Stat 280 (Forecasting Analytics) Exercise 2

Name: Veronica Bayani Student Number: 2009-00574

Solve the following problems with full codes and full outputs. 10 pts overall. Use the sale_app data (Sales of Appliances in Units Sold, Jan 2000 – Dec 2009) in PhilMonthlyData.csv.

Loading the data

```
library(tidyverse)
```

```
## Warning: package 'tidyverse' was built under R version 4.2.2
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.4.0      v purrr   0.3.4
## v tibble  3.1.8      v dplyr   1.0.10
## v tidyr   1.2.1      v stringr 1.4.1
## v readr   2.1.3      v forcats 0.5.2
```

```
## Warning: package 'ggplot2' was built under R version 4.2.2
```

```
## Warning: package 'readr' was built under R version 4.2.2
```

```
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
library(fpp2)
```

```
## Warning: package 'fpp2' was built under R version 4.2.2
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
## -- Attaching packages ------------------------------------------- fpp2 2.4 --
## v forecast  8.18     v expsmooth 2.3
## v fma       2.4
```

```
## Warning: package 'forecast' was built under R version 4.2.2
```

```
##
```

```
library(tinytex)
```

```
## Warning: package 'tinytex' was built under R version 4.2.2
```

```
philmon <- read.csv("PhilMonthlyData.csv", stringsAsFactors = FALSE, na.strings = c("NA"))
```

Split the data into training and test data set: Training dataset = Jan 2000 – Dec 2007; and Test dataset = Jan 2008 – Dec 2009.

Getting the Sales of Appliances in Units Sold from 2000 to 2009

```
saleapp <- ts(na.omit(philmon$sale_app),start=c(2000,1),frequency=12)
saleapp
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2000 474500.0 491300.0 561800.0 528900.0 663200.0 665700.0 524800.0 555000.0
## 2001 468800.0 435100.0 527800.0 491700.0 558200.0 559400.0 487100.0 465800.0
## 2002 424400.0 413300.0 482700.0 613800.0 690600.0 637600.0 564300.0 536900.0
## 2003 518495.7 423300.0 560250.0 601000.0 638400.0 596400.0 580100.0 491400.0
## 2004 509400.0 508500.0 668500.0 664300.0 660100.0 627100.0 565300.0 543800.0
## 2005 452500.0 441300.0 484600.0 537300.0 547400.0 519400.0 483200.0 446600.0
## 2006 369300.0 335300.0 405200.0 422300.0 462300.0 428100.0 386700.0 372400.0
## 2007 257400.0 265300.0 310000.0 283100.0 342200.0 326350.0 258700.0 254200.0
## 2008 299503.0 287403.0 346505.0 306229.0 322493.0 322071.0 290698.0 278103.0
## 2009 238044.7 217870.3 303801.5 321495.3 373995.3 348701.5 284970.3 261957.8
##           Sep      Oct      Nov      Dec
## 2000 493000.0 513000.0 581300.0 676900.0
## 2001 438200.0 536700.0 538900.0 568100.0
## 2002 527400.0 592100.0 654800.0 714600.0
## 2003 515200.0 642800.0 618300.0 730500.0
## 2004 557200.0 625300.0 708600.0 710100.0
## 2005 477300.0 471000.0 529400.0 592744.1
## 2006 351600.0 338800.0 426300.0 438500.0
## 2007 265950.0 304000.0 348000.0 373800.0
## 2008 314734.0 344043.5 391781.0 441736.5
## 2009 256926.5 306657.8 354395.3 404350.8
```

```
#train dataset
saleapp_train <- window(saleapp,start=2000,end=c(2007,12))
saleapp_train
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2000 474500.0 491300.0 561800.0 528900.0 663200.0 665700.0 524800.0 555000.0
## 2001 468800.0 435100.0 527800.0 491700.0 558200.0 559400.0 487100.0 465800.0
## 2002 424400.0 413300.0 482700.0 613800.0 690600.0 637600.0 564300.0 536900.0
## 2003 518495.7 423300.0 560250.0 601000.0 638400.0 596400.0 580100.0 491400.0
## 2004 509400.0 508500.0 668500.0 664300.0 660100.0 627100.0 565300.0 543800.0
## 2005 452500.0 441300.0 484600.0 537300.0 547400.0 519400.0 483200.0 446600.0
## 2006 369300.0 335300.0 405200.0 422300.0 462300.0 428100.0 386700.0 372400.0
## 2007 257400.0 265300.0 310000.0 283100.0 342200.0 326350.0 258700.0 254200.0
##           Sep      Oct      Nov      Dec
## 2000 493000.0 513000.0 581300.0 676900.0
## 2001 438200.0 536700.0 538900.0 568100.0
## 2002 527400.0 592100.0 654800.0 714600.0
## 2003 515200.0 642800.0 618300.0 730500.0
## 2004 557200.0 625300.0 708600.0 710100.0
```

```
## 2005 477300.0 471000.0 529400.0 592744.1
## 2006 351600.0 338800.0 426300.0 438500.0
## 2007 265950.0 304000.0 348000.0 373800.0
```

```
#test dataset
saleapp_test <- window(saleapp,start=2008,end=c(2009,12))
saleapp_test
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul      Aug
## 2008 299503.0 287403.0 346505.0 306229.0 322493.0 322071.0 290698.0 278103.0
## 2009 238044.7 217870.3 303801.5 321495.3 373995.3 348701.5 284970.3 261957.8
##           Sep      Oct      Nov      Dec
## 2008 314734.0 344043.5 391781.0 441736.5
## 2009 256926.5 306657.8 354395.3 404350.8
```

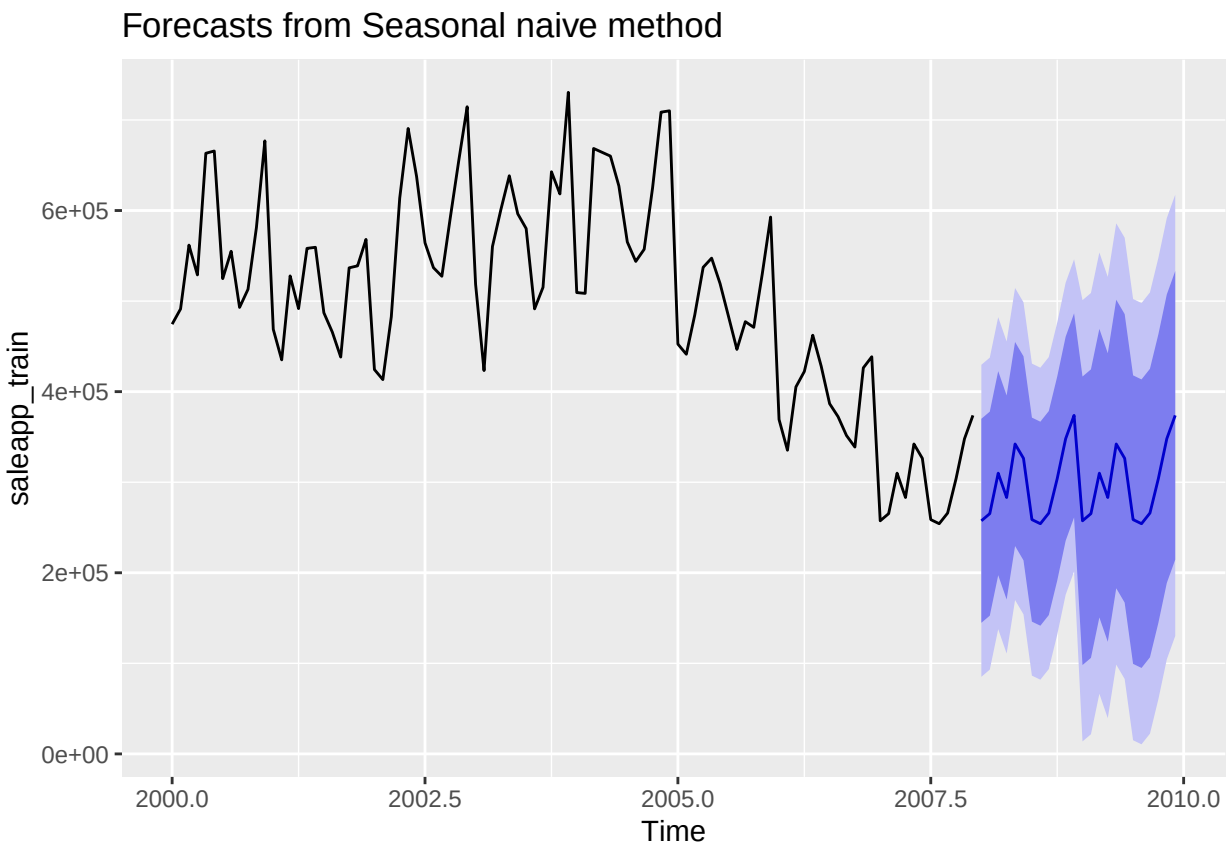
Decide which of the following models are the best-fitting model based on the test error measures RMSE, MAPE, and MAE:

a) [1 pt] Seasonal Naïve method

```
saleapp_seasonal_naive <- snaive(saleapp_train, h = 24)
saleapp_seasonal_naive
```

```
##          Point Forecast     Lo 80    Hi 80     Lo 95    Hi 95
## Jan 2008         257400 144772.83 370027.2  85151.61 429648.4
## Feb 2008         265300 152672.83 377927.2  93051.61 437548.4
## Mar 2008         310000 197372.83 422627.2 137751.61 482248.4
## Apr 2008         283100 170472.83 395727.2 110851.61 455348.4
## May 2008         342200 229572.83 454827.2 169951.61 514448.4
## Jun 2008         326350 213722.83 438977.2 154101.61 498598.4
## Jul 2008         258700 146072.83 371327.2  86451.61 430948.4
## Aug 2008         254200 141572.83 366827.2  81951.61 426448.4
## Sep 2008         265950 153322.83 378577.2  93701.61 438198.4
## Oct 2008         304000 191372.83 416627.2 131751.61 476248.4
## Nov 2008         348000 235372.83 460627.2 175751.61 520248.4
## Dec 2008         373800 261172.83 486427.2 201551.61 546048.4
## Jan 2009         257400  98121.13 416678.9  13803.99 500996.0
## Feb 2009         265300 106021.13 424578.9  21703.99 508896.0
## Mar 2009         310000 150721.13 469278.9  66403.99 553596.0
## Apr 2009         283100 123821.13 442378.9  39503.99 526696.0
## May 2009         342200 182921.13 501478.9  98603.99 585796.0
## Jun 2009         326350 167071.13 485628.9  82753.99 569946.0
## Jul 2009         258700  99421.13 417978.9  15103.99 502296.0
## Aug 2009         254200  94921.13 413478.9  10603.99 497796.0
## Sep 2009         265950 106671.13 425228.9  22353.99 509546.0
## Oct 2009         304000 144721.13 463278.9  60403.99 547596.0
## Nov 2009         348000 188721.13 507278.9 104403.99 591596.0
## Dec 2009         373800 214521.13 533078.9 130203.99 617396.0
```

```
autoplot(saleapp_seasonal_naive)
```



b) [1.5 pts] Linear Trend and Seasonal Dummies Regression Model

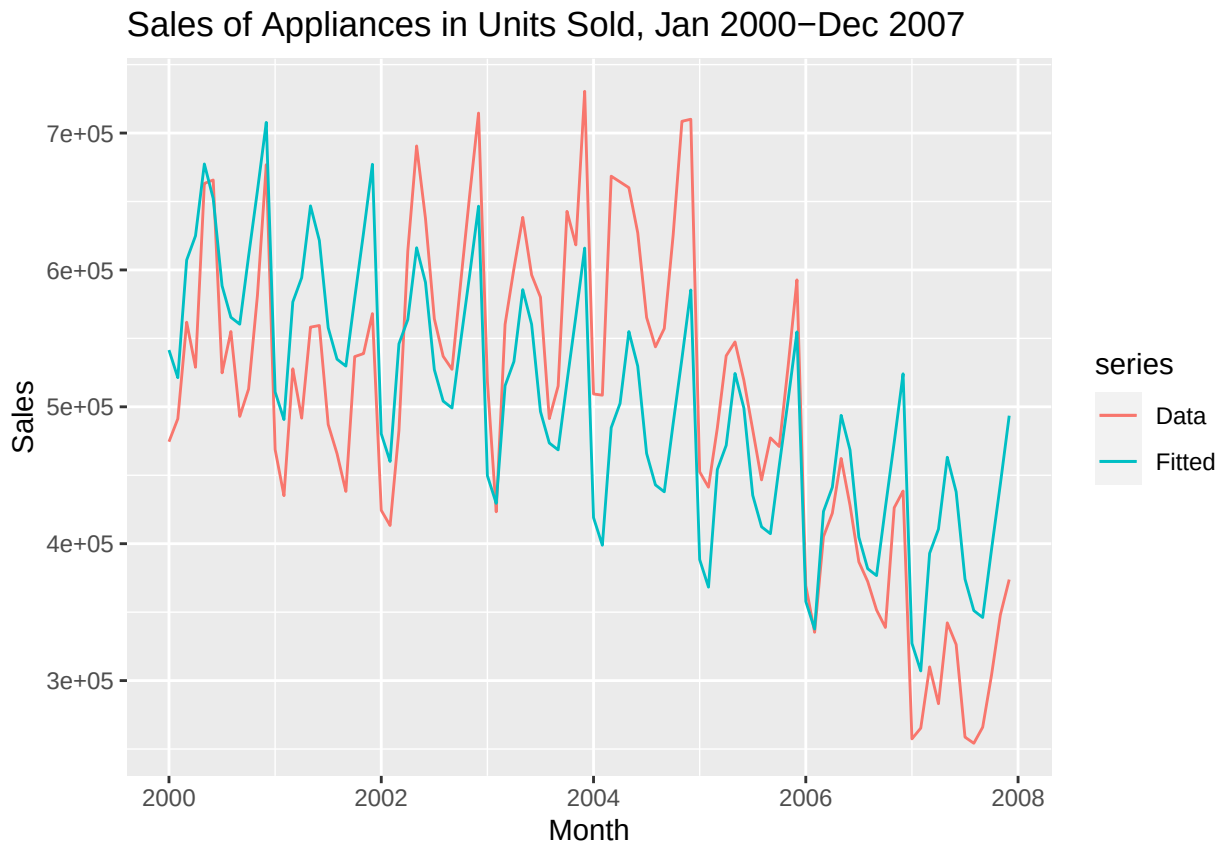
```
saleapp_b <- tslm(saleapp_train ~ trend + season)
summary(saleapp_b)
```

```
##
## Call:
## tslm(formula = saleapp_train ~ trend + season)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -127565  -67080   -7759   54142  183699
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 544035.8    32017.6  16.992  < 2e-16 ***
## trend        -2550.8      306.4  -8.326 1.46e-12 ***
## season2     -17623.6    41270.1  -0.427 0.670463
## season3      70858.5    41273.5   1.717 0.089745 .
## season4      91103.1    41279.2   2.207 0.030075 *
## season5     146153.9    41287.2   3.540 0.000659 ***
## season6     123411.0    41297.4   2.988 0.003688 **
## season7      62230.6    41309.9   1.506 0.135753
## season8      41769.0    41324.7   1.011 0.315073
## season9      39288.5    41341.7   0.950 0.344700
## season10     91570.6    41361.0   2.214 0.029576 *
```

```
## season11    141859.0    41382.5   3.428 0.000949 ***
## season12    194365.3    41406.4   4.694 1.05e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 82540 on 83 degrees of freedom
## Multiple R-squared:  0.5732, Adjusted R-squared:  0.5115
## F-statistic: 9.288 on 12 and 83 DF,  p-value: 4.442e-11
```

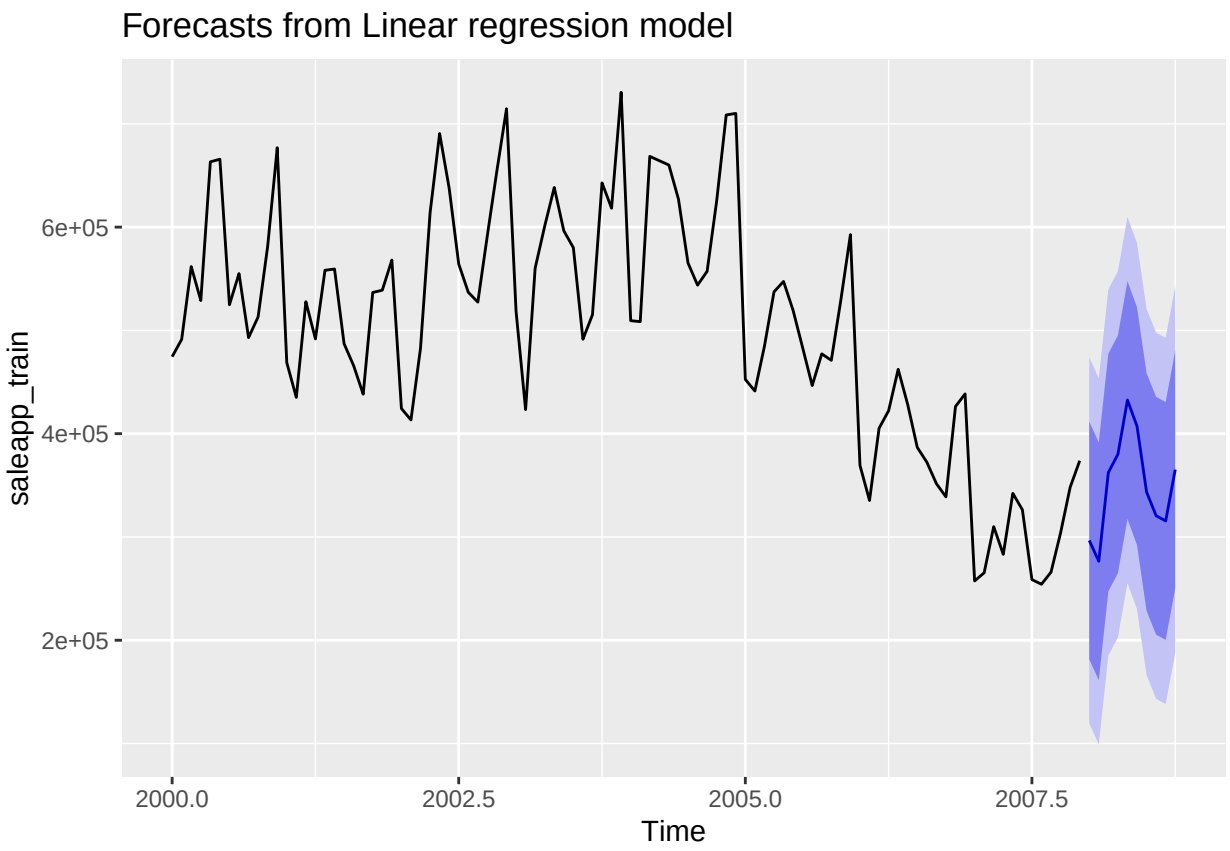
Time plot of actual sales of appliances in units sold and the fitted values from Linear Trend and Seasonal Dummies Regression Model.

```
autoplot(saleapp_train, series="Data") +
  autolayer(fitted(saleapp_b), series="Fitted") +
  xlab("Month") + ylab("Sales") +
  ggtitle("Sales of Appliances in Units Sold, Jan 2000-Dec 2007")
```



Forecasting with Linear Trend and Seasonal Dummies Regression Model and plotting the results

```
fcast_b <- forecast(saleapp_b)

autoplot(fcast_b)
```



c) [1.5 pts] Exponential Trend and Seasonal Dummies Regression Model

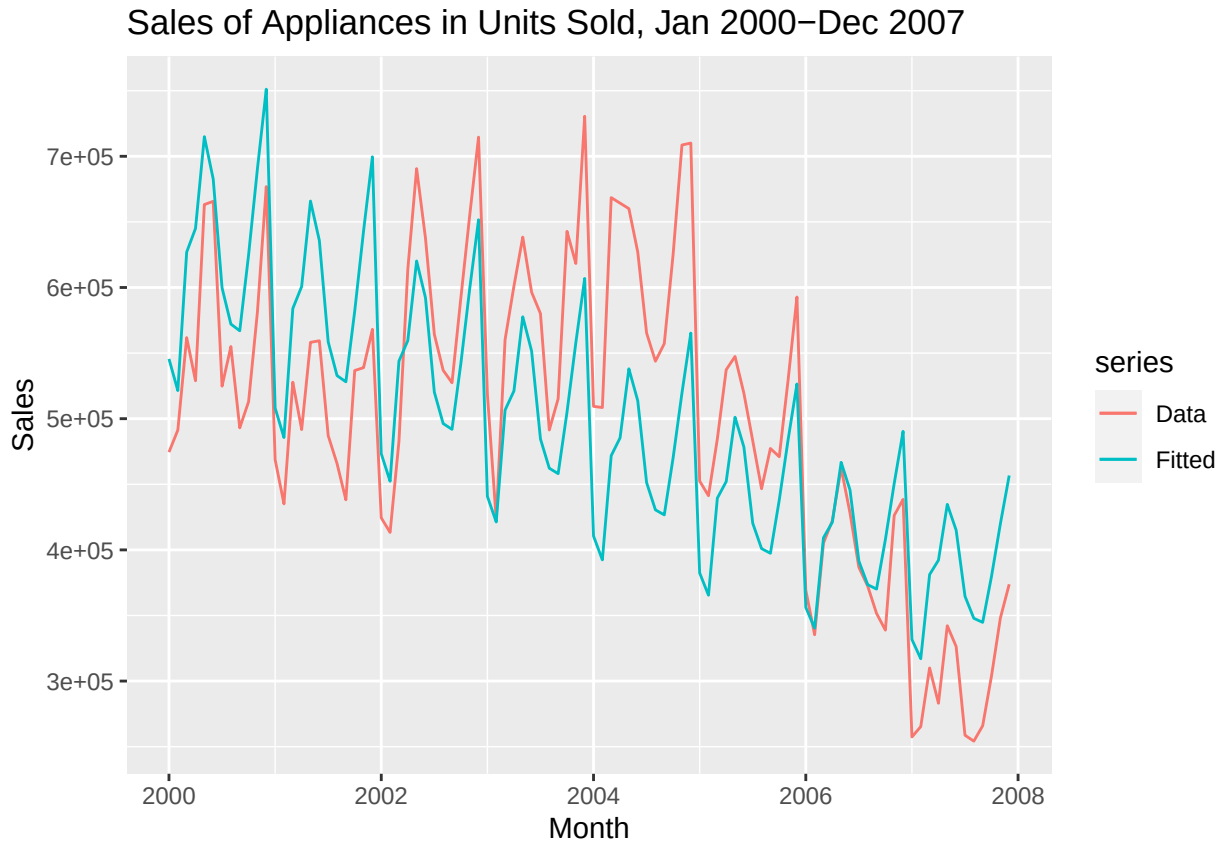
```
saleapp_c <- tslm(saleapp_train ~ trend + season, lambda=0)
summary(saleapp_c)
```

```
##
## Call:
## tslm(formula = saleapp_train ~ trend + season, lambda = 0)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.34320 -0.13508 -0.00617  0.11019  0.34846
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) 13.2155159  0.0693333 190.609  < 2e-16 ***
## trend       -0.0059238  0.0006635  -8.929 9.07e-14 ***
## season2     -0.0393694  0.0893694  -0.441  0.66070
## season3      0.1509284  0.0893768   1.689  0.09504 .
## season4      0.1852002  0.0893891   2.072  0.04138 *
## season5      0.2940941  0.0894063   3.289  0.00147 **
## season6      0.2537596  0.0894285   2.838  0.00571 **
## season7      0.1301765  0.0894556   1.455  0.14938
## season8      0.0889441  0.0894875   0.994  0.32315
## season9      0.0859231  0.0895244   0.960  0.33996
```

```
## season10     0.1891194  0.0895662   2.112  0.03774 *
## season11     0.2947595  0.0896129   3.289  0.00148 **
## season12     0.3849383  0.0896644   4.293 4.75e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1787 on 83 degrees of freedom
## Multiple R-squared:  0.5808, Adjusted R-squared:  0.5202
## F-statistic: 9.584 on 12 and 83 DF,  p-value: 2.232e-11
```

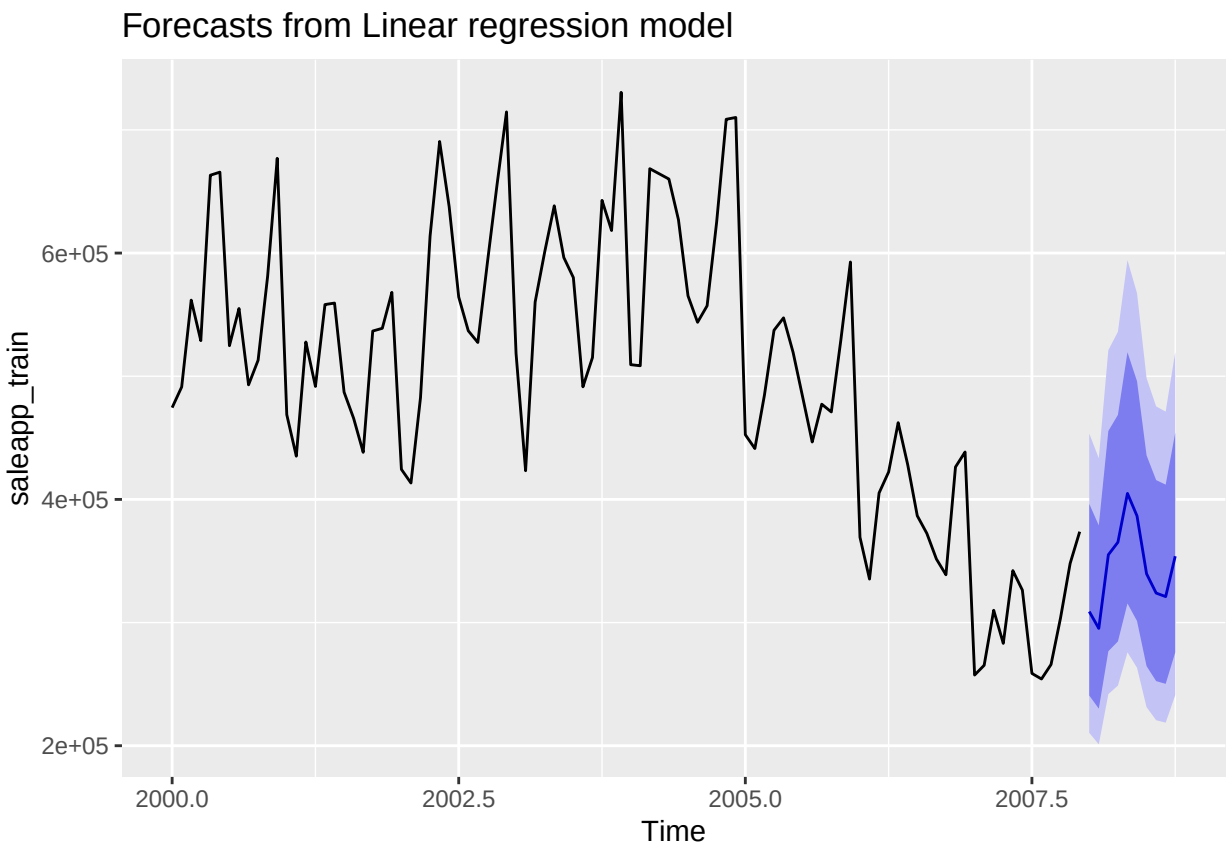
Time plot of actual sales of appliances in units sold and the fitted values from the Exponential Trend and Seasonal Dummies Regression Model

```
autoplot(saleapp_train, series="Data") +
  autolayer(fitted(saleapp_c), series="Fitted") +
  xlab("Month") + ylab("Sales") +
  ggtitle("Sales of Appliances in Units Sold, Jan 2000-Dec 2007")
```



Plotting the forecast from the Exponential Trend and Seasonal Dummies Regression Model

```
fcast_c <- forecast(saleapp_c)

autoplot(fcast_c)
```



d) [2 pts] Linear STL Model, t.window=13, s.window=13

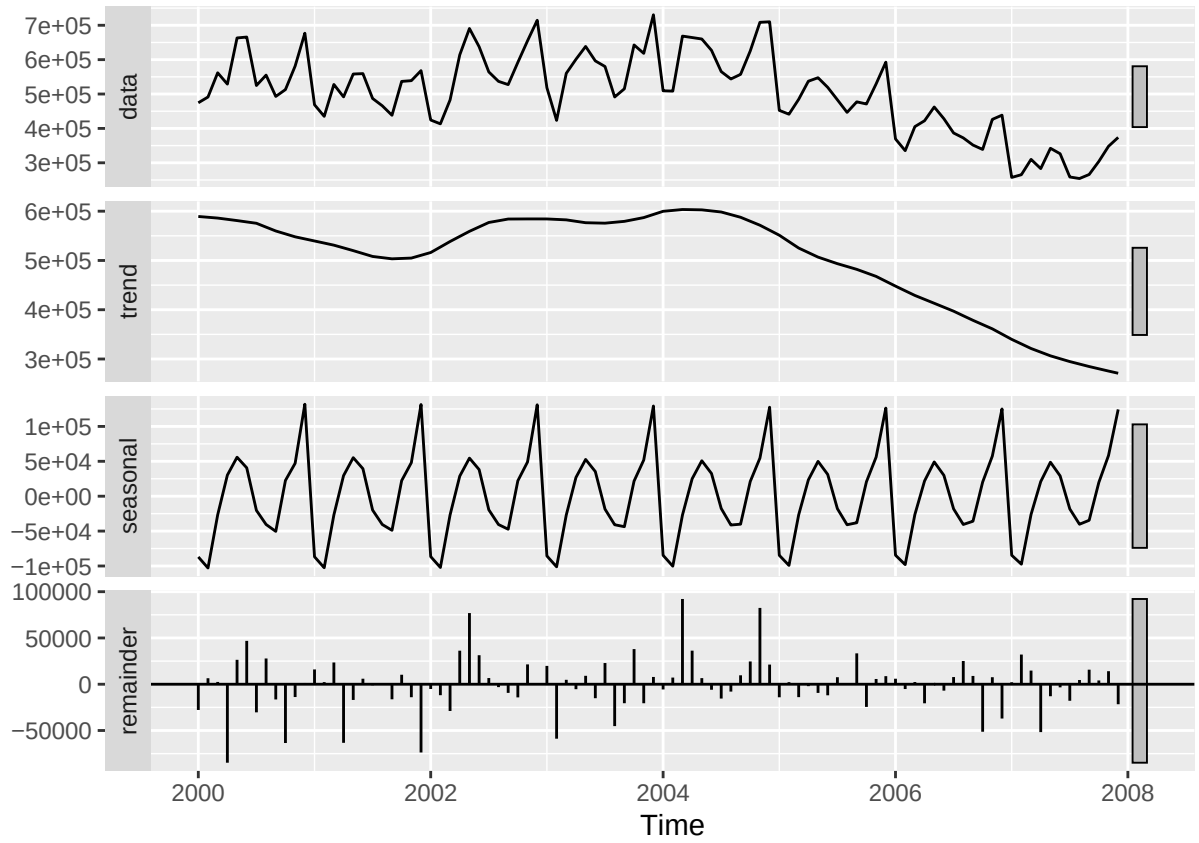
```
saleapp_STL <- saleapp_train %>%
  stl(t.window=13, s.window=13, robust=TRUE)
saleapp_STL
```

```
##  Call:
##  stl(x = ., s.window = 13, t.window = 13, robust = TRUE)
##
## Components
##              seasonal    trend   remainder
## Jan 2000  -86987.57 589290.5 -27802.9296
## Feb 2000 -102848.55 587651.6   6496.9501
## Mar 2000  -26680.38 586012.7   2467.6865
## Apr 2000   30221.86 583479.8 -84801.6857
## May 2000   55852.18 580947.0  26400.8570
## Jun 2000   40640.69 578196.9  46862.4252
## Jul 2000  -20279.84 575446.8 -30366.9824
## Aug 2000  -40489.72 567650.1  27839.6597
## Sep 2000  -50394.58 559853.3 -16458.7280
## Oct 2000   22589.62 553925.7 -63515.3498
## Nov 2000   47133.62 547998.2 -13831.7694
## Dec 2000  131756.74 543778.8   1364.4639
## Jan 2001  -86764.86 539559.4  16005.4266
## Feb 2001 -102493.43 535308.6   2284.8605
```

```
## Mar 2001  -26772.33 531057.7  23514.6216
## Apr 2001   29485.15 525513.1 -63298.2763
## May 2001   55202.78 519968.5 -16971.3142
## Jun 2001   39376.18 514047.7   5976.1275
## Jul 2001  -19890.17 508126.9  -1136.6862
## Aug 2001  -40545.50 505734.9    610.5756
## Sep 2001  -48863.58 503343.0 -16279.4082
## Oct 2001   22337.94 504106.5  10255.5959
## Nov 2001   48075.75 504869.9 -14045.6934
## Dec 2001  131347.59 510477.0 -73724.5900
## Jan 2002  -86546.44 516084.1  -5137.6181
## Feb 2002 -102144.34 527273.4 -11829.0090
## Mar 2002  -26872.05 538462.6 -28890.6019
## Apr 2002   28742.09 548808.4  36249.5512
## May 2002   54548.42 559154.1  76897.5095
## Jun 2002   38111.95 568133.8  31354.2056
## Jul 2002  -19494.98 577113.6   6681.3670
## Aug 2002  -40588.63 580585.8  -3097.1491
## Sep 2002  -47312.81 584057.9  -9345.1266
## Oct 2002   22111.84 584179.0 -14190.8687
## Nov 2002   49049.28 584300.1  21450.6003
## Dec 2002  130974.13 584282.4   -656.5017
## Jan 2003  -85608.94 584264.6  19839.9837
## Feb 2003 -101184.25 583319.4 -58835.1967
## Mar 2003  -26922.65 582374.3   4798.3874
## Apr 2003   26883.55 579485.9  -5369.4273
## May 2003   52702.04 576597.5   9100.4614
## Jun 2003   35345.46 576143.8 -15089.2322
## Jul 2003  -18510.12 575690.0  22920.0714
## Aug 2003  -40890.02 577533.8 -45243.7600
## Sep 2003  -43674.45 579377.5 -20503.0650
## Oct 2003   21647.39 583175.6  37977.0281
## Nov 2003   51872.50 586973.7 -20546.1516
## Dec 2003  129267.57 593362.8   7869.6024
## Jan 2004  -84719.32 599752.0  -5632.6834
## Feb 2004 -100272.72 601569.6   7203.0828
## Mar 2004  -27022.51 603387.3  92135.2321
## Apr 2004   24974.27 603047.7  36278.0580
## May 2004   50803.45 602708.1   6588.4855
## Jun 2004   32524.46 600532.8  -5957.2460
## Jul 2004  -17582.05 598357.5 -15475.4481
## Aug 2004  -41250.98 593007.2  -7956.2416
## Sep 2004  -40098.42 587656.9   9641.4792
## Oct 2004   21116.49 579580.8  24602.7214
## Nov 2004   54625.16 571504.6  82470.2067
## Dec 2004  127485.40 561376.1  21238.4615
## Jan 2005  -84664.13 551247.7 -14083.5203
## Feb 2005  -99158.68 538224.6   2234.0297
## Mar 2005  -26662.70 525201.6 -13938.9472
## Apr 2005   23367.91 516010.9  -2078.7825
## May 2005   49946.47 506820.1  -9366.5739
## Jun 2005   31180.84 500166.1 -11946.9771
## Jul 2005  -17884.47 493512.2   7572.3018
## Aug 2005  -40895.86 487726.6   -230.7271
```

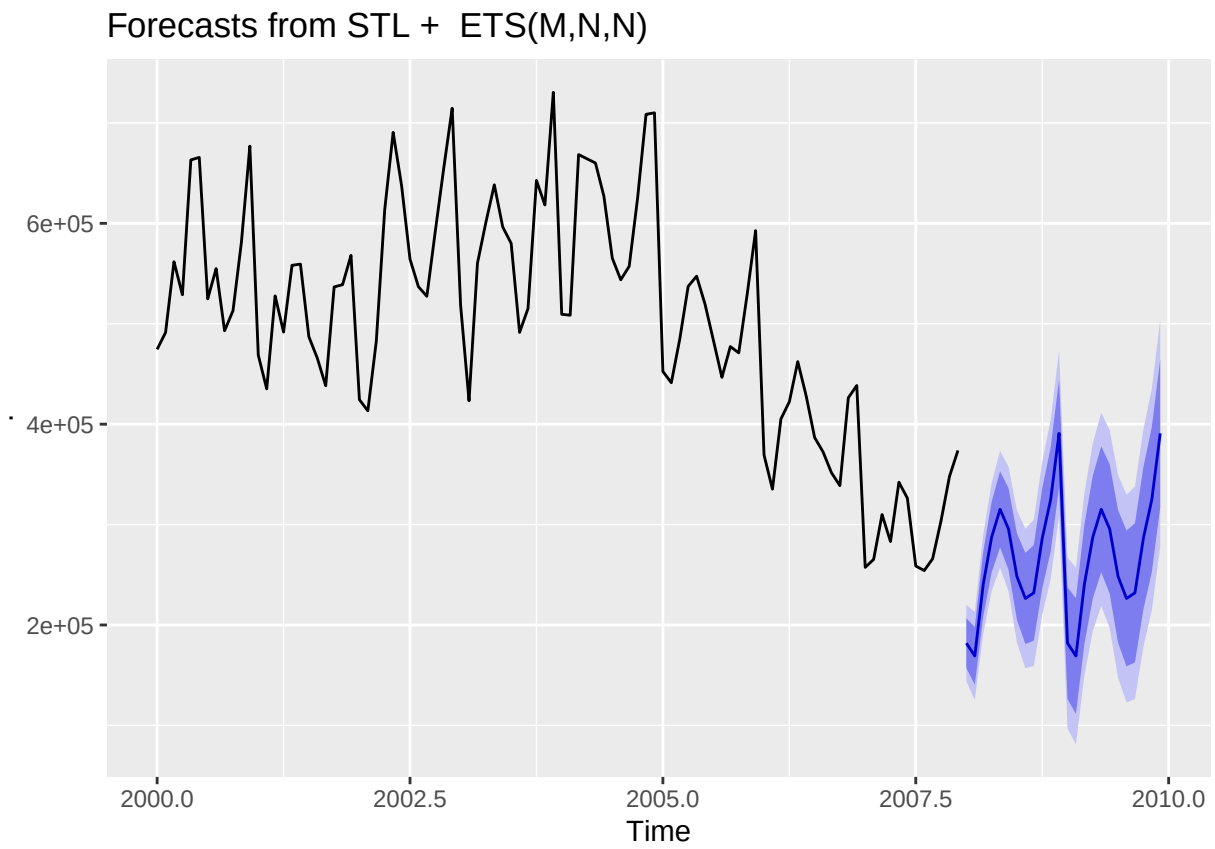
```
## Sep 2005  -38007.87 481941.0  33366.8550
## Oct 2005   20680.75 474801.8 -24482.5599
## Nov 2005   56060.78 467662.6   5676.6105
## Dec 2005  126240.22 457778.5   8725.2937
## Jan 2006  -84559.54 447894.5   5965.0804
## Feb 2006  -97990.90 438472.4  -5181.5152
## Mar 2006  -26244.82 429050.4   2394.4526
## Apr 2006   21818.22 421095.6 -20613.8592
## May 2006   49144.77 413140.9     14.3153
## Jun 2006   29885.76 405117.4  -6903.1875
## Jul 2006  -18145.08 397093.9   7751.1444
## Aug 2006  -40507.78 387810.3  25097.5010
## Sep 2006  -35893.18 378526.6   8966.5653
## Oct 2006   20262.37 369876.7 -51339.0567
## Nov 2006   57506.99 361226.8   7566.2488
## Dec 2006  125001.73 350563.2 -37064.8864
## Jan 2007  -84590.74 339899.5   2091.2015
## Feb 2007  -97383.33 330624.4  32058.9642
## Mar 2007  -26106.44 321349.2  14757.2534
## Apr 2007   20921.61 313902.6 -51724.2184
## May 2007   48718.96 306456.0 -12974.9810
## Jun 2007   29095.25 300632.4  -3377.6868
## Jul 2007  -18233.37 294808.8 -17875.4685
## Aug 2007  -40145.83 289775.3   4570.5634
## Sep 2007  -34536.00 284741.7  15744.3090
## Oct 2007   19747.93 280156.2   4095.8824
## Nov 2007   58337.12 275570.7  14092.1851
## Dec 2007  124322.07 271095.8 -21617.9011
```

```
autoplot(saleapp_STL)
```



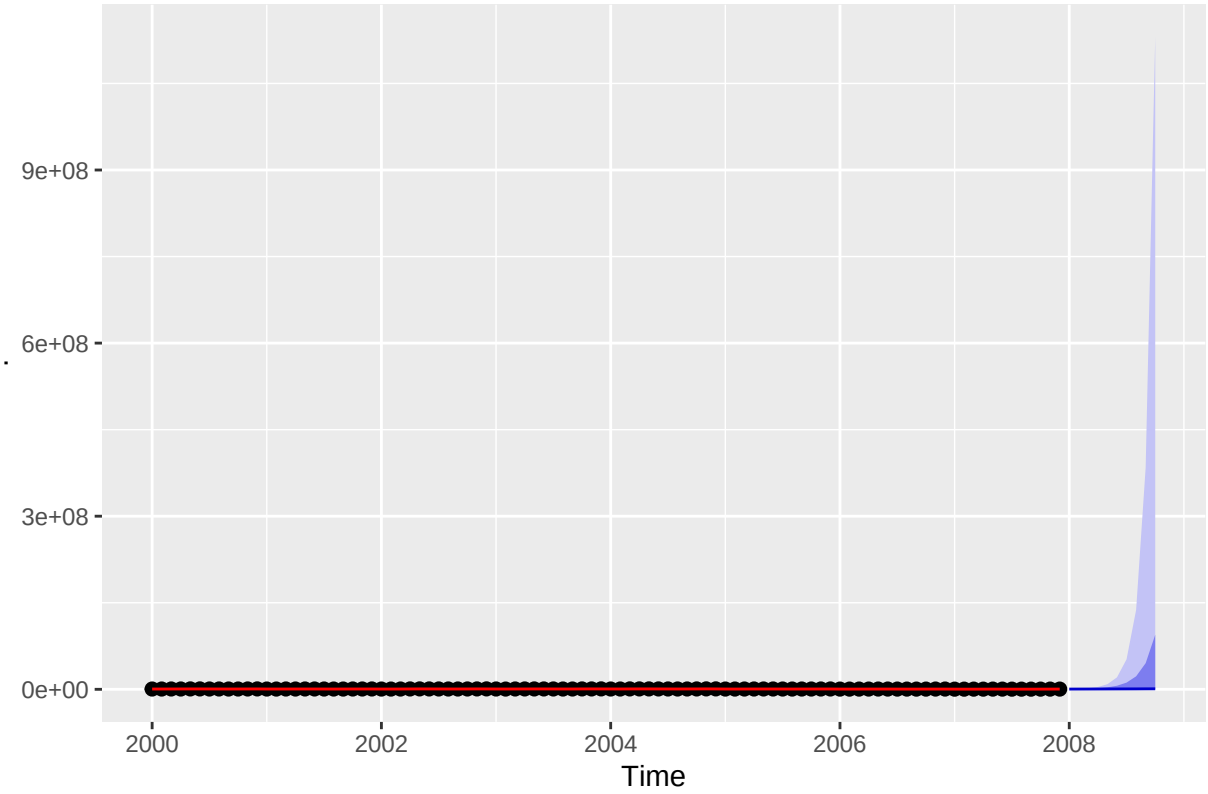
Plotting the STL FC

```
fcast_STL <- forecast(saleapp_STL)

autoplot(fcast_STL)
```



e) [2 pts] Exponential Natural Cubic Smoothing Splines Model (splinef)

```
saleapp_e <- saleapp_train %>% splinef(lambda=0)

autoplot(saleapp_e)
```



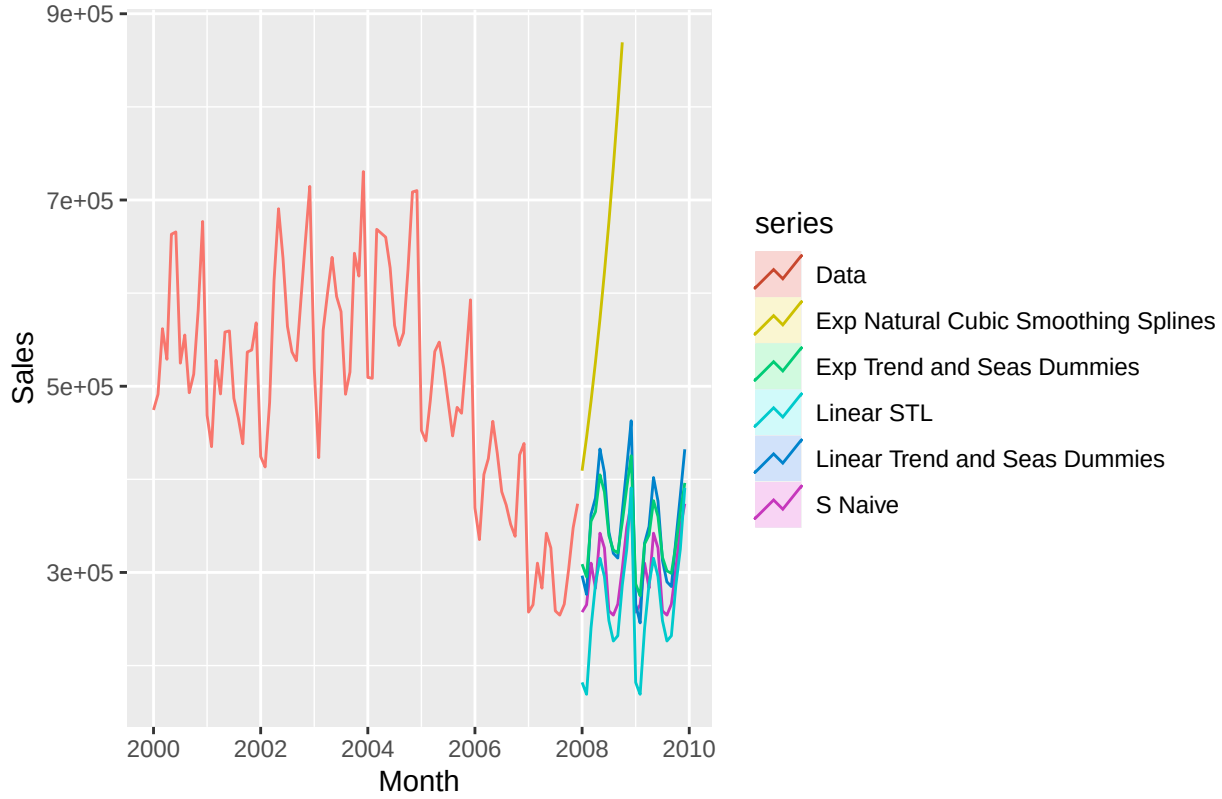
Plotting all of the forecasts in one chart:

```
autoplot(saleapp_train, series="Data") +
  autolayer(snaive(saleapp_train, h = 24), series="S Naive", PI=FALSE) +
  autolayer(forecast(saleapp_b, newdata = saleapp_test), series="Linear Trend and Seas Dummies", PI=FALS
  autolayer(forecast(saleapp_c, newdata = saleapp_test), series="Exp Trend and Seas Dummies", PI=FALSE)
  autolayer(forecast(saleapp_STL), series="Linear STL", PI=FALSE) +
  autolayer(forecast(saleapp_e, newdata = saleapp_test), series="Exp Natural Cubic Smoothing Splines", I
  xlab("Month") + ylab("Sales") +
  ggtitle("Figure 1. Forecast for Sales of Appliances in Units Sold, Jan 2008-Dec 2009")
```

```
## Warning in forecast.lm(saleapp_b, newdata = saleapp_test): newdata column names
## not specified, defaulting to first variable required.
```

```
## Warning in forecast.lm(saleapp_c, newdata = saleapp_test): newdata column names
## not specified, defaulting to first variable required.
```

Figure 1. Forecast for Sales of Appliances in Units Sold, Jan 2008−Dec 2



Computing for the RMSE, MAPE and MAE:

```
accuracy(saleapp_seasonal_naive, saleapp_test)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -37385.71 87883.45 77137.99 -10.895158 17.43934 1.0000000
## Test set      18352.78 31692.91 27185.53   4.979186  8.55356 0.3524273
##                   ACF1 Theil's U
## Training set 0.7891203        NA
## Test set     0.3805821 0.5849341
```

```
accuracy(fcast_b, saleapp_test)
```

```
##                         ME     RMSE      MAE        MPE     MAPE      MASE
## Training set -4.836576e-12 76746.24 66280.95  -2.792209 14.04815 0.8592516
## Test set     -3.882298e+04 54991.07 41597.52 -12.429475 13.38671 0.5392612
##                   ACF1 Theil's U
## Training set 0.8677971        NA
## Test set     0.6229627  1.757117
```

```
accuracy(fcast_c, saleapp_test)
```

```
##                     ME     RMSE      MAE        MPE     MAPE      MASE
## Training set   5813.244 81672.61 70555.38  -1.386488 14.45613 0.9146645
## Test set     -34267.341 43913.62 34267.34 -11.138067 11.13807 0.4442343
```

```
##                    ACF1 Theil's U
## Training set 0.8839281        NA
## Test set     0.5719927  1.403659
```

```
accuracy(fcast_STL, saleapp_test)
```

```
##                      ME     RMSE      MAE       MPE      MAPE      MASE
## Training set -5413.842 36153.62 26885.16 -1.61040  5.516279 0.3485333
## Test set     50893.451 59002.83 50893.45 16.46846 16.468459 0.6597715
##                     ACF1 Theil's U
## Training set -0.04599406        NA
## Test set      0.55752068  1.089705
```

```
accuracy(saleapp_e, saleapp_test)
```

```
##                       ME     RMSE       MAE        MPE     MAPE     MASE
## Training set  -12572.19 109631.4  82764.74  -2.932391 17.38422 1.072944
## Test set     -302684.28 335397.9 302684.28 -97.715675 97.71567 3.923933
##                     ACF1 Theil's U
## Training set -0.1486967        NA
## Test set      0.7256058  11.16306
```

Putting the results in a table for the summary:

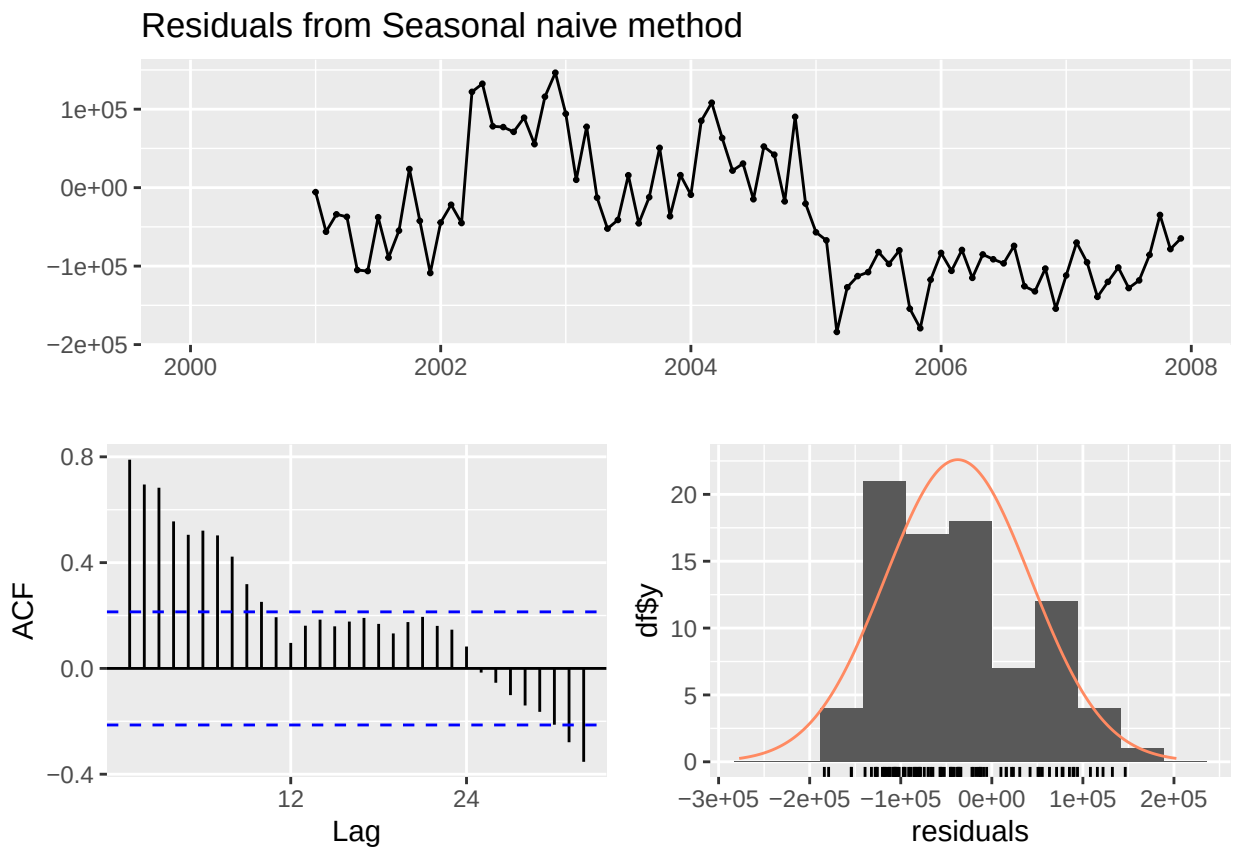
	RMSE	MAPE	MAE
Seasonal Naïve	31,693	9	27,186
Linear Trend and Seasonal Dummies Regression	54,991	13	41,598
Exponential Trend and Seasonal Dummies Regression	43,914	11	34,267
Linear STL Model	59,003	16	50,893
Exponential Natural Cubic Smoothing Splines Model	335,398	98	302,684

Figure 1: results

The seasonal naive model has the lowest RMSE, MAPE and MAE.

Checking the residuals for the seasonal naive model:

```
checkresiduals(saleapp_seasonal_naive)
```




```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 297.23, df = 19, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 19
```

The p value (2.2e-16) is less than 0.05 which means that there is evidence that the data is autocorrelated.

Checking the normality of the residuals,

```
nortest::ad.test(residuals(saleapp_seasonal_naive))
```

```
##
##  Anderson-Darling normality test
##
## data:  residuals(saleapp_seasonal_naive)
## A = 1.3651, p-value = 0.001449
```

The p value (2.2e-16) is less than 0.05 which means that there is evidence that the residuals are not normally distributed.

Write a short paragraph explaining your choice of the best-fitting model. [2 pts]

The best fitting model is the Seasonal Naive Model since it has the lowest RMSE, MAPE and MAE as compared to the other models. However, despite having the lowest errors, the seasonal naive model does not

comply with the properties that residuals should have for full extraction of the patterns from the time series. Specifically, there is sufficient evidence to conclude that autocorrelation is present based on the Ljung-Box test results and that the residuals are not normally distributed.

To further improve the performance of the model, the autocorrelation issues and non-normality of the residuals must be addressed. Some of the possible improvements that can be done to address these concerns are exploring the addition of other predictor variables in the model or by doing variable transformation.