Problem 1: Labour productivity

Problem 2: Monthly Average Auckland Temperatures

Code ▾

STATS 326

Assignment 04

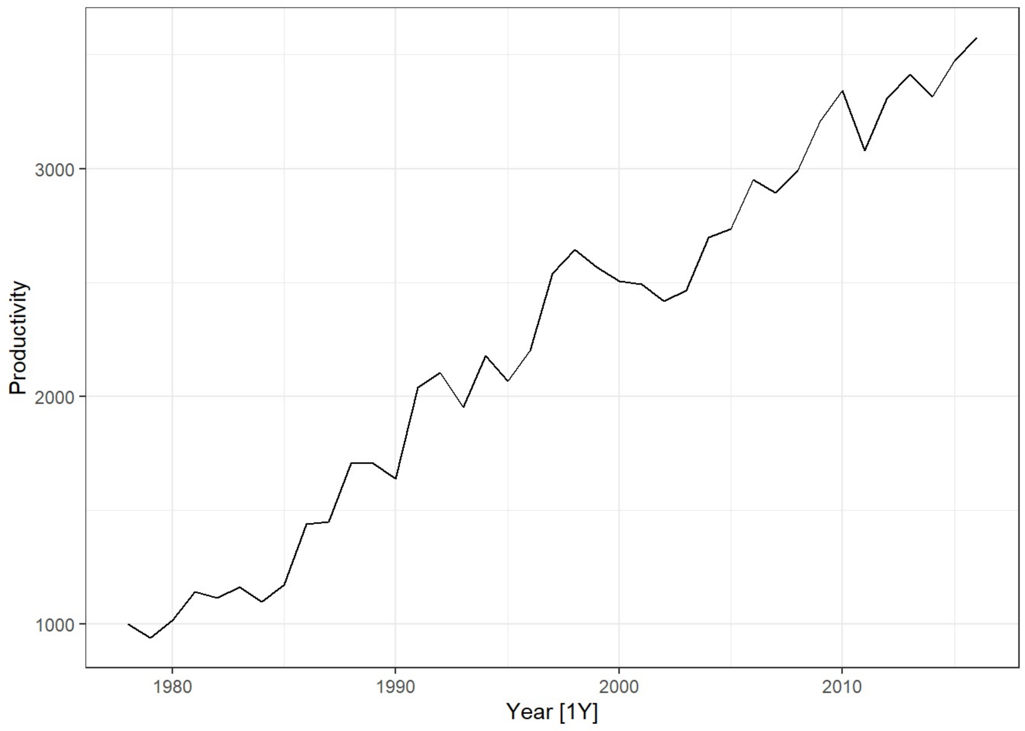
General comments:

- All the plots should be labelled appropriately (axes, legends, titles).
- Please submit both your `.Rmd`, and the generated output file `.html` or `.pdf` on Canvas before the due date/time.
- Please make sure that the `.Rmd` file compiles without any errors. The marker will not spend time fixing the bugs in your code.
- Please avoid specifying absolute paths.
- Your submission must be original, and if we recognize that you have copied answers from another student in the course, we will deduct your marks.
- You will need to use the `tidyverse` and `fpp3` libraries for this assignment.

Due: Friday 24 May 2024 at 16:00 PM (NZ time)

Problem 1: Labour productivity

In economics, productivity measures how much output has been produced per unit of input, and can therefore measure how efficient the economy is. The file `productivity.csv` contains the labour productivity (index) for primary industries in New Zealand, measured annually from 1978 to 2021. A training set called `train` has been created that withholds the years 2017–2021, and has been plotted below.



> 1. 4 Marks Based on the time plot above, think of two candidate ETS models to fit. Write down the reasons you think these models will be appropriate for the labour productivity series.

In oberving our time series plot, there appears to be an increasing, approximately linear trend with no seasonal component, therefore an ETS model with an additive trend and no seasonality makes sense. However, we are uncertain whether there are additive or multiplicative errors, hence the two candidate ETS models are ETS(A,A,N) and ETS(A,Ad,N). If the errors were multiplicative, we prefer to use a damped additive trend instead of multiplicative error component.

2. 6 Marks Fit these two candidate models as well as the automatic ETS model on your training set. Compare the models using AICc and output the parameter estimates from the model that has the best predictive ability. Name the best model and interpret the smoothing parameters.

Hide

```
#fit models
fit_productivity = train_productivity %>%
  model(ETS1 = ETS(Productivity ~ error("A") + trend("A") + season("N")),
        ETS2 = ETS(Productivity ~ error("A") + trend("Ad") + season("N")),
        auto.ETS = ETS(Productivity))

#compare AICc between models
fit_productivity %>% glance() %>% select(.model:BIC)
```

```
## # A tibble: 3 × 6
##   .model   sigma2 log_lik   AIC  AICc   BIC
##   <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 ETS1     19414.   -262.  534.  536.  542.
## 2 ETS2     20988.   -263.  538.  540.  548.
## 3 auto.ETS 19414.   -262.  534.  536.  542.
```

Hide

```
#store best model into variable and obtain smoothing parameters
AdditiveHolt.fit= train_productivity %>%
  model(ETS(Productivity ~ error("A") + trend("A") + season("N")))

report(AdditiveHolt.fit)
```

```
## Series: Productivity
## Model: ETS(A,A,N)
##   Smoothing parameters:
##     alpha = 0.4889942
##     beta  = 0.0001000043
##
##   Initial states:
##      l[0]     b[0]
##  880.7263 68.67191
##
##   sigma^2:  19414.32
##
##      AIC     AICc      BIC
## 533.7355 535.5536 542.0533
```

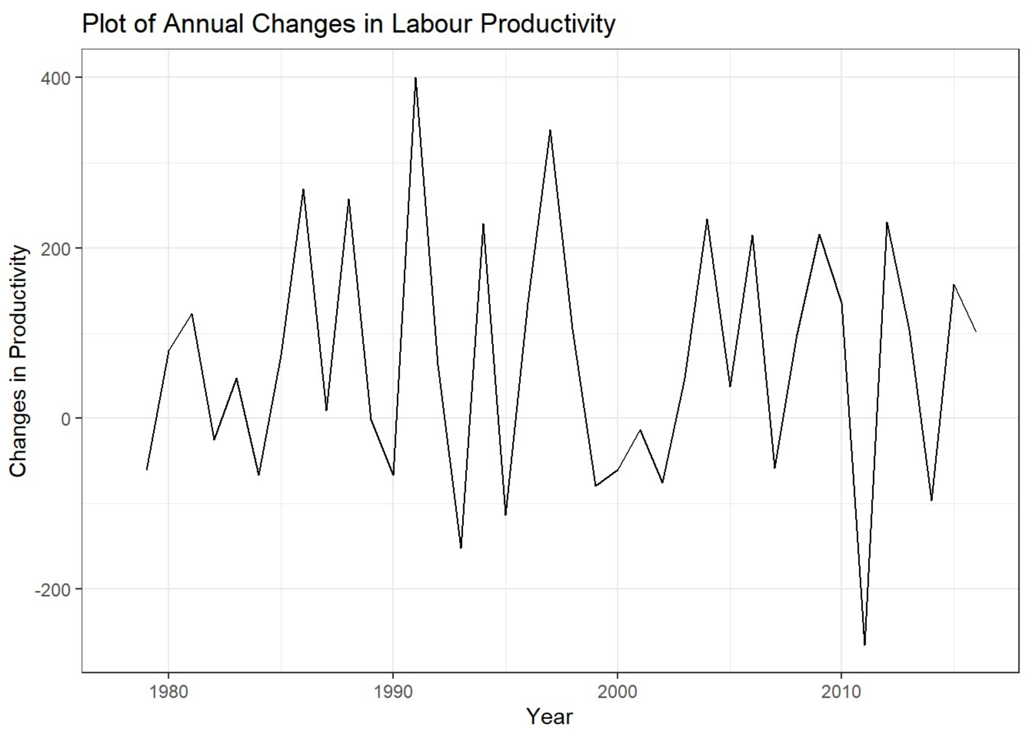
It appears that our first ETS model (ETS1) and automatic ETS model (auto.ETS) has the lowest AICc value of 535.55, and ETS2 has a higher AICc value of 540.27, which suggests it does not provide as accurate fit as our first and automatic ETS model. The first and automatic ETS model provides the most accurate fit, which has an additive error term, additive trend and no seasonality, this is also known as the Additive Holt-Winter's method, ETS(A,A,N).

Our smoothing parameters for alpha and beta are 0.489 and 0.0001, respectively. As we obtained a medium value of 0.489 for alpha, the level trend changes moderately to capture the underlying trend of our data. We also obtained a very small value of 0.0001 for beta, this means the slope rarely changes with time.

3. 5 Marks Plot the differenced productivity series from the training set and comment on whether you believe this series is stationary. Verify this using a KPSS test on the differenced productivity series, making sure you interpret the *p*-value.

Hide

```
#plot differenced productivity series
train_productivity %>%
  autoplot(difference(Productivity)) +
  ggtitle("Plot of Annual Changes in Labour Productivity") +
  xlab("Year") + ylab("Changes in Productivity") +
  theme_bw()
```



Hide

```
#KPSS test
train_productivity %>%
  mutate(diff_prod = difference(Productivity)) %>%
  features(diff_prod, unitroot_kpss)
```

```
## # A tibble: 1 × 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0520         0.1
```

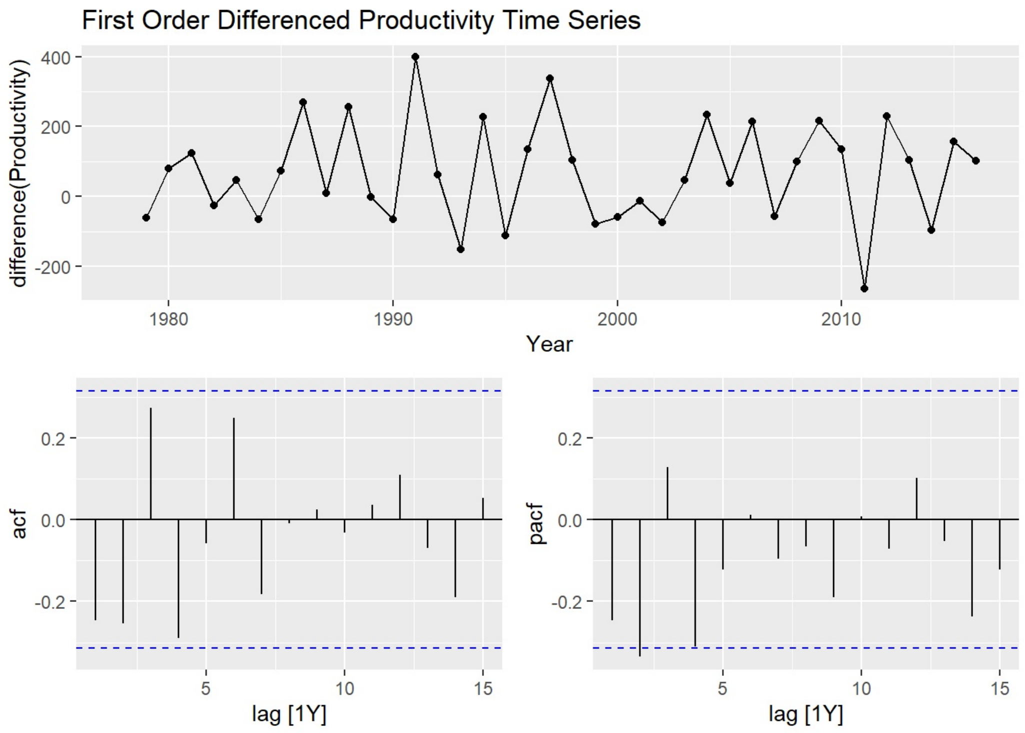
In observing our differenced productivity series, there appears to be no trend or seasonality present, hence our time series is considered stationary.

Furthermore, as we obtained a large p-value of 0.1, we do not have sufficient evidence to reject the null hypothesis that the data is stationary and non-seasonal. Hence, we conclude that our time series does not depend on time.

4. 7 Marks Plot the ACF and PACF plots for your differenced training data (you may use the `gg_tsdisplay` function). Analyse these plots to come up with two candidate ARIMA models. Explain why you have chosen these ARIMA models.

Hide

```
#ACF and PACF plot for differenced training data
train_productivity %>%
  gg_tsdisplay(difference(Productivity), plot_type = 'partial') +
  ggtitle("First Order Differenced Productivity Time Series")
```



After taking a first-order difference of our data, we obtained a stationary productivity time series. Hence, the order of differencing for an ARIMA model is d = 1.

In the pattern that we observe in the PACF for a non-seasonal AR(p), there is a significant spike at lag 2 and none after. The pattern that we observe in the ACF of a non-seasonal AR(p) is sinusoidal dampening. This would lead us to consider p = 2 AR terms, therefore one of the candidate models would be ARIMA(2,1,0).

In the pattern that we observe in the ACF for a non-seasonal MA(q), there doesn't appear to be any significant spikes as all the values are within the 0.1 threshold. The pattern that we observe in the PACF of a non-seasonal MA(q) is sinusoidal dampening. As there aren't any significant spikes and a decaying PACF, we would consider q = 0 MA terms. Therefore, our second candidate model would be ARIMA(0,1,0).

5. 6 Marks Fit your candidate model as well as the automatic ARIMA model on your training set. Compare the models using AICc and output the parameter estimates from the model that has the best predictive ability. Name the best model and write the fitted model equation using backshift notation.

Hide

```
#fit ARIMA models
fit_prod_arima <- train_productivity %>%
  model(arima210 = ARIMA(Productivity ~ pdq(2, 1, 0)),
        arima010 = ARIMA(Productivity ~ pdq(0, 1, 0)),
        stepwise = ARIMA(Productivity),
        search = ARIMA(Productivity, stepwise = FALSE))

#compare AICc values
glance(fit_prod_arima) %>% select(.model:BIC)
```

```
## # A tibble: 4 × 6
##   .model   sigma2 log_lik   AIC  AICc   BIC
##   <chr>     <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 arima210 18401.   -239.  486.  487.  493.
## 2 arima010 21037.   -243.  489.  489.  492.
## 3 stepwise 18873.   -240.  486.  487.  491.
## 4 search   18873.   -240.  486.  487.  491.
```

Hide

```
#fit ARIMA model with drift method
drift.fit <- train_productivity %>%
  model(ARIMA(Productivity ~ pdq(0, 1, 1)))

drift.fit %>% report()
```

```
## Series: Productivity
## Model: ARIMA(0,1,1) w/ drift
##
## Coefficients:
##           ma1  constant
##       -0.4888   68.8464
## s.e.   0.2121   11.4006
##
## sigma^2 estimated as 18873:  log likelihood=-240.09
## AIC=486.19   AICc=486.89   BIC=491.1
```

It appears that our second candidate model ARIMA(0,1,0) has the highest AICc value of 489.42 which suggests that it has the least accurate fit. The first candidate model, ARIMA(2,1,0) has a lower AICc value of 487.39 than ARIMA(0,1,0), this suggests it has a more accurate fit. The automatic ARIMA model, ARIMA(0,1,1) has the lowest AICc value of 486.89, this suggests it has the most accurate fit.

The best model is ARIMA(0,1,1), which is the model with drift method.

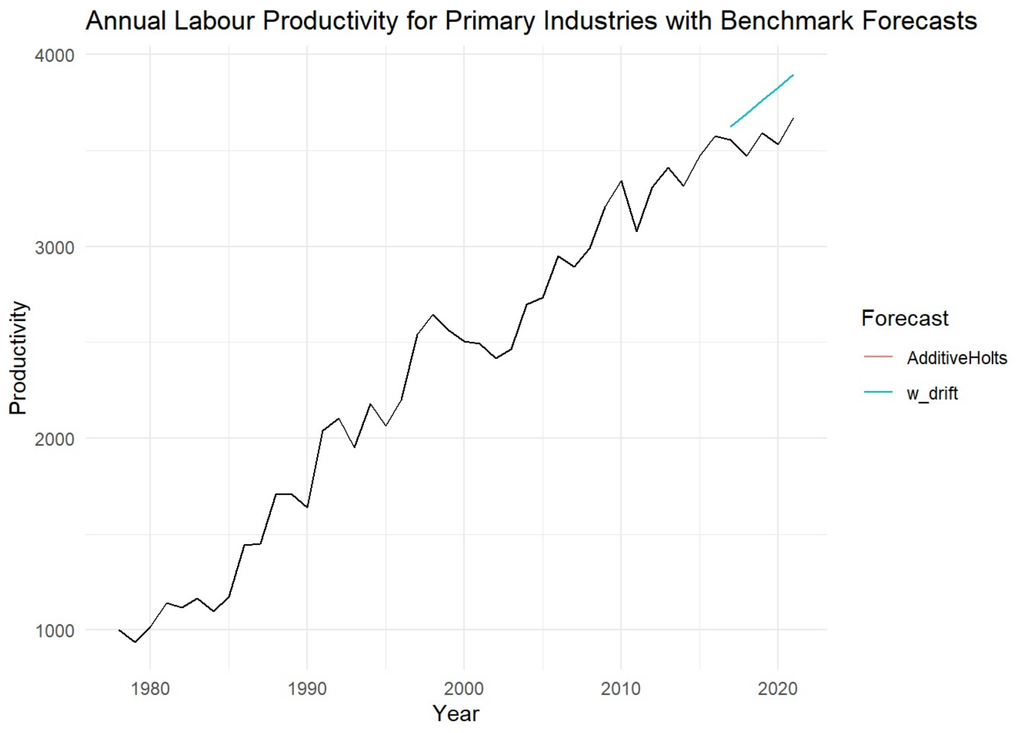
$$y_t - y_{t-1} = c + \varepsilon_t + \theta_1 \varepsilon_{t-1}(1-B)y_t = c + (1+\theta_1 B)\varepsilon_t(1-B)y_t = 68.85 + (1-0.49B)\varepsilon_t$$

6. 6 Marks Fit your best ETS and ARIMA model on the training data in the same `mable`. Forecast $h = 5$ periods into the future. Plot your forecasts. Compare your forecasts against the truth by computing RMSE, MAE, MAPE, and MASE. State which model has the best forecast accuracy.

Hide

```
#Fit ETS and ARIMA models, and forcast 5 periods
productivity.fc <- train_productivity %>%
  model(
    AdditiveHolts = ETS(Productivity ~ error("A") + trend("A") + season("N")),
    w_drift = ARIMA(Productivity ~ pdq(0, 1, 1))
    ) %>% forecast(h = 5)

#Plot forecasts
productivity.fc %>%
  autoplot(data_productivity, level = NULL) +
  xlab("Year")+ ylab("Productivity") +
  ggtitle("Annual Labour Productivity for Primary Industries with Benchmark Forecasts") +
  guides(colour=guide_legend(title = "Forecast")) +
  theme_minimal()
```


Annual Labour Productivity for Primary Industries with Benchmark Forecasts
4000
3000
2000
1000
Productivity
1980
1990
2000
2010
2020
Year
Forecast
AdditiveHolts
w_drift

Hide

```
#Compare forecast accuracy
prod.fc.measures <- productivity.fc %>%
  accuracy(data_productivity) %>%
  select(.model, RMSE, MAE, MAPE, MASE)

prod.fc.measures
```

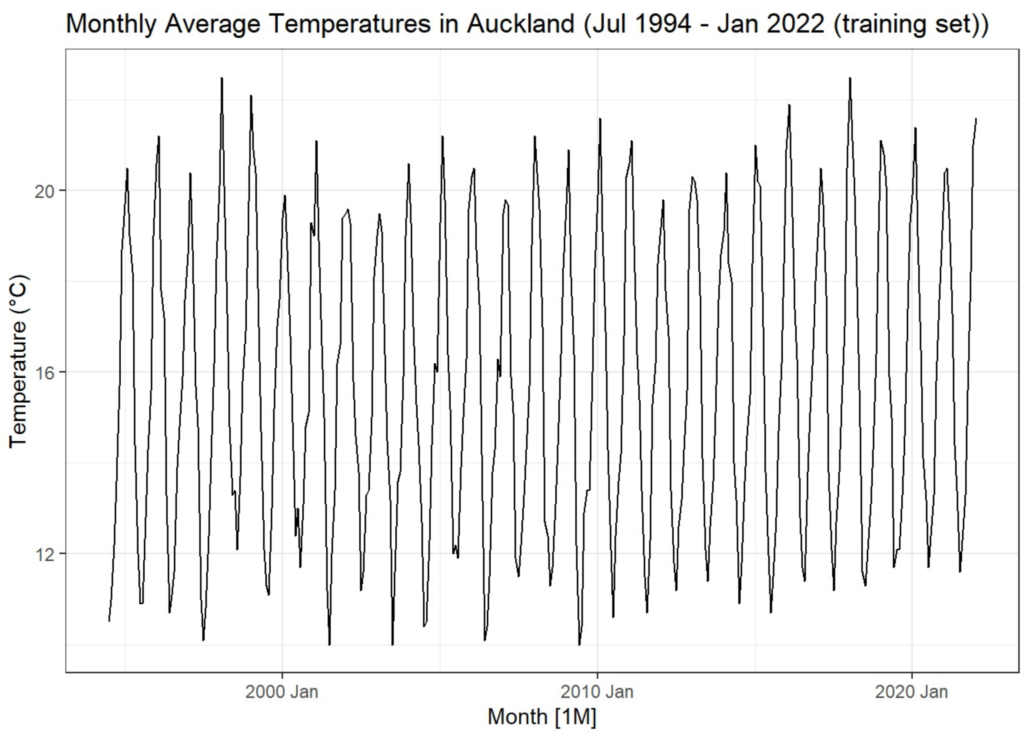
```
## # A tibble: 2 × 5
##   .model         RMSE   MAE  MAPE  MASE
##   <chr>         <dbl> <dbl> <dbl> <dbl>
## 1 AdditiveHolts  208.  194.  5.43  1.52
## 2 w_drift        209.  195.  5.47  1.53
```

It appears that the ETS Additive Holt-Winter's model has a lower forecast error across all four metrics in comparison to the ARIMA model with drift method. For RMSE, MAE, MAPE, MASE, the ETS model scored respectively, 208.1, 193.6, 5.43 and 1.52 which are slightly lower than the forecast errors of the ARIMA model with drift method, which are respectively 209.3, 194.7, 5.47 and 1.53. Hence, we prefer the Additive Holt-Winter's (ETS) model as it has the best forecast accuracy for our NZ labor productivity time series.

Possible marks for **Problem 1**: 34 Marks

Problem 2: Monthly Average Auckland Temperatures

In Assignments 1, 2, and 3, you investigated monthly average temperatures in Auckland. In this problem, you will do some further analysis. The data set `auckland_temps.csv` contains the monthly average temperatures in Auckland from July 1994 until January 2024. For this question do not Box-Cox transform the data. A training set called `train` has been created that withholds the most recent two years of data, and this has been plotted below.



1. 4 Marks Based on the time plot above, think of two candidate ETS models to fit. Write down why you think these models will be appropriate for the time series given.

In observing our monthly average temperature time series, there appears to be no trend but there is seasonality in which the temperature values fluctuate within the same range, this means the seasonality are additive effects. Therefore, an ETS model with no trend and an additive seasonal component makes sense. However, we are unsure whether the errors are additive or multiplicative. Hence, the two candidate ETS models are ETS(A,N,A) and ETS(M,N,A).

2. 6 Marks Fit these two candidate models as well as the automatic ETS model on your training set. Compare the models using AICc and output the parameter estimates from the model that has the best predictive ability. Interpret the smoothing parameters.

Hide

```
#fit models
fit_temp = train_temp %>%
  model(ETS1 = ETS(Temperature ~ error("A") + trend("N") + season("A")),
        ETS2 = ETS(Temperature ~ error("M") + trend("N") + season("A")),
        auto.ETS = ETS(Temperature))

#compare AICc between models
fit_temp %>% glance() %>% select(.model:BIC)
```

```
## # A tibble: 3 × 6
##   .model    sigma2 log_lik   AIC  AICc   BIC
##   <chr>      <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 ETS1     0.624     -875. 1780. 1781. 1837.
## 2 ETS2     0.00276   -883. 1795. 1797. 1852.
## 3 auto.ETS 0.624     -875. 1780. 1781. 1837.
```

Hide

```
#store best model into variable and obtain smoothing parameters
ETS.ANA= train_temp %>%
  model(ETS(Temperature ~ error("A") + trend("N") + season("A")))

report(ETS.ANA)
```

```
## Series: Temperature
## Model: ETS(A,N,A)
##   Smoothing parameters:
##     alpha = 0.115503
##     gamma = 0.0001022135
##
##   Initial states:
##     l[0]      s[0]     s[-1]    s[-2]    s[-3]    s[-4]    s[-5]    s[-6]
##  15.3319 -3.591093 -1.353245 1.097064 3.360583 4.939143 4.327193 2.925433
##      s[-7]     s[-8]     s[-9]    s[-10]    s[-11]
##  0.6546313 -1.134251 -2.561042 -4.033425 -4.630991
##
##   sigma^2:  0.6235
##
##      AIC     AICc      BIC
## 1779.845 1781.369 1836.877
```

It appears that our first ETS model (ETS1) and automatic ETS model (auto.ETS) has the lowest AICc value of 1781.40, and ETS2 has a higher AICc value of 1796.77, which suggests it does not provide as accurate fit as our first and automatic ETS model. The first and automatic ETS model provides the most accurate fit, which has an additive error term, no trend and additive seasonal component, ETS(A,N,A).

Our smoothing parameters for alpha and gamma are respectively 0.116 and 0.0001. As we obtained a relatively small value of 0.116 for alpha, the level trend changes rarely. As we obtained an extremely small value of 0.0001 for gamma, this means the seasonal component rarely changes with time. Both smoothing parameters suggests the fitted values are smooth.

3. 11 Marks Explore how many orders of seasonal differencing, D, and first differencing, d, are required to make the training data stationary by doing the following:

- Calculate the seasonal strength, F_s, on the training data. Report the value and interpret it in plain English.
- If $F_s > 0.64$, seasonally difference (using `lag = 12`) the training data and calculate F_s again. Determine if you need to seasonally difference again. If necessary, continue this process until you have found an appropriate order of seasonal differencing.
- Verify the order of seasonal differencing using the `unitroot_nsdiffs` feature on your training data. Report D.
- Plot your seasonally differenced training data and comment on whether you believe the time series has constant mean over time.
- Perform a KPSS unit root test on your seasonally differenced training data to determine if this series is stationary. Report your p-value and interpret it in plain English. (If your time series requires first differencing, difference it and repeat the KPSS test on the differenced series until stationarity is satistfied).
- Verify the order of first differencing using the `unitroot_ndiffs` feature on your seasonally differenced training data. Report d.

Hide

```
#obtain seasonal strength
train_temp %>% features(Temperature, feat_stl)
```

```
## # A tibble: 1 × 9
##   trend_strength seasonal_strength_year seasonal_peak_year seasonal_trough_year
##            <dbl>                  <dbl>              <dbl>                <dbl>
## 1          0.341                  0.965                  8                    1
## # ℹ 5 more variables: spikiness <dbl>, linearity <dbl>, curvature <dbl>,
## #   stl_e_acf1 <dbl>, stl_e_acf10 <dbl>
```

Hide

```
#seasonally difference data and find seasonal strength
train_temp %>% mutate(d_temp = difference(Temperature, 12)) %>%
  features(d_temp, feat_stl)
```

```
## # A tibble: 1 × 9
##   trend_strength seasonal_strength_year seasonal_peak_year seasonal_trough_year
##            <dbl>                  <dbl>              <dbl>                <dbl>
## 1          0.260                 0.0151                  0                   10
## # ℹ 5 more variables: spikiness <dbl>, linearity <dbl>, curvature <dbl>,
## #   stl_e_acf1 <dbl>, stl_e_acf10 <dbl>
```

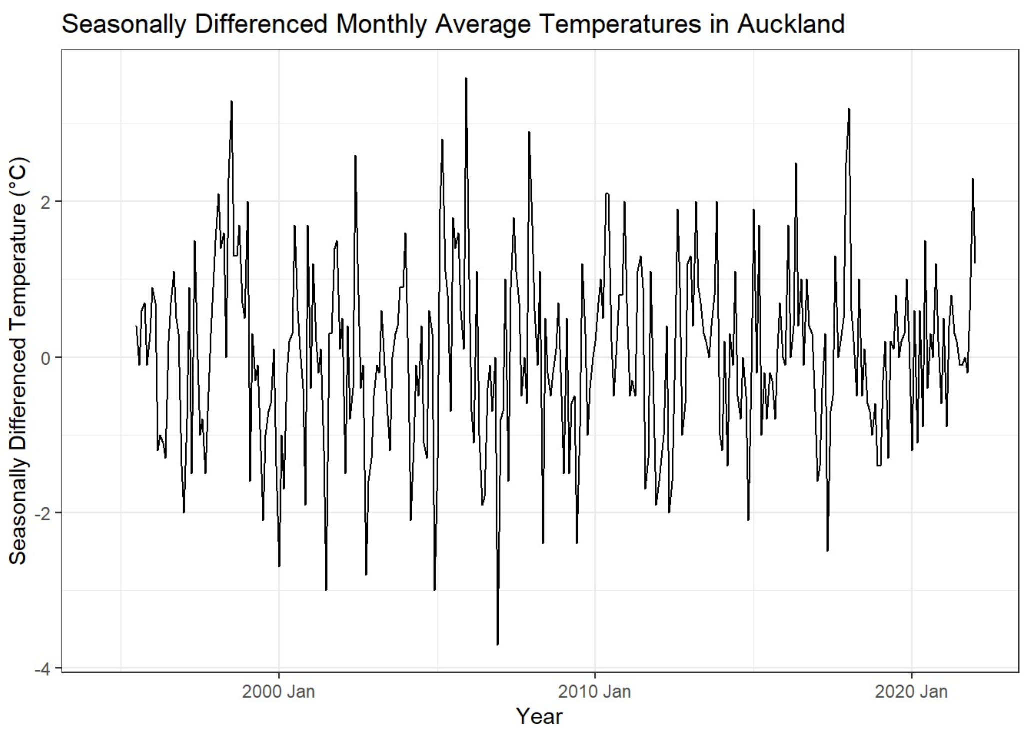
Hide

```
#verify order of seasonal differencing
train_temp %>%
  features(Temperature, unitroot_nsdiffs)
```

```
## # A tibble: 1 × 1
##   nsdiffs
##     <int>
## 1       1
```

Hide

```
#plot seasonally differenced temperature series
train_temp %>% mutate(d_temp = difference(Temperature, 12)) %>%
  autoplot(d_temp) +
  ggtitle("Seasonally Differenced Monthly Average Temperatures in Auckland") +
  xlab("Year") + ylab("Seasonally Differenced Temperature (\u00B0C)") +
  theme_bw()
```



Hide

```
#KPSS test
train_temp %>% mutate(d_temp = difference(Temperature, 12)) %>%
  features(d_temp, unitroot_kpss)
```

```
## # A tibble: 1 × 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0361         0.1
```

Hide

```
#verify order of first differencing for seasonally differenced temperatures
train_temp %>% mutate(d_temp = difference(Temperature, 12)) %>%
  features(d_temp, unitroot_nsdiffs)
```

```
## # A tibble: 1 × 1
##   nsdiffs
##     <int>
## 1       0
```

F_s is used to measure how strong the seasonality is in our data. As we obtained a seasonal strength value of $F_s = 0.965$, this indicates very strong seasonal effects in our temperature time series, hence one seasonal difference should be sufficient.

After applying a seasonal difference using lag = 12, we obtained a very low seasonal strength value of $F_s = 0.015$, this means applying seasonal difference once has effectively removed seasonality in our temperature time series.

As we obtained a value of $D = 1$ for the order of seasonal differencing, this suggests we should apply a seasonal difference once.

Our seasonally differenced temperature series appears to have reasonably constant variace and the mean also appears to be reasonably constant at zero.

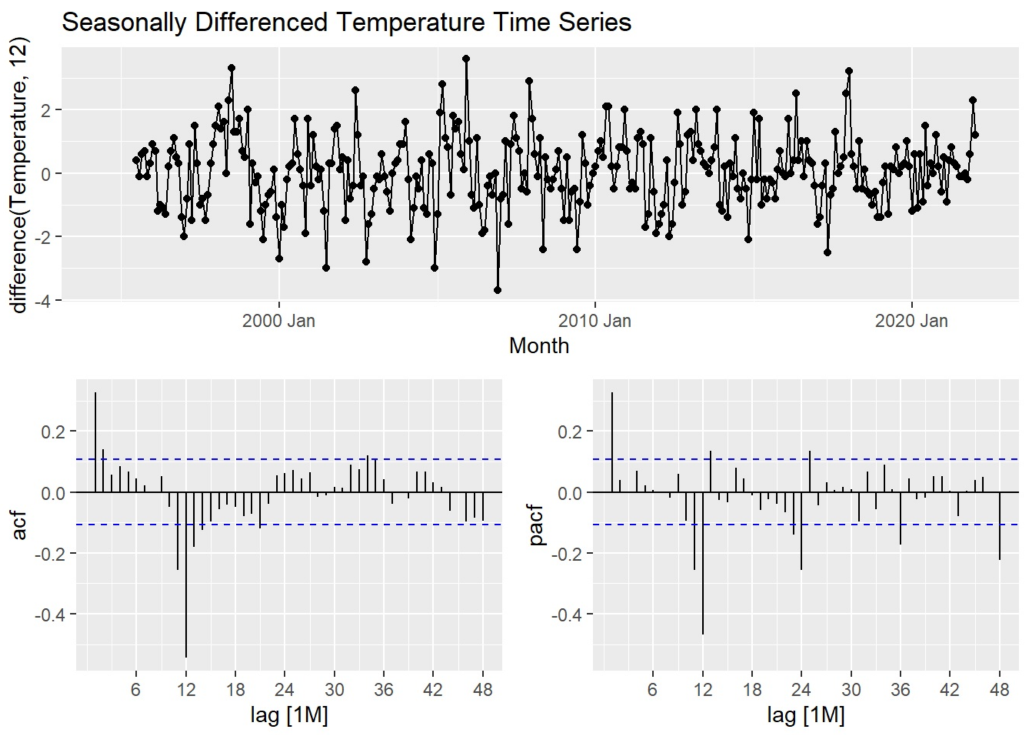
In the KPSS unit root test, we obtained a large p-value of 0.1, this means we do not have sufficient evidence to reject the null hypothesis that the seasonally differenced temperature series is stationary and non-seasonal. After applying a seasonal difference once, our data is independent of time and requires no further differencing.

In testing for the number of first order differencing for our seasonally differenced temperature series, we obtained a value of $d = 0$ order which suggests that no first differencing is required on our data.

> 4. 11 Marks Plot the ACF and PACF plots on your differenced training data (you may use the `gg_tsdisplay` function). Set `lag_max = 48`. Analyse these plots to come up with four candidate ARIMA models. Explain why you have chosen these ARIMA models.

Hide

```
#ACF and PACF plot for seasonally differenced temperature data
train_temp %>%
  gg_tsdisplay(difference(Temperature, 12), plot_type = 'partial', lag_max =48) +
  ggtitle("Seasonally Differenced Temperature Time Series")
```



Both the ACF and PACF plots shows a dampening sinusoidal wave. In determining the non-seasonal part of the seasonal ARIMA model, we firstly noted that it required no first order differencing, hence d = 0. In our ACF plot, there appears to be two significant spikes at lag 1 and 2 and none after that before the seasonal frequency at lag 12, hence q = 2 non-seasonal MA terms. Lag 11 is ignored. Similarly, in our PACF aplot, there appears to be a significant spike at lag 1 and none after that before the seasonal frequency at lag 12, hence p = 1 non-seasonal AR terms. Therefore, the two candidate non-seasonal parts of our ARIMA models are (0,0,2) and (1,0,0).

In determining the seasonal part of the seasonal ARIMA model, we firstly noted that it required a seasonal differencing once to obtain a stationary time series, hence D = 1. In our ACF plot, there appears to be a significant spike at only one seasonal freqency at lag 12, hence Q = 1 seasonal MA term. In our PACF plot, there appears to be significant spikes at lags 12, 24, 26 and 48 which are the seasonal frequencies, it reaches a limit at the fourth harmonic in lag 48, hence we require P = 4 seasonal AR terms. Therefore, the two candidate seasonal parts of our ARIMA models are (0,1,1) and (4,1,0)

Our four candidate seasonal ARIMA models are ARIMA(0,0,2)(0,1,1)_12, ARIMA(1,0,0)(0,1,1)_12, ARIMA(0,0,2)(4,1,0)_12 and ARIMA(1,0,0)(4,1,0)_12.

5. 8 Marks Fit your four candidate ARIMA models, as well as the automatic stepwise search model (noting that `stepwise = FALSE` will take too long to run) on the training data. Compare the models using AICc and output the parameter estimates from the model that has the best predictive ability. Write the fitted model equation using backshift notation. Note: If you get warnings in your code, you may have to include a constant in your model. You can do this by starting your model fit with `Temperature ~ 1 +`.

Hide

```
#fit seasonal ARIMA models
temp_fit <- train_temp %>%
  model(
    arima002011 = ARIMA(Temperature ~ 1 + pdq(0,0,2) + PDQ(0,1,1)),
    arima100011 = ARIMA(Temperature ~ 1 + pdq(1,0,0) + PDQ(0,1,1)),
    arima002410 = ARIMA(Temperature ~ 1 + pdq(0,0,2) + PDQ(4,1,0)),
    arima100410 = ARIMA(Temperature ~ 1 + pdq(1,0,0) + PDQ(4,1,0)),
    auto = ARIMA(Temperature, approximation = FALSE)
  )
```

```
## Warning in sqrt(diag(best$var.coef)): NaNs produced
```

Hide

```
#compare AICc values
glance(temp_fit) %>% select(.model:BIC)
```

```
## # A tibble: 5 × 6
##   .model      sigma2 log_lik   AIC  AICc   BIC
##   <chr>        <dbl>   <dbl> <dbl> <dbl> <dbl>
## 1 arima002011  0.586   -385.  780.  781.  799.
## 2 arima100011  0.586   -386.  780.  780.  795.
## 3 arima002410  0.707   -399.  814.  815.  845.
## 4 arima100410  0.708   -400.  814.  814.  840.
## 5 auto         0.714   -407.  819.  819.  826.
```

Hide

```
#fit best seasonal ARIMA model and find parameter estimates
arima100011 <- train_temp %>%
  model(ARIMA(Temperature ~ 1 + pdq(1,0,0) + PDQ(0,1,1)))

arima100011 %>% report()
```

```
## Series: Temperature
## Model: ARIMA(1,0,0)(0,1,1)[12] w/ drift
##
## Coefficients:
##          ar1     sma1  constant
##       0.2907  -1.0000    0.0180
## s.e.  0.0536   0.0622    0.0052
##
## sigma^2 estimated as 0.5863:  log likelihood=-385.92
## AIC=779.85   AICc=779.97   BIC=794.91
```

In observing our AICc values, our automatic ARIMA model has the highest AICc value of 818.74, this suggests the least accurate fit for our temperature time series. Our third and fourth ARIMA models, arima002410 and arima100410 have got similar AICc values of 814.96 and 814.23, respectively, which are lower than the AICc value of the automatic model. This suggests they have a slightly more accurate fit. Our first and second ARIMA models, arima002011 and arima100011 have an AICc value of 780.67 and 779.97, respectively, which are substantially lower than the AICc values of the other ARIMA models. This suggests they have a much more accurate than the other models. In comparing these two, our best model is arima100011 [ARIMA(1,0,0)(0,1,1)_12] which has the lowest AICc value of 779.97. This model provides the most accurate fit to our temperature time series.

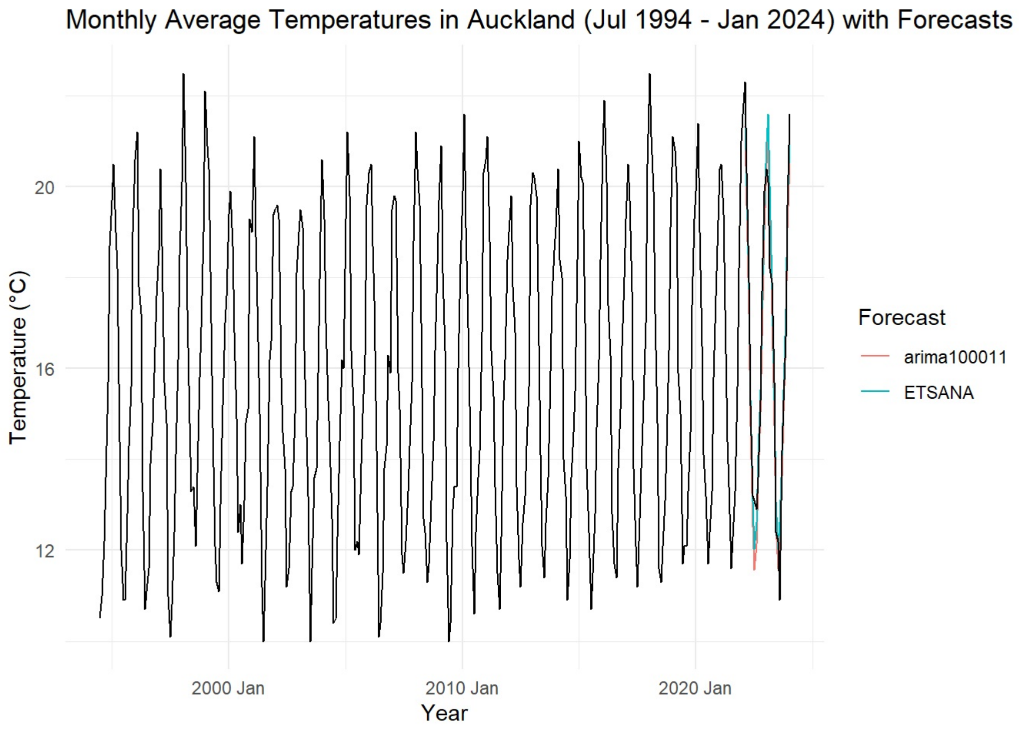
$$(1 - \phi_1 B)(1 - B^{12})(1 - B)y_t = c + (1 + \Theta_1 B^{12})\varepsilon_t(1 - 0.2907B)(1 - B^{12})(1 - B)y_t = 0.018 + (1 - B^{12})\varepsilon_t$$

6. 6 Marks Fit your best ETS and ARIMA model on the training data in the same `mable`. Forecast $h = 24$ periods into the future. Plot your forecasts. Compare your forecasts against the truth by computing RMSE, MAE, MAPE, and MASE. State which model has the best forecast accuracy.

Hide

```
#Fit ETS and ARIMA models, and forcast 24 periods
temp.fc <- train_temp %>%
  model(
    ETSANA = ETS(Temperature ~ error("A") + trend("N") + season("A")),
    arima100011 = ARIMA(Temperature ~ 1 + pdq(1,0,0) + PDQ(0,1,1))
    ) %>% forecast(h = 24)

#Plot forecasts
temp.fc %>%
  autoplot(data_temp, level = NULL) +
  xlab("Year")+ ylab("Temperature (\u00B0C)") +
  ggtitle("Monthly Average Temperatures in Auckland (Jul 1994 - Jan 2024) with Forecasts") +
  guides(colour=guide_legend(title = "Forecast")) +
  theme_minimal()
```



Hide

```
#Compare forecast accuracy
temp.fc.measures <- temp.fc %>%
  accuracy(data_temp) %>%
  select(.model, RMSE, MAE, MAPE, MASE)

temp.fc.measures
```

```
## # A tibble: 2 × 5
##   .model       RMSE   MAE  MAPE  MASE
##   <chr>       <dbl> <dbl> <dbl> <dbl>
## 1 ETSANA      0.728 0.530  3.39 0.589
## 2 arima100011 0.791 0.681  4.27 0.756
```

It appears that our ETS model, ETS(A,N,A), scored the lowest forecast errors across all four metrics. For our ETS model, the RMSE, MAE, MAPE and MASE forecast errors are respectively, 0.728, 0.530, 3.388 and 0.589, which in comparison are lower than the forecast errors for our ARIMA model. For our ARIMA model, ARIMA(1,0,0)(0,1,1)_12, the RMSE, MAE, MAPE and MASE forecast errors are respectively 0.791, 0.681, 4.269 and 0.756. Therefore, the ETS(A,N,A) has the best accuracy forecasts to our monthly average temperature time series.

Possible marks for **Problem 2**: 46 Marks

Total possible marks for **Assignment 4**: 80 Marks