Untitled

Adham Suliman

November 29, 2018

Create cmeS data with moving average filling monthw wtih no data.

```
df <- as.data.frame(read.csv('HW 7/cmeS.csv'))
(df2 <- df %>%
  group_by(Year, Month, division) %>%
  summarize(Avg_Price = mean(price)))
```

```
## # A tibble: 134 x 4
## # Groups:   Year, Month [?]
##     Year Month division Avg_Price
##    <int> <int> <fct>        <dbl>
##  1  2001     1 CME         188000
##  2  2001     2 CME         250000
##  3  2001     3 CME         250000
##  4  2001     5 CME         325000
##  5  2001     6 CME         375000
##  6  2001    11 CME         305000
##  7  2001    12 CME         360000
##  8  2002     1 CME         395000
##  9  2002     2 CME         400000
## 10  2002     4 CME         400000
## # ... with 124 more rows
```

```
df2$Date <- as.Date(strftime(strptime((paste(1, df2$Month, df2$Year)),"%d %m %Y"),"%Y-%m-%d"))
dfx <- data.frame(Month=1:156)
dfx$Date<- c(seq(as.Date("2001-01-01"), by = "month", length.out = 156))
df3 <- merge(x = dfx, y = df2, by = "Date", all.x = TRUE)
df3 <- df3[,c('Date','division','Avg_Price')]
df3$division <- "CME"
df3$Avg_Price  <- na.seadec(ts(df3$Avg_Price,frequency = 12), "ma")
df3$Date <- as.Date(df3$Date, format='%Y-%m-%d')
class(df3$Date)
```

```
## [1] "Date"
```

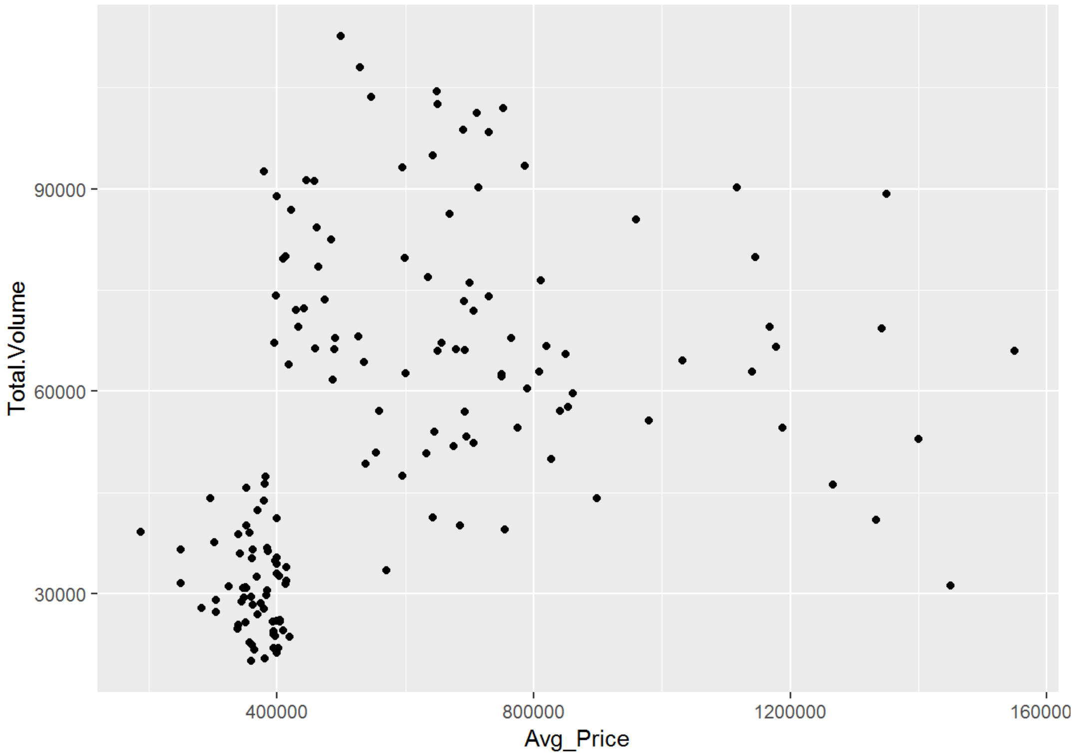
Data Wrangling

```
cont_class <- as.data.frame(read.csv('HW 7/Contracts_Classification.csv'))
cont_vol <- as.data.frame(read.csv('HW 7/Contracts_Volume.csv'))
contracts <- merge(cont_class,cont_vol,by.x="Commodity.Code",by.y="Commodity.Indicator")
contracts$Electronic.Volume[is.na(contracts$Electronic.Volume)] <- 0
contracts$Electronic.Volume <- as.integer(contracts$Electronic.Volume)
contracts$Date <- as.Date(as.character(contracts$Date), format='%m/%d/%Y')
contracts2 <-
  as.data.frame(contracts %>%
  group_by(Date,Division)%>%
  summarize(Electronic.Volume = mean(Electronic.Volume),Total.Volume = mean(Total.Volume)))
contracts2$Floor.Volume <- contracts2$Total.Volume - contracts2$Electronic.Volume
contracts2 <- contracts2[contracts2$Date >= '2001-01-01',]
CME <- inner_join(x=df3,y=contracts2[contracts2$Division=='CME',],by=('Date'))
```

Exploratory Analysis for CME: Avg price doesn't seem to be a good predictor of total volume.

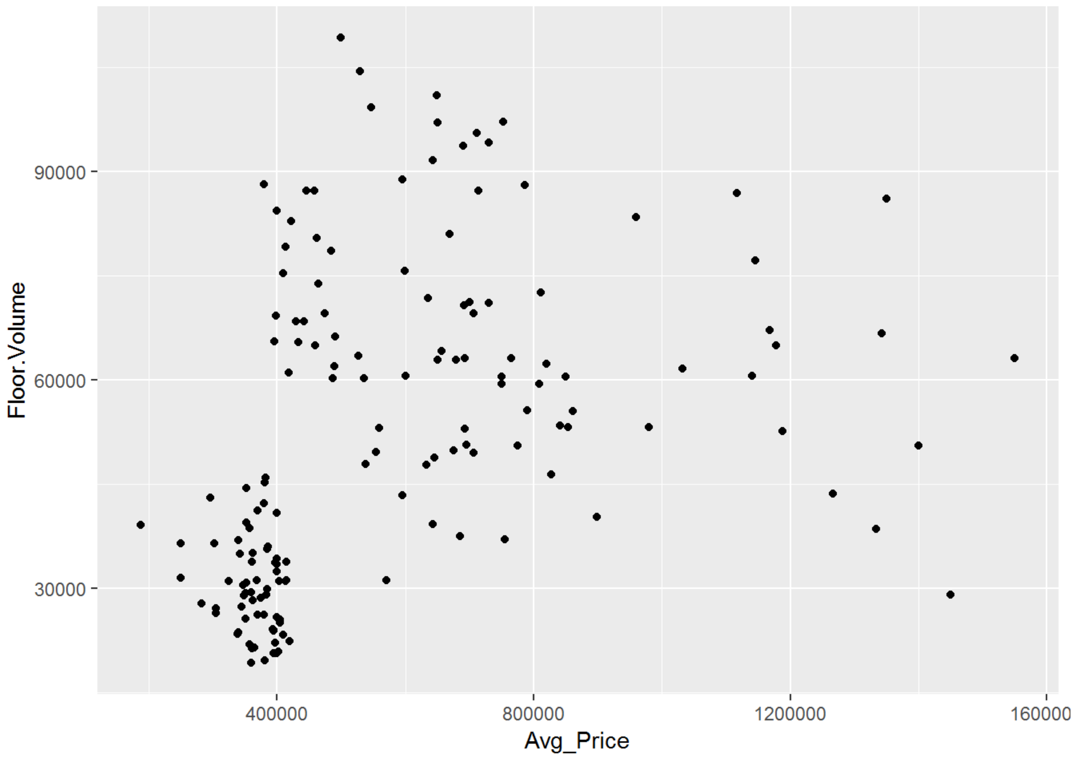
```
ggplot(CME,aes(x=Avg_Price,y=Total.Volume))+geom_point()
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```




```
ggplot(CME,aes(x=Avg_Price,y=Floor.Volume))+geom_point()
```

```
## Don't know how to automatically pick scale for object of type ts. Defaulting to continuous.
```



Split into Training and Test

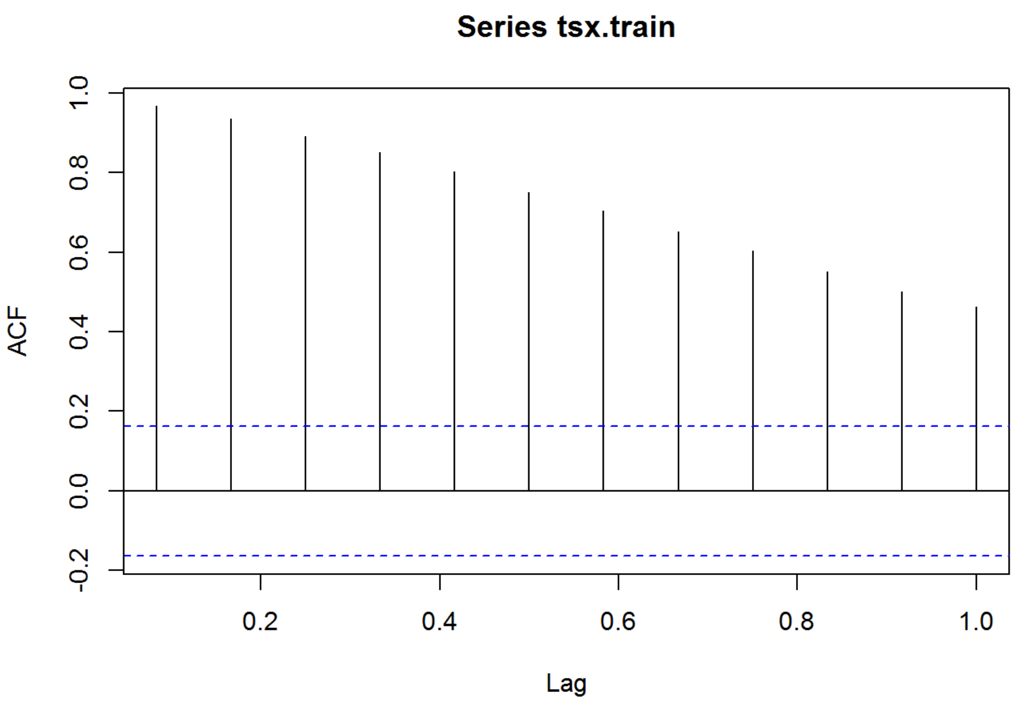
```
df.train <-   CME[CME$Date < '2013-01-01',]
df.test <- CME[CME$Date >= '2013-01-01',]
```

Exploratory Analysis continued: Since the Ljung-Box test comes back as significant when inputed with the average price for the TS, it can be stated that our resdiuals are not independent, and there is autocorrelation. The autocorrelation can also bee seen within the acf and pacf below due to the significant lags. The pacf is telling that it is the first lag and potentially the third that are causing the autocorrelation and dependence. The acf is telling that an arfima would be a good way to get rid of the autocorrelation.

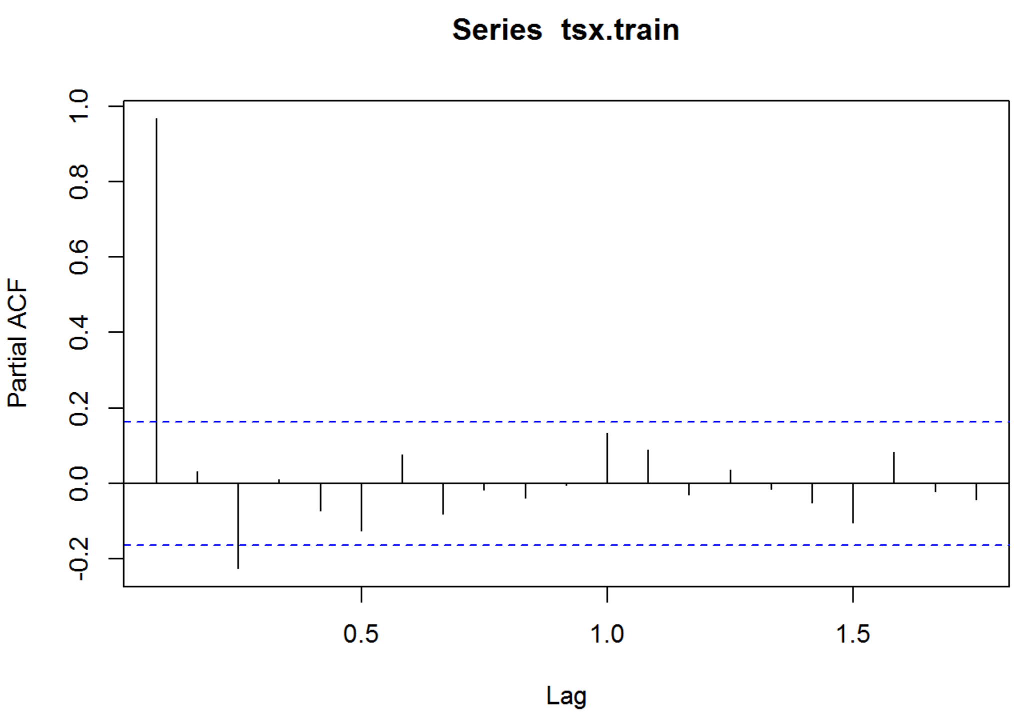
```
tsx.train <- ts(df.train$Avg_Price, frequency=12)
tsx.test <- ts(df.test$Avg_Price, frequency=12)
Box.test(tsx.train, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  tsx.train
## X-squared = 137.27, df = 1, p-value < 2.2e-16
```

```
acf(tsx.train, lag.max=12)
```




```
pacf(tsx.train)
```



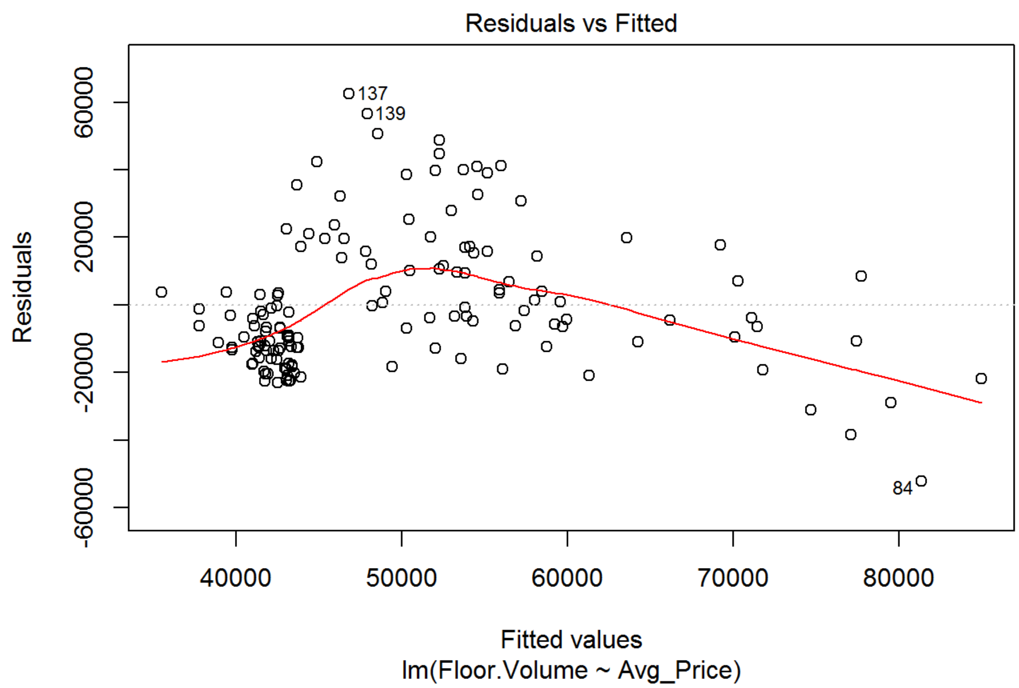
1. Pull in contracts_volume and run a Linear regression (seat price is independent, volume(s) dependent). Should we use the

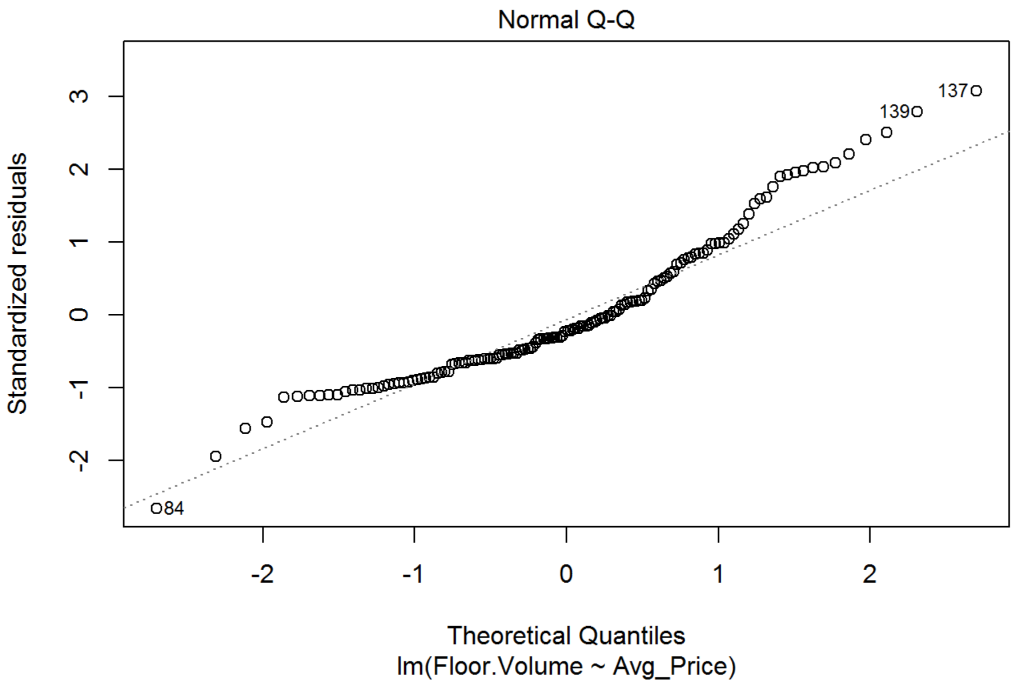
Electronic.Volume or Total.Volume? Linear Reggression is awful with an R-squared of .2. sMAPE = .455517

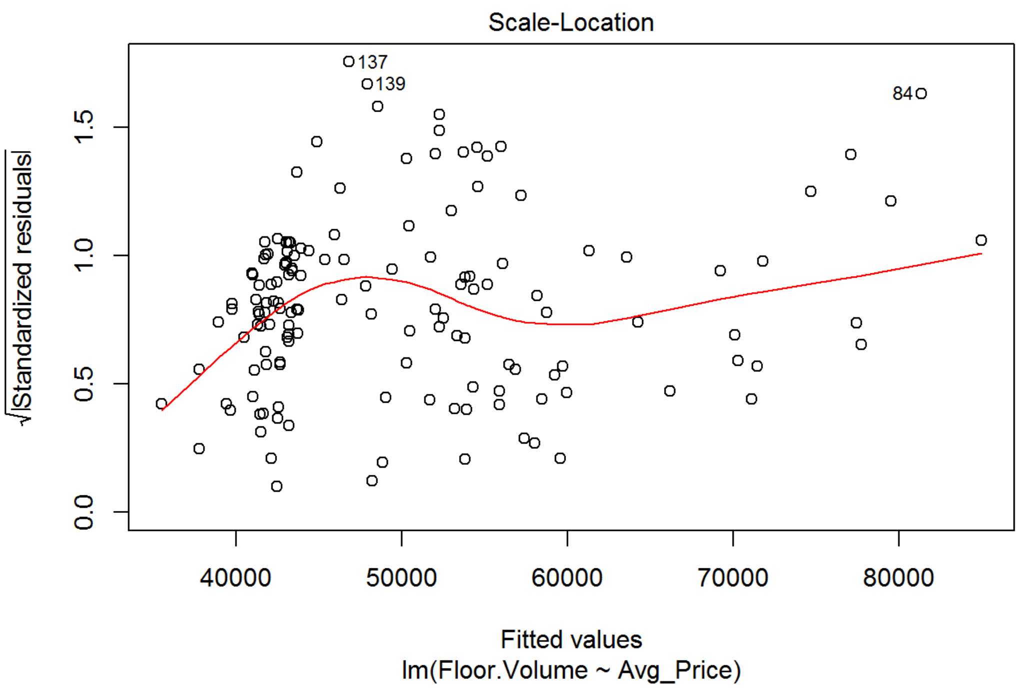
```
lm_price_by_seat <- lm(Floor.Volume ~ Avg_Price, df.train)
summary(lm_price_by_seat)
```

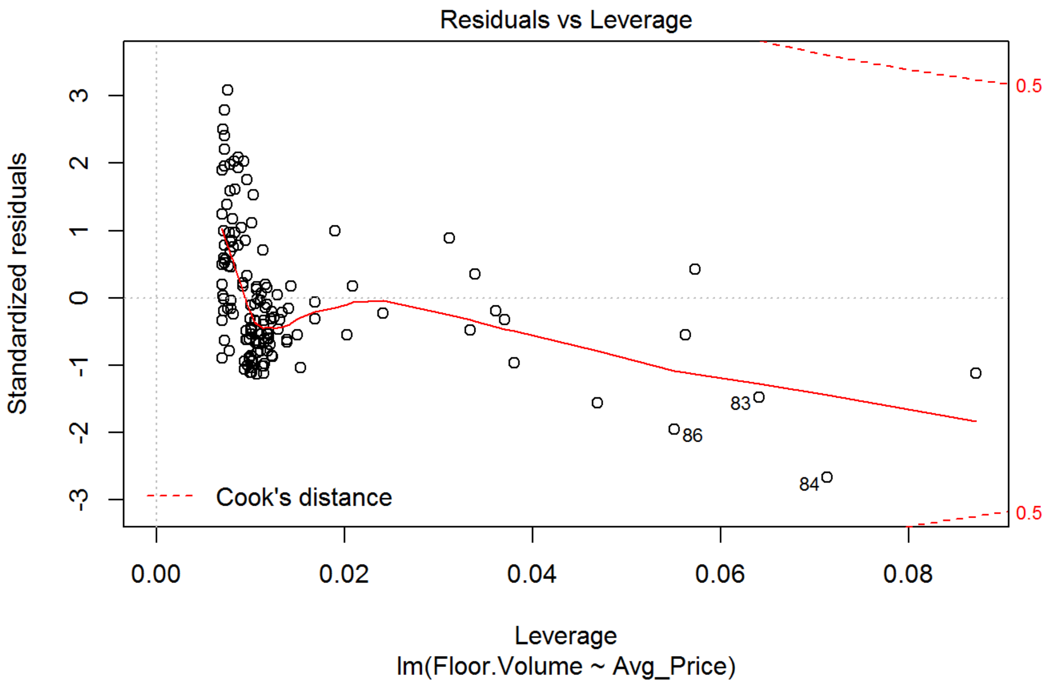
```
##
## Call:
## lm(formula = Floor.Volume ~ Avg_Price, data = df.train)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -52300 -13383  -4673  10869  62468
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 2.868e+04  3.909e+03   7.337 1.54e-11 ***
## Avg_Price   3.633e-02  5.989e-03   6.065 1.13e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20350 on 142 degrees of freedom
## Multiple R-squared:  0.2058, Adjusted R-squared:  0.2002
## F-statistic: 36.79 on 1 and 142 DF,  p-value: 1.131e-08
```

```
plot(lm_price_by_seat)
```




Normal Q-Q
Standardized residuals
Theoretical Quantiles
lm(Floor.Volume ~ Avg_Price)
137
139
84
-2
-1
0
1
2
3


Scale-Location
√|Standardized residuals|
Fitted values
lm(Floor.Volume ~ Avg_Price)
137
139
84
0.0
0.5
1.0
1.5
40000
50000
60000
70000
80000




```
x <- data.frame(Avg_Price=df.test$Avg_Price)
Metrics::smape(df.test$Floor.Volume,predict(lm_price_by_seat, newdata=x))
```

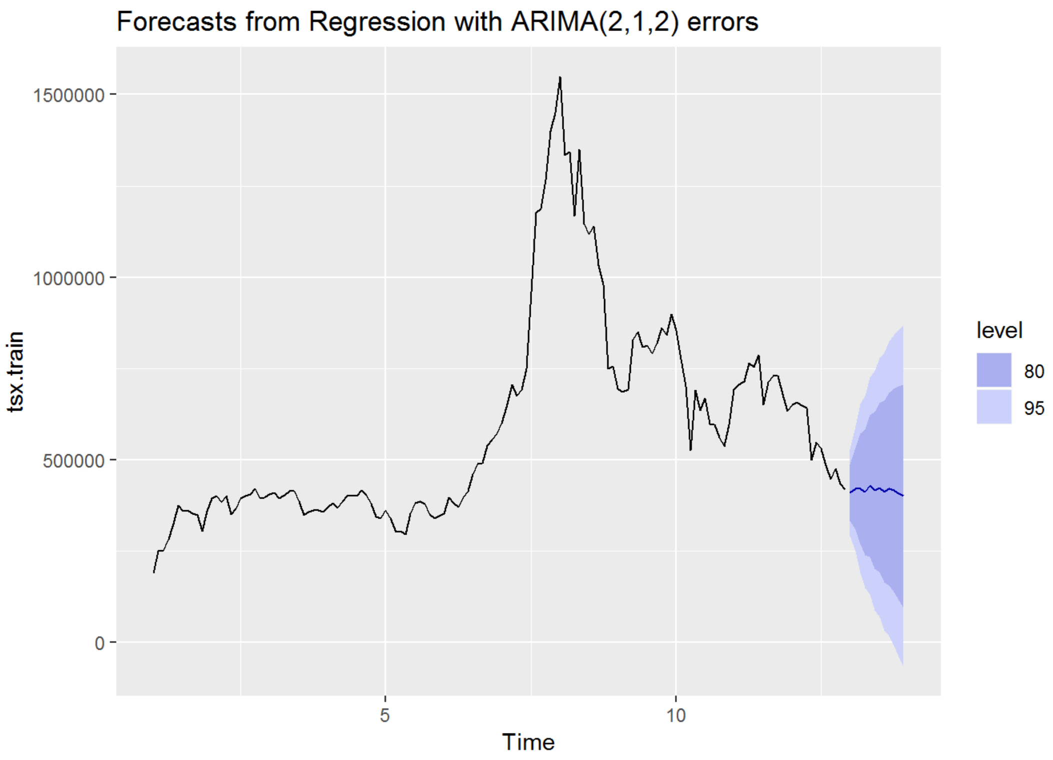
```
## [1] 0.4847074
```

2. Linear regression with ARMA errors (use arima with xreg)

```
(arima_xreg <- auto.arima(tsx.train, xreg=df.train$Floor.Volume))
```

```
## Series: tsx.train
## Regression with ARIMA(2,1,2) errors
##
## Coefficients:
##           ar1      ar2     ma1     ma2    xreg
##       -0.8995  -0.4098  0.9419  0.7042  0.6641
## s.e.   0.2260   0.1956  0.1856  0.1368  0.4234
##
## sigma^2 estimated as 3.681e+09:  log likelihood=-1775.52
## AIC=3563.03   AICc=3563.65   BIC=3580.81
```

```
arima_xreg_forecast <- forecast(arima_xreg,h=12 , xreg=(df.test$Total.Volume))
autoplot(arima_xreg_forecast)
```




```
smape(df.test$Avg_Price, arima_xreg_forecast$mean)
```

```
## [1] 0.1071452
```

3. Holt Winters

```
(HW <- HoltWinters(tsx.train))
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = tsx.train)
##
## Smoothing parameters:
##  alpha: 0.9312385
##  beta : 0
##  gamma: 1
##
## Coefficients:
##            [,1]
## a   446730.717
## b     5215.328
## s1   11253.641
## s2    8629.679
## s3    6285.122
## s4    1272.161
## s5   -3580.676
## s6  -25688.965
## s7    2363.641
## s8   26412.513
## s9    8949.894
## s10   -495.625
## s11 -49845.721
## s12 -27730.717
```

```
hw_forecast <- forecast(HW, h=12)
smape(df.test$Avg_Price, hw_forecast$mean)
```

```
## [1] 0.1271425
```

4)ARIMA

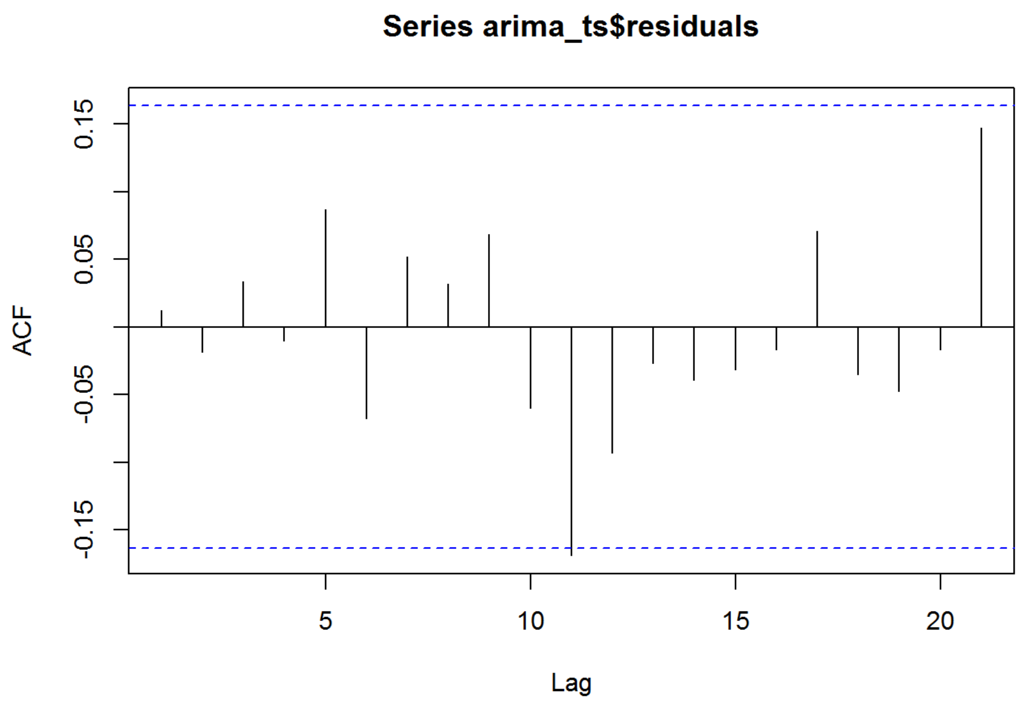
```r
(arima_ts <- auto.arima(df.train$Avg_Price))
```

```
## Series: df.train$Avg_Price
## ARIMA(0,1,4)
##
## Coefficients:
##          ma1     ma2      ma3     ma4
##       0.0078  0.3249  -0.1750  0.2501
## s.e.  0.0806  0.0803   0.0819  0.0798
##
## sigma^2 estimated as 3.646e+09:  log likelihood=-1775.37
## AIC=3560.73   AICc=3561.17   BIC=3575.55
```

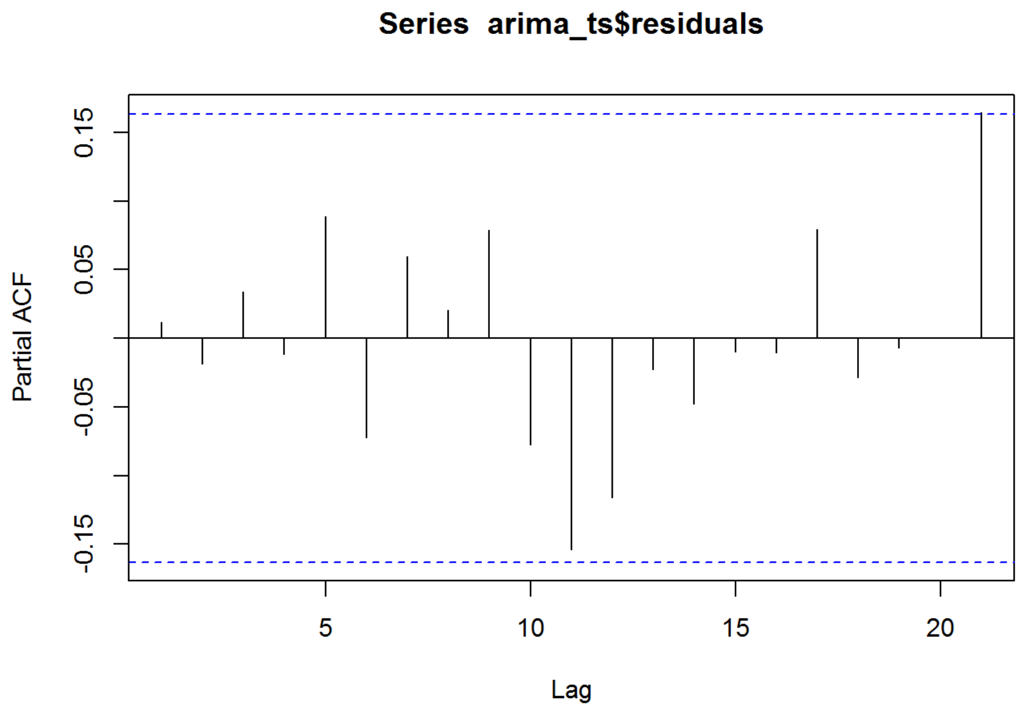
```r
Box.test(arima_ts$residuals, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  arima_ts$residuals
## X-squared = 0.020926, df = 1, p-value = 0.885
```

```r
acf(arima_ts$residuals)
```




```r
pacf(arima_ts$residuals)
```




```
smape(df.test$Avg_Price ,forecast(arima_ts, h=12)$mean)
```

```
## [1] 0.1289222
```

5. Seasonal Arima: sMAPE =

```
(seasonal_arima <- auto.arima(tsx.train, D=1))
```

```
## Series: tsx.train
## ARIMA(1,0,0)(2,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2
##       0.9709  -0.7350  -0.2809
## s.e.  0.0189   0.0845   0.0861
##
## sigma^2 estimated as 5.763e+09:  log likelihood=-1673.43
## AIC=3354.85   AICc=3355.17   BIC=3366.38
```

```
summary(seasonal_arima)
```

```
## Series: tsx.train
## ARIMA(1,0,0)(2,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2
##       0.9709  -0.7350  -0.2809
## s.e.  0.0189   0.0845   0.0861
##
## sigma^2 estimated as 5.763e+09:  log likelihood=-1673.43
## AIC=3354.85   AICc=3355.17   BIC=3366.38
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -1177.885 71853.3 46455.22 -0.5553849 6.996685 0.2389512
##                    ACF1
## Training set 0.01523917
```

```
smape(df.test$Avg_Price ,forecast(seasonal_arima, h=12)$mean)
```

```
## [1] 0.1234857
```

6)Fractional ARIMA (ARFIMA) - check applicability first using the ACF For CME, ARFIMA produces the lowest sMAPE. Therefore, it is the best model for the CME data.

```
arfima <- arfima(tsx.train)
smape(df.test$Avg_Price ,forecast(arfima, h=12)$mean)
```

```
## [1] 0.08762458
```

fgarch

```
garch_df <- diff((tsx.train))
modelspec <-
  ugarchspec( variance.model= list(model = "sGARCH", garchOrder= c(1, 1)),
  mean.model= list(armaOrder= c(1, 0), include.mean= FALSE),
  distribution.model= "norm")
model <- ugarchfit(spec=modelspec,data=garch_df)
```

```
## Warning in .makefitmodel(garchmodel = "sGARCH", f = .sgarchLLH, T = T, m = m, :
## rugarch-->warning: failed to invert hessian
```

```
garch_forecast <- ugarchforecast(model, n.ahead=12)
smape(tsx.test,garch_forecast@forecast$sigmaFor)
```

```
## [1] 1.620519
```

Conduct the same analysis done on cmeS as on iomS and immS

Create cmeS data with moving average filling monthw wtih no data.

```
df.iom <- as.data.frame(read.csv('HW 7/iomS.csv'))
(df.iom2 <- df.iom %>%
  group_by(Year, Month, division) %>%
  summarize(Avg_Price = mean(price)))
```

```
## # A tibble: 147 x 4
## # Groups:   Year, Month [?]
##    Year    Month   division Avg_Price
##    <fct>   <fct>   <fct>        <dbl>
##  1 #VALUE! #VALUE! IOM         356000
##  2 2001    1       IOM         130000
##  3 2001    10      IOM         231250
##  4 2001    11      IOM         243000
##  5 2001    12      IOM         246000
##  6 2001    2       IOM         170000
##  7 2001    3       IOM         242500
##  8 2001    4       IOM         291500
##  9 2001    6       IOM         260000
## 10 2001    7       IOM         260000
## # ... with 137 more rows
```

```
df.iom2$Date <- as.Date(strftime(strptime((paste(1, df.iom2$Month, df.iom2$Year)),"%d %m %Y"),"%Y-%m-%d"))
df.iom3 <- merge(x = dfx, y = df.iom2, by = "Date", all.x = TRUE)
df.iom3 <- df.iom3[,c('Date','division','Avg_Price')]
df.iom3$division <- "IOM"
df.iom3$Avg_Price  <- na.seadec(ts(df.iom3$Avg_Price,frequency = 12), "ma")
df.iom3$Date <- as.Date(df.iom3$Date, format='%Y-%m-%d')
IOM <- inner_join(x=df.iom3,y=contracts2[contracts2$Division=='IOM',],by=('Date'))
head(IOM)
```

```
##         Date division Avg_Price Division Electronic.Volume Total.Volume
## 1 2001-01-01      IOM  130000.0      IOM         1293.0000     294590.4
## 2 2001-02-01      IOM  170000.0      IOM         1044.6400     264763.4
## 3 2001-03-01      IOM  242500.0      IOM          918.6729     318127.1
## 4 2001-04-01      IOM  291500.0      IOM         1335.3030     319554.4
## 5 2001-05-01      IOM  252240.2      IOM         1483.1402     309745.1
## 6 2001-06-01      IOM  260000.0      IOM         1092.6768     334241.7
##   Floor.Volume
## 1     293297.4
## 2     263718.8
## 3     317208.4
## 4     318219.1
## 5     308262.0
## 6     333149.1
```

Create immS data with moving average filling monthw wtih no data.

```
df.imm <- as.data.frame(read.csv('HW 7/immS.csv'))
(df.imm2 <- df.imm %>%
  group_by(Year, Month, division) %>%
  summarize(Avg_Price = mean(price)))
```

```
## # A tibble: 146 x 4
## # Groups:   Year, Month [?]
##      Year Month division Avg_Price
##     <int> <int> <fct>        <dbl>
##  1  2001     1 IMM         183125
##  2  2001     2 IMM         225000
##  3  2001     3 IMM         292500
##  4  2001     5 IMM         305000
##  5  2001     6 IMM         355333.
##  6  2001     7 IMM         355000
##  7  2001     9 IMM         338333.
##  8  2001    10 IMM         316667.
##  9  2001    11 IMM         318333.
## 10  2001    12 IMM         344833.
## # ... with 136 more rows
```

```
df.imm2$Date <- as.Date(strftime(strptime((paste(1, df.imm2$Month, df.imm2$Year)),"%d %m %Y"),"%Y-%m-%d"))
df.imm3 <- merge(x = dfx, y = df.imm2, by = "Date", all.x = TRUE)
df.imm3 <- df.imm3[,c('Date','division','Avg_Price')]
df.imm3$division <- "IMM"
df.imm3$Avg_Price  <- na.seadec(ts(df.imm3$Avg_Price,frequency = 12), "ma")
df.imm3$Date <- as.Date(df.imm3$Date, format='%Y-%m-%d')
IMM <- inner_join(x=df.imm3,y=contracts2[contracts2$Division=='IMM',],by=('Date'))
head(IMM)
```

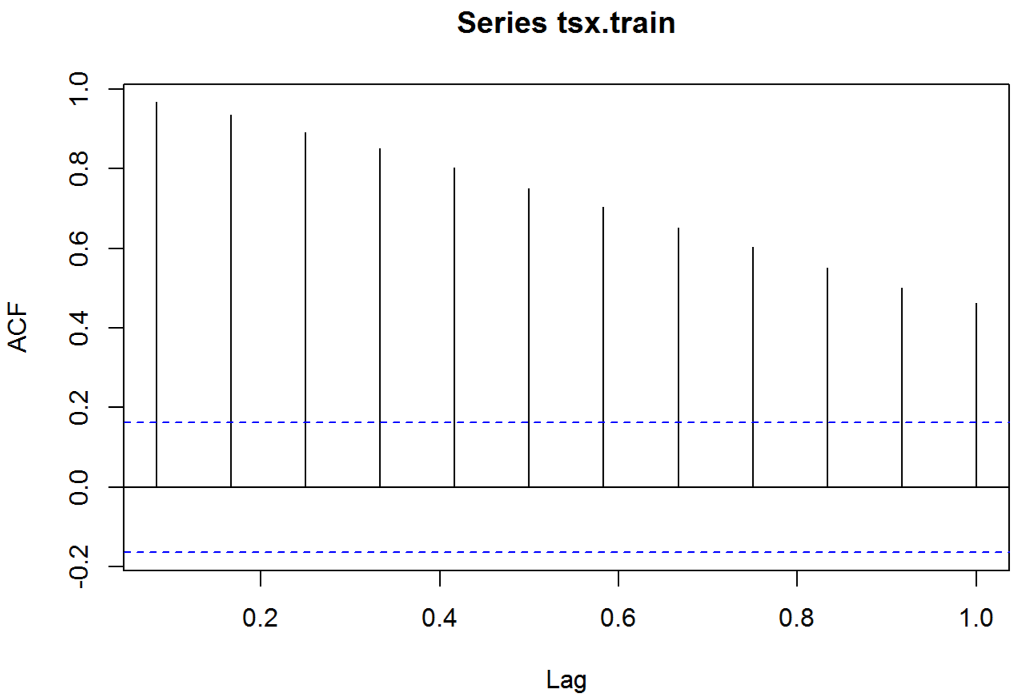
```
##         Date division Avg_Price Division Electronic.Volume Total.Volume
## 1 2001-01-01      IMM  183125.0      IMM          2499.545     701233.6
## 2 2001-02-01      IMM  225000.0      IMM          2091.125     565389.0
## 3 2001-03-01      IMM  292500.0      IMM          1901.424     674786.9
## 4 2001-04-01      IMM  286829.1      IMM          2719.424     643694.5
## 5 2001-05-01      IMM  305000.0      IMM          3024.030     685575.0
## 6 2001-06-01      IMM  355333.3      IMM          2597.344     685384.9
##   Floor.Volume
## 1     698734.0
## 2     563297.9
## 3     672885.5
## 4     640975.0
## 5     682551.0
## 6     682787.6
```

Create train and test TS for imm and iom

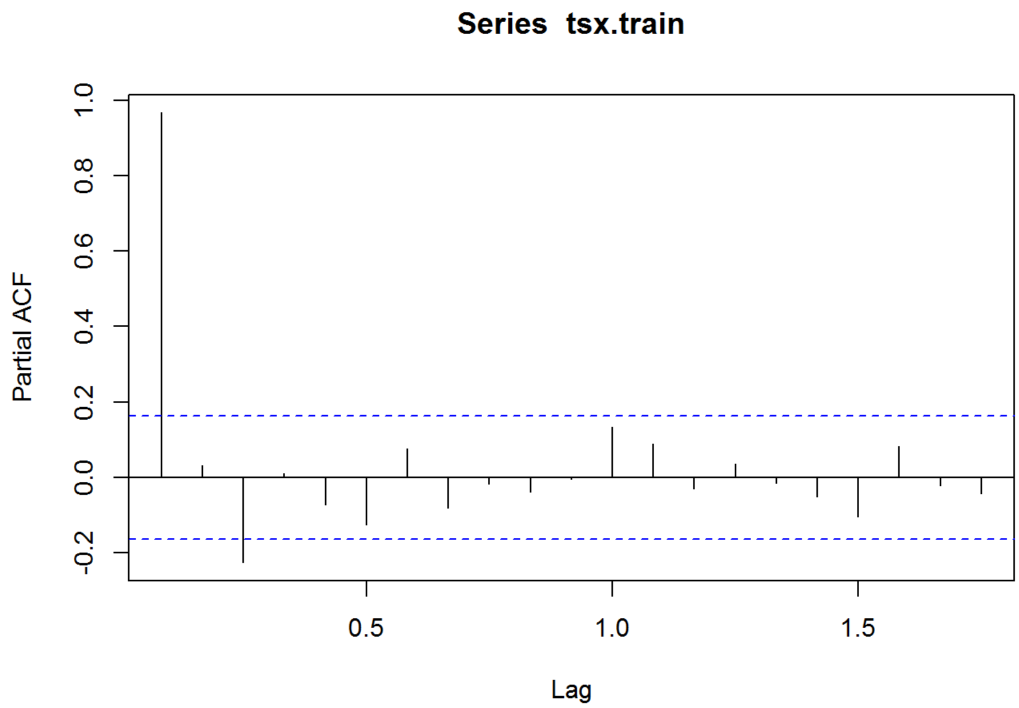
```
df.train.imm <-   IMM[IMM$Date < '2013-01-01',]
df.test.imm <- IMM[IMM$Date >= '2013-01-01',]
df.train.iom <-   IOM[IOM$Date < '2013-01-01',]
df.test.iom <- IOM[IOM$Date >= '2013-01-01',]

tsx.train.imm <- ts(df.train.imm$Avg_Price, frequency=12)
tsx.train.iom <- ts(df.train.iom$Avg_Price, frequency=12)

tsx.test.imm <- ts(df.test.imm$Avg_Price, frequency=12)
tsx.test.iom <- ts(df.test.iom$Avg_Price, frequency=12)

Box.test(tsx.train, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  tsx.train
## X-squared = 137.27, df = 1, p-value < 2.2e-16
```

```
acf(tsx.train, lag.max=12)
```




```
pacf(tsx.train)
```



The lm for IMM produces an R^2 of .4 while the lm of IOM produces an R^2 of .00873 which tells the imm floor volume is a better regessor than IOM's. IOM seems to have a lower sMAPE than IMM's forecast The lm produces the lowest sMAPE for IOM. Therefore it is the best model for the IOM data.

```
lm_imm <- lm(Floor.Volume ~ Avg_Price, df.train.imm)
lm_iom <- lm(Floor.Volume ~ Avg_Price, df.train.iom)

summary(lm_imm)
```

```
##
## Call:
## lm(formula = Floor.Volume ~ Avg_Price, data = df.train.imm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -626970 -233788  -37848  181624  873729
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 3.316e+05  7.571e+04   4.379  2.3e-05 ***
## Avg_Price   1.843e+00  1.737e-01  10.610  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 311400 on 142 degrees of freedom
## Multiple R-squared:  0.4422, Adjusted R-squared:  0.4383
## F-statistic: 112.6 on 1 and 142 DF,  p-value: < 2.2e-16
```

```
summary(lm_iom)
```

```
##
## Call:
## lm(formula = Floor.Volume ~ Avg_Price, data = df.train.iom)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -281742 -115197  -12384   98168  511563
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 4.892e+05  3.125e+04  15.655  < 2e-16 ***
## Avg_Price   3.312e-01  1.245e-01   2.659  0.00873 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 161000 on 142 degrees of freedom
## Multiple R-squared:  0.04744,    Adjusted R-squared:  0.04073
## F-statistic: 7.072 on 1 and 142 DF,  p-value: 0.008728
```

```
x <-x <- data.frame(Avg_Price=df.test$Avg_Price)
Metrics::smape(df.test$Floor.Volume,predict(lm_price_by_seat, newdata=x))
```

```
## [1] 0.4847074
```

```
Metrics::smape(df.test$Floor.Volume,predict(lm_price_by_seat, newdata=x))
```

```
## [1] 0.4847074
```

```
lm_imm_smape <- smape(df.test.imm$Floor.Volume,predict(lm_imm, newdata=data.frame(Avg_Price=df.test.iom$Avg_
Price)))
lm_iom_smape <- smape(df.test.iom$Floor.Volume,predict(lm_iom, newdata=data.frame(Avg_Price=df.test.imm$Avg_
Price)))
cbind(lm_imm_smape,lm_iom_smape)
```

```
##      lm_imm_smape lm_iom_smape
## [1,]    0.7131311    0.1178354
```

2. Linear regression with ARMA errors (use arima with xreg)

```
(arima_xreg.imm <- auto.arima(tsx.train.imm, xreg=df.train.imm$Floor.Volume))
```

```
## Series: tsx.train.imm
## Regression with ARIMA(1,0,1) errors
##
## Coefficients:
##          ar1     ma1  intercept    xreg
##       0.9525  0.3626  345799.43  0.0109
## s.e.  0.0261  0.0798   79586.54  0.0094
##
## sigma^2 estimated as 1.316e+09:  log likelihood=-1715.7
## AIC=3441.39   AICc=3441.83   BIC=3456.24
```

```
(arima_xreg.iom <- auto.arima(tsx.train.iom, xreg=df.train.iom$Floor.Volume))
```

```
## Series: tsx.train.iom
## Regression with ARIMA(0,1,1)(0,0,1)[12] errors
##
## Coefficients:
##          ma1     sma1     xreg
##       0.2458  -0.1369  -0.0088
## s.e.  0.0833   0.0863   0.0151
##
## sigma^2 estimated as 725718795:  log likelihood=-1660.33
## AIC=3328.66   AICc=3328.95   BIC=3340.51
```

```
arima_xreg_forecast_imm <- forecast(arima_xreg.imm,h=12 , xreg=(df.train.imm$Total.Volume))
arima_xreg_forecast_iom <- forecast(arima_xreg.iom,h=12 , xreg=(df.train.iom$Total.Volume))

imm_xreg_smape <- smape(df.train.imm$Avg_Price, arima_xreg_forecast_imm$mean)
iom_xreg_smape <- smape(df.train.iom$Avg_Price, arima_xreg_forecast_iom$mean)
cbind(imm_xreg_smape,iom_xreg_smape)
```

```
##      imm_xreg_smape iom_xreg_smape
## [1,]      0.2585058      0.9501456
```

3. Holt Winters

```
(HW_imm <- HoltWinters(tsx.train.imm))
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = tsx.train.imm)
##
## Smoothing parameters:
##  alpha: 0.9455428
##  beta : 0.008629903
##  gamma: 1
##
## Coefficients:
##            [,1]
## a   196237.5205
## b     -463.0422
## s1   -7822.4771
## s2    8944.8890
## s3    5654.1038
## s4    2706.8794
## s5   -2782.1351
## s6   -1563.4830
## s7   23028.9859
## s8   18975.7479
## s9    6613.7936
## s10 -19140.5732
## s11 -36718.0022
## s12 -26237.5205
```

```
(HW_iom <- HoltWinters(tsx.train.iom))
```

```
## Holt-Winters exponential smoothing with trend and additive seasonal component.
##
## Call:
## HoltWinters(x = tsx.train.iom)
##
## Smoothing parameters:
##  alpha: 0.9409175
##  beta : 0.008212475
##  gamma: 1
##
## Coefficients:
##            [,1]
## a    76503.4594
## b     -352.6329
## s1   -4745.5725
## s2    7522.6161
## s3    9456.6935
## s4    6798.4131
## s5    2339.1420
## s6   -2488.9796
## s7    7875.0189
## s8    9064.0109
## s9   -8563.7402
## s10 -19848.9352
## s11 -17688.8944
## s12 -12378.4594
```

```
hw_forecast_imm <- forecast(HW_imm, h=12)
hw_forecast_iom <- forecast(HW_iom, h=12)

HW_imm_smape <- smape(df.test.imm$Avg_Price,hw_forecast_imm$mean)
HW_iom_smape <- smape(df.test.iom$Avg_Price,hw_forecast_iom$mean)
cbind(HW_imm_smape,HW_iom_smape)
```

```
##      HW_imm_smape HW_iom_smape
## [1,]    0.1119201    0.2575716
```

4)ARIMA Both box test return the null hypothesis which states no autocorrelation. Auto.arima returns the lowest sMAPE for IMM. Therefore, it is the best model for the IMM data set.

```
(arima_imm <- auto.arima(df.train.imm$Avg_Price))
```

```
## Series: df.train.imm$Avg_Price
## ARIMA(1,1,0)
##
## Coefficients:
##          ar1
##       0.3594
## s.e.  0.0780
##
## sigma^2 estimated as 1.328e+09:  log likelihood=-1704.45
## AIC=3412.91   AICc=3412.99   BIC=3418.83
```

```
(arima_iom <- auto.arima(df.train.iom$Avg_Price))
```

```
## Series: df.train.iom$Avg_Price
## ARIMA(0,1,1)
##
## Coefficients:
##          ma1
##       0.2424
## s.e.  0.0821
##
## sigma^2 estimated as 729839811:  log likelihood=-1661.63
## AIC=3327.27   AICc=3327.35   BIC=3333.19
```

```
Box.test(arima_imm$residuals, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  arima_imm$residuals
## X-squared = 0.039796, df = 1, p-value = 0.8419
```

```
Box.test(arima_iom$residuals, type = c("Ljung-Box"))
```

```
##
##  Box-Ljung test
##
## data:  arima_iom$residuals
## X-squared = 0.0070409, df = 1, p-value = 0.9331
```

```
smape(df.test$Avg_Price ,forecast(arima_imm, h=12)$mean)
```

```
## [1] 0.9279901
```

```
smape(df.test$Avg_Price ,forecast(arima_iom, h=12)$mean)
```

```
## [1] 1.497419
```

5. Seasonal Arima: sMAPE =

```
(seasonal_arima.imm <- auto.arima(tsx.train.imm, D=1))
```

```
## Series: tsx.train.imm
## ARIMA(1,1,0)(2,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2
##       0.3539  -0.5642  -0.2462
## s.e.  0.0824   0.0845   0.0840
##
## sigma^2 estimated as 1.947e+09:  log likelihood=-1587.56
## AIC=3183.12   AICc=3183.44   BIC=3194.62
```

```
(seasonal_arima.iom <- auto.arima(tsx.train.iom, D=1))
```

```
## Series: tsx.train.iom
## ARIMA(2,0,0)(2,1,0)[12]
##
## Coefficients:
##          ar1      ar2     sar1     sar2
##       1.2056  -0.2409  -0.6947  -0.2596
## s.e.  0.0849   0.0849   0.0845   0.0830
##
## sigma^2 estimated as 1.064e+09:  log likelihood=-1561.23
## AIC=3132.46   AICc=3132.94   BIC=3146.87
```

```
summary(seasonal_arima.imm)
```

```
## Series: tsx.train.imm
## ARIMA(1,1,0)(2,1,0)[12]
##
## Coefficients:
##          ar1     sar1     sar2
##       0.3539  -0.5642  -0.2462
## s.e.  0.0824   0.0845   0.0840
##
## sigma^2 estimated as 1.947e+09:  log likelihood=-1587.56
## AIC=3183.12   AICc=3183.44   BIC=3194.62
##
## Training set error measures:
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -2263.718 41597.33 29760.49 -0.6944528 7.293641 0.2532689
##                     ACF1
## Training set 0.008373249
```

```
summary(seasonal_arima.iom)
```

```
## Series: tsx.train.iom
## ARIMA(2,0,0)(2,1,0)[12]
##
## Coefficients:
##          ar1      ar2     sar1     sar2
##       1.2056  -0.2409  -0.6947  -0.2596
## s.e.  0.0849   0.0849   0.0845   0.0830
##
## sigma^2 estimated as 1.064e+09:  log likelihood=-1561.23
## AIC=3132.46   AICc=3132.94   BIC=3146.87
##
## Training set error measures:
##                     ME     RMSE      MAE       MPE     MAPE      MASE
## Training set -1408.379 30754.26 21321.24 -1.325775 9.853546 0.2456216
##                     ACF1
## Training set 0.006125391
```

```
smape(df.test.imm$Avg_Price ,forecast(seasonal_arima.imm, h=12)$mean)
```

```
## [1] 0.7484162
```

```
smape(df.test.iom$Avg_Price ,forecast(seasonal_arima.iom, h=12)$mean)
```

```
## [1] 0.1558746
```

6)Fractional ARIMA (ARFIMA) - check applicability first using the ACF

```
arfima.imm <- arfima(tsx.train.imm)
arfima.iom <- arfima(tsx.train.iom)

arfima.imm.smape <- smape(df.test.imm$Avg_Price ,forecast(arfima.imm, h=12)$mean)
arfima.iom.smape <- smape(df.test.iom$Avg_Price ,forecast(arfima.iom, h=12)$mean)
cbind(arfima.imm.smape,arfima.iom.smape )
```

```
##      arfima.imm.smape arfima.iom.smape
## [1,]        0.1279411        0.5151476
```

sgarch for imm

```
garch_df.imm <- diff((tsx.train.imm))
model.imm <- ugarchfit(spec=modelspec,data=garch_df.imm)
```

```
## Warning in .makefitmodel(garchmodel = "sGARCH", f = .sgarchLLH, T = T, m = m, :
## rugarch-->warning: failed to invert hessian
```

```
garch_forecast.imm <- ugarchforecast(model.imm, n.ahead=12)
smape(tsx.test.imm,garch_forecast.imm@forecast$sigmaFor)
```

```
## [1] 1.631434
```

sgarch for iom

```
garch_df.iom <- diff((tsx.train.iom))
model.iom <- ugarchfit(spec=modelspec,data=garch_df.iom)
```

```
## Warning in .makefitmodel(garchmodel = "sGARCH", f = .sgarchLLH, T = T, m = m, :
## rugarch-->warning: failed to invert hessian
```

```
garch_forecast.iom <- ugarchforecast(model.iom, n.ahead=12)
smape(tsx.test.iom,garch_forecast.iom@forecast$sigmaFor)
```

```
## [1] 1.602248
```