantless_arima` = ARIMA(Passengers ~ 0 + pdq(3,2,1), method="ML"),
    `full_arima` = ARIMA(Passengers ~ 1 + pdq(1,1,1)),
  ))
```

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

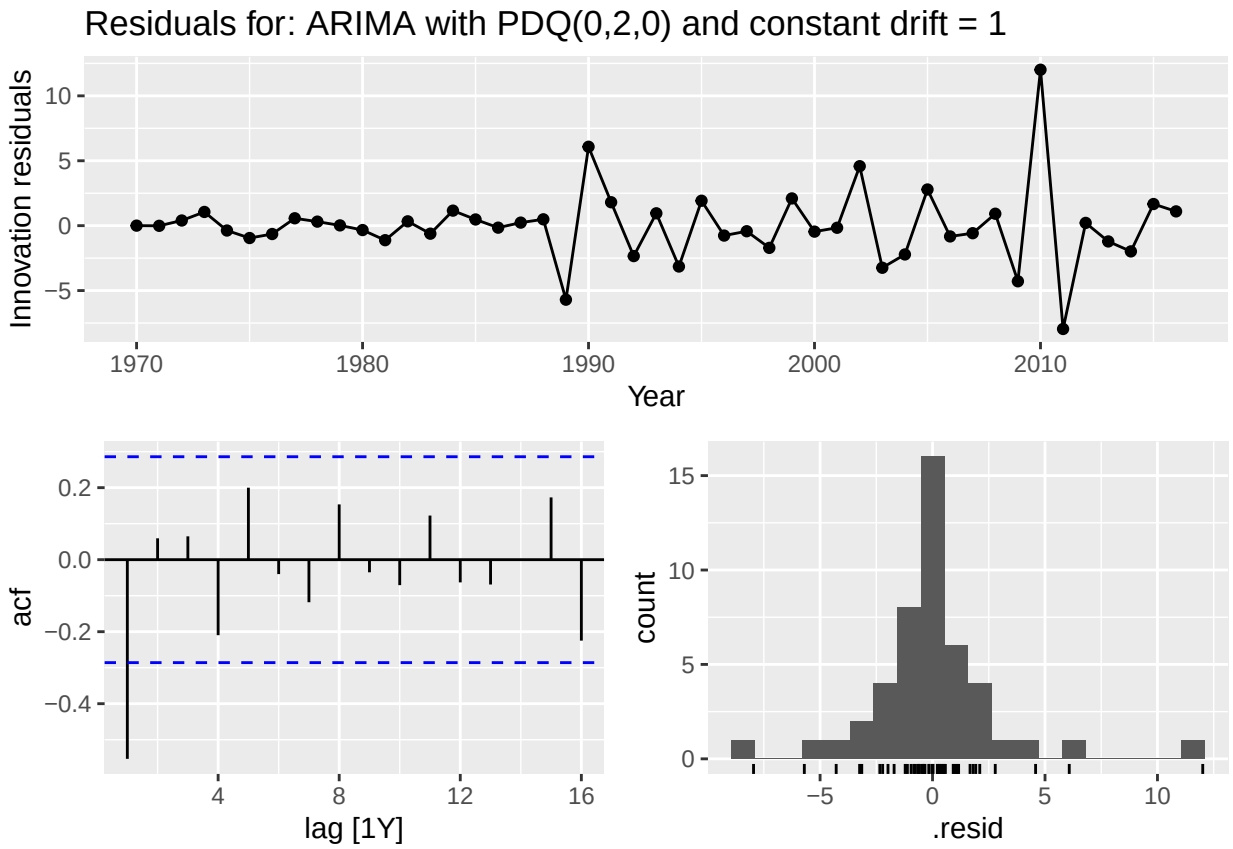
```
## # A mable: 1 x 5
##        arima_air               arima_up             arima_down
##          <model>                <model>                <model>
## 1 <ARIMA(0,2,1)> <ARIMA(0,2,0) w/ poly> <ARIMA(1,2,2) w/ poly>
## # i 2 more variables: constantless_arima <model>, full_arima <model>
```

```
modeler_airside |>
  select(arima_air) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: Standard ARIMA model")
```

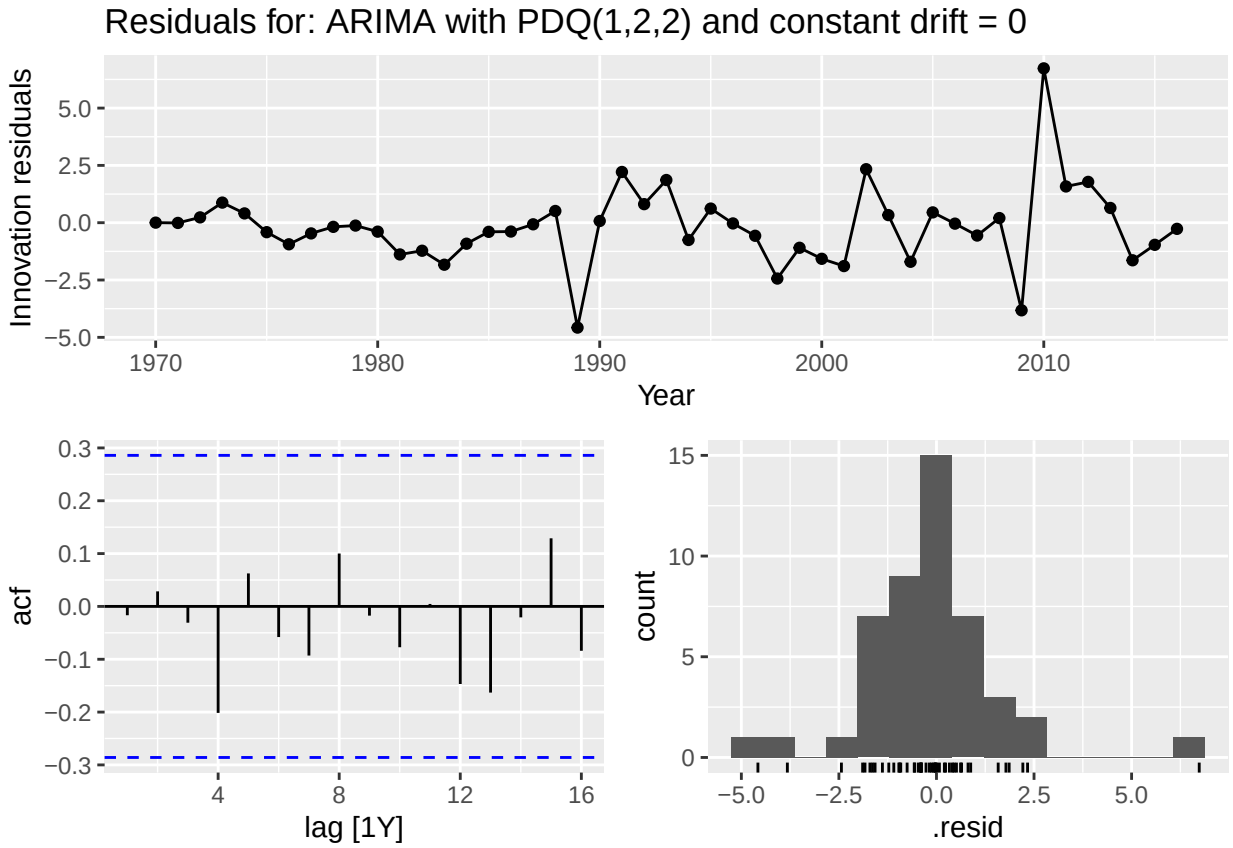



```
modeler_airside |>
  select(arima_up) |>
  gg_tsresiduals() +
```

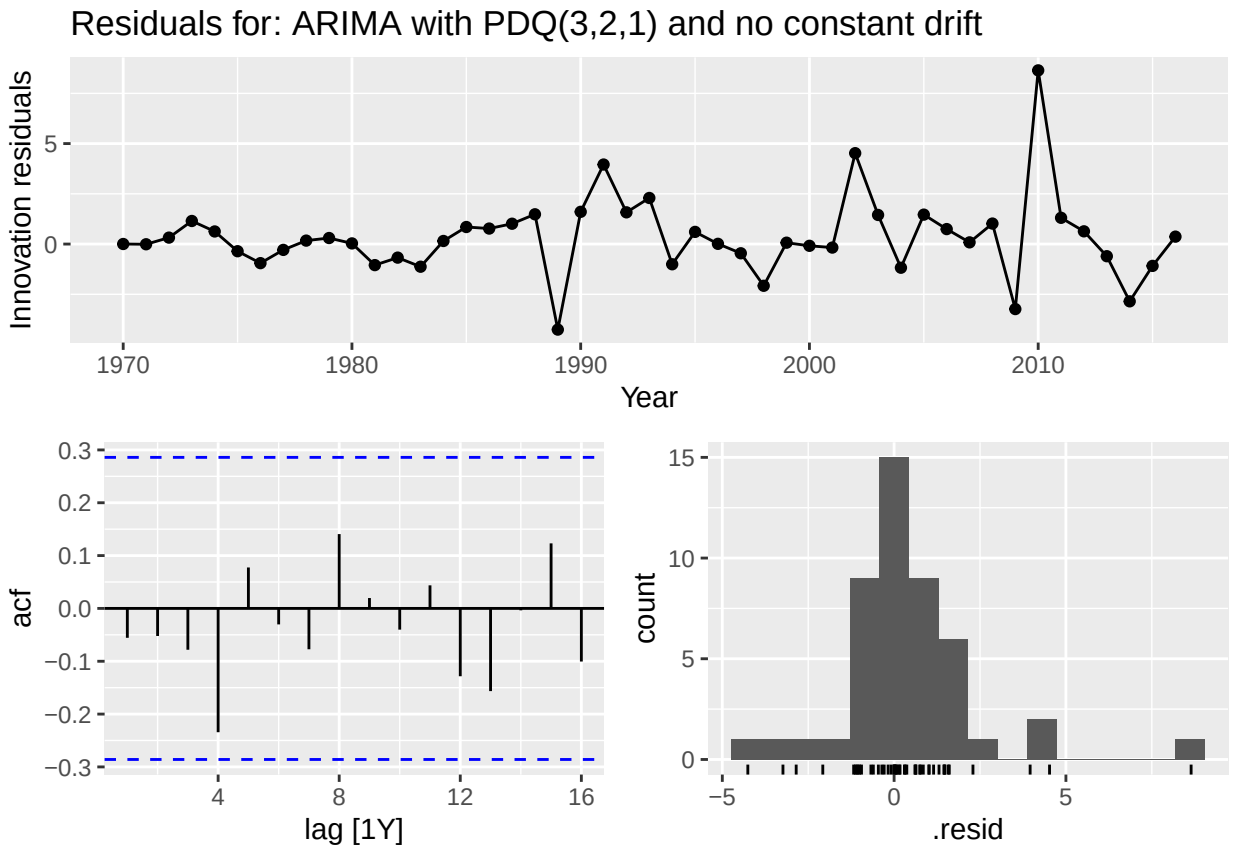
```
  ggtitle("Residuals for: ARIMA with PDQ(0,2,0) and constant drift = 1")
```



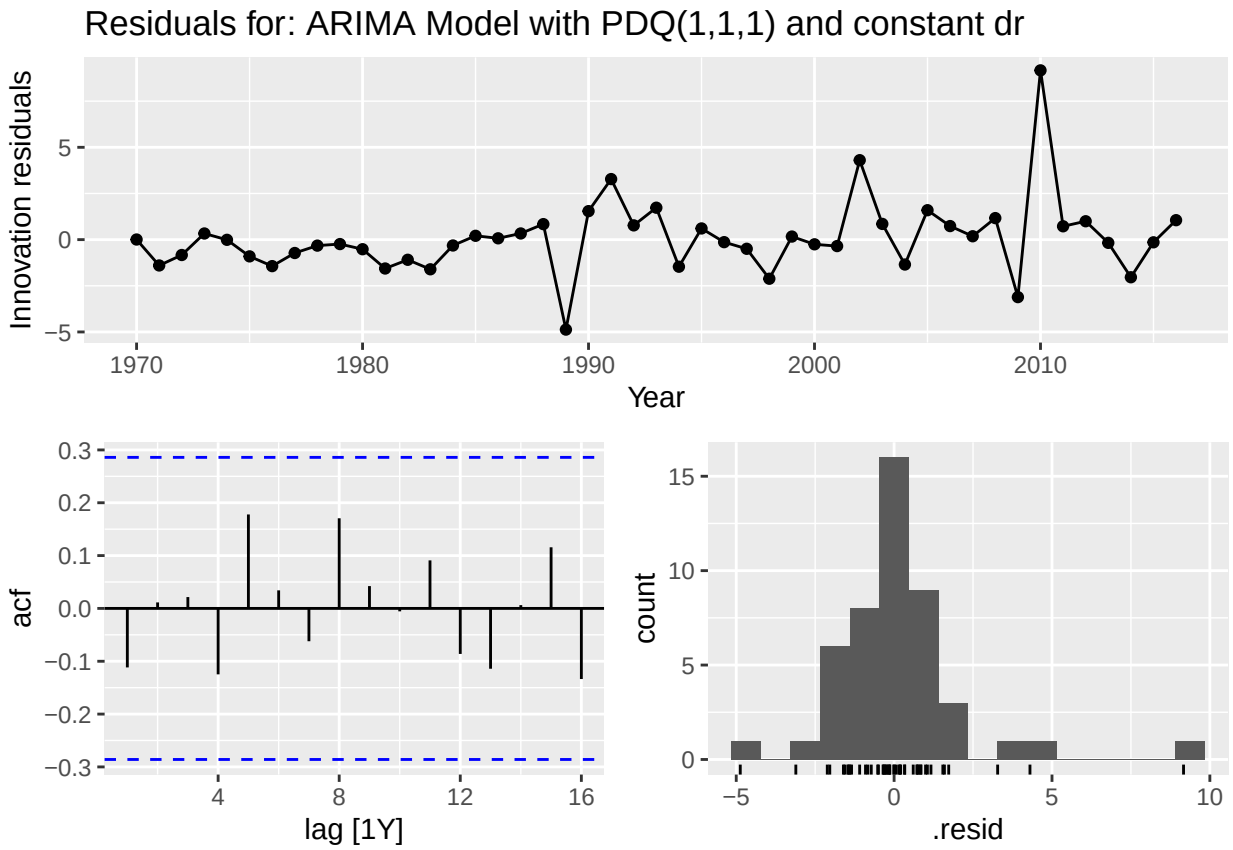

```
modeler_airside |>
  select(arima_down) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: ARIMA with PDQ(1,2,2) and constant drift = 0")
```




```
modeler_airside |>
  select(constantless_arima) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: ARIMA with PDQ(3,2,1) and no constant drift")
```

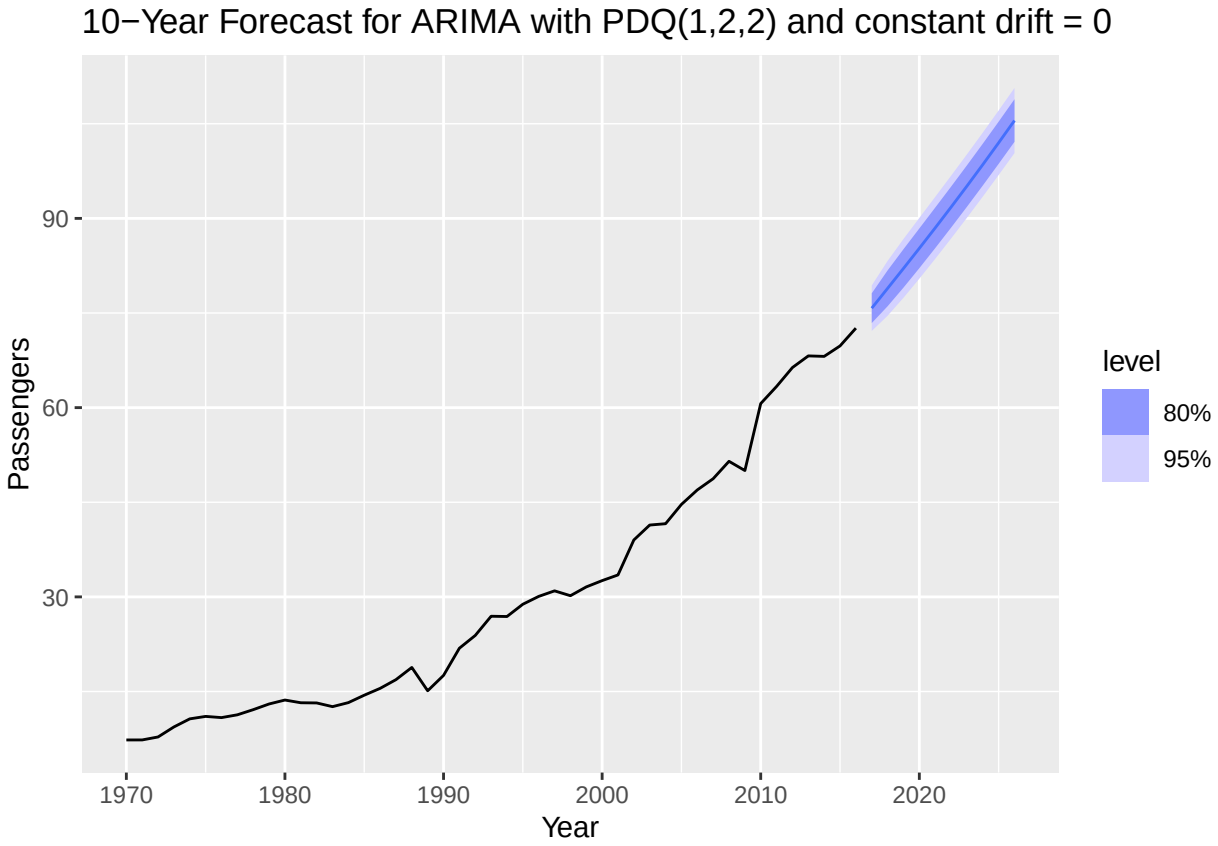



```
modeler_airside |>
  select(full_arima) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: ARIMA Model with PDQ(1,1,1) and constant dr")
```



I chose the ARIMA model with p = 1, d = 2, q = 2 and no constant because it best fit the data out of the models I tried. The data had a strong non-stationary trend, which required two differencing steps to stabilize. Including a constant didn't seem necessary given the nature of the trend. Overall, this model captured the key patterns in the data without overcomplicating it.

```
arima_down_forecast <- modeler_airside |>
     select(arima_down) |>
    forecast(h = 10)

arima_down_forecast |>
     autoplot(aus_airpassengers) +
     ggtitle("10-Year Forecast for ARIMA with PDQ(1,2,2) and constant drift = 0")
```



B. Write the model in terms of the backshift operator

```
modeler_airside |>
  select(arima_down) |>
  report()
```

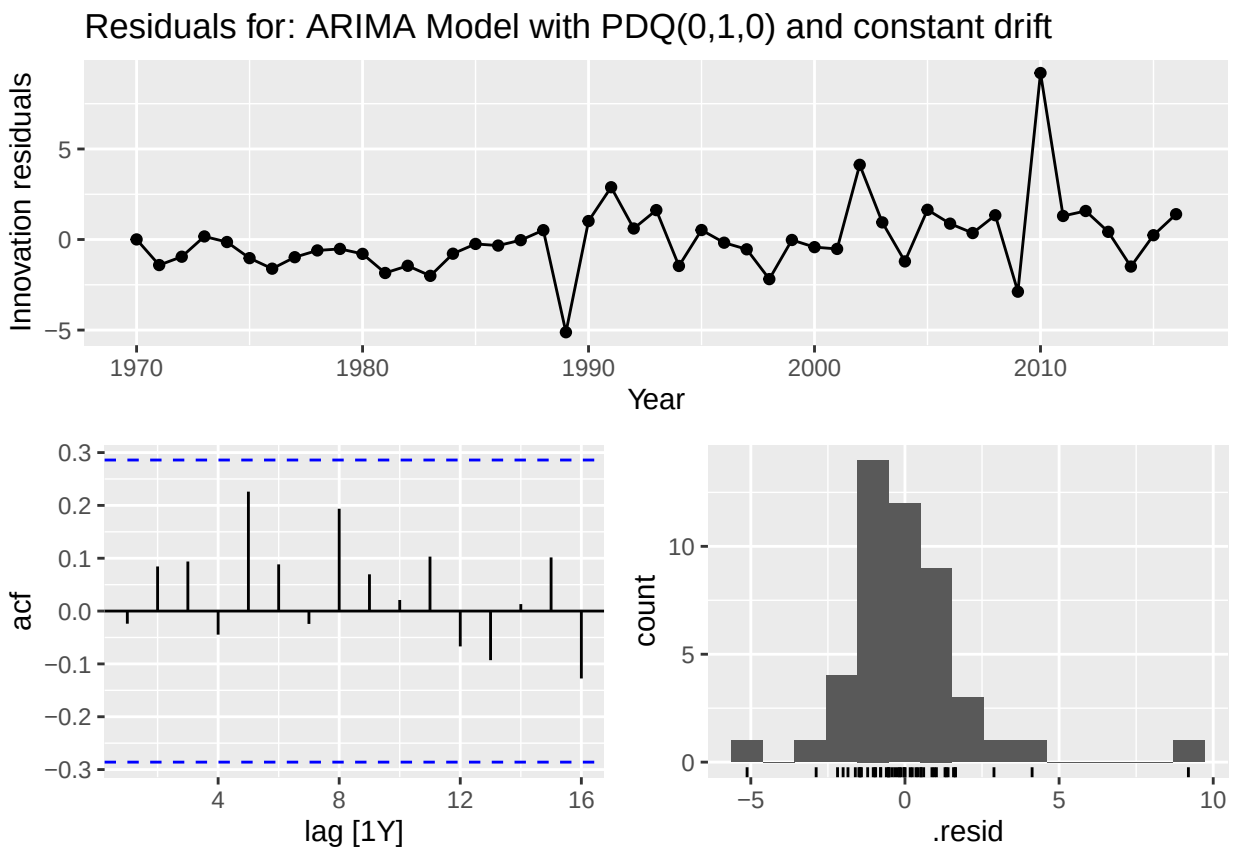
```
## Series: Passengers
## Model: ARIMA(1,2,2) w/ poly
##
## Coefficients:
##          ar1      ma1     ma2  constant
##       0.6317  -1.9951  0.9999    0.0241
## s.e.  0.1419   0.1089  0.1089    0.0022
##
## sigma^2 estimated as 3.263:  log likelihood=-92.54
## AIC=195.09   AICc=196.63   BIC=204.12
```

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

Answer: The ARIMA(0,1,0) model with constant drift shows a fairly stable set of residuals, although there are a few noticeable outliers in the later years. In contrast, the ARIMA(1,2,2) model with no drift captures the general pattern but the residuals exhibit less variance and more centralized clustering around zero. Both models have similar autocorrelation patterns, though the ARIMA(1,2,2) appears to have a slightly better fit with fewer large residual spikes.

```
modeler_drifter <- aus_airpassengers |>
  model(
```

```
    `arima_drifter` = ARIMA(Passengers ~ 1 + pdq(0,1,0))
  )

modeler_drifter |>
  select(arima_drifter) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: ARIMA Model with PDQ(0,1,0) and constant drift")
```



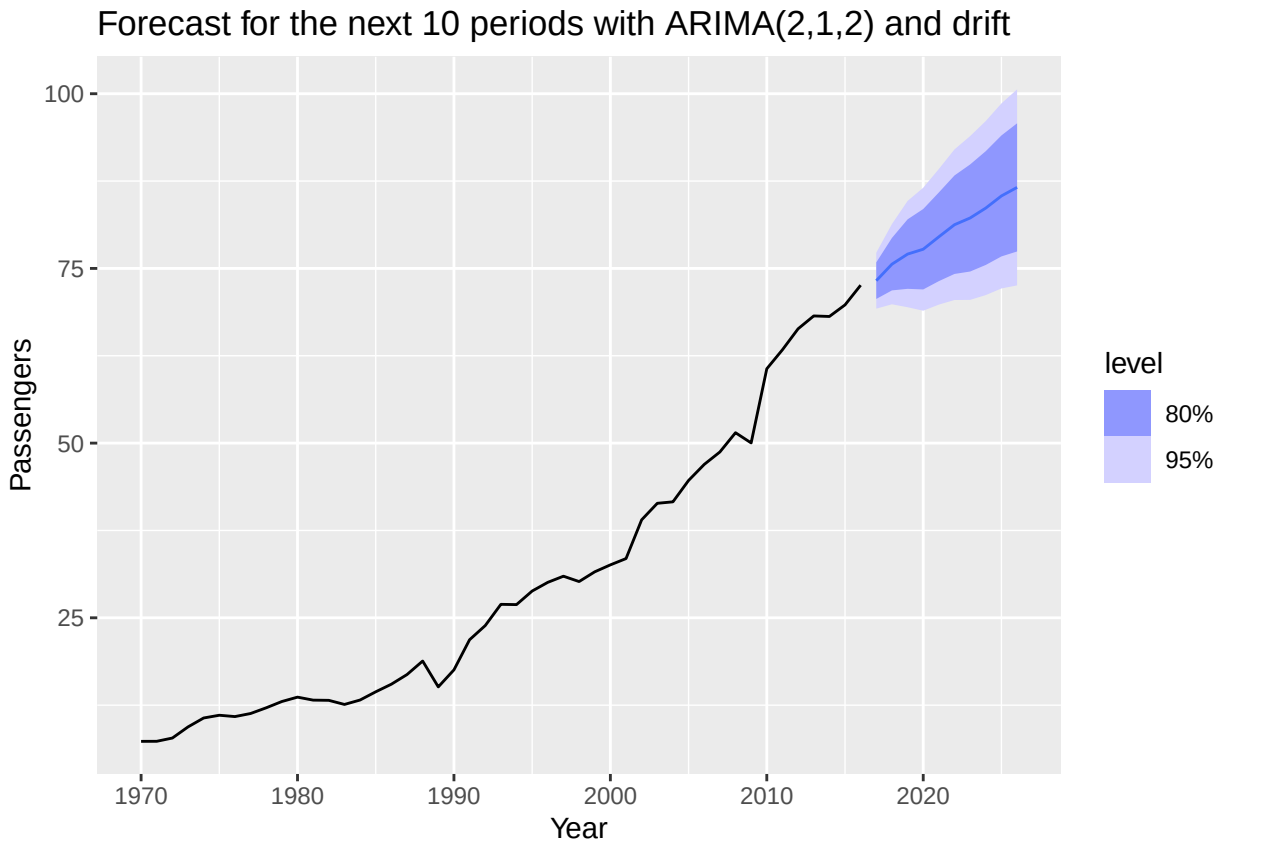

```
arima_down_forecast <- modeler_drifter %>%
  select(arima_drifter) %>%
  forecast(h = 10) %>%
  autoplot(aus_airpassengers) +
  ggtitle("Forecast for the next 10 periods with ARIMA(0,1,0) and drift")
```

D. Plot forecasts from an ARIMA(2,1,2) model with drift and compare these to parts a

Answer: In comparing the two ARIMA models, the forecast for ARIMA(2,1,2) with drift clearly shows a more pronounced upward trend with bigger confidence intervals as the projections go further out. This increased uncertainty is most likely driven by the higher order parameters.With the ARIMA(1,2,2), we have a model showing a similar upward trajectory but with smaller confidence intervals, a good sign for a better forecast. It's also benefiting from no drift, making the growth less steep.

```
fit_212 <- aus_airpassengers %>%
  model(arima_drifter = ARIMA(Passengers ~ pdq(2,1,2) + 1))

fit_212 %>%
  forecast(h = 10) %>%
```

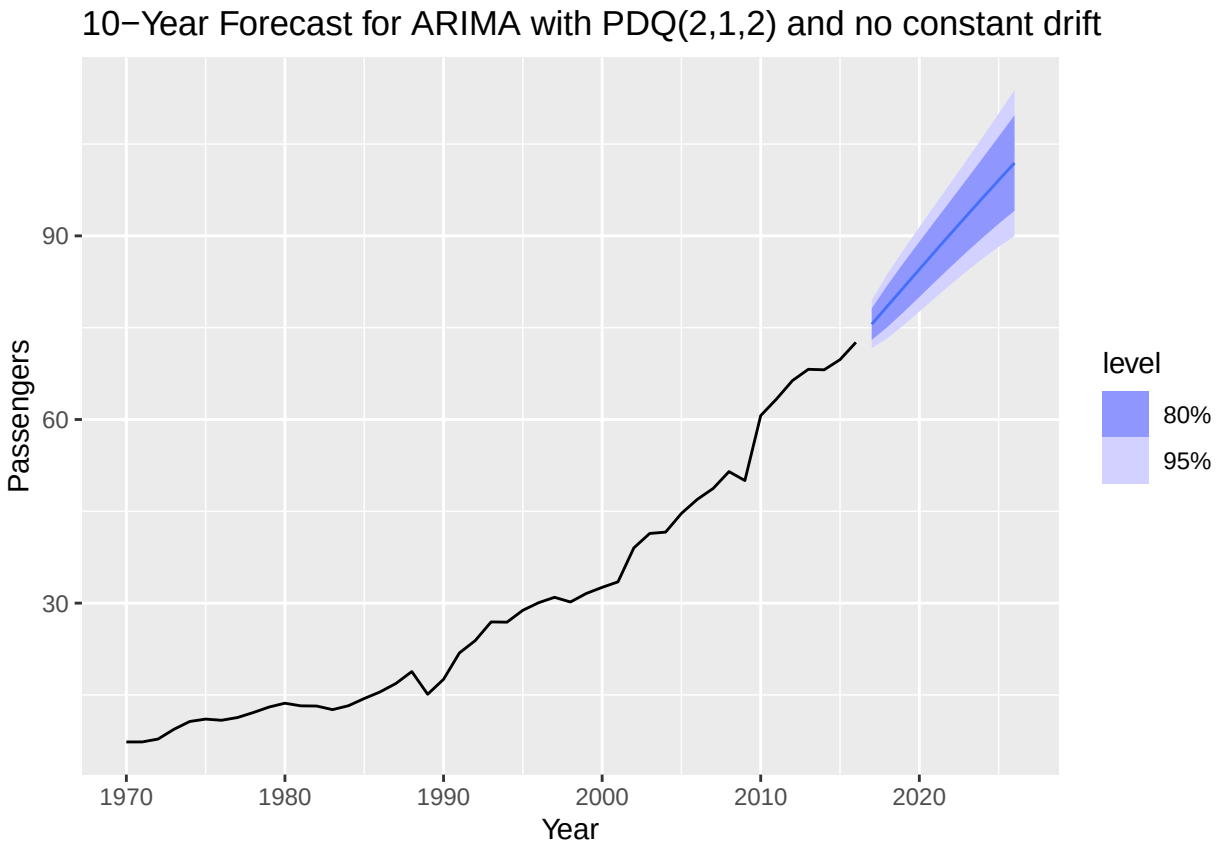
```
  autoplot(aus_airpassengers) +
  ggtitle("Forecast for the next 10 periods with ARIMA(2,1,2) and drift")
```




```
fit_212_forecast <- fit_212 %>%
  forecast(h = 10)
```

D.1 Remove the constant and see what happens Answer: Removing the drift created tighter confidence intervals and a forecast going more firmly up.The lack of drift is eliminating the downward pressure, which appears to create more variance as the data actually wants to be on the steeper trajectory.

```
fit_212_noconstant <- aus_airpassengers %>%
  model(
  `arima_noconstant` = ARIMA(Passengers ~ 0 + pdq(2,1,2), method="ML")
  )

arima_constantless_forecast <- fit_212_noconstant |>
     select(arima_noconstant) |>
    forecast(h = 10)

arima_constantless_forecast |>
     autoplot(aus_airpassengers) +
     ggtitle("10-Year Forecast for ARIMA with PDQ(2,1,2) and no constant drift")
```



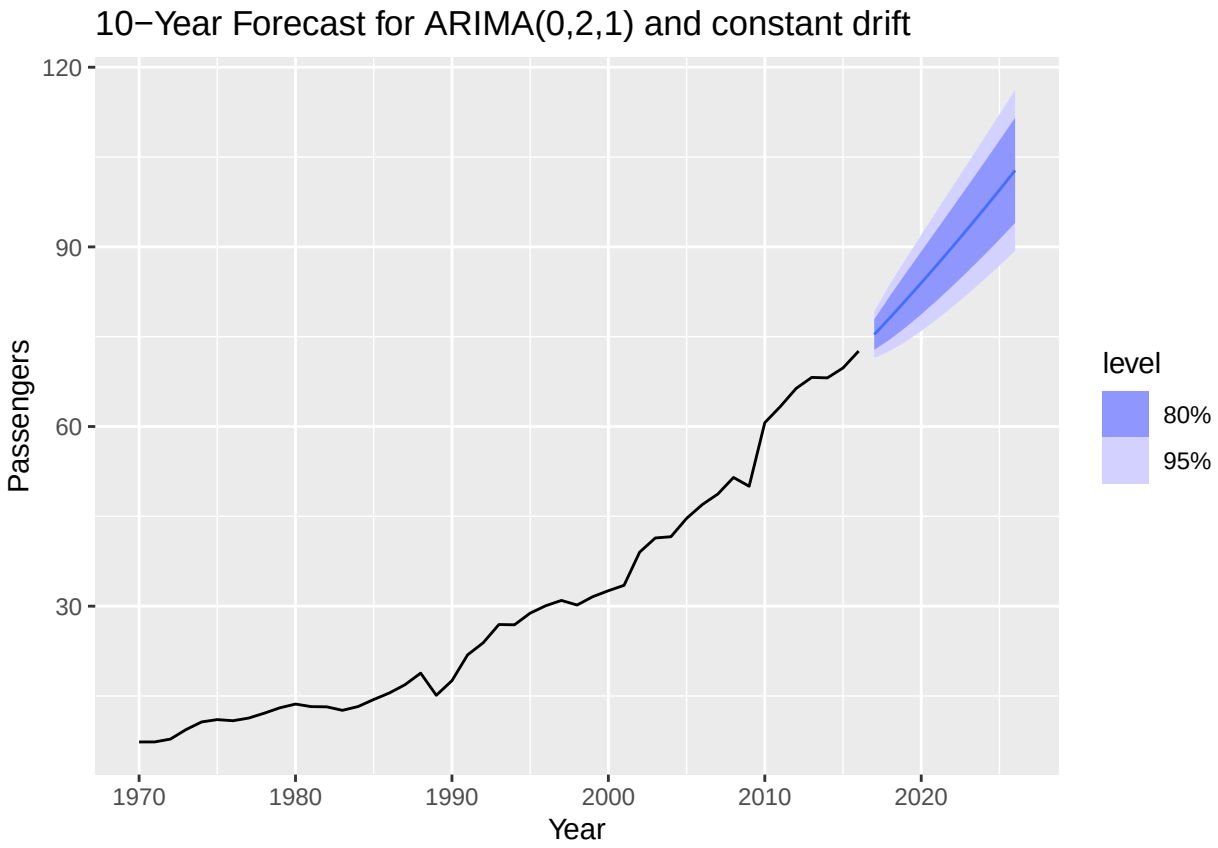
E. Plot forecasts from an ARIMA(0,2,1) model with a constant. What happens?

Answer: When comparing these two, the forecast for ARIMA(2,1,2) with no constant drift shows a more gradual upward trend with relatively narrow confidence intervals, suggesting a steadier forecast with lower uncertainty. On the other hand, the ARIMA(0,2,1) model with constant drift displays a steeper increase in the forecasted values, with wider confidence intervals, indicating higher variability and uncertainty in future outcomes. The inclusion of the constant drift in the second model likely accounts for the stronger upward trajectory, while the first model without drift projects a more conservative and stable growth pattern. Both models reflect the underlying upward trend, but the drift in the ARIMA(0,2,1) model amplifies the expected growth over the forecast horizon.

```
fit_021_constant <- aus_airpassengers %>%
  model(
  `arima_con_constant` = ARIMA(Passengers ~ 1 + pdq(0,2,1), method="ML")
  )
```

```
## 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.
```

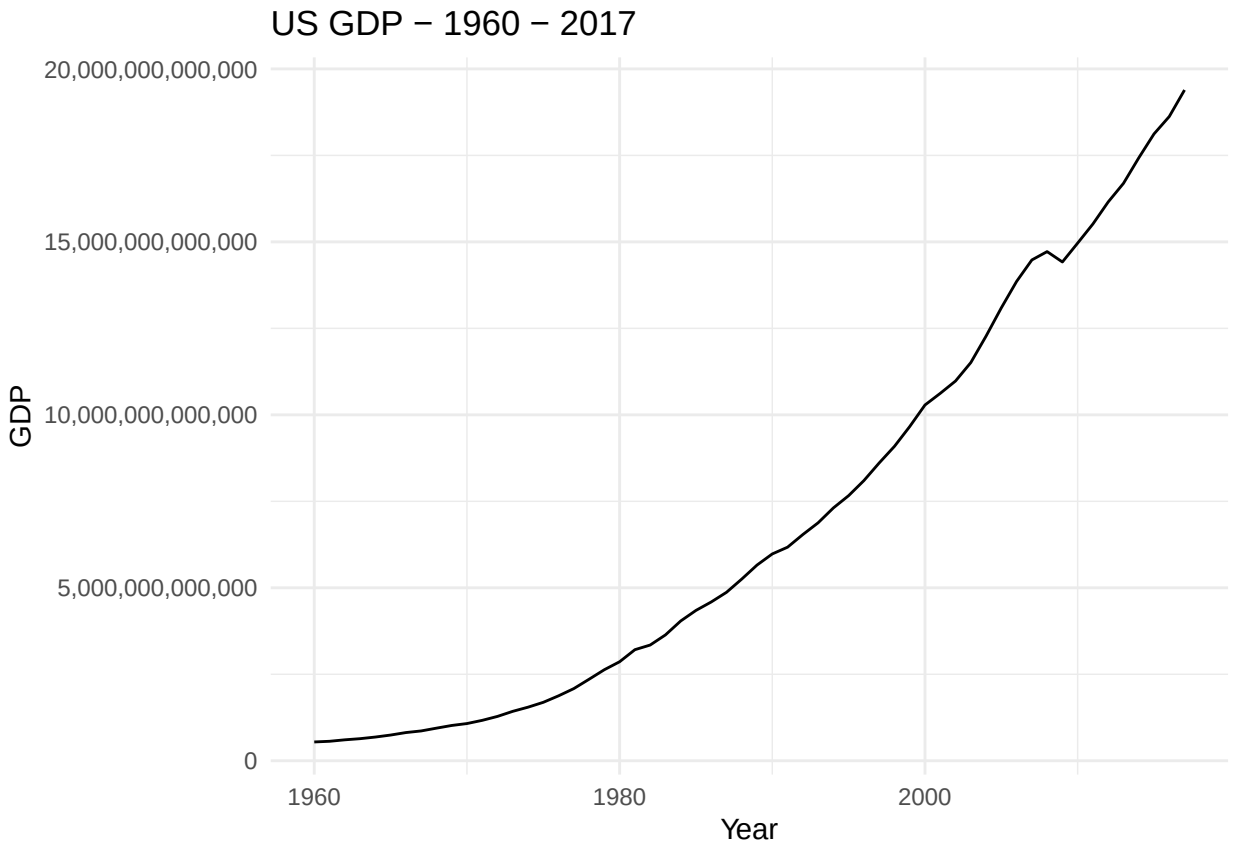
```
ohtwoone_forecast <- fit_021_constant |>
     select(arima_con_constant) |>
    forecast(h = 10)

ohtwoone_forecast |>
     autoplot(aus_airpassengers) +
     ggtitle("10-Year Forecast for ARIMA(0,2,1) and constant drift")
```



9.8 Analysis of United States GDP from global_economy

This exercise consists of component questions A through E

```
us_gdp <-
  global_economy |>
  filter(Code == "USA") |>
  select(GDP)

us_gdp %>%
  autoplot(GDP) +
    labs(title = "US GDP - 1960 - 2017",
       x = "Year",
       y = "GDP") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



A. If necessary, find a suitable Box-Cox transformation for the data

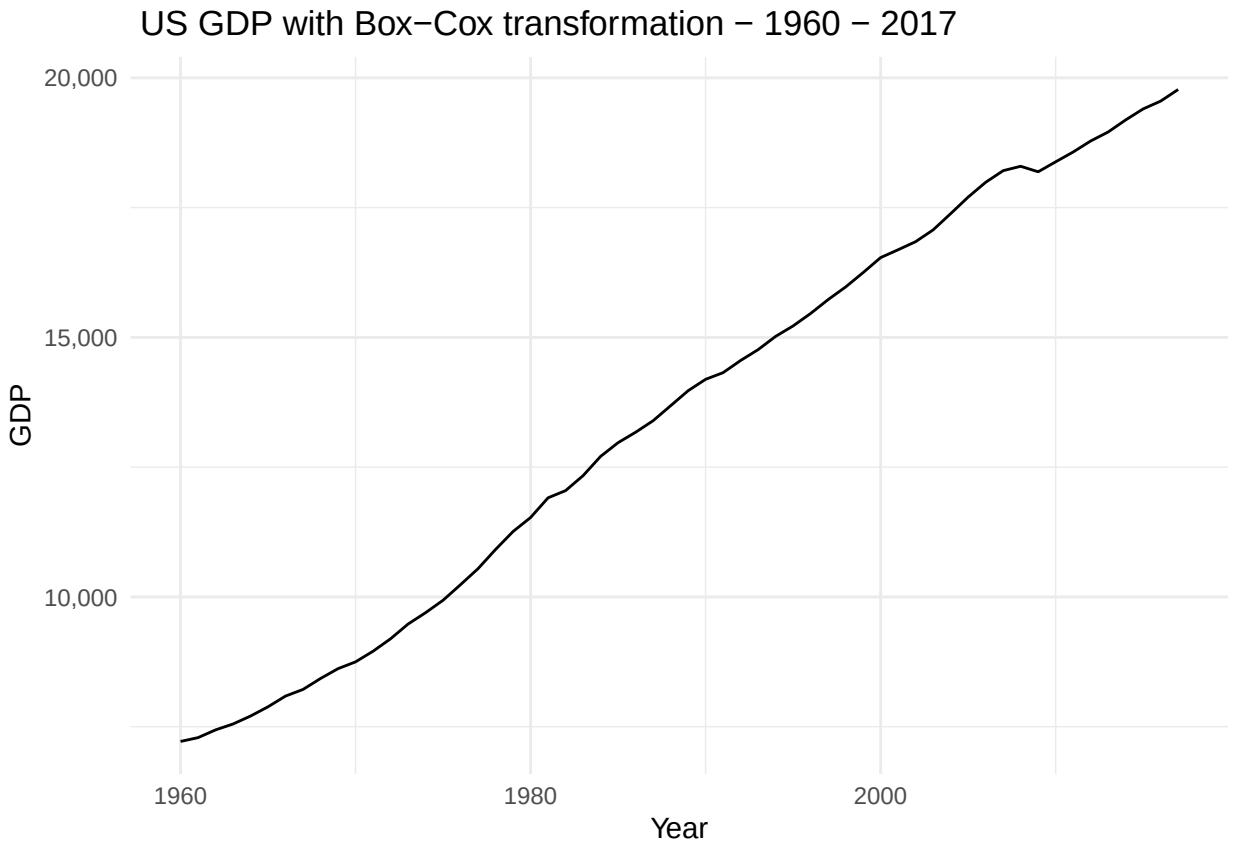
I will use a Box-Cox transformation because the GDP data shows an exponential growth pattern, caused increased variance over time. A Box-Cox will stabilize this variance and making the forecast more reliable by reducing skewness.

```
us_lambda <- us_gdp |>
  features(GDP, features = guerrero) |>
  pull(lambda_guerrero)

us_ndiffs <- us_gdp |>
  features(box_cox(GDP, us_lambda), features = unitroot_ndiffs) |>
  pull(ndiffs)

us_nsdiffs <- us_gdp |>
  features(box_cox(GDP, us_lambda), unitroot_nsdiffs) |>
  pull(nsdiffs)

cat("Optimal lambda: ", us_lambda,
    "\nOptimal first diffs ", us_ndiffs,
    "\nOptima; seasonal diffs: ", us_nsdiffs)
```

```
## Optimal lambda:  0.2819443
## Optimal first diffs  1
## Optima; seasonal diffs:  0
```

Optimal lambda: 0.2819443 Optimal first diffs 1 Optima; seasonal diffs: 0

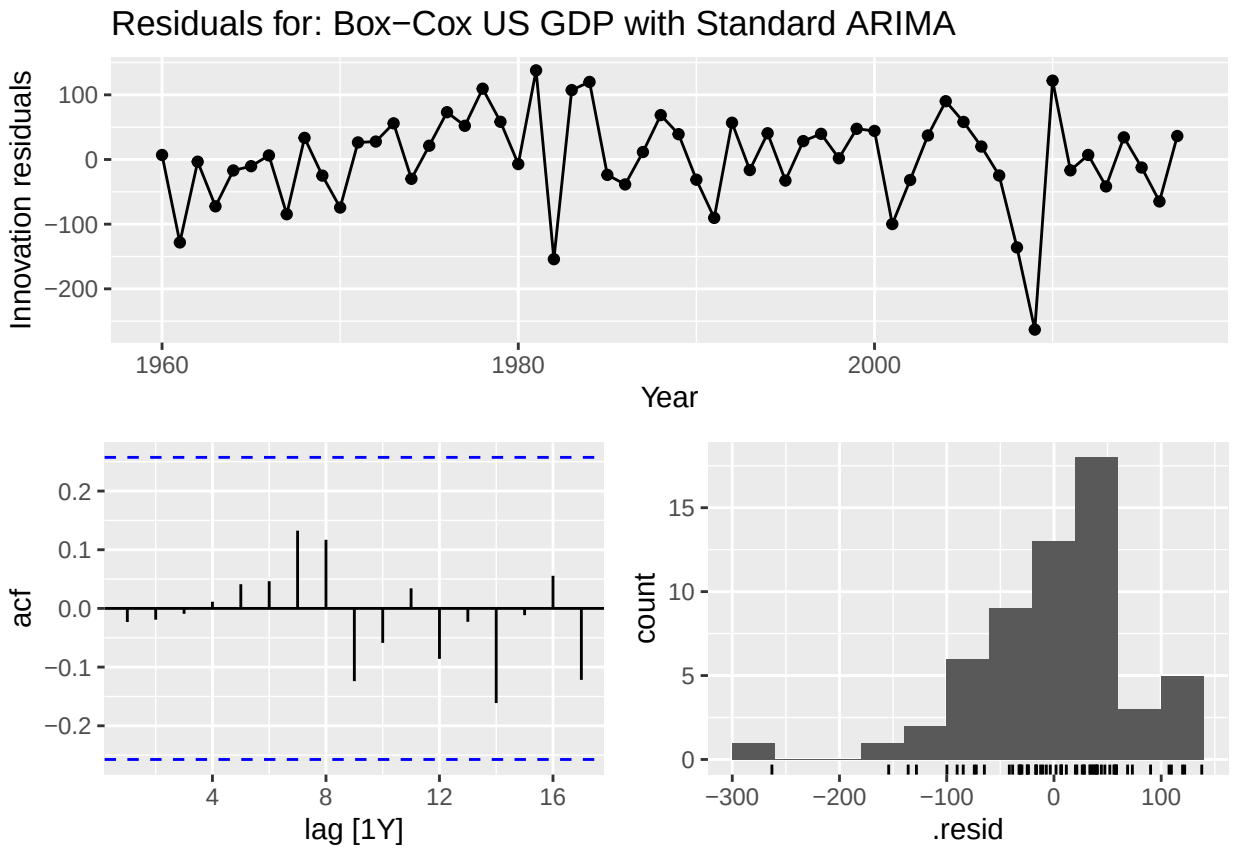
```
us_gdp_bx <- us_gdp |>
  mutate(box_gdp = box_cox(GDP,us_lambda))

us_gdp_bx %>%
  autoplot(box_gdp) +
    labs(title = " US GDP with Box-Cox transformation - 1960 - 2017",
       x = "Year",
       y = "GDP") +
    scale_y_continuous(labels = scales::comma_format()) +
    theme_minimal()
```



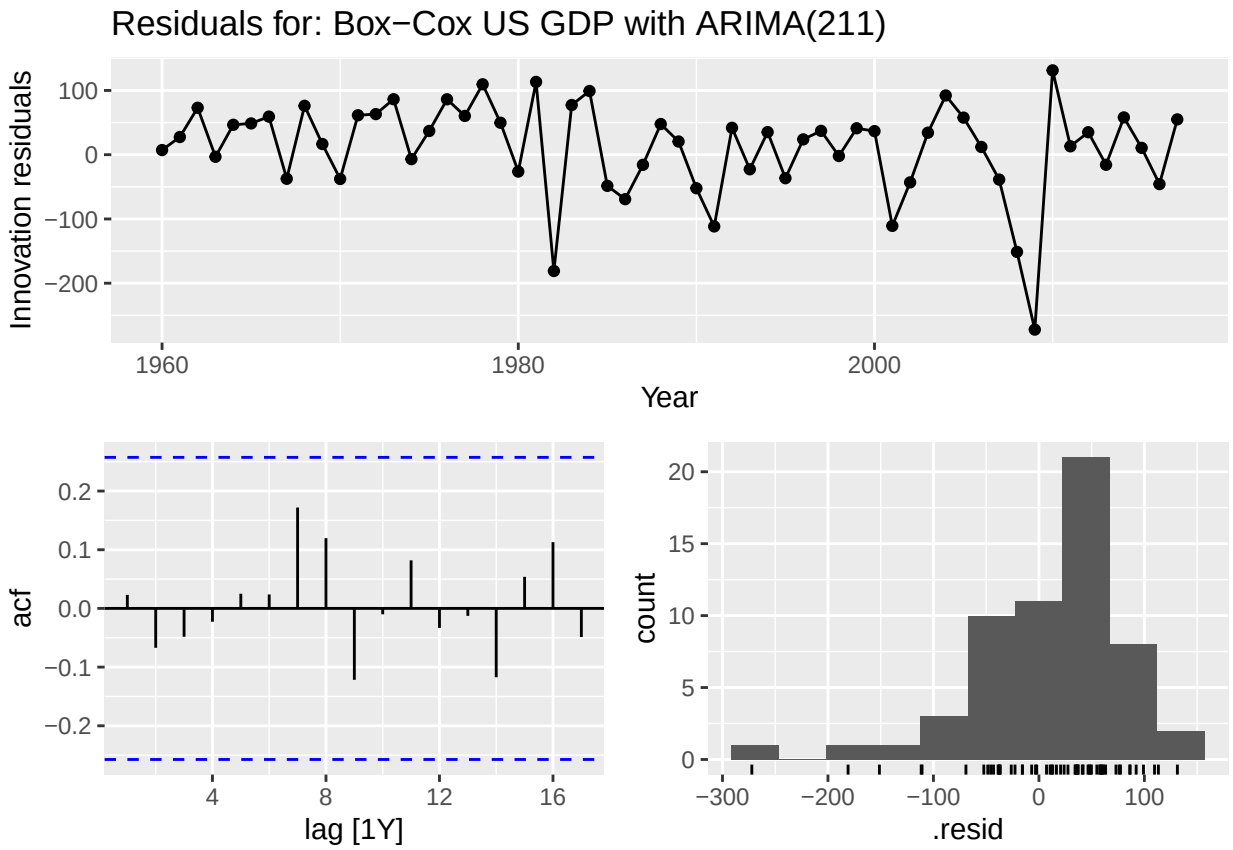
The Box-Cox transformation took it from an exponential growth arch to a more straight-line pattern.

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

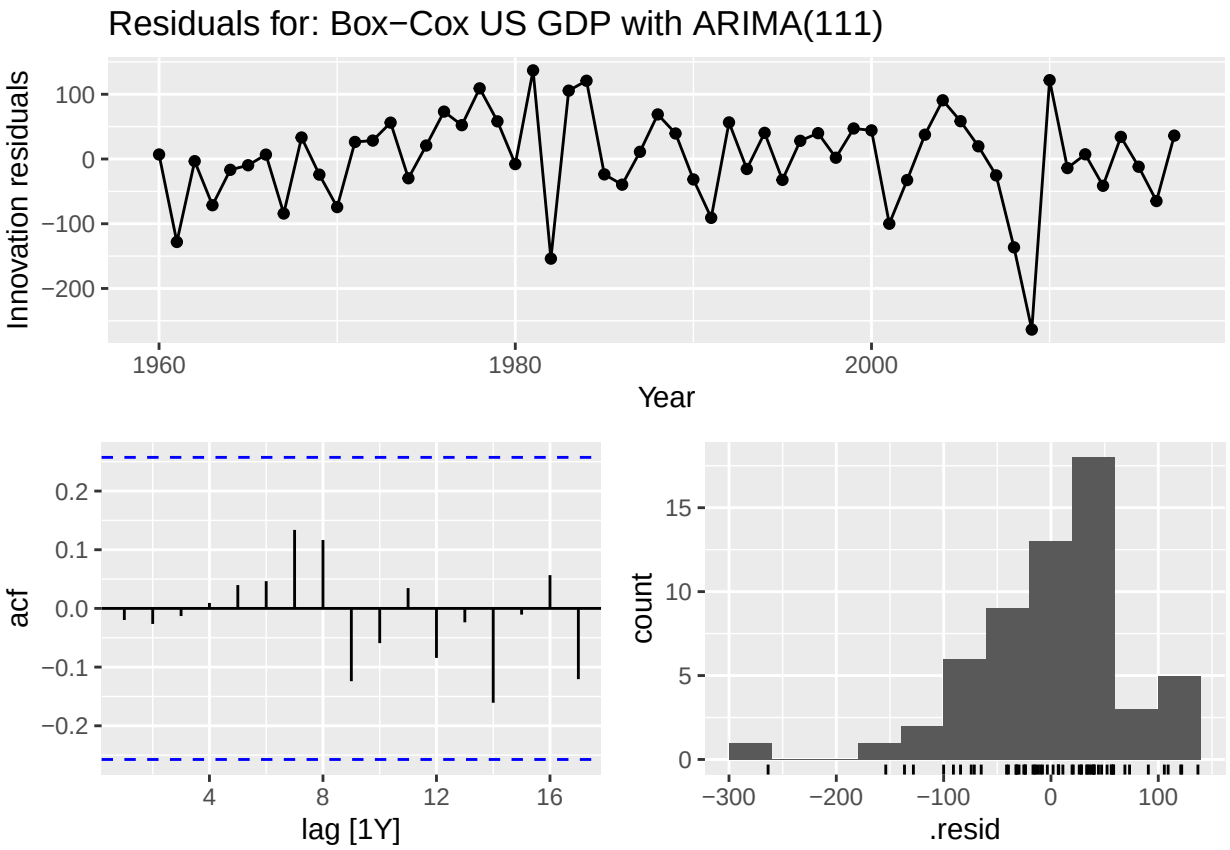
```
usgdp_modeler <- us_gdp_bx |>
  model(
    standard_arima = ARIMA(box_gdp),
    arima_211 = ARIMA(box_gdp ~ pdq(2,1,1) + 0),
    arima_111 = ARIMA(box_gdp ~ pdq(1,1,1)),
    arima_nogreed = ARIMA(box_gdp, greedy = FALSE)
  )

usgdp_modeler |>
  select(standard_arima) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: Box-Cox US GDP with Standard ARIMA")
```




```
usgdp_modeler |>
  select(arima_211) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: Box-Cox US GDP with ARIMA(211)")
```



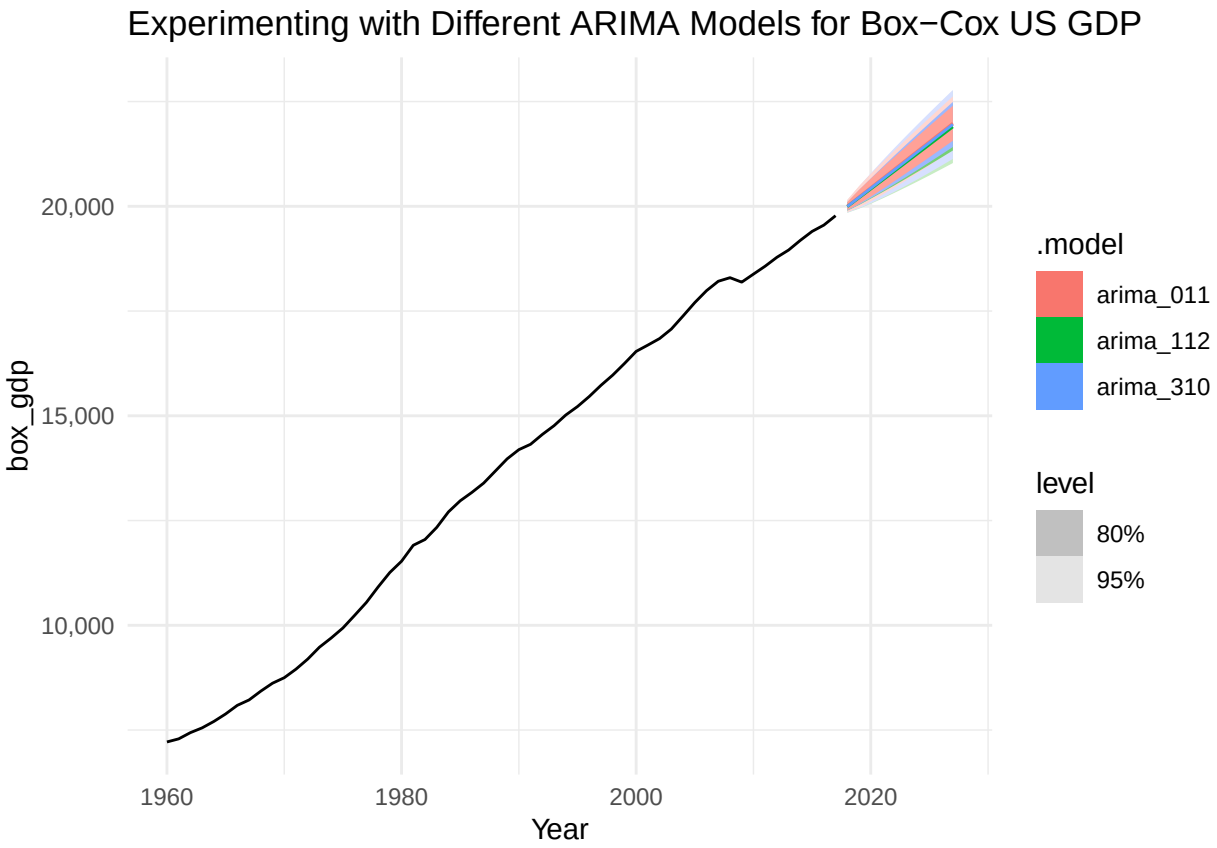

```
usgdp_modeler |>
  select(arima_111) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: Box-Cox US GDP with ARIMA(111)")
```



The ARIMA(1,1,1) model works best of the three because it has residuals with minimal autocorrelation. It also has its residuals tucked close to zero, indiciating a good overall fit. The histogram has something close to a normal distribution, save the sharp drop off the peak. It looks likw the model has managed to over overfitting while accurately showinf the patterns in the Box-Cox GDP.

C. Try some other plausible models by experimenting with the orders chosen

```
usgdpbx_model_redux <- us_gdp_bx |>
  model(
    arima_011 = ARIMA(box_gdp ~ pdq(0,1,1)),
    arima_310 = ARIMA(box_gdp ~ pdq(3,1,0)),
    arima_112 = ARIMA(box_gdp ~ pdq(1,1,2))
  )

usgdpbx_model_redux_fc <- usgdpbx_model_redux |>
  forecast(h = 10)

usgdpbx_model_redux_fc |>
  autoplot(us_gdp_bx) +
  labs(title = "Experimenting with Different ARIMA Models for Box-Cox US GDP") +
  scale_y_continuous(labels = scales::comma_format()) +
  theme_minimal()
```



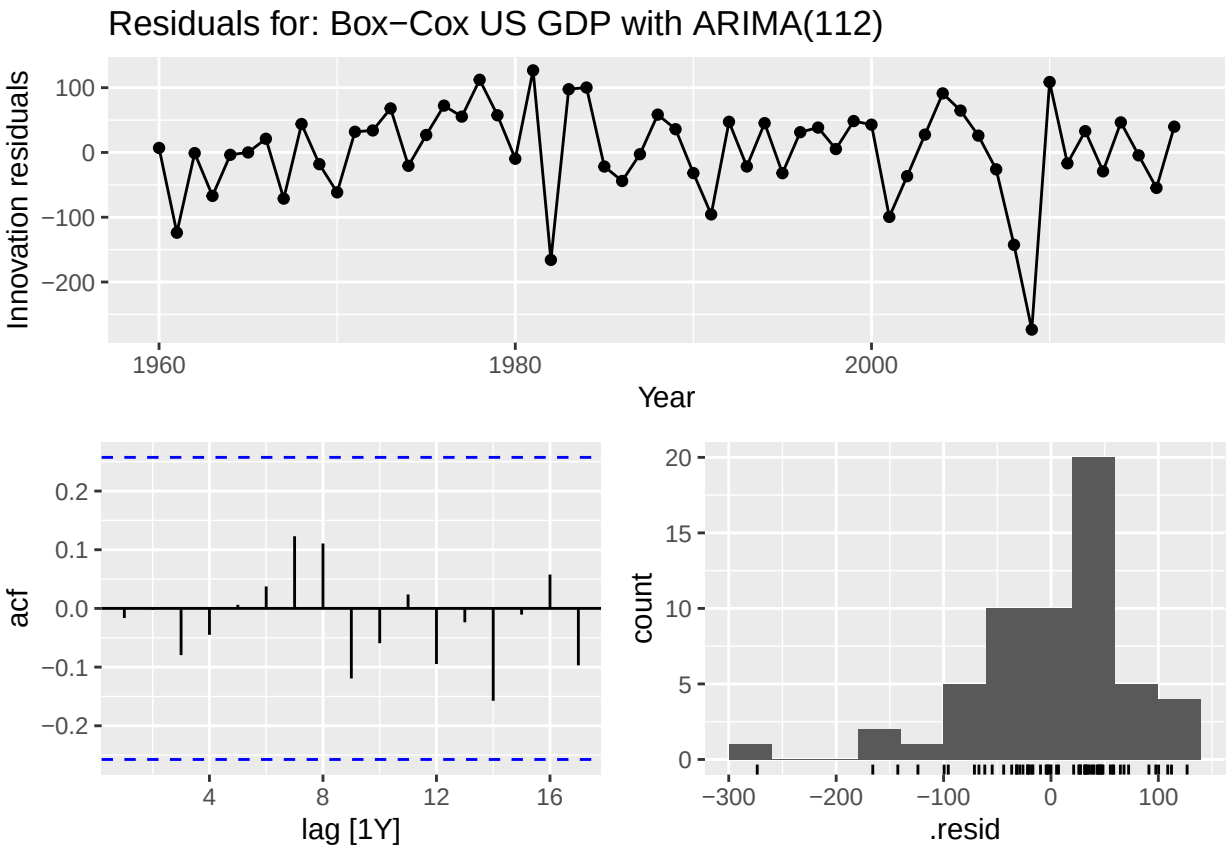
All three of these ARIMA models ahow a similar upward trend, which makes sense considering the overall growth trajectory in historical data. The confidence intervals are pretty narrow, indicating that each model is cpaturing the general growth pattern while only deviating slightly from each other. While ARIMA(3,1,0) forecasts more aggressively than the others, it's not that far off from what the other models projected.

D. Choose what you think is the best model and check the residual diagnostics

Answer:

The ARIMA(1,1,2) (green above) model looks like the best choice. It offers a balanced approach by incorporating both autoregressive and moving average components. The upward growth is accurately forecasted while the confidence intervals are the tightest of the three. The more complex ARIMA(3,1,0) (blue above) forecasts more aggressive growth but the added complexity doesn't appear to provide much upside.

```
usgdpbx_model_redux |>
  select(arima_112) |>
  gg_tsresiduals() +
  ggtitle("Residuals for: Box-Cox US GDP with ARIMA(112)")
```

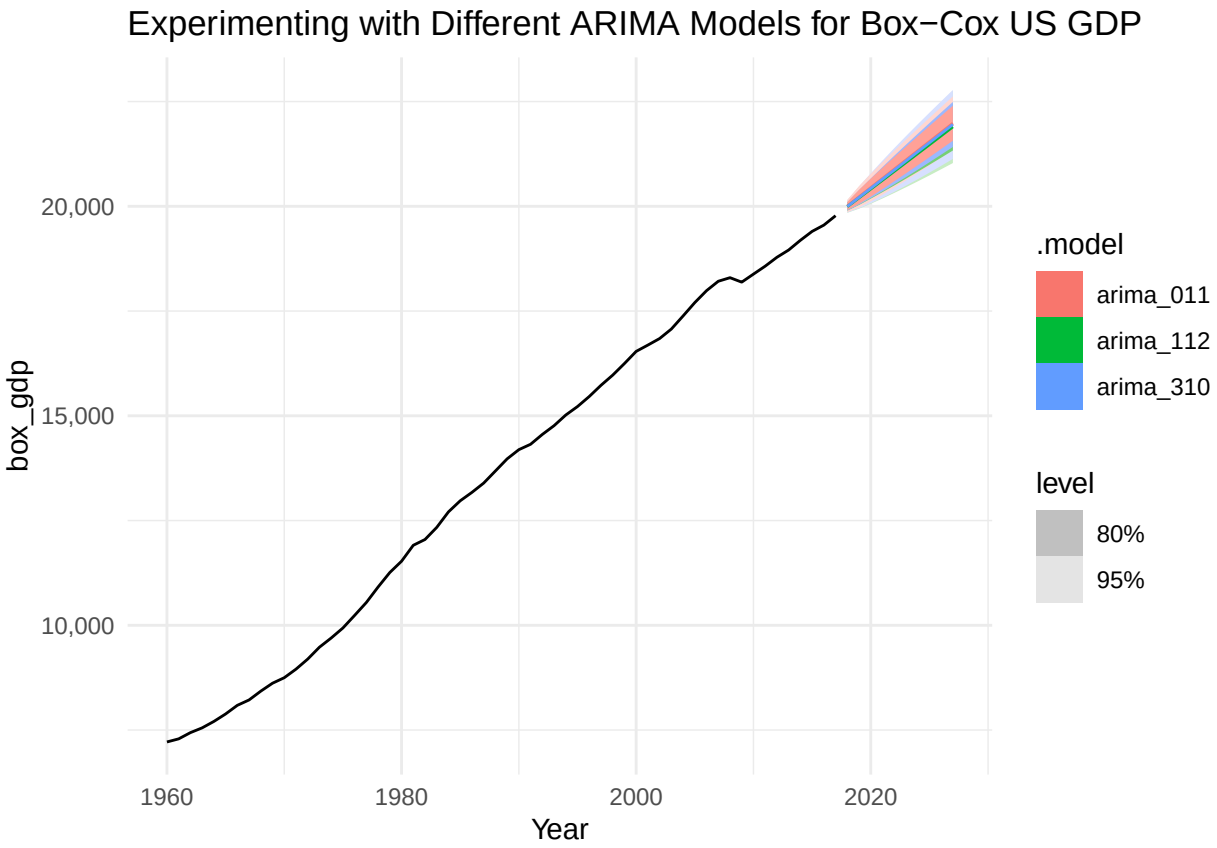


E. produce forecasts of your fitted model. Do the forecasts look reasonable?

Answer: I did this plot above to help answer an earlier component but I'll bring it down here so I can provide more commentary.

I do believe these forecasts are reasonable because they extend in the general direction I would expect the Box-Cox transformed US GDP to go in. The narrow confidence bands also points towards a dataset that has a clear pattern that can be foreasted relatively easily. There isn't the type of variation or noise in the historical data that would make forecasting a particular challenge.

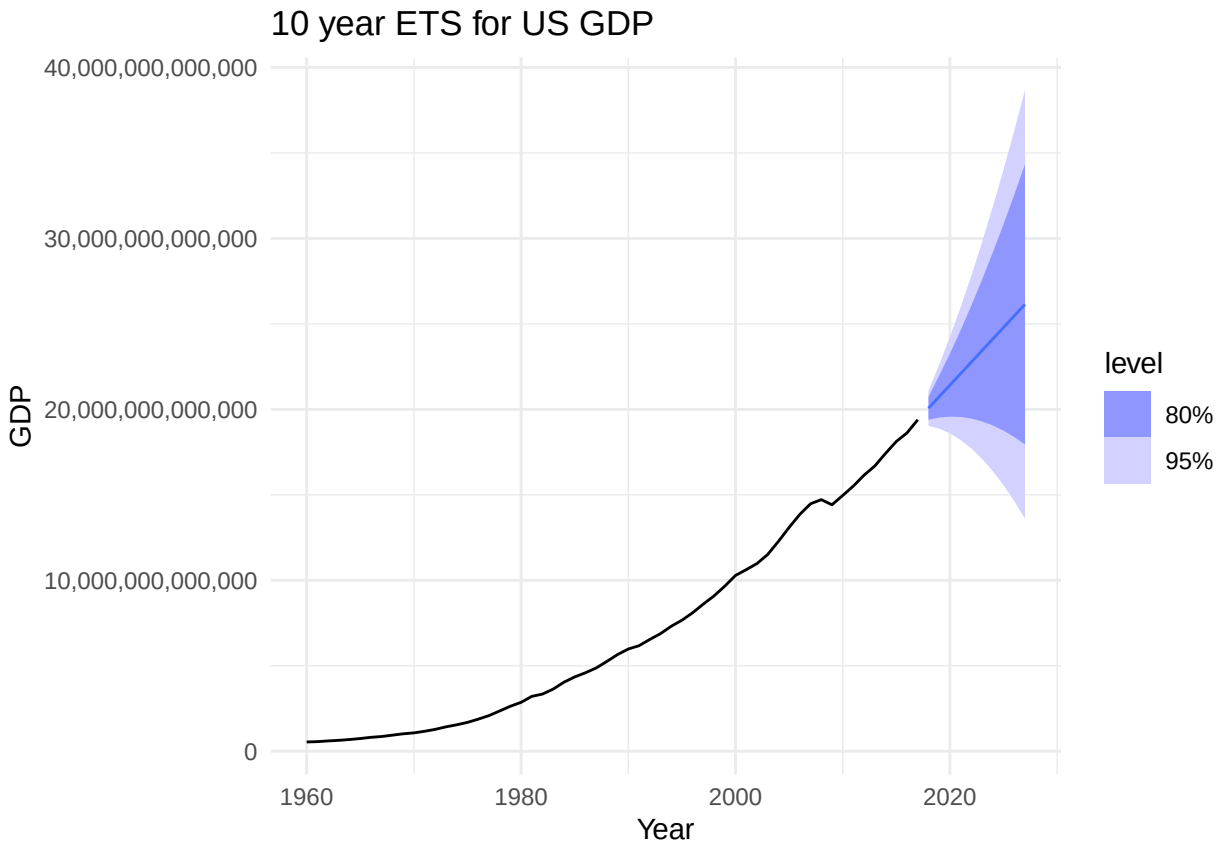
```
usgdpbx_model_redux_fc |>
  autoplot(us_gdp_bx) +
  labs(title = "Experimenting with Different ARIMA Models for Box-Cox US GDP") +
  scale_y_continuous(labels = scales::comma_format()) +
  theme_minimal()
```



F. Compare the results with what you would obtain using ETS() (with no transformation)

Answer: I can see why ARIMA reigns supreme. This ETS forecast has an extremelt large confidence band, including coomponents that send it on a sharp downward trajectory. I wouldn't take this forecast seriously if it was presented to me as a tool for decision making because it doesn't actually inform decision making.

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

usgdp_ets_fc <- usgdp_ets |>
  forecast(h = 10)

usgdp_ets_fc |>
  autoplot(us_gdp) +
  labs(title = "10 year ETS for US GDP") +
  scale_y_continuous(labels = scales::comma_format()) +
  theme_minimal()
```


10 year ETS for US GDP
40,000,000,000,000
30,000,000,000,000
20,000,000,000,000
10,000,000,000,000
0
GDP
1960
1980
2000
2020
Year
level
80%
95%