Assignment 2

Sri Seshadri

5/4/2018

Contents

Section 7/8 Question 7.1

a) Figure 1 shows the timeseries of the books data. The paperback series have change in levels more so than a trend. The hardcover series have a subtle upward trend. The saw tooth pattern in data shows seasonality, but with varying levels of peaks.

```
data("books")
autoplot(books,facet = T) + theme_bw() + geom_smooth(se = F)
```

```
## `geom_smooth()` using method = 'loess'
```

b, c,d and e.

Figure 2 shows the effect of alpha on the SSE (in sample). In the Paperback and Hardcover data when the intial values are set to "simple" we see that as the alpha increases the SSE decreases, that is the forecasts are closer to the actuals. However when the intial values are set to optimal, it is seen that in the Paperback data, the SSE decreases and then increases with increase in alpha.

The optimal alpha value is shown in Figure 2 as a triangle. When the initial values are obtained optimally and the alpha is selected optimally, then minimal SSE is obtained at relatively smaller values of alpha.

```
sesstats <- function(alpha,initial,name){
  data <- books
  fit <- ses(data[,name],alpha = alpha, initial = initial)
```

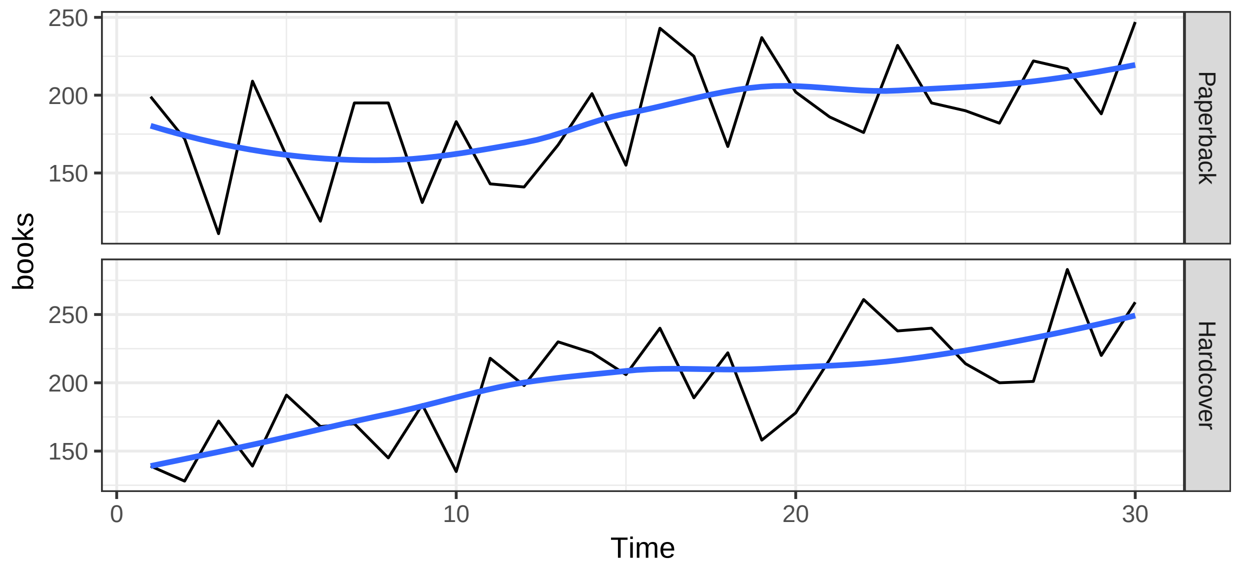


Figure 1: Time series of books

```
  stats <- data.frame(dataset = name,
                      initial = initial,
                      alpha = alpha,
                      SSE = (accuracy(fit)[2])^2*length(data[,name])
                      )
}


arggrid <- tibble(alpha = rep(seq(0.1,0.2,by= 0.02),4),
                      initial = rep(rep(c("simple","optimal"),each = 6),2),
                      name = rep(c("Paperback", "Hardcover"),each = 12)
                      )
SSE_Results <- purrr::pmap_df(arggrid,.f = sesstats)

SSE_Results %>%
  dplyr::mutate(dataset = as.factor(dataset)) %>%
  ggplot(mapping = aes(x = alpha, y = SSE,color = initial)) + geom_point() + geom_line() + facet_wrap(~
```

```
ses.Paperback.simple <- ses(books[,"Paperback"],initial = "simple", h = 4)
ses.Paperback.optimal <- ses(books[,"Paperback"],initial = "optimal", h = 4)
ses.Hardcover.simple <- ses(books[,"Hardcover"],initial = "simple", h = 4)
ses.Hardcover.optimal <- ses(books[,"Hardcover"],initial = "optimal", h = 4)

EstAlphaModels <- list(ses.Paperback.smpl = ses.Paperback.simple,
                       ses.Paperbacl.Opt = ses.Paperback.optimal,
                       ses.Hardcover.smpl = ses.Hardcover.simple,
                       ses.Hardcover.Opt = ses.Hardcover.optimal)



extractAlpha <- function(mdl){
  mdl %>% .$model %>% .$par %>% head(1)
}
```

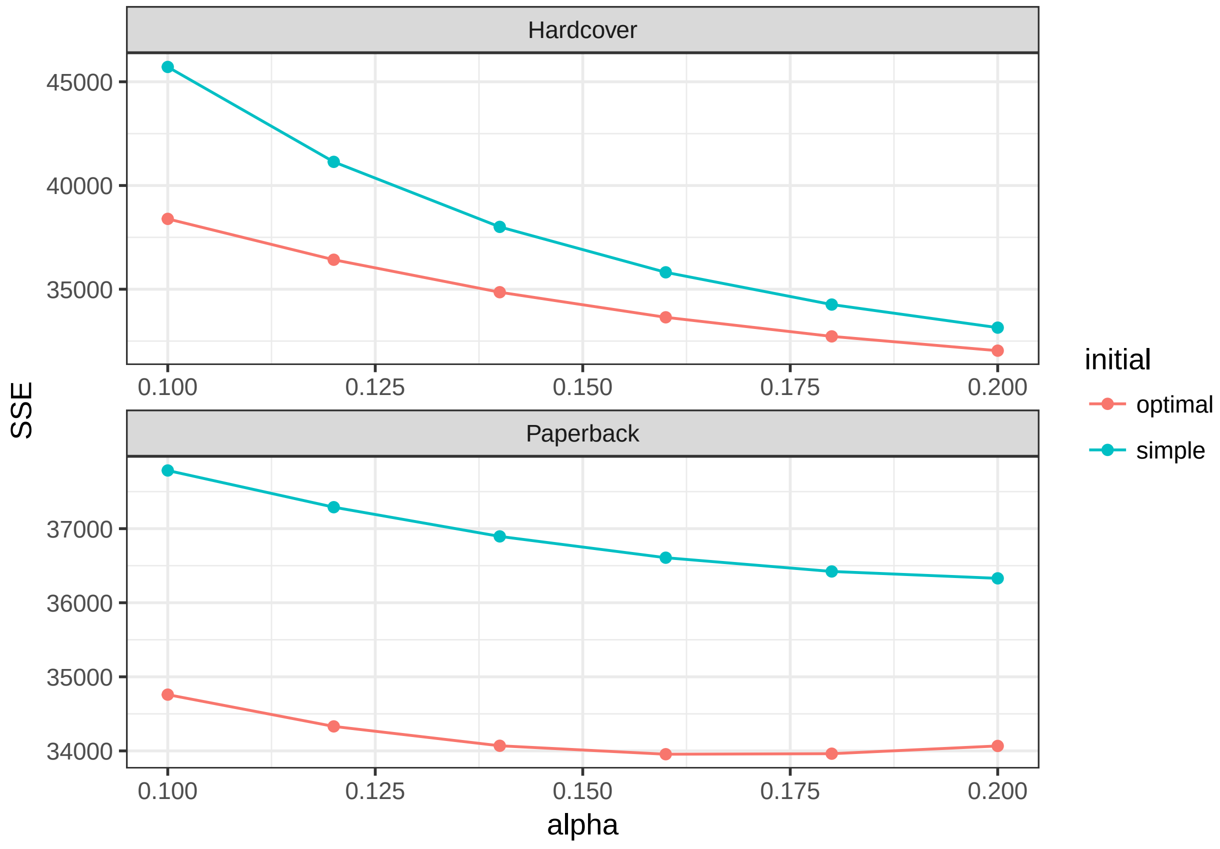


Figure 2: SSE vs alpha by simple and optimal initial values

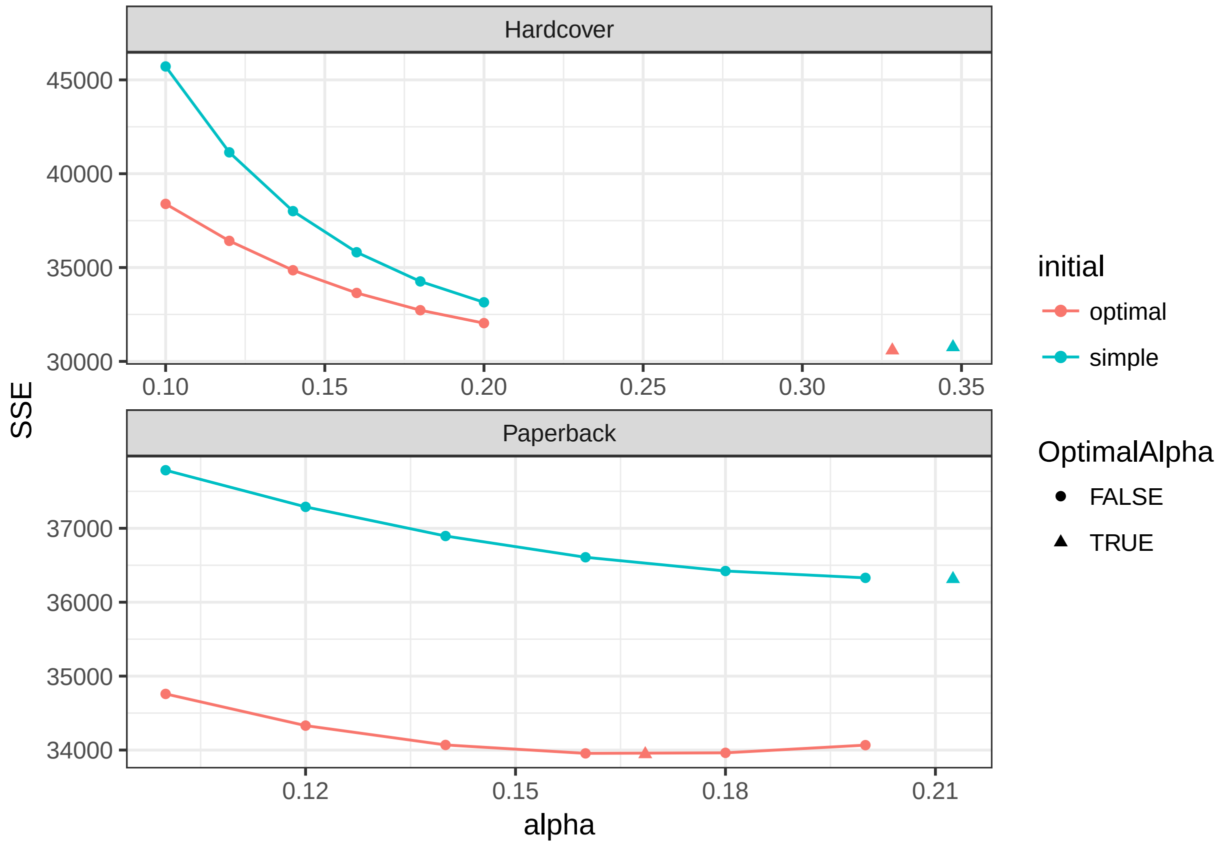


Figure 3: SSE vs alpha by simple and optimal initial values

```
dataset <- rep(c("Paperback","Hardcover"),each = 2)
initial <- rep(c("simple","optimal"),2)
alpha <- t(purrr::map_df(EstAlphaModels,extractAlpha))
RMSE <- t(purrr::map_df(EstAlphaModels,.f =  accuracy)[2,])
SSE <- RMSE^2*length(books[,1])

OptimalAlpha <- data.frame(dataset, initial,alpha = alpha,SSE,row.names = NULL)
SSE_Results <- rbind.data.frame(SSE_Results,OptimalAlpha) %>% dplyr::mutate(OptimalAlpha = c(rep(0,nrow


SSE_Results %>%
  dplyr::mutate(dataset = as.factor(dataset),
                OptimalAlpha = as.logical(OptimalAlpha)) %>%
  ggplot(mapping = aes(x = alpha, y = SSE,color = initial, shape = OptimalAlpha)) + geom_point() + geom_
```

```
par(mfrow = c(2,2))
plot(ses.Paperback.simple, main = "SES - Paperback, optimal alpha, initial : simple",cex.main = 1)
plot(ses.Paperback.optimal, main = "SES - Paperback, optimal alpha, initial : optimal",cex.main = 1)
plot(ses.Hardcover.simple, main = "SES - Hardcover, optimal alpha, initial : simple",cex.main = 1)
plot(ses.Hardcover.optimal,main = "SES - Hardcover, optimal alpha, initial : optimal",cex.main = 1)
```

Figure 3 shows that the forecasts are identical. The exponential smoothing methods rely on the actuals for forecasting the next period. Hence the forecasts are identical.

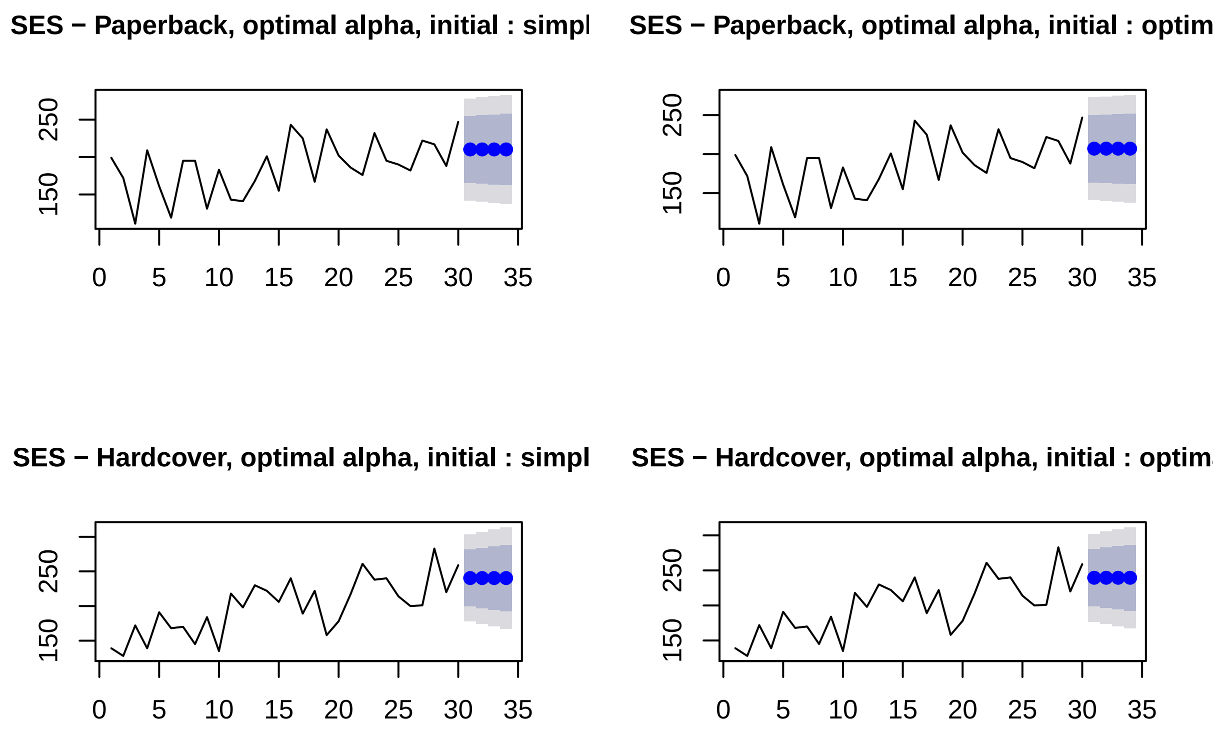


Figure 4: Forcasts with optimal alpha

It can be seen that the optimal settings of alpha and initial values have reduced the SSE.

Question 7.2

a) Figure 4 shows the forecasts for the next 4 days for the sales of Hardcover and paperback computed using Holt's method.

```
holt.paperback <- holt(books[,"Paperback"], h = 4)
holt.hardcover <- holt(books[,"Hardcover"], h = 4)
par(mfrow = c(1,2))
plot(holt.hardcover,main = "Holts method - Hardcover", cex.main = 1)
plot(holt.paperback, main = "Holts method - Paperback", cex.main = 1)
```

```
SSE_holt <- data.frame(dataset = c("Paperback", "Hardcover"),
                      SSE = c(sw_glance(holt.paperback %>% .$model)$RMSE^2*length(books[,"Hardcover"]),
                              sw_glance(holt.hardcover %>% .$model)$RMSE^2*length(books[,"Paperback"]))
                      )

SSE_holt_ses <-
  rbind.data.frame(
  SSE_Results %>% dplyr::filter(OptimalAlpha == 1) %>%
  dplyr::filter(initial == "optimal") %>%
  dplyr::select(dataset, SSE),
  SSE_holt
  ) %>%
  dplyr::mutate(Model = rep(c("Exponential Smoothing", "Holt"), each = 2))
```

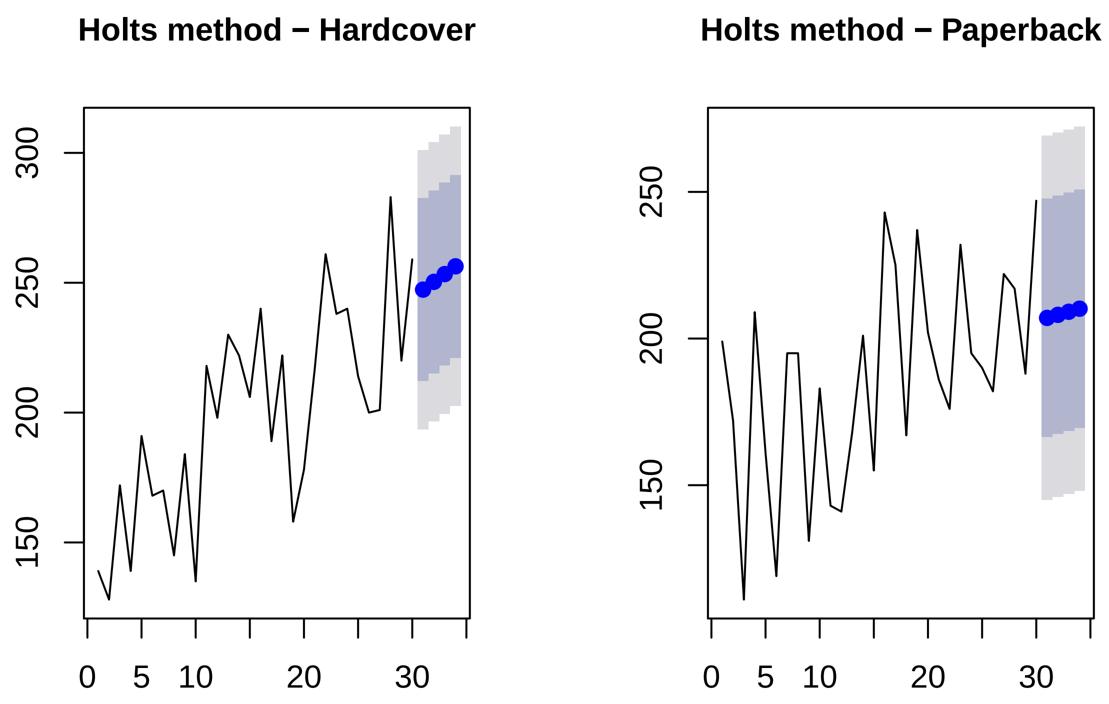


Figure 5: Holts method forecast for Hardcover & Paperback

```
Forecasts <- cbind(SSE_holt_ses,t(sapply(list(ses.Paperback.optimal,ses.Hardcover.optimal,holt.paperbac

knitr::kable(Forecasts, caption = "Holts vs simple exponential smoothing forecasts")
```

Table 1: Holts vs simple exponential smoothing forecasts

dataset	SSE	Model	Point.Forecast	Lo.80	Hi.80	Lo.95	Hi.95
Hardcover	30587.69	Exponential Smoothing	239.5602	198.639	280.4815	176.9766	302.1439
Hardcover	22581.83	Holt	247.3504	212.1899	282.5109	193.577	301.1237
Paperback	33944.82	Exponential Smoothing	207.1098	164.0013	250.2182	141.1811	273.0384
Paperback	30074.17	Holt	207.038	166.4617	247.6143	144.982	269.0941

b) Table 1 shows the forecasts of both the simple and double exponential (Holt's) forecasting methods. The SSE of the Holt's method is less compared to simple exponential smoothing. The holts method is able to predict the slope or trend in the data.

c) The forecasts of Holt's method is better for the reason mentioned above.

d)

Question 7.3

The holts method for several alpha and beta experiemnts with exponential trend turned on and off and damping turned on and off. Figure 4 shows the contour plot of RMSE vs alpha nad beta when exponential trend is turned on and off with damping and no damping.

The RMSE is insensitive when the damping is set to false and alpha is greater than 0.3. Which is an interesting observation. The best and worst performing model are plotted below (Figure 5).

```
data("eggs")
inputs <- expand.grid(alpha = seq(0.001, 0.5, length.out = 10),beta = seq(0.0001, 0.05, length.out = 10
```

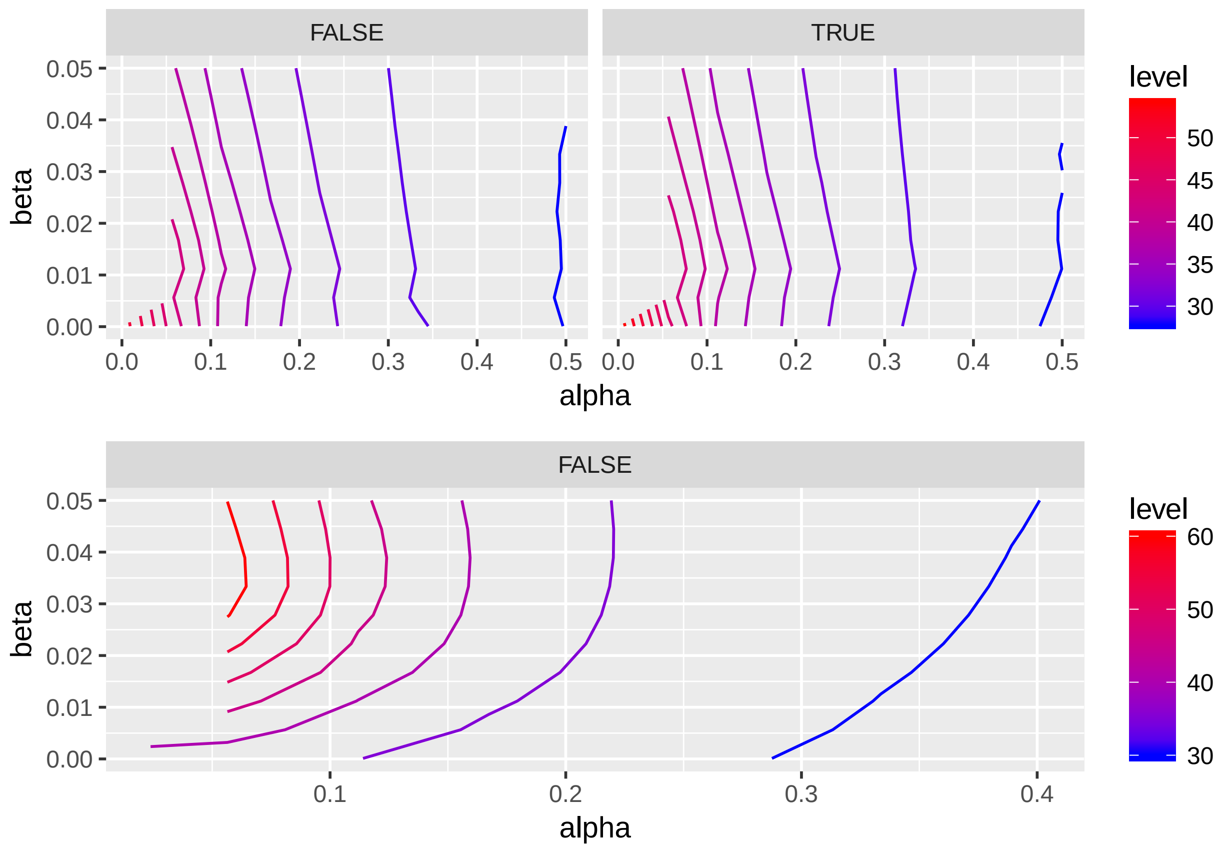


Figure 6: Top: Left:Contour plot of RMSE vs alpha & beta with NO damping when exponential trend is off, right : When exponetial trend is ON; Bottom: Contour plot of RMSE vs alpha & beta with damping when exponential trend is off

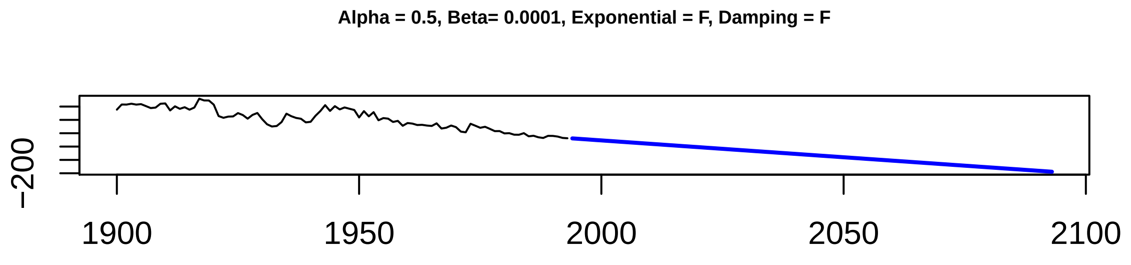
```
Results <- purrr::pmap(inputs,holtfct)
Accuracy <- lapply(1:273, function(x) Results[[x]]$result)
Accuracy <- do.call("rbind",Accuracy)

dampF <- Accuracy %>% dplyr::filter(damped == F) %>%  ggplot(data = .,aes(x = alpha, y = beta, z = RMSE

dampT <- Accuracy %>% dplyr::filter(damped == T) %>%  ggplot(data = .,aes(x = alpha, y = beta, z = RMSE
  facet_wrap(~exponential)

gridExtra::grid.arrange(dampT,dampF)
```

```
par(mfrow = c(2,1))
plot(Results[[which.min(Accuracy$RMSE)]]$model,PI = F, main = "Alpha = 0.5, Beta= 0.0001, Exponential =
plot(Results[[which.max(Accuracy$RMSE)]]$model, PI = F, col = "red", main = "Alpha = 0.06, Beta= 0.03, 
```



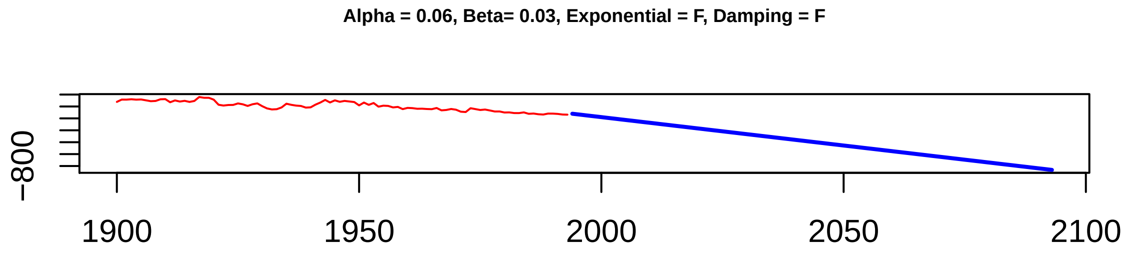


Figure 7: Top: Best model; Bottom: Worst model

Question 7.4

a) The ukcars data is non-staionary with seasonlity, trend and changes in level. Figure 6 shows the time series plot of the same

```
data("ukcars")
plot(ukcars)
```

b) The ukcars data is decomposed using the stl() method. Figure 7 shows the decomposition.

```
stlfit <- stl(ukcars,s.window = "periodic", robust = T)
autoplot(stlfit)
```

```
seasonalAdj <- stlfit$time.series[,"trend"]
```

c) Figure 8 shows the additive damped model forecast on the seasonally adjusted data as obtained from stl(). The summary of the model is shown below.

```
holtfit <- holt(seasonalAdj,h = 8,damped = T,exponential = F)
summary(holtfit)
```

```
##
## Forecast method: Damped Holt's method
##
```

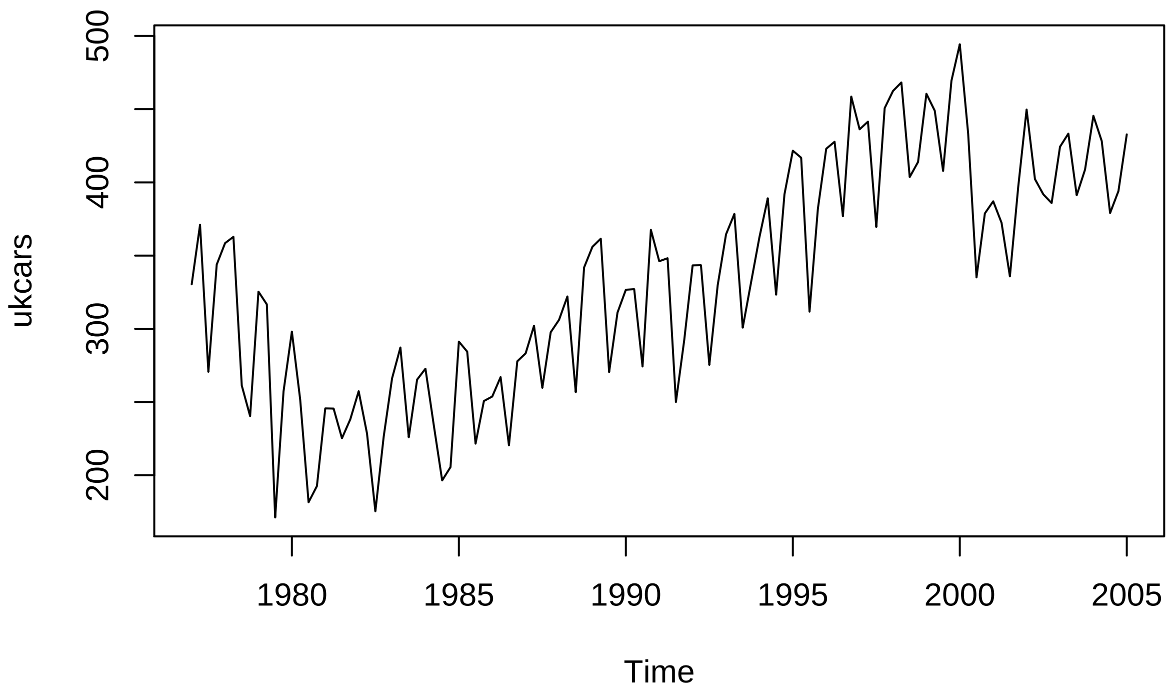


Figure 8: Time series of ukcars

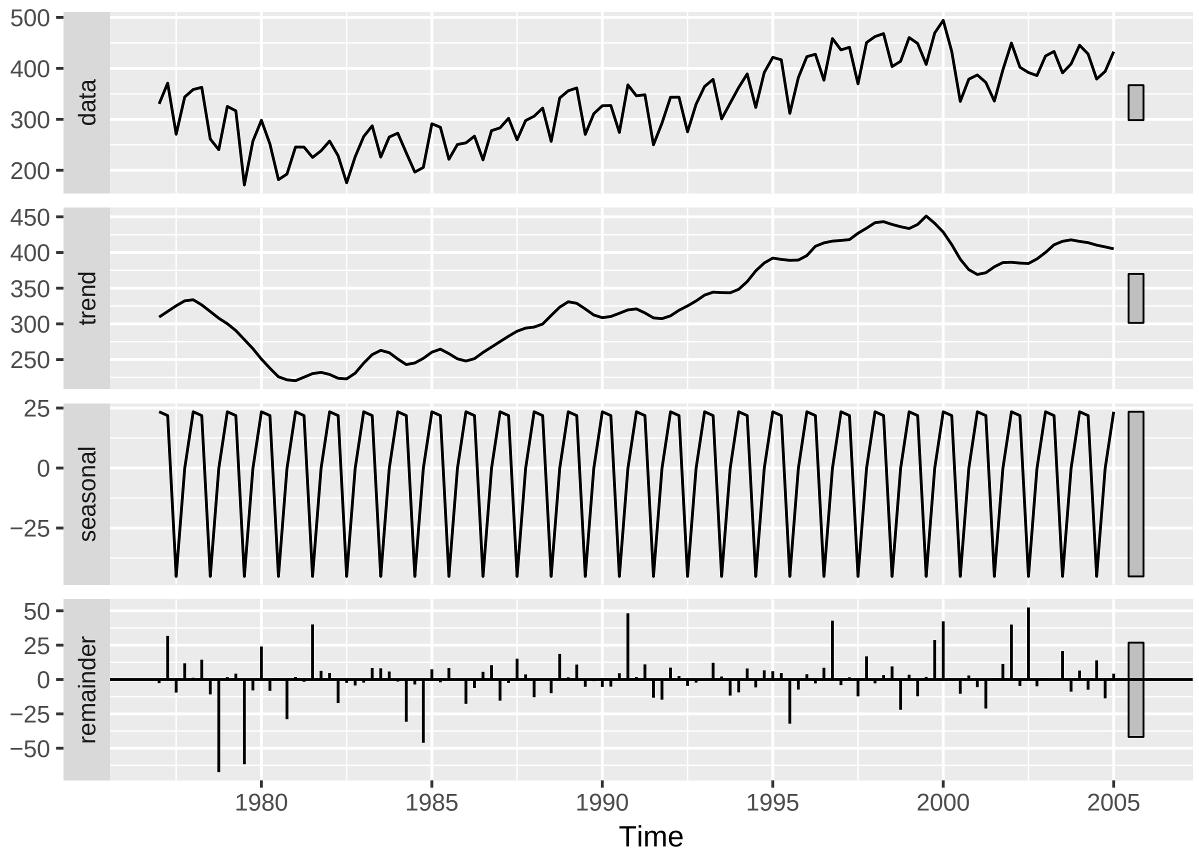


Figure 9: STL decomposition of ukcars

```
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = seasonalAdj, h = 8, damped = T, exponential = F)
##
##   Smoothing parameters:
##     alpha = 0.9999
##     beta  = 0.9999
##     phi   = 0.8
##
##   Initial states:
##     l = 329.7418
##     b = -2.964
##
##   sigma:  5.511
##
##      AIC     AICc      BIC
## 931.9176 932.7100 948.2819
##
## Error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.1347103 5.510954 4.057228 0.06727714 1.273959 0.2068624
##                   ACF1
## Training set 0.1835495
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005 Q2       402.8698 395.8072 409.9323 392.0685 413.6710
## 2005 Q3       401.0937 386.5521 415.6353 378.8543 423.3332
## 2005 Q4       399.6729 377.1260 422.2198 365.1904 434.1555
## 2006 Q1       398.5363 367.8288 429.2437 351.5733 445.4992
## 2006 Q2       397.6269 358.8134 436.4405 338.2667 456.9872
## 2006 Q3       396.8995 350.1527 443.6463 325.4064 468.3925
## 2006 Q4       396.3175 341.8756 450.7594 313.0558 479.5792
## 2007 Q1       395.8519 333.9858 457.7181 301.2358 490.4681
```

```
plot(holtfit, ylab = "Seasonally adjusted data")
```

```
# reseasonalize
seasoncomp <- stlfit$time.series[,"seasonal"]
seasonalcompfct <- snaive(seasoncomp, h= 8)
fcts <- holtfit$mean + seasonalcompfct$mean
reseasonlizedts <- ts(c(ukcars,fcts),start = c(1977,1), frequency = 4)
year <- c(rep(1977:2004, each = 4), rep(2005:2006, each = 4), 2007)
qtr <- c(rep(1:4,(2006-1977+1)),1)
data.frame(data = reseasonlizedts, year = year, qtr = qtr, year.qtr = year + qtr/10, forecast = c(rep(F
  ggplot(aes(x = year.qtr, y = data, color = forecast)) + geom_line()
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

```
#autoplot(reseasonlizedts, col= c(rep("grey",113),rep("blue",8)))
RMSE_Holtdamp <- sqrt(sum((ukcars - holtfit$fitted + seasonalcompfct$fitted)^2,na.rm = T)/113)
accuracy(holtfit)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
```

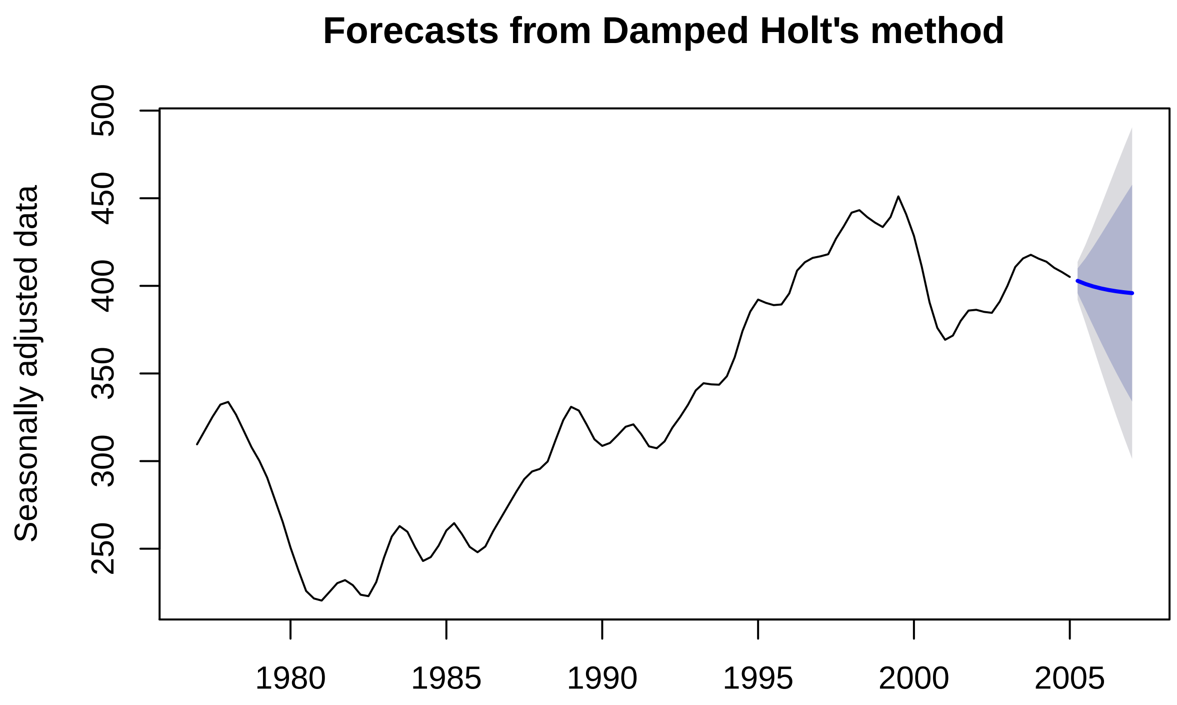


Figure 10: Forecast of seasonally adjusted data using Holts additive method with damping.

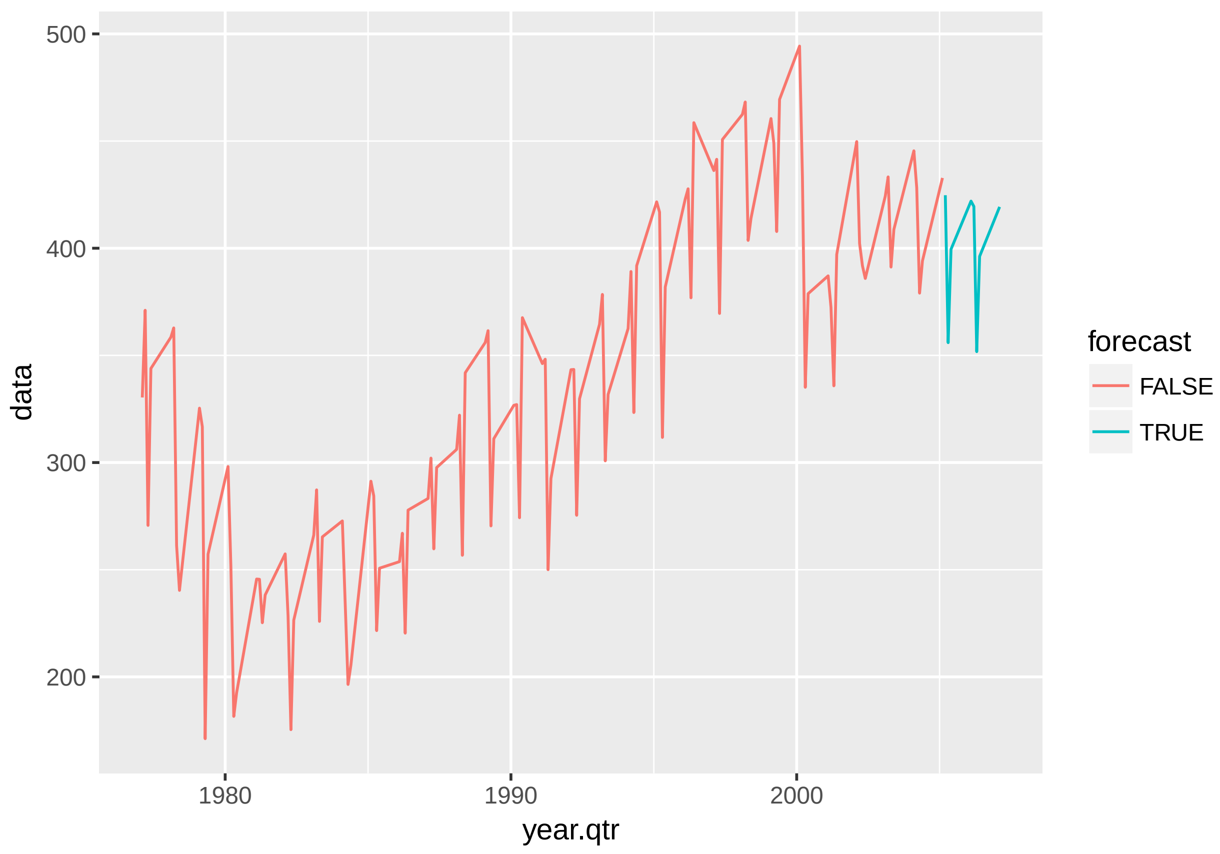


Figure 11: reseasonilized forecasts

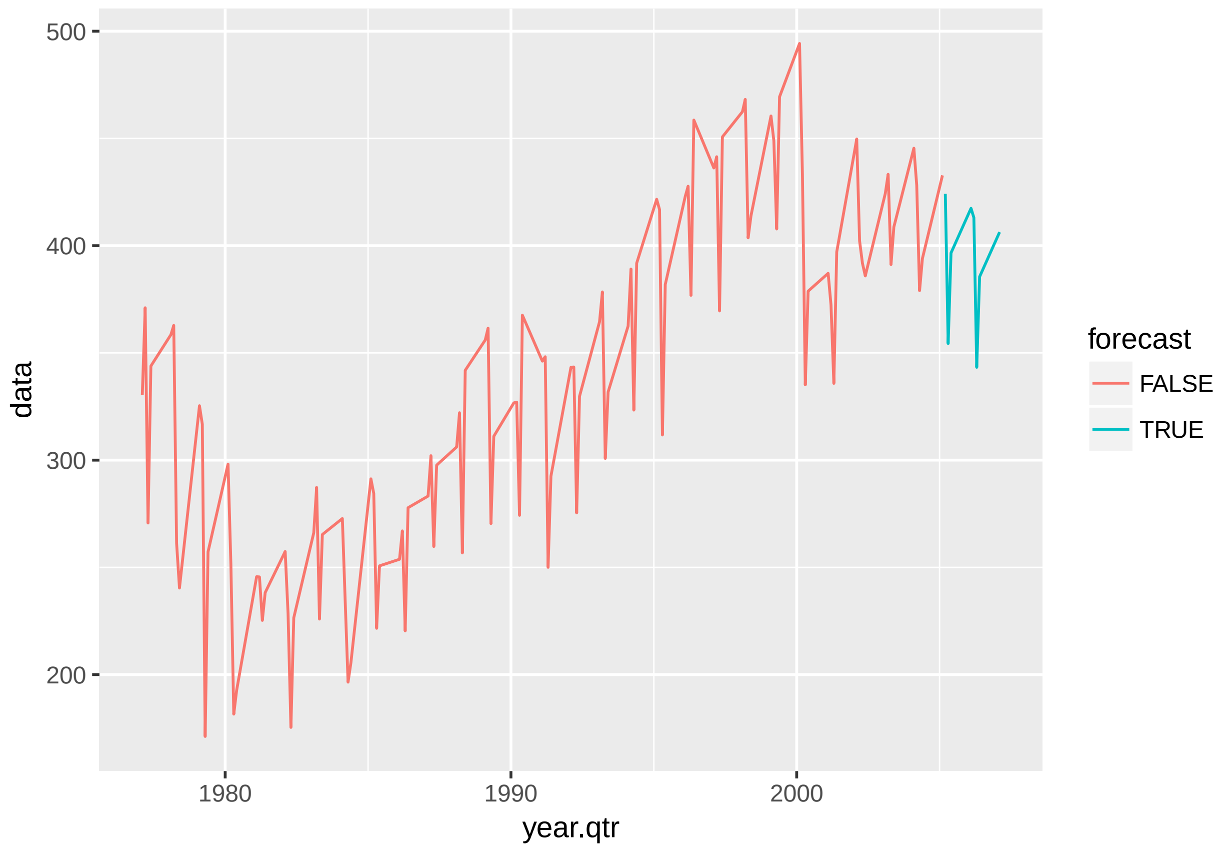


Figure 12: Reseasonlized forecasts - Holt's linear

```
## Training set 0.1347103 5.510954 4.057228 0.06727714 1.273959 0.2068624
##                   ACF1
## Training set 0.1835495
```

Figure 9 shows the reseasonlized forecasts and the RMSE for the model is 57

d)

Figure 10 shows the reseasonalized forecasts using linear Holt's method and the RMSE is 56.88. Subtly better than the forecast with damping.

```
holtlinear <- holt(seasonalAdj,h = 8,damped = F,exponential = F)
fcts <- holtlinear$mean + seasonalcompfct$mean
reseasonlizedts <- ts(c(ukcars,fcts),start = c(1977,1), frequency = 4)
year <- c(rep(1977:2004, each = 4), rep(2005:2006, each = 4), 2007)
qtr <- c(rep(1:4,(2006-1977+1)),1)
data.frame(data = reseasonlizedts, year = year, qtr = qtr, year.qtr = year + qtr/10, forecast = c(rep(F
  ggplot(aes(x = year.qtr, y = data, color = forecast)) + geom_line()
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```

```
RMSE_holtlinear <- sqrt(sum((ukcars - holtlinear$fitted + seasonalcompfct$fitted)^2,na.rm = T)/113)
accuracy(holtlinear)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
```

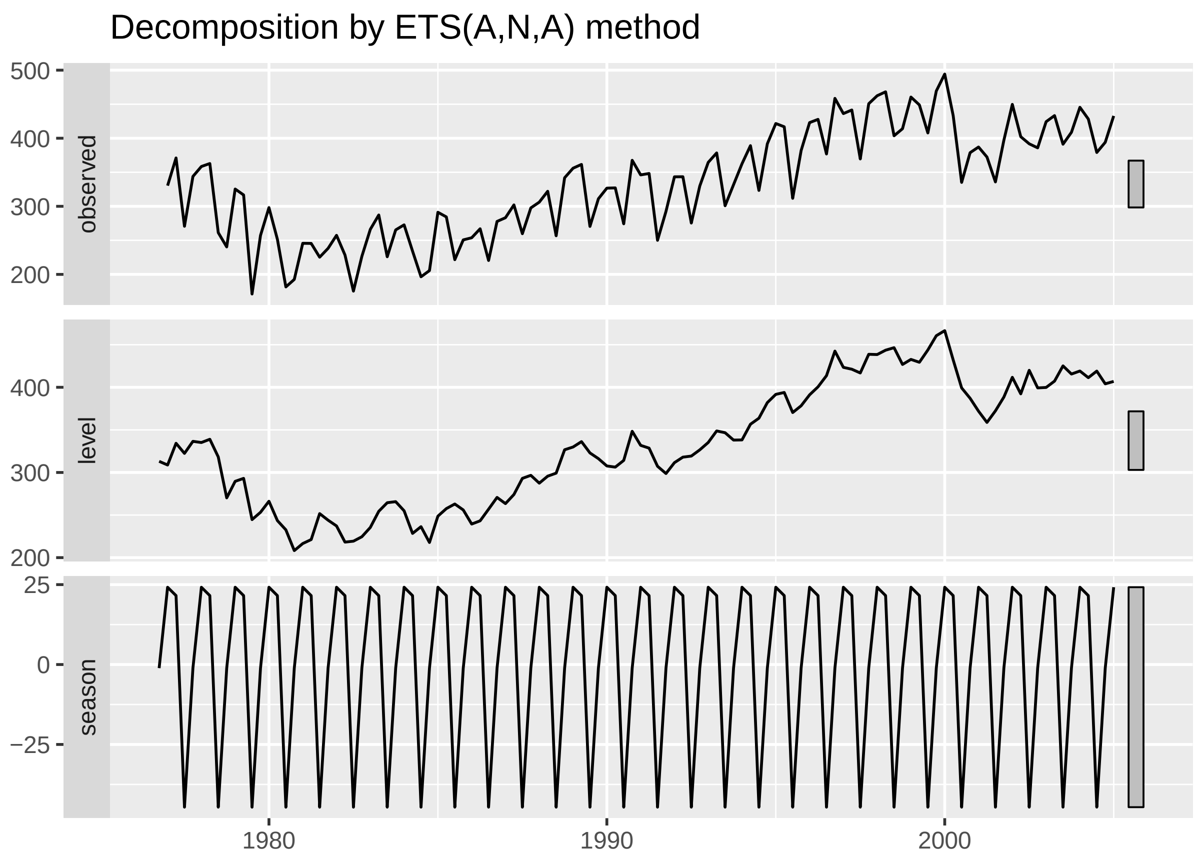


Figure 13: ETS model fit for ukcars

```
## Training set -0.1504732 5.056389 3.901235 -0.0008004432 1.218107 0.1989089
##                   ACF1
## Training set 0.3659724
```

d, e and f) Figure 11 shows that decomposition using ets(). The resulting (A,N,A) model shows an accuracy of 25.25, which is alomost half of the previous 2 models. The ets model is most reasonable in this case.

```r
ets.fit <- ets(ukcars, model = "ZZZ")
autoplot(ets.fit)
```

```r
plot(forecast(ets.fit, h = 8))
```

Comparing the models A,N,A models seems to be a better model from a RMSE perspective.

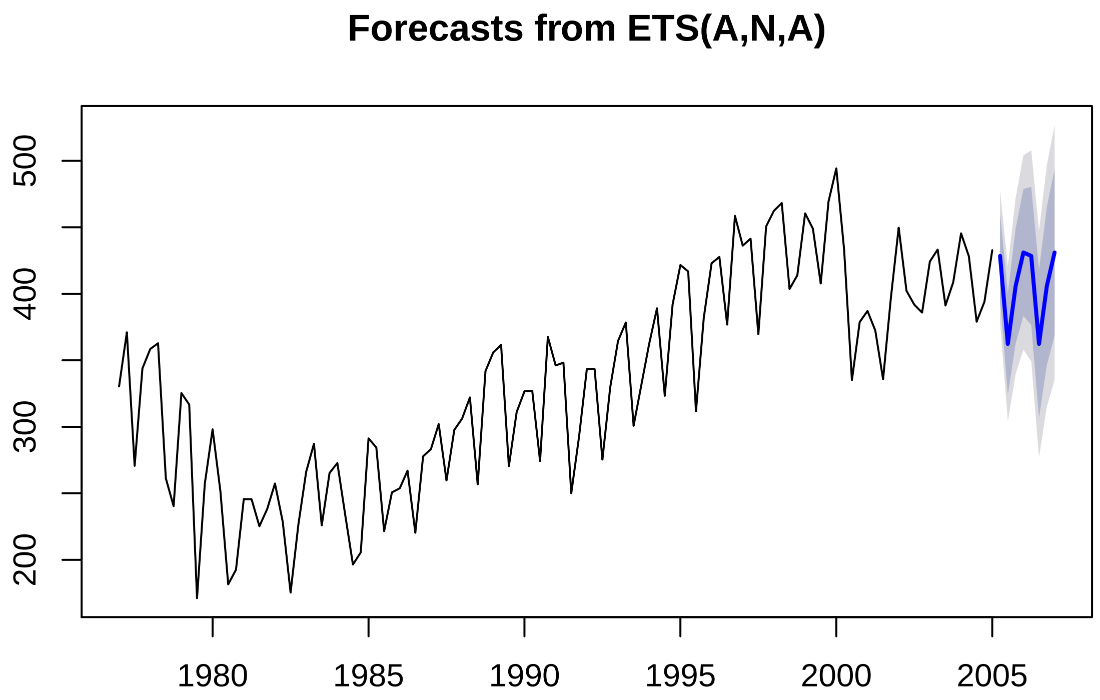


Figure 14: ETS model fit for ukcars

7.5

a) The data is a non stationert time series data with seasonality and trend. The variation on seasonal values increases with the level of the series.

```
data("visitors")
autoplot(visitors)
```

b & c) The holts winter forecast for the next two years is shown in the plot below.

The holt's winter method is necessary here because the variation on seasonal values increases with the level of the series.

```
hw.m.fit <- hw(visitors,seasonal = "multiplicative",h = 24)
plot(hw.m.fit)
```

d & e)

Figure 14 shows the experimentation with the settings of expoential & damped. Based on table 2, the Damped HW multiplicative method gives the best RMSE.

```
par(mfrow=c(2,2))
plot(hw.m.fit,cex.main = 0.7)
plot(hw(y = visitors,seasonal = "multiplicative", exponential = T),cex.main = 0.7)
```

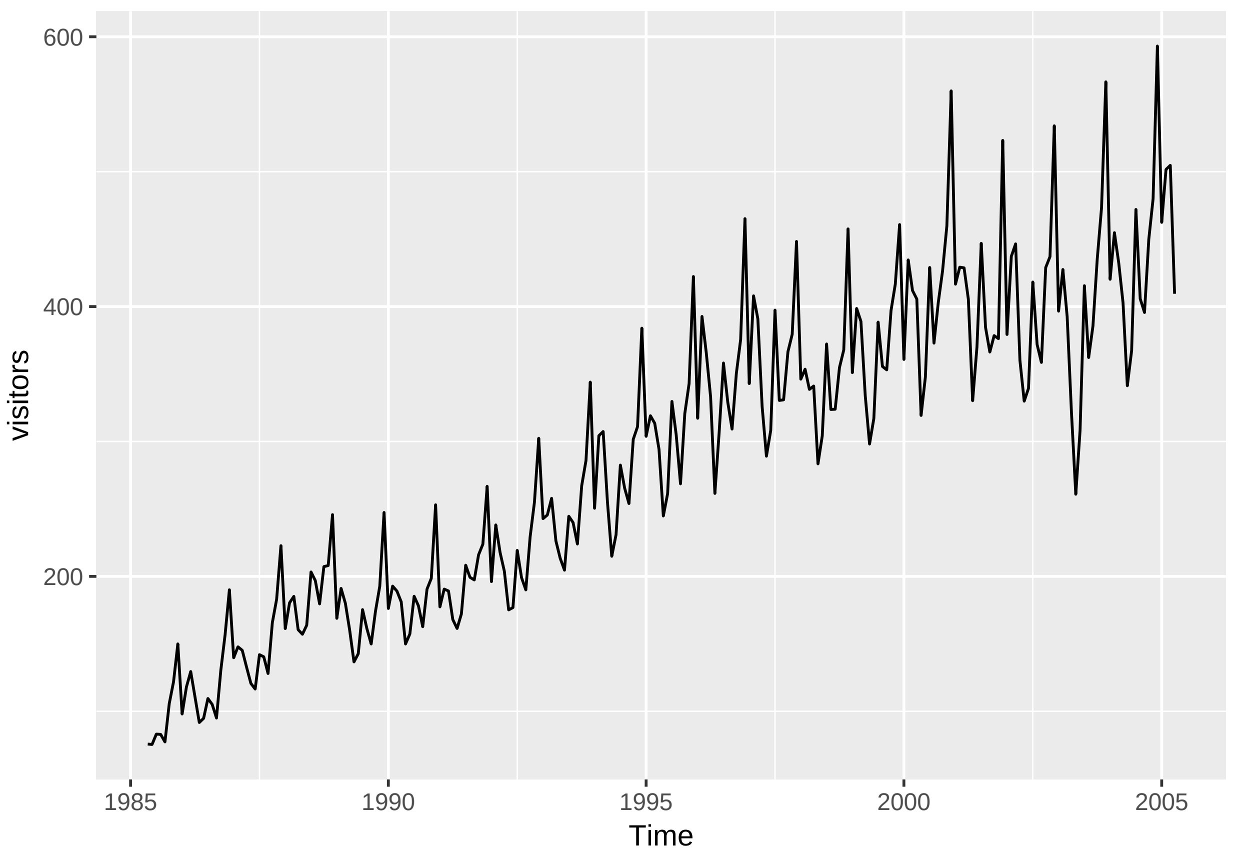


Figure 15: Time series of visitors

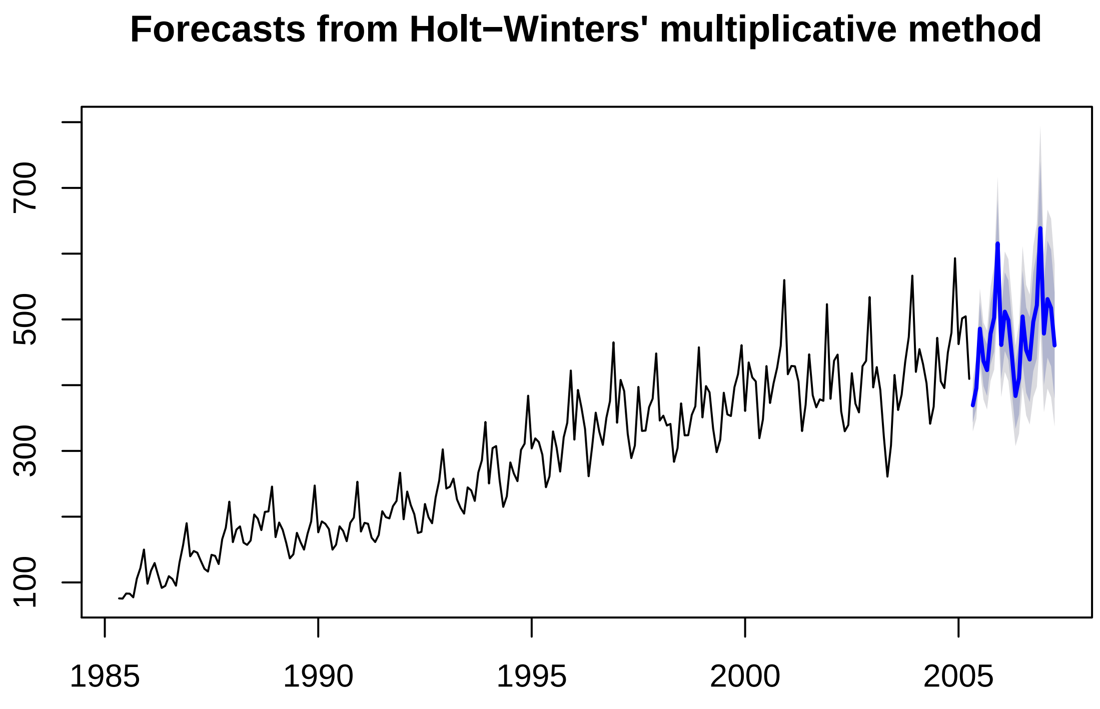


Figure 16: Holts - winters forecast

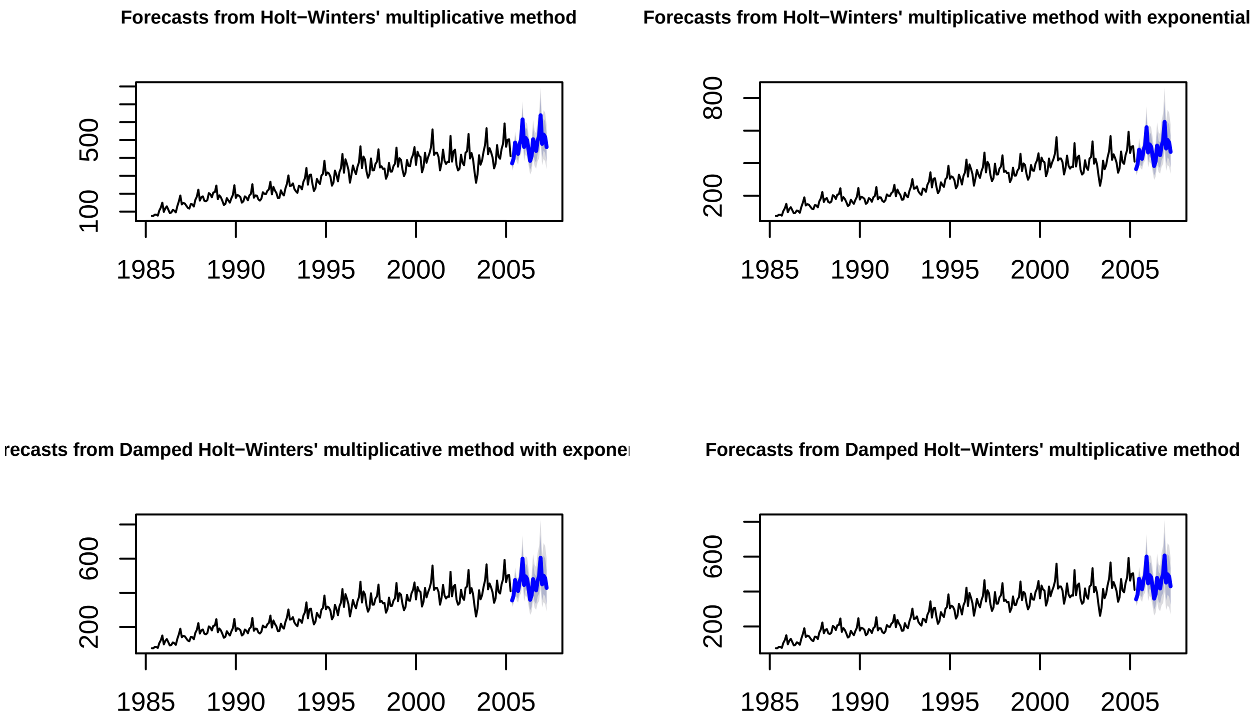


Figure 17: Experimentation with damping and exponential

```r
plot(hw(y = visitors,seasonal = "multiplicative", exponential = T, damped = T),cex.main = 0.7)
plot(hw(y = visitors,seasonal = "multiplicative", exponential = F, damped = T),cex.main = 0.7)
```

```r
Models <- list(hw.m.fit = hw.m.fit,hw.m.exp = hw(y = visitors,seasonal = "multiplicative", exponential =
               hw.m.exp.damp = hw(y = visitors,seasonal = "multiplicative", exponential = T, damped = T
               hw.m.damp = hw(y = visitors,seasonal = "multiplicative", exponential = F, damped = T))
df <- do.call("rbind",purrr::map(Models, ~sw_glance(.x$model)))

knitr::kable(df, caption = "Model Summary & Performance")
```

Table 2: Model Summary & Perfo

	model.desc	sigma	logLik	AIC
hw.m.fit	Holt-Winters' multiplicative method	0.0549701	-1299.795	2633.589
hw.m.exp	Holt-Winters' multiplicative method with exponential trend	0.0555770	-1299.884	2633.767
hw.m.exp.damp	Damped Holt-Winters' multiplicative method with exponential trend	0.0544229	-1295.373	2626.745
hw.m.damp	Damped Holt-Winters' multiplicative method	0.0542224	-1294.409	2624.818

f) Figure 15 shows the forecastss from the various models. Table 3 shows the respectinve RMSE. The hybrid model of STL-ETS on Box-Cox transformed data is the best from the RMSE perspective, however the residual signature makes it a poor model. Both the ETS models seems to work well for this data.

```r
ets.mdl <- ets(visitors)
ets.additive <- ets(visitors,additive.only = T,lambda = BoxCox.lambda(visitors))
snaive.mdl <- snaive(visitors,lambda = BoxCox.lambda(visitors))
stl.decop <- stl(x = BoxCox(visitors,BoxCox.lambda(visitors)),s.window = "periodic",robust = T)
ets.seasadj <- ets(stl.decop$time.series[,"trend"])
stl.ets <- ets.seasadj$fitted + stl.decop$time.series[,"seasonal"]
stl.ets.fitted <- InvBoxCox(stl.ets,BoxCox.lambda(visitors))
stl.ets.res <- visitors - stl.ets.fitted
```

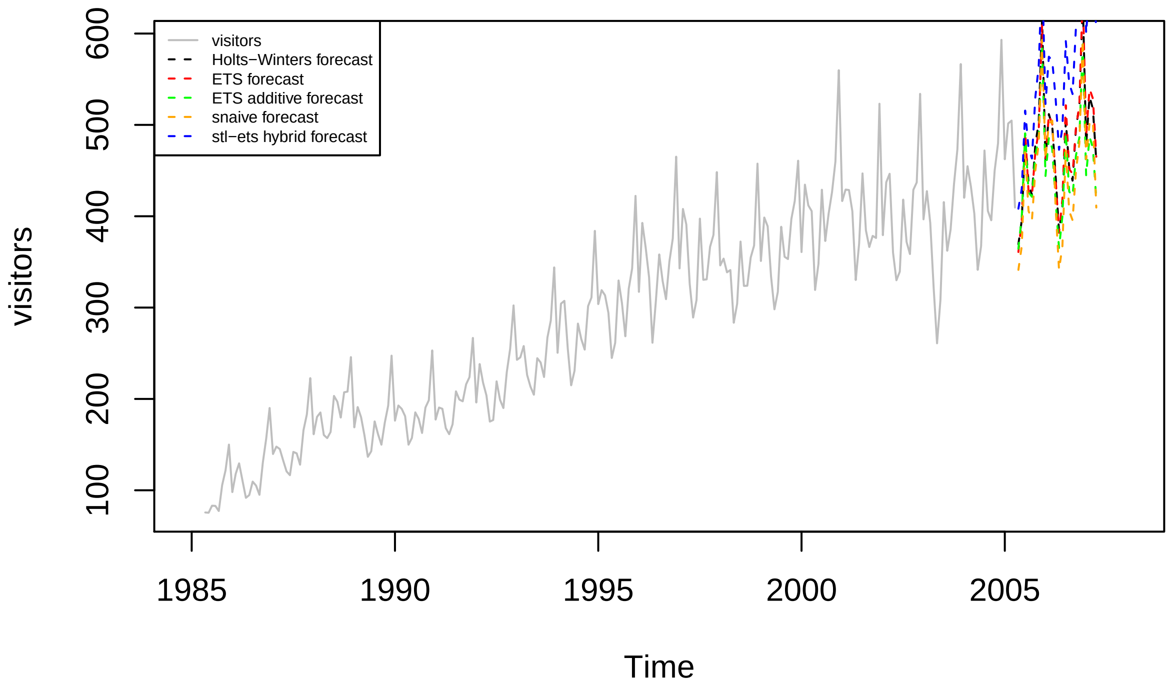


Figure 18: Forecast of visitors

```r
legend("topleft",legend = c("visitors", "Holts-Winters forecast",
                            "ETS forecast", "ETS additive forecast",
                            "snaive forecast","stl-ets hybrid forecast"),
       lty = c(1,rep(2,5)), col = c("grey","black","red","green","orange","blue"),cex = 0.5)
```

```r
# RSME generation
mdls <- list(ets.mdl,ets.additive)
#sw_glance(snaive.mdl$model)
mdls_RMSE <- purrr::reduce(purrr::map(mdls,sw_glance),bind_rows) %>% select(model.desc,RMSE) %>%
  dplyr::mutate(RMSE = round(RMSE,2)) %>%
  rbind.data.frame(
    data.frame(model.desc = c("snaive","stl.ets"),
    RMSE = round(c(accuracy(snaive.mdl)[2], sqrt(mean(stl.ets.res^2))),2)
    )

  )

knitr::kable(mdls_RMSE,caption = "RMSE for model fits")
```

Table 3: RMSE for model fits

model.desc	RMSE
ETS(M,A,M)	15.86
ETS(A,Ad,A)	15.58
snaive	32.57
stl.ets	14.32

```r
par(mfrow = c(2,3))
r1 <- checkresiduals(hw.m.fit)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 47.794, df = 8, p-value = 1.081e-07
```

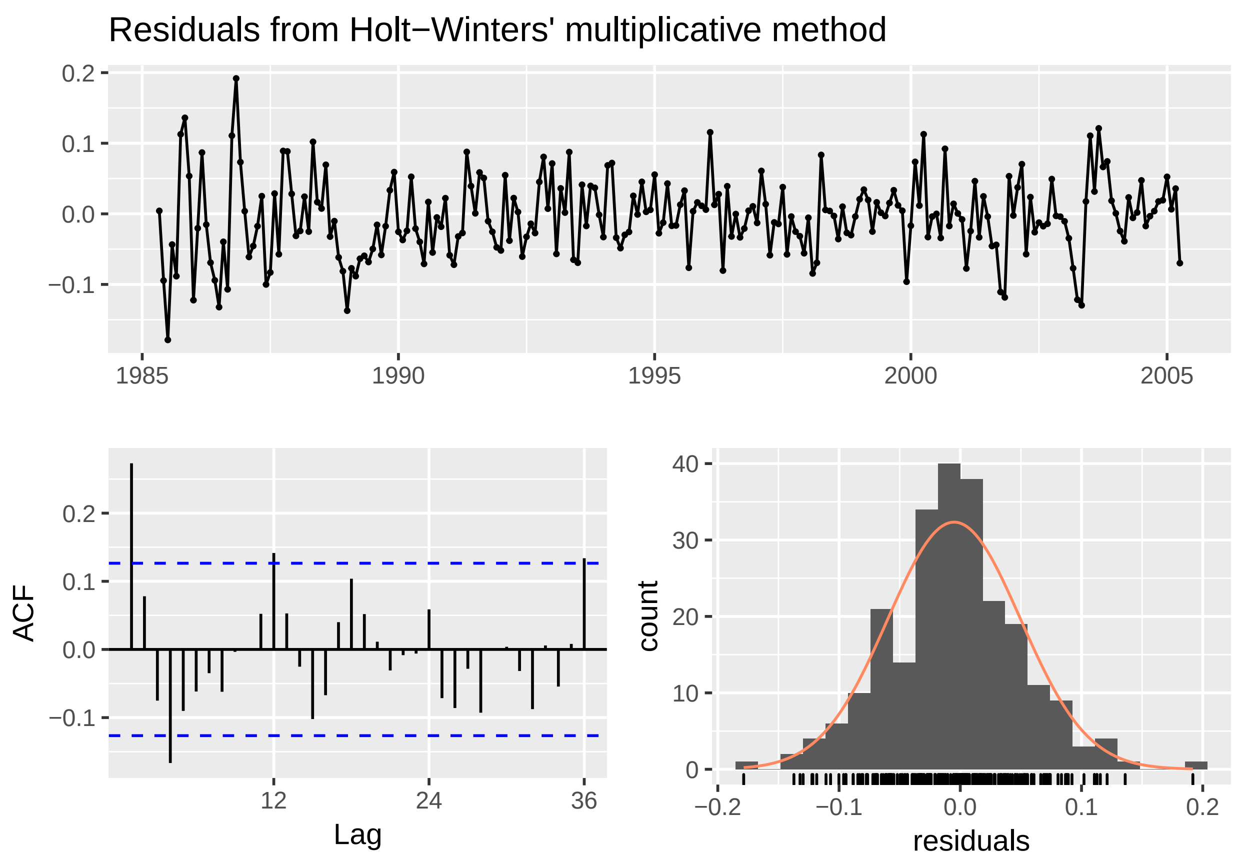


Figure 19: Residual Analysis of models

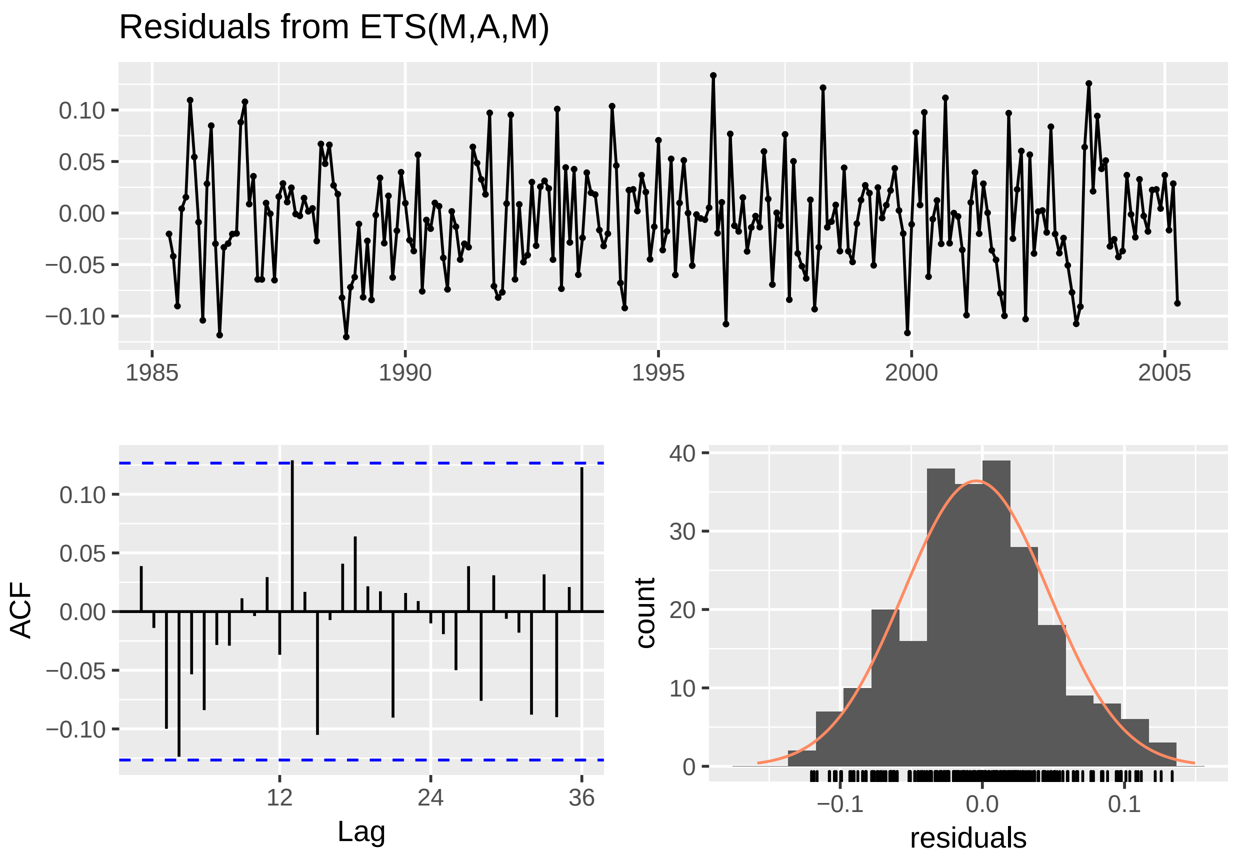


Figure 20: Residual Analysis of models

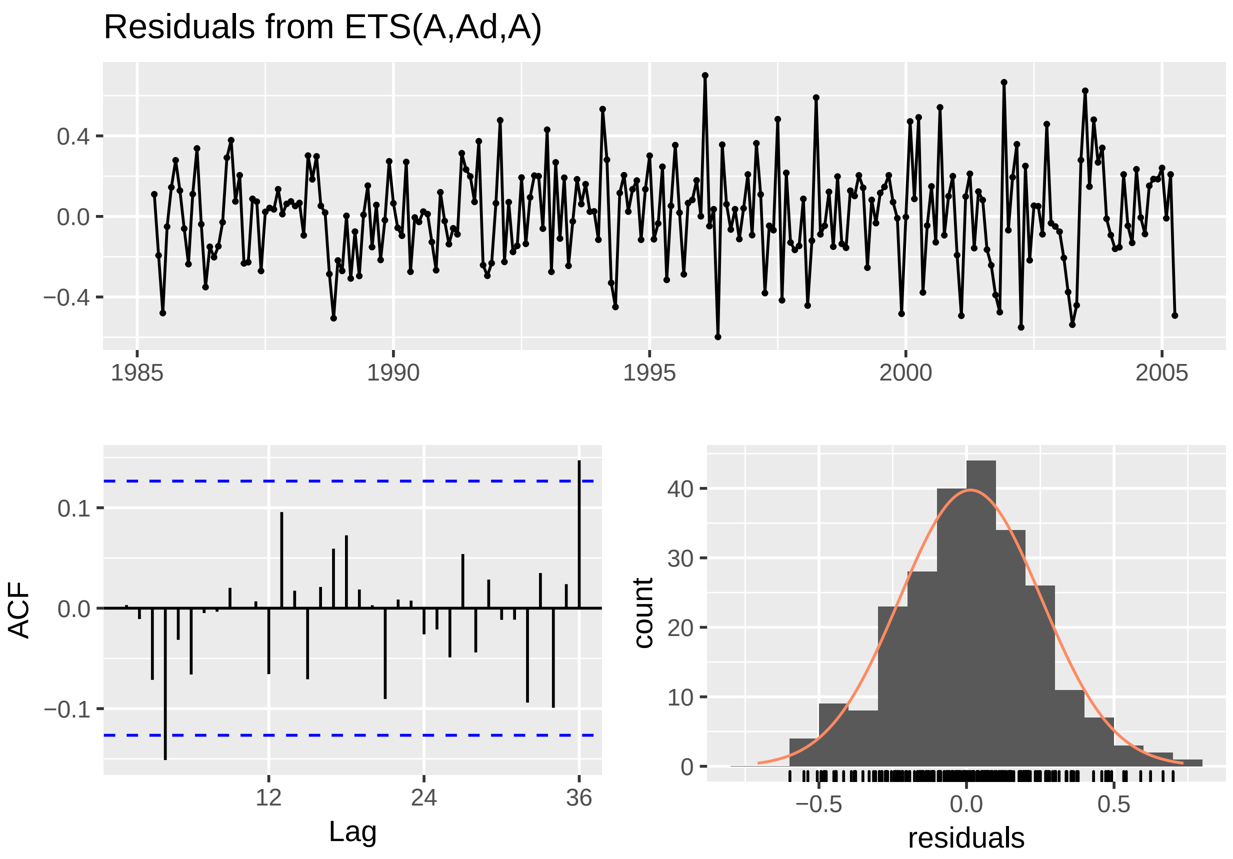


Figure 21: Residual Analysis of models

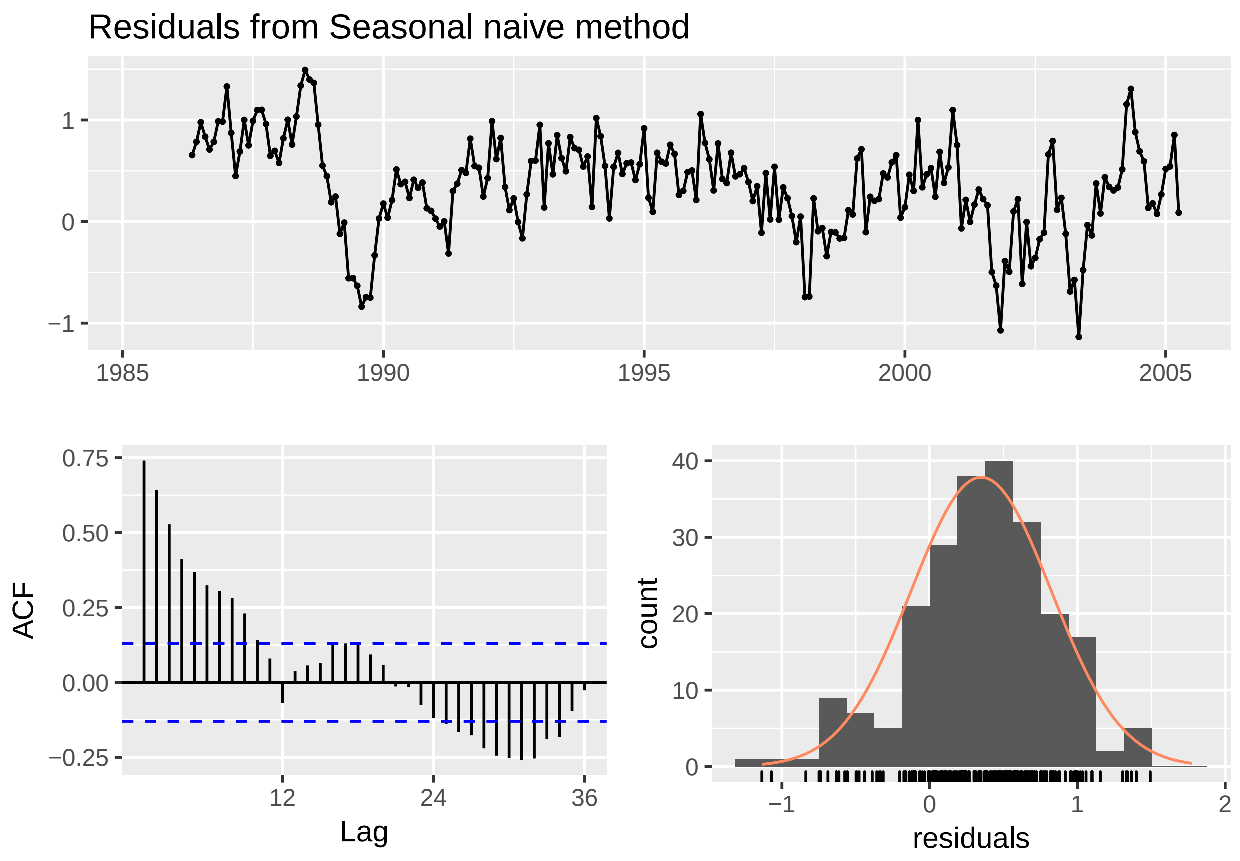


Figure 22: Residual Analysis of models

```
##
## Model df: 17.   Total lags used: 24
```

```
r4 <- checkresiduals(snaive.mdl)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 468.11, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

```
r5 <- checkresiduals(stl.ets.res)
```

Section 8, Q 8.5

a & b)

The auto regressive model of order k is given by

y_{t} = c + phi_{1}y_{t-1} + phi_{2}y_{t-2} + … + phi_{t-p}y_{t-p} + e_{t}

For AR(1) model …

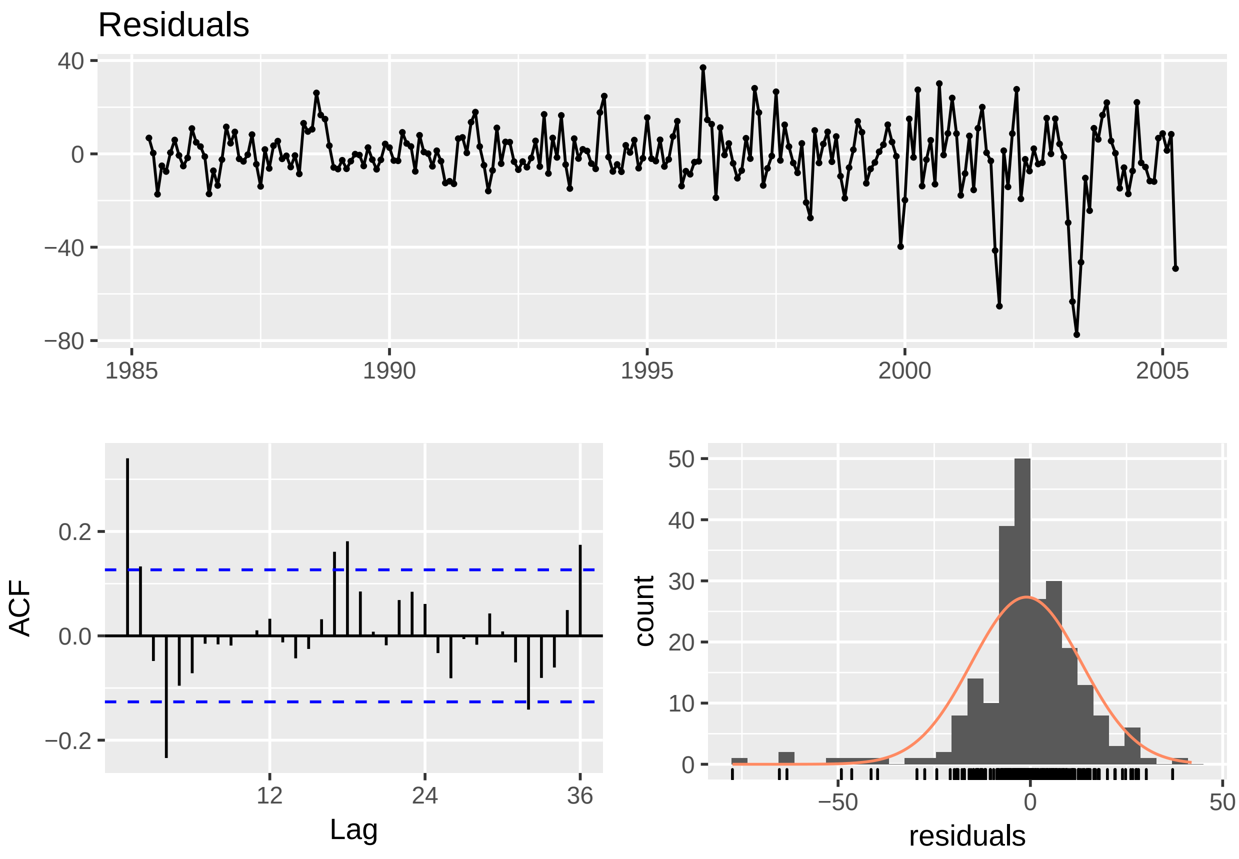


Figure 23: Residual Analysis of models

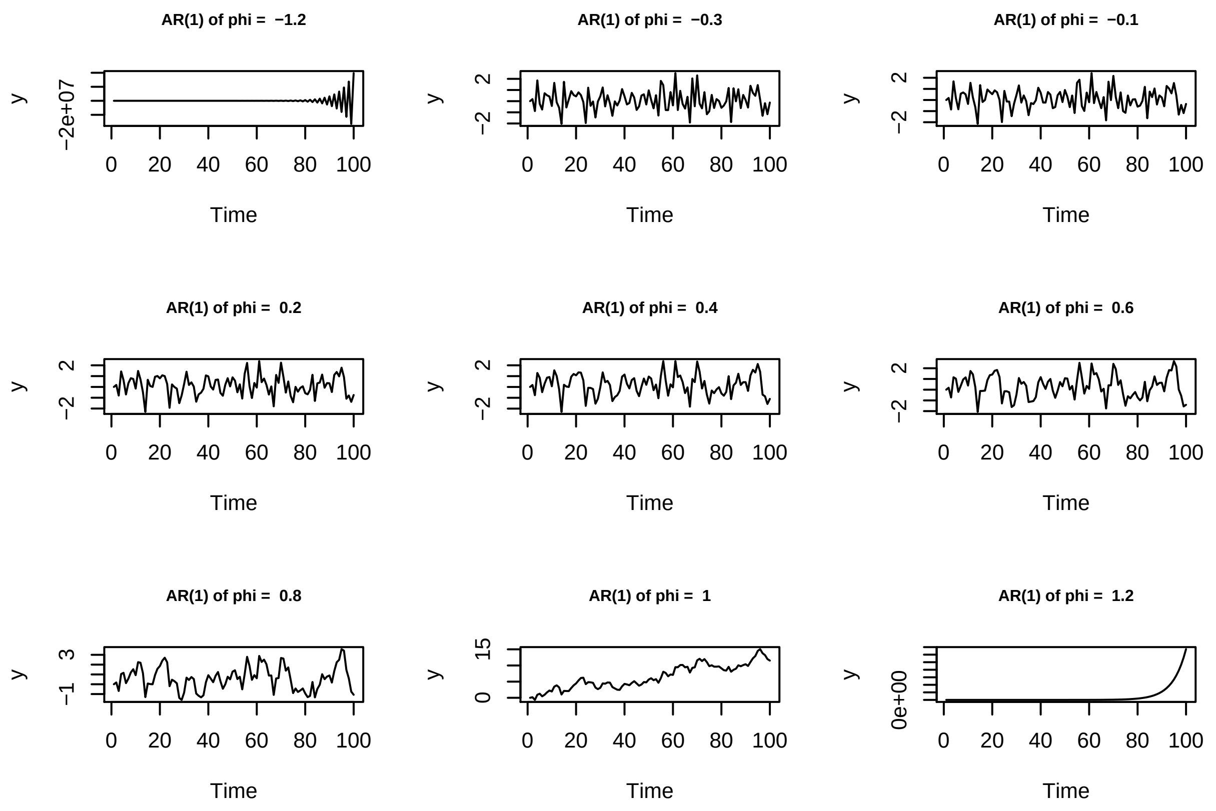


Figure 24: AR(1) simulations

y_{t} = c + phi_{1}y_{t-1} + e_{t}

Where e_t is normally distributed error with mean zero and variance (σ^2) 1. Here c is assumed to be 0.

```
AR1simul <- function(phi){
  set.seed(1)
  y <- ts(numeric(100))
e <- rnorm(100)
for(i in 2:100)
  y[i] <- phi*y[i-1] + e[i]
plot(y, main = paste("AR(1) of phi = ",phi), cex.main = 0.7)
}
# y <- AR1simul(0.05)
# set.seed(1)
# ytest <- arima.sim(list(ar=0.05),n = 100)
phi <- c(-1.2,-0.3, -0.1, 0.2, 0.4, 0.6, 0.8,1, 1.2)
par(mfrow = c(3,3))
test <- lapply(phi,AR1simul)
```

It can be seen from figure 17 as the ϕ_1 increases in magnitude the series becomes non-stationery. When $\phi_1 = 1$, the series becomes a randomwalk.

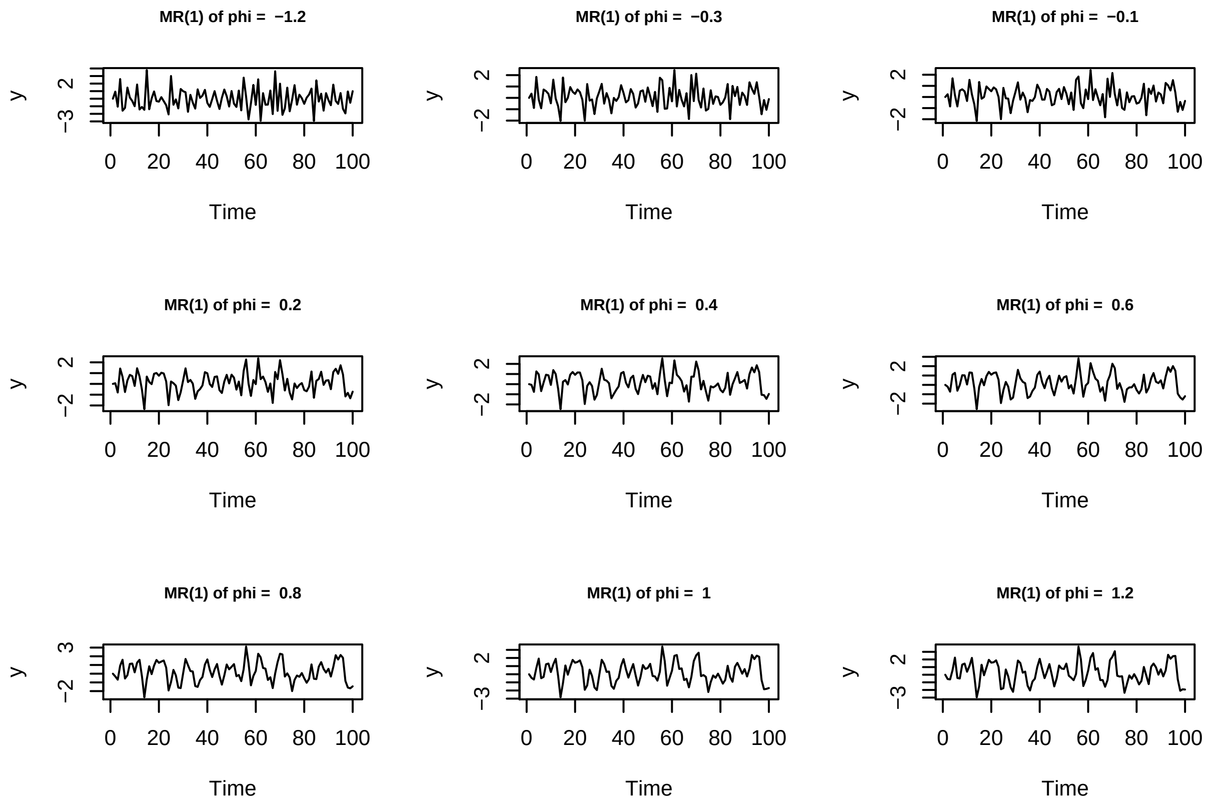
```
arimsim <- function(ars) {
  sim <- arima.sim(list(ma = ars),n = 100)
  ts.plot(sim)
}
```

```
test2 <- lapply(phi[10],arimsim)
```

c & d)

Figure 18 shows that as the ϕ_1 changes different time series are generated.

```
MR1simul <- function(phi){
  set.seed(1)
  y <- ts(numeric(100))
e <- rnorm(100)
for(i in 2:100)
  y[i] <- phi*e[i-1] + e[i]
plot(y, main = paste("MR(1) of phi = ",phi), cex.main = 0.7)
}
par(mfrow = c(3,3))
test <- lapply(phi,MR1simul)
```



e,f & g

ARMA(1,1) seems stationary, however the AR(2) is not.

```
ts.sim <- arima.sim(list(order = c(1,0,1), ar = 0.6, ma = 0.6), n = 100)
AR2simul <- function(phi1,phi2){
  set.seed(1)
  y <- ts(numeric(100))
e <- rnorm(100)
for(i in 3:100)
```

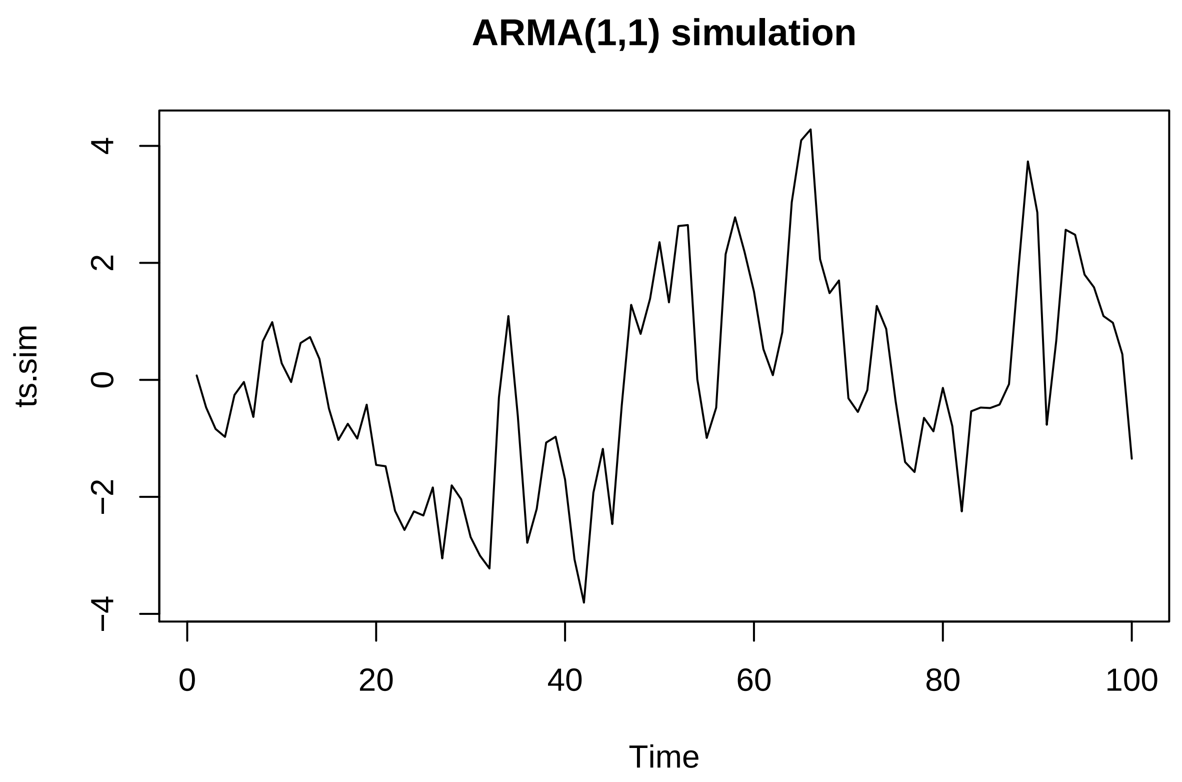


Figure 25: ARMA(1,1) and AR(2) comparison

```
  y[i] <- phi*y[i-1] + phi2*y[i-2] + e[i]
plot(y, main = paste("AR(2) of phi1 = ",phi1," & phi2 = ",phi2), cex.main = 0.7)

}


par(mfror = c(2,1))
ts.plot(ts.sim, main = "ARMA(1,1) simulation")
```

```
AR2simul(-0.8,0.3)
```

Question 8.6

a & d)

Figure 21 shows the time series of wmurders. I would guess with the changes in level like stock data, it would need at least one differencing. A naive model with forecast equaling the previous data would likely be better. I am thinking either 1 regressive part or 1 moving average (may be both?) might work best.

```
data("wmurders")
autoplot(wmurders)
```

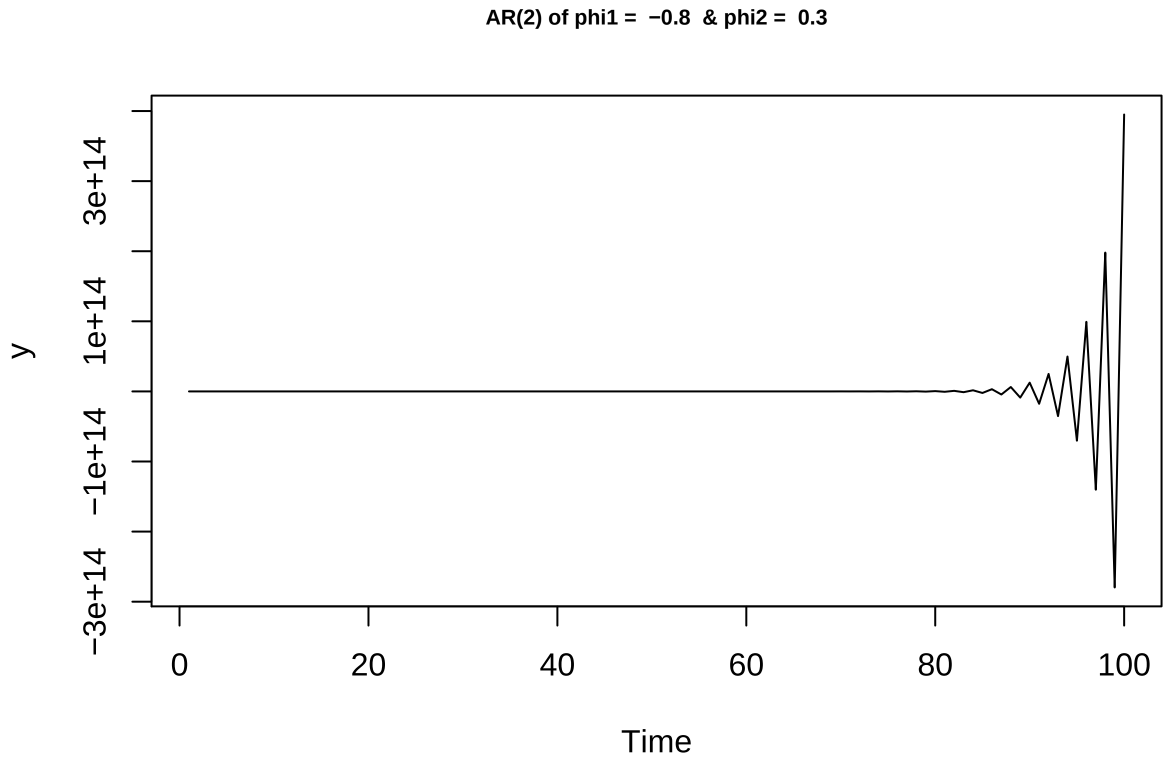
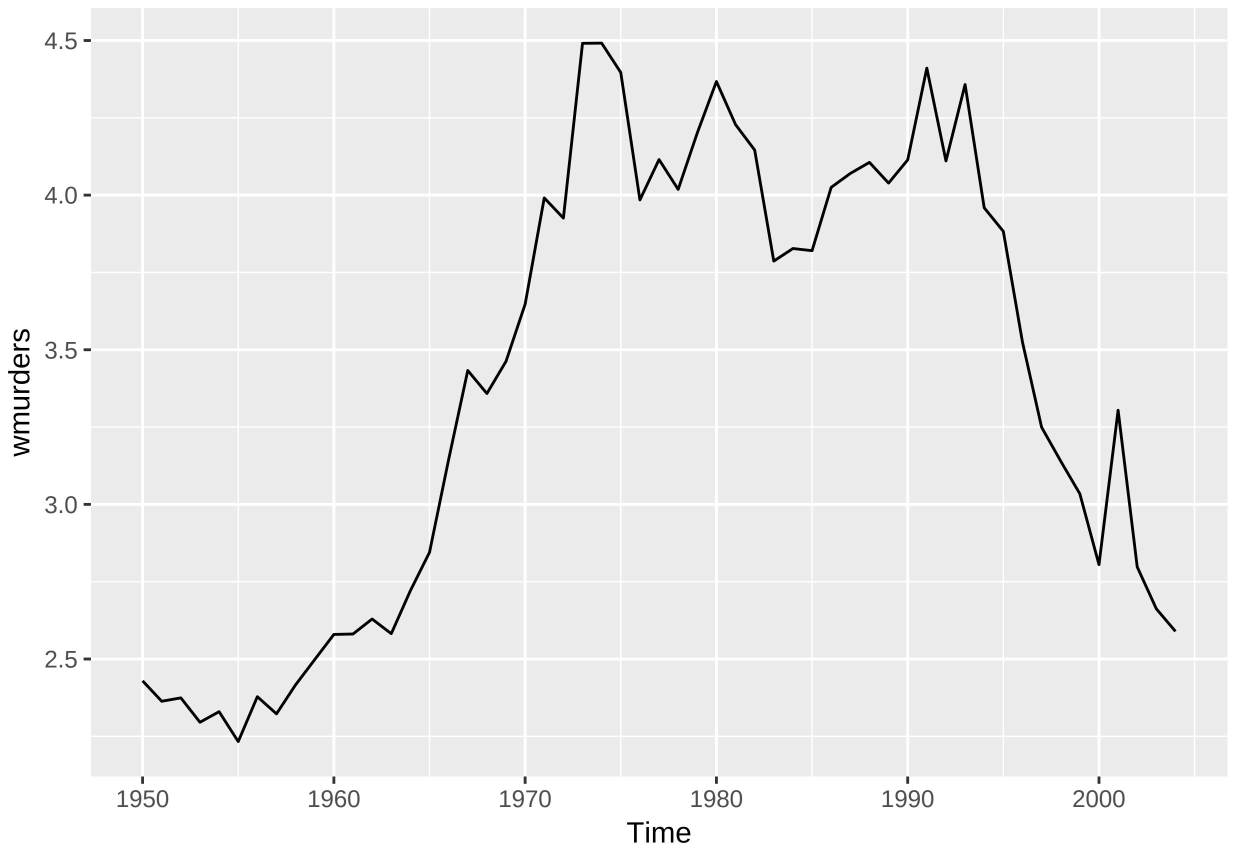


Figure 26: ARMA(1,1) and AR(2) comparison



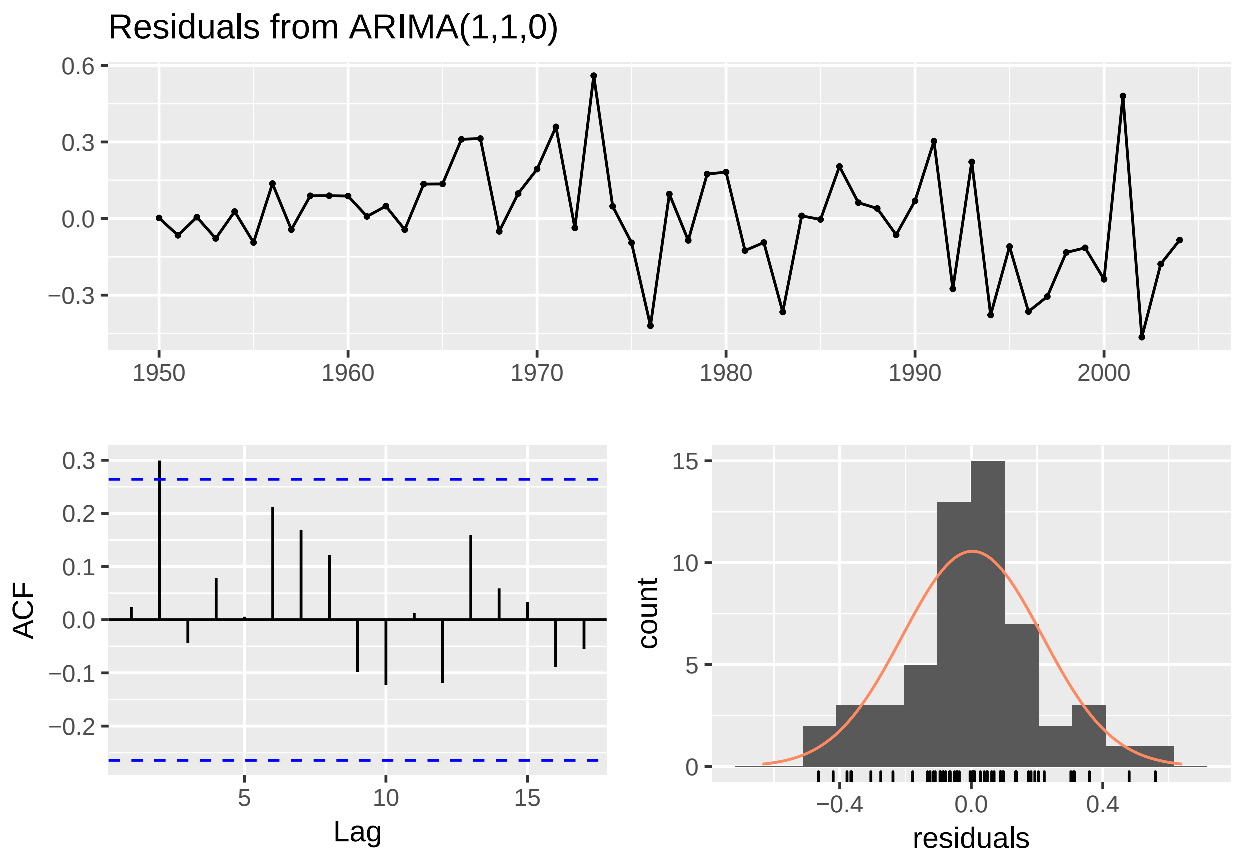


Figure 27: Residual checks for wmirders

Here we'll fit the ARIMA(1,1,0), ARIMA(0,1,1) and ARIMA(1,1,1). From the accuracy of the models, all the three models are close. From the residual perspective ARIMA(1,1,1) seems relatively better.

```
mdl1 <- Arima(wmurders,order = c(1,1,0))
mdl2 <- Arima(wmurders,order = c(0,1,1))
mdl3 <- Arima(wmurders,order = c(1,1,1))

purrr::reduce(.x = purrr::map(list(mdl1,mdl2,mdl3),.f = sw_glance),.f = bind_rows)
```

```
## # A tibble: 3 x 12
##   model.desc   sigma logLik    AIC   BIC      ME  RMSE   MAE     MPE  MAPE
##   <chr>        <dbl>  <dbl>  <dbl> <dbl>   <dbl> <dbl> <dbl>   <dbl> <dbl>
## 1 ARIMA(1,1,0) 0.215   6.92 - 9.85 -5.87 0.00331 0.211 0.160 -0.0623  4.65
## 2 ARIMA(0,1,1) 0.215   6.85 - 9.70 -5.72 0.00320 0.211 0.160 -0.0627  4.65
## 3 ARIMA(1,1,1) 0.212   8.18 -10.4  -4.39 0.00384 0.206 0.155 -0.0322  4.47
## # ... with 2 more variables: MASE <dbl>, ACF1 <dbl>
```

```
checkresiduals(mdl1)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)
## Q* = 13.281, df = 9, p-value = 0.1503
##
## Model df: 1.   Total lags used: 10
```

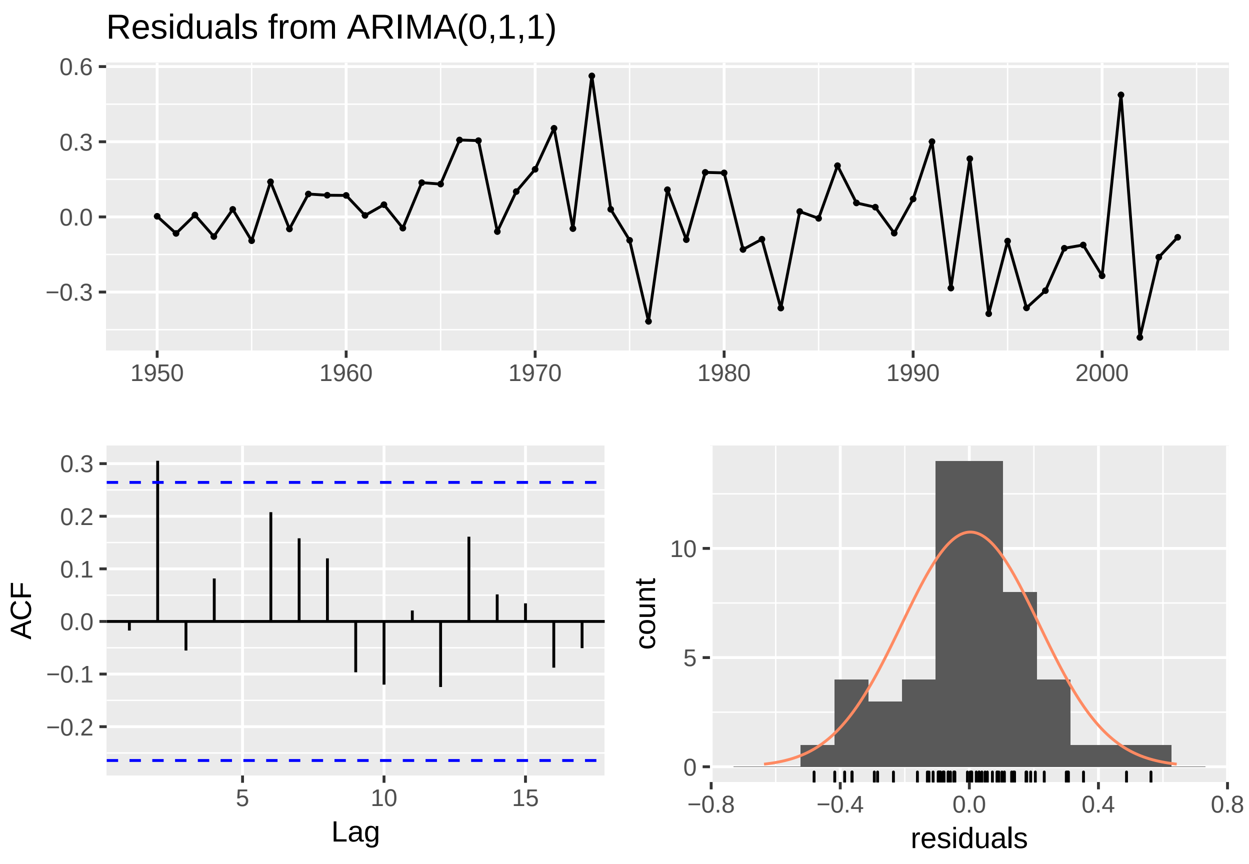


Figure 28: Residual checks for wmirders

```r
checkresiduals(mdl2)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,1,1)
## Q* = 13.122, df = 9, p-value = 0.1572
##
## Model df: 1.   Total lags used: 10
```

```r
checkresiduals(mdl3)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)
## Q* = 11.266, df = 8, p-value = 0.1871
##
## Model df: 2.   Total lags used: 10
```

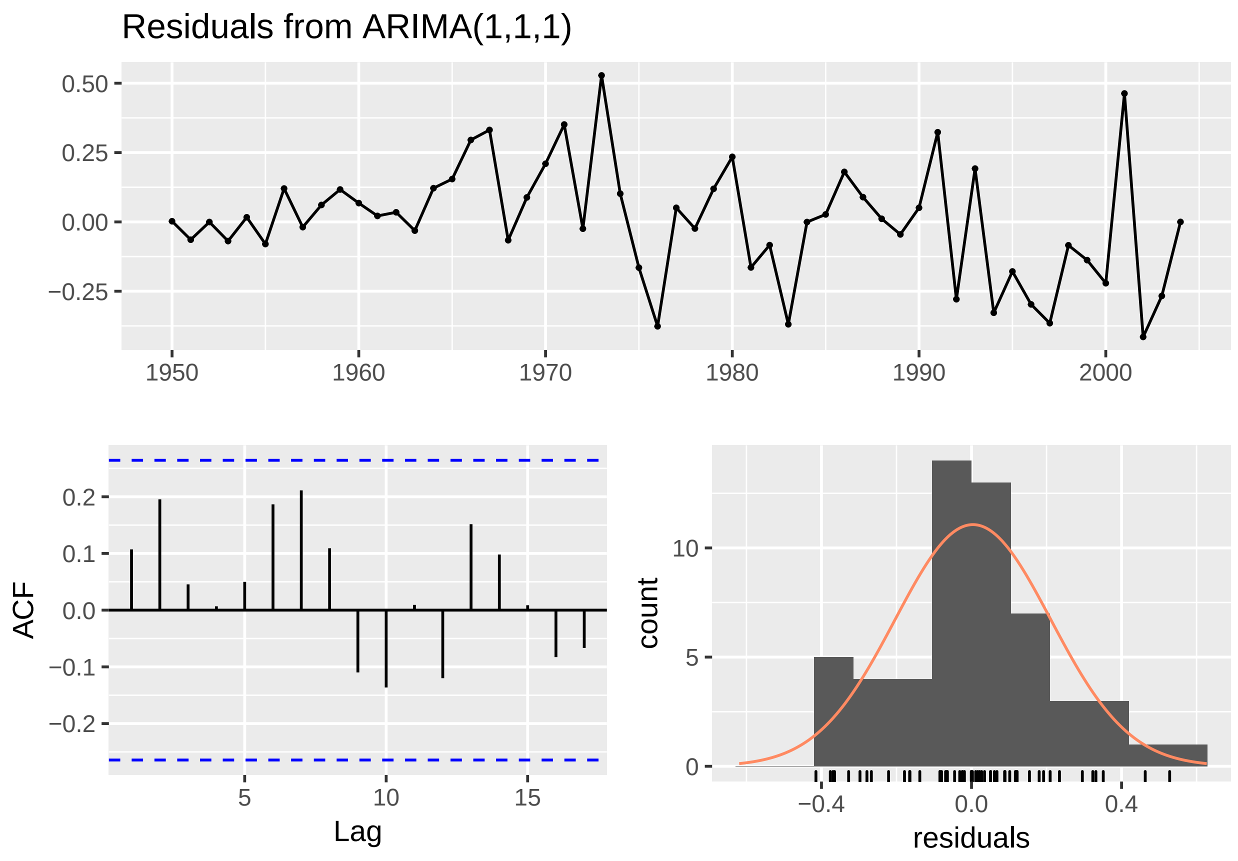


Figure 29: Residual checks for wmirders

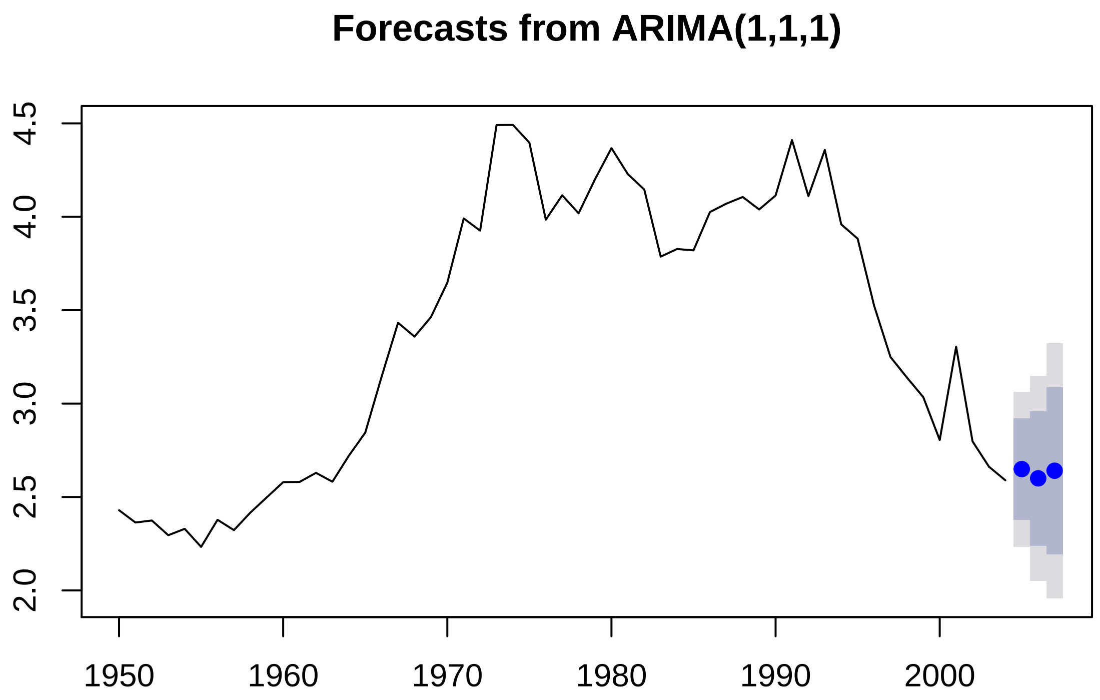


Figure 30: Forecast with Arima

b) Adding a constant adds a drift to the forecast. The time series is not contantly increasing / decreasing. Hence adding a constant may not make sense for this data.

c)

(1 - phi1B) (1-B)yt = (1 + theta1)et

e & f)

```r
plot(forecast(mdl3, h = 3))
```

```r
arimaauto <- auto.arima(wmurders)
accuracy(arimaauto)
```

```
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.01065956 0.2072523 0.1528734 -0.2149476 4.335214 0.9400996
##                    ACF1
## Training set 0.02176343
```

```r
checkresiduals(arimaauto)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,2,1)
## Q* = 12.419, df = 8, p-value = 0.1335
##
```

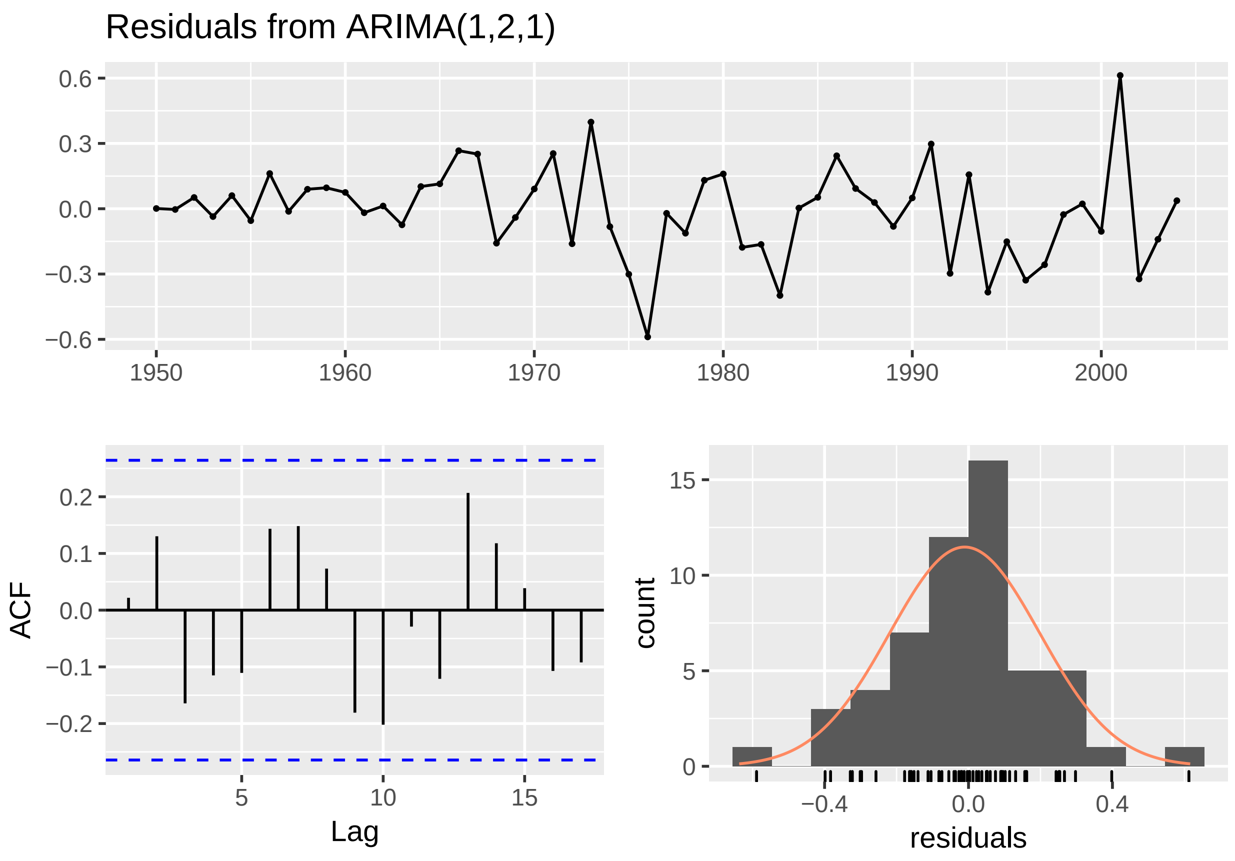


Figure 31: Auto Arima model

```
## Model df: 2.   Total lags used: 10
```

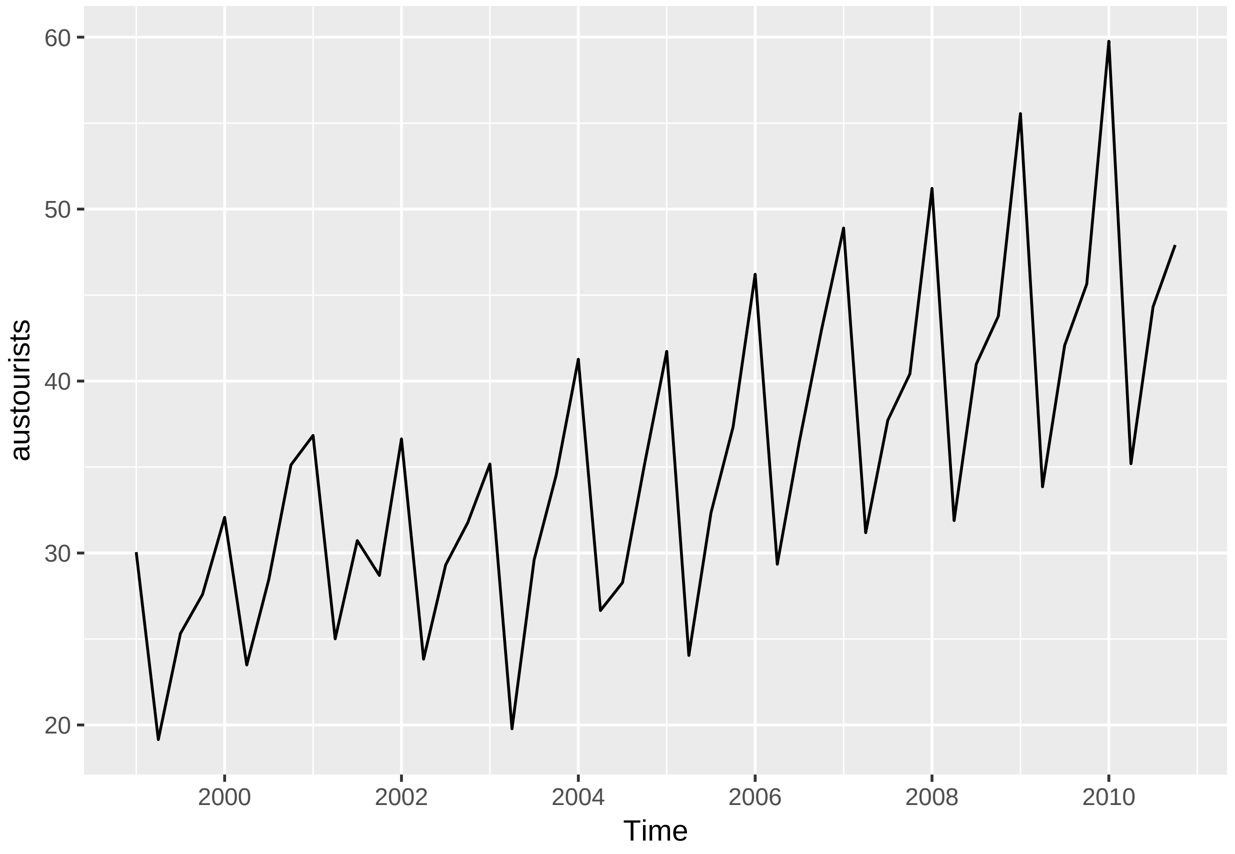
g) The ARIMA(1,1,1) and ARIMA(1,2,1) are very comparable in its forecast accuracy.

8.7

a)

The austourists data has both seasonality and trend.

```
data("austourists")
autoplot(austourists)
```



## b & c) The ACF captures the trend and nearly shows all lags are significant . . . The PACF removes the effect of the trend and shows tha tthe seasonality (lag 4) shows up as the key correlation factor.

```
par(mfrow = c(1,2))
Acf(austourists)
pacf(austourists)
```

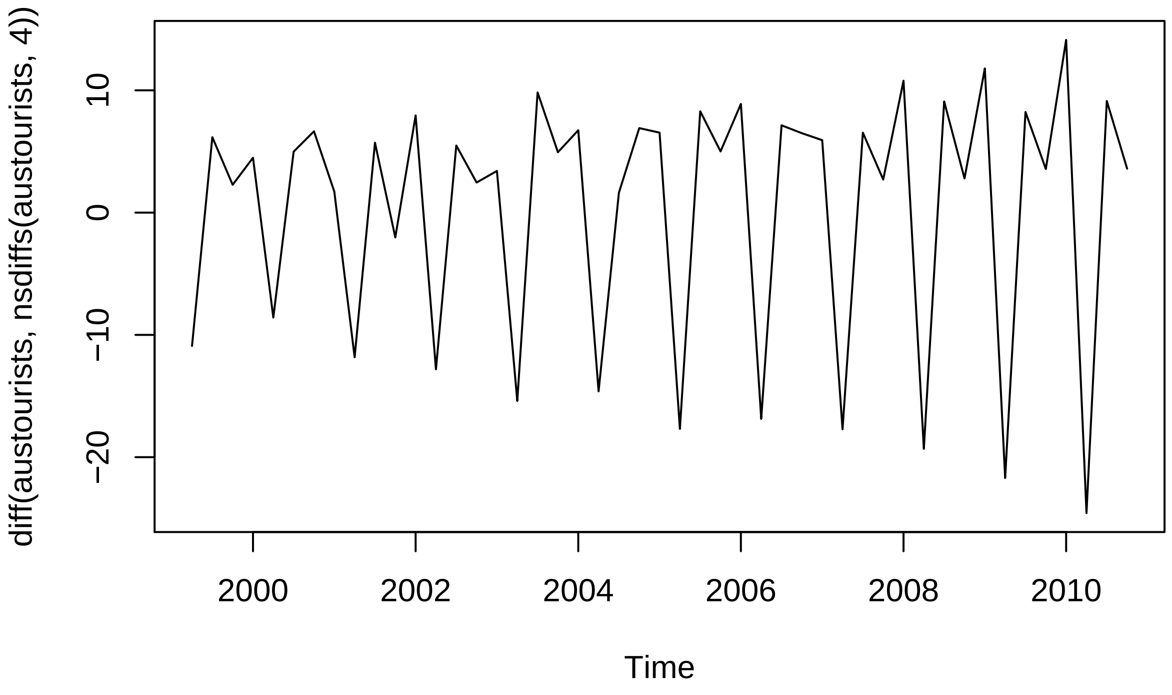
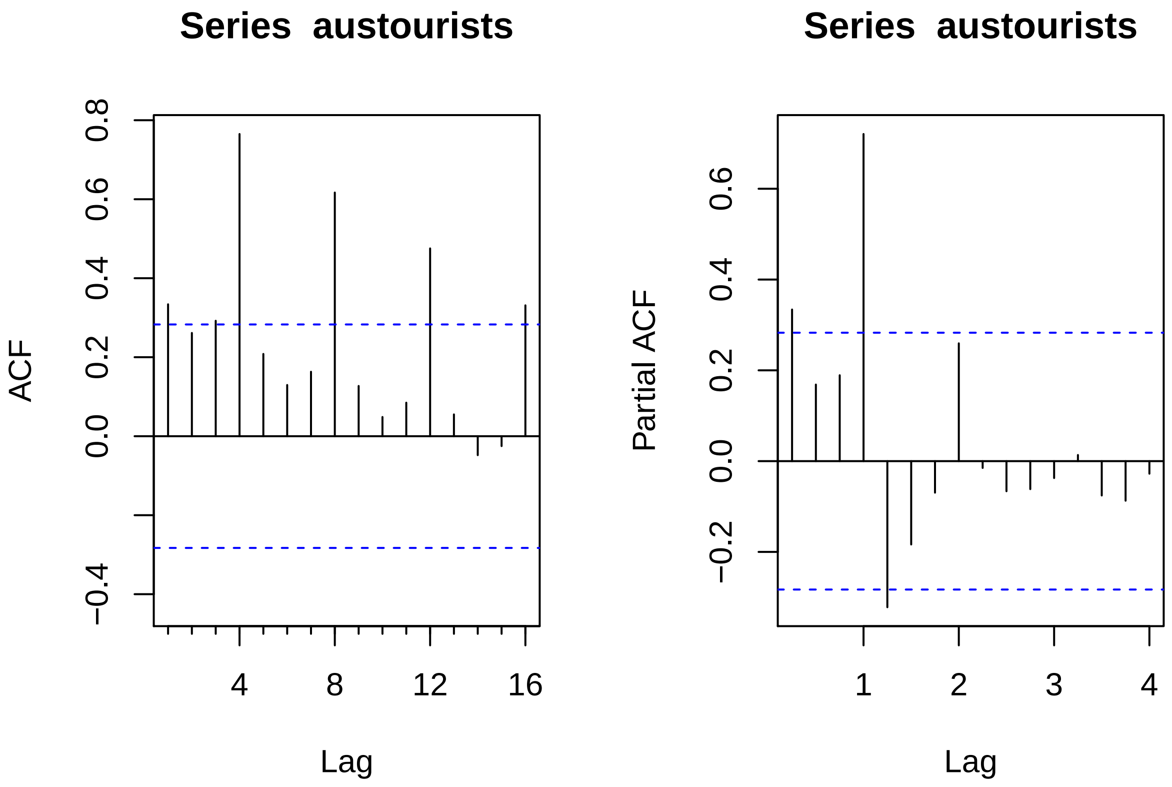


Figure 32: austourists



##d)

```
plot(diff(austourists,nsdiffs(austourists,4)))
```

The seasonal differences still show a seasonality in the data. ARIMA(1,0,0)(1 or 4?,1,0) may work.

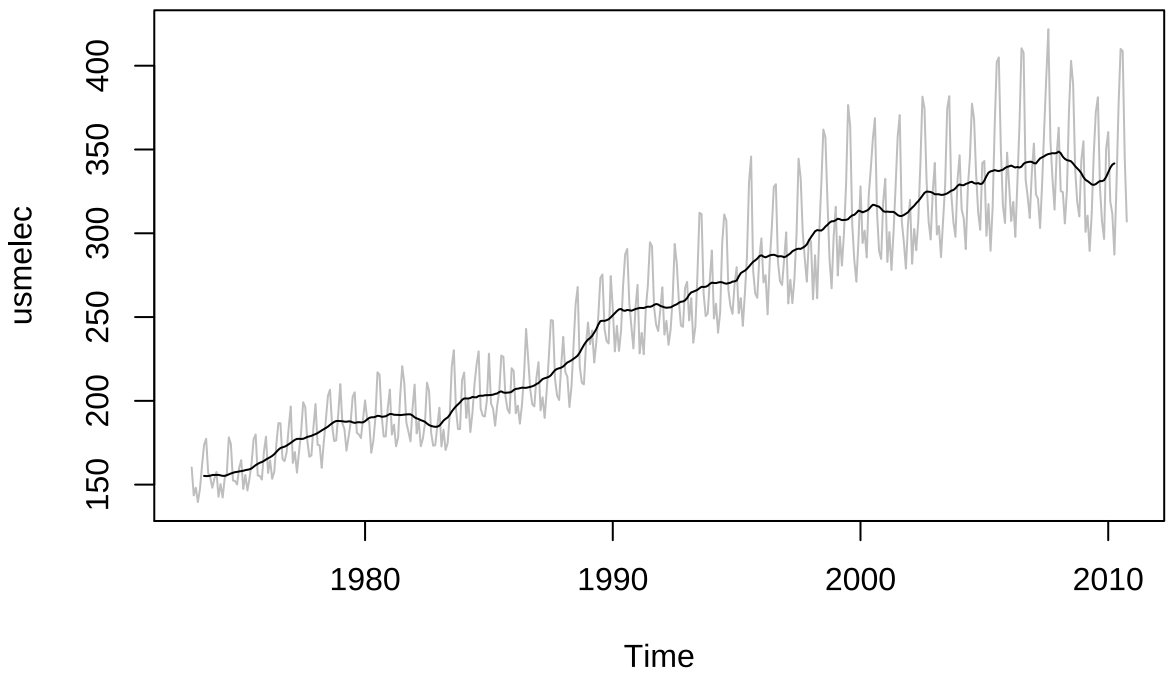


Figure 33: time series of usmelec

```r
auto.arima(austourists)
```

```
## Series: austourists
## ARIMA(1,0,0)(1,1,0)[4] with drift
##
## Coefficients:
##          ar1     sar1   drift
##       0.4493  -0.5012  0.4665
## s.e.  0.1368   0.1293  0.1055
##
## sigma^2 estimated as 5.606:  log likelihood=-99.47
## AIC=206.95   AICc=207.97   BIC=214.09
```

f)

(1-phi1B) (1-B^4)yt = et

8.8

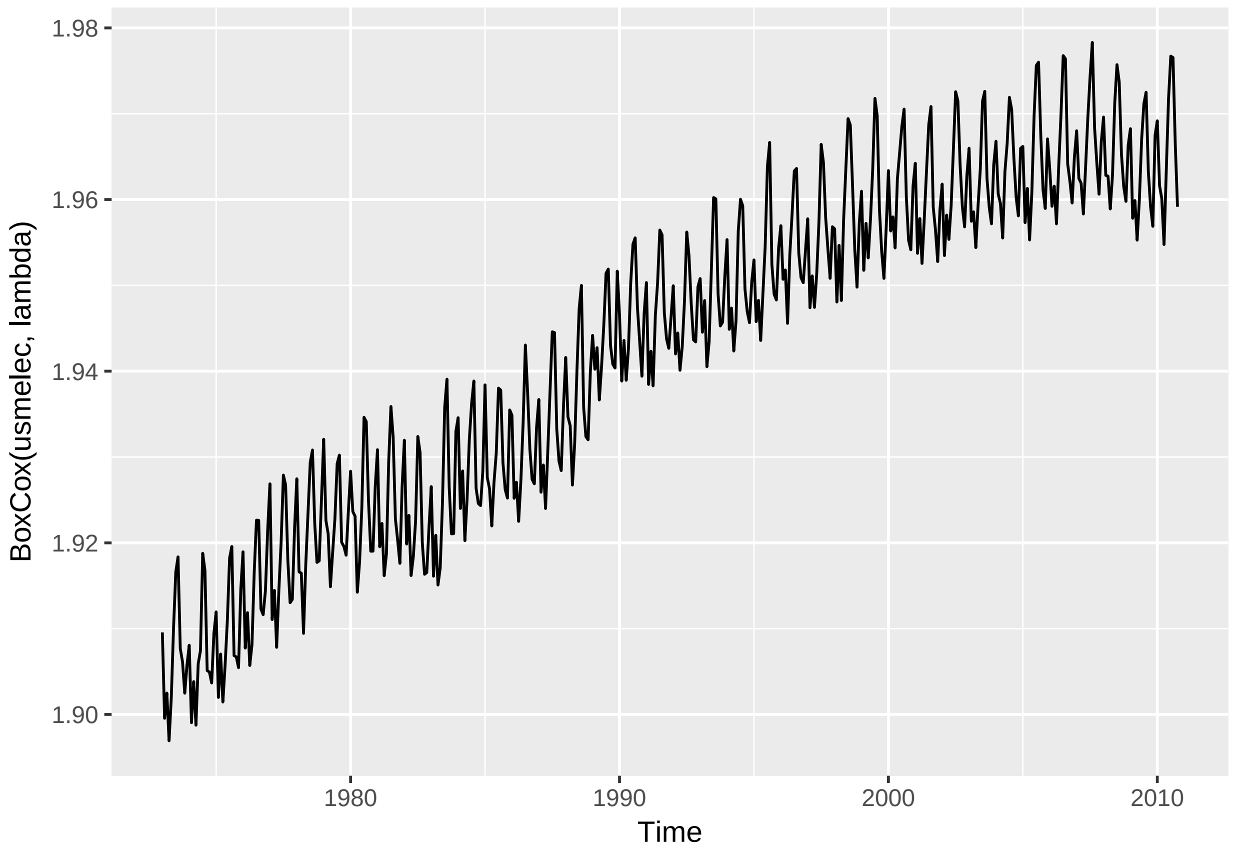
a) The time series shows increasing trend and the variation in peaks is proportional to time.

```r
data("usmelec")

plot(usmelec,col = "grey")
lines(forecast::ma(usmelec, 12))
```

b)

```
lambda <- BoxCox.lambda(usmelec)
autoplot(BoxCox(usmelec,lambda))
```



The boxcox transformation has reduced the variability in the peaks and valleys. There is seasons in the data. Hence not stationary. A 12 month differencing might help, as shown below.

```
usmelectrans <- BoxCox(usmelec,lambda)
plot(diff(usmelectrans,12))
```

d & e)

The auto.arima model came out to be the better from AIC perspective and RMSE perspective. The residuals look like whitenoise.

```
fit1 <- Arima(usmelectrans, order = c(1,0,1), seasonal = c(1,0,1))
fit2 <- Arima(usmelectrans, order = c(2,1,1), seasonal = c(2,0,1))
fit3 <- auto.arima(usmelectrans)
mdls2 <- list(fit1,fit2,fit3)

knitr::kable(purrr::reduce(purrr::map(mdls2,.f = sw_glance),bind_rows), caption = "Model comparison")
```

model.desc	sigma	logLik	AIC	BIC	ME	RMSE

Table 4: Model comparison

model.desc	sigma	logLik	AIC	BIC	ME	RMSE	
ARIMA(1,0,1)(1,0,1)[12] with non-zero mean	0.0019580	2165.253	-4318.505	-4293.797	0.0001228	0.0019472	0.0
ARIMA(2,1,1)(2,0,1)[12]	0.0018945	2179.817	-4345.634	-4316.823	-0.0000682	0.0018798	0.0
ARIMA(2,1,2)(0,0,2)[12] with drift	0.0031415	1966.482	-3916.965	-3884.038	0.0000079	0.0031137	0.0

```r
checkresiduals(fit3)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,1,2)(0,0,2)[12] with drift
## Q* = 224.05, df = 17, p-value < 2.2e-16
##
## Model df: 7.   Total lags used: 24
```

f & g)

Forecasts are reasonable in short terms and not on the long term. Forecast models with drift need to be used carefully.

```r
fcts <- forecast(fit3, h = 180)
plot(fcts$mean)
```

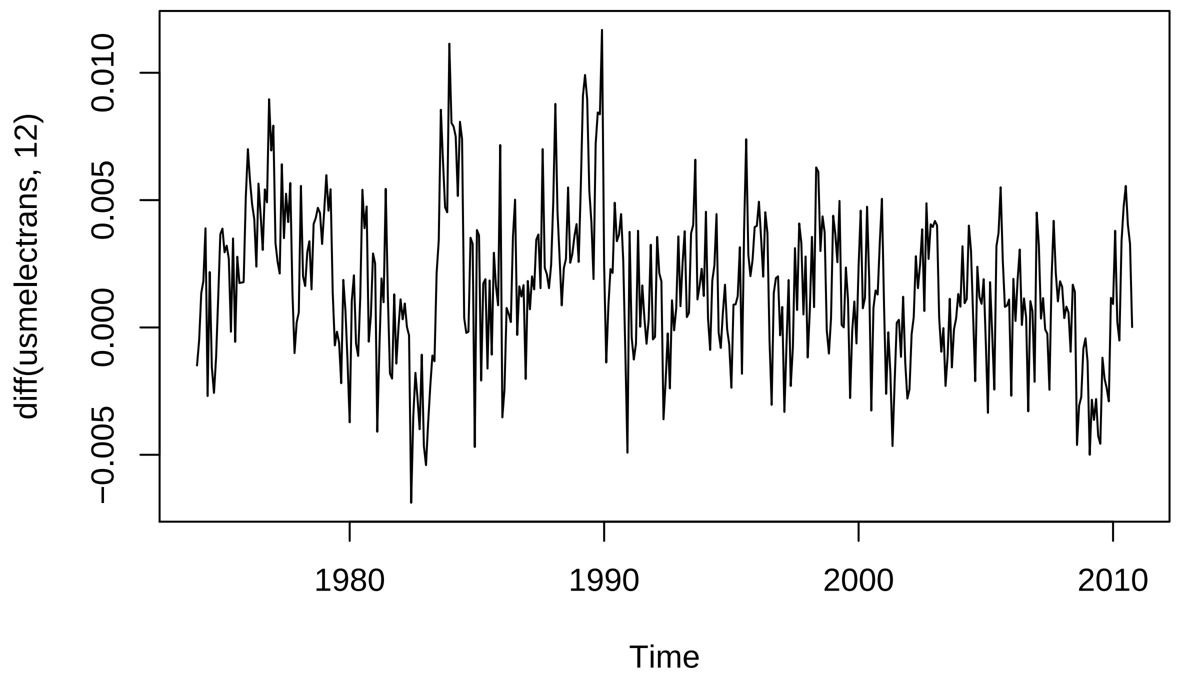


Figure 34: 12 month differencing of transformed usmelec

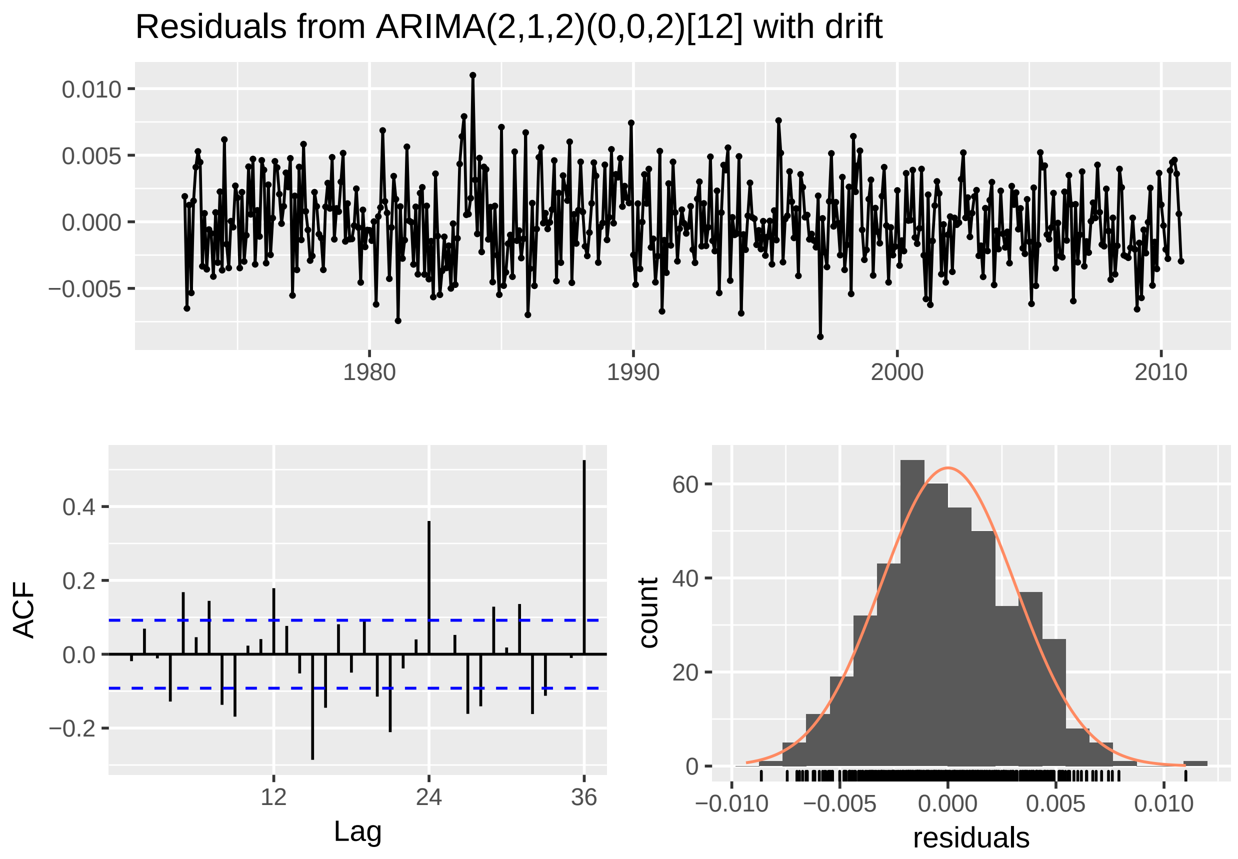


Figure 35: Residual check

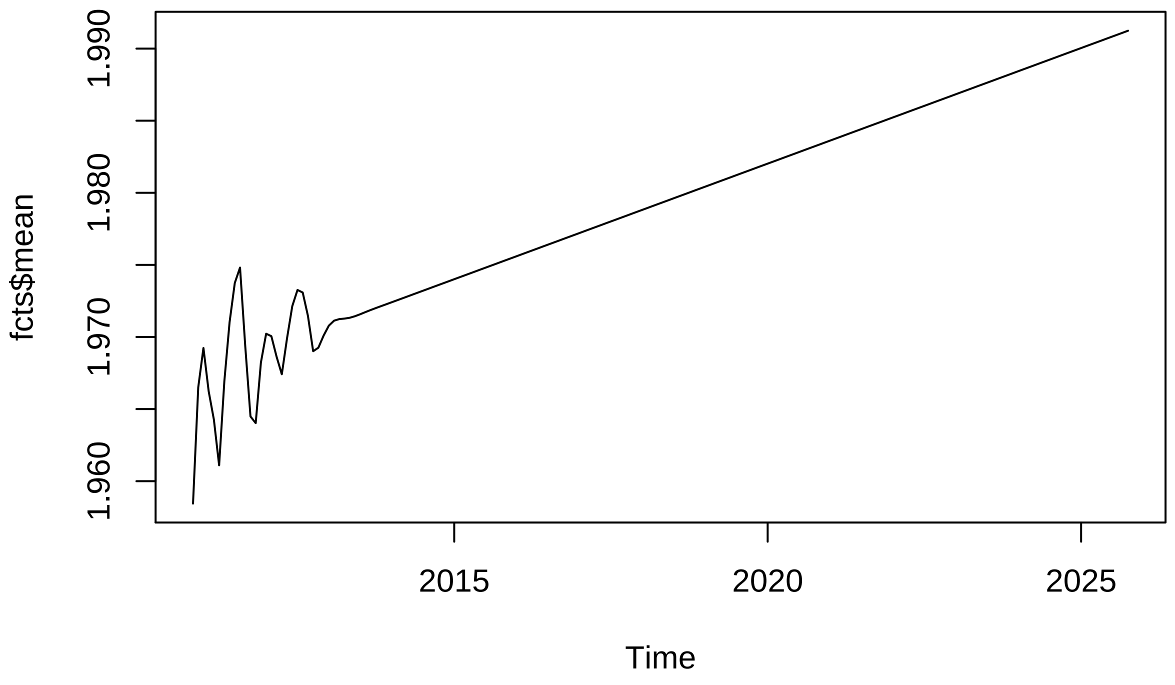


Figure 36: 15 year projection