2024-01

Naval Expeditionary Readiness Model

MacKinnon, Douglas J.; Davenport, Branden W.

Monterey, California. Naval Postgraduate School

https://hdl.handle.net/10945/72644

NPS-OR-23-013

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

NAVAL EXPEDITIONARY READINESS MODEL

by

Dr. Douglas J. MacKinnon and LCDR Branden W. Davenport, SC, USN

January 2024

Approved for public release. Distribution is unlimited.

Prepared for: OPNAV N81 - Integration of Capabilities and Resources.
This research is supported by funding from the Naval Postgraduate School, Naval Research Program (PE 0605853N/2098). NRP Project ID: NPS-23-N251-A

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ORGANIZATION.

1. REPORT DATE	**2. REPORT TYPE**	**3. DATES COVERED**	
		START DATE	**END DATE**
03 Dec 2023	Technical Report	2 Jan 2023	13 Jan 2024

3. TITLE AND SUBTITLE
Naval Expeditionary Readiness Model

5a. CONTRACT NUMBER	**5b. GRANT NUMBER**	**5c. PROGRAM ELEMENT NUMBER**
		0605853N/2098

5d. PROJECT NUMBER	**5e. TASK NUMBER**	**5f. WORK UNIT NUMBER**
NPS-23-N251-A		

6. AUTHOR(S)
Dr. Douglas J. MacKinnon, LCDR Branden W. Davenport

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	**8. PERFORMING ORGANIZATION REPORT NUMBER**
Naval Postgraduate School 1 University Circle Monterey, CA 93943-5000	NPS-OR-23-013

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	**10. SPONSOR/MONITOR'S ACRONYM(S)**	**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**
Naval Postgraduate School, Naval Research Program, Monterey, CA Integration of Capabilities and Resources (N81). Service Integration and Policy, Washington, DC	NRP; N81	NPS-23-N251-A

12. DISTRIBUTION/AVAILABILITY STATEMENT
Distribution Statement A: Approved for public release: distribution is unlimited.

13. SUPPLEMENTARY NOTES
The views expressed in this document are those of the authors and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

14. ABSTRACT

Naval expeditionary forces lack the ability to adequately estimate the level of spending required to achieve a minimum level of readiness. Currently the Navy Expeditionary Combat Enterprise (NECE) Capability Costing Model (NCCM) forecasts requirements using Excel Solver and data from the Optimized Fleet Response Plan (OFRP) and Certified Obligation Reports. To explore methods of improving requirement forecasts, this research limits its focus to one program, Explosive Ordinance Disposal (EOD), one component, active duty, and one training and testing data split. It then attempts multiple forecasting methods over multiple levels of cost aggregation. These forecasting methods include Exponential Smoothing, Autoregressive Integrated Moving Averages (ARIMA), and dynamic regression models. The analysis then evaluates models made with those methods using the accuracy measures of absolute error (MAE), mean absolute percentage error (MAPE), and mean absolute scaled error (MASE). It also attempts hierarchical models to forecast costs and evaluates those models in the same way. Finally, it calculates forecasts for two years in the future and compares those forecasts to actual costs. This final calculation mimics the process required in the Program Objective Memorandum (POM) process.

This technical report finds that the various models forecast at different levels of accuracy across different levels of cost aggregation. The most accurate model to forecast total EOD costs two years in the future is an ARIMA model. It possesses a 10 percent delta in its forecast. The best aggregated model is an exponential smoothing model for the Budget Submitting Office (BSO) 60 and the warfare pillars of personnel (P) and training (T). Its delta is three percent. However, some levels of aggregation are much worse, with the best model possessing a delta of 36 percent for BSO 70 for supply (S) and equipment (E) costs.

This technical report ends with several recommendations for future research.

15. SUBJECT TERMS

Naval Expeditionary Forces, Forecasting, Budget, Readiness

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**
a. REPORT	**b. ABSTRACT**	**C. THIS PAGE**		
U	U	U	UU	55

19a. NAME OF RESPONSIBLE PERSON	**19b. PHONE NUMBER** *(Include area code)*
Branden W. Davenport, LCDR, SC, USN	(574) 999-0453

THIS PAGE INTENTIONALLY LEFT BLANK

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

Ann E. Rondeau
President

Scott Gartner
Provost

The report entitled "Naval Expeditionary Readiness Model" was prepared for OPNAV N81 – Integration of Capabilities and Resources and funded by the Naval Postgraduate School, Naval Research Program (PE 0605853N/2098).

Further distribution of all or part of this report is authorized.

This report was prepared by:

MACKINNON.DOUGLAS.JOHN.1177107227 Digitally signed by MACKINNON.DOUGLAS.JOHN.1177107227 Date: 2024.02.01 12:26:52 -08'00'

Douglas J. MacKinnon
Research Associate Professor

DAVENPORT.BRANDEN.WESLEY.1459098565 Digitally signed by DAVENPORT.BRANDEN.WESLEY.1459098565 Date: 2024.02.04 16:17:15 -08'00'

Branden W. Davenport
LCDR, SC, USN

Reviewed by:

CARLYLE.WILLIAM.MATTHEW.1256369440 Digitally signed by CARLYLE.WILLIAM.MATTHEW.1256369440 DN: c=US, o=U.S. Government, ou=DoD, ou=PKI, ou=USN, cn=CARLYLE.WILLIAM.MATTHEW.1256369440 Date: 2024.02.16 10:51:47 -08'00'

Matthew Carlyle, Chair
Operations Research

Released by:

SMITH.KEVIN.B.1230026366 Digitally signed by SMITH.KEVIN.B.1230026366 Date: 2024.02.16 17:07:02 -08'00'

Kevin B. Smith
Vice Provost for Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Naval expeditionary forces lack the ability to adequately estimate the level of spending required to achieve a minimum level of readiness. Currently the Navy Expeditionary Combat Enterprise Capability Costing Model forecasts requirements using Excel Solver and data from the Optimized Fleet Response Plan and Certified Obligation Reports. To explore methods of improving requirement forecasts, this research limits its focus to one program, explosive ordinance disposal (EOD), one component, active duty, and one training and testing data split. It then attempts multiple forecasting methods over multiple levels of cost aggregation. These forecasting methods include exponential smoothing, autoregressive integrated moving averages (ARIMA), and dynamic regression models. The analysis then evaluates models made with those methods using the accuracy measures of absolute error, mean absolute percentage error, and mean absolute scaled error. It also attempts hierarchical models to forecast costs and evaluates those models in the same way. Finally, it calculates forecasts for two years in the future and compares those forecasts to actual costs. This final calculation mimics the process required in the Program Objective Memorandum process.

This technical report finds that the various models forecast at different levels of accuracy across different levels of cost aggregation. The best model to forecast total EOD costs, two years in the future, is an ARIMA model. It possesses a 10 percent difference in its forecast. The best aggregated model is an exponential smoothing model for the Budget Submitting Office (BSO) 60 and the warfare pillars of personnel (P) and training (T). Its delta is three percent. However, some levels of aggregation are much worse, with the best model possessing a difference of 36 percent for BSO 70 for supply (S) and equipment (E) costs.

This technical report ends with several recommendations for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

A. PROBLEM STATEMENT

Naval expeditionary forces lack the ability to adequately estimate the level of spending required to achieve a minimum level of readiness. Under the status quo, the Navy Expeditionary Combat Enterprise (NECE) Capability Costing Model (NCCM) forecasts routine requirements using Excel Solver and data from the Optimized Fleet Response Plan (OFRP) and Certified Obligation Reports. This model receives historical data obtained from Command Financial Management System (DFMS), Standardized Accounting and Reporting System – Field Level (STARS-FL), past OFRP schedules, and notional OFRP schedules. Using least-square-optimization and various constraints, Solver estimates the cost of each phase of the OFRP and then applies those costs to the notional OFRP schedule of each program. The reasoning behind the constraints used in the model is unclear. The sponsor also believes a more accurate model to forecast costs exists. The purpose of this research is to explore forecasting methods that may be able to improve the determination of requirements in the Program Objective Memorandum (POM) process.

B. ANALYTICAL TOOLS AND PROCESS

The first step in the analytical process of the author is to retrieve, review, and wrangle data. The author of this technical report received raw cost and OFRP data in the forms of CSV files directly from the NCCM tool. This analysis then combines yearly cost and OFRP data, identifies relevant columns, and then formats the data to be the appropriate data type. This analysis focuses on programs, BSOs, program elements (PE), components, and warfare pillars. The data is then divided into training and testing data. The analysis uses training data to determine the optimal coefficients in the models and then testing data to assess the quality of the models. The author also filters training and testing data into multiple data frames that represent different levels of cost aggregation: All explosive ordinance disposal (EOD) costs; BSO 60 costs across E/S (equipment and supply) and P/T (personnel and training) pillars; and BSO 70 costs across E/S and P/T pillars. Note that BSO 60 is the comptroller for EOD units on the East Coast while BSO 70 is the comptroller on the West Coast. The author chose the E/S and P/T pillars as levels of aggregation because the distinction between E and S is sometimes unclear.

Using the programming language R and the Fable package, this analysis builds models that can be divided into three broad categories: exponential smoothing, autoregressive integrated moving averages (ARIMA), and dynamic regression. The author defines various parameters across these model types and Fable determines the coefficients for those models based on the training data and various optimization criteria. The author then determines the best models within each model category using testing measures of absolute error, mean absolute percentage error, and mean absolute scaled error. Finally, using a two-year forecast, the author compares forecasted costs and actual costs. This technical report also explores hierarchical methods, but it produces worse results than the best-in-category approach above.

C. CONCLUSIONS AND RECOMMENDATIONS

This technical report finds that the various models forecast at different levels of accuracy across different levels of cost aggregation. The most accurate aggregated model to forecast all EOD costs two years in the future is an ARIMA model. Its delta when compared to actual costs is 10 percent. The most accurate disaggregated model is an exponential smoothing model for BSO 60 and the warfare pillars P/T. Its delta when compared to actual costs is three percent. However, some levels of aggregation are much less accurate. For example, the most accurate model for BSO 70 for S/T costs possess a delta of 36 percent.

Standard forecasting methods, therefore, can predict requirements at reasonable levels of accuracy for certain levels of aggregation. Before these methods can be implemented, however, further research is required to explore different levels of aggregation and different training and testing splits. For example, instead of aggregating based on pillars, aggregation based on Special Interest Code or List Item may produce superior models relative to aggregation based on warfare pillars. In the meantime, the forecasting methods in this technical report can serve as secondary forecasting methods to supplement the NECE NCCM.

I. INTRODUCTION

A. BACKGROUND

The Naval expeditionary forces lack the ability to adequately estimate the level of spending required to achieve a minimum level of readiness. Under the status quo, the Navy Expeditionary Combat Enterprise (NECE) Capability Costing Model (NCCM) forecasts routine requirements using Excel Solver and data from the Optimized Fleet Response Plan (OFRP) and Certified Obligation Reports. This model receives historical data obtained from Command Financial Management System (DFMS), Standardized Accounting and Reporting System – Field Level (STARS-FL), past OFRP schedules, and notional OFRP schedules. Using least-square-optimization and various constraints, Solver determines the cost of each phase of the OFRP and then applies those costs to the notional OFRP schedule of each program. Note that the output of this model only relates to costs in the P/S/T (personnel, supply, and training) pillars. A separate deterministic model is used for the E pillar. This deterministic method is based on equipment allowances and maintenance factors associated with that equipment.

The reasoning behind the constraints used in Solver is unclear, and the sponsor believes better models to forecast costs exist. The purpose of this research, therefore, is to explore forecasting methods to improve the determination of requirements in the POM process. Initially, this analysis forecast at highest levels of aggregation. It then attempts to forecast routine costs of BSO 60 and BSO 70 across E/S and P/T pillars. The analysis combined these pillars because the distinction between E/S is sometimes unclear or confused. Finally, the author uses automated and hierarchical methods to forecast costs. This method is much quicker but provides less control over the parameters of the forecast.

B. ANALYTICAL APPROACH

The raw data of this research is historical costs and the planned number of expeditionary units in each phase of the Optimized Fleet Response Plan. The raw data contains programs other than EOD and funds other than Operations and Maintenance, Navy (OMN). The data also includes granularity that is outside of the scope of this research. The author therefore filters and summarizes the raw data to information relevant to this report.

The raw data also marks some P/S/T pillar costs as "excluded" because they are not representative of routine costs. All E pillar costs are marked as excluded because a different method is used to predict them. This technical report considers all "included" historical P/S/T costs. However, it also considers all E pillar costs, including the non-routine ones. The inclusion of non-routine equipment costs, which are not explicitly identified in the raw cost data, is a limitation in the analysis. The author inflates historical costs to fiscal year (FY) 22 based on the approved Office of Secretary of Defense (OSD) inflation factors where available and Consumer Price Index factors where OSD rates are not available.

The author divides this raw data into training and testing data. It then further divides both into data related to all EOD costs, BSO 60 P/T and S/E costs, and BSO 70 P/T and S/E costs. The author further wrangles this cost data into a combined data frame with OFRP schedules—this is necessary for dynamic regression which uses exogeneous variables like the number of units in each phase of the OFRP.

Using this raw data, the analysis creates numerous models within the broad categories of exponential smoothing, ARIMA, and dynamic regression. It also uses top-down and bottom-up hierarchical models based on ARIMA and exponential models. The R programming package and Fable package are the main tools to create all these models. The best model under each broad modeling category is determined based on MAE, MAPE, MASE, and subjective judgment where necessary. These best models are then forecasted two years into the future and compared to actual costs in a step comparable to the POM process.

II. ANALYSIS

This section explores different forecasting methods across different aggregations of cost data. The first sub-section relates to aggregated EOD costs and will include most of the graphs and tables used in the analysis. Later sub-sections, however, will include most of these items referenced appendixes. This report displays R code when appropriate.

A. AGGREGATED EOD COSTS

The author created the raw data in the following way. Note that the PE below is the PE for EOD costs. The costs were filtered to those that are "included"—that is, routine—or equipment costs. The APPN refers exclusively to the active-duty element. Note that Tsibble is a special type of data frame used by the Fable package.

```
EOD_ALL <- ActualCosts_forecasting %>%
  filter(PE == "0204424N",
         APPN == "OMN",
         FY >= 2017,
         EXCLUDED_INCLUDED == "Included" | PILLAR == "E") %>%
  group_by(fyMonth) %>%
  summarise(totalCost = sum(inflatedCost)) %>%
  select(fyMonth,totalCost) %>%
  as_tsibble(index = fyMonth)
```

Figure 1: Aggregated Raw Data for All EOD

The next step is to attempt exponential smoothing models using the following code in Figure 2. Note that the four primary types of exponential models are additive, multiplicative, additive damped, and multiplicative damped. These models differ based on error, trend, and season parameters. The parameters of the next four models are determined automatically based on an algorithm within the Fable package. The difference between the last four models is what the algorithm attempts to minimize: likelihood, average mean squared error, mean squared error, and mean absolute error.

```
EOD_ExponentialFits <- EOD_ALL_Train %>%
  model(Add = ETS(totalCost ~ error("A") + trend("A") + season("A")),
        HWMult = ETS(totalCost ~ error("M") + trend("A") + season("M")),
        Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A")),
        HWMult_Damped = ETS(totalCost ~ error("M") + trend("Ad") + season("M")),
        ETSAuto_LIK = ETS(totalCost, opt_crit = "lik", ic = "aicc"),
        ETSAuto_AMSE = ETS(totalCost, opt_crit = "amse", ic = "aicc"),
        ETSAuto_MSE = ETS(totalCost, opt_crit = "mse", ic = "aicc"),
        ETSAuto_MAE = ETS(totalCost, opt_crit = "mae", ic = "aicc"))

forecastEOD <- EOD_ExponentialFits %>%
  forecast(h=24)

accuracy(forecastEOD,EOD_ALL) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

Figure 2: Exponential Models for EOD

The result is that multiplicative damped and additive damped models are the best models.

```
## # A tibble: 8 × 5
##   .model         .type      MAE  MAPE  MASE
##   <chr>          <chr>    <dbl> <dbl> <dbl>
## 1 HWMult_Damped  Test  4365339.  44.6 0.515
## 2 Add_Damped     Test  4890149.  45.6 0.577
## 3 HWMult         Test  4870673.  49.4 0.574
## 4 Add            Test  5465968.  51.1 0.645
## 5 ETSAuto_LIK    Test  5468214.  70.7 0.645
## 6 ETSAuto_MAE    Test  5468214.  70.7 0.645
## 7 ETSAuto_MSE    Test  5468214.  70.7 0.645
## 8 ETSAuto_AMSE   Test  5712984.  73.4 0.674
```

Figure 3: Table of EOD Exponential Model Accuracy

The best model, multiplicative damped, looks like the following. Note that the black line is actual costs and the blue line is forecasted costs.

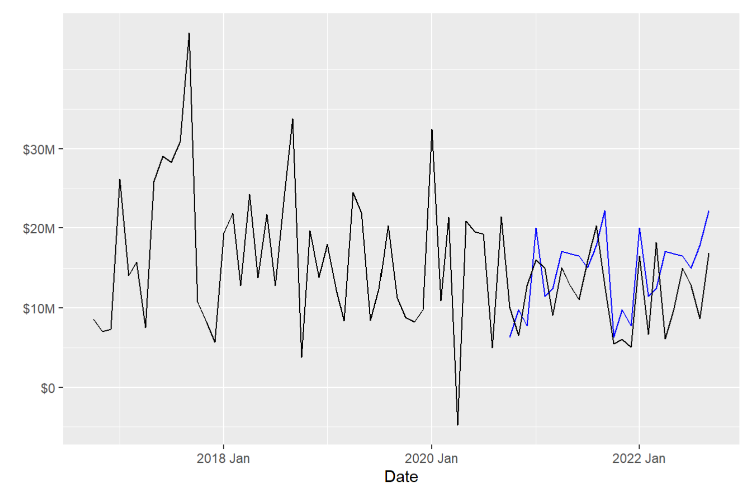


Figure 4: Graph of Best EOD Exponential Model

The next category of models is ARIMA models. These models require data that lacks trends and seasons. That is, the data must be “stationary.” Differencing and seasonal differencing can make non-stationary data stationary. After differencing, the data identifies the change from unit of time to the next unit of time, monthly in this case. Seasonal differencing identifies the change across seasons, yearly in this case. Fable contains a function that estimates the level of differencing required for the data to be stationary. Graphs of autocorrelation (ACF) and partial autocorrelation (PACF) are also useful because statistically significant ACF and PACF indicate that the data is not stationary.

Based on unitroot_ndiffs test in Fable, one differencing appears to be required for this data. Based on ACF and PACF graphs, however, no differencing appears to be required. Another stationary check will be conducted when the best model is identified. The final model must pass this test to be legitimate.

Figure 5 illustrates the ACF and PACF graphs. There does not appear to be any significant autocorrelation or partial autocorrelation. In addition, the charts do not provide any clear guidance on the order of the autoregressive or moving average parts of the ARIMA model. As a rule of thumb, a statistically significant ACF suggests the autoregressive term in the ARIMA model, and a statistically significant PACF suggests a weighted average term.

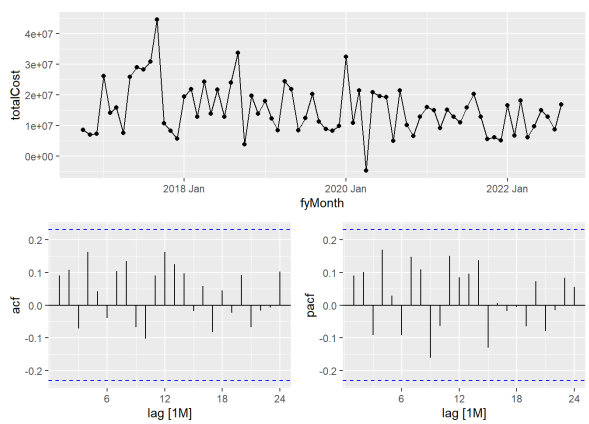


Figure 5: ACF/PACF Graphs for EOD ARIMA Models

The following code in Figure 6 creates the ARIMA models. The sixth model appears to be the best one.

```
EOD_ARIMAFits <- EOD_ALL_Train %>%
  model(stepwiseARIMA = ARIMA(totalCost,ic = "aicc", stepwise = TRUE),
        autoARIMA = ARIMA(totalCost, ic = "aicc",stepwise = FALSE, approximation = FALSE,),
        ARIMA1 = ARIMA(totalCost ~ pdq(1,1,1) +PDQ(1,1,1)),
        ARIMA2 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,0,1)),
        ARIMA3 = ARIMA(totalCost ~ pdq(1,0,0) + PDQ(0,1,1)),
        ARIMA4 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(1,1,0)),
        ARIMA5 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(1,1,0)),
        ARIMA6 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)),
        ARIMA7 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA8 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA9 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,1,1)),
        ARIMA10 = ARIMA(totalCost ~ pdq(1,1,1)),
        ARIMA11 = ARIMA(totalCost ~ pdq(0,1,1)))

ARIMAforecastEOD <- EOD_ARIMAFits %>%
  forecast(h=24)

accuracy(ARIMAforecastEOD,EOD_ALL) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

Figure 6: ARIMA Models for EOD

The accuracy of each ARIMA model are displayed in Figure 7.

```
## # A tibble: 13 × 5
##    .model        .type      MAE  MAPE   MASE
##    <chr>         <chr>    <dbl> <dbl>  <dbl>
##  1 ARIMA6        Test  3948069.  33.0  0.466
##  2 ARIMA3        Test  3986458.  33.3  0.470
##  3 ARIMA1        Test  4926561.  42.9  0.581
##  4 ARIMA4        Test  5212748.  44.9  0.615
```

Figure 7: Table of EOD Arima Model Accuracy

Figure 8 illustrates what the best model looks like.

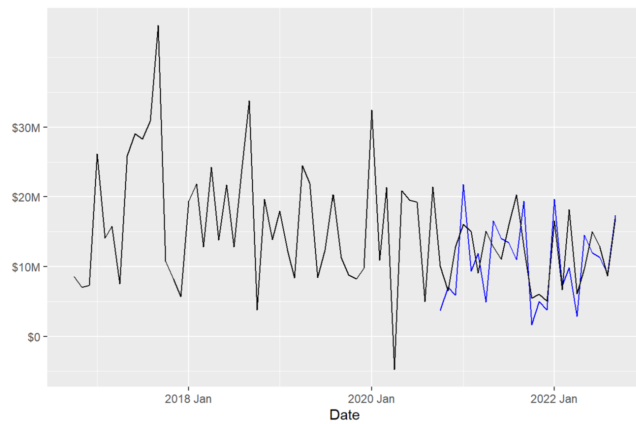


Figure 8: Graph of Best EOD ARIMA Model

Figure 9 provides the parameters of the best models.

```
## Series: totalCost
## Model: ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -1.0000  -0.5494
## s.e.   0.1174   0.3898
##
## sigma^2 estimated as 9.638e+13:  log likelihood=-616.33
## AIC=1238.66   AICc=1239.43   BIC=1243.33
```

Figure 9: Parameters for Best EOD ARIMA Model

The best model passes the Ljung-Box test, meaning that autocorrelation does not invalidate the model. The residuals displayed in Figure 10 also appear sufficiently normal.

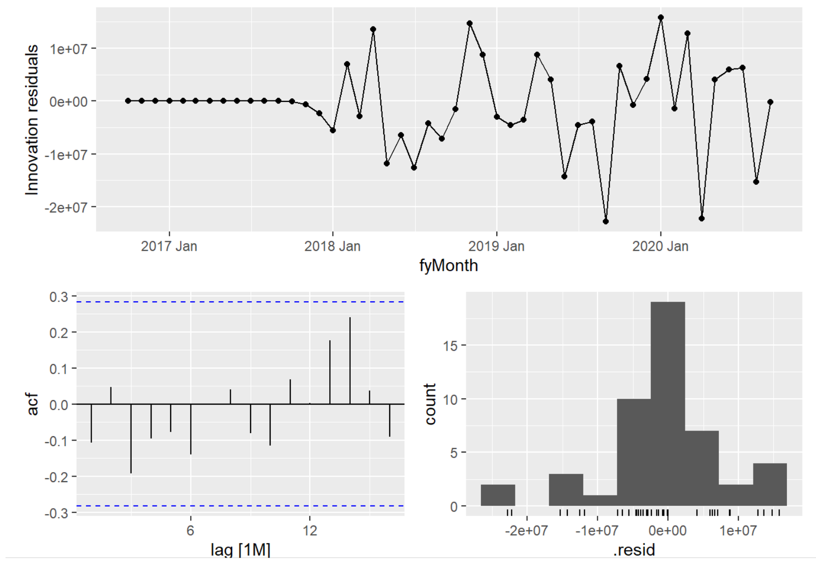


Figure 10: Residuals of Best EOD ARIMA Model

The final type of forecasting model is dynamic regression. In addition to the month and total cost, the raw data divides units across the maintenance, preparation, and readiness stages. Note that unit counts in these phases are the planned amount and not the actual—a certain amount of noise is, therefore, included in these numbers.

Dynamic regression includes relevant information other than past cost values and assumes that errors follow an ARIMA model. It requires that all the models in the variable be stationary. Unfortunately for ease of interpretation, this is not the case for any of the exogenous variables—they all fail the Ljung-Box test, some by massive amounts. After differencing the exogeneous variables, they are now sufficiently stationary. Figure 11 provides what the data frame looks like after one differencing. After differencing the exogeneous variables, they are now sufficiently stationary.

```
## # A tsibble: 6 x 5 [1M]
##    fyMonth maintenence preparation readiness totalCost
##      <mth>       <int>       <int>     <int>     <dbl>
## 1 2018 Feb           2          -4         2 21834786.
## 2 2018 Mar           2           1        -3 12796587.
## 3 2018 Apr          -9           7         2 24234735.
## 4 2018 May           0           0         0 13780909.
## 5 2018 Jun           7          -1        -6 21705880.
## 6 2018 Jul           2          -8         6 12828784.
```

Figure 11: Table of Raw EOD Data for Dynamic Regression

Figure 12 shows a few models. The analysis includes models with lagged values for readiness because the cost of "expending readiness" in deployments may not appear until sometime after the deployment and sustainment phases, which is referred to as “readiness” in the author’s model. The analysis also includes models that force the inclusion of pre-determined ARIMA errors.

```
EOD_AdvancedARIMA_Fit <- EOD_ALL_Train_DR %>%
  model(advancedARIMA1 = ARIMA(totalCost ~ 0 + diff(maintenence,differences = 1) + diff(readiness,differences = 1)),
        advancedARIMA2 = ARIMA(totalCost ~ 0 + maintenence + preparation),
        advancedARIMA3 = ARIMA(totalCost ~ 0 + preparation + readiness),
        advancedARIMA_maint = ARIMA(totalCost ~ 0 + maintenence),
        advancedARIMA_read = ARIMA(totalCost ~ 0 + readiness),
        advancedARIMA_prep = ARIMA(totalCost ~ 0 + preparation),
        advancedARIMA1_lag = ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6)),
        advancedARIMA3_lag = ARIMA(totalCost ~ 0 + preparation + lag(readiness,6)),
        advancedARIMA_read_lag = ARIMA(totalCost ~ 0 + lag(readiness,6)),
        advancedARIMA_misc1 = ARIMA(totalCost ~ 0 + pdq(1,0,0) + readiness),
        advancedARIMA_misc2 = ARIMA(totalCost ~ 0 + pdq(1,0,0) + preparation + readiness),
        advancedARIMA_misc3 = ARIMA(totalCost ~ 0 + pdq(1,1,0) + readiness),
        advancedARIMA_misc3 = ARIMA(totalCost ~ 0 + pdq(1,1,0) + preparation + readiness))

EOD_AdvancedARIMA_Forecast <- EOD_AdvancedARIMA_Fit %>%
  forecast(new_data = EOD_ALL_Test_DR,h = 24)

accuracy(EOD_AdvancedARIMA_Forecast,advancedARIMA_EOD) %>%
  select(.model,MAE,MAPE,MASE) %>%
  arrange(MAPE) %>%
  head(5)
```

Figure 12: Dynamic Regression Models for EOD

The best model appears to be “advancedARIMA1_lag” based on Figure 13..

```
## # A tibble: 5 × 4
##   .model                   MAE  MAPE  MASE
##   <chr>                  <dbl> <dbl> <dbl>
## 1 advancedARIMA1_lag  3707525.  41.1 0.400
## 2 advancedARIMA2      6014538.  45.5 0.649
## 3 advancedARIMA_maint 6241311.  47.2 0.673
## 4 advancedARIMA_prep  6115395.  47.8 0.660
## 5 advancedARIMA_read  6152107.  48.7 0.664
```

Figure 13: Table of EOD Dynamic Regression Model Accuracy

Figure 14 depicts what the best model looks like. The analysis includes the confidence levels to show that the confidence level expands dramatically with time.

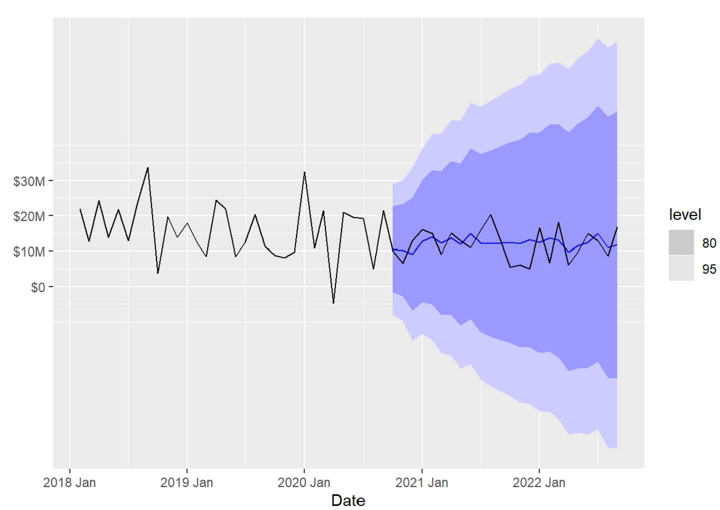


Figure 14: Graph of Best EOD Dynamic Regression Model

The parameters of the best model are given in Figure 15.

```
## Series: totalCost
## Model: LM w/ ARIMA(1,1,0) errors
##
## Coefficients:
##           ar1  maintenence  lag(readiness, 6)
##       -0.6172    -570391.1           87377.94
## s.e.   0.1544     311444.2          152213.58
##
## sigma^2 estimated as 8.995e+13:  log likelihood=-438.76
## AIC=885.52   AICc=887.06   BIC=891.25
```

Figure 15: Parameters for Best EOD Dynamic Regression Model

The residuals are represented in the graphs in Figure 16. The tail of the distribution is larger than residuals produced by other models, indicating that this method is not the best model.

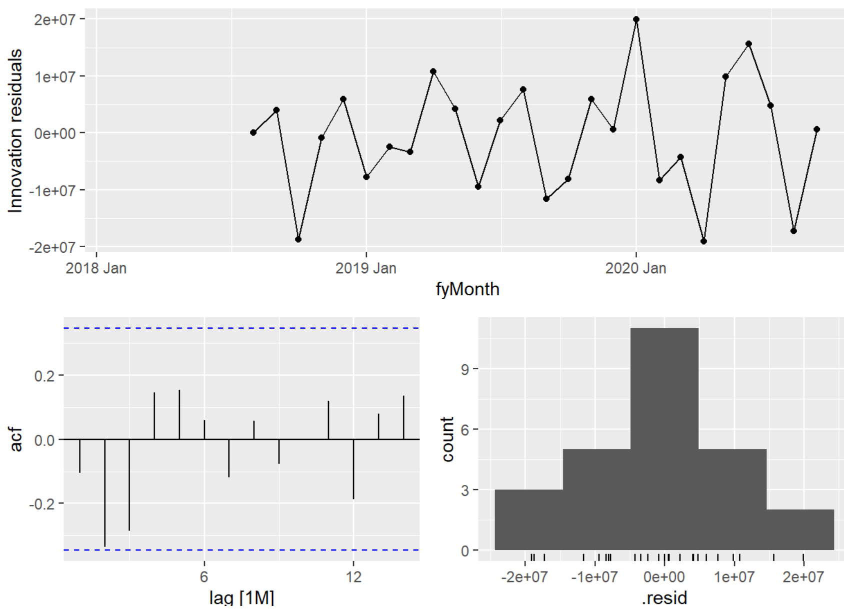


Figure 16: Residuals for Best EOD Dynamic Regression Model

In conclusion, several models provide strong modeling potential. ARIMA6, however, appears to be the best based on its simplicity and accuracy. Surprisingly, while calling for an ARIMA model, the result is simple exponential smoothing model with a seasonal element.

B. BSO 60: P/T PILLARS

After exploration of forecasts for overall expenditure, the next step is to take a step down in the hierarchy: where the BSO is 60 and the pillars are P/T. This is the first section where most of the figures, graphs, and tables will be included in the appendix. The data was wrangled in a similar fashion as overall costs.

A large outlier of -$6,180,000 is included in this dataframe. The List Item (LI) of 1C6C indicates that it relates to "Combat Support Forces," and the pillar is T. Because this cost is marked as "included," the author left it in the analysis. A large negative value under the T-pillar is likely due to recoupment of funds previously obligated to a training contract.

The first category of the model is exponential. Appendix-A1 is a list of several models and their accuracy. The best model is Additive damped. The MASE for the best model, however, is over 1. This indicates that the naive model outperforms the proposed models. Although this is typically an indication of poor model quality, the large outlier may be distorting accuracy calculations.

The graph of this best model is presented in Figure 17. Apart from the large outlier, it appears to be a better model than the accuracy models suggest. The residuals, shown in Appendix-A2, appear to be skewed right.

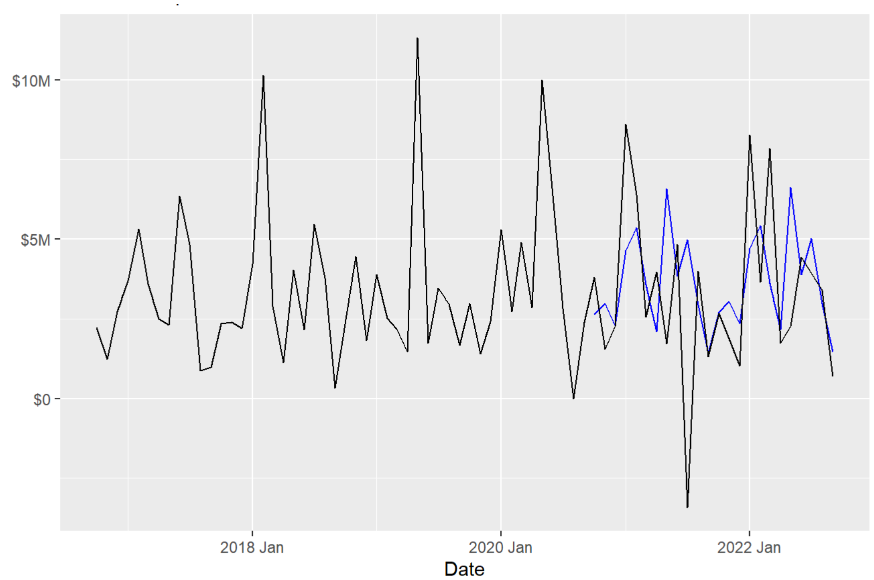


Figure 17: Graph of Best BSO60 (P/T) Exponential Model

The next broad model category is ARIMA. No differencing appears to be required based on the unitroot_ndiffs test. The ACF and PACF charts shown in Appendix-A3, however, show a less clear picture. The ACF and PACF suggest a possible bi-yearly seasonality rather than a yearly one.

Appendix-A4 contains the tested ARIMA models. The second model appears to be the best one. However, the MASE is still above one.

The best ARIMA model is seen in Figure 18.

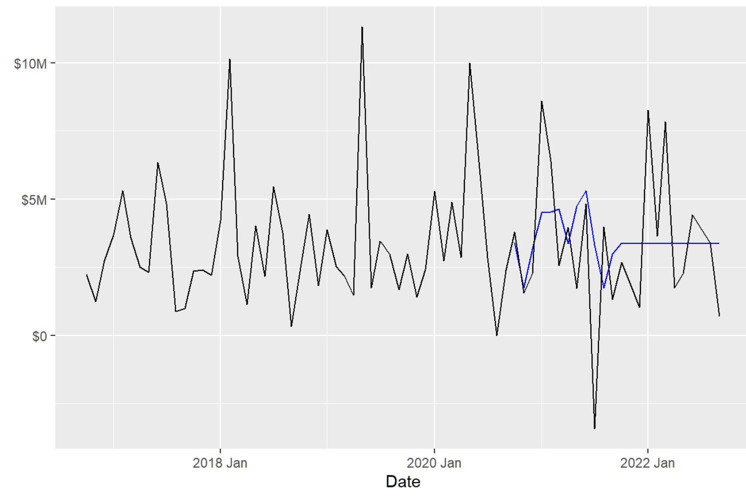


Figure 18: Graph of Best BSO60 (P/T) ARIMA Model

Its specific parameters are illustrated in Appendix-A5. This model passes the Ljung Box Test. In addition, the residuals appear to be reasonably normal but contain two outliers.

The final type of forecasting model is dynamic regression. Appendix-A6 contains several models and their testing accuracy. The "advancedARIMA_Read" is the best model. The graph of the model is in Appendix-A7 for readability. It appears to possess poor quality.

In conclusion, the best model appears to be exponential smoothing, although the accuracy measures for this model are still poor.

C. BSO 60: S/E PILLARS

The author wrangled this data frame in the same manner as before with one exception: A missing row was added because zero dollars appear to be spent on S/E in one month: Oct, 2018.

The first category is exponential. Appendix-A8 contains the modelling attempts and accuracy measures. The best model appears to be the multiplicative one.

This graph of this exponential model is in Figure 19. The residuals appear to be reasonable as illustrated in Appendix-A9.

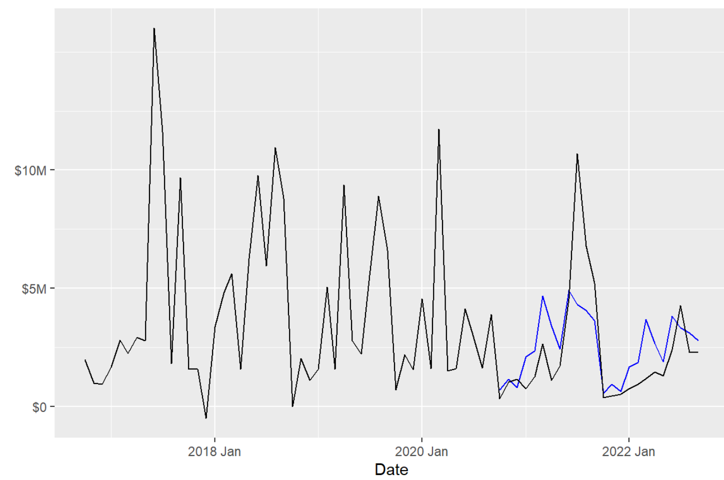


Figure 19: Graph of Best BSO60 (S/E) Exponential Model

The next model is the ARIMA model. No differencing seems to be required based on the unitroot_ndiffs test, but the ACF and PACF graphs in Appendix-A10 appear to indicate that autocorrelation may be a problem.

Appendix-A11 contains the ARIMA modeling attempts and their accuracy measures. The best model is in Figure 20.

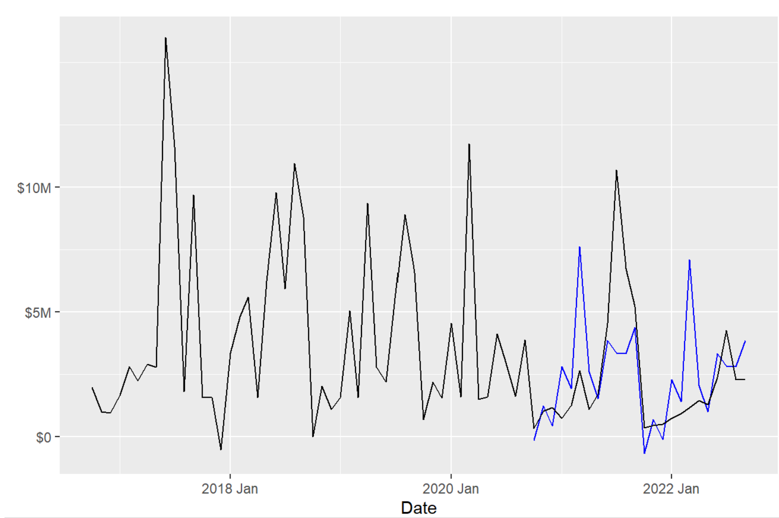


Figure 20: Graph of Best BSO60 (S/E) ARIMA Model

The parameters of this model are in Appendix-A12. The best model passes the Ljung_Box test, but only barely with an alpha of 0.05. This is an indication that this is not an appropriate model. The residuals are in Appendix-A13.

The final forecasting model is dynamic regression. Appendix-A14 contains the modeling attempts and their accuracy measures. The predictive power of these model categories appears to be comparable to other attempts. In addition, the measures of accuracy point to different models as the best one. The author subjectively chose advancedARIMA3 as the best model.

The graph of this model is in Figure 21. The parameters of the best model are contained in Appendix-A15.

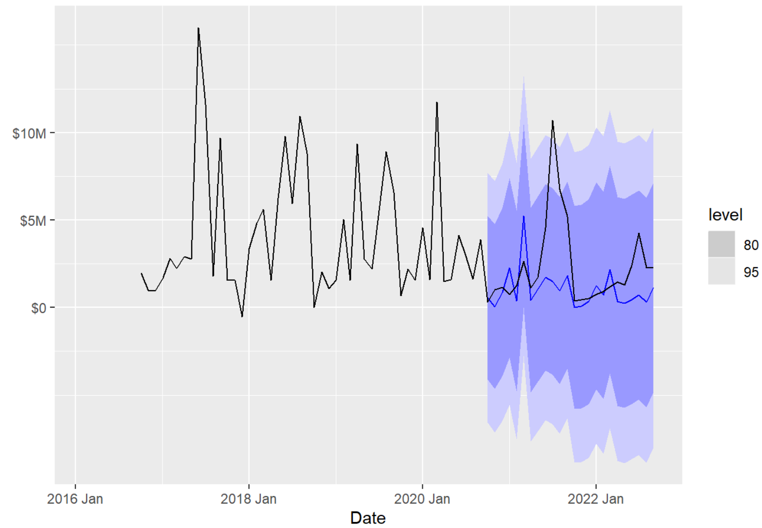


Figure 21: Graph of Best BSO60 (S/E) Dynamic Regression Model

Based on Appendix-A16, the residuals appear to be skewed right.

In conclusion, the exponential model appears to be the best model. However, the unusual shape of these execution costs create difficulty in predicting these costs.

D. BSO 70: P/T PILLARS

The author wrangled the data for this level of aggregation like before.

The first modeling attempt is exponential. Appendix-A17 contains several modeling attempts and their accuracy measures. The best model appears to be multiplicative.

The graph of the best exponential model according to accuracy measures is contained in Appendix-A18. It does not track the data well. Based on a subjective assessment, however, the second-best model, additive damped, is superior. It is graphed in Figure 22. Interestingly, both the best and second-best models tend to underestimate actual costs.

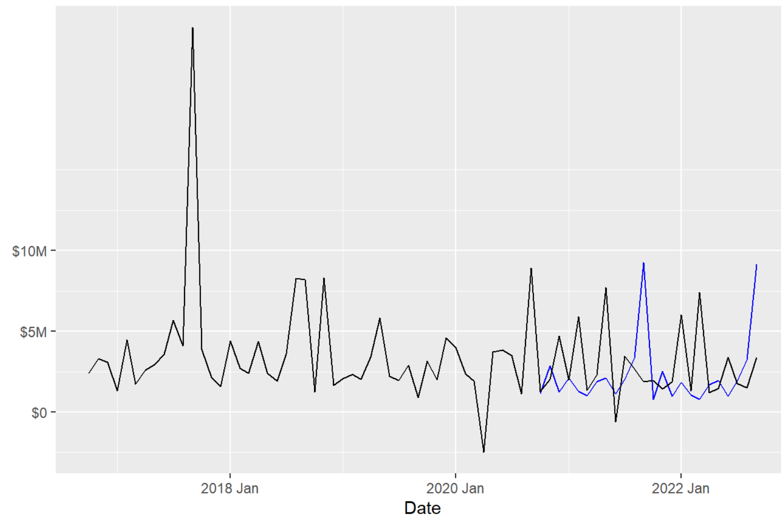


Figure 22: Graph of Best BSO70 (P/T) Exponential Model

The residuals of this model appear to be skewed left as shown in Appendix-A19.

The next set of models is ARIMA. No differencing appears to be required as shown by the unitroot_ndiffs test. The ACF and PACF graphs, contained in Appendix-A20, show the same thing.

Appendix-A21 shows several attempts at models their accuracy measures. The best model appears to be the sixth one, although the accuracy measures point to different models as the best. The graph of this model is shown in Appendix-A22. It does not fit the data well. The parameters of this model are in Appendix-A23.

The best model passes the Ljung_Box test. Appendix-A24 shows the residuals of the model. It appears sufficiently normal but contains several large outliers.

The final model type is dynamic regression. Appendix-A25 shows several attempts at models and their accuracy measures. The best model appears to be advancedARIMA1_lag.

The model is displayed in Figure 23. It does not fit the data well but may be the best model for this level of aggregation in cost data.

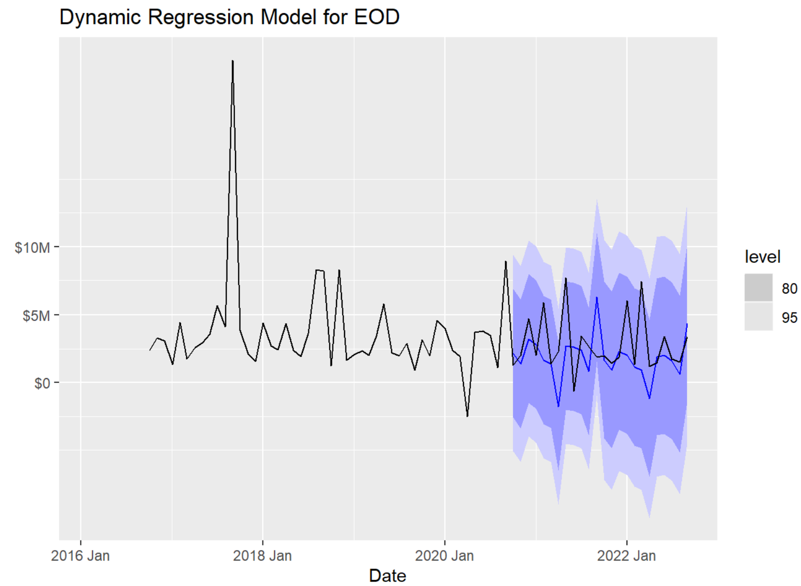


Figure 23: Graph of Best BSO70 (P/T) Dynamic Regression Model

The parameters of the models are in Appendix-A26. It contains one seasonal regression term as an error and two coefficients for maintenance and lagged readiness. Appendix-A27 contains the residuals—they appear reasonably normal. It also passes the Ljung Box Test.

In conclusion, the best model appears to be dynamic regression.

E. BSO 70: S/E PILLARS

The author wrangled the data for this level of aggregation like before.

The first modeling category is exponential. Appendix-A28 contains several modeling attempts and their accuracy measures. The graph of this best model looks like the multiplicative model and additive model (Figure 24). The author chose the additive model as the best one because it does not consistently overestimate actual costs.

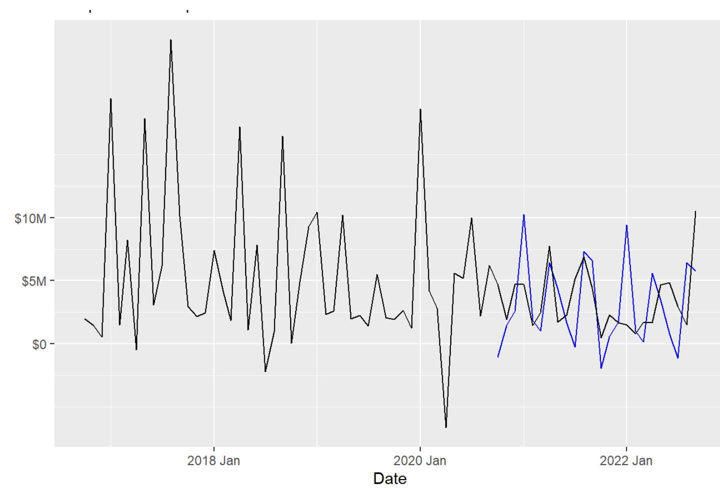


Figure 24: Graph of Best BSO70 (S/E) Exponential Model

The residuals appear to be normal based on Appendix-A29. It may be the most normal distribution yet.

The next model type is ARIMA. One differencing appears to be required to force the model to be stationary. The ACF and PACF plots contained in Appendix-A30 appear to be less clear, however. Appendix-A31 contains several ARIMA modeling attempts and their accuracy measures.

The best model appears to be the fifth one and is graphed in Figure 25.

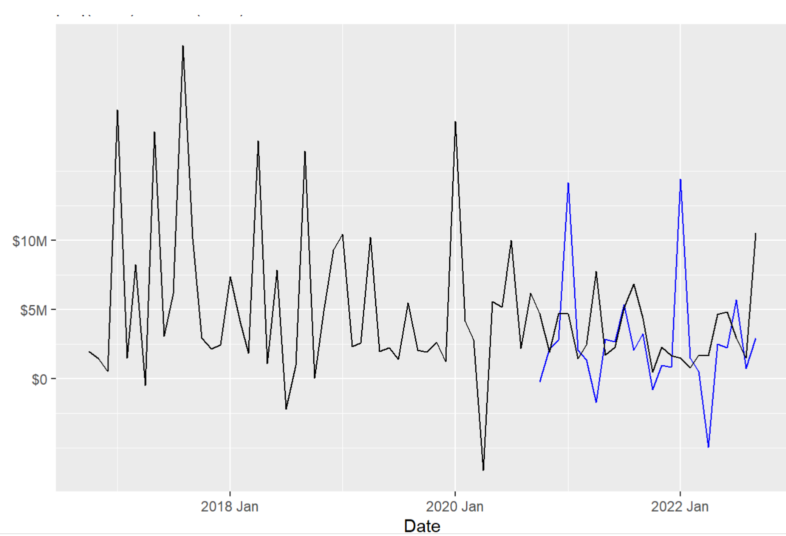


Figure 25: Graph of Best BSO70 (S/E) ARIMA Model

The parameters are in Appendix-A32. It passes the Ljung Box Test. The residuals, as shown in Appendix-A33, appear to be excessively flat.

The final type of model is dynamic regression. Appendix-A34 contains several dynamic regression modeling attempts and their accuracy measures. The best model appears to be the first one (Figure 26).

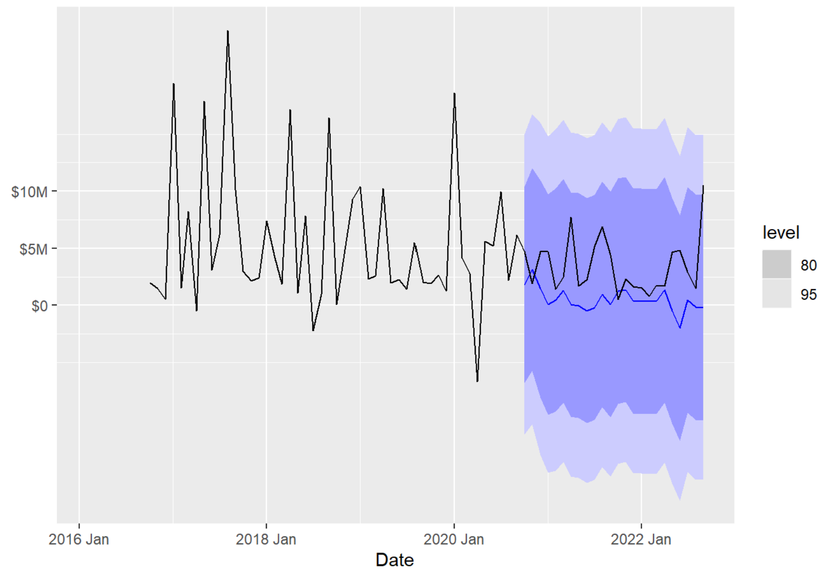


Figure 26: Graph of Best BSO70 (S/E) Dynamic Regression Model

This model consistently underestimates actual cost. The parameters of the best model are in Appendix-A34. The residuals are contained in Appendix-A35.

In conclusion, the best model appears to be the ARIMA model for this level of cost aggregation.

F. HIEARCHICAL METHODS

Hierarchical methods are useful because they are quick and automated. The algorithm, however, does not allow the inclusion of exogeneous variables. The author creates a training and test set as before, and then creates hierarchical exponential smoothing as well as hierarchical ARIMA models. There is only one method to determine the bottom-up forecasts. The top-town method contains four separate methods based on different estimation criteria.

For the sake of exploration, the following are the disaggregated costs across pillars and BSOs. No obvious pattern emerges as shown in Figure 27.

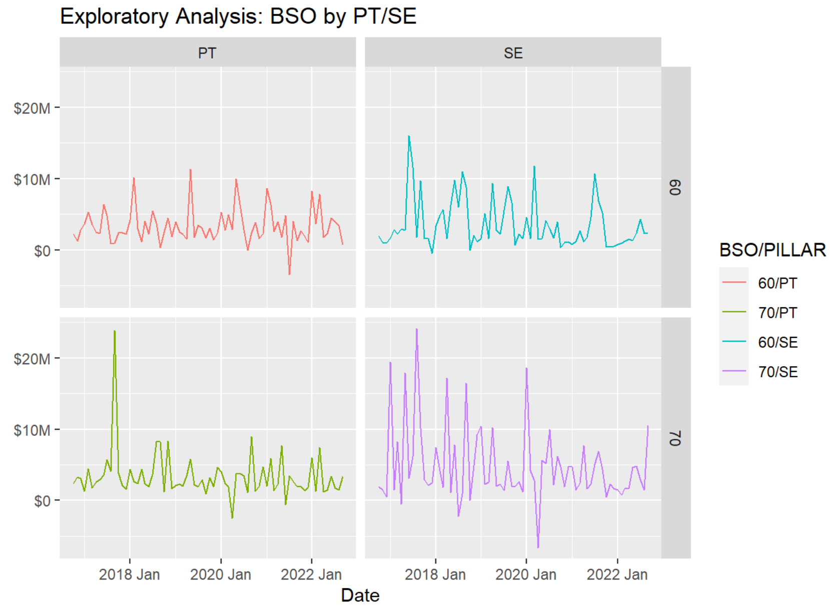


Figure 27: Graph of Hierarchical Data Exploration

Appendix-A36 contains hierarchical exponential models, and Appendices-A37 and A38 contain the accuracy from aggregated and disaggregated perspectives. Appendix-A39 shows their aggregated forecast. Their forecasts are in Figure 28. Some of the automated forecasts for BSO 60 and P/T pillar do well. The other models do not appear to be accurate.

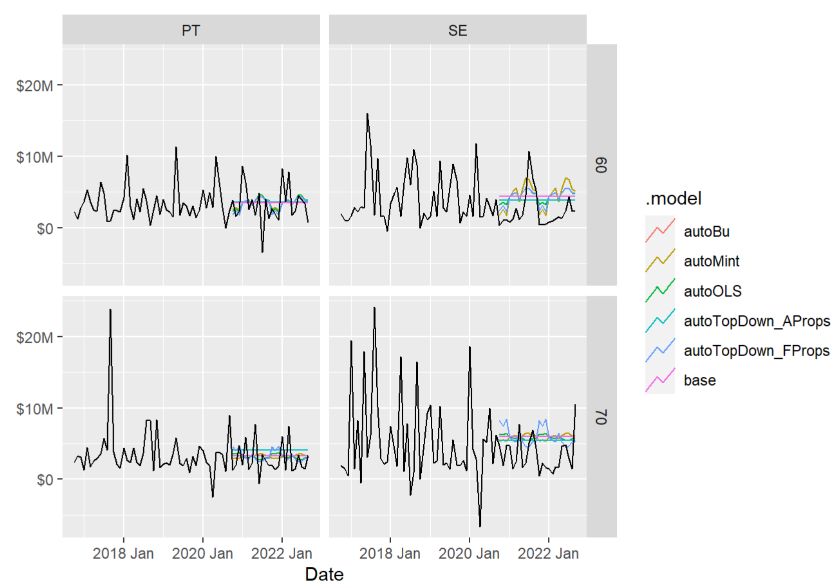


Figure 28: Graph of Hierarchical Exponential Models

The next hierarchical method is ARIMA methods. The author chose one model based on the best ARIMA model for overall costs. The second model is an automatically determined model. Appendix-A40 contains exponential models and their forecasts below and Appendix-A41 and A42 contains their accuracy measures. The graphs of the disaggregated forecasts are in Figure 29, and Appendix-A43 shows the aggregated forecasts.

Figure 29: Graph of Hierarchical ARIMA Models

In general, hierarchical ARIMA models appear to perform better than exponential models. However, only one aggregated and two disaggregated forecasts perform well.

III. RESULTS AND CONCLUSION

A. VALIDATING SHIP COUNT

The following is what the summary of the best-of-category model performances relative to the test set two years into the future. Notice that the best model performances are very strong while other areas are very poor (Figure 30).

Cost Set	Model Type	Model	MAE	MAPE	MASE	Predicted Cost	Delta (Predicted - Actual)	Delta Percent
EOD	Exponential	ETS(totalCost ~ error("M") + trend("Ad") + season("M")	4365339.00	44.57	0.51	$173,445,896.00	$ 46,134,806.00	36%
EOD	ARIMA	ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)	3948069.00	33.01	0.47	$114,130,211.00	$ (13,180,879.00)	-10%
EOD	Dynamic	ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6))	3707525.00	41.09	0.40	$148,243,588.00	$ 20,932,498.00	16%
EOD60_PT	Exponential	Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A"))	1898313.00	65.49	1.09	$43,999,300.00	$ 2,248,869.00	5%
EOD60_PT	ARIMA	ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,0,1))	1895014.00	75.90	1.08	$40,447,816.00	$ (1,302,615.00)	-3%
EOD60_PT	Dynamic	ARIMA(totalCost ~ 0 + readiness)	2408705.00	63.88	1.38	$17,238,442.00	$ (24,511,989.00)	-59%
EOD60_SE	Exponential	ETS(totalCost ~ error("M") + trend("A") + season("M"))	1248080.00	74.22	0.38	$26,912,125.00	$ 8,749,408.00	48%
EOD60_SE	ARIMA	ARIMA(totalCost ~ pdq(1,1,1) +PDQ(1,1,1))	1670622.00	95.34	0.51	$24,646,669.00	$ 6,483,952.00	36%
EOD60_SE	Dynamic	ARIMA(totalCost ~ 0 + readiness)	1768410.00	70.21	0.57	$7,803,036.00	$ (10,359,681.00)	-57%
EOD70_PT	Exponential	Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A"))	2167099.00	77.63	0.78	$26,862,215.00	$ (5,898,138.00)	-18%
EOD70_PT	ARIMA	ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1))	2063416.00	86.96	0.74	$18,621,928.00	$ (14,138,425.00)	-43%
EOD70_PT	Dynamic	advancedARIMA1_lag = ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6))	1910743.00	81.21	0.68	$18,209,107.00	$ (14,551,246.00)	-44%
EOD70_SE	Exponential	ETS(totalCost ~ error("A") + trend("A") + season("A"))	3179989.00	118.71	0.53	$31,886,067.00	$ (2,751,522.00)	-8%
EOD70_SE	ARIMA	ARIMA(totalCost ~ pdq(1,1,0) + PDQ(1,1,0))	2713753.00	117.18	0.45	$26,640,458.00	$ (7,997,131.00)	-23%
EOD70_SE	Dynamic	ARIMA(totalCost ~ 0 + maintenence + readiness)	3118256.00	85.85	0.52	$3,057,353.00	$ (31,580,236.00)	-91%

Figure 30: Summary of Standard Model Performances

The "autoMint" method appears to be the best method for hierarchical Exponential models and the "average proportions" method appears to be the best method for the hierarchical ARIMA model. Surprisingly this is true for the best aggregated prediction and best disaggregated predictions. The ARIMA models tend to be the better of the two sets of hierarchical models. The best and worst models are exponential ones: a delta of 1 percent and 199 percent as shown in Figure 30.

Cost Set	Model Type	Model	MAE	MAPE	MASE	Predicted Cost	Delta (Predicted - Actual)	Delta Percent
EOD	Exponential	autoMint	5541778.00	65.35	0.65	$ 206,942,682.00	$ 79,631,592.00	63%
EOD	ARIMA	top_down(ARIMA1,method = "average_proportions")	3948069.00	33.01	0.47	$ 114,130,212.00	$ (13,180,878.00)	-10%
EOD60_PT	Exponential	[redacted]				$ 42,250,077.00	$ 499,646.00	1%
EOD60_PT	ARIMA	[redacted]				$ 24,196,661.00	$ (17,553,770.00)	-42%
EOD70_PT	Exponential	[redacted]				$ 39,147,347.00	$ 6,386,994.00	19%
EOD70_PT	ARIMA	[redacted]				$ 27,439,533.00	$ (5,320,820.00)	-16%
EOD60_SE	Exponential	[redacted]				$ 54,304,707.00	$ 36,141,990.00	199%
EOD60_SE	ARIMA	[redacted]				$ 26,070,557.00	$ 7,907,840.00	44%
EOD70_SE	Exponential	[redacted]				$ 71,240,551.00	$ 36,602,962.00	106%
EOD70_SE	ARIMA	[redacted]				$ 36,423,460.00	$ 1,785,871.00	5%

Figure 31: Summary of Hierarchical Model Performances

Using the summary information above in Figure 31, traditional forecasting techniques appear to do well at forecasting FY22 costs for Aggregated EOD, EOD60_PT projections, and EOD70_SE. However, it does poorly with EOD60_SE and EOD70_PT. Overall, the additional work to create unique models at a disaggregated level appears to be worth the additional time to produce them.

IV. RECOMMENDATIONS FOR FUTURE WORK

The following are some recommendations for future work.

A. CREATE ADDITIONAL TRAINING AND TEST SETS

Create multiple training and testing splits to verify that the best models for one training/test split are consistently the best across the other splits. Future analysis, for example, can apply the models that are currently the best to predict FY23 costs. The training and testing split would be Oct 2021 rather than Oct 2020.

B. EXPLORE RELEVANCE OF MONTHLY ACCURACY MEASURES

Explore what accuracy metrics are the most meaningful in predicting yearly accuracy. Based on quick analysis of correlation between monthly testing accuracy and absolute values of percent deltas for FY22, MAE possesses the strongest negative correlation while MAPE has a weak positive one—a positive correlation indicates that MAPE may not be a meaningful measure if the goal is to predict yearly costs.

C. IDENTIFY AND REMOVE OUTLIERS

Work with sponsor to determine what outlier costs can be excluded from analysis of routine costs. The presence of extreme values (e.g., a monthly expenditure of -$6M) distorts budget accuracy.

D. FUTHER EXPLORE LEVELS OF AGGREATION

Further explore the appropriate level of aggregation to create accurate forecasts and apply forecasting techniques to other programs. The next step would be to create a forecast for each BSO or a budget for each division of the BSO budget besides pillars.

V. APPENDICES

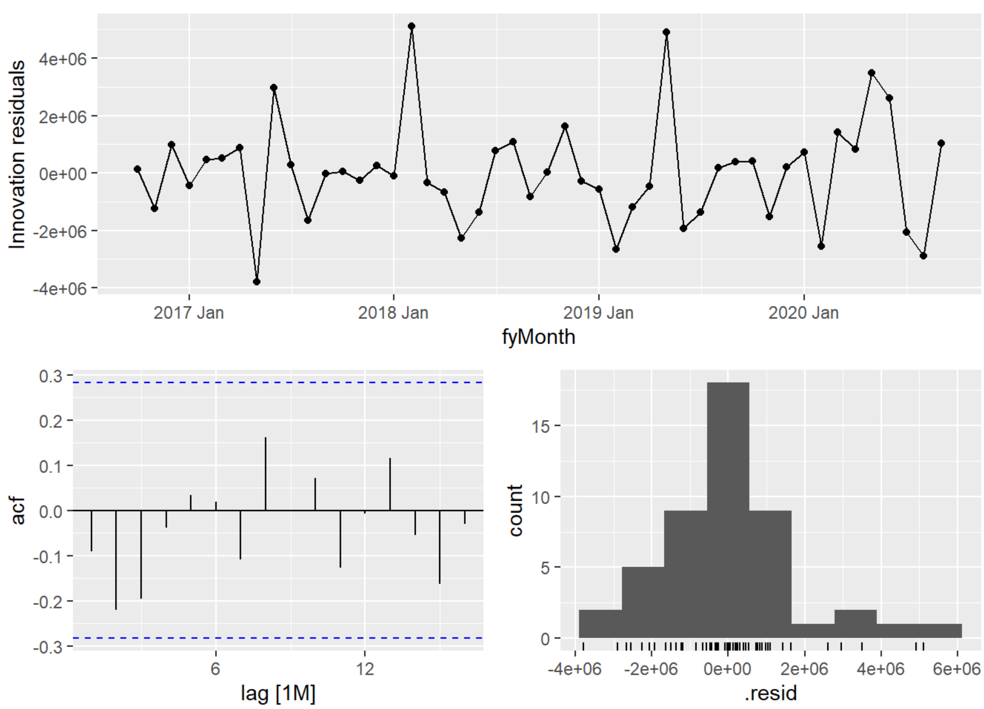
A. GRAPHS AND TABLES

1. BSO60 (P/T) Exponential Models and Accuracy

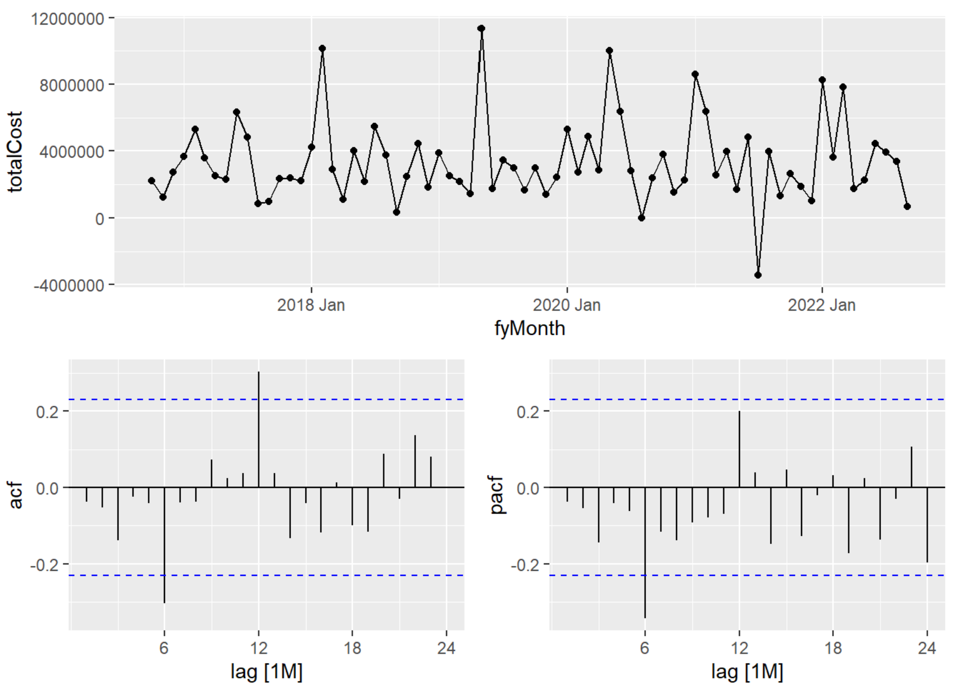
```
EOD_60PT_ExponentialFits <- EOD60_PT_Train %>%
  model(Add = ETS(totalCost ~ error("A") + trend("A") + season("A")),
        HWMult = ETS(totalCost ~ error("M") + trend("A") + season("M")),
        Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A")),
        HWMult_Damped = ETS(totalCost ~ error("M") + trend("Ad") + season("M")),
        ETSAuto_LIK = ETS(totalCost, opt_crit = "lik", ic = "aicc"),
        ETSAuto_AMSE = ETS(totalCost, opt_crit = "amse", ic = "aicc"),
        ETSAuto_MSE = ETS(totalCost, opt_crit = "mse", ic = "aicc"),
        ETSAuto_MAE = ETS(totalCost, opt_crit = "mae", ic = "aicc"))

forecastEOD_60PT <- EOD_60PT_ExponentialFits %>%
  forecast(h=24)

accuracy(forecastEOD_60PT,EOD60_PT) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 8 × 5
##   .model        .type       MAE  MAPE  MASE
##   <chr>         <chr>     <dbl> <dbl> <dbl>
## 1 Add_Damped    Test  1898313.  65.5  1.09
## 2 ETSAuto_MAE   Test  1948885.  65.6  1.12
## 3 HWMult_Damped Test  1875451.  66.8  1.07
## 4 HWMult        Test  2004874.  71.6  1.15
## 5 Add           Test  2015117.  79.0  1.15
## 6 ETSAuto_LIK   Test  1956773.  79.4  1.12
## 7 ETSAuto_AMSE  Test  1956773.  79.4  1.12
## 8 ETSAuto_MSE   Test  1956773.  79.4  1.12
```

2. Residuals for Best BSO60 (P/T) Exponential Model



3. ACF/PACF Charts for BSO60 (P/T)



4. BSO60 (P/T) ARIMA Models and Accuracy

```
EOD60_PT_ARIMAFits <- EOD60_PT_Train %>%
  model(stepwiseARIMA = ARIMA(totalCost,ic = "aicc", stepwise = TRUE),
        autoARIMA = ARIMA(totalCost, ic = "aicc",stepwise = FALSE, approximation = FALSE,),
        ARIMA1 = ARIMA(totalCost ~ pdq(1,1,1) +PDQ(1,1,1)),
        ARIMA2 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,0,1)),
        ARIMA3 = ARIMA(totalCost ~ pdq(1,0,0) + PDQ(0,1,1)),
        ARIMA4 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(1,1,0)),
        ARIMA5 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(1,1,0)),
        ARIMA6 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)),
        ARIMA7 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA8 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA9 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,1,1)))

ARIMAforecastEOD60_PT <- EOD60_PT_ARIMAFits %>%
  forecast(h=24)

accuracy(ARIMAforecastEOD60_PT,EOD60_PT) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 11 × 5
##    .model        .type      MAE  MAPE  MASE
##    <chr>         <chr>    <dbl> <dbl> <dbl>
##  1 ARIMA2        Test  1895014.  75.9  1.08
##  2 autoARIMA     Test  1902271.  76.2  1.09
##  3 stepwiseARIMA Test  1902271.  76.2  1.09
##  4 ARIMA9        Test  2170804.  78.7  1.24
##  5 ARIMA3        Test  2142456.  79.3  1.23
##  6 ARIMA7        Test  2186053.  79.3  1.25
##  7 ARIMA8        Test  2186053.  79.3  1.25
##  8 ARIMA5        Test  2411719.  84.3  1.38
##  9 ARIMA1        Test  2188521.  85.0  1.25
## 10 ARIMA6        Test  2252307.  90.7  1.29
## 11 ARIMA4        Test  2406013.  96.8  1.38
```

5. Parameters for Best BSO60 (P/T) ARIMA Model

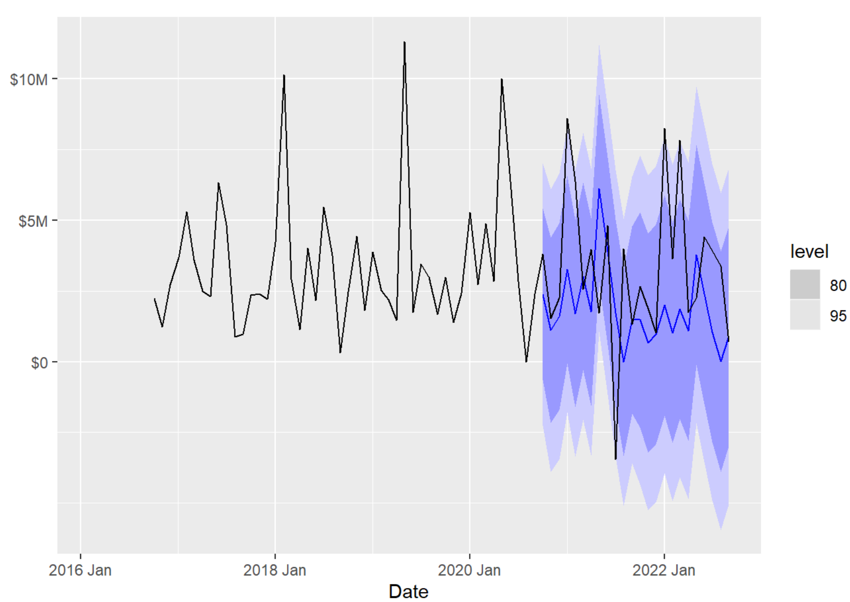
```
## Series: totalCost
## Model: ARIMA(1,0,1)(0,0,1)[12] w/ mean
##
## Coefficients:
##           ar1     ma1    sma1   constant
##       -0.3271  0.4347  0.6287  4471426.5
## s.e.   0.6243  0.5819  0.2404   612050.5
```

6. BSO60 (P/T) Dynamic Regression Models and Accuracy

```
EOD60_PT_AdvancedARIMA_Fit <- EOD60_PT_Train_DR %>%
  model(advancedARIMA1 = ARIMA(totalCost ~ 0 + maintenence + readiness),
        advancedARIMA2 = ARIMA(totalCost ~ 0 + maintenence + preparation),
        advancedARIMA3 = ARIMA(totalCost ~ 0 + preparation + readiness),
        advancedARIMA_maint = ARIMA(totalCost ~ 0 + maintenence),
        advancedARIMA_read = ARIMA(totalCost ~ 0 + readiness),
        advancedARIMA_prep = ARIMA(totalCost ~ 0 + preparation),
        advancedARIMA1_lag = ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6)),
        advancedARIMA3_lag = ARIMA(totalCost ~ 0 + preparation + lag(readiness,6)),
        advancedARIMA_read_lag = ARIMA(totalCost ~ 0 + lag(readiness,6)),
        advancedARIMA_misc1 = ARIMA(totalCost ~ 0 + pdq(1,0,1) + readiness),
        advancedARIMA_misc2 = ARIMA(totalCost ~ 0 + pdq(1,0,1) + preparation + readiness))

EOD60_PT_AdvancedARIMA_Forecast <- EOD60_PT_AdvancedARIMA_Fit %>%
  forecast(new_data = EOD60_PT_Test_DR)

accuracy(EOD60_PT_AdvancedARIMA_Forecast,advancedARIMA_EOD60_PT) %>%
  select(.model,MAE,MAPE,MASE) %>%
  arrange(MAPE) %>%
  head(5)
```

```
## # A tibble: 5 × 4
##   .model                     MAE  MAPE  MASE
##   <chr>                    <dbl> <dbl> <dbl>
## 1 advancedARIMA_read     2408705.  63.9  1.38
## 2 advancedARIMA_read_lag 2424229.  64.1  1.39
## 3 advancedARIMA3         2410710.  64.5  1.38
## 4 advancedARIMA3_lag     2429806.  64.7  1.39
## 5 advancedARIMA_prep     2411010.  64.8  1.38
```

7. Graph of Best Dynamic Regression Model for BSO60 (P/T)

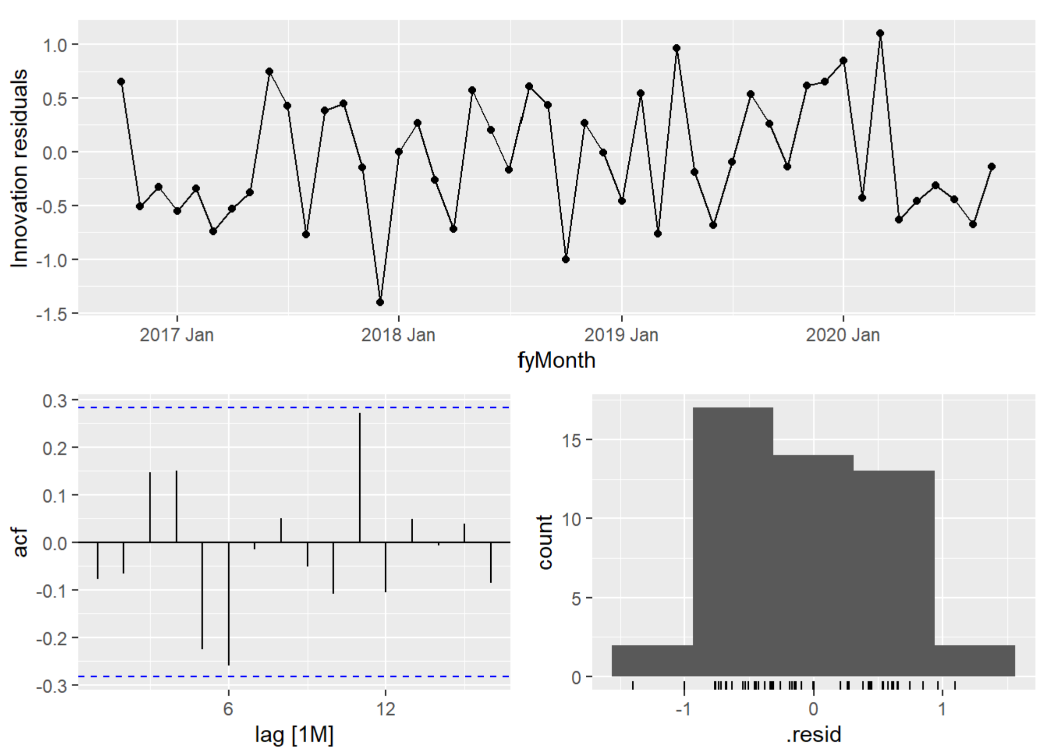


8. BSO60 (S/E) Exponential Models and Accuracy

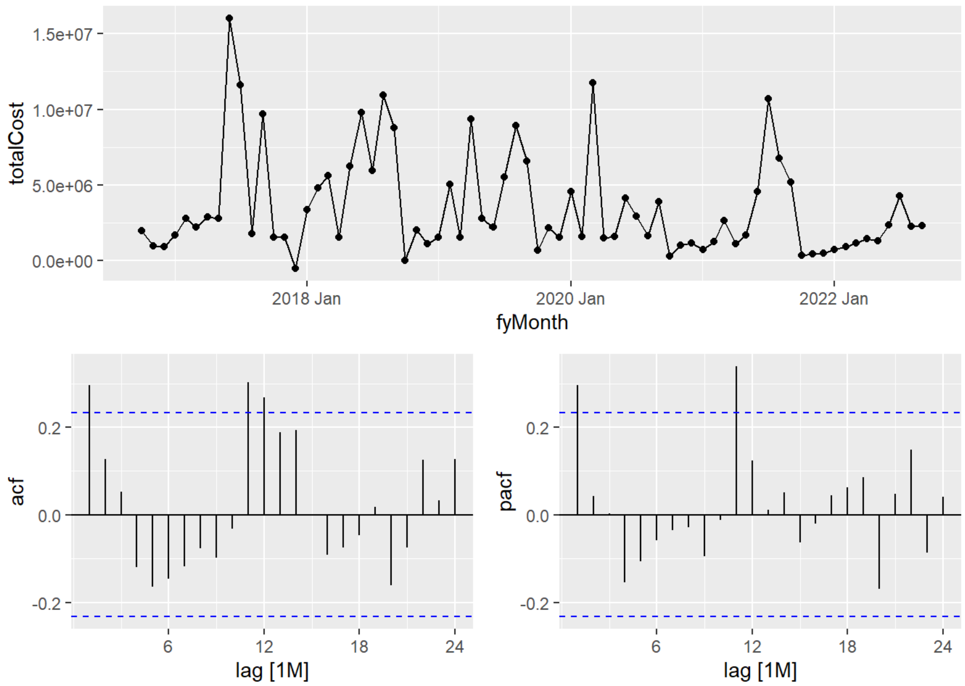
```
EOD_60SE_ExponentialFits <- EOD60_SE_Train %>%
  model(Add = ETS(totalCost ~ error("A") + trend("A") + season("A")),
        HWMult = ETS(totalCost ~ error("M") + trend("A") + season("M")),
        Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A")),
        HWMult_Damped = ETS(totalCost ~ error("M") + trend("Ad") + season("M")),
        ETSAuto_LIK = ETS(totalCost, opt_crit = "lik", ic = "aicc"),
        ETSAuto_AMSE = ETS(totalCost, opt_crit = "amse", ic = "aicc"),
        ETSAuto_MSE = ETS(totalCost, opt_crit = "mse", ic = "aicc"),
        ETSAuto_MAE = ETS(totalCost, opt_crit = "mae", ic = "aicc"))

forecastEOD_60SE <- EOD_60SE_ExponentialFits %>%
  forecast(h=24)

accuracy(forecastEOD_60SE,EOD60_SE) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 8 × 5
##   .model        .type      MAE  MAPE  MASE
##   <chr>         <chr>    <dbl> <dbl> <dbl>
## 1 HWMult        Test  1248080.  74.2 0.405
## 2 HWMult_Damped Test  1603638. 109.  0.521
## 3 Add           Test  2235040. 232.  0.726
## 4 Add_Damped    Test  2211214. 235.  0.718
## 5 ETSAuto_MAE   Test  2574530. 281.  0.836
## 6 ETSAuto_LIK   Test  2892127. 327.  0.939
## 7 ETSAuto_AMSE  Test  2892127. 327.  0.939
## 8 ETSAuto_MSE   Test  2892127. 327.  0.939
```

9. Residuals for Best BSO60 (S/E) Exponential Model



10. ACF/PACF Charts for BSO60 (S/E)



11. BSO60 (S/E) ARIMA Models and Accuracy

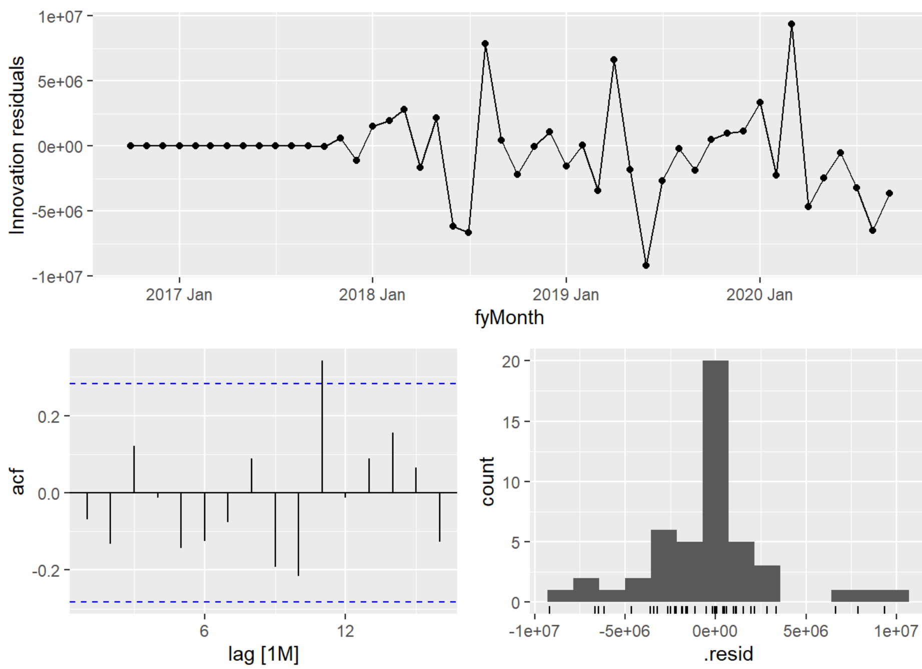
```
EOD60_SE_ARIMAFits <- EOD60_SE_Train %>%
  model(stepwiseARIMA = ARIMA(totalCost,ic = "aicc", stepwise = TRUE),
        autoARIMA = ARIMA(totalCost, ic = "aicc",stepwise = FALSE, approximation = FALSE,),
        ARIMA1 = ARIMA(totalCost ~ pdq(1,1,1) +PDQ(1,1,1)),
        ARIMA2 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,0,1)),
        ARIMA3 = ARIMA(totalCost ~ pdq(1,0,0) + PDQ(0,1,1)),
        ARIMA4 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(1,1,0)),
        ARIMA5 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(1,1,0)),
        ARIMA6 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)),
        ARIMA7 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA8 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA9 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,1,1)),
        ARIMA10 = ARIMA(totalCost ~ pdq(1,1,1)),
        ARIMA11 = ARIMA(totalCost ~ pdq(0,1,1)))

ARIMAforecastEOD60_SE <- EOD60_SE_ARIMAFits %>%
  forecast(h=24)

accuracy(ARIMAforecastEOD60_SE,EOD60_SE) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE) %>%
  head(5)
```

```
## # A tibble: 5 × 5
##   .model .type      MAE  MAPE  MASE
##   <chr>  <chr>    <dbl> <dbl> <dbl>
## 1 ARIMA1 Test  1670622.  95.3 0.542
## 2 ARIMA6 Test  1595294. 104.  0.518
## 3 ARIMA4 Test  1761455. 114.  0.572
## 4 ARIMA3 Test  2145640. 163.  0.697
## 5 ARIMA9 Test  2317075. 198.  0.752
```

12. Parameters for Best BSO60 (S/E) ARIMA Model

```
## Series: totalCost
## Model: ARIMA(0,1,1)(0,1,1)[12]
##
## Coefficients:
##           ma1     sma1
##       -0.9994  -0.3651
## s.e.   0.2929   0.2586
##
## sigma^2 estimated as 1.619e+13:  log likelihood=-583.77
## AIC=1173.54   AICc=1174.31   BIC=1178.2
```

13. Residuals for Best BSO60 (S/E) ARIMA Model



14. BSO60 (S/E) Dynamic Regression Models and Accuracy

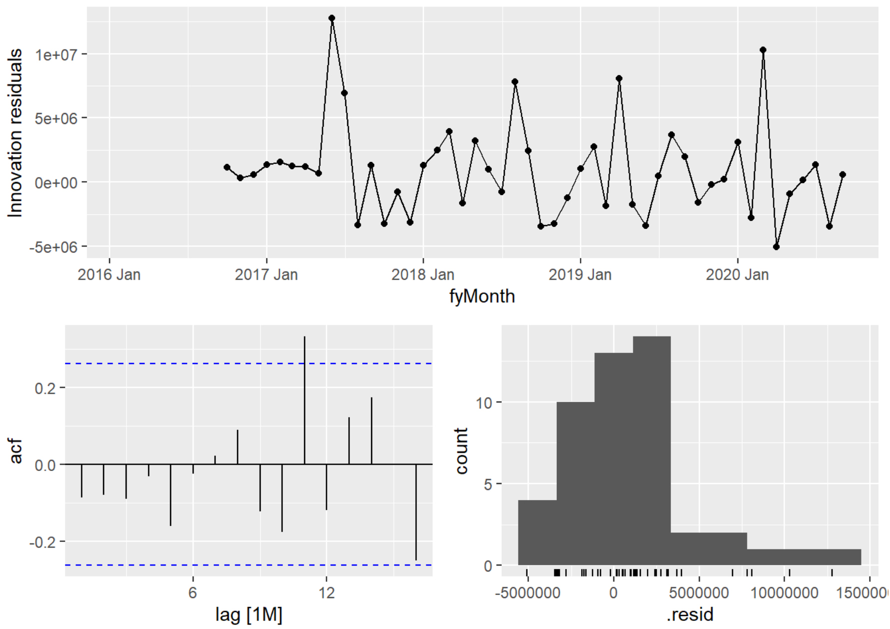
```
EOD60_SE_AdvancedARIMA_Fit <- EOD60_SE_Train_DR %>%
  model(advancedARIMA1 = ARIMA(totalCost ~ 0 + maintenence + readiness),
        advancedARIMA2 = ARIMA(totalCost ~ 0 + maintenence + preparation),
        advancedARIMA3 = ARIMA(totalCost ~ 0 + preparation + readiness),
        advancedARIMA_maint = ARIMA(totalCost ~ 0 + maintenence),
        advancedARIMA_read = ARIMA(totalCost ~ 0 + readiness),
        advancedARIMA_prep = ARIMA(totalCost ~ 0 + preparation),
        advancedARIMA1_lag = ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6)),
        advancedARIMA3_lag = ARIMA(totalCost ~ 0 + preparation + lag(readiness,6)),
        advancedARIMA_read_lag = ARIMA(totalCost ~ 0 + lag(readiness,6)))

EOD60_SE_AdvancedARIMA_Forecast <- EOD60_SE_AdvancedARIMA_Fit %>%
  forecast(new_data = EOD60_SE_Test_DR)

accuracy(EOD60_SE_AdvancedARIMA_Forecast,advancedARIMA_EOD60_SE) %>%
  select(.model,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 9 × 4
##   .model                     MAE  MAPE  MASE
##   <chr>                    <dbl> <dbl> <dbl>
## 1 advancedARIMA_read     1768410.  70.2 0.574
## 2 advancedARIMA3         1767217.  75.7 0.574
## 3 advancedARIMA3_lag     1807855.  79.3 0.587
## 4 advancedARIMA_read_lag 1724989.  87.7 0.560
## 5 advancedARIMA2         1903299.  97.6 0.618
## 6 advancedARIMA1_lag     1896616.  98.6 0.616
## 7 advancedARIMA_maint    1910553.  98.9 0.620
## 8 advancedARIMA1         1908744.  99.0 0.620
## 9 advancedARIMA_prep     2085062. 107.   0.677
```

15. Parameters of Best BSO60 (S/E) Dynamic Regression Model

```
## Series: totalCost
## Model: LM w/ ARIMA(3,0,0)(1,0,0)[12] errors
##
## Coefficients:
##          ar1     ar2     ar3    sar1  preparation  readiness
##       0.1341  0.1952  0.3284  0.4288    -74580.33   75774.16
## s.e.  0.1763  0.1427  0.1472  0.2131    119373.81  106276.27
##
## sigma^2 estimated as 1.32e+13:  log likelihood=-795.72
## AIC=1605.45   AICc=1607.78   BIC=1619.62
```

16. Residuals for Best BSO60 (S/E) Dynamic Regression Model

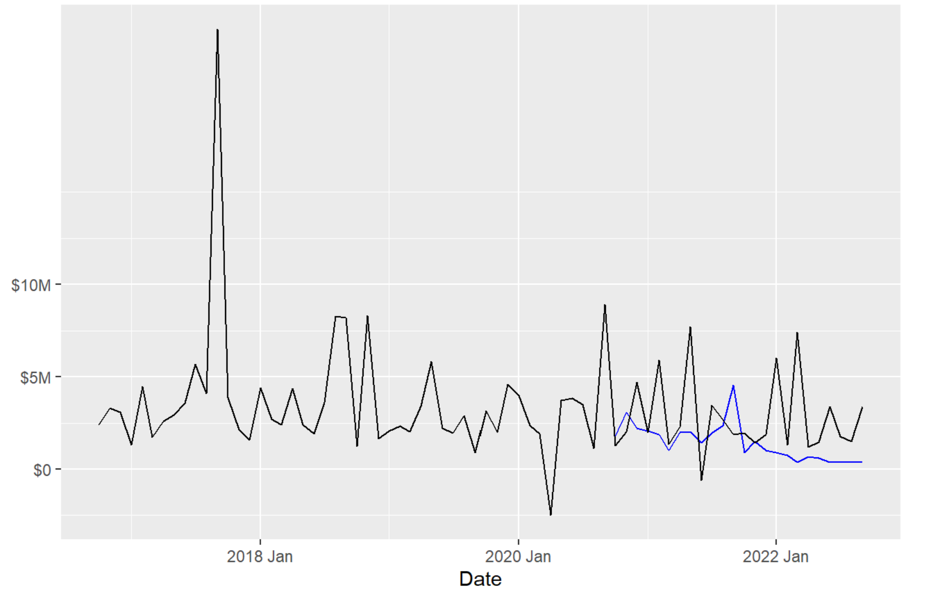


17. BSO70 (P/T) Exponential Model and Accuracy

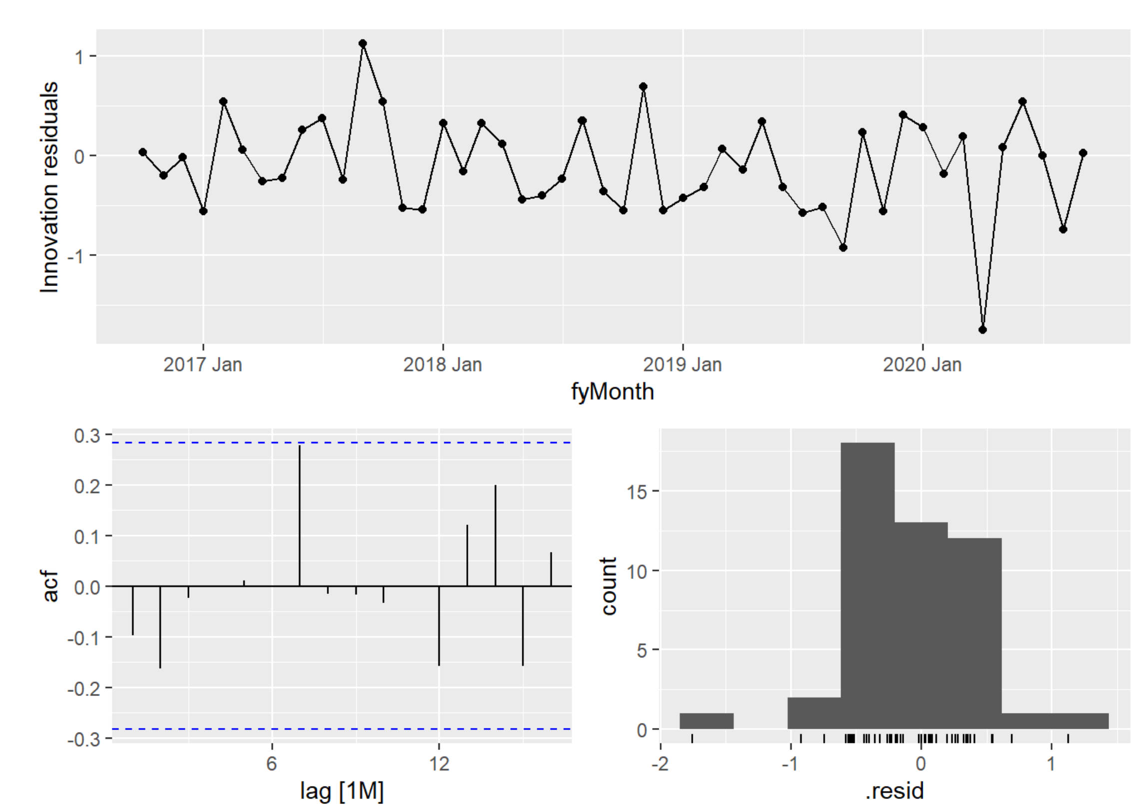
```
EOD_70PT_ExponentialFits <- EOD70_PT_Train %>%
  model(Add = ETS(totalCost ~ error("A") + trend("A") + season("A")),
        HWMult = ETS(totalCost ~ error("M") + trend("A") + season("M")),
        Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A")),
        HWMult_Damped = ETS(totalCost ~ error("M") + trend("Ad") + season("M")),
        ETSAuto_LIK = ETS(totalCost, opt_crit = "lik", ic = "aicc"),
        ETSAuto_AMSE = ETS(totalCost, opt_crit = "amse", ic = "aicc"),
        ETSAuto_MSE = ETS(totalCost, opt_crit = "mse", ic = "aicc"),
        ETSAuto_MAE = ETS(totalCost, opt_crit = "mae", ic = "aicc"))

forecastEOD_70PT <- EOD_70PT_ExponentialFits %>%
  forecast(h=24)

accuracy(forecastEOD_70PT,EOD70_PT) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 8 × 5
##   .model        .type      MAE  MAPE  MASE
##   <chr>         <chr>    <dbl> <dbl> <dbl>
## 1 HWMult        Test  1899789.  67.3 0.680
## 2 Add_Damped    Test  2167099.  77.6 0.775
## 3 HWMult_Damped Test  1983863.  85.7 0.710
## 4 ETSAuto_MAE   Test  1739338.  93.1 0.622
## 5 ETSAuto_MSE   Test  1777922.  97.4 0.636
## 6 ETSAuto_LIK   Test  1777922.  97.4 0.636
## 7 ETSAuto_AMSE  Test  1837039. 103.  0.657
## 8 Add           Test  2881601. 104.  1.03
```

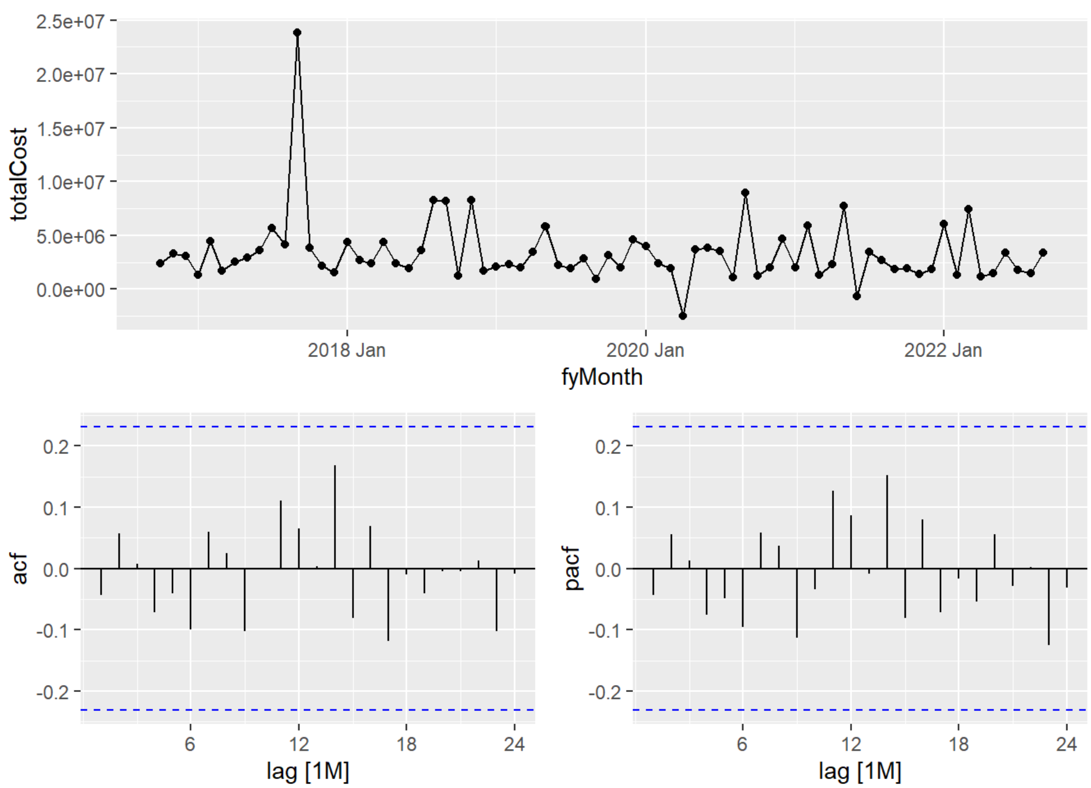
18. Graph of Best Exponential Model for BSO70 (P/T)



19. Residuals of Best BSO70 (P/T) Exponential Models



20. ACF/PACF Charts for BSO70 (P/T)

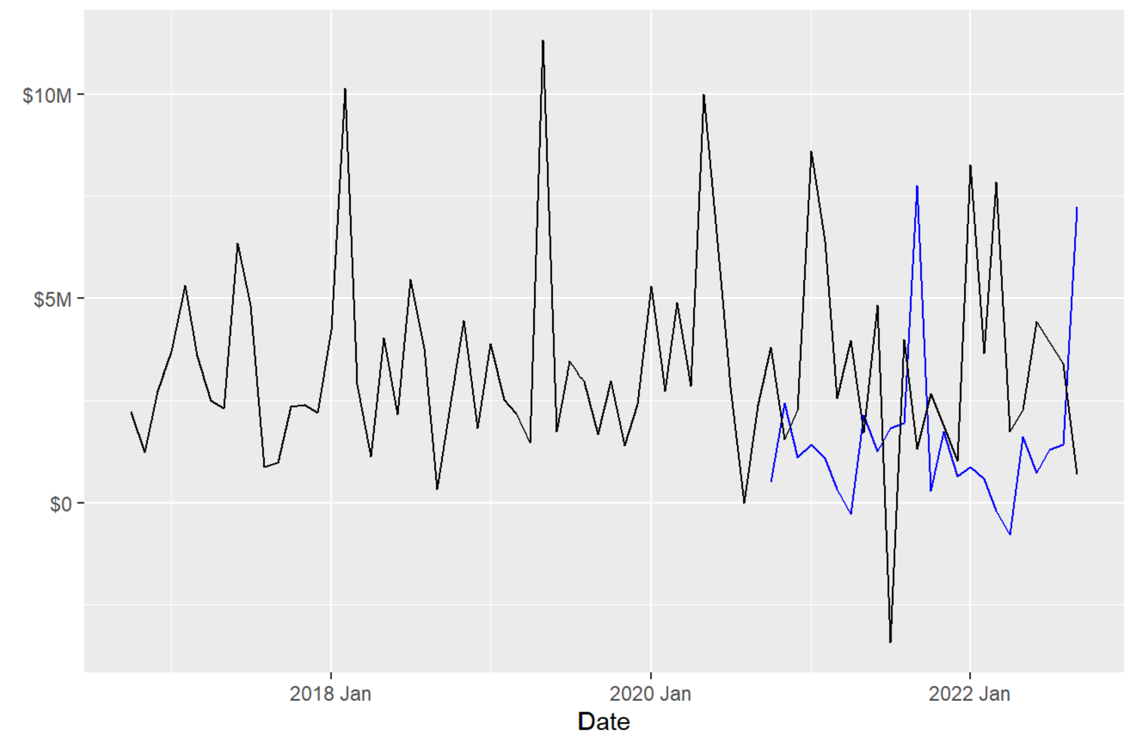


21. BSO70 (P/T) ARIMA Models and Their Accuracy

```
EOD70_PT_ARIMAFits <- EOD70_PT_Train %>%
  model(stepwiseARIMA = ARIMA(totalCost,ic = "aicc", stepwise = TRUE),
        autoARIMA = ARIMA(totalCost, ic = "aicc",stepwise = FALSE, approximation = FALSE,),
        ARIMA1 = ARIMA(totalCost ~ pdq(1,1,1) +PDQ(1,1,1)),
        ARIMA2 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,0,1)),
        ARIMA3 = ARIMA(totalCost ~ pdq(1,0,0) + PDQ(0,1,1)),
        ARIMA4 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(1,1,0)),
        ARIMA5 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(1,1,0)),
        ARIMA6 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)),
        ARIMA7 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA8 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA9 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,1,1)),
        ARIMA10 = ARIMA(totalCost ~ pdq(1,1,1)),
        ARIMA11 = ARIMA(totalCost ~ pdq(0,1,1)),
        ARIMA12 = ARIMA(totalCost ~ pdq(1,1,0)))

ARIMAforecastEOD70_PT <- EOD70_PT_ARIMAFits %>%
  forecast(h=24)

accuracy(ARIMAforecastEOD70_PT,EOD70_PT) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE) %>%
  head(5)
```

```
## # A tibble: 5 × 5
##   .model  .type      MAE  MAPE  MASE
##   <chr>   <chr>    <dbl> <dbl> <dbl>
## 1 ARIMA7  Test  2304971.  84.5 0.825
## 2 ARIMA8  Test  2304971.  84.5 0.825
## 3 ARIMA6  Test  2063416.  87.0 0.738
## 4 ARIMA10 Test  1879214.  98.5 0.672
## 5 ARIMA4  Test  2238587. 103.  0.801
```

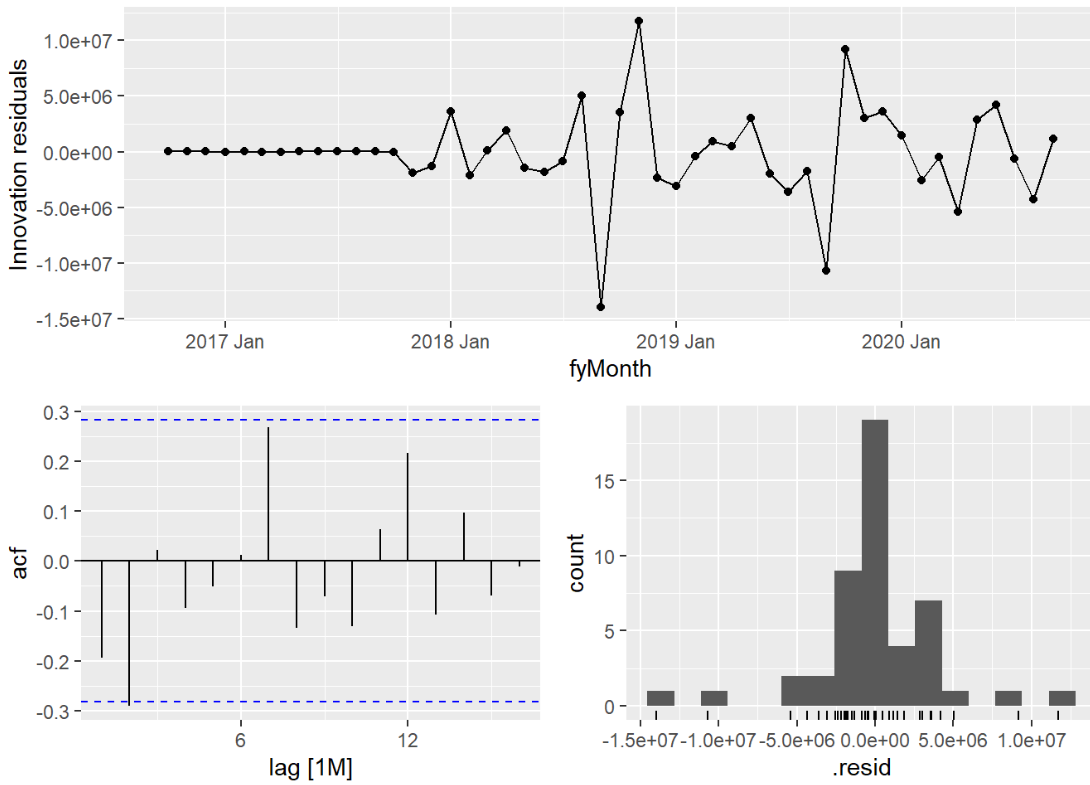
22. Graph of Best BSO70 (P/T) ARIMA Model



23. Parameters of Best BSO70 (P/T) ARIMA Model

```
## Series: totalCost
## Model: ARIMA(1,1,0)(0,1,1)[12]
##
## Coefficients:
##            ar1     sma1
##        -0.4850  -0.7020
## s.e.    0.1445   0.8476
```

24. Residuals of Best BSO70 (P/T) ARIMA Model



25. BSO70 (P/T) Dynamic Regression Models and Accuracy

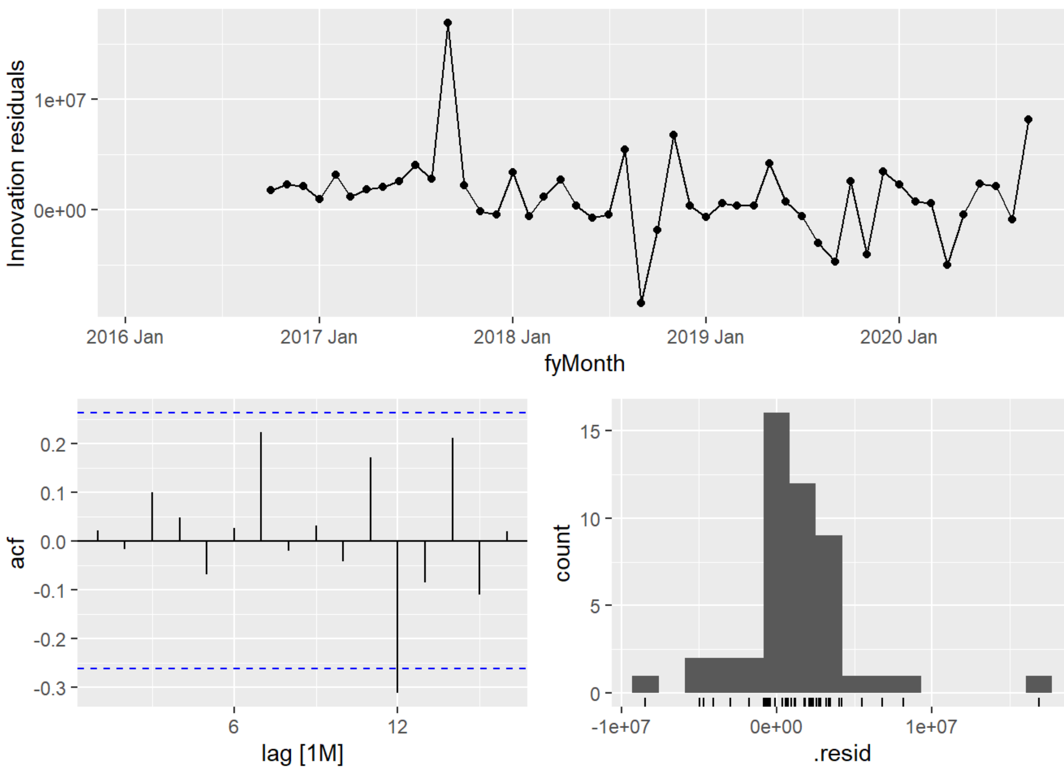
```
EOD70_PT_AdvancedARIMA_Fit <- EOD70_PT_Train_DR %>%
  model(advancedARIMA1 = ARIMA(totalCost ~ 0 + maintenence + readiness),
        advancedARIMA2 = ARIMA(totalCost ~ 0 + maintenence + preparation),
        advancedARIMA3 = ARIMA(totalCost ~ 0 + preparation + readiness),
        advancedARIMA_maint = ARIMA(totalCost ~ 0 + maintenence),
        advancedARIMA_read = ARIMA(totalCost ~ 0 + readiness),
        advancedARIMA_prep = ARIMA(totalCost ~ 0 + preparation),
        advancedARIMA1_lag = ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6)),
        advancedARIMA3_lag = ARIMA(totalCost ~ 0 + preparation + lag(readiness,6)),
        advancedARIMA_read_lag = ARIMA(totalCost ~ 0 + lag(readiness,6)))

EOD70_PT_AdvancedARIMA_Forecast <- EOD70_PT_AdvancedARIMA_Fit %>%
  forecast(new_data = EOD70_PT_Test_DR)

accuracy(EOD70_PT_AdvancedARIMA_Forecast,advancedARIMA_EOD70_PT) %>%
  select(.model,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 9 × 4
##   .model                     MAE  MAPE  MASE
##   <chr>                    <dbl> <dbl> <dbl>
## 1 advancedARIMA1_lag     1910743.  81.2 0.684
## 2 advancedARIMA_maint    1911712.  81.2 0.684
## 3 advancedARIMA_read_lag 1918032.  81.8 0.686
## 4 advancedARIMA2         1965337.  85.9 0.703
## 5 advancedARIMA_prep     1969308.  86.0 0.705
## 6 advancedARIMA3_lag     1968296.  86.3 0.704
## 7 advancedARIMA1         1985732.  87.5 0.711
## 8 advancedARIMA_read     1990856.  87.9 0.712
## 9 advancedARIMA3         2131504.  99.6 0.763
```

26. Parameters of Best BSO70 (P/T) Dynamic Regression Model

```
## Series: totalCost
## Model: LM w/ ARIMA(0,0,0)(1,0,0)[12] errors
##
## Coefficients:
##          sar1  maintenence  lag(readiness, 6)
##        0.7036     34985.42          -6530.948
## s.e.   0.1089    166743.48         154168.300
```

27. Residuals of Best BSO70 (P/T) Dynamic Regression Model

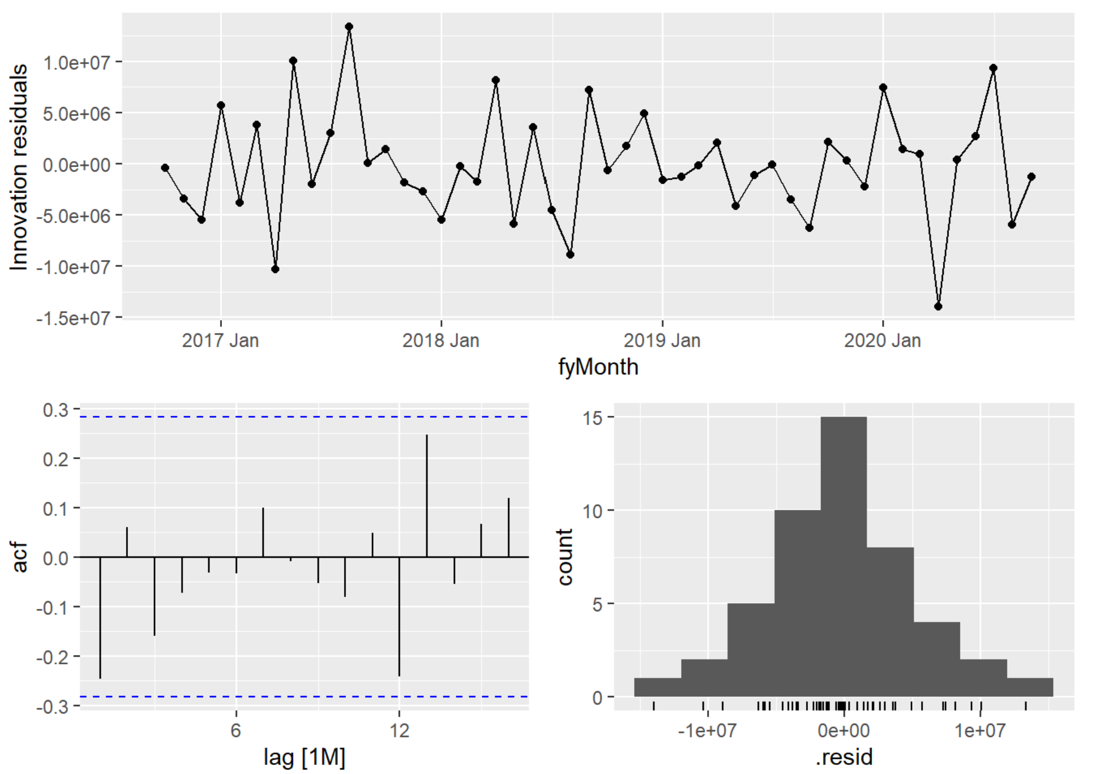


28. BSO70 (S/E) Exponential Models

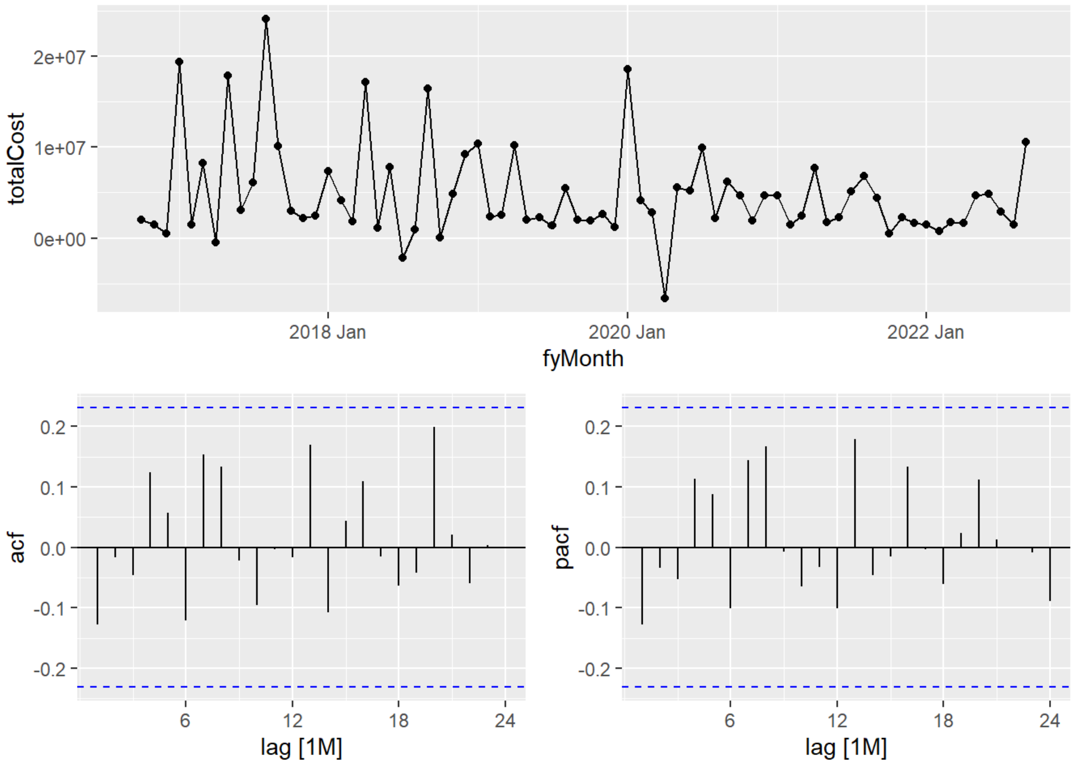
```
EOD_70SE_ExponentialFits <- EOD70_SE_Train %>%
  model(Add = ETS(totalCost ~ error("A") + trend("A") + season("A")),
        HWMult = ETS(totalCost ~ error("M") + trend("A") + season("M")),
        Add_Damped = ETS(totalCost ~ error("A") + trend("Ad") + season("A")),
        HWMult_Damped = ETS(totalCost ~ error("M") + trend("Ad") + season("M")),
        ETSAuto_LIK = ETS(totalCost, opt_crit = "lik", ic = "aicc"),
        ETSAuto_AMSE = ETS(totalCost, opt_crit = "amse", ic = "aicc"),
        ETSAuto_MSE = ETS(totalCost, opt_crit = "mse", ic = "aicc"),
        ETSAuto_MAE = ETS(totalCost, opt_crit = "mae", ic = "aicc"))

forecastEOD_70SE <- EOD_70SE_ExponentialFits %>%
  forecast(h=24)

accuracy(forecastEOD_70SE,EOD70_SE) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 8 × 5
##   .model        .type      MAE  MAPE  MASE
##   <chr>         <chr>    <dbl> <dbl> <dbl>
## 1 Add           Test  2713753.  117. 0.449
## 2 HWMult        Test  2333835.  117. 0.386
## 3 Add_Damped    Test  2605055.  119. 0.431
## 4 HWMult_Damped Test  2836411.  145. 0.470
## 5 ETSAuto_AMSE  Test  2820412.  177. 0.467
## 6 ETSAuto_LIK   Test  3130149.  196. 0.518
## 7 ETSAuto_MAE   Test  3130156.  196. 0.518
## 8 ETSAuto_MSE   Test  3130156.  196. 0.518
```

29. Residuals for Best BSO70(S/E) Exponential Model



30. ACF/PACF Charts for BSO70 (S/E)



31. BSO70 (S/E) ARIMA Models

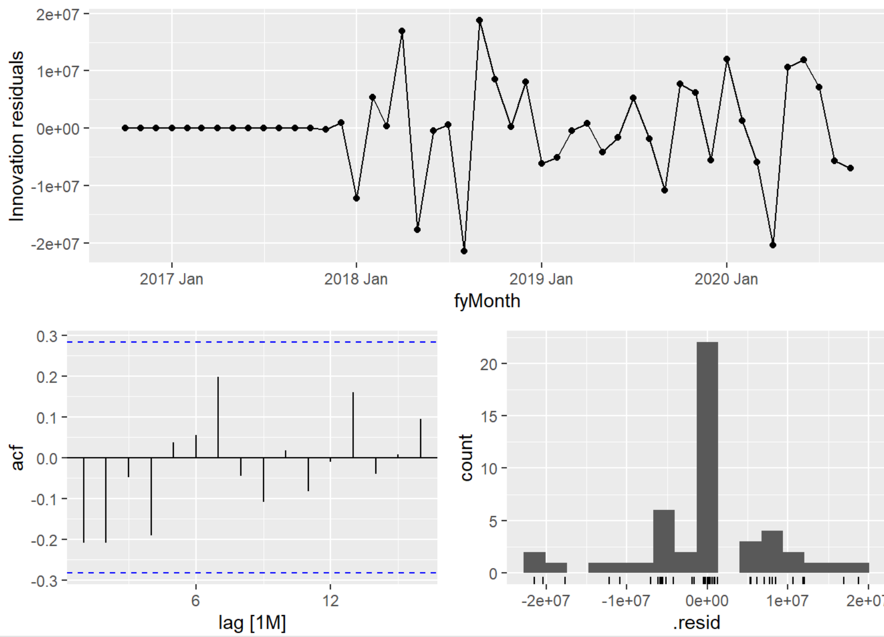
```
EOD70_SE_ARIMAFits <- EOD70_SE_Train %>%
  model(stepwiseARIMA = ARIMA(totalCost,ic = "aicc", stepwise = TRUE),
        autoARIMA = ARIMA(totalCost, ic = "aicc",stepwise = FALSE, approximation = FALSE,),
        ARIMA1 = ARIMA(totalCost ~ pdq(1,1,1) +PDQ(1,1,1)),
        ARIMA2 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,0,1)),
        ARIMA3 = ARIMA(totalCost ~ pdq(1,0,0) + PDQ(0,1,1)),
        ARIMA4 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(1,1,0)),
        ARIMA5 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(1,1,0)),
        ARIMA6 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)),
        ARIMA7 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA8 = ARIMA(totalCost ~ pdq(1,1,0) + PDQ(0,1,1)),
        ARIMA9 = ARIMA(totalCost ~ pdq(1,0,1) + PDQ(0,1,1)),
        ARIMA10 = ARIMA(totalCost ~ pdq(1,1,1)),
        ARIMA11 = ARIMA(totalCost ~ pdq(0,1,1)))

ARIMAforecastEOD70_SE <- EOD70_SE_ARIMAFits %>%
  forecast(h=24)

accuracy(ARIMAforecastEOD70_SE,EOD70_SE) %>%
  select(.model,.type,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 13 × 5
##    .model        .type      MAE  MAPE   MASE
##    <chr>         <chr>    <dbl> <dbl>  <dbl>
##  1 ARIMA5        Test  3179989.  119.  0.527
##  2 ARIMA4        Test  3304076.  123.  0.547
##  3 ARIMA6        Test  3029841.  127.  0.502
##  4 ARIMA1        Test  3218317.  134.  0.533
##  5 ARIMA9        Test  2829920.  139.  0.469
##  6 ARIMA3        Test  2804719.  139.  0.464
##  7 ARIMA10       Test  2521171.  160.  0.417
##  8 autoARIMA     Test  2763455.  174.  0.458
##  9 stepwiseARIMA Test  2763455.  174.  0.458
## 10 ARIMA2        Test  2797522.  176.  0.463
## 11 ARIMA7        Test  4505393.  198.  0.746
## 12 ARIMA8        Test  4505393.  198.  0.746
```

32. Parameters for Best BSO70 (S/E) ARIMA Model

```
## Series: totalCost
## Model: ARIMA(1,1,0)(1,1,0)[12]
##
## Coefficients:
##           ar1      sar1
##       -0.6393   -0.3749
## s.e.   0.1247    0.2063
```

33. Residuals for Best BSO70 (S/E) ARIMA Model

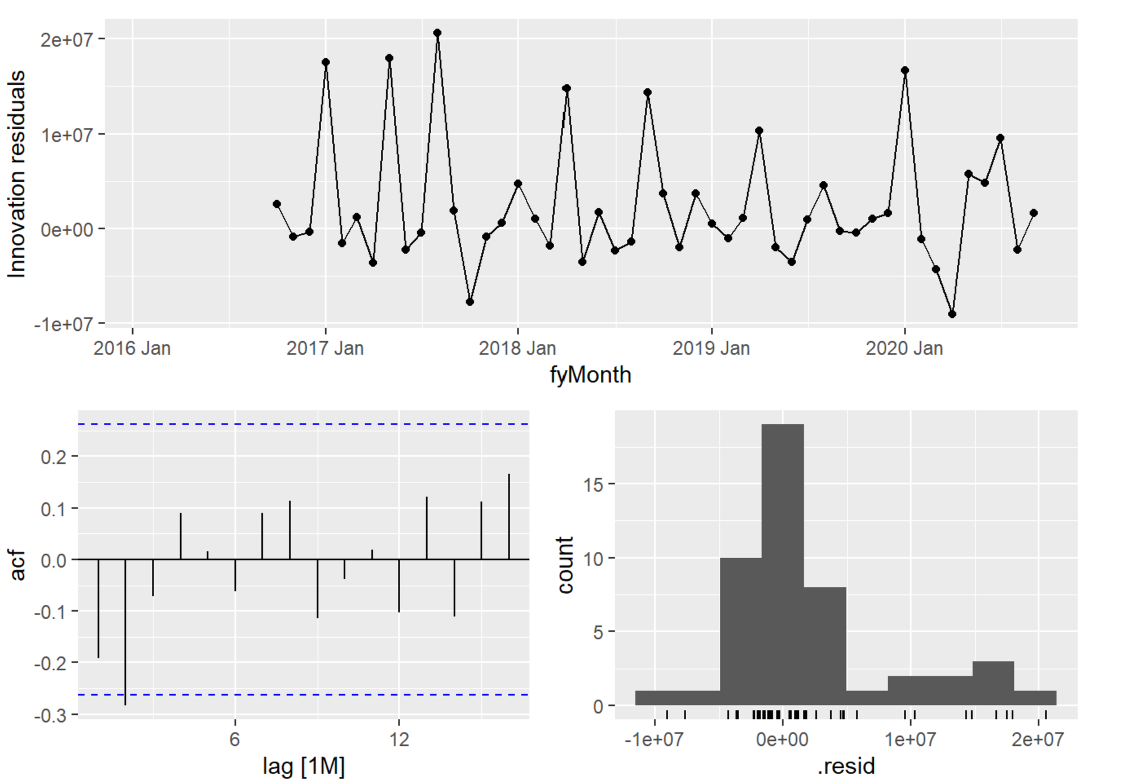


34. BSO70 (S/E) Dynamic Regression Models

```
EOD70_SE_AdvancedARIMA_Fit <- EOD70_SE_Train_DR %>%
  model(advancedARIMA1 = ARIMA(totalCost ~ 0 + maintenence + readiness),
        advancedARIMA2 = ARIMA(totalCost ~ 0 + maintenence + preparation),
        advancedARIMA3 = ARIMA(totalCost ~ 0 + preparation + readiness),
        advancedARIMA_maint = ARIMA(totalCost ~ 0 + maintenence),
        advancedARIMA_read = ARIMA(totalCost ~ 0 + readiness),
        advancedARIMA_prep = ARIMA(totalCost ~ 0 + preparation),
        advancedARIMA1_lag = ARIMA(totalCost ~ 0 + maintenence + lag(readiness,6)),
        advancedARIMA3_lag = ARIMA(totalCost ~ 0 + preparation + lag(readiness,6)),
        advancedARIMA_read_lag = ARIMA(totalCost ~ 0 + lag(readiness,6)),
        advancedARIMA_misc1 = ARIMA(totalCost ~  pdq(1,1,0) + readiness),
        advancedARIMA_misc1 = ARIMA(totalCost ~  pdq(1,1,0) + maintenence))

EOD70_SE_AdvancedARIMA_Forecast <- EOD70_SE_AdvancedARIMA_Fit %>%
  forecast(new_data = EOD70_SE_Test_DR)

accuracy(EOD70_SE_AdvancedARIMA_Forecast,advancedARIMA_EOD70_SE) %>%
  select(.model,MAE,MAPE,MASE) %>%
  arrange(MAPE)
```

```
## # A tibble: 10 × 4
##    .model                     MAE  MAPE  MASE
##    <chr>                    <dbl> <dbl> <dbl>
##  1 advancedARIMA1         3118256.  85.8 0.516
##  2 advancedARIMA_maint    3032589.  88.0 0.502
##  3 advancedARIMA2         3048438.  89.3 0.505
##  4 advancedARIMA_prep     3122652.  90.3 0.517
##  5 advancedARIMA_read_lag 3127309.  90.4 0.518
##  6 advancedARIMA1_lag     3090660.  90.7 0.512
##  7 advancedARIMA3_lag     3140094.  91.7 0.520
##  8 advancedARIMA_read     3203076.  91.9 0.530
##  9 advancedARIMA3         3214866.  92.2 0.532
## 10 advancedARIMA_misc1    2273215. 141.   0.376
```

35. Residuals for Best BSO70 (S/E) Dynamic Regression Model



36. Hierarchical Exponential Models

```
ActualCosts_Hiearcichal_Fit <- ActualCosts_Hiearcichal_Train %>%
  model(base = ETS(totalCost)) %>%
  reconcile(autoBu = bottom_up(base),
            autoTopDown_FProps = top_down(base,method = "forecast_proportions"),
            autoTopDown_AProps = top_down(base,method = "average_proportions"),
            autoOLS = min_trace(base, method = "ols"),
            autoMint = min_trace(base,method = "mint_shrink"))
```

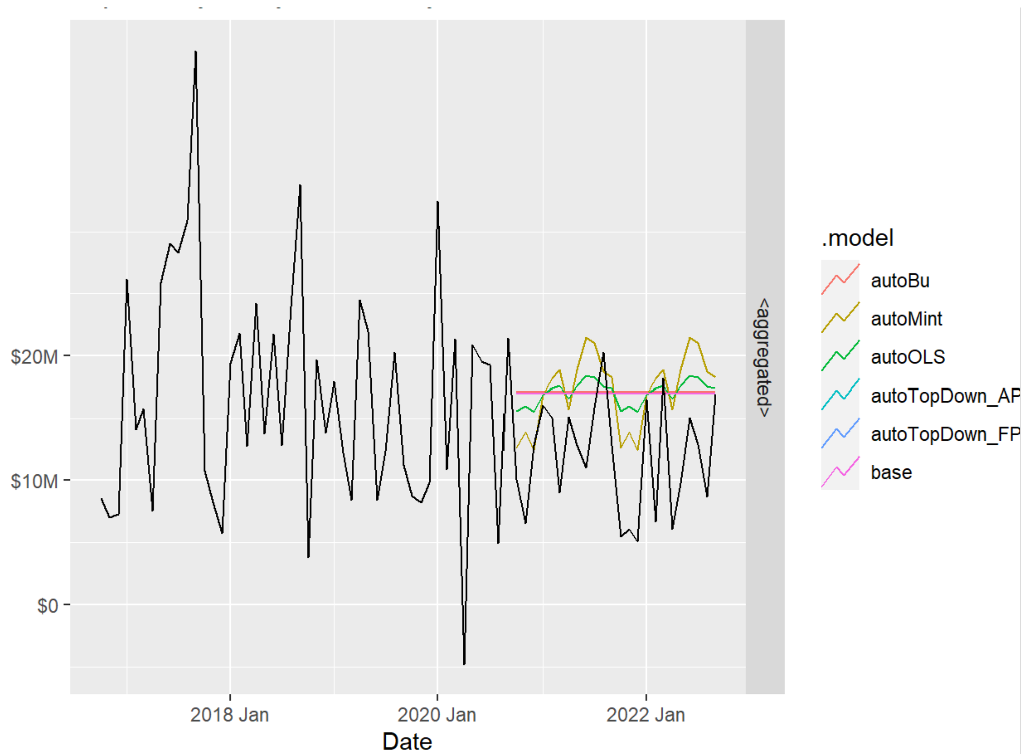
37. Accuracy of Aggregated, Hierarchical Exponential Models

```
## # A tibble: 6 × 7
##   .model             BSO          PILLAR       .type      mae  mase  mape
##   <chr>              <chr*>       <chr*>       <chr>    <dbl> <dbl> <dbl>
## 1 autoMint           <aggregated> <aggregated> Test  5541778. 0.654  65.4
## 2 autoOLS            <aggregated> <aggregated> Test  5463207. 0.644  68.9
## 3 autoTopDown_AProps <aggregated> <aggregated> Test  5468214. 0.645  70.7
## 4 autoTopDown_FProps <aggregated> <aggregated> Test  5468214. 0.645  70.7
## 5 base               <aggregated> <aggregated> Test  5468214. 0.645  70.7
## 6 autoBu             <aggregated> <aggregated> Test  5596913. 0.660  72.2
```

38. Average Accuracy of Disaggregated, Hierarchical Exponential Models

```
## # A tibble: 6 × 4
##   .model                 mae  mase  mape
##   <chr>                <dbl> <dbl> <dbl>
## 1 autoMint           2334155. 0.763  147.
## 2 autoTopDown_FProps 2418099. 0.784  158.
## 3 autoOLS            2380058. 0.780  162.
## 4 autoTopDown_AProps 2366962. 0.801  167.
## 5 autoBu             2439243. 0.803  175.
## 6 base               2439243. 0.803  175.
```

39. Graph of Aggregated, Exponential Hierarchical Models



40. Hierarchical ARIMA Models

```
ActualCosts_Hiearcichal_Fit <- ActualCosts_Hiearcichal_Train %>%
  model(ARIMA1 = ARIMA(totalCost ~ pdq(0,1,1) + PDQ(0,1,1)),
        ARIMA2 = ARIMA(totalCost)) %>%
  reconcile(ARIMA1_bu = bottom_up(ARIMA1),
            ARIMA1_topDown_FProps = top_down(ARIMA1,method = "forecast_proportions"),
            ARIMA1_topDown_AProps = top_down(ARIMA1,method = "average_proportions"),
            ARIMA1_ols = min_trace(ARIMA1, method = "ols"),
            ARIMA1_mint = min_trace(ARIMA1,method = "mint_shrink"),
            ARIMA2_bu = bottom_up(ARIMA2),
            ARIMA2_topDown_FProps = top_down(ARIMA2,method = "forecast_proportions"),
            ARIMA2_topDown_AProps = top_down(ARIMA2,method = "average_proportions"),
            ARIMA2_ols = min_trace(ARIMA2, method = "ols"),
            ARIMA2_mint = min_trace(ARIMA2,method = "mint_shrink"))

ActualCosts_Hiearcichal_Forcast <- ActualCosts_Hiearcichal_Fit %>%
  forecast(h = 24)
```

41. Accuracy of Aggregated, Hierarchical ARIMA Models

```
## # A tibble: 12 × 7
##    .model                BSO          PILLAR       .type      mae  mase  mape
##    <chr>                 <chr*>       <chr*>       <chr>    <dbl> <dbl> <dbl>
##  1 ARIMA1                <aggregated> <aggregated> Test  3948069. 0.466  33.0
##  2 ARIMA1_topDown_AProps <aggregated> <aggregated> Test  3948069. 0.466  33.0
##  3 ARIMA1_topDown_FProps <aggregated> <aggregated> Test  3948069. 0.466  33.0
##  4 ARIMA1_ols            <aggregated> <aggregated> Test  4016441. 0.474  33.3
##  5 ARIMA1_mint           <aggregated> <aggregated> Test  4180360. 0.493  34.8
##  6 ARIMA1_bu             <aggregated> <aggregated> Test  4344676. 0.512  36.3
##  7 ARIMA2_bu             <aggregated> <aggregated> Test  5475270. 0.646  68.0
##  8 ARIMA2_mint           <aggregated> <aggregated> Test  5910688. 0.697  71.6
##  9 ARIMA2_ols            <aggregated> <aggregated> Test  5949786. 0.702  72.1
## 10 ARIMA2                <aggregated> <aggregated> Test  6365799. 0.751  74.7
## 11 ARIMA2_topDown_AProps <aggregated> <aggregated> Test  6365799. 0.751  74.7
## 12 ARIMA2_topDown_FProps <aggregated> <aggregated> Test  6365799. 0.751  74.7
```

42. Average Accuracy of Disaggregated, Hierarchical ARIMA Models

```
## # A tibble: 5 × 4
##   .model                     mae  mase  mape
##   <chr>                    <dbl> <dbl> <dbl>
## 1 ARIMA1_topDown_AProps 1849393. 0.677  83.2
## 2 ARIMA1                2235215. 0.762 102.
## 3 ARIMA1_bu             2235215. 0.762 102.
## 4 ARIMA1_ols            2233427. 0.763 102.
## 5 ARIMA1_mint           2237251. 0.762 103.
```