Forecasting model for Consumer Goods Appliances by Optimal Reconciliation for Hierarcical and Grouped Times Series through Trace minimization

PGP BABI-K19 - Group 4

15 January 2020

```
library(readr)
library(tidyverse)
```

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

```
## -- Attaching packages -------------------------------------------- tidyverse 1.3.0 --
```

```
## v ggplot2 3.2.1     v dplyr   0.8.3
## v tibble  2.1.3     v stringr 1.4.0
## v tidyr   1.0.0     v forcats 0.4.0
## v purrr   0.3.3
```

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

```
## -- Conflicts ----------------------------------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(fabletools)
```

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

```
library(fpp3)
```

```
## Warning: package 'fpp3' was built under R version 3.6.2
```

```
## -- Attaching packages ---------------------------------------------------- fpp3 0.1 --
```

```
## v lubridate   1.7.4     v feasts      0.1.1
## v tsibble     0.8.5     v fable       0.1.1
## v tsibbledata 0.1.0
```

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

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

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

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

```
## -- Conflicts ------------------------------------------------------ fpp3_conflicts --
## x lubridate::date()       masks base::date()
## x dplyr::filter()         masks stats::filter()
## x tsibble::id()           masks dplyr::id()
## x tsibble::interval()     masks lubridate::interval()
## x dplyr::lag()            masks stats::lag()
## x tsibble::new_interval() masks lubridate::new_interval()
```

REading the data and converting it into a time series tibble format

```
## Parsed with column specification:
## cols(
##   product = col_character(),
##   date = col_character(),
##   city = col_character(),
##   sales = col_double()
## )
```

```
## [1] "tbl_ts"     "tbl_df"     "tbl"        "data.frame"
```

We can see from above output that data is converted into tbl_ts class.We need to check whether there are any gaps in the time series segments.

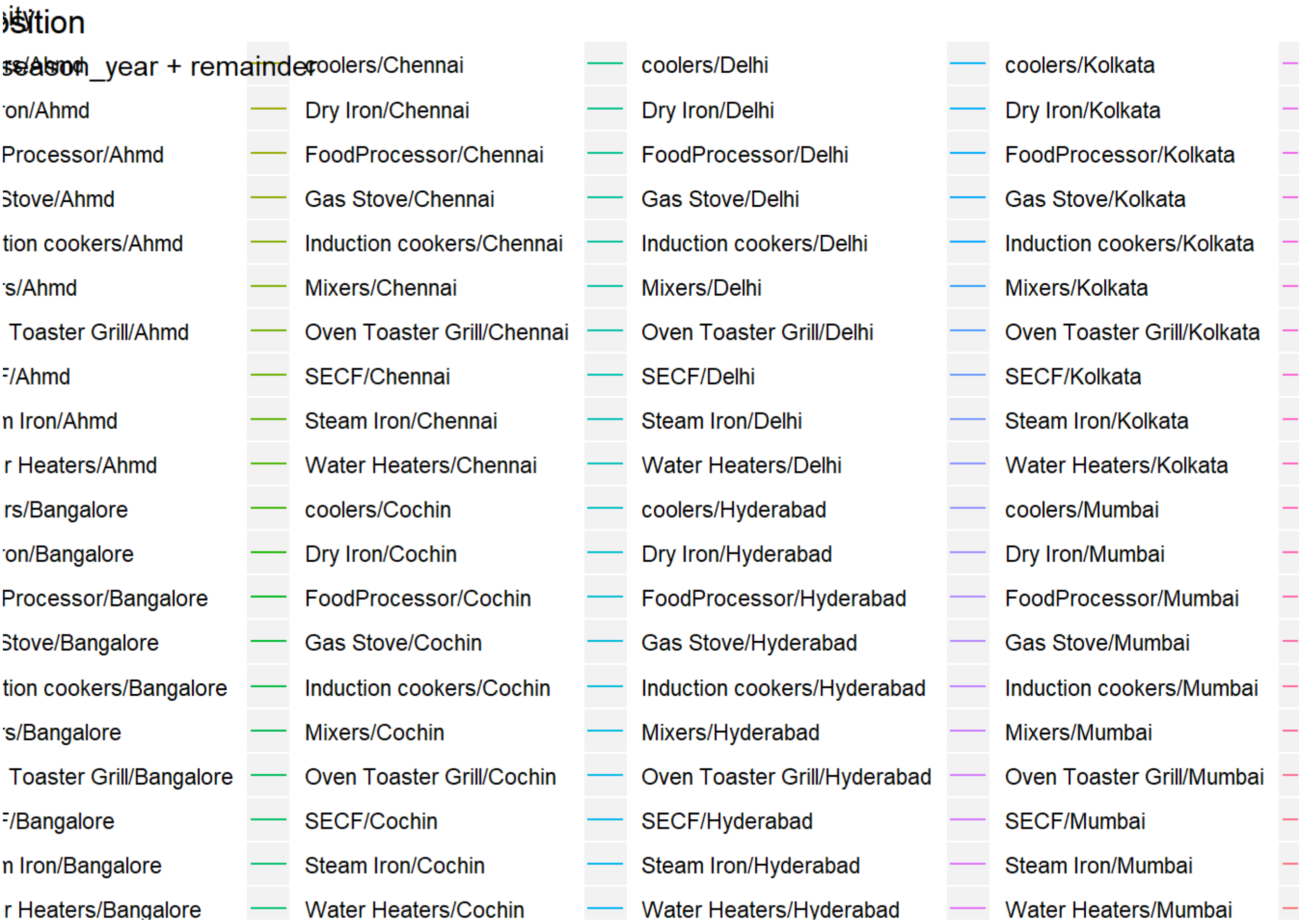
```
has_gaps(data_t10, .full = T)
```

product <chr>	city <chr>	.gaps <lgl>
coolers	Ahmd	FALSE
coolers	Bangalore	FALSE
coolers	Chennai	FALSE
coolers	Cochin	FALSE
coolers	Delhi	FALSE
coolers	Hyderabad	FALSE
coolers	Kolkata	FALSE
coolers	Mumbai	FALSE
coolers	Patna	FALSE
coolers	Pune	FALSE

1-10 of 100 rows Previous **1** 2 3 4 5 6 ... 10 Next

No gaps are detected in the series time segments. We will derive the time series features from the data set like strength of trend and seasonality. Also we will apply STL decomposition for checking the components of time series

```
#Time Series Components

data_t10 %>%
  STL(sales ~ trend(window=21) + season(window = 13), robust = TRUE) %>%
  autoplot()
```




```
#Time Series features

data_t10 %>%
  features(sales, feature_set(tags = "stl"))
```

product <chr>	city <chr>	trend_strength <dbl>	seasonal_strength_year <dbl>	spikiness <dbl>
coolers	Ahmd	0.3824289	0.9677495	2.702754e+19
coolers	Bangalore	0.3749252	0.9749263	4.580445e+19
coolers	Chennai	0.3244190	0.9728752	4.055376e+20
coolers	Cochin	0.3701564	0.9444576	8.620348e+19
coolers	Delhi	0.3519970	0.8484469	6.915335e+18
coolers	Hyderabad	0.2927994	0.8434662	4.541726e+20

product <chr>	city <chr>	trend_strength <dbl>	seasonal_strength_year <dbl>	spikiness <dbl>
coolers	Kolkata	0.4073242	0.4547291	3.118761e+22
coolers	Mumbai	0.3157606	0.9431247	3.577547e+20
coolers	Patna	0.4096202	0.4551642	4.288666e+20
coolers	Pune	0.1079552	0.9372398	1.736126e+19

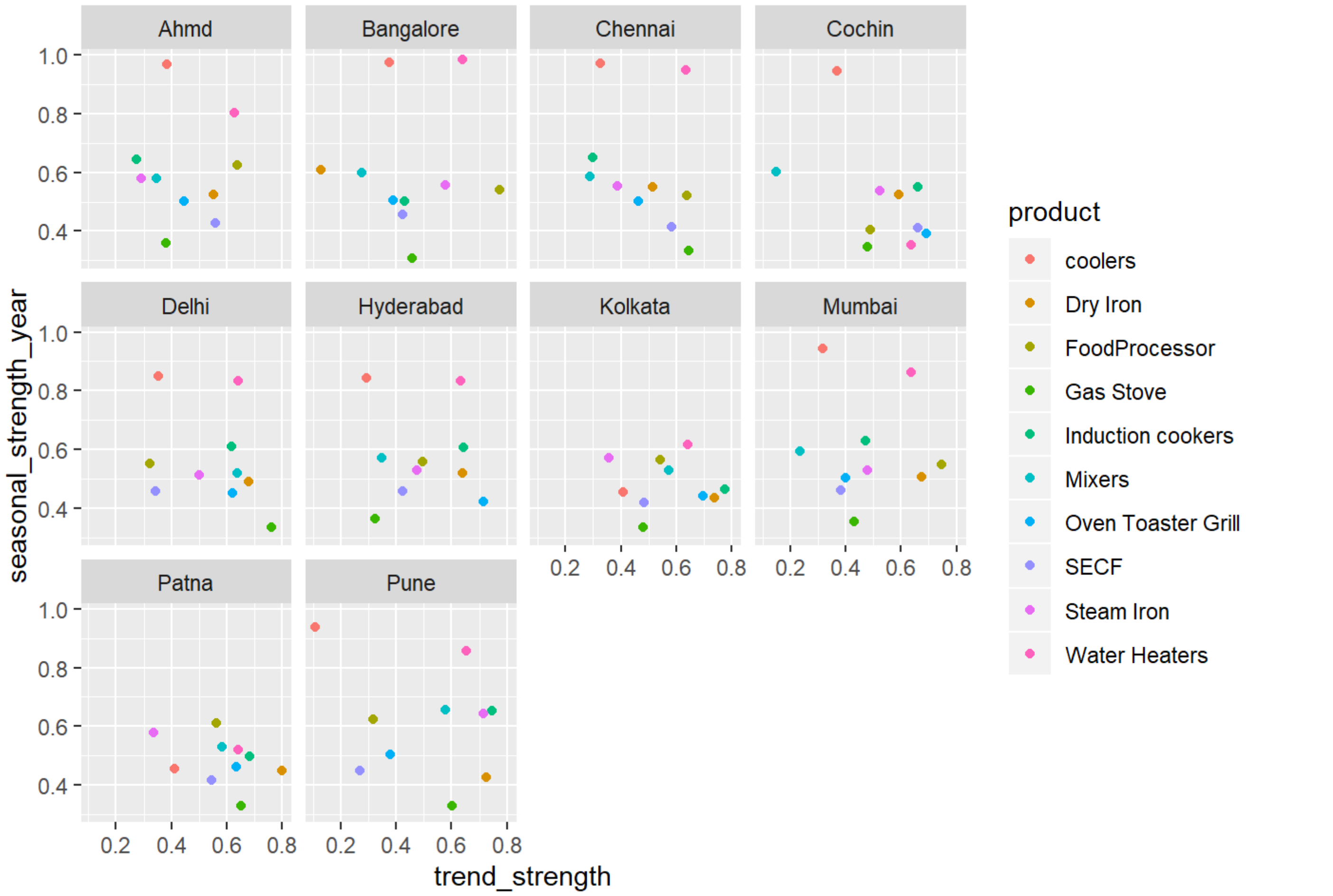
1-10 of 100 rows | 1-6 of 9 columns Previous 1 2 3 4 5 6 ... 10 Next

```
#Measuring STL features and Trend & Seasonality strength

data_t10 %>%
  features(sales, feat_stl)
```

product <chr>	city <chr>	trend_strength <dbl>	seasonal_strength_year <dbl>	spikiness <dbl>
coolers	Ahmd	0.3824289	0.9677495	2.702754e+19
coolers	Bangalore	0.3749252	0.9749263	4.580445e+19
coolers	Chennai	0.3244190	0.9728752	4.055376e+20
coolers	Cochin	0.3701564	0.9444576	8.620348e+19
coolers	Delhi	0.3519970	0.8484469	6.915335e+18
coolers	Hyderabad	0.2927994	0.8434662	4.541726e+20
coolers	Kolkata	0.4073242	0.4547291	3.118761e+22
coolers	Mumbai	0.3157606	0.9431247	3.577547e+20
coolers	Patna	0.4096202	0.4551642	4.288666e+20
coolers	Pune	0.1079552	0.9372398	1.736126e+19

1-10 of 100 rows | 1-6 of 9 columns Previous 1 2 3 4 5 6 ... 10 Next

```
#Plotting the features

data_t10 %>%
  features(sales, feat_stl) %>%
  ggplot(aes(x = trend_strength, y = seasonal_strength_year, col = product))+
  geom_point() + facet_wrap(vars(city))
```




```
#Measuring the average trend strength of the product

data_t10 %>%
  features(sales, feat_stl) %>%
  group_by(product) %>%
  summarise(avg_trend_str = mean(trend_strength))
```

product <chr>	avg_trend_str <dbl>
coolers	0.3337386
Dry Iron	0.6042122
FoodProcessor	0.5526996
Gas Stove	0.5213903
Induction cookers	0.5599806
Mixers	0.4022176
Oven Toaster Grill	0.5428758
SECF	0.4673534
Steam Iron	0.4646600
Water Heaters	0.6382907

1-10 of 10 rows

```
#Measuring the average seasonal strength of the product

data_t10 %>%
  features(sales, feat_stl) %>%
  group_by(product) %>%
  summarise(avg_season_str = mean(seasonal_strength_year))
```

product <chr>	avg_season_str <dbl>
coolers	0.8342180
Dry Iron	0.5033640
FoodProcessor	0.5541772
Gas Stove	0.3382473
Induction cookers	0.5811531
Mixers	0.5765536
Oven Toaster Grill	0.4683841
SECF	0.4367305
Steam Iron	0.5586238
Water Heaters	0.7610801

1-10 of 10 rows

```
#Measuring the average trend strength of the city
data_t10 %>%
  features(sales, feat_stl) %>%
  group_by(city) %>%
  summarise(avg_trend_str = mean(trend_strength))
```

city <chr>	avg_trend_str <dbl>
Ahmd	0.4496741
Bangalore	0.4470481
Chennai	0.4777565
Cochin	0.5253164
Delhi	0.5477875
Hyderabad	0.4990429
Kolkata	0.5695075
Mumbai	0.4771078
Patna	0.5848740
Pune	0.5093037

1-10 of 10 rows

Now we need to divide the data into test and train components and use aggts() function in order to aggregate the data into base time series. We will use 3 years of data as train data and 1 year of data for validation.

```
#Training and Testing for t5 data

t10_train <- data_t10 %>%
  group_by(product, city) %>%
  slice(1:36)

t10_test <- data_t10 %>%
  group_by(product, city) %>%
  slice(37:48)

#Creating aggregates for Forecasting

data_t10_agg <- data_t10 %>%
  aggregate_key( product * city , sales = sum(sales))


train_agg <- t10_train %>%
  aggregate_key( product * city , sales = sum(sales))

test_agg <- t10_test %>%
  aggregate_key( product * city , sales = sum(sales))
```

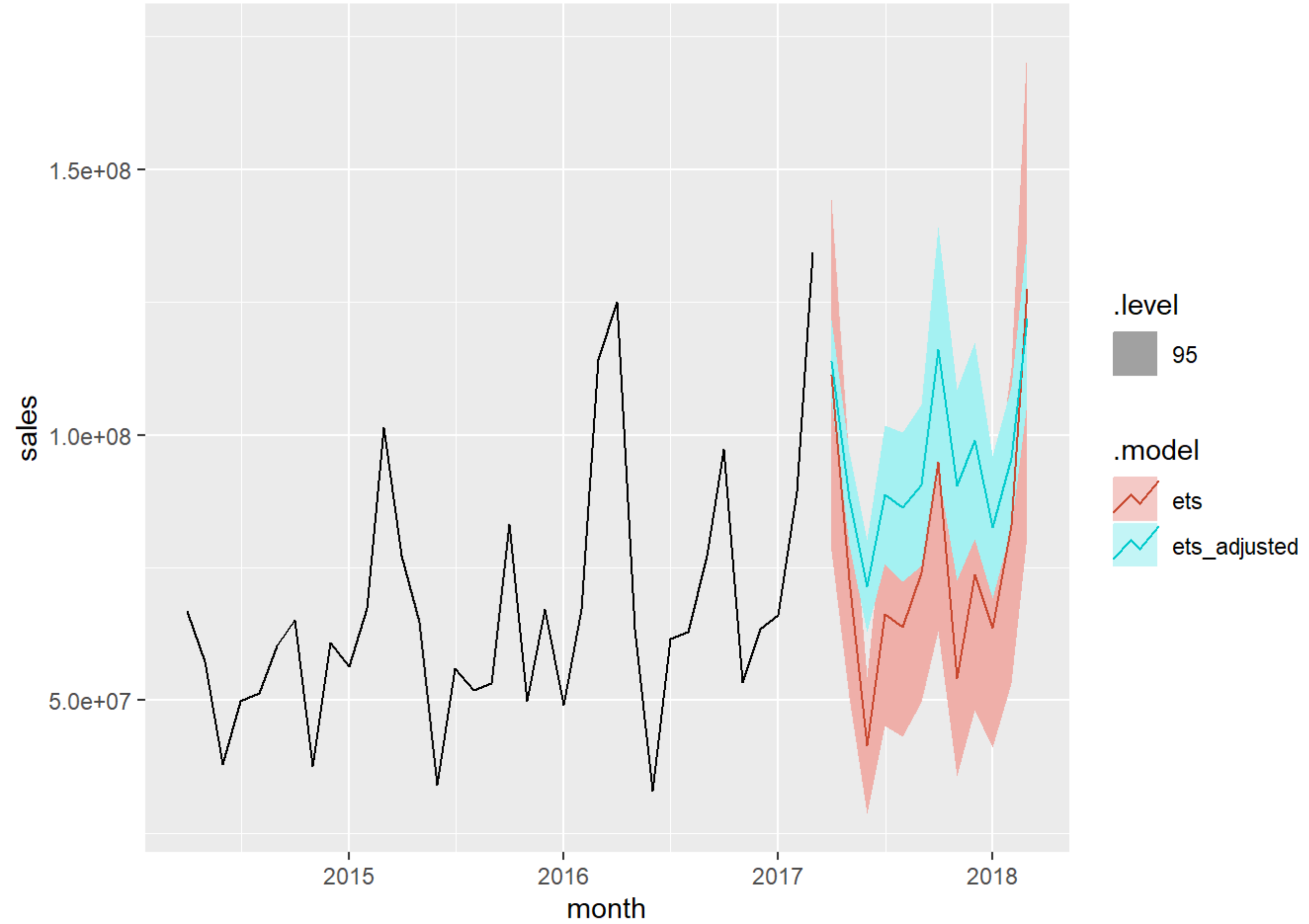
ETS Modelling:

We will use both simple ETS approach and Optimal reconciliation approach for ETS in modelling.

```
fc <- t10_train %>%
  aggregate_key(product*city, sales= sum(sales)) %>%
  model(ets = ETS(sales)) %>%
  reconcile(ets_adjusted = min_trace(ets)) %>%
  forecast(h=12)
```

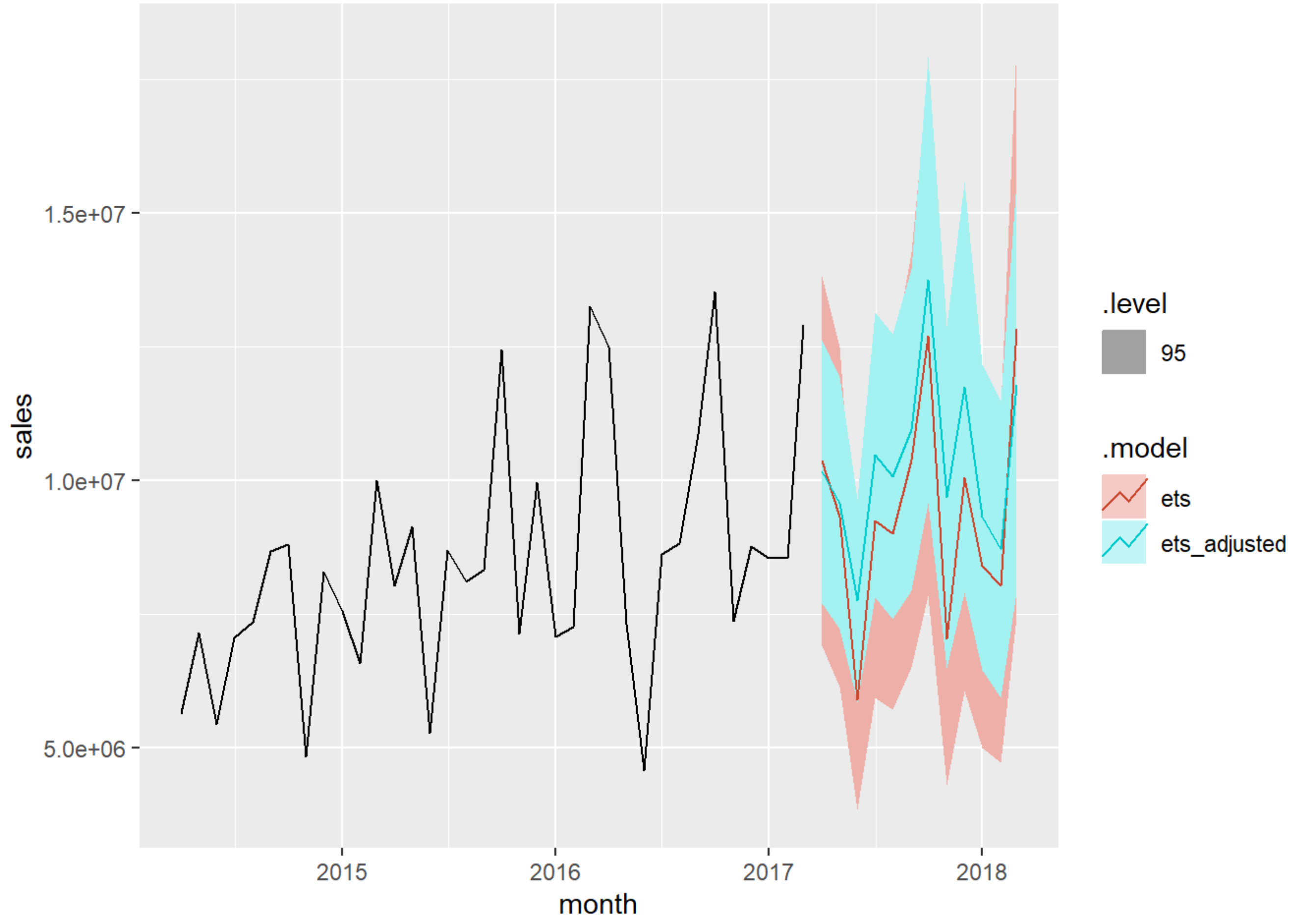
```
## Warning: Reconciliation in fable is highly experimental. The interface will
## likely be refined in the near future.
```

```
fc %>%
  filter(is_aggregated(product) & is_aggregated(city)) %>%
  autoplot(train_agg, level=95)
```

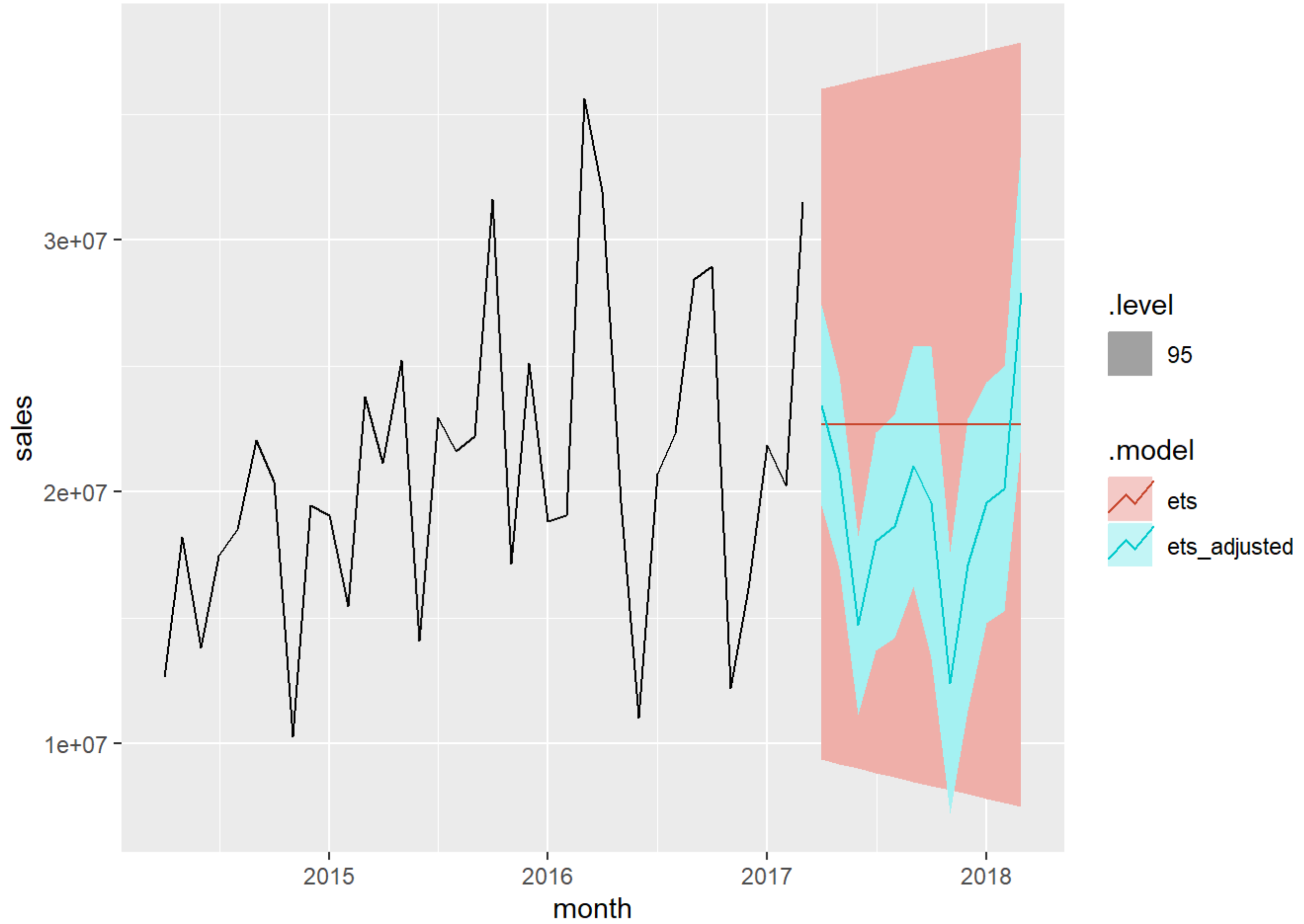


Forcasting Results for City- Kolkata & Forecasting results for Product Mixers.

```
#By City - Kolkata

fc %>%
  filter(is_aggregated(product) & city=="Kolkata") %>%
  autoplot(train_agg, level=95)
```




```
#By Product - Mixers

fc %>%
  filter(is_aggregated(city) & product=="Mixers") %>%
  autoplot(train_agg, level=95)
```



Forecast Evaluation - ETS Modelling

```
fc %>%
  accuracy(test_agg)
```

.mo… <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>
ets	coolers	Ahmd	Test	-5.571225e+05	1030529.27	1005403.70
ets	coolers	Bangalore	Test	5.794945e+05	1054898.46	597562.46
ets	coolers	Chennai	Test	-2.729410e+06	3083707.64	2815368.05
ets	coolers	Cochin	Test	-1.726321e+06	2066058.81	1876919.63
ets	coolers	Delhi	Test	-1.949519e+06	2222370.59	1999515.57
ets	coolers	Hyderabad	Test	-5.803882e+06	6313687.32	5803881.80
ets	coolers	Kolkata	Test	9.951736e+05	5157514.46	3928689.76
ets	coolers	Mumbai	Test	8.173727e+05	1569149.71	910745.25
ets	coolers	Patna	Test	2.753177e+05	1759744.55	1371478.37
ets	coolers	Pune	Test	2.467975e+04	129958.81	98088.78

1-10 of 242 rows | 1-8 of 11 columns Previous 1 2 3 4 5 6 ... 25 Next

```
fc %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(RMSE = mean(RMSE)) %>%
  arrange(RMSE)
```

.model <chr>	RMSE <dbl>
ets_adjusted	1528802
ets	1638783

2 rows

```
fc %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAPE = mean(MAPE)) %>%
  arrange(MAPE)
```

.model <chr>	MAPE <dbl>
ets	Inf
ets_adjusted	Inf

2 rows

```
fc %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MASE = mean(MASE)) %>%
  arrange(MASE)
```

.model <chr>	MASE <dbl>
ets	NaN
ets_adjusted	NaN

2 rows

```
fc %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAE = mean(MAE)) %>%
  arrange(MAE)
```

.model <chr>	MAE <dbl>
ets_adjusted	1289956

.model <chr>	MAE <dbl>
ets	1396718

2 rows

```
fc %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(ME = mean(ME)) %>%
  arrange(ME)
```

.model <chr>	ME <dbl>
ets_adjusted	17830.14
ets	91590.98

2 rows

ARIMA Modelling:

```
#ARIMA Modelling

fc1 <- t10_train %>%
  aggregate_key(product*city, sales= sum(sales)) %>%
  model(arima = ARIMA(sales)) %>%
  reconcile(arima_adjusted = min_trace(arima)) %>%
  forecast(h=12)
```

```
## Warning: Reconciliation in fable is highly experimental. The interface will
## likely be refined in the near future.
```

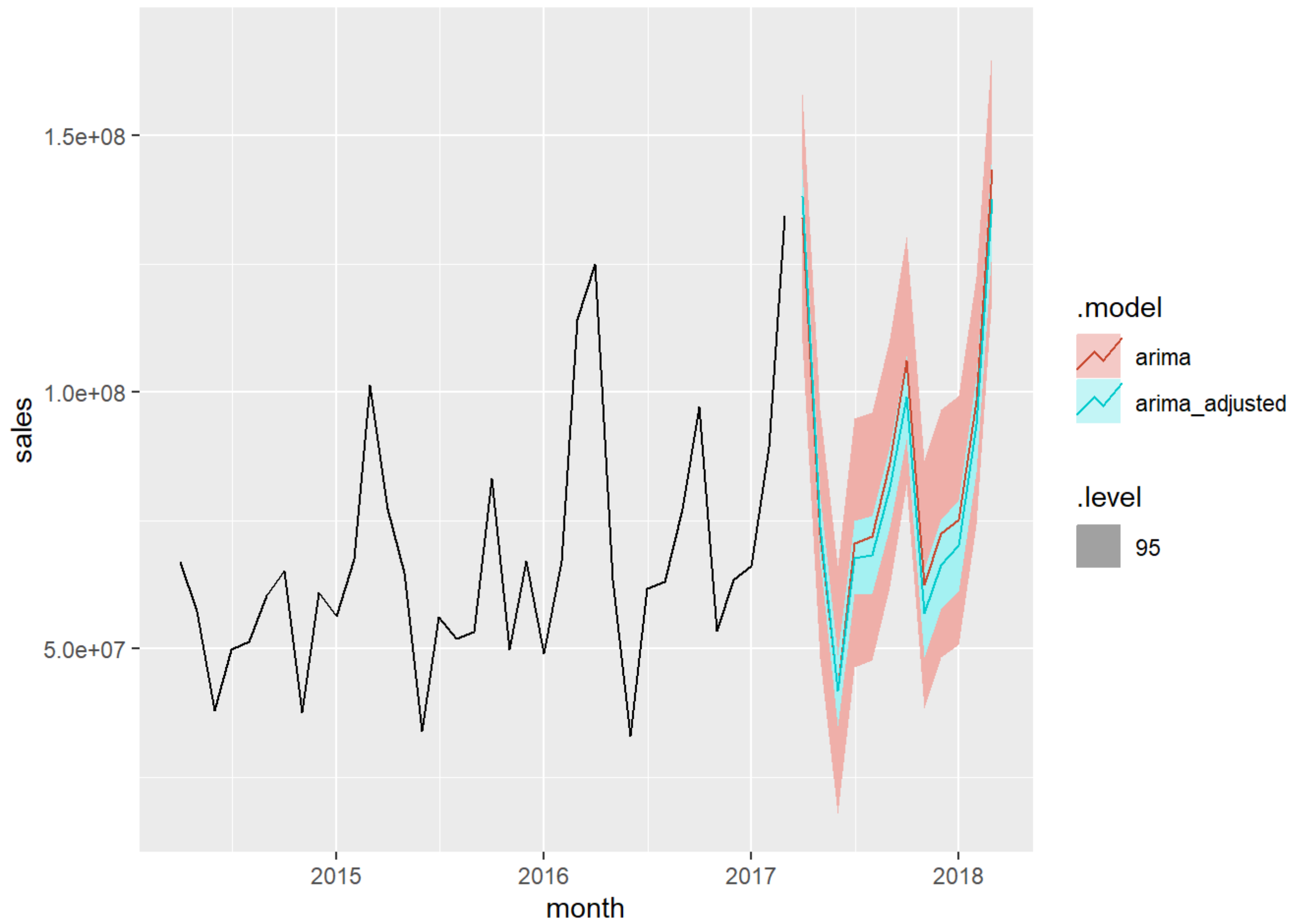
```
fc1
```

product <S3: agg_key>	city <S3: agg_key>	.model <chr>	month <S3: yearmonth>	sales <dbl>	.distribution <S3: fcdist>
coolers	Ahmd	arima	2017 Apr	6.352033e+06	<S3: fcdist>
coolers	Ahmd	arima	2017 May	2.340689e+06	<S3: fcdist>
coolers	Ahmd	arima	2017 Jun	4.739231e+05	<S3: fcdist>
coolers	Ahmd	arima	2017 Jul	2.719731e+05	<S3: fcdist>
coolers	Ahmd	arima	2017 Aug	1.739571e+05	<S3: fcdist>
coolers	Ahmd	arima	2017 Sep	1.112650e+05	<S3: fcdist>
coolers	Ahmd	arima	2017 Oct	7.837836e+04	<S3: fcdist>
coolers	Ahmd	arima	2017 Nov	8.468782e+04	<S3: fcdist>
coolers	Ahmd	arima	2017 Dec	6.527336e+04	<S3: fcdist>

product	city	.model	month	sales	.distribution
<S3: agg_key>	<S3: agg_key>	<chr>	<S3: yearmonth>	<dbl>	<S3: fcdist>
coolers	Ahmd	arima	2018 Jan	7.849179e+05	<S3: fcdist>

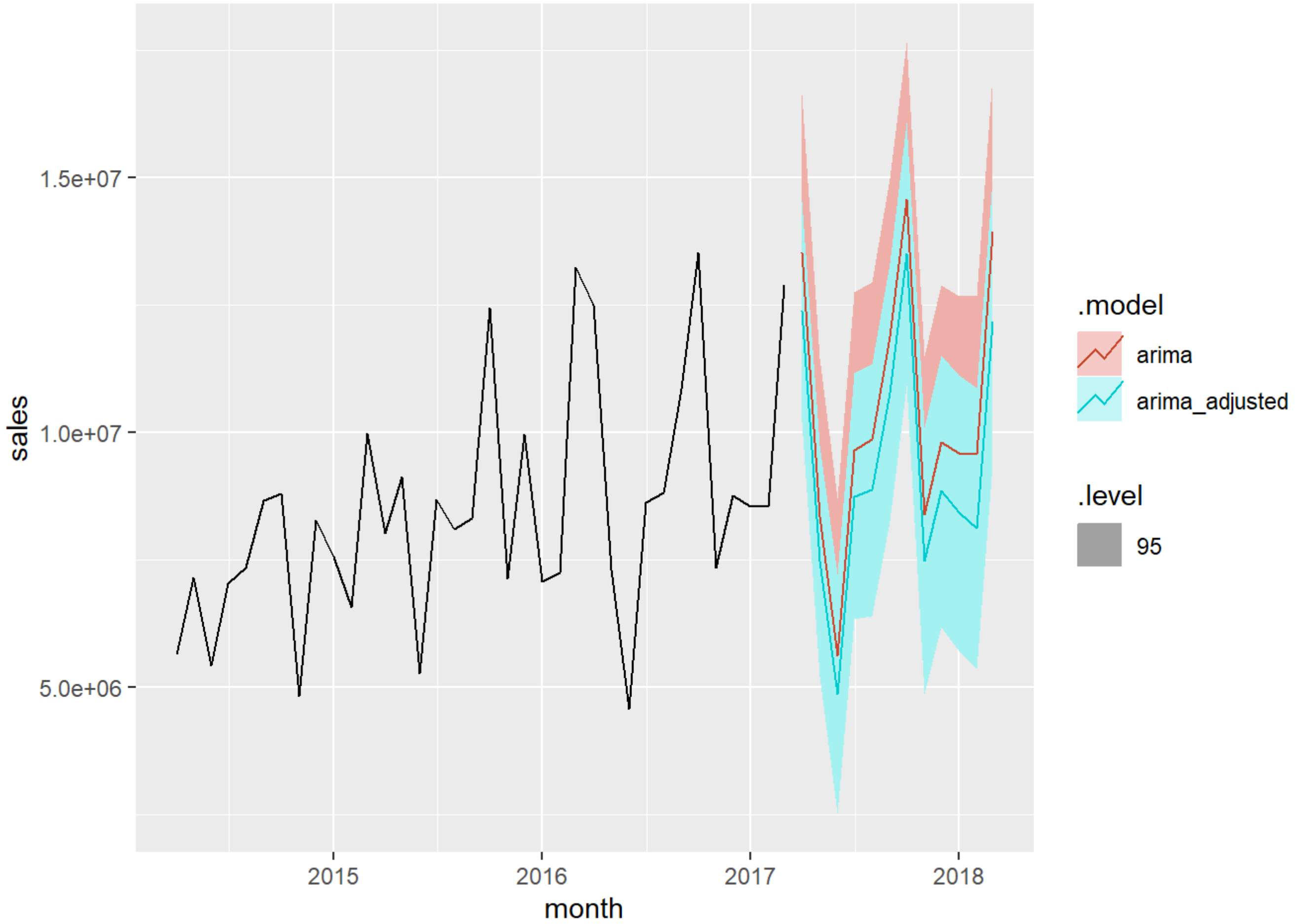
1-10 of 2,904 rows Previous **1** 2 3 4 5 6 ... 291 Next

```
fc1 %>%
  filter(is_aggregated(product) & is_aggregated(city)) %>%
  autoplot(train_agg, level=95)
```

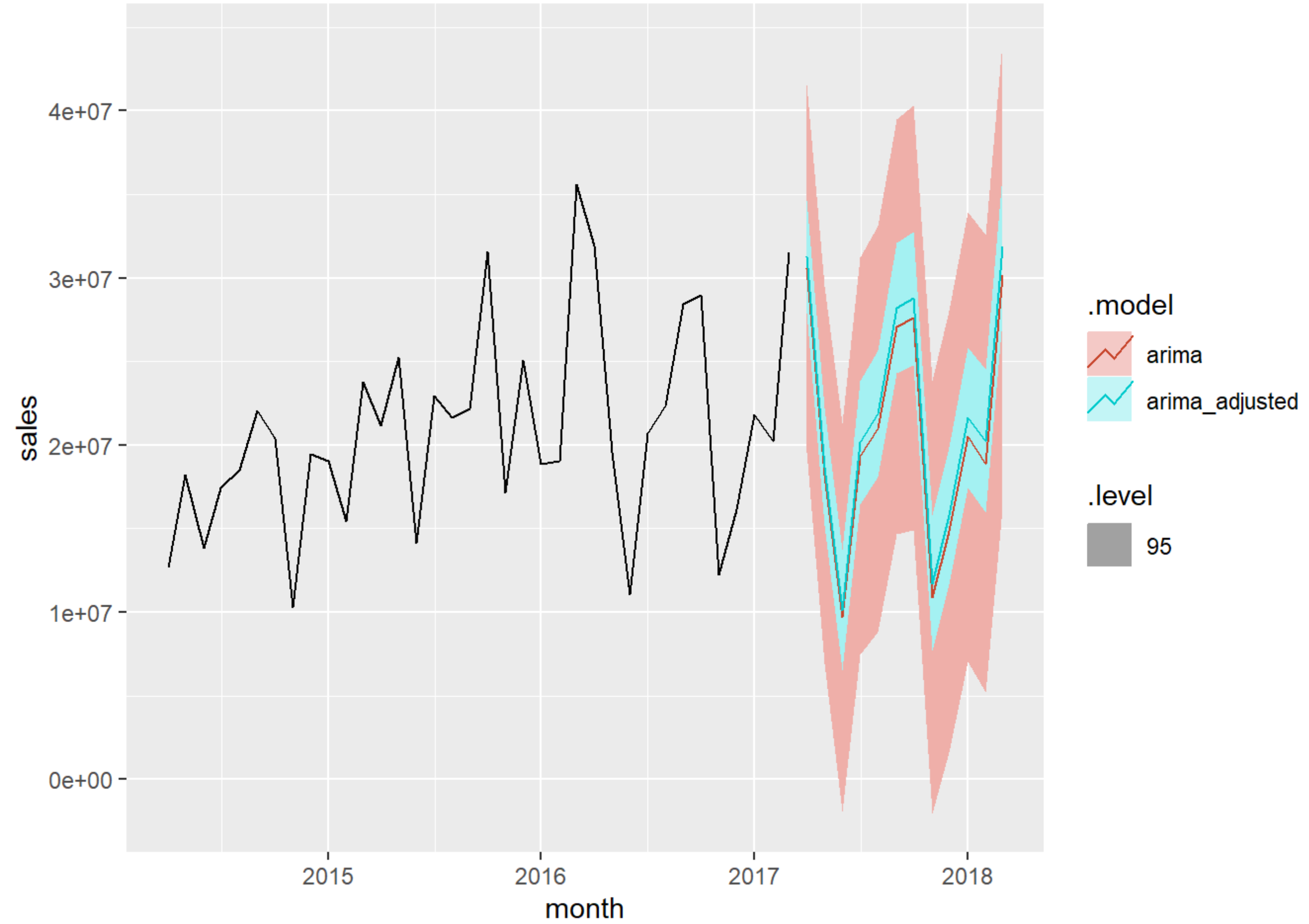


Plotting the ARIMA forecasting by product and city

```
#By City

fc1 %>%
  filter(is_aggregated(product) & city=="Kolkata") %>%
  autoplot(train_agg, level=95)
```




```
#By Product

fc1 %>%
  filter(is_aggregated(city) & product=="Mixers") %>%
  autoplot(train_agg, level=95)
```



Forecast Evaluation:

```
fc1 %>%
  accuracy(test_agg)
```

.mo… <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>
arima	coolers	Ahmd	Test	41602.834	439931.55	295531.85
arima	coolers	Bangalore	Test	660398.156	1172717.27	660398.16
arima	coolers	Chennai	Test	-861419.150	1266315.85	911553.05
arima	coolers	Cochin	Test	-625439.243	887716.88	790482.50
arima	coolers	Delhi	Test	-350582.982	486336.74	403428.72
arima	coolers	Hyderabad	Test	-1193349.210	1581665.38	1193349.21
arima	coolers	Kolkata	Test	2881775.542	5333746.94	3030830.49
arima	coolers	Mumbai	Test	315908.936	602919.94	315908.94
arima	coolers	Patna	Test	863436.769	1627877.98	929430.10
arima	coolers	Pune	Test	-177507.788	349131.75	177507.79

1-10 of 242 rows | 1-8 of 11 columns Previous 1 2 3 4 5 6 ... 25 Next

```
fc1 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(RMSE = mean(RMSE)) %>%
  arrange(RMSE)
```

.model <chr>	RMSE <dbl>
arima	1354859
arima_adjusted	1387081

2 rows

```
fc1 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAPE = mean(MAPE)) %>%
  arrange(MAPE)
```

.model <chr>	MAPE <dbl>
arima	Inf
arima_adjusted	Inf

2 rows

```
fc1 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MASE = mean(MASE)) %>%
  arrange(MASE)
```

.model <chr>	MASE <dbl>
arima	NaN
arima_adjusted	NaN

2 rows

```
fc1 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAE = mean(MAE)) %>%
  arrange(MAE)
```

.model <chr>	MAE <dbl>
arima	1127788

.model <chr>	MAE <dbl>
arima_adjusted	1164003

2 rows

```
fc1 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(ME = mean(ME)) %>%
  arrange(ME)
```

.model <chr>	ME <dbl>
arima	430148.2
arima_adjusted	431920.5

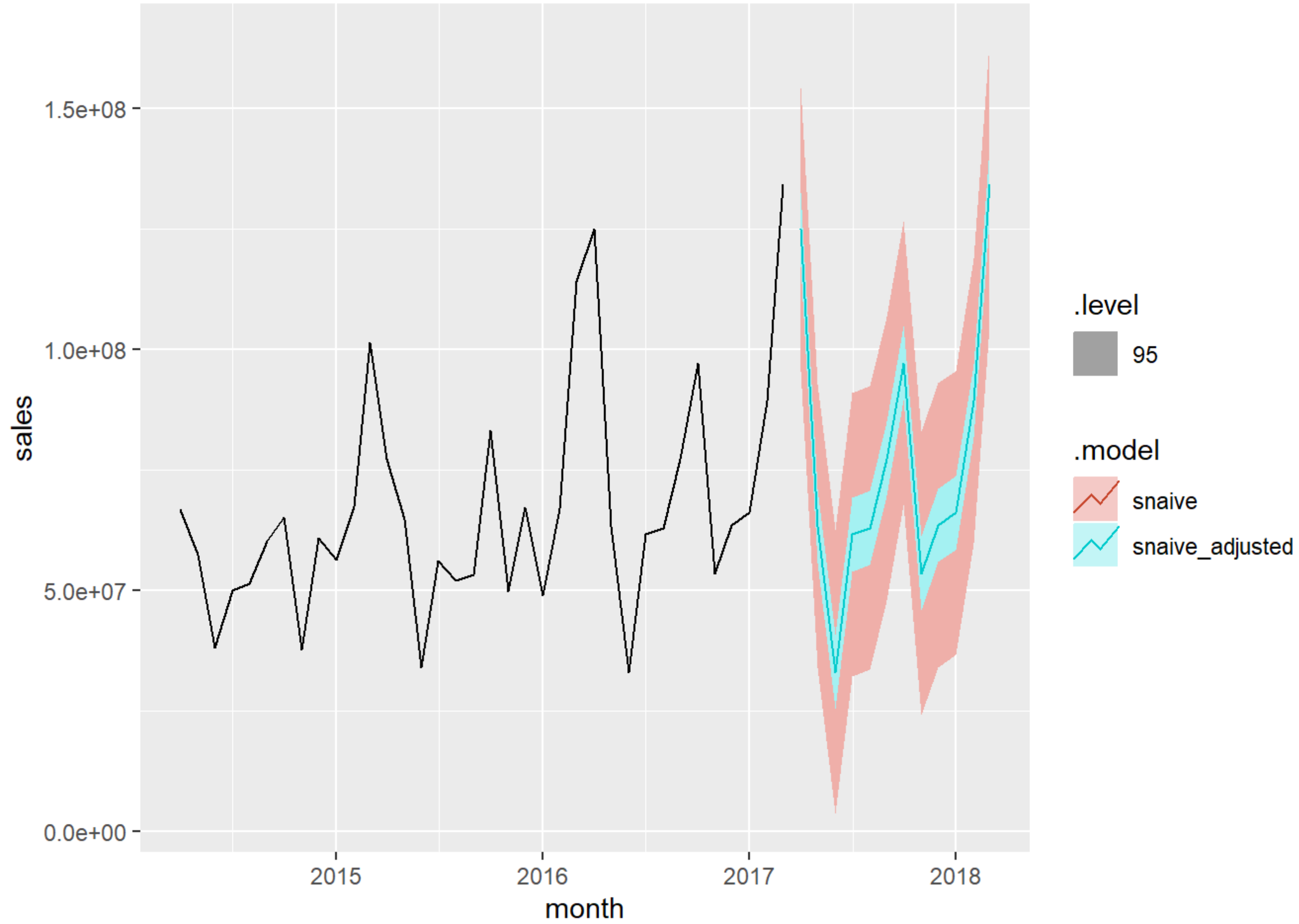
2 rows

SNAIVE Modelling:

```
fc2 <- t10_train %>%
  aggregate_key(product*city, sales= sum(sales)) %>%
  model(snaive = SNAIVE(sales)) %>%
  reconcile(snaive_adjusted = min_trace(snaive)) %>%
  forecast(h=12)
```

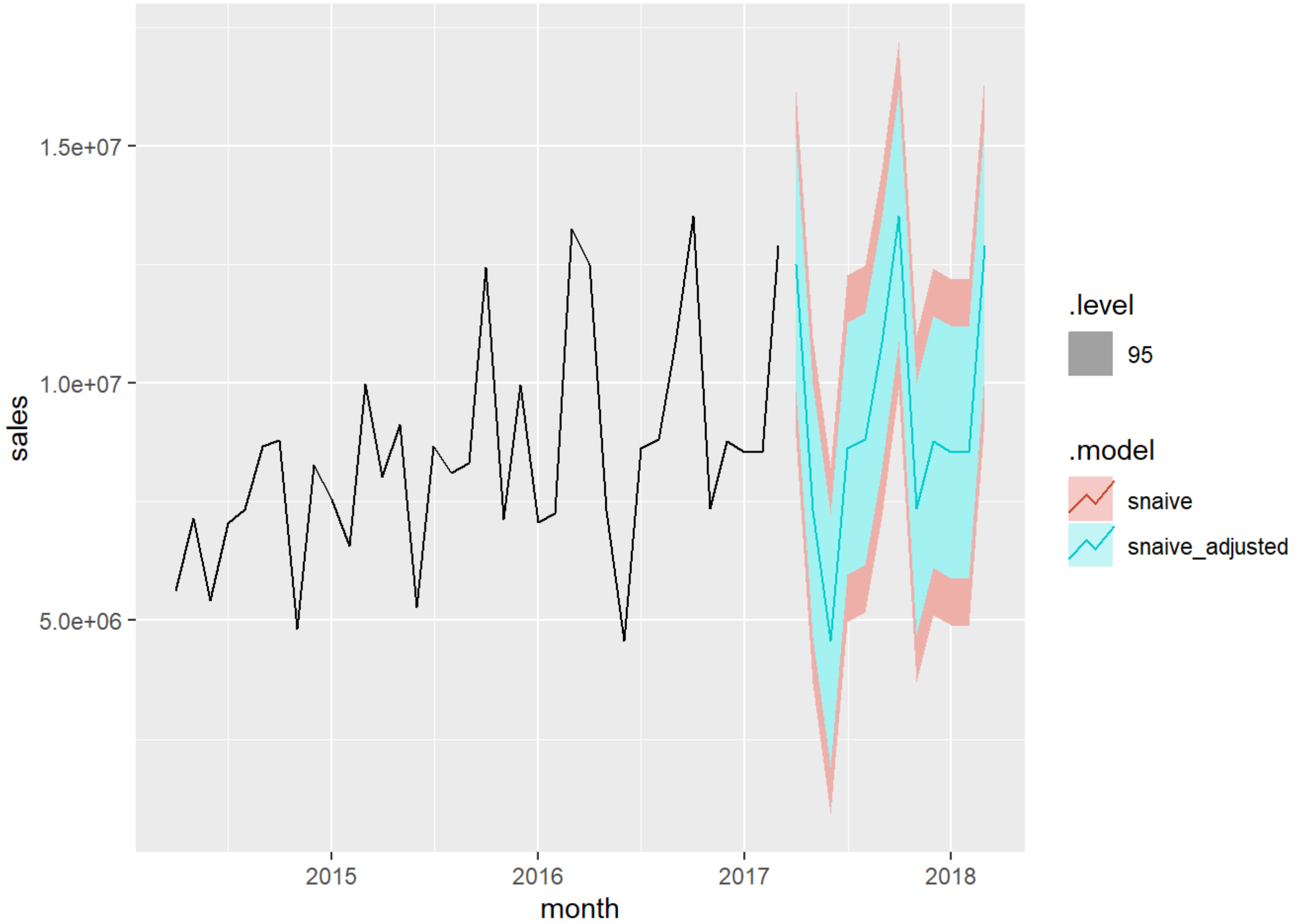
```
## Warning: Reconciliation in fable is highly experimental. The interface will
## likely be refined in the near future.
```

```
fc2 %>%
  filter(is_aggregated(product) & is_aggregated(city)) %>%
  autoplot(train_agg, level=95)
```

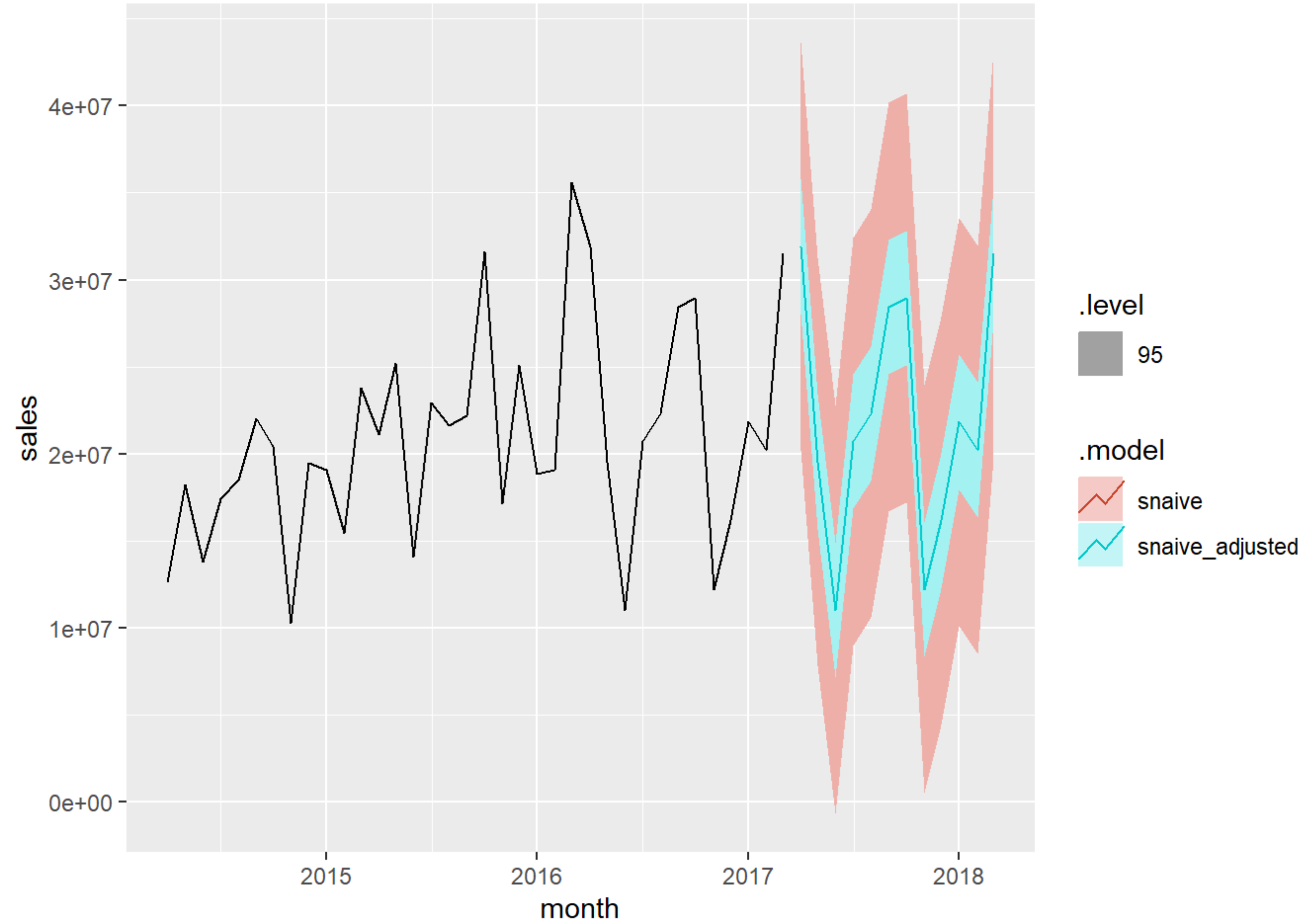


Plotting the SNAIVE Forecasting results by City Wise and Product Wise

```
#By City

fc2 %>%
  filter(is_aggregated(product) & city=="Kolkata") %>%
  autoplot(train_agg, level=95)
```




```
#By Product

fc2 %>%
  filter(is_aggregated(city) & product=="Mixers") %>%
  autoplot(train_agg, level=95)
```



Forecast Evaluation for SNAIVE:

```
fc2 %>%
  accuracy(test_agg)
```

.mo… <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>
snaive	coolers	Ahmd	Test	280815.500	498665.11	280815.50
snaive	coolers	Bangalore	Test	575740.083	1022383.98	575740.08
snaive	coolers	Chennai	Test	74996.917	133177.34	74996.92
snaive	coolers	Cochin	Test	214236.000	380434.72	214236.00
snaive	coolers	Delhi	Test	73251.667	130078.46	73251.67
snaive	coolers	Hyderabad	Test	-137693.250	244512.30	137693.25
snaive	coolers	Kolkata	Test	2849294.750	5059702.82	2849294.75
snaive	coolers	Mumbai	Test	406580.833	721995.75	406580.83
snaive	coolers	Patna	Test	962359.500	1708932.61	962359.50
snaive	coolers	Pune	Test	-131636.167	233756.08	131636.17

1-10 of 242 rows | 1-8 of 11 columns Previous **1** 2 3 4 5 6 ... 25 Next

```
fc2 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(RMSE = mean(RMSE)) %>%
  arrange(RMSE)
```

.model <chr>	RMSE <dbl>
snaive	1366817
snaive_adjusted	1366817

2 rows

```
fc2 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAPE = mean(MAPE)) %>%
  arrange(MAPE)
```

.model <chr>	MAPE <dbl>
snaive	66.98614
snaive_adjusted	Inf

2 rows

```
fc2 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MASE = mean(MASE)) %>%
  arrange(MASE)
```

.model <chr>	MASE <dbl>
snaive	NaN
snaive_adjusted	NaN

2 rows

```
fc2 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAE = mean(MAE)) %>%
  arrange(MAE)
```

.model <chr>	MAE <dbl>
snaive	1127574

.model <chr>	MAE <dbl>
snaive_adjusted	1127574

2 rows

```
fc2 %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(ME = mean(ME)) %>%
  arrange(ME)
```

.model <chr>	ME <dbl>
snaive_adjusted	617333.4
snaive	617333.4

2 rows

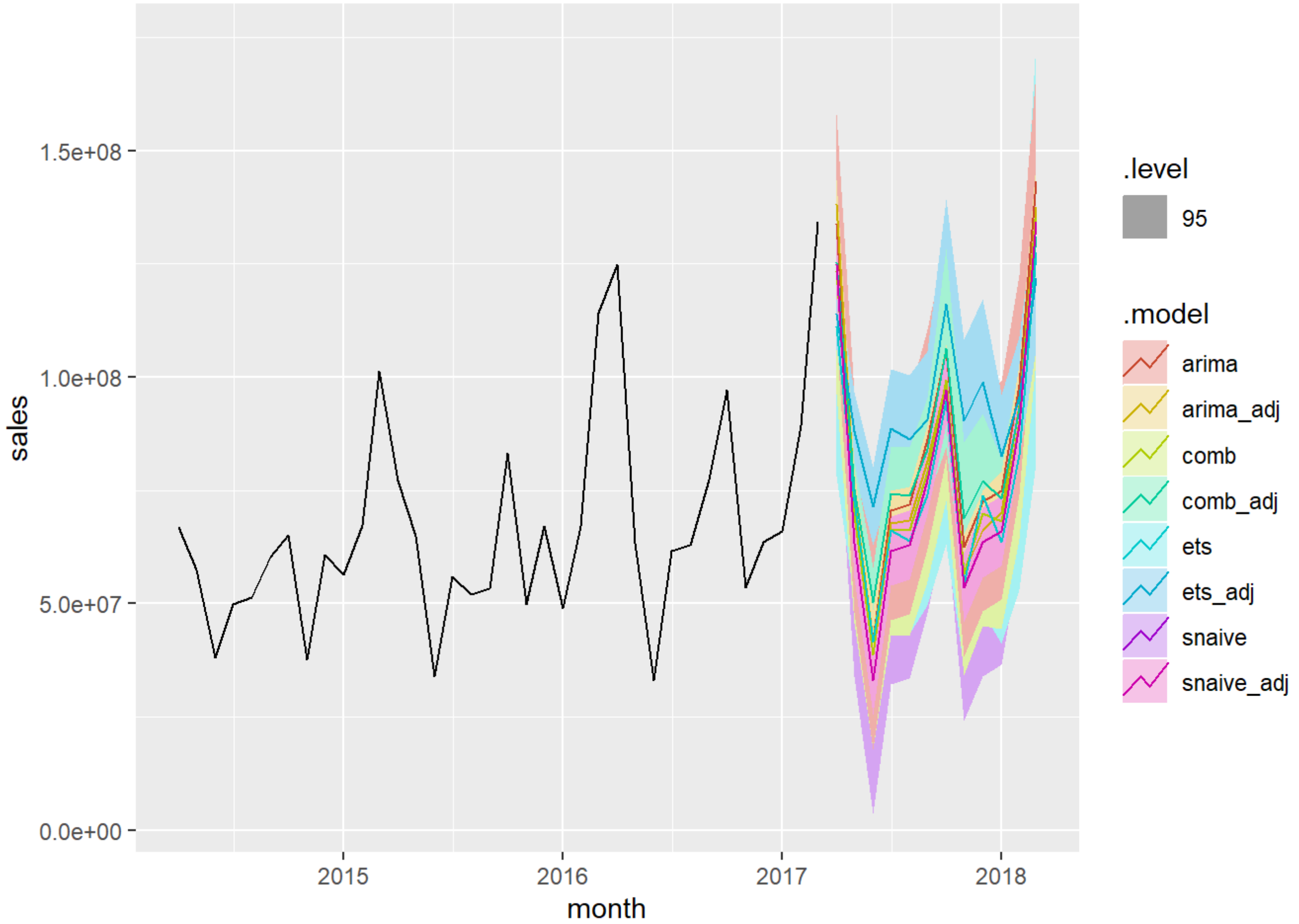
We are getting MAPE as infinity because some of the observations in the time series are zero. Also, MASE is displayed as NaN due the requirement of minimum of 13 observations in validation data for calculation. We will now check the Ensemble approach. We take a simple average of all the forecasting results and create an ensemble model.

Ensemble Approach:

```
fc_comb <- train_agg %>%
  model(
    ets = ETS(sales),
    arima = ARIMA(sales),
    snaive = SNAIVE(sales)
  ) %>%
  mutate(
    comb = (ets+arima+snaive)/3
  ) %>%
  reconcile(
    ets_adj = min_trace(ets),
    arima_adj = min_trace(arima),
    snaive_adj = min_trace(snaive),
    comb_adj = min_trace(comb)
  ) %>%
  forecast(h = 12)
```

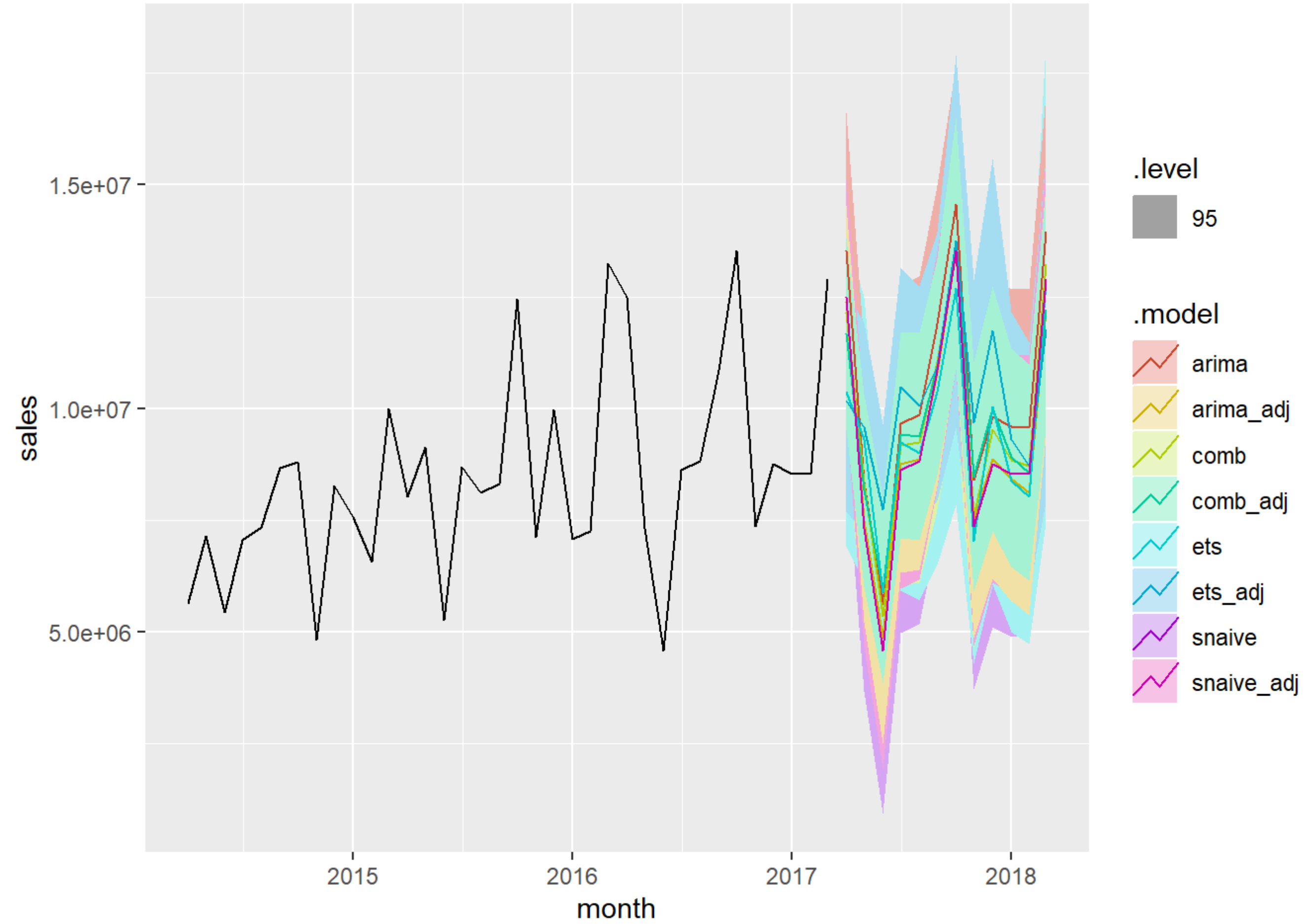
```
## Warning: Reconciliation in fable is highly experimental. The interface will
## likely be refined in the near future.
```

```
fc_comb %>%
  filter(is_aggregated(product) & is_aggregated(city)) %>%
  autoplot(train_agg, level=95)
```

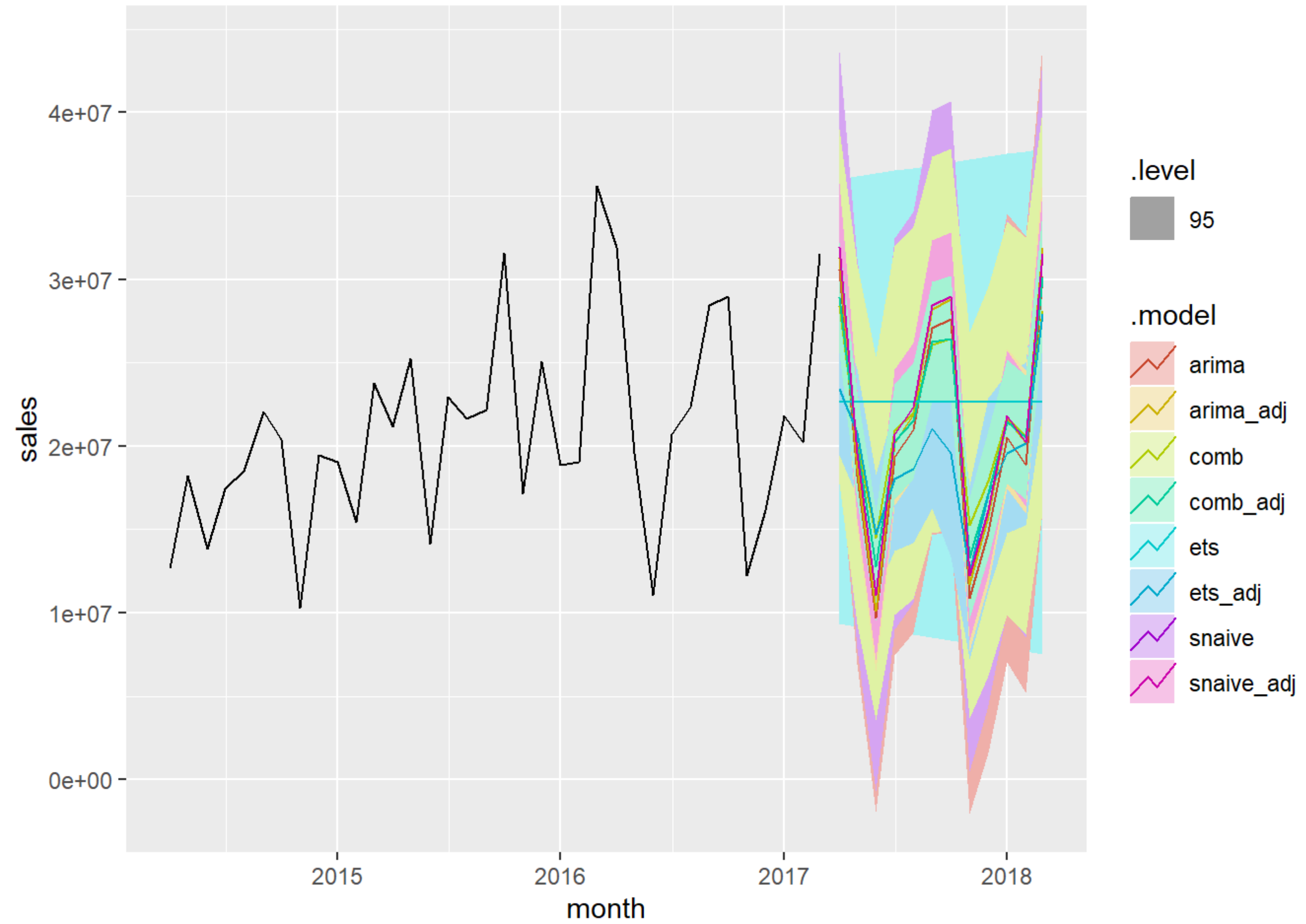


Forecast Plotting of Ensemble Model by City and Product

```
#By City

fc_comb %>%
  filter(is_aggregated(product) & city=="Kolkata") %>%
  autoplot(train_agg, level=95)
```




```
#By Product

fc_comb %>%
  filter(is_aggregated(city) & product=="Mixers") %>%
  autoplot(train_agg, level=95)
```



Forecast Evaluation - Ensemble Model

```
fc_comb %>%
  accuracy(test_agg)
```

.mo… <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>
arima	coolers	Ahmd	Test	4.160283e+04	439931.55	295531.85
arima	coolers	Bangalore	Test	6.603982e+05	1172717.27	660398.16
arima	coolers	Chennai	Test	-8.614192e+05	1266315.85	911553.05
arima	coolers	Cochin	Test	-6.254392e+05	887716.88	790482.50
arima	coolers	Delhi	Test	-3.505830e+05	486336.74	403428.72
arima	coolers	Hyderabad	Test	-1.193349e+06	1581665.38	1193349.21
arima	coolers	Kolkata	Test	2.881776e+06	5333746.94	3030830.49
arima	coolers	Mumbai	Test	3.159089e+05	602919.94	315908.94
arima	coolers	Patna	Test	8.634368e+05	1627877.98	929430.10
arima	coolers	Pune	Test	-1.775078e+05	349131.75	177507.79

1-10 of 968 rows | 1-8 of 11 columns Previous **1** 2 3 4 5 6 ... 97 Next

```
fc_comb %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(RMSE = mean(RMSE)) %>%
  arrange(RMSE)
```

.model <chr>	RMSE <dbl>
comb_adj	1327512
arima	1354859
snaive	1366817
snaive_adj	1366817
comb	1367792
arima_adj	1387081
ets_adj	1528802
ets	1638783

8 rows

```
fc_comb %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAPE = mean(MAPE)) %>%
  arrange(MAPE)
```

.model <chr>	MAPE <dbl>
snaive	66.98614
arima	Inf
arima_adj	Inf
comb	Inf
comb_adj	Inf
ets	Inf
ets_adj	Inf
snaive_adj	Inf

8 rows

```
fc_comb %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MASE = mean(MASE)) %>%
  arrange(MASE)
```

.model <chr>	MASE <dbl>
arima	NaN
arima_adj	NaN
comb	NaN
comb_adj	NaN
ets	NaN
ets_adj	NaN
snaive	NaN
snaive_adj	NaN

8 rows

```
fc_comb %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(MAE = mean(MAE)) %>%
  arrange(MAE)
```

.model <chr>	MAE <dbl>
comb_adj	1122383
snaive	1127574
snaive_adj	1127574
arima	1127788
comb	1158951
arima_adj	1164003
ets_adj	1289956
ets	1396718

8 rows

```
fc_comb %>%
  accuracy(test_agg) %>%
  group_by(.model) %>%
  summarise(ME = mean(ME)) %>%
  arrange(ME)
```

.model <chr>	ME <dbl>
ets_adj	17830.14
ets	91590.98

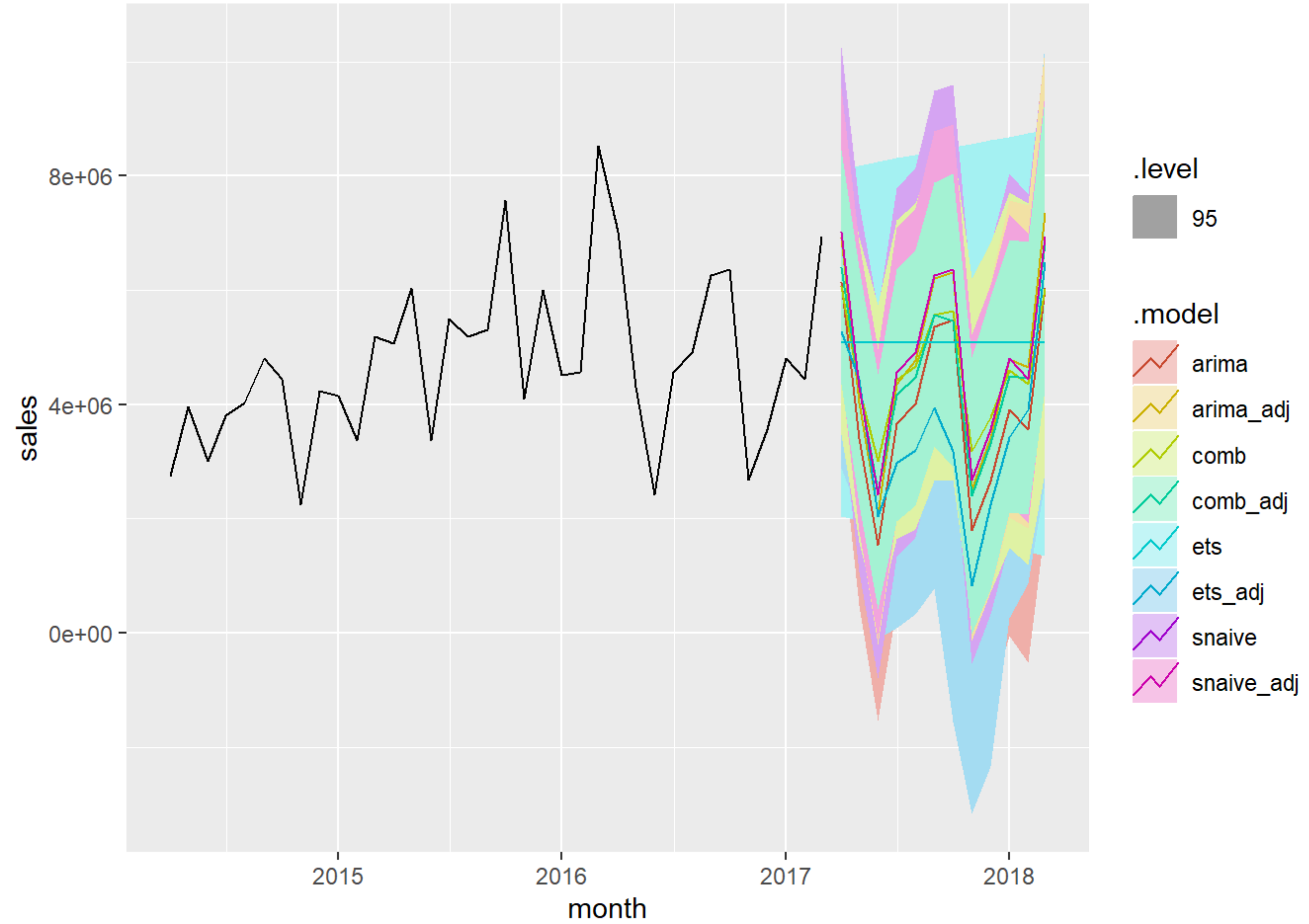
.model <chr>	ME <dbl>
comb_adj	326960.92
comb	379690.87
arima	430148.21
arima_adj	431920.50
snaive_adj	617333.41
snaive	617333.41

8 rows

In order to better understand the forecasting results and accuracy measurements we will test our models on three individual series foor 4 Products and 4 Cities Combinations. They are Mixers - Kolkata, Coolers - Mumbai, Dry Iron - Bangalore, and Water Heaters - Hyderabad

```
# Kolkate - Mixers

kol_mix <- fc_comb %>%
  filter(product == "Mixers" & city == "Kolkata")

fc_comb %>%
  filter(product == "Mixers" & city == "Kolkata") %>%
  autoplot(train_agg, ylab = "Mixer Sales Forecasting for Kolkata", level = 95)
```

```
## Warning: Ignoring unknown parameters: ylab
```

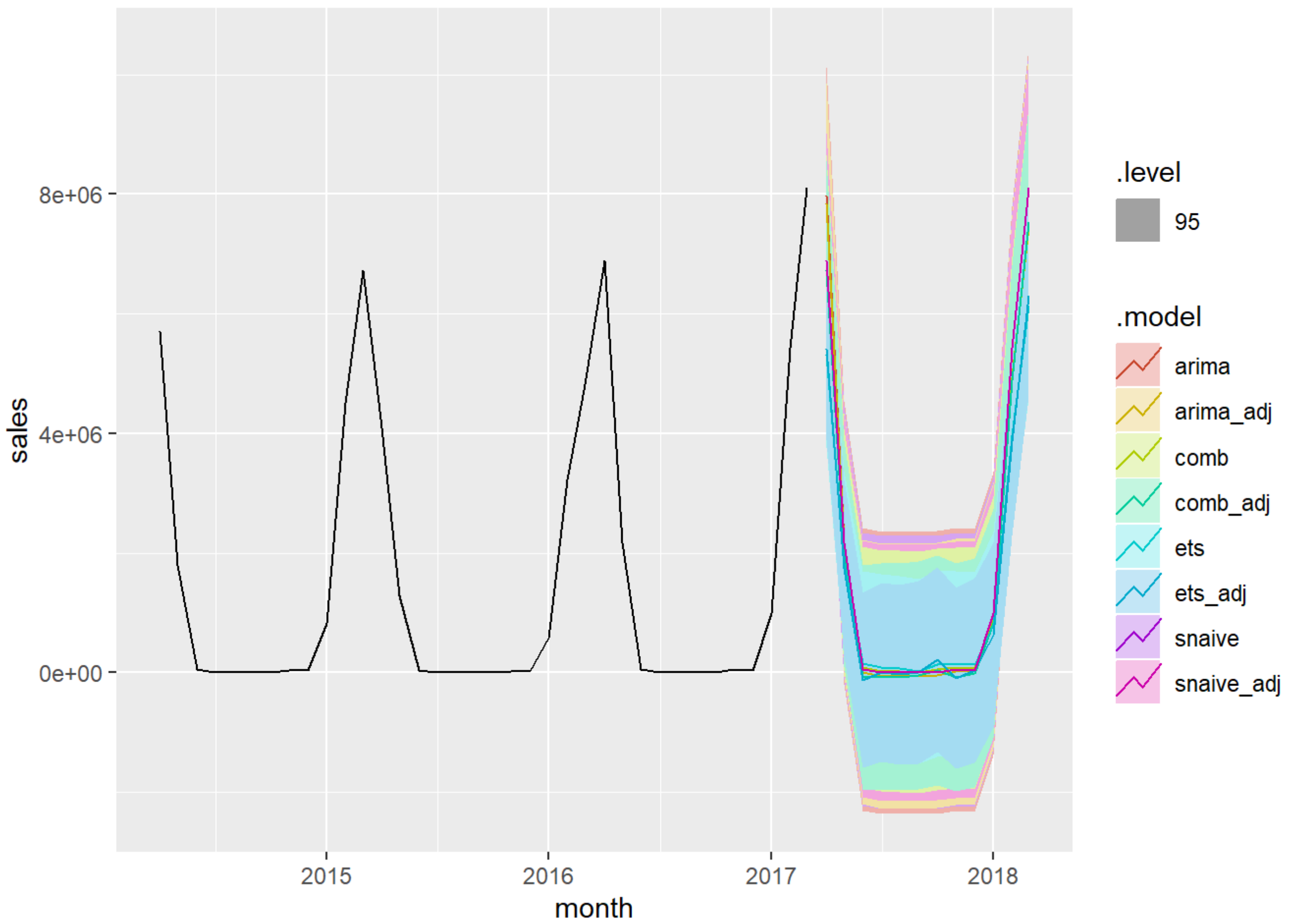



```
fc_comb %>%
  filter(product == "Mixers" & city == "Kolkata") %>%
  accuracy(test_agg) %>%
  group_by(.model)
```

.model <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>	MAPE <dbl>	M <
arima	Mixers	Kolkata	Test	3781391	4800647	4207430	40.29986	52.21885	N
arima_adj	Mixers	Kolkata	Test	2963681	4245159	3817853	27.54177	49.16791	N
comb	Mixers	Kolkata	Test	3113374	4256827	3615220	29.61970	43.48695	N
comb_adj	Mixers	Kolkata	Test	3216345	4440325	3861354	30.95288	47.89634	N
ets	Mixers	Kolkata	Test	2669597	4045273	3258668	21.97189	37.42417	N
ets_adj	Mixers	Kolkata	Test	4248567	5456969	4563194	45.52152	54.05654	N
snaive	Mixers	Kolkata	Test	2889134	4134536	3656750	26.58733	46.71463	N
snaive_adj	Mixers	Kolkata	Test	2889134	4134536	3656750	26.58733	46.71463	N

8 rows | 1-10 of 11 columns

```
#Mumbai - Coolers

mum_col <- fc_comb %>%
  filter(product == "coolers" & city == "Mumabi")

fc_comb %>%
  filter(product == "coolers" & city == "Mumbai") %>%
  autoplot(train_agg, level = 95)
```



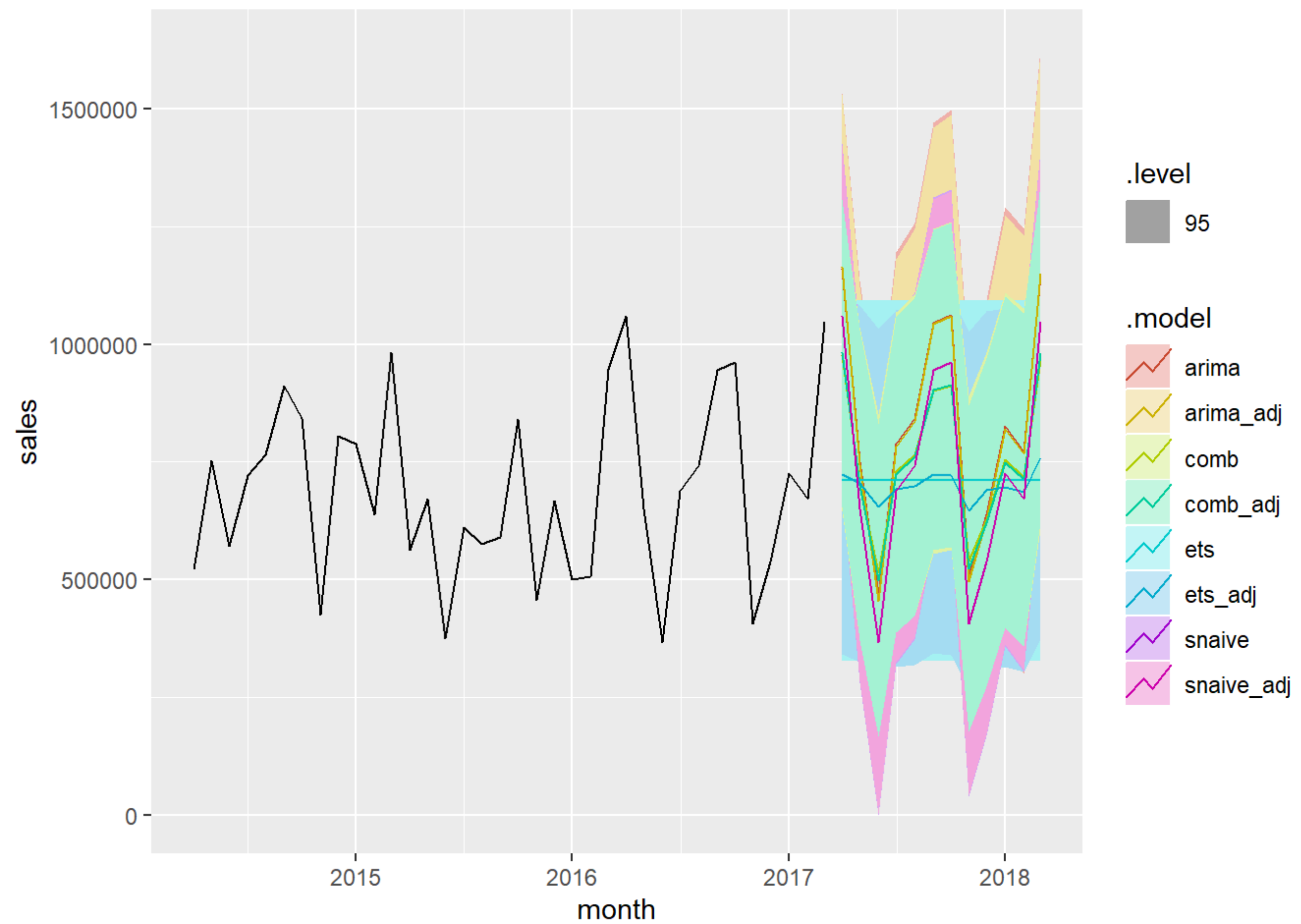

```
fc_comb %>%
  filter(product == "coolers" & city == "Mumbai") %>%
  accuracy(test_agg) %>%
  group_by(.model)
```

.model <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>	MA <d
arima	coolers	Mumbai	Test	315908.9	602919.9	315908.9	Inf	
arima_adj	coolers	Mumbai	Test	360932.9	614047.5	360932.9	Inf	
comb	coolers	Mumbai	Test	513287.5	949504.3	540530.5	-Inf	
comb_adj	coolers	Mumbai	Test	581771.2	952607.9	586816.1	Inf	
ets	coolers	Mumbai	Test	817372.7	1569149.7	910745.2	-Inf	
ets_adj	coolers	Mumbai	Test	887273.5	1549095.4	924707.1	NaN	

.model <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>	MA <d
snaive	coolers	Mumbai	Test	406580.8	721995.8	406580.8	17.03949	17.039
snaive_adj	coolers	Mumbai	Test	406580.8	721995.8	406580.8	NaN	

8 rows | 1-10 of 11 columns

```
#Bangalore - Dry Iron

ban_di <- fc_comb %>%
  filter(product == "Dry Iron" & city == "Bangalore")

fc_comb %>%
  filter(product == "Dry Iron" & city == "Bangalore") %>%
  autoplot(train_agg, level = 95)
```




```
fc_comb %>%
  filter(product == "Dry Iron" & city == "Bangalore") %>%
  accuracy(test_agg) %>%
  group_by(.model)
```

.model <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>

.model <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>	
arima	Dry Iron	Bangalore	Test	-133268.9048	340913.1	236873.9	-43.92867	5
arima_adj	Dry Iron	Bangalore	Test	-128575.5576	341707.9	236522.3	-43.16077	5
comb	Dry Iron	Bangalore	Test	-58651.5652	294359.4	221841.2	-31.20615	4
comb_adj	Dry Iron	Bangalore	Test	-55466.9210	295594.4	223054.8	-30.55804	4
ets	Dry Iron	Bangalore	Test	-9809.7075	292995.8	216913.5	-23.16664	4
ets_adj	Dry Iron	Bangalore	Test	805.3293	284943.3	216321.6	-20.81119	4
snaive	Dry Iron	Bangalore	Test	-32876.0833	315502.8	230740.4	-26.52314	4
snaive_adj	Dry Iron	Bangalore	Test	-32876.0833	315502.8	230740.4	-26.52314	4

8 rows | 1-10 of 11 columns

```
##Hyderabad - Water Heaters

fc_comb %>%
  filter(product == "Water Heaters" & city == "Hyderabad") %>%
  accuracy(test_agg) %>%
  group_by(.model)
```

.model <chr>	product <S3: agg_key>	city <S3: agg_key>	.type <chr>	ME <dbl>	RMSE <dbl>	MAE <dbl>	MPE <dbl>	I
arima	Water Heaters	Hyderabad	Test	937104.7	1215551	937104.7	105.03816	105.
arima_adj	Water Heaters	Hyderabad	Test	938550.7	1210332	938550.7	113.85903	113.
comb	Water Heaters	Hyderabad	Test	928817.2	1213290	928817.2	89.20649	89.
comb_adj	Water Heaters	Hyderabad	Test	933417.0	1217447	933417.0	90.94454	90.
ets	Water Heaters	Hyderabad	Test	932551.4	1221545	932551.4	82.05673	82.
ets_adj	Water Heaters	Hyderabad	Test	936136.1	1226010	936136.1	82.24833	82.
snaive	Water Heaters	Hyderabad	Test	916795.6	1202886	916795.6	80.52457	80.
snaive_adj	Water Heaters	Hyderabad	Test	916795.6	1202886	916795.6	80.52457	80.

8 rows | 1-10 of 11 columns