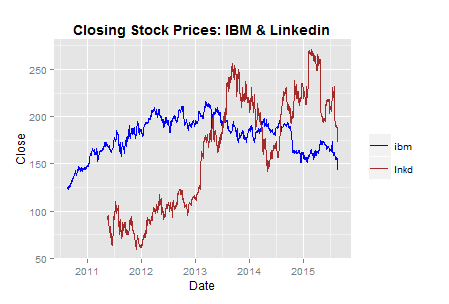
Time Series(Fastenal)

Concept: What is Time Series?

A time series is a series of [data points](https://en.wikipedia.org/wiki/Data_point) indexed (or listed or graphed) in time order. Most commonly, a time series is a sequence taken at successive equally spaced points in time. Thus it is a sequence of [discrete-time](https://en.wikipedia.org/wiki/Discrete-time) data. (Wikipedia)

example: The stock prices.



Our problem: To use 4 years(48 months) usages to predict next month’s usage for Fastenal top sale items.

Tools: arima model.(R)

What is arima model?

In statistics and econometrics, and in particular in time series analysis, an autoregressive integrated moving average (ARIMA) model is a generalization of an autoregressive moving average (ARMA) model. Both of these models are fitted to time series data either to better understand the data or to predict future points in the series (forecasting). ARIMA models are applied in some cases where data show evidence of non-stationarity, where an initial differencing step (corresponding to the "integrated" part of the model) can be applied one or more times to eliminate the non-stationarity.(Wikipedia)

Code of arima model(model 1.1):

fit <- auto.arima(ts)

my.array <- forecast(fit)

forecast(fit) will include next 10 months’ 80% confidence level, 95% confidence level, mean value of confidence level.

Mean value of prediction value: my.array$mean[1]

Max value of 80 % confidence level: my.array$upper[1]

Min value of 80% confidence level: my.array$lower[1]

What is the confidence level?

It is an observed interval (i.e., it is calculated from the observations), in principle different from [sample](https://en.wikipedia.org/wiki/Sample_(statistics)) to sample, that potentially includes the unobservable true parameter of interest.(Wikipedia)

80% confidence level for our project is mean: out of 100 trials, the outcome will be in the prediction interval 80 times.

Our advanced time series model (model1.2)based on arima model(model1.1):

Code:

library(fpp)

x <- c(49)

tep <- 0

pridictRange <- 0

errorRange <- 0

pCount <- 0

eCount <- 0

#errorCollect <- 0

total <- 0

errout <- 48

for(div in 1:9){

div <- div\*0.1

for (num in 1:69){

e <- c()

for(err in 25:48){

low <- c(num)

upp <- c(num)

fit <- auto.arima(small[20:err,num])

my.array <- forecast(fit)

e <- append(e,as.numeric(my.array$mean[1])-as.numeric(small[err+1,num]))

}

fiterr <- auto.arima(e)

err.array <- forecast(fiterr)

min <- as.numeric(my.array$mean[1])-as.numeric(err.array$upper[1])

if(pridictRange > errorRange){

max <- as.numeric(my.array$mean[1])-as.numeric(err.array$lower[1])

point <- min + (max-min)\*div

# errorCollect <- errorCollect+abs(point-as.numeric(small[errout+1,num]))

e <-c()

total<- total + as.numeric(small[48,num])

}

print("total")

print(total)

print(div)

#print(errorCollect)

# errorCollect <- 0

}

What advanced model(model 1.2) do?

Instead of just using auto.arima() to get prediction value.

Advanced model:

1. Collect errors for each months(30-48): P(mean)- Usages
2. Do prediction on errors: Predict error for 49 month -> E(max), E(min).
3. Get new confidence interval: P(mean) + E(max); P(mean) + P(min)
4. Normal Model: P(mean) = arima()

The code is trying to make auto.arima() better.

Why model 1.2 better than model 1.1?

1. Accuracy: Predict months 30-48(with same 62 items as normal model)
   1. Real value is in prediction interval 948 times -- 80.5% --
   2. Real value is in not prediction interval 230 times
   3. 80.5%(Advanced Model) > 78.1%(Normal model)

2. Prediction Interval :

Advanced Model < Normal Model (711 times)

Normal Model < Advanced Model (529 times)

3. Summary:

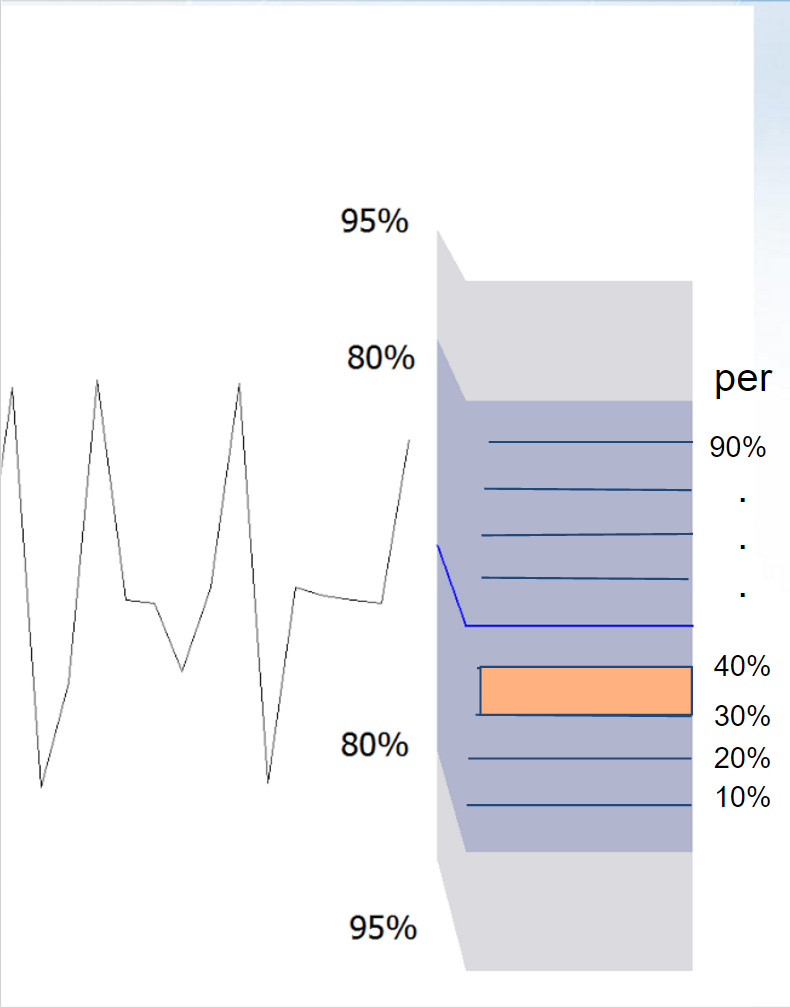
Advanced Model is more accurate than normal model;

The confidence Interval of Advanced Model is smaller than normal model.

Disadvantages of advanced model(model 1.2): It takes longer than normal model(model 1.1)

Predict a number instead of an interval:

1. Divided prediction interval into 10 parts.
2. Find the point at 10%, 20%, 30%, 40%....90%

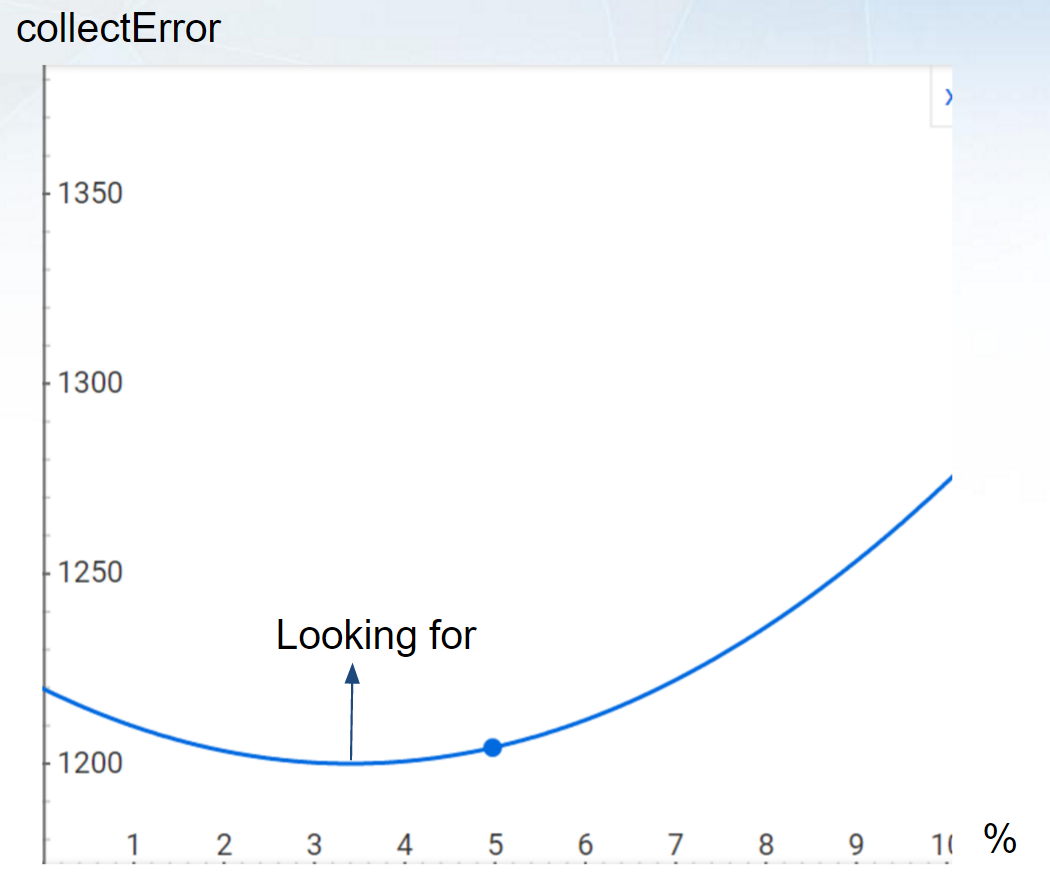


3. Collect errors when per is(10%, 20%.....): Collect Errors: collectError += abs(realValue - predictValue).

For example(month 48 -- 69 items): Total sale: 39468.

minErrors(48): 16412.

Most time: 30% can get minimum error, 40% can get second minimum error. Therefore, we guess there might be a minimum point between 30% and 40%.



Future:

1. Run more items.
2. Get prices for items.
3. Do prediction depends on prices and quantities.