Demand Prediction using Time Series (Weekly)

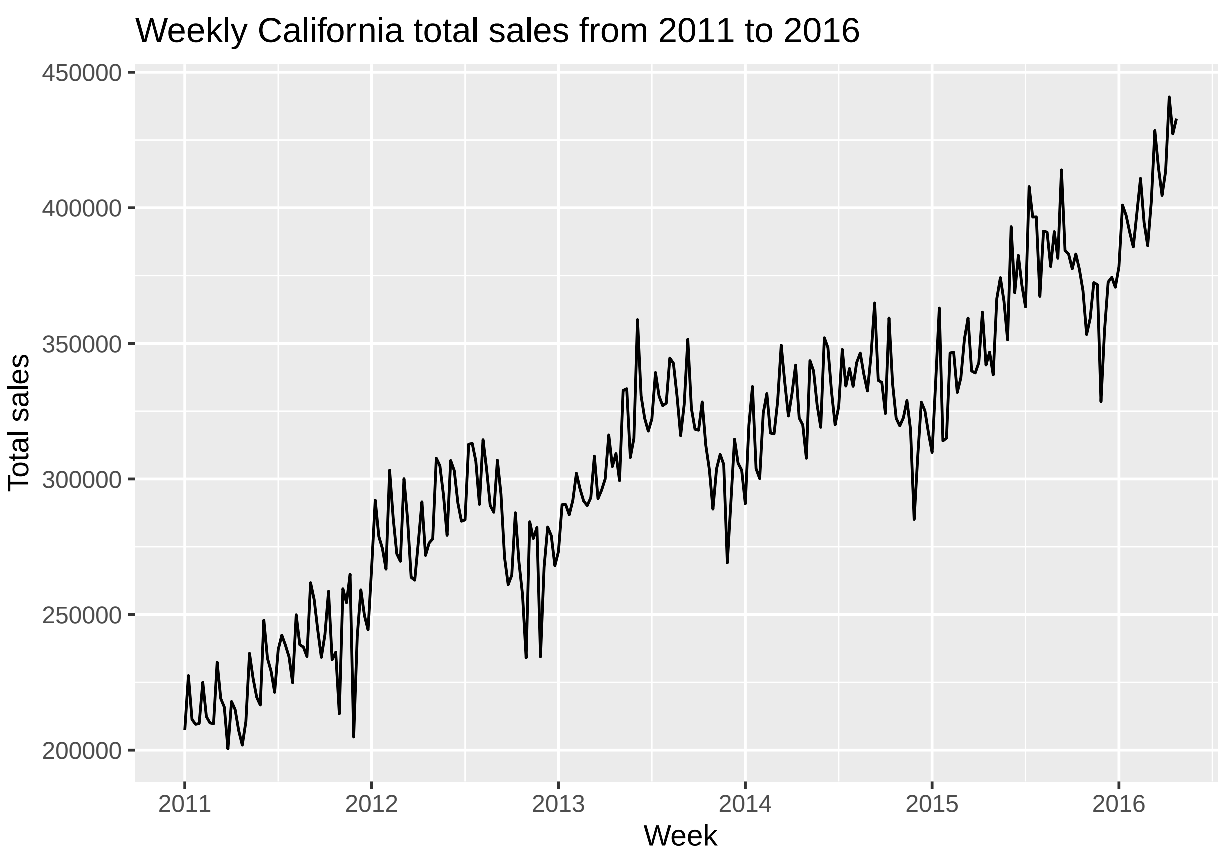
11/21/2020

```
library(forecast)
```

```
## Warning: package 'forecast' was built under R version 3.6.2
```

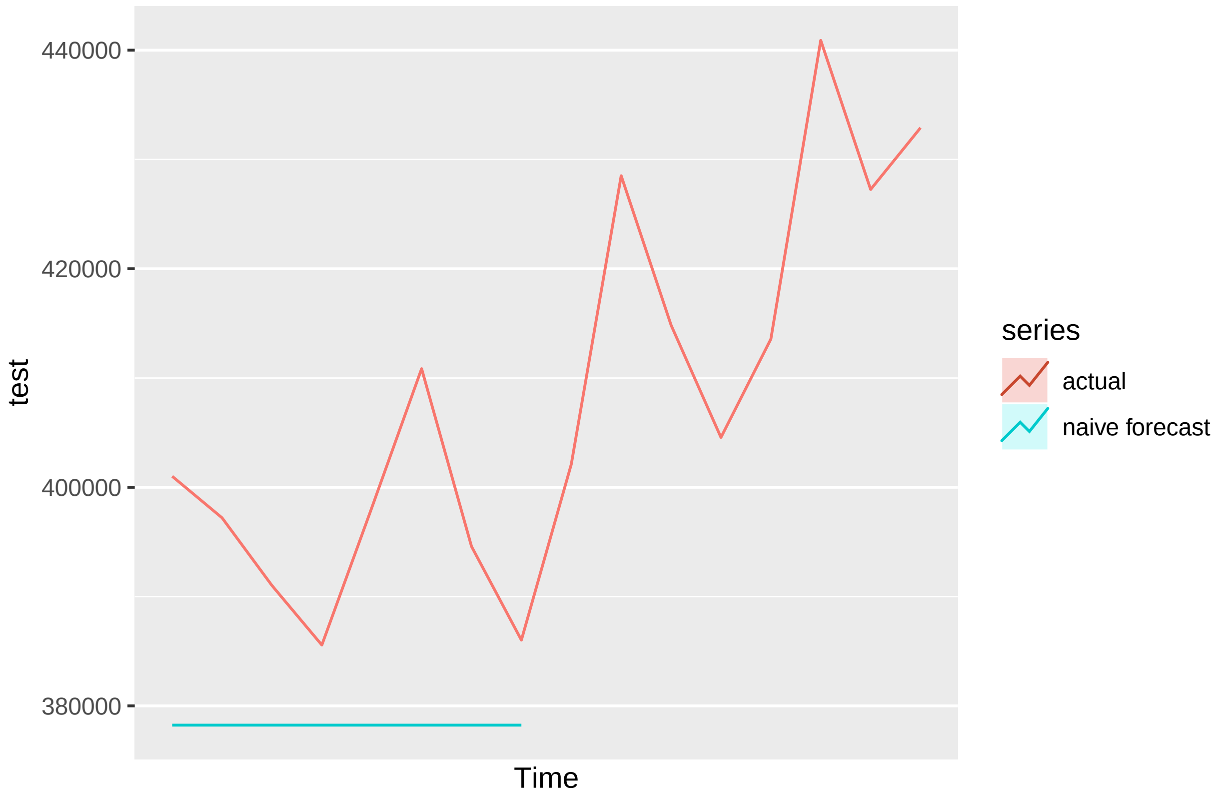
```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(ggplot2)
#library(DataCombine)

data = read.csv("weekly_new.csv")
data = data[1:277,]
ca.ts = ts(data$CA_total, start=c(2011, 1), frequency = 52)
tx.ts = ts(data$TX_total, start=c(2011, 1), frequency = 52)
wi.ts = ts(data$WI_total, start=c(2011, 1), frequency = 52)

autoplot(ca.ts, main="Weekly California total sales from 2011 to 2016", xlab="Week", ylab="Total sales")
```




```
# split train and test data
# use the last 16 weeks as test
ntest = 16
ntrain = length(ca.ts) - ntest
```

```r
train = window(ca.ts, end=c(2011, ntrain), frequency=52)
test = window(ca.ts, start=c(2011, ntrain+1), frequency=52)

# naive forecast
na.m = naive(train)
na.pred = forecast(na.m, h=8)
autoplot(test, series="actual") +
  autolayer(na.pred, PI=FALSE, series="naive forecast")
```



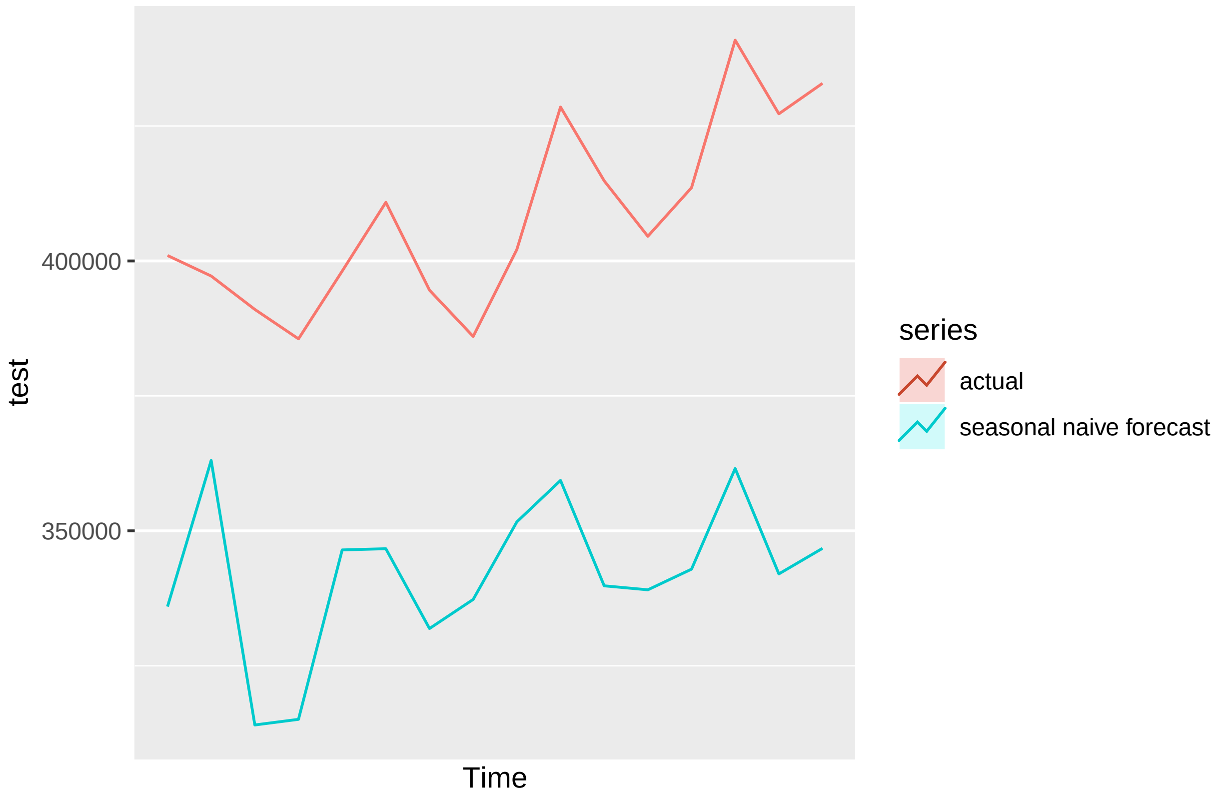

```r
accuracy(na.pred, test)
```

```
##                      ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   656.8782 17639.36 13691.72 0.04768451 4.598270 0.4064933
## Test set     17311.0625 18982.72 17311.06 4.33971765 4.339718 0.5139481
##                    ACF1 Theil's U
## Training set -0.2658419        NA
## Test set      0.1131553  1.794335
```

```r
na.pred = rep(378236.3, 16)
accuracy(na.pred, test)
```

```
##                ME     RMSE      MAE     MPE    MAPE      ACF1 Theil's U
## Test set 29827.59 34076.03 29827.59 7.16085 7.16085 0.5739171  2.394375
```

```r
# seasonal naive
nas.m = snaive(train)
nas.pred = forecast(nas.m, h=ntest)
autoplot(test, series="actual") +
  autolayer(nas.pred, PI=FALSE, series="seasonal naive forecast")
```




```
accuracy(nas.pred, test)
```

```
##                    ME     RMSE      MAE      MPE     MAPE     MASE
## Training set 33508.69 39065.68 33682.51 10.58723 10.64202 1.000000
## Test set     65963.01 67361.30 65963.01 16.11113 16.11113 1.958376
##                   ACF1 Theil's U
## Training set 0.8306041        NA
## Test set     0.2354716  4.714615
```

```
# Moving Average
library(zoo)
```

```
## Warning: package 'zoo' was built under R version 3.6.2

##
## Attaching package: 'zoo'

## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

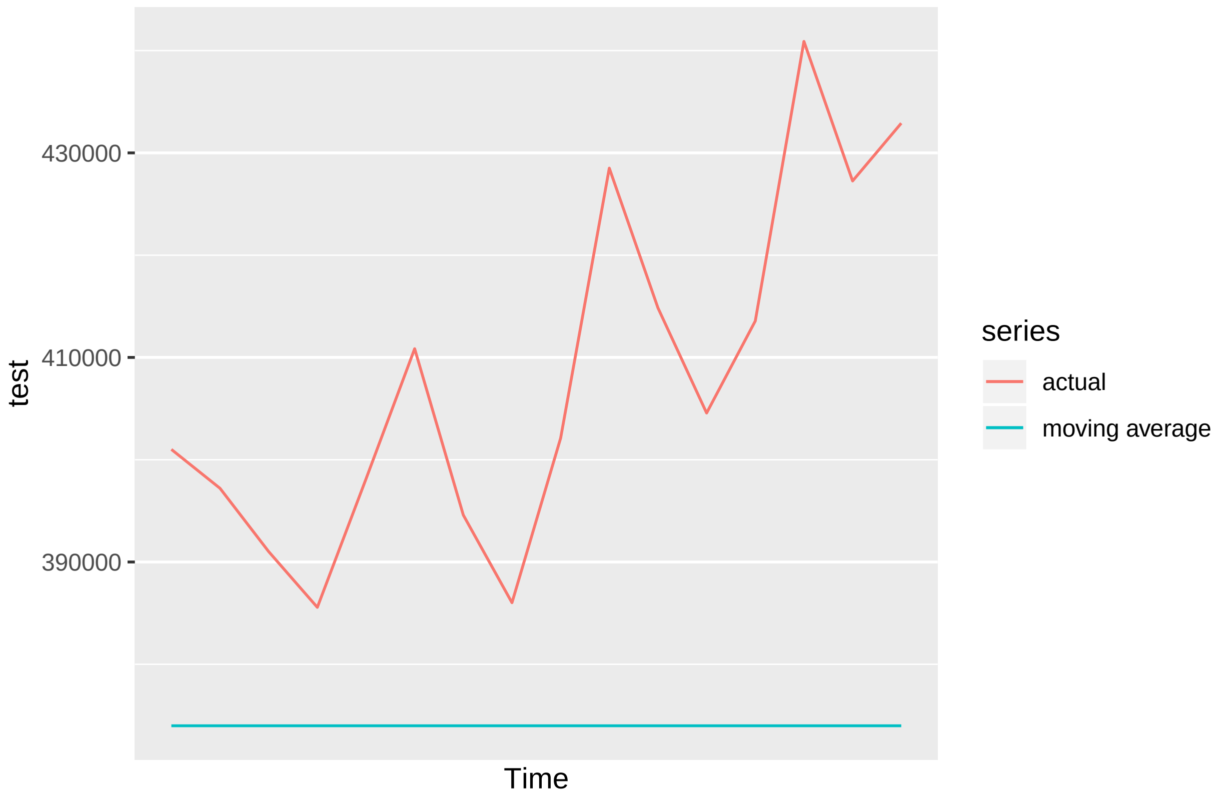
```
# w=4
ma = rollmean(train, k=4, align="right")
last.ma = tail(ma, 1)
ma.pred = ts(rep(last.ma, ntest), start=c(2011, ntrain+1), frequency = 52)
ma.pred
```

```
## Time Series:
## Start = c(2016, 2)
## End = c(2016, 17)
## Frequency = 52
```

```
##  [1] 373986.9 373986.9 373986.9 373986.9 373986.9 373986.9 373986.9
##  [8] 373986.9 373986.9 373986.9 373986.9 373986.9 373986.9 373986.9
## [15] 373986.9 373986.9
```

```
autoplot(test, series="actual") +
  autolayer(ma.pred, PI=FALSE, series="moving average")
```

```
## Warning: Ignoring unknown parameters: PI
```




```
# train accuracy
accuracy(ma, train)
```

```
##                ME     RMSE      MAE       MPE     MAPE       ACF1
## Test set 924.5439 12522.62 10019.76 0.1450757 3.364848 0.04839778
##          Theil's U
## Test set 0.7030077
```

```
# test accuracy
accuracy(ma.pred, test)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 34076.95 37851.41 34076.95 8.203869 8.203869 0.5739171  2.659076
```

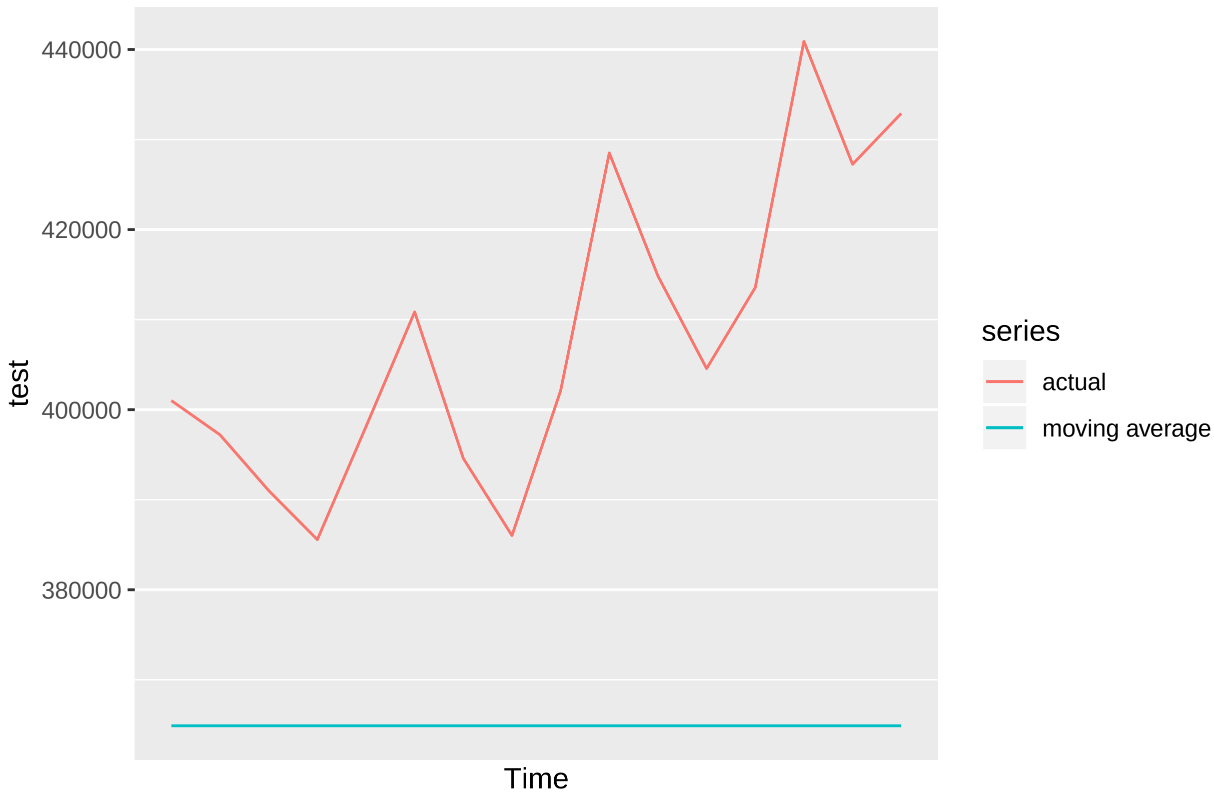
```
# w=52
ma = rollmean(train, k=52, align="right")
last.ma = tail(ma, 1)
ma.pred = ts(rep(last.ma, ntest), start=c(2011, ntrain+1), frequency = 52)
ma.pred
```

```
## Time Series:
## Start = c(2016, 2)
## End = c(2016, 17)
```

```
## Frequency = 52
##  [1] 364892 364892 364892 364892 364892 364892 364892 364892 364892 364892
## [11] 364892 364892 364892 364892 364892 364892
```

```
autoplot(test, series="actual") +
  autolayer(ma.pred, PI=FALSE, series="moving average")
```

```
## Warning: Ignoring unknown parameters: PI
```




```
# train accuracy
accuracy(ma, train)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 16401.47 26754.56 21702.46 4.862535 6.751236 0.6544885  1.491003
```

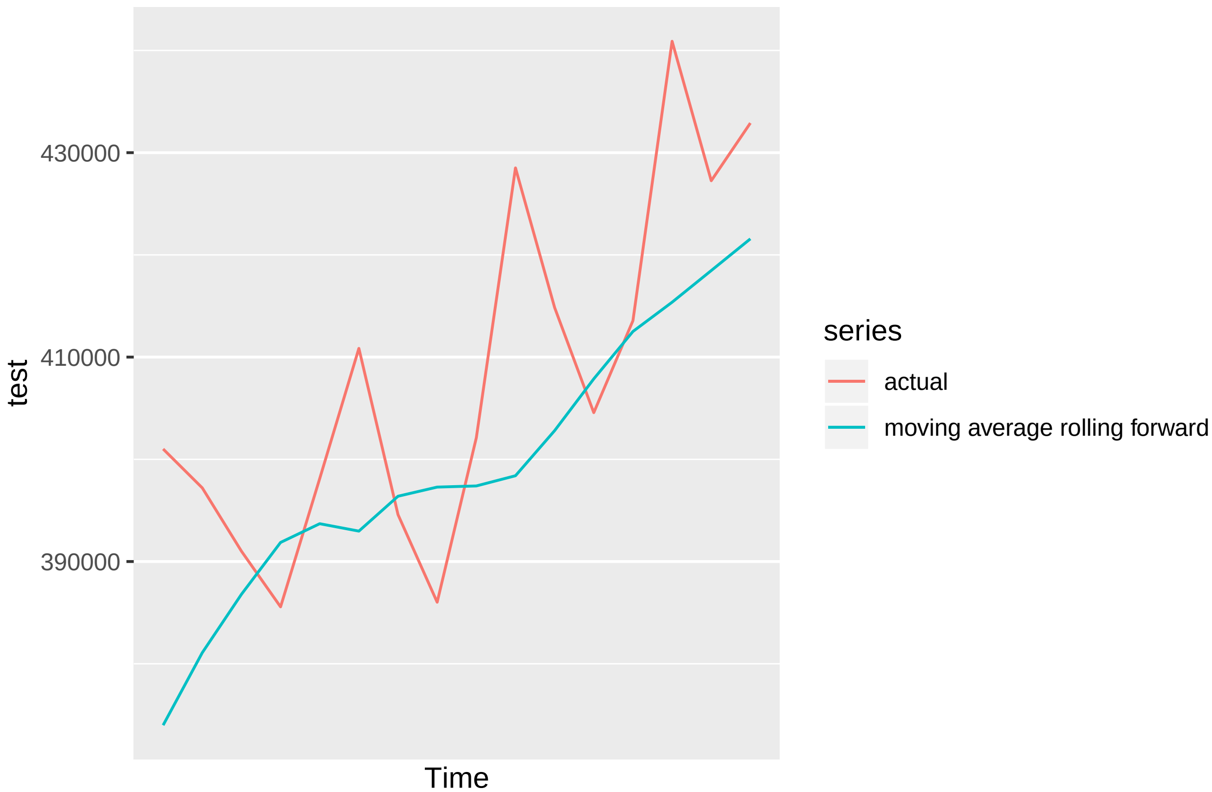
```
# test accuracy
accuracy(ma.pred, test)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 43171.91 46209.35 43171.91 10.43625 10.43625 0.5739171  3.246187
```

```
#########################################
# Moving Average rolling forward, k=4
mar.pred = rep(NA, ntest)
# start the for loop
for(i in 1:ntest){

  # Split the data into training and validation
  nTrain = length(ca.ts) - ntest + (i-1)
  train.ts = window(ca.ts, start=c(2011, 1), end=c(2011, nTrain))
```

```
  # Fit a trailing average smoother
  ma.trailing.roll = rollmean(train.ts, k=4, align="right")

  # Find the last moving average in the taining period
  last.ma = tail(ma.trailing.roll, 1)

  # Use the last moving average as the prediction for each month in the validation   period
  mar.pred[i] = last.ma

}

mar.pred = ts(mar.pred, start=c(2011, ntrain+1), frequency = 52)
autoplot(test, series="actual") +
  autolayer(mar.pred, PI=FALSE, series="moving average rolling forward")
```

```
## Warning: Ignoring unknown parameters: PI
```




```
accuracy(ma.trailing.roll, train)
```

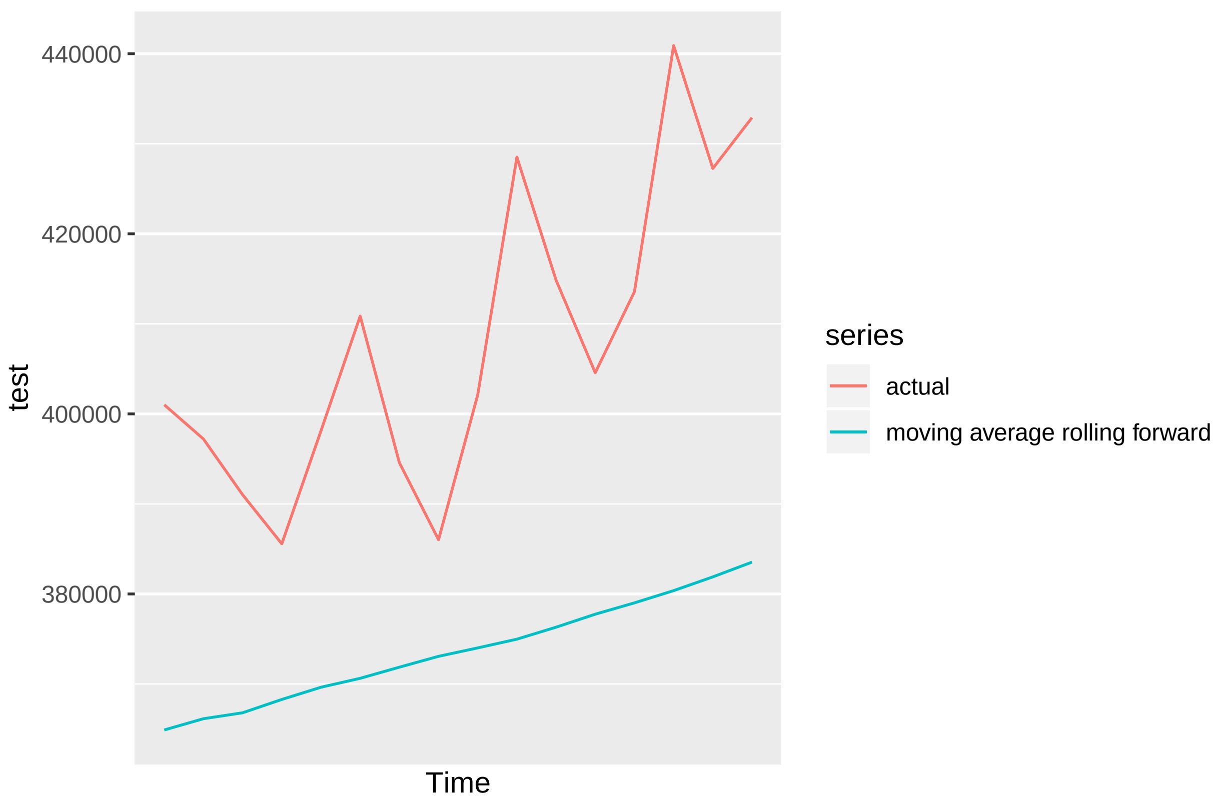
```
##                 ME     RMSE      MAE       MPE     MAPE       ACF1
## Test set 924.5439 12522.62 10019.76 0.1450757 3.364848 0.04839778
##          Theil's U
## Test set 0.7030077
```

```
accuracy(mar.pred, test)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 8785.652 14706.69 11619.29 2.083544 2.811678 0.1348512 0.9488963
```

```
###########################################
# Moving Average rolling forward, k=52
mar.pred = rep(NA, ntest)
# start the for loop
for(i in 1:ntest){

  # Split the data into training and validation
  nTrain = length(ca.ts) - ntest + (i-1)
  train.ts = window(ca.ts, start=c(2011, 1), end=c(2011, nTrain))

  # Fit a trailing average smoother
  ma.trailing.roll = rollmean(train.ts, k=52, align="right")

  # Find the last moving average in the taining period
  last.ma = tail(ma.trailing.roll, 1)

  # Use the last moving average as the prediction for each month in the validation   period
  mar.pred[i] = last.ma

}

mar.pred = ts(mar.pred, start=c(2011, ntrain+1), frequency = 52)
autoplot(test, series="actual") +
  autolayer(mar.pred, PI=FALSE, series="moving average rolling forward")
```

```
## Warning: Ignoring unknown parameters: PI
```



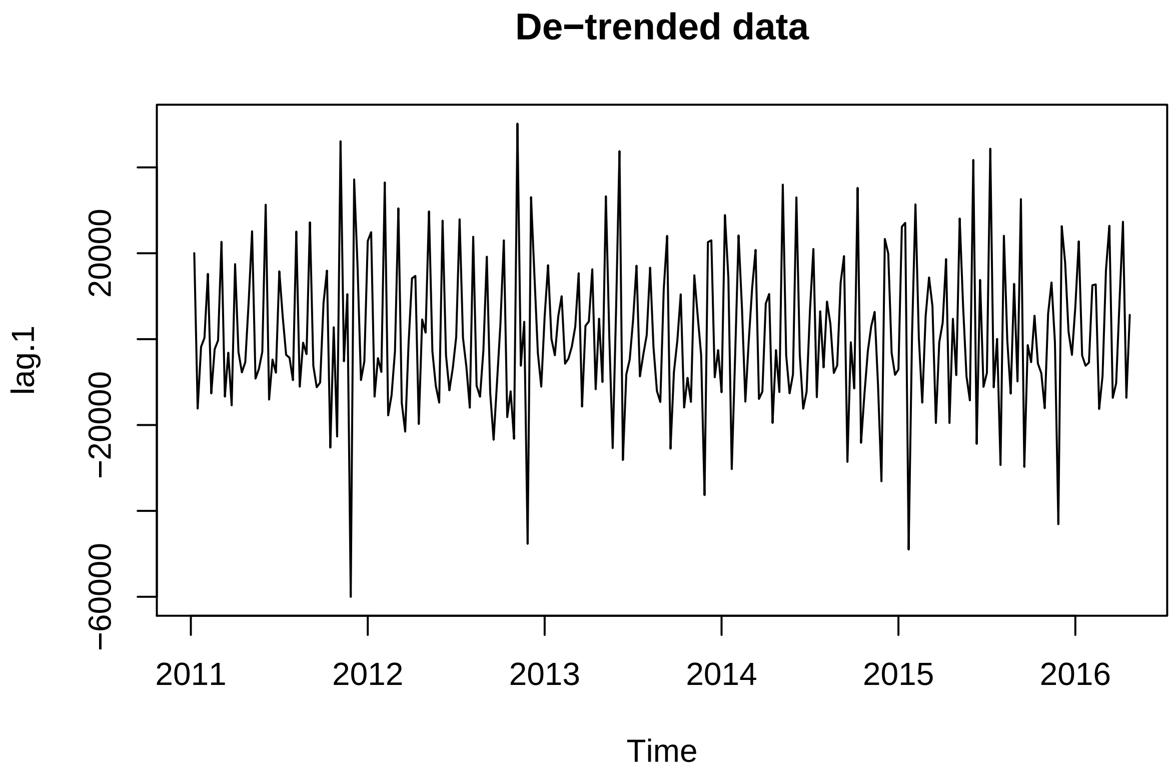

```
accuracy(ma.trailing.roll, train)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 16401.47 26754.56 21702.46 4.862535 6.751236 0.6544885  1.491003
```

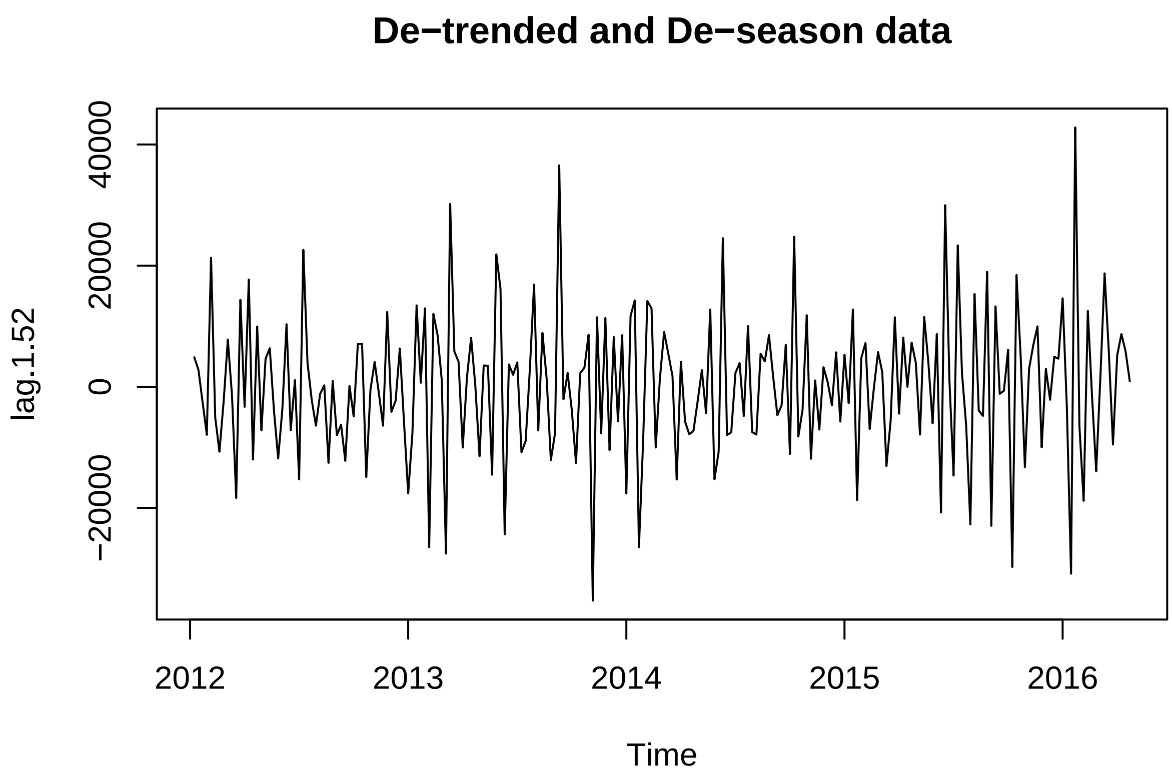
```r
accuracy(mar.pred, test)
```

```
##                 ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 34367.53 36641.44 34367.53 8.316868 8.316868 0.3742239  2.552762
```

```r
#####################################
# Simple exponential smoothing
# remove trend and seasonality
lag.1 = diff(ca.ts, lag=1)
plot(lag.1, main="De-trended data")
```




```r
lag.1.52 = diff(lag.1, lag=52)
plot(lag.1.52, main="De-trended and De-season data")
```




```
# split data into train and test
dntest = 16
dntrain = length(lag.1.52) - dntest
dtrain = window(lag.1.52, start=c(2012, 2), frequency=52)
dtest = window(lag.1.52, start=c(2012, dntrain+2), frequency=52)

# simple exponential smoothing
m1 = ets(dtrain, model="ANN")
m1.pred = forecast.ets(m1, h=ntest, level=0)
m1.pred = ts(m1.pred[[2]], start=c(2012, dntrain+2), frequency = 52)

accuracy(m1.pred, dtest)
```

```
##                   ME     RMSE      MAE      MPE     MAPE       ACF1 Theil's U
## Test set 990.8134 16125.76 11664.85 99.21369 99.21369 -0.3656072  1.085964
```

```
##########################################
# Holt-Winters Model, ANN
m2 = ets(train, model="ZZZ")
```

```
## Warning in ets(train, model = "ZZZ"): I can't handle data with frequency
## greater than 24. Seasonality will be ignored. Try stlf() if you need
## seasonal forecasts.
```

```
summary(m2)
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = train, model = "ZZZ")
##
##   Smoothing parameters:
```

```
##     alpha = 0.3426
##
##   Initial states:
##     l = 215514.0815
##
##   sigma:  15362.41
##
##      AIC     AICc      BIC
## 6488.245 6488.338 6498.938
##
## Training set error measures:
##                    ME     RMSE      MAE       MPE    MAPE      MASE
## Training set 1739.868 15303.44 11900.19 0.3815539 3.98793 0.3533048
##                   ACF1
## Training set 0.0691773
```

```
m2.pred = forecast.ets(m2, h=ntest, level=0)
m2.pred[[2]]
```

```
## Time Series:
## Start = c(2016, 2)
## End = c(2016, 17)
## Frequency = 52
##  [1] 371113 371113 371113 371113 371113 371113 371113 371113 371113 371113
## [11] 371113 371113 371113 371113 371113 371113
```

```
accuracy(m2.pred[[2]], test)
```

```
##                ME     RMSE      MAE      MPE     MAPE      ACF1 Theil's U
## Test set 36950.86 40458.09 36950.86 8.909278 8.909278 0.5739171   2.84205
```

```
##########################################
# Holt-Winters Model, ANN
m2 = ets(train, model="AAN")
summary(m2)
```

```
## ETS(A,A,N)
##
## Call:
##  ets(y = train, model = "AAN")
##
##   Smoothing parameters:
##     alpha = 0.3256
##     beta  = 1e-04
##
##   Initial states:
##     l = 211589.4729
##     b = 714.1022
##
##   sigma:  15317.67
##
##      AIC     AICc      BIC
## 6488.699 6488.934 6506.521
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
```

```
## Training set -301.5052 15199.84 11996.24 -0.3028275 4.038871 0.3561565
##                     ACF1
## Training set 0.08407806
```

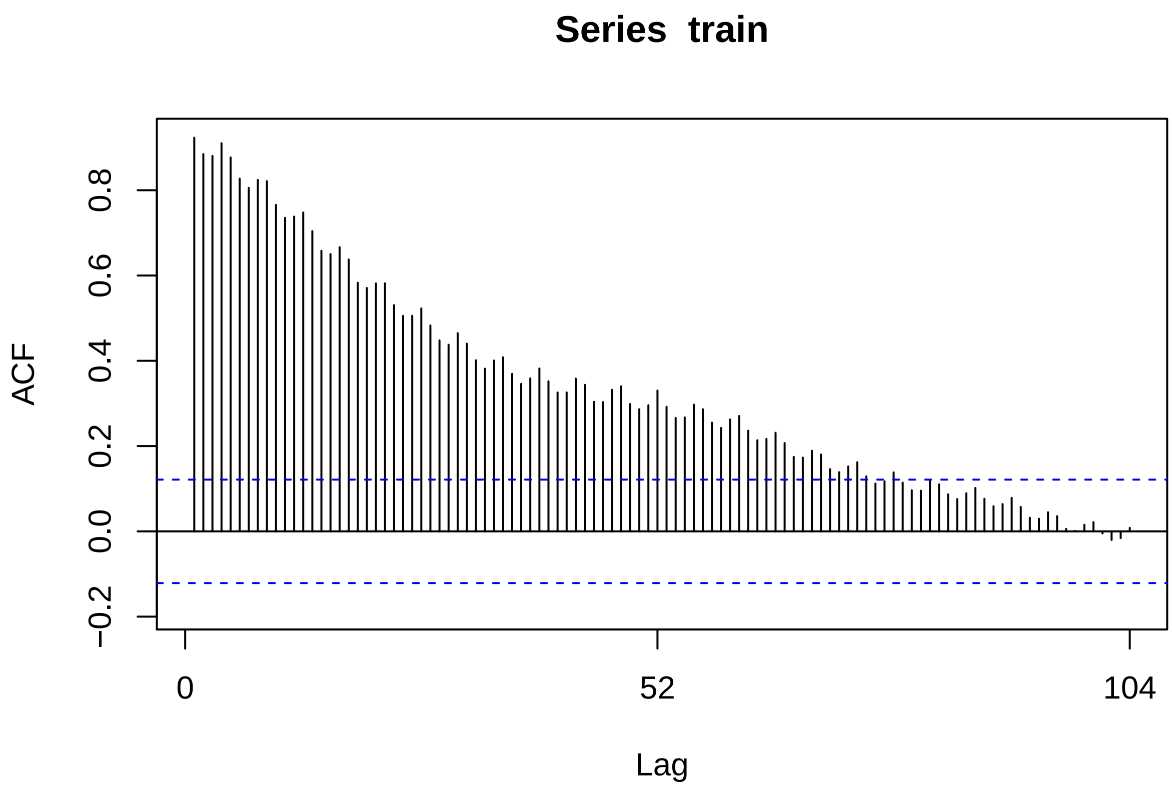
```
m2.pred = forecast.ets(m2, h=ntest, level=0)
m2.pred[[2]]
```

```
## Time Series:
## Start = c(2016, 2)
## End = c(2016, 17)
## Frequency = 52
##  [1] 372954.9 373661.1 374367.4 375073.6 375779.8 376486.0 377192.2
##  [8] 377898.5 378604.7 379310.9 380017.1 380723.3 381429.5 382135.8
## [15] 382842.0 383548.2
```

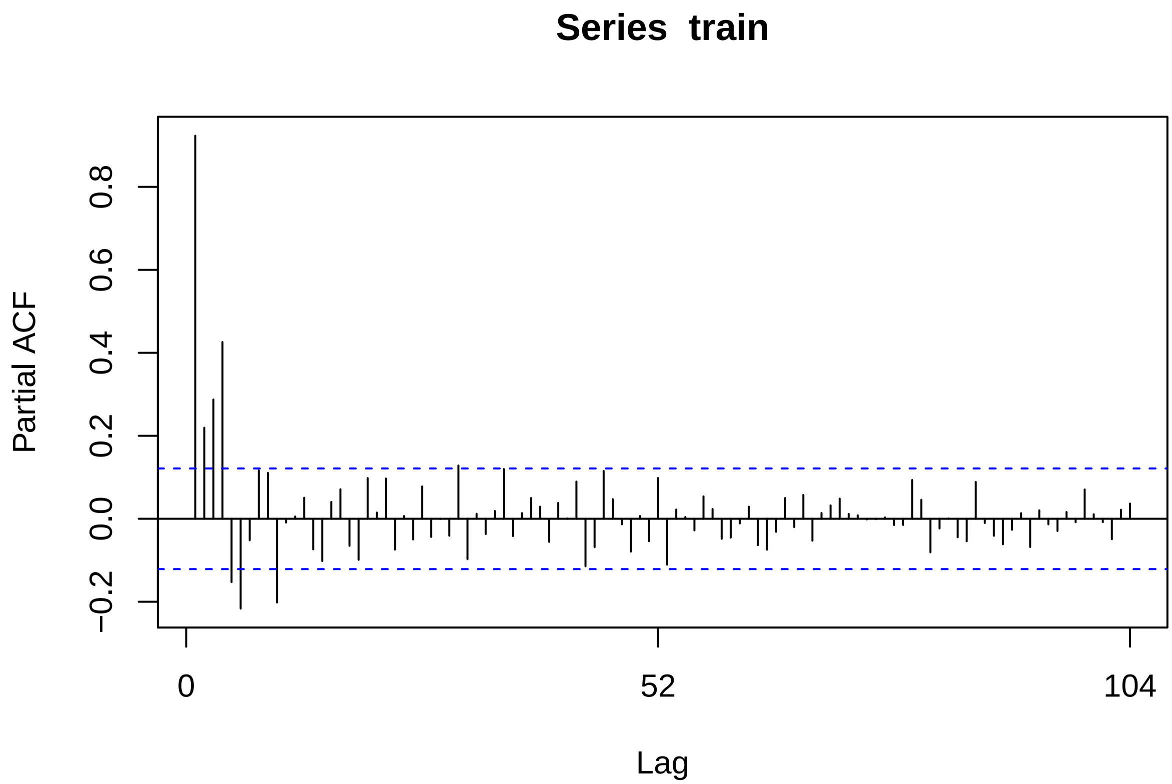
```
accuracy(m2.pred[[2]], test)
```

```
##                ME     RMSE      MAE     MPE    MAPE      ACF1 Theil's U
## Test set 29812.32 33001.42 29812.32 7.18121 7.18121 0.4723404  2.306854
```

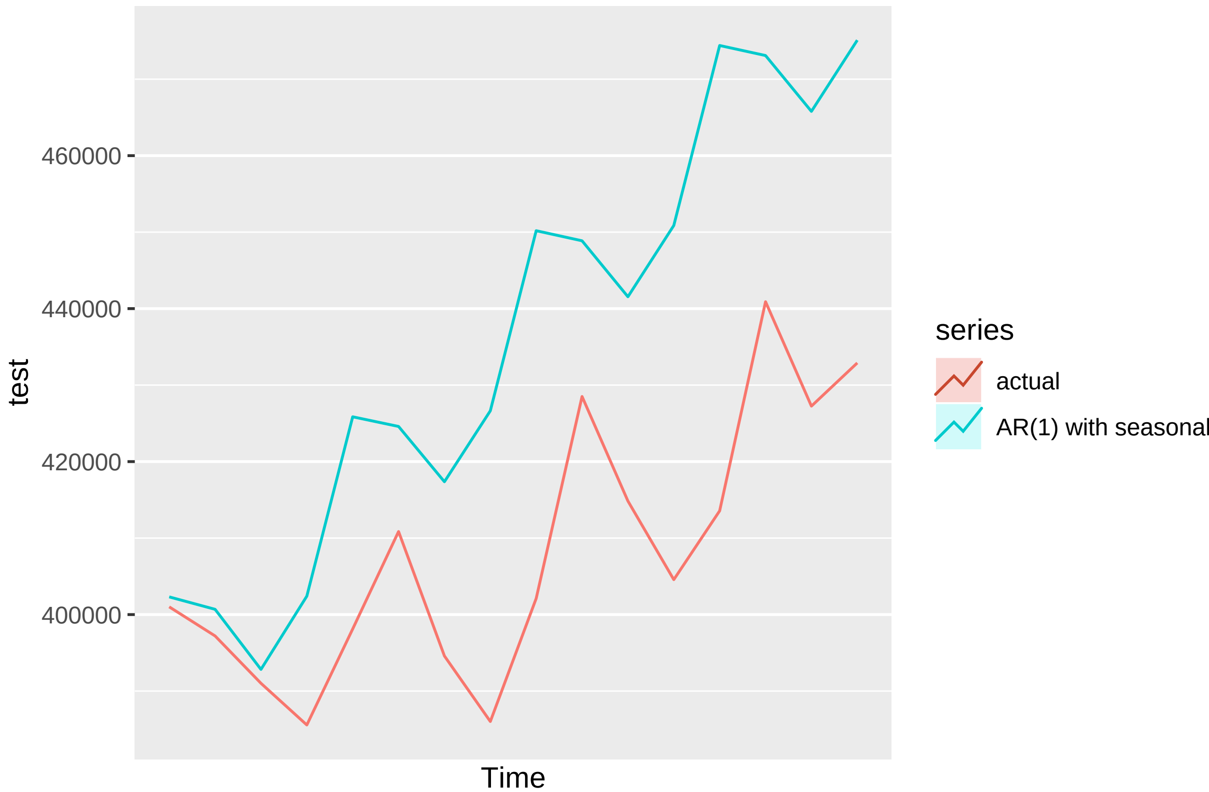
```
###########################################
Acf(train)
```




```
Pacf(train)
```



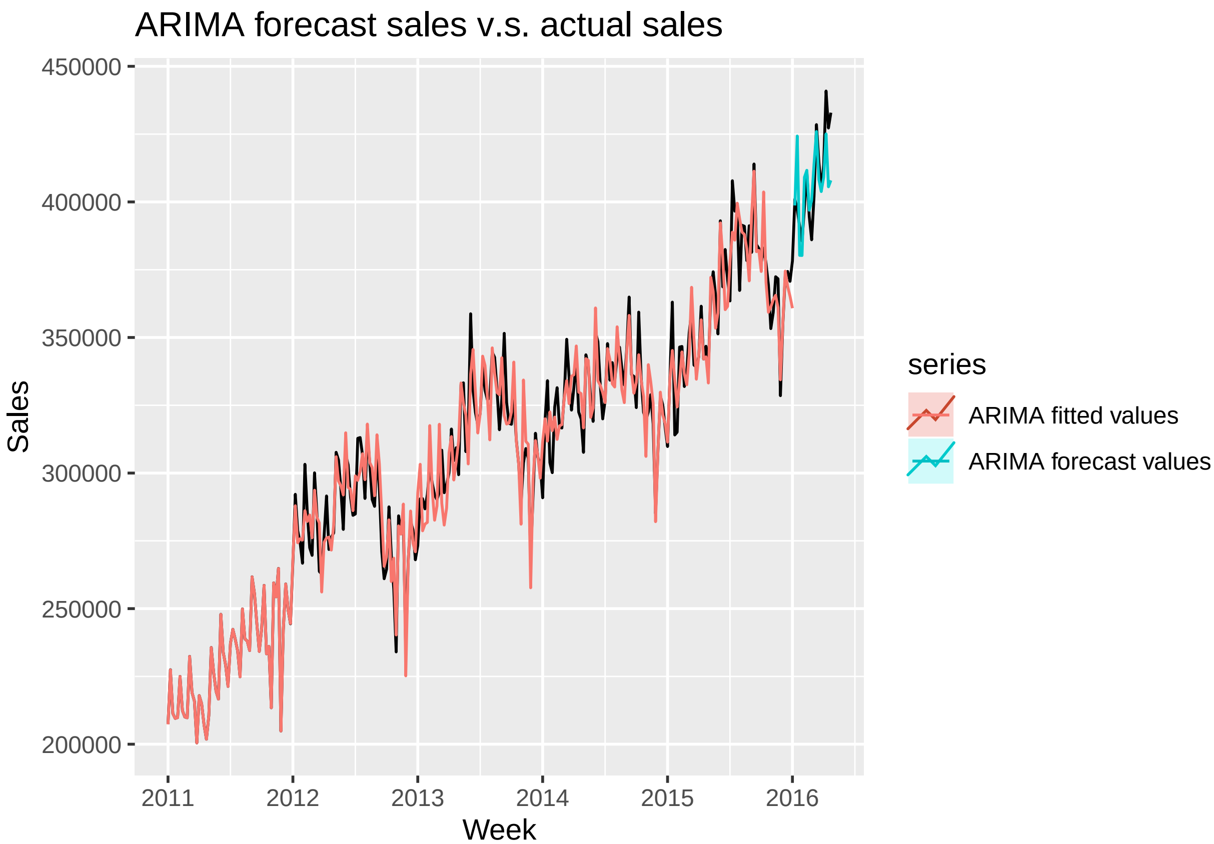

```r
# Arima Models
# AR(1) with seasonal component
ar1s = Arima(train, order=c(1, 1, 0),
             seasonal=list(order=c(1,1,0), period=4))
ar1s.pred = forecast(ar1s, h=ntest)
autoplot(test, series="actual") +
  autolayer(ar1s.pred, PI=FALSE, series="AR(1) with seasonal")
```




```
# compute the accuracy
accuracy(ar1s.pred, test)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set    138.7736 15473.72 12045.83 -0.06973452 4.001668 0.3576286
## Test set     -27719.6480 32609.12 27719.65 -6.75045051 6.750451 0.8229685
##                    ACF1 Theil's U
## Training set -0.1079529        NA
## Test set      0.5561199  2.364642
```

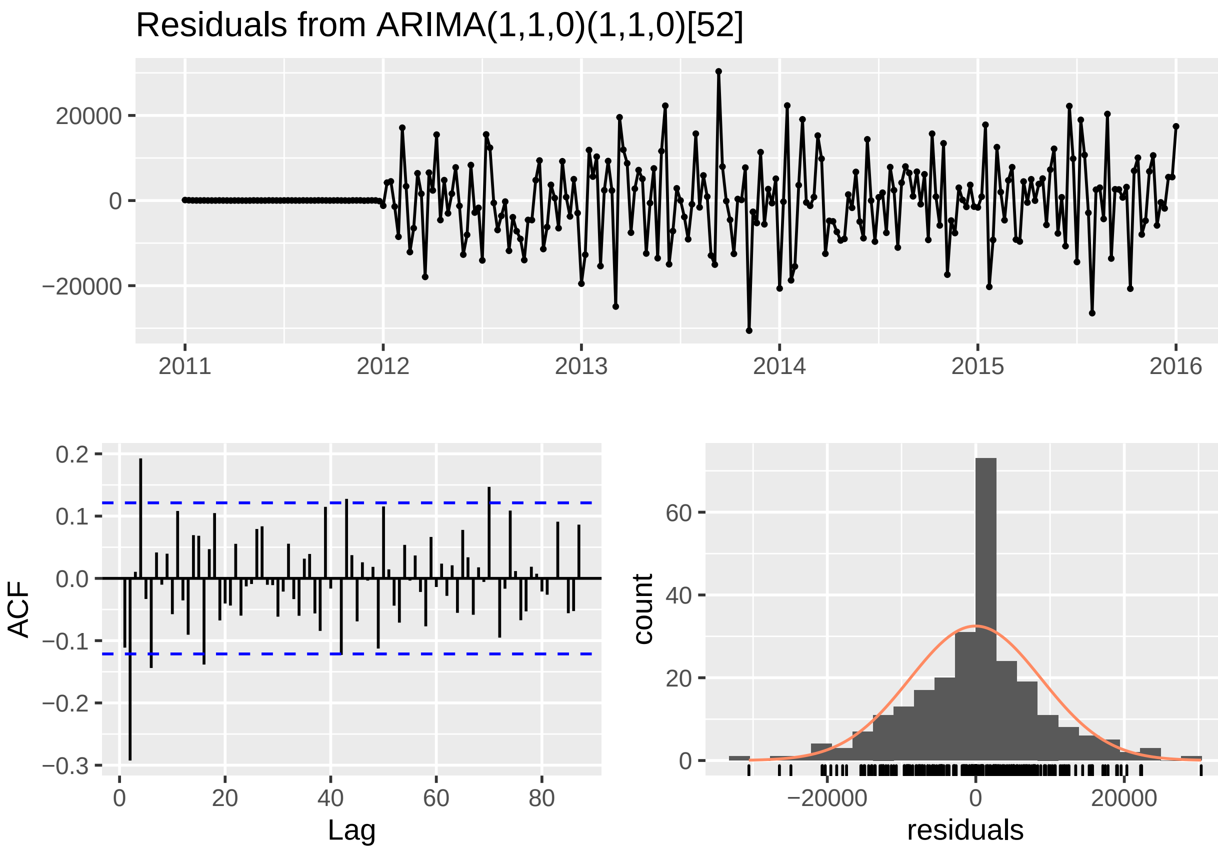
```
# AR(1) with seasonal component
ar1s = Arima(train, order=c(1, 1, 0),
             seasonal=list(order=c(1,1,0), period=52))
ar1s.pred = forecast(ar1s, h=ntest)
autoplot(ca.ts, series="actual values", colour="black",
         main="ARIMA forecast sales v.s. actual sales",
         xlab="Week",
         ylab="Sales") +
  autolayer(ar1s.pred, PI=FALSE, series="ARIMA forecast values")+
  autolayer(ar1s$fitted, series="ARIMA fitted values")
```




```r
# compute the accuracy
accuracy(ar1s.pred, test)
```

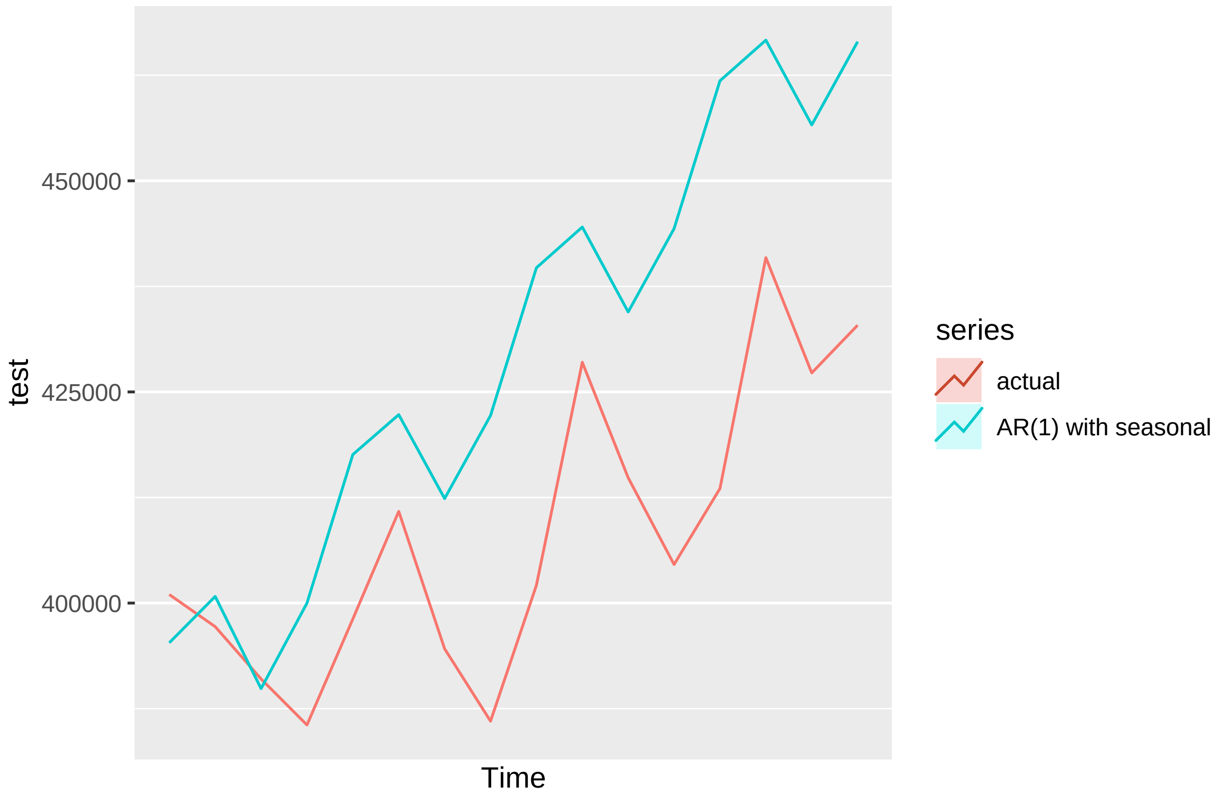
```
##                       ME      RMSE       MAE         MPE     MAPE      MASE
## Training set   -38.92526  8856.813  6143.531 -0.06971565 1.921195 0.1823953
## Test set      1724.76296 13170.261 10176.063  0.34397835 2.479117 0.3021171
##                     ACF1 Theil's U
## Training set -0.1112363        NA
## Test set      0.2963335 0.9438178
```

```r
checkresiduals(ar1s)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from ARIMA(1,1,0)(1,1,0)[52]
## Q* = 102.36, df = 50, p-value = 1.824e-05
##
## Model df: 2.   Total lags used: 52
```

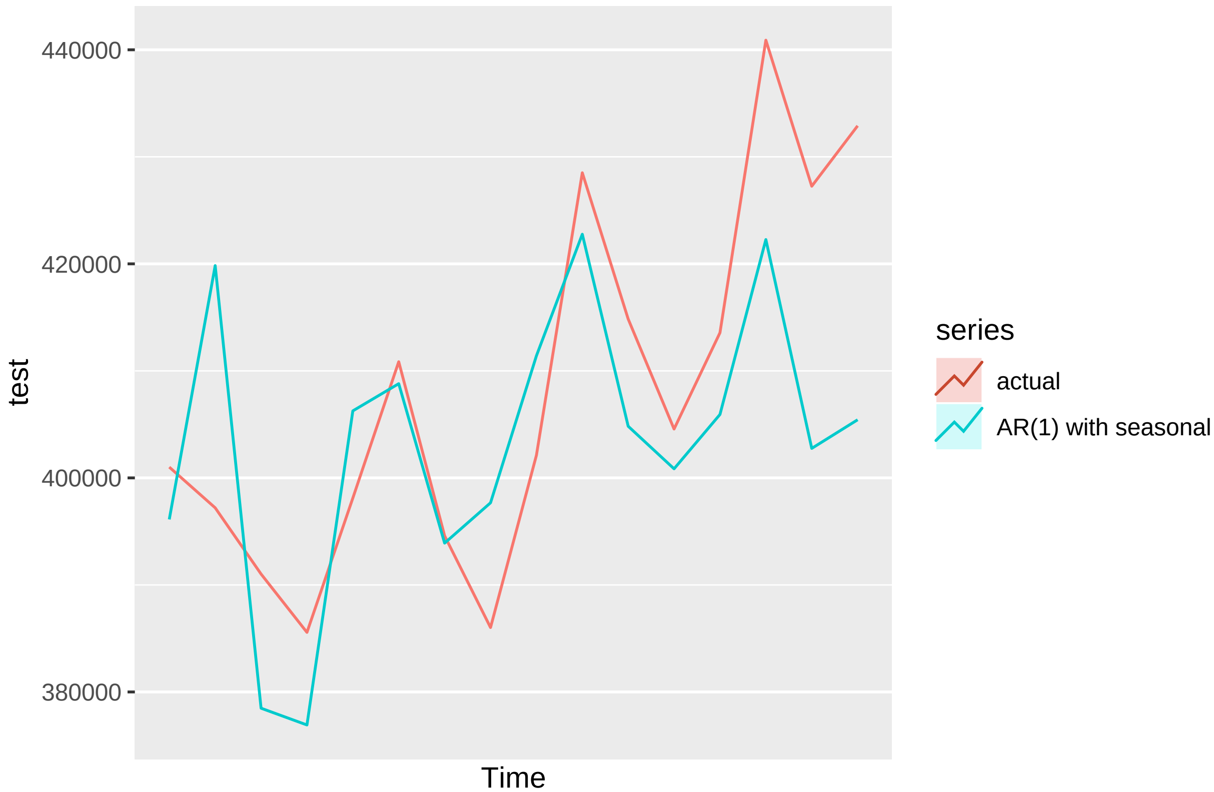
```
# AR(1) with seasonal component
ar1s = Arima(train, order=c(2, 1, 0),
             seasonal=list(order=c(1,1,0), period=4))
ar1s.pred = forecast(ar1s, h=ntest)
autoplot(test, series="actual") +
  autolayer(ar1s.pred, PI=FALSE, series="AR(1) with seasonal")
```




```
# compute the accuracy
accuracy(ar1s.pred, test)
```

```
##                      ME     RMSE      MAE         MPE     MAPE      MASE
## Training set   157.4983 14994.10 11730.79 -0.04671704 3.912661 0.3482753
## Test set     -21625.5185 26232.69 22482.49 -5.26710886 5.481722 0.6674825
##                     ACF1 Theil's U
## Training set -0.04295734        NA
## Test set      0.54558652  1.900552
```

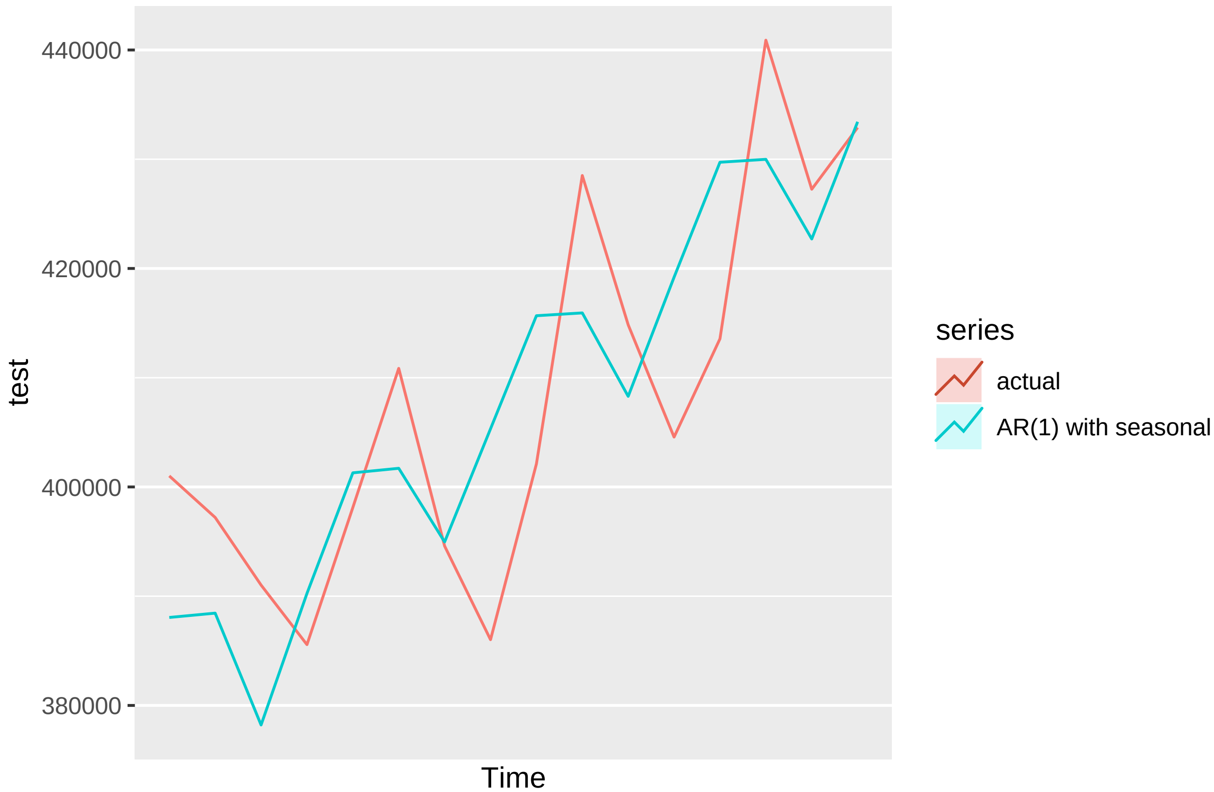
```
# AR(1) with seasonal component
ar1s = Arima(train, order=c(2, 1, 0),
             seasonal=list(order=c(1,1,0), period=52))
ar1s.pred = forecast(ar1s, h=ntest)
autoplot(test, series="actual") +
  autolayer(ar1s.pred, PI=FALSE, series="AR(1) with seasonal")
```




```
# compute the accuracy
accuracy(ar1s.pred, test)
```

```
##                       ME      RMSE       MAE         MPE     MAPE      MASE
## Training set  -66.23973  8555.365  5989.144 -0.07897297 1.878589 0.1778117
## Test set     4674.33475 13620.115 11139.783  1.06815719 2.702365 0.3307290
##                     ACF1 Theil's U
## Training set -0.06216826        NA
## Test set      0.32446033 0.9640792
```

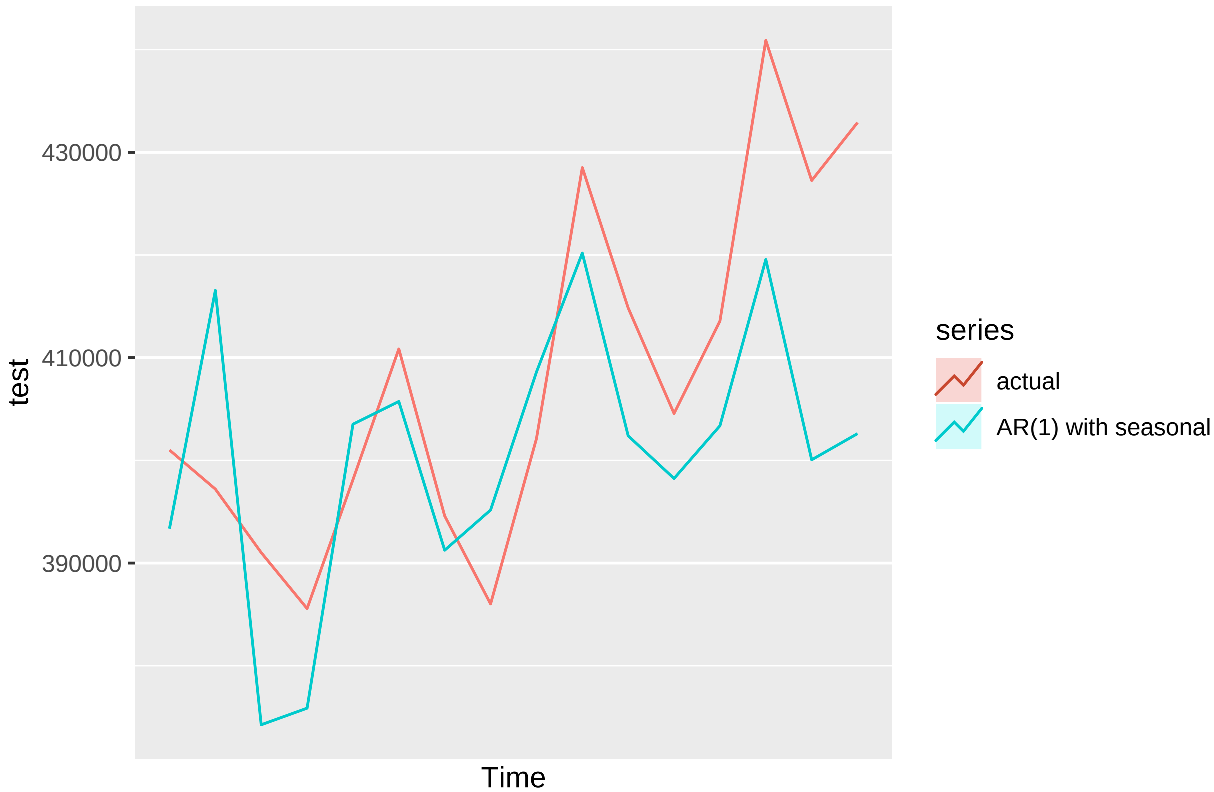
```
# AR(1) with seasonal component
ar1s = Arima(train, order=c(3, 1, 0),
             seasonal=list(order=c(1,1,0), period=4))
ar1s.pred = forecast(ar1s, h=ntest)
autoplot(test, series="actual") +
  autolayer(ar1s.pred, PI=FALSE, series="AR(1) with seasonal")
```




```r
# compute the accuracy
accuracy(ar1s.pred, test)
```

```
##                    ME     RMSE       MAE         MPE     MAPE      MASE
## Training set 147.6371 14590.69 11359.854 -0.03856936 3.770563 0.3372627
## Test set     364.8981 10890.11  9417.718  0.05402675 2.319994 0.2796026
##                     ACF1 Theil's U
## Training set -0.04303516        NA
## Test set      0.18083377 0.7655673
```

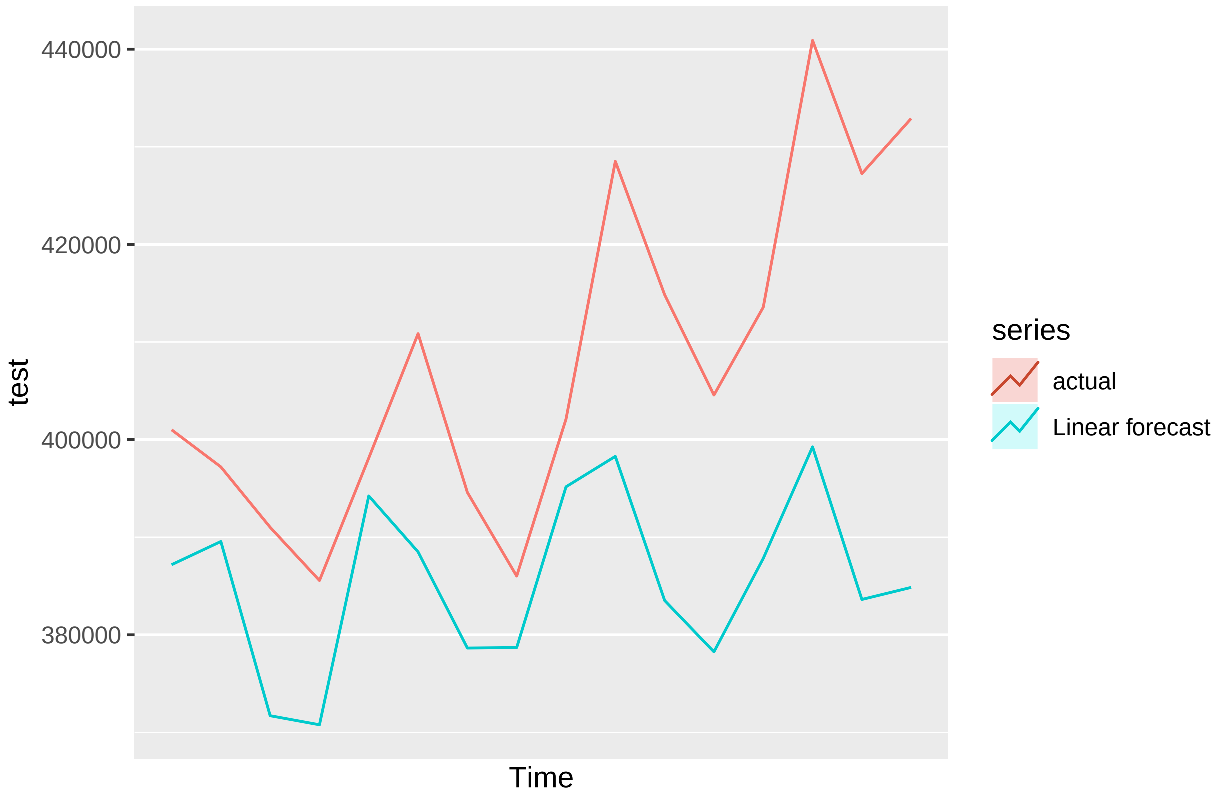
```r
# AR(1) with seasonal component
ar1s = Arima(train, order=c(3, 1, 0),
             seasonal=list(order=c(1,1,0), period=52))
ar1s.pred = forecast(ar1s, h=ntest)
autoplot(test, series="actual") +
  autolayer(ar1s.pred, PI=FALSE, series="AR(1) with seasonal")
```




```
# compute the accuracy
accuracy(ar1s.pred, test)
```

```
##                     ME      RMSE       MAE         MPE     MAPE      MASE
## Training set -111.4656  8337.967  5839.946 -0.09720462 1.822989 0.1733821
## Test set     7395.6393 14751.258 12443.853  1.73576978 3.012165 0.3694455
##                     ACF1 Theil's U
## Training set 0.004662744        NA
## Test set     0.315201926  1.034376
```

```
################################################
######## Linear Model #############################
m0 = tslm(train ~ trend+season)
m0.pred = forecast(m0, h=ntest)
autoplot(test, series="actual") +
  autolayer(m0.pred, PI=FALSE, series="Linear forecast")
```




```
accuracy(m0.pred, test)
```

```
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 1.562098e-12 13049.85 10944.37 -0.2471374 3.809508 0.3249273
## Test set     2.243199e+04 26108.15 22431.99  5.3942682 5.394268 0.6659832
##                   ACF1 Theil's U
## Training set 0.7696460        NA
## Test set     0.5892909  1.827701
```

```
##############################################################
######### Linear Regression with external variables #########
##############################################################
# put all variables
ca.food = ts(data$CA_FOODS, start=c(2011, 1), frequency = 52)
ca.hobbie = ts(data$CA_HOBBIES, start=c(2011, 1), frequency = 52)
ca.household = ts(data$CA_HOUSEHOLD, start=c(2011, 1), frequency = 52)
total.revenue = ts(data$Total.Revenue, start=c(2011, 1), frequency = 52)
total.event = ts(data$total_event, start=c(2011, 1), frequency = 52)


newdata = ts.intersect(ca.ts,
                       ca.lag1 = lag(ca.ts, -1),
                       tx.lag1 = lag(tx.ts, -1),
                       wi.lag1 = lag(wi.ts, -1),
                       ca.food.lag1 = lag(ca.food, -1),
                       ca.hobbie.lag1 = lag(ca.hobbie, -1),
                       ca.household.lag1 = lag(ca.household, -1),
                       total.revenue.lag1 = lag(total.revenue, -1),
                       total.event.lag1 = lag(total.event, -1),
```

```r
                        total.event)

newdata.df = data.frame(newdata)
ntrain = nrow(newdata.df) - ntest
train.new = newdata.df[1:ntrain,]
test.new = newdata.df[ntrain+1:nrow(newdata.df),]
test.new = test.new[1:ntest,]
ca.ts = ts(newdata.df$ca.ts, start=c(2011,2), frequency=52)
train.ts = ts(train.new$ca.ts, start=c(2011,2), frequency = 52)
test.ts = ts(test.new$ca.ts, start=c(2011, 2+ntrain), frequency = 52)

####################################
# fit linear model
# MAPE = 2.89
m1 = lm(ca.ts ~ ca.lag1+tx.lag1+wi.lag1+ca.food.lag1+ca.hobbie.lag1+ca.household.lag1+total.revenue.lag
summary(m1)
```

```
##
## Call:
## lm(formula = ca.ts ~ ca.lag1 + tx.lag1 + wi.lag1 + ca.food.lag1 +
##     ca.hobbie.lag1 + ca.household.lag1 + total.revenue.lag1 +
##     total.event, data = newdata.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -55973 -10085  -1128  10352  39979
##
## Coefficients: (2 not defined because of singularities)
##                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)         3.830e+04  7.808e+03   4.906 1.61e-06 ***
## ca.lag1             1.229e+00  1.927e-01   6.378 7.78e-10 ***
## tx.lag1            -1.930e-01  1.080e-01  -1.787   0.0751 .
## wi.lag1            -9.495e-02  6.338e-02  -1.498   0.1352
## ca.food.lag1       -4.406e-01  2.485e-01  -1.774   0.0773 .
## ca.hobbie.lag1      6.430e-01  5.052e-01   1.273   0.2042
## ca.household.lag1          NA         NA      NA       NA
## total.revenue.lag1         NA         NA      NA       NA
## total.event        -2.195e+03  1.490e+03  -1.473   0.1418
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16110 on 269 degrees of freedom
## Multiple R-squared:  0.9107, Adjusted R-squared:  0.9087
## F-statistic: 457.1 on 6 and 269 DF,  p-value: < 2.2e-16
```

```r
m1.pred = predict(m1, test.new)
```

```
## Warning in predict.lm(m1, test.new): prediction from a rank-deficient fit
## may be misleading
```

```r
accuracy(m1$fitted.values, train.new$ca.ts)
```

```
##                ME     RMSE      MAE        MPE     MAPE
## Test set -649.33 15941.13 12386.82 -0.4830225 4.248428
```

```r
accuracy(m1.pred, test.new$ca.ts)
```

```
##                  ME     RMSE      MAE      MPE     MAPE
## Test set 10551.61 15371.05 12019.49 2.539595 2.898962
```

```r
####################################
# fit linear model
# MAPE = 2.89
m1 = lm(ca.ts ~ ca.lag1+tx.lag1+wi.lag1+ca.food.lag1+ca.hobbie.lag1+total.event, data=newdata.df)
summary(m1)
```

```
##
## Call:
## lm(formula = ca.ts ~ ca.lag1 + tx.lag1 + wi.lag1 + ca.food.lag1 +
##     ca.hobbie.lag1 + total.event, data = newdata.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -55973 -10085  -1128  10352  39979
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)      3.830e+04  7.808e+03   4.906 1.61e-06 ***
## ca.lag1          1.229e+00  1.927e-01   6.378 7.78e-10 ***
## tx.lag1         -1.930e-01  1.080e-01  -1.787   0.0751 .
## wi.lag1         -9.495e-02  6.338e-02  -1.498   0.1352
## ca.food.lag1    -4.406e-01  2.485e-01  -1.774   0.0773 .
## ca.hobbie.lag1   6.430e-01  5.052e-01   1.273   0.2042
## total.event     -2.195e+03  1.490e+03  -1.473   0.1418
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16110 on 269 degrees of freedom
## Multiple R-squared:  0.9107, Adjusted R-squared:  0.9087
## F-statistic: 457.1 on 6 and 269 DF,  p-value: < 2.2e-16
```

```r
m1.pred = predict(m1, test.new)
accuracy(m1$fitted.values, train.new$ca.ts)
```

```
##                ME     RMSE      MAE        MPE     MAPE
## Test set -649.33 15941.13 12386.82 -0.4830225 4.248428
```

```r
accuracy(m1.pred, test.new$ca.ts)
```

```
##                  ME     RMSE      MAE      MPE     MAPE
## Test set 10551.61 15371.05 12019.49 2.539595 2.898962
```

```r
####################################
# fit linear model
# MAPE = 2.97
m1 = lm(ca.ts ~ ca.lag1+tx.lag1+wi.lag1+ca.food.lag1+total.event, data=newdata.df)
summary(m1)
```

```
##
## Call:
## lm(formula = ca.ts ~ ca.lag1 + tx.lag1 + wi.lag1 + ca.food.lag1 +
##     total.event, data = newdata.df)
```

```
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -54335  -9701  -1340  10545  38913
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.756e+04  7.796e+03   4.819 2.41e-06 ***
## ca.lag1       1.448e+00  8.711e-02  16.625  < 2e-16 ***
## tx.lag1      -1.890e-01  1.081e-01  -1.748   0.0815 .
## wi.lag1      -6.682e-02  5.947e-02  -1.124   0.2621
## ca.food.lag1 -7.168e-01  1.211e-01  -5.917 9.90e-09 ***
## total.event  -1.921e+03  1.476e+03  -1.302   0.1940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16130 on 270 degrees of freedom
## Multiple R-squared:  0.9101, Adjusted R-squared:  0.9085
## F-statistic:   547 on 5 and 270 DF,  p-value: < 2.2e-16
```

```
m1.pred = predict(m1, test.new)
accuracy(m1$fitted.values, train.new$ca.ts)
```

```
##                  ME     RMSE      MAE        MPE     MAPE
## Test set -639.6102 15978.81 12444.18 -0.4809176 4.266645
```

```
accuracy(m1.pred, test.new$ca.ts)
```

```
##                 ME    RMSE     MAE     MPE     MAPE
## Test set 10393.67 15588.9 12289.2 2.50378 2.965932
```

```
####################################
# fit linear model
# MAPE = 2.99
m1 = lm(ca.ts ~ ca.lag1+tx.lag1+ca.food.lag1+total.event, data=newdata.df)
summary(m1)
```

```
##
## Call:
## lm(formula = ca.ts ~ ca.lag1 + tx.lag1 + ca.food.lag1 + total.event,
##     data = newdata.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -53228  -9826  -1320  10572  40463
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.168e+04  6.884e+03   6.056 4.65e-09 ***
## ca.lag1       1.428e+00  8.528e-02  16.745  < 2e-16 ***
## tx.lag1      -2.440e-01  9.640e-02  -2.532   0.0119 *
## ca.food.lag1 -7.111e-01  1.211e-01  -5.872 1.25e-08 ***
## total.event  -2.082e+03  1.470e+03  -1.417   0.1577
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 16140 on 271 degrees of freedom
## Multiple R-squared:  0.9097, Adjusted R-squared:  0.9084
## F-statistic: 682.7 on 4 and 271 DF,  p-value: < 2.2e-16
```

```
m1.pred = predict(m1, test.new)
accuracy(m1$fitted.values, train.new$ca.ts)
```

```
##                  ME     RMSE      MAE        MPE     MAPE
## Test set -590.224 16017.87 12441.12 -0.4696937 4.262766
```

```
accuracy(m1.pred, test.new$ca.ts)
```

```
##                  ME     RMSE      MAE      MPE     MAPE
## Test set 9591.141 15596.33 12428.35 2.303799 2.998391
```

```
####################################
# fit linear model
# MAPE = 2.94
m1 = lm(ca.ts ~ ca.lag1+tx.lag1+ca.food.lag1, data=newdata.df)
summary(m1)
```

```
##
## Call:
## lm(formula = ca.ts ~ ca.lag1 + tx.lag1 + ca.food.lag1, data = newdata.df)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -56188  -9385  -1051  10645  39353
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   4.132e+04  6.891e+03   5.996 6.42e-09 ***
## ca.lag1       1.450e+00  8.399e-02  17.265  < 2e-16 ***
## tx.lag1      -2.687e-01  9.499e-02  -2.828  0.00503 **
## ca.food.lag1 -7.271e-01  1.208e-01  -6.020 5.62e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 16170 on 272 degrees of freedom
## Multiple R-squared:  0.9091, Adjusted R-squared:  0.9081
## F-statistic: 906.3 on 3 and 272 DF,  p-value: < 2.2e-16
```

```
m1.pred = predict(m1, test.new)
accuracy(m1$fitted.values, train.new$ca.ts)
```

```
##                   ME     RMSE      MAE        MPE     MAPE
## Test set -571.4412 16110.58 12413.14 -0.4687906 4.250787
```

```
accuracy(m1.pred, test.new$ca.ts)
```

```
##                 ME     RMSE      MAE      MPE     MAPE
## Test set 9285.92 15084.14 12166.46 2.230467 2.936501
```

```
######################################
# final model with all the data #
# We could use this final model to make prediction in real business settings
##################################
final.m = Arima(ca.ts, order=c(1, 1, 0),
```

```
             seasonal=list(order=c(1,1,0), period=52))
summary(final.m)
```

```
## Series: ca.ts
## ARIMA(1,1,0)(1,1,0)[52]
##
## Coefficients:
##           ar1     sar1
##       -0.4004  -0.3726
## s.e.   0.0615   0.0694
##
## sigma^2 estimated as 103913790:  log likelihood=-2377.57
## AIC=4761.14   AICc=4761.25   BIC=4771.36
##
## Training set error measures:
##                    ME    RMSE      MAE         MPE     MAPE      MASE
## Training set 52.68919 9121.75 6336.057 -0.04978241 1.941208 0.1766316
##                    ACF1
## Training set -0.1233591
```

```
final.m.pred = forecast(final.m, h=16)
final.m.pred
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2016.327       421602.8 408538.9 434666.7 401623.3 441582.3
## 2016.346       453227.7 437995.1 468460.4 429931.5 476524.0
## 2016.365       456466.0 438283.9 474648.2 428658.9 484273.2
## 2016.385       446420.4 426092.9 466747.9 415332.2 477508.6
## 2016.404       434381.3 411975.1 456787.4 400114.0 468648.5
## 2016.423       472874.9 448619.0 497130.9 435778.6 509971.2
## 2016.442       456228.0 430234.4 482221.6 416474.2 495981.8
## 2016.462       458850.1 431235.2 486465.0 416616.7 501083.5
## 2016.481       447296.4 418147.5 476445.3 402717.0 491875.9
## 2016.500       444892.9 414287.8 475497.9 398086.4 491699.3
## 2016.519       480538.3 448542.8 512533.7 431605.4 529471.1
## 2016.538       468464.4 435136.7 501792.1 417494.1 519434.7
## 2016.558       470913.2 436304.4 505521.9 417983.6 523842.7
## 2016.577       450074.9 414230.8 485919.0 395256.1 504893.7
## 2016.596       468432.9 431394.7 505471.1 411787.9 525077.9
## 2016.615       469508.7 431313.7 507703.7 411094.5 527922.9
```