В [1]:

```
library(fpp2)
```

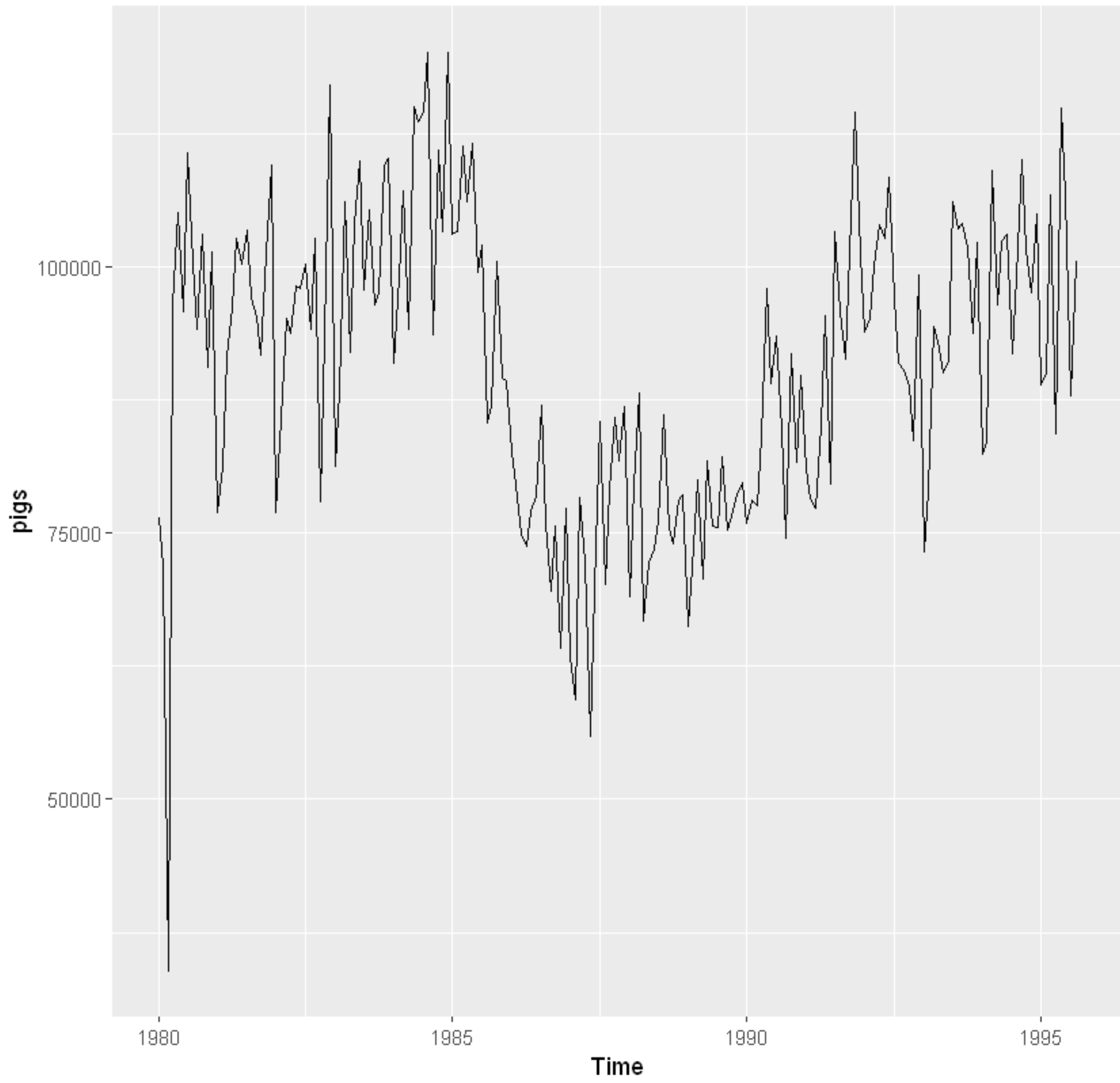
```
Warning message:
"package 'fpp2' was built under R version 3.6.3"Registered S3 method overwri
tten by 'xts':
  method     from
  as.zoo.xts zoo
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo
-- Attaching packages ---------------------------------------------------
-------------------------------- fpp2 2.4 --
v ggplot2   3.3.3     v fma       2.4
v forecast  8.13      v expsmooth 2.3
Warning message:
"package 'ggplot2' was built under R version 3.6.3"Warning message:
"package 'forecast' was built under R version 3.6.3"Warning message:
"package 'fma' was built under R version 3.6.3"Warning message:
"package 'expsmooth' was built under R version 3.6.3"
```

1. Consider the `pigs` series— the number of pigs slaughtered in Victoria each month.

> a. Use the `ses` function in R to find the optimal values of α and ℓ_0, and generate forecasts for the next four months.

В [3]:

```
autoplot(pigs)
```

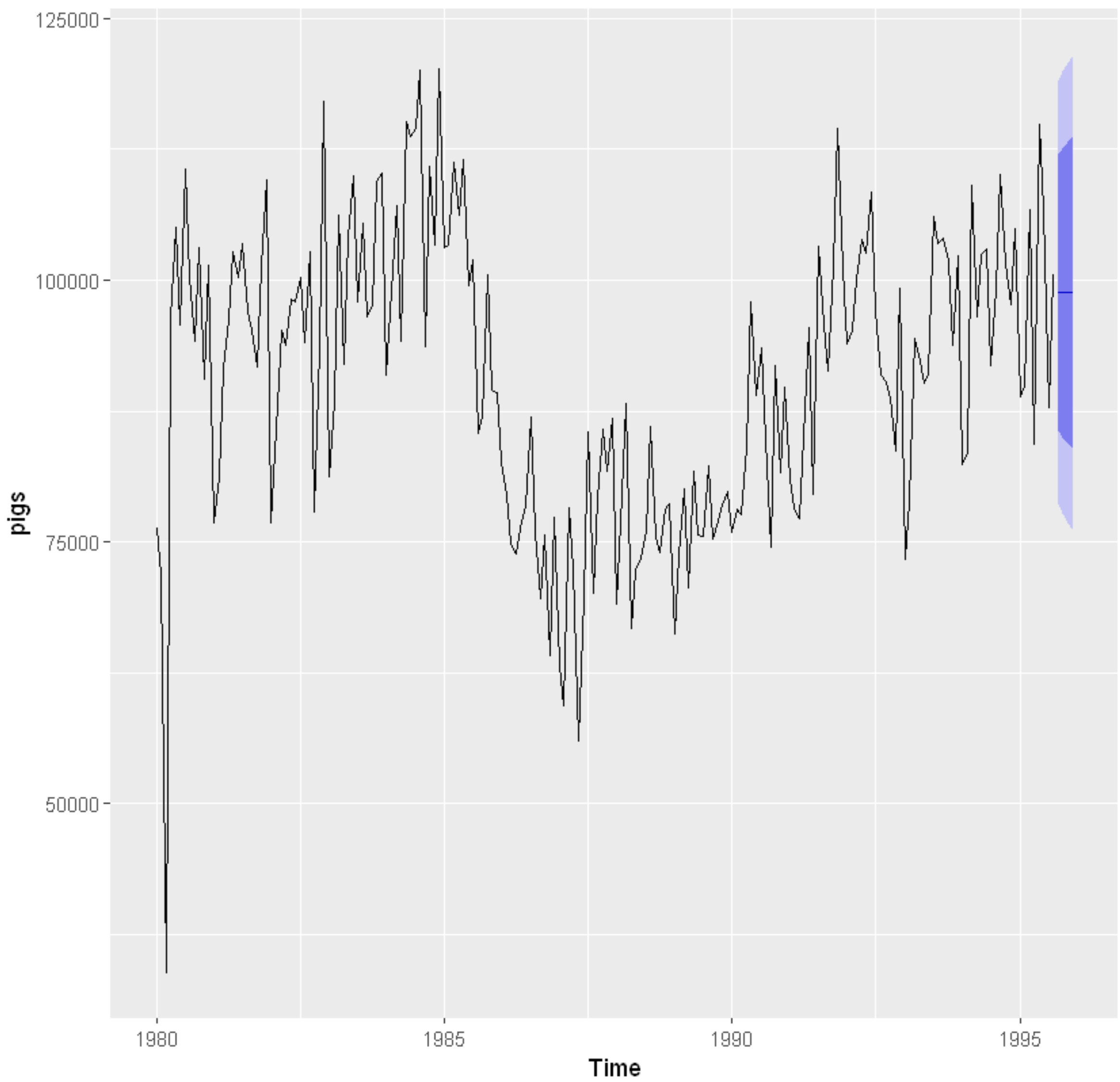


В [4]:

```
fc <- ses(pigs, h=4)
```

В [5]:

```
autoplot(pigs) + autolayer(fc)
```



В [6]:

```
fc
```

```
         Point Forecast     Lo 80    Hi 80    Lo 95    Hi 95
Sep 1995       98816.41 85605.43 112027.4 78611.97 119020.8
Oct 1995       98816.41 85034.52 112598.3 77738.83 119894.0
Nov 1995       98816.41 84486.34 113146.5 76900.46 120732.4
Dec 1995       98816.41 83958.37 113674.4 76092.99 121539.8
```

В [7]:

```
summary(fc)
```

```
Forecast method: Simple exponential smoothing

Model Information:
Simple exponential smoothing 

Call:
 ses(y = pigs, h = 4) 

  Smoothing parameters:
    alpha = 0.2971 

  Initial states:
    l = 77260.0561 

  sigma:  10308.58

     AIC     AICc      BIC 
4462.955 4463.086 4472.665 

Error measures:
                   ME    RMSE      MAE       MPE     MAPE      MASE       AC
F1
Training set 385.8721 10253.6 7961.383 -0.922652 9.274016 0.7966249 0.012822
39

Forecasts:
         Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
Sep 1995       98816.41 85605.43 112027.4 78611.97 119020.8
Oct 1995       98816.41 85034.52 112598.3 77738.83 119894.0
Nov 1995       98816.41 84486.34 113146.5 76900.46 120732.4
Dec 1995       98816.41 83958.37 113674.4 76092.99 121539.8
```

$\alpha = 0.2971$

$\ell_0 = 77260.0561$

b. Compute a 95% prediction interval for the first forecast using $\hat{y} \pm 1.96s$ where s is the standard deviation of the residuals. Compare your interval with the interval produced by R.

В [8]:

```
s <- sqrt(var(fc$residuals))
m <- fc$mean[1]
c(lb = m - 1.96 * s, ub = m + 1.96 * s)
```

lb
78679.9672534162
ub
118952.844969765

В [9]:

```
s1 <- sqrt(sum(fc$residuals ^ 2) / (length(pigs) - 2))
c(lb = m - 1.96 * s1, ub = m + 1.96 * s1)
```

lb
78611.5971896414
ub
119021.21503354

2. Write your own function to implement simple exponential smoothing. The function should take arguments `y` (the time series), `alpha` (the smoothing parameter α) and `level` (the initial level ℓ_0). It should return the forecast of the next observation in the series. Does it give the same forecast as `ses`?

В [10]:

```
fses <- function(y, alpha, level){
    n <- length(y)
    yt <- numeric(n + 1)
    yt[1] <- level
    for (i in 2:(n + 1)){
        yt[i] <- alpha * y[i - 1] + (1 - alpha) * yt[i - 1]
    }
    return(yt[n + 1])
}
```

В [11]:

```
fc$model$par
```

alpha
0.297148833372095
l
77260.0561458528

В [12]:

```
fc$mean[1]
```

98816.4061115907

В [13]:

```
fses(pigs, 0.297148833372095, 77260.0561458528)
```

98816.4061115907

3. Modify your function from the previous exercise to return the sum of squared errors rather than the forecast of the next observation. Then use the `optim` function to find the optimal values of α and ℓ_0. Do you get the same values as the `ses` function?

В [14]:

```
fses_se <- function(par, y){
    alpha <- par[1]
    level <- par[2]
    n <- length(y)
    yt <- numeric(n)
    yt[1] <- level
    for (i in 2:n){
        yt[i] <- alpha * y[i - 1] + (1 - alpha) * yt[i - 1]
    }
    return(sum((y - yt) ^2))
}
```

В [15]:

```
fc$model$par
```

alpha
0.297148833372095
l
77260.0561458528

В [16]:

```
optim(c(0.1, pigs[1]), fses_se, y=pigs)$par
```

0.297118970754748 77265.8747808378

В [17]:

```
fses(pigs, 0.297118970754748, 77265.8747808378)
```

98816.4356357345

4. Combine your previous two functions to produce a function which both finds the optimal values of α and ℓ_0, and produces a forecast of the next observation in the series.

В [18]:

```
fses1 <- function(y, h=4){
    par <- optim(c(0.1, y[1]), fses_se, y=y)$par
    alpha <- par[1]
    level <- par[2]
    n <- length(y)
    yt <- numeric(n + 1)
    yt[1] <- level
    for (i in 2:(n + 1)){
        yt[i] <- alpha * y[i - 1] + (1 - alpha) * yt[i - 1]
    }
    return(rep(yt[n + 1], h))
}
```

В [19]:

```
fses1(pigs, h=4)
```

98816.4356357345 98816.4356357345 98816.4356357345 98816.4356357345

В [20]:

```
ses(pigs, h=4)
```

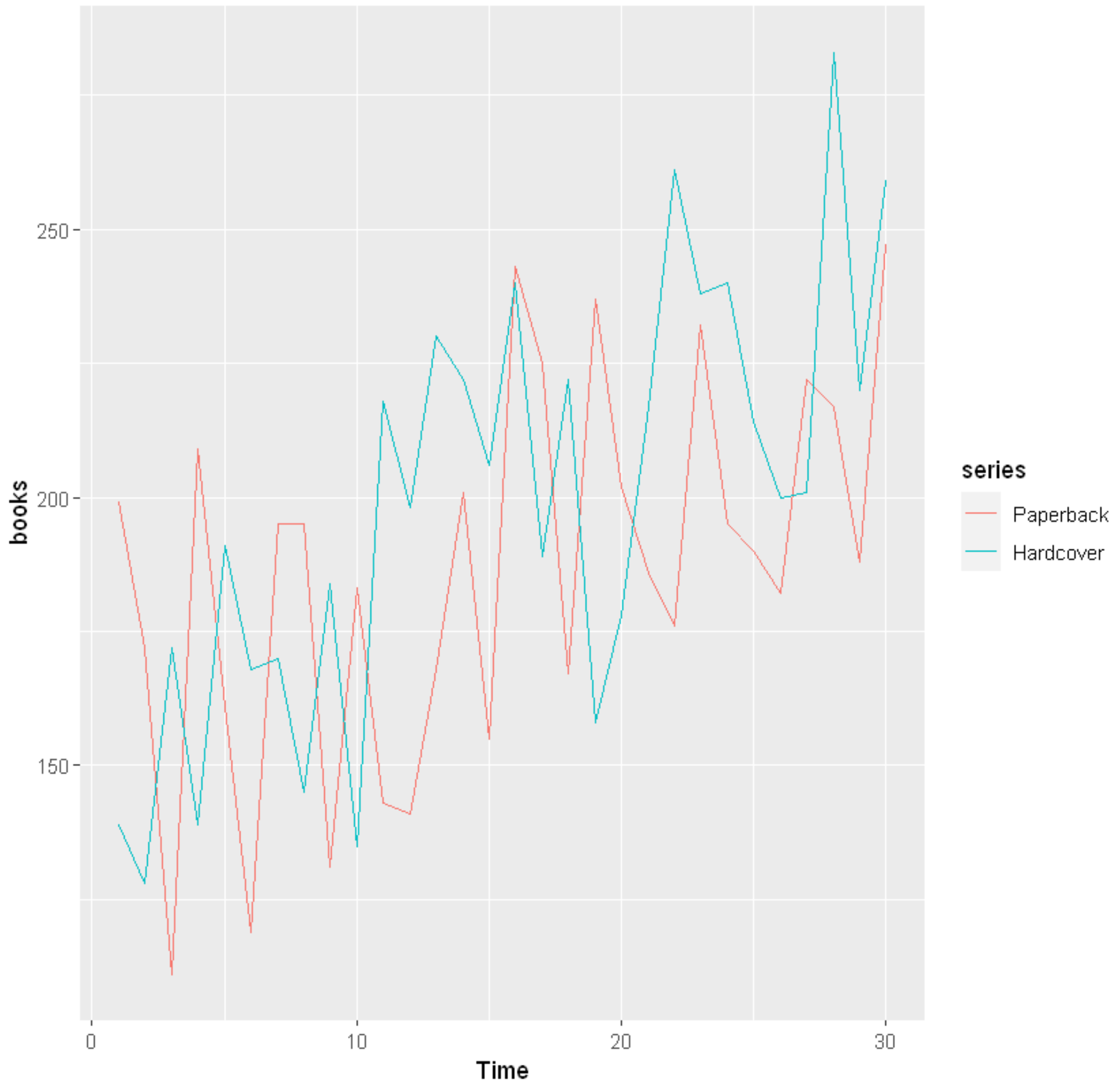
	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
Sep 1995	98816.41	85605.43	112027.4	78611.97	119020.8
Oct 1995	98816.41	85034.52	112598.3	77738.83	119894.0
Nov 1995	98816.41	84486.34	113146.5	76900.46	120732.4
Dec 1995	98816.41	83958.37	113674.4	76092.99	121539.8

5. 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.

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

В [21]:

```
autoplot(books)
```



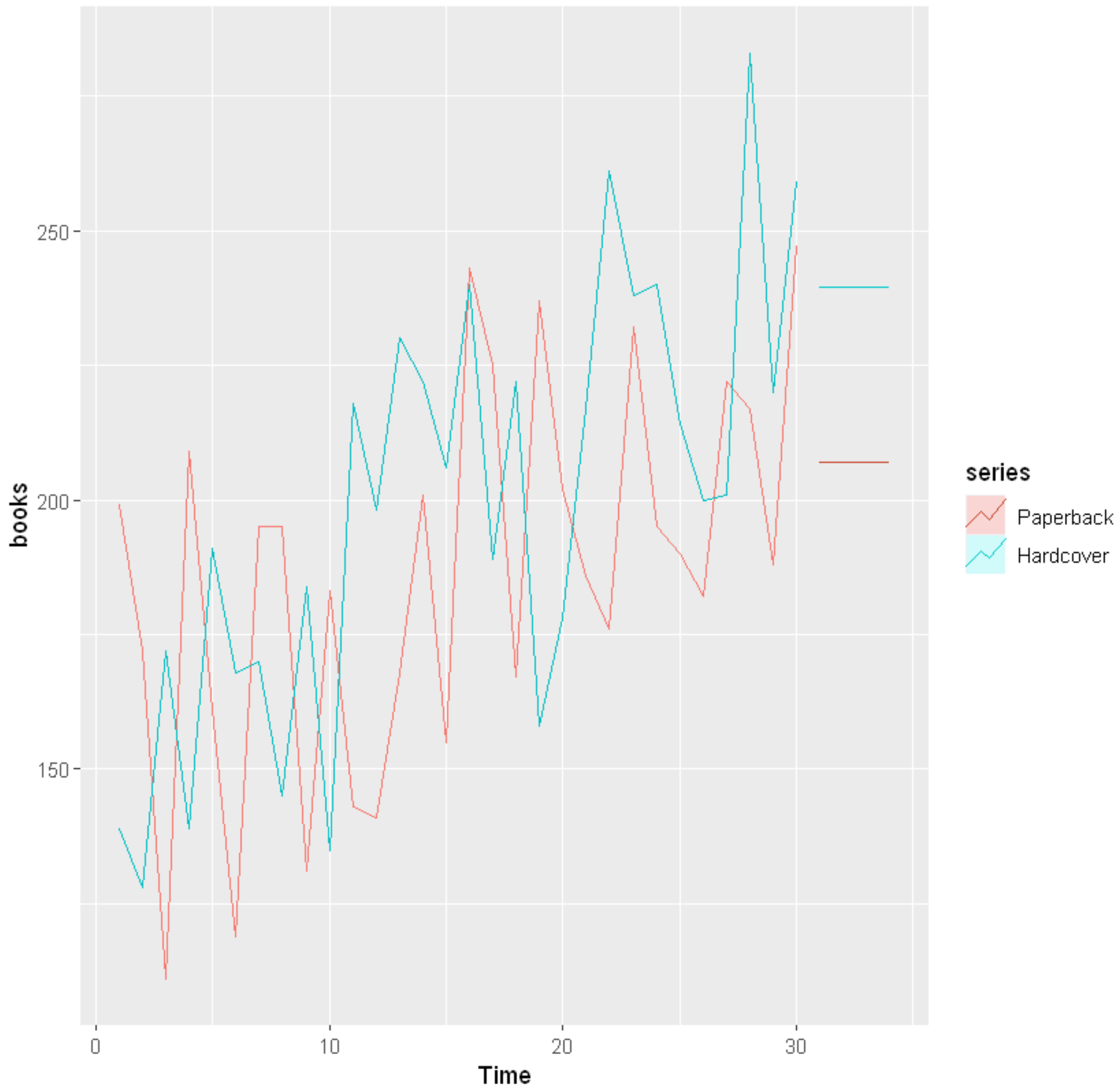
b. Use the `ses()` function to forecast each series, and plot the forecasts.

В [22]:

```
fchd <- ses(books[,"Hardcover"], h=4)
fcpb <- ses(books[,"Paperback"], h=4)
```

В [23]:

```
autoplot(books) +
autolayer(fchd, series = "Hardcover", PI=FALSE) +
autolayer(fcpb, series = "Paperback", PI=FALSE)
```



c. Compute the RMSE values for the training data in each case.

В [24]:

```
round(accuracy(fchd),2)
round(accuracy(fcpb),2)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	9.17	31.93	26.77	2.64	13.39	0.8	-0.14

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	7.18	33.64	27.84	0.47	15.58	0.7	-0.21

6.

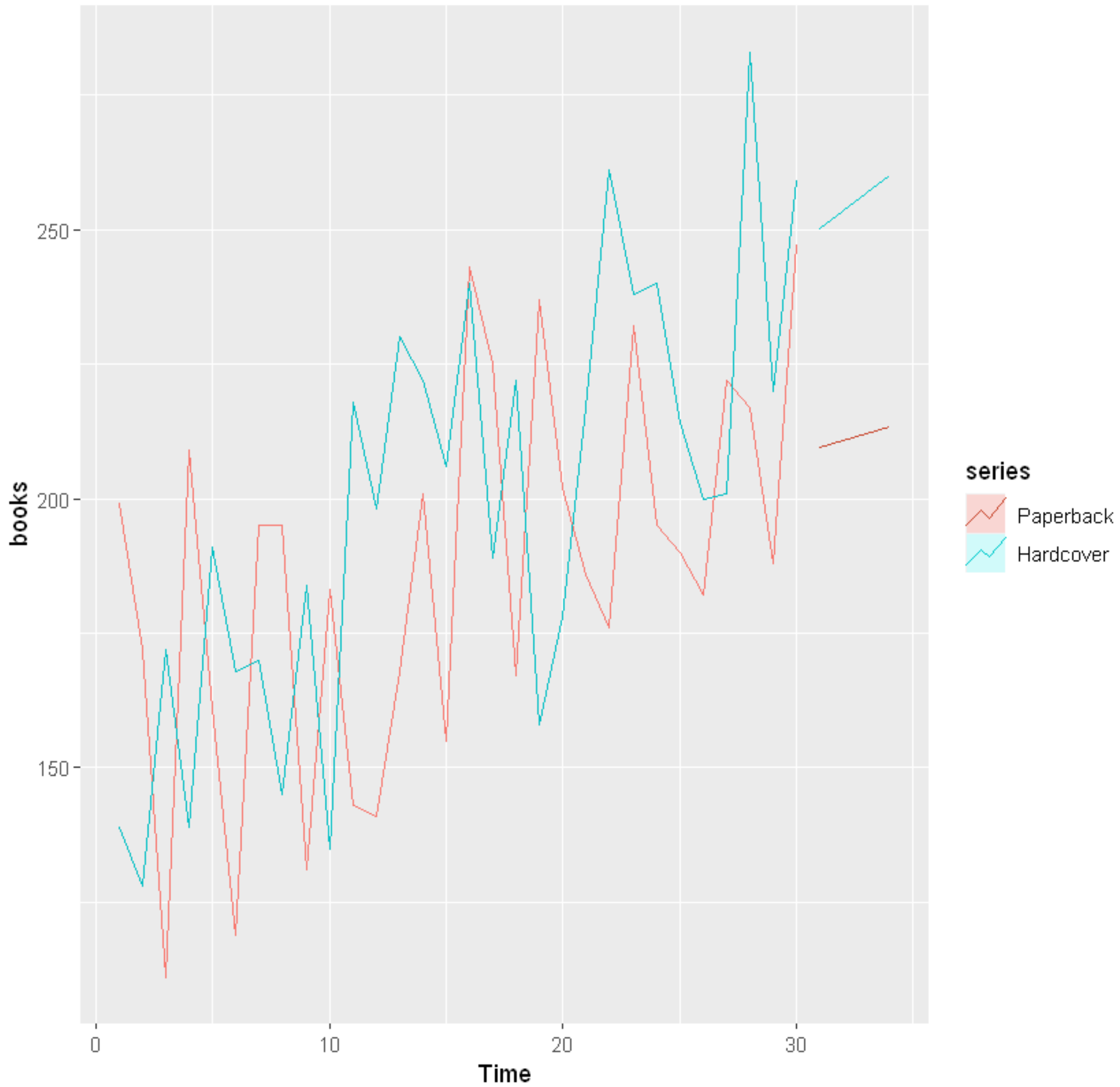
> a. Now apply Holt’s linear method to the `paperback` and `hardback` series and compute four-day forecasts in each case.

В [25]:

```
hh <- holt(books[,"Hardcover"], h=4)
hp <- holt(books[,"Paperback"], h=4)
```

В [26]:

```
autoplot(books) +
autolayer(hh, series = "Hardcover", PI=FALSE) +
autolayer(hp, series = "Paperback", PI=FALSE)
```



> b. Compare the RMSE measures of Holt's method for the two series to those of simple exponential smoothing in the previous question. (Remember that Holt's method is using one more parameter than SES.) Discuss the merits of the two forecasting methods for these data sets.

В [27]:

```
round(accuracy(hh),2)
round(accuracy(hp),2)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-0.14	27.19	23.16	-2.11	12.16	0.69	-0.03

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.72	31.14	26.18	-5.51	15.58	0.66	-0.18

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

Так как для обоих рядов имеет место тренд, поэтому RMSE для метода Холта меньше.

> d. Calculate a 95% prediction interval for the first forecast for each series, using the RMSE values and assuming normal errors. Compare your intervals with those produced using `ses` and `holt` .

В [28]:

```
shd <- sqrt(fchd$model$mse)
mhd <- fchd$mean[1]
c(lb = mhd - 1.96 * shd, ub = mhd + 1.96 * shd)
```

lb
176.97530263223
ub
302.144881371293

В [29]:

```
spb <- sqrt(fcpb$model$mse)
mpb <- fcpb$mean[1]
c(lb = mpb - 1.96 * spb, ub = mpb + 1.96 * spb)
```

lb
141.179798900711
ub
273.039531089726

В [30]:

```
shh <- sqrt(hh$model$mse)
mhh <- hh$mean[1]
c(lb = mhh - 1.96 * shh, ub = mhh + 1.96 * shh)
```

lb
196.874459367927
ub
303.473285056784

В [31]:

```
shp <- sqrt(hp$model$mse)
mhp <- hp$mean[1]
c(lb = mhp - 1.96 * shp, ub = mhp + 1.96 * shp)
```

lb
148.438396409231
ub
270.495134632871

7. For this exercise, use data set `eggs`, the price of a dozen eggs in the United States from 1900–1993. Experiment with the various options in the `holt()` function to see how much the forecasts change with damped trend, or with a Box-Cox transformation. Try to

develop an intuition of what each argument is doing to the forecasts.

[Hint: use `h=100` when calling `holt()` so you can clearly see the differences between the various options when plotting the forecasts.]

Which model gives the best RMSE?

В [32]:

```
autoplot(eggs)
```

В [33]:

```
fe1 <- holt(eggs, h=100)
```

В [34]:

```
accuracy(fe1)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	0.04499087	26.58219	19.18491	-1.142201	9.653791	0.9463626	0.01348202

В [35]:

```
fe2 <- holt(eggs, h=100, damped = TRUE)
```

В [36]:

```
accuracy(fe2)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-2.891496	26.54019	19.2795	-2.907633	10.01894	0.9510287	-0.003195358

В [37]:

```
fe2$model$par
```

alpha
0.846211920621314
beta
0.00401630172303531
phi
0.800000007622614
l
276.984167407755
b
4.99657927483563

В [38]:

```
fe3 <- holt(eggs, h=100, alpha=0.5, beta=0.5, damped = TRUE)
accuracy(fe3)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-1.359818	31.06105	23.2205	-1.566944	11.79842	1.145432	0.1565599

В [39]:

```
fe4 <- holt(eggs, h=100, alpha=0.5, beta=0.1, damped = TRUE)
accuracy(fe4)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.015718	28.28655	20.47675	-3.01732	10.63521	1.010088	0.2643378

В [40]:

```
fe5 <- holt(eggs, h=100, alpha=0.5, beta=0.05, damped = TRUE)
accuracy(fe5)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.722539	28.03246	20.41912	-3.611935	10.73974	1.007245	0.280374

В [41]:

```
fe6 <- holt(eggs, h=100, alpha=0.9, beta=0.001, damped = TRUE)
accuracy(fe6)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.405292	26.52701	19.64428	-2.9945	10.14573	0.9690229	-0.05782288

В [42]:

```
fe7 <- holt(eggs, h=100, alpha=0.9, beta=0.0001, damped = TRUE)
accuracy(fe7)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.352562	26.51669	19.59936	-2.981654	10.13096	0.9668069	-0.05646822

В [43]:

```
fe8 <- holt(eggs, h=100, alpha=0.9, beta=0.0001, phi=0.8, damped = TRUE)
accuracy(fe8)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.208492	26.51226	19.52392	-2.933841	10.10505	0.9630854	-0.05510603

В [44]:

```
fe9 <- holt(eggs, h=100, alpha=0.9, beta=0.0001, damped = FALSE)
accuracy(fe9)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-0.006124764	26.46502	19.3541	-1.181171	9.807844	0.9547087	-0.05226493

В [45]:

```
fe10 <- holt(eggs, h=100, alpha=0.9, beta=0.0001, damped = FALSE, lambda=0.2)
accuracy(fe10)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	0.8589922	26.39539	19.29629	-0.9689305	9.789108	0.951857	-0.04817418

В [46]:

```
fe11 <- holt(eggs, h=100, alpha=0.9, beta=0.0001, damped = FALSE, lambda=0.01)
accuracy(fe11)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	1.20373	26.38563	19.28933	-0.8511079	9.779588	0.9515139	-0.04609538

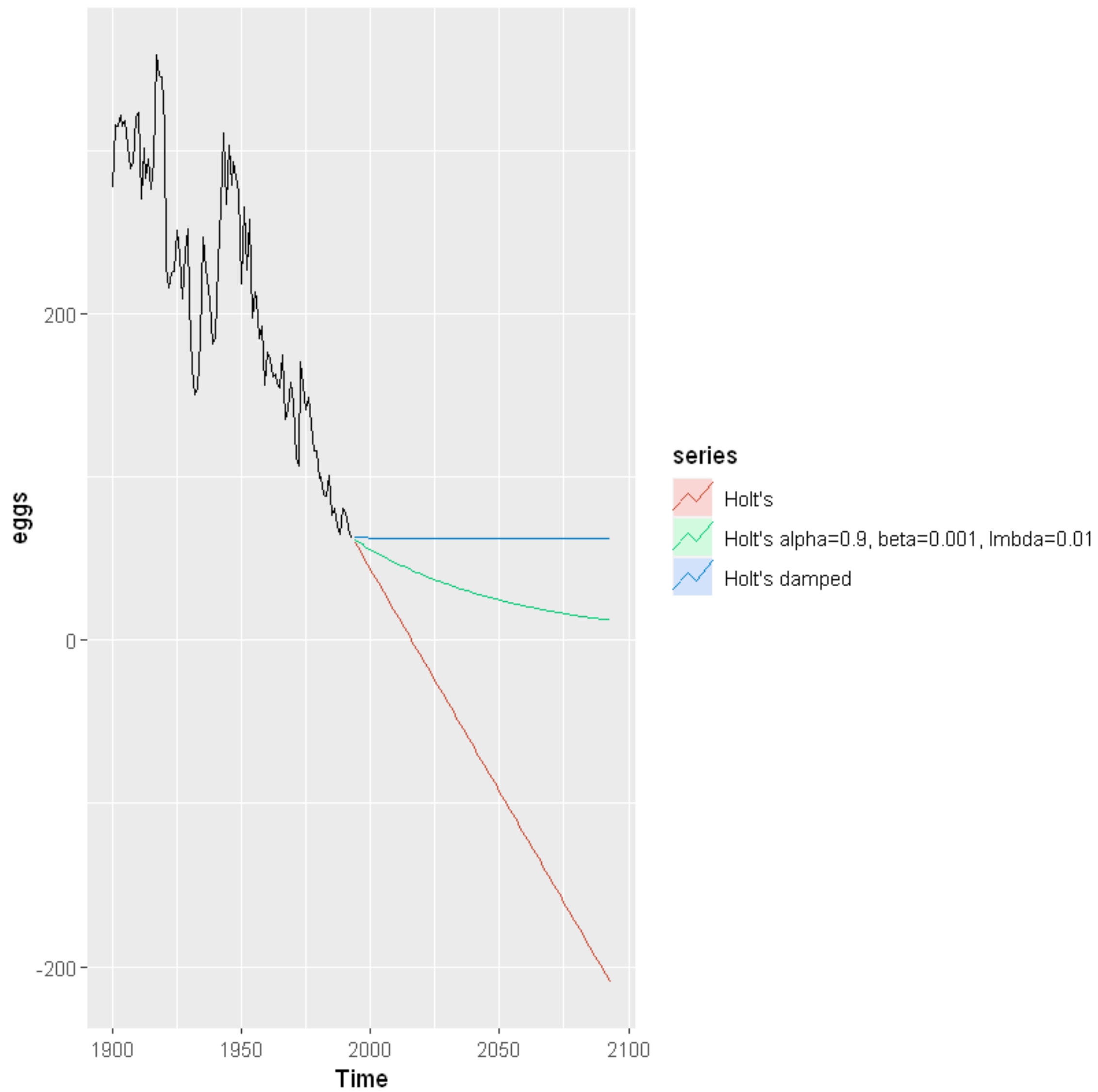
В [47]:

```
fe12 <- holt(eggs, h=100, alpha=0.9, beta=0.0001, damped = TRUE, lambda=0.8)
accuracy(fe12)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-3.185132	26.51349	19.52612	-2.919051	10.10338	0.9631941	-0.05508185

В [48]:

```
autoplot(eggs) +
autolayer(fe1, series = "Holt's", PI=FALSE) +
autolayer(fe2, series = "Holt's damped", PI=FALSE) +
autolayer(fe11, series = "Holt's alpha=0.9, beta=0.001, lmbda=0.01", PI=FALSE)
```



8. Recall your retail time series data (from Exercise 3 in Section 2.10).

> a. Why is multiplicative seasonality necessary for this series?

В [49]:

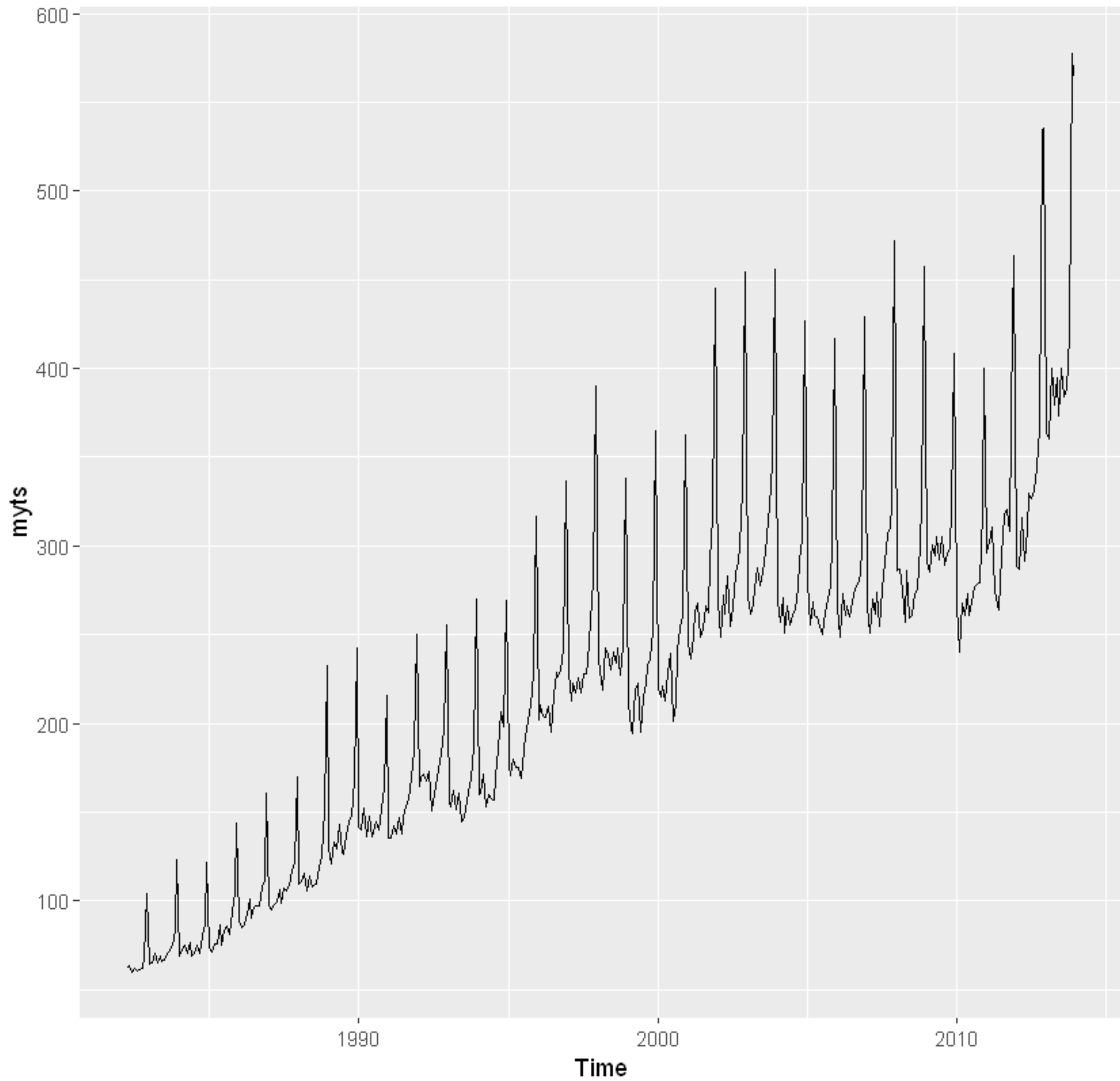
```
retaildata<-readxl::read_excel("retail.xlsx", skip=1)
```

В [50]:

```
myts <- ts(retaildata[,"A3349873A"],
  frequency=12, start=c(1982,4))
```

В [51]:

```
autoplot(myts)
```



Сезонные колебания увеличиваются пропорцианально уровню ряда.

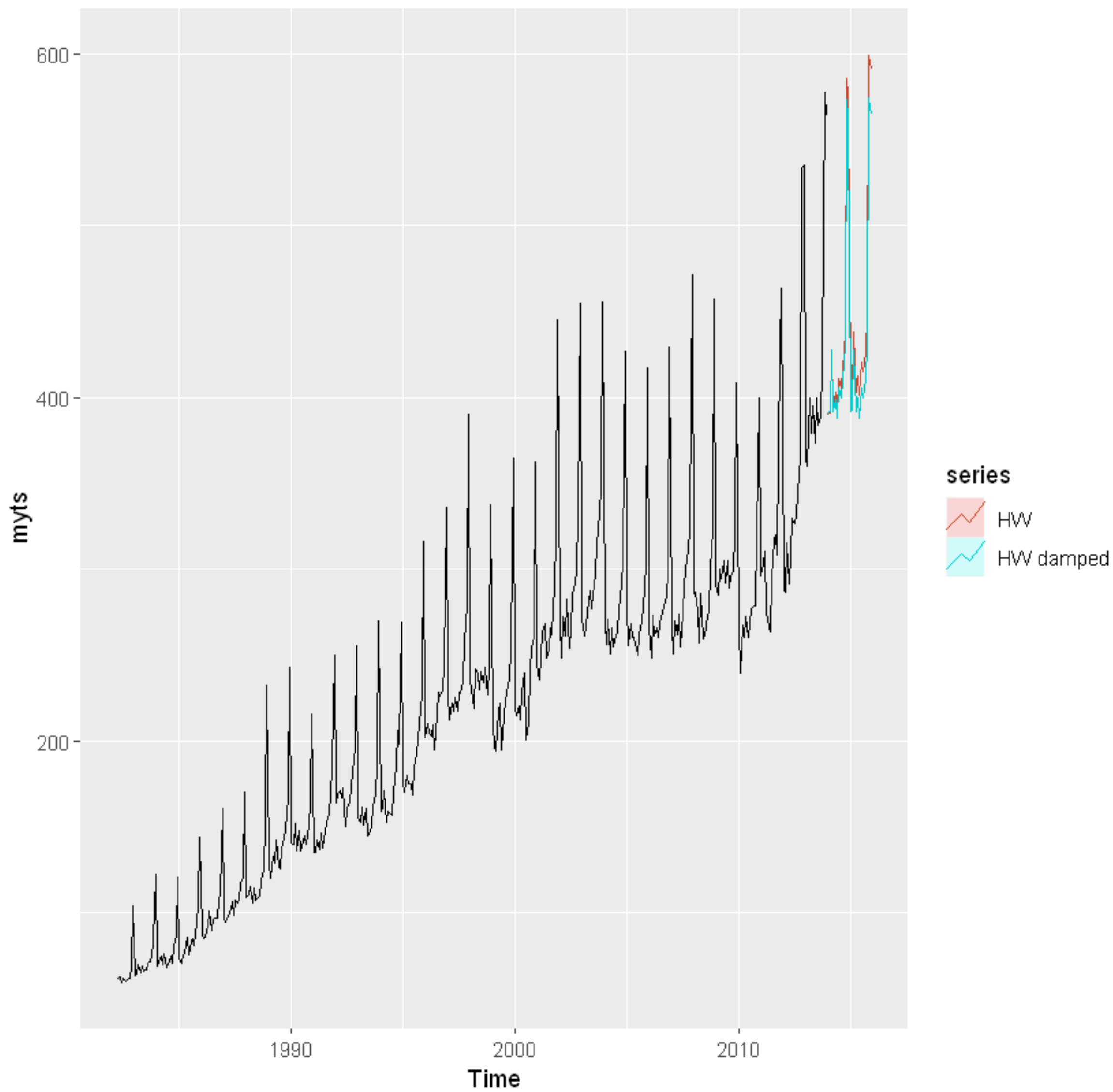
> b. Apply Holt-Winters' multiplicative method to the data. Experiment with making the trend damped.

В [52]:

```
hw1 <- hw(myts, seasonal = 'multiplicative', damped=FALSE)
hw2 <- hw(myts, seasonal = 'multiplicative', damped=TRUE)
```

В [53]:

```
autoplot(myts) +
autolayer(hw1, series="HW", PI=FALSE) +
autolayer(hw2, series="HW damped", PI=FALSE)
```



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

В [54]:

```
accuracy(hw1)
accuracy(hw2)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	0.1170648	13.29378	8.991856	-0.1217735	3.918351	0.4748948	0.08635577

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	1.414869	13.30494	9.042151	0.6105987	3.959617	0.4775511	0.04077895

d. Check that the residuals from the best method look like white noise.

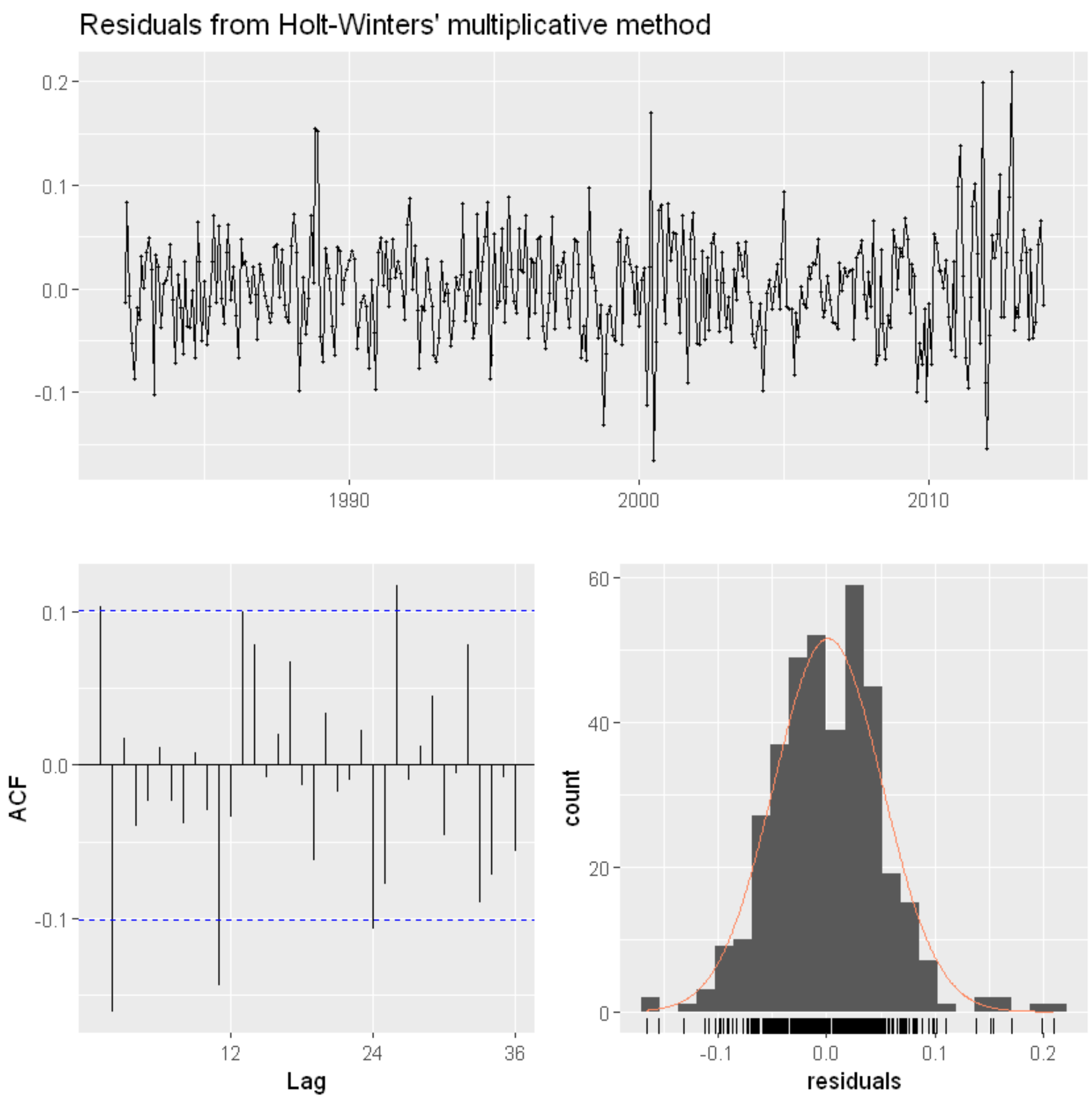
В [55]:

```
checkresiduals(hw1)
```

```
	Ljung-Box test

data:  Residuals from Holt-Winters' multiplicative method
Q* = 40.405, df = 8, p-value = 2.692e-06

Model df: 16.   Total lags used: 24
```



> e. Now find the test set RMSE, while training the model to the end of 2010. Can you beat the seasonal naïve approach from Exercise 8 in Section 3.7)?

В [56]:

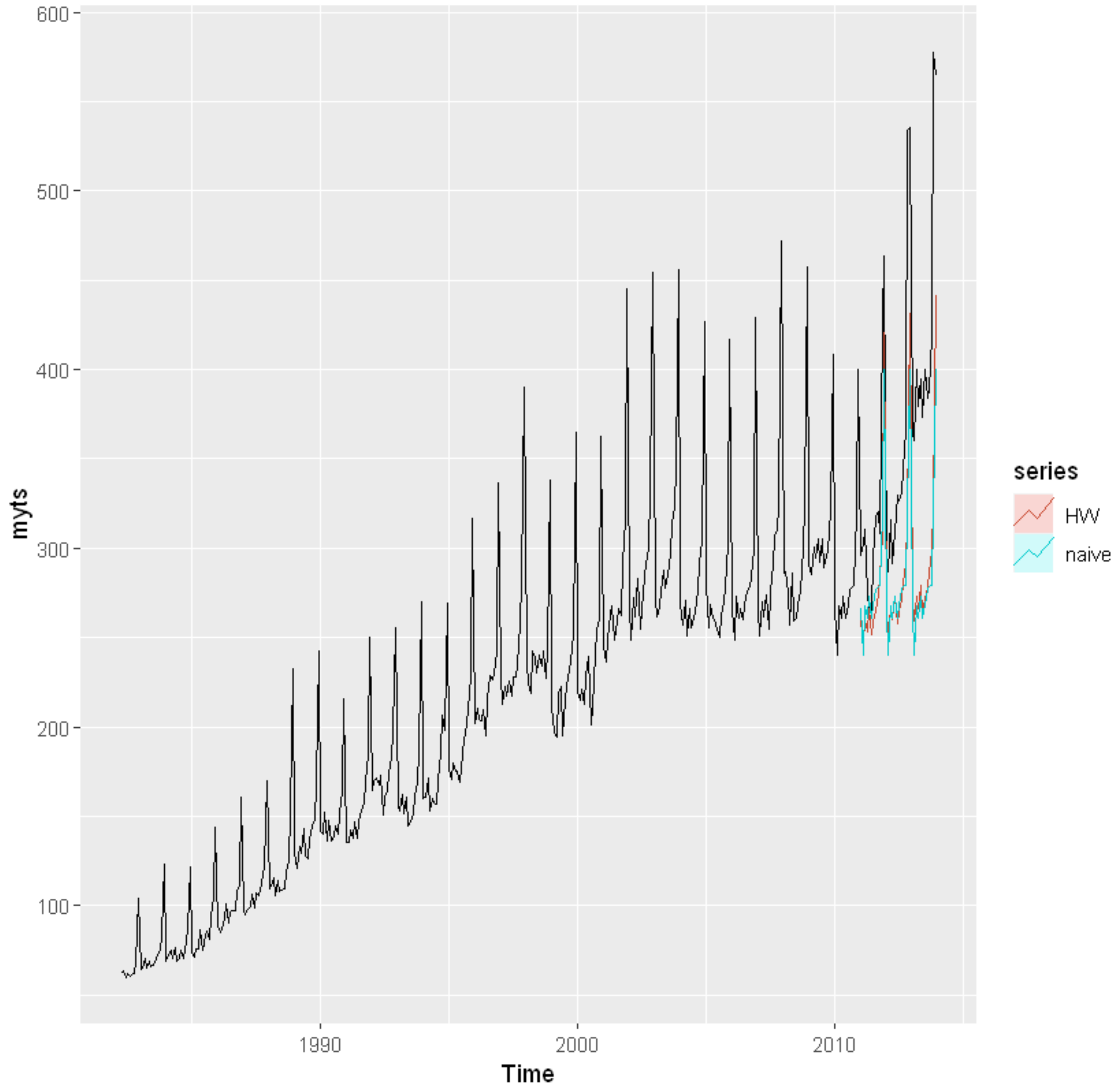
```
ts_train <- window(myts, end=c(2010, 12))
```

В [57]:

```
hw3 <- hw(ts_train, h=36, seasonal = 'multiplicative', damped=FALSE)
fcn <- snaive(ts_train, h=36)
```

B [58]:

```
autoplot(myts) +
autolayer(hw3, series="HW", PI=FALSE) +
autolayer(fcn, series="naive", PI=FALSE)
```



В [59]:

```
test <- window(myts, start=c(2011,1))
```

В [60]:

```
sum(sqrt((test - hw3$mean)^2)) / length(test)
sum(sqrt((test - fcn$mean)^2)) / length(test)
```

78.3406836524533

82.0666666666667

9. For the same retail data, try an STL decomposition applied to the Box-Cox transformed series, followed by ETS on the seasonally adjusted data. How does that compare with your best previous forecasts on the test set?

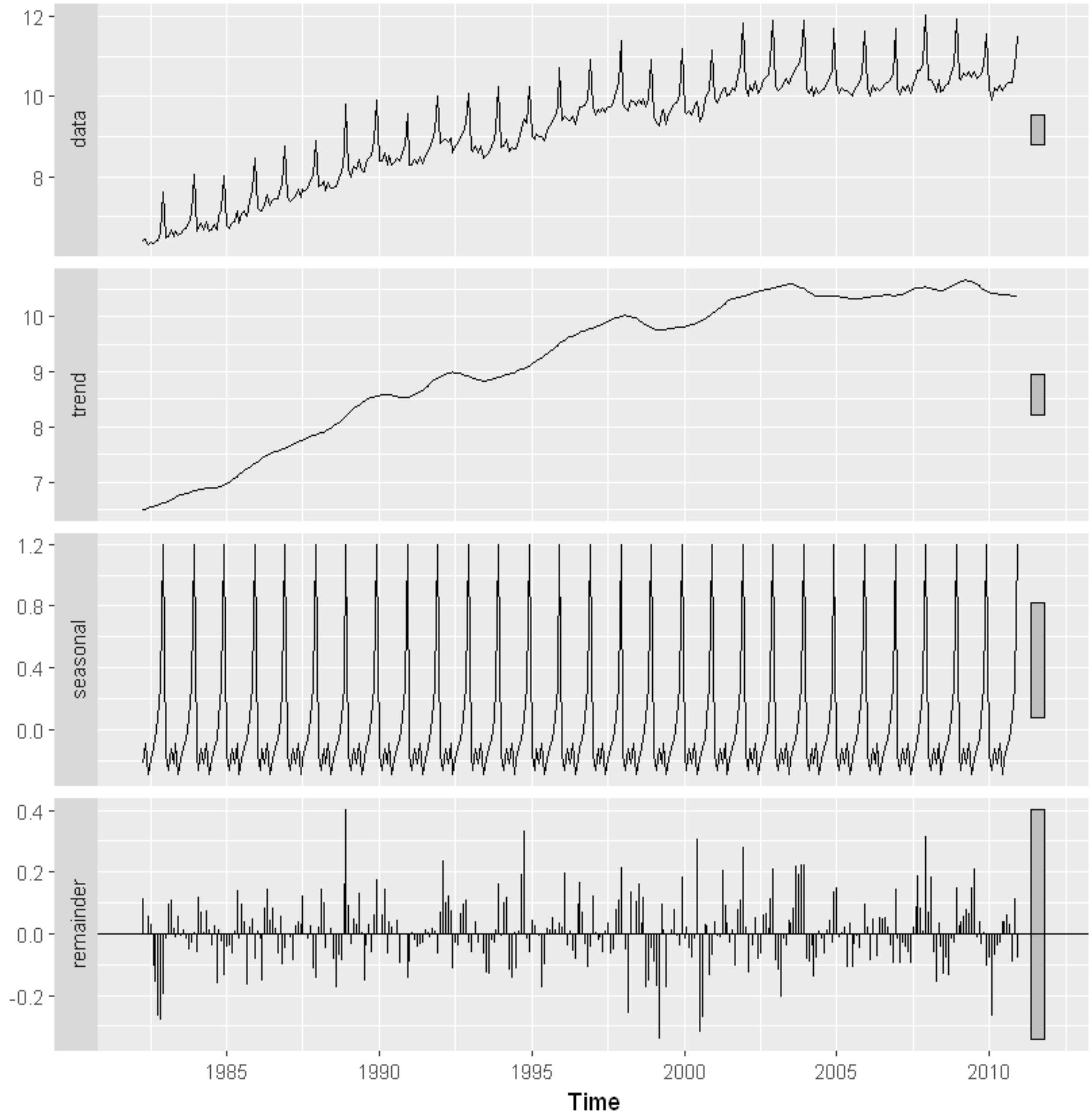
В [61]:

```
train <- ts(as.vector(myts), start=c(1982,4), end=c(2010,12), frequency = 12)
lambda <- BoxCox.lambda(train)
lambda
```

0.197968156308491

В [62]:

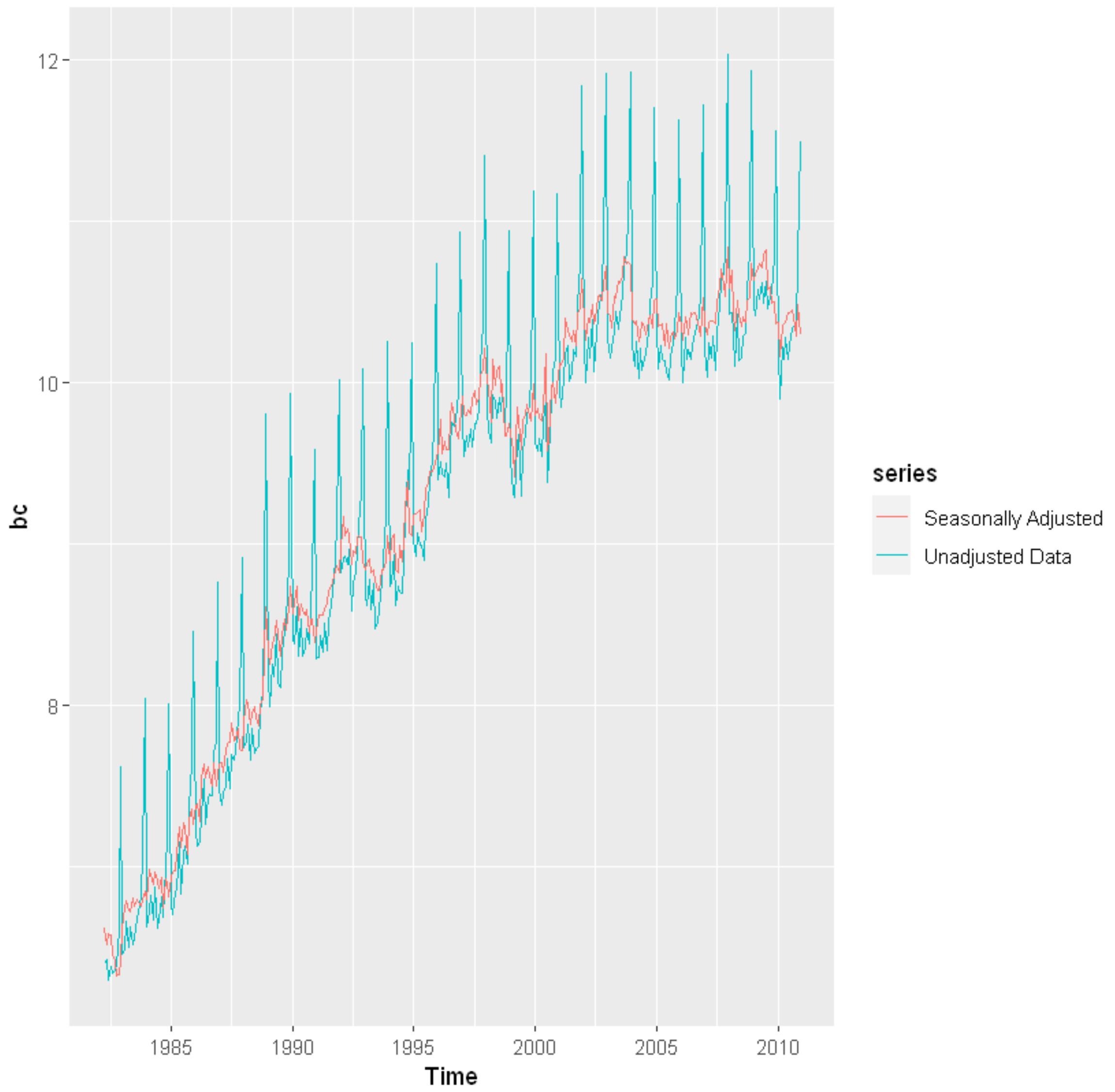
```
bc <- BoxCox(train, lambda)
bcstl <- stl(bc, s.window='periodic', robust=TRUE)
autoplot(bcstl)
```



В [63]:

```
bcsadj <- bc - bcstl$time.series[,'seasonal']

autoplot(bc, series='Unadjusted Data') +
  autolayer(bcsadj, series='Seasonally Adjusted')
```



В [64]:

```
fets <- ets(bcsadj)
summary(fets)
```

```
ETS(M,A,N)

Call:
 ets(y = bcsadj)

  Smoothing parameters:
    alpha = 0.6333
    beta  = 1e-04

  Initial states:
    l = 6.567
    b = 0.0134

  sigma:  0.0129

     AIC     AICc      BIC
543.5141 543.6911 562.7319

Training set error measures:
                       ME      RMSE        MAE          MPE      MAPE      MAS
E
Training set -0.003878286 0.1172707 0.0899321 -0.03866332 0.9882063 0.383223
1
                   ACF1
Training set 0.01864534
```

В [65]:

```
fcets <- forecast(fets, h=36)$mean
fcets <- InvBoxCox(fcets, lambda=lambda)
fcets
```

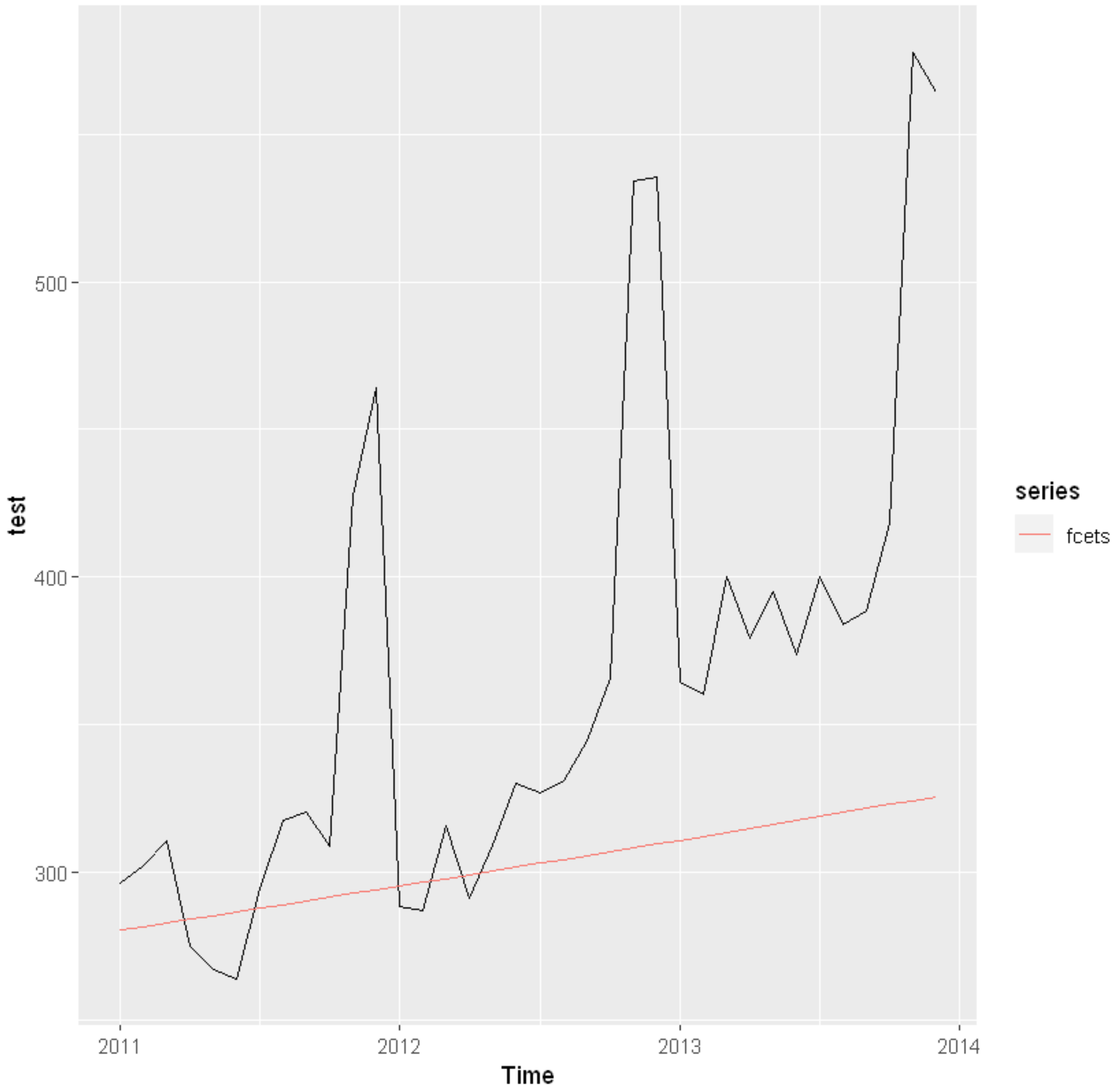
```
ERROR while rich displaying an object: Error in repr_matrix_generic(obj,
"\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matched by mult
iple actual arguments

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.ts(obj)
9. repr_ts_generic(obj, repr_markdown.matrix, ...)
10. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep
2011	280.4265	281.6456	282.8690	284.0966	285.3286	286.5648	287.8053	289.0500	290.2992
2012	295.3390	296.6098	297.8850	299.1647	300.4487	301.7371	303.0300	304.3273	305.6291
2013	310.8809	312.2051	313.5338	314.8670	316.2048	317.5472	318.8941	320.2456	321.6016

В [66]:

```
autoplot(test) +
autolayer(fcets)
```



В [67]:

```
sum(sqrt((test - fcets)^2)) / length(test)
```

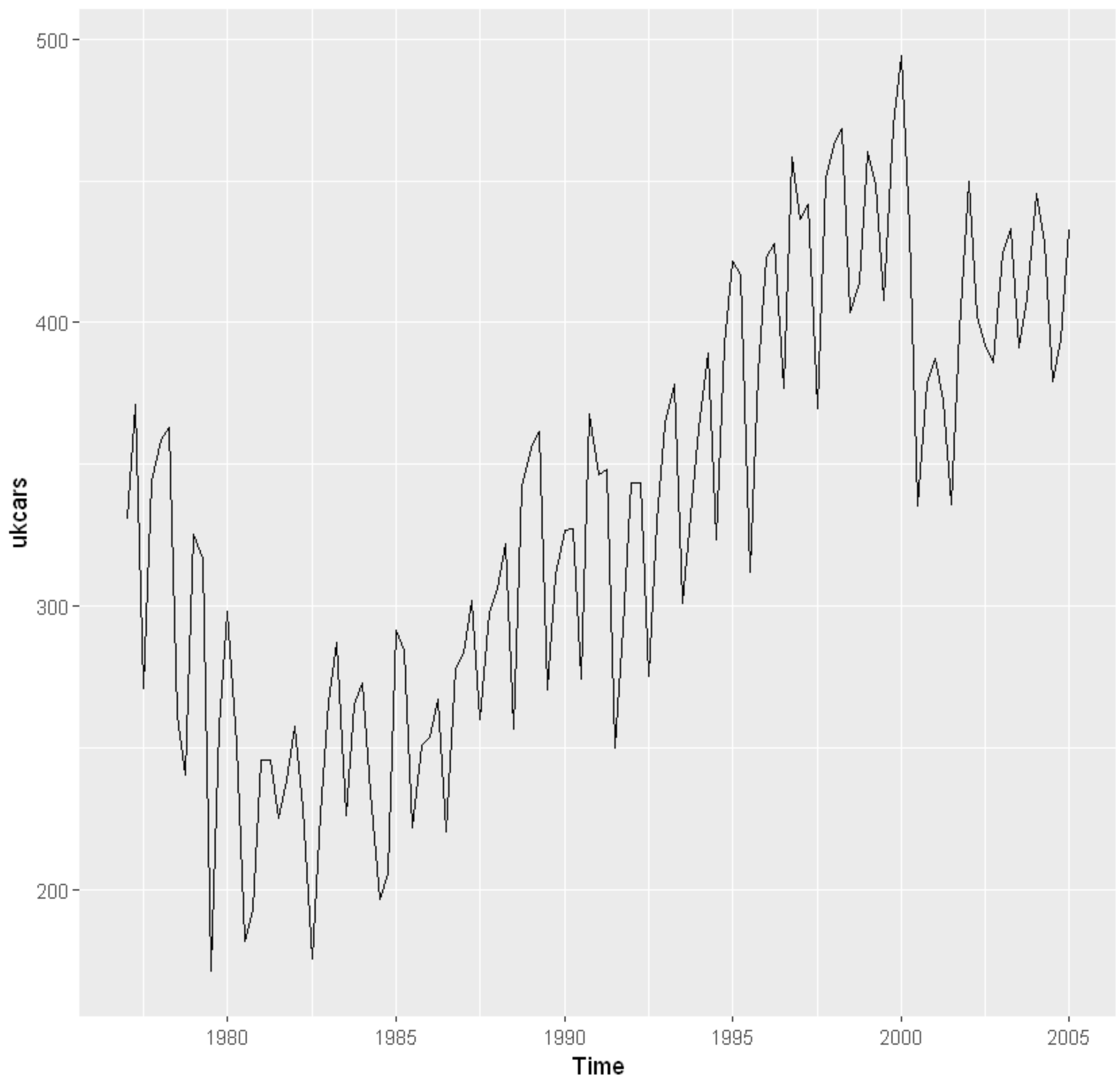
65.8001368355712

10. For this exercise, use the quarterly UK passenger vehicle production data from 1977Q1--2005Q1 (data set `ukcars`).

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

В [68]:

```
autoplot(ukcars)
```



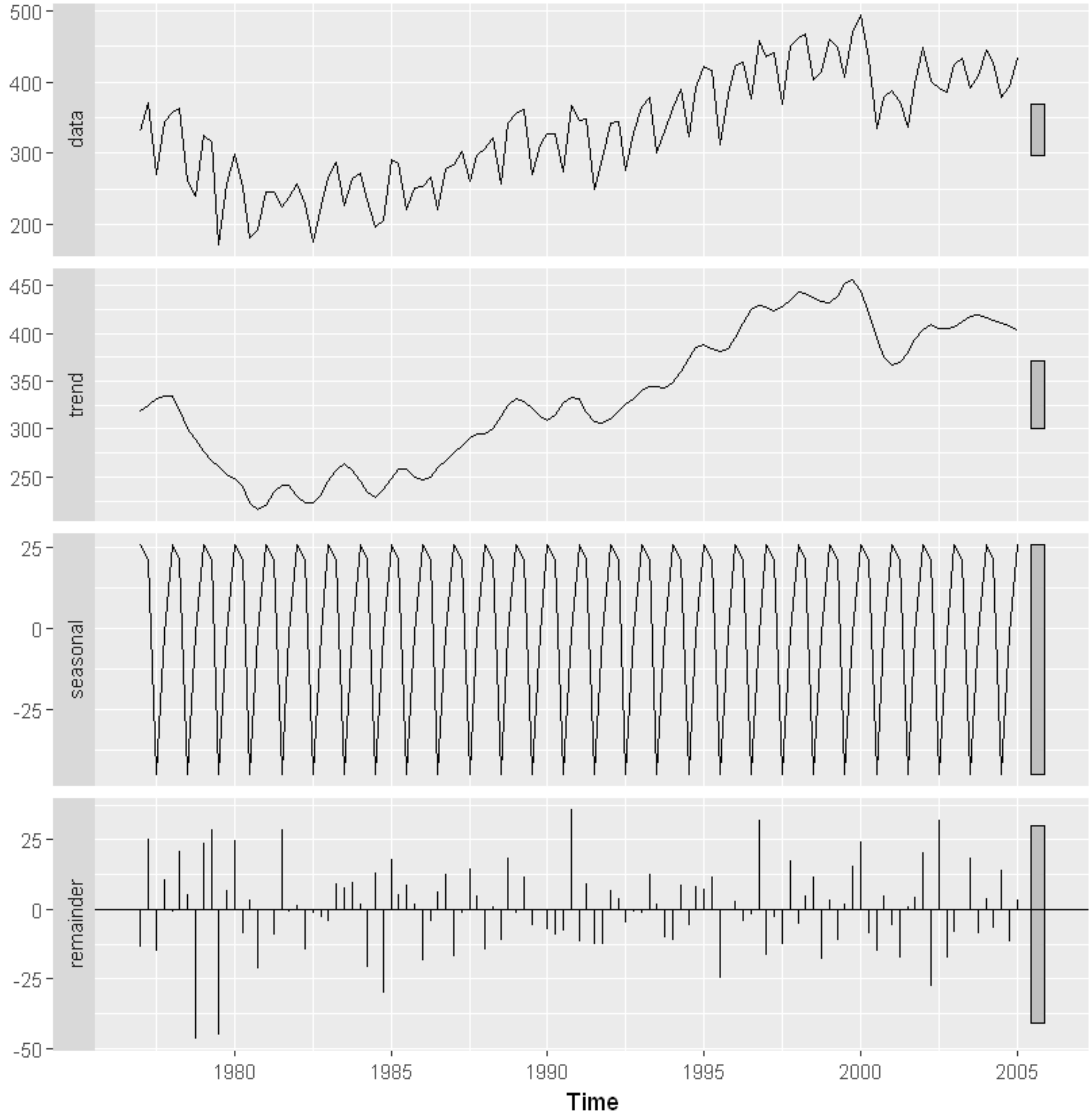
> b. Decompose the series using STL and obtain the seasonally adjusted data.

В [69]:

```
ukstl <- stl(ukcars, s.window = "periodic")
```

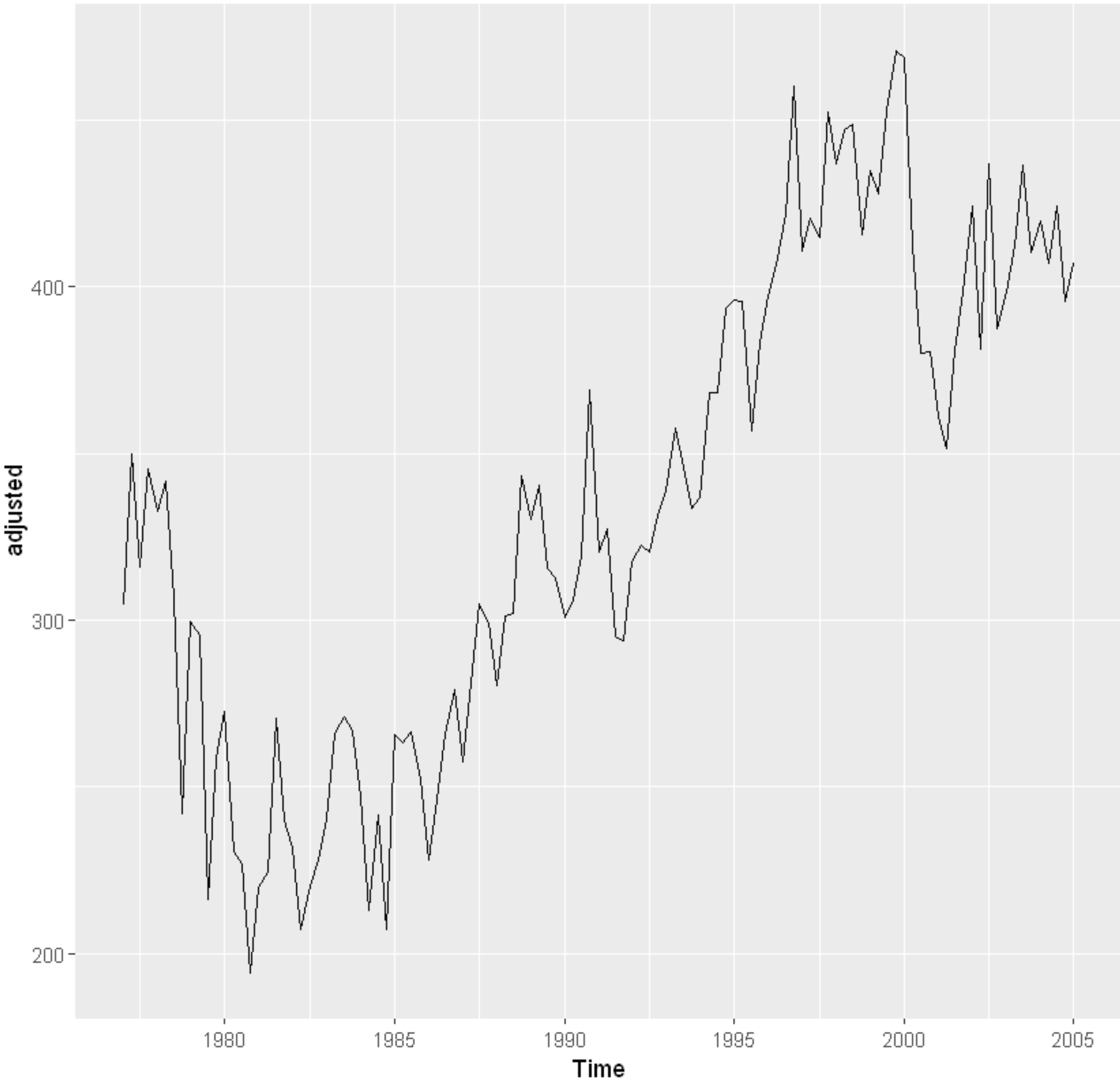
В [70]:

```
autoplot(ukstl)
```



В [71]:

```
adjusted <- seasadj(ukstl)
autoplot(adjusted)
```



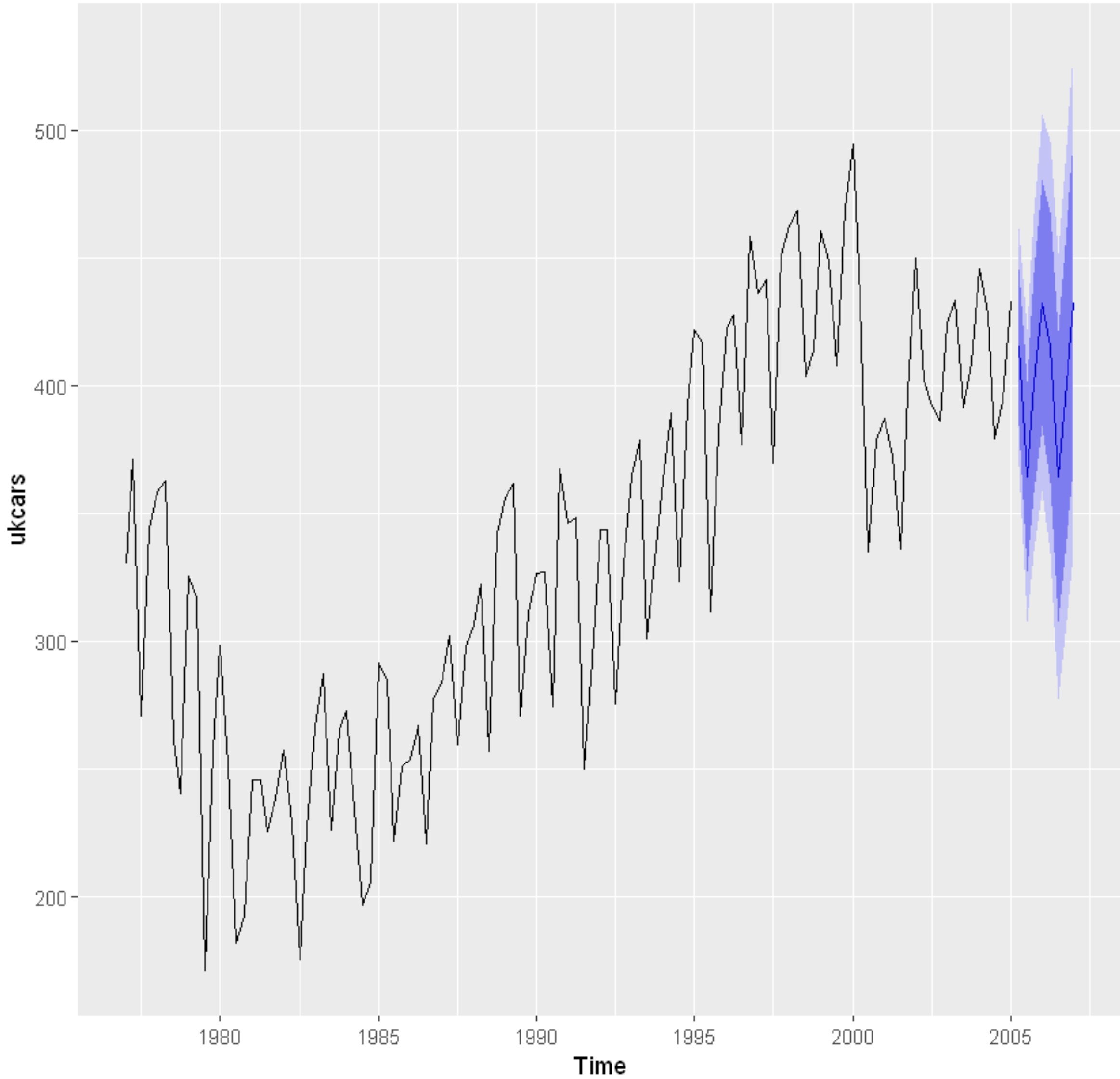
c. Forecast the next two years of the series using an additive damped trend method applied to the seasonally adjusted data. (This can be done in one step using `stlf` with arguments `etsmodel="AAN", damped=TRUE`.)

В [72]:

```
fcst_damped <- stlf(ukcars, etsmodel="AAN", damped=TRUE)
```

В [73]:

```
autoplot(ukcars) +
autolayer(fcst_damped)
```



В [74]:

```
fcst_damped
```

```
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2005 Q2       415.0289 384.4576 445.6001 368.2742 461.7836
2005 Q3       364.2543 326.8700 401.6386 307.0799 421.4287
2005 Q4       400.8059 357.6702 443.9416 334.8355 466.7762
2006 Q1       432.5663 384.3596 480.7731 358.8404 506.2922
2006 Q2       415.0250 362.2312 467.8189 334.2838 495.7663
2006 Q3       364.2507 307.2369 421.2646 277.0556 451.4459
2006 Q4       400.8026 339.8596 461.7456 307.5983 494.0069
2007 Q1       432.5633 367.9289 497.1976 333.7136 531.4130
```

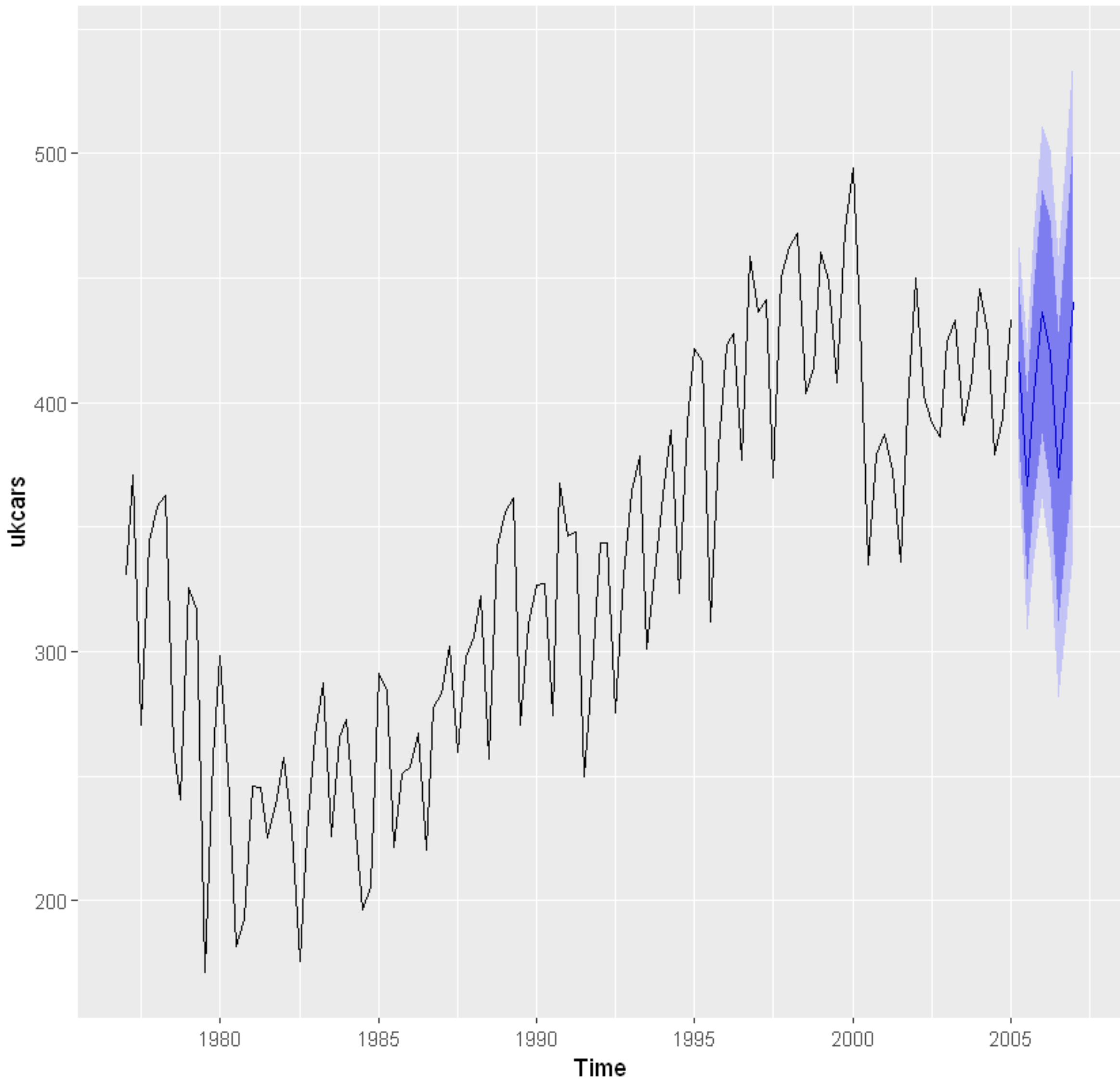
> d. Forecast the next two years of the series using Holt’s linear method applied to the seasonally adjusted data (as before but with `damped=FALSE`).

В [75]:

```
fcst <- stlf(ukcars, etsmodel="AAN", damped=FALSE)
```

В [76]:

```
autoplot(ukcars) +
autolayer(fcst)
```



В [77]:

```
fcst
```

```
        Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
2005 Q2       416.1915 385.7950 446.5881 369.7040 462.6791
2005 Q3       366.3343 328.7907 403.8780 308.9163 423.7524
2005 Q4       403.8032 360.2690 447.3375 337.2233 470.3831
2006 Q1       436.4809 387.6847 485.2771 361.8535 511.1083
2006 Q2       419.8568 366.3120 473.4016 337.9671 501.7465
2006 Q3       369.9996 312.0932 427.9060 281.4394 458.5598
2006 Q4       407.4685 345.5056 469.4313 312.7045 502.2325
2007 Q1       440.1461 374.3755 505.9167 339.5587 540.7336
```

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

В [78]:

```
ukets <- ets(ukcars)
```

В [79]:

```
ukets
```

```
ETS(A,N,A)

Call:
 ets(y = ukcars)

  Smoothing parameters:
    alpha = 0.6199
    gamma = 1e-04

  Initial states:
    l = 314.2568
    s = -1.7579 -44.9601 21.1956 25.5223

  sigma:  25.9302

     AIC     AICc      BIC
1277.752 1278.819 1296.844
```

> f. Compare the RMSE of the ETS model with the RMSE of the models you obtained using STL decompositions. Which gives the better in-sample fits?

В [80]:

```
accuracy(fcst_damped)
accuracy(fcst)
accuracy(ukets)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	1.551267	23.32113	18.48987	0.04121971	6.042764	0.602576	0.02262668

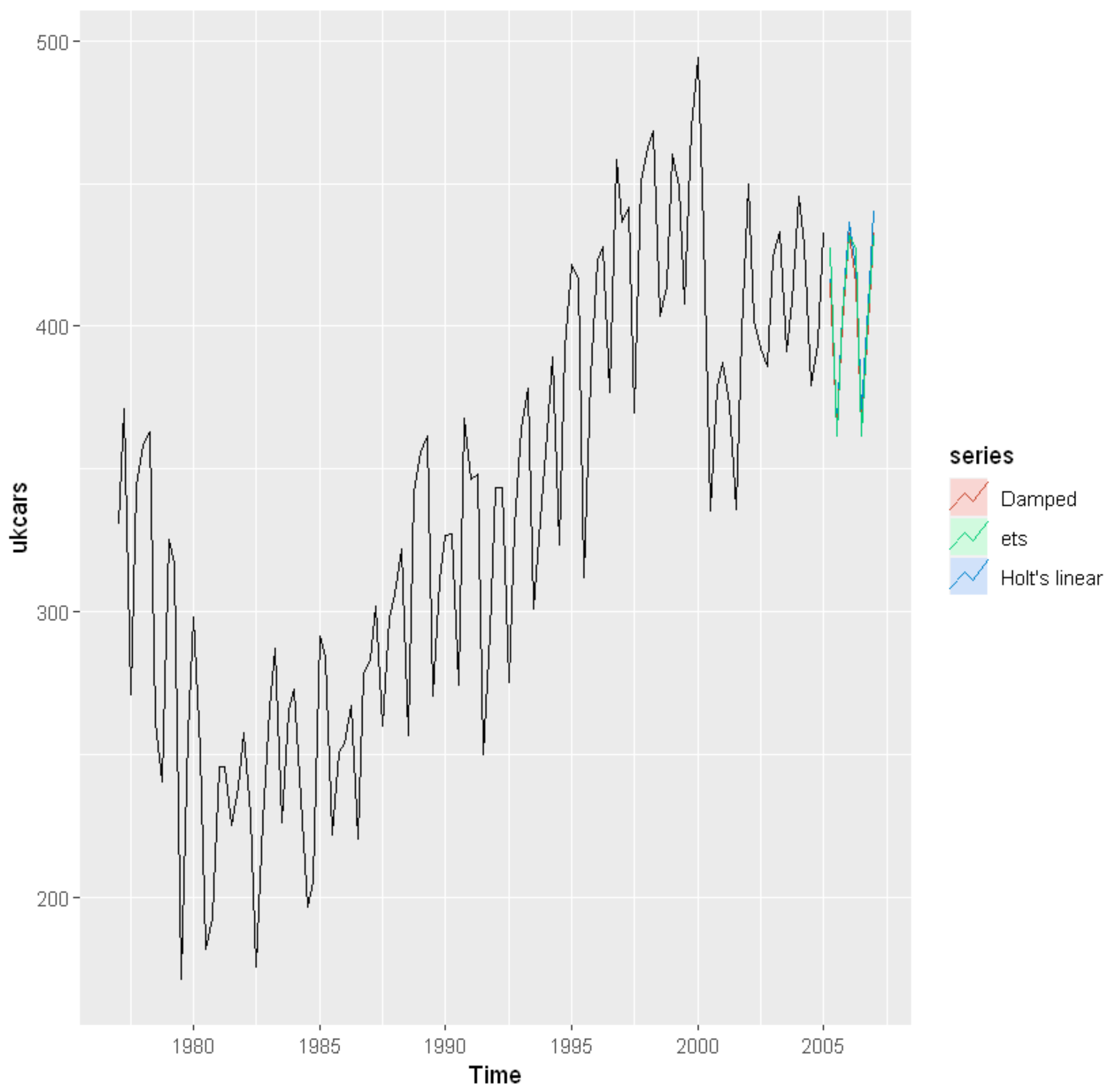
	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	-0.3412727	23.295	18.1605	-0.5970778	5.98018	0.5918418	0.02103582

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1
Training set	1.313884	25.23244	20.17907	-0.1570979	6.629003	0.6576259	0.02573334

g. Compare the forecasts from the three approaches? Which seems most reasonable?

В [81]:

```
autoplot(ukcars) +
autolayer(fcst, PI=FALSE, series="Holt's linear") +
autolayer(fcst_damped, PI=FALSE, series="Damped") +
autolayer(forecast(ukets), PI=FALSE, series="ets")
```



h. Check the residuals of your preferred model.

В [82]:

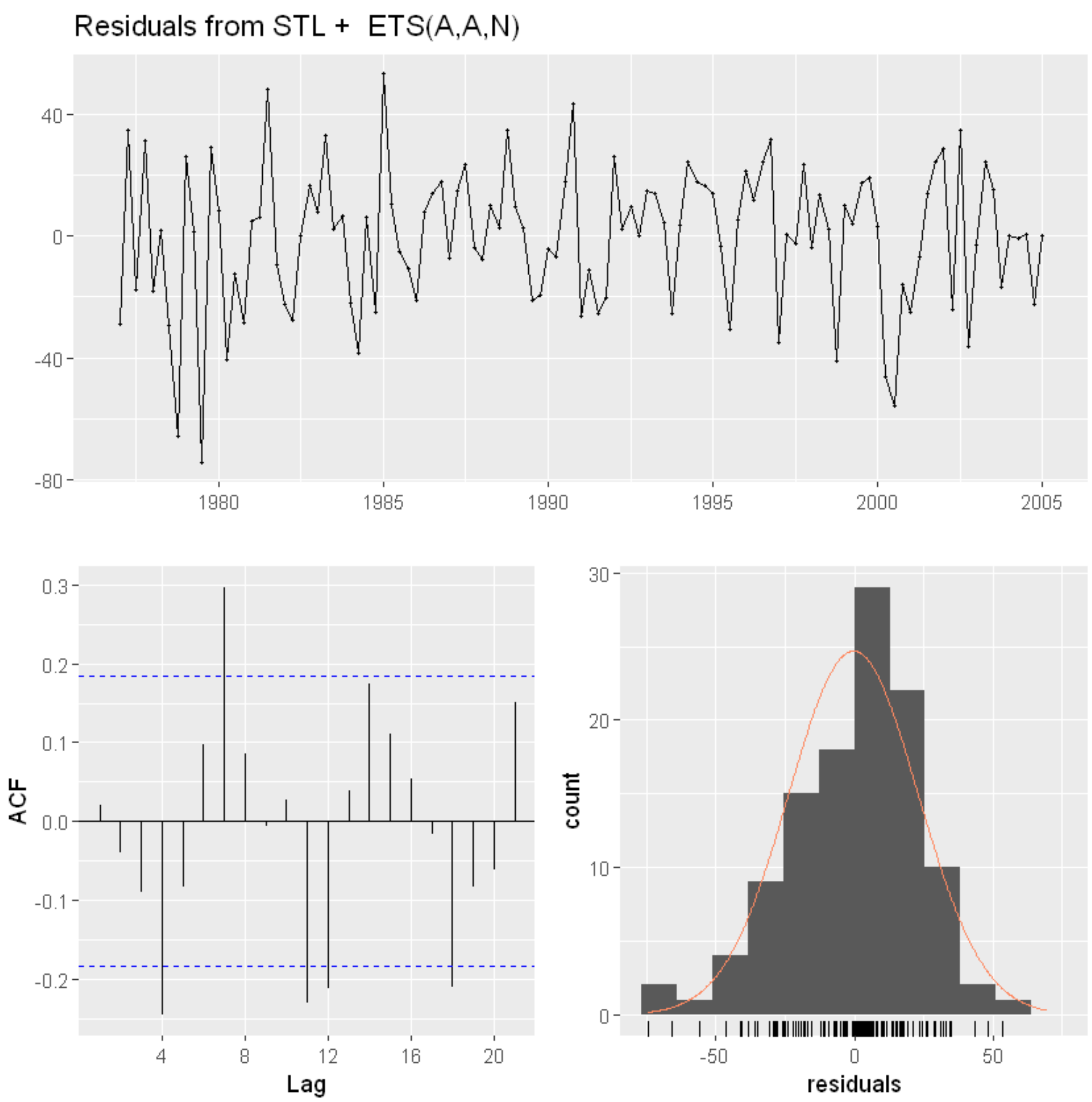
```
checkresiduals(fcst)
```

```
Warning message in checkresiduals(fcst):
"The fitted degrees of freedom is based on the model used for the seasonally
adjusted data."


        Ljung-Box test

data:  Residuals from STL +  ETS(A,A,N)
Q* = 22.061, df = 4, p-value = 0.0001949

Model df: 4.   Total lags used: 8
```



В [83]:

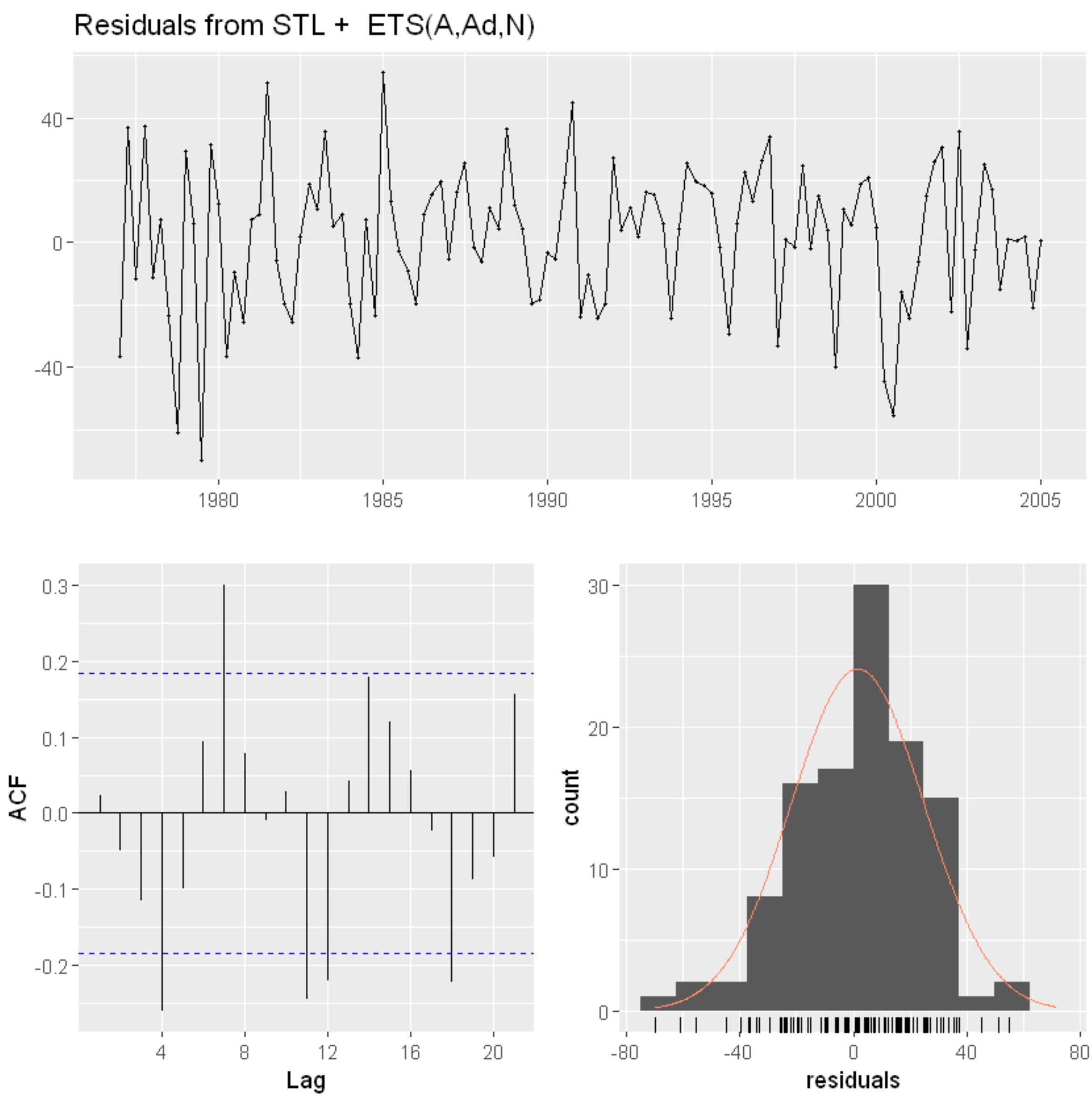
```
checkresiduals(fcst_damped)
```

```
Warning message in checkresiduals(fcst_damped):
"The fitted degrees of freedom is based on the model used for the seasonally
adjusted data."


	Ljung-Box test

data:  Residuals from STL +  ETS(A,Ad,N)
Q* = 24.138, df = 3, p-value = 2.337e-05

Model df: 5.   Total lags used: 8
```



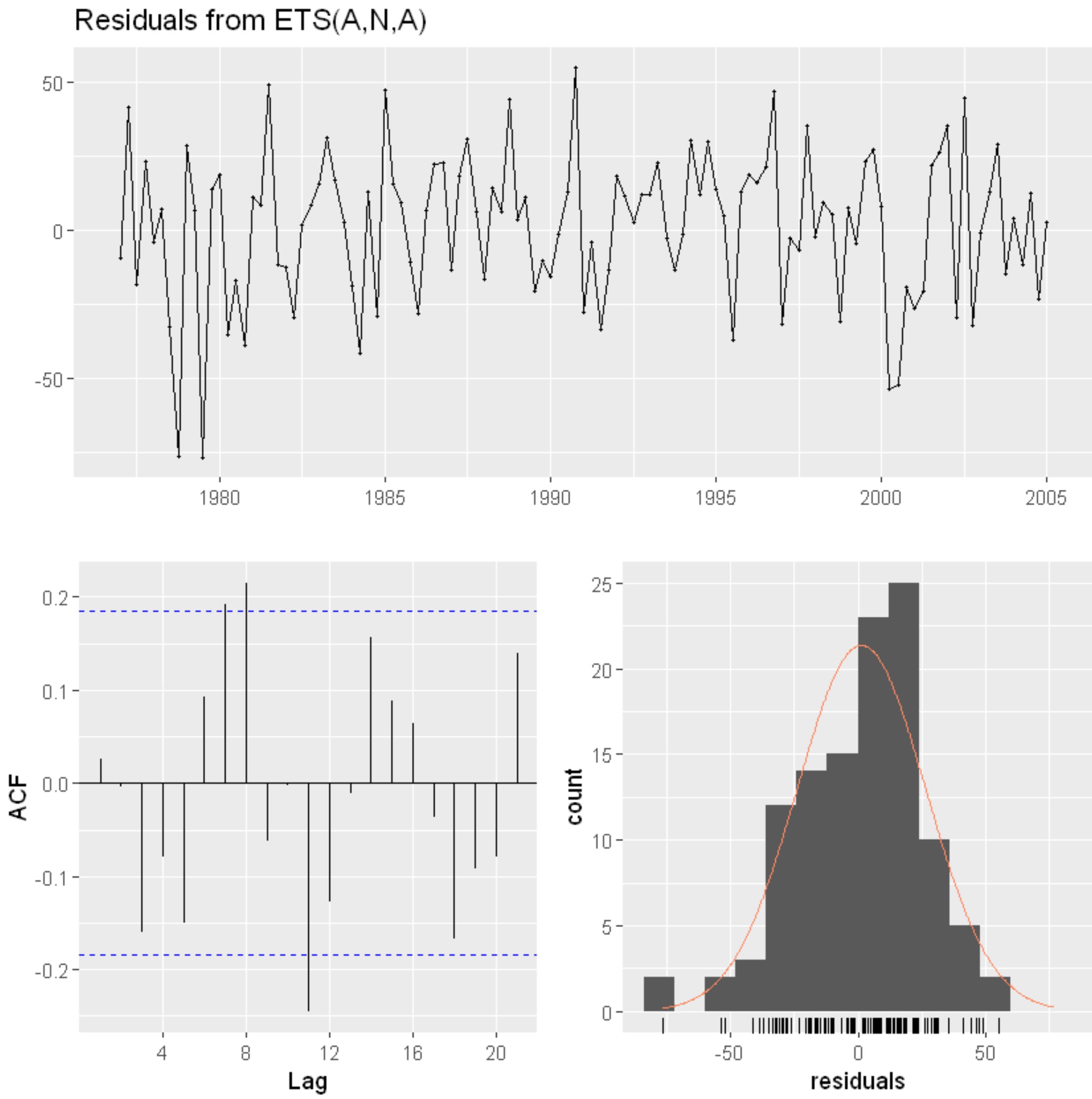
В [84]:

```
checkresiduals(ukets)
```

```
        Ljung-Box test

data:  Residuals from ETS(A,N,A)
Q* = 18.324, df = 3, p-value = 0.0003772

Model df: 6.   Total lags used: 9
```

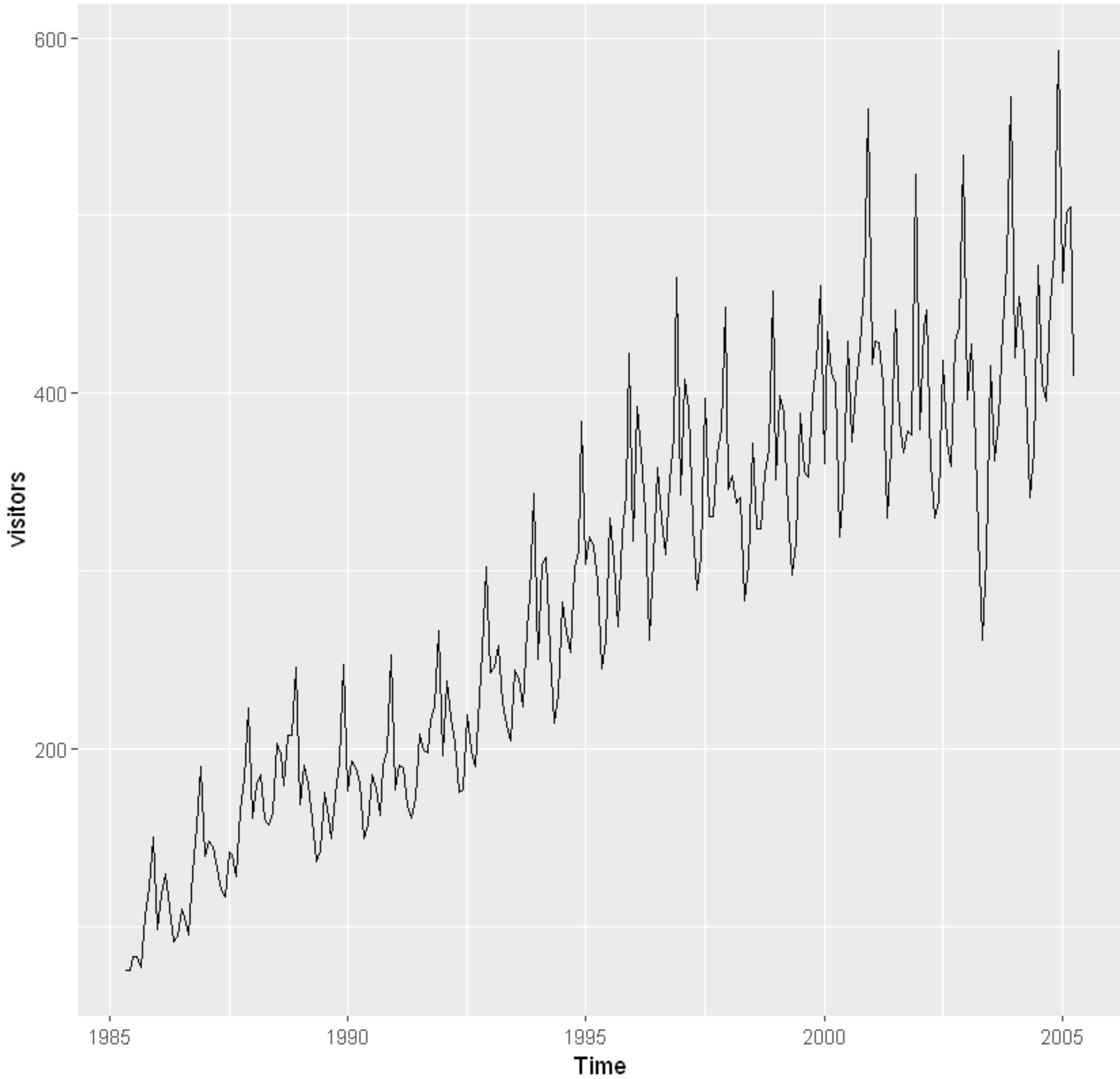


11. For this exercise use data set `visitors` , the monthly Australian short-term overseas visitors data, May 1985–April 2005.

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

В [85]:

```
autoplot(visitors)
```



b. Split your data into a training set and a test set comprising the last two years of available data. Forecast the test set using Holt-Winters' multiplicative method.

В [86]:

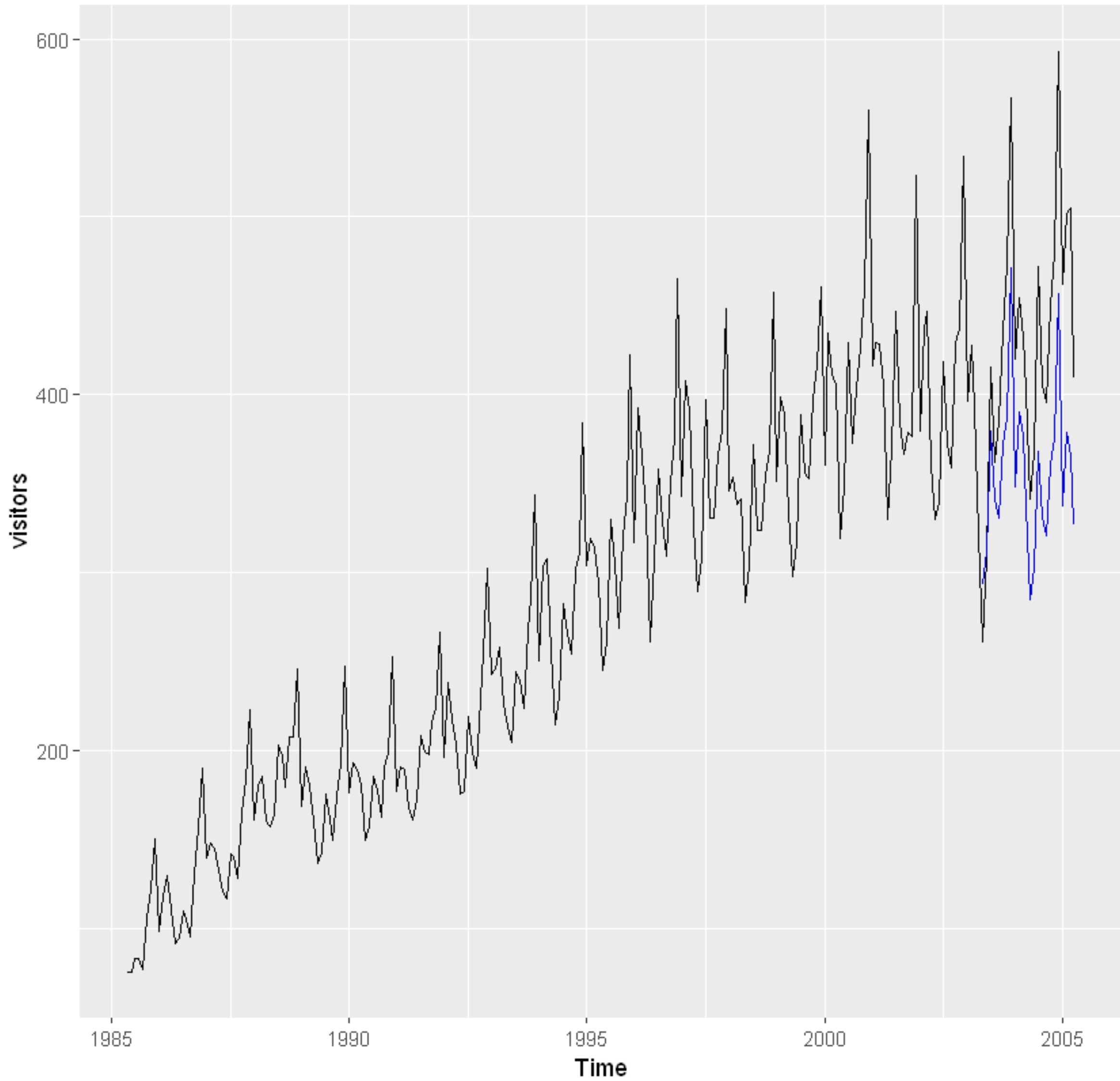
```
vtrain <- window(visitors, end=end(visitors)-c(2,0))
vtest <- window(visitors, start=end(visitors)-c(2,0))
```

В [87]:

```
vfcst <- hw(vtrain, h=24, seasonal = "multiplicative")
```

В [88]:

```
autoplot(visitors) +
autolayer(vfcst, PI=FALSE)
```



c. Why is multiplicative seasonality necessary here?

d. Forecast the two-year test set using each of the following methods:

```
i) an ETS model;
```

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

```
iii) a seasonal naïve method;
```

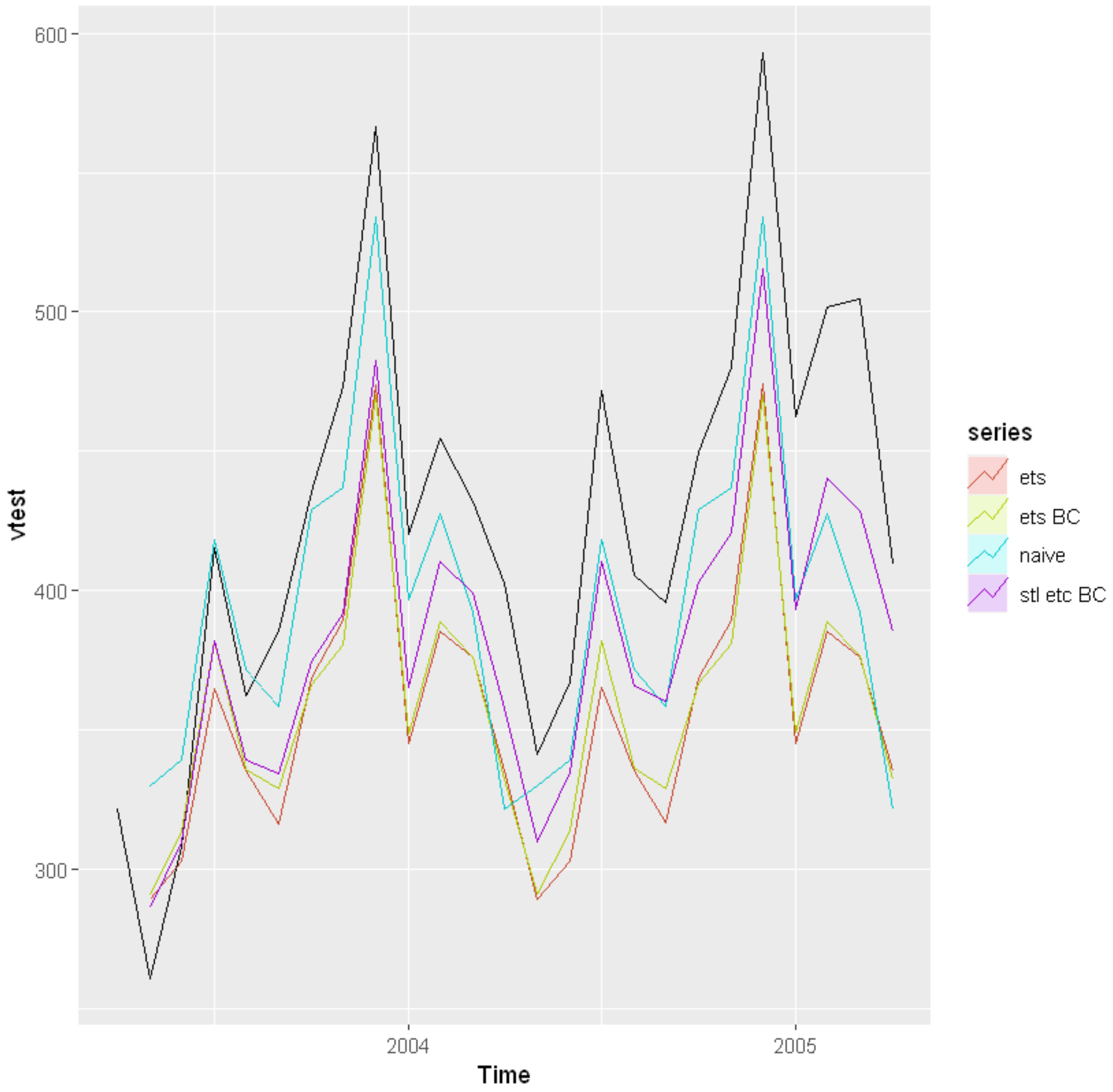
```
iv) an STL decomposition applied to the Box-Cox transformed data foll
owed by an ETS model applied to the seasonally adjusted (transformed)
data.
```

В [89]:

```
fets <- forecast(ets(vtrain), h=24)
fetsbc <- forecast(ets(vtrain, lambda = BoxCox.lambda(vtrain)), h=24)
fnaive <- snaive(vtrain, h=24)
fstlets <- stlf(vtrain, lambda = BoxCox.lambda(vtrain))
```

В [90]:

```
autoplot(vtest) +
autolayer(fets, series="ets", PI=FALSE) +
autolayer(fetsbc, series="ets BC", PI=FALSE) +
autolayer(fnaive, series="naive", PI=FALSE) +
autolayer(fstlets, series="stl etc BC", PI=FALSE)
```



e. Which method gives the best forecasts? Does it pass the residual tests?

В [91]:

```
sum(sqrt((vtest - fets$mean)^2)) / length(vtest)
sum(sqrt((vtest - fetsbc$mean)^2)) / length(vtest)
sum(sqrt((vtest - fnaive$mean)^2)) / length(vtest)
sum(sqrt((vtest - fstlets$mean)^2)) / length(vtest)
```

71.570734759382

69.51925582919

40.556

46.0477875850091

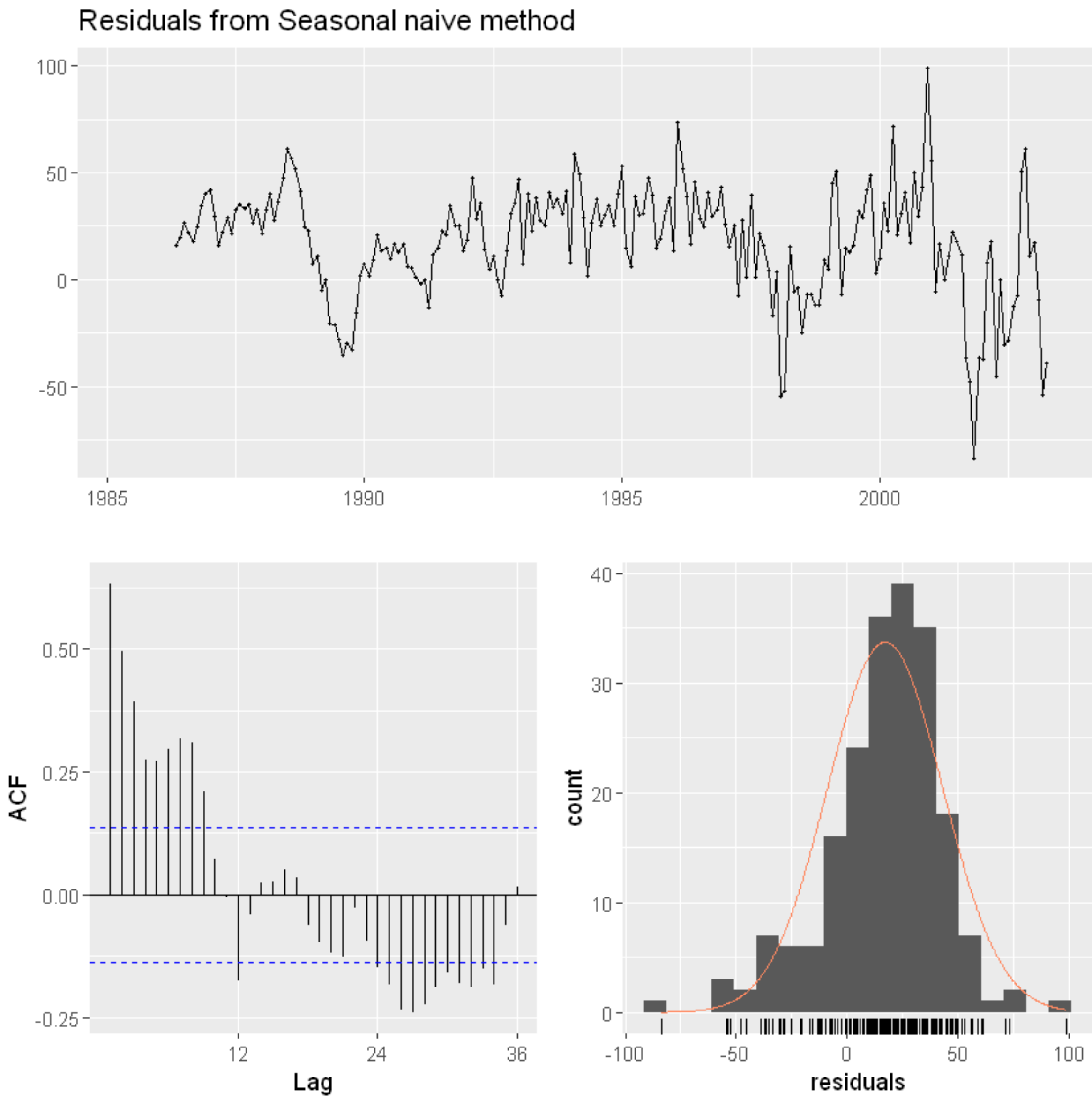
В [92]:

```
checkresiduals(fnaive)
```

```
	Ljung-Box test

data:  Residuals from Seasonal naive method
Q* = 295.02, df = 24, p-value < 2.2e-16

Model df: 0.   Total lags used: 24
```



f. Compare the same four methods using time series cross-validation with the `tsCV` function instead of using a training and test set. Do you come to the same conclusions?

В [93]:

```
fun1 <- function(y, h){
    forecast(ets(y), h=h)
}
fun2 <- function(y, h){
    forecast(ets(y, lambda=BoxCox.lambda(y)), h=h)
}
```

В [96]:

```
e1 <- tsCV(visitors, fun1)
```

В [97]:

```
e2 <- tsCV(visitors, fun2)
```

В [98]:

```
e3 <- tsCV(visitors, snaive)
```

В [99]:

```
e4 <- tsCV(visitors, stlf, lambda=BoxCox.lambda(visitors))
```

В [102]:

```
e <- matrix(NA, ncol=4, nrow=length(visitors))
```

В [103]:

```
e[, 1] <- e1
e[, 2] <- e2
e[, 3] <- e3
e[, 4] <- e4
```

В [104]:

```
colMeans(e^2, na.rm = T)
```

343.355194360265 355.865197478754 1074.97066945607 282.587317068726

12.

The `fets()` function below returns ETS forecasts.

```
fets <- function(y, h) {
  forecast(ets(y), h = h)
}
```

> a. Apply `tsCV()` for a forecast horizon of $h = 4$, for both ETS and seasonal naïve methods to

> the `qcement` data, (Hint: use the newly created `fets()` and the existing `snaive()` functions as your forecast function arguments.)

B [105]:

```
fets <- function(y, h) {
  forecast(ets(y), h = h)
}
```

B [106]:

```
e1 <- tsCV(qcement, fets, h=4)
```

B [107]:

```
e2 <- tsCV(qcement, snaive, h=4)
```

B [108]:

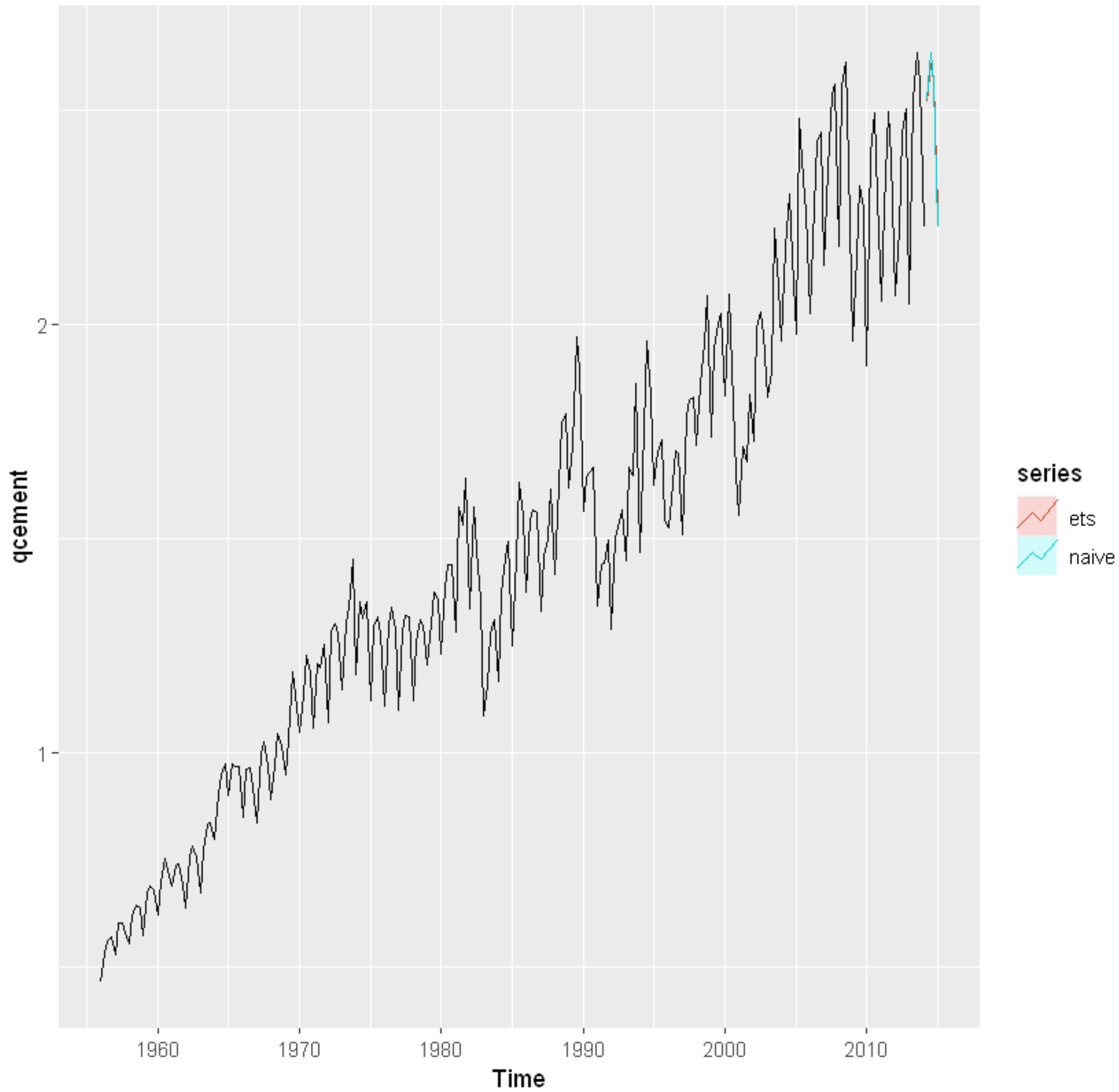
```
colMeans(e1^2, na.rm = T)
colMeans(e2^2, na.rm = T)
```

h=1
0.00695951102352166
h=2
0.0105922770179574
h=3
0.0141177205070902
h=4
0.0184510736000398

h=1
0.0177924267241379
h=2
0.0178281304347826
h=3
0.0179635175438596
h=4
0.0181065

В [109]:

```
autoplot(qcement) +
autolayer(fets(qcement, h=4), series="ets", PI=FALSE) +
autolayer(snaive(qcement, h=4), series="naive", PI=FALSE)
```



13. Compare `ets`, `snaive` and `stlf` on the following six time series. For `stlf`, you might need to use a Box-Cox transformation. Use a test set of three years to

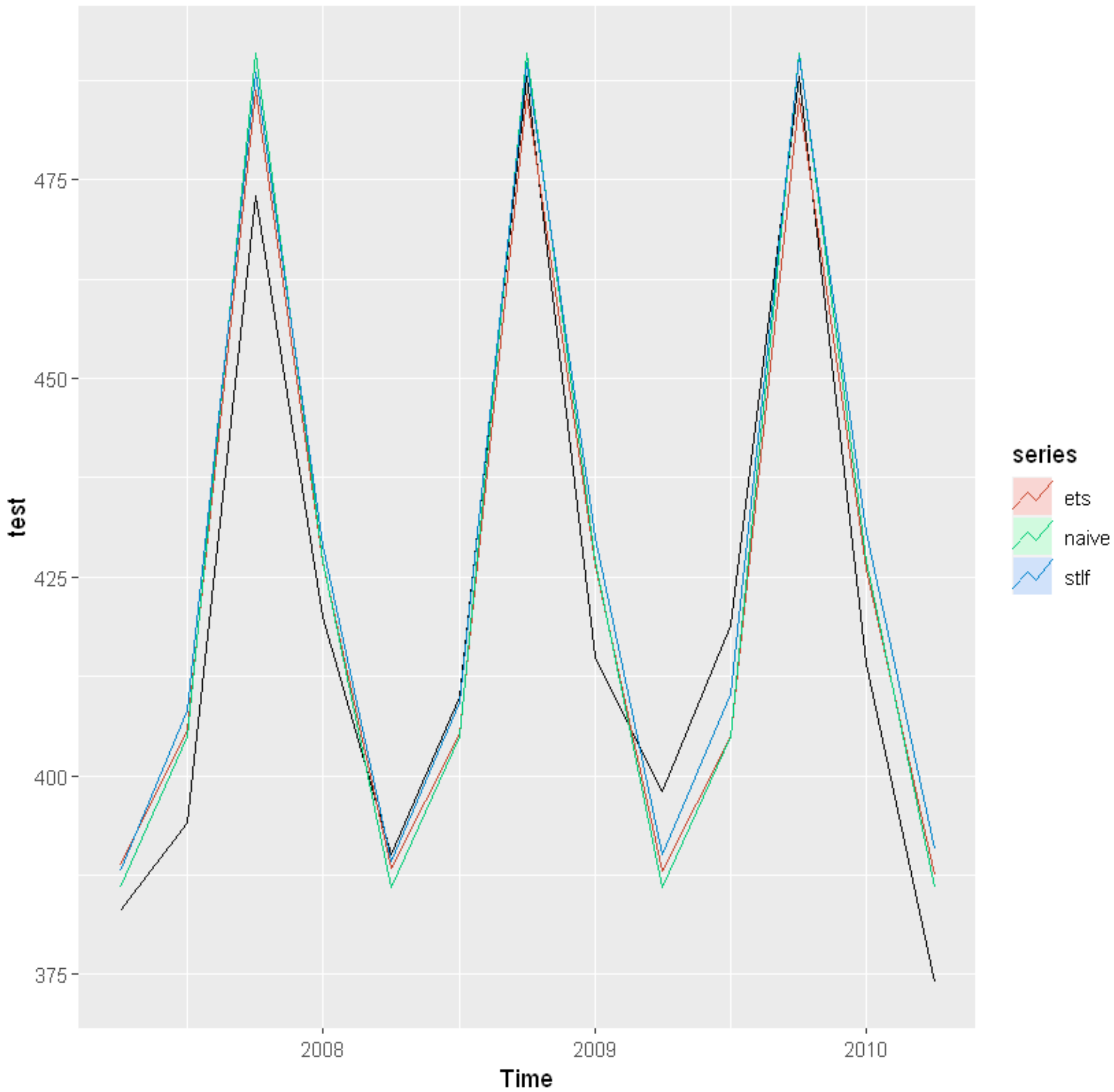
decide what gives the best forecasts. `ausbeer`, `bricksq`, `dole`, `a10`, `h02`, `usmelec`.

В [121]:

```
train <- window(ausbeer, end=end(ausbeer)-c(3,1))
test <- window(ausbeer, start=end(ausbeer)-c(3,0))
fc1 <- forecast(ets(train), h=13)
fc2 <- snaive(train, h=13)
fc3 <- stlf(train, lambda=BoxCox.lambda(train), h=13)
```

В [122]:

```
autoplot(test) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE) +
autolayer(fc3, series="stlf", PI=FALSE)
```



В [123]:

```
accuracy(fc1, test)
accuracy(fc2, test)
accuracy(fc3, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil'
Training set	-0.3277264	15.814416	12.011993	-0.05972334	2.871560	0.7556841	-0.1953342	
Test set	-3.0004013	9.565905	8.477236	-0.74556768	2.054088	0.5333097	0.3932191	0.1892

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's
Training set	3.368159	19.71778	15.89552	0.9127974	3.786236	1.0000000	-0.0002469795	I
Test set	-3.615385	10.22441	9.00000	-0.8396570	2.162967	0.5661972	0.3517322372	0.20493

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.6182399	13.6352	10.479361	0.1464515	2.510741	0.6592650	-0.1620239	NA
Test set	-6.0188687	10.6846	8.827108	-1.4519045	2.140312	0.5553204	0.3225159	0.2081258

В [125]:

```
bricksq
```

```
ERROR while rich displaying an object: Error in repr_matrix_generic(obj,
"\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matched by mult
iple actual arguments

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.ts(obj)
9. repr_ts_generic(obj, repr_markdown.matrix, ...)
10. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
```

	Qtr1	Qtr2	Qtr3	Qtr4
1956	189	204	208	197
1957	187	214	227	223
1958	199	229	249	234
1959	208	253	267	255
1960	242	268	290	277
1961	241	253	265	236
1962	229	265	275	258
1963	231	263	308	313
1964	293	328	349	340
1965	309	349	366	340
1966	302	350	362	337
1967	326	358	359	357
1968	341	380	404	409
1969	383	417	454	428
1970	386	428	434	417
1971	385	433	453	436

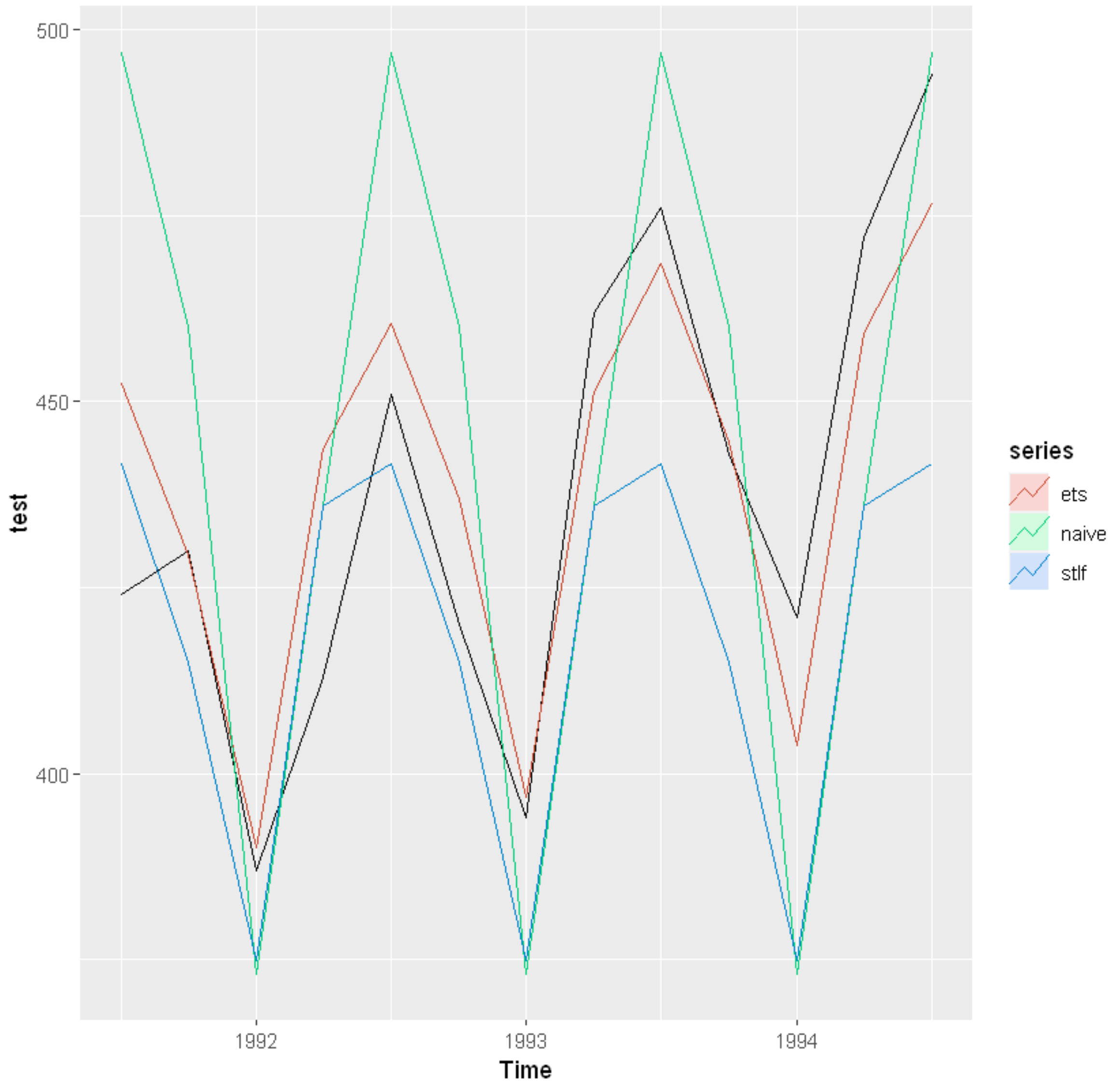
	Qtr1	Qtr2	Qtr3	Qtr4
1972	399	461	476	477
1973	452	461	534	516
1974	478	526	518	417
1975	340	437	459	449
1976	424	501	540	533
1977	457	513	522	478
1978	421	487	470	482
1979	458	526	573	563
1980	513	551	589	564
1981	519	581	581	578
1982	500	560	512	412
1983	303	409	420	413
1984	400	469	482	484
1985	447	507	533	503
1986	443	503	505	443
1987	415	485	495	458
1988	427	519	555	539
1989	511	572	570	526
1990	472	524	497	460
1991	373	436	424	430
1992	387	413	451	420
1993	394	462	476	443
1994	421	472	494	

В [126]:

```
train <- window(bricksq, end=end(bricksq)-c(3,1))
test <- window(bricksq, start=end(bricksq)-c(3,0))
fc1 <- forecast(ets(train), h=13)
fc2 <- snaive(train, h=13)
fc3 <- stlf(train, lambda=BoxCox.lambda(train), h=13)
```

В [127]:

```
autoplot(test) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE) +
autolayer(fc3, series="stlf", PI=FALSE)
```



В [128]:

```
accuracy(fc1, test)
accuracy(fc2, test)
accuracy(fc3, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.3488635	21.78068	15.44195	-0.1147142	3.800434	0.4228153	0.1788342	NA
Test set	-2.0223864	15.37817	12.23429	-0.5894459	2.803553	0.3349866	0.3270557	0.3889101

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	7.014493	50.05229	36.52174	1.565362	8.940406	1.0000000	0.8044755	NA
Test set	-8.307692	35.22019	30.61538	-1.902989	7.072799	0.8382784	0.2844705	0.8266998

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	1.400714	21.41328	15.39837	0.3616624	3.731307	0.4216221	0.1883923	NA
Test set	18.732439	28.44946	24.97868	4.1172636	5.612660	0.6839399	0.4221968	0.7767417

В [129]:

```
dole
```

```
ERROR while rich displaying an object: Error in repr_matrix_generic(obj,
"\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matched by mult
iple actual arguments

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.ts(obj)
9. repr_ts_generic(obj, repr_markdown.matrix, ...)
10. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1956	4742	6128	6494	5379	6011	7003	9164	10333	9614	9545
1957	15711	13135	13077	15453	15995	18071	20291	20175	18975	17928
1958	29856	26879	24485	27745	27282	29418	29908	29278	26002	23826
1959	31486	28207	27669	27559	27924	27528	27410	24887	21904	19598
1960	23781	20020	18177	17732	16765	16310	14897	12940	11465	10364
1961	19257	20941	29718	35025	45110	57154	61499	62090	59561	48531
1962	56755	49740	45870	49136	47256	46324	45453	42333	36851	33952
1963	46178	40482	36394	37142	36424	38188	37174	31869	26575	21758
1964	28649	24226	21955	19937	18287	18129	17072	14924	12491	11160
1965	15831	13698	12111	12690	12585	12855	12137	10977	9993	9614
1966	19490	17611	16206	17560	18082	19482	19200	18918	17375	16122
1967	24911	21969	21956	20944	22200	24002	22951	20143	17187	15287
1968	26943	23735	20744	21090	21502	21275	19426	16798	14209	13357
1969	23460	19551	15898	16012	16054	15910	13873	11854	10138	9942
1970	17778	13854	12681	11328	11946	13043	12785	11937	11383	10282
1971	18337	16779	15504	17258	18264	19184	19453	18741	19087	18171

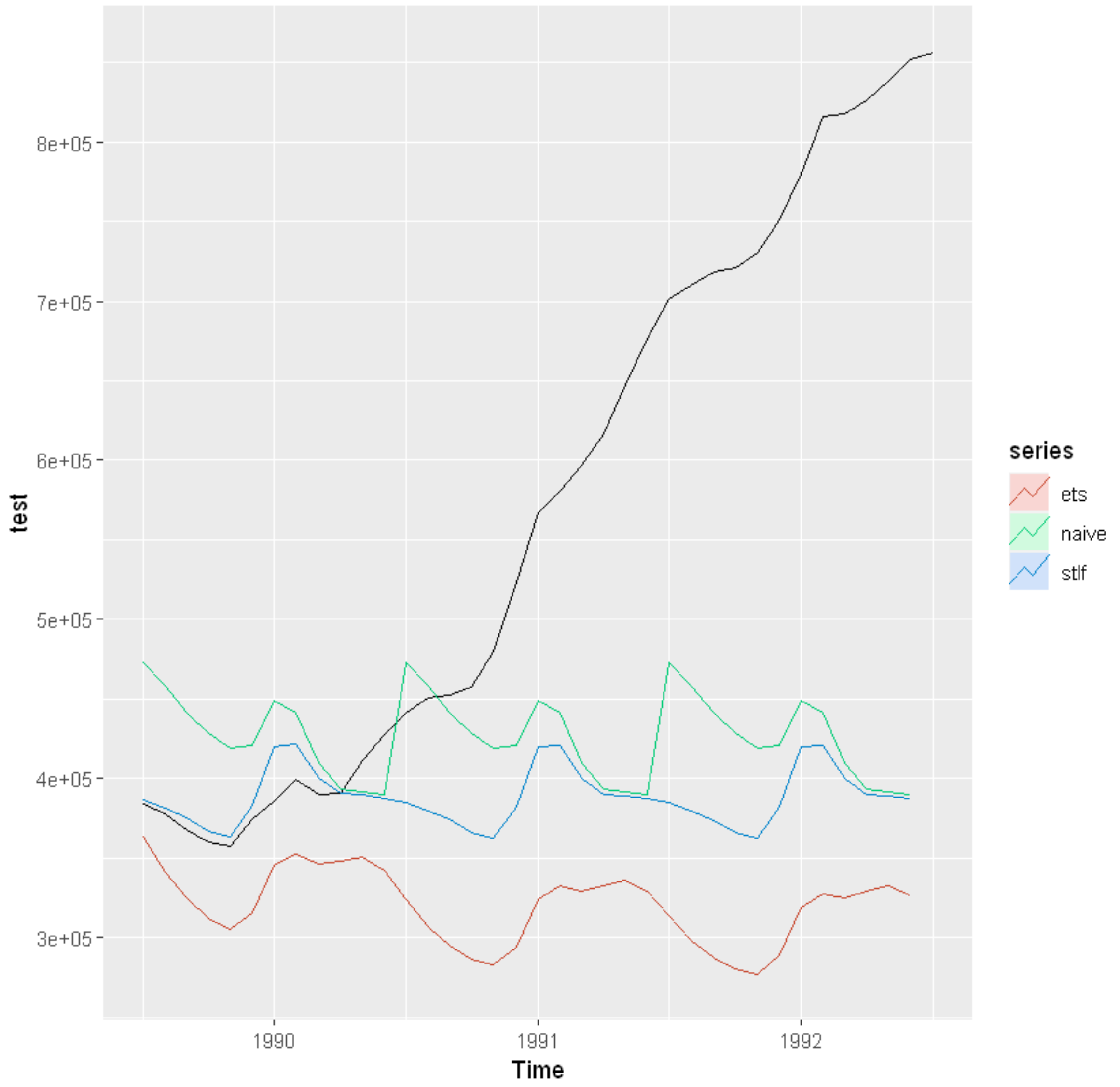
	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1972	37486	37303	37639	36536	35850	41581	42979	42490	37992	32454
1973	48622	39868	34511	37234	36675	37945	36593	31669	28682	25944
1974	46847	38315	32600	33349	30598	32009	37599	45999	54945	68394
1975	182260	184177	157547	168471	159020	160748	169631	170927	179898	176471
1976	248619	215342	192024	178765	182397	188423	197159	198648	195864	194125
1977	229415	245395	236383	226807	239984	250309	253809	254863	249551	254085
1978	269896	298455	290356	283308	272384	286091	290718	285424	284642	279874
1979	341877	357463	334400	332572	318905	311232	310000	303800	299566	286241
1980	334495	334265	316776	309300	308989	311232	313943	303555	290386	283822
1981	339700	347400	325500	315200	314900	314500	313700	318500	306000	299500
1982	351425	372288	358536	356004	375626	390664	404840	421856	446341	465959
1983	601931	632837	622819	622162	633272	635002	634020	622103	610379	599100
1984	674424	667059	626653	602100	600344	584506	580347	570553	565348	555279
1985	636841	636342	599092	580700	568574	561400	553644	541022	534700	522587
1986	609987	603156	578700	568400	569966	569761	573989	573735	566245	556055
1987	623079	619978	592892	582102	561698	550850	536522	525650	515893	492248
1988	517127	511023	493993	483400	481469	475070	472806	458767	441201	428578
1989	448572	441100	409708	393323	391918	390001	383839	377968	368060	360246
1990	385727	398961	390149	391108	411171	427931	441335	450824	452304	457658
1991	567249	580777	596890	616326	647415	676706	701677	709801	718748	720754
1992	779868	816124	818102	826297	838390	851831	856505			

В [137]:

```
train <- window(dole, end=end(dole)-c(3,1))
test <- window(dole, start=end(dole)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
fc3 <- stlf(train, lambda=BoxCox.lambda(train), h=36)
```

В [138]:

```
autoplot(test) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE) +
autolayer(fc3, series="stlf", PI=FALSE)
```



В [139]:

```
accuracy(fc1, test)
accuracy(fc2, test)
accuracy(fc3, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Th
Training set	-43.97203	16514.08	9559.249	0.5003715	6.235696	0.3051564	0.5075433	
Test set	245990.37348	302478.73	245990.373	37.7926695	37.792670	7.8526600	0.9366090	12.

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	12867.03	56276.27	31325.74	3.418356	27.74820	1.000000	0.9782284	NA
Test set	139826.81	224940.16	172348.47	17.407268	25.98169	5.501817	0.9179536	8.981789

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	The
Training set	164.504	6161.745	3538.65	0.1792235	3.731988	0.112963	-0.07099662	
Test set	177799.662	243916.217	183456.09	25.1736039	26.651081	5.856401	0.93667203	9.76

В [140]:

```
a10
```

```
ERROR while rich displaying an object: Error in repr_matrix_generic(obj,
"\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matched by mult
iple actual arguments

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.ts(obj)
9. repr_ts_generic(obj, repr_markdown.matrix, ...)
10. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Au
1991							3.526591	3.18089
1992	5.088335	2.814520	2.985811	3.204780	3.127578	3.270523	3.737851	3.55877
1993	6.192068	3.450857	3.772307	3.734303	3.905399	4.049687	4.315566	4.56218
1994	6.731473	3.841278	4.394076	4.075341	4.540645	4.645615	4.752607	5.35060
1995	6.749484	4.216067	4.949349	4.823045	5.194754	5.170787	5.256742	5.85527
1996	8.329452	5.069796	5.262557	5.597126	6.110296	5.689161	6.486849	6.30056
1997	8.524471	5.277918	5.714303	6.214529	6.411929	6.667716	7.050831	6.70491
1998	8.798513	5.918261	6.534493	6.675736	7.064201	7.383381	7.813496	7.43189
1999	10.391416	6.421535	8.062619	7.297739	7.936916	8.165323	8.717420	9.07096
2000	12.511462	7.457199	8.591191	8.474000	9.386803	9.560399	10.834295	10.64375
2001	14.497581	8.049275	10.312891	9.753358	10.850382	9.961719	11.443601	11.65923
2002	16.300269	9.053485	10.002449	10.788750	12.106705	10.954101	12.844566	12.19650
2003	16.828350	9.800215	10.816994	10.654223	12.512323	12.161210	12.998046	12.51727
2004	18.003768	11.938030	12.997900	12.882645	13.943447	13.989472	15.339097	15.37076
2005	20.778723	12.154552	13.402392	14.459239	14.795102	15.705248	15.829550	17.55470
2006	23.486694	12.536987	15.467018	14.233539	17.783058	16.291602	16.980282	18.61218

	Jan	Feb	Mar	Apr	May	Jun	Jul	Au
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

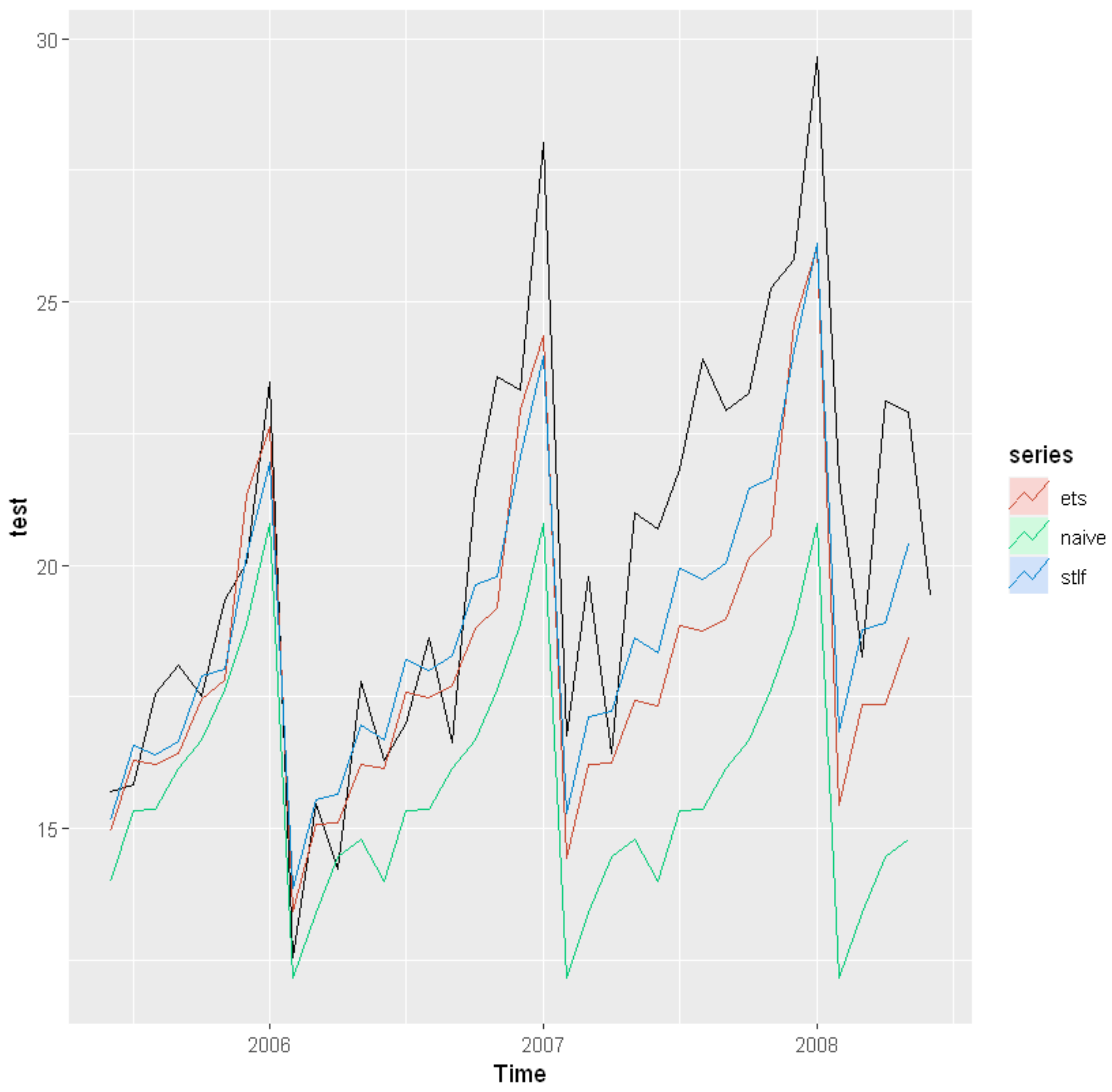
В [142]:

```
train <- window(a10, end=end(a10)-c(3,1))
test <- window(a10, start=end(a10)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
fc3 <- stlf(train, lambda=BoxCox.lambda(train), h=36)
autoplot(test) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE) +
autolayer(fc3, series="stlf", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
accuracy(fc3, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil'
Training set	0.04453347	0.4871696	0.3514114	0.2302407	3.978799	0.359998	-0.07201835	
Test set	1.94820204	2.8252641	2.2379827	8.5915374	10.409452	2.292667	0.30367534	0.7715

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	0.9528301	1.173254	0.9761483	10.86243	11.15917	1.000000	0.3820970	NA
Test set	4.3576261	5.218219	4.3701650	20.09919	20.18728	4.476948	0.6762667	1.410315

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil
Training set	0.005501358	0.4156385	0.3076362	-0.1384512	3.601905	0.3151531	-0.1345679	
Test set	1.383049872	2.2597694	1.8653501	5.6620330	8.769762	1.9109290	0.2170432	0.6122


30
25
20
15
test
2006
2007
2008
Time
series
ets
naive
stlf

В [144]:

```
h02
```

```
ERROR while rich displaying an object: Error in repr_matrix_generic(obj,
"\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matched by mult
iple actual arguments

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.ts(obj)
9. repr_ts_generic(obj, repr_markdown.matrix, ...)
10. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Au
1991							0.4297950	0.400906
1992	0.6601190	0.3362200	0.3513480	0.3798080	0.3618010	0.4105340	0.4833887	0.475463
1993	0.7515028	0.3875543	0.4272832	0.4138902	0.4288588	0.4701264	0.5092097	0.558443
1994	0.8193253	0.4376698	0.5061213	0.4704912	0.5106963	0.5405138	0.5581189	0.672852
1995	0.8031126	0.4752582	0.5525723	0.5271078	0.5612498	0.5889776	0.6231336	0.740837
1996	0.9372759	0.5287616	0.5593399	0.5778717	0.6149274	0.5941888	0.7077584	0.719502
1997	0.8468335	0.4638225	0.4852732	0.5280586	0.5623365	0.5885704	0.6694804	0.677993
1998	0.8005444	0.4905572	0.5244080	0.5366495	0.5520905	0.6033656	0.6812454	0.678075
1999	0.8930815	0.5126960	0.6529959	0.5739764	0.6392384	0.7038719	0.7706482	0.846185
2000	0.9696557	0.5732915	0.6185068	0.6189957	0.6652092	0.7265201	0.8558649	0.865984
2001	1.0438053	0.5106472	0.6725690	0.6484701	0.7041147	0.6994307	0.8519259	0.907705
2002	1.1458676	0.5755844	0.6411646	0.6798621	0.7679384	0.7520959	0.9180636	0.924367
2003	1.0781449	0.5782962	0.6433333	0.6633674	0.7505160	0.8007456	0.9163610	0.916886
2004	1.1301252	0.6679887	0.7490143	0.7399860	0.7951286	0.8568028	1.0015932	0.994864
2005	1.1706900	0.5976390	0.6525900	0.6705050	0.6952480	0.8422630	0.8743360	1.006497
2006	1.2306910	0.5871350	0.7069590	0.6396410	0.8074050	0.7979700	0.8843120	1.049648

	Jan	Feb	Mar	Apr	May	Jun	Jul	Au
2007	1.2233190	0.5977530	0.7043980	0.5617600	0.7452580	0.8379340	0.9541440	1.078219
2008	1.2199410	0.7618220	0.6494350	0.8278870	0.8162550	0.7621370		

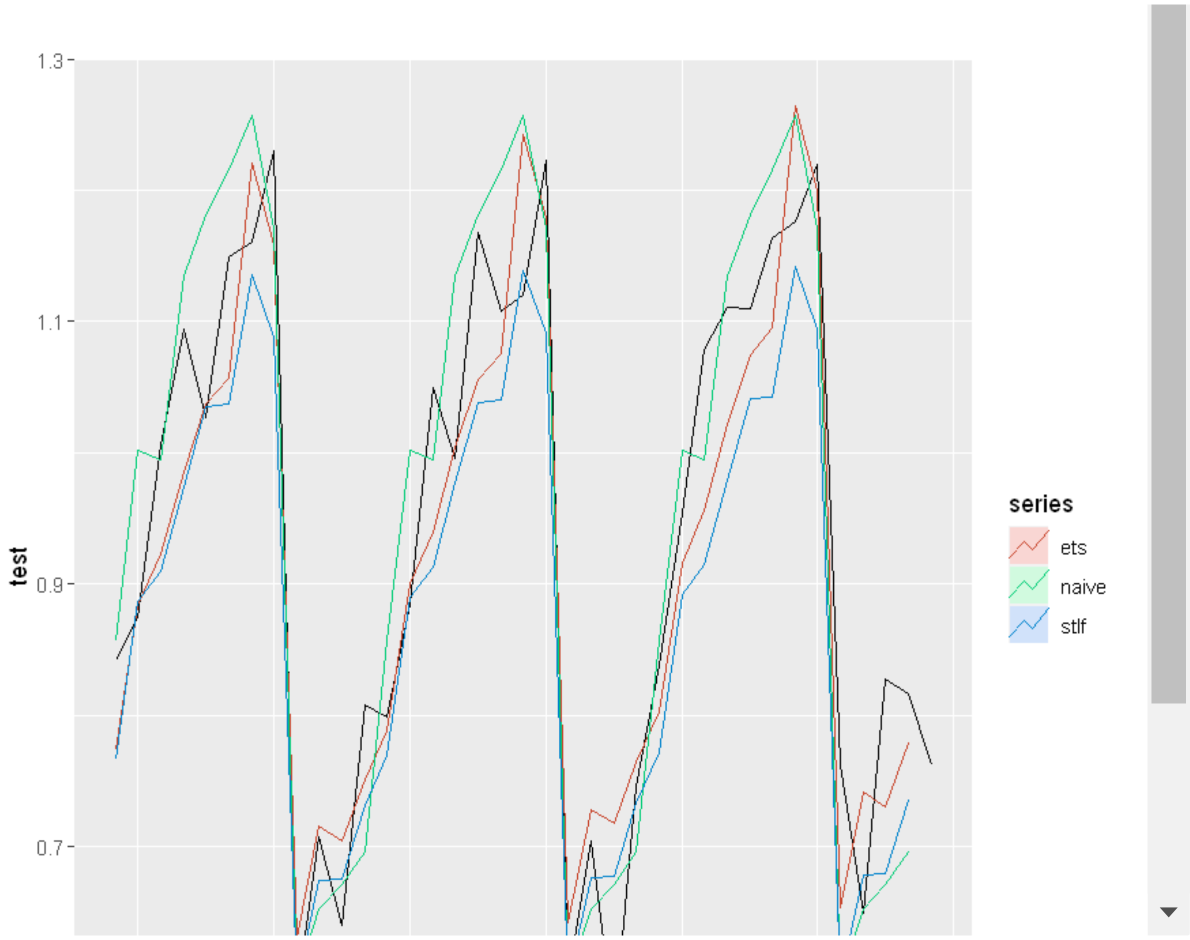
В [145]:

```
train <- window(h02, end=end(h02)-c(3,1))
test <- window(h02, start=end(h02)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
fc3 <- stlf(train, lambda=BoxCox.lambda(train), h=36)
autoplot(test) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE) +
autolayer(fc3, series="stlf", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
accuracy(fc3, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	T
Training set	-0.005839258	0.04721347	0.03485033	-0.9933135	4.659098	0.5858134	0.1342509	
Test set	0.018880781	0.07293169	0.06130399	1.1734152	6.834075	1.0304839	-0.1304120	0.4

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Thei
Training set	0.03915094	0.07144133	0.05949049	5.265019	8.197994	1.000000	0.39326999	
Test set	-0.01378079	0.08480448	0.07059648	-1.163027	7.739285	1.186685	0.05515531	0.513

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	T
Training set	0.001484422	0.03917380	0.02950828	-0.09917989	4.006421	0.4960167	-0.1400721	
Test set	0.060479825	0.08988643	0.07333716	5.67983815	7.692105	1.2327543	-0.1474802	0.


1.3
1.1
0.9
0.7
test
series
ets
naive
stlf

В [148]:

```
usmelec
```

```
ERROR while rich displaying an object: Error in repr_matrix_generic(obj,
"\n%s%s\n", sprintf("|%%s\n|%s|\n", : formal argument "cols" matched by mult
iple actual arguments

Traceback:
1. FUN(X[[i]], ...)
2. tryCatch(withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler), error = outer_handler)
3. tryCatchList(expr, classes, parentenv, handlers)
4. tryCatchOne(expr, names, parentenv, handlers[[1L]])
5. doTryCatch(return(expr), name, parentenv, handler)
6. withCallingHandlers({
 .     if (!mime %in% names(repr::mime2repr))
 .         stop("No repr_* for mimetype ", mime, " in repr::mime2repr")
 .     rpr <- repr::mime2repr[[mime]](obj)
 .     if (is.null(rpr))
 .         return(NULL)
 .     prepare_content(is.raw(rpr), rpr)
 . }, error = error_handler)
7. repr::mime2repr[[mime]](obj)
8. repr_markdown.ts(obj)
9. repr_ts_generic(obj, repr_markdown.matrix, ...)
10. repr_func(m, ..., rows = nrow(m), cols = ncol(m))
```

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1973	160.218	143.539	148.158	139.589	147.395	161.244	173.733	177.365	156.875	154.197
1974	157.555	142.748	150.342	142.312	153.813	156.440	178.247	174.119	152.467	152.196
1975	164.623	147.349	155.760	146.495	153.531	162.717	177.057	179.931	155.441	155.188
1976	178.609	156.966	164.467	153.467	157.664	173.674	186.691	186.639	165.237	164.009
1977	196.665	162.949	169.437	157.117	169.596	181.031	199.168	196.363	176.498	166.645
1978	198.108	173.746	173.461	160.013	175.549	188.585	202.947	206.659	185.802	176.013
1979	209.987	186.587	183.154	170.260	178.409	186.976	202.522	205.101	180.975	179.953
1980	200.296	188.961	187.745	169.017	176.066	189.748	217.058	215.629	191.698	178.761
1981	206.758	179.860	185.834	172.841	178.139	203.021	220.655	210.639	187.051	181.558
1982	209.694	180.546	187.968	172.877	177.480	186.447	210.865	205.892	180.875	173.172
1983	195.871	172.725	182.769	170.669	174.725	191.367	220.447	230.193	195.817	183.137
1984	216.924	189.810	200.387	181.381	192.550	209.967	221.526	229.532	195.411	191.142
1985	228.148	198.488	195.250	185.173	197.123	205.682	227.004	226.286	202.712	194.995
1986	217.761	192.582	197.115	186.370	197.647	215.334	242.954	225.402	206.905	197.960
1987	223.041	194.281	202.130	189.792	206.407	225.908	248.196	247.881	213.221	203.215
1988	238.188	217.183	214.294	196.297	208.704	233.066	257.742	267.929	220.392	210.814

	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct
1989	246.766	233.783	241.946	222.869	234.825	250.919	273.458	275.420	242.794	235.669
1990	255.187	229.499	244.761	229.770	241.774	268.992	287.448	290.655	258.357	244.372
1991	269.214	228.290	240.561	227.799	254.873	269.099	294.648	292.159	256.027	245.378
1992	267.773	239.514	247.733	233.406	242.412	261.077	293.617	281.927	259.925	244.994
1993	271.021	248.015	261.248	234.695	244.326	275.360	312.225	311.450	264.032	250.553
1994	289.768	249.172	257.998	240.637	252.745	294.162	311.257	307.605	266.262	256.528
1995	279.773	252.307	261.343	244.736	264.288	286.258	330.416	345.780	277.575	263.978
1996	296.923	270.685	275.019	251.613	282.266	302.717	327.708	329.286	283.151	271.446
1997	300.574	258.131	272.258	258.284	272.914	299.092	344.516	332.899	301.057	284.971
1998	295.260	260.590	286.878	261.230	299.640	328.903	361.936	357.366	318.924	284.446
1999	315.814	274.820	298.145	280.719	300.098	328.924	376.538	364.031	305.516	283.935
2000	327.994	294.169	301.580	285.578	322.954	339.054	356.528	368.669	312.447	289.452
2001	332.493	282.940	300.707	278.079	300.492	327.694	357.614	370.533	306.929	294.734
2002	319.941	281.826	302.549	289.848	307.675	341.023	381.542	374.586	331.279	307.059
2003	341.989	299.249	304.317	285.756	307.545	328.694	374.396	381.816	323.136	306.741
2004	346.546	314.280	308.812	290.560	327.380	345.085	377.332	368.439	335.622	312.450
2005	343.121	298.500	317.458	289.562	315.062	363.672	402.274	404.941	350.218	316.398
2006	328.658	307.333	318.730	297.858	330.616	364.260	410.421	407.763	332.055	321.567
2007	353.531	323.230	320.471	303.129	330.203	362.755	393.226	421.797	355.394	332.615
2008	362.998	325.106	324.630	305.865	325.245	373.109	402.900	388.987	338.056	318.547
2009	354.993	300.887	310.603	289.537	311.306	347.658	372.542	381.221	327.401	307.040
2010	360.957	319.735	312.168	287.800	327.936	375.759	409.725	408.884	346.045	307.921
2011	363.105	313.293	318.710	302.400	323.627	367.727	418.693	406.541	337.961	308.727
2012	340.919	310.151	309.040	295.940	337.530	361.506	416.515	396.108	334.735	312.157
2013	348.642	309.601	325.372	298.261	322.118	356.400				

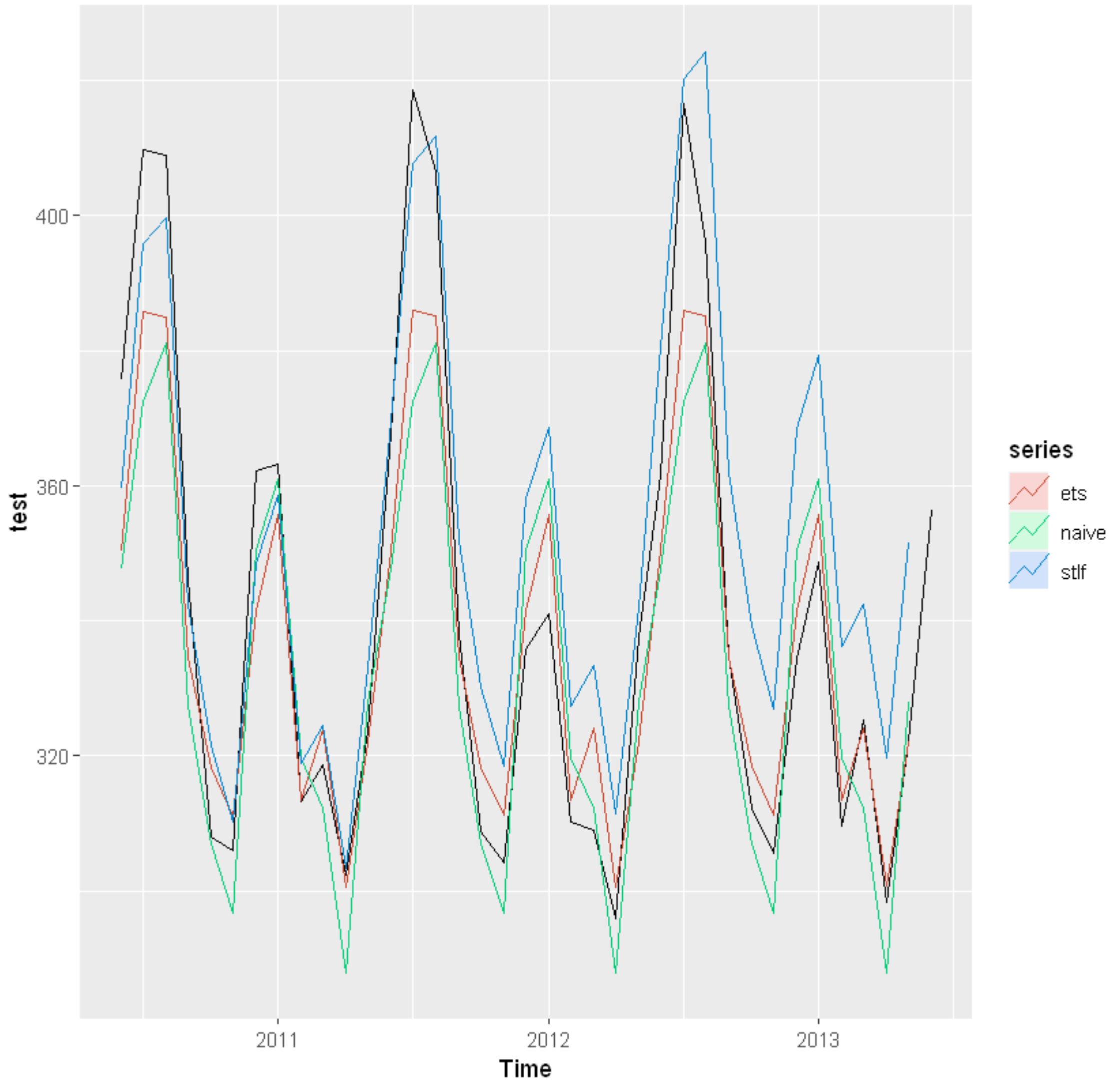
В [149]:

```
train <- window(usmelec, end=end(usmelec)-c(3,1))
test <- window(usmelec, start=end(usmelec)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
fc3 <- stlf(train, lambda=BoxCox.lambda(train), h=36)
autoplot(test) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE) +
autolayer(fc3, series="stlf", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
accuracy(fc3, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's
Training set	0.4949346	7.298956	5.450133	0.1444447	2.133156	0.6086471	0.04427954	N
Test set	4.1089753	13.617773	10.387622	0.8909161	2.870954	1.1600444	0.58075337	0.388240

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	4.868522	11.52062	8.954503	1.988132	3.502571	1.000000	0.4839319	NA
Test set	8.355472	17.78925	14.060694	2.178254	3.912180	1.570237	0.5411975	0.5084651

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	The
Training set	-0.1444685	6.298561	4.733474	-0.05361971	1.838786	0.5286137	0.08417934	
Test set	-11.7022560	18.308198	15.717834	-3.67771229	4.716249	1.7552994	0.67082880	0.579



14.

a. Use `ets()` on the following series: `bicoal` , `chicken` , `dole` , `usdeaths` , `lynx` , `ibmclose` , `eggs` .

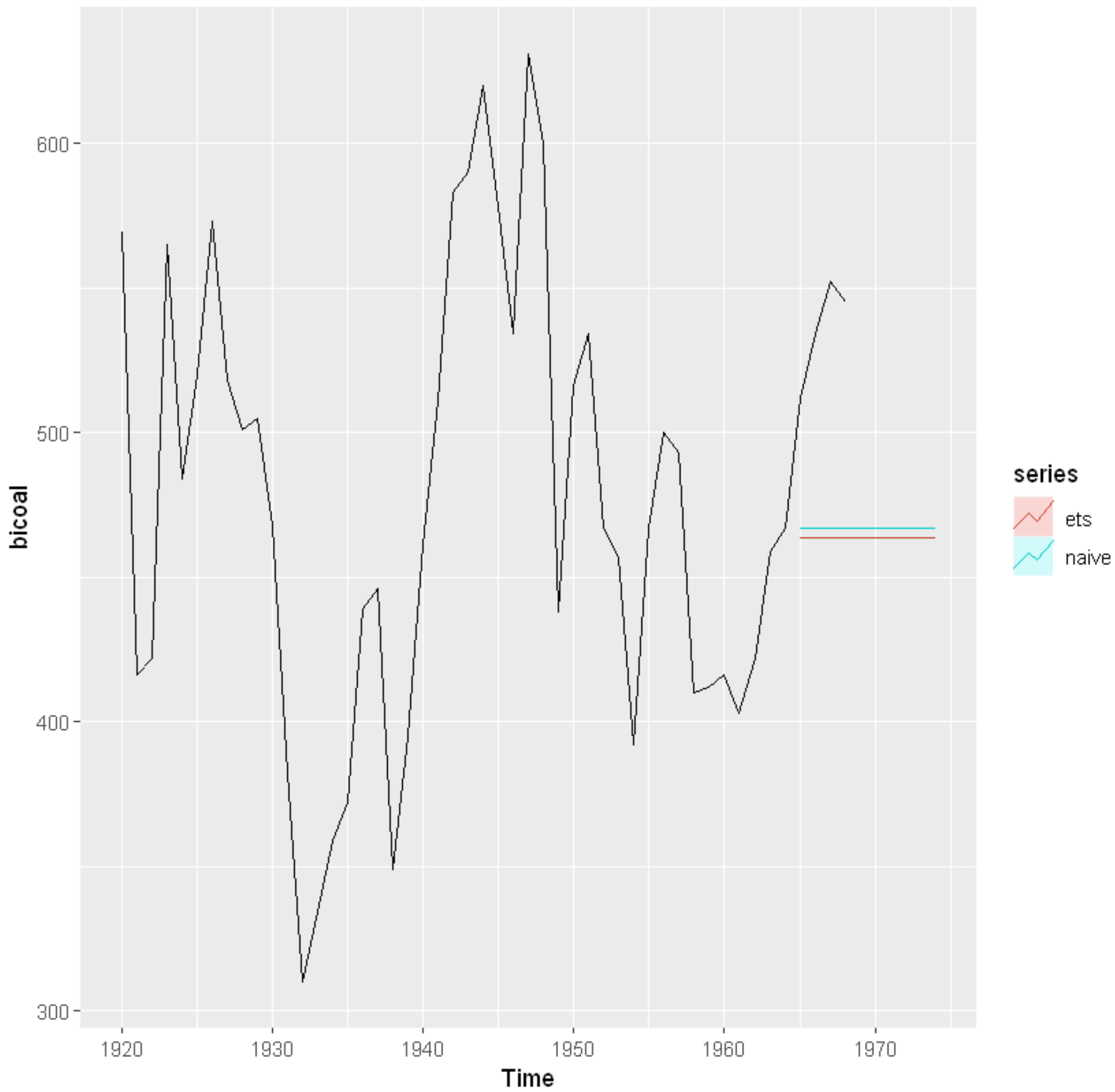
```
Does it always give good forecasts?
```

В [42]:

```
train <- window(bicoal, end=end(bicoal)-c(3,1))
test <- window(bicoal, start=end(bicoal)-c(3,0))
fc1 <- forecast(ets(train))
fc2 <- snaive(train, h=10)
autoplot(bicoal) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-2.13692	61.40706	48.37468	-1.495224	10.60489	0.9936908	0.03345824	NA
Test set	72.04094	73.61443	72.04094	13.376356	13.37636	1.4798325	0.17827925	4.646749

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-2.318182	62.78245	48.68182	-1.365742	10.66286	1.000000	-0.07866075	NA
Test set	68.750000	70.39709	68.75000	12.761589	12.76159	1.412232	0.17827925	4.456501

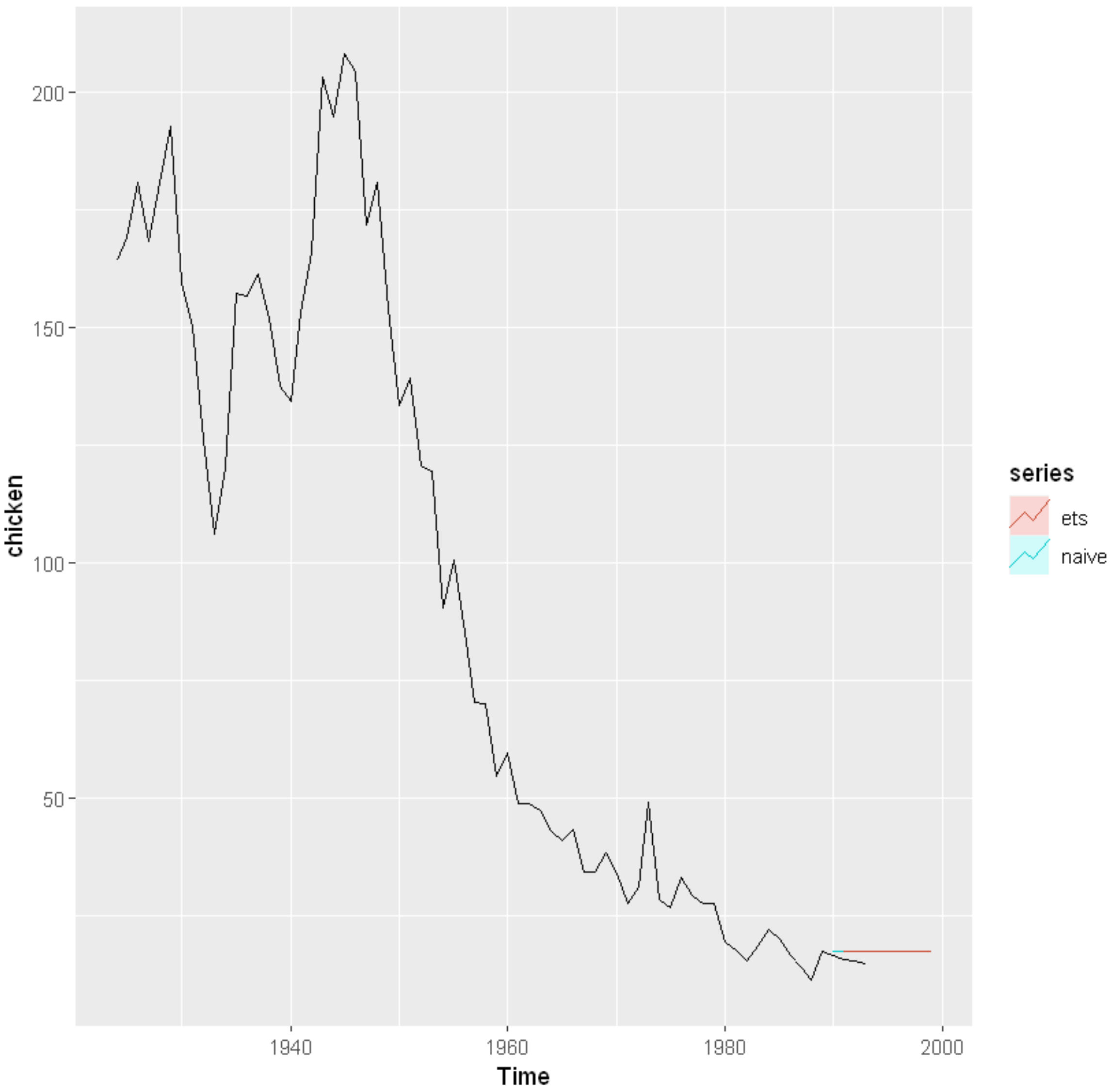


В [43]:

```
train <- window(chicken, end=end(chicken)-c(3,1))
test <- window(chicken, start=end(chicken)-c(3,0))
fc1 <- forecast(ets(train))
fc2 <- snaive(train)
autoplot(chicken) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's
Training set	-2.156730	13.776049	10.069824	-4.864515	13.62123	0.9914847	0.004282254	N
Test set	-1.871887	1.984711	1.871887	-12.272975	12.27297	0.1843078	0.200171649	3.6802

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's
Training set	-2.258154	13.870590	10.15631	-4.980539	13.788938	1.0000000	0.0001132474	I
Test set	-1.310000	1.364001	1.31000	-8.212344	8.212344	0.1289839	-0.5000000000	2.2236



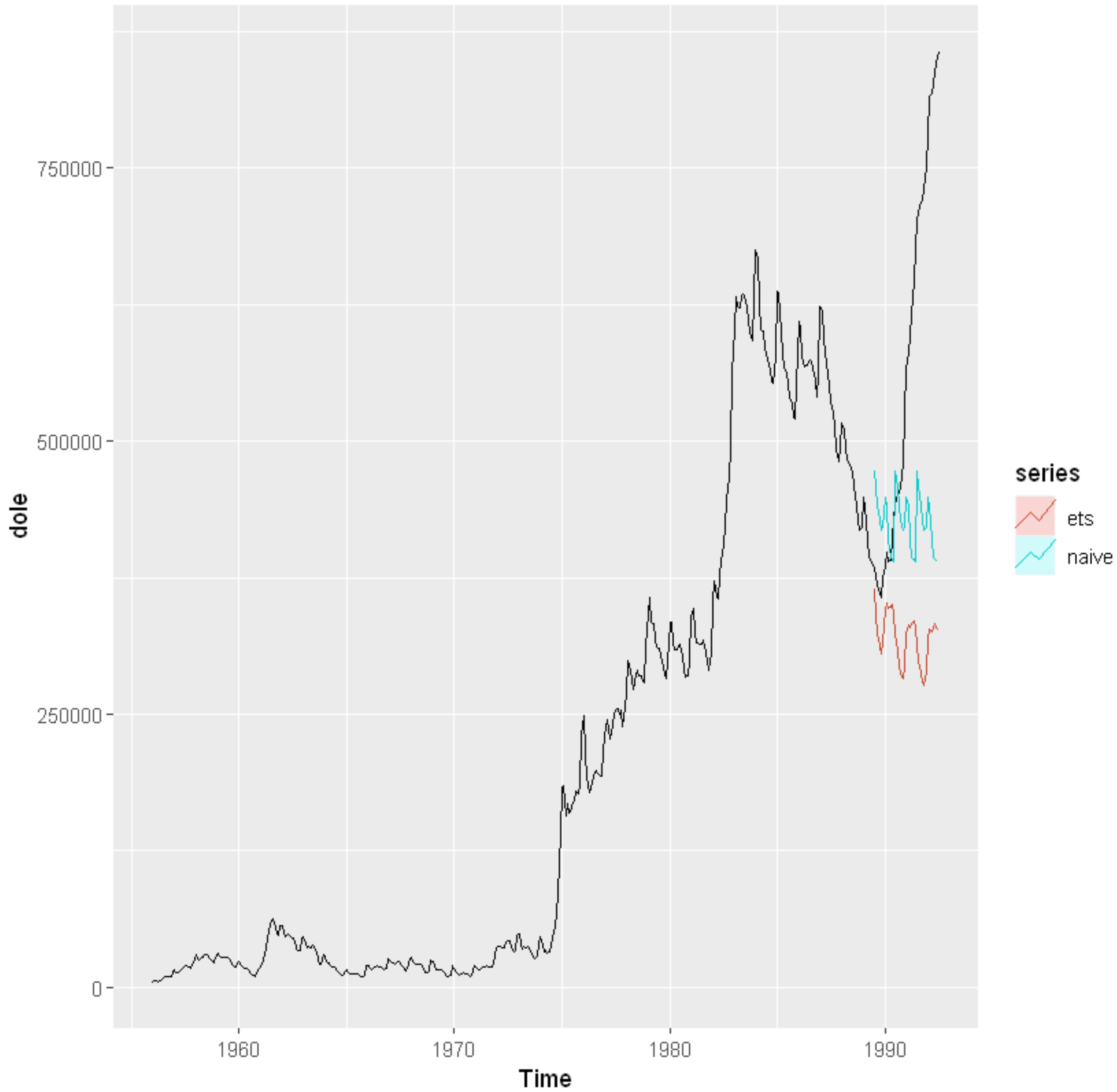
chicken
200
150
100
50
1940
1960
1980
2000
Time
series
ets
naive

В [44]:

```
train <- window(dole, end=end(dole)-c(3,1))
test <- window(dole, start=end(dole)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
autoplot(dole) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Th
Training set	-43.97203	16514.08	9559.249	0.5003715	6.235696	0.3051564	0.5075433	
Test set	245990.37348	302478.73	245990.373	37.7926695	37.792670	7.8526600	0.9366090	12.

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	12867.03	56276.27	31325.74	3.418356	27.74820	1.000000	0.9782284	NA
Test set	139826.81	224940.16	172348.47	17.407268	25.98169	5.501817	0.9179536	8.981789

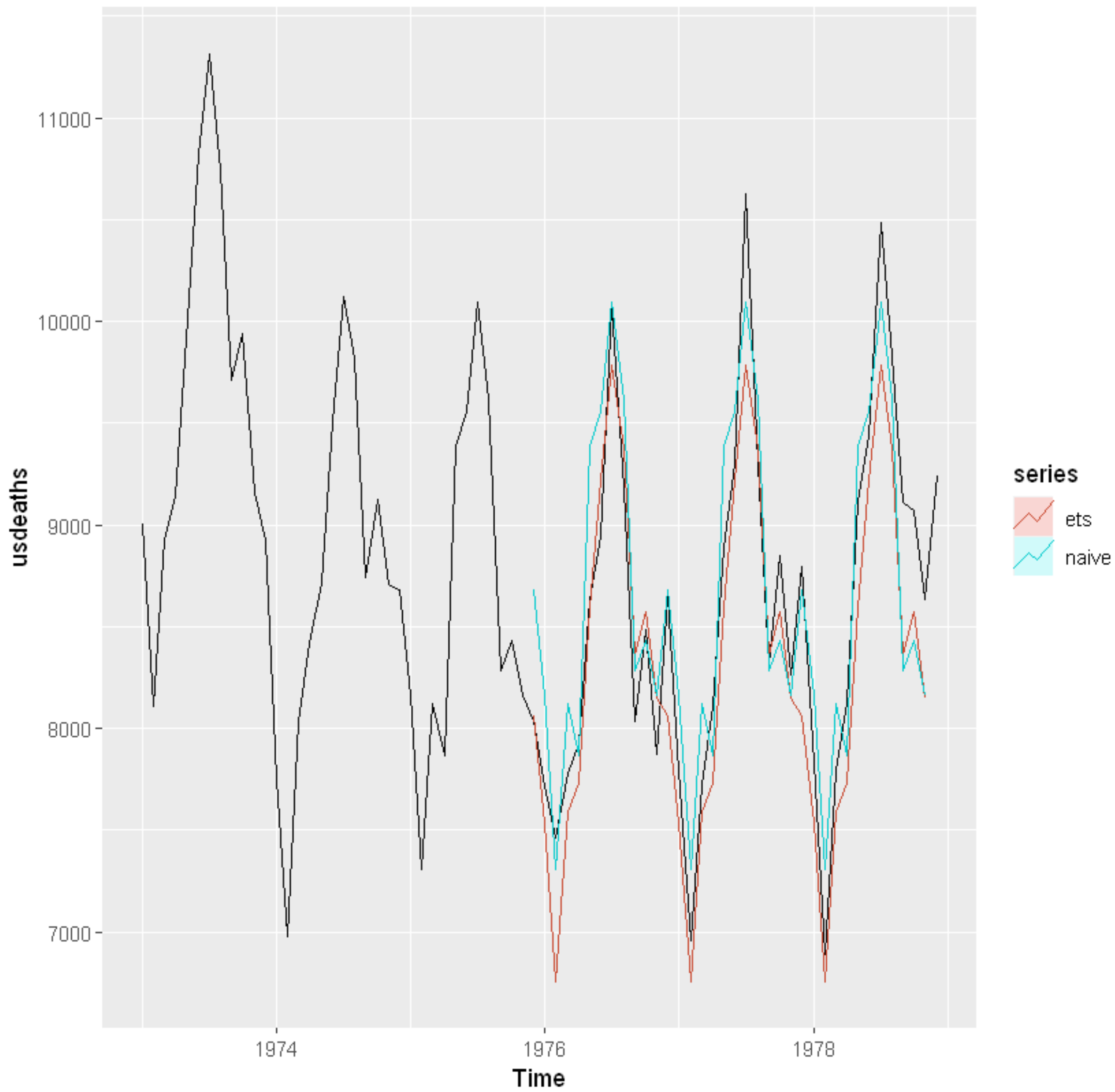


В [45]:

```
train <- window(usdeaths, end=end(usdeaths)-c(3,1))
test <- window(usdeaths, start=end(usdeaths)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
autoplot(usdeaths) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-57.97644	267.1174	198.7885	-0.6934528	2.290095	0.3003045	0.2266130	NA
Test set	244.89495	387.0590	317.8936	2.7884475	3.653449	0.4802334	0.1898319	0.5227897

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-528.04348	774.4619	661.9565	-6.155489	7.736768	1.0000000	0.7121250	NA
Test set	-83.05556	389.6411	332.6111	-1.159097	3.912874	0.5024667	0.2914318	0.5109459

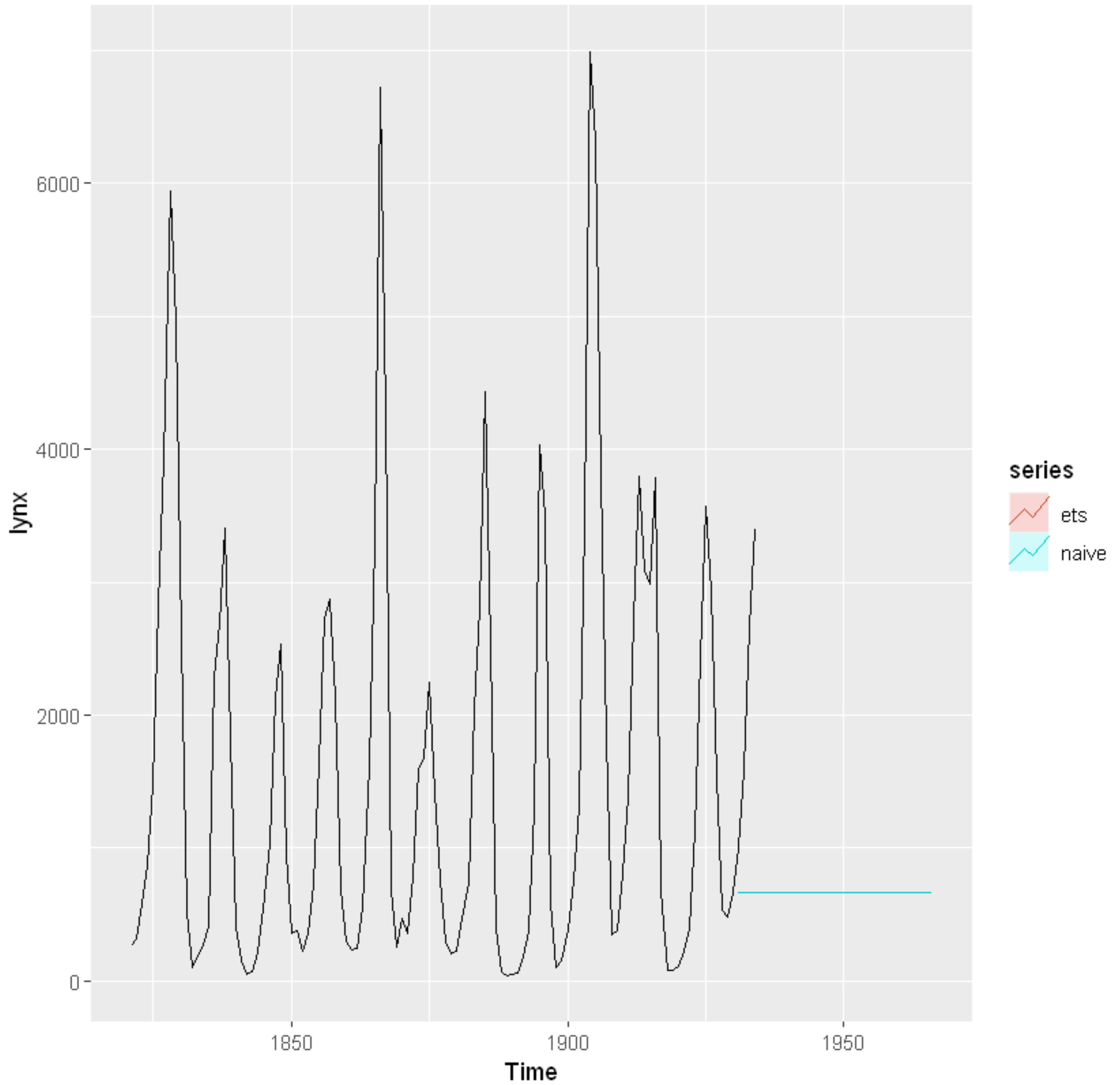


В [46]:

```
train <- window(lynx, end=end(lynx)-c(3,1))
test <- window(lynx, start=end(lynx)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
autoplot(lynx) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-15.55449	1212.021	847.8289	-55.2329	103.8471	1.013827	0.362455	NA
Test set	1498.76770	1762.840	1498.7677	61.9400	61.9400	1.792214	0.288011	1.997488

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	3.605505	1200.733	836.2661	-48.54511	97.60713	1.000000	0.3732181	NA
Test set	1498.750000	1762.825	1498.7500	61.93899	61.93899	1.792193	0.2880110	1.997467

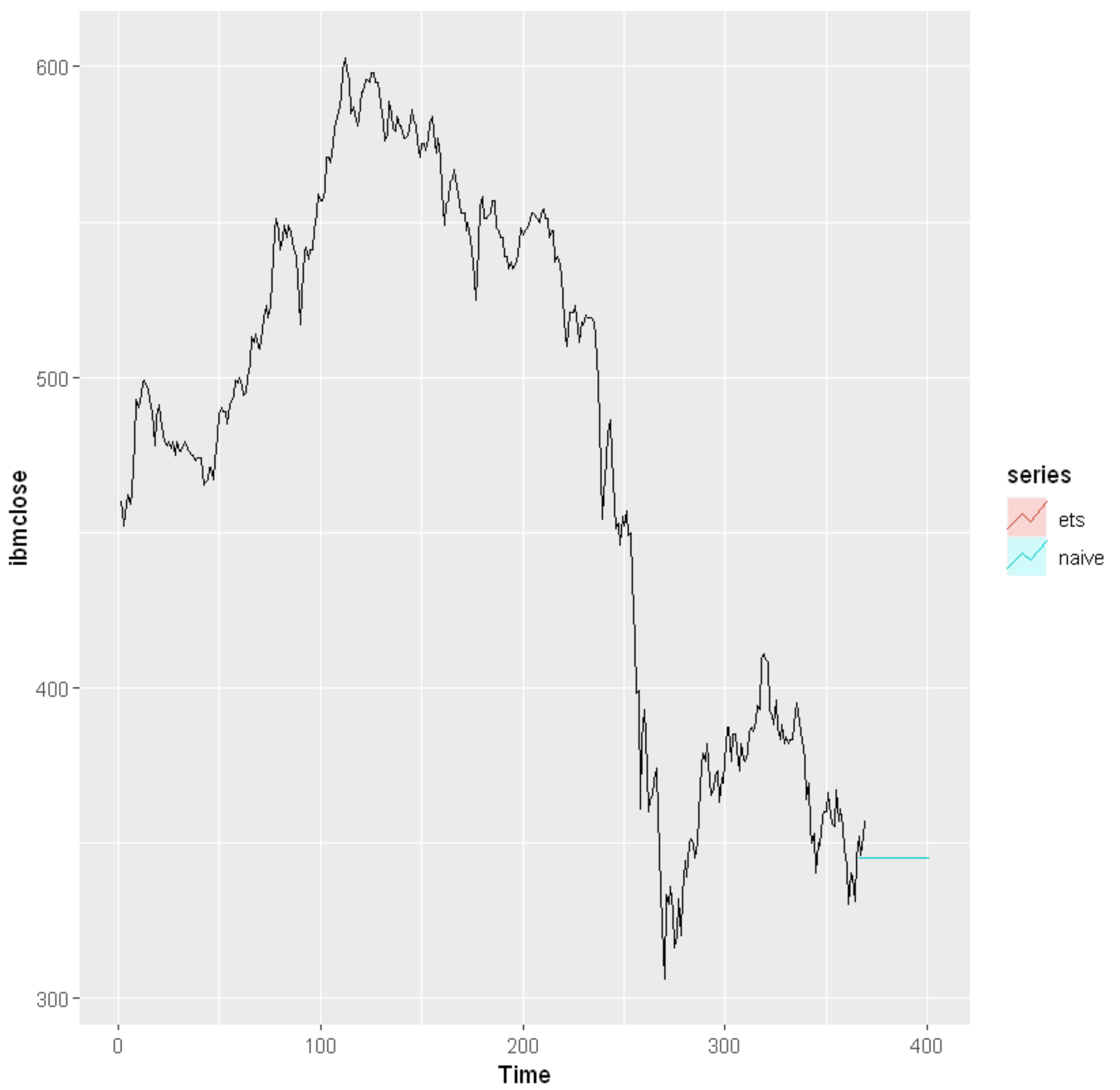


В [47]:

```
train <- window(ibmclose, end=end(ibmclose)-c(3,1))
test <- window(ibmclose, start=end(ibmclose)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
autoplot(ibmclose) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's
Training set	-0.3151359	7.256047	5.186389	-0.09460953	1.188325	0.9972772	0.08316537	N
Test set	6.7514051	7.795446	6.751405	1.90730816	1.907308	1.2982100	-0.02572016	1.41152

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-0.3159341	7.265945	5.200549	-0.09485176	1.191574	1.00000	0.08310933	NA
Test set	6.7500000	7.794229	6.750000	1.90690865	1.906909	1.29794	-0.02572016	1.411315



В [48]:

```
train <- window(eggs, end=end(eggs)-c(3,1))
test <- window(eggs, start=end(eggs)-c(3,0))
fc1 <- forecast(ets(train), h=36)
fc2 <- snaive(train, h=36)
autoplot(eggs) +
autolayer(fc1, series="ets", PI=FALSE) +
autolayer(fc2, series="naive", PI=FALSE)
accuracy(fc1, test)
accuracy(fc2, test)
```

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's
Training set	-2.669337	27.11594	19.999007	-2.701814	10.15031	0.9532364	-0.005649341	N
Test set	-7.721222	10.52020	8.541861	-12.105496	13.13399	0.4071409	0.303445465	2.00730

	ME	RMSE	MAE	MPE	MAPE	MASE	ACF1	Theil's U
Training set	-2.207079	27.54438	20.98011	-2.307694	10.59464	1.0000000	-0.1474276	NA
Test set	-9.932500	12.23568	9.93250	-15.277607	15.27761	0.4734245	0.3034455	2.329413