```r
require(tseries)
require(forecast)
require(MASS)
nbins = 32


# ##################################################################################
HIST = function( x, ...) {
    hist(x, freq=FALSE, ...)
    f.den <- function(t) dnorm(t, mean=mean(x,na.rm=TRUE), sd=sd(x,na.rm=TRUE))
    curve(f.den, add=TRUE, col="darkblue", lwd=2)
}
# ##################################################################################


# ##################################################################################
# preprocess the transactions in sql to generate various time/season factor attributes
# such as week number, quarter number, etc. using extract( x from_ts)
# ##################################################################################
transactions = read.csv('DATA/VERIZON_TRANSACTIONS_EXTENDED_JOINED.csv')
transactions[,'transaction_count'] = ts(transactions[,'transaction_count'])
for (col in c('transaction_year', 'transaction_quarter', 'transaction_doy'
             #'transaction_month', 'transaction_week', 'transaction_dow',
               )) {
    transactions[,col] = as.factor(transactions[,col])
}
summary(transactions)


# ##############################################################
# exploratory visualization
# ##############################################################
graphics.off()
par(mfrow=c(3, 1))
plot.ts(transactions$transaction_count)
plot.ts(log(transactions$transaction_count))
plot.ts(diff(log(transactions$transaction_count,) -1))


# ##############################################################
# exploratory modeling
# ##############################################################
p0 = lm(log(transaction_count) ~
        transaction_daynum + transaction_week + transaction_month, data = transactions
)
p1 = ets(log(transactions$transaction_count))
p2 = auto.arima(log(transactions$transaction_count), d=1,
                seasonal=TRUE, max.order=31,
                trace=TRUE, approximation=FALSE) #stepwise=FALSE,
summary(p0)
summary(p1)
summary(p2)


# ##############################################################
# basic comparative inspection of models w/o anova
# ##############################################################
y = log(transactions[,"transaction_count"])
minlim = -max(y)/20
maxlim =  max(y)/20
graphics.off()
par(mfrow=c(3,2))
acf(residuals(p0), lag.max=90, main='ACF: lm(daynum, week, month)')
HIST(residuals(p0),  breaks=nbins, xlim=c(minlim, maxlim))
acf(residuals(p1), lag.max=90, main='ACF: ewma(ANN)')
HIST(residuals(p1),  breaks=nbins, xlim=c(minlim, maxlim))
acf(residuals(p2), lag.max=90, main='ACF: auto.arima(3,1,2)')
HIST(residuals(p2),  breaks=nbins, xlim=c(minlim, maxlim))
```

```
# ###############################################################
# the transaction total is an interesting time series:
#       requiring log due to multiplicative trend
#       then, requiring differentiation due to reach stationary
#       then, exhibiting monthly seasonality
#       thenm exhibiting gaps during month
#       then, exhibiting irregularities (some days without activity)
#       then, exhibiting short range dependencies (at beginning of month, one week on
previous month)
#       then, exhibiting long -range dependencies (at end of month, 3 weeks before wit
hin same month)
# ###############################################################





# ###############################################################
# first, gaps in data are interpolated, created in python using script
# reload a transformed dataset which inserts on the gaps interpolated values
# using the four adjacent points to the data item x: [-2, x, +2]
# ###############################################################
transactions = read.csv('DATA/VERIZON_INTERPOLATED.csv', sep=' ')
colnames(transactions) = c("rownumber", 'orig_transaction_count', 'transaction_date',
                           'orig_daynum', 'transaction_count', 'transaction_daynum',
                           'transaction_origin' )
x = as.ts(transactions$transaction_count, frequency=31)
x_log = log(x)


# ###############################################################
# since long-term dependencies present, find out significant lag effects with arma
# ###############################################################
lag_findings = arma(diff(x_log), lag=list(ar=c(1,9,14,22,23,30), ma=c(1,3,16,24)), inc
lude.intercept=FALSE)
summary(lag_findings)


# ###############################################################
# using knowledge about significant lags and trial/error after,
# build ARIMA to implicit differentiation and address monthly
# seasonality, using a seasonal autoregressive and seasonal
# moving average
# ###############################################################
p3 = Arima( x_log, order=c(30,1,3), seasonal=list(order=c(1,0,1), period=12),
            fixed =c( NA,     NA,     0,     NA,    NA,      0,      0,       0,
                      NA,     NA,     0,     NA,    NA,      NA,     NA,      NA,
                      0,      0,      0,     0,     0,       NA,     NA,      0,
                      0,      0,      0,     0,     0,       NA,
                      NA,     NA,     NA,
                      NA,     NA
                      ))
summary(p3)


# ###############################################################
# comparative residual analysis between approaches
# ########################################################### (i
graphics.off()
par(mfrow=c(4,2))
acf(residuals(p0), lag.max=90, main="lm(year, month, daynum)")
HIST(residuals(p0),  breaks=nbins, xlim=c(minlim, maxlim))
acf(residuals(p1), lag.max=90, main="ewma(ANN)")
HIST(residuals(p1),  breaks=nbins, xlim=c(minlim, maxlim))
acf(residuals(p2), lag.max=90, main="auto.arima(3.1.2)")
HIST(residuals(p2),  breaks=nbins, xlim=c(minlim, maxlim))
acf(residuals(p3), lag.max=90, main="seasonal.arima(30,1,3) (1,0,1)[12] sign.lags")
```

```r
HIST(residuals(p3),  breaks=nbins, xlim=c(minlim, maxlim))


# ##############################################################
# basic goodness of fit
# ##############################################################
print(summary(p3))
Box.test(residuals(p3), type="Ljung")
Box.test (residuals(p3), lag = 1, type = "Ljung")
accuracy(p3)
tsdiag(p3)


# ##############################################################
# visualization summary essay of the findings
# ##############################################################
nf <- layout(matrix(c(1,1,1,3, 2,2,2,3, 4,4,5,5, 6,6,6,6), 4,4, byrow=TRUE), TRUE)
layout.show(nf)
plot.ts(diff(x_log),           main='T1: input signal: diff(log(transaction_count))')
plot.ts(scale(residuals(p3)),  main='T2: std.residuals(fitted_arima_model)')
HIST(residuals(p3), breaks=32, main="T3: histogram of fitted_arima_model residuals")
acf(residuals(p3), lag.max=90, main="T4: acf of arima residuals")
pacf(residuals(p3), lag.max=90,main="T5: pacf of arima residuals")
plot(forecast(p3, h=31),       main="T6: input signal along with forecast values")


# ##############################################################
# computation of actual predicted/forecast values
# ##############################################################
march_data = forecast(p3, h=31)
march_dates = seq(as.Date("2015/3/1"), as.Date("2015/3/31"), "days")
xcount_vals =  exp(as.data.frame(march_data)[,1])
predicted_vals = cbind( as.data.frame(march_data), xcount_vals, as.data.frame(march_da
tes))
march_forecasts = predicted_vals[ -c(5, 11, 14, 17, 24, 27, 29, 30, 31), ]
print ( march_forecasts )
```