HW 2

Rahul Sangole

April 29, 2018

Contents

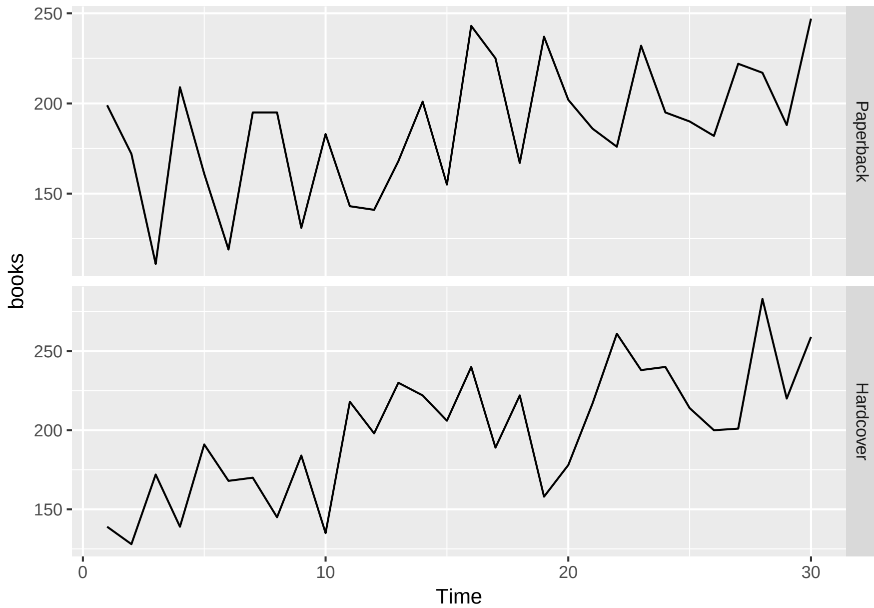
Section 7.8

Data set books contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days' sales for paperback and hardcover books

Plot the series and discuss the main features of the data.

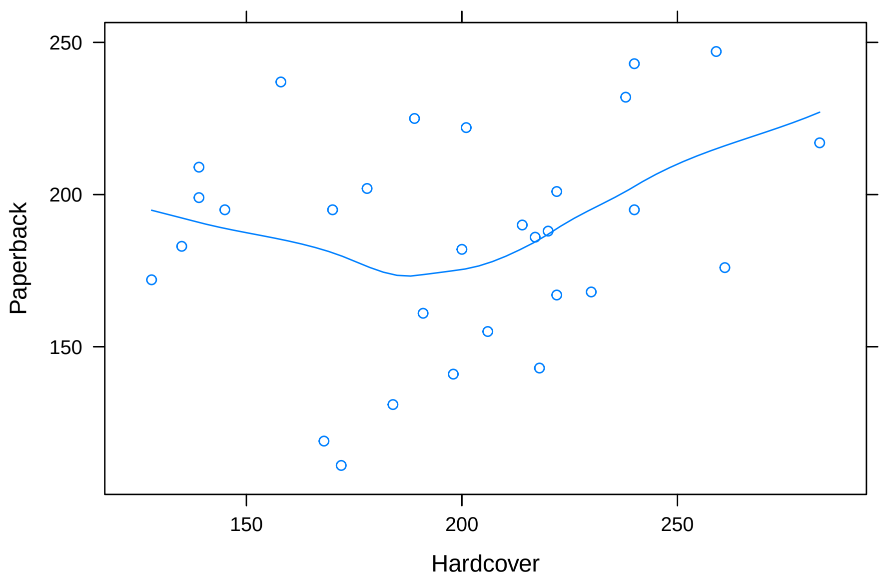
Both the time series show a linear upward trend. Visually, the `Paperback` seems to have some regular patters (seasonality) approximately equal to 3 days.

```
autoplot(books, facets = T)
```



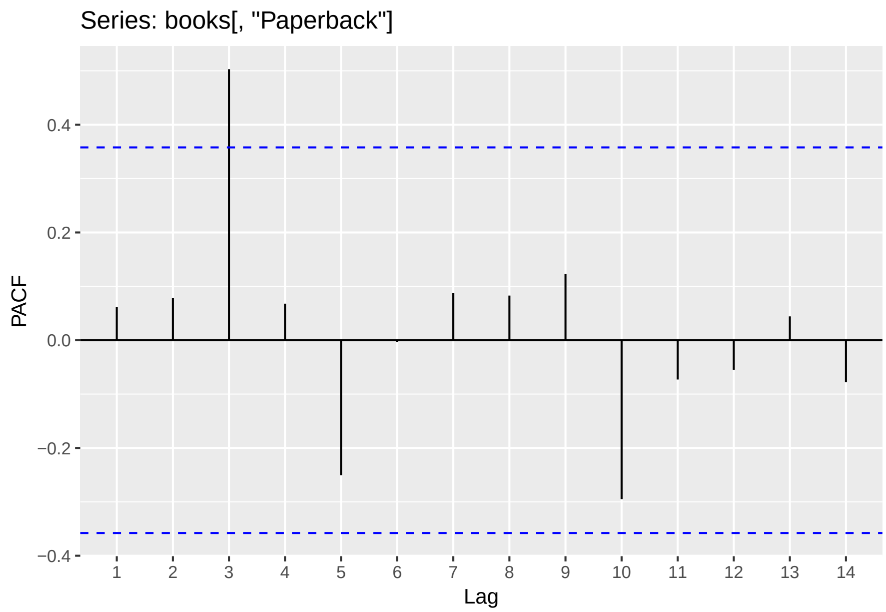
There doesn't seem to be any linear correlation between the two series.

```
lattice::xyplot(Paperback~Hardcover, as.data.frame(books), type=c('p','smooth'))
```

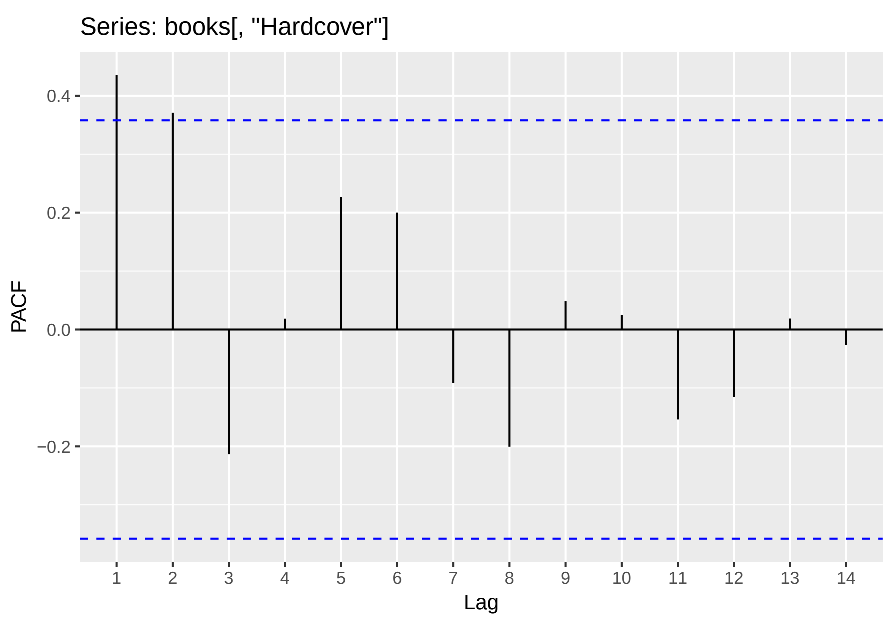


The PACF plots do show that for `Paperback`, there is a 3-order autocorrelation in the signal which is significant. For `Hardcover`, there is a 1st and 2nd order significance.

```
ggPacf(books[,'Paperback'])
```



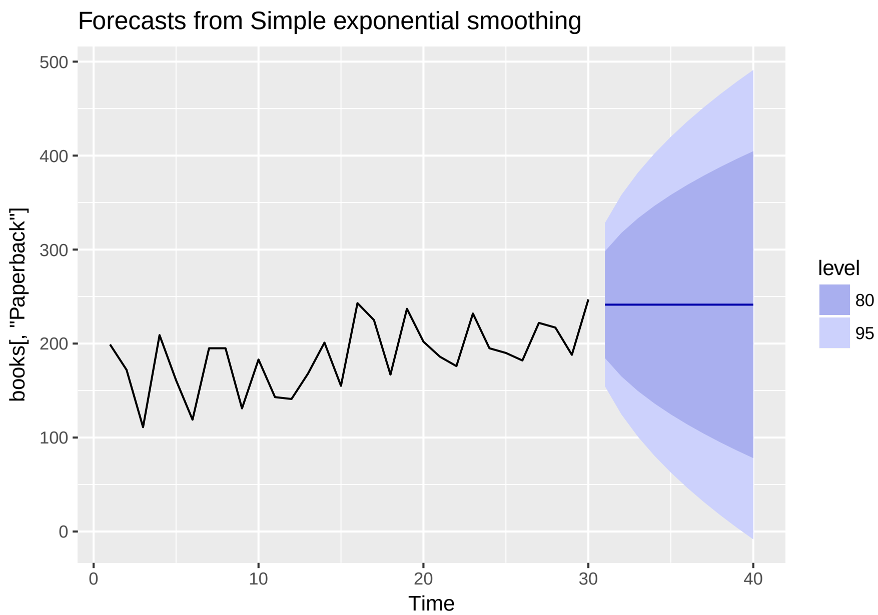

```r
ggPacf(books[,'Hardcover'])
```



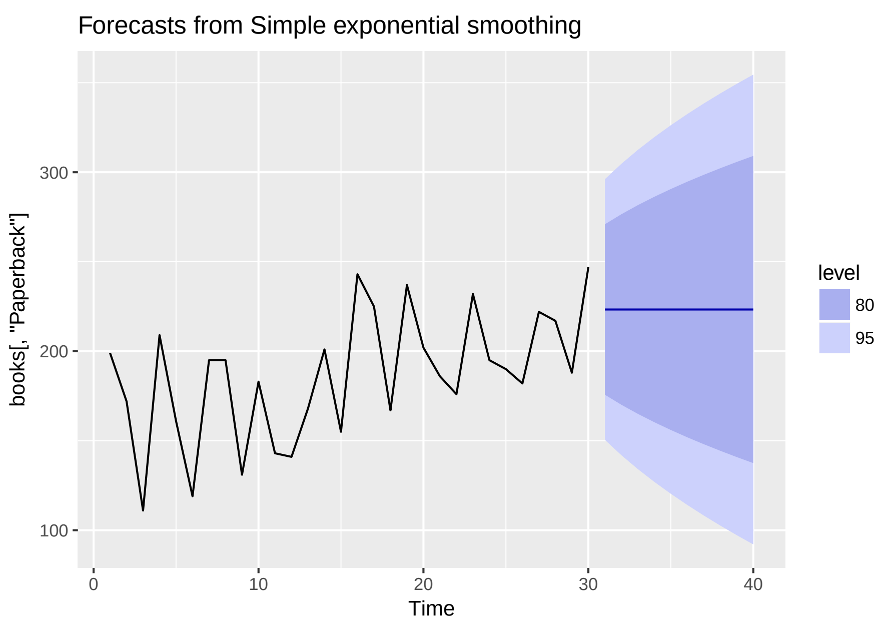
Use simple exponential smoothing with the ses function (setting initial="simple") and explore different values of alpha for the paperback series.

Here are three settings - a=0.9, a=0.5, and a=0.01. As `alpha` increases, so does the uncertainty of the prediction since `ses` will look back further in time. Though at alpha = 0.01, the point estimate seems quite low.

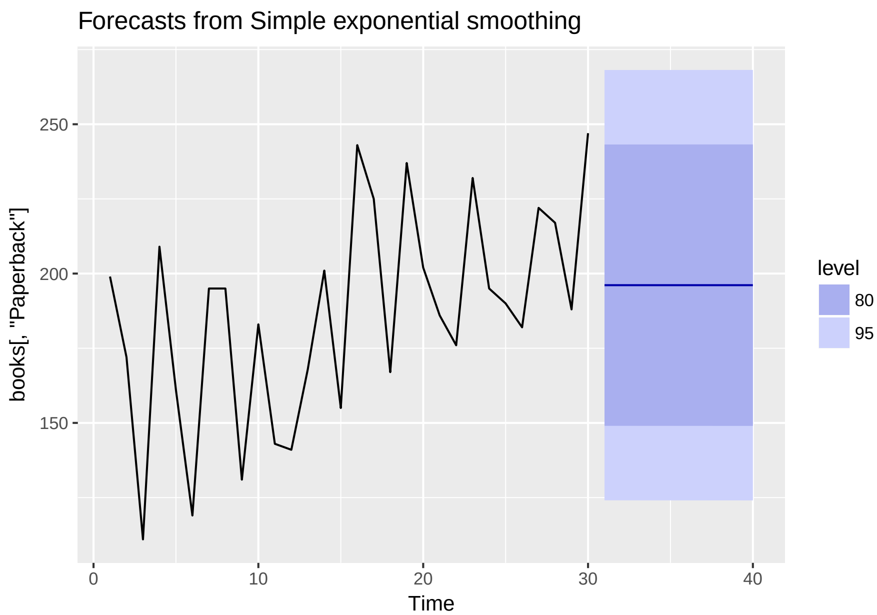
```r
ses(books[,'Paperback'], initial = 'simple', alpha = .9)  %>% forecast() %>% autoplot()
```




```r
ses(books[,'Paperback'], initial = 'simple', alpha = .5)  %>% forecast() %>% autoplot()
```




```r
ses(books[,'Paperback'], initial = 'simple', alpha = .01) %>% forecast() %>% autoplot()
```



Record the within-sample SSE for the one-step forecasts. Plot SSE against alpha and find which value of alpha works best. What is the effect of alpha on the forecasts?

We can see a typical curve as seen during parameter tuning. The SSE is minimum at an alpha value of about 0.2.

```
sse_list <- c()
a_list <- seq(0.001, 0.999, length.out = 20)
for (a_sel in a_list) {
    ses(books[,'Paperback'], initial = 'simple', alpha = a_sel)$model$SSE -> sse
    sse_list <- c(sse_list, sse)
}
plot(a_list, sse_list, type='b')
points(a_list[which.min(sse_list)], sse_list[which.min(sse_list)], col='red', pch=20)
```

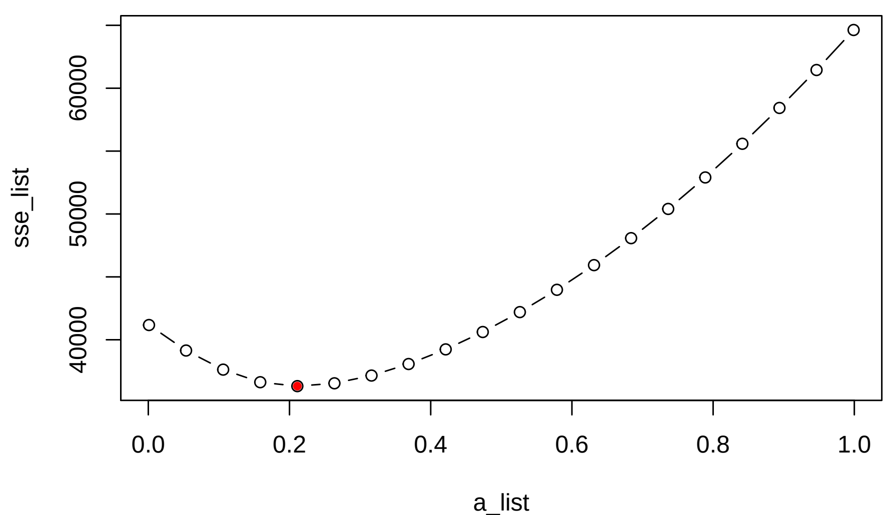


Table 1: 1st line: Auto-alpha. 2nd line: a=0.01

.rownames	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	1.749509	34.79175	28.64424	-2.770157	16.56938	0.7223331	-0.1268119
Training set	-9.605222	36.75235	28.40972	-9.519865	17.64342	0.7164189	0.0845229

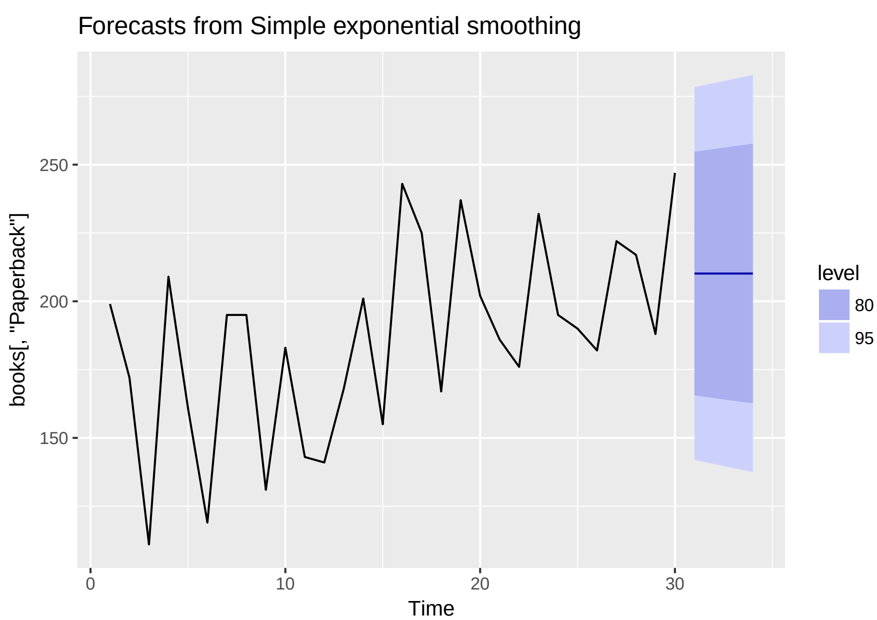
Now let ses select the optimal value of alpha. Use this value to generate forecasts for the next four days. Compare your results with 2.

Alpha selected is 0.215. The point estimate for this forecast seem better than the #2 results. Prediction interval widths are about similar. RMSE is smaller for the auto-alpha selection.

```
ses(books[,'Paperback'], initial = 'simple', alpha = NULL)$model$par['alpha']
```

```
##     alpha
## 0.2125115
```

```
ses(books[,'Paperback'], initial = 'simple', alpha = NULL)%>% forecast(h=4)%>% autoplot()
```




```
bind_rows(ses(books[,'Paperback'], initial = 'simple', alpha = NULL) %>%
                forecast(h=4) %>% accuracy() %>% sweep::sw_tidy(),
          ses(books[,'Paperback'], initial = 'simple', alpha = 0.01) %>%
                forecast(h=4) %>% accuracy() %>% sweep::sw_tidy()) %>%
    knitr::kable(format = 'latex',caption = '1st line: Auto-alpha. 2nd line: a=0.01')
```

Repeat but with initial="optimal". How much difference does an optimal initial level make?

Setting the initial to optimal reduces RMSE by ~1 unit, and MASE by ~0.02 units.

Table 2: 1st line: Optimal initial. 2nd line: Simple initial

.rownames	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	7.175981	33.63769	27.84310	0.4736071	15.57784	0.7021303	-0.2117522
Training set	1.749509	34.79175	28.64424	-2.7701566	16.56938	0.7223331	-0.1268119

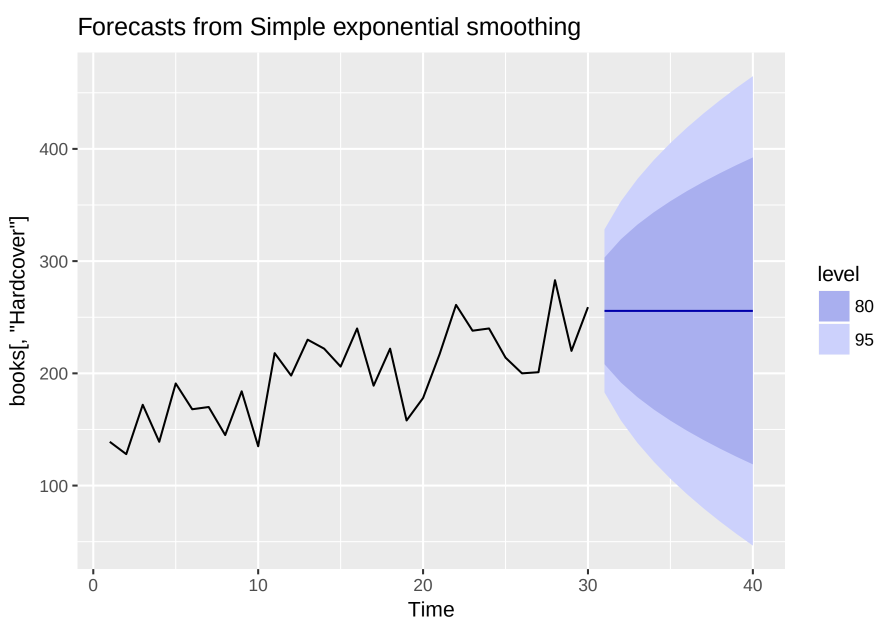
```
ses(books[,'Paperback'], initial = 'optimal', alpha = NULL) -> fit_optimal
ses(books[,'Paperback'], initial = 'simple', alpha = NULL) -> fit_simple
bind_rows(fit_optimal %>% forecast(h=4) %>% accuracy() %>% sweep::sw_tidy(),
          fit_simple %>% forecast(h=4) %>% accuracy() %>% sweep::sw_tidy()) %>%
    knitr::kable(format = 'latex',
                 caption = '1st line: Optimal initial. 2nd line: Simple initial')
```

Repeat steps (b)–(d) with the hardcover series.

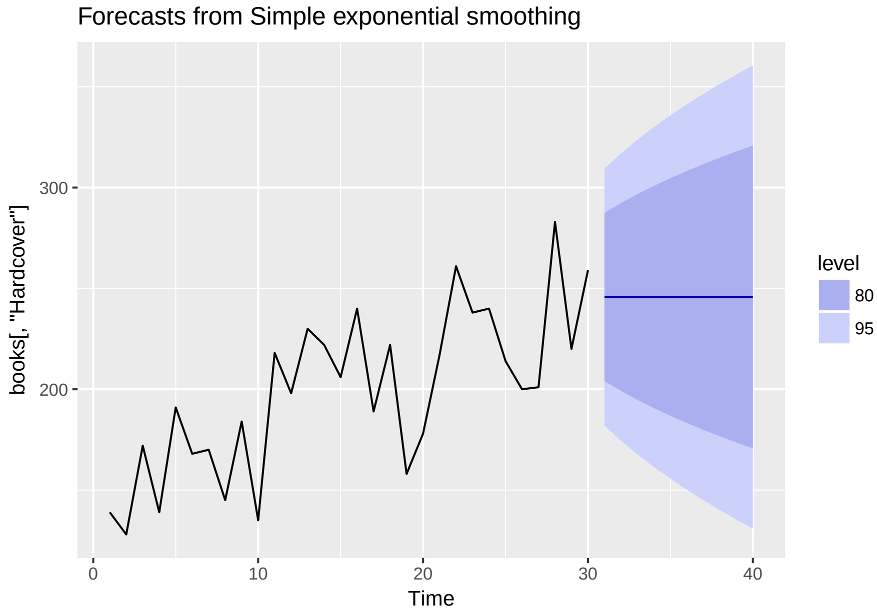
Use simple exponential smoothing with the ses function (setting initial="simple") and explore different values of alpha for the paperback series.

This series does better with a higher value of alpha.

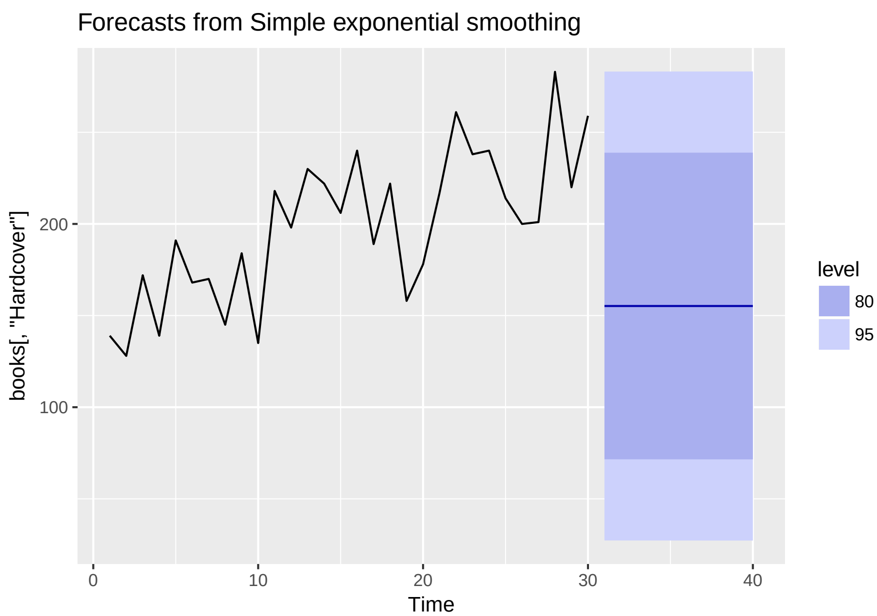
```
ses(books[,'Hardcover'], initial = 'simple', alpha = .9)  %>% forecast() %>% autoplot()
```




```
ses(books[,'Hardcover'], initial = 'simple', alpha = .5)  %>% forecast() %>% autoplot()
```




```
ses(books[,'Hardcover'], initial = 'simple', alpha = .01) %>% forecast() %>% autoplot()
```

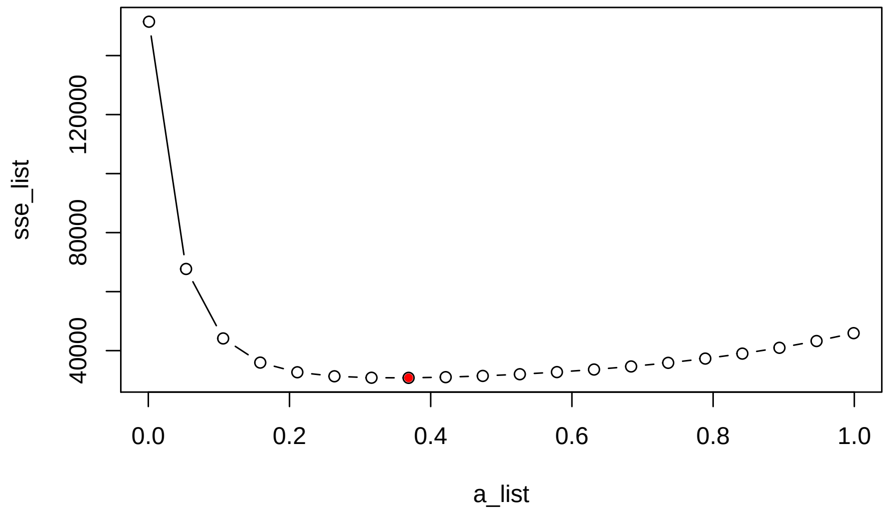


Record the within-sample SSE for the one-step forecasts. Plot SSE against alpha and find which value of alpha works best. What is the effect of alpha on the forecasts?

We can see a typical curve as seen during parameter tuning. The SSE is minimum at an alpha value of about 0.35.

```
sse_list <- c()
a_list <- seq(0.001, 0.999, length.out = 20)
for (a_sel in a_list) {
    ses(books[,'Hardcover'], initial = 'simple', alpha = a_sel)$model$SSE -> sse
    sse_list <- c(sse_list, sse)
```

```
}
plot(a_list, sse_list, type='b')
points(a_list[which.min(sse_list)], sse_list[which.min(sse_list)], col='red', pch=20)
```



Now let ses select the optimal value of alpha. Use this value to generate forecasts for the next four days. Compare your results with 2.

Alpha selected is 0.347. The point estimate for this forecast seem better than previous results. Prediction interval widths are about similar. RMSE is smaller for the auto-alpha selection.

```
ses(books[,'Hardcover'], initial = 'simple', alpha = NULL)$model$par['alpha']
```

```
##     alpha
## 0.3473308
```

```
ses(books[,'Hardcover'], initial = 'simple', alpha = NULL)%>% forecast(h=4)%>% autoplot()
```

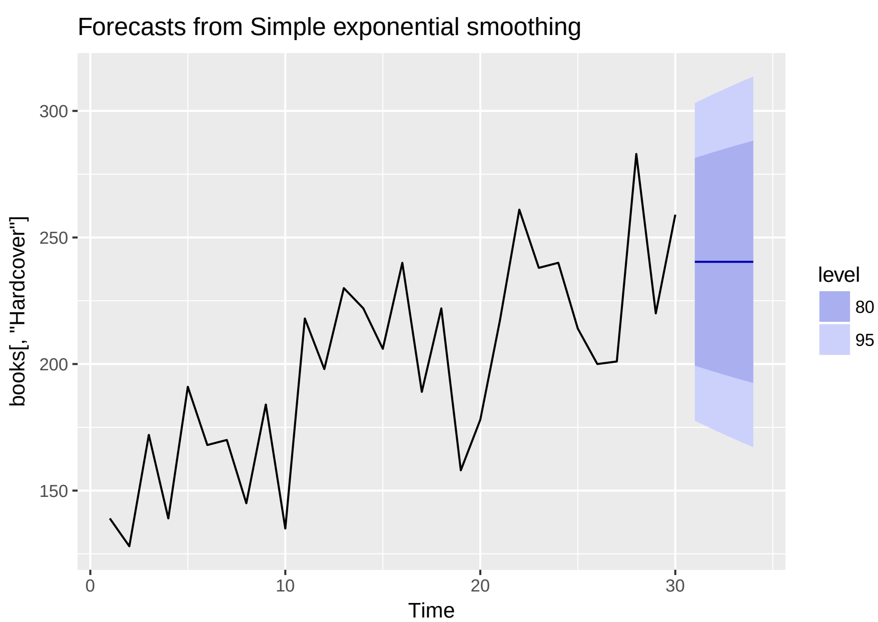


Table 3: 1st line: Auto-alpha. 2nd line: a=0.01

.rownames	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	9.729512	32.01982	26.34467	3.1042066	13.05063	0.7860035	-0.1629044
Training set	4.320299	37.06280	30.11231	0.2818677	15.24951	0.8984127	-0.5483903

Table 4: 1st line: Optimal initial. 2nd line: Simple initial

.rownames	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	9.166735	31.93101	26.77319	2.636189	13.39487	0.7987887	-0.1417763
Training set	9.729512	32.01982	26.34467	3.104207	13.05063	0.7860035	-0.1629044

```
bind_rows(ses(books[,'Hardcover'], initial = 'simple', alpha = NULL) %>%
              forecast(h=4) %>% accuracy() %>% sweep::sw_tidy(),
          ses(books[,'Hardcover'], initial = 'simple', alpha = 0.9) %>%
              forecast(h=4) %>% accuracy() %>% sweep::sw_tidy()) %>%
    knitr::kable(format = 'latex',caption = '1st line: Auto-alpha. 2nd line: a=0.01')
```

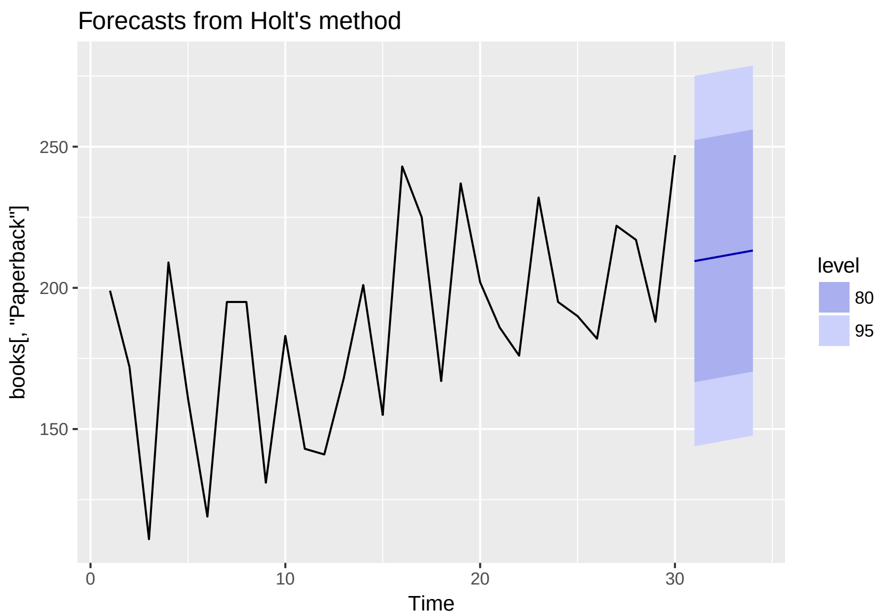
Repeat but with initial="optimal". How much difference does an optimal initial level make?

Setting the initial to optimal reduces RMSE by ~2 units, but MASE increased by ~0.01 units.

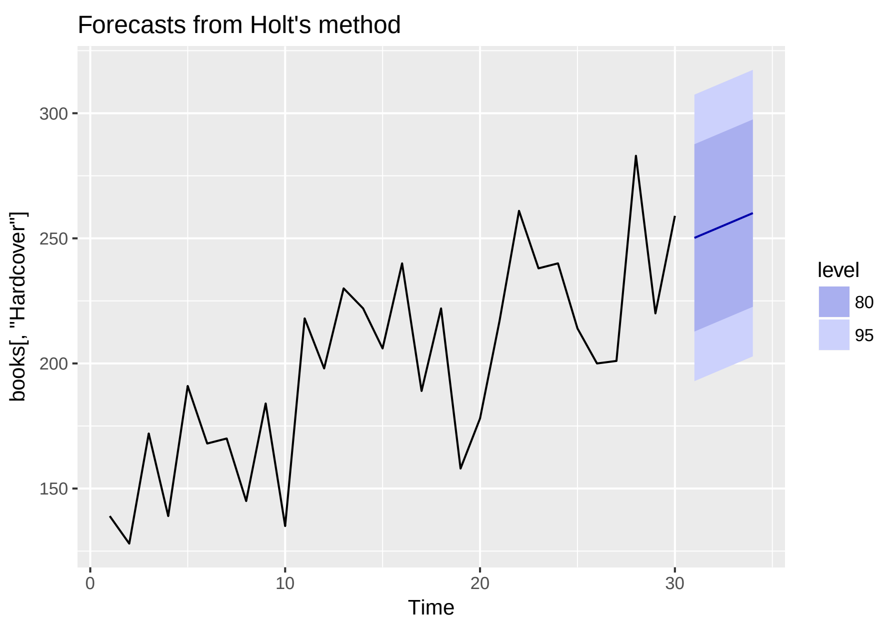
```
ses(books[,'Hardcover'], initial = 'optimal', alpha = NULL) -> fit_optimal
ses(books[,'Hardcover'], initial = 'simple', alpha = NULL) -> fit_simple
bind_rows(fit_optimal %>% forecast(h=4) %>% accuracy() %>% sweep::sw_tidy(),
          fit_simple %>% forecast(h=4) %>% accuracy() %>% sweep::sw_tidy()) %>%
    knitr::kable(format = 'latex',
                 caption = '1st line: Optimal initial. 2nd line: Simple initial')
```

Apply Holt's linear method to the paperback and hardback series and compute four-day forecasts in each case.

```
holt(books[,'Paperback']) %>% forecast(h=4) %>% autoplot()
```




```r
holt(books[,'Hardcover']) %>% forecast(h=4) %>% autoplot()
```



Compare the SSE measures of Holt's method for the two series to those of simple exponential smoothing in the previous question. Discuss the merits of the two forecasting methods for these data sets.

The `holt` method definitely has a lower SSE than the `sse` method since `holt` can account for the upward trend in both the timeseries. `ses` unfortunately, cannot account for this trend.

```r
ses(books[,'Paperback']) %>% residuals() %>% .^2 %>% sum()
```

```
## [1] 33944.82
```

```r
ses(books[,'Hardcover']) %>% residuals() %>% .^2 %>% sum()
```

```
## [1] 30587.69
```

```r
holt(books[,'Paperback']) %>% residuals() %>% .^2 %>% sum()
```

```
## [1] 29085.24
```

```r
holt(books[,'Hardcover']) %>% residuals() %>% .^2 %>% sum()
```

```
## [1] 22184.72
```

Compare the forecasts for the two series using both methods. Which do you think is best?

Again, the forecasts created by `holt` have the right upward trend as would be expected from the

Calculate a 95% prediction interval for the first forecast for each series using both methods, assuming normal errors. Compare your forecasts with those produced by R.

Using the `forecast` function:

```r
holt(books[,'Paperback']) %>% forecast(h=1)
```

```
##    Point Forecast    Lo 80    Hi 80   Lo 95    Hi 95
## 31       209.4668 166.6035 252.3301 143.913 275.0205
```

```r
holt(books[,'Hardcover']) %>% forecast(h=1)
```

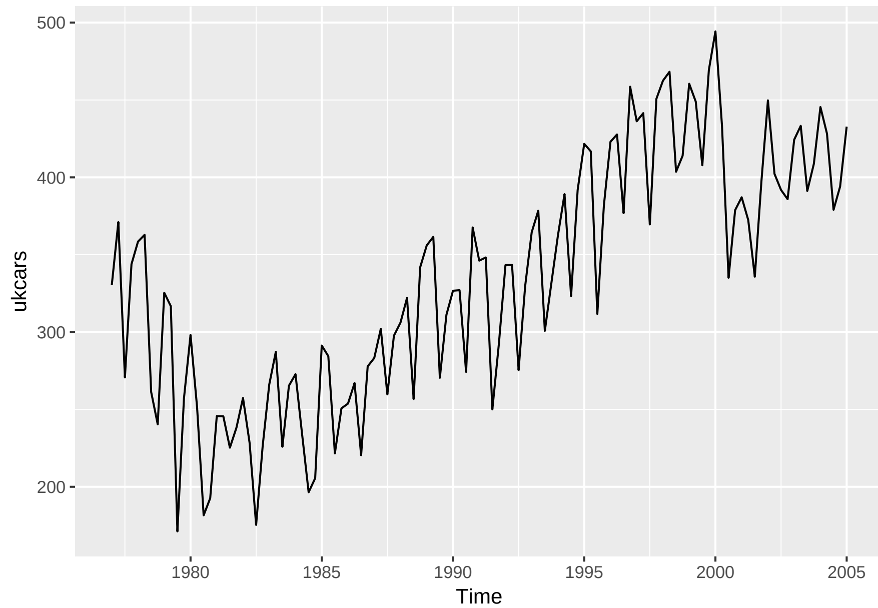
```
##    Point Forecast   Lo 80    Hi 80    Lo 95    Hi 95
## 31       250.1739 212.739 287.6087 192.9222 307.4256
```

For this exercise, use the quarterly UK passenger vehicle production data from 1977:1–2005:1 (data set ukcars).

Plot the data and describe the main features of the series.

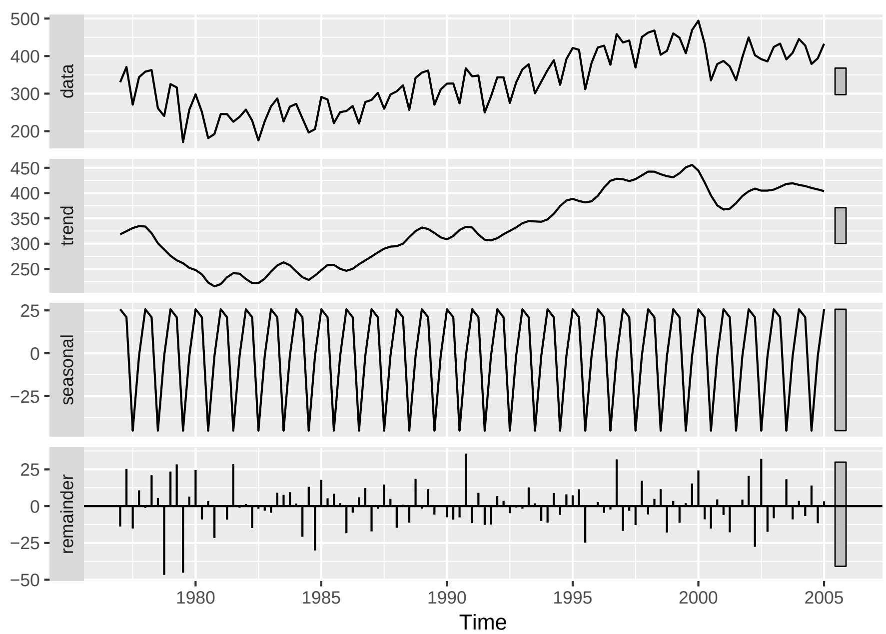
- Between 1977 and 1980 there is a downward trend
- From 1980 to 2000, thre is a steady linear increasing trend
- Something happens in 2000 which causes a sharp decline for a year and picks back up
- There is a quarterly seasonality we can see

```r
autoplot(ukcars)
```



Decompose the series using STL and obtain the seasonally adjusted data.

```
stl_fit <- stl(ukcars, s.window = 'periodic')
autoplot(stl_fit)
```




```
sadjusted <- seasadj(stl_fit)
```

Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of the one-step forecasts from your method.

```r
holtfit <- holt(sadjusted, h = 8, damped = T, exponential = F)
seasonaladjustments <- ukcars-sadjusted
holtfit_w_seasonal <- holtfit$mean + seasonaladjustments[2:9]
holtfit_w_seasonal
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2005          427.4813 361.3417 404.9096
## 2006 432.1322 427.4794 361.3399 404.9080
## 2007 432.1308
```

Parameters of the fit are here. RMSE is 25.15986.

```r
summary(holtfit)
```

```
##
## Forecast method: Damped Holt's method
##
## Model Information:
## Damped Holt's method
##
## Call:
##  holt(y = sadjusted, h = 8, damped = T, exponential = F)
##
##   Smoothing parameters:
##     alpha = 0.5717
##     beta  = 1e-04
##     phi   = 0.9136
##
##   Initial states:
##     l = 346.4959
##     b = -8.9299
##
##   sigma:  25.7357
##
##      AIC     AICc      BIC
## 1275.101 1275.894 1291.466
##
## Error measures:
##                    ME     RMSE      MAE       MPE     MAPE     MASE
## Training set 2.385682 25.15986 20.51592 0.2663159 6.573144 0.668604
##                    ACF1
## Training set 0.03563559
##
## Forecasts:
```

```
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005 Q2       406.4670 373.4854 439.4486 356.0260 456.9080
## 2005 Q3       406.4664 368.4736 444.4593 348.3614 464.5715
## 2005 Q4       406.4659 364.0486 448.8833 341.5942 471.3377
## 2006 Q1       406.4655 360.0424 452.8885 335.4675 477.4634
## 2006 Q2       406.4651 356.3546 456.5755 329.8277 483.1024
## 2006 Q3       406.4647 352.9194 460.0099 324.5743 488.3551
## 2006 Q4       406.4643 349.6911 463.2376 319.6371 493.2915
## 2007 Q1       406.4640 346.6361 466.2919 314.9651 497.9629
```

Forecast the next two years of the series using Holt's linear method applied to the seasonally adjusted data. Then reseasonalize the forecasts. Record the parameters of the method and report the RMSE of of the one-step forecasts from your method.

```
holtfit_additive <- holt(sadjusted, h = 8, damped = F, exponential = F)
seasonaladjustments <- ukcars-sadjusted
holtfit_additive_w_seasonal <- holtfit_additive$mean + seasonaladjustments[2:9]
holtfit_additive_w_seasonal
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2005          428.6470 363.3398 407.7401
## 2006 435.7950 431.9744 366.6671 411.0674
## 2007 439.1224
```

Parameters of the fit are here. RMSE is 25.26.

```
summary(holtfit_additive)
```

```
##
## Forecast method: Holt's method
##
## Model Information:
## Holt's method
##
## Call:
##  holt(y = sadjusted, h = 8, damped = F, exponential = F)
##
##   Smoothing parameters:
##     alpha = 0.6049
##     beta  = 1e-04
##
##   Initial states:
##     l = 334.5744
##     b = 0.8354
##
##   sigma:  25.7197
##
##      AIC     AICc      BIC
```

```
## 1274.003 1274.563 1287.640
##
## Error measures:
##                      ME     RMSE      MAE       MPE     MAPE    MASE
## Training set -0.311188 25.26041 20.10954 -0.638754 6.490918 0.65536
##                    ACF1
## Training set 0.03183994
##
## Forecasts:
##         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005 Q2       407.6327 374.6716 440.5939 357.2230 458.0425
## 2005 Q3       408.4646 369.9401 446.9890 349.5465 467.3826
## 2005 Q4       409.2964 365.9149 452.6779 342.9501 475.6427
## 2006 Q1       410.1282 362.3798 457.8767 337.1033 483.1531
## 2006 Q2       410.9601 359.2107 462.7095 331.8162 490.1039
## 2006 Q3       411.7919 356.3282 467.2556 326.9675 496.6163
## 2006 Q4       412.6237 353.6783 471.5692 322.4744 502.7731
## 2007 Q1       413.4556 351.2217 475.6894 318.2771 508.6340
```

Now use ets() to choose a seasonal model for the data.

ETS selects an A-Ad-A model - Additive error, Additive damped seasonal and Additive trend component.

```
ets(ukcars, model = 'ZZZ',damped = T) %>% summary()
```

```
## ETS(A,Ad,A)
##
## Call:
##  ets(y = ukcars, model = "ZZZ", damped = T)
##
##   Smoothing parameters:
##     alpha = 0.5814
##     beta  = 1e-04
##     gamma = 1e-04
##     phi   = 0.9284
##
##   Initial states:
##     l = 343.6012
##     b = -5.3444
##     s=-1.1652 -45.1153 21.2507 25.0298
##
##   sigma:  26.2512
##
##      AIC     AICc      BIC
## 1283.319 1285.476 1310.593
##
## Training set error measures:
```

```
##                      ME     RMSE      MAE     MPE     MAPE      MASE
## Training set 2.009896 25.18409 20.44382 0.10939 6.683841 0.6662543
##                      ACF1
## Training set 0.03323651
```

Compare the RMSE of the fitted model with the RMSE of the model you obtained using an STL decomposition with Holt's method. Which gives the better in-sample fits?

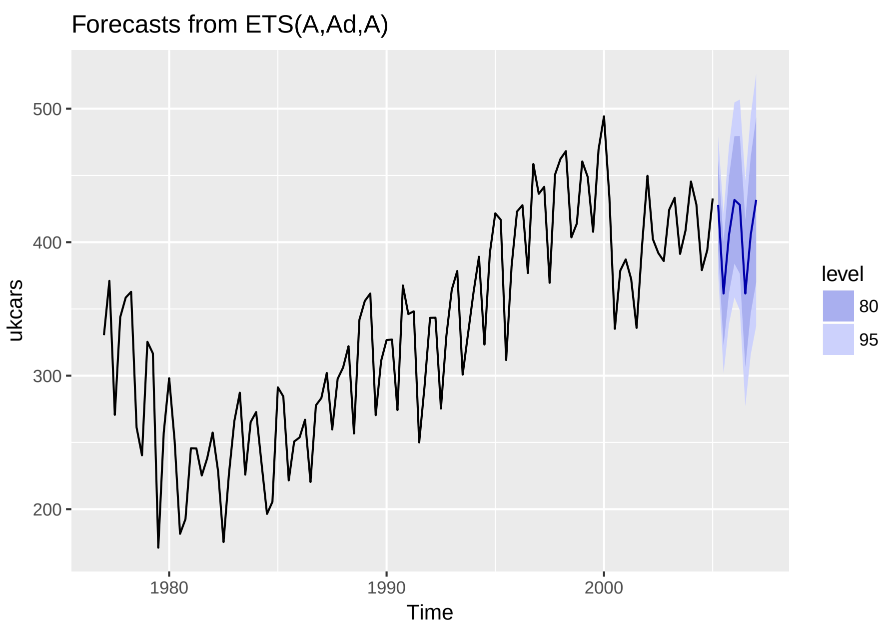
- RMSE of ETS A-Ad-A model = 25.18409
- RMSE of Holt Addive Damped modelo on STL decomposed data = 25.15986

The ETS gives *marginally* worse results.

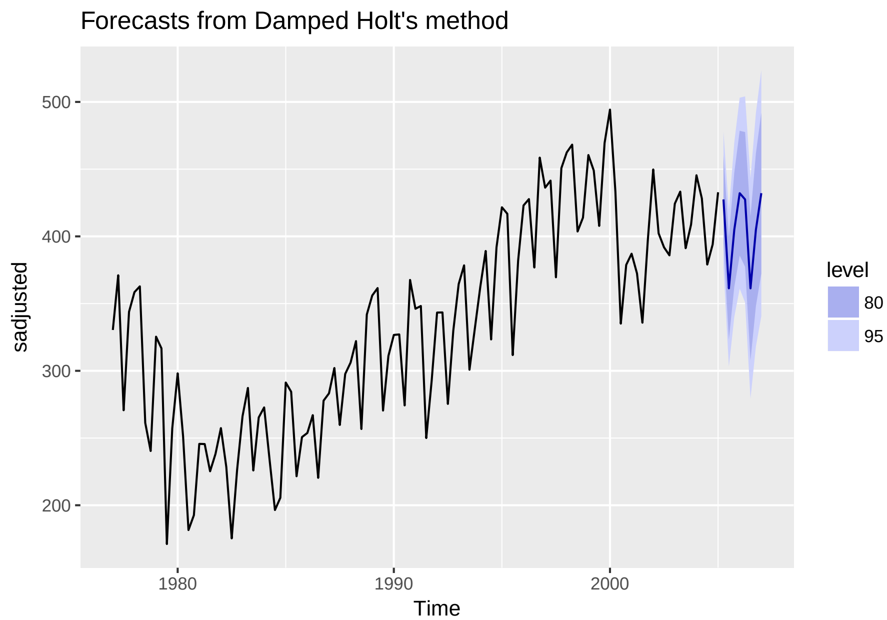
Compare the forecasts from the two approaches? Which seems most reasonable?

Given how close the RMSE values are we expect very similar forecasts. And we can see this in the plots. They are virtually indistinguishable.

```
holtfit$mean  <- holtfit$mean + seasonaladjustments[2:9]
holtfit$upper <- holtfit$upper + seasonaladjustments[2:9]
holtfit$lower <- holtfit$lower + seasonaladjustments[2:9]
holtfit$x  <- holtfit$x + seasonaladjustments
ets(ukcars, model = 'ZZZ',damped = T) %>% forecast() %>% autoplot()
```




```
holtfit %>% autoplot()
```

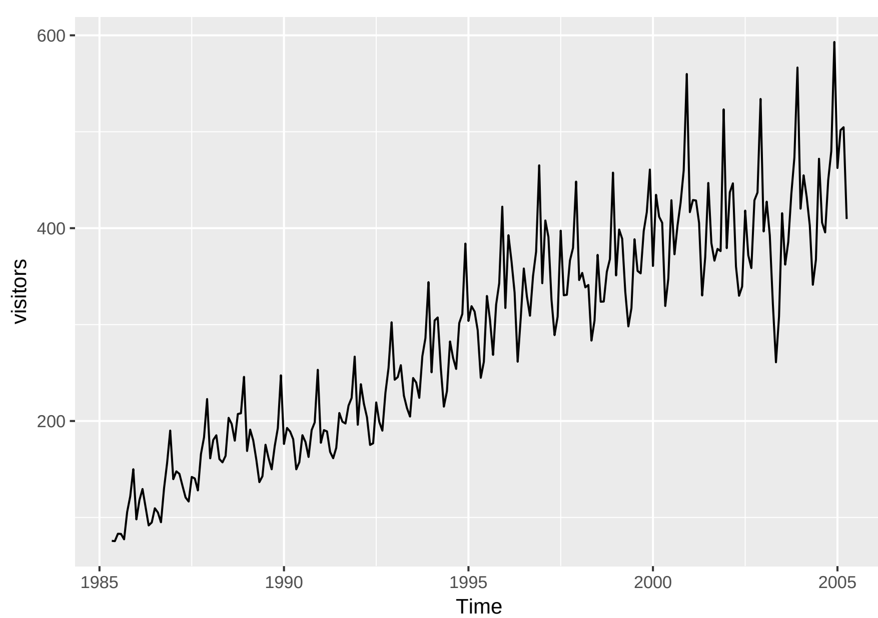


For this exercise, use the monthly Australian short-term overseas visitors data, May 1985–April 2005. (Data set: visitors.)

Make a time plot of your data and describe the main features of the series.

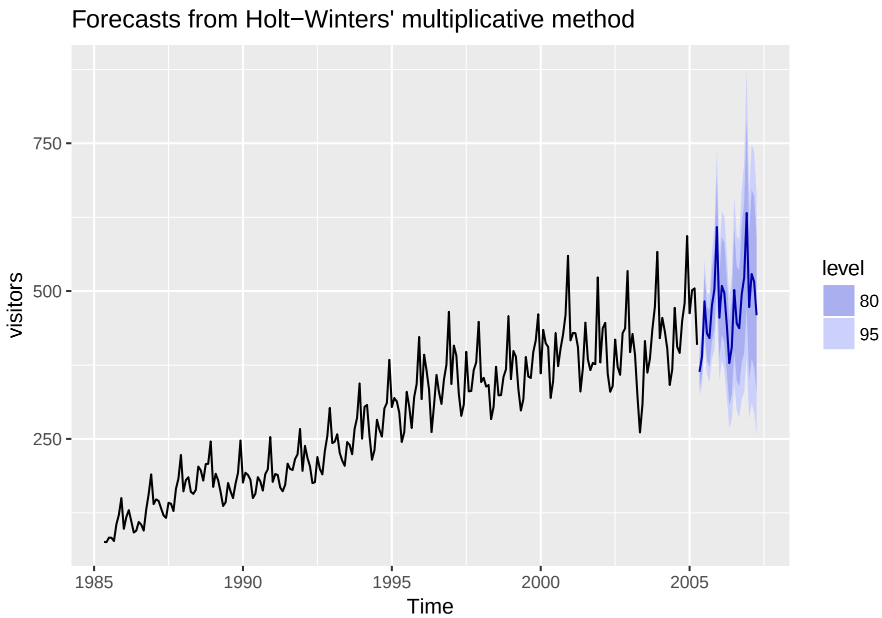
- Increasing trend, almost linearly increasing
- Clear seasonality (yearly) # Increasing variance of the seasonality over time
- Sharp drop mid-2004 - something odd happened here

```
autoplot(visitors)
```



Forecast the next two years using Holt-Winters' multiplicative method.

```
hw(visitors, seasonal = 'm', damped = F) %>%
    forecast(h=12*2) %>% autoplot()
```

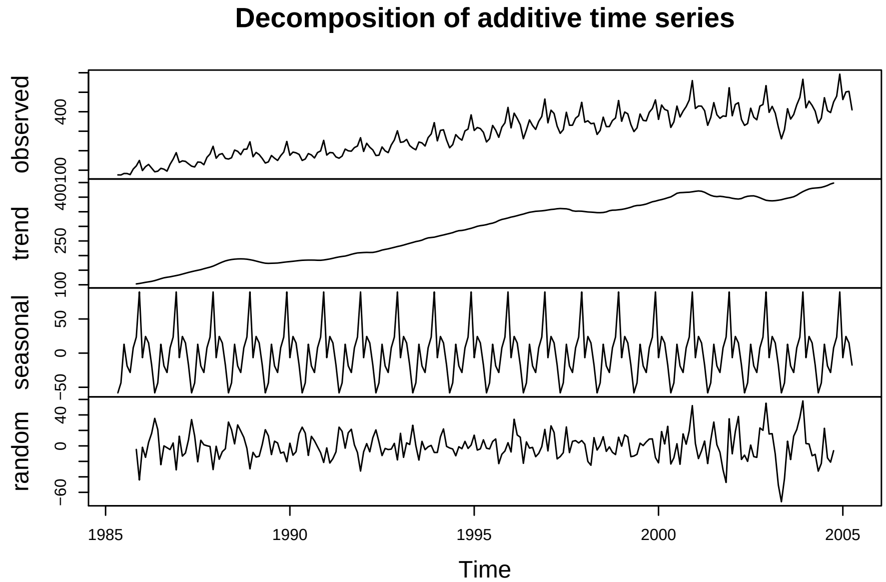


Why is multiplicative seasonality necessary here?

This is because the seasonality keeps increasing over time. We can visually see this if we keep the seasonality constant (using `decompose`). Look at the residuals - the magnitude keeps increasing over time.

Multiplicative seasonality allows for it to increase over time.

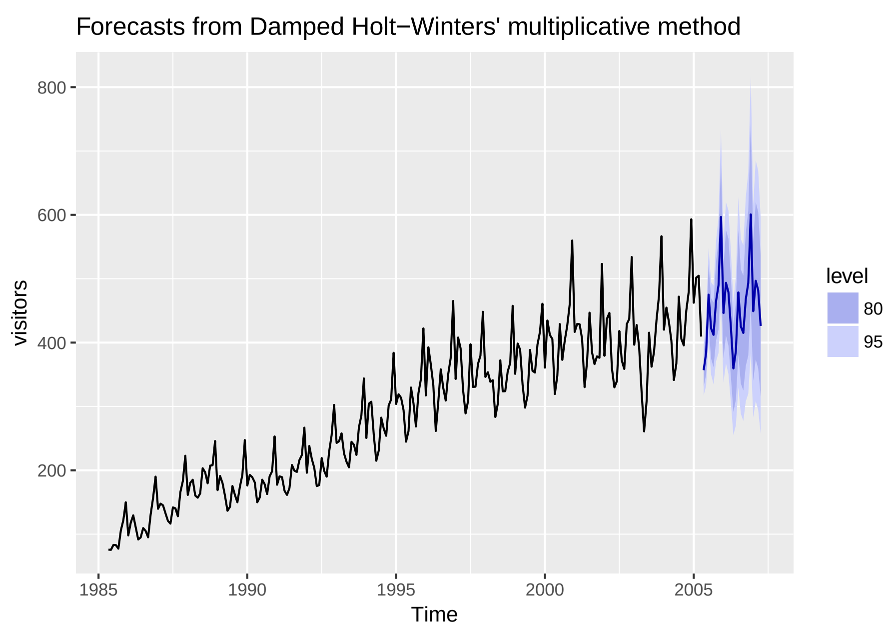
```
decompose(visitors) %>% plot
```



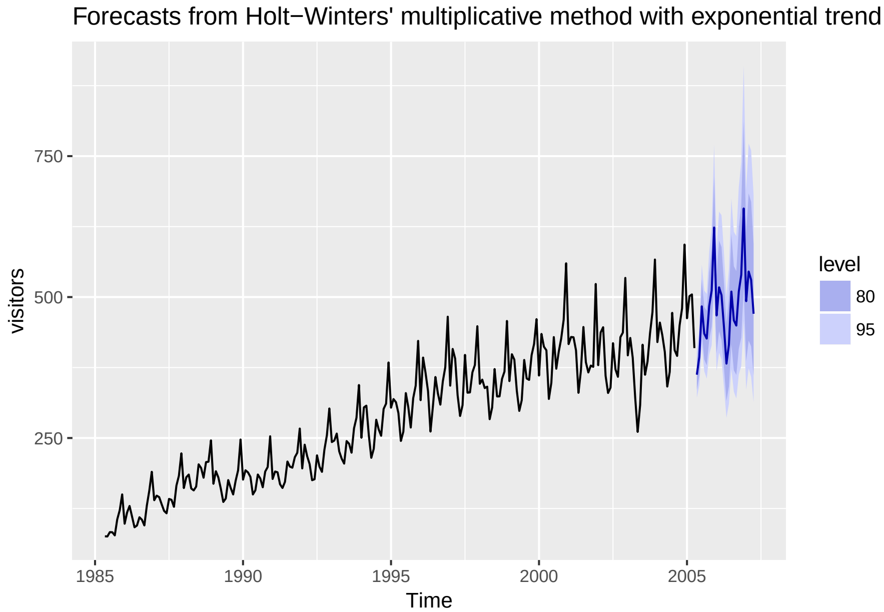
Experiment with making the trend exponential and/or damped.

As expected, the exponential method overpredicts the point estimates than the damped forecast. But, the exponential method seems to have a smaller prediction interval.

```
hw(visitors, seasonal = 'm', damped = T) %>%
    forecast(h=12*2) %>% autoplot()
```




```
hw(visitors, seasonal = 'm', exponential = T) %>%
    forecast(h=12*2) %>% autoplot()
```



Compare the RMSE of the one-step forecasts from the various methods. Which do you prefer?

The RMSEs are fairly similar for all three models. Based on RMSE I would pick the damped Holt Winters model. MASE for this model is the lowest too.

```
bind_rows(
hw(visitors, seasonal = 'm') %>% forecast(h=12*2) %>%
    accuracy() %>% sweep::sw_tidy(),
hw(visitors, seasonal = 'm', damped = T) %>% forecast(h=12*2) %>%
    accuracy() %>% sweep::sw_tidy(),
hw(visitors, seasonal = 'm', exponential = T) %>% forecast(h=12*2) %>%
    accuracy() %>% sweep::sw_tidy()
) %>% mutate(.rownames = c('hw_m','hw_m_d','hw_m_e')) %>%
    rename(model = .rownames)
```

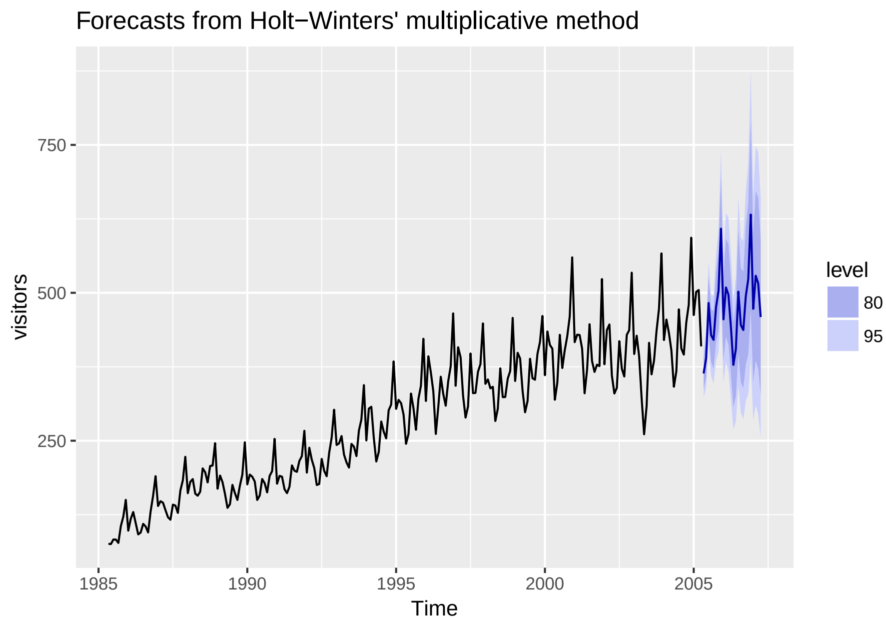
```
## Warning: package 'bindrcpp' was built under R version 3.4.4
```

model	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
hw_m	-0.0949571	14.66220	10.97229	-0.3070136	4.188878	0.4051965	0.0799886
hw_m_d	1.2864553	14.41189	10.67154	0.2674105	4.065573	0.3940899	-0.0207396
hw_m_e	0.0076814	14.62367	10.77736	0.0531410	4.095109	0.3979977	0.0867989

Now fit each of the following models to the same data:

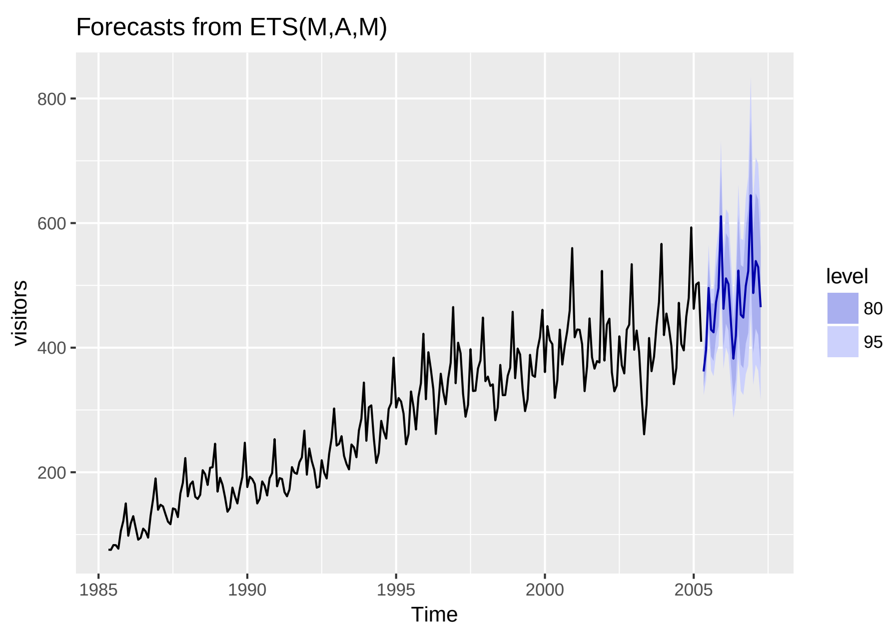
a multiplicative Holt-Winters' method;

```
hw_m <- hw(visitors, seasonal = 'm') %>% forecast(h=12*2)
hw_m %>% autoplot()
```



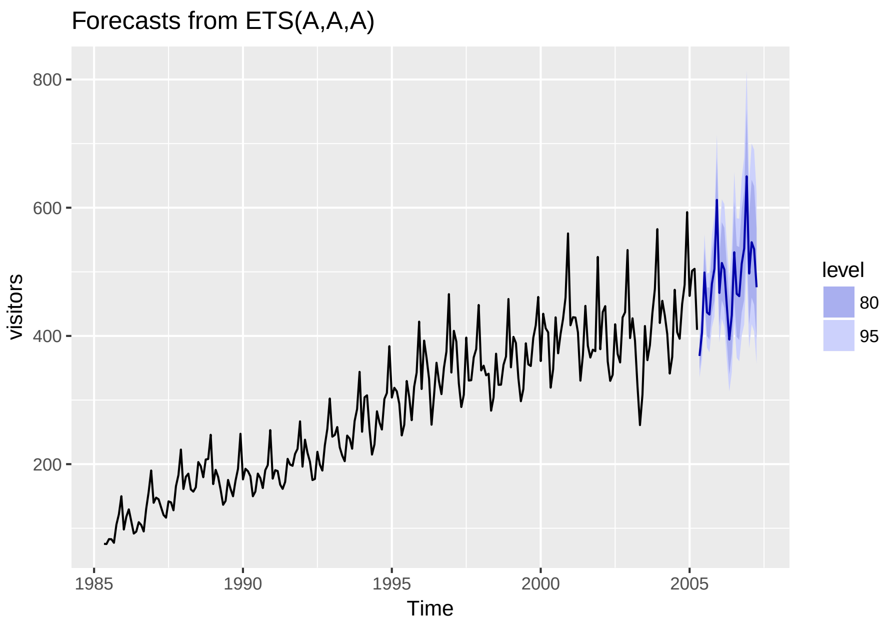
an ETS model;

```
ets_fit <- ets(visitors) %>% forecast(h=12*2)
ets_fit %>% autoplot()
```



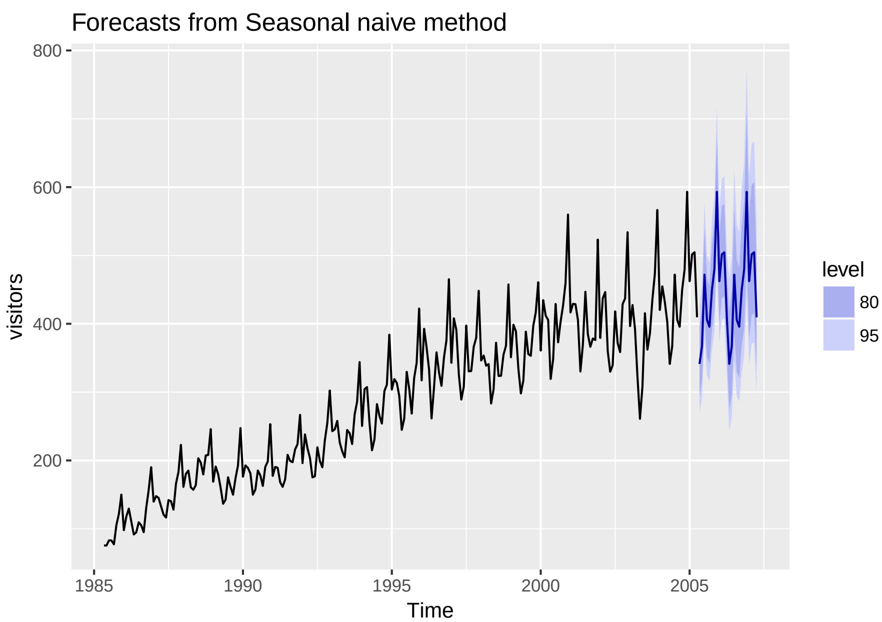
an additive ETS model applied to a Box-Cox transformed series;

```
ets_box_fit <- ets(visitors, lambda = 'auto') %>% forecast(h=12*2)
ets_box_fit %>% autoplot()
```



a seasonal naive method applied to the Box-Cox transformed series;

```
snaive_box_fit <- snaive(visitors, lambda = 'auto') %>% forecast(h=12*2)
snaive_box_fit %>% autoplot()
```



an STL decomposition applied to the Box-Cox transformed data followed by an ETS model applied to the seasonally adjusted (transformed) data.

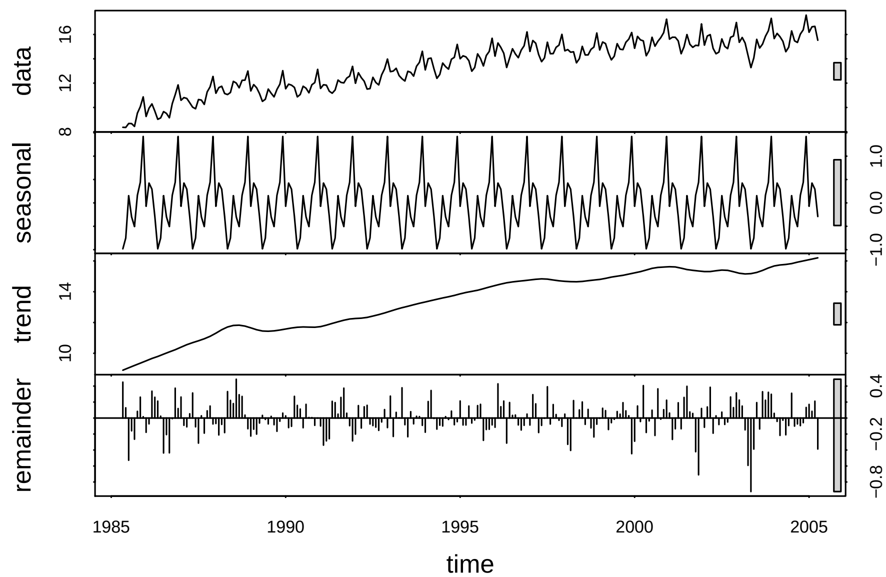
This code performs the needed actions. The `stl` function removes the trend from the signal very well. The residuals look fairly random visually.

After adjusting for seasonality, we can see that an ETS model is an A-Ad-N model: no seasonality with a linear damped trend. Now, although we have an additive trend component, the beta coeef is 1e-4, so practically, the forecast is flat as we can see in the plot.

```
BoxCox.lambda(x = visitors)
```

```
## [1] 0.2775249
```

```
visitors_boxed <- BoxCox(x = visitors, lambda = 0.2775249)
visitors_stl <- stl(visitors_boxed, s.window = 'periodic')
plot(visitors_stl)
```

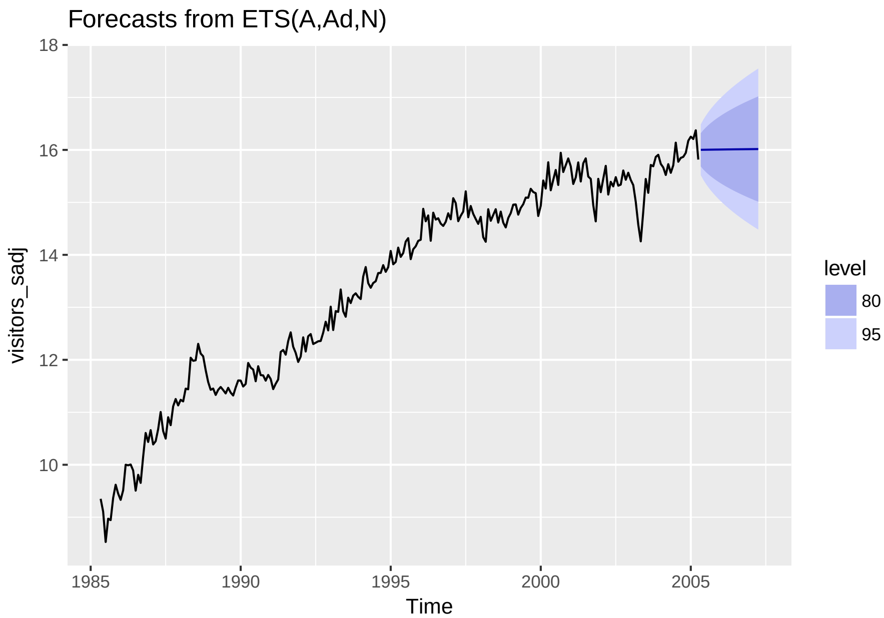



```
visitors_sadj <- seasadj(visitors_stl)
ets_sadj <- ets(visitors_sadj, model = 'ZZZ', damped = T)
ets_sadj %>% summary()
```

```
## ETS(A,Ad,N)
##
## Call:
##  ets(y = visitors_sadj, model = "ZZZ", damped = T)
##
##   Smoothing parameters:
##     alpha = 0.6262
##     beta  = 1e-04
##     phi   = 0.98
##
##   Initial states:
##     l = 9.0722
##     b = 0.0898
##
##   sigma:  0.2471
##
##      AIC     AICc      BIC
## 651.1832 651.5437 672.0670
```

```
##
## Training set error measures:
##                      ME      RMSE       MAE        MPE     MAPE      MASE
## Training set 0.01699032 0.2444638 0.1882544 0.08865111 1.433917 0.3847848
##                    ACF1
## Training set 0.01828507
```

```
ets_sadj %>% forecast() %>% autoplot()
```

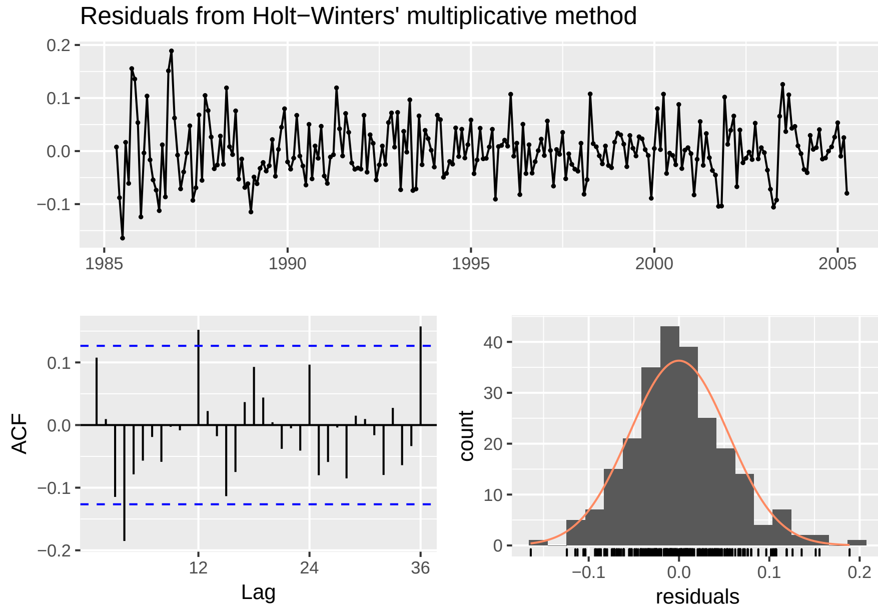


For each model, look at the residual diagnostics and compare the forecasts for the next two years. Which do you prefer?

The residuals tell us this:

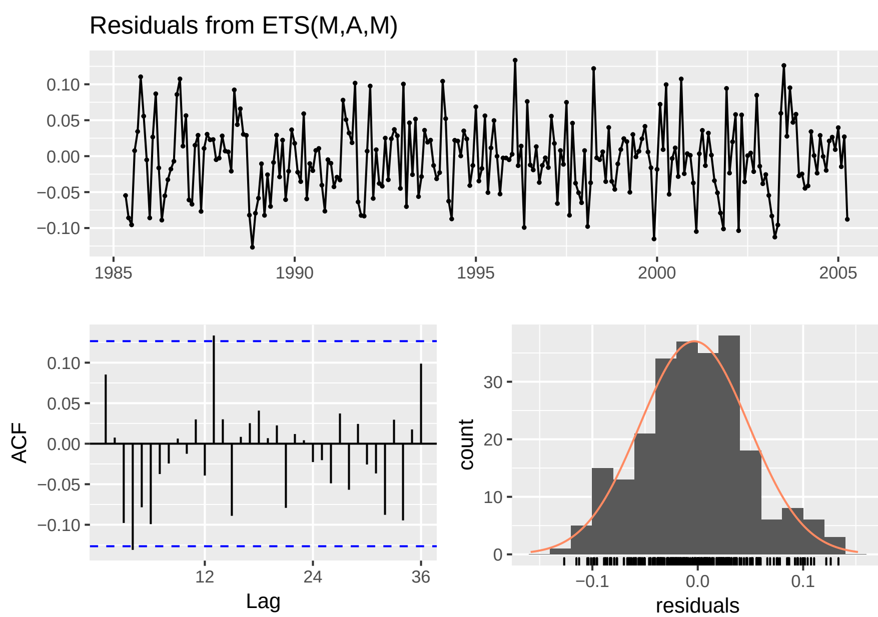
- HW - There is significant autocorrelation in the model (12,24,36 periods). P-value is low too.
- ETS - Almost no autocorrelation. Just a bit at 12 periods. P-value is low, but we do have a large sample size, so the test will be sensitive.
- ETS-BoxCoxed - Almost no autocorrelation.
- SNaive - LOTS of autocorrelation (as expected).
- ETS SAdjusted - No autocorrelation.

```
checkresiduals(hw_m)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Holt-Winters' multiplicative method
## Q* = 35.143, df = 8, p-value = 2.518e-05
##
## Model df: 16.   Total lags used: 24
```

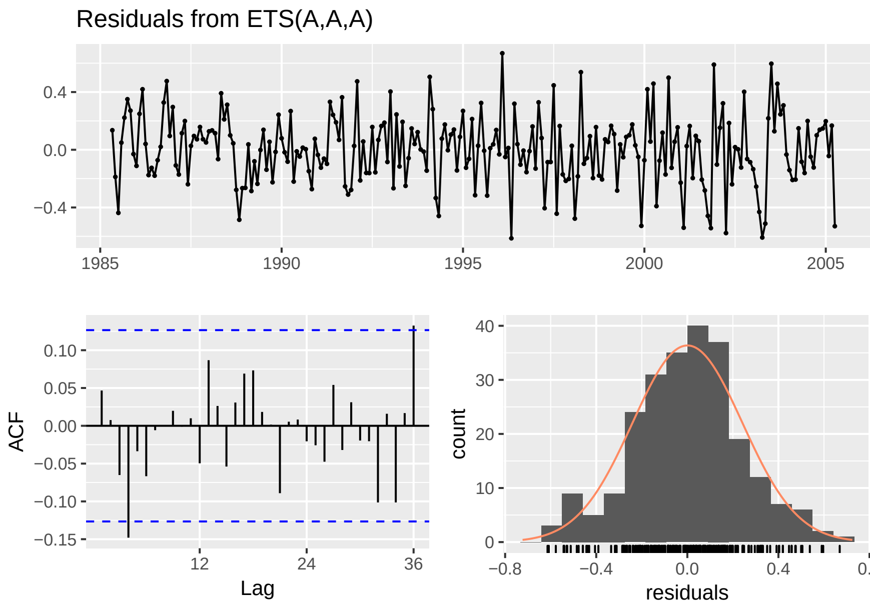
```r
checkresiduals(ets_fit)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(M,A,M)
```

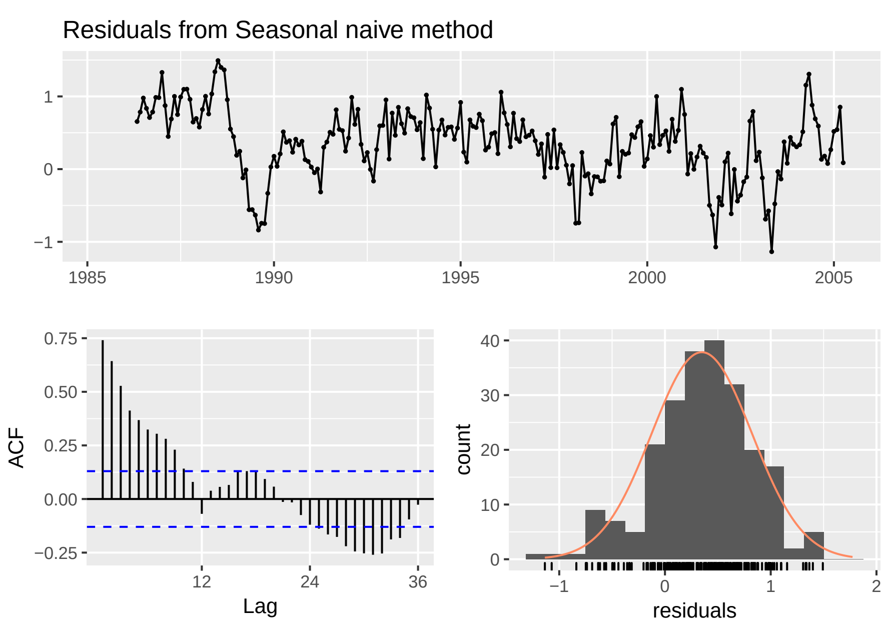
```
## Q* = 22.938, df = 8, p-value = 0.003444
##
## Model df: 16.   Total lags used: 24
```

```r
checkresiduals(ets_box_fit)
```



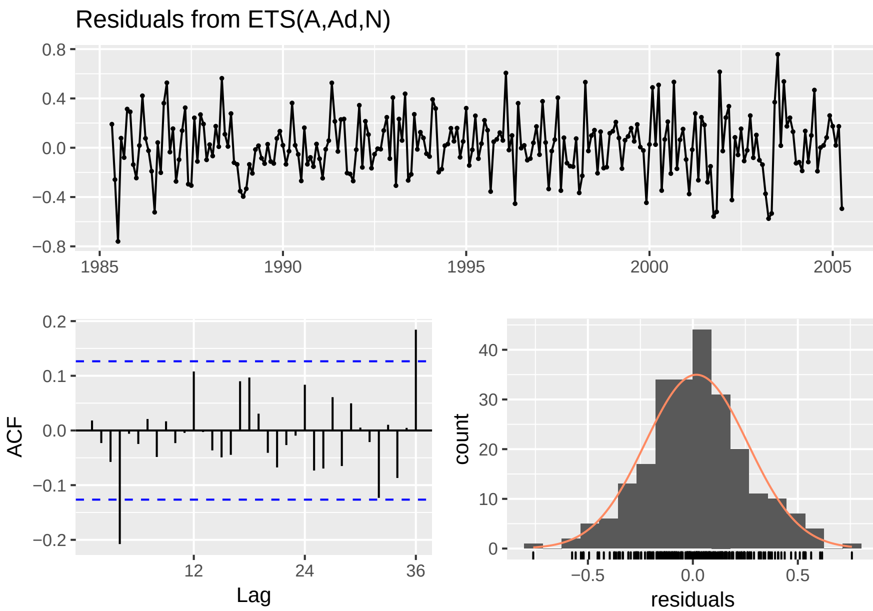

```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,A,A)
## Q* = 17.189, df = 8, p-value = 0.0282
##
## Model df: 16.   Total lags used: 24
```

```r
checkresiduals(snaive_box_fit)
```



```
##
##  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
```

```
checkresiduals(ets_sadj)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ETS(A,Ad,N)
## Q* = 25.716, df = 19, p-value = 0.1383
##
## Model df: 5.   Total lags used: 24
```

Putting everything together, we can see that:

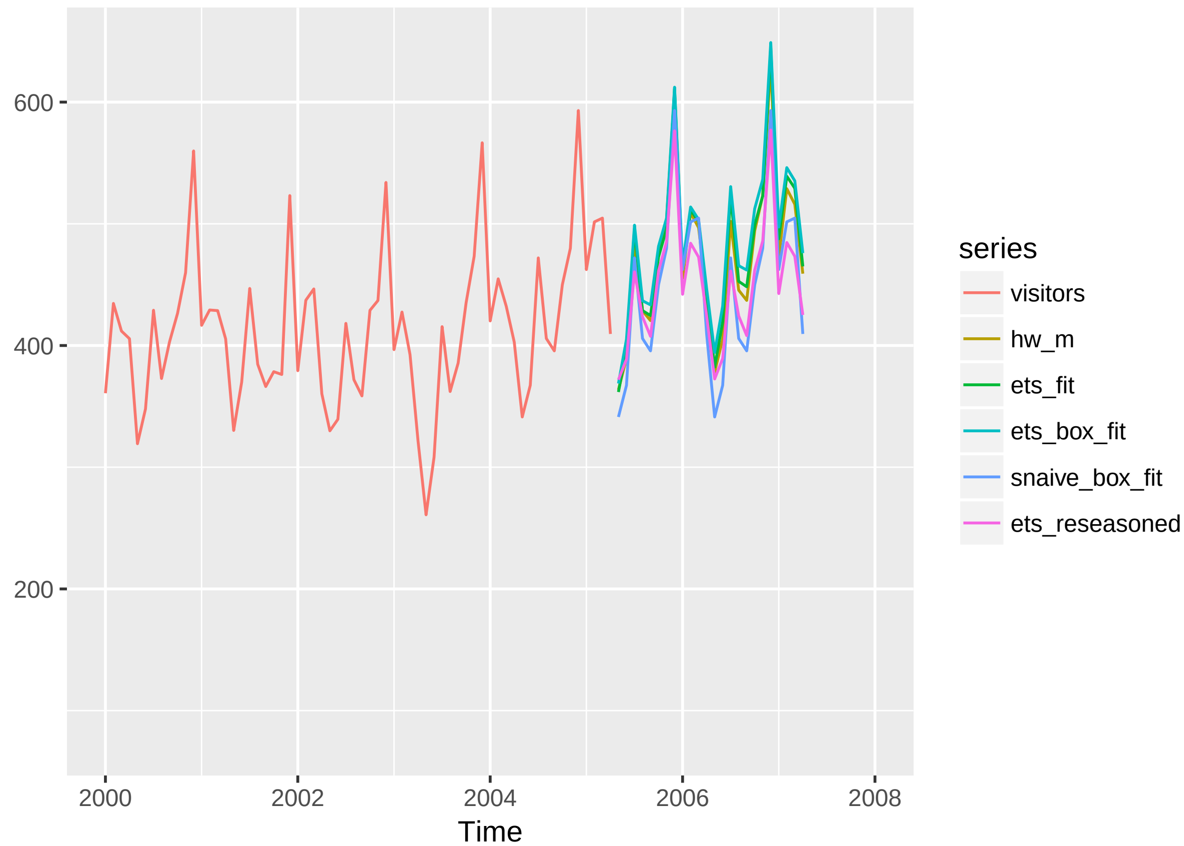
- Seasonal naive underpredicts since it can't account for the increasing trend component
- The BoxCox+STL+ETS method also underpredicts
- HW seems reasonable
- ETS seems reasonable as well
- ETS-BoxCoxed seems to over predict

Given this information and the residuals, I would probably pick the ETS model. It has almost no autocorrelation and seems to fit the data the best.

```
hw_m = hw_m %>% forecast(h = 12*2) %>% sweep::sw_tidy() %>%
    pull('Point Forecast') %>% ts(frequency = 12, start=c(2005,5))
ets_fit = ets_fit %>% forecast(h = 12*2) %>% sweep::sw_tidy() %>%
    pull('Point Forecast') %>% ts(frequency = 12, start=c(2005,5))
```

```r
ets_box_fit = ets_box_fit %>% forecast(h = 12*2) %>% sweep::sw_tidy() %>%
    pull('Point Forecast') %>% ts(frequency = 12, start=c(2005,5))
snaive_box_fit = snaive_box_fit %>% forecast(h = 12*2) %>% sweep::sw_tidy() %>%
    pull('Point Forecast') %>% ts(frequency = 12, start=c(2005,5))
ets_reseasoned = ets_sadj %>% forecast(h = 12*2) %>% sweep::sw_tidy() %>%
    pull('Point Forecast') %>% ts(frequency = 12, start=c(2005,5))
ets_reseasoned = (ets_reseasoned + visitors_stl$time.series %>%
                      tail(24) %>% .[,'seasonal'] %>%
                      as.numeric()) %>% InvBoxCox(lambda = 0.2775249)

ts.union(visitors, hw_m, ets_fit, ets_box_fit, snaive_box_fit, ets_reseasoned) %>%
    autoplot(size=2) + xlim(c(2000,2008))
```



Section 8.11

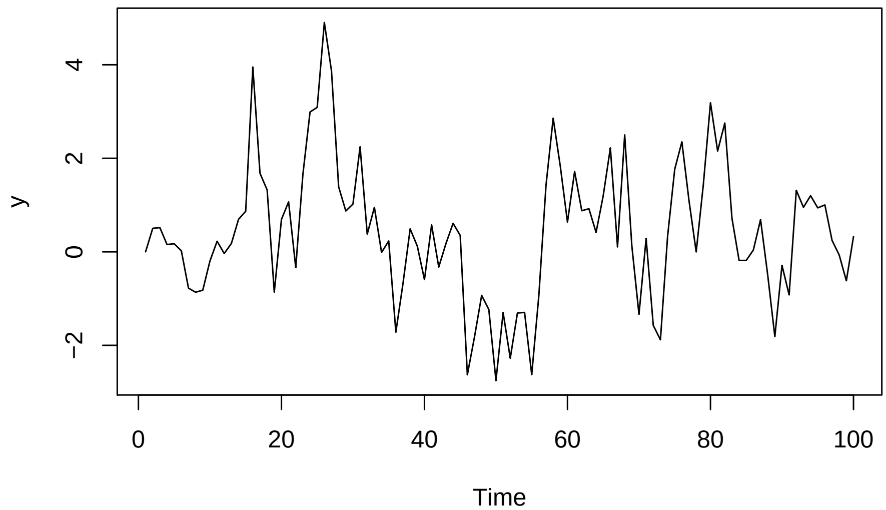
Use R to simulate and plot some data from simple ARIMA models.

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 y0=0.

```r
y <- ts(data = numeric(100))
e <- rnorm(n = 100, mean = 0, sd = 1)
for(i in 2:100)
  y[i] <- 0.6*y[i-1] + e[i]
```

Produce a time plot for the series. How does the plot change as you change phi_1?

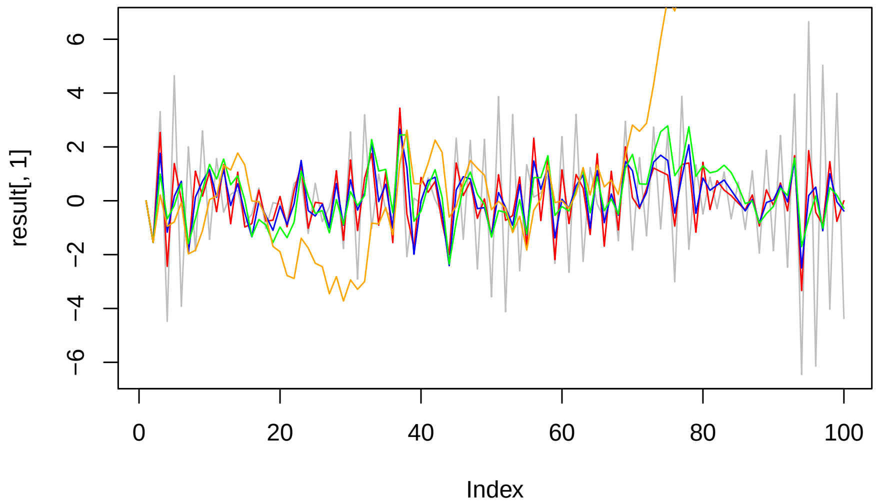
```r
plot(y)
```




```r
phi1 <- seq(-1, 1, by = 0.5)
phi1
```

```
## [1] -1.0 -0.5  0.0  0.5  1.0
```

```r
y <- ts(data = numeric(100))
e <- rnorm(n = 100, mean = 0, sd = 1)
result <- matrix(nrow = 100,ncol = length(phi1))
for(p in seq_along(phi1)){
    for(i in 2:100)
      y[i] <- phi1[p]*y[i-1] + e[i]
    result[,p] <- y
}
plot (result[,1], type='l', col='gray')
lines(result[,2], type='l', col='red')
lines(result[,3], type='l', col='blue')
lines(result[,4], type='l', col='green')
lines(result[,5], type='l', col='orange')
```



Over the range of values of phi_1 from -1 to 1:

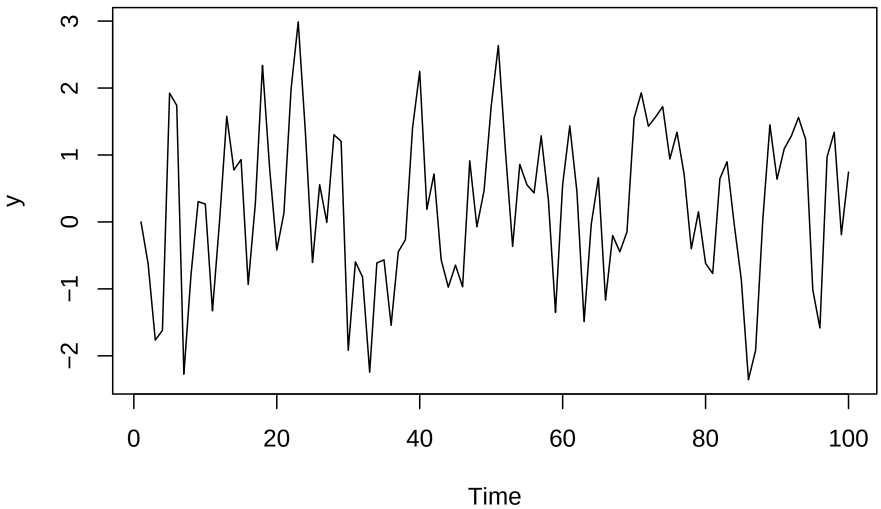
- when c = 0, phi < 0 – we can see that the trend occilates between +ve and -ve. (gray and red lines)
- when c = 0, phi = 0 – white noise
- when c = 0, phi > 0 – random walk (no drift) (green and orange lines)

Write your own code to generate data from an MA(1) model with theta_1=0.6 and sigma_2=1.

```
y <- ts(data = numeric(100))
e <- rnorm(n = 100, mean = 0, sd = 1)
for (i in 2:100){
    y[i] <- 0.6*e[i-1] + e[i]
}
```

Produce a time plot for the series. How does the plot change as you change theta_1?

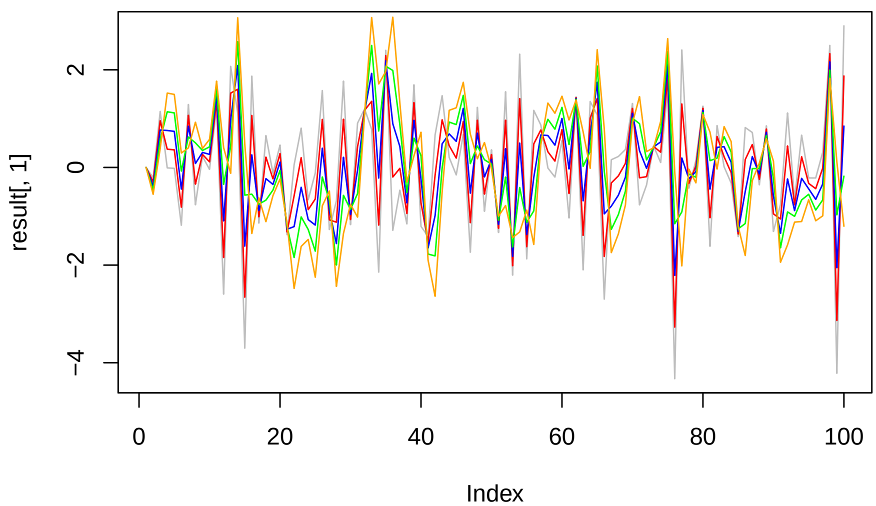
```
plot(y)
```



```r
theta1 <- seq(-1, 1, by = 0.5)
theta1
```

```
## [1] -1.0 -0.5  0.0  0.5  1.0
```

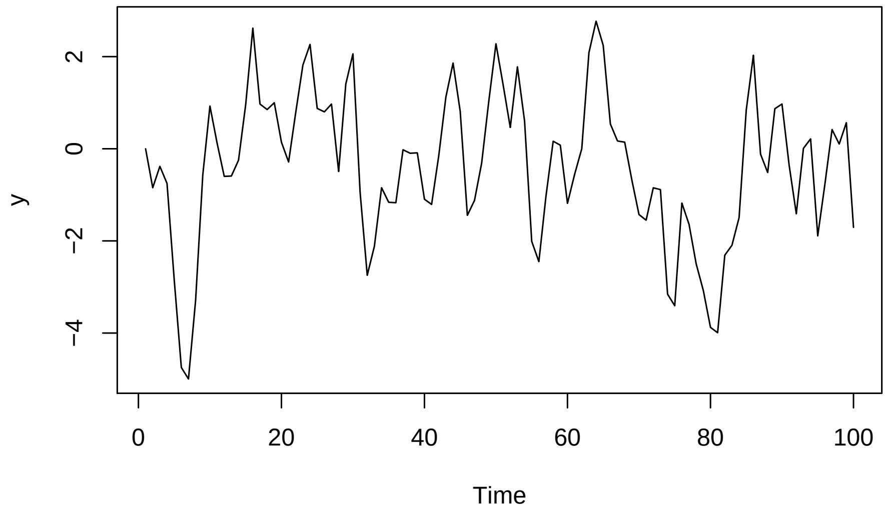
```r
y <- ts(data = numeric(100))
e <- rnorm(n = 100, mean = 0, sd = 1)
result <- matrix(nrow = 100,ncol = length(theta1))
for(tht in seq_along(theta1)){
    for(i in 2:100)
      y[i] <- theta1[tht]*e[i-1] + e[i]
    result[,tht] <- y
}
plot (result[,1], type='l', col='gray')
lines(result[,2], type='l', col='red')
lines(result[,3], type='l', col='blue')
lines(result[,4], type='l', col='green')
lines(result[,5], type='l', col='orange')
```



There doesn't seem to any concrete relationship between theta and the result.

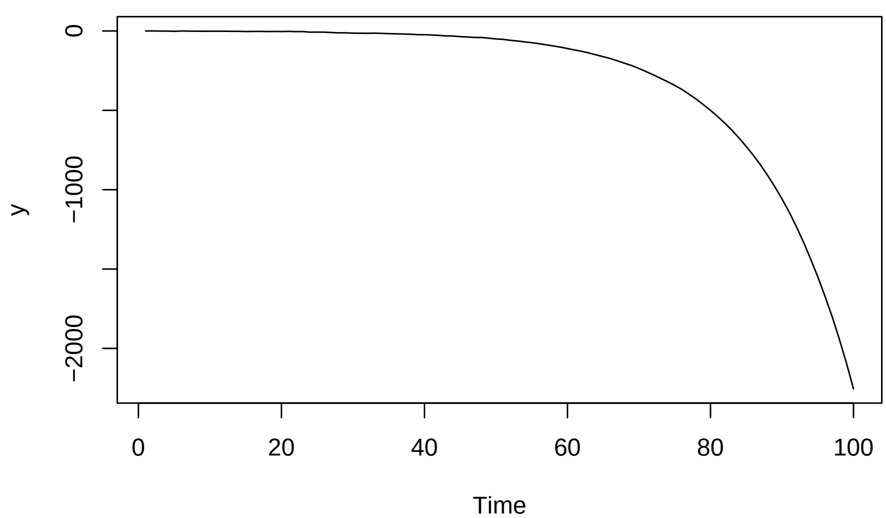
Generate data from an ARMA(1,1) model with phi_1 = 0.6 and theta_1=0.6 and sigma_2=1.

```
y <- ts(numeric(100))
e <- rnorm(100)
for(i in 2:100)
    y[i] <- 0.6 * y[i - 1] + 0.6 * e[i - 1] + e[i]
plot(y)
```



Generate data from an AR(2) model with phi_1=-0.8 and phi_2=0.3 and sigma_2=1. (Note that these parameters will give a non-stationary series.)

```
y <- ts(numeric(100))
e <- rnorm(100)
for(i in 3:100)
    y[i] <- 0.8 * y[i - 1] + 0.3 * y[i - 2] + e[i]
plot(y)
```



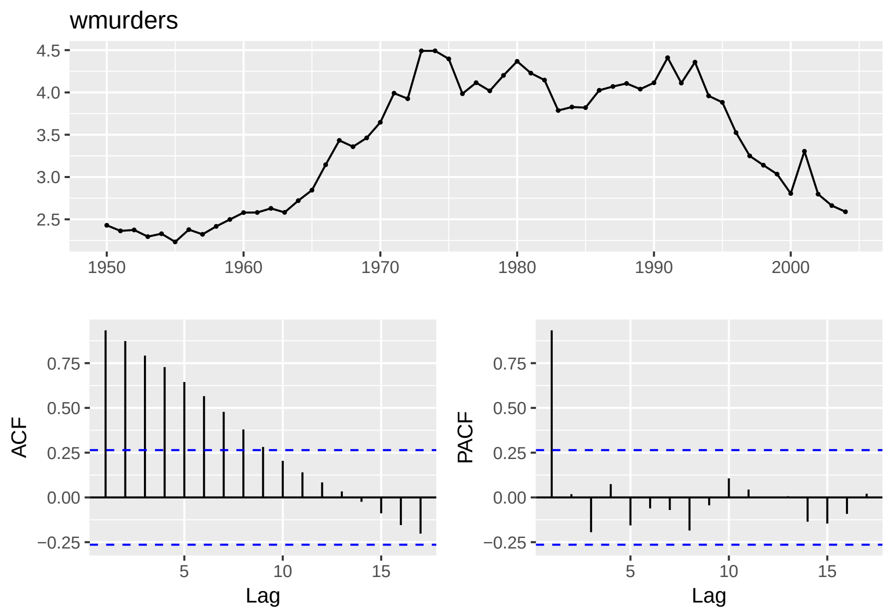
Graph the latter two series and compare them.

Graphed above. While the ARMA(1,1) gives a stationary series, the AR(2) is clearly non-stationary.

Consider the number of women murdered each year (per 100,000 standard population) in the United States (data set wmurders).

By studying appropriate graphs of the series in R, find an appropriate ARIMA(p,d,q) model for these data.

```r
ggtsdisplay(wmurders)
```



- The time series has trending behaviour. It doesn't seem to have any seasonal behavior.

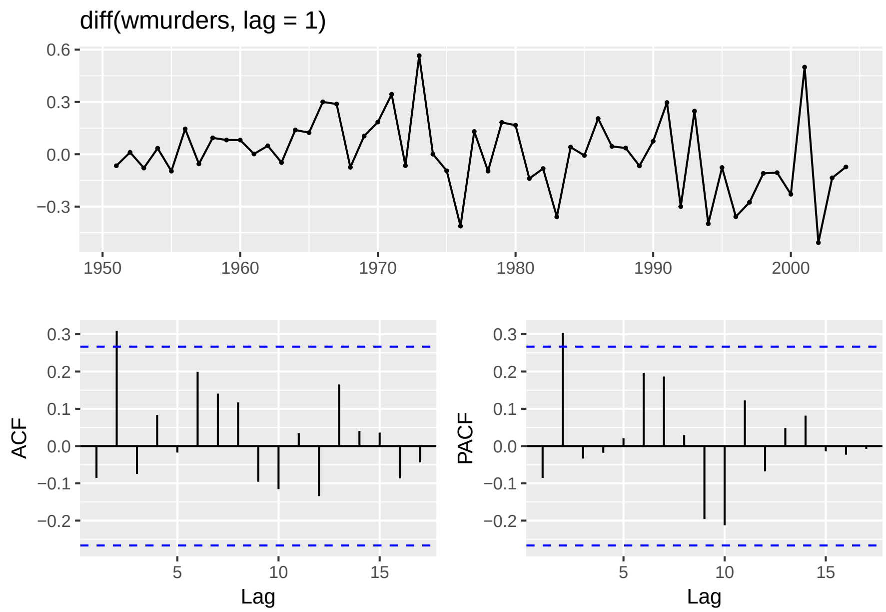
```r
kpss.test(wmurders)
```

```
## Warning in kpss.test(wmurders): p-value smaller than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  wmurders
## KPSS Level = 1.2017, Truncation lag parameter = 1, p-value = 0.01
```

The KPSS test shows us that we must reject the H0: data are stationary. Differencing the time series by 1:

```r
ggtsdisplay(diff(wmurders,lag = 1))
```



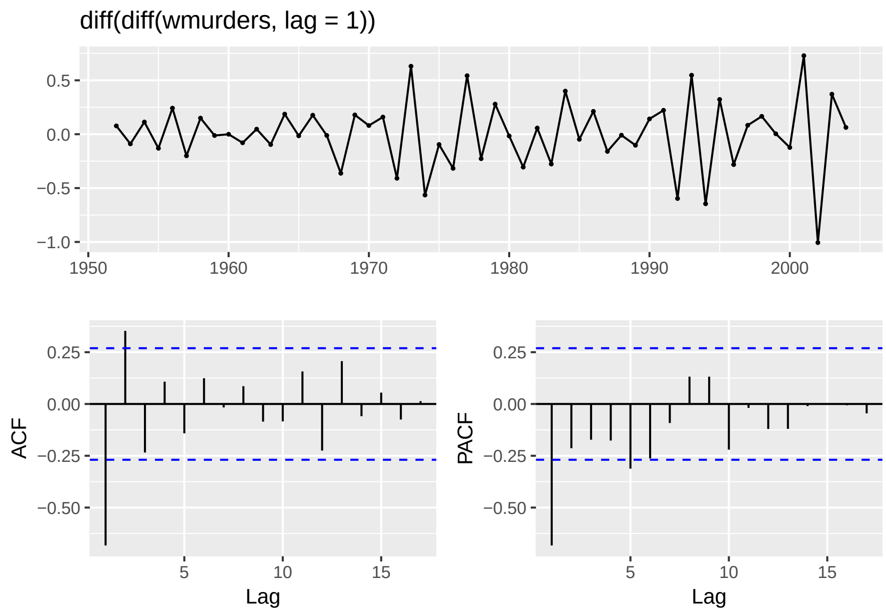
- After differencing, the time series seems to be fairly stationary
- ACF shows us a significant 2nd order component. The PACF also shows a 2nd order significant component.

```
kpss.test(diff(wmurders,1))
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  diff(wmurders, 1)
## KPSS Level = 0.58729, Truncation lag parameter = 1, p-value =
## 0.02379
```

Trying one more differencing scheme. Double differencing - Differencing the difference by 1:

```
ggtsdisplay(diff(diff(wmurders,lag = 1)))
```



- PACF reduces exponentially.
- ACF has a significant order 2, and no significant orders after that.

Thus, I can guess that an ARIMA(0, 2, 2) might fit the data well.

Should you include a constant in the model? Explain.

Yes, I think we should include a constant. With a constant value included, with a d=1 or d=2, a long term forecast can follow a line or quadratic curve. The time series seems to be downward trending, so a constant will help. If c = 0, with d = 0 or d = 1, we won't get a downward trend. (With c = 0 and d = 2, we can still get a linear trend).

Write this model in terms of the backshift operator.

```
(1 - B)^2 y_t = c + (1 + theta1 x B + theta2 x B^2) x e_t
```

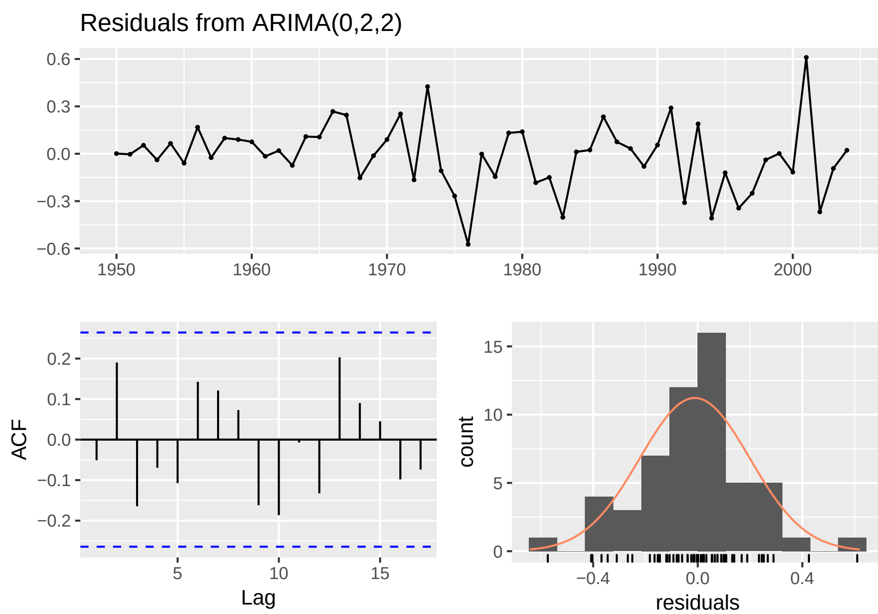
Fit the model using R and examine the residuals. Is the model satisfactory?

```r
fit1 <- Arima(wmurders, order = c(0, 2, 2))
summary(fit1)
```

```
## Series: wmurders
## ARIMA(0,2,2)
##
## Coefficients:
##           ma1     ma2
##       -1.0181  0.1470
## s.e.   0.1220  0.1156
##
```

```
## sigma^2 estimated as 0.04702:  log likelihood=6.03
## AIC=-6.06   AICc=-5.57   BIC=-0.15
##
## Training set error measures:
##                       ME      RMSE       MAE        MPE     MAPE      MASE
## Training set -0.0113461 0.2088162 0.1525773 -0.2403396 4.331729 0.9382785
##                     ACF1
## Training set -0.05094066
```

```
checkresiduals(fit1)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(0,2,2)
## Q* = 11.764, df = 8, p-value = 0.1621
##
## Model df: 2.   Total lags used: 10
```

The Ljung-Box test shows a p-val > 0.05, so there is little chance of autocorrelation in the residuals. Visually looking at the ACF plot, we can see some there is periodic elements in the residuals. The model seems adequate.

Forecast three times ahead. Check your forecasts by hand to make sure you know how they have been calculated.

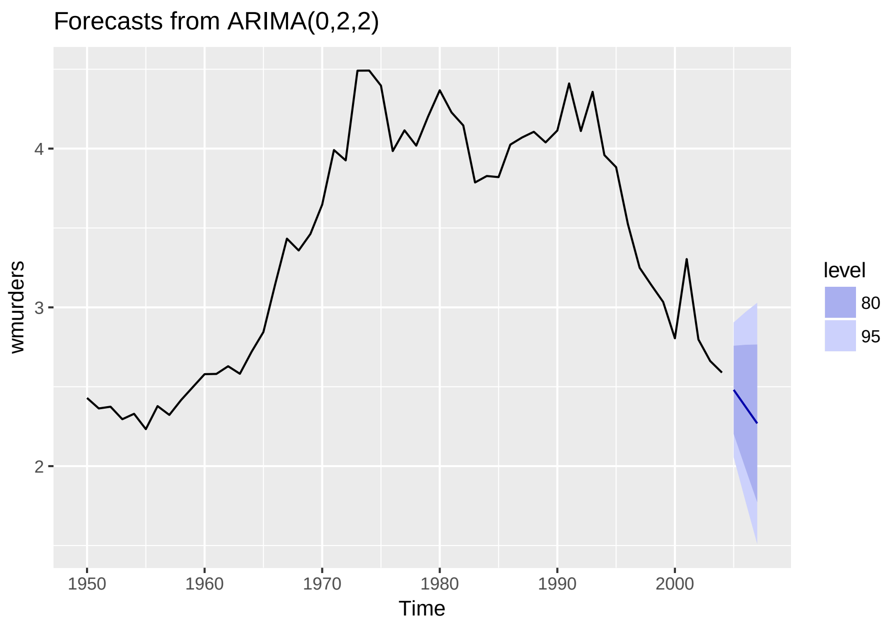
```
fit1 %>% forecast(h = 3)
```

```
##      Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2005       2.480525 2.202620 2.758430 2.055506 2.905544
## 2006       2.374890 1.985422 2.764359 1.779250 2.970531
```

```
## 2007       2.269256 1.772305 2.766206 1.509235 3.029276
```

Create a plot of the series with forecasts and prediction intervals for the next three periods shown.

```
fit1 %>% forecast(h = 3) %>% autoplot()
```



Does auto.arima give the same model you have chosen? If not, which model do you think is better?

If we allow `stepwise`:

```
auto.arima(wmurders, seasonal = F, allowdrift = T)
```

```
## Series: wmurders
## ARIMA(1,2,1)
##
## Coefficients:
##          ar1      ma1
##      -0.2434  -0.8261
## s.e.  0.1553   0.1143
##
## sigma^2 estimated as 0.04632:  log likelihood=6.44
## AIC=-6.88   AICc=-6.39   BIC=-0.97
```

If we force a thorough investigation:

```
auto.arima(wmurders, seasonal = T, stepwise = F, allowdrift = T)
```

```
## Series: wmurders
```

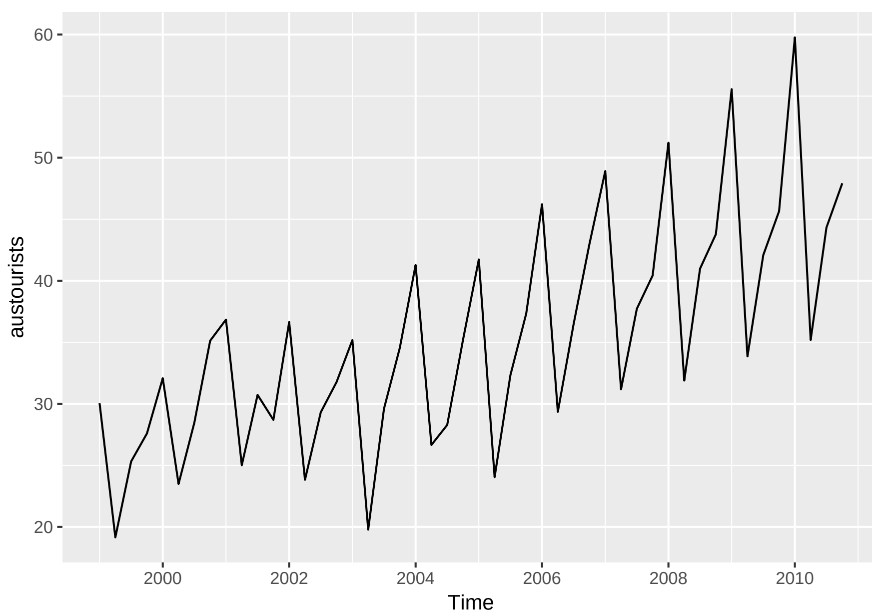
```
## ARIMA(0,2,3)
##
## Coefficients:
##           ma1     ma2      ma3
##       -1.0154  0.4324  -0.3217
## s.e.   0.1282  0.2278   0.1737
##
## sigma^2 estimated as 0.04475:  log likelihood=7.77
## AIC=-7.54   AICc=-6.7   BIC=0.35
```

The ARIMA(0,2,3) has the lowest AICc value (-6.7) against my selection of ARIMA(0,2,2) with an AICc of -5.7.

Consider the quarterly number of international tourists to Australia for the period 1999–2010. (Data set austourists.)

Describe the time plot.

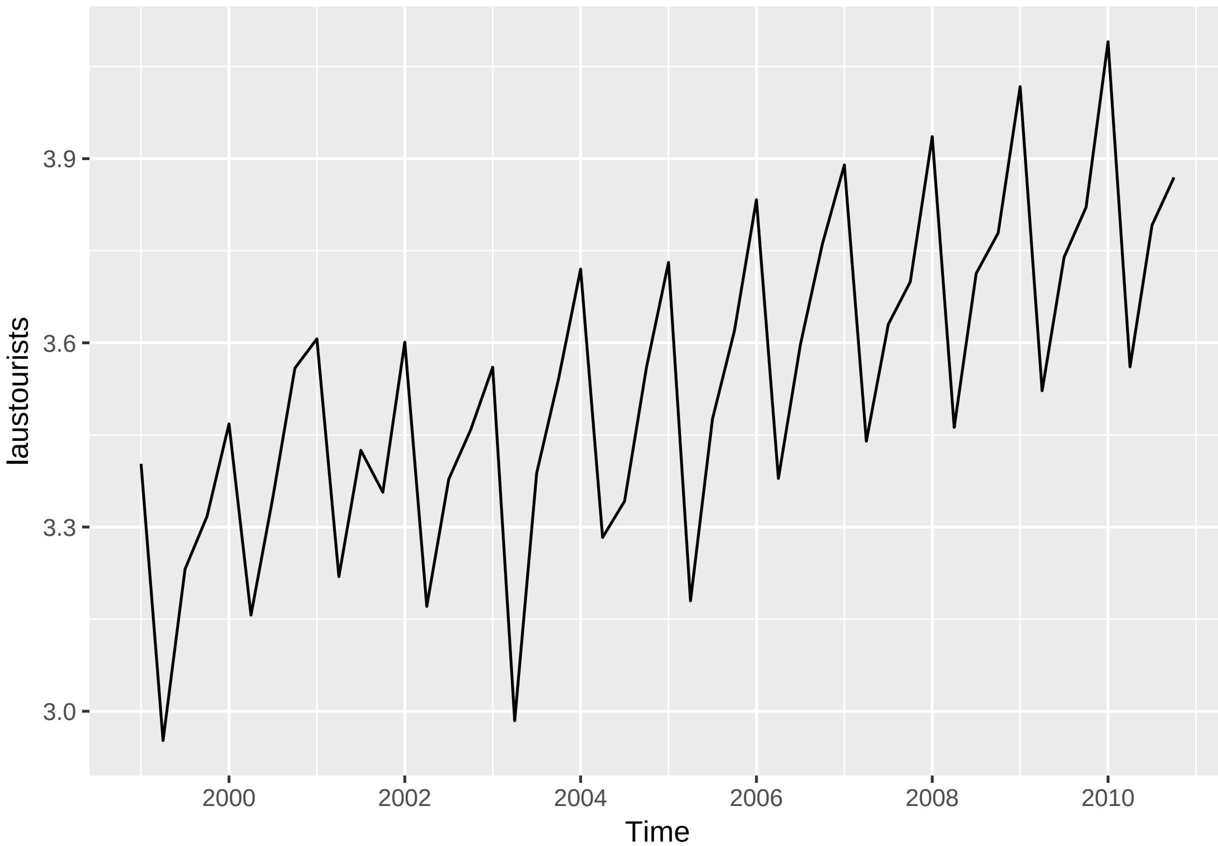
```r
autoplot(austourists)
```



- Trend component exists
- Seasonality exists
- Seasonality increases over time

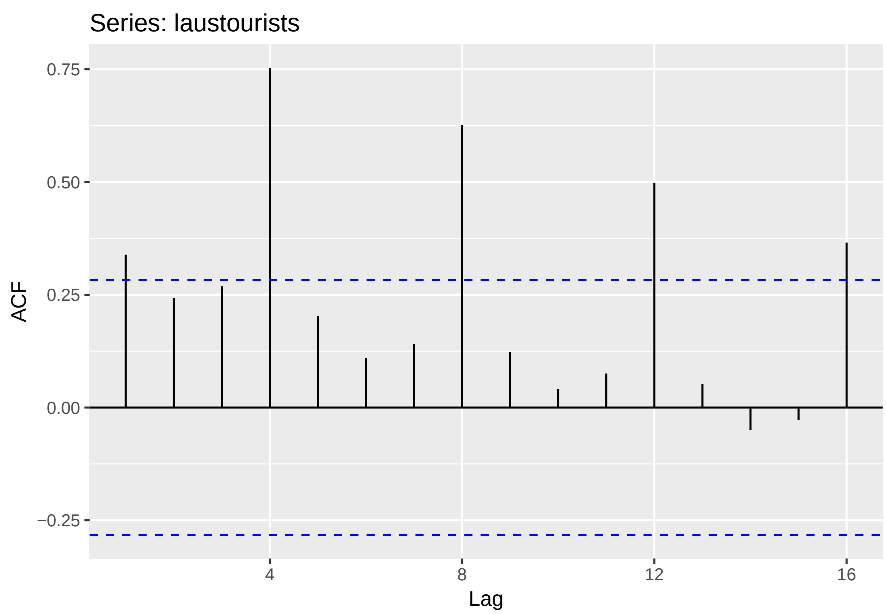
I'm going to log transform the data

```r
laustourists <- log(austourists)
autoplot(laustourists)
```



What can you learn from the ACF graph?

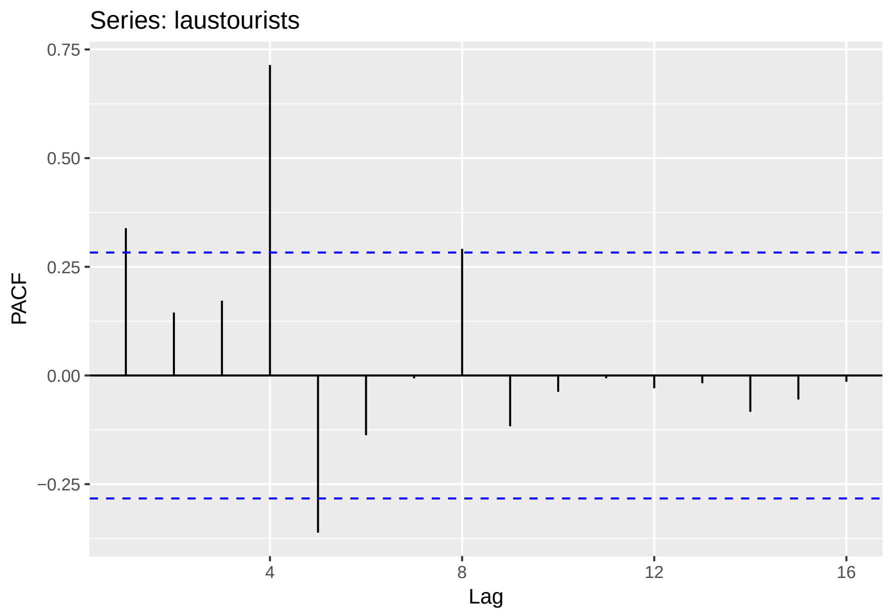
```
ggAcf(laustourists)
```



Very strong seasonality - 4, 8, 12, 16 seen in the components. Reducing strength of the contributions as expected.

What can you learn from the PACF graph?

```r
ggPacf(laustourists)
```

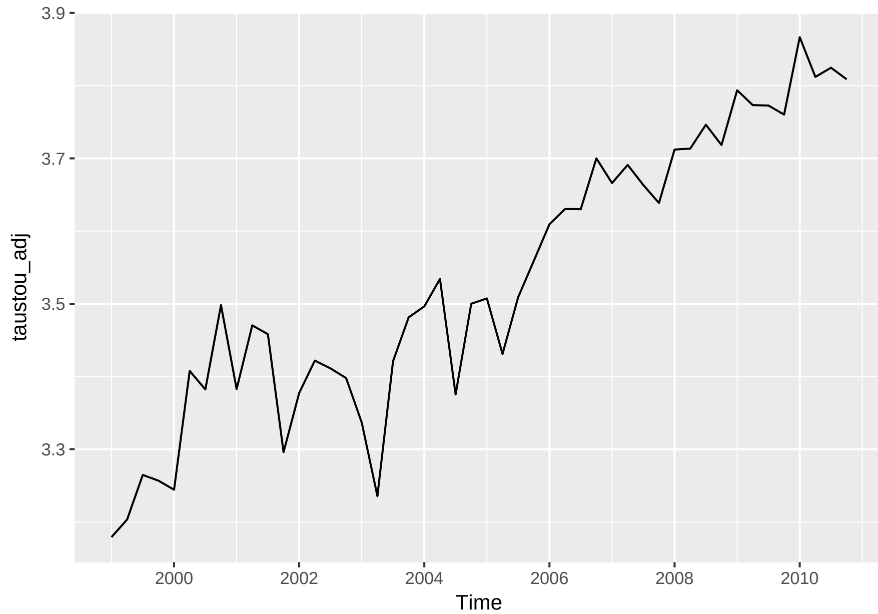


Strong seasonal components at lag=4. Perhaps a strong non-seasonal component at lag=5.

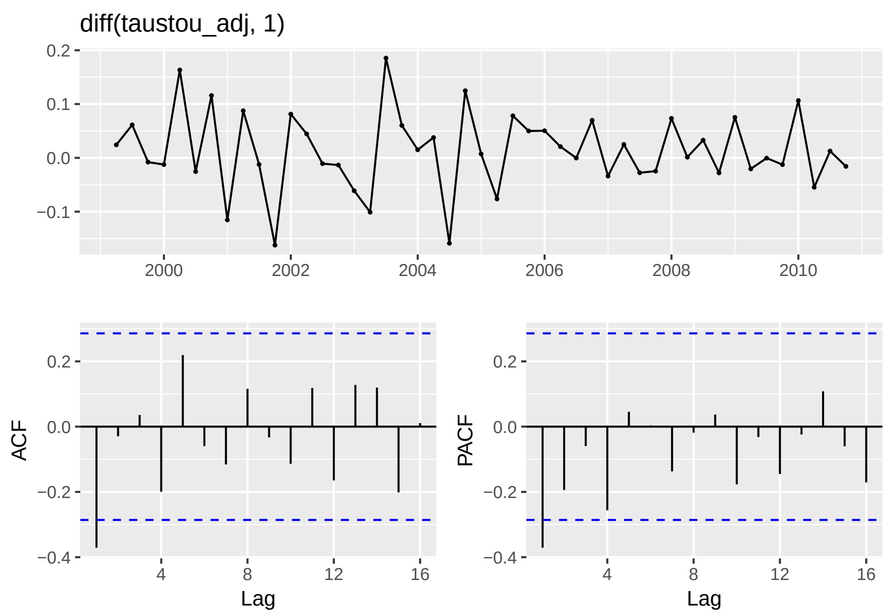
Produce plots of the seasonally differenced data (1-B^4)Yt. What model do these graphs suggest?

After adjusting for seasonality, and differencing the data once to make it stationary, the ACF and PACF plots tell me a significant lag of 1 exists. It's hard for me to pick of p=1 or q=1 based on this plot alone.

```r
taustou_adj <- seasadj(stl(laustourists, s.window = 'periodic'))
autoplot(taustou_adj)
```




```r
ggtsdisplay(diff(taustou_adj,1))
```



Perhaps I might choose a ARIMA(1,0,0)(0,1,1)[4] or a ARIMA(0,0,1)(0,1,1)[4]. But I can't pick without more analysis.

Does auto.arima give the same model that you chose? If not, which model do you think is better?

```r
auto.arima(laustourists, stepwise = F)
```

```
## Series: laustourists
## ARIMA(1,0,0)(0,1,1)[4] with drift
```

```
##
## Coefficients:
##          ar1     sma1   drift
##       0.4154  -0.9043  0.0128
## s.e.  0.1387   0.2711  0.0011
##
## sigma^2 estimated as 0.004541:  log likelihood=54.52
## AIC=-101.05   AICc=-100.02   BIC=-93.91
```

It's similar to what I picked, and I tend to agree with it's selection.

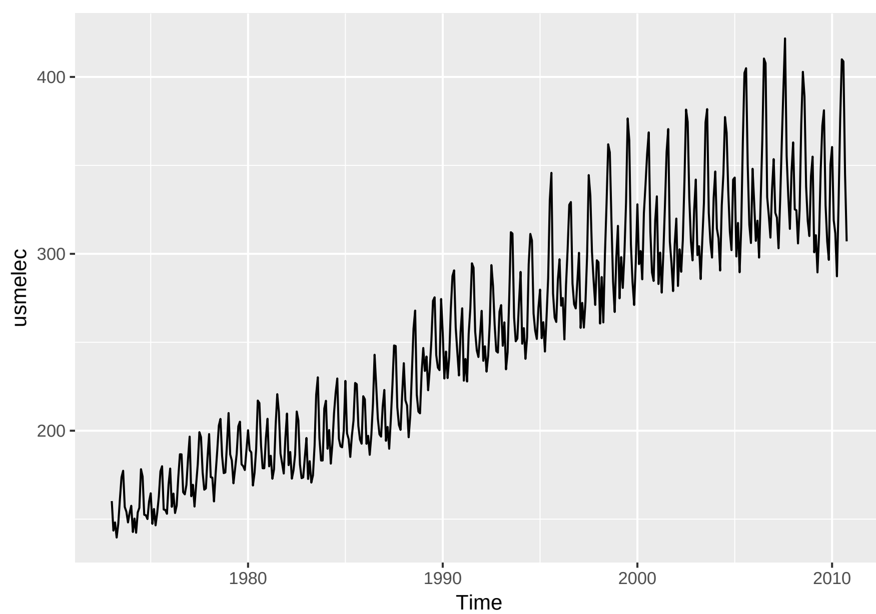
Write the model in terms of the backshift operator, and then without using the backshift operator.

```
(1 - phi1 x B)(1 - PHI1 x B^4)(1 - B^4)y_t = (1 + THETA1 x B^4)e_t
```

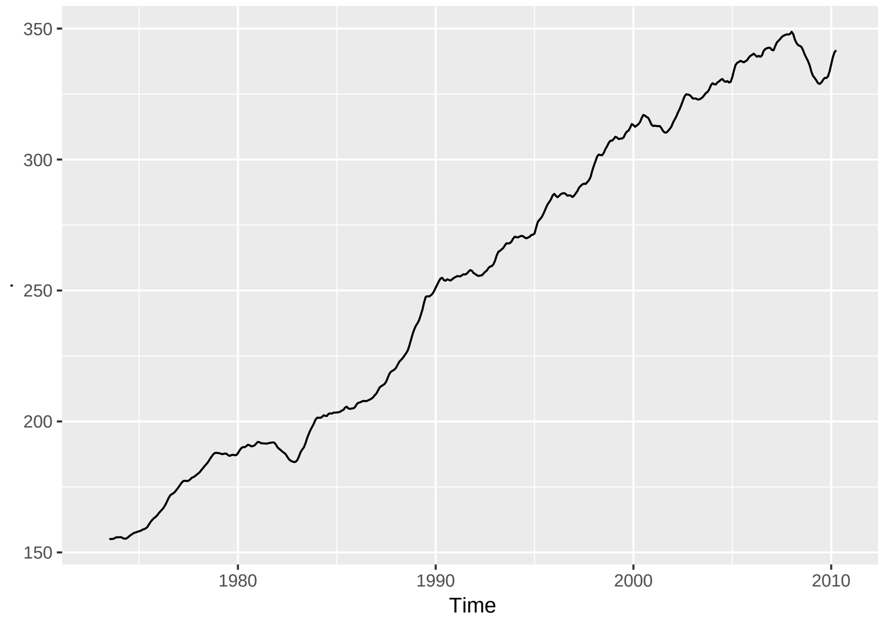
Consider the total net generation of electricity (in billion kilowatt hours) by the U.S. electric industry (monthly for the period 1985–1996). (Data set usmelec.) In general there are two peaks per year: in mid-summer and mid-winter.

Examine the 12-month moving average of this series to see what kind of trend is involved.

```
autoplot(usmelec)
```




```
ma(usmelec, order = 12) %>% autoplot()
```



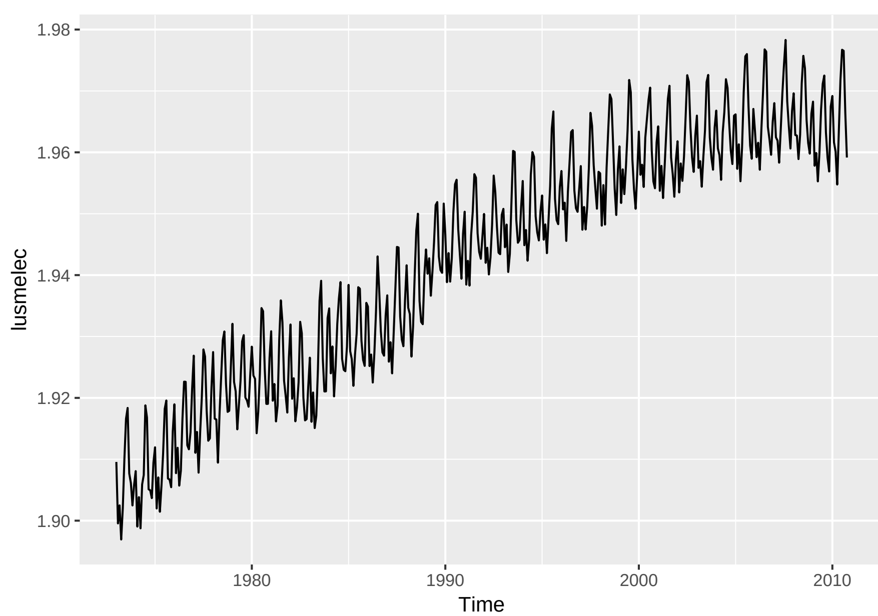
Do the data need transforming? If so, find a suitable transformation.

Yes, there is an increasing seasonality. Looks like an inverse sqrt will work.

```
BoxCox.lambda(usmelec)
```

```
## [1] -0.4772402
```

```
lusmelec <- BoxCox(usmelec, lambda = -0.4772402)
autoplot(lusmelec)
```



Are the data stationary? If not, find an appropriate differencing which yields stationary data.

The data are not stationary. A difference of 1 is needed to make it stationary according to the KPSS test.

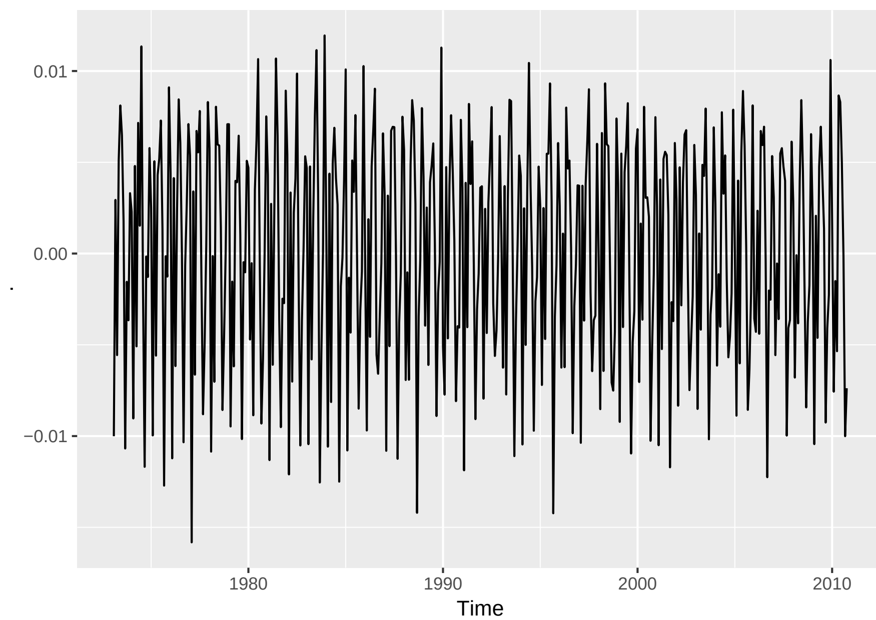
```
diff(lusmelec,1) %>%  kpss.test()
```

```
## Warning in kpss.test(.): p-value greater than printed p-value

##
##  KPSS Test for Level Stationarity
##
## data:  .
## KPSS Level = 0.022613, Truncation lag parameter = 4, p-value = 0.1
```

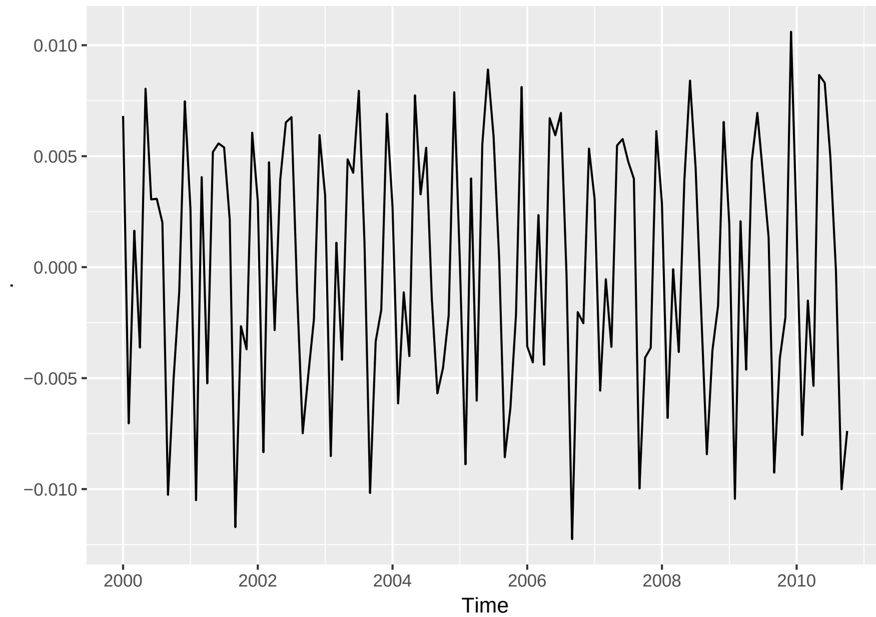
Visually, the signal looks stationary.

```
lusmelec_diff <- diff(lusmelec)
lusmelec_diff %>% autoplot()
```



Zooming in a bit more, I think I can see seasonality which a diff=1 hasn't got rid of.

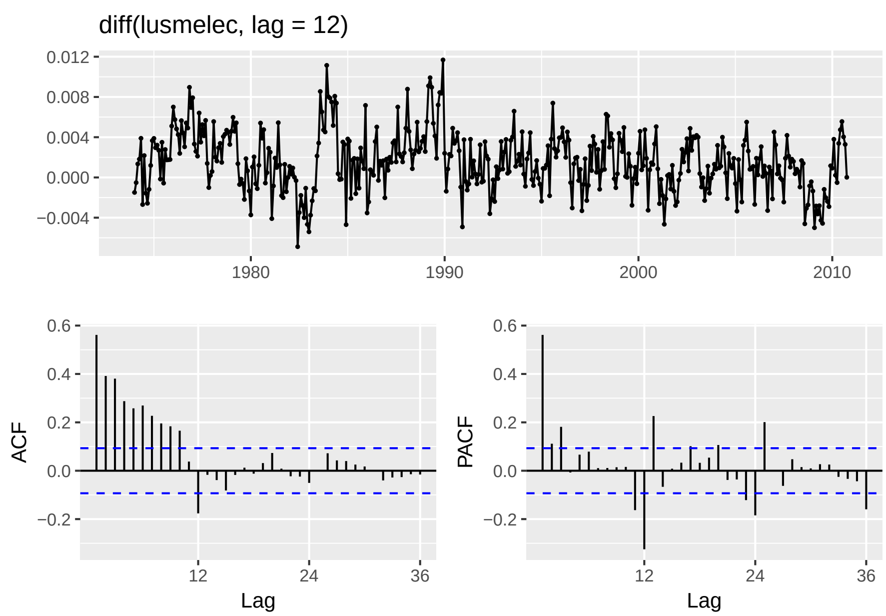
```
lusmelec_diff %>% window(start=2000) %>% autoplot()
```



If we look at the ACF, PACF plots, we can see two patterns: * a seasonal pattern at 12, 24, 36 * another seasonal pattern at 3, 6, 9, . . .

Looks like we have a complex dual-seasonal pattern even after differencing.

```
ggtsdisplay(diff(lusmelec,lag = 12))
```



The PACF plot shows lags at 12, 24, 36 which are seasonal lags. Perhaps an AR(3) term for seasonal is needed. For non-seasonal component, considering the drop in ACF values, the most significant PACF value is for lag = 1. Perhaps an ARIMA(1,0,0)(0,1,3)[12] is a good guess.

Identify a couple of ARIMA models that might be useful in describing the time series. Which of your models is the best according to their AIC values?

It looks like ARIMA(1,0,1)(1,1,1)[12] has the lowest AIC of -4240.98, after some investigation.

```
Arima(lusmelec, order = c(1,0,0), seasonal = list(order=c(0,1,3), period=12))
```

```
## Series: lusmelec
## ARIMA(1,0,0)(0,1,3)[12]
##
## Coefficients:
##          ar1     sma1     sma2    sma3
##       0.9843  -0.8726  -0.0957  0.1128
## s.e.  0.0106   0.0512   0.0581  0.0502
##
## sigma^2 estimated as 4.358e-06:  log likelihood=2098.67
## AIC=-4187.35   AICc=-4187.21   BIC=-4166.89
```

```
Arima(lusmelec, order = c(1,0,0), seasonal = list(order=c(0,1,2), period=12))
```

```
## Series: lusmelec
## ARIMA(1,0,0)(0,1,2)[12]
##
## Coefficients:
##          ar1     sma1     sma2
##       0.9858  -0.8526  -0.0223
## s.e.  0.0100   0.0551   0.0531
##
## sigma^2 estimated as 4.398e-06:  log likelihood=2096.22
## AIC=-4184.43   AICc=-4184.34   BIC=-4168.07
```

```
Arima(lusmelec, order = c(1,0,0), seasonal = list(order=c(1,1,2), period=12))
```

```
## Series: lusmelec
## ARIMA(1,0,0)(1,1,2)[12]
##
## Coefficients:
##          ar1     sar1     sma1     sma2
##       0.9864  -0.4831  -0.3578  -0.4591
## s.e.  0.0099   0.3726   0.3533   0.3042
##
## sigma^2 estimated as 4.401e-06:  log likelihood=2096.53
## AIC=-4183.06   AICc=-4182.92   BIC=-4162.6
```

```
Arima(lusmelec, order = c(1,0,1), seasonal = list(order=c(0,1,3), period=12))
```

```
## Series: lusmelec
## ARIMA(1,0,1)(0,1,3)[12]
##
## Coefficients:
##          ar1      ma1     sma1     sma2    sma3
```

```
##       0.9965  -0.5114  -0.8071  -0.1279  0.0895
## s.e.  0.0037   0.0645   0.0512   0.0577  0.0508
##
## sigma^2 estimated as 3.881e-06:  log likelihood=2126.49
## AIC=-4240.98   AICc=-4240.79   BIC=-4216.44
```

```r
Arima(lusmelec, order = c(1,0,1), seasonal = list(order=c(1,1,1), period=12))
```

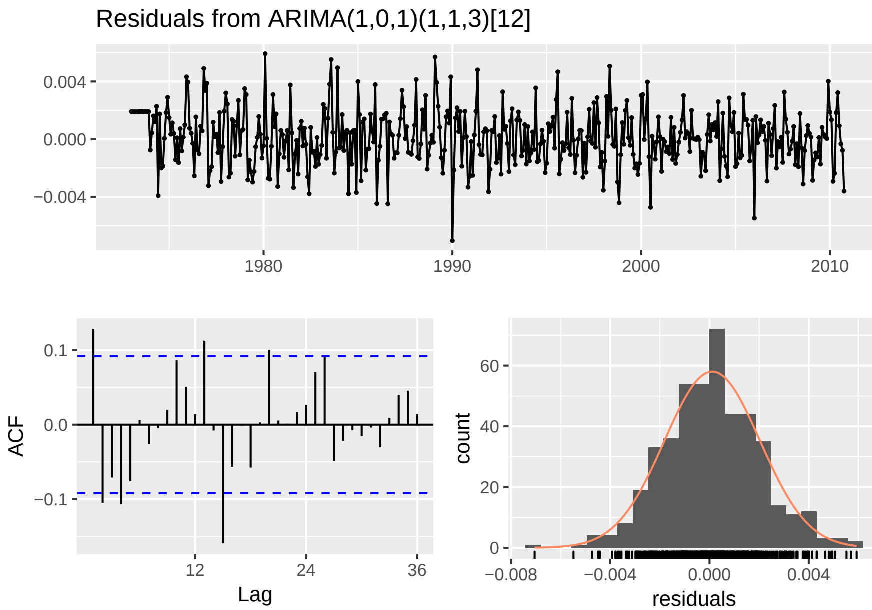
```
## Series: lusmelec
## ARIMA(1,0,1)(1,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1
##       0.9971  -0.5191  0.0702  -0.8720
## s.e.  0.0033   0.0645  0.0568   0.0309
##
## sigma^2 estimated as 3.903e-06:  log likelihood=2124.75
## AIC=-4239.51   AICc=-4239.37   BIC=-4219.05
```

Estimate the parameters of your best model and do diagnostic testing on the residuals. Do the residuals resemble white noise? If not, try to find another ARIMA model which fits better.

```r
fit_selected <- Arima(lusmelec, order = c(1,0,1), seasonal = list(order=c(1,1,3), period=12))
fit_selected %>% summary()
```

```
## Series: lusmelec
## ARIMA(1,0,1)(1,1,3)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sma1    sma2    sma3
##       0.9959  -0.5069  0.3740  -1.1789  0.1630  0.1226
## s.e.  0.0042   0.0651  0.3026   0.3007  0.2565  0.0506
##
## sigma^2 estimated as 3.881e-06:  log likelihood=2127.04
## AIC=-4240.07   AICc=-4239.81   BIC=-4211.43
##
## Training set error measures:
##                        ME        RMSE         MAE        MPE       MAPE
## Training set 0.0001088471 0.001930585 0.001511984 0.00564225 0.07789407
##                   MASE      ACF1
## Training set 0.5754233 0.1284949
```

```r
fit_selected %>% checkresiduals()
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,0,1)(1,1,3)[12]
## Q* = 54.46, df = 18, p-value = 1.556e-05
##
## Model df: 6.   Total lags used: 24
```

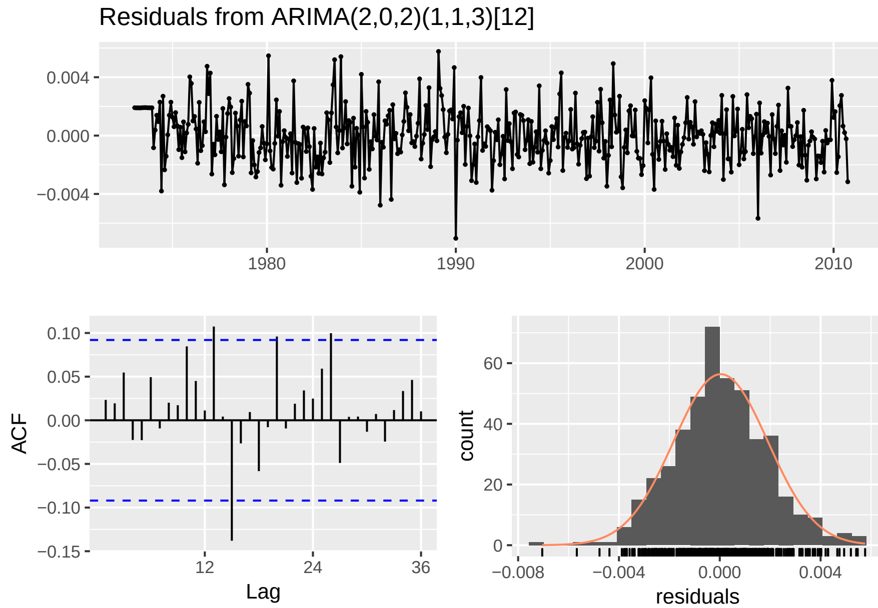
Residuals show a left skew and the ACF plot shows significant lags at 1,2,14 etc. Trying out another model: an ARIMA(2,0,2)(1,1,3)[12] improved the AIC to -4264.

```r
fit_selected <- Arima(lusmelec, order = c(2,0,2), seasonal = list(order=c(1,1,3), period=12))
fit_selected %>% summary()
```

```
## Series: lusmelec
## ARIMA(2,0,2)(1,1,3)[12]
##
## Coefficients:
##          ar1      ar2      ma1      ma2    sar1     sma1    sma2    sma3
##       1.3027  -0.3035  -0.7186  -0.0789  0.4392  -1.2694  0.2476  0.1115
## s.e.  0.1826   0.1822   0.1872   0.1259  0.3036   0.3025  0.2690  0.0489
##
## sigma^2 estimated as 3.656e-06:  log likelihood=2141.09
## AIC=-4264.18   AICc=-4263.77   BIC=-4227.36
##
## Training set error measures:
##                        ME        RMSE         MAE         MPE       MAPE
## Training set 3.731734e-05 0.001869441 0.001448626 0.001974489 0.07462978
##                   MASE       ACF1
```

```
## Training set 0.5513111 0.02330224
```

```r
fit_selected %>% checkresiduals()
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(2,0,2)(1,1,3)[12]
## Q* = 29.996, df = 16, p-value = 0.01802
##
## Model df: 8.   Total lags used: 24
```

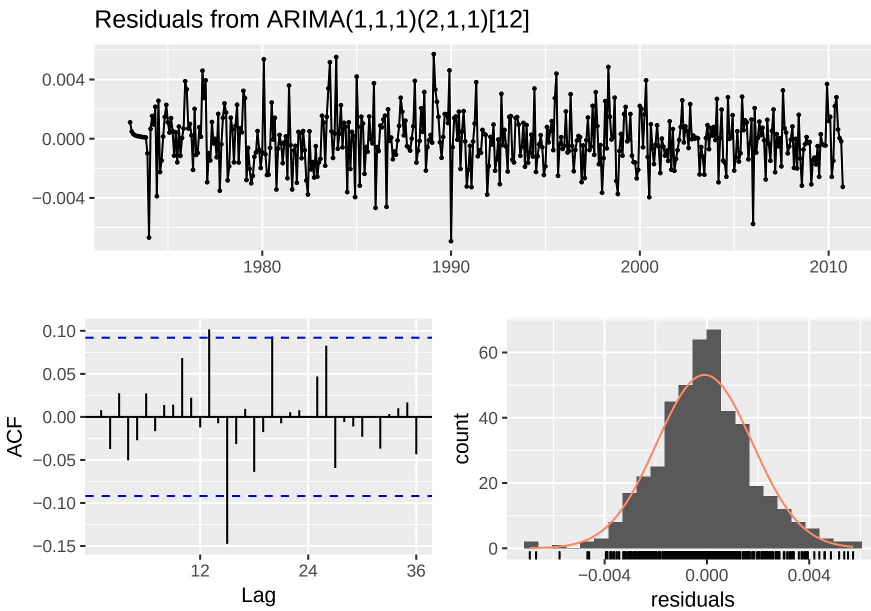
What does an auto.arima say? ARIMA(1,1,1)(2,1,1)[12]… but the AICc is worse than the model I chose.

```r
fit_selected <- auto.arima(lusmelec, stepwise = F)
fit_selected %>% summary()
```

```
## Series: lusmelec
## ARIMA(1,1,1)(2,1,1)[12]
##
## Coefficients:
##          ar1      ma1    sar1     sar2     sma1
##       0.4002  -0.8295  0.0269  -0.1016  -0.8485
## s.e.  0.0652   0.0385  0.0579   0.0553   0.0366
##
## sigma^2 estimated as 3.653e-06:  log likelihood=2135.33
## AIC=-4258.65   AICc=-4258.46   BIC=-4234.12
##
## Training set error measures:
##                         ME        RMSE         MAE          MPE       MAPE
## Training set -9.026359e-05 0.001873122 0.001423716 -0.004660559 0.07332098
```

```
##                     MASE        ACF1
## Training set 0.5418307 0.007791069
```

```
fit_selected %>% checkresiduals()
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,1)(2,1,1)[12]
## Q* = 27.628, df = 19, p-value = 0.09086
##
## Model df: 5.   Total lags used: 24
```

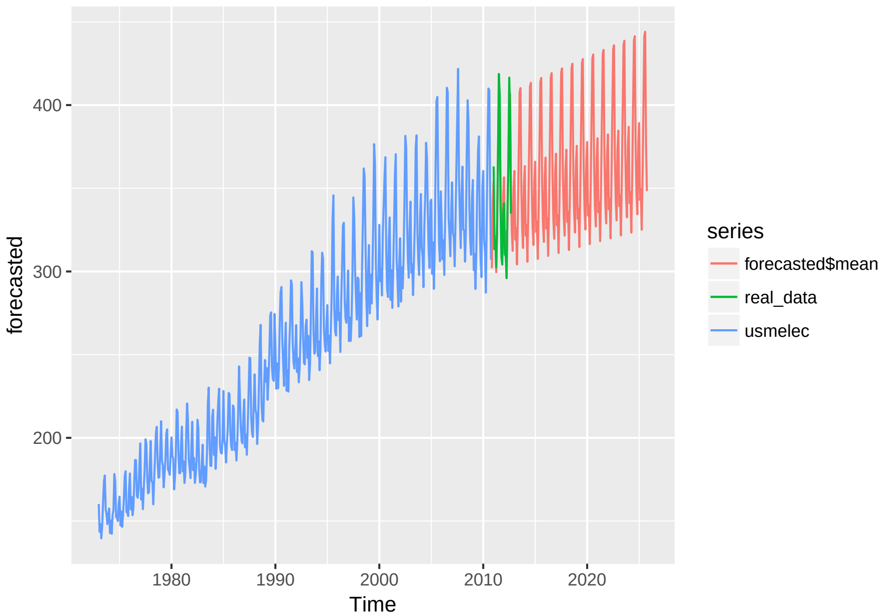
Forecast the next 15 years of generation of electricity by the U.S. electric industry. Get the latest figures from http://data.is/zgRWCO to check on the accuracy of your forecasts.

The forecasted model works quite well!! The green line shows the real data for the past few years, while the red line is my forecasts. I am quite pleased.

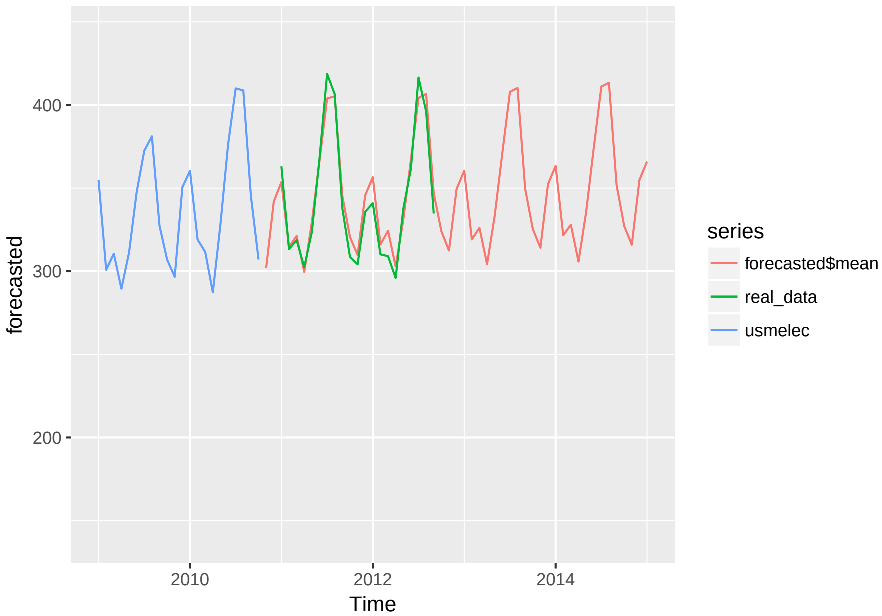
```
fit_selected <- Arima(lusmelec, order = c(2,0,2),
                      seasonal = list(order=c(1,1,3), period=12))
forecasted <- fit_selected %>% forecast(h=15*12, lambda = -0.4772402)
```

```
## Warning in InvBoxCox(pred$pred, lambda, biasadj, var(residuals(object), :
## biasadj information not found, defaulting to FALSE.
```

```
forecasted <- ts.union(usmelec, forecasted$mean)
load("~/GDrive NU/413 Time Series/413_RProject/newdata.Rdata")
real_data <- ts(newdata$y, start=c(2011,1), frequency = 12)
autoplot(forecasted)+autolayer(real_data)
```




```r
autoplot(forecasted)+autolayer(real_data)+xlim(c(2009,2015))
```



How many years of forecasts do you think are sufficiently accurate to be usable?

Probably a year or two years out. Because forecasting assumes the underlying data generating process remains unchanged. If this changes, then the model is useless.