0 Smoothing Model

```
library(zoo)
```

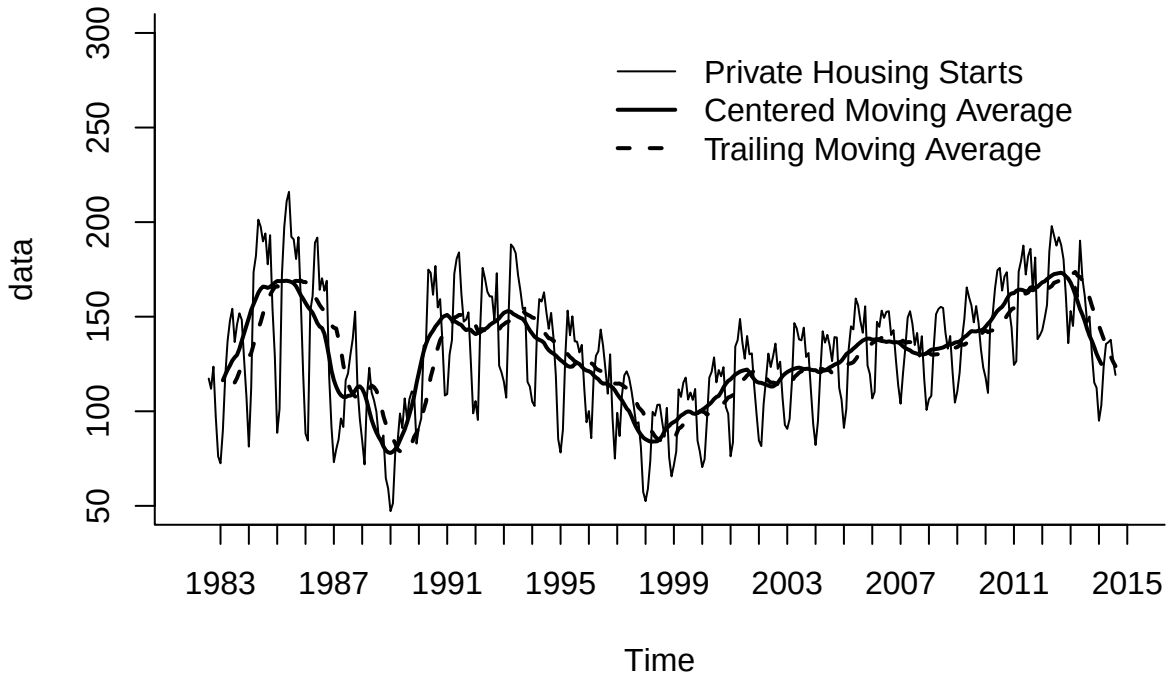
```
##
## Attaching package: 'zoo'
```

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

```
library(forecast)
library(ggplot2)
df = read.csv("data.csv")  # PrivateHousingStarts
data = ts(df$Private.Housing.Starts, start = c(1982,08), frequency = 12)
```

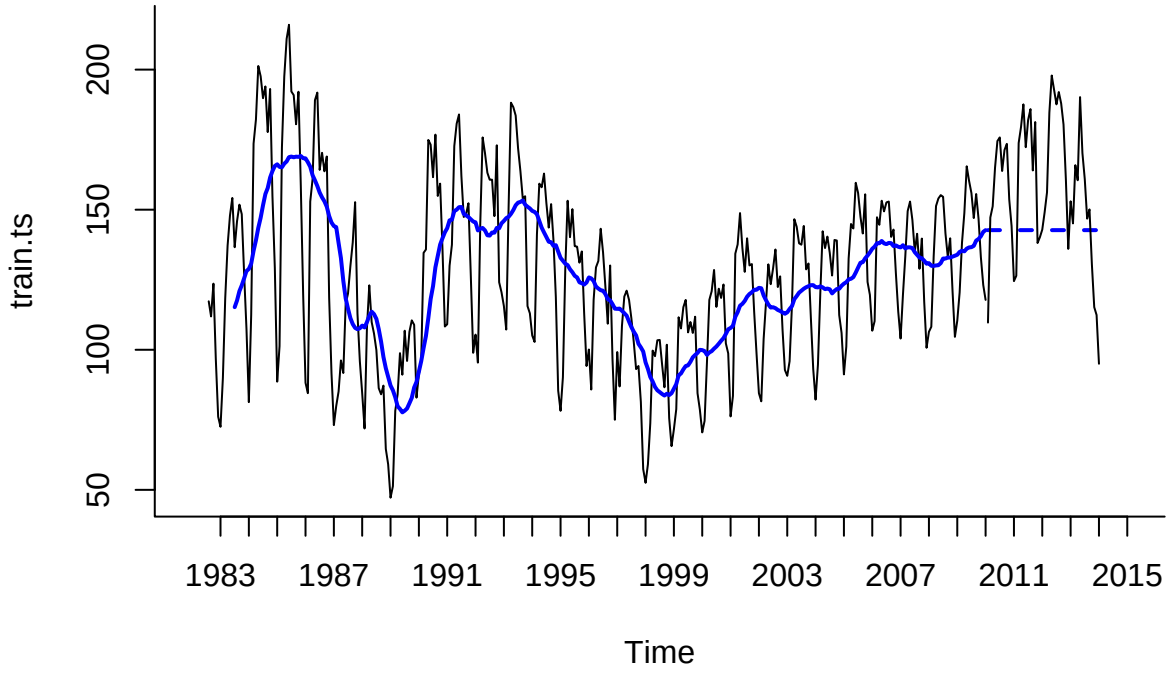
1 Moving Average

```
ma.trailing = rollmean(data, k = 12, align = "right")
ma.center = ma(data, order = 12)   ## ma is in forecast library

plot(data, ylim = c(50,300) ,bty = "l", xaxt = "n", xlim = c(1982, 2015),main = "")
axis(1,at = seq(1983, 2015,1), labels = format(seq(1983, 2015,1)))
lines(ma.center, lwd = 2)
lines(ma.trailing, lwd = 2, lty = 2)
legend(1996,300,c("Private Housing Starts","Centered Moving Average", "Trailing Moving Average"), lty =
```



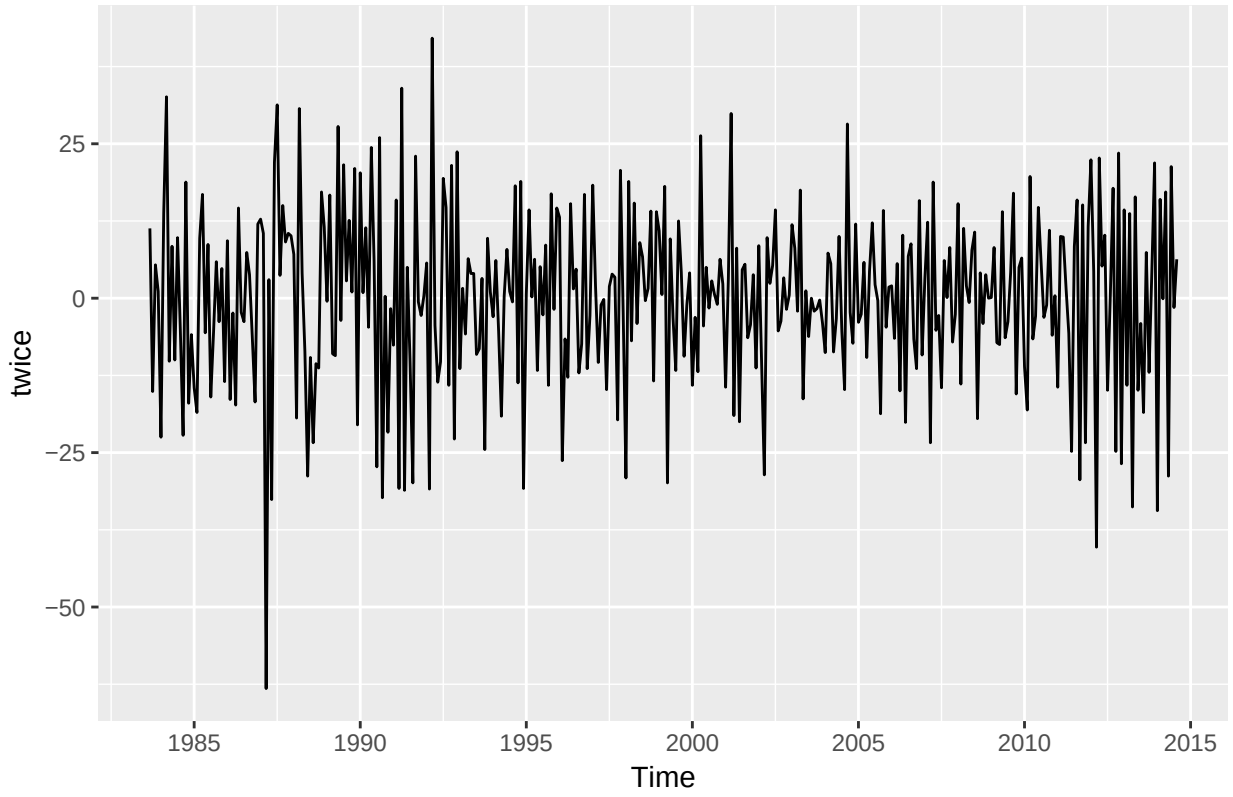
2 Using validation set, naive method

```
nvalid = 48
ntrain = length(data) - nvalid
train.ts = window(data, start = c(1982,08), end = c(1982, ntrain))
valid.ts = window(data, start = c(1982, ntrain+1), end = c(1982, ntrain+nvalid))
ma.trailing = rollmean(train.ts, k = 12, align = "right")
last.ma = tail(ma.trailing,1)
ma.trailing.pred = ts(rep(last.ma, nvalid), start = c(1982, ntrain+1),end = c(1982, ntrain+nvalid),freq
plot(train.ts, xlab = "Time", bty = "l", xaxt = "n",xlim = c(1982, 2015))
axis(1,at = seq(1983, 2015,1), labels = format(seq(1983, 2015,1)))
lines(ma.trailing, lwd = 2, col = 'blue')
lines(ma.trailing.pred, lwd = 2, col = "blue", lty=2)
lines(valid.ts)
```



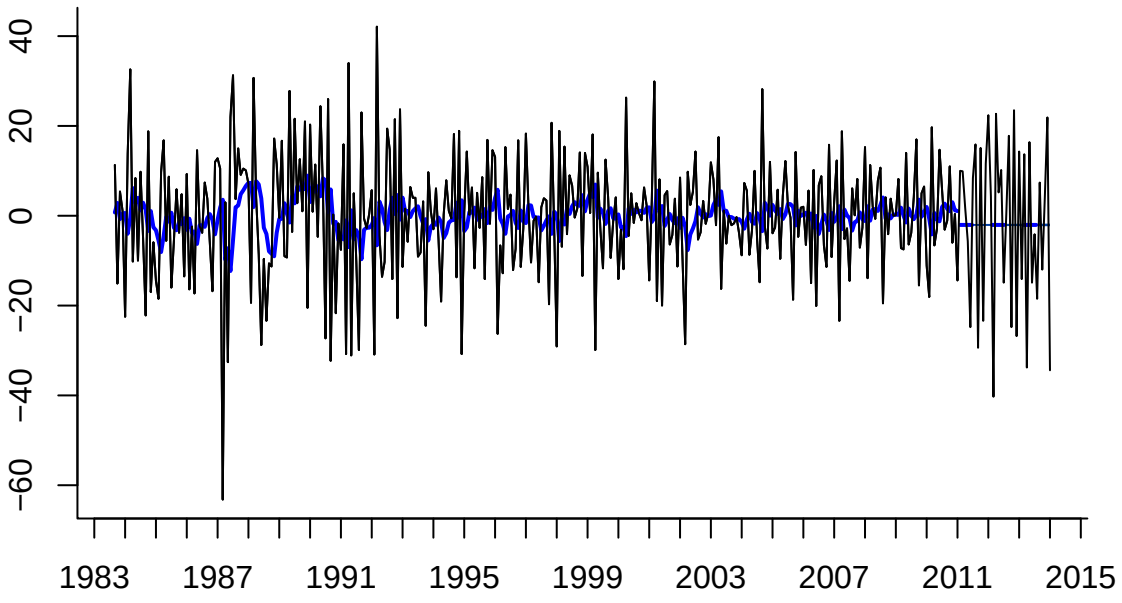
3 Simple Exponential Smoothing, user Twice differenced data

```
total = nrow(df)
df["First"] = 0
df["Second"] = 0
for(i in 13:total){
  df[i,3] = df[i,2] - df[i-12,2]
}

for(i in 14:total){
  df[i,4] = df[i,3] - df[i-1,3]
}

twice = ts(df$Second[14:total], start = c(1983,9), frequency = 12)
autoplot(twice)
```



OR:

```
diff.twice.ts = diff(diff(data, lag = 12), lag = 1)
nvalid = 36
ntrain = length(diff.twice.ts) - nvalid
train.ts = window(diff.twice.ts, start = c(1983,9), end = c(1983,ntrain+1))
valid.ts = window(diff.twice.ts, start = c(1983,9), end = c(1983, ntrain+1+nvalid))
ses = ets(train.ts, model = "ANN", alpha = 0.2)
ses.pred = forecast(ses, h = nvalid, level = 1)

plot(ses.pred, bty = "l", xaxt = "n", main = "", flty = 2)
axis(1, at = seq(1983,2015,1), labels = format(seq(1983,2015,1)))
lines(ses.pred$fitted, lwd = 2, col = "blue")
lines(valid.ts)
```



Optimized Alpha, ets will automatically optimize the alpha

```
ses.opt = ets(train.ts, model = "ANN")
ses.opt.pred = forecast(ses.opt, h = nvalid, level=1)
ses.opt
```

```
## ETS(A,N,N)
##
## Call:
##  ets(y = train.ts, model = "ANN")
##
##   Smoothing parameters:
##     alpha = 1e-04
##
##   Initial states:
##     l = -0.0633
##
##   sigma:  13.7374
##
##      AIC     AICc      BIC
## 3634.936 3635.009 3646.324
```

```
accuracy(ses.pred,valid.ts)
```

```
##                       ME     RMSE      MAE           MPE         MAPE
## Training set -0.04229546 14.88718 11.41804 2.658360e+12 1.017380e+13
## Test set      0.24581895 18.64225 16.42662 1.193587e+02 1.193587e+02
##                    MASE       ACF1 Theil's U
## Training set 0.6465274 -0.4037169        NA
## Test set     0.9301297 -0.5311250  1.099226
```

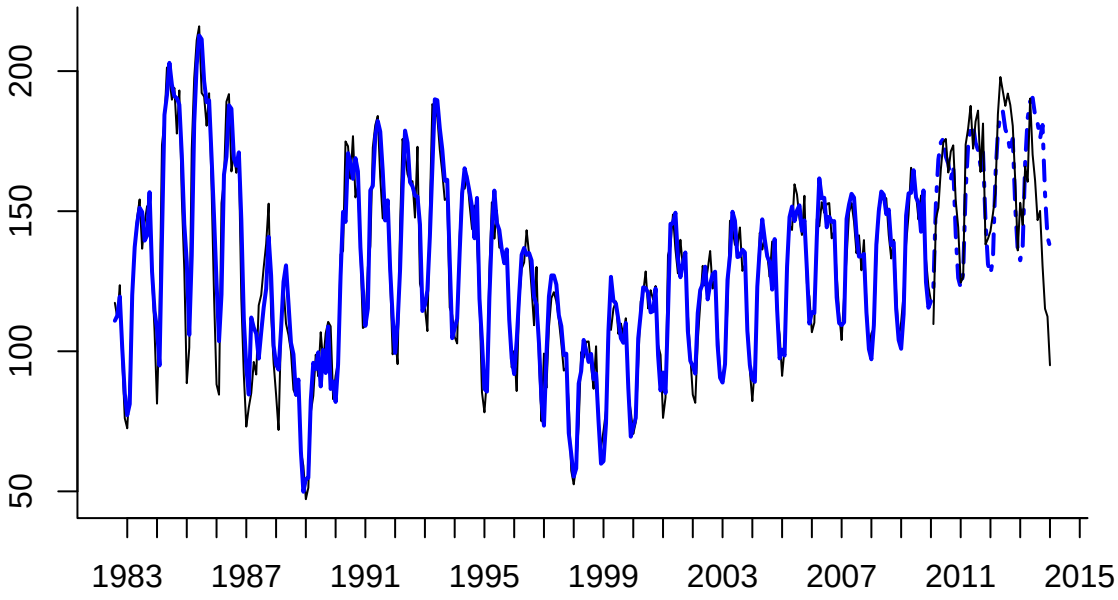
```
accuracy(ses.opt.pred,valid.ts)
```

```
##                       ME     RMSE      MAE           MPE         MAPE
## Training set -0.001656481 13.69555 10.56024 -2.045142e+12 2.045142e+12
```

```
## Test set     -1.733908351 18.72110 16.09666  1.005999e+02 1.005999e+02
##                     MASE       ACF1 Theil's U
## Training set 0.5979558 -0.3512214        NA
## Test set     0.9114465 -0.5311250   1.00724
```

4 Advanced Exponential Smoothing, considering trend, level and seasonality

```
nvalid = 48
ntrain = length(data) - nvalid
train.ts = window(data, start = c(1982,08), end = c(1982, ntrain))
valid.ts = window(data, start = c(1982, ntrain+1), end = c(1982, ntrain+nvalid))
hwin = ets(train.ts, model = "MAA")
hwin.pred = forecast(hwin, h = nvalid, level = 0)
plot(hwin.pred, bty = "l", xaxt = "n", main="", flty = 12)
axis(1, at = seq(1983,2019,1), labels = format(seq(1983,2019,1)))
lines(hwin.pred$fitted, lwd = 2, col = "blue")
lines(valid.ts)
```




```
hwin
```

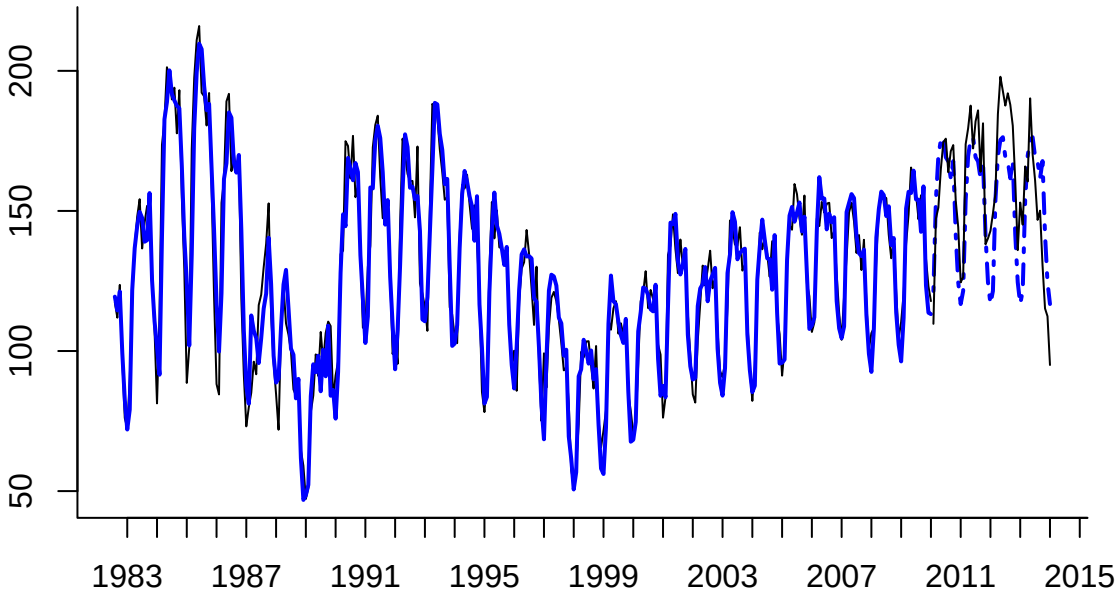
```
## ETS(M,A,A)
##
## Call:
##  ets(y = train.ts, model = "MAA")
##
##   Smoothing parameters:
##     alpha = 0.7673
##     beta  = 0.0065
##     gamma = 1e-04
##
##   Initial states:
##     l = 95.351
```

```
##     b = 2.9308
##     s = 15.6508 22.3843 21.5088 15.8373 0.3781 -28.8891
##           -33.5903 -29.7201 -13.8271 11.0436 6.6064 12.6172
##
##   sigma:  0.0927
##
##      AIC     AICc      BIC
## 3535.805 3537.766 3600.389
```

model = "ZZZ", automaticaly select model ,first error, second trend, third seasonal

damped = T, sometimes if choose an exact model type, need to set restrict to be true

```
auto = ets(train.ts, model = "ZZZ")
auto.pred = forecast(auto, h = nvalid, level = 0)
plot(auto.pred, bty = "l", xaxt = "n", main="", flty = 12)
axis(1, at = seq(1983,2019,1), labels = format(seq(1983,2019,1)))
lines(auto.pred$fitted, lwd = 2, col = "blue")
lines(valid.ts)
```



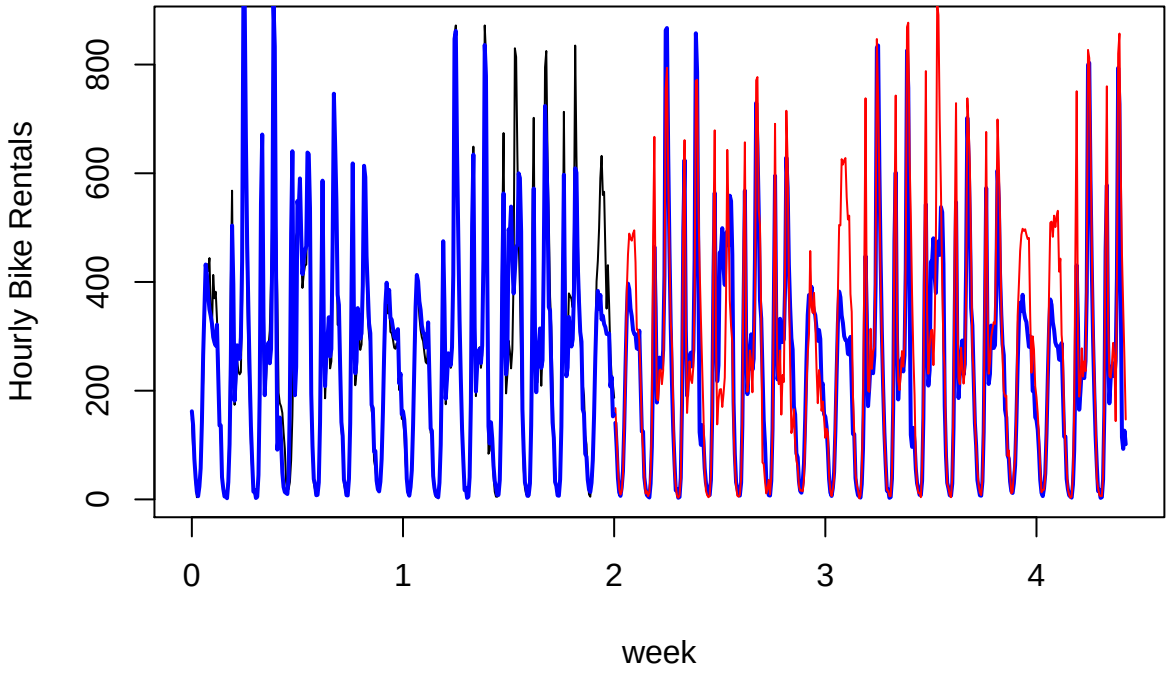
5 Multi seasonal, hour-of-day, day-of-week, nested periods, longer period divided by the shorter period must be an integer

```
bike.hourly.df = read.csv("BikeSharingHourly.csv")
ntotal = length(bike.hourly.df$cnt[13004:13747])
bike.hourly.msts = msts(bike.hourly.df$cnt[13004:13747], seasonal.periods = c(24,168), start = c(0,1))
ntain = 21*24
nvalid =ntotal - ntrain
ytrain.msts = window(bike.hourly.msts, start = c(0,1), end = c(0,ntrain))
yvalid.msts = window(bike.hourly.msts, start = c(0, ntrain+1), end = c(0,ntotal))
```

```
bike.hourly.dshw.pred = dshw(ytrain.msts, h = nvalid)
##  These two means are equal : bike.hourly.dshw.pred$mean - bike.hourly.dshw.pred.mean
bike.hourly.dshw.pred.mean = msts(bike.hourly.dshw.pred$mean, seasonal.periods = c(24,168), start = c(0
accuracy(bike.hourly.dshw.pred.mean, yvalid.msts)
```

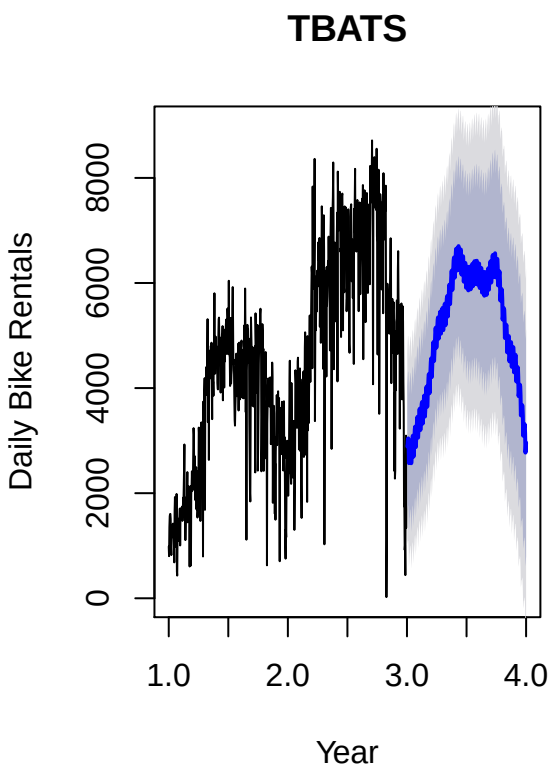
```
##                 ME     RMSE      MAE       MPE     MAPE      ACF1 Theil's U
## Test set 30.29324 103.3914 65.16892 0.4644289 28.99982 0.7332371 0.4381055
```

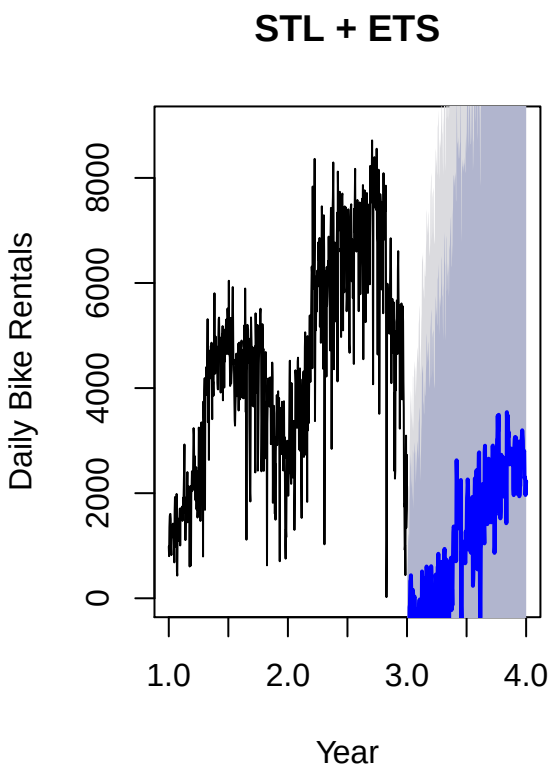
```
plot(ytrain.msts, xlim = c(0,4+3/7), xlab = "week", ylab = "Hourly Bike Rentals")
lines(bike.hourly.dshw.pred.mean, lwd = 2, col = "blue")
lines(yvalid.msts, lwd = 1, col = "red")
lines(bike.hourly.dshw.pred$fitted, lwd = 2, col = "blue")
```



tbats & stlm for not nested periods

```
bike.daily.df = read.csv("BikeSharingDaily.csv")
bike.daily.msts = msts(bike.daily.df$cnt, seasonal.periods = c(7, 365.25))

bike.daily.tbats = tbats(bike.daily.msts)
bike.daily.tbats.pred = forecast(bike.daily.tbats, h = 365)

bike.daily.stlm = stlm(bike.daily.msts, s.window = "periodic", method = "ets")
bike.daily.stlm.pred = forecast(bike.daily.stlm,h = 365)
par(mfrow = c(1,2))
plot(bike.daily.tbats.pred, ylim = c(0,9000) ,xlab = "Year", ylab = "Daily Bike Rentals", main = "TBATS
plot(bike.daily.stlm.pred, ylim = c(0,9000) ,xlab = "Year", ylab = "Daily Bike Rentals", main = "STL + E
```


TBATS
Daily Bike Rentals
0
2000
4000
6000
8000
1.0
2.0
3.0
4.0
Year


STL + ETS
Daily Bike Rentals
0
2000
4000
6000
8000
1.0
2.0
3.0
4.0
Year