Economics 430 - Homework - 3

Anshika Sharma, UCLA ID: 305488635

Contents

Part I.

I -Introduction (describe the data, provide some background on the topic, etc.).

I downloaded the time series data on **retail sales of hardware stores located in the United States** from FRED. The data is reported in millions of USD. As is required by the question, the data shows an upward/increasing trend and it shows the presence of seasonality. The seasonality is expected since construction of houses and other outdoor projects tend to peak in summer months and hence the retail sales of hardware stores also peaks in the summer months. The frequency of the data is monthly and goes form January 1992 to August 2020. We see a big spike in the sales in 2020 inspite of the pandemic. One of the reasons as per the article I read was that and I quote: Hardware and home improvement retailers have been deemed essential businesses and allowed to remain open. People were not able to travel, go on vacation and eat at restaurants, so they were spending their time and money sprucing up their homes and yards. Garden centers have not been allowed to stay open in many areas, which lets hardware retailers capitalize on the surging demands for flowers and plants. Big-box stores in states such as Michigan have been restricted from selling non-essential items like paint and gardening supplies, leaving all that business to stores under 50,000 square feet. Unquote

Part II.

Part II - Results (answers and plots). Consists of two parts:

1. Modeling and Forecasting Trend

(a) Show a time-series plot of your data

We first begin by downloading and reading in the csv file of the data on retail sales of hardware stores from the **FRED** site.

```
Hardware <- read_csv("C:\\Users\\anshi\\Desktop\\HW3\\Hardware.csv")
```

```
#Changing the column names for our own ease
names(Hardware) = c("observation_date","retail_sales")
head(Hardware)
```

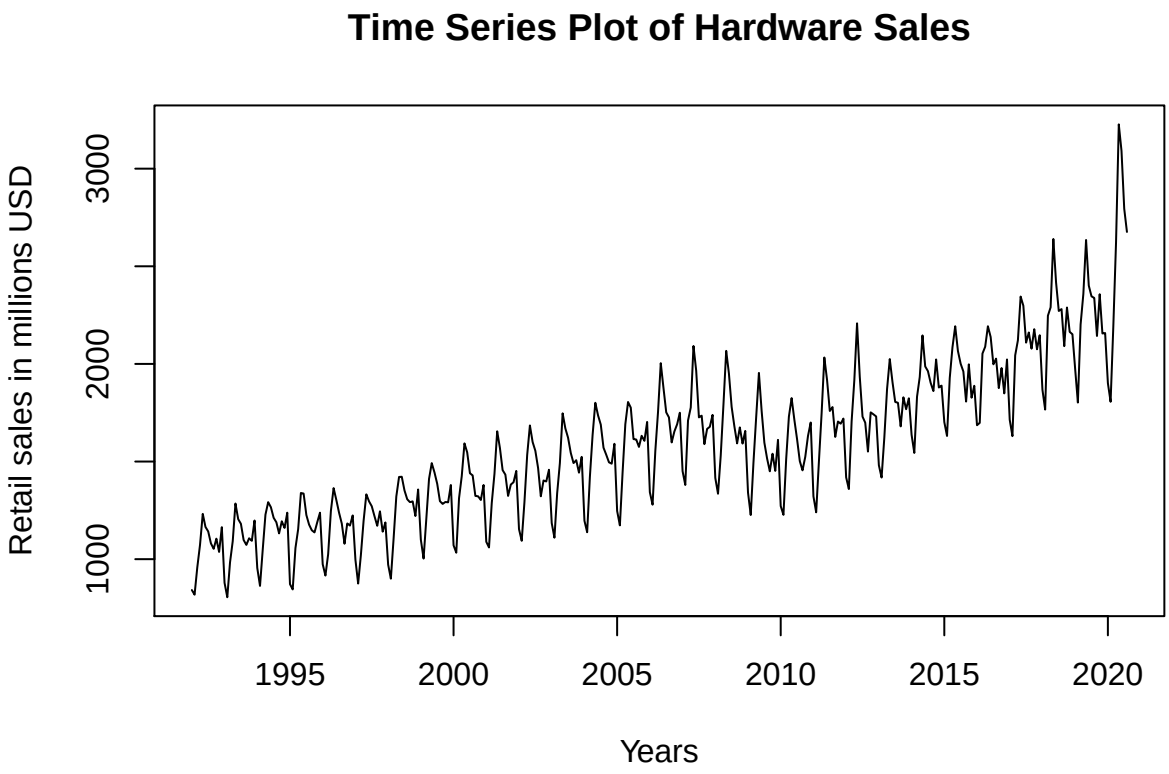
```
## # A tibble: 6 x 2
##   observation_date retail_sales
##   <chr>                   <dbl>
## 1 01-01-1992                842
## 2 01-02-1992                818
## 3 01-03-1992                958
## 4 01-04-1992               1073
## 5 01-05-1992               1232
## 6 01-06-1992               1165
```

Plotting the Time-series of the hard-ware store data: Note that we first need to convert our data into time series object so that the appropriate tests can be conducted and forecasts can be made. This is a very crucial step, without which analysis can't be made. This can be easily done in R by the following code. Since the data is monthly data, we set the frequency to 12 to include the number of months. I saved the time series data as "hardware" for ease.

```
#Converting the date into Time-Series
hardware <-ts(Hardware$retail_sales,start=1992,freq=12)
head(hardware, 12)
```

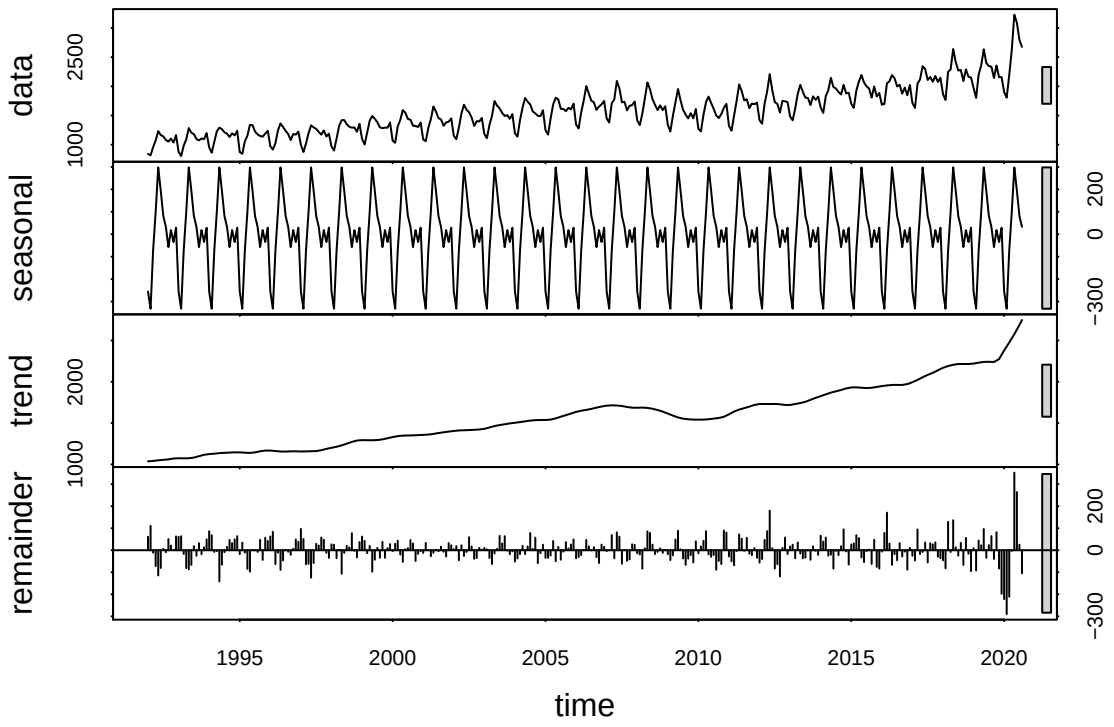
```
##       Jan  Feb  Mar  Apr  May  Jun  Jul  Aug  Sep  Oct  Nov  Dec
## 1992  842  818  958 1073 1232 1165 1142 1081 1053 1105 1037 1164
```

```
#plotting the time series data
plot(hardware, main = "Time Series Plot of Hardware Sales",
     ylab ="Retail sales in millions USD",
     xlab = "Years")
```



As can be seen in the plot above, the data exhibits an upward trend. The graph also suggests that there is seasonality in the data. A spike can be seen in the middle of every year.

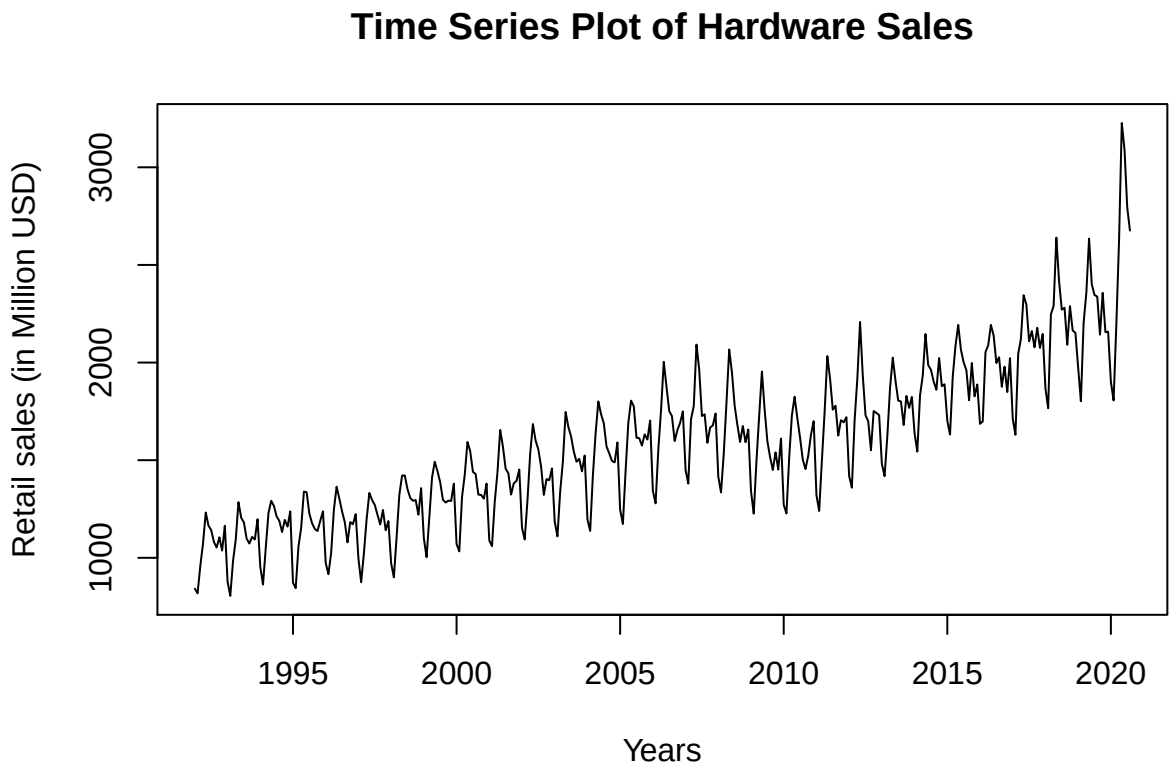
```r
#STL plot
plot(stl(hardware, s.window = "periodic"))
```



To further confirm the presence of trend and seasonality, I plotted the STL and it clearly indicates an upward trend and presence of seasonality.

(b) Does your plot in (a) suggest that the data are co-variance stationary? Explain your answer.

```
#Plotting the data
plot(hardware, main = "Time Series Plot of Hardware Sales",
     ylab ="Retail sales (in Million USD)",
     xlab = "Years")
```




```
#Carrying out the KPSS test as a sanity check
#Null Hypothesis: Stationary ; Alternative Hypothesis: Non-stationary
kpss.test(hardware)
```

```
## Warning in kpss.test(hardware): p-value smaller than printed p-value
```

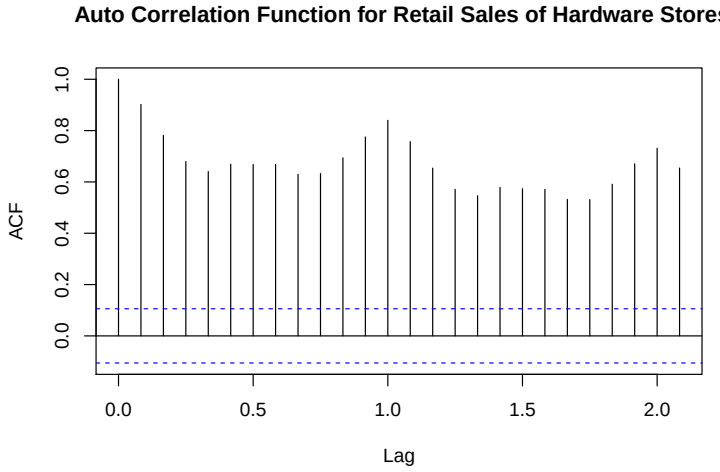
```
##
##  KPSS Test for Level Stationarity
##
## data:  hardware
## KPSS Level = 5.1125, Truncation lag parameter = 5, p-value = 0.01
```

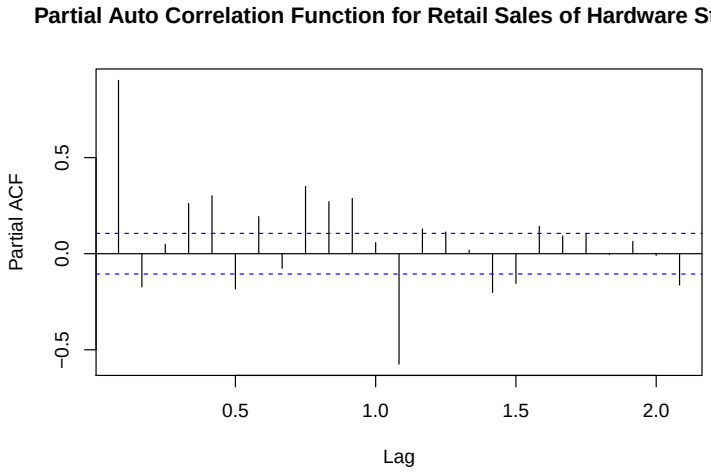
Looking at the graph, it does not look co-variance stationary. Stationarity requires that the data should mean-revert(constant mean), but it doesn't seem to. Rather, the mean is increasing over time. There is evidence of volatility in the data. So, it is not variance constant and the variance seems to be increasing over time. Thus, the violation of these two conditions tells us that the data is not co-variance stationary. The

KPSS test, as a sanity check also suggests that the data is not co-variance stationary since the p value for kpss test is $0.01 < 0.05$. So we reject the null hypothesis and say that the data is not covariance stationary.

(c) Plot and discuss the ACF and PACF of your data.

```
#tsdisplay(hardware) - shows data plot, ACF and PACF plot but to zoom in,
                       #we plot it separately.
acf(hardware,
main = " Auto Correlation Function for Retail Sales of Hardware Stores")

pacf(hardware,
main = "Partial Auto Correlation Function for Retail Sales of Hardware Stores")
```



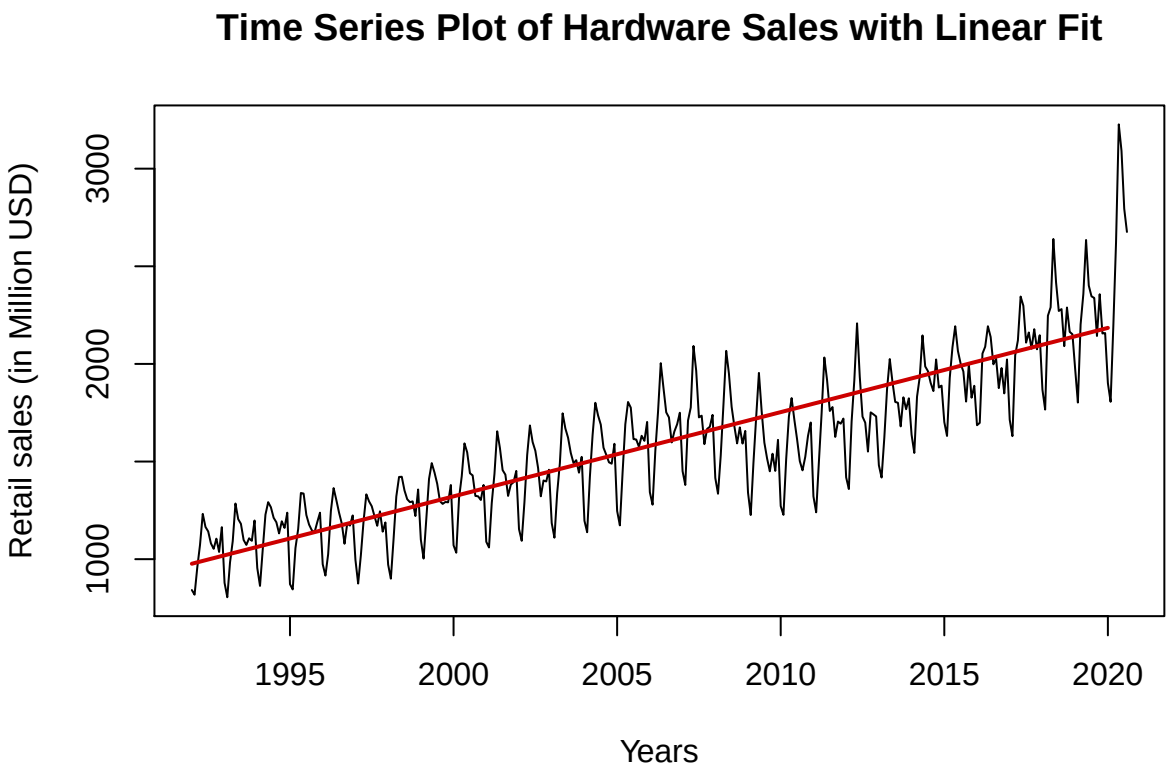


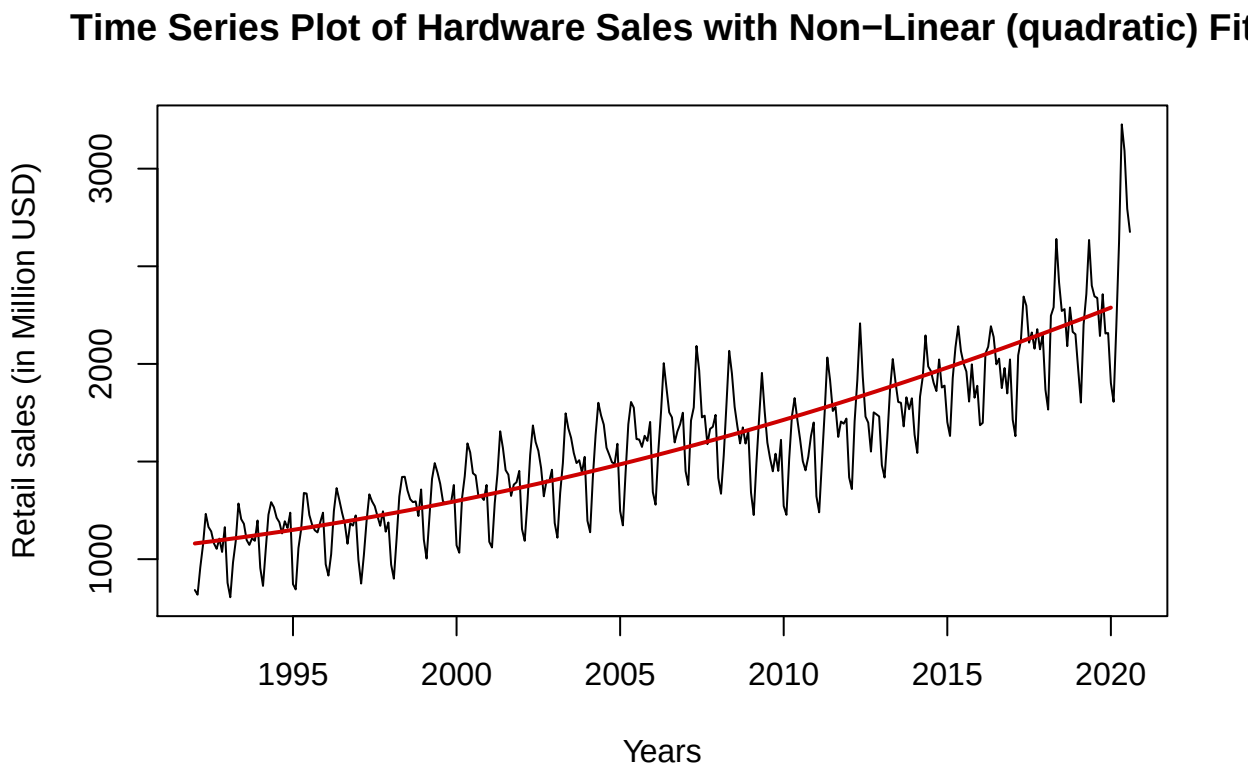
Both the ACF and PACF suggest a high dependence on how retail sales in hardware stores have changed over time. More specifically, retail sales in period t are highly dependent on retail sales from previous periods. In this case, auto-correlation remains positive and decreases as lags increase. In other words, When the data has a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series tend to have positive values that slowly decrease as the lags increase. Our data has a clear increasing trend over time, which is captured in the ACF plot with positive autocorrelations that are decreasing as the lag increases. The PACF suggests that the

correlation coefficient fluctuates between positive and negative and auto-correlation becomes smaller as lag increases.When data are seasonal, the auto-correlations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags. Our data has a clear seasonal component, which is captured in the ACF plot as the auto-correlations are increasing and decreasing at regular intervals.

(d) Fit a linear and nonlinear (e.g., polynomial, exponential, quadratic + periodic, etc.) model to your series. In one window, show both figures of the original times series plot with the respective fit.

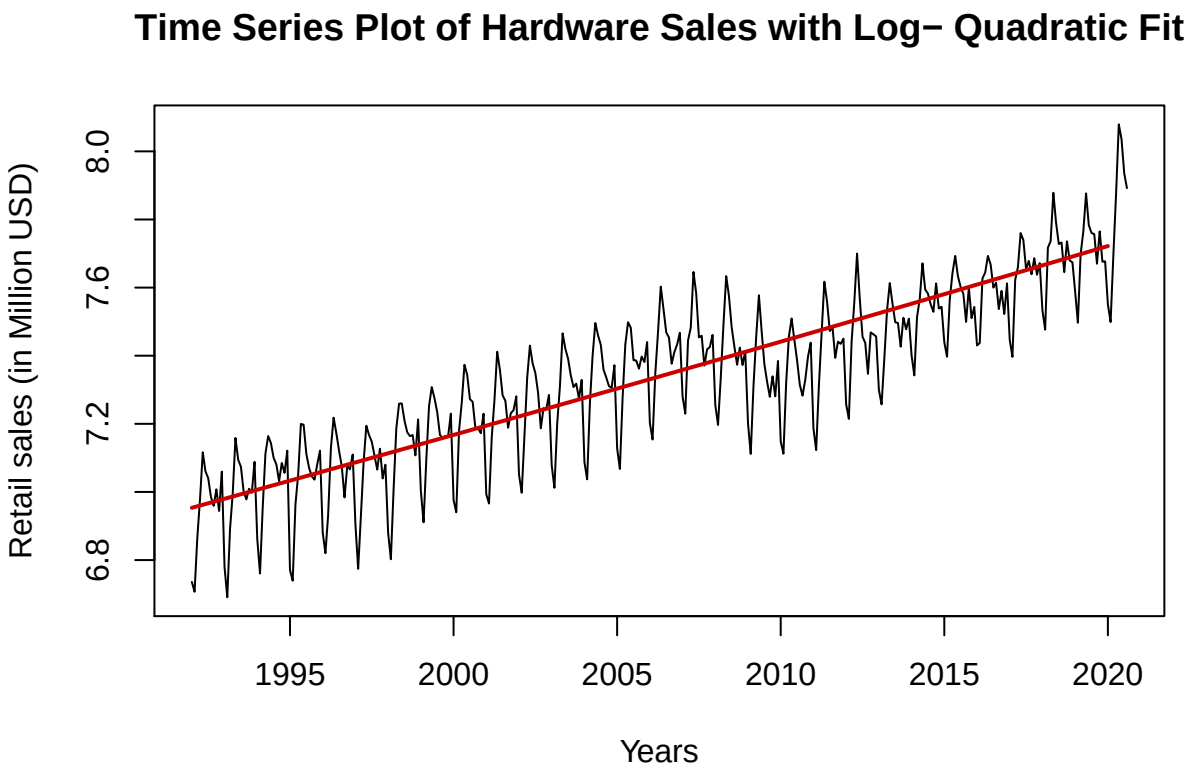
```
#creating a dummy variable for time
t <-seq(1992, 2020,length=length(hardware))

#Fitting and developing a model:

#The commands below are used to create models to attempt and see how the models
#fit relative to the data. We will use the lines command to fit the model to the
#data and see visually how it looks.

#Model with linear fit
model1 <- lm(hardware ~ t, data = hardware)

#Model with non-linear (quadratic) fit
model2 <- lm(hardware ~ t + I(t^2), data = hardware)


#Fitting a linear - trend model to time series and plotting it
plot(hardware, main = "Time Series Plot of Hardware Sales with Linear Fit",
     ylab ="Retail sales (in Million USD)",
     xlab = "Years")
lines(t,model1$fit,col="red3",lwd=2)

#Fitting a non-linear (quadratic) - trend model to time series and plotting it
plot(hardware,
     main = "Time Series Plot of Hardware Sales with Non-Linear (quadratic) Fit",
     ylab ="Retail sales (in Million USD)",
     xlab = "Years")
lines(t,model2$fit,col="red3",lwd=2)
```

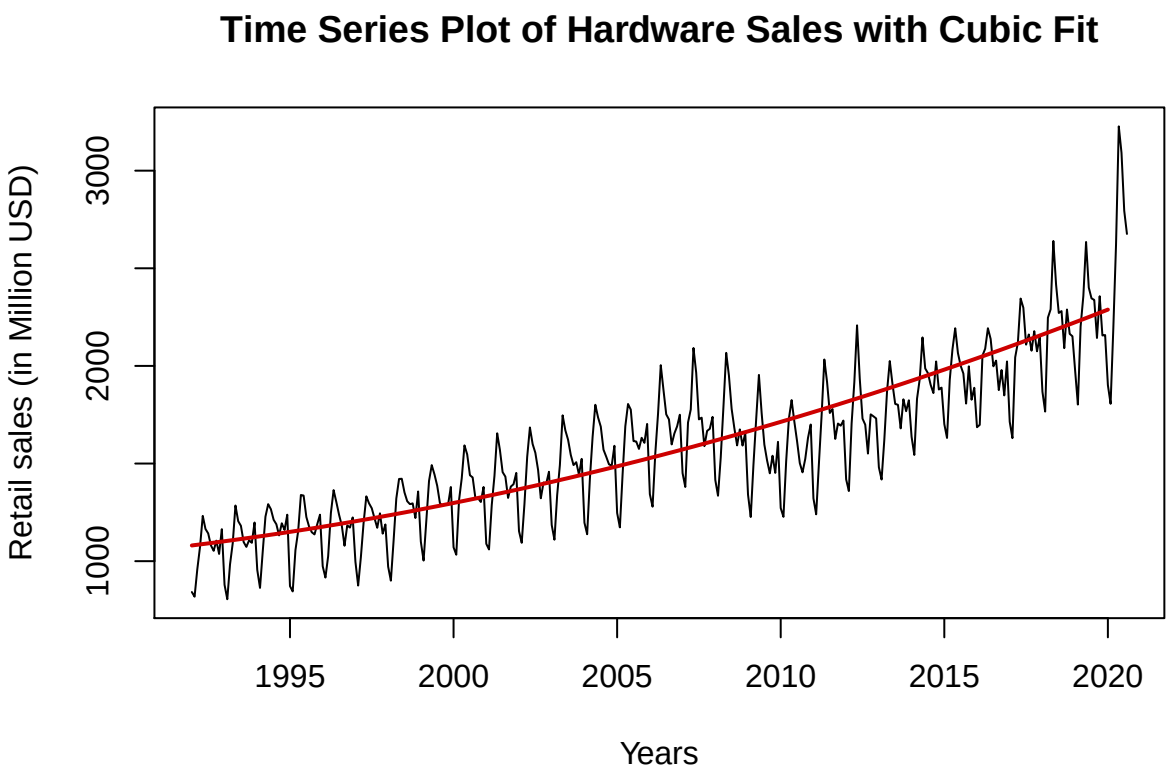

Time Series Plot of Hardware Sales with Linear Fit
Retail sales (in Million USD)
1000
2000
3000
1995
2000
2005
2010
2015
2020
Years



For the non-linear model, I tried a-few other models as well (to see if any fits the data better than quadratic) like:

```
#Note:
#For the non-linear model, I tried a-few other models as well like:
#Log-quadratic
try.mod.1 <- lm(log(hardware) ~ t + I(t^2)   , data = hardware)
plot(log(hardware),
     main = "Time Series Plot of Hardware Sales with Log- Quadratic Fit",
     ylab ="Retail sales (in Million USD)",
     xlab = "Years")
lines(t,try.mod.1$fit,col="red3",lwd=2)
#cubic
try.mod.2 <- lm(hardware ~ t + I(t^2) + I(t^3) , data = hardware)
plot(hardware, main = "Time Series Plot of Hardware Sales with Cubic Fit",
     ylab ="Retail sales (in Million USD)",
     xlab = "Years")
lines(t,try.mod.2$fit,col="red3",lwd=2)
```


Time Series Plot of Hardware Sales with Log− Quadratic Fit
Retail sales (in Million USD)
6.8
7.2
7.6
8.0
1995
2000
2005
2010
2015
2020
Years



I concluded that they don't give a fit better than the quadratic model, hence, I stick to quadratic fit, that is, model 2.

As can be seen above, we developed and fitted models to our data set. And then, we can choose the model that fits the data well to carry out the forecast. The better the model fits the data, the better forecast we get. The summary statistics and other parameters of the models will be discussed in the other part, but in this part as seen above, we are just doing a visual representation.

(e) For each model, plot the respective residuals vs. fitted values and discuss your observations.

```
#plotting the respective residuals vs. fitted values for model1
plot(model1$fitted.values,model1$residuals,type='l', xlab="Fitted Values",
     ylab = "Residuals", main = "Residuals vs. Fitted Values with Linear Fit")
abline(h = mean(model1$residuals), col = "red")
legend("topleft","Mean of Residuals",col = "red",
       bty='n',lty = 1)

#plotting the respective residuals vs. fitted values for model2
plot(model2$fitted.values,model2$residuals,type='l', xlab="Fitted Values",
     ylab = "Residuals", main = "Residuals vs. Fitted Values with Non-Linear Fit")
abline(h = mean(model2$residuals), col = "red")
legend("topleft","Mean of Residuals",col = "red",
       bty='n',lty = 1)

#Plotting Residuals against Time
plot(t, model1$residuals, main="Residuals vs Time for Model 1 ",
     xlab ="Time", ylab = "Residuals")
plot(t, model2$residuals, main="Residuals vs Time for Model 2 ",
```

```
     xlab = "Time", ylab = "Residuals")

#Tests for a sanity check
adf.test(model1$residuals)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  model1$residuals
## Dickey-Fuller = -5.71, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(model2$residuals)
```

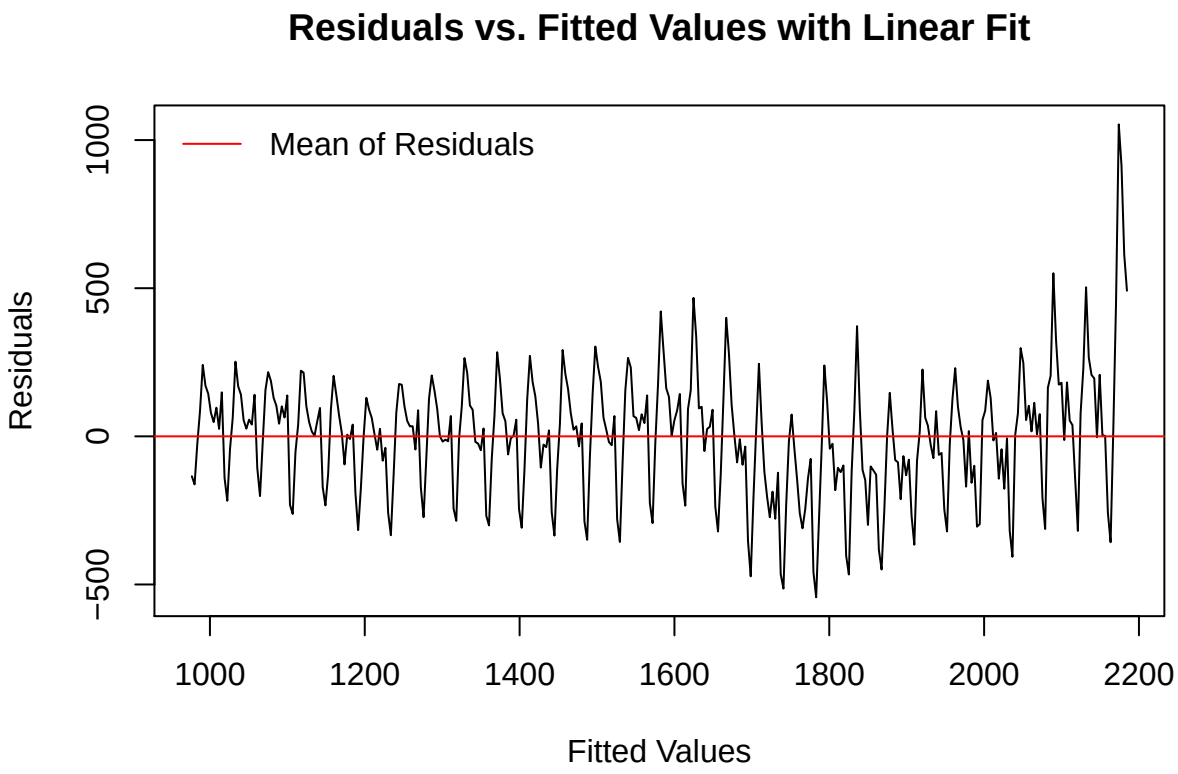
```
##
##  Augmented Dickey-Fuller Test
##
## data:  model2$residuals
## Dickey-Fuller = -6.5411, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

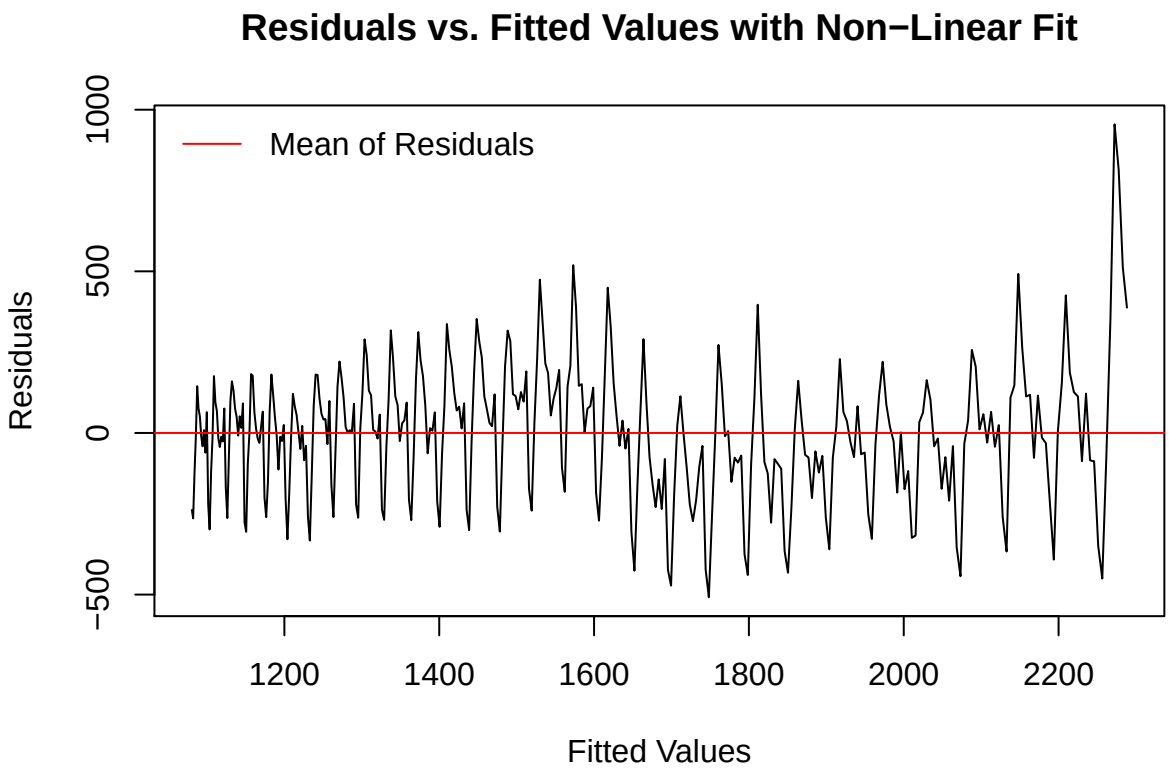
```
kpss.test(model1$residuals)
```

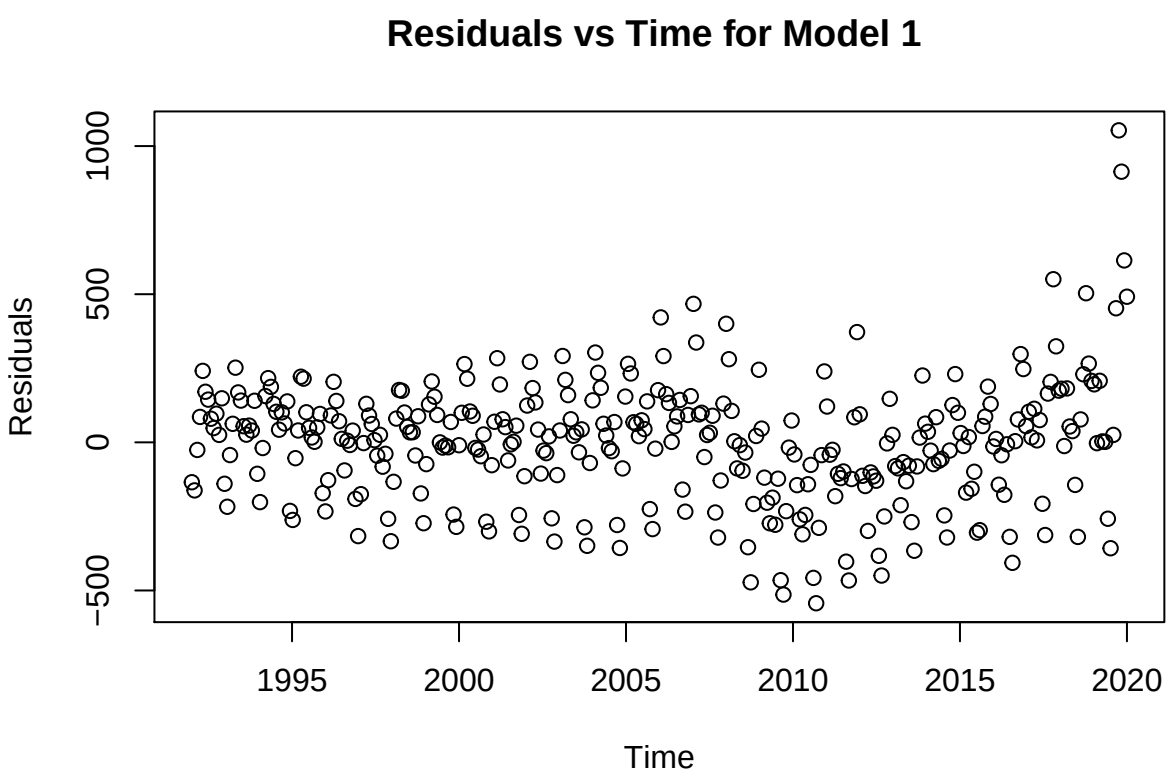
```
##
##  KPSS Test for Level Stationarity
##
## data:  model1$residuals
## KPSS Level = 0.2931, Truncation lag parameter = 5, p-value = 0.1
```

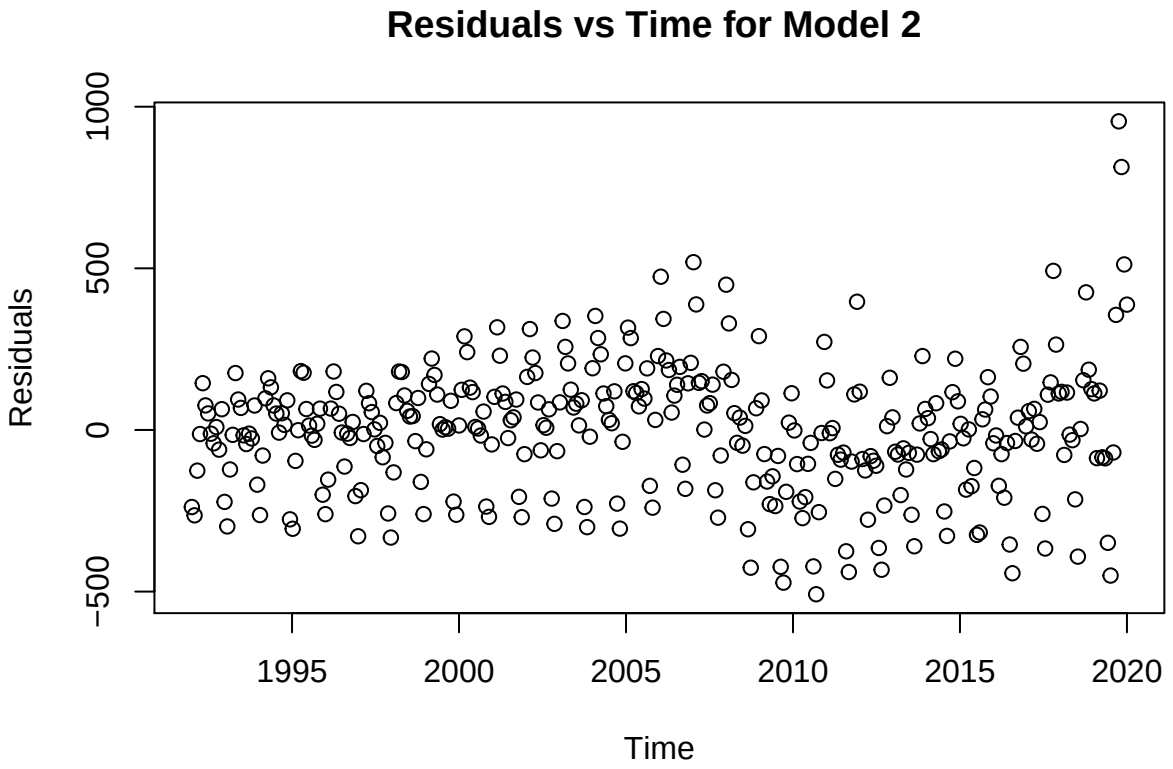
```
kpss.test(model1$residuals)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  model1$residuals
## KPSS Level = 0.2931, Truncation lag parameter = 5, p-value = 0.1
```


Residuals vs. Fitted Values with Linear Fit
Mean of Residuals
Residuals
1000
500
0
−500
1000
1200
1400
1600
1800
2000
2200
Fitted Values


Residuals vs. Fitted Values with Non−Linear Fit
Mean of Residuals
Residuals
1000
500
0
−500
1200
1400
1600
1800
2000
2200
Fitted Values

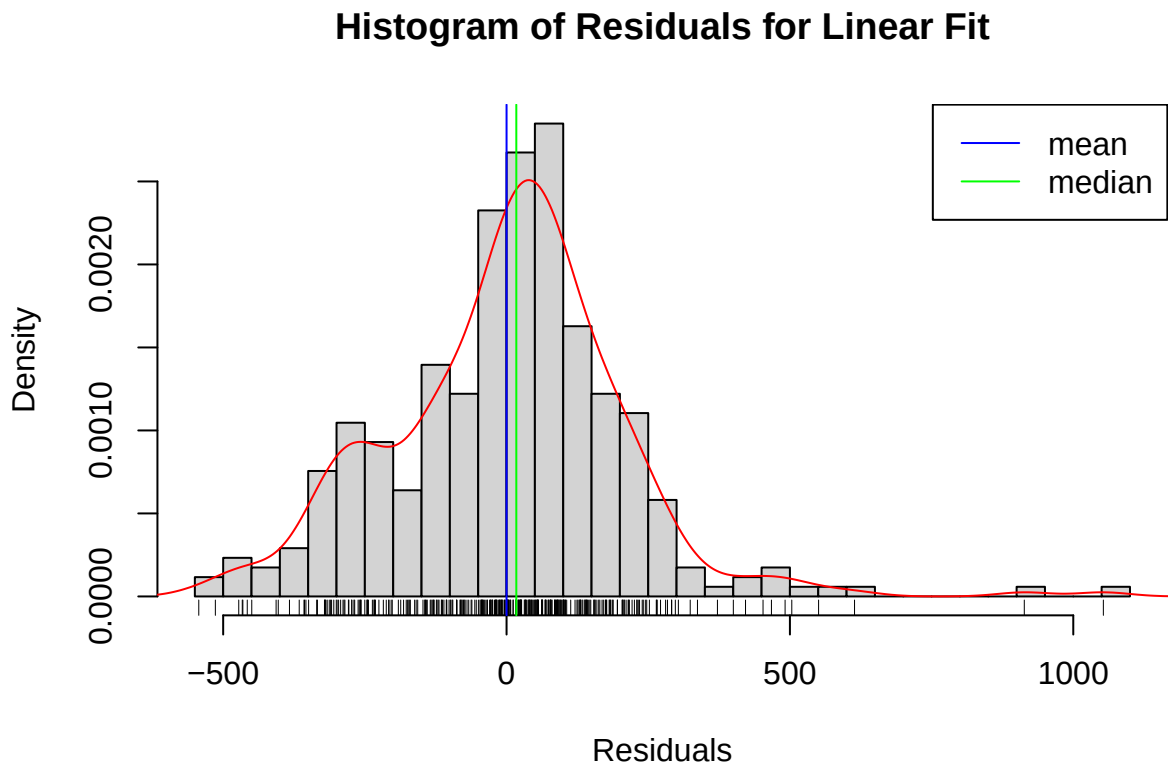

Residuals vs Time for Model 1
Residuals
1000
500
0
−500
1995
2000
2005
2010
2015
2020
Time



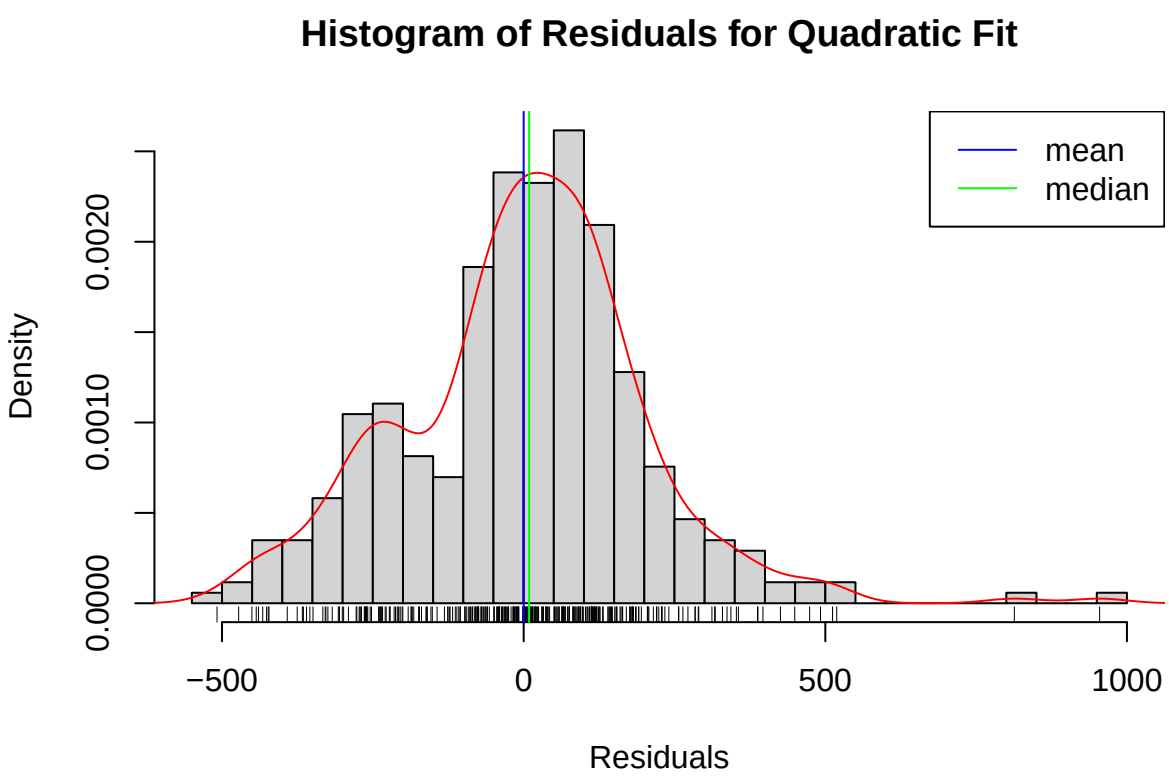
As can be seen in the first two graphs above, the residuals have been plotted against the fitted values for each model that was developed and chosen based on visual representation in the previous part. The red line in the plots is the mean of the residual which is generated using ablibne. The mean is close to zero and hence the command, h = 0 was carried out inside the abline. That line is usually made since it can be used as a benchmark comparison to see how residuals behave. I noticed that there is still some structure/pattern remaining in the residuals. One can incorporate this structure into the model to improve the fit. Adding seasonality would likely help make the residuals more random (i.e. white noise). After applying the model fits, the residuals no longer have an increasing trend and the mean is more constant over time. Although it appears so for the most part of it, but there is some volatility at higher fitted values for year 2020 which could be probably owing to surge in sales due to pandemic. The adf and Kpss test suggest covariance stationary residuals.

(f) For each model, plot a histogram of the residuals and discuss your observations.

```
#Plot histogram for linear fit
hist(model1$residuals,breaks = "FD",xlab="Residuals",
main="Histogram of Residuals for Linear Fit",probability = TRUE)
lines(density(model1$residuals),col="red")
rug(model1$residuals)
abline(v=mean(model1$residuals),col="blue")
abline(v=median(model1$residuals),col="green")
legend("topright",c("mean","median"),col=c("blue","green"),lty=c(1,1))
```




```
#Plot histogram for quadratic fit
hist(model2$residuals,breaks = "FD",xlab="Residuals",
main="Histogram of Residuals for Quadratic Fit",probability = TRUE)
lines(density(model2$residuals),col="red")
rug(model2$residuals)
abline(v=mean(model2$residuals),col="blue")
abline(v=median(model2$residuals),col="green")
legend("topright",c("mean","median"),col=c("blue","green"),lty=c(1,1))
```



The histograms are plotted using the hist function in R from MASS library. The code as used above is straightforward and easy to understand. The histogram of residuals for both the linear and quadratic fits are fairly normal, for the most part of it. This is evident form the bell shaped curve. To confirm, I plotted the mean and the median which are fairly close together. This shows that the residuals seem to be normal. This is also suggested by the rug-plots and skewness of the histograms. As the mean and median are closer together in the quadratic fit, this also indicates that the residuals for the quadratic fit are slightly more normally distributed. From this one can say that the quadratic trend seems to be a better fit for the model. We also see that the mean (blue) is approximately zero for both the linear and quadratic fits, indicating that the forecast will be unbiased.

(g) For each model, discuss the associated diagnostic statistics (R2,t−distribution,F−distribution,etc.)

```
#stargazer(model1, model2, type = "text")
summary(model1)
```

```
##
## Call:
## lm(formula = hardware ~ t, data = hardware)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -543.05 -121.25   17.25  104.66 1053.06
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -84954.338   2743.481  -30.97   <2e-16 ***
## t               43.138      1.368   31.54   <2e-16 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205.6 on 342 degrees of freedom
## Multiple R-squared:  0.7442, Adjusted R-squared:  0.7434
## F-statistic: 994.9 on 1 and 342 DF,  p-value: < 2.2e-16
```

```
summary(model2)
```

```
##
## Call:
## lm(formula = hardware ~ t + I(t^2), data = hardware)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -508.25 -106.07    9.06  113.72  954.66
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.122e+06  7.400e+05   4.218 3.16e-05 ***
## t           -3.154e+03  7.378e+02  -4.275 2.49e-05 ***
## I(t^2)       7.969e-01  1.839e-01   4.333 1.94e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.5 on 341 degrees of freedom
## Multiple R-squared:  0.7575, Adjusted R-squared:  0.7561
## F-statistic: 532.7 on 2 and 341 DF,  p-value: < 2.2e-16
```

The R^2 and Adjusted R^2 for both model 1 and model 2 are almost the same. The R^2 for model 1 is 0.744 and for model 2 is 0.758. The adjusted R^2 for model 1 is 0.743 and for model 2 is 0.756. This indicates that the model is a good fit. But we know that we can still improve our fit by incorporating seasonality. The coefficients for both the models are statistically significant at 1% level of significance. This individual significance of variables can be seen via their t statistic using the t-distribution. We could also look straight at their p-values, each of them being small and suggesting that they indeed are statistically significant. The F-statistic is large for both models, 994.914 & 532.7 respectively and the small p-value suggests that there is overall significance of the model and that we should keep all the variables.

(h) Select a trend model using AIC and one using BIC (show the values obtained from each criterion). Do the selected models agree?

AIC is an estimate of a constant plus the relative distance between the unknown true likelihood function of the data and the fitted likelihood function of the model, so that a lower AIC means a model is considered to be closer to the truth. BIC is an estimate of a function of the posterior probability of a model being true, under a certain Bayesian setup, so that a lower BIC means that a model is considered to be more likely to be the true model.

```
#creating a data frame to comopare the values of AIC and BIC for both models
data.frame(AIC(model1, model2),BIC(model1, model2))
```

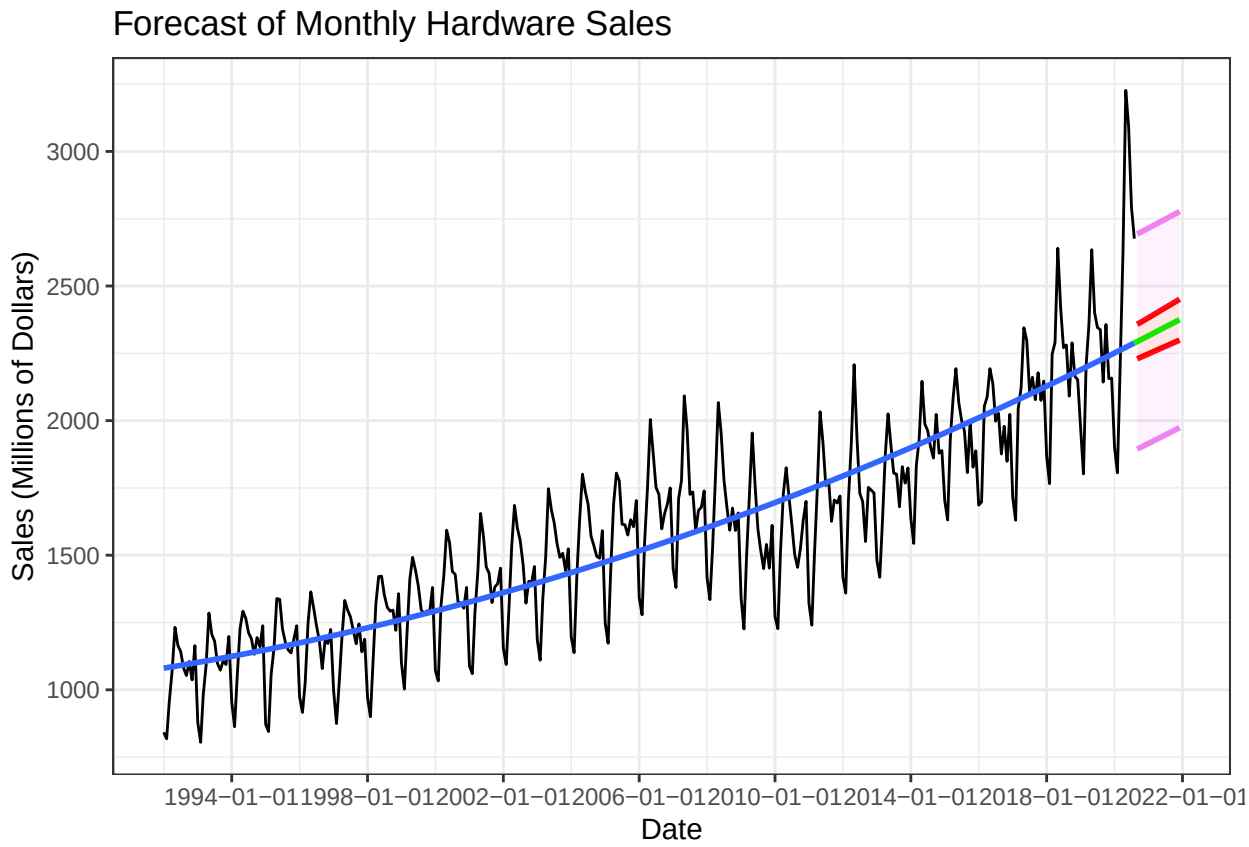
```
##        df      AIC df.1      BIC
## model1  3 4644.550    3 4656.072
## model2  4 4628.113    4 4643.476
```

Both AIC and BIC suggest that the quadratic fit is better, so we go ahead with it.

(i) Use your preferred model to forecast h-steps (at least 16) ahead. Your forecast should include the respective uncertainty prediction interval. Depending on your data, h will be in days, months, years, etc.

```
#create new hardware data frame and time dummy
hardware2 = Hardware
colnames(hardware2)[2] = "sales"
hardware2$t = 1:nrow(hardware2)
m4 = lm(sales ~ t + I(t^2), hardware2)
t2 = 1:length(hardware)

#solve for prediction intervals
tn2 = data.frame(t = 345:(345+15))
pred_p = predict(m4, tn2, level = 0.95, interval="prediction")
pred_c = predict(m4, tn2, level = 0.95, interval="confidence")

#create separate data frames to store original and forecasted data
known = data.frame(t = 1:344)
known$fc = NA
#store original data
for(i in 1:nrow(known)){
known[i, "fc"] = hardware[i]
}
#store forecasted values
new = cbind(data.frame(pred_c), data.frame(pred_p))
new = new[-4]
colnames(new) = c("fit", "clwr", "cupr", "plwr", "pupr")
new$t = 1:nrow(new)
new$t = new$t + 344
temp = data.frame(fit = model2$fitted.values[344], clwr = NA, cupr = NA, plwr = NA, pupr = NA,
t = 344) #connect forecast to original data for plot
new = rbind(temp, new)
#create date for x-axis
known$year = c(rep(1992, 12),rep(1993, 12),rep(1994, 12),rep(1995, 12), rep(1996, 12),
rep(1997, 12), rep(1998, 12),
rep(1999, 12), rep(2000, 12), rep(2001, 12), rep(2002, 12), rep(2003, 12),
```

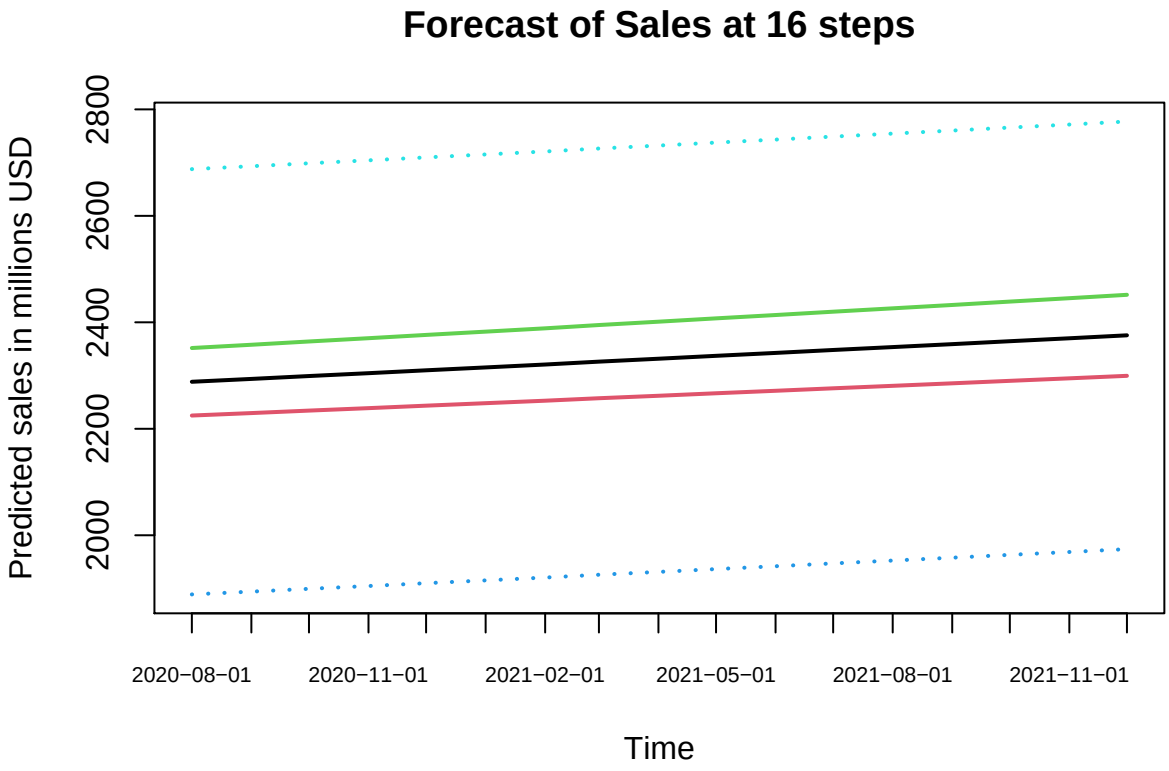
```
rep(2004, 12), rep(2005, 12),
rep(2006, 12), rep(2007, 12), rep(2008, 12), rep(2009, 12), rep(2010, 12),
rep(2011, 12), rep(2012, 12),
rep(2013, 12), rep(2014, 12), rep(2015, 12), rep(2016, 12), rep(2017, 12),
rep(2018, 12), rep(2019,12), rep(2020,8))

known$month = known$t %% 12
known[known$month == 0, "month"] = 12
known$day = rep(1, 344)
known$date = paste(as.character(known$year),
as.character(known$month), as.character(known$day), sep = "-")
known$date = as.Date(known$date)
new$year = c(rep(2020, 5), rep(2021, 12))
new$month = new$t %% 12
new[new$month == 0, "month"] = 12
new$day = rep(1, 17)
new$date = paste(as.character(new$year),
as.character(new$month), as.character(new$day), sep = "-")
new$date = as.Date(new$date)
#plot
ggplot()+
geom_line(data = known, aes(x = date, y = fc))+
geom_smooth(data = known, aes(x = date, y = fc), method = "lm",
formula = y ~ x + I(x^2), se = FALSE)+
geom_line(data = new, aes(x = date, y = fit), color = "green", size = 1)+
geom_line(data = new, aes(x = date, y = clwr), color = "red", size = 1)+
geom_line(data = new, aes(x = date, y = cupr), color = "red", size = 1)+
geom_line(data = new, aes(x = date, y = plwr), color = "violet", size = 1)+
geom_line(data = new, aes(x = date, y = pupr), color = "violet", size = 1)+
geom_ribbon(data = new, aes(x = date, y = fit, ymin = clwr, ymax = cupr),
fill = "red", alpha = 0.1)+
geom_ribbon(data = new, aes(x = date, y = fit, ymin = cupr, ymax = pupr),
fill = "violet", alpha = 0.1)+
geom_ribbon(data = new, aes(x = date, y = fit, ymin = plwr, ymax = clwr),
fill = "violet", alpha = 0.1)+
theme_bw()+
labs(
title = "Forecast of Monthly Hardware Sales",
x = "Date",
y = "Sales (Millions of Dollars)"
)+
scale_x_date(breaks = date_breaks("4 years"))
```



I carried out a 16 step forecast using the quadratic trend model and since I have data till August 2020, I carried forecast till the month of December 2021. We first create a new data frame called hardware2 and a time dummy. We redefine the trend model and save it as m4. We then solve for the prediction intervals. This is done by using the "predict" function and then the argument "interval" defines whether a prediction or a confidence interval is to be produced. As seen above, separate data frames are created for original and forecasted data and the values are stored in the respective data frames. Then both are connected for plot. Then, an essential step was to compute the date variables for the x - axis and this was accomplished using the "as.date" function. The dates got saved in the date column. Finally, last set of codes involved plotting the data and fitting the trend model and making a forecast.The ggplot package was used to carry out the same. The confidence intervals saved under "cupr" and "clwr" are represented by the red lines and prediction intervals saved under "pupr" and "plwr", which appear to be wider as expected are shown by the violet lines. The forecast is shown by the green line. The model forecasts the value at 16th step to be 2375.167 USD.

```
#The following codes are for making an additional graph showing just the
#Forecast of the sales at 16-steps using the quadratic trend model.
#creating data frame
Forecast = data.frame(t = seq(2020.000, 2021.312, by =0.082))

#Confidence and Prediction Intervals
Prediction = predict(lm(hardware ~ t + I(t^2)), Forecast, se.fit = TRUE)
pred.plim = predict(lm(hardware ~ t + I(t^2)), Forecast, level = 0.95,
                    interval ="prediction")
pred.clim = predict(lm(hardware ~ t + I(t^2)), Forecast, level = 0.95,
                    interval ="confidence")

#Plot
```

```r
matplot(new$date, cbind(pred.clim, pred.plim[,-1]), lty = c(1,1,1,3,3),
             type = "l", lwd =2,
             ylab = "Predicted sales in millions USD",
             xlab = "Time",
             main = "Forecast of Sales at 16 steps", xaxt="n")
axis(1, new$date, format(new$date), "%b %d", cex.axis = 0.7)
```



The above graph is an additional graph showing just the Forecast of the sales at 16-steps using the quadratic trend model. The dotted blue lines show the prediction interval, the green and red lines show the confidence interval and black line shows the forecast.

2. Modeling and Forecasting Seasonality

(a) Construct and test (by looking at the diagnostic statistics) a model with a full set of seasonal dummies.

```r
fit = tslm(hardware~season) #model with a full set of seasonal dummies
summary(fit)
```

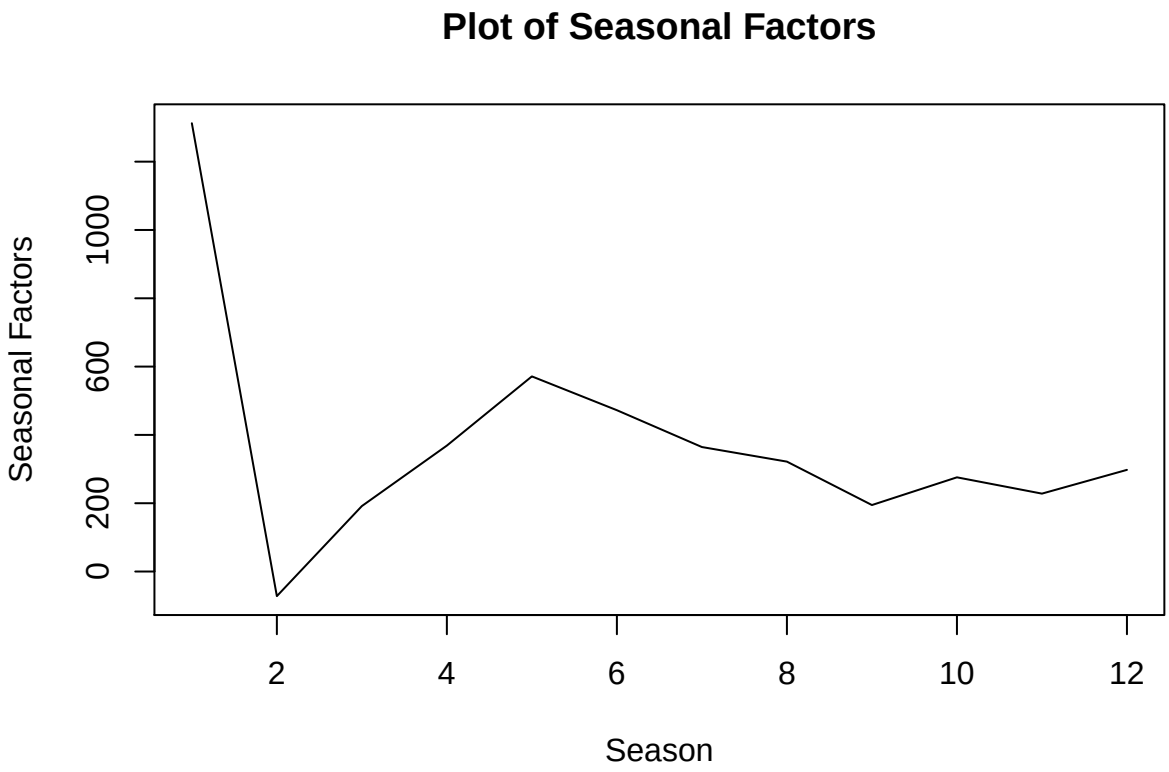
```
##
## Call:
## tslm(formula = hardware ~ season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -651.79 -293.04  -17.16  203.12 1343.21
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1312.48      69.25  18.952  < 2e-16 ***
## season2       -72.00      97.94  -0.735 0.462763
## season3       191.66      97.94   1.957 0.051197 .
## season4       368.66      97.94   3.764 0.000198 ***
## season5       571.31      97.94   5.833 1.29e-08 ***
## season6       472.31      97.94   4.823 2.16e-06 ***
## season7       364.45      97.94   3.721 0.000233 ***
## season8       321.79      97.94   3.286 0.001126 **
## season9       194.70      98.81   1.970 0.049620 *
## season10      275.80      98.81   2.791 0.005554 **
## season11      228.09      98.81   2.308 0.021592 *
## season12      297.66      98.81   3.012 0.002790 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 372.9 on 332 degrees of freedom
## Multiple R-squared:  0.1831, Adjusted R-squared:  0.1561
## F-statistic: 6.766 on 11 and 332 DF,  p-value: 2.868e-10
```

The adjusted R^2 is much smaller than that of the quadratic fit. This means that most of the model was explained by the trend and not by seasonality alone. The p-value associated with the F-statistic is small so we should keep all of the seasons. I noticed that the most significant seasons are the summer months. Additionally, the coefficients for the summer months are larger. Intuitively this makes sense as One would expect hardware store sales to peak during these months due to construction and outdoor projects.

(b) Plot the estimated seasonal factors and interpret your plot.

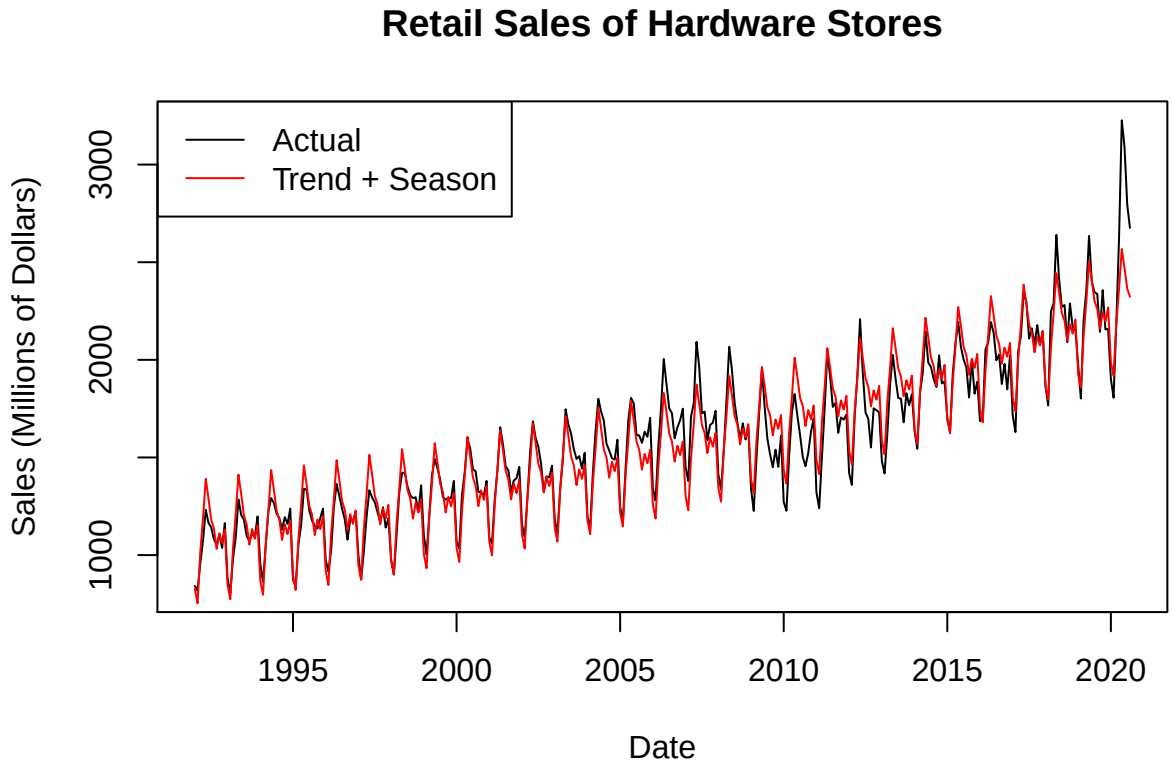
```
#Plot estimated seasonal factors
plot(fit$coefficients,type='l',ylab="Seasonal Factors",xlab="Season",
    main="Plot of Seasonal Factors")
```



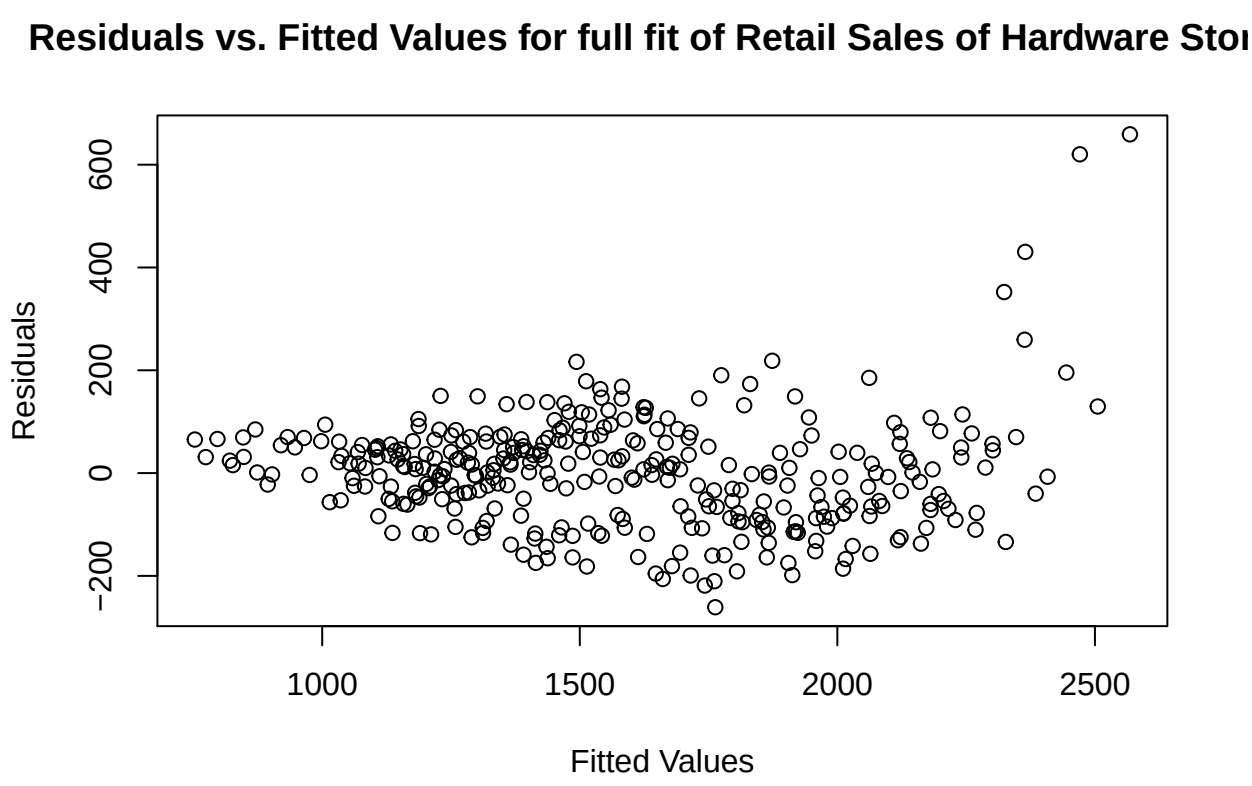
It can be seen that the summer months have higher sales while the winter and fall months have lower hardware sales. This is what was expected, however the fall and winter sales still aren't as low as expected. This is probably due to a continuous need for hardware store products in those months as well.

(c)In order to improve your model, add the trend model from problem 1 to your seasonal model. We will refer to this model as the full model. For the full model, plot the respective residuals vs. fitted values and discuss your observations.

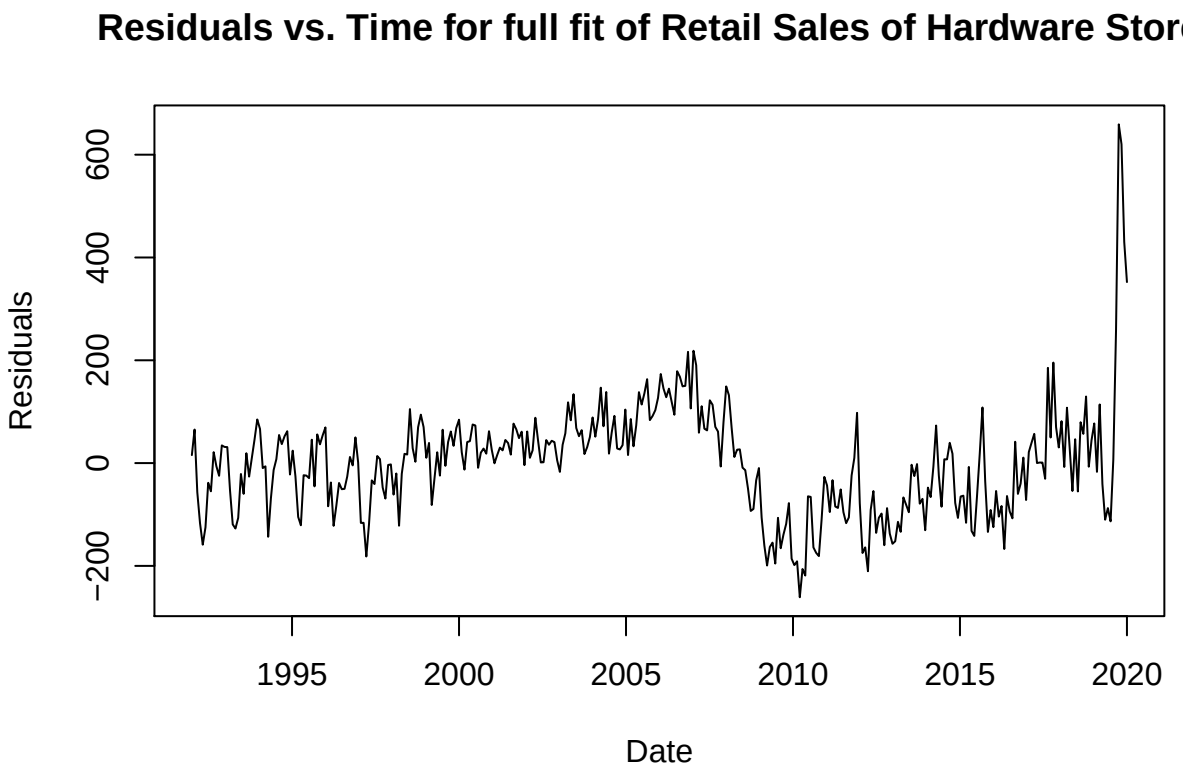
```
#create full model with season and trend and call it fit.full
fit.full = tslm(hardware~t+I(t^2)+season)

#plot the full fit onto original data
plot(hardware,ylab="Sales (Millions of Dollars)", xlab = "Date",
main = "Retail Sales of Hardware Stores")
lines(fit.full$fitted.values,col="red")
legend("topleft",c("Actual","Trend + Season"),col=c("black","red"),lty = c(1,1))
```




```
#plot residuals vs. fitted values for full fit of hardware stores sales
plot(fit.full$fitted.values,fit.full$residuals, ylab= "Residuals",
xlab = "Fitted Values",
main="Residuals vs. Fitted Values for full fit of Retail Sales of Hardware Stores")
```




```
#plot residuals vs. time for full fit of hardware stores sales
plot(t,fit.full$residuals,type='l',ylab="Residuals", xlab = "Date",
main = "Residuals vs. Time for full fit of Retail Sales of Hardware Stores")
```




```
kpss.test(fit.full$residuals)
```

```
##
##  KPSS Test for Level Stationarity
##
## data:  fit.full$residuals
## KPSS Level = 0.35887, Truncation lag parameter = 5, p-value = 0.09488
```

```
adf.test(fit.full$residuals) #suggests not covariance stationary
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  fit.full$residuals
## Dickey-Fuller = -1.8982, Lag order = 6, p-value = 0.6197
## alternative hypothesis: stationary
```

The first plot shows the original time series with our fitted model, which includes a trend and seasonal component. It shows that our model fits the data well. The second plot shows residuals versus fitted observations in a scatterplot. I noticed that the variance of the residuals is increasing for larger fitted values meaning that the model doesn't estimate well for larger values of sales. The third plot shows residuals versus time. A similar pattern was noticed with the variance of the residuals increasing with time. This indicates that our model was unable to estimate the drop in sales around 2009 (recession) because we were using a quadratic trend. A spike in the residuals was seen in 2020 which can be explained by the surge in the sales

during Covid-19 since people didn't go out and spend much on recreational activities and instead stayed and spent money on sprucing up their homes.This also shows that our model was unable to estimate the surge. As for the stationarity of the residuals, graphically as well as by conducting adf and kpss tests, we can state they they aren't covariance stationary.

(d) Interpret the respective summary statistics including the error metrics of your full model.

```
summary(fit.full) #summary statistics for full model
```

```
##
## Call:
## tslm(formula = hardware ~ t + I(t^2) + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -261.01  -68.91    4.04   59.15  659.03
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  3.096e+06  4.033e+05   7.676 1.87e-13 ***
## t           -3.128e+03  4.021e+02  -7.779 9.42e-14 ***
## I(t^2)       7.904e-01  1.002e-01   7.886 4.61e-14 ***
## season2     -7.547e+01  2.867e+01  -2.632  0.00889 **
## season3      1.847e+02  2.867e+01   6.442 4.17e-10 ***
## season4      3.582e+02  2.867e+01  12.493  < 2e-16 ***
## season5      5.574e+02  2.867e+01  19.438  < 2e-16 ***
## season6      4.549e+02  2.867e+01  15.863  < 2e-16 ***
## season7      3.435e+02  2.868e+01  11.979  < 2e-16 ***
## season8      2.973e+02  2.868e+01  10.368  < 2e-16 ***
## season9      1.914e+02  2.893e+01   6.614 1.51e-10 ***
## season10     2.690e+02  2.893e+01   9.297  < 2e-16 ***
## season11     2.178e+02  2.893e+01   7.526 5.01e-13 ***
## season12     2.838e+02  2.893e+01   9.810  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 109.2 on 330 degrees of freedom
## Multiple R-squared:  0.9304, Adjusted R-squared:  0.9277
## F-statistic: 339.4 on 13 and 330 DF,  p-value: < 2.2e-16
```

```
#error metrics
library(forecast)
accuracy(fit.full)
```

```
##                        ME     RMSE      MAE        MPE     MAPE     MASE
## Training set -1.452456e-14 106.9395 77.98366 -0.2568624 4.860879 1.054091
##                    ACF1
## Training set 0.7838178
```

From the summary table, we see that all the coefficients for trend and seasonal components are statistically significant. This is likely due to the model displaying a heavy seasonal component, which is why the coefficients for the seasons are so large. The adjusted R^2 is 0.9277 which is significantly higher than when we just had the trend or seasonal components alone. The F-stat, 339.4 is also significant because its p-value

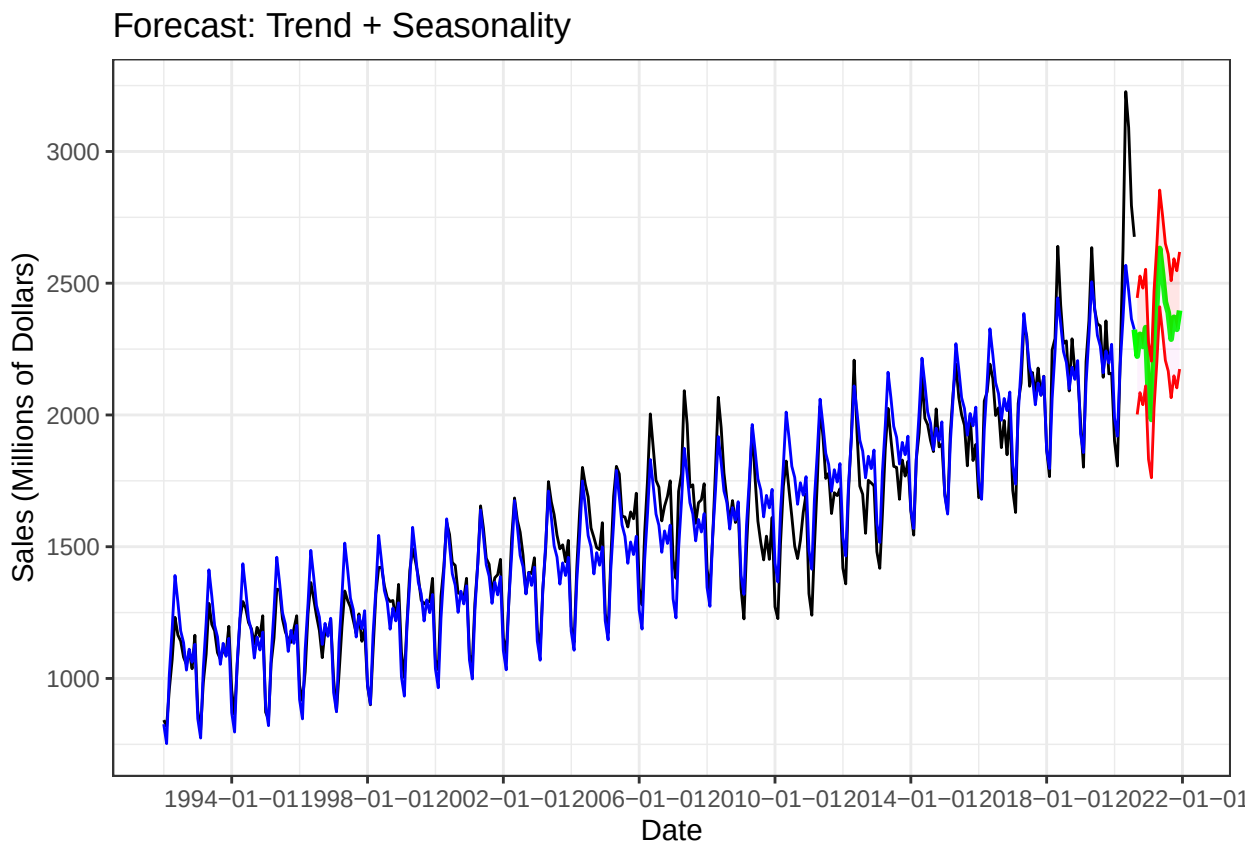
is low, therefore we should include all trend and seasonal components. As for error metrics, we want to see how well a model does in terms of how accurate the model is when using a training set. For getting the error metrics, I used the accuracy function from the forecast library as shown above. A ME value very close to zero was derived, which measures the average of all the errors in the model and the smaller the value, the better it is. RMSE of 106.9395 was derived which seems to be quite high. Considering the high R^2 of the model, there might have been overfitting. A MAPE of 4.860879 was obtained which measures how accurate a forecast model is and the smaller the number due to the errors being minimized, the better the model is at prediction. A MASE of around 1 was derived. It measures the mean absolute error produced by the forecast. The lower the MASE, the better because a higher value would suggest that the actual forecast would do worst out of the sample.

(e) Use the full model to forecast h-steps (at least 16) ahead. Your forecast should include the respective prediction interval.

```
#re-define time dummy
t2 = data.frame(t = 1:(nrow(Hardware) + 16))
t2$month = t2$t %% 12
t2[t2$month == 0, "month"] = 12
t2[1:344, "val"] = Hardware$retail_sales

#fit model using trend and create seasonal dummies
fit2 = lm(val ~ t + I(t^2) + factor(month), t2[1:344, ])
t2[1:344, "fit"] = fit2$fitted.values

t2$season1 = 0
t2$season2 = 0
t2$season3 = 0
t2$season4 = 0
t2$season5 = 0
t2$season6 = 0
t2$season7 = 0
t2$season8 = 0
t2$season9 = 0
t2$season10 = 0
t2$season11 = 0
t2$season12 = 0

t2[t2$month == 1, "season1"] = 1
t2[t2$month == 2, "season2"] = 1
t2[t2$month == 3, "season3"] = 1
t2[t2$month == 4, "season4"] = 1
t2[t2$month == 5, "season5"] = 1
t2[t2$month == 6, "season6"] = 1
t2[t2$month == 7, "season7"] = 1
t2[t2$month == 8, "season8"] = 1
t2[t2$month == 9, "season9"] = 1
t2[t2$month == 10, "season10"] = 1
t2[t2$month == 11, "season11"] = 1
t2[t2$month == 12, "season12"] = 1

#using the model to calculate the fit value for each row
for(i in 345:nrow(t2)){
t2[i, "fit"] =
fit2$coefficients[1]+
fit2$coefficients[2]*t2[i, "t"]+
```

```
fit2$coefficients[3]*(t2[i, "t"]^2)+
fit2$coefficients[4]*t2[i, "season2"]+
fit2$coefficients[5]*t2[i, "season3"]+
fit2$coefficients[6]*t2[i, "season4"]+
fit2$coefficients[7]*t2[i, "season5"]+
fit2$coefficients[8]*t2[i, "season6"]+
fit2$coefficients[9]*t2[i, "season7"]+
fit2$coefficients[10]*t2[i, "season8"]+
fit2$coefficients[11]*t2[i, "season9"]+
fit2$coefficients[12]*t2[i, "season10"]+
fit2$coefficients[13]*t2[i, "season11"]+
fit2$coefficients[14]*t2[i, "season12"]
}

#get prediction intervals
prediction = predict(fit2, t2, interval="predict")

t2$lwr = NA
t2$upr = NA

t2[345:nrow(t2), "lwr"] = prediction[ 345:nrow(t2), 2]
t2[345:nrow(t2), "upr"] = prediction[ 345:nrow(t2), 3]

#creating date variable for x-axis
t2$day = 1
t2$year = NA
t2[345:nrow(t2), "year"] = c(rep(2020, 4), rep(2021, 12))

t2$date = NA
t2[1:344, "date"] = as.character(Hardware$observation_date)
t2[345:nrow(t2), "date"] = paste(as.character(t2[345:nrow(t2), "day"]),
                                 as.character(t2[345:nrow(t2), "month"]),
                                 as.character(t2[345:nrow(t2), "year"]),
                                 sep = "-")

t2$date = as.Date(t2$date, format = "%d-%m-%Y")

#plot
ggplot()+
geom_line(data = t2[1:344,], mapping = aes(x = date, y = val))+
geom_line(data = t2[1:344,], mapping = aes(x = date, y = fit), color = "blue")+
geom_line(data = t2[344:nrow(t2), ], mapping = aes(x = date, y = fit),
color = "green", size = 1)+
geom_line(data = t2[344:nrow(t2), ], mapping = aes(x = date, y = upr), color = "red" )+
geom_line(data = t2[344:nrow(t2), ], mapping = aes(x = date, y = lwr), color = "red")+
theme_bw()+
geom_ribbon(data = t2[344:nrow(t2), ],
            aes(x = date, y = fit, ymin = fit, ymax = upr), fill = "red",
            alpha = 0.1)+
geom_ribbon(data = t2[344:nrow(t2), ],
            aes(x = date, y = fit, ymin = lwr, ymax = fit),
            fill = "violet", alpha = 0.1)+
theme_bw()+
```

```
labs(
title = "Forecast: Trend + Seasonality",
x = "Date",
y = "Sales (Millions of Dollars)"
)+
scale_x_date(breaks = date_breaks("4 years"))
```

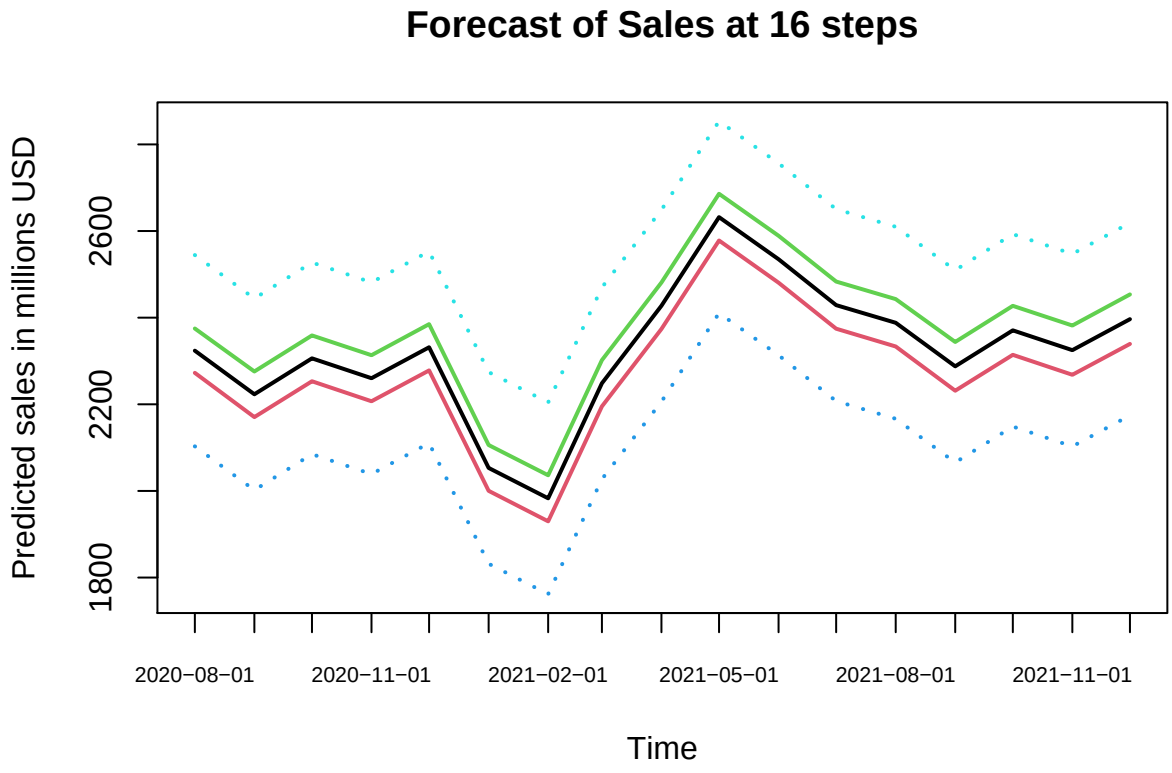


I carried out the 16 step forecast for the retail sales of hardware stores using the trend and seasonality fit model. So i forecasted till December 2021. In the first step, we redefine the time dummy and initially save the values of sales in the data frame, t2. Then a model is fitted using quadratic trend and an additional column for the fitted values is added in the t2 data frame. Then I created 12 seasonal dummies since the frequency of my data is 12, that is, it is monthly. In the next step, using the loop command, I used the model to calculate the fit value for each row. The prediction intervals were computed and saved under "lwr" and "upr" for lower limit and upper limit of the interval respectively.This is represented by the red line. Then, an essential step was to compute the date variables for the x - axis and this was accomplished using "as.date" function. The dates got saved in the dates column. Finally, last set of codes involved plotting the data and fitting the trend and seasonality model and making a forecast.The ggplot package was used to carry out the same. The forecast (as shown by the green line) is in alignment with our intuition. The values see a dip in winter months, especially January and February and again a rise in the summer months. The model forecasts the value at 16th step to be 2396.506 USD.

```
#The following codes are for making an additional graph showing just the
#Forecast of the sales at 16-steps using the trend and seasonality fit model.
#Subset of t2
tk = t2[344:nrow(t2), ]
```

```r
#calculating the intervals
prediction = predict(fit2, tk, interval="predict")
pred.plim = predict(fit2, tk, level = 0.95,
                    interval ="prediction")
pred.clim = predict(fit2, tk, level = 0.95,
                    interval ="confidence")
```

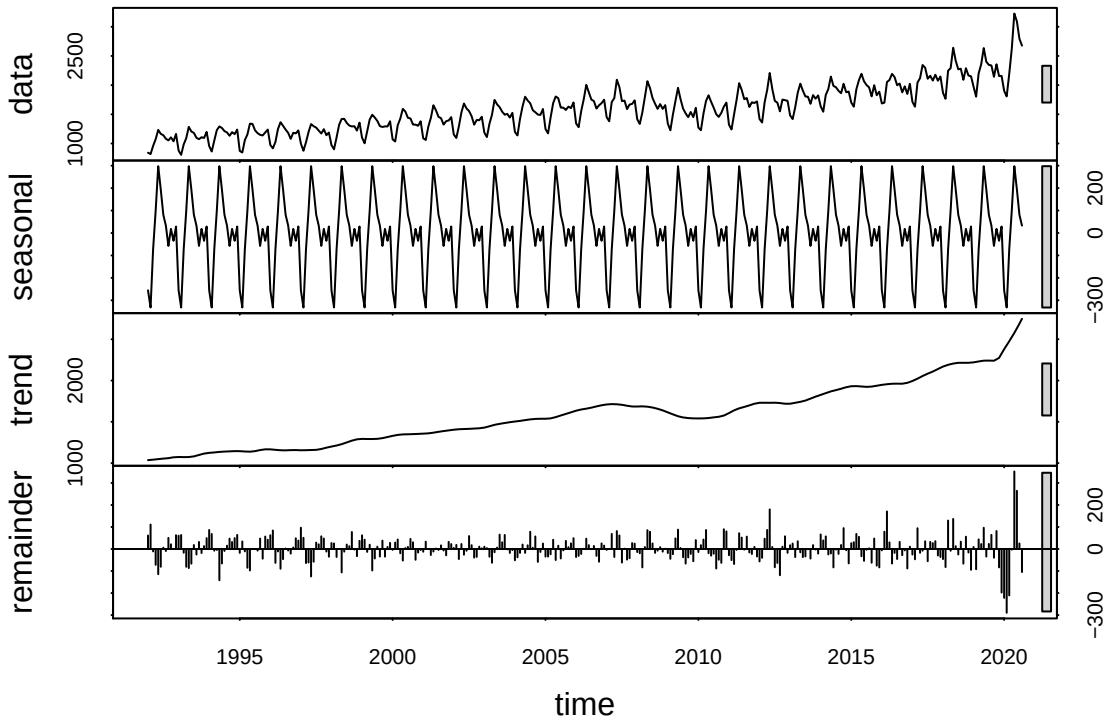
```r
#plot
plot.new()
matplot(tk$date, cbind(pred.clim, pred.plim[,-1]), lty = c(1,1,1,3,3),
             type = "l", lwd =2, ylab = "Predicted sales in millions USD",
              xlab = "Time",
              main = "Forecast of Sales at 16 steps", xaxt = "n")
axis(1, tk$date, format(tk$date), "%b %d", cex.axis = 0.7)
```



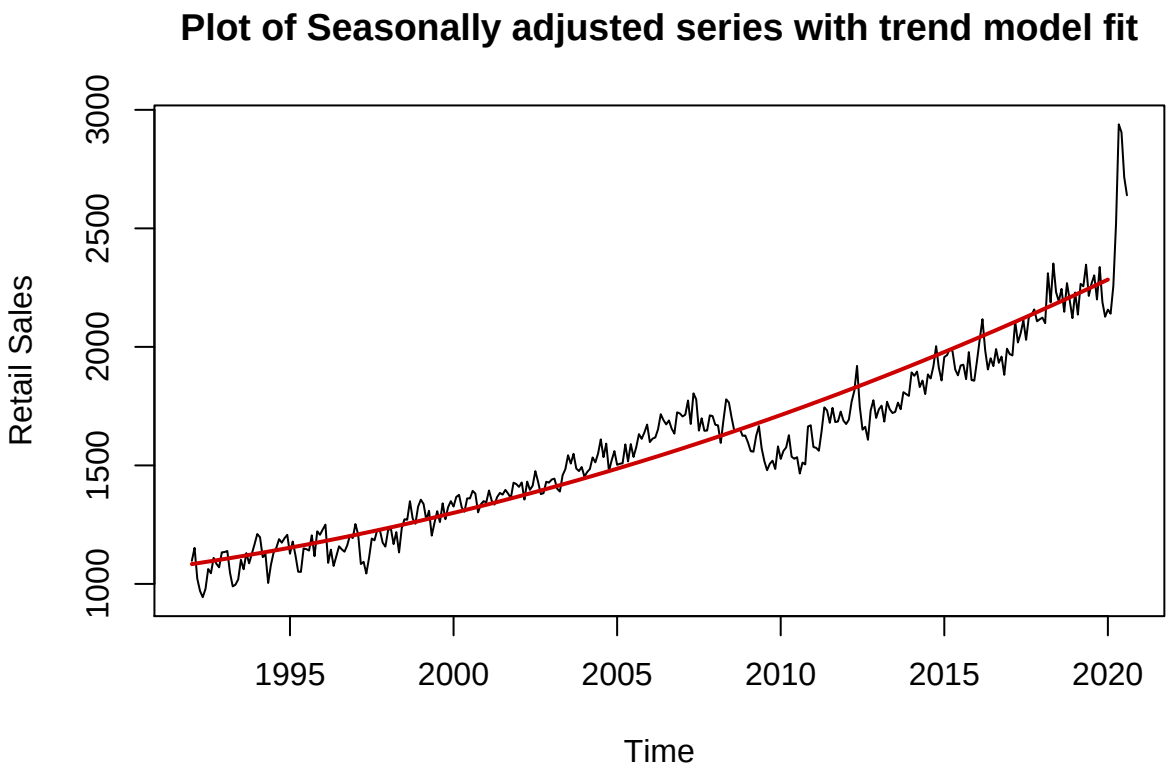
The above graph is an additional graph showing just the Forecast of the sales at 16-steps using the full model (trend and seasonality fit model). The dotted blue lines show the prediction interval, the green and red lines show the confidence interval and black line shows the forecast. The forecast suggests a drop in sales in the month of February, 2021 and rise in summer months and peak in May. It again shows a decline especially in fall.

(f) Plot the STL decomposition plot, and based on it, choose between an additive and multiplicative seasonal adjustment. Perform the correction, and plot the seasonally adjusted series. Based on this adjustment, would your trend model from problem 1 still be appropriate? Explain your answer in detail.

```
#Plotting the STL plot to see if we want to additive or multiplicative
plot(stl(hardware,s.window="periodic"))
```




```
#Carrying out seasonal adjustment
tsc <- decompose(hardware, "additive")
tscseasonallyadjusted <- hardware - tsc$seasonal

#fitting trend model to seasonally adjusted data
mod.2 <- lm(tscseasonallyadjusted ~ t + I(t^2))

#plot
plot(tscseasonallyadjusted, xlab = "Time", ylab = "Retail Sales",
     main = "Plot of Seasonally adjusted series with trend model fit")
lines(t,mod.2$fit,col="red3",lwd=2)
```




```
#Comparison
stargazer(model2, mod.2, type = "text")
```

```
##
## ===============================================================
##                                         Dependent variable:
##                                     ---------------------------
##                                        hardware     tscseasonallyadjusted
##                                          (1)                (2)
## ---------------------------------------------------------------
## t                                    -3,153.864***       -3,139.684***
##                                       (737.832)           (396.201)
##
## I(t2)                                   0.797***           0.793***
##                                         (0.184)            (0.099)
##
## Constant                            3,121,586.000***   3,107,639.000***
##                                     (740,037.200)      (397,385.300)
##
## ---------------------------------------------------------------
## Observations                             344                344
## R2                                       0.758              0.915
## Adjusted R2                              0.756              0.914
## Residual Std. Error (df = 341)         200.482            107.655
## F Statistic (df = 2; 341)            532.699***         1,824.084***
```

```
## ================================================================
## Note:                                    *p<0.1; **p<0.05; ***p<0.01
```

I first plot the STL plot for my time series data in order to see what adjustment I should go ahead with. As can be seen the STL plot represents the trend and seasonality of the data separately. After looking at the seasonality component of the STL plot, I can see that the seasonal fluctuations are not varying with time. Hence, the "additive" seasonal adjustment would be the right choice given my data. So I perform the correction by subtracting the seasonal component form the series and plot the seasonally adjusted time series. As for the remaining part of the question, I fitted the model2 (quadratic trend model from question one) onto the seasonally adjusted data and it seems to fit really well. As a sanity check, I even ran a regression of the quadratic model using the seasonally adjusted data , which I refer to as mod.2. Note that the y variable is that of seasonally adjusted hardware sales. As can be seen from the table, the R^2 and adjusted R^2 also improve tremendously. So, I would conclude that I would use trend model even after seasonal adjustment has been carried out or in other words, it is still appropriate.

Part III.

III. Conclusions and Future Work (state your conclusion regarding your final model and forecast, and provide some insight as to how it could be improved).

Our final model that incorporates both seasonality and trend is a good fit given our high adjusted R^2. Including both trend and seasonality was a good option as seasonality alone or trend alone did not fit data so well as compared to the final model. The model with a full set of seasonal dummies had a very low R^2 suggesting that seasonality alone can't explain the entire model. On the other hand, considering the quadratic trend model had a higher R^2, most of the model was explained by trend and not seasonality. Also, even on seasonally adjusted data, the trend model would be appropriate as was seen in the last part of question 2. The forecast shows the same result as we would expect, rise in sales in summer months and lower sale in winter and fall. The model forecasts the value at 16th step to be 2396.506 USD.However, the model becomes less accurate after the 2009 great recession. These drop in sales are not captured by our model. Another thing that our model could not capture was a spike in sales seen in 2020. The reasons for the same could be that we used a quadratic trend model. If we were able to capture the drop in sales during the great recession and spike in sales during 2020 pandemic, we could significantly improve both our model and our forecast. So in retrospect, one could go with choosing a model that would capture both.

Part IV.

IV. References (include the source of your data and any other resources).

[1] FRED Economic data on Retail Sales: Hardware Stores (MRTSSM44413USN).2020

fred.stlouisfed.org/series/MRTSSM44413USN. Accessed November 15 2020

[2] Jensen, C. Special Report: COVID-19 Update - Hardware Retailers Are Thriving During the Pandemic 2020 https://thehardwareconnection.com/special-report-covid-19-update-hardware-retailers-are-thriving-during-the-pandemic/. Accessed November 29 2020

[3] Online resources for R: https://a-little-book-of-r-for-time-series.readthedocs.io/en/latest/src/timeseries.html

[4] Hill,R.C., Griffith,W.E., and Lim,G.C. 2018. The Principles of Econometrics. John Wiley & Sons, New Jersey,pp. : 417-480.

Part V.

V. R/Python Source code. - As an be seen throughout the assignment codes with detailed explanation have been provided for each part.