Forecasting product demand

Taral Desai

05/08/2021

Required libraries

```
library(readr)
library(xts)
```

```
## Loading required package: zoo
```

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

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

```
library(forecast)
```

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

Importing Data

There are a lot of ways to import data into R! Once the data is imported into R, we need to transform the data into an xts object to help with analysis. These xts objects are so much easier to plot and manipulate.

```
Bev <- read_csv("Downloads/Bev.csv")
```

```
## Rows: 176 Columns: 14
```

```
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (14): M.hi.p, M.lo.p, MET.hi.p, MET.lo.p, MET.sp.p, SEC.hi.p, SEC.lo.p, ...
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
head(Bev)
```

```
## # A tibble: 6 x 14
##   M.hi.p M.lo.p MET.hi.p MET.lo.p MET.sp.p SEC.hi.p SEC.lo.p  M.hi  M.lo MET.hi
##    <dbl>  <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl> <dbl> <dbl>  <dbl>
## 1   59.2   29.2     63.7     26.0     50.1     58.6     29.2   458  1455   2037
## 2   56.3   26.3     60.3     25.5     48.8     54.6     26.3   477  1756   1700
## 3   56.3   26.2     60.8     25.7     48.6     57.9     26.2   539  2296   1747
## 4   49.3   26.2     55.1     26.5     47.7     49.7     26.2   687  3240   2371
## 5   61.3   25.9     65.1     25.7     50.8     63.7     25.9   389  2252   1741
```

```
## 6   61.4   27.4     67.9     26.2     52.6     68.4     27.4   399  1901   2072
## # ... with 4 more variables: MET.lo <dbl>, MET.sp <dbl>, SEC.hi <dbl>,
## #   SEC.lo <dbl>
```

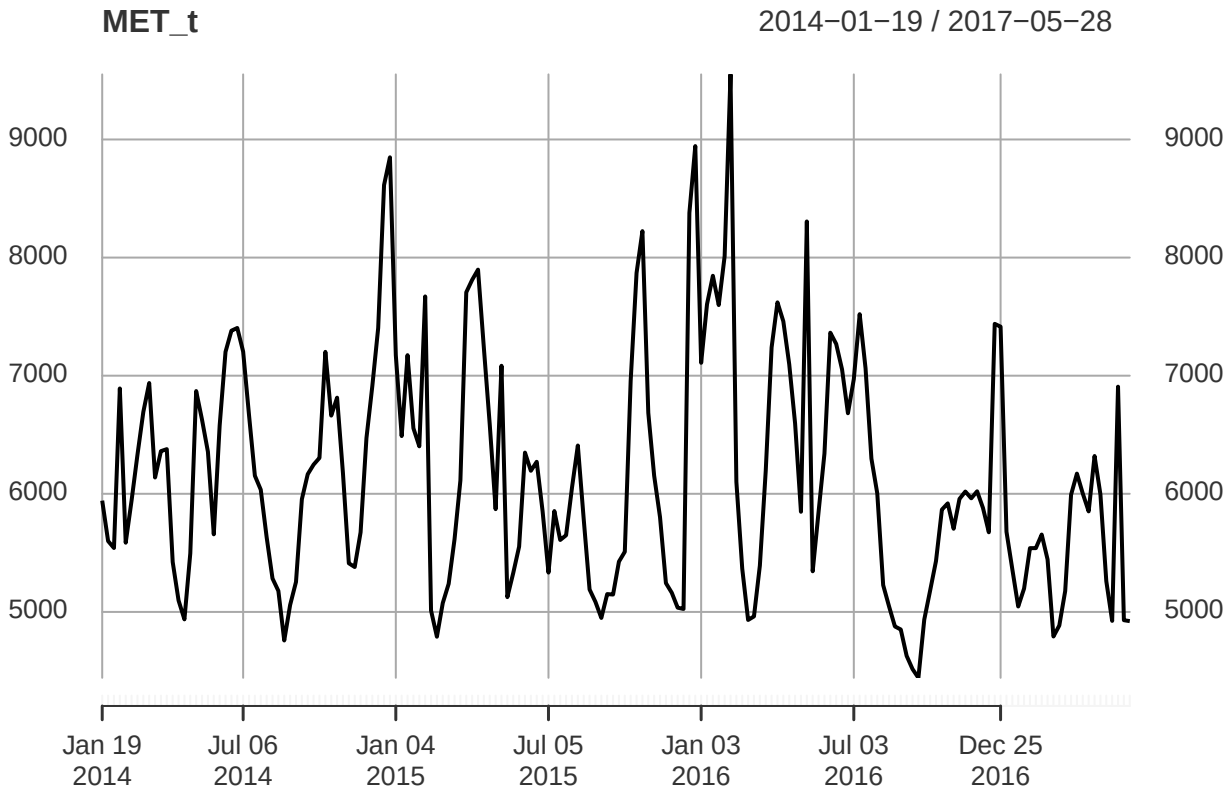
```
# Load xts package
library(xts)

# Create the dates object as an index for your xts object
dates <- seq(as.Date("2014-01-19"), length = 176, by = "weeks")

# Create an xts object called bev_xts
bev_xts <- xts(Bev, order.by = dates)
head(bev_xts[,"M.hi"], n = 10)
```

```
##            M.hi
## 2014-01-19  458
## 2014-01-26  477
## 2014-02-02  539
## 2014-02-09  687
## 2014-02-16  389
## 2014-02-23  399
## 2014-03-02  392
## 2014-03-09  417
## 2014-03-16  568
## 2014-03-23  583
```

Visualising Data

There are three products in the metropolitan areas - high end, low end, and specialty. The specialty product is not sold any where else in the state. The column names for the sales of these three products are MET.hi, MET.lo, and MET.sp respectively. Before looking at each one of these products individually, let's plot how total sales are going in the metropolitan region.

```
# Creating the individual region sales as their own objects
MET_hi <- bev_xts[,"MET.hi"]
MET_lo <- bev_xts[,"MET.lo"]
MET_sp <- bev_xts[,"MET.sp"]

# Sum the region sales together
MET_t <- MET_hi + MET_lo + MET_sp

# Plot the metropolitan region total sales
plot(MET_t)
```



ARIMA MODEL

-Auto Regressive Models -Integrated -Moving Average

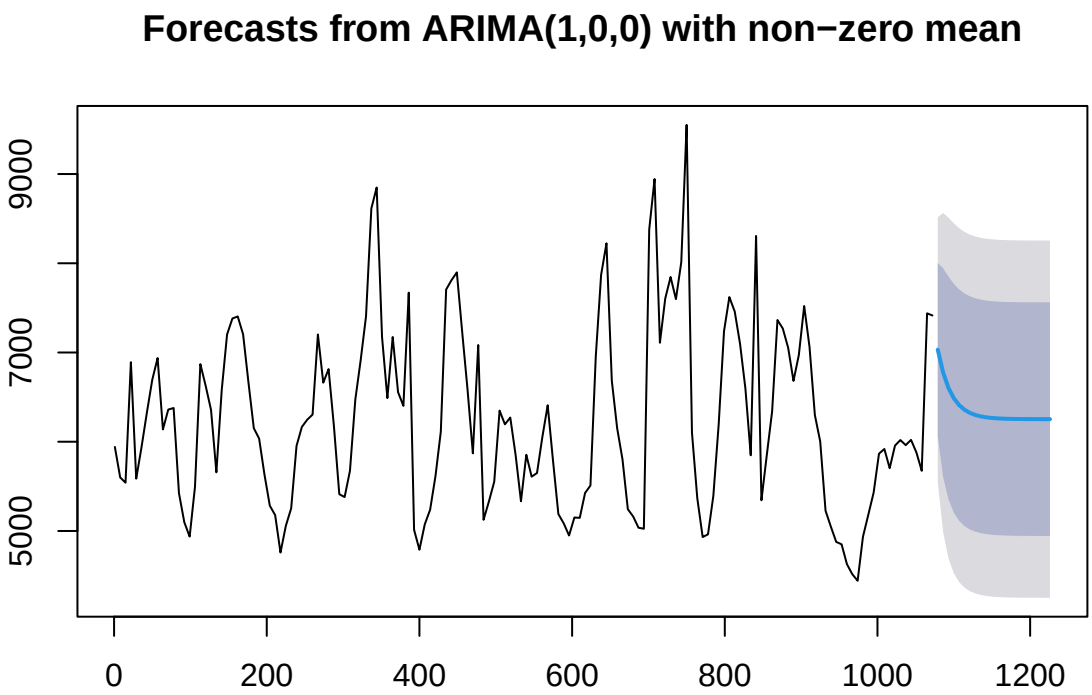
```
# Split the data into training and validation
MET_t_train <- MET_t[index(MET_t) < "2017-01-01"]
MET_t_valid <- MET_t[index(MET_t) >= "2017-01-01"]

# Use auto.arima() function for metropolitan sales training data
MET_t_model = auto.arima(MET_t_train)
MET_t_model
```

```
## Series: MET_t_train
## ARIMA(1,0,0) with non-zero mean
##
## Coefficients:
##          ar1       mean
##       0.6706  6252.8350
## s.e.  0.0594   181.7487
##
## sigma^2 estimated as 574014:  log likelihood=-1238.86
## AIC=2483.72   AICc=2483.88   BIC=2492.83
```

FORECASTING

```
# Forecasting first 22 weeks of 2017
forecast_MET_t <- forecast(MET_t_model, h = 22)

# Plot the forecast
```

```
plot(forecast_MET_t)
```




```
# Convert to numeric for ease
for_MET_t <- as.numeric(forecast_MET_t$mean)
v_MET_t <- as.numeric(MET_t_valid)

# Calculate the MAE
MAE <- mean(abs(for_MET_t - v_MET_t))

# Calculate the MAPE
MAPE <- 100*mean(abs((for_MET_t - v_MET_t)/v_MET_t))


print(MAE)
```

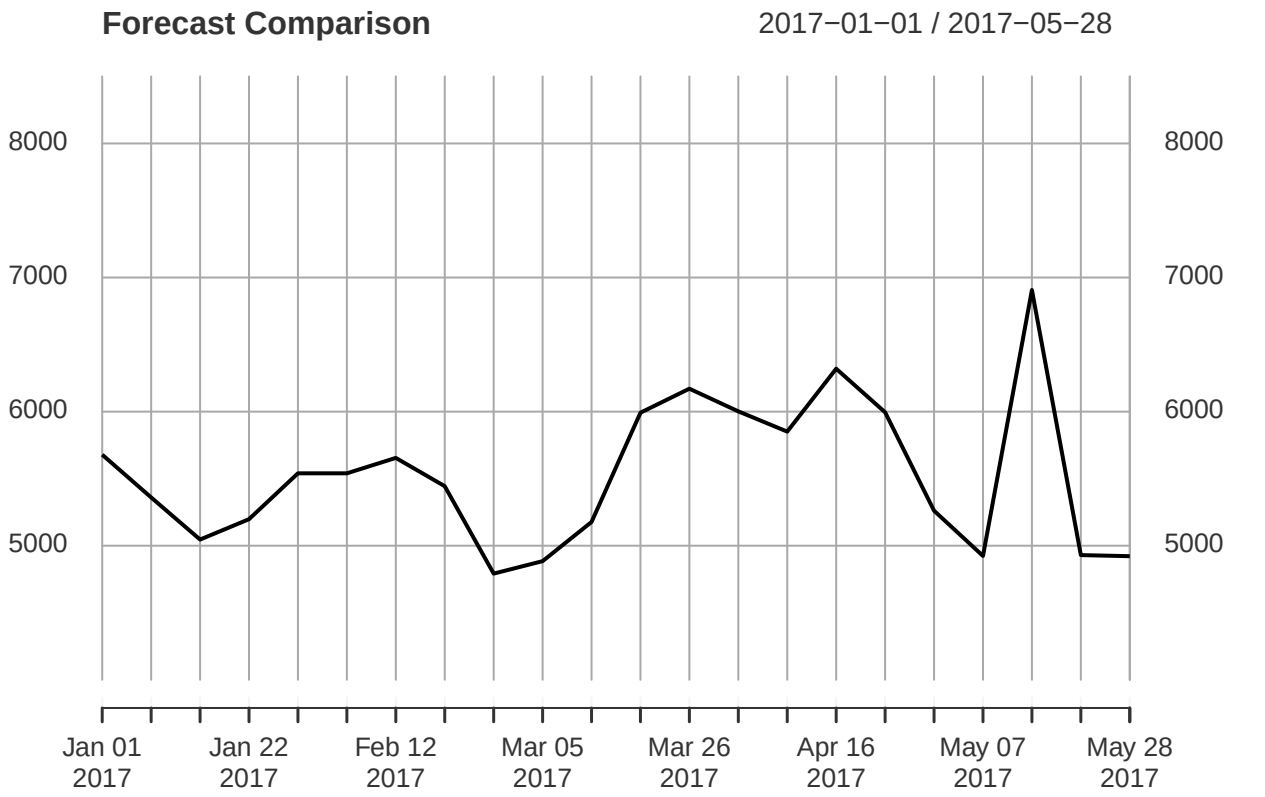
```
## [1] 898.8403
```

```
print(MAPE)
```

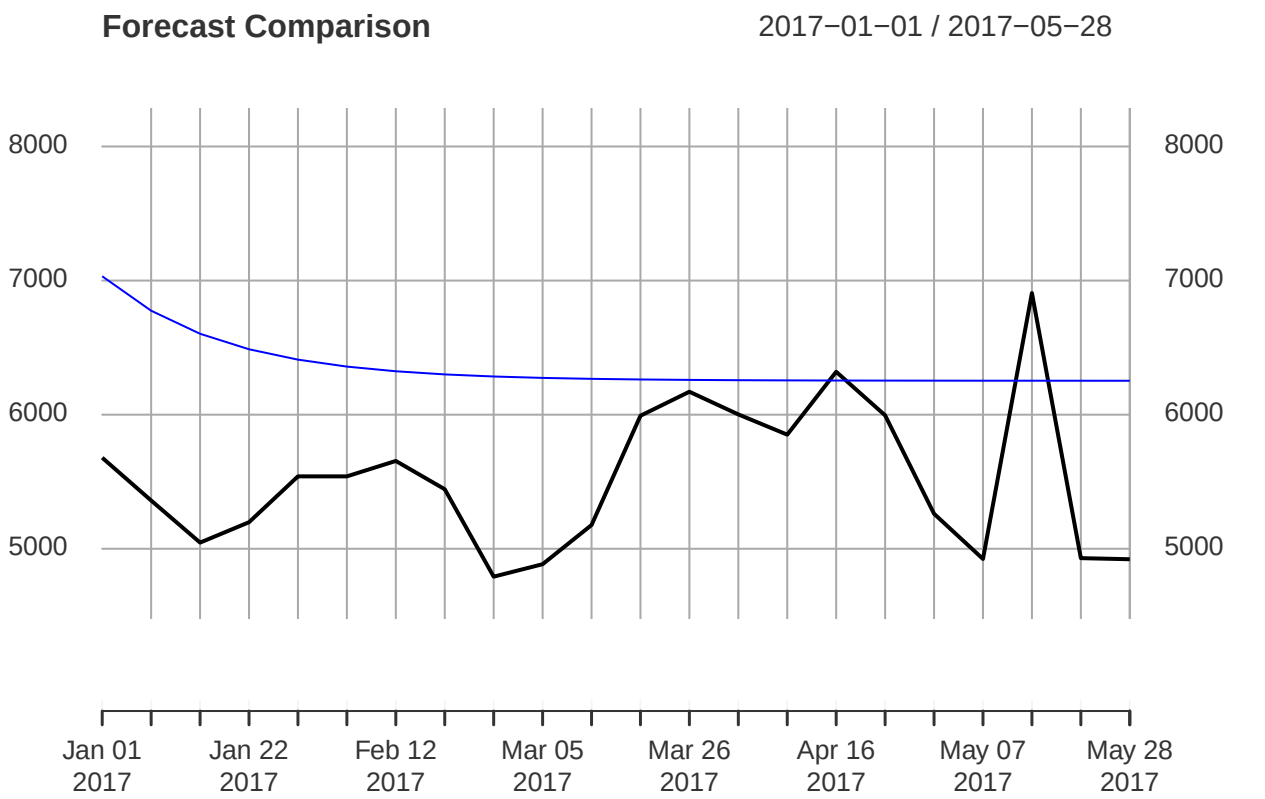
```
## [1] 17.10332
```

FORECAST COMPARISON

```
# Convert forecast to an xts object
for_dates <- seq(as.Date("2017-01-01"), length = 22, by = "weeks")
for_MET_t_xts <- xts(forecast_MET_t$mean, order.by = for_dates)

# Plot the validation dataset
plot(MET_t_valid, main = 'Forecast Comparison', ylim = c(4000, 8500))
```

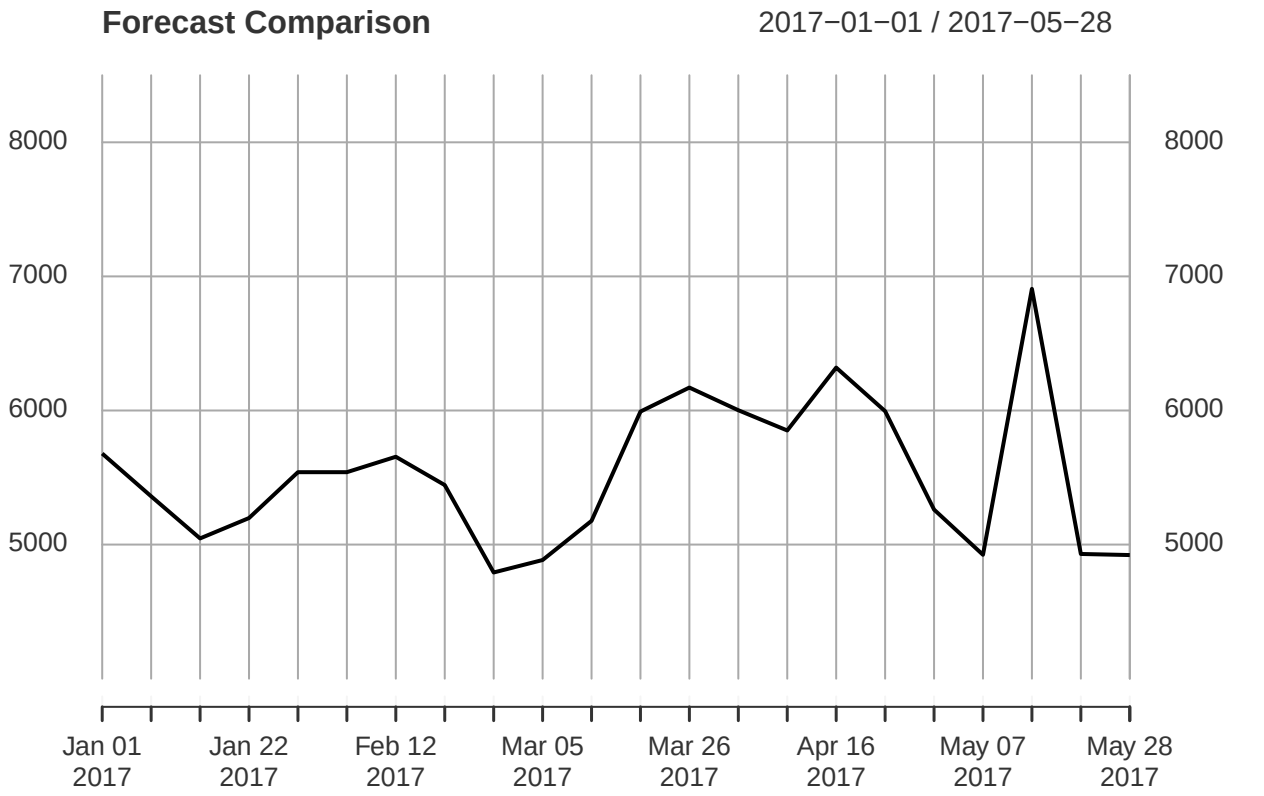



```
# Overlay the forecast of 2017
lines(for_MET_t_xts, col = "blue")
```

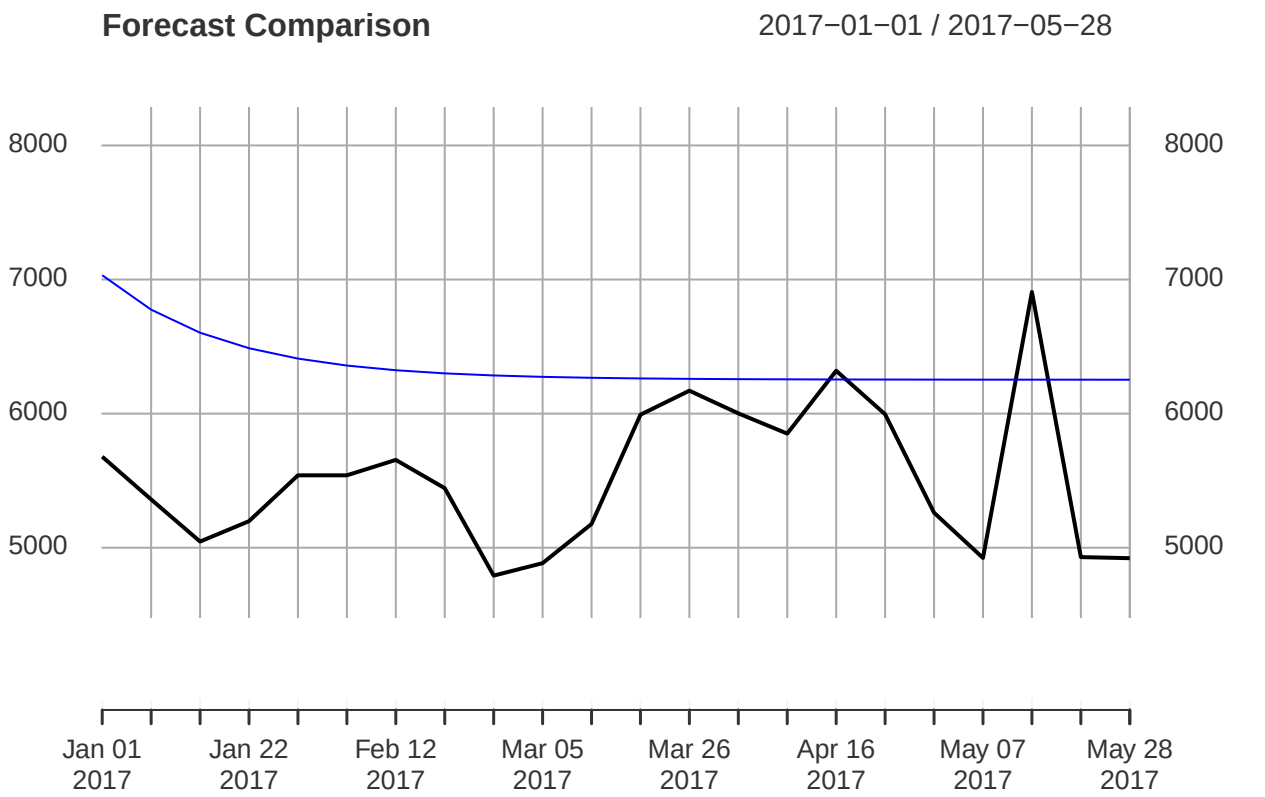


CONFIDENCE INTERVAL FOR FORECAST

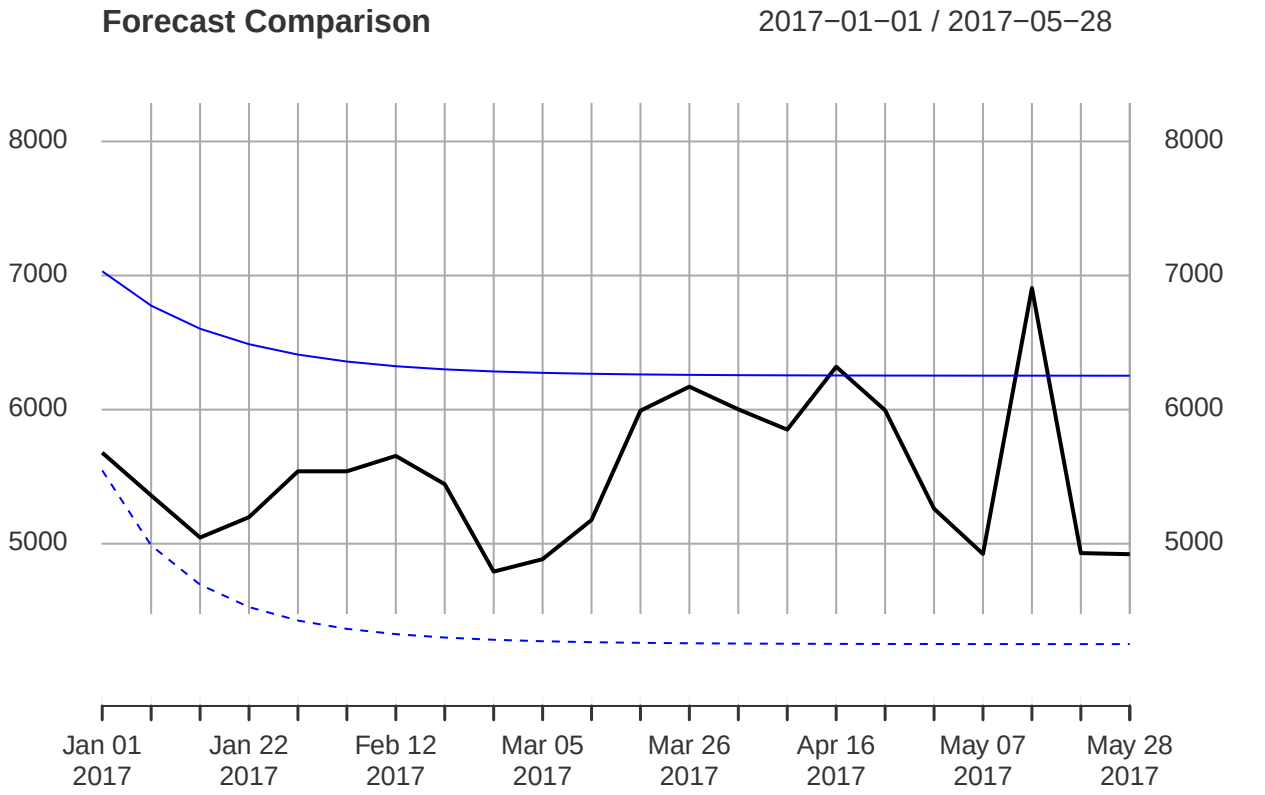
```
# Plot the validation data set
plot(MET_t_valid, main = 'Forecast Comparison', ylim = c(4000, 8500))
```



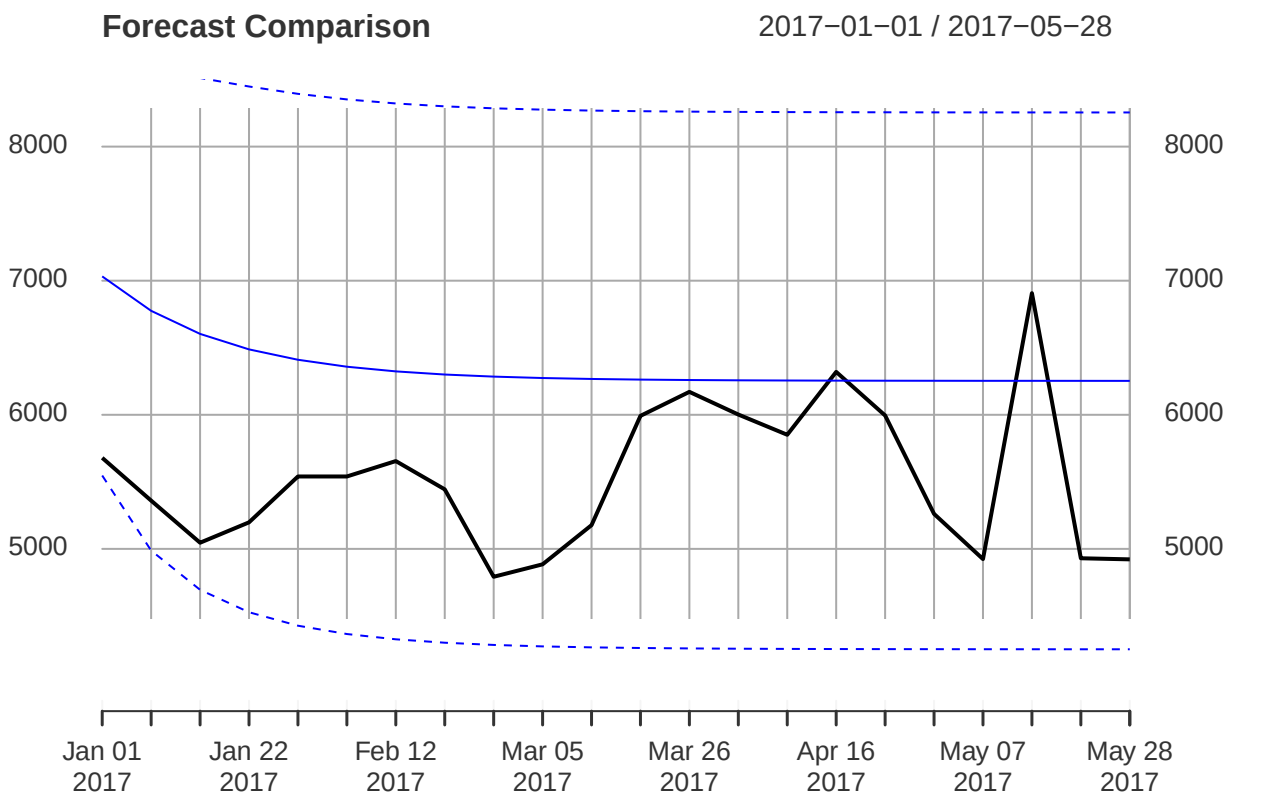

```
# Overlay the forecast of 2017
lines(for_MET_t_xts, col = "blue")
```




```
# Convert the limits to xts objects
lower <- xts(forecast_MET_t$lower[,2], order.by = for_dates)
upper <- xts(forecast_MET_t$upper[,2], order.by = for_dates)

# Adding confidence intervals of forecast to plot
lines(lower, col = "blue", lty = "dashed")
```




```
lines(upper, col = "blue", lty = "dashed")
```



PRICE ELASTICITY Price Elasticity is the economic measure of how much demand "reacts" to change in price

CALCULATING PRICE ELASTICITY

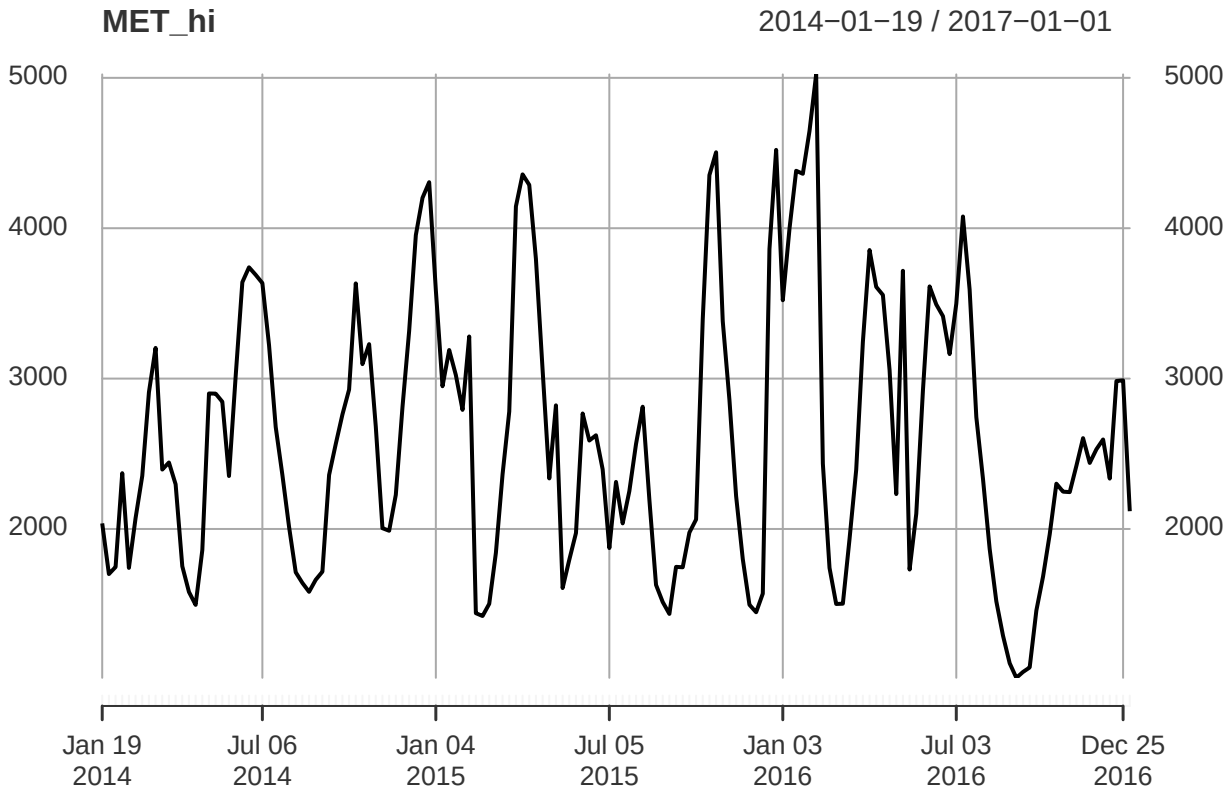
```
bev_xts_train <- bev_xts[index(MET_t) <= "2017-01-01"]
MET_hi<-MET_hi[index(MET_t) <= "2017-01-01"]
# Save the prices of each product
l_MET_hi_p <- as.vector(log(bev_xts_train[,"MET.hi.p"]))

# data frame
MET_hi_train <- data.frame(as.vector(log(MET_hi)), l_MET_hi_p)
colnames(MET_hi_train) <- c("log_sales", "log_price")

# Calculating regression
model_MET_hi <- lm(log_sales ~ log_price, data = MET_hi_train)

model_MET_hi
```

```
##
## Call:
## lm(formula = log_sales ~ log_price, data = MET_hi_train)
##
## Coefficients:
## (Intercept)    log_price
##      13.951       -1.511
```

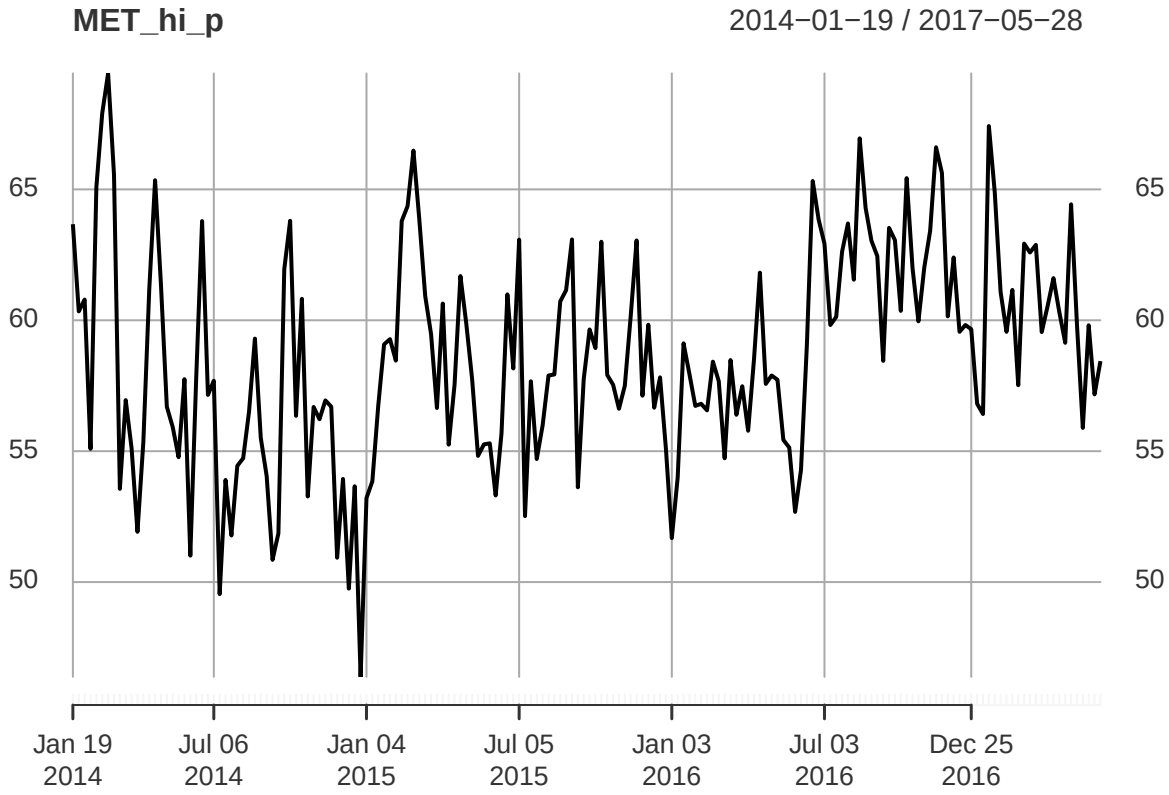
High end product is elastic in the metropoliton region

Visualize holiday / promotion effects

```
MET_hi_p<- bev_xts[,"MET.hi.p"]
# Plotting product's sales
plot(MET_hi)
```




```
# Plotting product's price
plot(MET_hi_p)
```



holiday / promotional effect variables

Products would be more desired around the weeks of Christmas, New Year's, and Valentine's Day.

```
# Date indices for New Year's week
n.dates <- as.Date(c("2014-12-28", "2015-12-27", "2016-12-25"))

# xts objects for New Year's week
newyear <- as.xts(rep(1, 3), order.by = n.dates)

# date indices for valentines's week
n1.dates <- as.Date(c("2014-02-09", "2015-02-08", "2016-02-07"))

# xts objects for valentine's week
valentine <- as.xts(rep(1, 3), order.by = n1.dates)

# date indices for christmas's week
n2.dates <- as.Date(c("2014-12-21", "2015-12-20", "2016-12-18"))

# xts objects for christmas's week
christmas <- as.xts(rep(1, 3), order.by = n2.dates)

# date indices for New mother's week
n3.dates <- as.Date(c("2014-05-04", "2015-05-03", "2016-05-01"))

# xts objects for mother's week
mother <- as.xts(rep(1, 3), order.by = n3.dates)

# sequence of dates for merging
dates_train <- seq(as.Date("2014-01-19"), length = 155, by = "weeks")

# Merge training dates into New Year's object
newyear <- merge(newyear, dates_train, fill = 0)
valentine <- merge(valentine, dates_train, fill = 0)
christmas<- merge(christmas, dates_train, fill = 0)
mother<- merge(mother, dates_train, fill = 0)
# MET_hi_train_2 by adding new year
MET_hi_train_2 <- data.frame(MET_hi_train, as.vector(newyear))
MET_hi_train_2 <- data.frame(MET_hi_train_2, as.vector(valentine))
MET_hi_train_2 <- data.frame(MET_hi_train_2, as.vector(christmas))
MET_hi_train_2 <- data.frame(MET_hi_train_2, as.vector(mother))

colnames(MET_hi_train_2)[3:6] <- c("newyear","valentine","christmas","mother")
head(MET_hi_train_2)
```

```
##   log_sales log_price newyear valentine christmas mother
## 1  7.619233  4.153713       0         0         0      0
## 2  7.438384  4.099995       0         0         0      0
## 3  7.465655  4.107425       0         0         0      0
## 4  7.771067  4.008968       0         1         0      0
## 5  7.462215  4.175771       0         0         0      0
## 6  7.636270  4.218183       0         0         0      0
```

```
# Building regressions for the products
model_MET_hi_full <- lm(log_sales ~log_price + newyear+valentine+christmas+mother, data = MET_hi_train_1
```

```
summary(model_MET_hi_full)
```

```
##
## Call:
## lm(formula = log_sales ~ log_price + newyear + valentine + christmas +
##     mother, data = MET_hi_train_2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.82983 -0.22921  0.00521  0.23269  0.64287
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.15229    1.55842   8.440 2.59e-14 ***
## log_price   -1.31923    0.38305  -3.444 0.000744 ***
## newyear      0.35796    0.19663   1.820 0.070698 .
## valentine    0.30558    0.19392   1.576 0.117193
## christmas    0.38349    0.19383   1.979 0.049715 *
## mother      -0.09268    0.19378  -0.478 0.633161
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.3319 on 149 degrees of freedom
## Multiple R-squared:  0.1471, Adjusted R-squared:  0.1185
## F-statistic: 5.139 on 5 and 149 DF,  p-value: 0.0002231
```

For 0.01 as significant,The Mother's Day promotion is significantly helping increase sales.

Forecasting with regression

Future Predictor Variables

```
bev_xts_valid<-bev_xts[index(MET_t) >="2017-01-01"]

# Subset the validation prices
l_MET_hi_p_valid <- as.vector(log(bev_xts_valid[,"MET.hi.p"]))

# Create a validation data frame
MET_hi_valid <- data.frame(l_MET_hi_p_valid)
colnames(MET_hi_valid) <- "log_price"
```

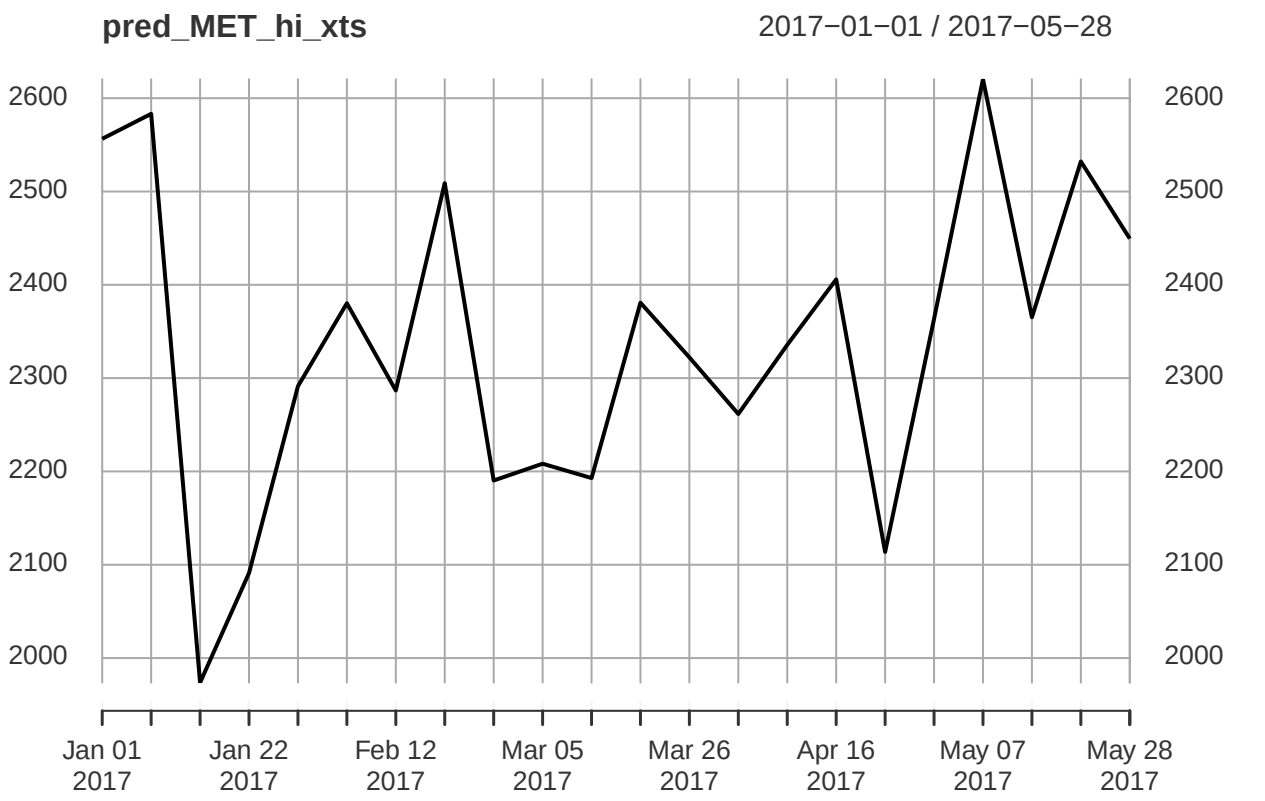
Forecast future values of demand

```
# Predict the log of sales for your high end product
pred_MET_hi <- predict(model_MET_hi, newdata = MET_hi_valid)

# Convert predictions out of log scale
pred_MET_hi <- exp(pred_MET_hi)
```

Visualizing forecasts of regression

```
# Convert to an xts object
dates_valid <- seq(as.Date("2017-01-01"), length = 22, by = "weeks")
pred_MET_hi_xts <- xts(pred_MET_hi, order.by = dates_valid)
```

```
# Plot the forecast
plot(pred_MET_hi_xts)
```



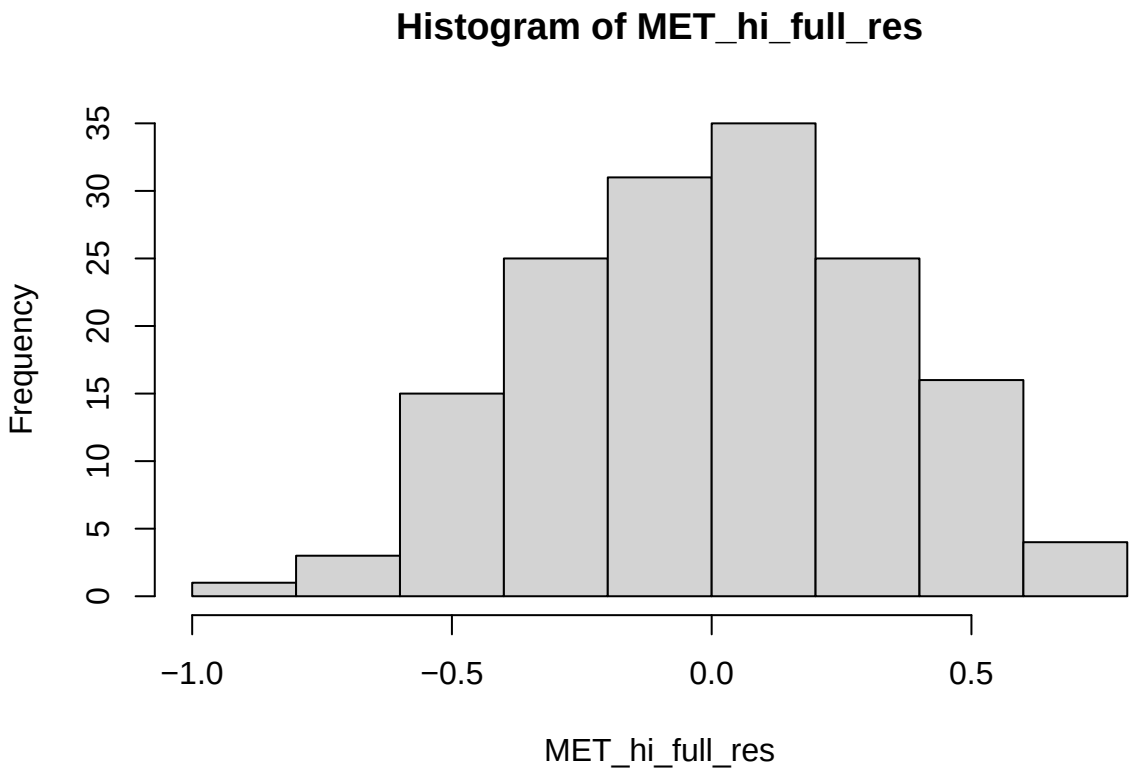

```
# Calculate and print the MAPE
MET_hi_v <- bev_xts_valid[,"MET.hi"]

MAPE <- 100*mean(abs((pred_MET_hi_xts - MET_hi_v)/MET_hi_v))
print(MAPE)
```

```
## [1] 29.49333
```

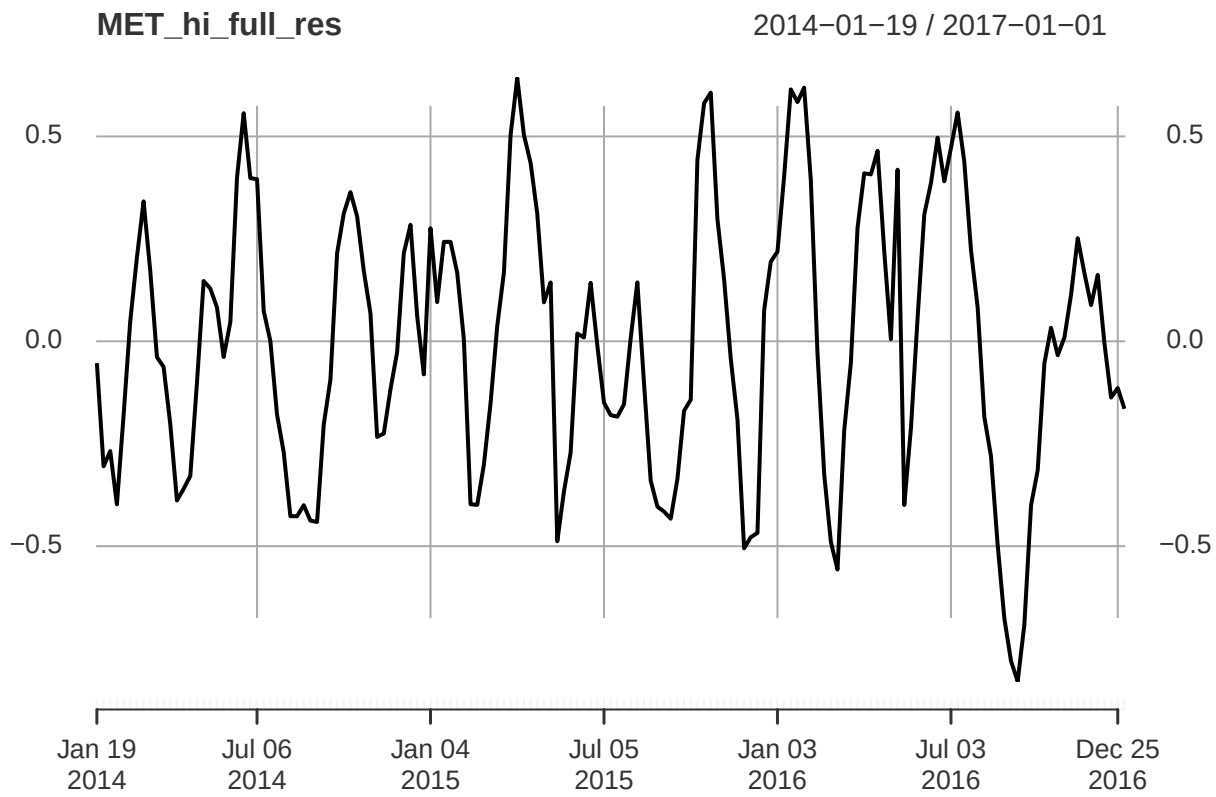
Regression residuals

Ways to reduce residuals further: 1 - Add more important variables to the model 2 - Use time series if your residuals are related over time

```
# Calculating the residuals from the model
MET_hi_full_res <- residuals(model_MET_hi_full)

# Converting the residuals to an xts object
MET_hi_full_res <- xts(MET_hi_full_res, order.by = dates_train)

# Plotting the histogram of the residuals
hist(MET_hi_full_res)
```




```
# Plotting the residuals over time
plot(MET_hi_full_res)
```



Forecasting Residuals

Demand = Inputs + Errors

ARIMA model on the residuals

```
# Building an ARIMA model on the residuals: MET_hi_arima
MET_hi_arima <- auto.arima(MET_hi_full_res)

# summary of the model
summary(MET_hi_arima)
```

```
## Series: MET_hi_full_res
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.5958  -0.7979  -0.6855  0.1847
## s.e.  0.0824   0.0660   0.1149  0.0944
##
## sigma^2 estimated as 0.02809:  log likelihood=58.07
## AIC=-106.15   AICc=-105.75   BIC=-90.93
##
## Training set error measures:
##                        ME      RMSE       MAE      MPE     MAPE      MASE
## Training set 0.0004663883 0.1654188 0.1223865 140.8364 240.4996 0.4557276
##                       ACF1
## Training set -0.0008817228
```

Forecasting residuals with time series

```
# Forecasting 22 weeks with your model: for_MET_hi_arima
for_MET_hi_arima <- forecast(MET_hi_arima, h = 22)

# Printing first 10 observations
head(for_MET_hi_arima, n = 10)
```

```
## $method
## [1] "ARIMA(2,0,2) with zero mean"
##
## $model
## Series: MET_hi_full_res
## ARIMA(2,0,2) with zero mean
##
## Coefficients:
##          ar1      ar2      ma1     ma2
##       1.5958  -0.7979  -0.6855  0.1847
## s.e.  0.0824   0.0660   0.1149  0.0944
##
## sigma^2 estimated as 0.02809:  log likelihood=58.07
## AIC=-106.15   AICc=-105.75   BIC=-90.93
##
## $level
## [1] 80 95
##
## $mean
## Time Series:
## Start = 1086
```

```
## End = 1233
## Frequency = 0.142857142857143
##  [1] -0.136268773 -0.092743464 -0.039268171  0.011338153  0.049426761
##  [6]  0.069828704  0.071994327  0.059170946  0.036979248  0.011797650
## [11] -0.010680006 -0.026456954 -0.033698398 -0.032665483 -0.025238988
## [16] -0.014211902 -0.002540578  0.007285809  0.013653972  0.015975593
## [21]  0.014599118  0.010550034
##
## $lower
## Time Series:
## Start = 1086
## End = 1233
## Frequency = 0.142857142857143
##             80%        95%
## 1086 -0.3510510 -0.4647498
## 1093 -0.3831881 -0.5369402
## 1100 -0.3811275 -0.5620969
## 1107 -0.3550208 -0.5489595
## 1114 -0.3228893 -0.5199816
## 1121 -0.3024881 -0.4995808
## 1128 -0.3038990 -0.5028849
## 1135 -0.3258095 -0.5296058
## 1142 -0.3587063 -0.5681695
## 1149 -0.3918024 -0.6054553
## 1156 -0.4178738 -0.6334291
## 1163 -0.4342817 -0.6501710
## 1170 -0.4415812 -0.6575013
## 1177 -0.4415836 -0.6580517
## 1184 -0.4361986 -0.6537475
## 1191 -0.4273209 -0.6460076
## 1198 -0.4170889 -0.6365376
## 1205 -0.4078196 -0.6275632
## 1212 -0.4015066 -0.6212793
## 1219 -0.3992336 -0.6190321
## 1226 -0.4009009 -0.6208533
## 1233 -0.4054182 -0.6256184
##
## $upper
## Time Series:
## Start = 1086
## End = 1233
## Frequency = 0.142857142857143
##             80%       95%
## 1086 0.07851345 0.1922123
## 1093 0.19770120 0.3514533
## 1100 0.30259116 0.4835606
## 1107 0.37769709 0.5716358
## 1114 0.42174285 0.6188351
## 1121 0.44214555 0.6392382
## 1128 0.44788764 0.6468735
## 1135 0.44415137 0.6479477
## 1142 0.43266477 0.6421280
## 1149 0.41539766 0.6290506
## 1156 0.39651374 0.6120691
```

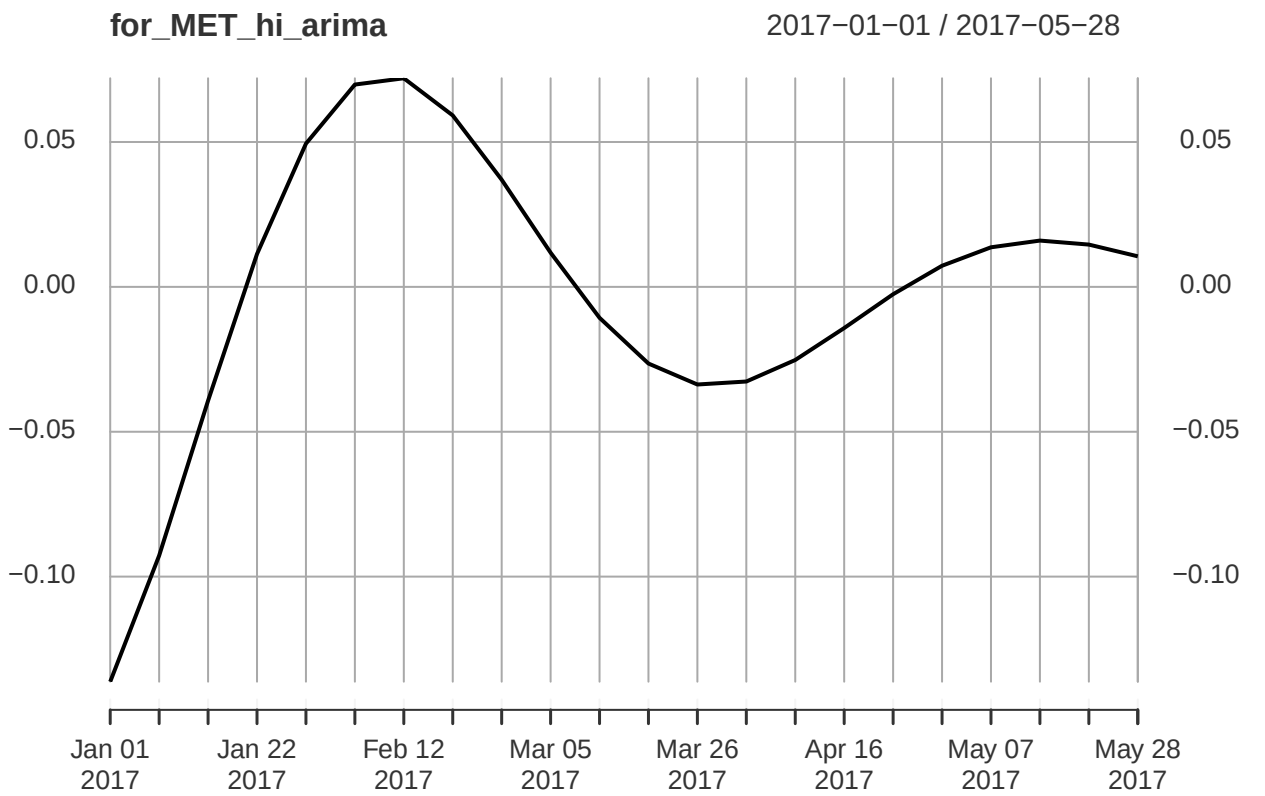
```
## 1163 0.38136776 0.5972571
## 1170 0.37418441 0.5901045
## 1177 0.37625259 0.5927207
## 1184 0.38572066 0.6032696
## 1191 0.39889708 0.6175838
## 1198 0.41200776 0.6314564
## 1205 0.42239123 0.6421348
## 1212 0.42881454 0.6485873
## 1219 0.43118481 0.6509833
## 1226 0.43009912 0.6500515
## 1233 0.42651824 0.6467185
##
## $x
## Time Series:
## Start = 1
## End = 1079
## Frequency = 0.142857142857143
##   [1] -0.0533442088 -0.3050608930 -0.2679871364 -0.3980448555 -0.1812638336
##   [6]  0.0487425483  0.2051215050  0.3415775235  0.1714681549 -0.0385858904
##  [11] -0.0624785947 -0.2023539900 -0.3885669752 -0.3600401298 -0.3283381564
##  [16] -0.1003193106  0.1472824543  0.1286635485  0.0824543503 -0.0383002670
##  [21]  0.0469779007  0.3993772091  0.5567473039  0.3982821656  0.3956714718
##  [26]  0.0741412362  0.0004864105 -0.1788583480 -0.2708136227 -0.4267616935
##  [31] -0.4268953659 -0.3999353308 -0.4377335990 -0.4409216052 -0.2031917150
##  [36] -0.0935248518  0.2146422752  0.3118464924  0.3641170628  0.3048746711
##  [41]  0.1720782482  0.0681841192 -0.2333446247 -0.2250716806 -0.1171176626
##  [46] -0.0261711766  0.2151540586  0.2844470758  0.0616015672 -0.0807641450
##  [51]  0.2760139638  0.0956579486  0.2427131211  0.2427211338  0.1676879751
##  [56]  0.0048155086 -0.3976951940 -0.3992417320 -0.3010502944 -0.1509957681
##  [61]  0.0370795404  0.1679619085  0.5029658814  0.6428668289  0.5038715438
##  [66]  0.4355756364  0.3109350095  0.0951141209  0.1436510983 -0.4878872747
##  [71] -0.3667051047 -0.2712003185  0.0190210221  0.0094009769  0.1427646989
##  [76] -0.0104327379 -0.1495626102 -0.1802545958 -0.1839169330 -0.1537727380
##  [81]  0.0075616554  0.1438794981 -0.1028232792 -0.3400548766 -0.4033425330
##  [86] -0.4157300649 -0.4329825702 -0.3365833229 -0.1697871337 -0.1424971977
##  [91]  0.4432309818  0.5809489241  0.6068156467  0.2988315363  0.1508324283
##  [96] -0.0427930796 -0.1899449266 -0.5052421045 -0.4780856884 -0.4676284344
## [101]  0.0757290321  0.1947596827  0.2186748480  0.4031612037  0.6148695398
## [106]  0.5839252433  0.6189314047  0.3932293468 -0.0310876831 -0.3238893750
## [111] -0.4892635176 -0.5569608604 -0.2174044237 -0.0520399604  0.2770585422
## [116]  0.4100168813  0.4073895382  0.4649143890  0.2197494353  0.0052051896
## [121]  0.4188936386 -0.3995365101 -0.2121645996  0.0571389176  0.3092226847
## [126]  0.3861154326  0.4971073758  0.3904229043  0.4717369130  0.5584993450
## [131]  0.4384207068  0.2226571630  0.0829540952 -0.1846279742 -0.2805017447
## [136] -0.4971907976 -0.6762871901 -0.7803955184 -0.8298344251 -0.6916916287
## [141] -0.4004567388 -0.3146740355 -0.0538671680  0.0330318515 -0.0339825342
## [146]  0.0092446785  0.1137550423  0.2517780814  0.1667640868  0.0883695791
## [151]  0.1621840145 -0.0048418026 -0.1373305993 -0.1139955377 -0.1647439283
##
## $series
## [1] "MET_hi_full_res"
##
## $fitted
## Time Series:
```

```
## Start = 1
## End = 1079
## Frequency = 0.142857142857143
##   [1] -0.025881221 -0.074526052 -0.308481955 -0.251067030 -0.313291467
##   [6] -0.088978617  0.152358715  0.277715270  0.347389016  0.133468968
##  [11] -0.112948913 -0.135297921 -0.217773847 -0.353922551 -0.291865028
##  [16] -0.212806442  0.018051059  0.247273767  0.192983892  0.082774495
##  [21] -0.064333683  0.006856630  0.351332116  0.501487438  0.300033895
##  [26]  0.228989484 -0.073584728 -0.137765862 -0.243959206 -0.278633280
##  [31] -0.368358299 -0.327955581 -0.259059910 -0.270234642 -0.270347027
##  [36] -0.049999034  0.055128061  0.299765296  0.347564955  0.323115652
##  [41]  0.211544739  0.055020764 -0.044811830 -0.295107147 -0.255818190
##  [46] -0.089448733  0.033933086  0.251688869  0.293268063  0.036197022
##  [51] -0.140657302  0.197670958  0.079317758  0.180141277  0.180955120
##  [56]  0.094579522 -0.067035338 -0.428400888 -0.400855483 -0.224883240
##  [61] -0.032956699  0.145295078  0.235848479  0.489692454  0.568906451
##  [66]  0.364000313  0.231962499  0.107721062 -0.073088687  0.002439481
##  [71] -0.517035640 -0.389526768 -0.193521946  0.122911631  0.116901873
##  [76]  0.181625293  0.005870612 -0.159278249 -0.182646565 -0.152670435
##  [81] -0.098118517  0.062118332  0.187045719 -0.065079901 -0.325668995
##  [86] -0.369870613 -0.324499724 -0.293340925 -0.182031498 -0.018760540
##  [91] -0.004835800  0.491002634  0.594532875  0.513003149  0.141768353
##  [96] -0.043525298 -0.187470712 -0.267138902 -0.491943172 -0.413272626
## [101] -0.324946468  0.209275207  0.334343178  0.270169373  0.356346513
## [106]  0.506869727  0.436148153  0.410701631  0.179400327 -0.222315473
## [111] -0.461315318 -0.521936780 -0.479560107 -0.088701383  0.113725455
## [116]  0.378463487  0.441779648  0.352355909  0.333334276  0.078366886
## [121] -0.137868249  0.269140344 -0.410593349 -0.279326653  0.066481982
## [126]  0.343624540  0.385144507  0.416294458  0.264803369  0.294638600
## [131]  0.372195718  0.257340405  0.041499863 -0.080110157 -0.281516688
## [136] -0.320311633 -0.448161964 -0.558797289 -0.595969573 -0.582178970
## [141] -0.409791190 -0.113762038 -0.043172556  0.135341212  0.163853099
## [146]  0.036131121  0.023751801  0.107489913  0.228737663  0.134361615
## [151]  0.028034353  0.087845021 -0.048817887 -0.171736570 -0.128268764
##
## $residuals
## Time Series:
## Start = 1
## End = 1079
## Frequency = 0.142857142857143
##   [1] -0.0274629878 -0.2305348410  0.0404948191 -0.1469778255  0.1320276336
##   [6]  0.1377211651  0.0527627899  0.0638622540 -0.1759208614 -0.1720548582
##  [11]  0.0504703184 -0.0670560688 -0.1707931278 -0.0061175790 -0.0364731285
##  [16]  0.1124871317  0.1292313951 -0.1186102186 -0.1105295421 -0.1210747617
##  [21]  0.1113115840  0.3925205790  0.2054151874 -0.1032052724  0.0956375771
##  [26] -0.1548482480  0.0740711384 -0.0410924864 -0.0268544163 -0.1481284139
##  [31] -0.0585370674 -0.0719797500 -0.1786736890 -0.1706869628  0.0671553117
##  [36] -0.0435258177  0.1595142139  0.0120811968  0.0165521076 -0.0182409810
##  [41] -0.0394664907  0.0131633552 -0.1885327943  0.0700354659  0.1387005277
##  [46]  0.0632775567  0.1812209725  0.0327582071 -0.2316664955 -0.1169611668
##  [51]  0.4166712654 -0.1020130095  0.1633953633  0.0625798569 -0.0132671446
##  [56] -0.0897640133 -0.3306598555  0.0291591562  0.0998051886  0.0738874720
##  [61]  0.0700362396  0.0226668302  0.2671174020  0.1531743751 -0.0650349068
##  [66]  0.0715753231  0.0789725103 -0.0126069414  0.2167397857 -0.4903267553
```

```
##  [71]  0.1503305351  0.1183264492  0.2125429679 -0.1135106545  0.0258628263
##  [76] -0.1920580308 -0.1554332225 -0.0209763465 -0.0012703677 -0.0011023026
##  [81]  0.1056801729  0.0817611665 -0.2898689982 -0.2749749761 -0.0776735379
##  [86] -0.0458594519 -0.1084828458 -0.0432423982  0.0122443643 -0.1237366578
##  [91]  0.4480667815  0.0899462904  0.0122827718 -0.2141716132  0.0090640755
##  [96]  0.0007322185 -0.0024742146 -0.2381032020  0.0138574833 -0.0543558087
## [101]  0.4006755005 -0.0145155244 -0.1156683304  0.1329918311  0.2585230264
## [106]  0.0770555159  0.1827832516 -0.0174722843 -0.2104880104 -0.1015739015
## [111] -0.0279481996 -0.0350240801  0.2621556832  0.0366614229  0.1633330867
## [116]  0.0315533947 -0.0343901099  0.1125584800 -0.1135848403 -0.0731616968
## [121]  0.5567618881 -0.6686768544  0.1984287496  0.3364655711  0.2427407031
## [126]  0.0424908927  0.1119628683 -0.0258715535  0.2069335444  0.2638607446
## [131]  0.0662249889 -0.0346832417  0.0414542319 -0.1045178170  0.0010149434
## [136] -0.1768791650 -0.2281252264 -0.2215982297 -0.2338648520 -0.1095126586
## [141]  0.0093344515 -0.2009119974 -0.0106946116 -0.1023093609 -0.1978356337
## [146] -0.0268864425  0.0900032410  0.1442881679 -0.0619735759 -0.0459920358
## [151]  0.1341496613 -0.0926868237 -0.0885127120  0.0577410321 -0.0364751642
```

Visualizing residual forecasts

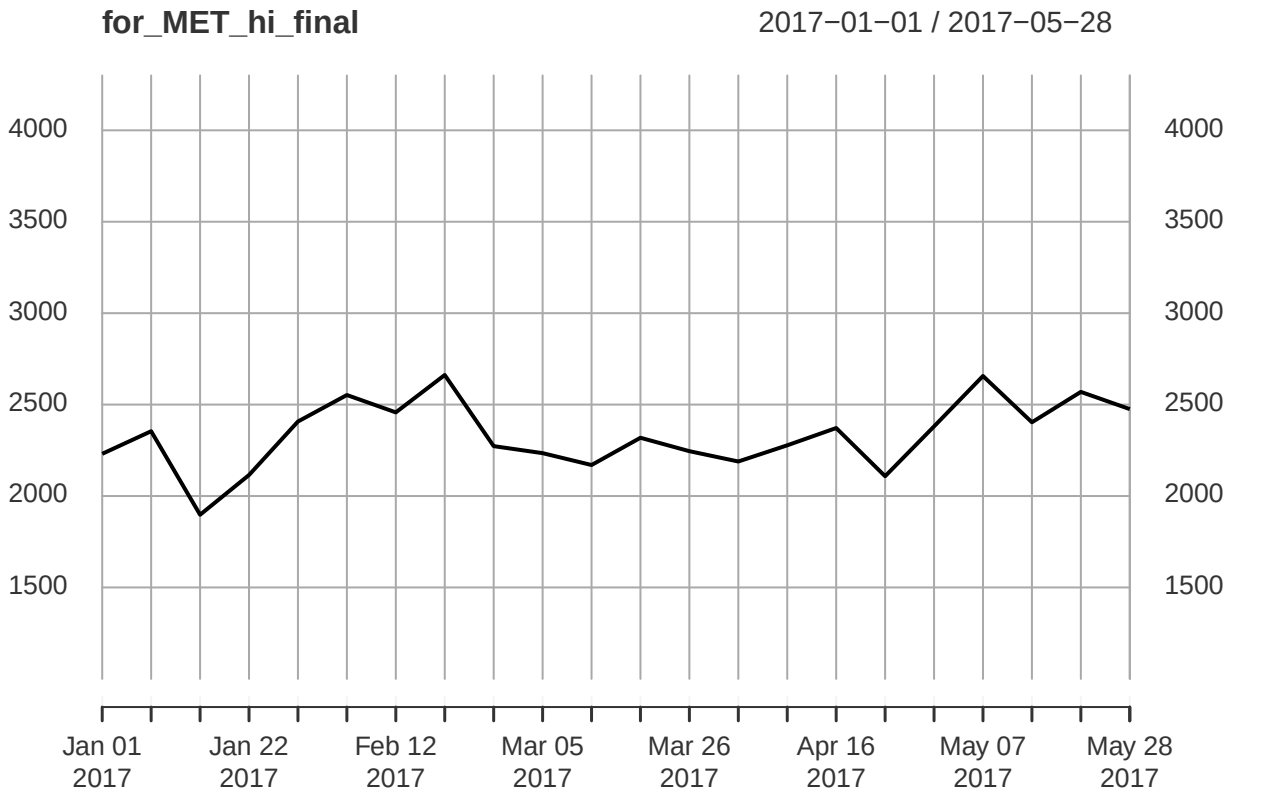
```
# Converting forecasts into an xts object
dates_valid <- seq(as.Date("2017-01-01"), length = 22, by = "weeks")
for_MET_hi_arima <- xts(for_MET_hi_arima$mean, order.by = dates_valid)

# Plotting the forecast
plot(for_MET_hi_arima)
```



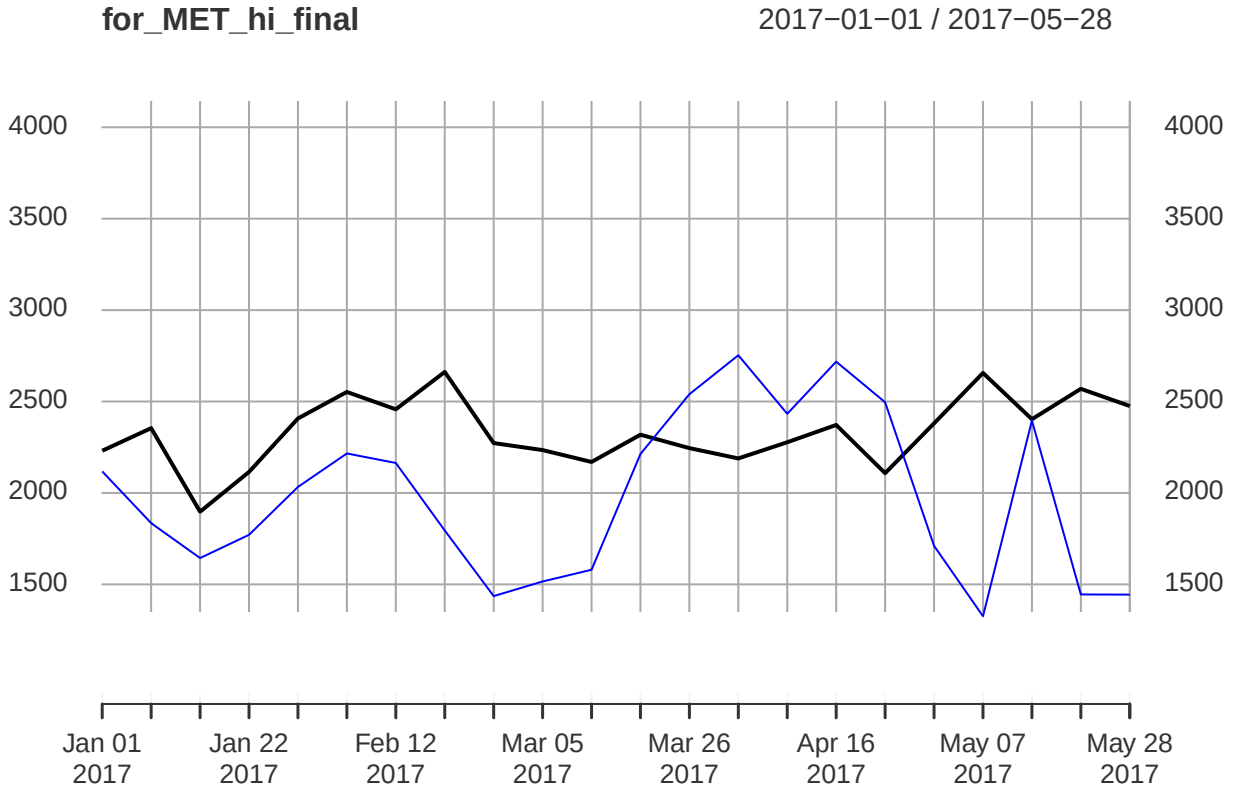
Transfer functions and Ensembling

Combining residuals from regression & time series

```
# Converting residual forecast to the exponential version
for_MET_hi_arima <- exp(for_MET_hi_arima)

# Multiplying forecasts together
for_MET_hi_final <- pred_MET_hi_xts * for_MET_hi_arima

# Plotting the final forecast
plot(for_MET_hi_final, ylim = c(1000, 4300))
```




```
# Overlaying the validation data set
lines(MET_hi_v, col = "blue")
```



Calculating transfer function MAPE and MAE

```
# Calculating the MAE
MAE <- mean(abs(for_MET_hi_final - MET_hi_v))
print(MAE)
```

```
## [1] 511.9467
```

```
# Calculating the MAPE
MAPE <- 100*mean(abs((for_MET_hi_final - MET_hi_v)/MET_hi_v))
print(MAPE)
```

```
## [1] 30.33902
```

Arima Forecasting

```
# Building an ARIMA model using the auto.arima function
MET_hi_model_arima <- auto.arima(MET_hi)

# Forecasting the ARIMA model you just built above
for_MET_hi <- forecast(MET_hi_model_arima, h = 22)

# forecast as an xts object
dates_valid <- seq(as.Date("2017-01-01"), length = 22, by = "weeks")
for_MET_hi_xts <- xts(for_MET_hi$mean, order.by = dates_valid)

# Calculating the MAPE of the forecast
MAPE <- 100*mean(abs((for_MET_hi_xts - MET_hi_v)/MET_hi_v))
print(MAPE)
```

```
## [1] 36.92598
```

Ensembling of Forecasts

```
# Ensembling the two forecasts together
for_MET_hi_en <- 0.5*(for_MET_hi_xts + pred_MET_hi_xts)

# Calculating the MAE and MAPE
MAE <- mean(abs(for_MET_hi_en - MET_hi_v))
print(MAE)
```

```
## [1] 536.2768
```

```
MAPE <- 100*mean(abs((for_MET_hi_en - MET_hi_v)/MET_hi_v))
print(MAPE)
```

```
## [1] 32.63149
```