Data 624: Week 6 Homework

Angrand, Burke, Deboch, Groysman, Karr

October 12, 2019

Week 6 Assignment

Chapter 8 HA 8.1, 8.2,8.6, 8.8

8.1 Figure 8.31 shows the ACFs for 36 random numbers, 360 random numbers and 1,000 random numbers.

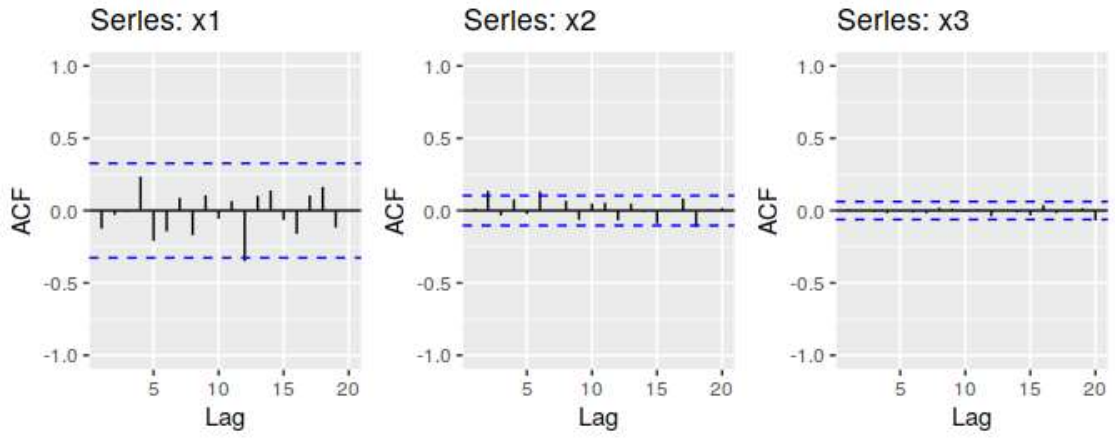


Fig. 8.31

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

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

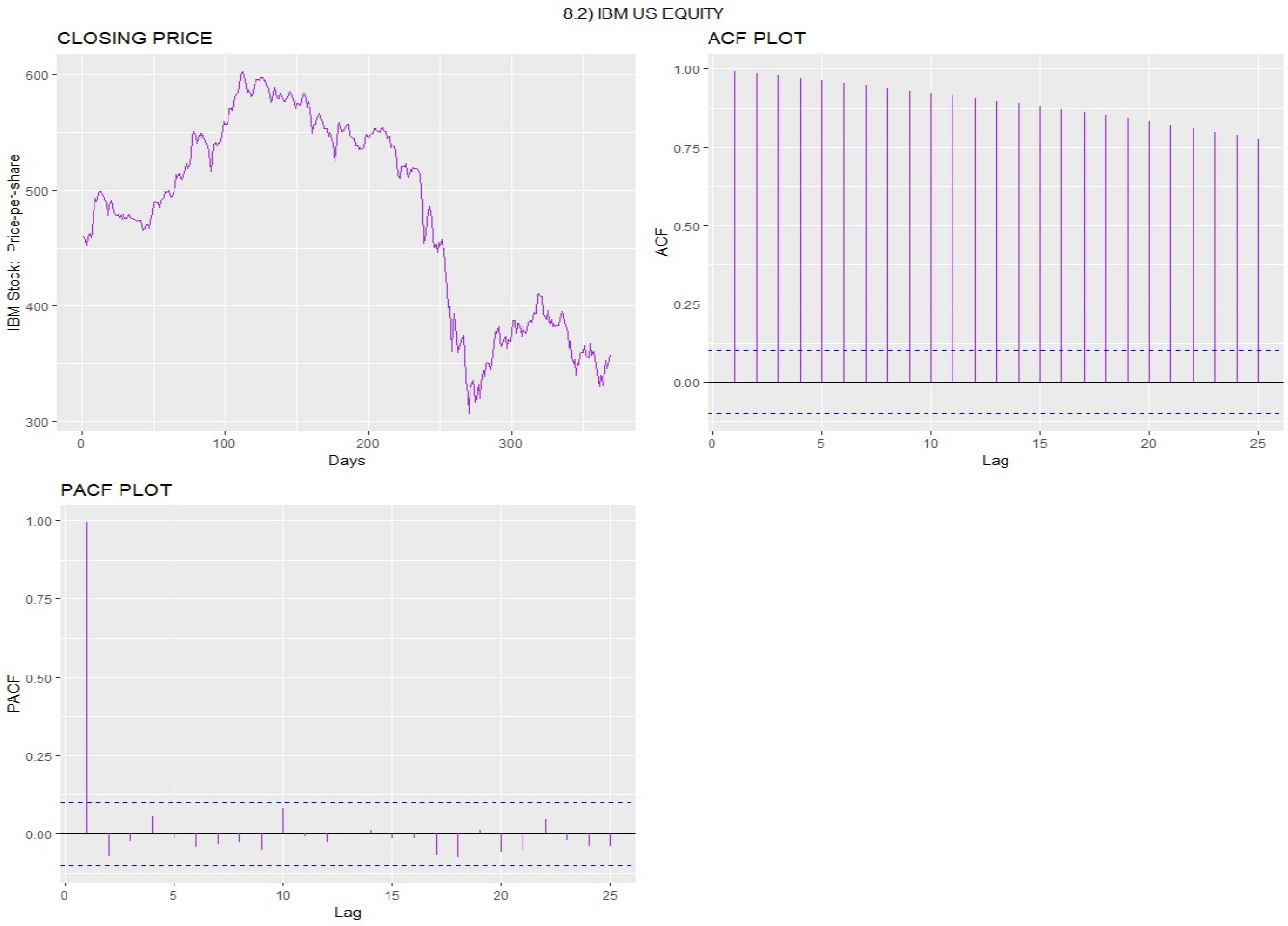
- Autocorrelation is the measurement of how correlated some lag in the time series is to the current lag. For ACF a previous point in time can either be directly or indirectly impacting the current value. For all three series objects, the autocorrelations, displayed with the ACF plots, are within the 95% error terms (essentially no different than zero). The confirms that all three series objects are essentially random and fit the definition of "White Noise" or ARIMA(0, 0, 0). The main difference between all three series objects is the number of samples. The smaller the number samples, the larger the ACF bars as the error terms are a function of the number of samples in a series $(\frac{\pm 1.96}{\sqrt{N}})$.

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 error terms are a function of the number of samples in a series, $\frac{\pm 1.96}{\sqrt{N}}$. The larger the number of samples, N, the closer the significance level/error terms are to zero. The error terms are larger for smaller data set, meaning higher series autocorrelation is needed to reject the null hypothesis that the series autocorrelation is zero. The larger the timeseries, the greater the denominator and the smaller the critical value.

*8.2 A classic example of a non-stationary series is the daily closing IBM stock price series (data set **ibmclose**). Use R to plot the daily closing prices for IBM stock and the ACF and PACF. Explain how each plot shows that the series is non-stationary and should be differenced.*

```
ibm.autoplot<- autoplot(ibmclose,col="purple") + ggtitle("CLOSING PRICE")+xla
b("Days") + ylab("IBM Stock:  Price-per-share")
ibm.acf <- ggAcf(ibmclose,col="purple") + ggtitle("ACF PLOT")
ibm.Pacf <- ggPacf(ibmclose,col="purple") + ggtitle("PACF PLOT")
grid.arrange(
  ibm.autoplot,
  ibm.acf,
  ibm.Pacf,
  nrow = 2,
  top = "8.2) IBM US EQUITY")
```



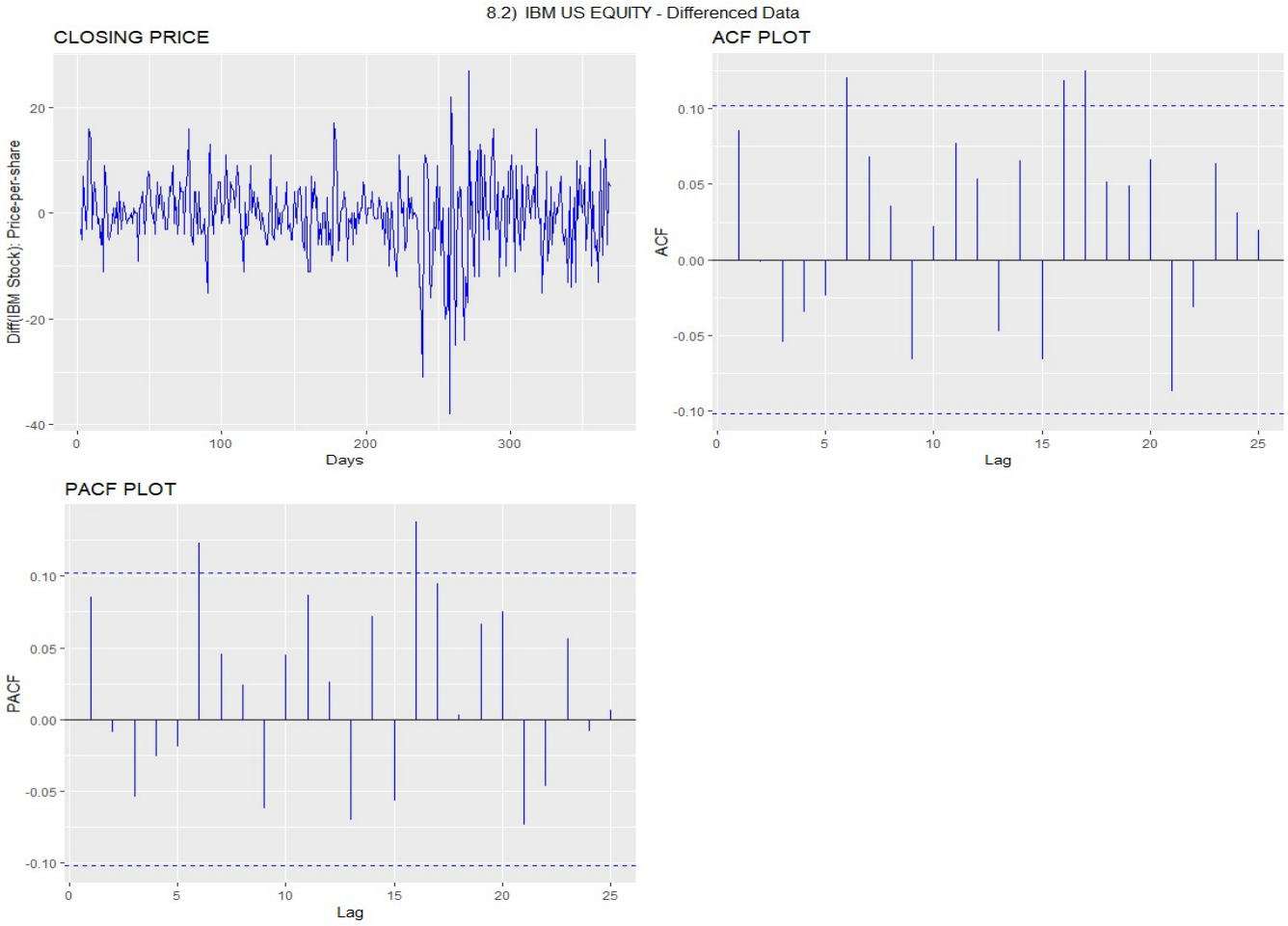
- **IBM Closing Price Plot** Non-stationary data has a mean that is either increasing or decreasing with time. Stationary data has a constant mean/variance the whole series. It does not matter when observed, it should look much the same at any point in time with no predictable patterns in the long-term. In the IBM US EQUITY closing price time series, there is clearly a decreasing trend over time, indicating that this time series needs to be differenced. Notice below in the "Differenced Data", the differenced data exhibits a roughly horizontal plot.

- **IBM Closing Price ACF Plot** ACF plots are great for testing of stationarity. For the IBM trended data, the lags are well outside the 95% "error terms" confidence interval meaning that they are meaningfully and serially correlated. The IBM ACF decreases slowly and its ACF lags are large and positive

- **IBM Closing Price PACF Plot** PACF plots are also great for testing of stationarity. For the IBM trended data, the first lag should be equal to the ACF first lag. The first PACF lag is close to one and all other PACF lags are close to zero, suggesting that the data is non-stationary.

```
ibm.autoplot<- autoplot(ibmclose%>% diff(1),col="blue" ) + ggtitle("CLOSING P
RICE")+xlab("Days") + ylab("Diff(IBM Stock): Price-per-share")
```

```r
ibm.acf <- ggAcf(ibmclose%>% diff(1),col="blue" ) + ggtitle("ACF PLOT")
ibm.Pacf <- ggPacf(ibmclose%>% diff(1),col="blue" ) + ggtitle("PACF PLOT")
grid.arrange(
  ibm.autoplot,
  ibm.acf,
  ibm.Pacf,
  nrow = 2,
  top = "8.2)  IBM US EQUITY - Differenced Data")
```



8.6 Use R to simulate and plot some data from simple ARIMA models.

a. Use the following R code to generate data from an AR(1) model with $\phi_1 = 0.6$ and $\sigma^2 = 1$ and $y_1 = 0$

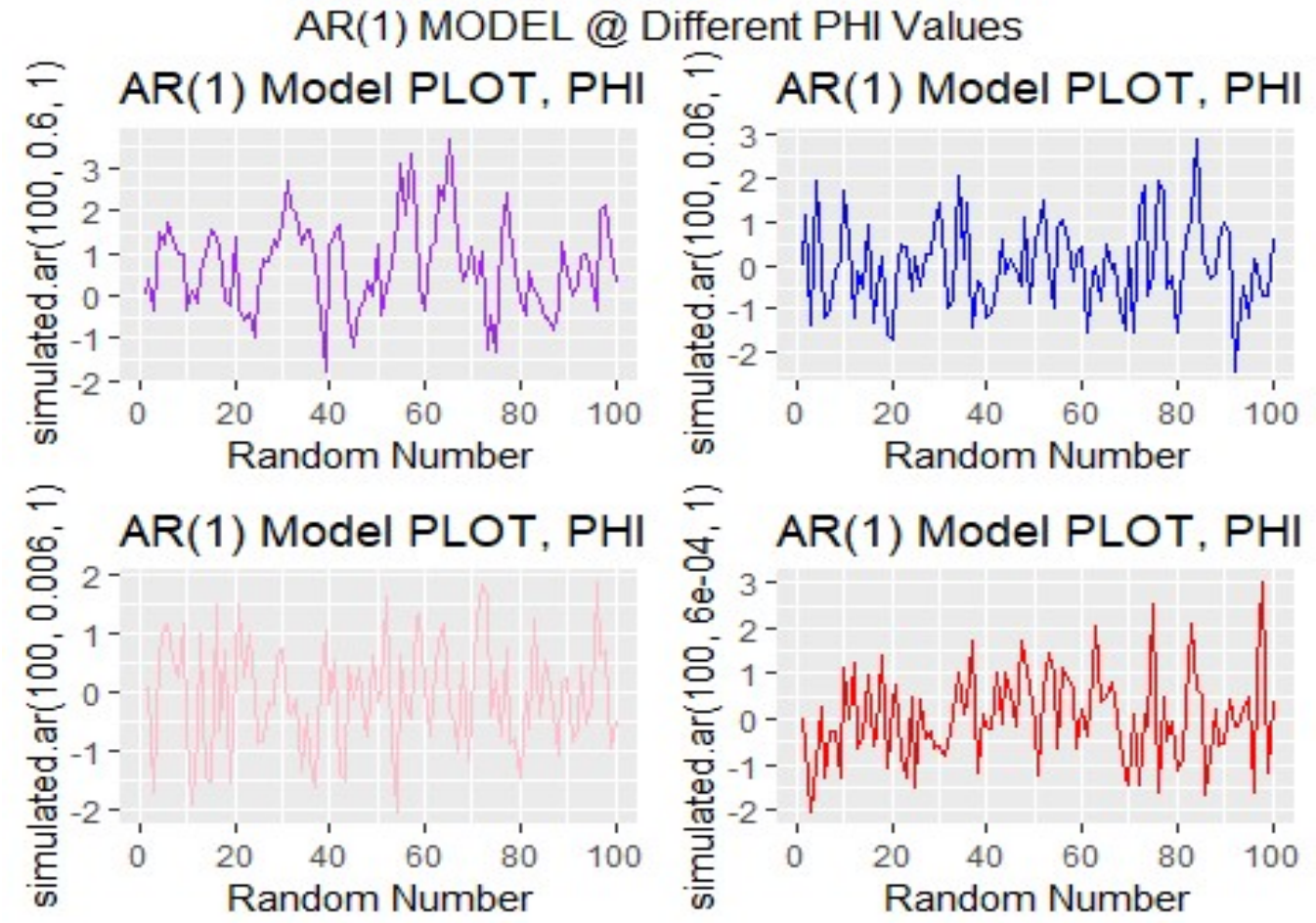
```r
#created simulated.ar function from the given code so it can be reused in later components of the exercise
simulated.ar <- function(N,phi, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)

  for( i in 2:N){
```

```
   y[i] <- phi*y[i-1] + e[i]

  }
  return(y)

}

initial.ar <- simulated.ar(100, 0.6, 1)
autoplot(initial.ar,col="purple")+ ggtitle("AR(1) Model Plot")+
  xlab("Random Number") +
  ylab("AR(1) model")
```

b. Produce a time plot for the series. How does the plot change as you change ϕ_1 ?

```
initial.ar <- autoplot(simulated.ar(100, 0.6, 1),col="purple")+ ggtitle("AR(1
) Model PLOT, PHI = 0.6")+ xlab("Random Number")
second.ar <- autoplot(simulated.ar(100,  0.06, 1), col="blue")+ ggtitle("AR(1
) Model PLOT, PHI = 0.06")+ xlab("Random Number")
third.ar <- autoplot(simulated.ar(100,  0.006, 1),col="pink")+ ggtitle("AR(1)
Model PLOT, PHI = 0.006")+ xlab("Random Number")
fourth.ar <- autoplot(simulated.ar(100,  0.0006, 1),col="red")+ ggtitle("AR(1
) Model PLOT, PHI = 0.0006")+ xlab("Random Number")

grid.arrange(
  initial.ar,
  second.ar,
  third.ar,
```

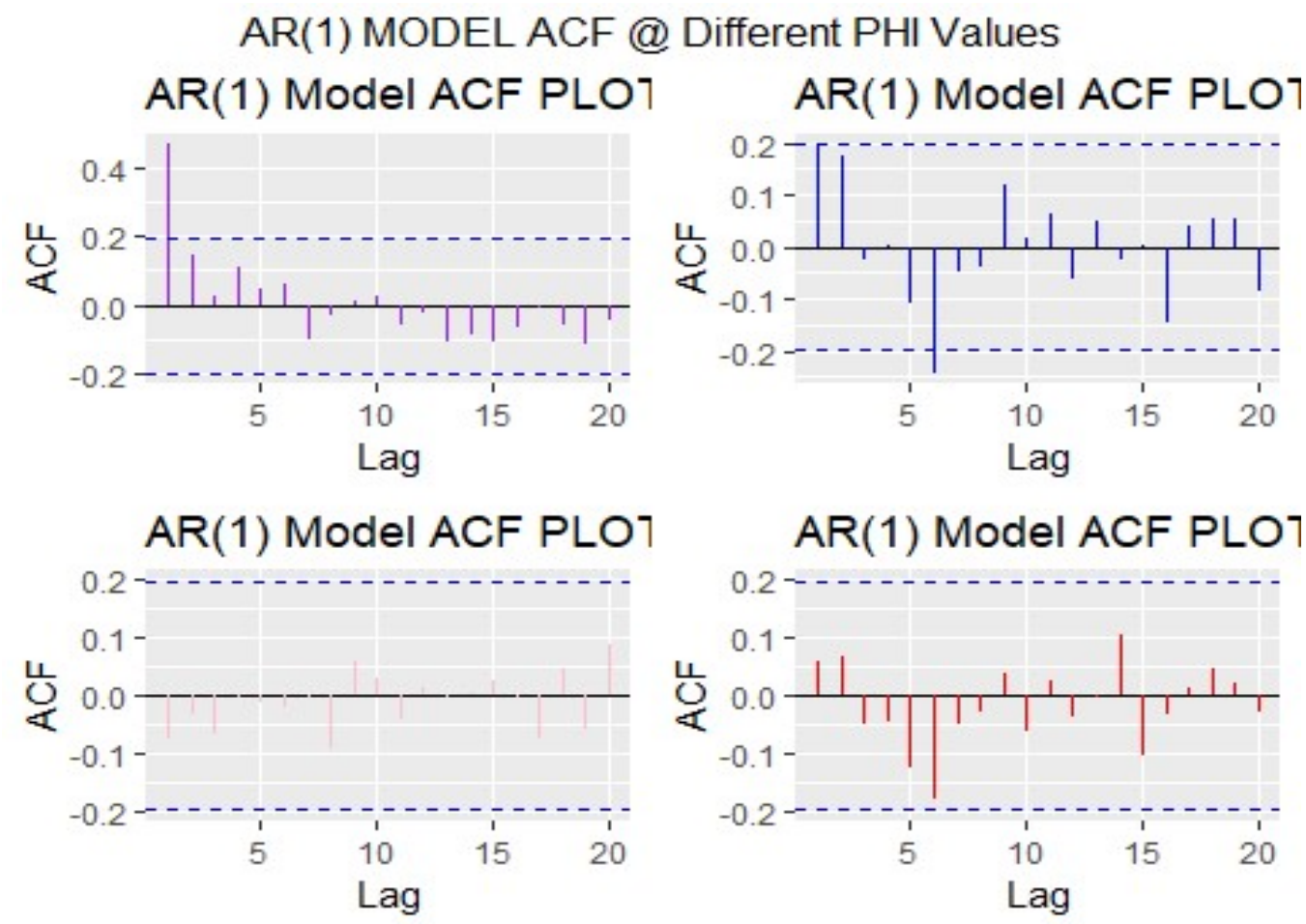
```
  fourth.ar,
  nrow = 2,
  top = "AR(1) MODEL @ Different PHI Values")
```



- Visually, it appears that with smaller ϕ_1 the more random the data. As per the above exercises, the auto correlation lags should be higher for the larger ϕ_1 values. The below ACF plots confirms the higher ACF plots.A change in ϕ_1 is directly proportional to variance in the timeseries.

```
initial.ar.acf <- ggAcf(simulated.ar(100, 0.6, 1),col="purple")+ ggtitle("AR(
1) Model ACF PLOT, PHI = 0.6")
second.ar.acf <- ggAcf(simulated.ar(100,  0.06, 1), col="blue")+ ggtitle("AR(
1) Model ACF PLOT, PHI = 0.06")
third.ar.acf <- ggAcf(simulated.ar(100,  0.006, 1),col="pink")+ ggtitle("AR(1
) Model ACF PLOT, PHI = 0.006")
fourth.ar.acf <- ggAcf(simulated.ar(100,  0.0006, 1),col="red")+ ggtitle("AR(
1) Model ACF PLOT, PHI = 0.0006")

grid.arrange(
  initial.ar.acf,
  second.ar.acf,
  third.ar.acf,
```

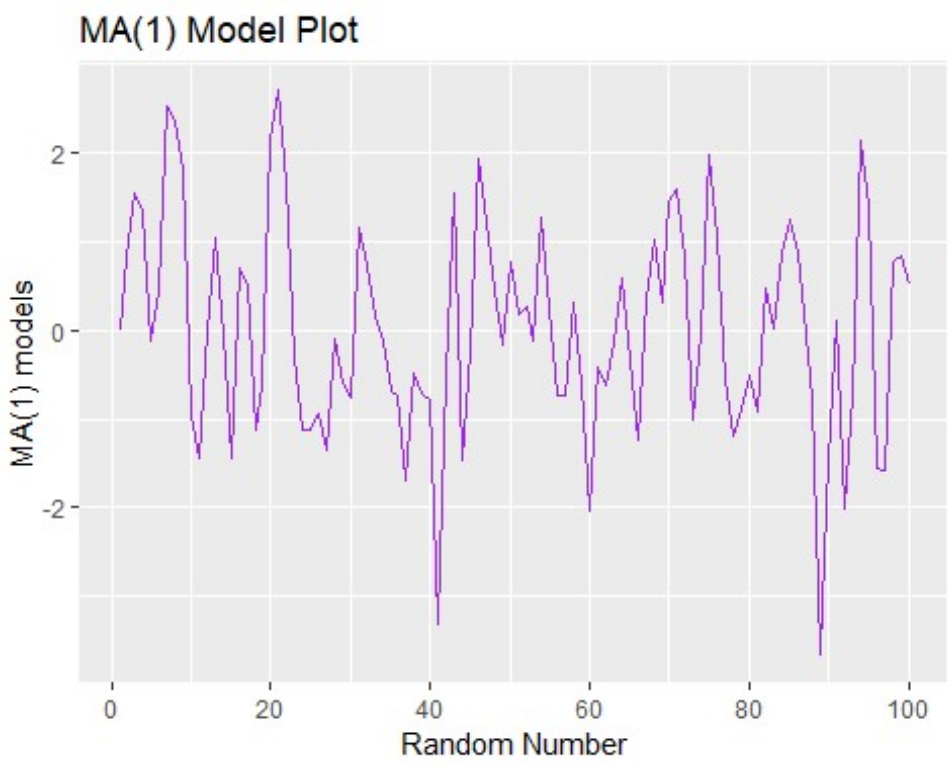
```
  fourth.ar.acf,
  nrow = 2,
  top = "AR(1) MODEL ACF @ Different PHI Values")
```



c. Write your own code to generate data from an MA(1) model with $\theta_1 = 0.6$ and $\sigma^2 = 1$

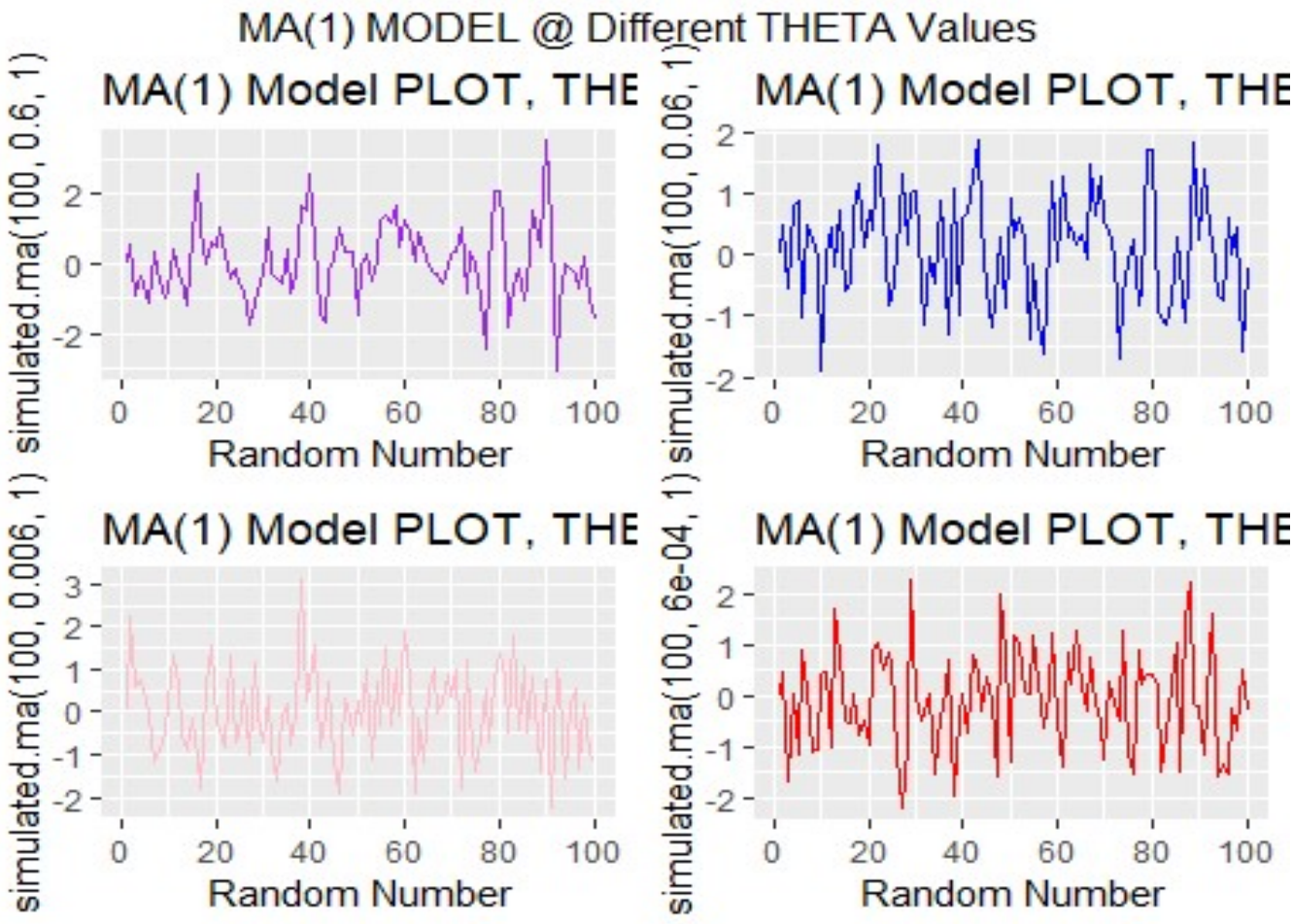
```
simulated.ma <- function(N,theta, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)
  e[1] <- 0
  for( i in 2:N){
   y[i] <- theta*e[i-1] + e[i]

  }
  return(y)

}

initial.ma <- simulated.ma(100, 0.6, 1)
autoplot(initial.ma,col="purple")+ ggtitle("MA(1) Model Plot")   +
```

```
  xlab("Random Number") +
  ylab("MA(1) models")
```



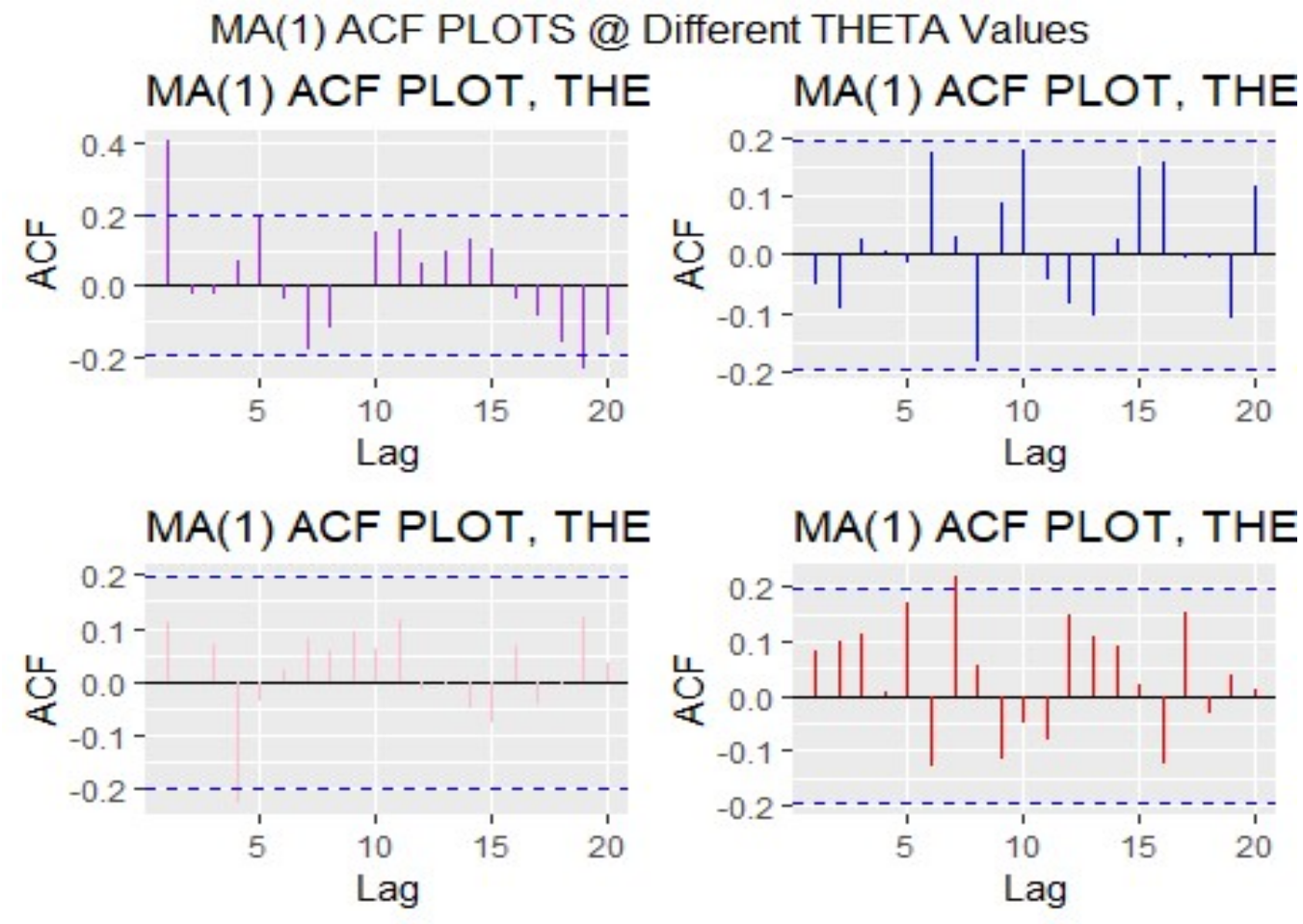
d. Produce a time plot for the series. How does the plot change as you change θ_1 ?

```
initial.ma <- autoplot(simulated.ma(100, 0.6, 1),col="purple")+ ggtitle("MA(1
) Model PLOT, THETA = 0.6")+ xlab("Random Number")
second.ma <- autoplot(simulated.ma(100,  0.06, 1), col="blue")+ ggtitle("MA(1
) Model PLOT, THETA = 0.06")+ xlab("Random Number")
third.ma <- autoplot(simulated.ma(100,  0.006, 1),col="pink")+ ggtitle("MA(1)
Model PLOT, THETA = 0.006")+ xlab("Random Number")
fourth.ma <- autoplot(simulated.ma(100,  0.0006, 1),col="red")+ ggtitle("MA(1
) Model PLOT, THETA = 0.0006")+ xlab("Random Number")

grid.arrange(
  initial.ma,
  second.ma,
  third.ma,
  fourth.ma,
  nrow = 2,
  top = "MA(1) MODEL @ Different THETA Values")
```



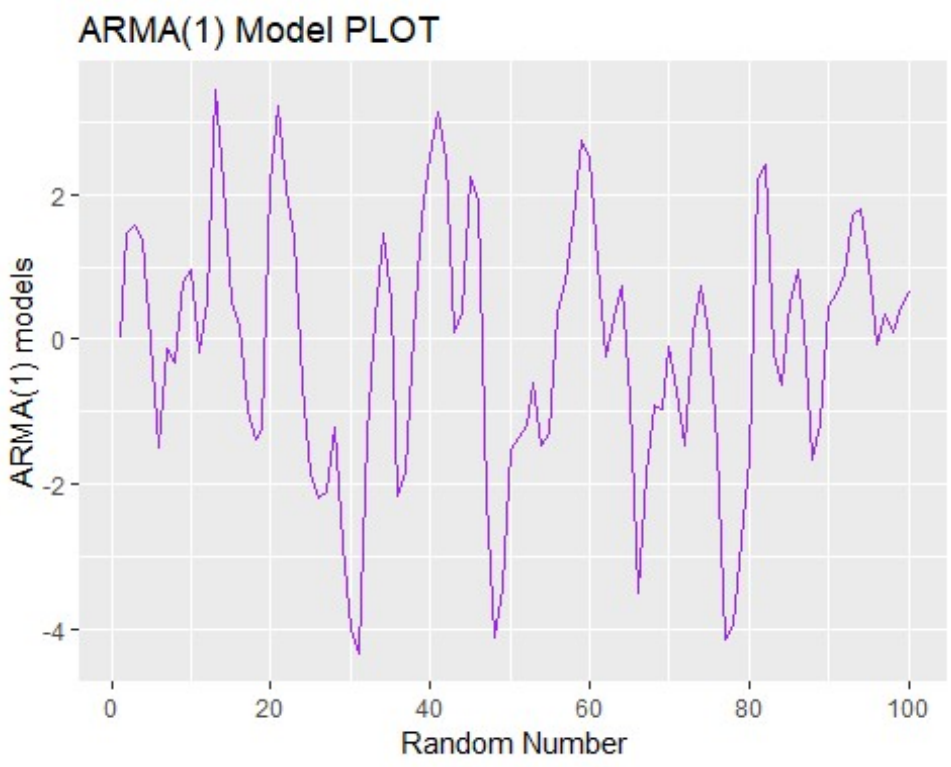
- Similar to AR(1), the smaller θ_1 in the MA(1), the more random the data appears. Below are ACF plots of the MA(1) plots to confirm.

```
initial.ma.acf <- ggAcf(simulated.ma(100, 0.6, 1),col="purple")+ ggtitle("MA(
1) ACF PLOT, THETA = 0.6")
second.ma.acf <- ggAcf(simulated.ma(100,  0.06, 1), col="blue")+ ggtitle("MA(
1) ACF PLOT, THETA = 0.06")
third.ma.acf <- ggAcf(simulated.ma(100,  0.006, 1),col="pink")+ ggtitle("MA(1
) ACF PLOT, THETA = 0.006")
fourth.ma.acf <- ggAcf(simulated.ma(100,  0.0006, 1),col="red")+ ggtitle("MA(
1) ACF PLOT, THETA = 0.0006")

grid.arrange(
  initial.ma.acf,
  second.ma.acf,
  third.ma.acf,
  fourth.ma.acf,
  nrow = 2,
  top = "MA(1) ACF PLOTS @ Different THETA Values")
```



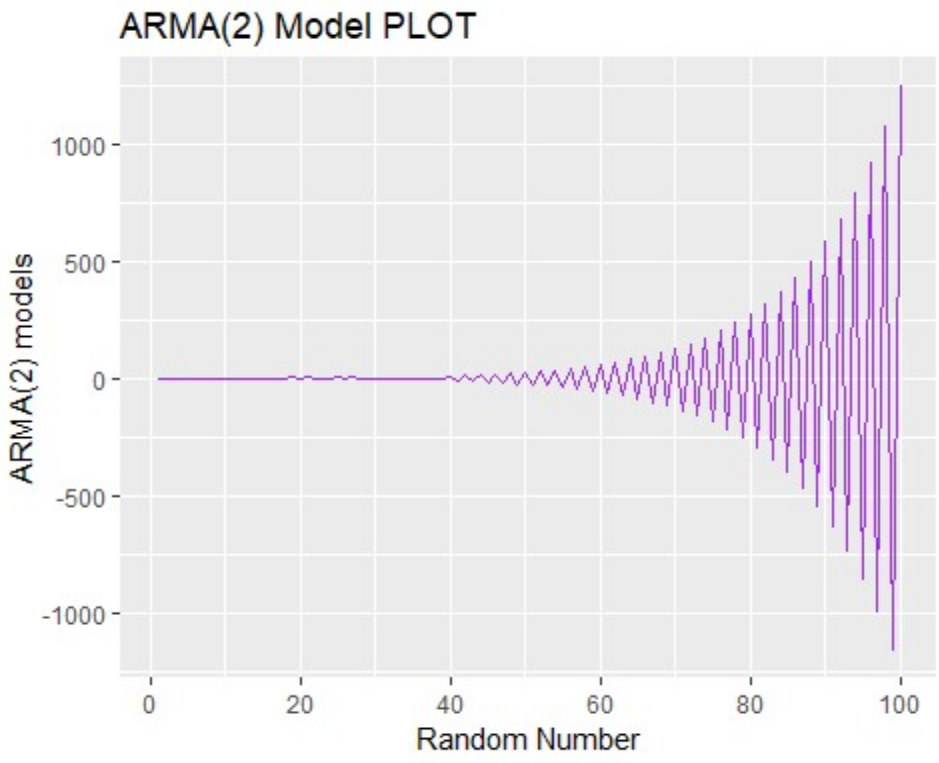
e. Generate data from an ARMA(1,1) model with $\phi_1 = 0.6, \theta_1 = 0.6$ and $\sigma^2 = 1$

```
simulated.arma <- function(N, phi, theta, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)

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

  }
  return(y)

}

initial.arma <- simulated.arma(100, 0.6, 0.6, 1)
autoplot(initial.arma,col="purple")+ ggtitle("ARMA(1) Model PLOT")  +
  xlab("Random Number") +
  ylab("ARMA(1) models")
```



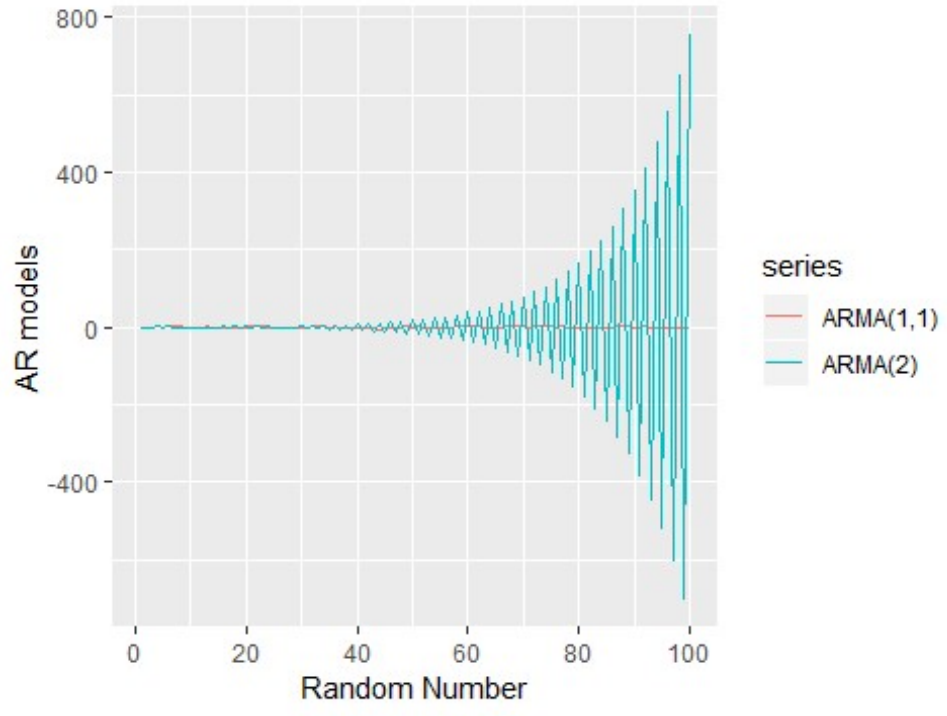
f. Generate data from an ARMA(2) model with $\phi_1 = -0.8$, $\phi_2 = 0.3$ and $\sigma^2 = 1$

```
simulated.arma.ns <- function(N, phi1, phi2, sd ){
  y <- ts(numeric(N))
  e <- rnorm(N, sd = sd)

  #2:N ran an error, changed to 3 and worked. Length zero error
  for( i in 3:N){
    y[i] <- phi1*y[i-1] + phi2*y[i-2] + e[i]


  }
  return(y)

}

non.standard.arma <- simulated.arma.ns(100, -0.8, 0.3, 1)
autoplot(non.standard.arma,col="purple")+ ggtitle("ARMA(2) Model PLOT")  +
  xlab("Random Number") +
  ylab("ARMA(2) models")
```



g. Graph the latter two series and compare them.

```
autoplot(
  simulated.arma(100, 0.6, 0.6, 1), series = "ARMA(1,1)") +
  autolayer(simulated.arma.ns(100, -0.8, 0.3, 1), series = "ARMA(2)")  +
  xlab("Random Number") +
  ylab("AR models")
```



- The AR(2) series exhibits non-stationary traits with a sinusoidal pattern amptitude curve that increases exponentially over time. The AR(2) series appears to have seasonality. The ARMA(1) series does not have this apparent seasonality and it appears to be random and much more stationary than the AR(2) series.

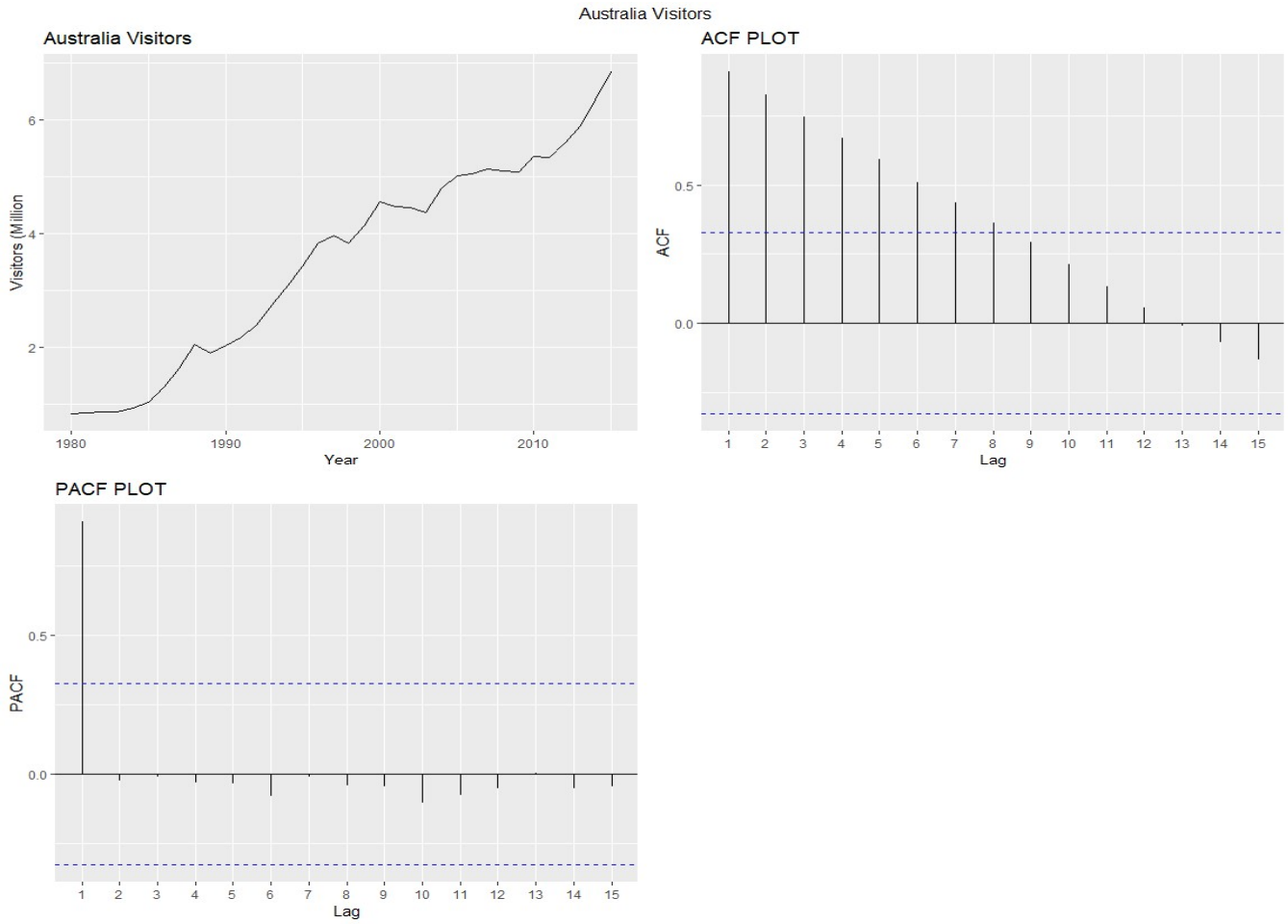
8.8 Consider austa, the total international visitors to Australia (in millions) for the period 1980-2015.

a. Use auto.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.

- ARIMA(0,1,1) was chosen for the auto-arima model. Residuals appear to show stationarity, confirming that the model is satisfactory. The residual plots appear to be random and the ACF & PACF indicate that the model is free of auto-correlation.

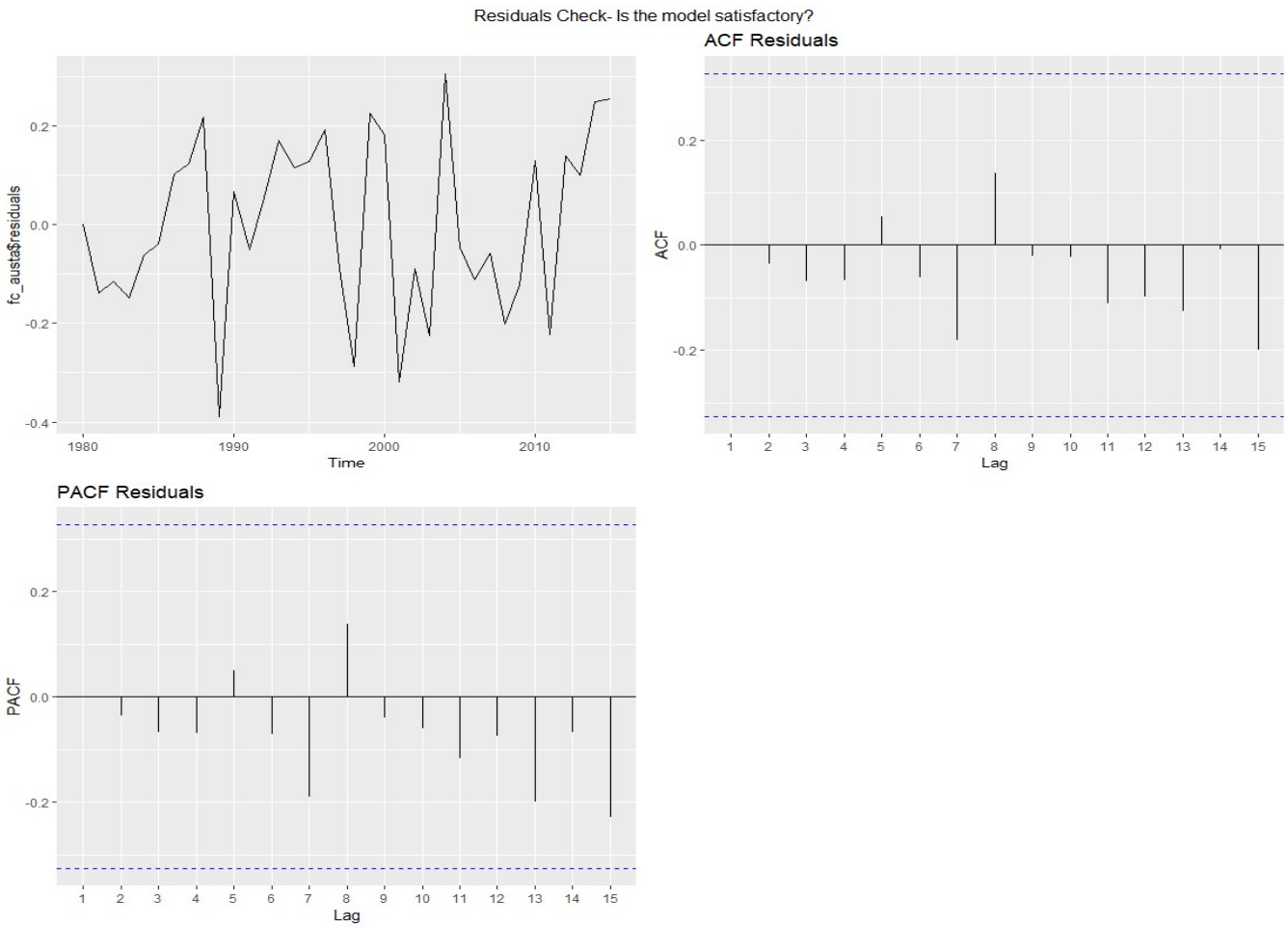
```
# plot

visitor.autoplot<- autoplot(austa) + ggtitle("Australia Visitors")+xlab("Year
") +
  ylab("Visitors (Million")
visitor.acf <- ggAcf(austa) + ggtitle("ACF PLOT")
visitor.Pacf <- ggPacf(austa) + ggtitle("PACF PLOT")
grid.arrange(
  visitor.autoplot,
  visitor.acf,
  visitor.Pacf,
```

```
  nrow = 2,
  top = "Australia Visitors")
```




```
#fit model with auto.arima

(fc_austa <- auto.arima(austa, seasonal=F))

## Series: austa
## ARIMA(0,1,1) with drift
##
## Coefficients:
##          ma1   drift
##       0.3006  0.1735
## s.e.  0.1647  0.0390
##
## sigma^2 estimated as 0.03376:  log likelihood=10.62
## AIC=-15.24   AICc=-14.46   BIC=-10.57

#check residuals to make sure the model is satisfactory

res <- autoplot(fc_austa$residuals)
acf.model <- ggAcf(ts(fc_austa$residuals))+ ggtitle("ACF Residuals")
```

```r
pacf.model <- ggPacf(ts(fc_austa$residuals))+ggtitle("PACF Residuals")

grid.arrange(
  res,
  acf.model,
  pacf.model,
  nrow = 2,
  top = "Residuals Check- Is the model satisfactory?")
```

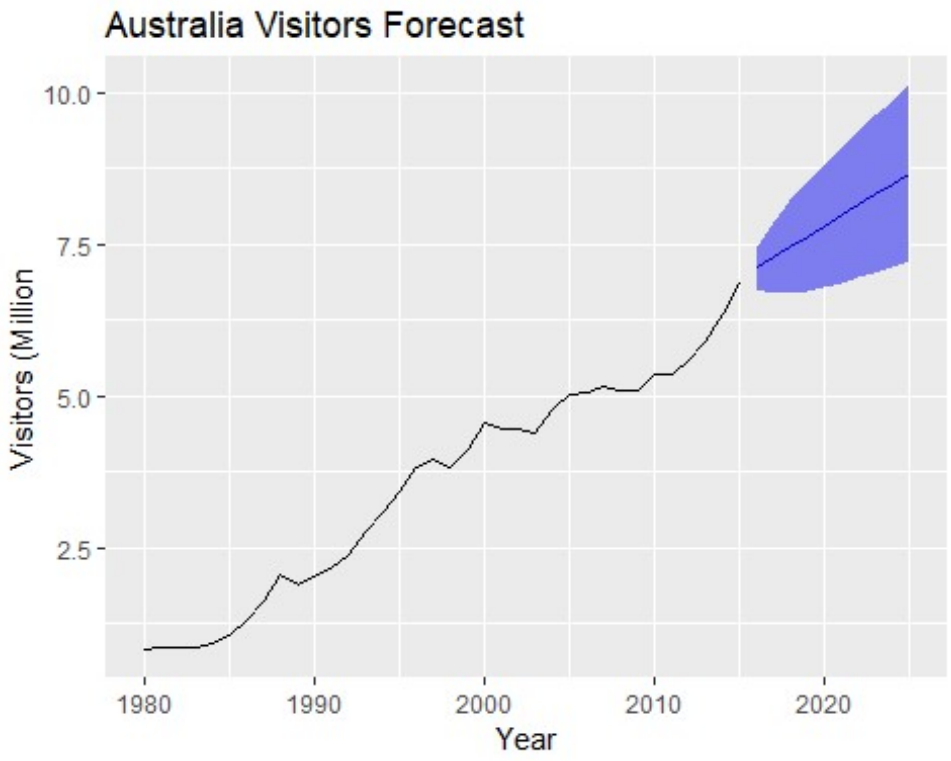



```r
# set the confidence level to 95% and the years out 10 years (h)
visitor.forecast <- forecast(fc_austa, level = c(95), h=10 )
data.frame(visitor.forecast)
```

```
##      Point.Forecast    Lo.95     Hi.95
## 2016       7.108647 6.748536  7.468758
## 2017       7.282129 6.691337  7.872922
## 2018       7.455612 6.701695  8.209529
## 2019       7.629094 6.741543  8.516644
## 2020       7.802576 6.799031  8.806120
## 2021       7.976058 6.868603  9.083513
## 2022       8.149540 6.947121  9.351960
## 2023       8.323023 7.032609  9.613436
```

```
## 2024       8.496505 7.123725  9.869284
## 2025       8.669987 7.219512 10.120462
```

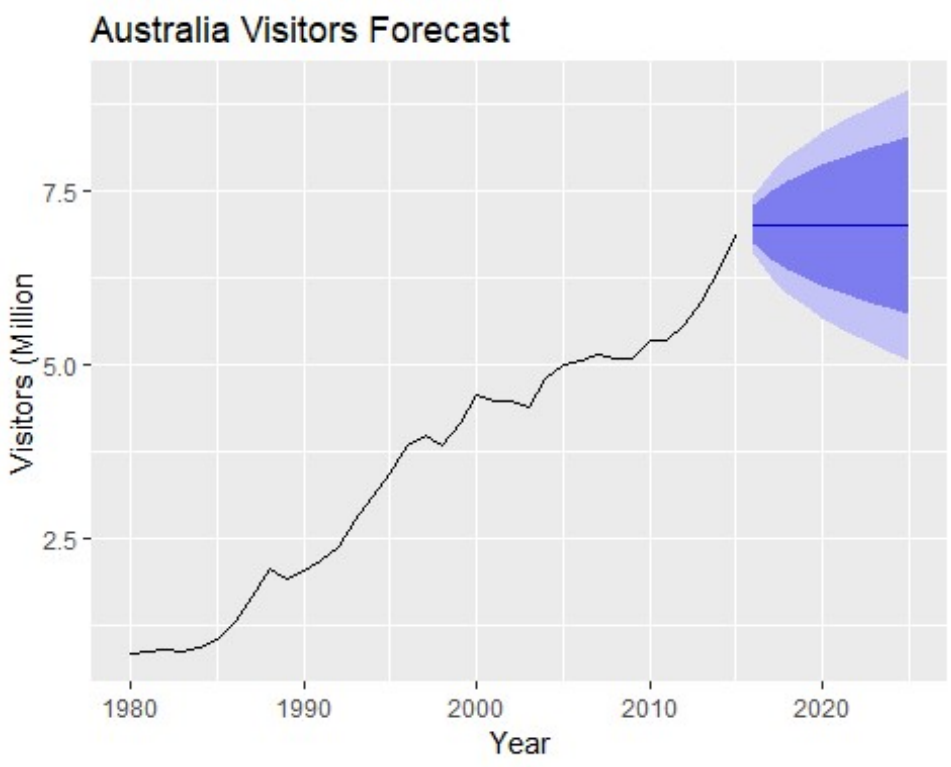
```
autoplot(visitor.forecast) + ggtitle("Australia Visitors Forecast")+xlab("Yea
r") +
  ylab("Visitors (Million")
```



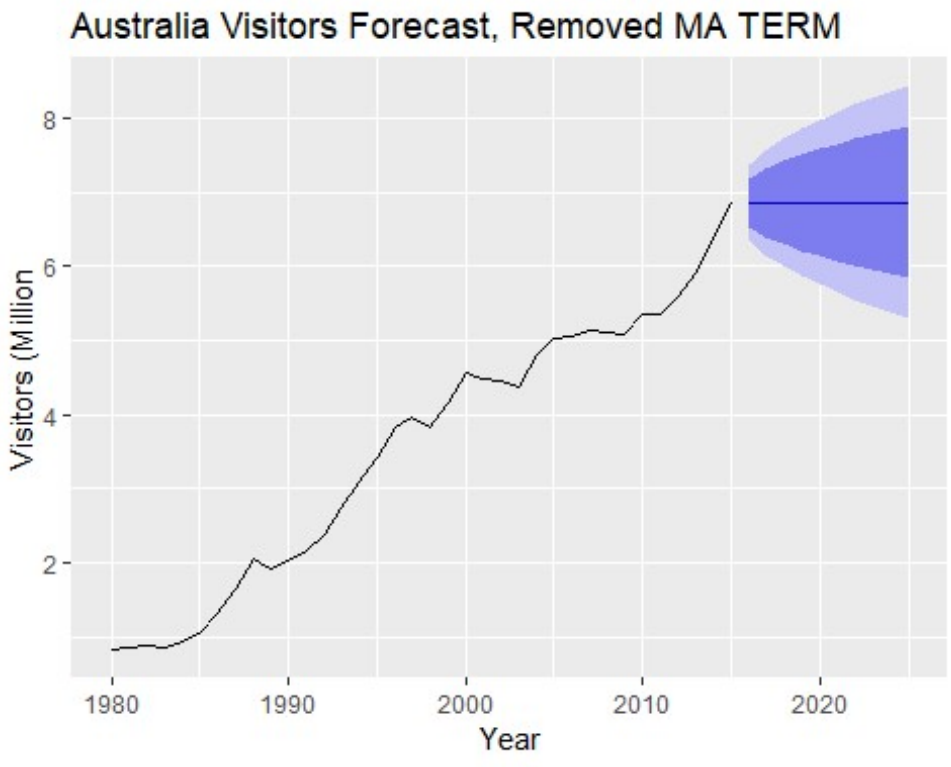
b. Plot forecasts from an ARIMA(0,1,1) model with no drift and compare these to part a. Remove the MA term and plot again.

- The for forecasts yield similar results but removing the MA parameter, slightly reduces the range of the forecast. Neither plot picks up trend.

```
fc_austa<-forecast(arima(austa,order =c(0, 1, 1)), h = 10)
autoplot(fc_austa)+ ggtitle("Australia Visitors Forecast")+xlab("Year") +
  ylab("Visitors (Million")
```



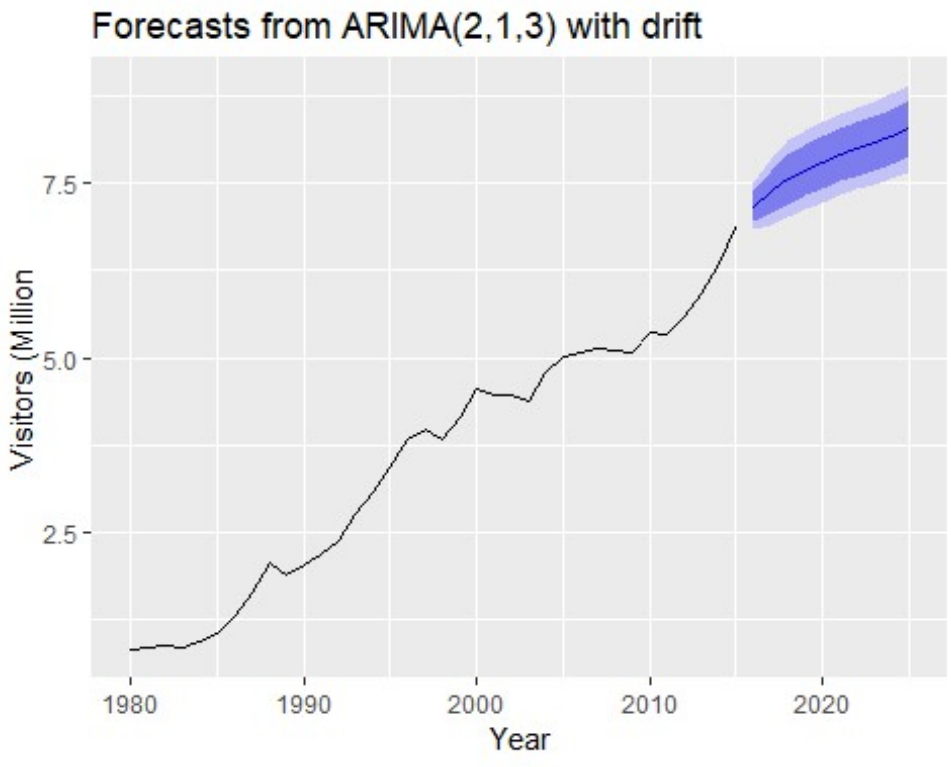

```
# remove moving average term.
fc_austa<-forecast(arima(austa,order=c(0, 1, 0)), h = 10)
autoplot(fc_austa)+ ggtitle("Australia Visitors Forecast, Removed MA TERM")+x
lab("Year") +
  ylab("Visitors (Million")
```



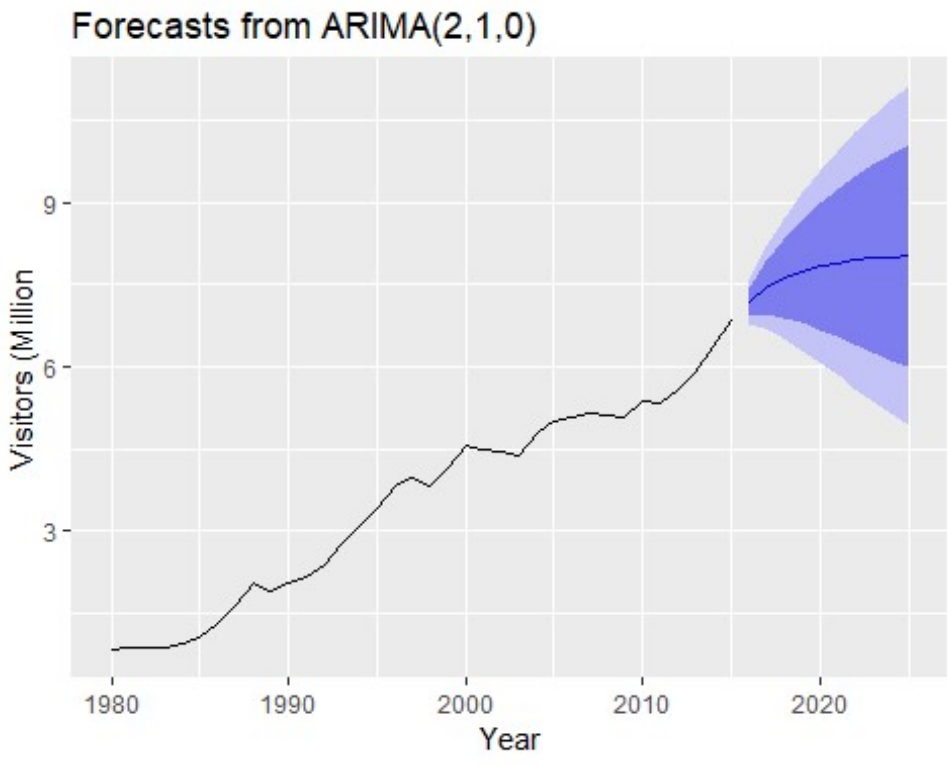
c. Plot forecasts from an ARIMA(2,1,3) model with drift. Remove the constant and see what happens.

- The forecast range significantly expands with the drift removed

```
fc_austa<-forecast(Arima(austa,order =c(2, 1, 3),include.drift=TRUE),h = 10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```



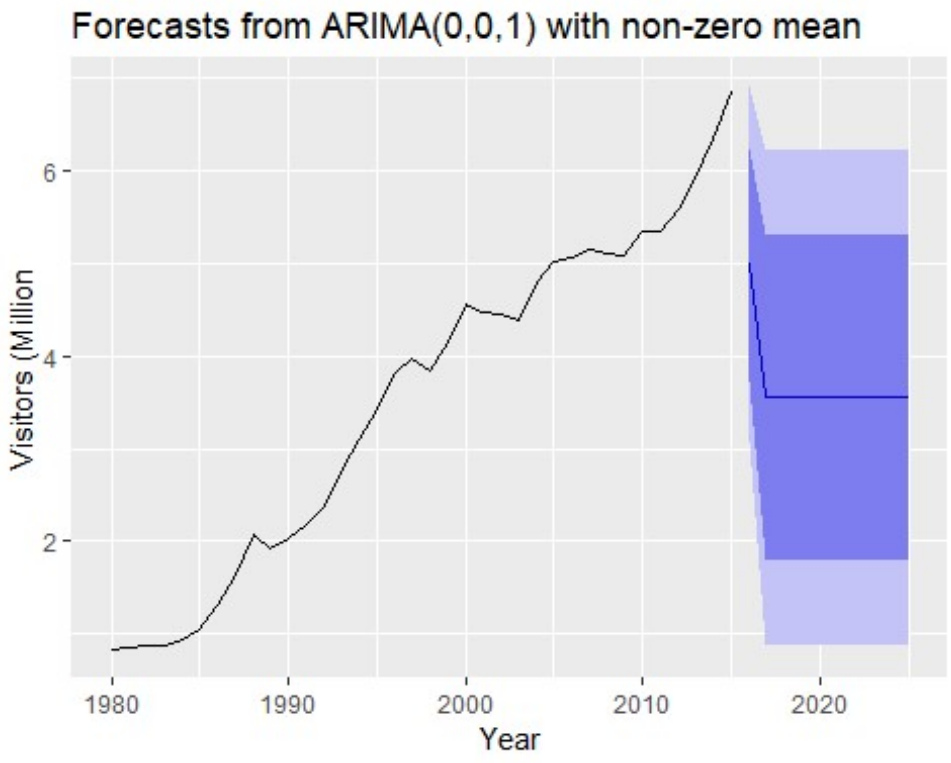

```r
# remove moving average term.
fc_austa$model$coef[6] <- 0
fc_austa<-forecast(Arima(austa,order =c(2, 1, 0)),h = 10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```



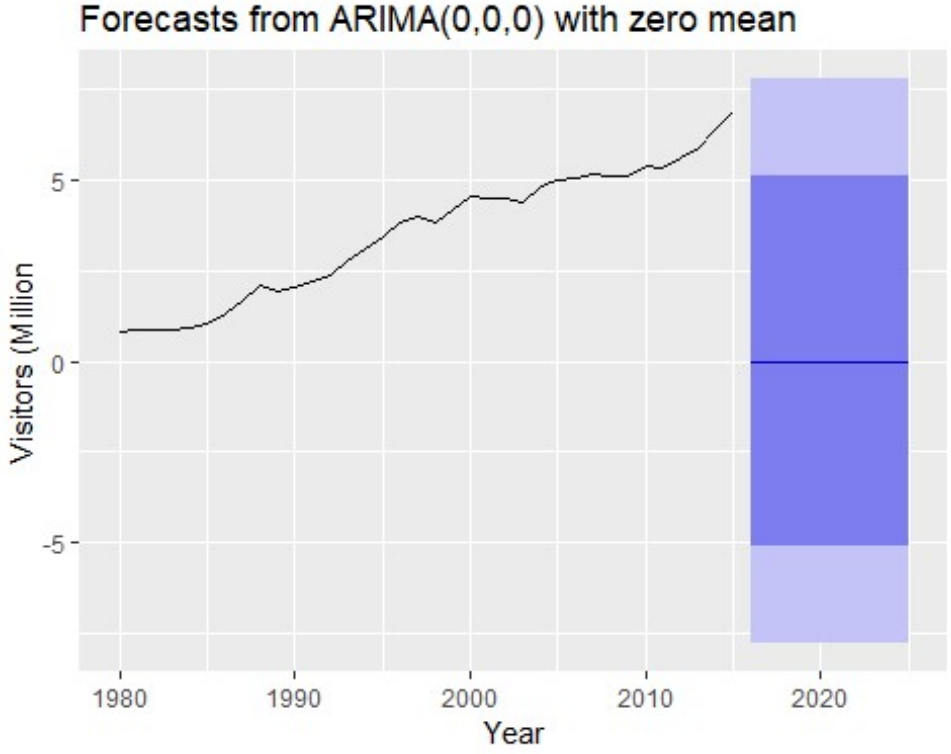
d. Plot forecasts from an ARIMA(0,0,1) model with a constant. Remove the MA term and plot again.

- The plot without constant shifts are below trend but with the MA term, the forecast shifts upward. The MA still doesn't pick up the trend line though.

```
fc_austa<-forecast(Arima(austa,order=c(0,0,1),include.constant=TRUE),h=10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```




```
fc_austa<-forecast(Arima(austa,order=c(0,0,0),include.constant=FALSE),h=10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```



e. Plot forecasts from an ARIMA(0,2,1) model with no constant.

- This plot picks up the trendline and is well fitted.

```
fc_austa<-forecast(Arima(austa,order=c(0,2,1),include.constant=FALSE),h=10)
autoplot(fc_austa)+xlab("Year") +
  ylab("Visitors (Million")
```