Tractor Sales Forecasting

Himank Jain

17/02/2020

Buiseness Problem:

PowerHorse, a tractor and farm equipment manufacturing company, was established a few years after World War II. The company has shown a consistent growth in its revenue from tractor sales since its inception. However, over the years the company has struggled to keep it's inventory and production cost down because of variability in sales and tractor demand. The management at PowerHorse is under enormous pressure from the shareholders and board to reduce the production cost.

Develop models to forecast next 2 years sales.

Data:

The dataset consists of 144 observations having the total monthwise sales data of Tractors for a period of past 12 years.

Loading Required Libraries:

```
library(data.table)
library(ggplot2)
library(fpp2)
library(forecast)
library(stats)
library(tseries)
```

Exploratory Analysis:

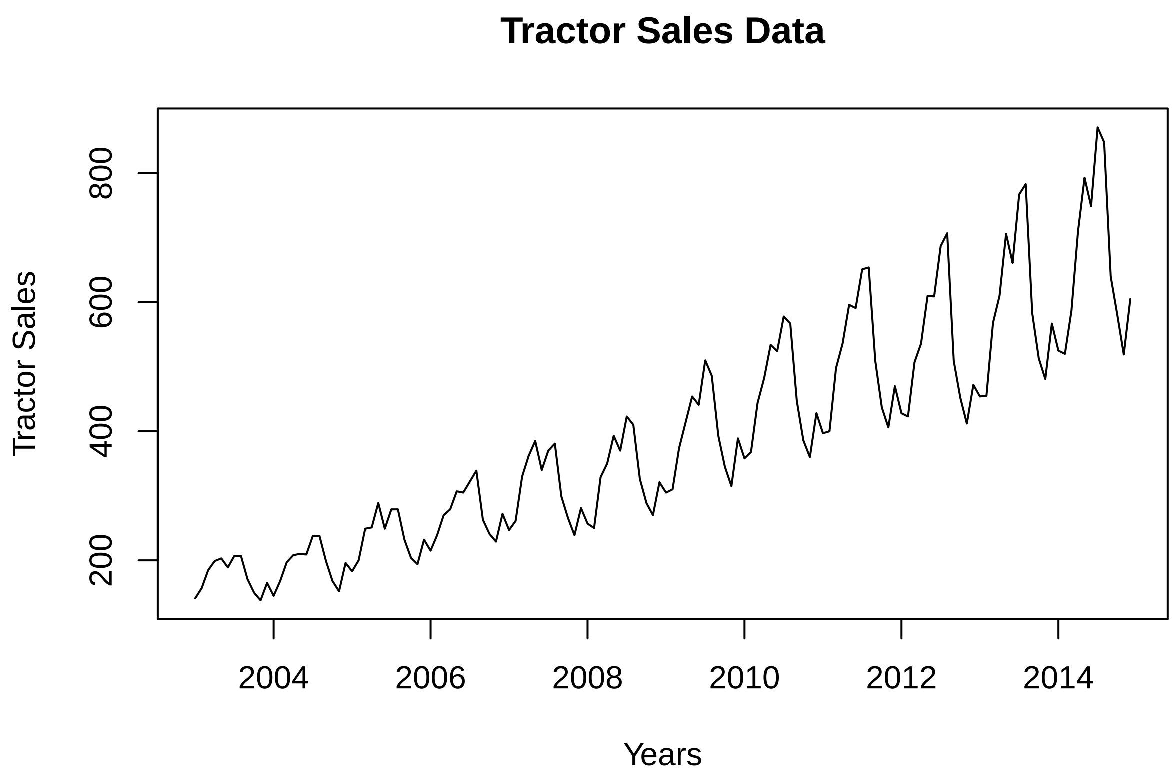
```
sales<-read.csv("Tractor-Sales.csv")
head(sales)
```

```
##   Month.Year Number.of.Tractor.Sold
## 1     Jan-03                    141
## 2     Feb-03                    157
## 3     Mar-03                    185
## 4     Apr-03                    199
## 5     May-03                    203
## 6     Jun-03                    189
```

```r
# Converting data to timeseries:
sales_ts=ts(sales[,2],start=c(2003,1),frequency=12)
sales_ts
```

```
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2003 141 157 185 199 203 189 207 207 171 150 138 165
## 2004 145 168 197 208 210 209 238 238 199 168 152 196
## 2005 183 200 249 251 289 249 279 279 232 204 194 232
## 2006 215 239 270 279 307 305 322 339 263 241 229 272
## 2007 247 261 330 362 385 340 370 381 299 266 239 281
## 2008 257 250 329 350 393 370 423 410 326 289 270 321
## 2009 305 310 374 414 454 441 510 486 393 345 315 389
## 2010 358 368 444 482 534 524 578 567 447 386 360 428
## 2011 397 400 498 536 596 591 651 654 509 437 406 470
## 2012 428 423 507 536 610 609 687 707 509 452 412 472
## 2013 454 455 568 610 706 661 767 783 583 513 481 567
## 2014 525 520 587 710 793 749 871 848 640 581 519 605
```

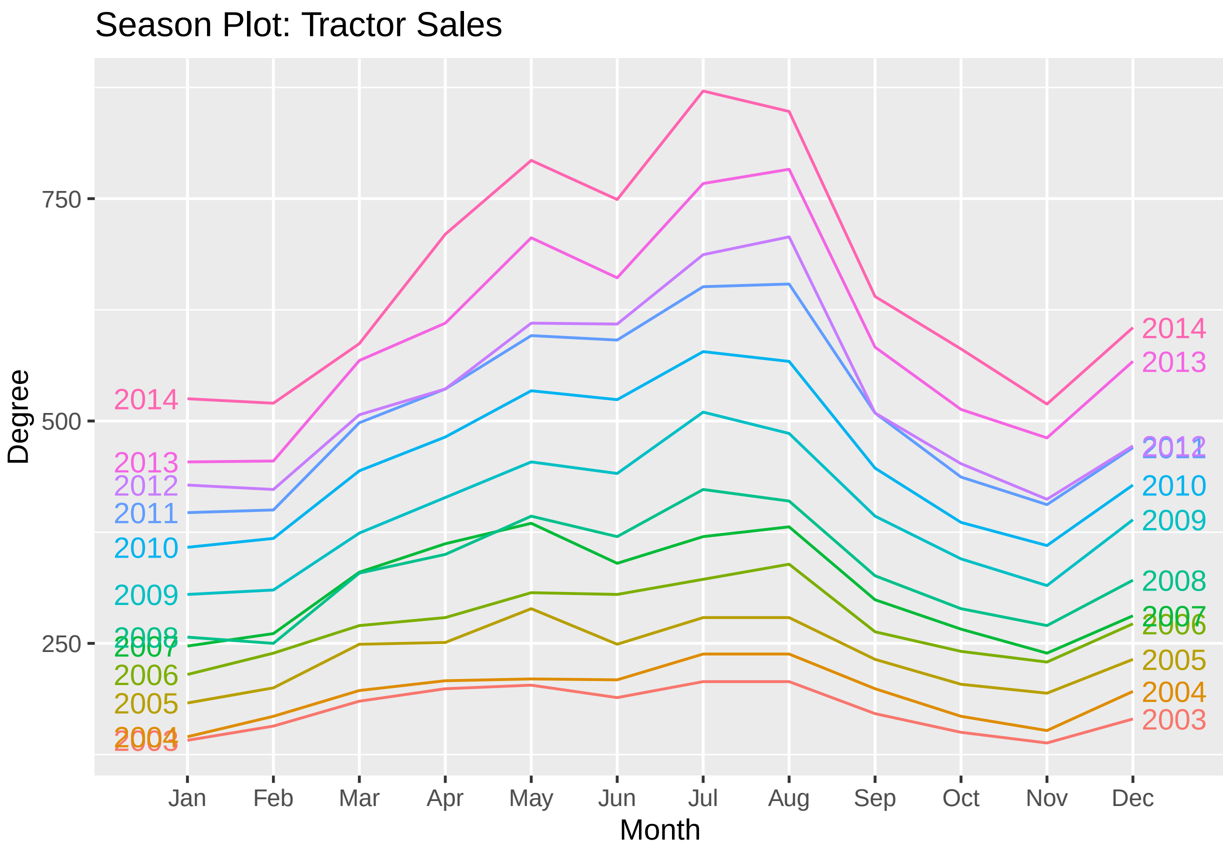
```r
plot(sales_ts,xlab="Years",ylab="Tractor Sales",main="Tractor Sales Data")
```



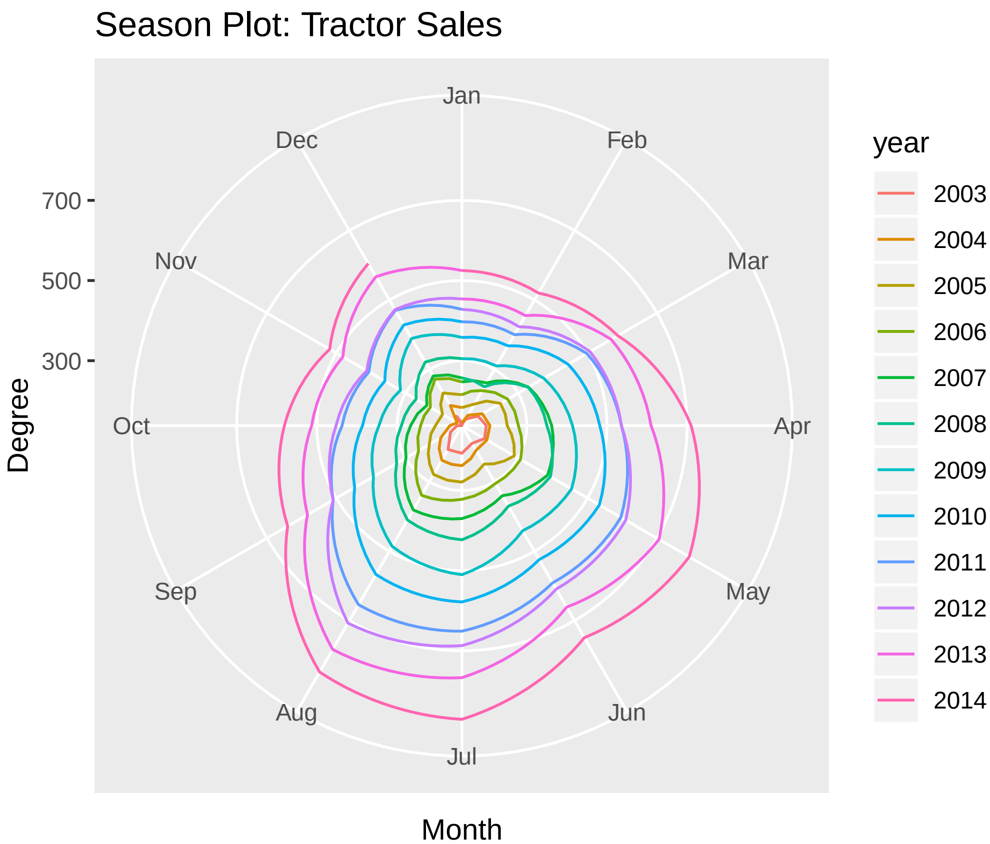
#

Sesonalty:

```r
ggseasonplot(sales_ts,year.labels = TRUE,year.labels.left = TRUE)+ylab("Degree")+
  ggtitle("Season Plot: Tractor Sales")
```

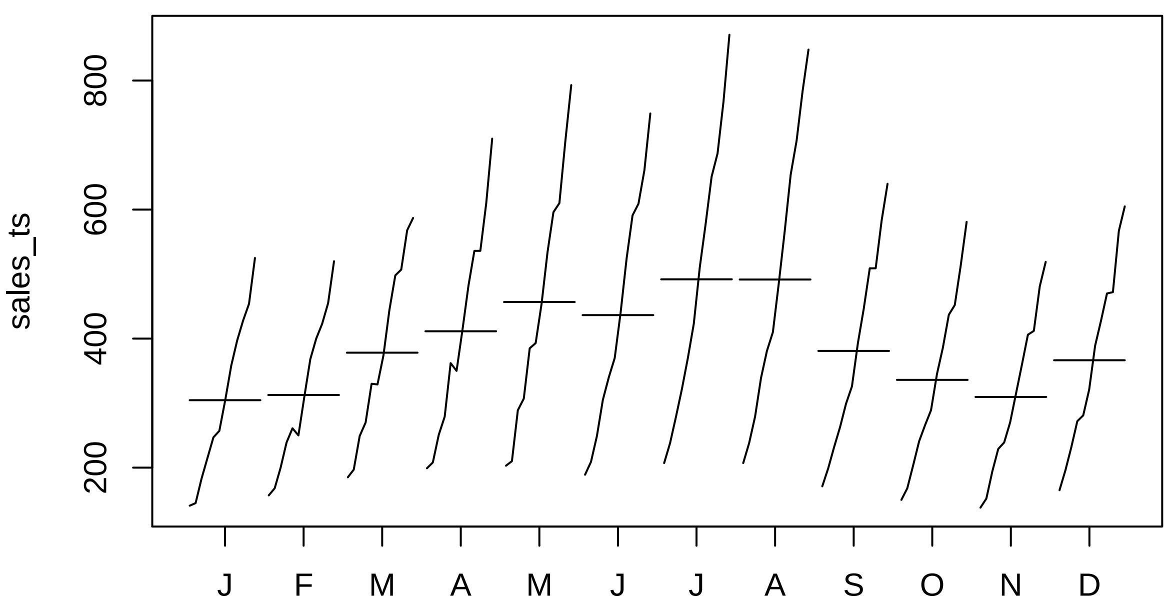



```
ggseasonplot(sales_ts,polar=TRUE  )+ylab("Degree")+
  ggtitle("Season Plot: Tractor Sales")
```



July & August seem to be the peek months for sales.

```r
monthplot(sales_ts)
```



Irregular components at April 2008 and Feb 2008

Decomposition:

```r
sales_decompose<-decompose(sales_ts,type='multiplicative')
sales_decompose
```

```
## $x
##      Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec
## 2003 141 157 185 199 203 189 207 207 171 150 138 165
## 2004 145 168 197 208 210 209 238 238 199 168 152 196
## 2005 183 200 249 251 289 249 279 279 232 204 194 232
## 2006 215 239 270 279 307 305 322 339 263 241 229 272
## 2007 247 261 330 362 385 340 370 381 299 266 239 281
## 2008 257 250 329 350 393 370 423 410 326 289 270 321
## 2009 305 310 374 414 454 441 510 486 393 345 315 389
## 2010 358 368 444 482 534 524 578 567 447 386 360 428
## 2011 397 400 498 536 596 591 651 654 509 437 406 470
## 2012 428 423 507 536 610 609 687 707 509 452 412 472
## 2013 454 455 568 610 706 661 767 783 583 513 481 567
## 2014 525 520 587 710 793 749 871 848 640 581 519 605
##
## $seasonal
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2003 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2004 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2005 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2006 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2007 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2008 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2009 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2010 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2011 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2012 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2013 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
## 2014 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
##            Aug       Sep       Oct       Nov       Dec
## 2003 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2004 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2005 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2006 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2007 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2008 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2009 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2010 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2011 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2012 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2013 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
## 2014 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
##
## $trend
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2003       NA       NA       NA       NA       NA       NA 176.1667 176.7917
## 2004 182.5417 185.1250 187.5833 189.5000 190.8333 192.7083 195.5833 198.5000
```

```
## 2005 219.3750 222.7917 225.8750 228.7500 232.0000 235.2500 238.0833 241.0417
## 2006 254.7083 259.0000 262.7917 265.6250 268.6250 271.7500 274.7500 277.0000
## 2007 301.2500 305.0000 308.2500 310.7917 312.2500 313.0417 313.8333 313.7917
## 2008 317.6250 321.0417 323.3750 325.4583 327.7083 330.6667 334.3333 338.8333
## 2009 365.0417 371.8333 377.7917 382.9167 387.1250 391.8333 396.8750 401.5000
## 2010 431.8333 438.0417 443.6667 447.6250 451.2083 454.7083 457.9583 460.9167
## 2011 485.0417 491.7083 497.9167 502.6250 506.6667 510.3333 513.3750 515.6250
## 2012 521.5000 525.2083 527.4167 528.0417 528.9167 529.2500 530.4167 532.8333
## 2013 561.0833 567.5833 573.8333 579.4583 584.8750 591.7083 598.6250 604.2917
## 2014 635.8333 642.8750 647.9583 653.1667 657.5833 660.7500       NA       NA
##           Sep      Oct      Nov      Dec
## 2003 177.7500 178.6250 179.2917 180.4167
## 2004 202.0000 205.9583 211.0417 216.0000
## 2005 243.5417 245.5833 247.5000 250.5833
## 2006 280.4167 286.3750 293.0833 297.7917
## 2007 313.2917 312.7500 312.5833 314.1667
## 2008 343.2083 347.7500 352.9583 358.4583
## 2009 406.8333 412.5833 418.7500 425.5417
## 2010 464.5000 469.0000 473.8333 479.2083
## 2011 516.9583 517.3333 517.9167 519.2500
## 2012 536.7083 542.3333 549.4167 555.5833
## 2013 607.7917 612.7500 620.5417 627.8333
## 2014       NA       NA       NA       NA
##
## $random
##            Jan       Feb       Mar       Apr       May       Jun       Jul
## 2003        NA        NA        NA        NA        NA        NA 0.9506481
## 2004 0.9647844 1.0754101 1.0373675 1.0157964 0.9280175 0.9675514 0.9845058
## 2005 1.0131838 1.0638023 1.0889096 1.0154655 1.0505119 0.9442738 0.9480874
## 2006 1.0252260 1.0935233 1.0148756 0.9720482 0.9637914 1.0012869 0.9481812
## 2007 0.9958506 1.0140761 1.0574784 1.0779330 1.0397990 0.9689581 0.9538407
## 2008 0.9827493 0.9228019 1.0049631 0.9952340 1.0113376 0.9982505 1.0236085
## 2009 1.0148029 0.9879689 0.9778672 1.0005726 0.9889987 1.0040734 1.0396559
## 2010 1.0069112 0.9955482 0.9885231 0.9965182 0.9980568 1.0280793 1.0211159
## 2011 0.9941130 0.9640117 0.9879463 0.9869002 0.9920077 1.0331462 1.0259341
## 2012 0.9968131 0.9544180 0.9495433 0.9393970 0.9725987 1.0265608 1.0478829
## 2013 0.9827720 0.9499738 0.9777396 0.9742270 1.0179648 0.9966026 1.0366059
## 2014 1.0028600 0.9585322 0.8948529 1.0059743 1.0169825 1.0112834        NA
##            Aug       Sep       Oct       Nov       Dec
## 2003 0.9518221 1.0000639 1.0039910 1.0054376 1.0126675
## 2004 0.9746837 1.0241013 0.9752383 0.9408307 1.0047590
## 2005 0.9409344 0.9902753 0.9931437 1.0239120 1.0251688
## 2006 0.9948719 0.9749743 1.0061501 1.0206586 1.0113837
## 2007 0.9870311 0.9921186 1.0168693 0.9987762 0.9903890
## 2008 0.9836599 0.9874179 0.9935999 0.9992553 0.9915759
## 2009 0.9840065 1.0041931 0.9997427 0.9826341 1.0122018
## 2010 1.0000182 1.0003755 0.9840004 0.9924600 0.9889606
## 2011 1.0310768 1.0235368 1.0099312 1.0240054 1.0022613
## 2012 1.0786369 0.9858723 0.9964441 0.9795610 0.9407027
## 2013 1.0533250 0.9971375 1.0009557 1.0125355 0.9999961
## 2014        NA        NA        NA        NA        NA
##
## $figure
##  [1] 0.8233333 0.8438594 1.0123702 1.0805564 1.1857930 1.1209126 1.2360238
```

```
##  [8] 1.2301349 0.9619639 0.8364099 0.7655329 0.9031095
##
## $type
## [1] "multiplicative"
##
## attr(,"class")
## [1] "decomposed.ts"
```

Decomposition gives the follwing components:

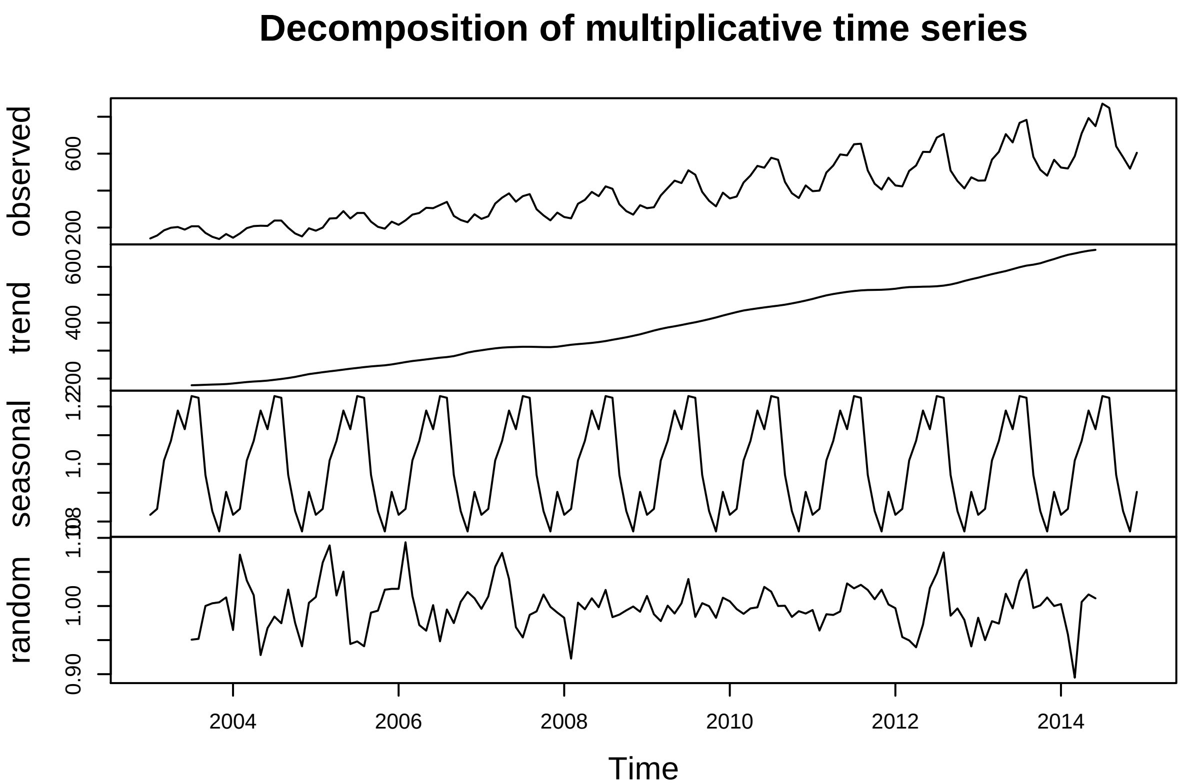
$x: The original data

$seasonal: Sales percentange relative to annual trend. Therefore, In january sales are 18% less than annual trend.

$trend: continuous trend of sales yearly

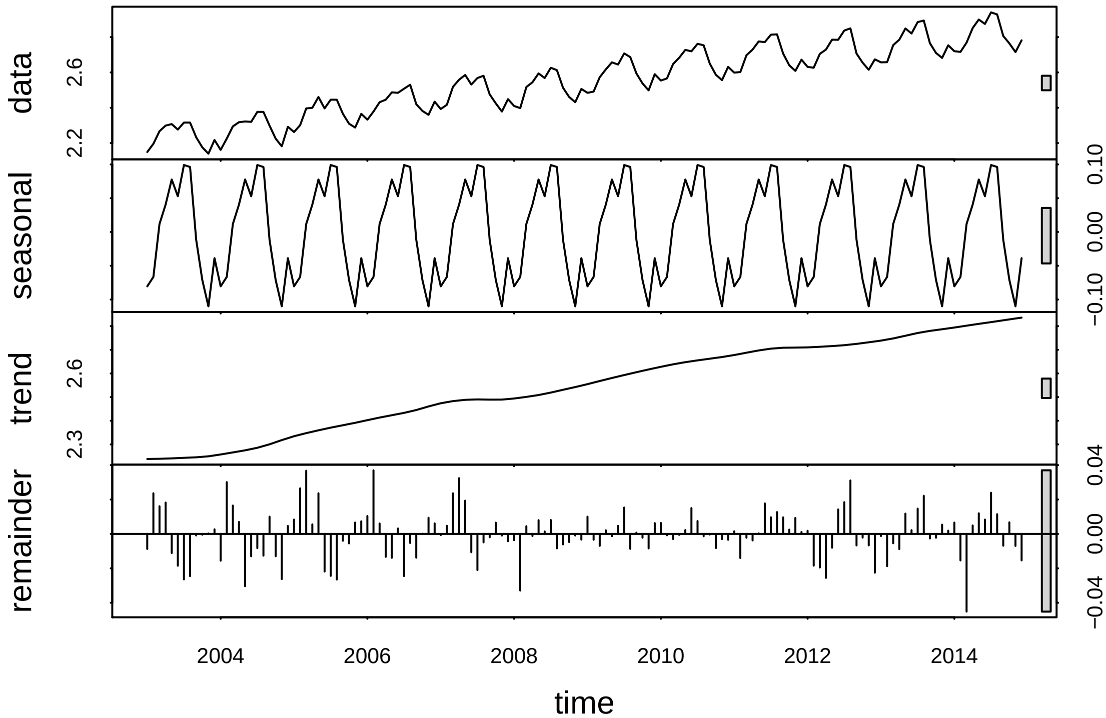
$random: Peculiarity i.e. Variation in data having accounted for seasonality & trend.

```r
plot(sales_decompose)
```



- The trend is generally growing over the years.
- The random error is within 10% of total sales for the years.
- Therefore, If the company has a high budget,we can confidently plan the production to be the forecasted sales +/- 10%.
- If costs are high and company has a low budget, then we can plan the production to be 450 and be confident all tractors are going to be sold.

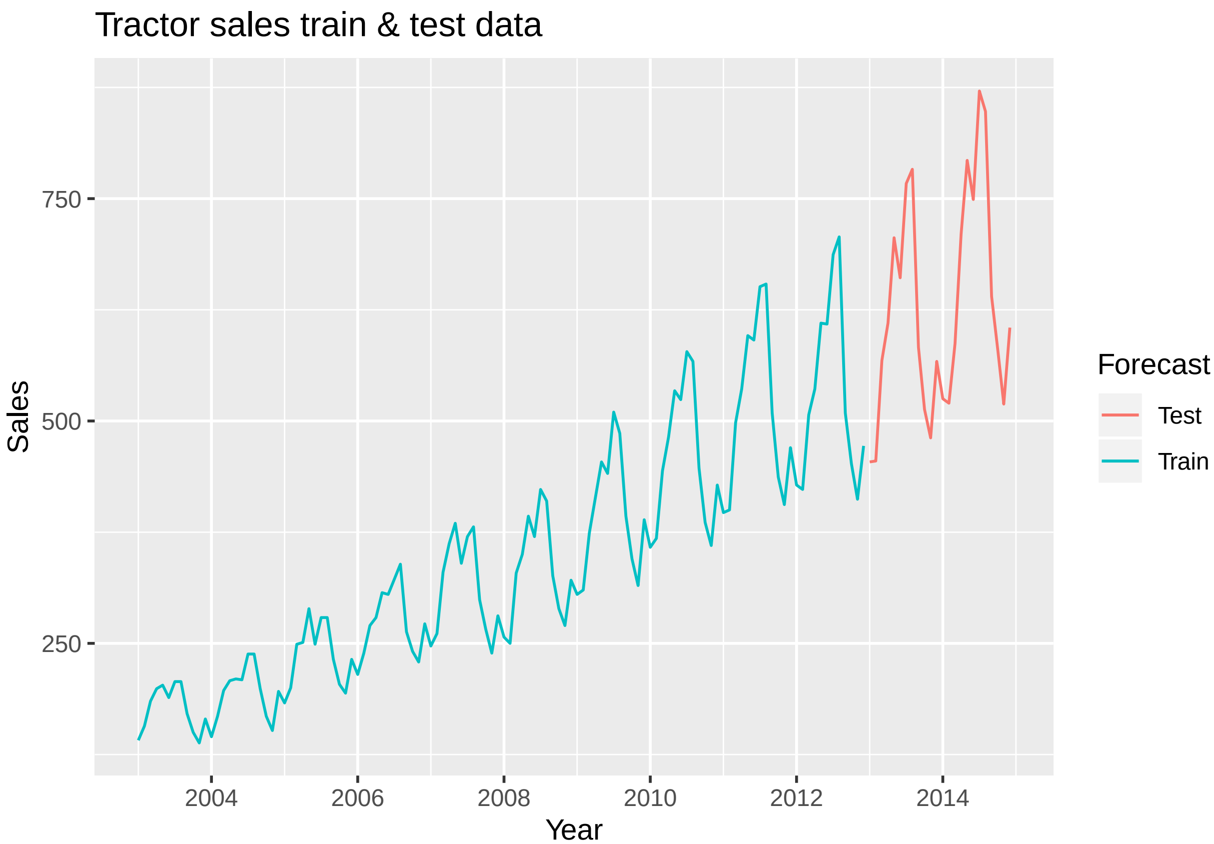
```r
## additive model on logarithm scale.
sales_decom_log<-stl(log10(sales_ts),s.window='p')
plot(sales_decom_log)
```



Train Test Split:

```r
ts_train<-window(sales_ts,start=c(2003,1),end=c(2012,12),freq=12)
ts_test<-window(sales_ts,start=c(2013,1),freq=12)
```

```r
autoplot(ts_train,,series="Train")+autolayer(ts_test,series="Test")+ggtitle("Tractor sales train & test
```



Forecasting:

Random Walk with Drift:

A random walk is defined as a process where the current value of a variable is composed of the past value plus an error term defined as a white noise (a normal variable with zero mean and variance one). Algebraically a random walk is represented as follows:

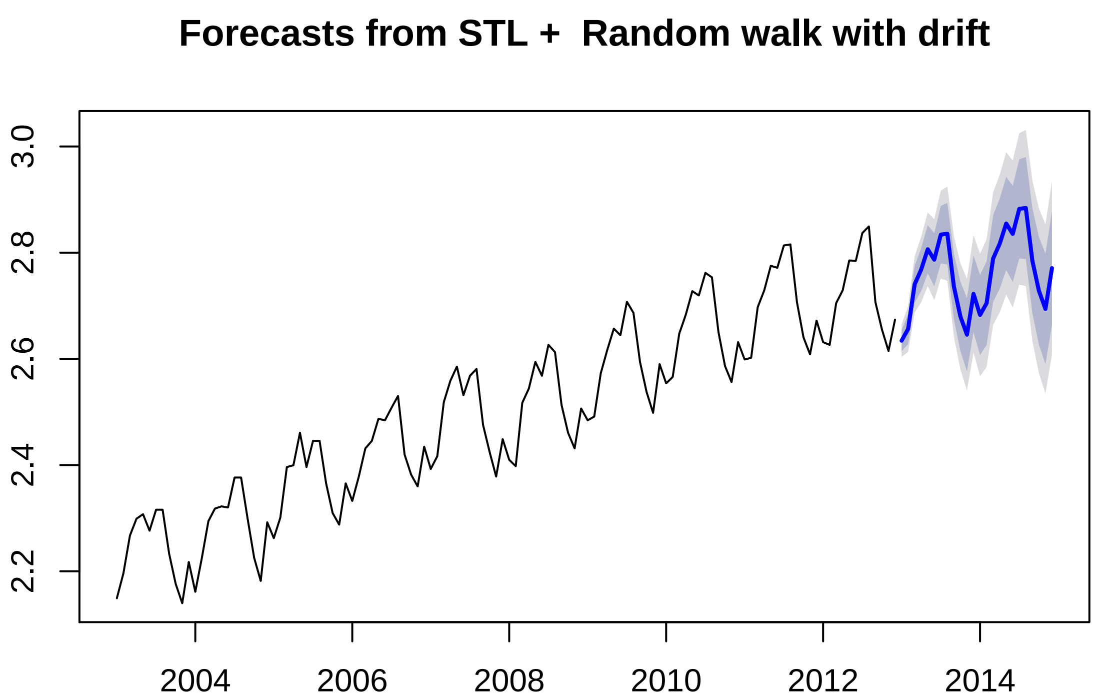
$$y_t = y_{t-1} + \epsilon_t$$

```
ts_decompose_train_log<-stl(log10(ts_train),s.window='p')
ts_train_stl<-forecast(ts_decompose_train_log,method='rwdrift',h=24)
ts_train_stl
```

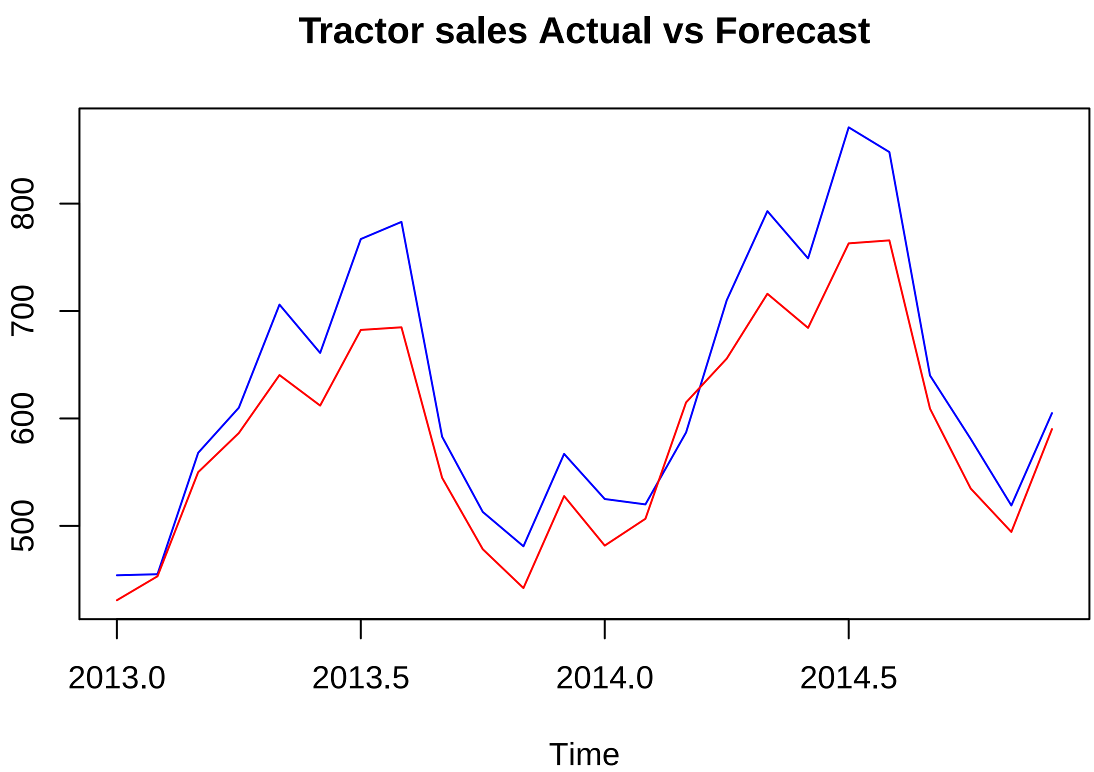
```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2013       2.634223 2.614268 2.654178 2.603705 2.664742
## Feb 2013       2.656170 2.627831 2.684509 2.612830 2.699511
## Mar 2013       2.740345 2.705493 2.775197 2.687043 2.793647
## Apr 2013       2.768221 2.727812 2.808631 2.706420 2.830023
## May 2013       2.806446 2.761082 2.851811 2.737068 2.875825
## Jun 2013       2.786750 2.736855 2.836646 2.710441 2.863059
## Jul 2013       2.834038 2.779928 2.888149 2.751284 2.916793
## Aug 2013       2.835597 2.777520 2.893674 2.746775 2.924419
## Sep 2013       2.736191 2.674346 2.798035 2.641608 2.830773
## Oct 2013       2.679677 2.614231 2.745123 2.579586 2.779768
```

```
## Nov 2013       2.645540 2.576632 2.714448 2.540154 2.750925
## Dec 2013       2.722442 2.650192 2.794693 2.611945 2.832940
## Jan 2014       2.682723 2.607234 2.758213 2.567273 2.798174
## Feb 2014       2.704671 2.626034 2.783308 2.584406 2.824936
## Mar 2014       2.788845 2.707140 2.870550 2.663888 2.913802
## Apr 2014       2.816722 2.732021 2.901423 2.687183 2.946261
## May 2014       2.854947 2.767314 2.942580 2.720923 2.988970
## Jun 2014       2.835251 2.744744 2.925758 2.696832 2.973669
## Jul 2014       2.882539 2.789210 2.975867 2.739805 3.025272
## Aug 2014       2.884097 2.787996 2.980199 2.737122 3.031072
## Sep 2014       2.784691 2.685860 2.883522 2.633542 2.935840
## Oct 2014       2.728177 2.626657 2.829697 2.572916 2.883439
## Nov 2014       2.694040 2.589868 2.798212 2.534723 2.853357
## Dec 2014       2.770943 2.664154 2.877732 2.607623 2.934263
```

```r
plot(ts_train_stl)
```




```r
vec2<-10^(cbind(log10(ts_test),as.data.frame(ts_train_stl)[,1]))
ts.plot(vec2,col=c('blue','red'),main='Tractor sales Actual vs Forecast')
```




```
RMSE<-round(sqrt(sum(((vec2[,1]-vec2[,2])^2)/length(vec2[,1]))),4)
MAPE<-round(mean(abs(vec2[,1]-vec2[,2])/vec2[,1]),4)
paste("Accuracy Measures: RMSE: ",RMSE,"and MAPE: ",MAPE)
```

```
## [1] "Accuracy Measures: RMSE:  53.5697 and MAPE:  0.0687"
```

Therefore The forecast on average is about 6.87% away from actual sales. Underpredicting by 6.87%

```
10^as.data.frame(ts_train_stl)
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2013       430.7478 411.4036 451.0015 401.5176 462.1059
## Feb 2013       453.0753 424.4548 483.6255 410.0434 500.6231
## Mar 2013       549.9774 507.5662 595.9325 486.4554 621.7943
## Apr 2013       586.4370 534.3324 643.6224 508.6510 676.1185
## May 2013       640.3927 576.8754 710.9035 545.8428 751.3203
## Jun 2013       611.9986 545.5753 686.5088 513.3830 729.5572
## Jul 2013       682.3988 602.4595 772.9452 564.0057 825.6444
## Aug 2013       684.8525 599.1280 782.8426 558.1813 840.2698
## Sep 2013       544.7418 472.4397 628.1091 438.1350 677.2882
## Oct 2013       478.2741 411.3685 556.0613 379.8272 602.2372
## Nov 2013       442.1196 377.2524 518.1405 346.8602 563.5404
## Dec 2013       527.7672 446.8813 623.2935 409.2090 680.6747
## Jan 2014       481.6410 404.7943 573.0765 369.2095 628.3102
## Feb 2014       506.6065 422.7013 607.1667 384.0657 668.2453
## Mar 2014       614.9577 509.4955 742.2500 461.1990 819.9780
## Apr 2014       655.7250 539.5362 796.9350 486.6117 883.6107
## May 2014       716.0556 585.2125 876.1528 525.9246 974.9224
```

```
## Jun 2014       684.3067 555.5763 842.8648 497.5447  941.1730
## Jul 2014       763.0248 615.4747 945.9477 549.2944 1059.9178
## Aug 2014       765.7684 613.7558 955.4307 545.9117 1074.1685
## Sep 2014       609.1035 485.1320 764.7549 430.0727  862.6614
## Oct 2014       534.7825 423.3086 675.6120 374.0378  764.6081
## Nov 2014       494.3564 388.9272 628.3650 342.5493  713.4396
## Dec 2014       590.1233 461.4809 754.6261 405.1567  859.5330
```

Therefore 95% confidence interval : forecast +/- 1.96 * RMSE i.e. For January 2013 Forecast interval: 430 +/- 1.96 * 53.56

Exponential Smoothing:

Exponential smoothing is a popular forecasting method for short-term predictions. Such forecasts of future values are based on past data whereby the most recent observations are weighted more than less recent observations. As part of this weighting, constants are being smoothed. This is different from the simple moving average method, in which every data point has equal weight in the average calculation. Exponential smoothing introduces the idea of building a forecasted value as the average figure from differently weighted data points for the average calculation.

There are different exponential smoothing methods that differ from each other in the components of the time series that are modeled. * Single Exponential Smoothing * Double Exponential Smoothing * Triple Exponential Smoothing

Simple exponential smoothing (SES) uses only one smoothing constant, double exponential smoothing or Holt exponential smoothing uses two smoothing constants and triple exponential smoothing or Holt-Winters exponential smoothing accordingly uses three smoothing constants.

Single Exponential Smoothing:

Simple exponential smoothing assumes that the time series data has only a level and some error (or remainder) but no trend or seasonality The smoothing parameter α determines the distribution of weights of past observations and with that how heavily a given time period is factored into the forecasted value. If the smoothing parameter is close to 1, recent observations carry more weight and if the smoothing parameter is closer to 0

$$F_t = F_{t-1} + \alpha * (A_{t-1} - F_{t-1})$$

or

$$F_t = \alpha * A_{t-1} + (1 - \alpha) * F_{t-1}$$

F_{t-1} = forecast for the previous period,

A_{t-1} = Actual demand for the period,

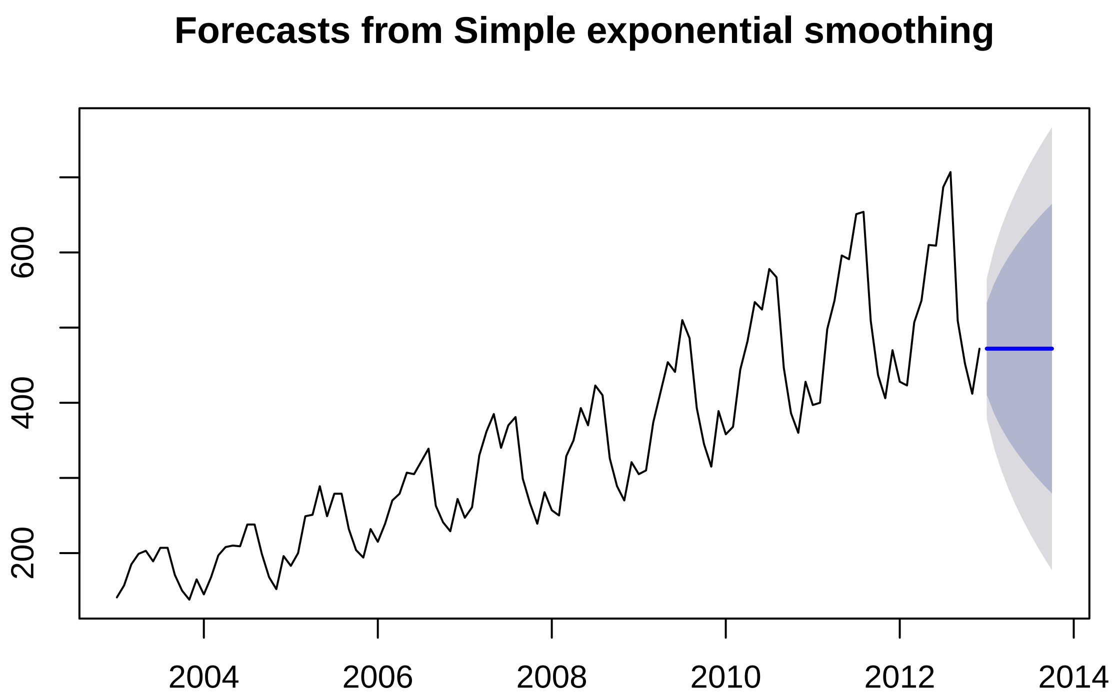
a = weight (between 0 and 1). The closer to zero, the smaller the weight.

```
ts_train_ses<-ses(ts_train,hs=24)
ts_train_ses
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2013        471.994 411.0351 532.9529 378.7654 565.2226
## Feb 2013        471.994 385.7894 558.1986 340.1555 603.8325
## Mar 2013        471.994 366.4172 577.5708 310.5282 633.4598
```

```
## Apr 2013        471.994 350.0854 593.9026 285.5509 658.4371
## May 2013        471.994 335.6967 608.2913 263.5453 680.4427
## Jun 2013        471.994 322.6883 621.2997 243.6506 700.3374
## Jul 2013        471.994 310.7258 633.2622 225.3556 718.6324
## Aug 2013        471.994 299.5914 644.3966 208.3269 735.6611
## Sep 2013        471.994 289.1336 654.8544 192.3332 751.6548
## Oct 2013        471.994 279.2425 664.7455 177.2060 766.7820
```

```
plot(ts_train_ses)
```



It can be Observed that all Point forecasts are the same.

Holt Method (Double Exponential Smoothing):

Holt exponential smoothing is a time series forecasting approach that fits time series data with an overall level as well as a trend. Additionally, to simple exponential smoothing, which uses smoothing parameter α only there is also a β smoothing parameter for the exponential decay of the modeled trend component. This β smoothing parameter ranges between 0 and 1, with higher values indicating more weight to recent observations. It accounts for level and trend but not seasonality.

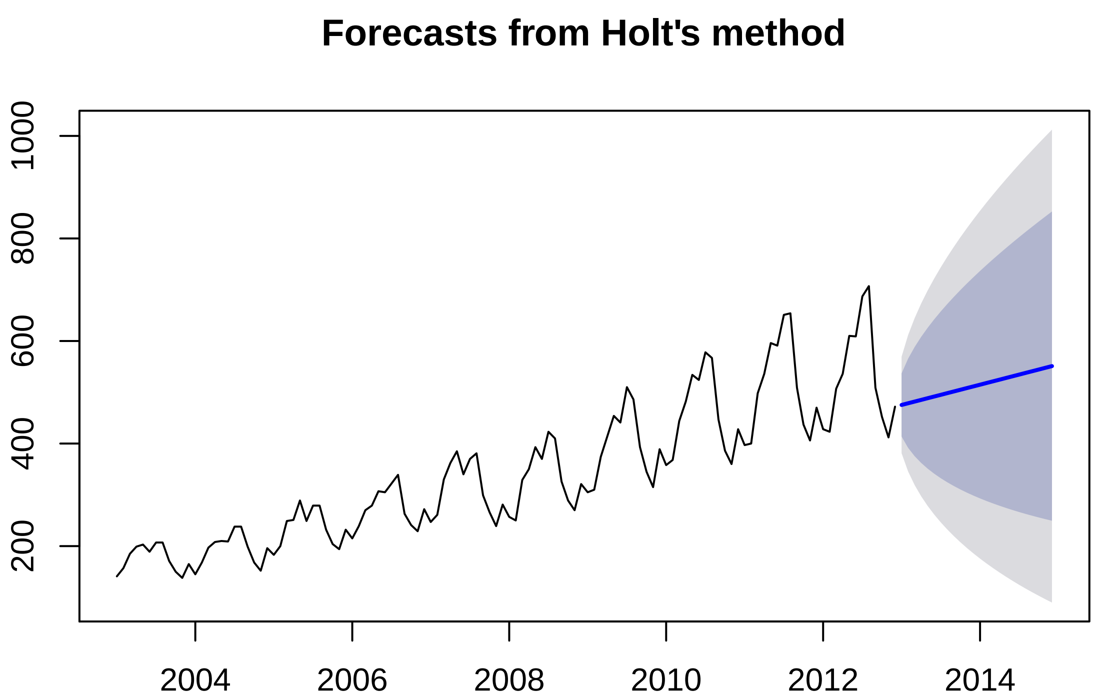
$$b_t = \gamma(S_t - S_{t-1}) + (1 - \gamma)b_{t-1}$$

```
ts_train_holt<-holt(ts_train,h=24)
ts_train_holt
```

```
##          Point Forecast    Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2013       475.2876 413.8077 536.7674 381.26220 569.3129
## Feb 2013       478.5814 391.6360 565.5269 345.60982 611.5530
## Mar 2013       481.8753 375.3856 588.3650 319.01336 644.7372
## Apr 2013       485.1691 362.2001 608.1382 297.10414 673.2342
```

```
## May 2013       488.4630 350.9728 625.9532 278.18993  698.7361
## Jun 2013       491.7569 341.1365 642.3773 261.40286  722.1109
## Jul 2013       495.0507 332.3539 657.7476 246.22739  743.8741
## Aug 2013       498.3446 324.4055 672.2838 232.32764  764.3616
## Sep 2013       501.6385 317.1386 686.1383 219.47030  783.8066
## Oct 2013       504.9323 310.4423 699.4224 207.48552  802.3792
## Nov 2013       508.2262 304.2327 712.2197 196.24507  820.2073
## Dec 2013       511.5201 298.4447 724.5955 185.64935  837.3908
## Jan 2014       514.8139 293.0264 736.6015 175.61914  854.0087
## Feb 2014       518.1078 287.9359 748.2797 166.09023  870.1254
## Mar 2014       521.4017 283.1386 759.6648 157.00968  885.7937
## Apr 2014       524.6955 278.6055 770.7856 148.33327  901.0578
## May 2014       527.9894 274.3122 781.6666 140.02359  915.9552
## Jun 2014       531.2833 270.2378 792.3287 132.04866  930.5179
## Jul 2014       534.5771 266.3642 802.7900 124.38089  944.7734
## Aug 2014       537.8710 262.6758 813.0662 116.99628  958.7457
## Sep 2014       541.1649 259.1588 823.1709 109.87380  972.4559
## Oct 2014       544.4587 255.8010 833.1164 102.99489  985.9226
## Nov 2014       547.7526 252.5918 842.9134  96.34309  999.1621
## Dec 2014       551.0465 249.5214 852.5715  89.90369 1012.1892
```

```r
plot(ts_train_holt)
```



Holt Winter's Method (triple exponential model):

Holt-Winters exponential smoothing is a time series forecasting approach that takes the overall level, trend and seasonality of the underlying dataset into account for its forecast. Hence, the Holt-Winters model has three smoothing parameters indicating the exponential decay from most recent to older observations: α for the level component, β for the trend component, and γ or the seasonality component. It accounts for level and trend and seasonality.

level:

$$L_t = \alpha * (y_t - S_{t-s}) + (1 - \alpha)(L_{t-1} + b_{t-1})$$

trend:

$$b_t = \beta * (L_t - L_{t-1}) + (1 - \beta) * b_{t-1})$$

seasonal:

$$S_t = \gamma(y_t - L_t) + (1 - \gamma)S_{t-s}$$

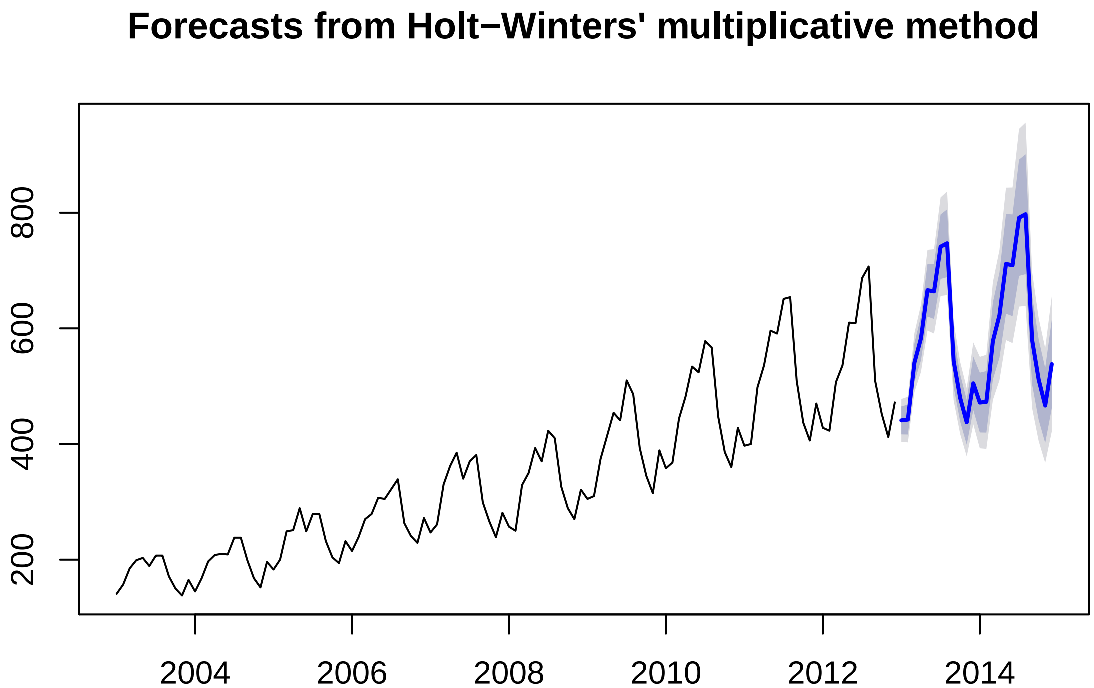
forecast:

$$F_{t+k} = L_t + k * b_t + S_{t+k-s}$$

```r
ts_train_hw<-hw(ts_train,h=24,seasonal="multiplicative")
ts_train_hw
```

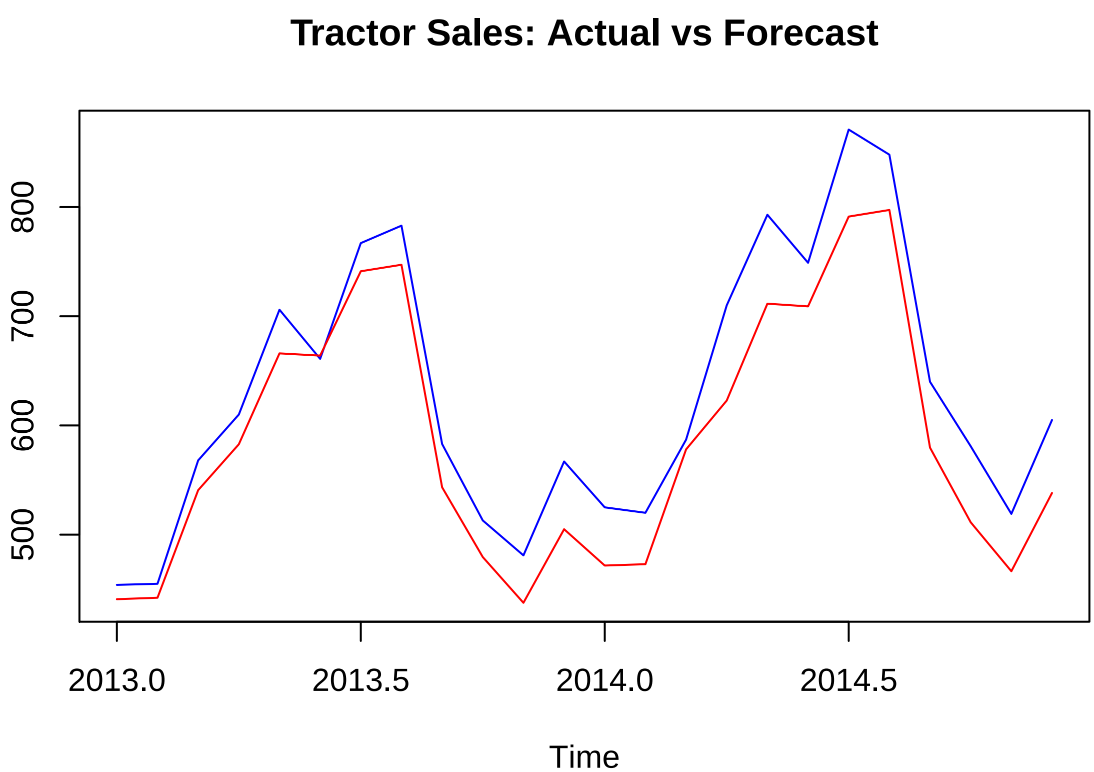
```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## Jan 2013       440.8777 416.6841 465.0713 403.8768 477.8786
## Feb 2013       442.2152 416.4192 468.0111 402.7637 481.6667
## Mar 2013       540.7288 507.3398 574.1179 489.6647 591.7929
## Apr 2013       582.7737 544.8184 620.7291 524.7261 640.8214
## May 2013       666.0299 620.4189 711.6408 596.2740 735.7858
## Jun 2013       663.9650 616.2837 711.6463 591.0428 736.8872
## Jul 2013       741.2300 685.5440 796.9160 656.0656 826.3944
## Aug 2013       747.1779 688.5796 805.7761 657.5596 836.7962
## Sep 2013       543.3566 498.9564 587.7568 475.4523 611.2609
## Oct 2013       479.4786 438.7255 520.2318 417.1520 541.8052
## Nov 2013       437.5753 398.9513 476.1993 378.5050 496.6457
## Dec 2013       504.9576 458.7353 551.1799 434.2666 575.6486
## Jan 2014       471.6816 419.9860 523.3772 392.6200 550.7432
## Feb 2014       472.9365 419.7595 526.1135 391.6093 554.2637
## Mar 2014       578.0813 511.4267 644.7359 476.1419 680.0207
## Apr 2014       622.8040 549.1967 696.4113 510.2314 735.3766
## May 2014       711.5228 625.3641 797.6816 579.7544 843.2913
## Jun 2014       709.0645 621.1309 796.9980 574.5817 843.5472
## Jul 2014       791.2990 690.8398 891.7582 637.6599 944.9381
## Aug 2014       797.3708 693.7814 900.9603 638.9444 955.7973
## Sep 2014       579.6577 502.6271 656.6884 461.8495 697.4660
## Oct 2014       511.3378 441.8559 580.8197 405.0744 617.6011
## Nov 2014       466.4928 401.7011 531.2845 367.4024 565.5832
## Dec 2014       538.1485 461.7771 614.5199 421.3485 654.9484
```

```r
plot(ts_train_hw)
```



Accuracy of Holt Winter's method:

```r
vec<-cbind(ts_test,as.data.frame(ts_train_hw)[,1])
ts.plot(vec,col=c("blue","red"),main="Tractor Sales: Actual vs Forecast")
```



Underprediction! This means the trend seems to be increasing faster than what historical data is suggesting.

```
RMSE<-round(sqrt(sum(((vec[,1]-vec[,2])^2)/length(vec[,1]))),4)
MAPE<-round(mean(abs(vec[,1]-vec[,2])/vec[,1]),4)
paste("Accuracy Measures: RMSE: ",RMSE,"and MAPE: ",MAPE)
```

```
## [1] "Accuracy Measures: RMSE:  49.7553 and MAPE:  0.0703"
```

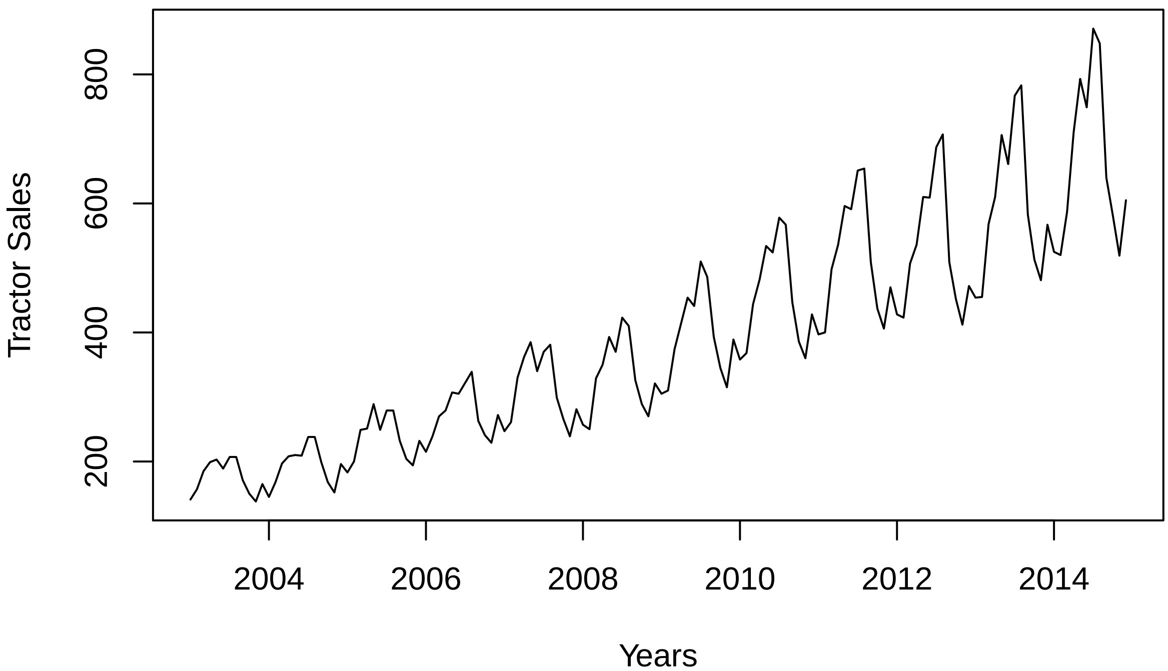
In analytics perspective, RMSE tracks variance while MAPE tracks bias.

Comparing rwdrift model to hold winter model, we observe that rwdrift model has less bias but high variance while holt winter model has less variance and high bias.

Stationarity:

Original data:

```
plot(sales_ts, xlab='Years', ylab = 'Tractor Sales')
```



Checking Stationarity using Augmented Dickey–Fuller test:

```
adf.test(sales_ts)
```

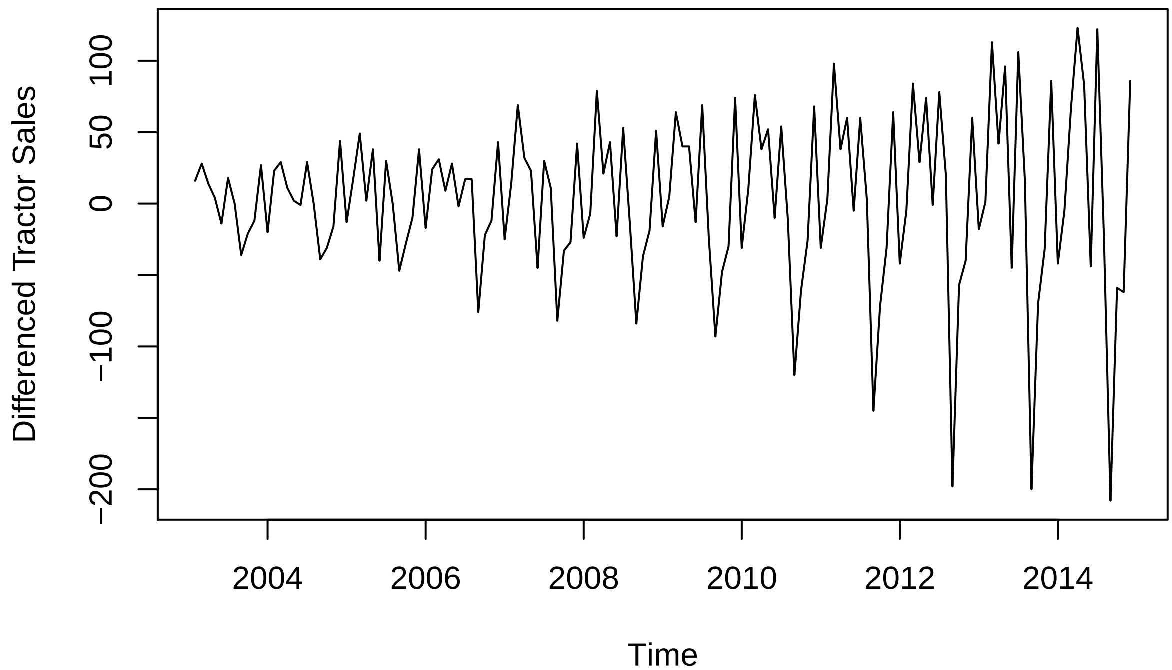
```
## Warning in adf.test(sales_ts): p-value smaller than printed p-value

##
##  Augmented Dickey-Fuller Test
##
## data:  sales_ts
## Dickey-Fuller = -12.783, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

Therefore Data is not stationary. Let's make it stationary for ARIMA modelling.

Trend Removal:

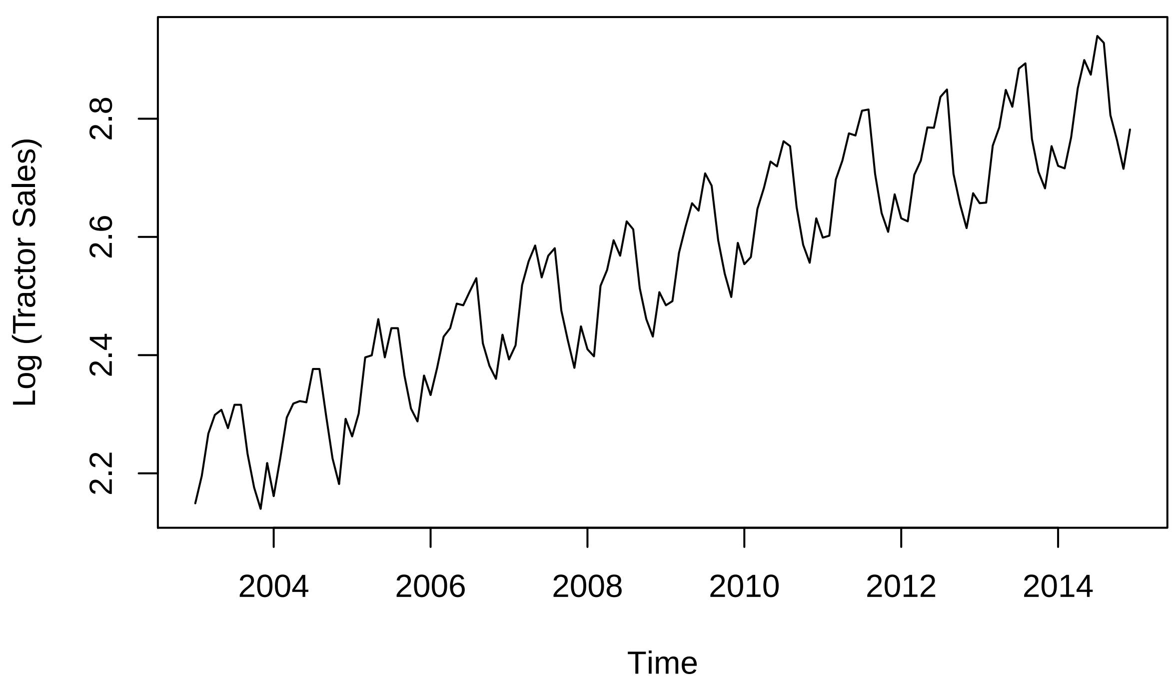
Removing the upward trend through 1st order differencing.

```r
plot(diff(sales_ts),ylab='Differenced Tractor Sales')
```



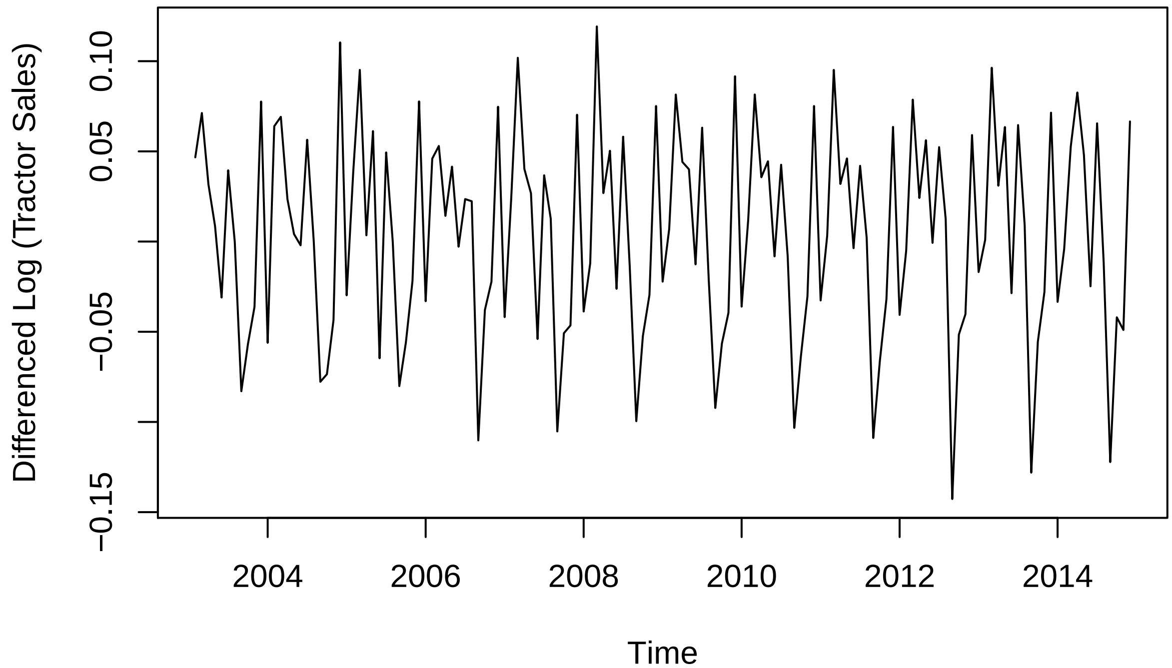
Log transform data to make data stationary on variance Now, the above series is not stationary on variance i.e. variation in the plot is increasing as we move towards the right of the chart. We need to make the series stationary on variance to produce reliable forecasts through ARIMA models.

```r
plot(log10(sales_ts),ylab='Log (Tractor Sales)')
```



Difference log transform data to make data stationary on both mean and variance:

```
plot(diff(log10(sales_ts)),ylab='Differenced Log (Tractor Sales)')
```

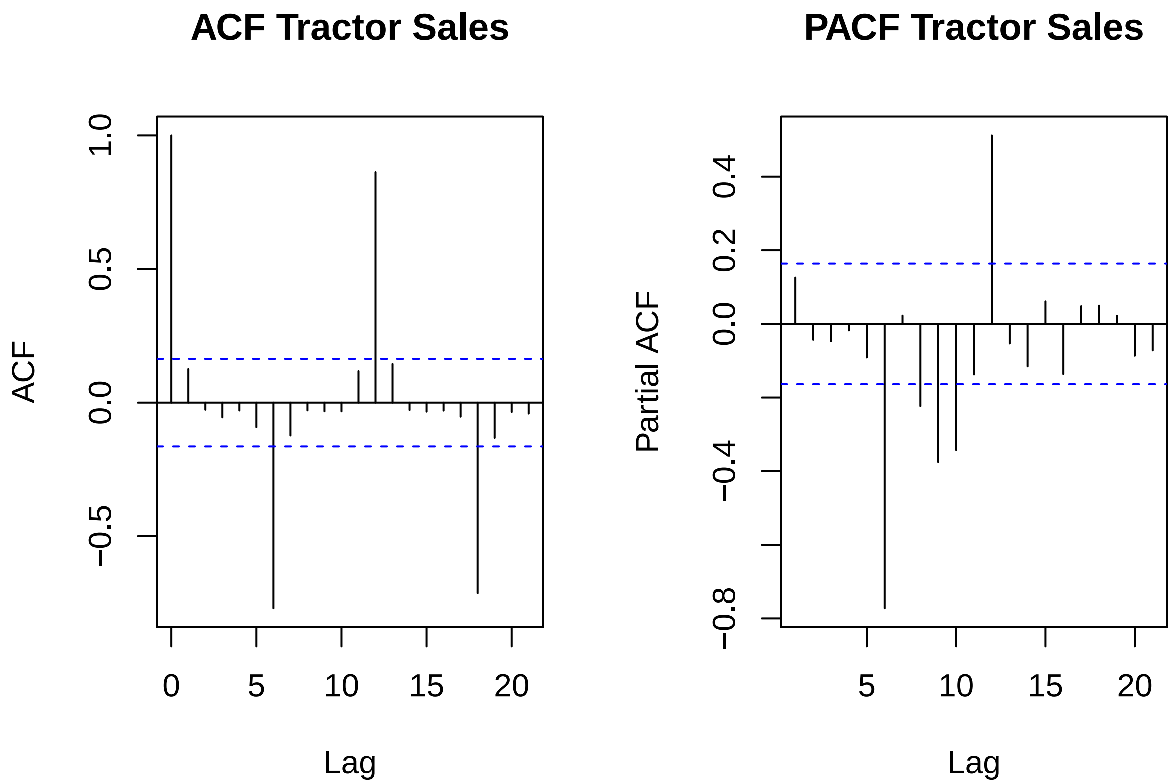


Now, The time series looks stationary. The Integrated part (I) of ARIMA model will be equal to 1 as we used 1st order difference to make series stationary.

ACF & PACF:

Plot ACF(Autocorrelation factor) and PACF(Partial Autocorrelation factor) to identify potential AR and MA components in the residuals.

```
par(mfrow=c(1,2))
acf(ts(diff(log10(sales_ts))),main='ACF Tractor Sales')
pacf(ts(diff(log10(sales_ts))),main='PACF Tractor Sales')
```



There are spikes beyond significant zones. Therefore residuals are not random. Hence AR MA models can be used to extract this information.

ARIMA:

Testing arima model on ts_test data:

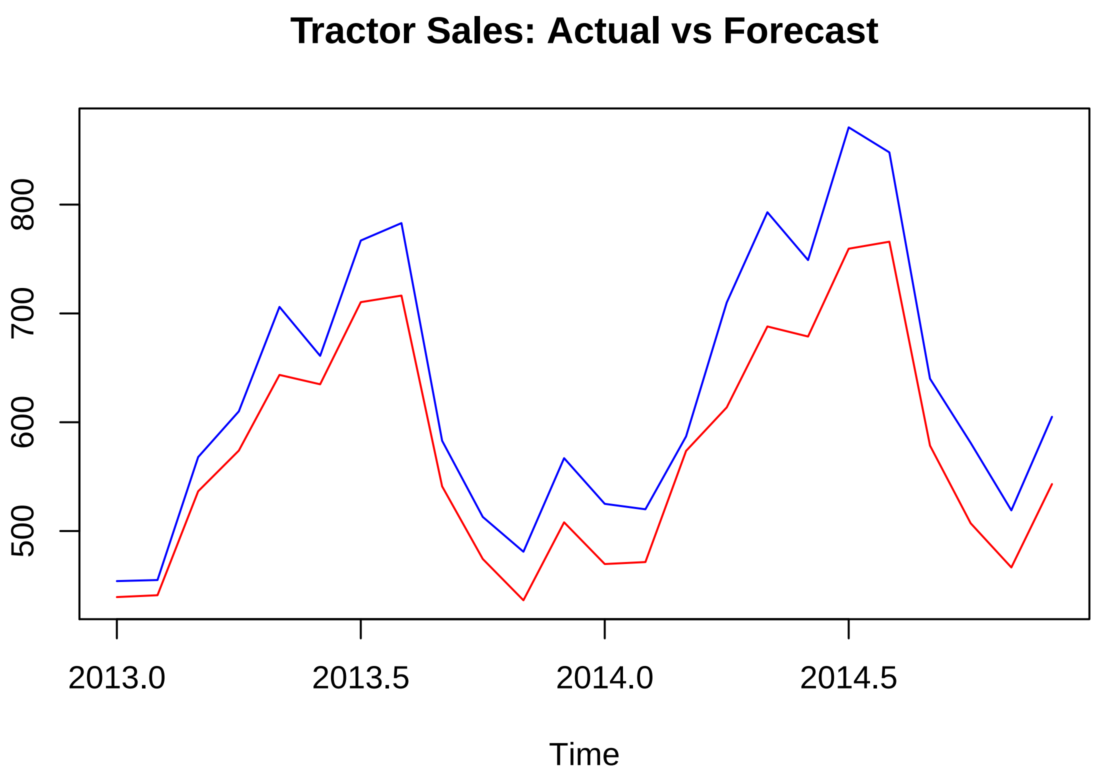
```
ARIMAfit = auto.arima(log10(ts_train), approximation=FALSE,trace=FALSE)
summary(ARIMAfit)
```

```
## Series: log10(ts_train)
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.3471  -0.5355
## s.e.   0.0996   0.0876
##
## sigma^2 estimated as 0.0002688:  log likelihood=287.06
## AIC=-568.12   AICc=-567.89   BIC=-560.11
##
## Training set error measures:
##                         ME       RMSE        MAE          MPE      MAPE
## Training set -6.71787e-05 0.01533676 0.01139577 -0.002628277 0.4559706
##                   MASE        ACF1
## Training set 0.2131231 0.002369841
```

```r
pred<-forecast(ARIMAfit,h=24)
vec<-10^cbind(log10(ts_test),as.data.frame(pred)[,1])
vec
```

```
##          log10(ts_test) as.data.frame(pred)[, 1]
## Jan 2013            454                 439.3281
## Feb 2013            455                 441.0230
## Mar 2013            568                 536.5490
## Apr 2013            610                 573.9486
## May 2013            706                 643.5105
## Jun 2013            661                 634.8851
## Jul 2013            767                 710.3604
## Aug 2013            783                 716.3897
## Sep 2013            583                 541.0908
## Oct 2013            513                 474.3412
## Nov 2013            481                 436.3978
## Dec 2013            567                 508.0340
## Jan 2014            525                 469.7022
## Feb 2014            520                 471.5143
## Mar 2014            587                 573.6447
## Apr 2014            710                 613.6301
## May 2014            793                 688.0014
## Jun 2014            749                 678.7796
## Jul 2014            871                 759.4731
## Aug 2014            848                 765.9192
## Sep 2014            640                 578.5006
## Oct 2014            581                 507.1361
## Nov 2014            519                 466.5693
## Dec 2014            605                 543.1583
```

```r
ts.plot(vec,col=c("blue","red"),main="Tractor Sales: Actual vs Forecast")
```




```r
RMSE<-round(sqrt(sum(((vec[,1]-vec[,2])^2)/length(vec[,1]))),4)
MAPE<-round(mean(abs(vec[,1]-vec[,2])/vec[,1]),4)
paste("Accuracy Measures: RMSE: ",RMSE,"and MAPE: ",MAPE)
```

```
## [1] "Accuracy Measures: RMSE:  61.0551 and MAPE:  0.0851"
```

Using arima to forecast values of 2015, 2016 & 2017

```r
ARIMAfit = auto.arima(log10(sales_ts), approximation=FALSE,trace=FALSE)
summary(ARIMAfit)
```

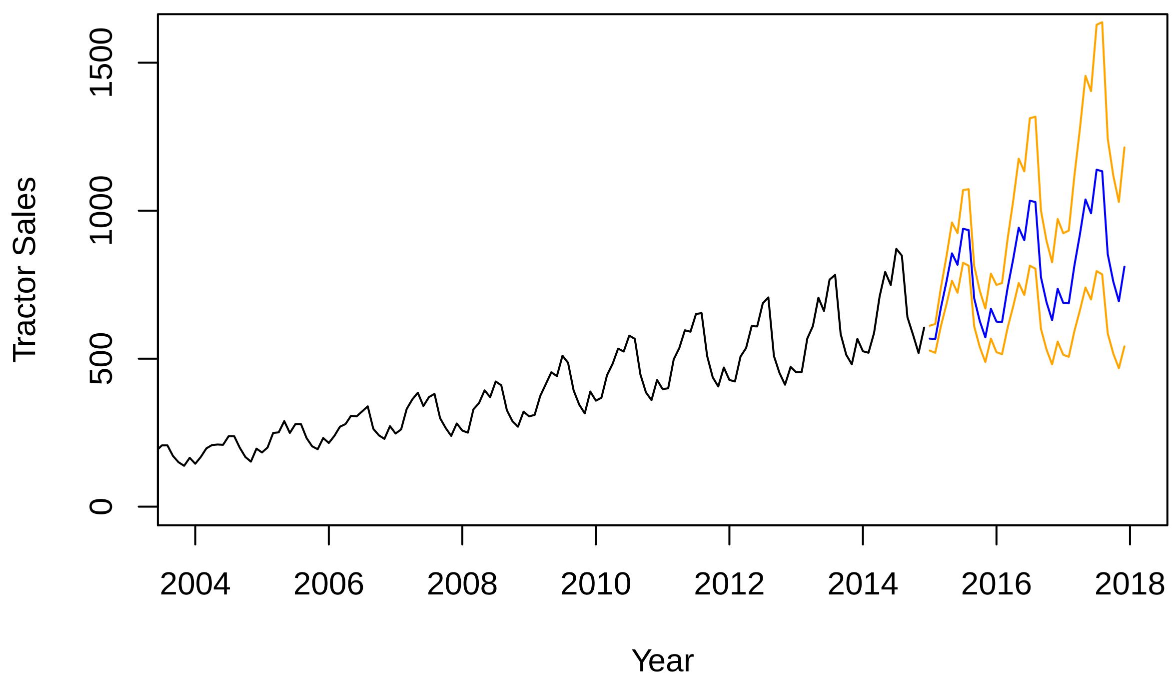
```
## Series: log10(sales_ts)
## ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.4047  -0.5529
## s.e.   0.0885   0.0734
##
## sigma^2 estimated as 0.0002571:  log likelihood=354.4
## AIC=-702.79   AICc=-702.6   BIC=-694.17
##
## Training set error measures:
##                        ME       RMSE        MAE         MPE      MAPE      MASE
## Training set 0.0002410698 0.01517695 0.01135312 0.008335713 0.4462212 0.2158968
##                    ACF1
## Training set 0.01062604
```

Forecasting:

```
par(mfrow = c(1,1))
pred = predict(ARIMAfit, n.ahead = 36) #36 months makes 3 years
pred
```

```
## $pred
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2015 2.754168 2.753182 2.826608 2.880192 2.932447 2.912372 2.972538 2.970585
## 2016 2.796051 2.795065 2.868491 2.922075 2.974330 2.954255 3.014421 3.012468
## 2017 2.837934 2.836948 2.910374 2.963958 3.016213 2.996138 3.056304 3.054351
##           Sep      Oct      Nov      Dec
## 2015 2.847264 2.797259 2.757395 2.825125
## 2016 2.889147 2.839142 2.799278 2.867008
## 2017 2.931030 2.881025 2.841161 2.908891
##
## $se
##             Jan        Feb        Mar        Apr        May        Jun
## 2015 0.01603508 0.01866159 0.02096153 0.02303295 0.02493287 0.02669792
## 2016 0.03923008 0.04159145 0.04382576 0.04595157 0.04798329 0.04993241
## 2017 0.06386474 0.06637555 0.06879478 0.07113179 0.07339441 0.07558934
##             Jul        Aug        Sep        Oct        Nov        Dec
## 2015 0.02835330 0.02991723 0.03140337 0.03282229 0.03418236 0.03549035
## 2016 0.05180825 0.05361850 0.05536960 0.05706700 0.05871534 0.06031866
## 2017 0.07772231 0.07979828 0.08182160 0.08379608 0.08572510 0.08761165
```

```
plot(sales_ts,type='l',xlim=c(2004,2018),ylim=c(1,1600),xlab = 'Year',ylab = 'Tractor Sales')
lines(10^(pred$pred),col='blue')
lines(10^(pred$pred+2*pred$se),col='orange')
lines(10^(pred$pred-2*pred$se),col='orange')
```

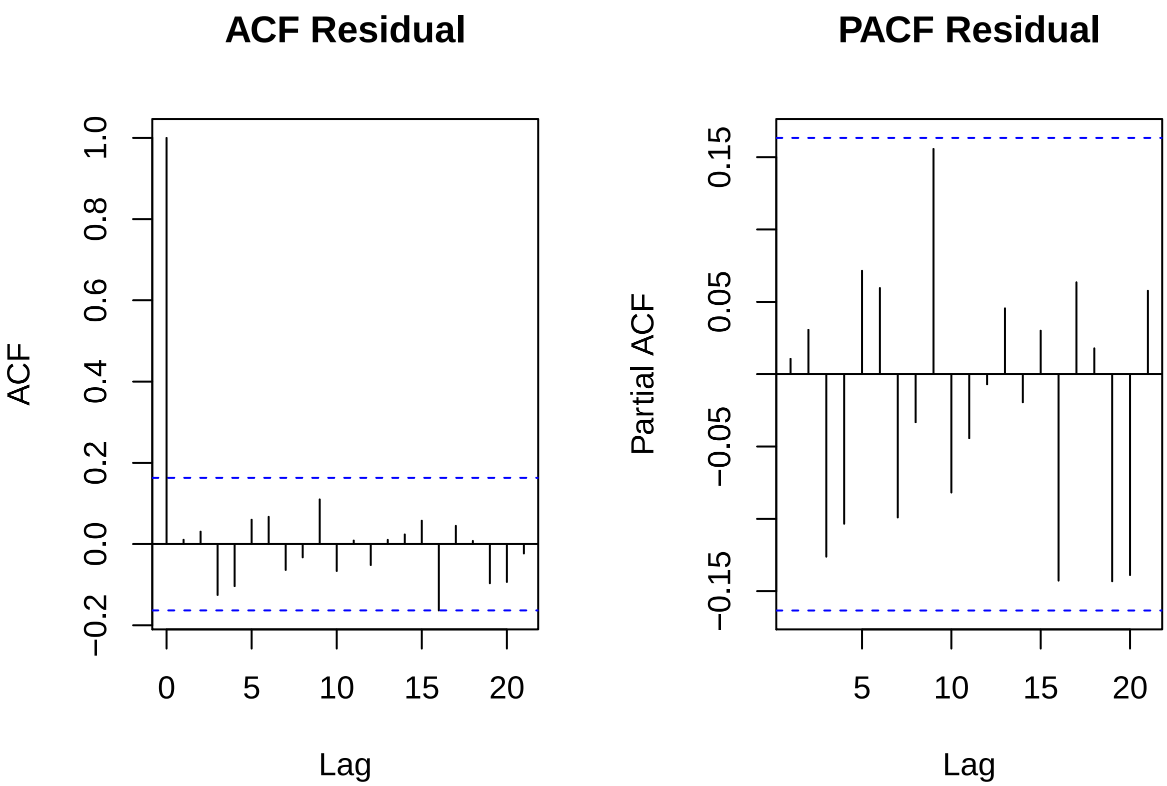


The above is the output with forecasted values of tractor sales in blue. Also, the range of expected error (i.e. 2 times standard deviation) is displayed with orange lines on either side of predicted blue line.

Assumptions while forecasting: Forecasts for a long period of 3 years is an ambitious task. The major assumption here is that the underlining patterns in the time series will continue to stay the same as predicted in the model. A short-term forecasting model, say a couple of business quarters or a year, is usually a good idea to forecast with reasonable accuracy. A long-term model like the one above needs to evaluated on a regular interval of time (say 6 months). The idea is to incorporate the new information available with the passage of time in the model.

Plot ACF and PACF for residuals of ARIMA model to ensure no more information is left for extraction:

```r
par(mfrow=c(1,2))
acf(ts(ARIMAfit$residuals),main='ACF Residual')
pacf(ts(ARIMAfit$residuals),main='PACF Residual')
```



Since there are no spikes outside the insignificant zone for both ACF and PACF plots we can conclude that residuals are random with no information or juice in them. Hence our ARIMA model is working good and predictions were successfully made.

Inference:

- Tractor sales seem to have both seasonality and trend.
- The sales are on average more during the months of July and August compared to other months by about 23% of annual trend.
- If the company has a high budget,we can confidently plan the production to be the forecasted sales +/- 10%.

- If costs are high and company has a low budget, then we can plan the production to be 450 and be confident all tractors are going to be sold.
- Furthermore, a cost matrix and cost curve can be used to minimize the costs considering both cases.