Assignment 3

Team 23: Hetian Chen, Teun Zwier, Spyros Avlonitis, Dragos Pop

30 November 2021

Exercise 1

1.1) Here we have a seasonal $ARIMA(0,1,1)(2,1,0)_{12}$ model. We extract the d,q for the non-seasonal part by looking into the $(1-B)$ and $(1+\theta_1 B)$ parts of the equation. Also we can see that the autogression part is missing and thus p is 0. For the seasonal part we can see that the period is 12 and the d is 1 by looking into $(1-B^{12})$. Also, p is 2 as described in $(1-\phi_1 B^{12}-\phi_2 B^{24})$, and q is 0 as the MA part is missing.

1.2) First we apply the difference to the model equation:

$$(1-B)(1-B^{12})y_t^* = b_1^*(1-B)(1-B^{12})x_{1,t}^* + b_2^*(1-B)(1-B^{12})x_{2,t}^* + \eta_t$$

Then converting the backshift operator:

$$\begin{aligned}(y_t^* - y_{t-1}^* - y_{t-12}^* + y_{t-13}^*) = \\ b_1(x_{1,t}^* - x_{1,t-1}^* - x_{1,t-12}^* + x_{1,t-13}^*) + b_2(x_{2,t}^* - x_{2,t-1}^* - x_{2,t-12}^* + x_{2,t-13}^*) + (1-B)(1-B^{12})\eta_t\end{aligned}$$

Next, we apply the autoregression $(1-\Phi_1 B^{12}-\Phi_2 B^{24})$

$$\begin{aligned}(1-\Phi_1 B^{12}-\Phi_2 B^{24})(y_t^* - y_{t-1}^* - y_{t-12}^* + y_{t-13}^*) = \\ b_1(1-\Phi_1 B^{12}-\Phi_2 B^{24})(x_{1,t}^* - x_{1,t-1}^* - x_{1,t-12}^* + x_{1,t-13}^*)+ \\ b_2(1-\Phi_1 B^{12}-\Phi_2 B^{24})(x_{2,t}^* - x_{2,t-1}^* - x_{2,t-12}^* + x_{2,t-13}^*)+ \\ (1-\Phi_1 B^{12}-\Phi_2 B^{24})(1-B)(1-B^{12})\eta_t = \\ (1-\Phi_1 B^{12}-\Phi_2 B^{24})(b_1(x_{1,t}^* - x_{1,t-1}^* - x_{1,t-12}^* + x_{1,t-13}^*)+ \\ b_2(x_{2,t}^* - x_{2,t-1}^* - x_{2,t-12}^* + x_{2,t-13}^*))+ \\ (1+\theta_1 B)\epsilon_t = \\ (1-\Phi_1 B^{12}-\Phi_2 B^{24})(b_1(x_{1,t}^* - x_{1,t-1}^* - x_{1,t-12}^* + x_{1,t-13}^*)+ \\ b_2(x_{2,t}^* - x_{2,t-1}^* - x_{2,t-12}^* + x_{2,t-13}^*))+ \\ (\epsilon_t + \theta_1\epsilon_{t-1})\end{aligned}$$

Above we should convert the backshift operator as we did before, however we will consider it a trivial task and skip it to save some space. Next we should keep y_t^* on the one LHS and move everything to the RHS.

$$\begin{aligned}y_t^* = (\Phi_1 B^{12}-\Phi_2 B^{24})y_t^* \\ +(1-\Phi_1 B^{12}-\Phi_2 B^{24})((y_{t-1}^* + y_{t-12}^* - y_{t-13}^*) \\ +b_1(x_{1,t}^* - x_{1,t-1}^* - x_{1,t-12}^* + x_{1,t-13}^*) + b_2(x_{2,t}^* - x_{2,t-1}^* - x_{2,t-12}^* + x_{2,t-13}^*)) + (\epsilon_t + \theta_1\epsilon_{t-1})\end{aligned}$$

1.3) Then, in order to forecast electricity demand for the next 12 months, we should first rewrite the equation by replacing t with $T+h$. Then,on the right hand side of the equation, we should replace future observations with their forecasts, future errors with zero, and past errors with the corresponding residuals. Finally, we can teratively solve it for h=1,2,...,11,12 and get the forecast for the next 12 months.

Exercise 2.1)

First, get sample from dataset and find most optimal lambda.

```
set.seed(11345678)
myseries <- aus_retail %>% filter(`Series ID` == sample(aus_retail$`Series ID`,1))
myseries %>% features(Turnover, guerrero)
```

```
## # A tibble: 1 x 3
##   State    Industry                                    lambda_guerrero
##   <chr>    <chr>                                                 <dbl>
## 1 Victoria Electrical and electronic goods retailing            -0.0918
```

-0.092 was found to be the most optimal lambda value for this specific seed, so close to a logarithmic transformation. Now, try model for order k = 0 - 6.

```
fit_box <- myseries %>%
  model (K_1 = ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 1) + PDQ(0 ,0 ,0)) ,
         K_2 = ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 2) + PDQ(0 ,0 ,0)) ,
         K_3 = ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 3) + PDQ(0 ,0 ,0)) ,
         K_4 = ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 4) + PDQ(0 ,0 ,0)) ,
         K_5 = ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 5) + PDQ(0 ,0 ,0)) ,
         K_6 = ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 6) + PDQ(0 ,0 ,0)))
glance(fit_box) %>% select(.model, sigma2, log_lik, AIC, AICc, BIC)
```

```
## # A tibble: 6 x 6
##   .model  sigma2 log_lik    AIC   AICc    BIC
##   <chr>    <dbl>   <dbl>  <dbl>  <dbl>  <dbl>
## 1 K_1    0.00596    505.  -993.  -992.  -956.
## 2 K_2    0.00358    618. -1214. -1214. -1169.
## 3 K_3    0.00237    710. -1394. -1393. -1341.
## 4 K_4    0.00155    804. -1577. -1575. -1511.
## 5 K_5    0.00146    818. -1608. -1607. -1551.
## 6 K_6    0.00117    869. -1705. -1704. -1640.
```

Looking at the table, the AICc is lowest for K = 6, so this model is chosen. For k = 6, the automatically chosen ARIMA-model has parameters (2,1,1)

Exercise 2.2)

Looking at the residuals in graph 2.1, there is no clear pattern in residuals in top graph. The residuals seem to be following a normal distribution, being roughly bell shaped with tails of equal length. There seem to be some significant autocorrelations. To check if the residuals resemble white noise, we can use the Ljung-box test.

```
fit_box %>% select(K_6) %>% augment() %>%
  features(.innov, ljung_box, dof = 16, lag = 24)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 K_6       63.9  7.96e-11
```

The Ljung Box test is significant. This result, combined with the ACF plot, suggests that the residuals do not resemble white noise.

Exercise 2.3)

To compare the models, we forecast the last 2 years of the dataset.

```
train <- myseries %>% filter(Month <= yearmonth( "2016 dec" ))
fit_compare <- train %>%
  model(dyn_regr= ARIMA(box_cox(Turnover, -0.0918) ~ fourier(K = 6) + PDQ(0 ,0 ,0)),
        ets = ETS(box_cox(Turnover, -0.0918)),
        arima = ARIMA(box_cox(Turnover, -0.0918)))
fc_compare <- fit_compare %>% forecast(h=24)
fc_compare %>% accuracy(myseries)
```

```
## # A tibble: 3 x 12
##   .model State Industry .type     ME  RMSE   MAE    MPE  MAPE  MASE RMSSE   ACF1
##   <chr>  <chr> <chr>    <chr>  <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 arima  Vict~ Electri~ Test   -2.56  22.3  16.7 -0.549  3.33 0.768 0.796 0.114
## 2 dyn_r~ Vict~ Electri~ Test  -28.4   41.1  32.8 -6.02   6.91 1.51  1.47  0.0957
## 3 ets    Vict~ Electri~ Test  -23.1   37.4  26.2 -4.77   5.42 1.20  1.33  0.360
```

Looking at the accuracy of the fits, the ARIMA model does the best for this train/test split for all error measures. All the forecasts seem to be reasonably close to the real data, looking at the error measures and the graphs 2.2, 2.3 and 2.4. The ARIMA model with seasonality may show better results here because of change in seasonality, which the fourier terms do not account for.

Exercise 3.1)

First we get the training set and fit harmonic regression with trend models with multiple values for parameter K. And then we rank the models according to their AICc values:

```
train_set <- us_gasoline %>% filter_index(~ '2004 W52')
train_fit <- train_set %>%
  model(
    'K_5' = ARIMA(Barrels ~ trend() + fourier(K = 5)),
    'K_6' = ARIMA(Barrels ~ trend() + fourier(K = 6)),
    'K_7' = ARIMA(Barrels ~ trend() + fourier(K = 7)),
    'K_8' = ARIMA(Barrels ~ trend() + fourier(K = 8)),
    'K_9' = ARIMA(Barrels ~ trend() + fourier(K = 9)))
train_fit %>% glance() %>% arrange(AICc)
```

```
## # A tibble: 5 x 8
##   .model sigma2 log_lik   AIC  AICc   BIC ar_roots  ma_roots
##   <chr>   <dbl>   <dbl> <dbl> <dbl> <dbl> <list>    <list>
## 1 K_7    0.0717   -64.7  165.  166.  248. <cpl [0]> <cpl [1]>
## 2 K_8    0.0717   -63.7  167.  169.  259. <cpl [0]> <cpl [1]>
## 3 K_9    0.0717   -62.7  169.  171.  270. <cpl [0]> <cpl [1]>
## 4 K_6    0.0724   -69.5  171.  172.  244. <cpl [0]> <cpl [1]>
## 5 K_5    0.0747   -81.5  193.  194.  262. <cpl [0]> <cpl [53]>
```

As we can see, the harmonic regression model with K = 7 has the lowest AICc value. Thus we pick this model.

Now we plot the fitted values against the observed values of the training set in graph **3.1** in appendix.

As we can see in the plot, the harmonic regression model captures the trend quite well. The fitted values are also quite close to the observed values in seasonal pattern, but the observed magnitude of the observed seasonality is a bit bigger than the fitted values.

In graph **3.2** in appendix, we could see the residuals of the harmonic regression model are close to white noise, although there appears to be some upward trend in the ACF plot.

Now we check the residuals of the harmonic regression model with the Ljung-Box test:

```
dof_1 <- train_fit %>% select('K_7') %>% tidy() %>% nrow()
train_fit %>% select('K_7') %>% augment() %>%
  features(.innov, ljung_box, dof = dof_1, lag = 52)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 K_7       55.0    0.0168
```

The p-value for Ljung-Box test is 0.02. Thus, at significance level of 95%, we can reject the null hypothesis and claim that the residuals are not white noise. However, at significance level of 99%, we can consider the residuals as white noise and calculate forecasts and prediction intervals.

Exercise 3.2)

Now we forecast the next year (2005) with data generated from the *fourier()* function. And the we plot the forecasts agains the observed values:

```
fc_1 <- train_fit %>% select('K_7') %>%
  forecast(xreg = fourier(train_set, K = 7, h = 52)) %>%
  filter(year(Week) == 2005)
```

The forecasts could be seen in graph **3.3** in appendix. The forecasts capture the trend quite well, except for the drop in 2005. The forecasts are quite smooth in seasonality compared to the observed values.

Now we fit a harmonic regression with a piecewise linear time trend to the full gasoline series. First we need to find the knots. In graph **3.4** in appendix, we could see two knots: the first being around 2006-2007 with the second one around 2012-2014.

Now we fit the models and we rank the models according to their AICc values. Note that as we have already learnt that K = 7 is optimal in part(1), we will continue to use K = 7 for fitting:

```
hybrid_fit <- us_gasoline %>%
  model(
    pw_1 = TSLM(Barrels ~ trend(knots = c(2006, 2012)) + fourier(K = 7)),
    pw_2 = TSLM(Barrels ~ trend(knots = c(2006, 2013)) + fourier(K = 7)),
    pw_3 = TSLM(Barrels ~ trend(knots = c(2006, 2014)) + fourier(K = 7)),
    pw_4 = TSLM(Barrels ~ trend(knots = c(2007, 2012)) + fourier(K = 7)),
    pw_5 = TSLM(Barrels ~ trend(knots = c(2007, 2013)) + fourier(K = 7)),
    pw_6 = TSLM(Barrels ~ trend(knots = c(2007, 2014)) + fourier(K = 7)))
hybrid_fit %>% glance() %>% arrange(AICc) %>%
  select(-sigma2, -statistic, -p_value, -deviance, -df.residual, -rank)
```

```
## # A tibble: 6 x 9
##   .model r_squared adj_r_squared    df log_lik    AIC   AICc    BIC     CV
##   <chr>      <dbl>         <dbl> <int>   <dbl>  <dbl>  <dbl>  <dbl>  <dbl>
## 1 pw_1       0.859         0.857    18   -172. -3463. -3462. -3364. 0.0776
## 2 pw_3       0.859         0.857    18   -172. -3463. -3462. -3364. 0.0776
## 3 pw_2       0.859         0.857    18   -172. -3463. -3462. -3364. 0.0776
## 4 pw_6       0.858         0.857    18   -173. -3461. -3460. -3362. 0.0777
## 5 pw_4       0.858         0.857    18   -173. -3461. -3460. -3362. 0.0777
## 6 pw_5       0.858         0.857    18   -173. -3461. -3460. -3361. 0.0777
```

In the metrics table, we could find that *pw_1*, which has 2006 and 2012 as two knots, has the lowest AICc value. Thus this is the best model for harmonic regression with a piecewise linear time trend.

Exercise 3.3)

Now we use *ARIMA()* function instead of *TSLM()* to allow for correlated errors, with the same predictor variables in part(2).
We start from the simple model with *PDQ(0, 0, 0)* and check the residuals of the model with Ljung-Box test:

```
new_fit_0 <- us_gasoline %>%
  model('model' = ARIMA(Barrels ~ trend(knots = c(2006, 2012)) +
                          fourier(K = 7) + PDQ(0, 0, 0)))
dof_2 <- new_fit_0 %>% tidy() %>% nrow()
new_fit_0 %>% augment() %>% features(.innov, ljung_box, dof = dof_2, lag = 52)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 model     65.9  0.000388
```

The residuals are plotted in graph **3.5** in appendix, which look close to white noise. However, the p-value of the Ljung-Box test is model, 65.89578580848, 0.000387974539491798, which rejects the null hypothesis that the residuals are white noise, at significance lever 99%.

We try to modify our model. First we plot the ACF and PACF of first order difference of the original series in graph **3.6** in appendix. There is 1 significant lag in ACF plot and 3 significant lags in PACF plot. So we consider adding *pdq(3, 1, 1)* to the ARIMA model. Also we add MA(1) to the seasonal ARIMA.
Eventually come up with these parameters:

```
new_fit <- us_gasoline %>%
  model(
    'model' = ARIMA(Barrels ~ trend(knots = c(2006, 2012)) +
                      fourier(K = 7) + pdq(3, 1, 1) + PDQ(0, 0, 1)))
dof_3 <- new_fit %>% tidy() %>% nrow()
new_fit %>% augment() %>% features(.innov, ljung_box, dof = dof_3, lag = 52)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 model     50.3    0.0116
```

The residuals are plotted in graph **3.7** in appendix, which look even closer to white noise. The p-value of the Ljung-Box test is 0.012. So at significance level 99%, the null hypothesis is not rejected and we can consider the residuals as white noise.

Now we use this model to forecast the next year (52 weeks):

```
new_fc <- new_fit %>%
  forecast(xreg = fourier(us_gasoline, K = c(7), h = 52)) %>%
  filter_index(. ~ '2018 W03')
```

As we can see in graph **3.8** in appendix, the model performs well in forecasting the seasonality but not too well with the trend.

Exercise 4.1)

After the series were parsed, they were plotted. Since the variation does not increase over time in any of the time series, no transformation was applied.

```
my_cols <- c('NN3_101','NN3_102','NN3_103','NN3_104','NN3_105', 'NN3_106','NN3_107','NN3_108','NN3_10
my_data <- read_excel("/Users/dragos/Desktop/Applied Forecasting in Complex Systems/Assignments/Assig
data1 = my_data[1] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data2 = my_data[2] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data3 = my_data[3] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data4 = my_data[4] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data5 = my_data[5] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data6 = my_data[6] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data7 = my_data[7] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data8 = my_data[8] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data9 = my_data[9] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data10 = my_data[10] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
data11 = my_data[11] %>%  mutate(i = 1:n()) %>% as_tsibble(index = i)
```

Exercise 4.2)

First, a new column representing the month was added as index for each time series and then the data was sliced in the train and test sets. Next, the six models, namely Naive, Mean, Simple Exponential Smoothing, Damped Exponential Smoothing, Seasonal Exponential Smoothing, and ARIMA were fitted to the train data. Lastly, the models were used to generate forecasts for the next 18 months, which were later evaluated in raport with the true values from the test set. Accordingly, one can see in the tables below the metrics for each time series.

```
for(j in 1:ncol(my_data))
{
  this_ts <- my_data[j][!is.na(my_data[j]), ] %>%
    mutate(i = seq(as.Date('1979-01-01'), by = 'months', length = n())) %>% mutate(i = yearmonth(i))
    as_tsibble(index = i)

  this_name = colnames(this_ts)[1]
  colnames(this_ts)[1] <- 'NN'
  colnames(this_ts)[2] <- 'Date'

  this_train_set <- this_ts[1:(nrow(this_ts)-18), ]
```

```
  this_test_set <- this_ts[(nrow(this_ts)-17):nrow(this_ts), ]

  this_fit <- this_train_set %>%
    model(
      Mean = MEAN(NN),
      Naive = NAIVE(NN),
      Simple_ETS = ETS(NN ~ error("A") + trend("N") + season("N")),
      Damped_ETS = ETS(NN ~ error('A') + trend('Ad') + season('N')),
      Season_ETS = ETS(NN ~ error('A') + trend('A') + season('A')),
      ARIMA = ARIMA(NN))

  m1 <- this_fit %>% forecast(this_test_set) %>% accuracy(this_ts) %>% select(.model, RMSE, MAE, MAPE

  this_fc <- this_fit %>% forecast(this_test_set)
  this_fc_2 <- this_fc %>% as_tsibble() %>% select(.model, Date, .mean) %>% pivot_wider(names_from =
  this_fc_3 <- bind_cols(this_fc_2, this_test_set['NN'])
  colnames(this_fc_3)[8] <- 'True_Value'
  this_fc_4 <- this_fc_3 %>% pivot_longer(c(Mean, Naive, Simple_ETS, Damped_ETS, Season_ETS, ARIMA),
  this_fc_4 <- this_fc_4 %>% mutate(E = 200*abs(NN-True_Value)/(NN+True_Value))
  m2 <- this_fc_4 %>% as_tibble() %>% group_by(.model) %>% summarise(sMAPE = mean(E))
  m <- bind_cols(m1, m2[2])
  m <- m %>% mutate(MSE = RMSE ^ 2)
  print(this_name);print(m)
}
```

```
## [1] "NN3_101"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE     MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 ARIMA       151. 121.   2.28  2.31  22780.
## 2 Damped_ETS  224. 179.   3.34  3.41  50129.
## 3 Mean        336. 292.   5.45  5.65 112906.
## 4 Naive       243. 196.   3.64  3.74  59088.
## 5 Season_ETS  110.  86.3  1.63  1.64  12011.
## 6 Simple_ETS  228. 181.   3.38  3.46  52023.
## [1] "NN3_102"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE      MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 ARIMA      1064. 1026.  25.1  21.0 1131113.
## 2 Damped_ETS 2462. 2189.  44.7  43.6 6063355.
## 3 Mean       2122. 1882.  48.6  37.1 4501752.
## 4 Naive      2484. 2203.  44.6  43.9 6171828.
## 5 Season_ETS 1558. 1480.  36.4  28.1 2427178.
## 6 Simple_ETS 2463. 2190.  44.7  43.6 6067879.
## [1] "NN3_103"
## # A tibble: 6 x 6
##   .model       RMSE    MAE  MAPE sMAPE        MSE
##   <chr>       <dbl>  <dbl> <dbl> <dbl>      <dbl>
## 1 ARIMA       6867.  3840.  30.1  24.5  47150651.
## 2 Damped_ETS 22255. 14154.  77.5  87.3 495268413.
## 3 Mean       18734. 15206. 232.  100.  350968233.
```

```
## 4 Naive      22253. 14154.  77.5  87.3 495214338.
## 5 Season_ETS  5048.  3370.  31.6  29.5  25481642.
## 6 Simple_ETS 22253. 14154.  77.5  87.3 495215835.
## [1] "NN3_104"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE      MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 ARIMA       607.  346.  5.37  5.44  368533.
## 2 Damped_ETS 2281. 1663. 47.7  29.9  5203135.
## 3 Mean       1822. 1547. 35.9  30.0  3318760.
## 4 Naive      2281. 1663. 47.6  29.9  5200868.
## 5 Season_ETS  629.  455.  8.22  7.96  395932.
## 6 Simple_ETS 2281. 1663. 47.6  29.9  5200950.
## [1] "NN3_105"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE    MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 ARIMA       165.  151.  3.31  3.27 27224.
## 2 Damped_ETS  180.  154.  3.34  3.33 32436.
## 3 Mean        210.  153.  3.23  3.32 44281.
## 4 Naive       177.  146.  3.15  3.16 31168.
## 5 Season_ETS  140.  111.  2.38  2.40 19488.
## 6 Simple_ETS  180.  156.  3.38  3.37 32547.
## [1] "NN3_106"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE     MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 ARIMA       260.  229.  4.57  4.56  67739.
## 2 Damped_ETS  336.  289.  5.58  5.75 112799.
## 3 Mean        279.  228.  4.59  4.52  77639.
## 4 Naive       279.  228.  4.58  4.52  77778.
## 5 Season_ETS  336.  250.  4.86  5.08 113083.
## 6 Simple_ETS  335.  288.  5.57  5.74 112483.
## [1] "NN3_107"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE    MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 ARIMA       179.  144.  4.07  3.98 32048.
## 2 Damped_ETS  260.  220.  6.27  6.03 67484.
## 3 Mean        275.  237.  6.74  6.46 75388.
## 4 Naive       260.  221.  6.28  6.03 67629.
## 5 Season_ETS  279.  245.  6.97  6.66 77604.
## 6 Simple_ETS  260.  221.  6.28  6.03 67627.
## [1] "NN3_108"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE      MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 ARIMA      1294. 1098.  27.0  29.7 1675204.
## 2 Damped_ETS 1352. 1146.  27.8  31.2 1827256.
## 3 Mean       1291. 1092.  26.9  29.5 1667877.
## 4 Naive      1139.  938.  24.5  25.0 1296326.
## 5 Season_ETS 1592. 1362.  31.7  38.7 2534566.
## 6 Simple_ETS 1358. 1151.  27.9  31.4 1843354.
```

```
## [1] "NN3_109"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE     MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 ARIMA       288.  247.  7.78  8.09  82893.
## 2 Damped_ETS  378.  289. 10.2   9.38 143189.
## 3 Mean        688.  624. 21.4  18.8  473744.
## 4 Naive       406.  312. 11.0  10.0  165017.
## 5 Season_ETS  257.  220.  7.19  7.22  66251.
## 6 Simple_ETS  397.  303. 10.7   9.79 157689.
## [1] "NN3_110"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE     MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>   <dbl>
## 1 ARIMA       921.  779.  44.3  33.6 848138.
## 2 Damped_ETS  899.  746.  40.6  32.5 808370.
## 3 Mean        922.  779.  44.4  33.6 849276.
## 4 Naive       917.  774.  43.8  33.5 840099.
## 5 Season_ETS  873.  680.  30.8  28.9 762094.
## 6 Simple_ETS  924.  781.  44.6  33.7 853203.
## [1] "NN3_111"
## # A tibble: 6 x 6
##   .model      RMSE   MAE  MAPE sMAPE      MSE
##   <chr>      <dbl> <dbl> <dbl> <dbl>    <dbl>
## 1 ARIMA       649.  531.  17.8  16.4  420806.
## 2 Damped_ETS  838.  690.  22.2  21.1  702283.
## 3 Mean        931.  837.  28.6  25.3  866055.
## 4 Naive      1013.  671.  17.3  20.5 1025470.
## 5 Season_ETS  455.  336.  11.0  10.4  206846.
## 6 Simple_ETS  854.  719.  23.6  21.9  729577.
```

Exercise 4.3)

For the first time series, namely NN3_101, all the metrics are pointing to the Seasonal Exponential Smoothing model as the best models, registering the smallest values in comparison with the other models, followed by the ARIMA model.

Regarding the second series, the table shows the ARIMA model outperforms by far the other five models.

According to the metrics in the third table, one can see that again the seasonal SES has the lowest values for all the evaluation metrics, being followed by the ARIMA This means that, as in the first time series, the Seasonal Exponential Smoothing model makes the most accurate forecast, as expected considering the strong seasonality of the data as seen in figure **4.3**.

When it comes to the fourth time series, the ARIMA is closely followed by the Seasonal ETS, with the other four models being far behind.

For NN3_105, Seasonal Exponential Smoothing seems to provide very accurate predictions in comparison with the other models. The best model in case of the sixth series is ARIMA model, followed unexpectedly by the simple methods Mean and Naive.

For the seventh time series, ARIMA makes the most accurate predictions by far, as opposed to NN3_108, where the Naive model outperforms the rest, followed by the simple Mean model.

From the ninth table, one can notice that Seasonal ETS has the lowest values for all the metrics, meaning that its predictions score the best against the true values from the test set.

Next, the tenth time series is best predicted by the Damped Exponential Smoothing model, which seems to learn from the slightly decreasing trend of the data.

Finally, the last table shows the lowest errors for the Seasonal Exponential Smoothing model.

Appendix

Graphs for Exercise 2

```
fit_box %>% select(K_6) %>% gg_tsresiduals() + labs(title = "Graph 2.1")
```

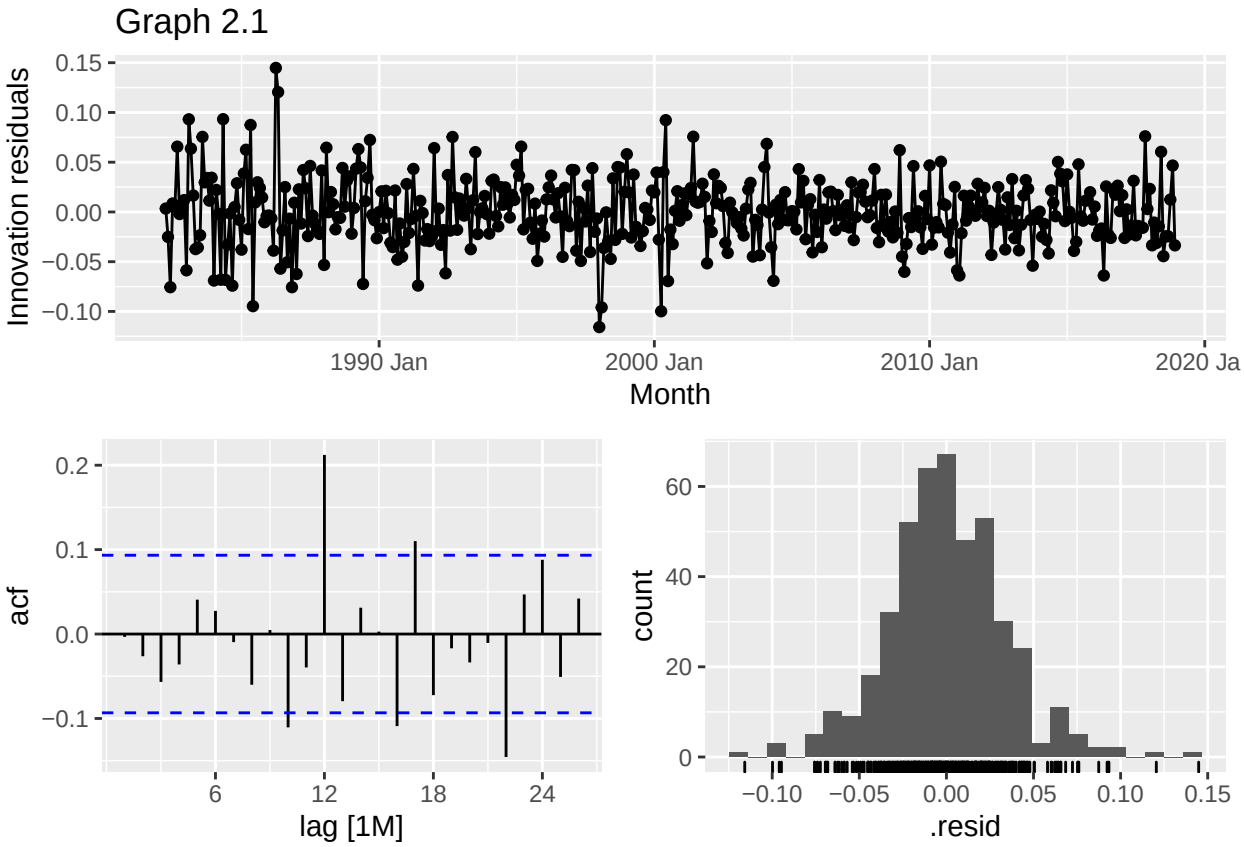

Figure 1: Figure 2.1

```
test_compare <- myseries %>% filter(Month > yearmonth( "2016 dec" ))
fc_compare %>% filter(.model == "dyn_regr") %>%
  autoplot(test_compare)+ labs(title="2.2 - Dynamic regression forecast")
```

```
fc_compare %>% filter(.model == "arima") %>%
  autoplot(test_compare)+ labs(title="2.3 - Arima forecast")
```

```
fc_compare %>% filter(.model == "ets") %>%
  autoplot(test_compare)+ labs(title="2.4 - ETS forecast")
```

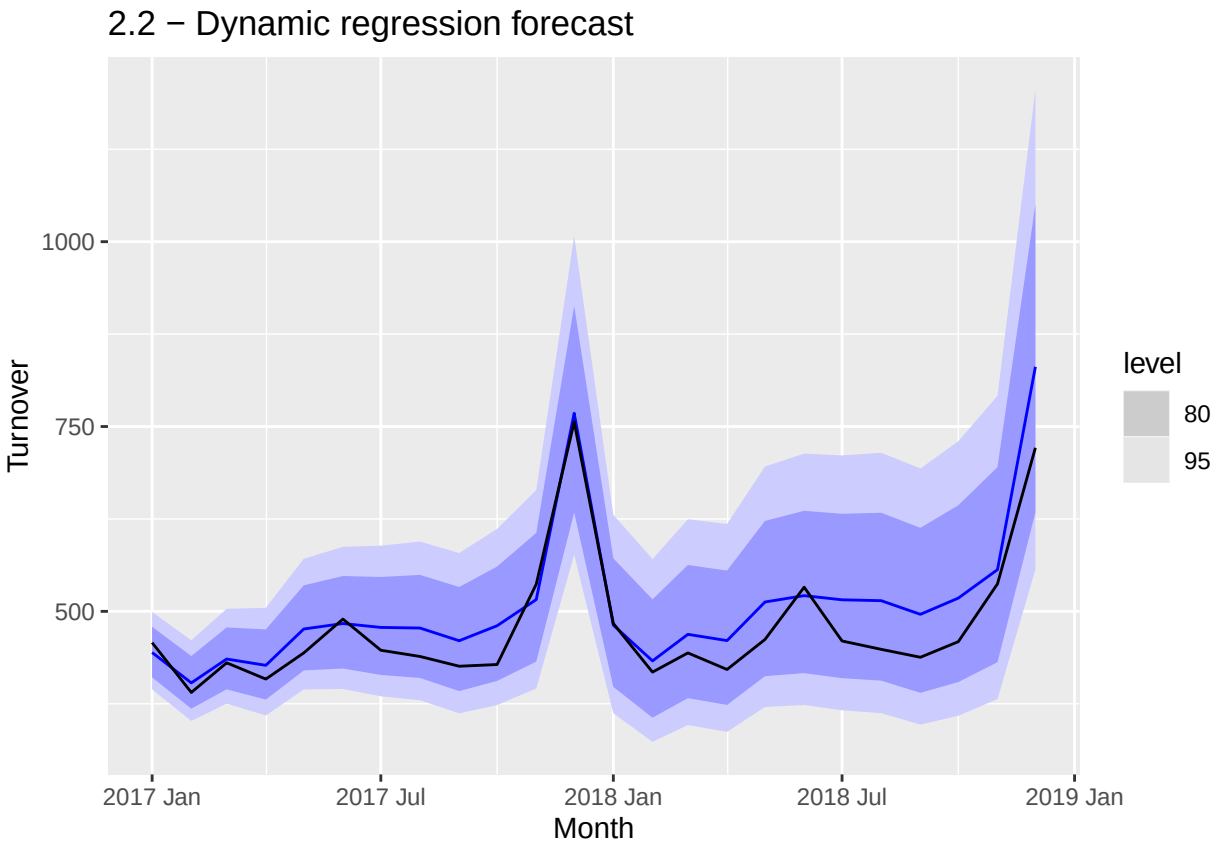

Figure 2: Figure 2.2

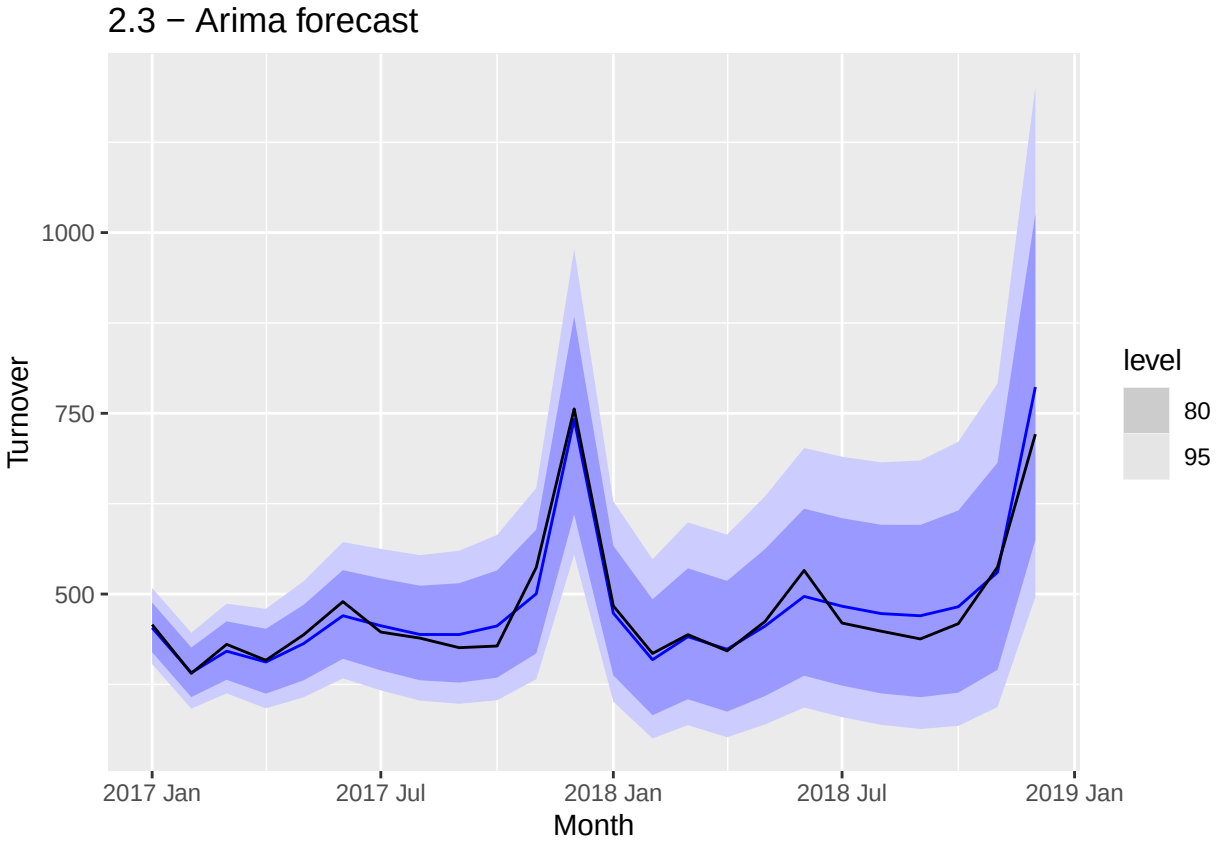

Figure 3: Figure 2.3

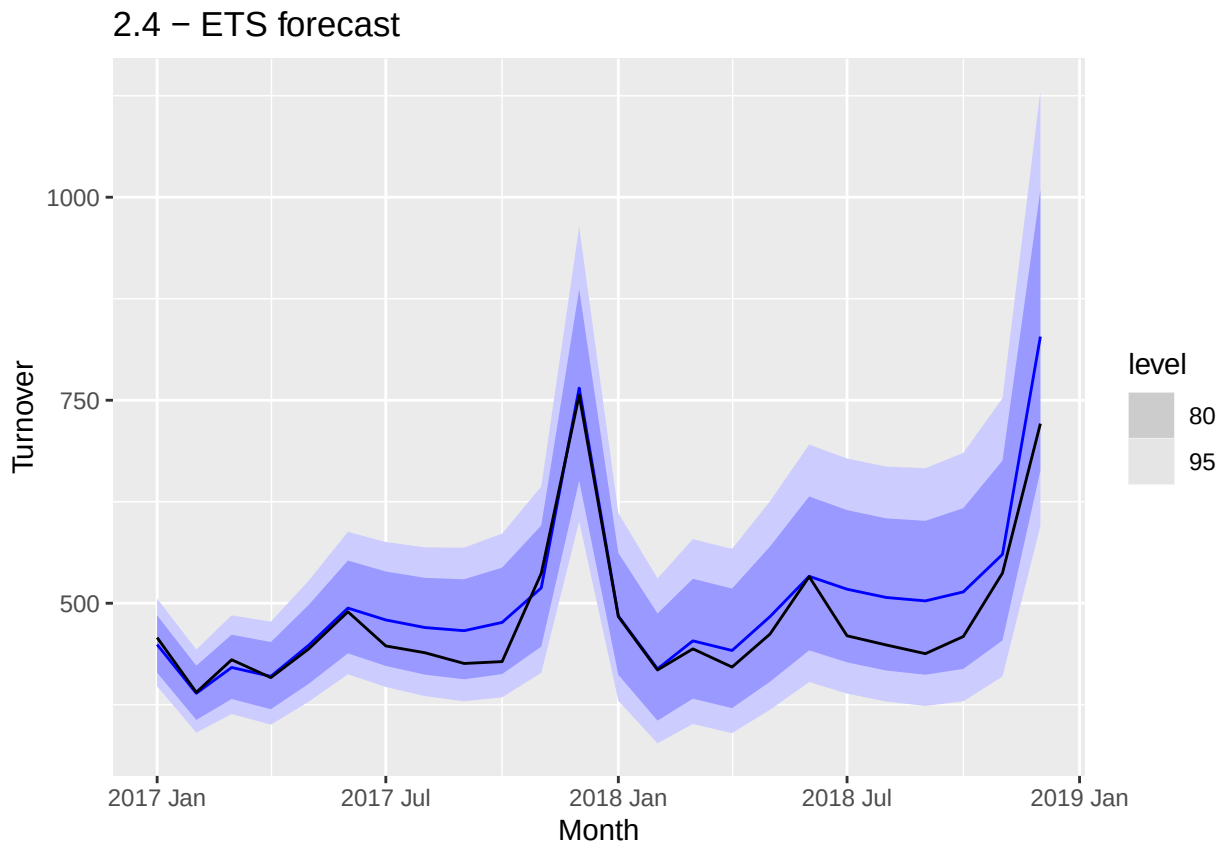

Figure 4: Figure 2.4

Graphs for Exercise 3

Graph 3.1 - Observed values vs. fitted values of harmonic regression model with trend:

```
train_fit %>% select('K_7') %>% augment() %>% rename(Observed = Barrels, Fitted = .fitted) %>%
  pivot_longer(Observed: Fitted, names_to = 'Type', values_to = 'Barrels') %>%
  autoplot(Barrels) +
  labs(title = '3.1 - Harmonic Regression Model - Fitted vs. Observed') +
  ylab('Million Barrels')
```

Graph 3.2 - Residuals of harmonic regression model with K = 7

```
train_fit %>% select('K_7') %>% gg_tsresiduals() +
  labs(title = '3.2 - Residuals of Harmonic Regression Model')
```

Graph 3.3 - Forecasts and observed values of 2005

```
test_set <- us_gasoline %>% filter(year(Week) == 2005)
test_set %>% autoplot(Barrels) + geom_line(data = fc_1, aes(y = .mean)) +
  autolayer(fc_1, alpha = 0.4) +
  labs(title = '3.3 - Forecast vs. Observed Values of 2005') +
  ylab('Million Barrels')
```

Graph 3.4 - US Gasoline Weekly Supplies

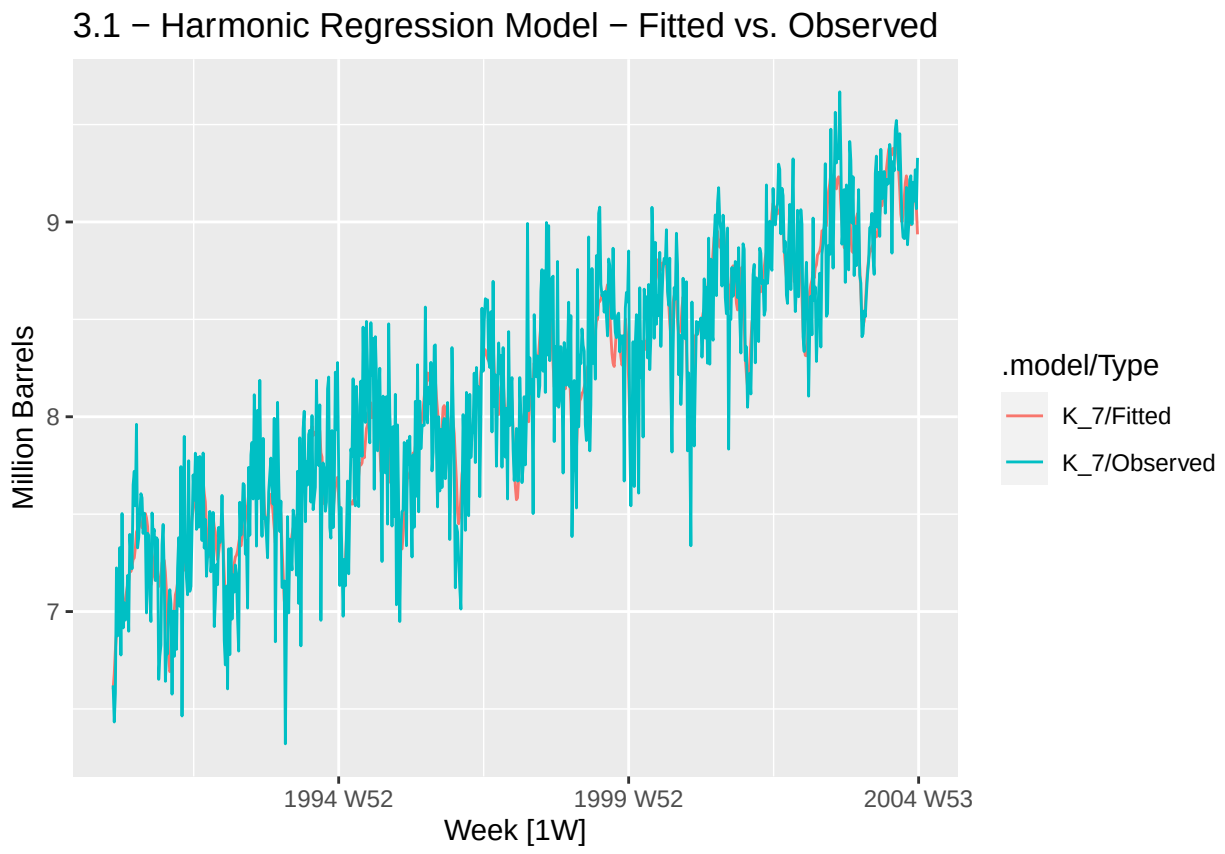

Figure 5: Figure 3.1

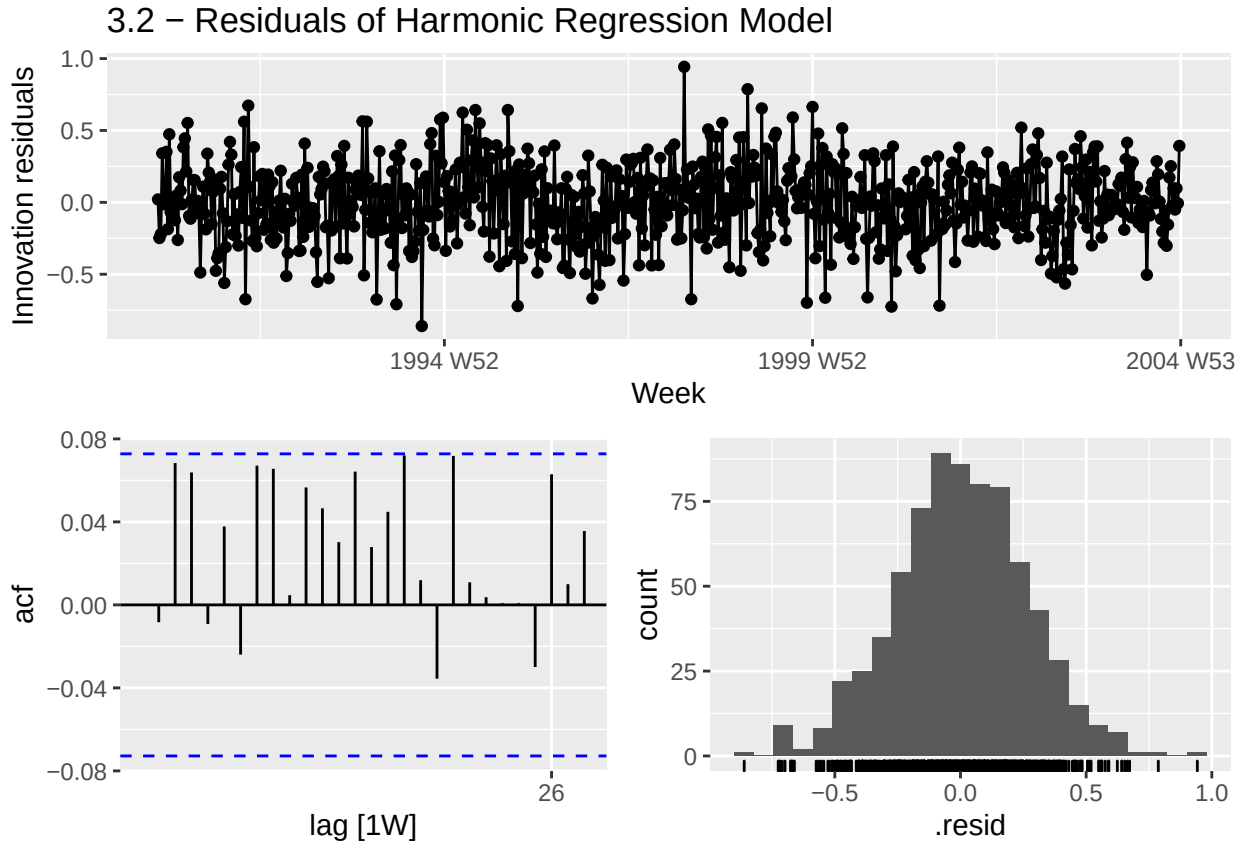

Figure 6: Figure 3.2

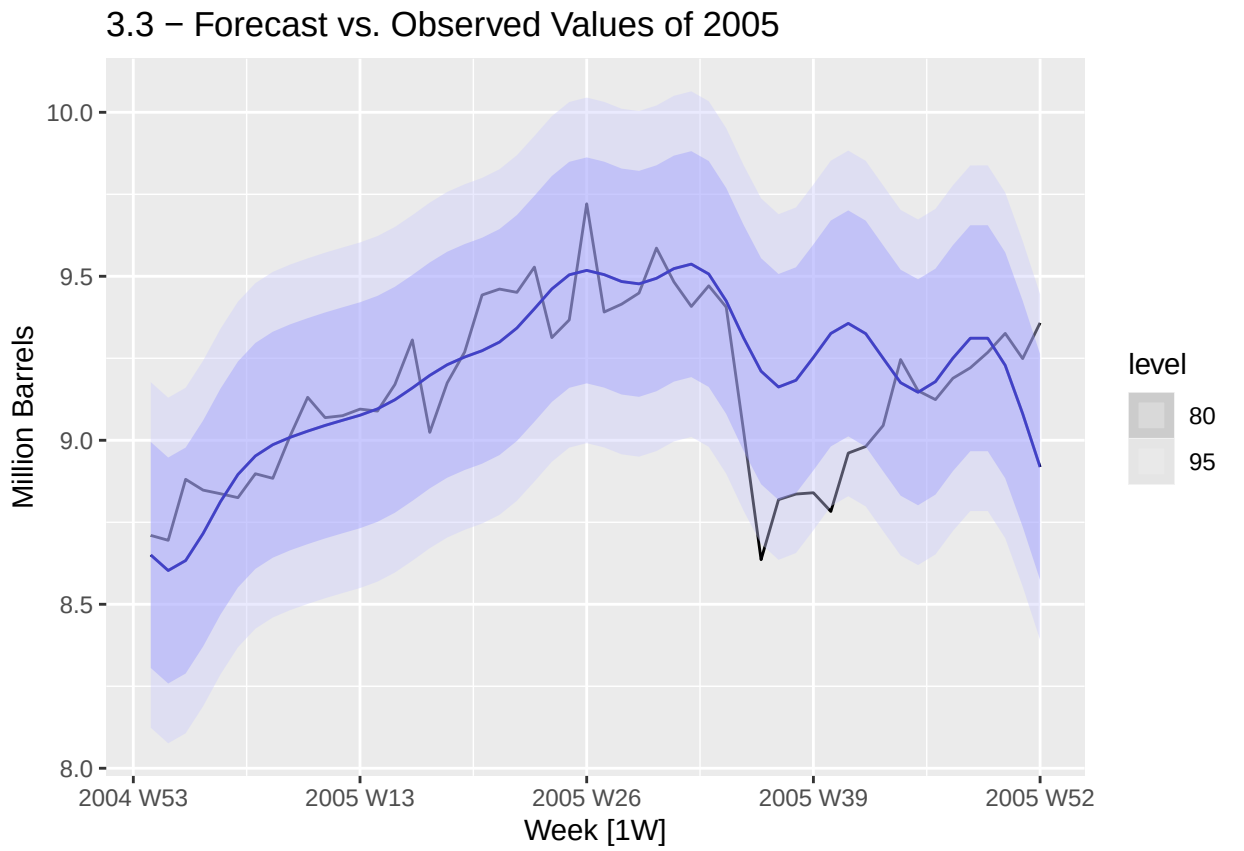

Figure 7: Figure 3.3

```
us_gasoline %>% autoplot(Barrels) +
  labs(title = '3.4 - US Gasoline Weekly Supplies') +
  ylab('Million Barrels')
```

Graph 3.5 - Residuals of the simple model with *PDQ(0, 0, 0)*

```
new_fit_0 %>% gg_tsresiduals() + labs(title = '3.5 - Residuals of simeple model')
```

Graph 3.6 - ACF and PACF plots of US gasoline

```
us_gasoline %>% gg_tsdisplay(difference(Barrels), plot_type = 'partial') +
  labs(title = '3.6 - ACF & PACF of US Gasoline')
```

```
## Warning: Removed 1 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 1 rows containing missing values (geom_point).
```

Graph 3.7 - Residuals of the optimized model

```
new_fit %>% gg_tsresiduals() + labs(title = '3.7 - Residuals of final model')
```

Graph 3.8 - Forecast of the next year

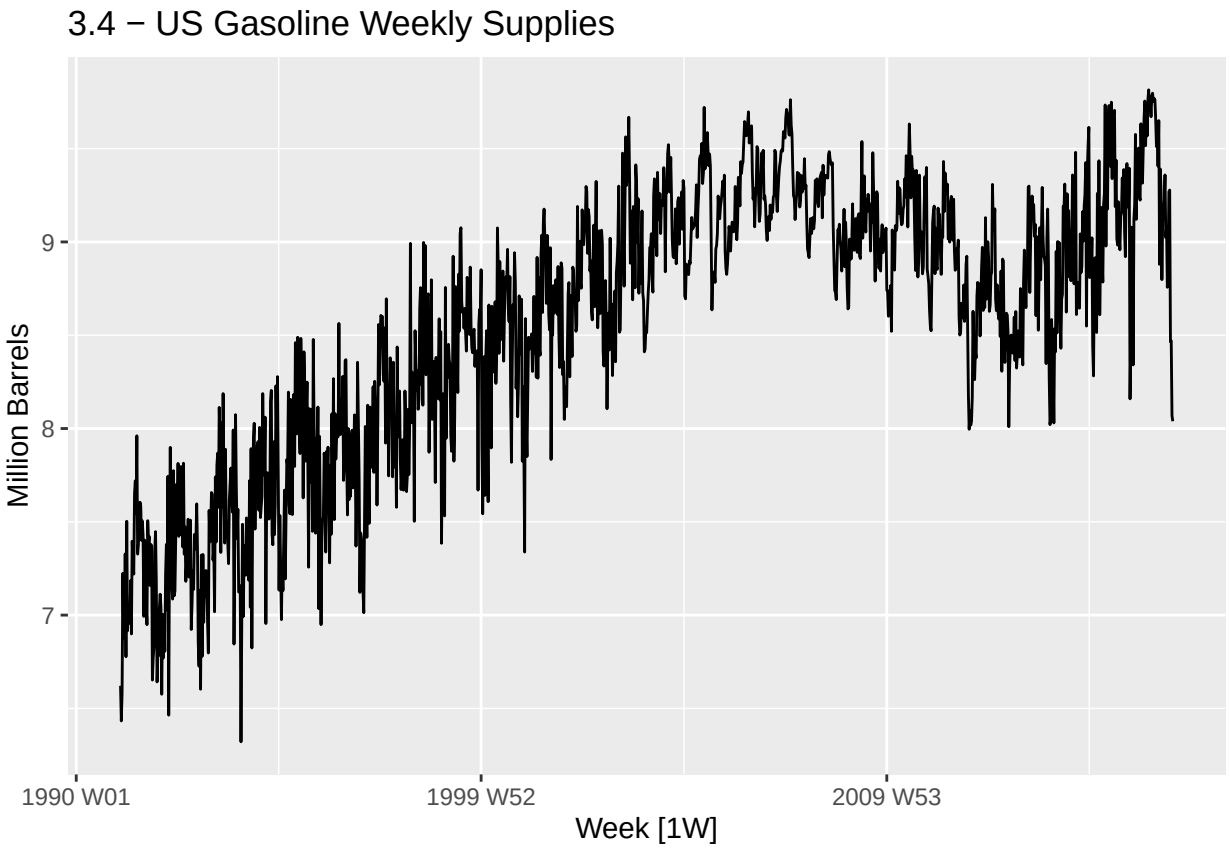

Figure 8: Figure 3.4

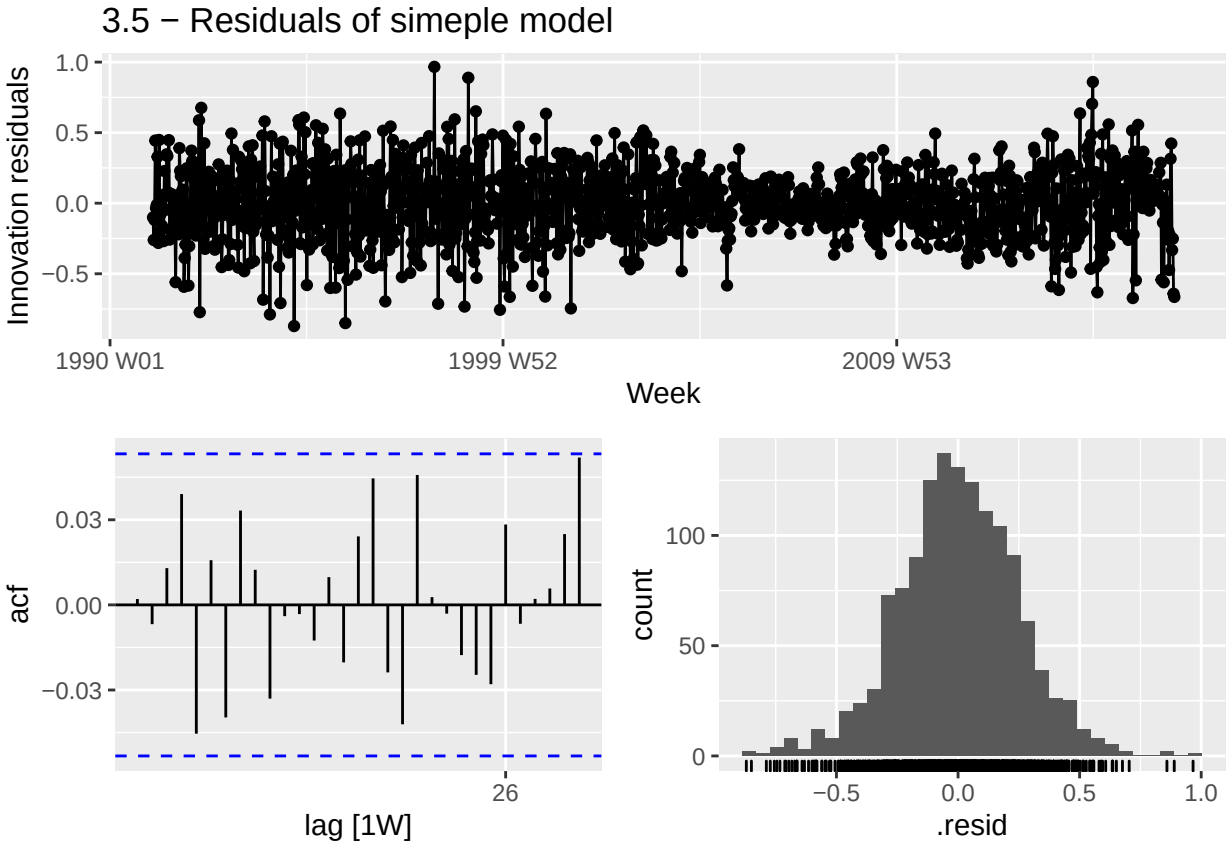

Figure 9: Figure 3.5

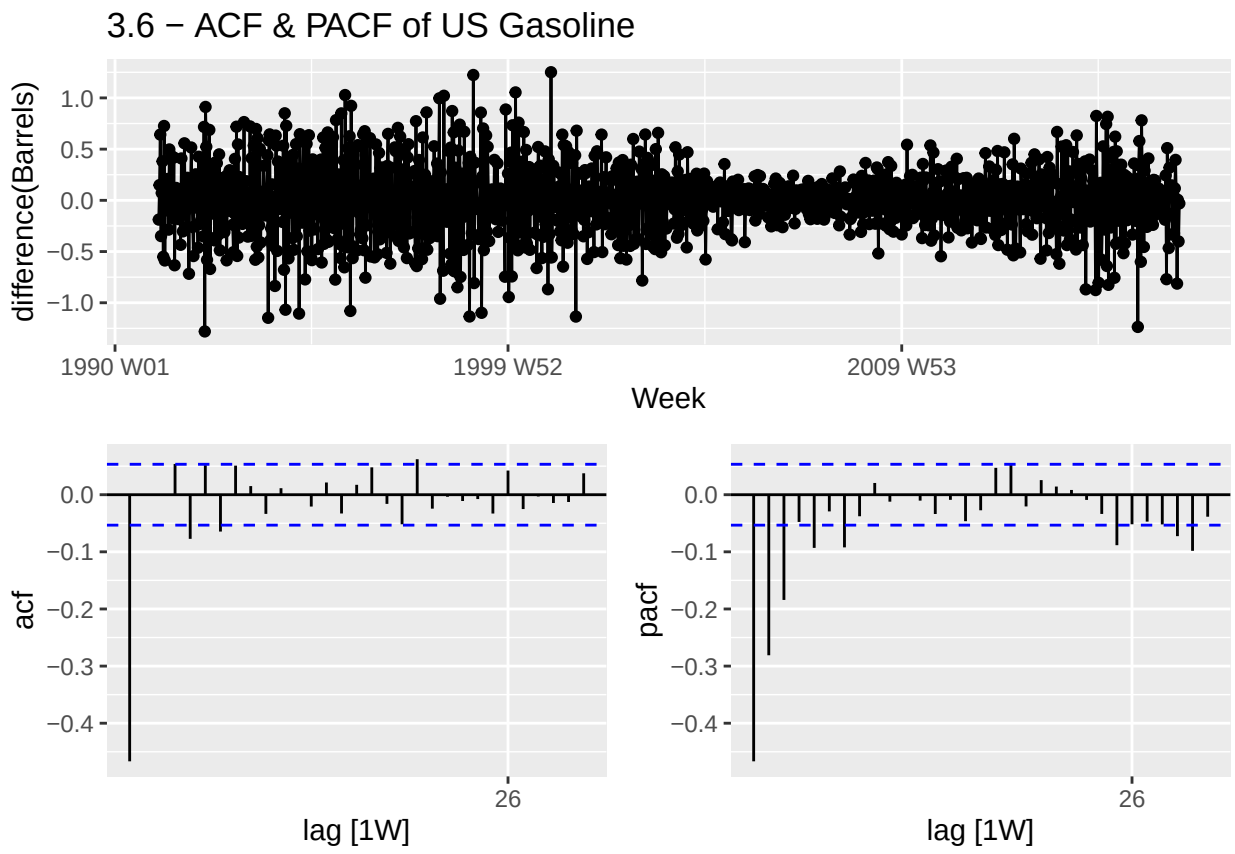

Figure 10: Figure 3.6

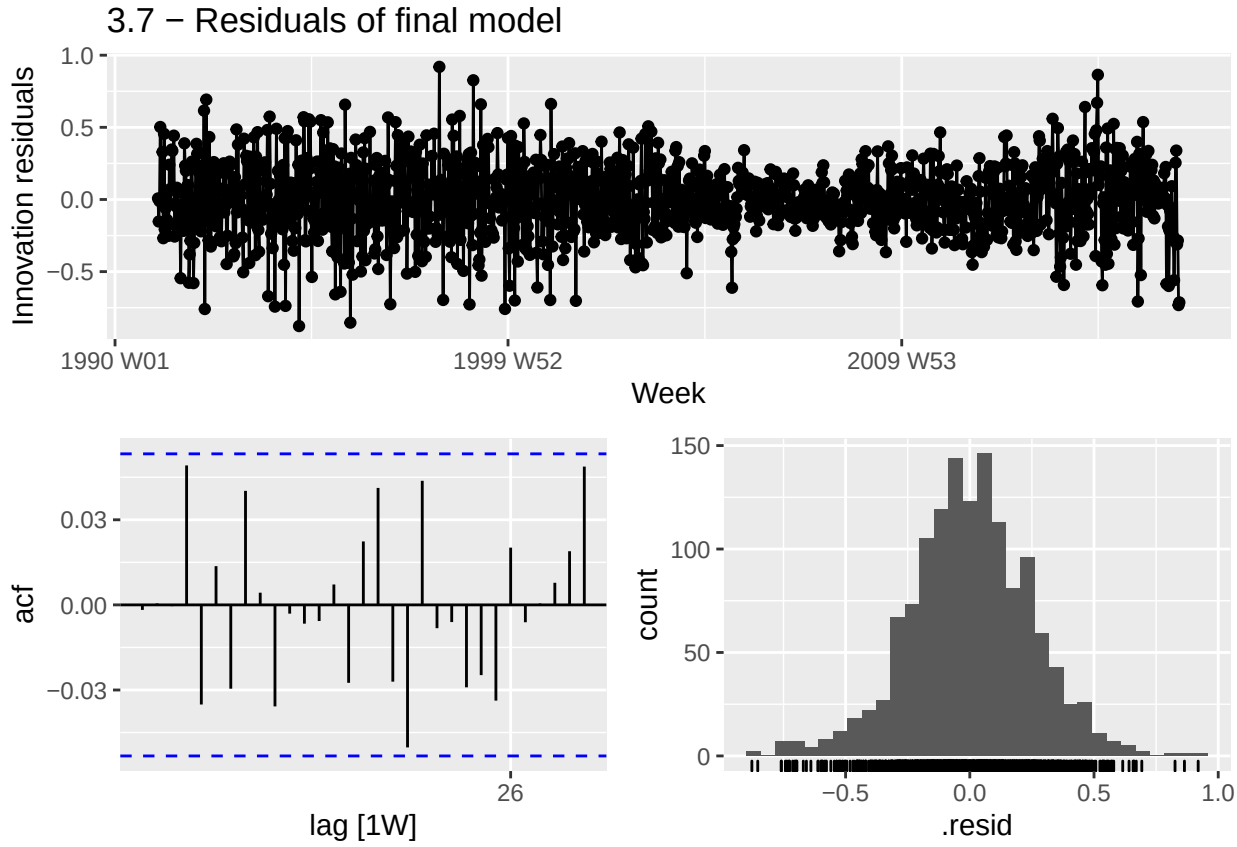

Figure 11: Figure 3.7

```
us_gasoline %>% autoplot(Barrels) + geom_line(data = new_fc, aes(y = .mean)) +
  autolayer(new_fc) +
  labs(title = '3.8 - Forecast of US Gasoline') +
  ylab('Million Barrels')
```

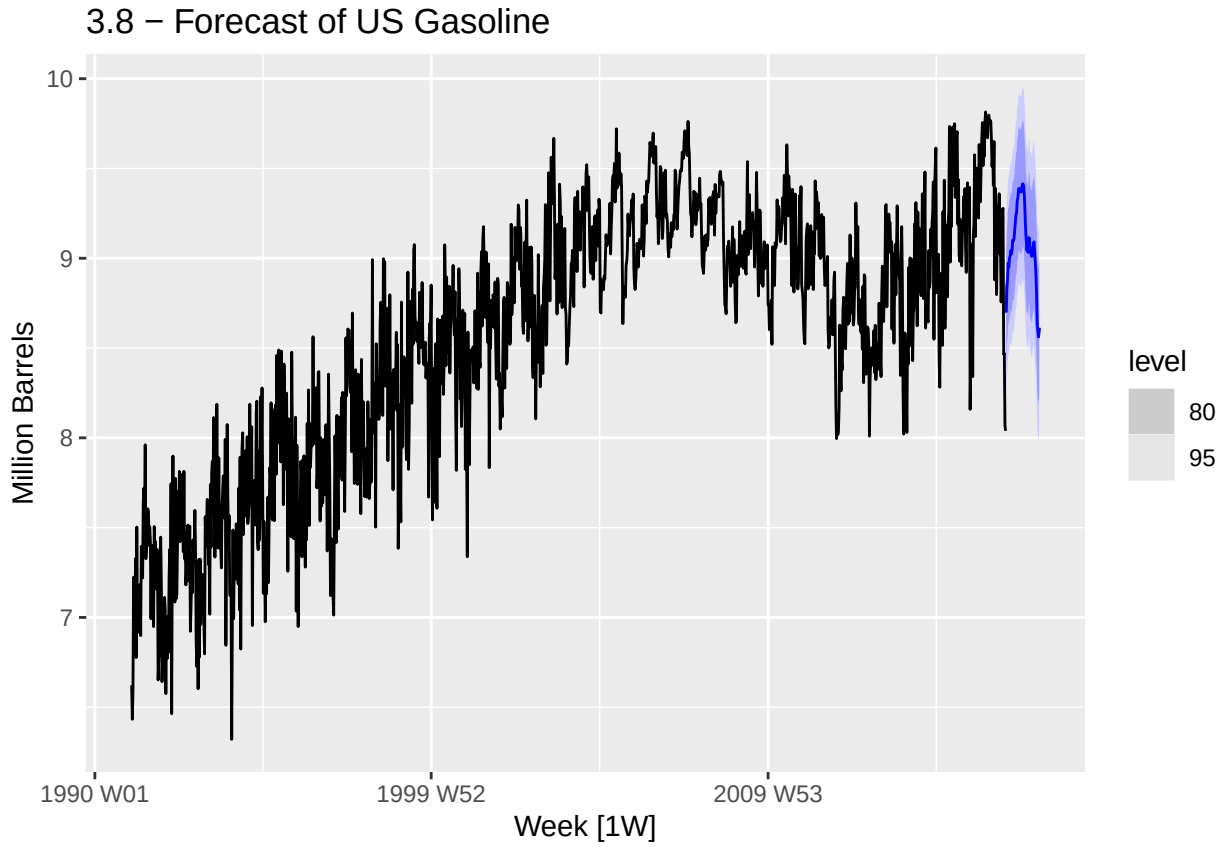

Figure 12: Figure 3.8

Graphs for Exercise 4

```
## Warning: Removed 11 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 10 row(s) containing missing values (geom_path).
```

```
## Warning: Removed 3 row(s) containing missing values (geom_path).
```

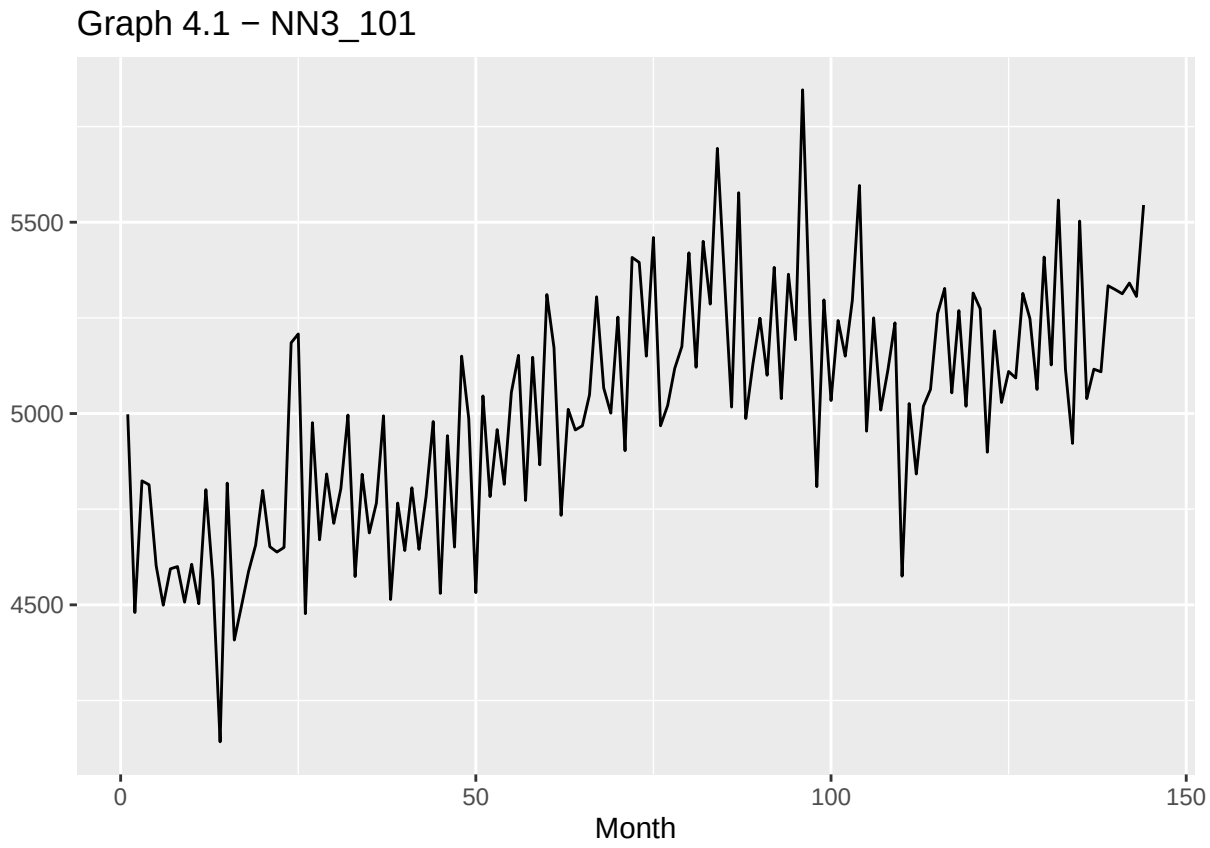

Figure 13: Figure 4.1

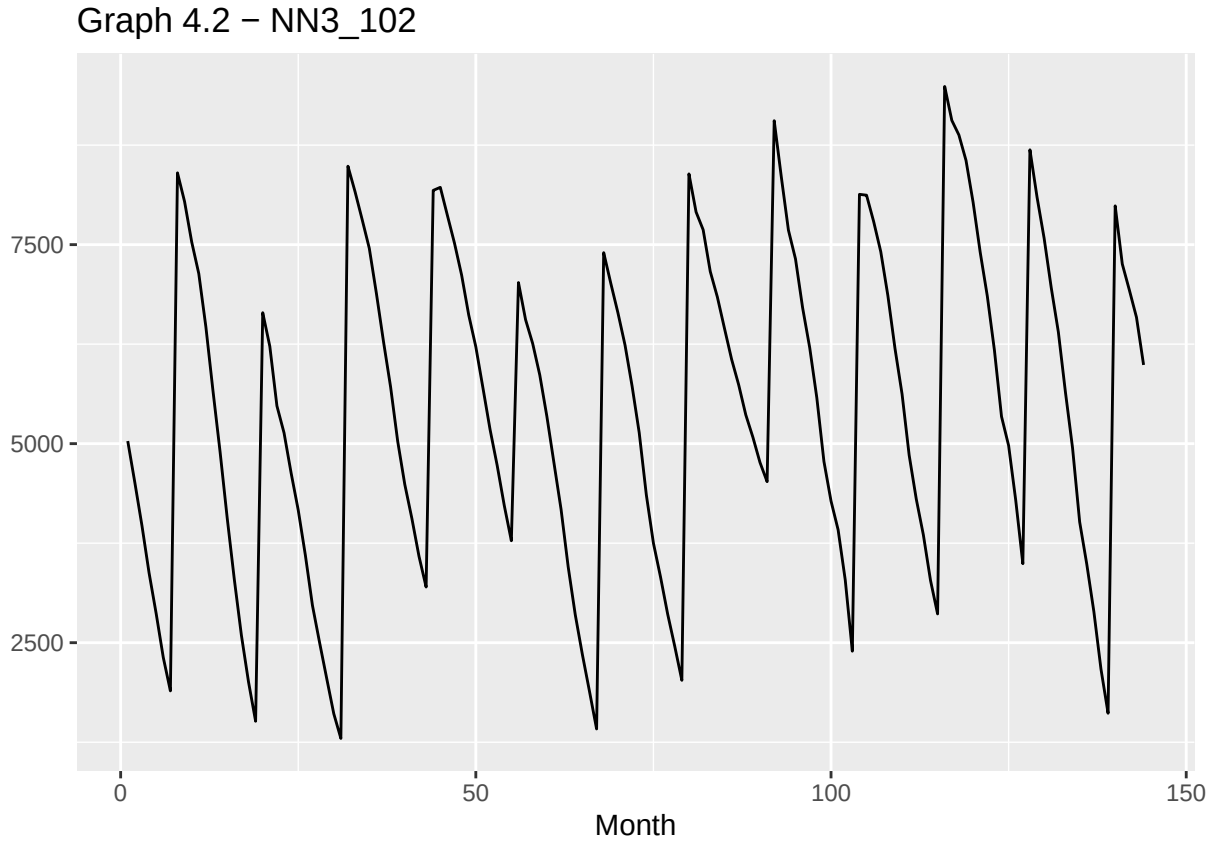

Figure 14: Figure 4.2

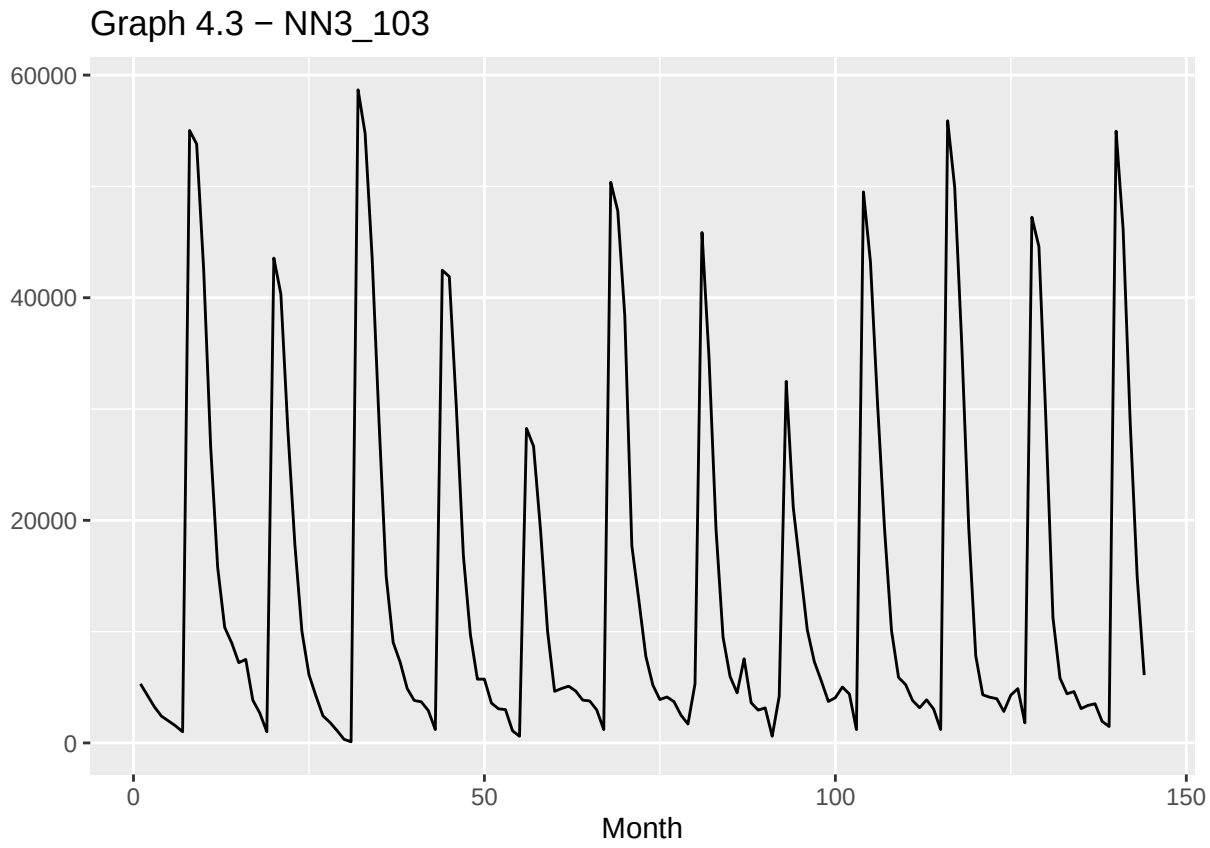

Figure 15: Figure 4.3

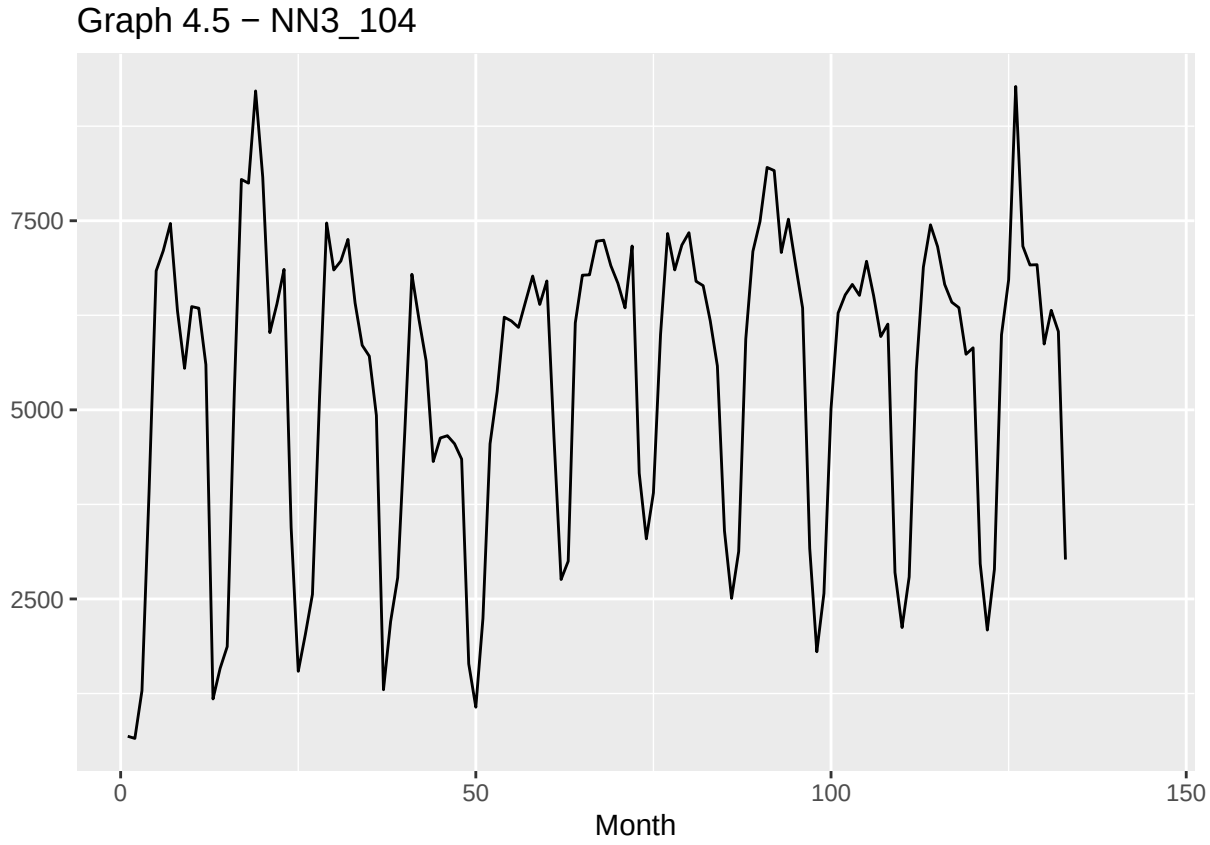

Figure 16: Figure 4.4

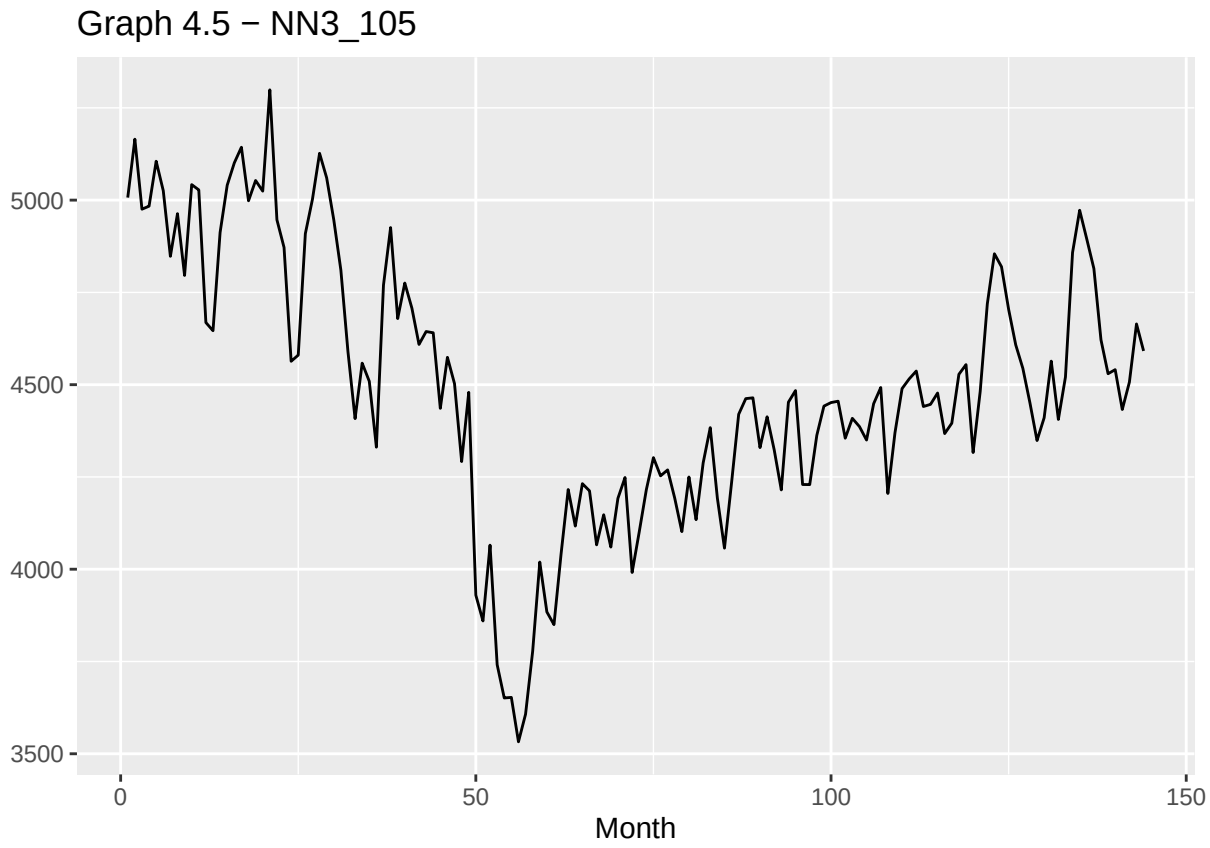

Figure 17: Figure 4.5

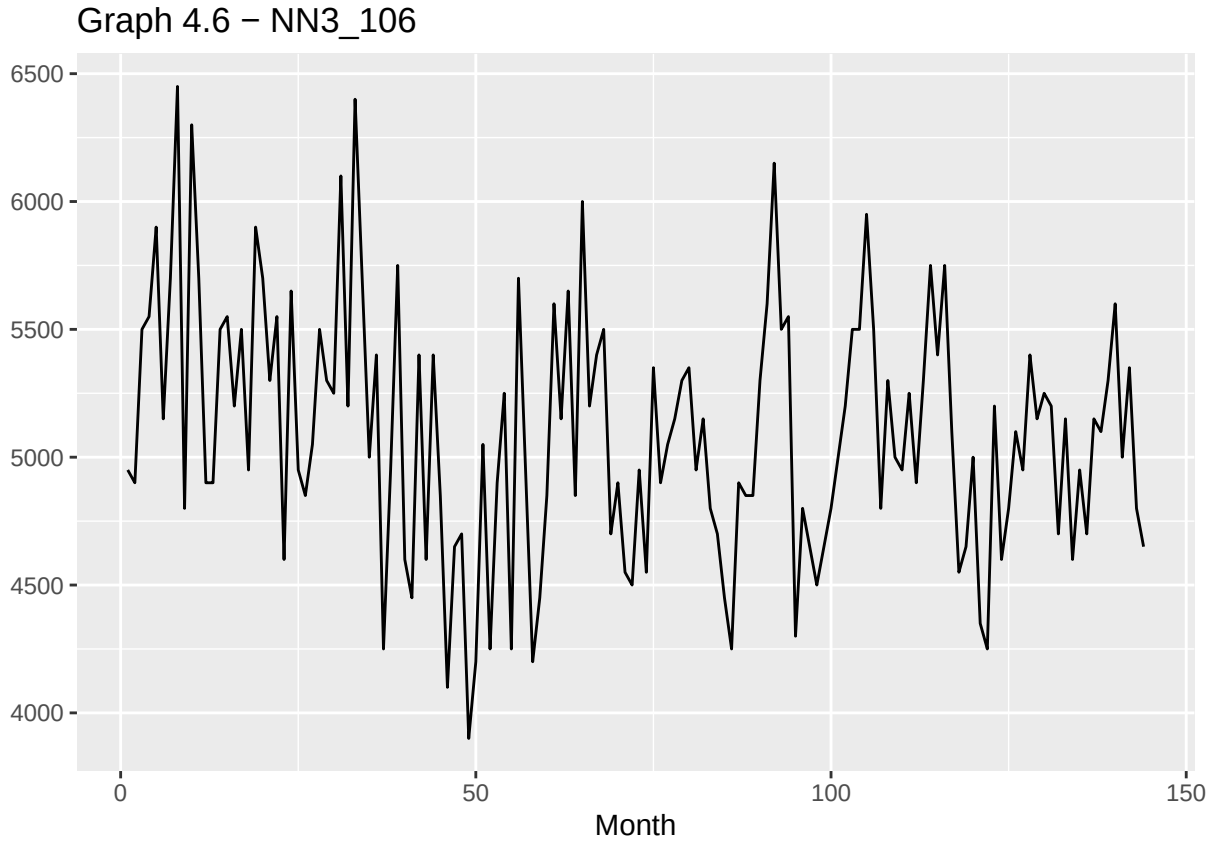

Figure 18: Figure 4.6

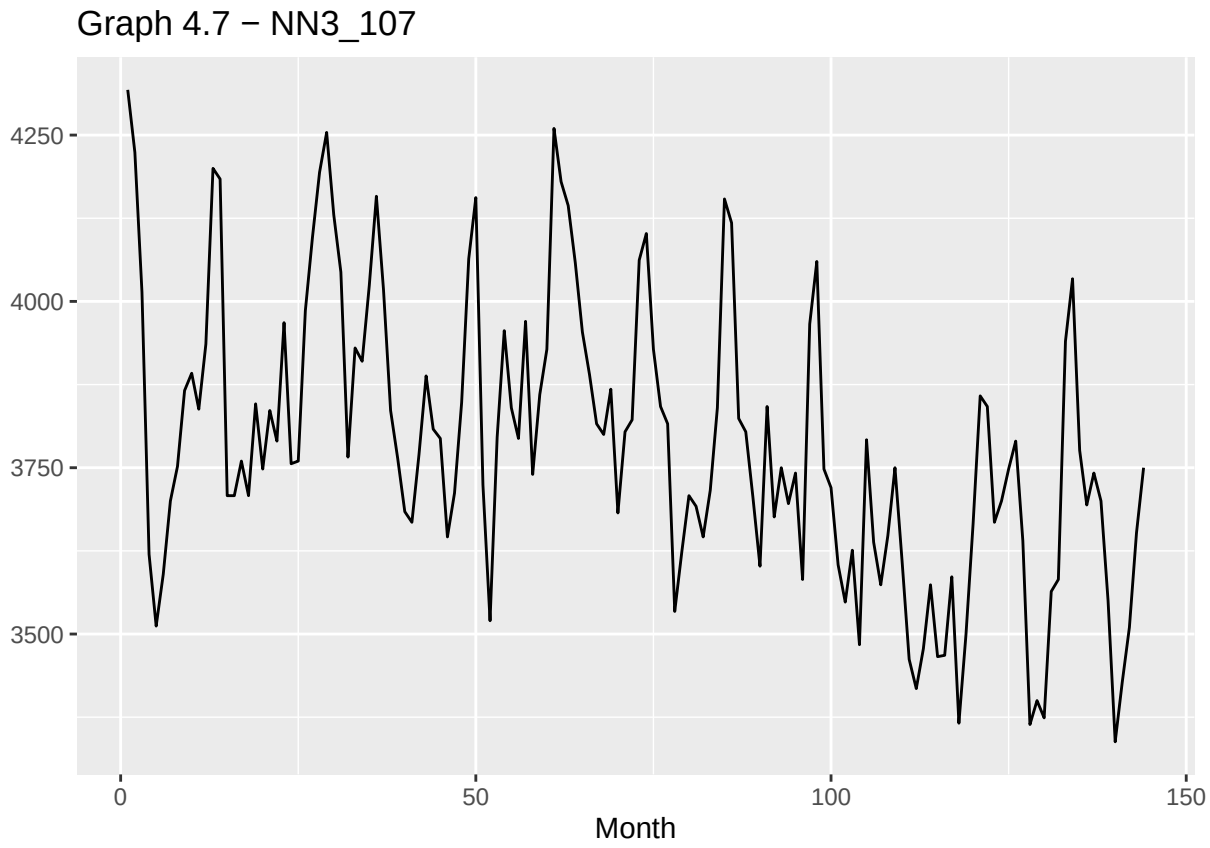

Figure 19: Figure 4.7

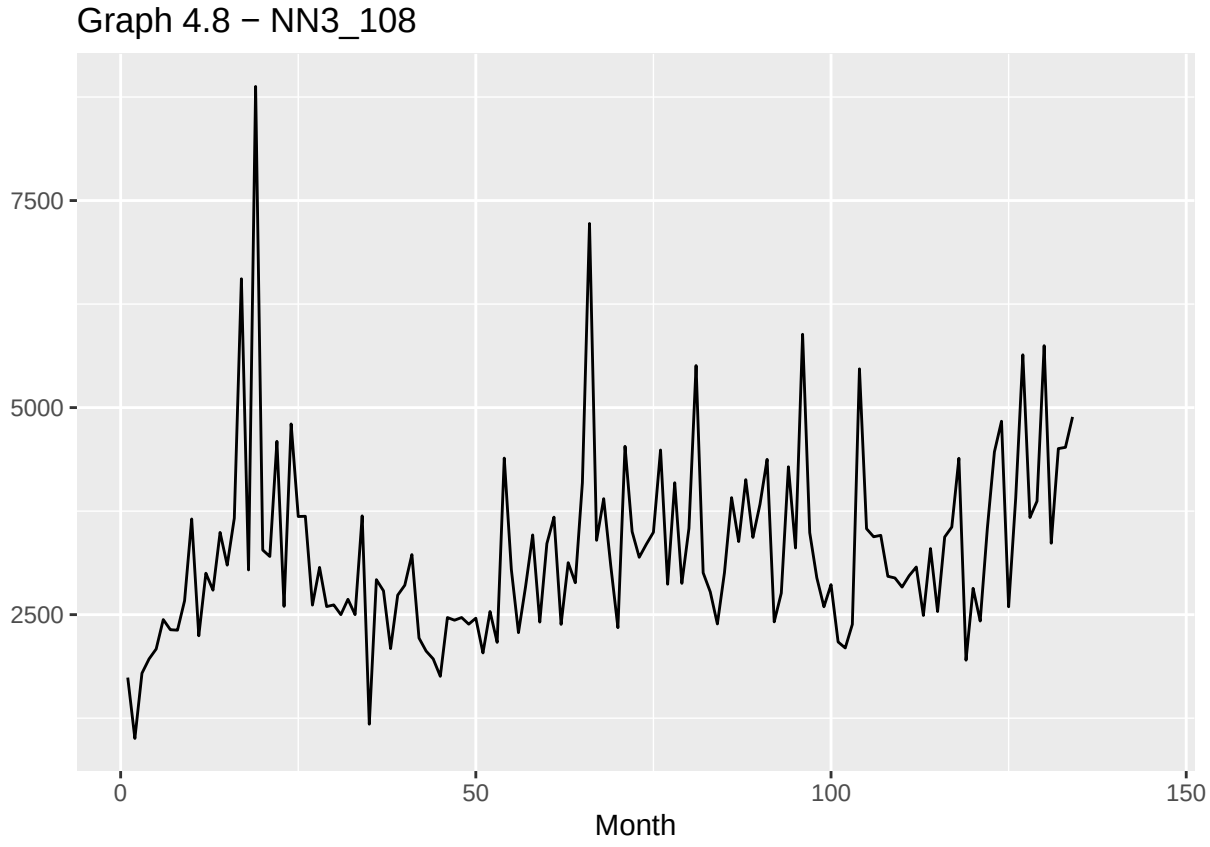

Figure 20: Figure 4.8

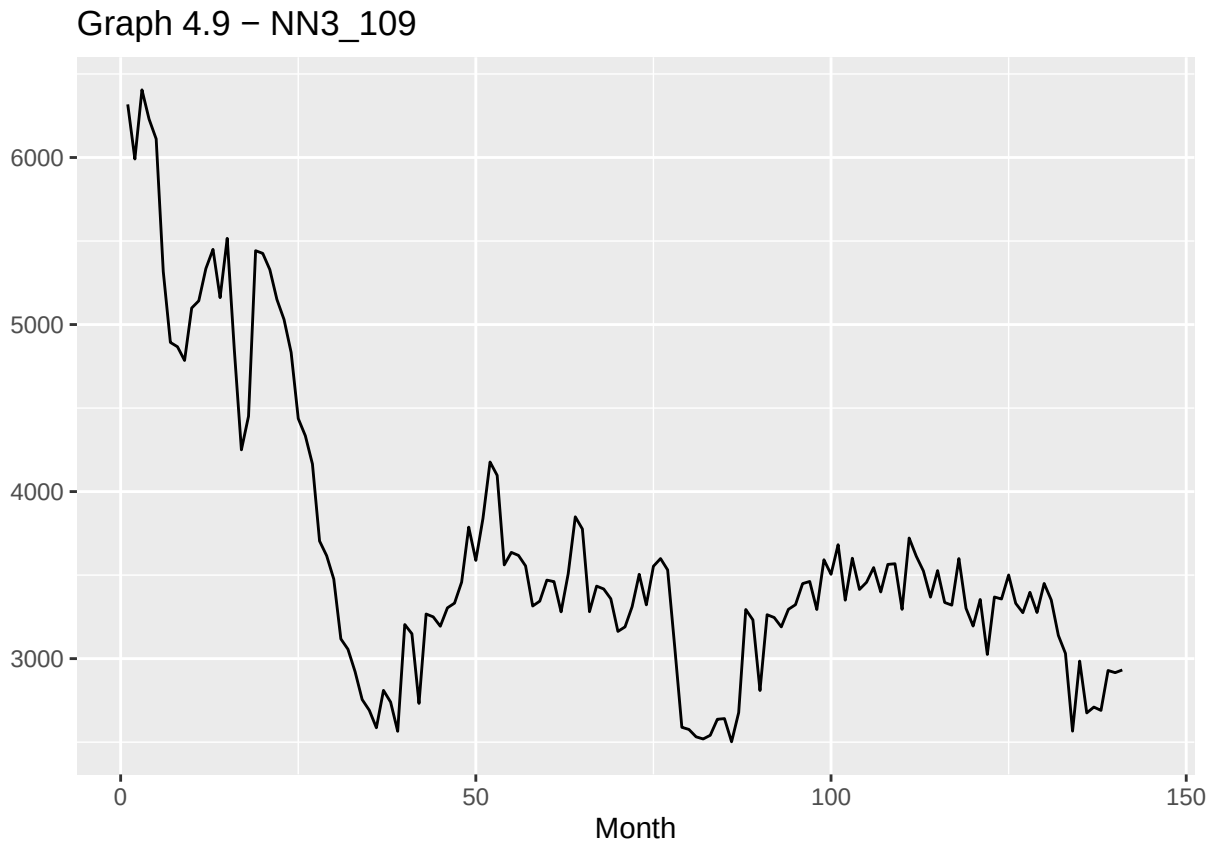

Figure 21: Figure 4.9

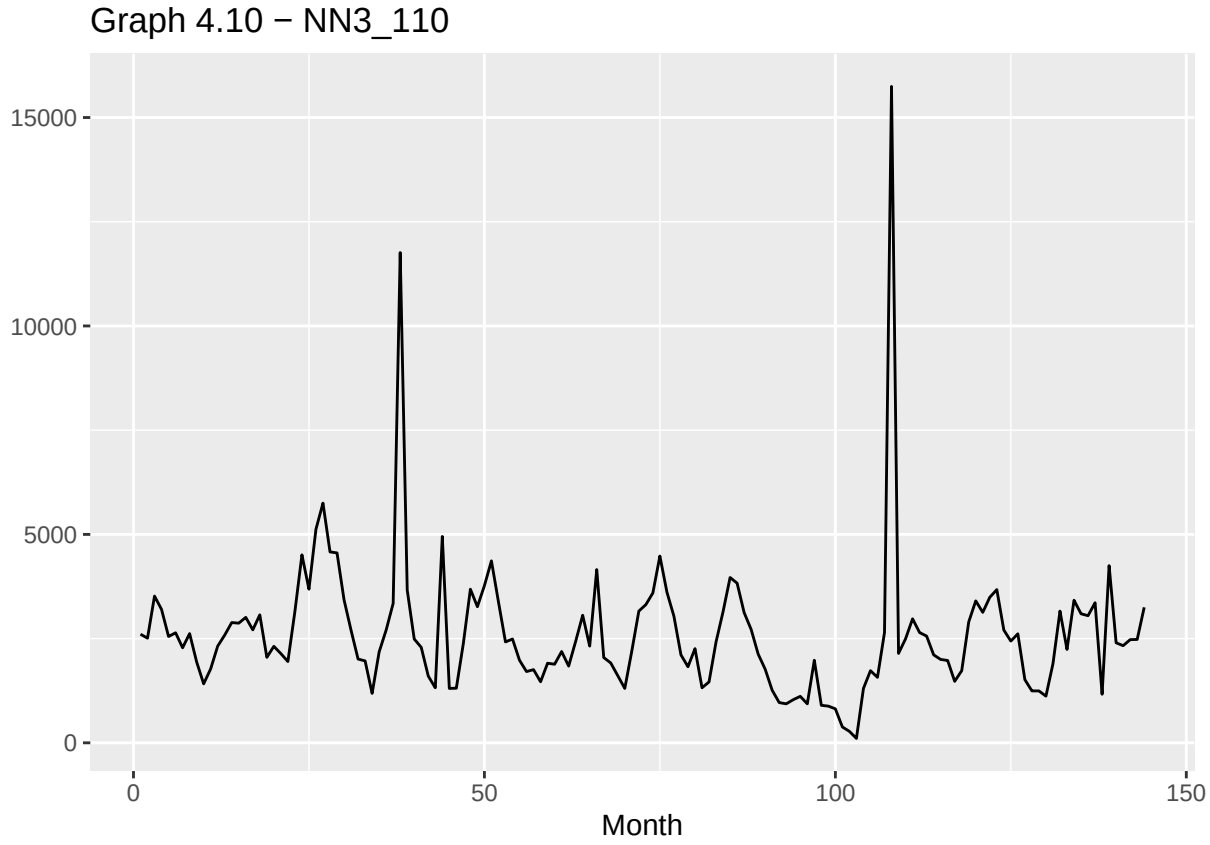

Figure 22: Figure 4.10

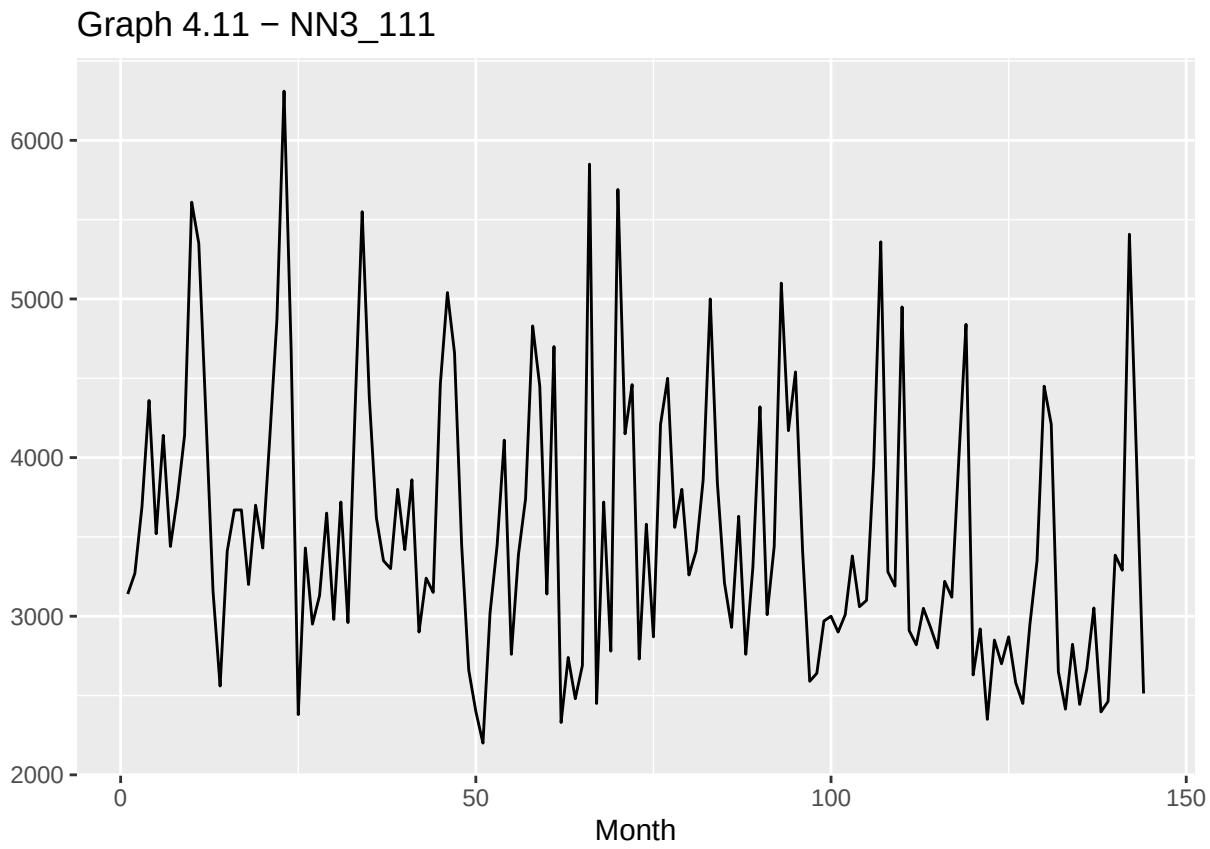

Figure 23: Figure 4.11