Projecting Early 2021 Stock Prices for Lots-of-stuff Inc.

Najiyullah Sanee

February 10, 2021

1 Executive Summary

After a quick recovery from events of early 2020 that caused market-wide volatility, Lots-of-stuff Incorporated (LOSI) stock prices are back on track continuing an increasing trend as in the past 4 years (mostly). With global adjustments to the 2020 pandemic and a gradual return to normalcy, there is no concerns of another sudden drop for projections of early 2021 prices. The best predictive model attained was first (daily trading day) differencing with autoregressive moving average ARMA(1,1) noise. The model predicts the next 10 day closing prices will grow slowly from 96.84 to 97.24 by day 10, and has uncertainty on those forecasts that grows from ± 1.64 to ± 4.32 by day 10 for a 95% prediction interval.

2 Exploratory Data Analysis

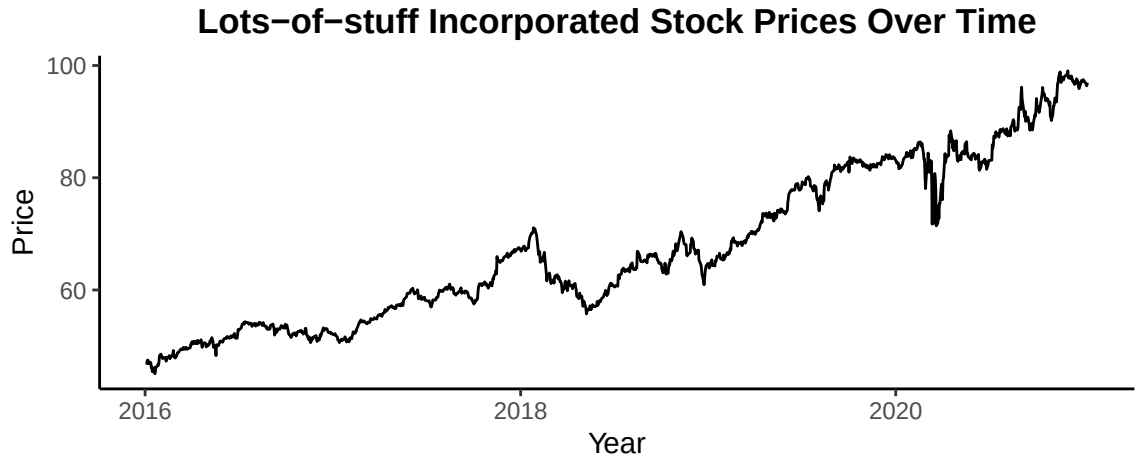

Figure 1 : LOSI trading days closing stock prices tracked from January 2016 to Jan 2021.

The closing prices time series in *Figure 1* shows an overall increasing trend annually, but there is a notable downturn in 2018 and a sudden drop of prices in 2020. The downturn in 2018 specifically slowed the increasing trend of prices, but the volatility of prices in 2020 is which occurs in March is more of a concern. The volatility did not seem to affect the overall trend but it will result in a lot of noise that could affect our models. Lastly, it is quite hard to see the presence of any seasonality patterns in the trend and periodogram of the prices indicates only a strong trend, so we have no reason to consider seasonal effects in our modeling.

3 Modeling Approaches

The most apparent observation from the EDA is the mostly consistent increasing trend either liner or quadratic. A filtering approach such as differencing can remove the entire trend so we can model remaining noise as needed. Alternatively, an estimating filter can be used to model the trend and obtain ideally stable residuals.

3.1 Model I - Differencing

A simple first differencing of the prices eliminates all trend but has a unstable variance for early parts of 2020 due to volatility. The mean looks to be constant and the variance ignoring the volatile period is also constant so we can consider it stationary nonetheless. The autocorrelation function (ACF) and partial autocorrelation function (PACF) of the differenced prices show significant autocorrelations at lags $1, 2, 7-10$ as seen in *Figure 2* below.

3.1.1 First Difference with ARMA(1,1)

Since the dominant value for both ACF and PACF is at lag 1, autogressive moving average (ARMA) with autoregressive (AR) order of at least 1 were checked for goodness of fit but none were much better than the initial $ARMA(1,1)$. The function $auto.arima()$ also suggested $ARMA(1,1)$ whose estimates are reflected by the red circles on the ACF/PACF plots in *Figure 2* but seems to only fit with autocorrelations for the first two lags. To confirm it's goodness of fit, we can use the Ljung-Box-Pierce test graphically represented in the left panel of *Figure 3* below. It uses a null hypothesis that a specified model is a good fit for the noise, so if the model choice is good it should have p-values that are above the zero-line for most short term lags. For $ARMA(1,1)$, the results show insignificant p-values up to lag 6, and that is acceptable for short term autocorrelation we'd like to model.

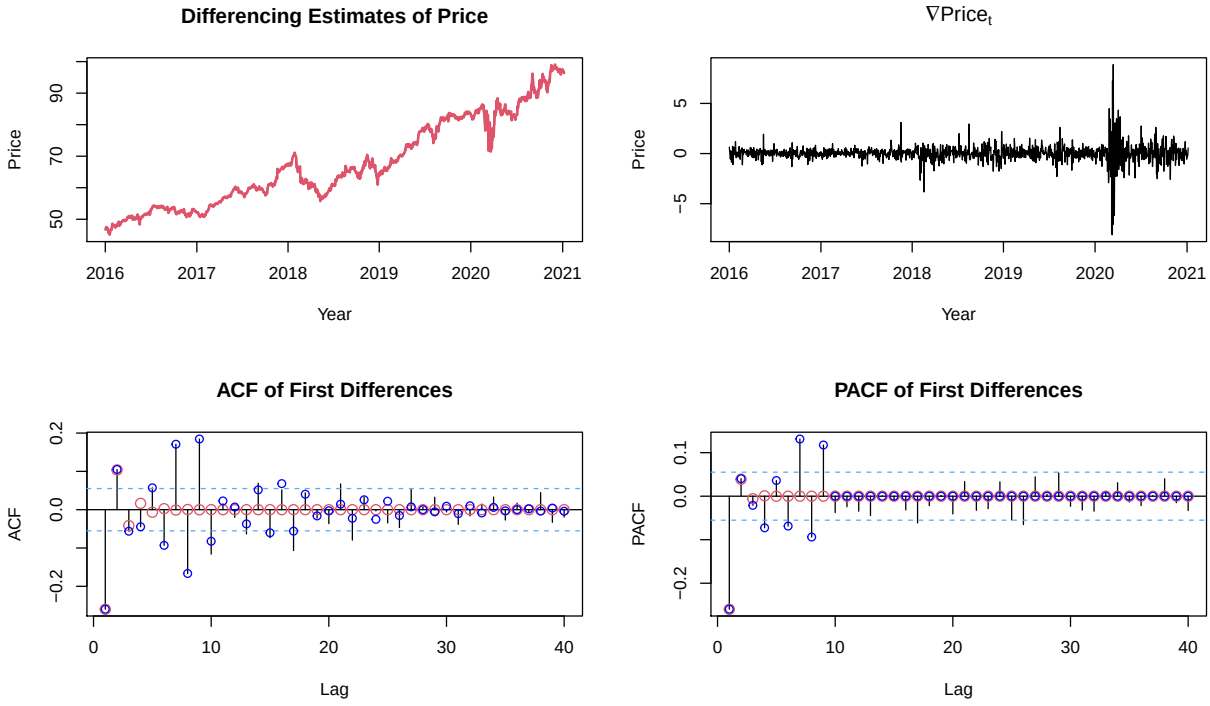

Figure 2 Top Left : LOSI stock prices overlayed with differencing estimates in red. Top Right: The first differences of closing stock prices over the years. Bottom Left: Autocorrelation function plot of first

differences on prices. Bottom Right: Partial autocorrelation function plot of first differences. *The blue circles are autocorrelation estimates for an AR(9) model, and the red circles are estimates from an ARMA(1,1) model.*

3.1.2 First Difference with AR(9)

The ACF/PACF of first differences has an odd set of significant values at lags 7-10 which are not being captured by $ARMA(1,1)$. This is likely the effect of the volatility observed in March 2020 as noted before. However, we can still try to incorporate this into the model, so both higher orders of AR and moving average models were considered for a better fit. The function $ar()$ provided the best fitting coefficients for an $AR(9)$ model, which isn't as parsimonious as desired but fits significant autocorrelations from lags $7-10$ quite well as reflected by the blue circles on the ACF/PACF plots in *Figure 2*. We also confirm it's goodness of fit with the Ljung-Box-Pierce test in the right panel of *Figure 3* which has very insignificant p-values up to lag 10.

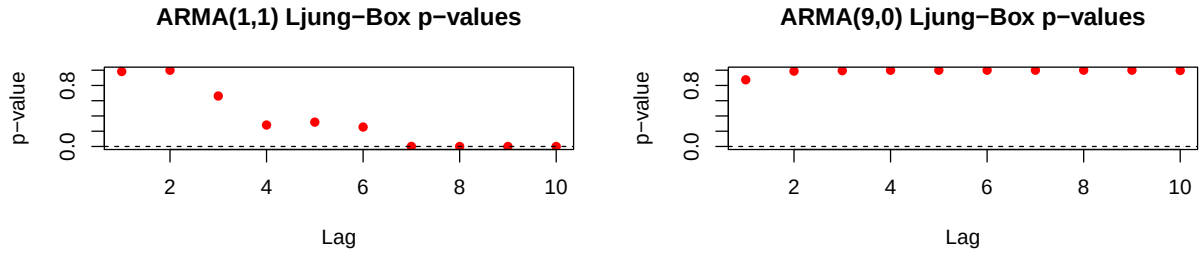

Figure 3 Ljung-Box test results of fitting $ARMA(1,1)$ and $ARMA(9,0)$ after first differencing on LOSI closing stock prices.

3.2 Model II - Exponential Filter

The trend can be estimated with an approach that still relies on past observations of prices for forecasting. Similar to differencing, exponential smoothing applies a one sided filter with a specified number of lags to the data and estimates $Price_t$ using decaying weights for a weighed sum of the past observations.

$$\hat{Price}_t = \left(\frac{(1-\alpha)}{\alpha} \sum_{i=1}^{15} \alpha^i Price_{t-i} \right) + X_t$$

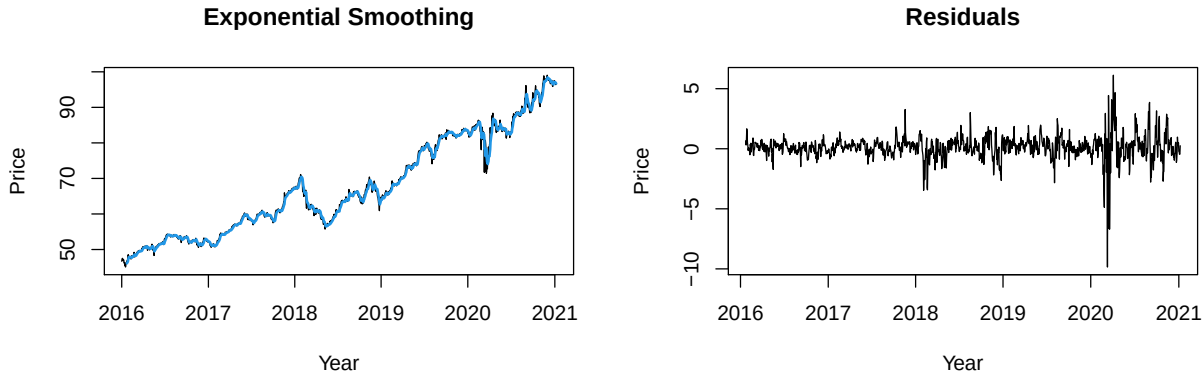

Figure 4 Left: Exponential smoothing estimates of LOSI closing stock prices after April 5 2020 in blue overlaid on the prices data. Right: Residuals of the smoothing estimates.

The value used for alpha is 0.67 which by trial and error seems to give an ideal balance of bias and variance for the number of lags being used (15), and we have reasonably stable residuals X_t left to model. The mean of the residuals is constant close to 0 but variance grows slightly after the volatile period in 2020. This is likely continued noise after the volatility so not a major heteroskedasticity issue and perhaps an ideal ARMA model can incorporate the noise.

3.2.1 Modeling Residuals - ARMA(1,2)

The autocorrelation plots for the residuals seen in *Figure 5* below, suggest an ARMA model with an AR order of perhaps 2, but once again we also have significant values around lags 7-10 in the PACF. An attempted $AR(2)$ model did not fit well, so an $ARMA(1,2)$ was fitted with coefficients by suggestion of $auto.arima()$ and is reflected by the red circles. The theoretical values for this model fit well with the lags $1-3$ in the ACF and PACF then decay to 0 very quickly afterwards. It's diagnostics with sarima has Ljung-Box p-values insignificant for the first 6 lags which is an acceptable fit.

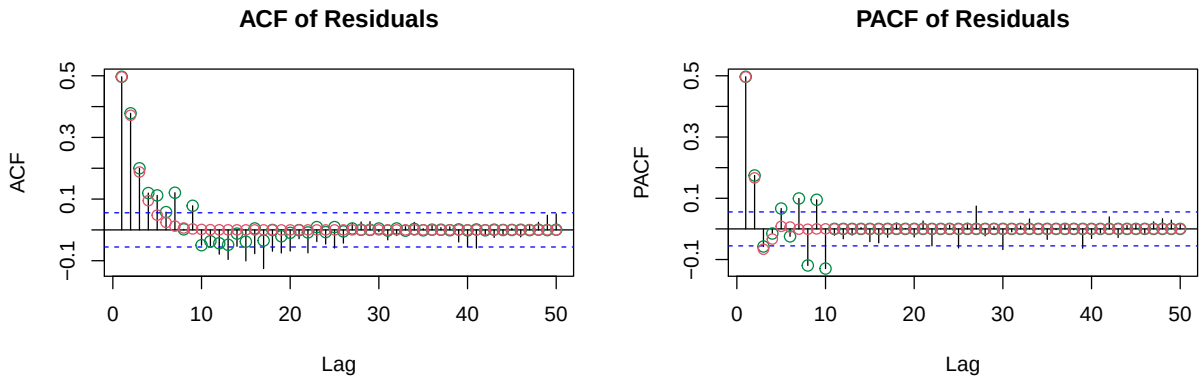

Figure 5 ACF and PACF values for exponential smoothing residuals. Green circles are autocorrelation values for an $AR(10)$ model, and red circles are autocorrelation values for an $ARMA(1,2)$ model.

3.2.2 Modeling Residuals - AR(10)

The exponential filter in use covers up to 15 lagged $Price_t$ points, so we may want a noise model that captures any significant autocorrelations at that many lags. A high order AR was able to capture all significant autocorrelations for the differencing model, so a similar approach can also be taken here. By trial and error of orders using the function $sarima()$, and the diagnostics of the residuals results in the best fit per the Ljung-Box-Pierce test at an AR order of 10.

The function $ar()$ specified with a starting and max order of 10 provided coefficients for the $AR(10)$ model fitted to the residuals with green circles in *Figure 5* above. It's autocorrelation values are about the same as that of $ARMA(1,2)$ for the first 3 lags, but it also fits the peculiar autocorrelations at lags $7-10$ before decaying.

4 Model Selection

For final predictions of stock prices, the best model will be selected by comparing cross validation root mean squared prediction errors (RMSPE). The cross validation process will start by reserving the first 1000 trading days as a training set, then make forecasts for rolling segments of 10 days ahead. The total out-of-sample forecasts will be 26 for a total of 260 days. The sum of root mean squared errors for the 26 forecasts will be averaged for a final CV-RMSPE. The model with the lowest CV-RMSPE will be regarded as the best model for predictions.

Model Choice	Root Mean Squared Prediction Error (RMSPE)
ARIMA(1,1,1)	2.818174
ARIMA(9,1,0)	2.907550
Exponential Filter + ARMA(1,2)	2.824818
Exponential Filter + AR(10)	2.854738

Table 1: Cross-validated out-of-sample root mean squared prediction error for the four models under consideration. *ARIMA is Daily Trading Day Differencing*

From the results in Table 1, it seems the model choices using parsimonious ARMA result in a better out-of-sample predictive performance. The RMSPE for all four choices are quite close, but the best model based on this metric is the first difference (Daily Trading Day Differencing) of stock prices combined with an ARMA(1,1) model for the noise.

5 Results

The final model selected will estimate a new price by using the previous trading day price with additive noise from the ARMA(1,1) model fitted with the best estimated coefficients as well as the average first difference of all previously observed prices. The model can be mathematically expressed as follows:

$$\hat{Price}_t = Price_{t-1} + \bar{V} + X_t$$

$\bar{V}$ is the sample mean of previously observed prices first differences and the stationary process X_t modeled as $ARMA(1,1)$ can be expressed as $X_t = \phi X_{t-1} + W_t - \theta W_{t-1}$ with W_t white noise.

5.1 Estimation of Final Model Parameters

With first order differencing and $ARMA(1,1)$, the final model has only 3 parameters estimated using as follows:

Parameter	Estimate	Standard Error	Description
ϕ	-0.3995	0.0897	AR Coefficient
θ	0.1499	0.0961	MA Coefficient
σ^2	0.671		Variance of White Noise

Table 2: Estimates of parameters used in the daily differencing model + stationary noise model.

5.2 Final Model Predictions

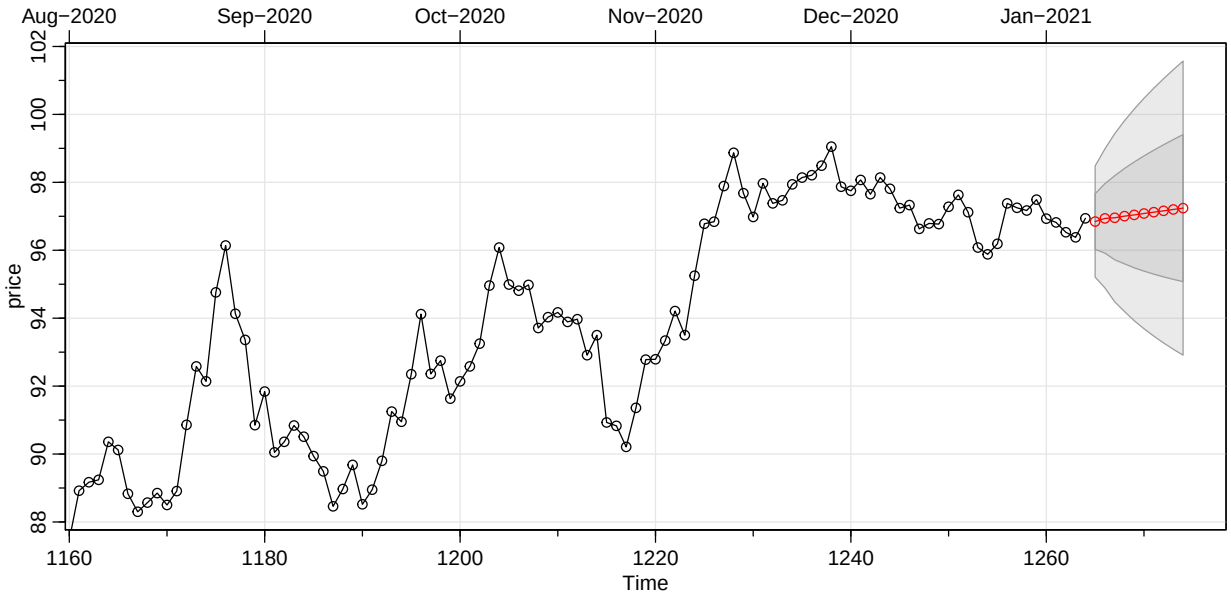

Figure 6 LOSI closing stock prices for most recent 104 trading days with forecasts of the next 10 days (01-11-21 to 01-25-21) in red. The x-axis are trading day time points starting at 2020-08-11. The forecasts have one std.error (68%) and two std.error (95%) prediction intervals reflected by the dark and light grey bands respectively.

As seen in *Figure 6*, final daily differencing model with ARMA(1,1) predicts a slight price drop in the next trading day. Then it forecasts that over the following 9 days, the price will grow slowly from 96.8 reaching 97.2 closer to the price level in early December of 2020. The uncertainty of these forecasts also grows for farther days. The models exact forecasts do not have a lot of noise since they are increasing consistently. The past 8 days closing prices leading up to the 10 days of the forecast were nearly consistently declining, and the possibility of such a continued decline is within the 95% prediction interval.

6 Appendix

6.1 Code

```
#Loading dataset
stock_dat <- read.csv("data_stock.csv")
blank_stock <- read.csv("blank_stock.csv")

stock_dat$Date <- stock_dat$Date %>% as.Date()
blank_stock$Date <- blank_stock$Date %>% as.Date()

stock_dat$t_index <- (1:nrow(stock_dat))
blank_stock$t_index <- ((nrow(stock_dat)+1) : (nrow(stock_dat)+nrow(blank_stock)))
```

```
#Univariate time series plot of stock prices
stock_dat %>% ggplot(aes(x=Date, y=Price)) +
  geom_path() +
  theme_classic() +
  labs(title="Lots-of-stuff Incorporated Daily Stock Prices Over Time", x="Year") +
  theme(plot.title = element_text(face='bold', hjust = 0.5))
```

```
#change date formats
stock_dat$year <- stock_dat$Date %>% lubridate::year()
stock_dat$month <- stock_dat$Date %>% lubridate::month()
stock_dat$day <- stock_dat$Date %>% lubridate::day()

blank_stock$year <- blank_stock$Date %>% lubridate::year()
blank_stock$month <- blank_stock$Date %>% lubridate::month()
blank_stock$day <- blank_stock$Date %>% lubridate::day()

#EDA Plots
#par(mfrow = c(2, 3))
#stock_dat %>% filter(year == 2016) %>% select(Price) %>%
#  plot.ts(main="2016", xlab="Day")
#stock_dat %>% filter(year == 2017) %>% select(Price) %>%
#  plot.ts(main="2017", xlab="Day")
#stock_dat %>% filter(year == 2018) %>% select(Price) %>%
#  plot.ts(main="2018", xlab="Day")
#stock_dat %>% filter(year == 2019) %>% select(Price) %>%
#  plot.ts(main="2019", xlab="Day")
#stock_dat %>% filter(year == 2020) %>% select(Price) %>%
#  plot.ts(main="2020", xlab="Day")
```

```
price = stock_dat$Price
stock_recent <- stock_dat %>% filter(t_index >= 1072)
recent_price <- stock_recent$Price
```

```
# Remove trend with First Difference
price.diff <- diff(price, differences = 1)
```

```r
#Estimate prices using difference formula
diff.estimates <- NA
for (i in 2:length(price)) {
  diff.estimates[i] = mean(price.diff) + price[i-1]
}
```

```r
max_lag=40
par(mfrow = c(2, 2))

#Stock prices plot with model estimates
price %>% plot.ts(main = "Differencing Estimates of Price",
                  xlab="Year",
                  ylab="Price",
                  axes=F)
diff.estimates %>% lines(col=2, lwd=2)
box()
axis(2)
axis(1,at = c(1, 253, 504, 755, 1007, 1260), labels = 2016:2021)

#Residuals plot
price.diff %>% plot.ts(main = expression(paste(nabla,"Price"[t])),
                  xlab="Year",
                  ylab="Price",
                  axes=F)
box()
axis(2)
axis(1,at = c(1, 253, 504, 755, 1007, 1260), labels = 2016:2021)



# Get first difference acf/pacf values
diff.acf <- acf(price.diff, ylim=c(-1,1), lag.max=max_lag, plot=F)$acf
diff.pacf <- pacf(price.diff, ylim=c(-1,1), lag.max = max_lag, plot=F)$acf

#First Noise Model
diff.auto_arima <- auto.arima(price.diff)


est_acf <- stats::ARMAacf(ar=diff.auto_arima$coef[1],
                          ma=diff.auto_arima$coef[2],
                          lag.max = max_lag, pacf=F)


est_pacf <- stats::ARMAacf(ar=diff.auto_arima$coef[1],
                           ma=diff.auto_arima$coef[2],
                           lag.max = max_lag, pacf=T)


#Second Noise Model
ar_est <- ar(price.diff)
ar.phis <- ar_est$ar
ma.thetas <- c()
```

```
est_acf2 <- stats::ARMAacf(ar=ar.phis, ma=ma.thetas, lag.max = max_lag, pacf=F)
est_pacf2 <- stats::ARMAacf(ar=ar.phis, ma=ma.thetas, lag.max = max_lag, pacf=T)


L = 1.96/sqrt(length(price.diff))

plot(1:max_lag, diff.acf,
     xlab = "Lag",
     ylab = "ACF",
     type = "h",
     ylim = c(min(diff.acf), max(diff.acf)),
     main = "ACF of First Differences")
abline(h = c(0, -L, L), lty = c(1, 2, 2), col = c('black','steelblue2', 'steelblue2'))
points((1:max_lag), est_acf[-1], col = 2, cex=1.3)
points((1:max_lag), est_acf2[-1], col = 'blue1', cex=1)


plot(1:max_lag, diff.pacf,
     xlab = "Lag",
     ylab = "PACF",
     type = "h",
     ylim = c(min(diff.pacf), max(diff.pacf)),
     main = "PACF of First Differences")
abline(h = c(0, -L, L), lty = c(1, 2, 2), col = c('black','steelblue2', 'steelblue2'))
points((1:max_lag), est_pacf, col = 2, cex=1.3)
points((1:max_lag), est_pacf2, col = 'blue1', cex=1)
```

```
#Ljung-Box tests
test = arima(price.diff, order=c(1,0,1))
test2 = arima(price.diff, order=c(9,0,0))

par(mfrow = c(1, 2))
#Arma(1,1)
lags = (1:10)
pvalues = NA
for (i in lags) {
  result =Box.test(test$residuals, lag=i, type="Ljung-Box")
  pvalues[i] = result$p.value

}
plot(lags, pvalues,
     xlab = "Lag",
     ylab = "p-value",
     type = "p",
     col = 'red',
     pch = 16,
     ylim = c(0, 1),
     main = "ARMA(1,1) Ljung-Box p-values")
abline(h = 0, lty = 2, col = 'black')

#Arma(9,0)
lags = (1:10)
pvalues = NA
```

```r
for (i in lags) {
  result =Box.test(test2$residuals, lag=i, type="Ljung-Box")
  pvalues[i] = result$p.value

}
plot(lags, pvalues,
     xlab = "Lag",
     ylab = "p-value",
     type = "p",
     col = 'red',
     pch = 16,
     ylim = c(0, 1),
     main = "ARMA(9,0) Ljung-Box p-values")
abline(h = 0, lty = 2, col = 'black')
```

```r
w = 0.67 ; w.lags = 15
price = (price)
exp.weights =  (w^(1:w.lags))
exp.weights = exp.weights/sum(exp.weights)
exp_smooth = stats::filter(price, sides = 1, filter=c(0,exp.weights))
exp_residual = (price - exp_smooth) %>% na.omit()


par(mfrow = c(1, 2))

#Stock prices plot with smoothing estimates
price %>% plot.ts(main = "Exponential Smoothing",
                  xlab="Year",
                  ylab="Price",
                  axes=F)
exp_smooth %>% lines(col=4, lwd=2)
box()
axis(2)
#axis(1,at = c(1, 190), labels = 2020:2021)
axis(1,at = c(1, 253, 504, 755, 1007, 1260), labels = 2016:2021)

#Residuals plot
exp_residual %>% plot.ts(main = "Residuals",
                  xlab="Year",
                  ylab="Price",
                  axes=F)
box()
axis(2)
#axis(1,at = c(1, 190), labels = 2020:2021)
axis(1,at = c(1, 253, 504, 755, 1007, 1260), labels = 2016:2021)
```

```r
max_lag=50

test = ar(exp_residual)


#Get exponential model residual acf/pacf values
exp_residual.acf <- acf(exp_residual, ylim=c(-1,1), lag.max=max_lag, plot=F)$acf
```

```r
exp_residual.pacf <- pacf(exp_residual, ylim=c(-1,1), lag.max = max_lag, plot=F)$acf

#AR based on observations
ar.phis <- test$ar
ar1_acf <- stats::ARMAacf(ar=ar.phis,lag.max = max_lag, pacf=F)
ar1_pacf <- stats::ARMAacf(ar=ar.phis, lag.max = max_lag, pacf=T)


#Auto Arima suggestion
exp_res.auto_arma <- auto.arima(exp_residual)

ar.phis <- c(0.5050) ; ma.thetas <- c(-0.0824, 0.1664)
auto_arma_acf <- stats::ARMAacf(ar=ar.phis, ma=ma.thetas, lag.max = max_lag, pacf=F)
auto_arma_pacf <- stats::ARMAacf(ar=ar.phis, ma=ma.thetas, lag.max = max_lag, pacf=T)


L = 1.96/sqrt(length(exp_residual))

par(mfrow = c(1, 2))
#Exponential Smoothing Residual ACF plot
plot(1:max_lag, exp_residual.acf,
     xlab = "Lag",
     ylab = "ACF",
     type = "h",
     ylim = c(min(exp_residual.acf), max(exp_residual.acf)),
     main = "ACF of Residuals")
abline(h = c(0, -L, L), lty = c(1, 2, 2), col = c('black','blue', 'blue'))
points((1:max_lag), ar1_acf[-1], col = 'springgreen4', cex=1.2)
points((1:max_lag), auto_arma_acf[-1],pch=1,  col = 2, cex=1.1)

#Exponential Smoothing Residual PACF plot
plot(1:max_lag, exp_residual.pacf,
     xlab = "Lag",
     ylab = "PACF",
     type = "h",
     ylim = c(min(exp_residual.pacf), max(exp_residual.pacf)),
     main = "PACF of Residuals")
abline(h = c(0, -L, L), lty = c(1, 2, 2), col = c('black','blue', 'blue'))
points((1:max_lag), ar1_pacf, col = 'springgreen4', cex=1.2)
points((1:max_lag), auto_arma_pacf, pch=1, col = 2, cex=1.1)
```

```r
n = length(price)
mid_point <- as.integer(n*0.1)*5
cut_offs = seq(from = 1000, to = 1250, by=10)

#ARIMA(1,1,1) cross validation
mse_sum = 0
for (i in cut_offs) {
  preds = sarima.for(price[1:i] , n.ahead=10,
                     p=1,d=1,q=1,D=0,Q=0, plot=F)

  mse_sum = mse_sum + mean((price[(i+1):(i+10)] - preds$pred)^2)
}
```

```
root_mses[1] = sqrt(mse_sum/length(cut_offs))

#ARIMA(9,1,0) cross validation
mse_sum = 0
for (i in cut_offs) {
  preds = sarima.for(price[1:i] , n.ahead=10,
                     p=9,d=1,q=0,D=0,Q=0, plot=F)

  mse_sum = mse_sum + mean((price[(i+1):(i+10)] - preds$pred)^2)
}
root_mses[2] = sqrt(mse_sum/length(cut_offs))
```

```
#Exponential Filter predictor
filter_pred <- function(data, weights, noise_forecasts, forecasts_num) {
  data_w_forecasts = c(data, rep(NA, forecasts_num))
  n = length(data)
  m = length(weights)
  for (i in 1:forecasts_num) {
    data_w_forecasts[n+i] = weights%*%data_w_forecasts[(n+i-1):(n+i-m)] +
      noise_forecasts$pred[i]
  }
  return(data_w_forecasts[(n+1):(n+forecasts_num)])
}


n = length(price)
mid_point <- as.integer(n*0.1)*5
cut_offs = seq(from = 1000, to = 1250, by=10)


#Exponential Smoothing with ARMA(1,2) CV
mse_sum = 0
for (i in cut_offs) {
  prices = price[1:i]
  smooth = stats::filter(prices, sides = 1, filter=c(0,exp.weights))
  residual = (prices - smooth) %>% na.omit()

  noise_preds = sarima.for(residual, n.ahead = 10,
                               p=1,d=0,q=2,P=0,D=0,Q=0, plot = F)

  mse_sum = mse_sum + mean((price[(i+1):(i+10)] -  filter_pred(prices,
                                                       exp.weights,
                                                       noise_preds, 10))^2)
}

root_mses[3] = sqrt(mse_sum/length(cut_offs))
```

```
#Exponential Smoothing with AR(10) CV
mse_sum = 0
for (i in cut_offs) {
```

```
  prices = price[1:i]
  smooth = stats::filter(prices, sides = 1, filter=c(0,exp.weights))
  residual = (prices - smooth) %>% na.omit()

  noise_preds = sarima.for(residual, n.ahead = 10,
                           p=10,d=0,q=0,P=0,D=0,Q=0, plot = F)

  mse_sum = mse_sum + mean((price[(i+1):(i+10)] -  filter_pred(prices,
                                                    exp.weights,
                                                    noise_preds, 10))^2)
}

root_mses[4] = sqrt(mse_sum/length(cut_offs))
```

```
#RMSE table
rmse = matrix(root_mses, nrow=4,ncol = 1)
colnames(rmse) = "RMSPE"
rownames(rmse) = c(
        "ARIMA(1,1,1)  ",
        "ARIMA(9,1,0)",
        "Exponential Filter + ARMA(1,2) ",
        "Exponential Filter + AR(10)"
        )
knitr::kable(rmse,caption = "Cross-validated out-of-sample root mean squared prediction error for the f
```

```
#Forecasting with ARIMA(1,1,1)
par(oma=c(0,0,2,0))
attempt = sarima.for(price, n.ahead=10, p=1,d=1,q=1,S=0,P=0,Q=0)$pred
axis(3,at = c(1160, 1180, 1200, 1220, 1240, 1260), labels = c("Aug-2020", "Sep-2020", "Oct-2020", "Nov-
```

```
#Forecasting with Exponential Smoothing + Arma(1,2)
data_w_forecasts = c(exp_smooth, rep(NA, 10))
n = length(exp_smooth)
m = length(exp.weights)
for (i in 1:10) {
    data_w_forecasts[n+i] = exp.weights%*%data_w_forecasts[(n+i-1):(n+i-m)]
  }

#par(oma=c(0,0,2,0))
#attempt = sarima.for(price, n.ahead=10, p=1,d=0,q=2,S=0,P=0,Q=0, xreg = exp_smooth, newxreg = data_w_f
#axis(3,at = c(1160, 1180, 1200, 1220, 1240, 1260), labels = c("Aug-2020", "Sep-2020", "Oct-2020", "Nov
```