Assignment5_APrakash

2023-10-29

```
#Business Forecasting Assignment 5 - Decomposition - Aarthi Prakash

#Creating Time series from data points
library(fpp)
```

```
## Loading required package: forecast
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
## Loading required package: fma
```

```
## Loading required package: expsmooth
```

```
## Loading required package: lmtest
```

```
## Loading required package: zoo
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
## Loading required package: tseries
```

```
library(fpp2)
```

```
## ── Attaching packages ─────────────────────────────────────── fpp2 2.5 ──
```

```
## ✔ ggplot2 3.4.3
```

```
##
```

```
##
## Attaching package: 'fpp2'
```

```
## The following objects are masked from 'package:fpp':
##
##     ausair, ausbeer, austa, austourists, debitcards, departures,
##     elecequip, euretail, guinearice, oil, sunspotarea, usmelec
```

```
#Import Updated Data
TotalRevenueForHospitalsData <- read.csv("~/Documents/Rutgers MBA 2022/Fall 2023/[0] Bus
iness Forecasting/Week 3/Assignment3/TotalRevenueForHospitalsData.csv", header=FALSE)
View(TotalRevenueForHospitalsData)

TotalRevenueForHospitalsData = TotalRevenueForHospitalsData[-1,]
TotalRevenueForHospitalsData$V2 <- as.numeric(TotalRevenueForHospitalsData$V2)
#colnames(TotalRevenueForHospitalsData) <- ('Date','TotalRevenue')
str(TotalRevenueForHospitalsData)
```

```
## 'data.frame':    75 obs. of  2 variables:
##  $ V1: chr  "10/1/04" "1/1/05" "4/1/05" "7/1/05" ...
##  $ V2: num  150574 152362 153319 157568 157783 ...
```

```
HospitalRevenue_ts <- ts(TotalRevenueForHospitalsData$V2, frequency = 4, start = c(2004,
4))
HospitalRevenue_ts
```

```
##        Qtr1   Qtr2   Qtr3   Qtr4
## 2004                      150574
## 2005 152362 153319 157568 157783
## 2006 160081 162764 165671 169284
## 2007 172192 174276 177107 179300
## 2008 182691 182492 183249 183727
## 2009 189127 193135 196132 196246
## 2010 194837 198871 203507 205443
## 2011 207047 209254 207742 213088
## 2012 218241 217307 221418 219884
## 2013 221548 223201 224781 228865
## 2014 225710 231055 236131 240621
## 2015 245589 245955 246042 246142
## 2016 251591 258288 257860 263405
## 2017 264861 263605 269610 272875
## 2018 273332 278449 284171 279932
## 2019 292520 296140 298630 304038
## 2020 290093 285250 312235 324531
## 2021 320690 333048 336951 339175
## 2022 323197 312036 332687 350285
## 2023 354716 361160
```

```
#Decomposition Commands
#stl_decomp Takes 2 Arguments
?stl
stl_decomp <- stl(HospitalRevenue_ts,s.window ="periodic")
#needs time series for decomposition, s.window is either periodic or has to be odd and > 7
#Generally use periodic always

# Table View
stl_decomp
```

```
##  Call:
##  stl(x = HospitalRevenue_ts, s.window = "periodic")
##
## Components
##           seasonal    trend    remainder
## 2004 Q4  1181.5959 150182.5   -790.09780
## 2005 Q1  -581.0745 152250.7    692.32837
## 2005 Q2 -1387.7900 154269.9    436.88701
## 2005 Q3   787.2590 156207.4    573.34152
## 2005 Q4  1181.5959 158254.4  -1652.97089
## 2006 Q1  -581.0745 160549.7    112.33701
## 2006 Q2 -1387.7900 163084.6   1067.23861
## 2006 Q3   787.2590 165888.3  -1004.52105
## 2006 Q4  1181.5959 168836.0   -733.55593
## 2007 Q1  -581.0745 171882.2    890.85941
## 2007 Q2 -1387.7900 174579.9   1083.92861
## 2007 Q3   787.2590 176953.5   -633.72414
## 2007 Q4  1181.5959 179309.4  -1191.03964
## 2008 Q1  -581.0745 181329.9   1942.20598
## 2008 Q2 -1387.7900 182650.4   1229.34970
## 2008 Q3   787.2590 183652.9  -1191.11703
## 2008 Q4  1181.5959 185692.3  -3146.85788
## 2009 Q1  -581.0745 188945.0    763.03790
## 2009 Q2 -1387.7900 192354.9   2167.89268
## 2009 Q3   787.2590 194505.5    839.24666
## 2009 Q4  1181.5959 195692.5   -628.08390
## 2010 Q1  -581.0745 197291.7  -1873.63498
## 2010 Q2 -1387.7900 199529.4    729.38818
## 2010 Q3   787.2590 202244.6    475.15756
## 2010 Q4  1181.5959 204967.2   -705.81470
## 2011 Q1  -581.0745 206979.1    648.96207
## 2011 Q2 -1387.7900 208411.9   2229.84034
## 2011 Q3   787.2590 210424.3  -3469.58119
## 2011 Q4  1181.5959 212962.6  -1056.22047
## 2012 Q1  -581.0745 216010.0   2812.12267
## 2012 Q2 -1387.7900 218545.3    149.46538
## 2012 Q3   787.2590 219608.9   1021.80175
## 2012 Q4  1181.5959 220663.5  -1961.14148
## 2013 Q1  -581.0745 221917.8    211.26077
## 2013 Q2 -1387.7900 223526.3   1062.52976
## 2013 Q3   787.2590 225135.6  -1141.82380
## 2013 Q4  1181.5959 226547.0   1136.38378
## 2014 Q1  -581.0745 228901.2  -2610.08215
## 2014 Q2 -1387.7900 231833.2    609.60822
## 2014 Q3   787.2590 235827.5   -483.76421
## 2014 Q4  1181.5959 240205.7   -766.30113
## 2015 Q1  -581.0745 243570.3   2599.77943
## 2015 Q2 -1387.7900 245477.6   1865.14630
## 2015 Q3   787.2590 246498.3  -1243.56563
## 2015 Q4  1181.5959 248595.5  -3635.12749
## 2016 Q1  -581.0745 252045.2    126.89845
## 2016 Q2 -1387.7900 255867.1   3808.66181
```

```
## 2016 Q3   787.2590 259407.7  -2334.98754
## 2016 Q4  1181.5959 261750.3    473.15199
## 2017 Q1  -581.0745 264002.5   1439.54522
## 2017 Q2 -1387.7900 266483.3  -1490.47159
## 2017 Q3   787.2590 268747.8     74.89543
## 2017 Q4  1181.5959 271670.8     22.55780
## 2018 Q1  -581.0745 275301.0  -1387.93538
## 2018 Q2 -1387.7900 278304.0   1532.83942
## 2018 Q3   787.2590 281311.1   2072.63515
## 2018 Q4  1181.5959 285605.1  -6854.66549
## 2019 Q1  -581.0745 290058.9   3042.19164
## 2019 Q2 -1387.7900 295128.0   2399.79442
## 2019 Q3   787.2590 297855.7    -12.99790
## 2019 Q4  1181.5959 296291.7   6564.70114
## 2020 Q1  -581.0745 295558.3  -4884.27279
## 2020 Q2 -1387.7900 299329.4 -12691.64555
## 2020 Q3   787.2590 307042.6   4405.13158
## 2020 Q4  1181.5959 317201.2   6148.18687
## 2021 Q1  -581.0745 325686.4  -4415.30717
## 2021 Q2 -1387.7900 330750.3   3685.49584
## 2021 Q3   787.2590 333460.5   2703.23713
## 2021 Q4  1181.5959 331065.4   6927.95974
## 2022 Q1  -581.0745 326819.8  -3041.76035
## 2022 Q2 -1387.7900 326843.6 -13419.84181
## 2022 Q3   787.2590 333101.9  -1202.18544
## 2022 Q4  1181.5959 343727.3   5376.09509
## 2023 Q1  -581.0745 354489.3    807.74592
## 2023 Q2 -1387.7900 365134.5  -2586.68294
```

```
#Prints a lot of data - seasonality, trend, remainder = C + I together
attributes(stl_decomp) #see other values this stores -
```

```
## $names
## [1] "time.series" "weights"     "call"        "win"         "deg"
## [6] "jump"        "inner"       "outer"
##
## $class
## [1] "stl"
```

```
stl_decomp$time.series
```

```
##           seasonal    trend    remainder
## 2004 Q4  1181.5959 150182.5   -790.09780
## 2005 Q1  -581.0745 152250.7    692.32837
## 2005 Q2 -1387.7900 154269.9    436.88701
## 2005 Q3   787.2590 156207.4    573.34152
## 2005 Q4  1181.5959 158254.4  -1652.97089
## 2006 Q1  -581.0745 160549.7    112.33701
## 2006 Q2 -1387.7900 163084.6   1067.23861
## 2006 Q3   787.2590 165888.3  -1004.52105
## 2006 Q4  1181.5959 168836.0   -733.55593
## 2007 Q1  -581.0745 171882.2    890.85941
## 2007 Q2 -1387.7900 174579.9   1083.92861
## 2007 Q3   787.2590 176953.5   -633.72414
## 2007 Q4  1181.5959 179309.4  -1191.03964
## 2008 Q1  -581.0745 181329.9   1942.20598
## 2008 Q2 -1387.7900 182650.4   1229.34970
## 2008 Q3   787.2590 183652.9  -1191.11703
## 2008 Q4  1181.5959 185692.3  -3146.85788
## 2009 Q1  -581.0745 188945.0    763.03790
## 2009 Q2 -1387.7900 192354.9   2167.89268
## 2009 Q3   787.2590 194505.5    839.24666
## 2009 Q4  1181.5959 195692.5   -628.08390
## 2010 Q1  -581.0745 197291.7  -1873.63498
## 2010 Q2 -1387.7900 199529.4    729.38818
## 2010 Q3   787.2590 202244.6    475.15756
## 2010 Q4  1181.5959 204967.2   -705.81470
## 2011 Q1  -581.0745 206979.1    648.96207
## 2011 Q2 -1387.7900 208411.9   2229.84034
## 2011 Q3   787.2590 210424.3  -3469.58119
## 2011 Q4  1181.5959 212962.6  -1056.22047
## 2012 Q1  -581.0745 216010.0   2812.12267
## 2012 Q2 -1387.7900 218545.3    149.46538
## 2012 Q3   787.2590 219608.9   1021.80175
## 2012 Q4  1181.5959 220663.5  -1961.14148
## 2013 Q1  -581.0745 221917.8    211.26077
## 2013 Q2 -1387.7900 223526.3   1062.52976
## 2013 Q3   787.2590 225135.6  -1141.82380
## 2013 Q4  1181.5959 226547.0   1136.38378
## 2014 Q1  -581.0745 228901.2  -2610.08215
## 2014 Q2 -1387.7900 231833.2    609.60822
## 2014 Q3   787.2590 235827.5   -483.76421
## 2014 Q4  1181.5959 240205.7   -766.30113
## 2015 Q1  -581.0745 243570.3   2599.77943
## 2015 Q2 -1387.7900 245477.6   1865.14630
## 2015 Q3   787.2590 246498.3  -1243.56563
## 2015 Q4  1181.5959 248595.5  -3635.12749
## 2016 Q1  -581.0745 252045.2    126.89845
## 2016 Q2 -1387.7900 255867.1   3808.66181
## 2016 Q3   787.2590 259407.7  -2334.98754
## 2016 Q4  1181.5959 261750.3    473.15199
## 2017 Q1  -581.0745 264002.5   1439.54522
## 2017 Q2 -1387.7900 266483.3  -1490.47159
```

```
## 2017 Q3   787.2590 268747.8     74.89543
## 2017 Q4  1181.5959 271670.8     22.55780
## 2018 Q1  -581.0745 275301.0  -1387.93538
## 2018 Q2 -1387.7900 278304.0   1532.83942
## 2018 Q3   787.2590 281311.1   2072.63515
## 2018 Q4  1181.5959 285605.1  -6854.66549
## 2019 Q1  -581.0745 290058.9   3042.19164
## 2019 Q2 -1387.7900 295128.0   2399.79442
## 2019 Q3   787.2590 297855.7    -12.99790
## 2019 Q4  1181.5959 296291.7   6564.70114
## 2020 Q1  -581.0745 295558.3  -4884.27279
## 2020 Q2 -1387.7900 299329.4 -12691.64555
## 2020 Q3   787.2590 307042.6   4405.13158
## 2020 Q4  1181.5959 317201.2   6148.18687
## 2021 Q1  -581.0745 325686.4  -4415.30717
## 2021 Q2 -1387.7900 330750.3   3685.49584
## 2021 Q3   787.2590 333460.5   2703.23713
## 2021 Q4  1181.5959 331065.4   6927.95974
## 2022 Q1  -581.0745 326819.8  -3041.76035
## 2022 Q2 -1387.7900 326843.6 -13419.84181
## 2022 Q3   787.2590 333101.9  -1202.18544
## 2022 Q4  1181.5959 343727.3   5376.09509
## 2023 Q1  -581.0745 354489.3    807.74592
## 2023 Q2 -1387.7900 365134.5  -2586.68294
```

```
stl_decomp$weights
```

```
##  [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
## [39] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
```

```
stl_decomp$call
```

```
## stl(x = HospitalRevenue_ts, s.window = "periodic")
```

```
stl_decomp$win
```

```
##   s   t   l
## 751   7   5
```

```
stl_decomp$deg
```

```
## s t l
## 0 1 1
```

```
stl_decomp$jump
```

```
##  s  t  l
## 76  1  1
```

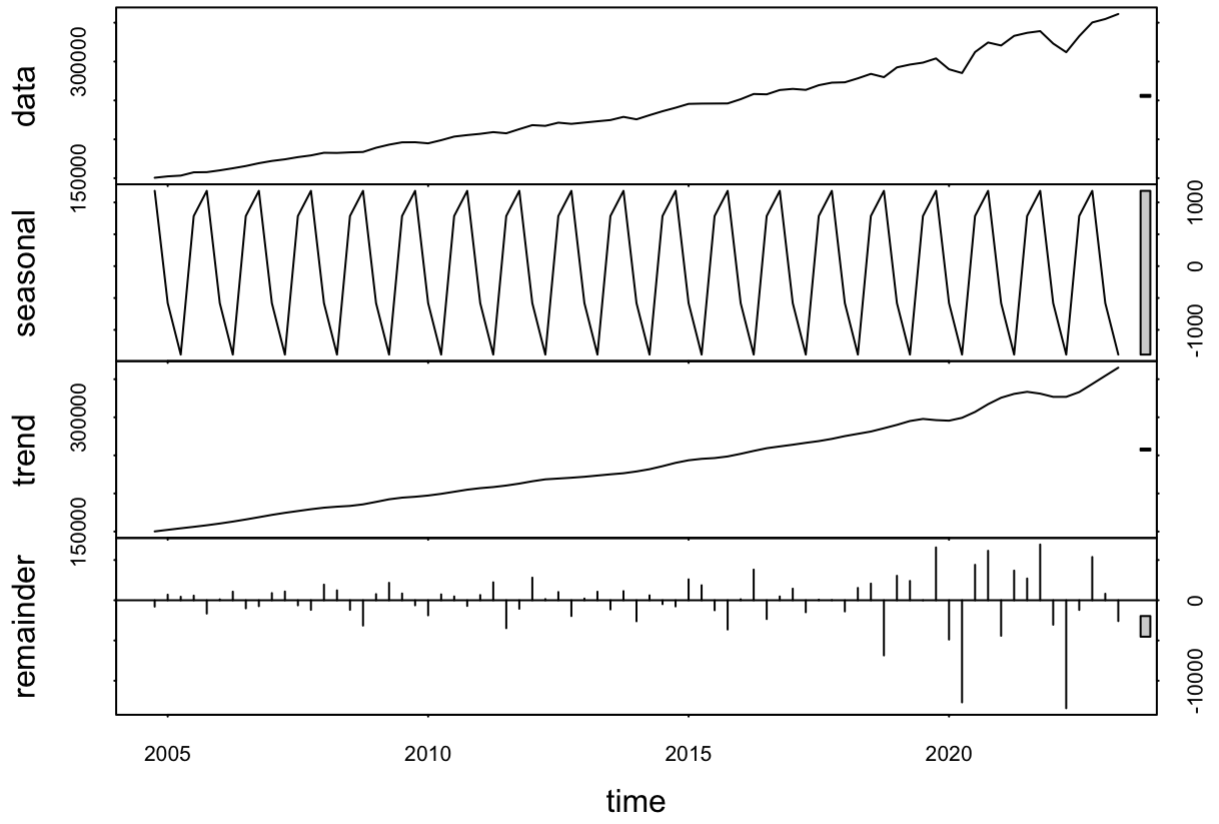
```
stl_decomp$inner
```

```
## [1] 2
```

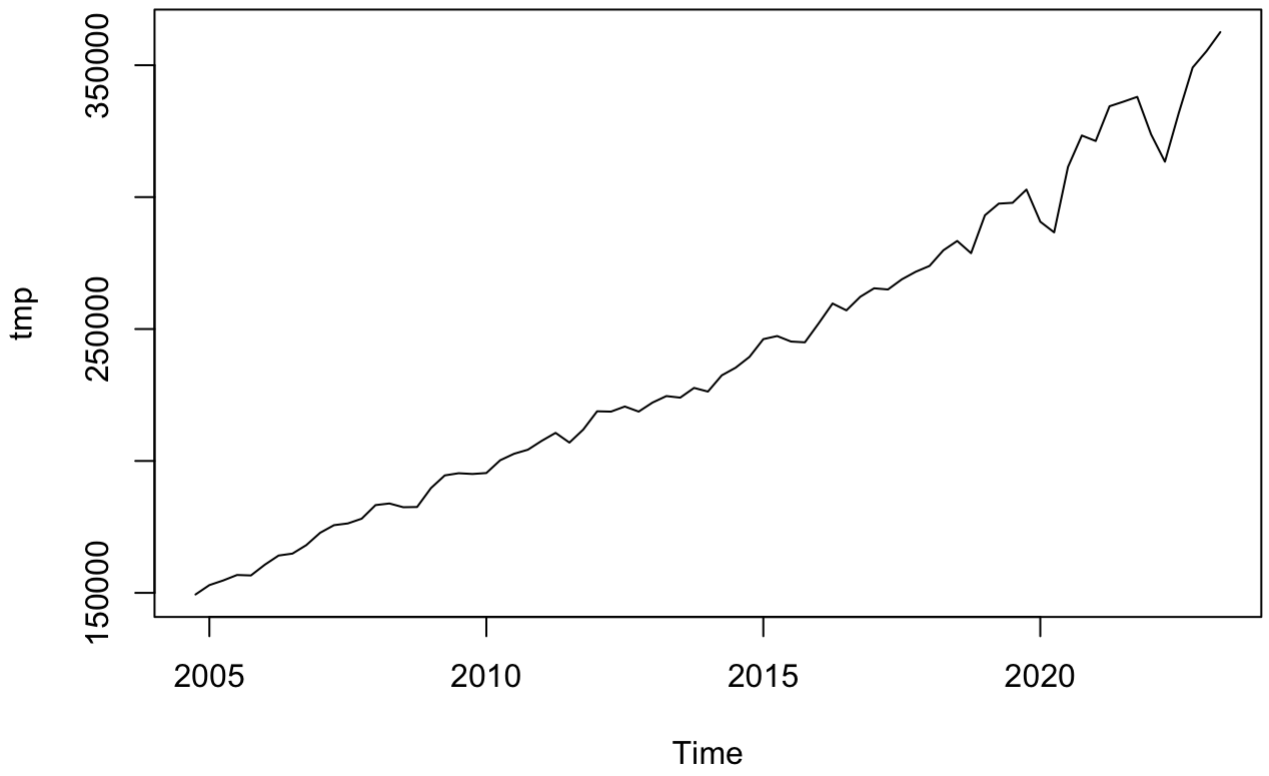
```
stl_decomp$outer
```

```
## [1] 0
```

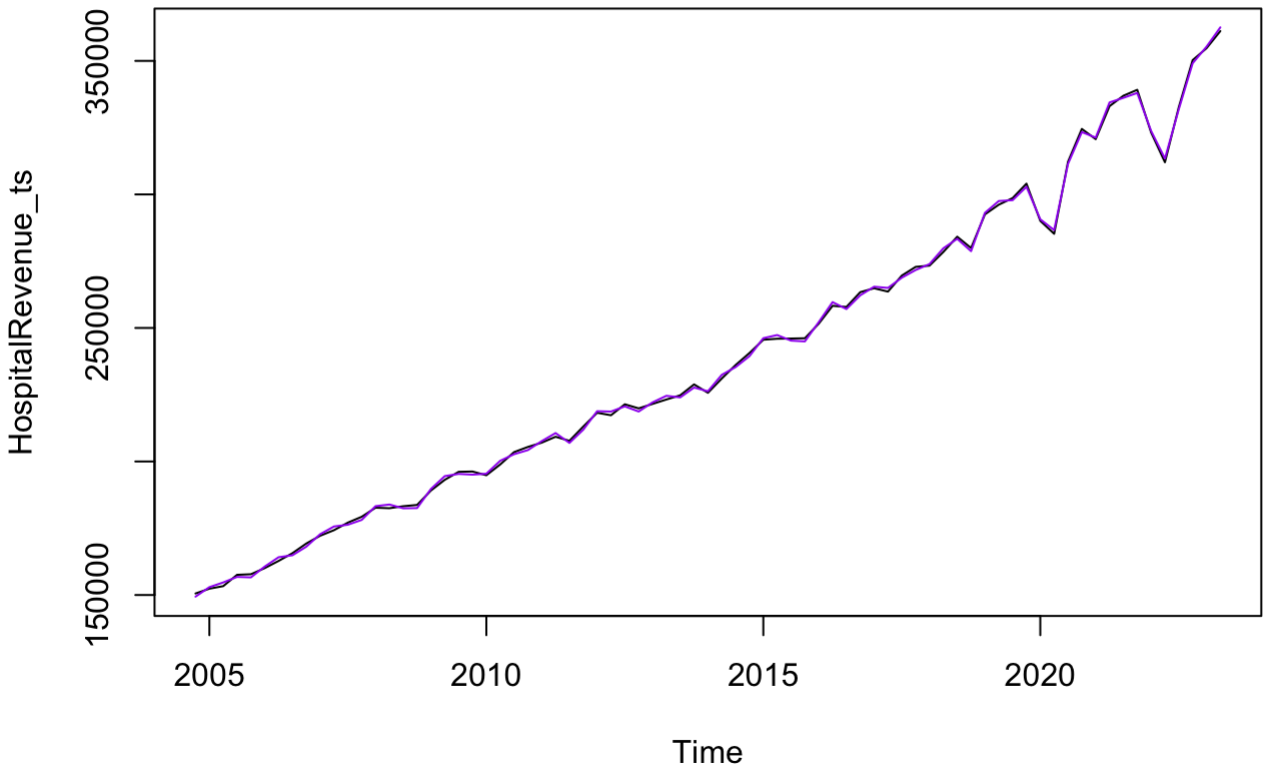
```
# Graph View
plot(stl_decomp)
```



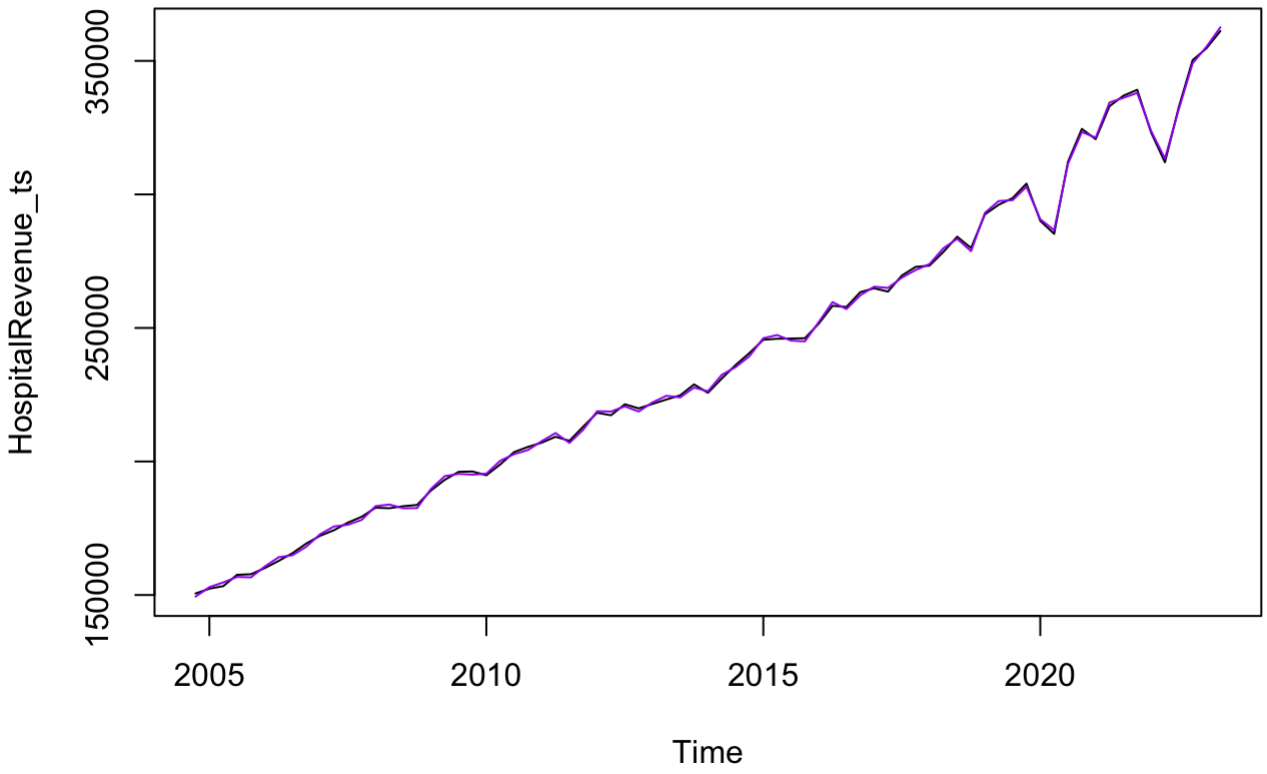
```
#How to read graph
#Find the highest and lowest in trend - 15000 to 30000 - trend is upwards and very simil
ar to the data

#Print out a seasonal adjustment
tmp <- seasadj(stl_decomp)
# shows seasonally adjusted data
plot(tmp) #plots seasonaly adjusted data
```




```
# Plot a line on the graph
plot(HospitalRevenue_ts)
lines(seasadj(stl_decomp), col = 'purple')
```



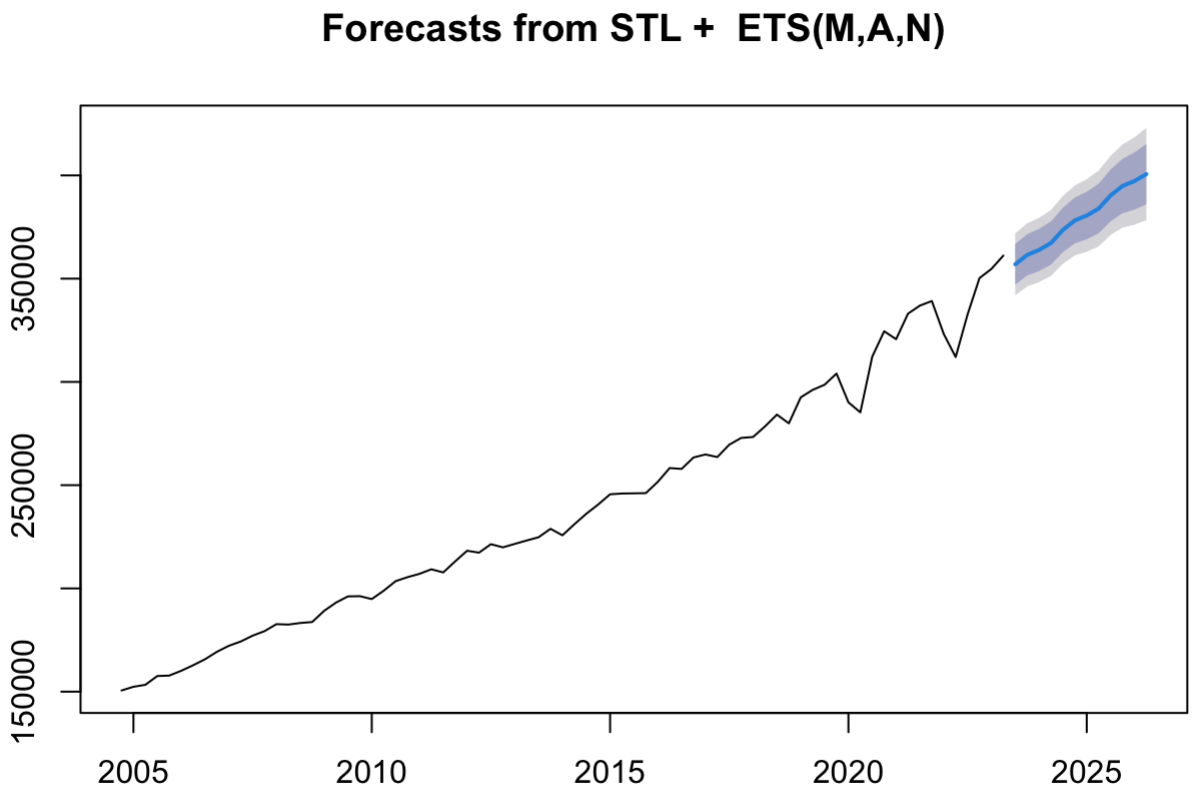

```
plot(HospitalRevenue_ts)
lines(tmp, col = 'purple')
```




```
#When plotting the original time series and the seasonally adjusted line - they are very
similar

#Forecast with the seasonally adjusted data
#Default period forecast
f_stl <- forecast(stl_decomp)

# you can pass the # of period - 15 months
f_stl <- forecast(stl_decomp,h=12)

#Print it out or graph it
f_stl
```

```
##          Point Forecast    Lo 80    Hi 80    Lo 95    Hi 95
## 2023 Q3       356940.0 347126.2 366753.7 341931.1 371948.8
## 2023 Q4       361503.1 351525.8 371480.3 346244.1 376762.0
## 2024 Q1       363909.1 353732.7 374085.5 348345.7 379472.6
## 2024 Q2       367271.2 356851.5 377690.8 351335.7 383206.6
## 2024 Q3       373614.9 362900.4 384329.4 357228.5 390001.4
## 2024 Q4       378178.0 367110.8 389245.3 361252.2 395103.9
## 2025 Q1       380584.1 369101.3 392066.8 363022.7 398145.5
## 2025 Q2       383946.1 371981.6 395910.7 365647.9 402244.3
## 2025 Q3       390289.9 377775.1 402804.7 371150.2 409429.6
## 2025 Q4       394853.0 381718.6 407987.4 374765.7 414940.3
## 2026 Q1       397259.1 383435.8 411082.3 376118.2 418399.9
## 2026 Q2       400621.1 386040.5 415201.7 378322.0 422920.2
```

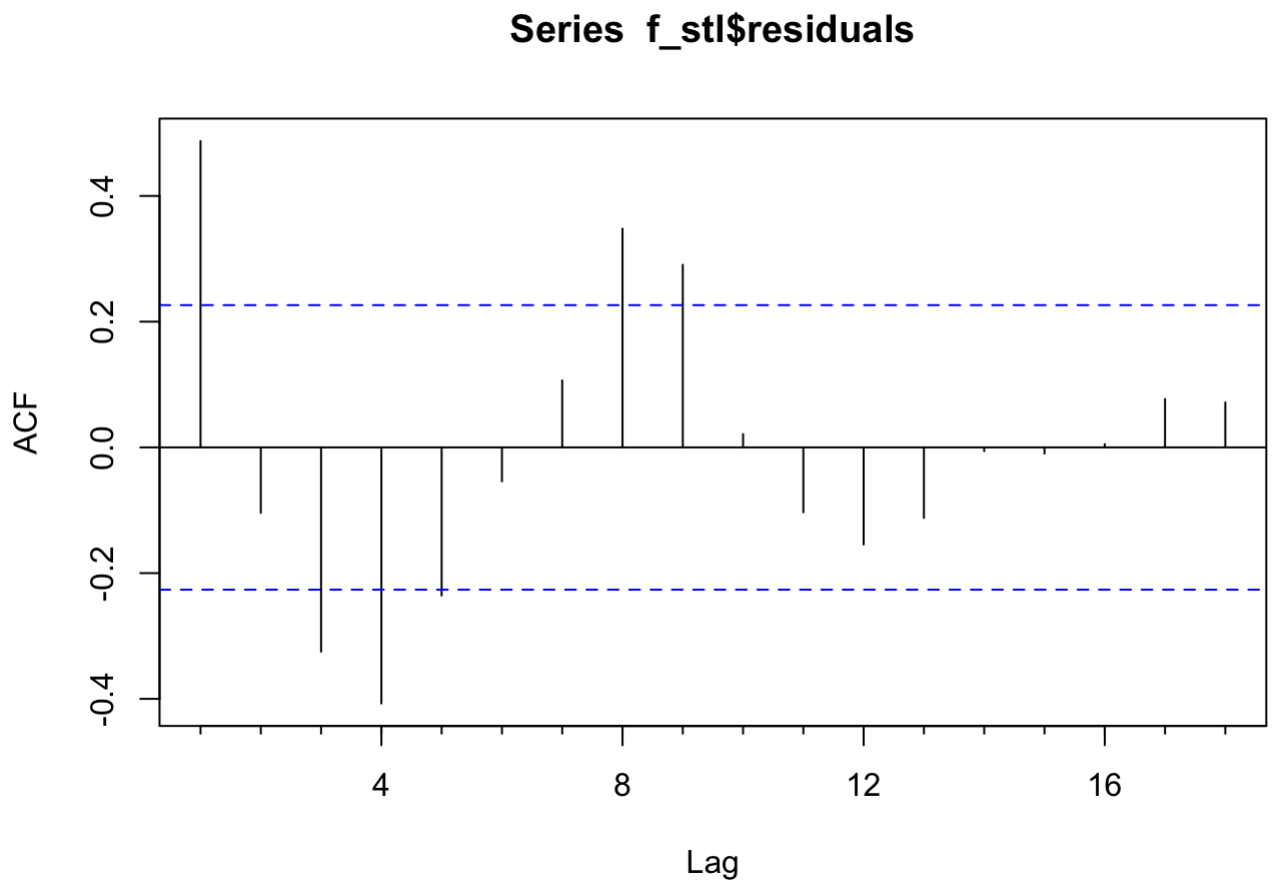
```
plot(f_stl)
```



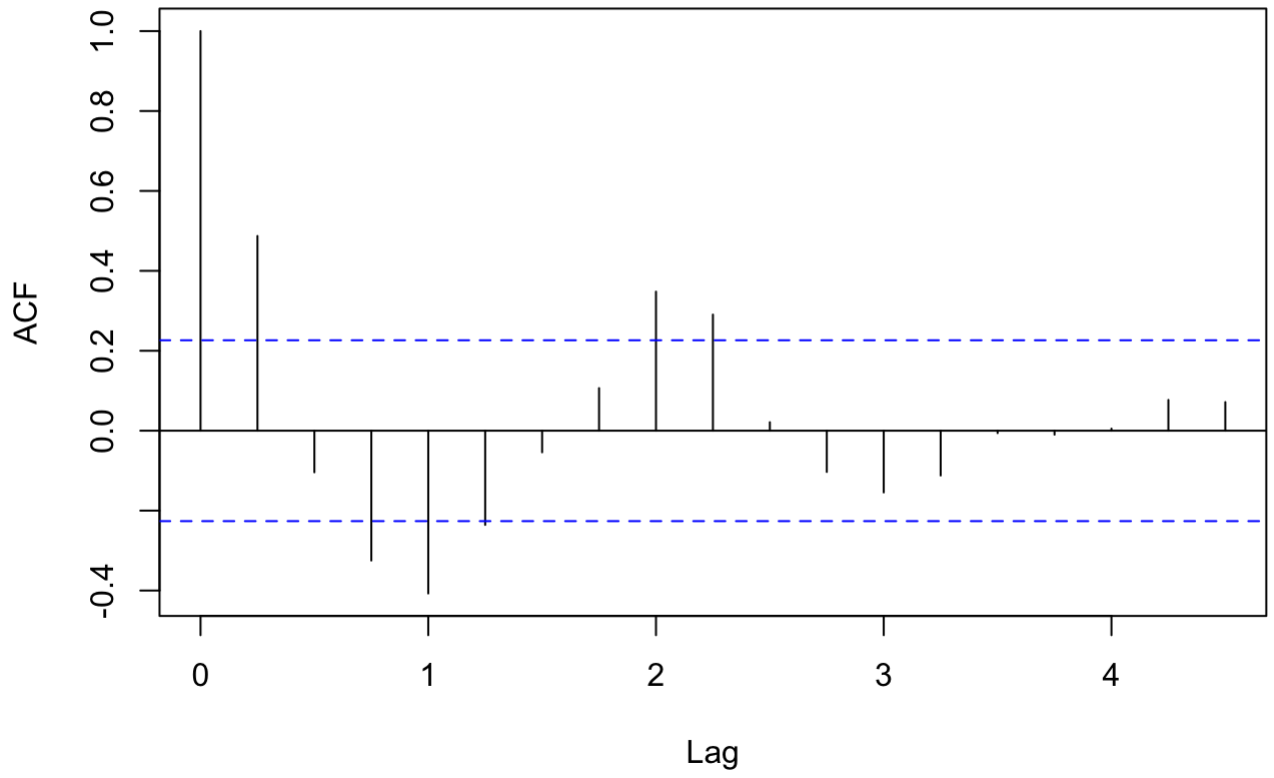

```
#plot is synthesizing using the components - starts looking like the historical data its elf
accuracy(f_stl) #MAPE is 1.523904 which seems pretty good
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 733.793 6197.329 4105.288 0.2036045 1.523904 0.3380558 0.4872655
```

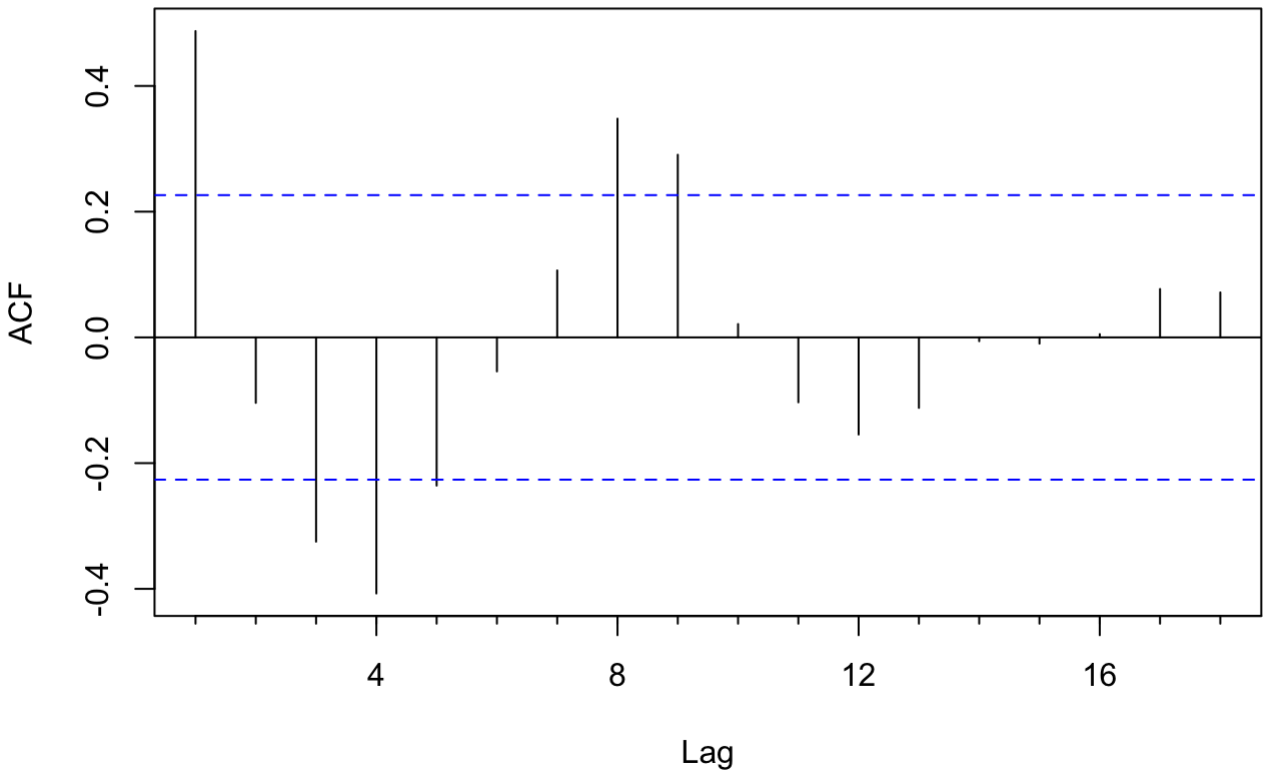
```
#Residual analysis for forecast
Acf(f_stl$residuals) #Removes the lag 0 which has the original data and will always be 1
```




```
acf(f_stl$residuals)
```




```
tmp <- Acf(f_stl$residuals)
```



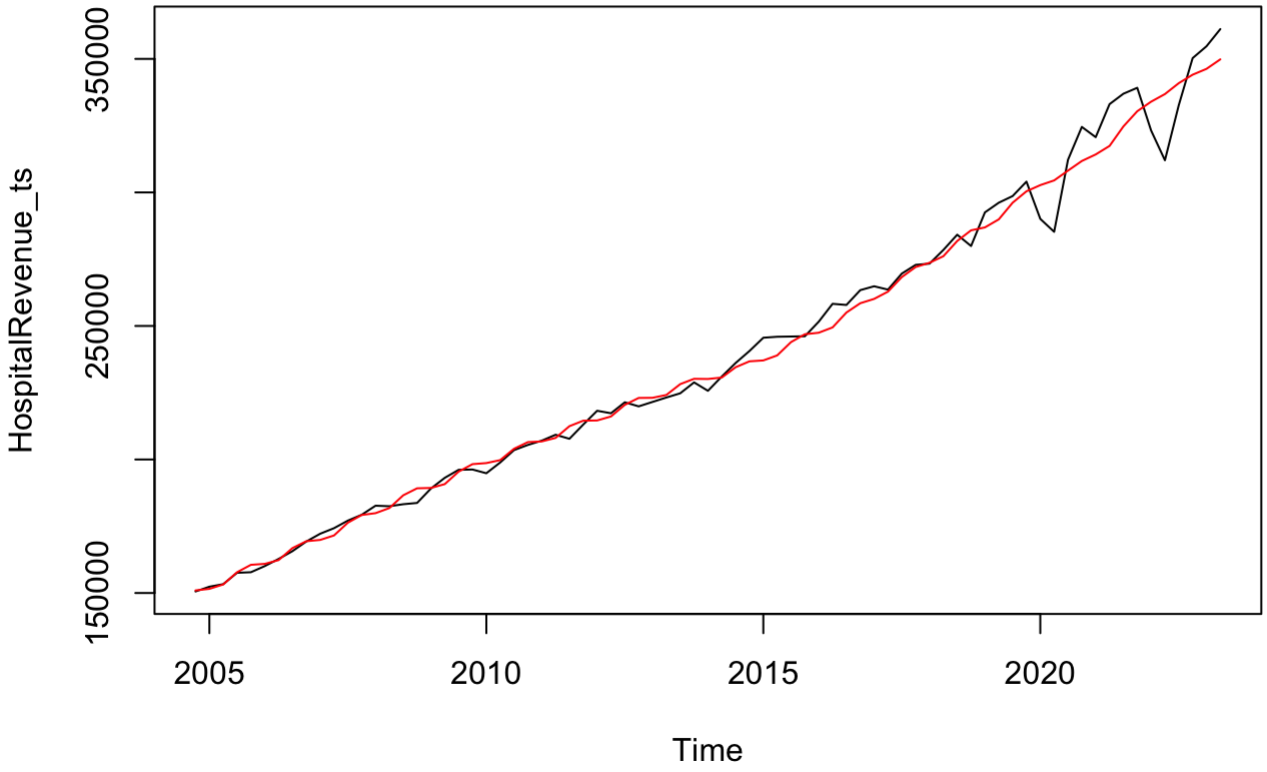

```
tmp
```

```
##
## Autocorrelations of series 'f_stl$residuals', by lag
##
##      0      1      2      3      4      5      6      7      8      9     10
##  1.000  0.487 -0.104 -0.325 -0.407 -0.236 -0.054  0.107  0.348  0.291  0.021
##     11     12     13     14     15     16     17     18
## -0.103 -0.155 -0.112 -0.006 -0.010  0.005  0.077  0.072
```

```
#Shows that there is something non-random happening in M3, M4, M6
f_stl$method
```

```
## [1] "STL +  ETS(M,A,N)"
```

```
#"STL +  ETS(M,A,N)"

#Plot original time series and what decomposition predicted
plot(HospitalRevenue_ts)
lines(f_stl$fitted, col = 'Red')
```




```
accuracy(f_stl) #MAPE is low at 1.523904
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 733.793 6197.329 4105.288 0.2036045 1.523904 0.3380558 0.4872655
```

```
#decomposition through stl function uses Loess function


#Another way to run decomposition model#
# There is more than one way to do things
decomp_elec <- decompose(HospitalRevenue_ts) #Underlying math is different from stl func
tion
attributes(decomp_elec)
```

```
## $names
## [1] "x"        "seasonal" "trend"    "random"   "figure"   "type"
##
## $class
## [1] "decomposed.ts"
```

```
decomp_elec
```

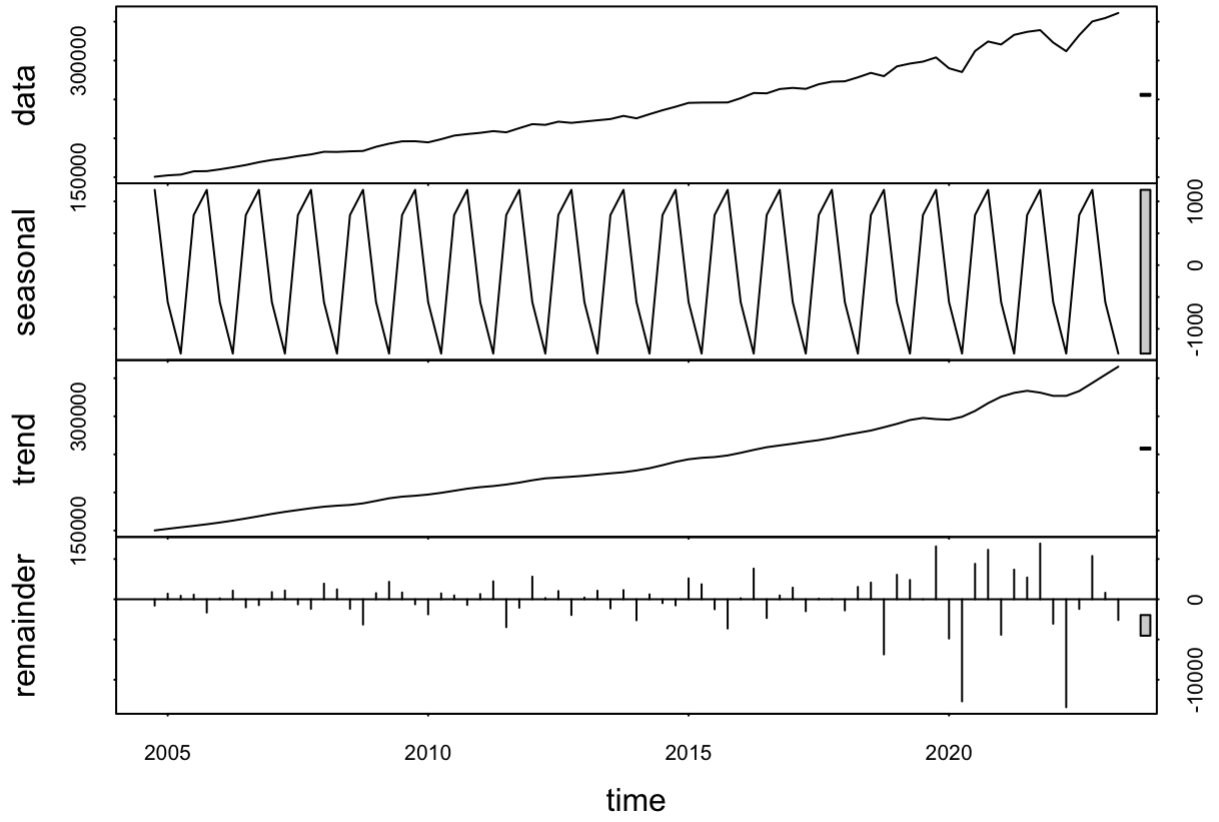
```
## $x
##        Qtr1   Qtr2   Qtr3   Qtr4
## 2004                       150574
## 2005 152362 153319 157568 157783
## 2006 160081 162764 165671 169284
## 2007 172192 174276 177107 179300
## 2008 182691 182492 183249 183727
## 2009 189127 193135 196132 196246
## 2010 194837 198871 203507 205443
## 2011 207047 209254 207742 213088
## 2012 218241 217307 221418 219884
## 2013 221548 223201 224781 228865
## 2014 225710 231055 236131 240621
## 2015 245589 245955 246042 246142
## 2016 251591 258288 257860 263405
## 2017 264861 263605 269610 272875
## 2018 273332 278449 284171 279932
## 2019 292520 296140 298630 304038
## 2020 290093 285250 312235 324531
## 2021 320690 333048 336951 339175
## 2022 323197 312036 332687 350285
## 2023 354716 361160
##
## $seasonal
##            Qtr1       Qtr2       Qtr3       Qtr4
## 2004                                     1482.7714
## 2005  -753.6407 -1553.1452   824.0145  1482.7714
## 2006  -753.6407 -1553.1452   824.0145  1482.7714
## 2007  -753.6407 -1553.1452   824.0145  1482.7714
## 2008  -753.6407 -1553.1452   824.0145  1482.7714
## 2009  -753.6407 -1553.1452   824.0145  1482.7714
## 2010  -753.6407 -1553.1452   824.0145  1482.7714
## 2011  -753.6407 -1553.1452   824.0145  1482.7714
## 2012  -753.6407 -1553.1452   824.0145  1482.7714
## 2013  -753.6407 -1553.1452   824.0145  1482.7714
## 2014  -753.6407 -1553.1452   824.0145  1482.7714
## 2015  -753.6407 -1553.1452   824.0145  1482.7714
## 2016  -753.6407 -1553.1452   824.0145  1482.7714
## 2017  -753.6407 -1553.1452   824.0145  1482.7714
## 2018  -753.6407 -1553.1452   824.0145  1482.7714
## 2019  -753.6407 -1553.1452   824.0145  1482.7714
## 2020  -753.6407 -1553.1452   824.0145  1482.7714
## 2021  -753.6407 -1553.1452   824.0145  1482.7714
## 2022  -753.6407 -1553.1452   824.0145  1482.7714
## 2023  -753.6407 -1553.1452
##
## $trend
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2004                                   NA
## 2005       NA 154356.9 156222.9 158368.4
## 2006 160561.9 163012.4 165963.9 168916.8
## 2007 171785.2 174466.8 177031.1 179370.5
```

```
## 2008 181165.2 182486.4 183844.2 185979.1
## 2009 188919.9 192095.1 194373.8 195804.5
## 2010 197443.4 199514.9 202190.8 205014.9
## 2011 206842.1 208327.1 210682.0 213087.9
## 2012 215804.0 218363.0 219625.9 220776.0
## 2013 221933.1 223476.1 225119.0 226621.0
## 2014 229021.5 231909.8 235864.1 240211.5
## 2015 243312.9 245241.9 246682.2 248974.1
## 2016 251993.0 255628.1 259444.8 261768.1
## 2017 263901.5 266554.0 268796.6 271711.0
## 2018 275386.6 278088.9 281369.5 285979.4
## 2019 289998.1 294818.8 297528.6 295864.0
## 2020 296203.4 300465.6 306851.9 316651.2
## 2021 325715.5 330635.5 332779.4 330466.2
## 2022 327306.8 328162.5 333491.1 343571.5
## 2023       NA       NA
##
## $random
##             Qtr1        Qtr2        Qtr3        Qtr4
## 2004                                             NA
## 2005          NA    515.2702    521.1105  -2068.1464
## 2006    272.7657   1304.7702  -1116.8895  -1115.5214
## 2007   1160.3907   1362.3952   -748.1395  -1553.2714
## 2008   2279.3907   1558.7702  -1419.2645  -3734.8964
## 2009    960.7657   2593.0202    934.2355  -1041.2714
## 2010  -1852.7343    909.2702    492.2355  -1054.6464
## 2011    958.5157   2480.0202  -3764.0145  -1482.6464
## 2012   3190.6407    497.1452    968.1105  -2374.7714
## 2013    368.5157   1278.0202  -1162.0145    761.2286
## 2014  -2557.8593    698.3952   -557.1395  -1073.2714
## 2015   3029.7657   2266.2702  -1464.2645  -4314.8964
## 2016    351.6407   4213.0202  -2408.7645    154.1036
## 2017   1713.1407  -1395.8548    -10.6395   -318.7714
## 2018  -1300.9843   1913.2702   1977.4855  -7530.1464
## 2019   3275.5157   2874.3952    277.3605   6691.2286
## 2020  -5356.7343 -13662.4798   4559.1105   6396.9786
## 2021  -4271.8593   3965.6452   3347.6105   7225.9786
## 2022  -3356.1093 -14573.3548  -1628.1395   5230.7286
## 2023          NA          NA
##
## $figure
## [1]  1482.7714  -753.6407 -1553.1452   824.0145
##
## $type
## [1] "additive"
##
## attr(,"class")
## [1] "decomposed.ts"
```

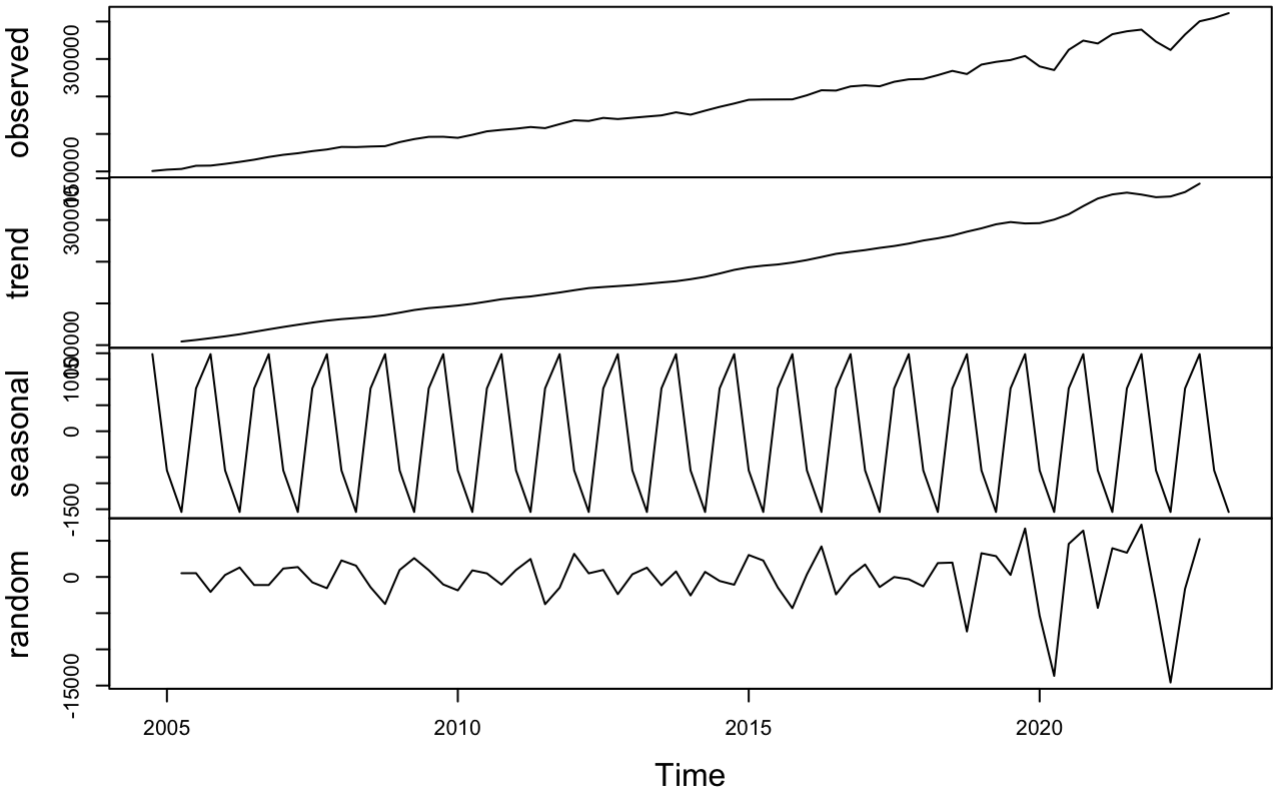
```
?decompose
decomp_elec$figure
```

```
## [1]  1482.7714  -753.6407 -1553.1452   824.0145
```

```
plot(stl_decomp)
```



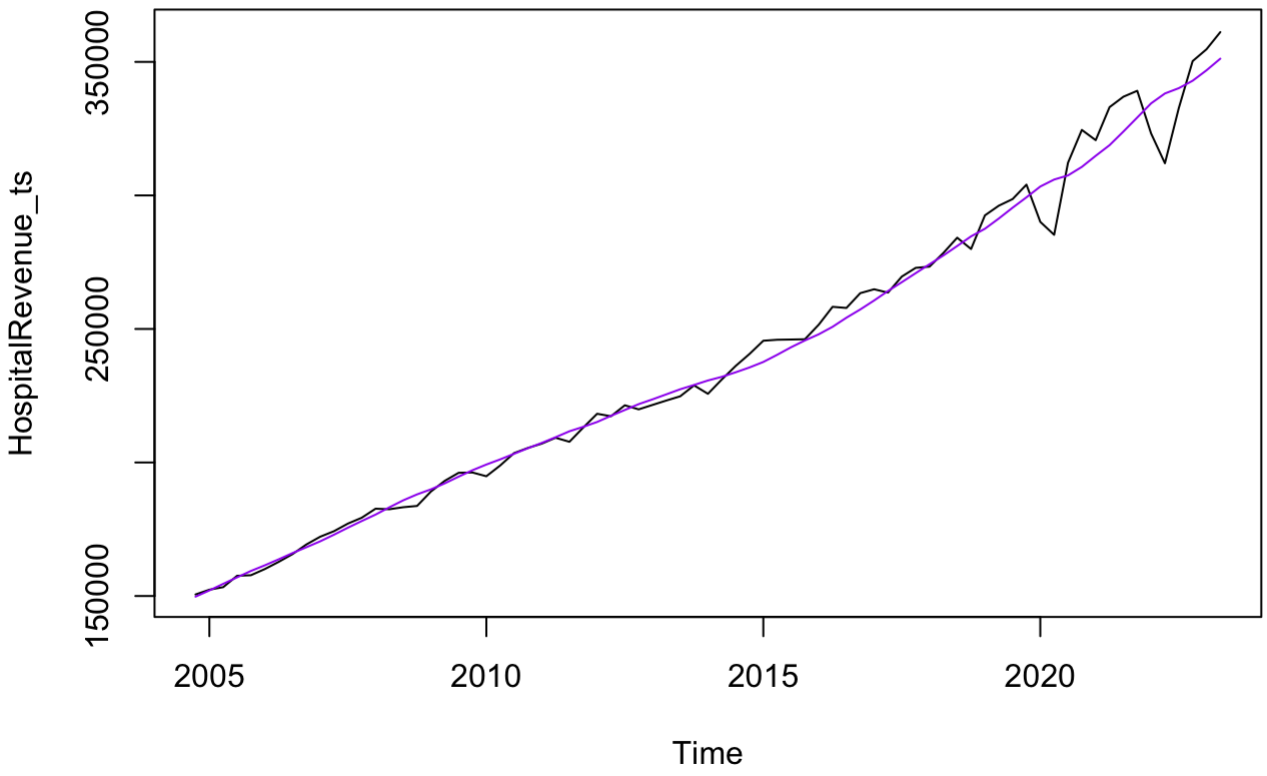

```
plot(decomp_elec) #Plot is little different from stl function
```




```
seasadj(decomp_elec)
```

```
##          Qtr1     Qtr2     Qtr3     Qtr4
## 2004                               149091.2
## 2005 153115.6 154872.1 156744.0 156300.2
## 2006 160834.6 164317.1 164847.0 167801.2
## 2007 172945.6 175829.1 176283.0 177817.2
## 2008 183444.6 184045.1 182425.0 182244.2
## 2009 189880.6 194688.1 195308.0 194763.2
## 2010 195590.6 200424.1 202683.0 203960.2
## 2011 207800.6 210807.1 206918.0 211605.2
## 2012 218994.6 218860.1 220594.0 218401.2
## 2013 222301.6 224754.1 223957.0 227382.2
## 2014 226463.6 232608.1 235307.0 239138.2
## 2015 246342.6 247508.1 245218.0 244659.2
## 2016 252344.6 259841.1 257036.0 261922.2
## 2017 265614.6 265158.1 268786.0 271392.2
## 2018 274085.6 280002.1 283347.0 278449.2
## 2019 293273.6 297693.1 297806.0 302555.2
## 2020 290846.6 286803.1 311411.0 323048.2
## 2021 321443.6 334601.1 336127.0 337692.2
## 2022 323950.6 313589.1 331863.0 348802.2
## 2023 355469.6 362713.1
```

```
#f_decomp <- forecast(decomp_elec,h=12) #Won't work - forecast only takes the stl compon
ent
f_decomp <- forecast(seasadj(decomp_elec),h=12) #BUT can forecast seasonally adjusted (b
ut takes seasonality out)

plot(HospitalRevenue_ts)
lines(f_decomp$fitted,col = 'purple')
```




```
accuracy(f_decomp) #MAPE = 1.550416
```

```
##                     ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 730.5078 6196.565 4150.932 0.2004013 1.550416 0.3418145 0.4878301
```

```
accuracy(f_stl) #MAPE = 1.523904
```

```
##                    ME     RMSE      MAE       MPE     MAPE      MASE      ACF1
## Training set 733.793 6197.329 4105.288 0.2036045 1.523904 0.3380558 0.4872655
```

```
#stl is better than decompositon - according to help it tells that the stl function is m
ore sophisticated than decomposition function


#HW - run on own data
#if data doesn't have seasonality you may get an error because data doesn't have seasona
lity
#Provide proof that there is no seasonality in data if that is true
```

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com (http://rmarkdown.rstudio.com).

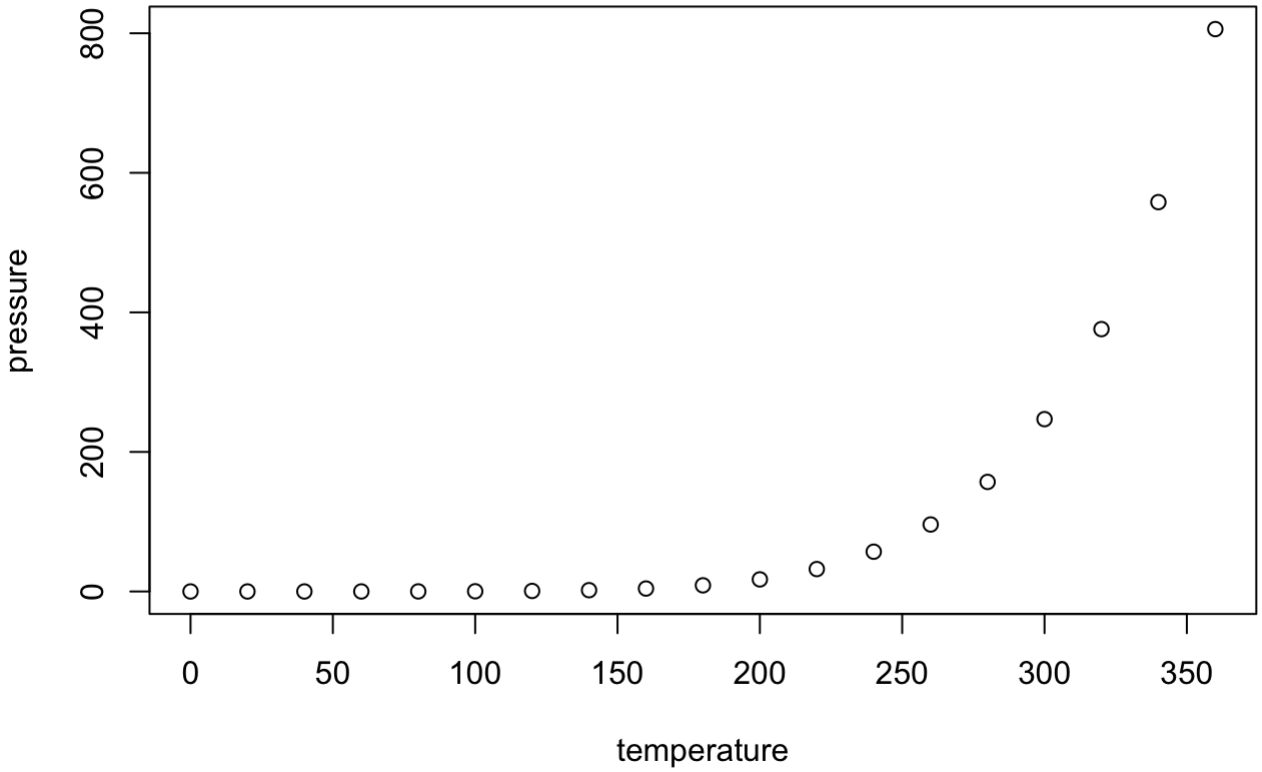
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.