Assignment for 6510 (Feng Gu T00751197)

Feng Gu

2024-11-08

Contents

0.1 Question 1

0.1.1 To the Mean forecast method:

To use Mean forecast method, we have to set some assumptions as follows:

Mean forecast method always assumes the time series is surrounding with the constant mean value, hence using the mean as the forecast for future value. The model is usually written as:

$$y_{T+h} = \bar{y} + \epsilon_t, \quad \epsilon_t \sim N(0, \sigma^2)$$

Given the observations up to time T, we can use this observed values to estimate the parameters for this model.

$$\bar{y} = E(y_t|T)$$

The estimated mean - $\bar{y}$ should be the mean of the sample.

This estimate has its variance due to the sampling way and the noise from the real world. So we can estimate the variance for the $\bar{y}$.

To the mean statistics, it comes from $\bar{y} = \frac{1}{N}\sum_{t=1}^{T} y_t$, where y_t is individual observation from the whole population.

We assume the ϵ_t that influences each y_t follows the identically independent distribution, we have:

$$y_t = \hat{y}_t + \epsilon = \mu_y + \epsilon_t$$

Hence, the variance of $\bar{y}$ can be expressed as:

$$\begin{aligned} Var(\bar{y}) &= \frac{1}{N} Var(\sum_{i=1}^{T} y_i) \\ &= \frac{1}{T^2}[\sum^{T} Var(y_i) + 2\sum_{j}^{T}\sum_{i}^{T} Cov(y_i, y_j)] \\ &= \frac{1}{N} Var(y_i) \\ &= \frac{\sigma^2}{N} \end{aligned}$$

Now, we use the observations to calculate the estimated mean value as the forecast and we have to add one more ϵ to that individual forecast because it is one time observing like the other $y_t's$.

$$\begin{aligned} \hat{y}_{T+h} &= \bar{y}_{|T} \\ y_{T+h} &= \bar{y}_{|T} + \epsilon \end{aligned}$$

The $\bar{y}_{|T}$ represents the mean calculated from the given T period observations.

Hence, the variance of T+h forecast is:

$$Var(y_{T+h}) = Var(\bar{y}_T + \epsilon) = (1 + \frac{1}{N})\hat{\sigma}^2$$

0.1.2 To the Naive forecast method

The Naive model is:

$$\begin{aligned} y_{T+h} &= y_{T+h-1} + \epsilon_{T+h} \\ &... \\ y_{T+1} &= y_T + \epsilon_{T+1} \end{aligned}$$

So, the closed expression for y_{T+h} is:

$$y_{T+h} = y_T + \sum_{i=1}^{h} \epsilon_i$$

where $\epsilon_i \sim N(0, \sigma^2)$ to all $i's$.

Hence:

$$\begin{aligned} &\text{Var}(y_{T+h}) \\ &= \text{Var}\left(y_T + \sum_{i=1}^{h} \epsilon_i\right) \\ &= \text{Var}(y_T) + \text{Var}\left(\sum_{i=1}^{h} \epsilon_i\right) \\ &= \sum_{i=1}^{h} \text{Var}(\epsilon_i) + 2\sum_{i=1}^{h}\sum_{j=1}^{h} \text{Cov}(\epsilon_i, \epsilon_j) \\ &= h\sigma^2 \end{aligned}$$

Finally, the standard variance of y_{T+h} would be $\sqrt{h\sigma^2}$.

0.1.3 To the Seasonal Naive method

The model expression is:

$$\hat{y}_{T+h} = y_{T+h-m(k+1)}$$

where m is the seasonal period and k is the integer part in $(h-1)/m$.

So we can have the expressions as follows:

$$\begin{aligned} y_{T+h} &= y_{T+h-m(k+1)} + \epsilon_{T+h} \\ &... \\ y_{T+1} &= y_{T+1-m} + \epsilon_{T+1} \end{aligned}$$

The closed form to expression y_{T+h} should be:

$$y_{T+h} = y_{T+h-m(k+1)} + \sum_{t=1}^{k+1} \epsilon_t$$

$$
\begin{aligned}
&Var(y_{T+h}) \\
&= Var(\sum_{t=1}^{k+1} \epsilon_t) \\
&= \sum_{t=1}^{k+1} Var(\epsilon_t) + 0 \\
&= (k+1)\sigma^2
\end{aligned}
$$

The standard variance for $y_{T+h} in $Seasonal Naive method is $\sqrt{k+1}\sigma$.

0.2 Question 2

0.2.1 Unit 3 exercise on # 57

To get the estimate β in least squares error method, we use the summation of squared errors as the criterion to optimize the parameter β. Hence, we write the criterion in a function form about *beta*:

$$\sum_{i=1}^{T} \epsilon_t^2 = \sum_{t=1}^{T} (y_t - \beta x_t)^2$$

Taking the derivative with the respect to *beta* and making it equal to 0, we have:

$$\frac{\partial \sum \epsilon_t^2}{\partial \beta} = 2(\sum_{t=1}^{T} (y_t x_t) + \beta \sum_{t=1}^{T} x_t^2) = 0 \beta = \frac{\sum^T y_t x_t}{\sum^T x_t^2}$$

0.2.1.1 To the variance of $\hat{\beta}$

The estimated $\hat{\beta}$ can be written as:

$$\hat{\beta} = \frac{\sum^T (\hat{y}_t + \epsilon_t) x_t}{\sum^T x_t^2} = \frac{\sum^T \hat{y}_t x_t + \sum^T \epsilon_t x_t}{\sum x_t^2}$$

Hence, the variance of $\hat{\beta}$ can be written as:

$$
\begin{aligned}
&Var(\hat{\beta}) \\
&= Var(\frac{\sum^T \hat{y}_t x_t + \sum^T \epsilon_t x_t}{\sum x_t^2}) \\
&= \frac{Var(\sum \epsilon_t x_t)}{\sum x_t^2} \\
&= \frac{\sum x_t^2 Var(\epsilon_t)}{\sum x_t^2} \\
&= \frac{\sigma^2 \sum x_t^2}{(\sum x_t^2)^2} \\
&= \frac{\sigma^2}{\sum x_t^2}
\end{aligned}
$$

Hence, the variance of $\hat{\beta}$ is $\frac{\sigma^2}{\sum x_t^2}$.

0.2.2 Unit4 Part exercise on # 40

Given this seasonal MA(1) model, we can check the three properties on this to decide if the series generated by this mode is a stationary time series.

1. Constant mean

$$E(Z_t - \mu) = E(a_t - \theta a_{t-12}) = E(a_t) - \theta E(a_{t-12}) = 0$$

where, the $a_t's$ are all 'iid', $a_t \sim N(0, \sigma^2)$.

2. Constant variance

$$Var(Z_t - \mu) = Var(a_t - \theta a_{t-12}) = (1 + \theta^2)\sigma^2$$

3. Autocorrelation only matters with $|t - s|$

Take the covariance between $Z_t - \mu$ and $Z_{t-s} - \mu$, we have:

$$\begin{aligned} &Cov(Z_t - \mu, Z_{t-s} - \mu) \\ &= Cov(a_t, a_{t-s}) - \theta Cov(a_t, a_{t-12-s}) \\ &- \theta Cov(a_{t-12}, a_{t-s}) + \theta^2 Cov(a_{t-12}, a_{t-12-s}) \end{aligned}$$

Denote γ_k as the Covariance between y_t and y_{t-k}/t_{t+k}, then:

$$\gamma_k = \begin{cases} (1 + \theta^2)\sigma^2, & \text{if } s = 0 \\ -\theta\sigma^2, & \text{if } |s| = 12 \end{cases}$$

0.2.2.1 Show the autocorrelations are the same Because we have calculated the γ_k for the first form, we can convert all θ to $\frac{1}{\theta}$ to get the second form.

$$\begin{aligned} &Cov(Z_t - \mu, Z_{t-s} - \mu) \\ &= Cov(a_t, a_{t-s}) - \frac{1}{\theta} Cov(a_t, a_{t-12-s}) \\ &- \frac{1}{\theta} Cov(a_{t-12}, a_{t-s}) + \frac{1}{\theta^2} Cov(a_{t-12}, a_{t-12-s}) \end{aligned}$$

Then, we got:

$$\gamma_k = \begin{cases} (1 + \frac{1}{\theta^2})\sigma^2, & \text{if } s = 0 \\ -\frac{1}{\theta}\sigma^2, & \text{if } |s| = 12 \end{cases}$$

Now, calculate the correlation of ρ_k for this second one, we have:

$$\rho_k = -\frac{\sigma^2}{\theta} \frac{\theta^2}{(\theta^2 + 1)\sigma^2} = -\frac{\theta}{\theta^2 + 1}$$

If we calculate the ρ_k for the first one, where the coefficient is θ, we have:

$$\rho_k = \frac{-\theta\sigma^2}{(1 + \theta^2)\sigma^2} = \frac{-\theta}{1 + \theta^2}$$

which is the same as the second one.

0.2.3 Unit4 Part A exercise on # 63

To this AR(2) model, we can multiply both sides with y_{t-k} and take expectation on them, assuming the series have been centered.

$$y_{t-k}y_t = \phi_{21}y_{t-k}y_{t-1} + \phi_{22}y_{t-k}y_{t-2} + y_{t-k}a_t$$

take $E(.)$ on both sides, we have:

$$\gamma_k = \begin{cases} \phi_{21}y_{k-1} + \phi_{22}y_{k-2} + 0, \text{if } k > 0 \\ \phi_{21}y_{k-1} + \phi_{22}y_{k-2} + \sigma^2, \text{if } k = 0 \end{cases}$$

Take k = 1 and 2 into the expression, we have:

$$\gamma_1 = \phi_{21}\gamma_0 + \phi_{22}\gamma_1\gamma_2 = \phi_{21}\gamma_1 + \phi_{22}\gamma_0$$

By taking the equal transformation to eliminate γ_0 and having $\rho_k = \frac{\gamma_k}{\gamma_0}$, we can finally get the expression for ϕ_{22}:

$$\phi_{22} = \frac{\rho_2 - \rho_1^2}{1 - \rho_1^2}$$

0.2.4 Unit4 Part A exercise on # 80 and # 81

0.2.4.1 exercise on #80

0.2.4.2 (a) Multiply both sides with Z_k, we have:

$$Z_tZ_k = \phi_1Z_{t-1}Z_k + \phi_2Z_{t-2}Z_k + Z_ka_t$$

Take expectation on both sides:

$$\gamma_{|t-k|} = \phi_1\gamma_{|t-k-1|} + \phi_2\gamma_{|t-k-2|} + E(Z_ka_t)$$

where:

$$E(Z_ka_t) = \begin{cases} \sigma^2, k = t \\ 0, k \neq t \end{cases}$$

Now, take $k = t - 1$ into the expression, we have:

$$\gamma_1 = \phi_1\gamma_0 + \phi_2\gamma_1$$

Divide both sides with γ_0 and simplify, we have:

$$\rho_1 = \frac{\phi_1}{1 - \phi_2}$$

0.2.4.3 (b) Divided by γ_0 to the expression for $\gamma_{|t-k|}$, we get (assuming $k = t - k$):

$$\rho_k = \phi_1\rho_{k-1} + \phi_2\rho_{k-2}\phi_1 = \frac{1}{7}, \quad \phi_2 = \frac{2}{49}$$

The question is how to express ρ_k with a closed form.

To solve this, we use the characteristic equation for this AR model.

$$1 - \phi_1 x - \phi_2 x^2 = 0.$$

The roots of x are the components about r_i in the closed form:

$$\rho_k = Ar_1^k + (1 - A)r_2^k r_1 = \frac{1}{x_1}, r_2 = \frac{1}{x_2}$$

Solve the $x's$, we got roots:

$$x_1 = \frac{7}{2}, \quad x_2 = -7r_1 = \frac{2}{7}, \quad r_2 = -\frac{1}{7}$$

Hence, we have:

$$\rho_k = A\left(\frac{2}{7}\right)^k + (1 - A)\left(-\frac{1}{7}\right)^k.$$

(c)

Because $\rho_0 = 1, \rho_1 = \frac{\phi_1}{1-\phi_2} = \frac{7}{47}$, we have:

$$1 = A + (1 - A)\frac{7}{47} = A \cdot \frac{2}{7} + (1 - A) \cdot \left(-\frac{1}{7}\right).$$

We got:

$$A = \frac{32}{47}$$

To get the $Var(Z_t)$, when we take $k = t$ into this equation, dividing both sides with γ_0 and take the known ρ_1, ρ_2, we would get the exrpession for σ^2

$$\gamma_{|t-k|} = \phi_1\gamma_{|t-k-1|} + \phi_2\gamma_{|t-k-2|} + E(Z_k a_t)$$

Given $\rho_1 = 0.1489, \sigma_a^2 = 5$, we have:

$$\text{Var}(Z_t) = \frac{\sigma_a^2}{1 - \phi_1^2 - \phi_2^2 - 2\phi_1\phi_2\rho_1} \approx 5.121$$

To calculate ρ_10, we just take $k = 10$ with the closed expression for ρ_k:

$$\rho_{10} = \frac{32}{47}\left(\frac{2}{7}\right)^{10} + \left(1 - \frac{32}{47}\right)\left(-\frac{1}{7}\right)^{10} \approx 2.468 \times 10^{-6}.$$

exercise on #81

All right, this is almost the total same question as the former one! The only thing we need to notice is that the $|t - k|$ should be 12 and 24 to get non-zero covariance. And the all following work are expanded based on this change.

0.2.4.4 (a)

$$\rho_k = \phi_1 \rho_{k-12} + \phi_2 \rho_{k-24}, \quad k = 12, 24, ... \phi_1 = \frac{1}{7}, \quad \phi_2 = \frac{2}{49}$$

So, $\rho_{12} = \frac{\Phi_1}{1-\Phi_2} = 0.1489$.

0.2.4.5 (b)

To find the close form of ρ_k, solve the characteristic equation:

$$x^2 - \frac{1}{7}x - \frac{2}{49} = 0 r_1 = \frac{1}{x_1}, r_2 = \frac{1}{x_2} r_1 = \frac{2}{7}, \quad r_2 = -\frac{1}{7}$$

$$\rho_k = A\left(\frac{2}{7}\right)^{k/12} + (1-A)\left(-\frac{1}{7}\right)^{k/12}$$

(c)

Repeat the same steps to find $A, Var(Z_t), \rho_{60}$.

$$\rho_1 = Ar_1 + (1-A)r_2 A = \frac{32}{47}$$

$$\text{Var}(Z_t) = \frac{\sigma_a^2}{1 - \Phi_1^2 - \Phi_2^2} \approx 5.113$$

To calculate ρ_{60}, take $k = 60$ into the $\rho_k = A\left(\frac{2}{7}\right)^{k/12} + (1-A)\left(-\frac{1}{7}\right)^{k/12}$, where $A = \frac{32}{47}$.

$$\rho_{60} = \frac{32}{47}\left(\frac{2}{7}\right)^5 + (1 - \frac{32}{47})\left(-\frac{1}{7}\right)^5 \approx 0.00128$$

0.3 Question 3

0.3.1 Summary

In conclusion, i first checked the trend and stationarity of this series, and made 4 transformations on the original data and did differ on them separately. Then, i built 4 benchmark models on the transformed data and two ARIMA models on the transformed-differed data.

After comparing some criterion about the model performance and the complex in explaining the results, i choose the ARIMA(1,1,4) model that was built on the centered-differed data.

Finally, i forecasted the future 5 differences, accumulated them on top of the last observation and added the mean value back to get the future 5 forecasts.

0.3.2 Exact steps

First, convert the data into a tsibble object in R.

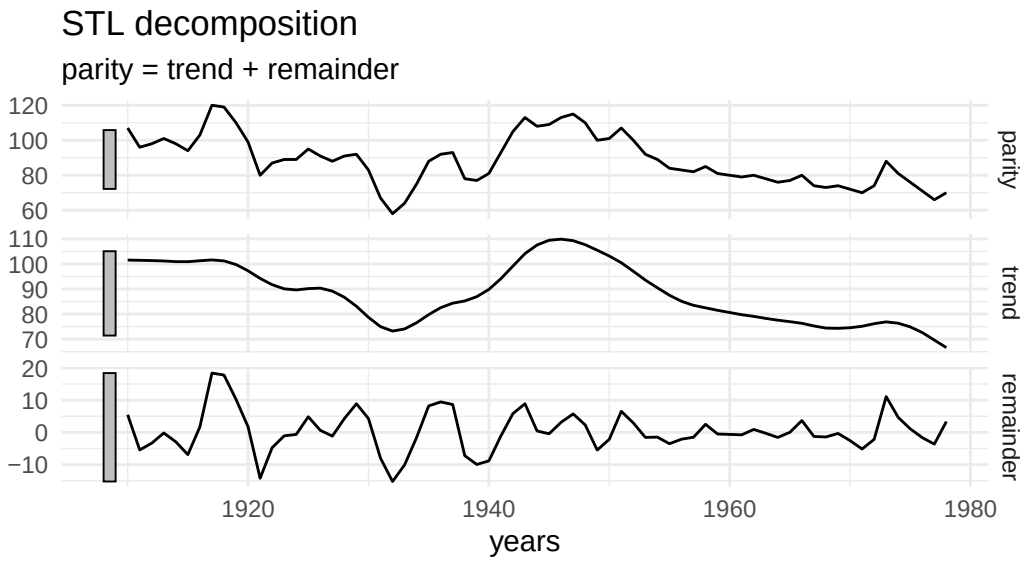
```
# Create the year index
years <-  seq(as.Date("1910-01-01"), as.Date("1978-01-01"), by = "year")

# Create the data as a tsibble
farm_parity <- c(
  107, 96, 98, 101, 98, 94, 103, 120, 119, 110,
  99, 80, 87, 89, 89, 95, 91, 88, 91, 92,
  83, 67, 58, 64, 75, 88, 92, 93, 78, 77,
```

```
  81, 93, 105, 113, 108, 109, 113, 115, 110, 100,
  101, 107, 100, 92, 89, 84, 83, 82, 85, 81,
  80, 79, 80, 78, 76, 77, 80, 74, 73, 74,
  72, 70, 74, 88, 81, 76, 71, 66, 70
)

# Extract the year component in the date object
parity_tibble <- tibble(years = year(years), parity = farm_parity)

# convert the tibble to tsibble
parity_tsibble <- parity_tibble |> as_tsibble(index = years)

# check the result
head(parity_tsibble)
```

```
## # A tsibble: 6 x 2 [1Y]
##   years parity
##   <dbl>  <dbl>
## 1  1910    107
## 2  1911     96
## 3  1912     98
## 4  1913    101
## 5  1914     98
## 6  1915     94
```

Use STL to decompose the original data.

```
parity_tsibble |>
  model(
    STL(parity ~ trend(window = 7) +
          season(window = "periodic"), # the annual data does have an explicit season to be appointed
          robust = TRUE)
  ) |>
  components() |>
  autoplot() +
  theme_minimal()
```



From the plot, there is no seasonarity in this time series. But we can not determine whether or not the series having a trend and being stationary.

To this two features, we can use ACF and PACF and other statistics to judge them. *Here, i want to use the STL features method from the reference book section 4.3:*

```
parity_stl_features <- parity_tsibble |> features(parity, feat_stl)
print(parity_stl_features)
```

```
## # A tibble: 1 x 6
##   trend_strength spikiness linearity curvature stl_e_acf1 stl_e_acf10
##            <dbl>     <dbl>     <dbl>     <dbl>      <dbl>       <dbl>
## 1          0.814     0.994     -56.9     -19.4      0.503       0.802
```

The trend_strength is 0.8140, which is a relatively high value in range $[0, 1]$, so we have reasons to believe this series has a trend inside itself.

To check the stationarity of this series, we can use KPSS to test the property.

```
parity_tsibble |> features(parity, unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.571      0.0256
```

The p-value of kpss is 0.0256 (<0.05), hence we can reject the null hypothesis: "the time series is stationary". It means we have to do some transformation work to make it a stationary model if we want to build ARIMA model on it.

We have known that the series has trend and is non-stationary, the following steps would be building and comparing different models. In order to build these models properly, we should transform the data to make them good in statistical properties.

```
# Transform the data
# 1. Center the data
parity_tsibble <-  parity_tsibble |>
  mutate(centered_parity = parity - mean(parity))

# 2. Do box-cox transformation to stabilize the variance over time
parity_cox_box_lambda <- parity_tsibble |>
  features(centered_parity, guerrero)

parity_tsibble <- parity_tsibble |>
  mutate(
    centered_bx_parity = box_cox(centered_parity, parity_cox_box_lambda[1,1]))

# 3. Keep a box-cox transformation on the uncentered series
parity_ori_cox_box_lambda <- parity_tsibble |> features(parity, guerrero)

parity_tsibble <- parity_tsibble |>
  mutate(
  bx_parity = box_cox(parity, parity_ori_cox_box_lambda[1,1])
)

# 4. Pivot the tsibble longer to make the following modeling work easier
parity_tsibble <- parity_tsibble |>
  pivot_longer(
    cols = c(parity, centered_parity, centered_bx_parity, bx_parity),
    names_to = "transform_type",
    values_to = "values"
  )
# Convert to a tsibble with a new key (e.g., `new_key_column`)
parity_tsibble <- as_tsibble(
  parity_tsibble,
  index = years,
  key = transform_type)
```

Now, we build models on these differently transformed series, here i choose an ARIMA model with p,d,q values chosen by myself and compare it with benchmark method - mean, naive and drift.

```
# Applying ARIMA model needs the series stationary
# take difference one the original data to make it stationary
parity_tsibble |>
  filter(transform_type == "centered_bx_parity") |>
  features(difference(values), unitroot_kpss)
```

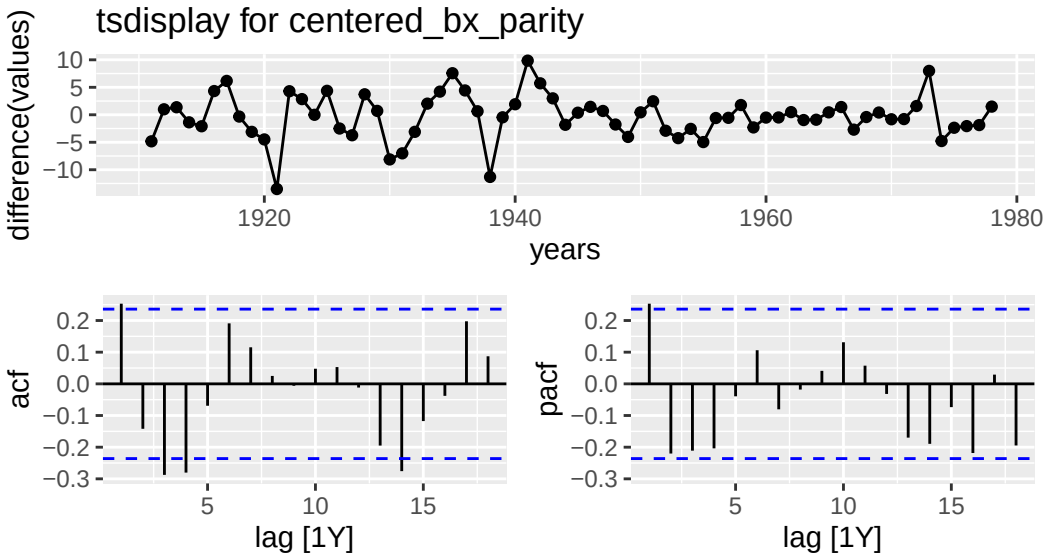
```
## # A tibble: 1 x 3
##   transform_type     kpss_stat kpss_pvalue
##   <chr>                  <dbl>       <dbl>
## 1 centered_bx_parity    0.0437         0.1
```

The kpss_pvalue is 0.1, which is larger than 0.05, kind of supports the stationarity for this transformed time series.

```
# Draw the ACF and PACF plots to search p, q
parity_tsibble |>
  filter(transform_type == "centered_bx_parity") |>
  gg_tsdisplay(difference(values), plot_type = "partial") +
  labs(title = "tsdisplay for centered_bx_parity")
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_line()`).
```

```
## Warning: Removed 1 row containing missing values or values outside the scale range
## (`geom_point()`).
```



The ACF is significant until after lag 4, the PACF is only significant on lag 1, i will build ARIMA(1,1,4) and ARIMA(1,1,0) and compare them to choose the best.

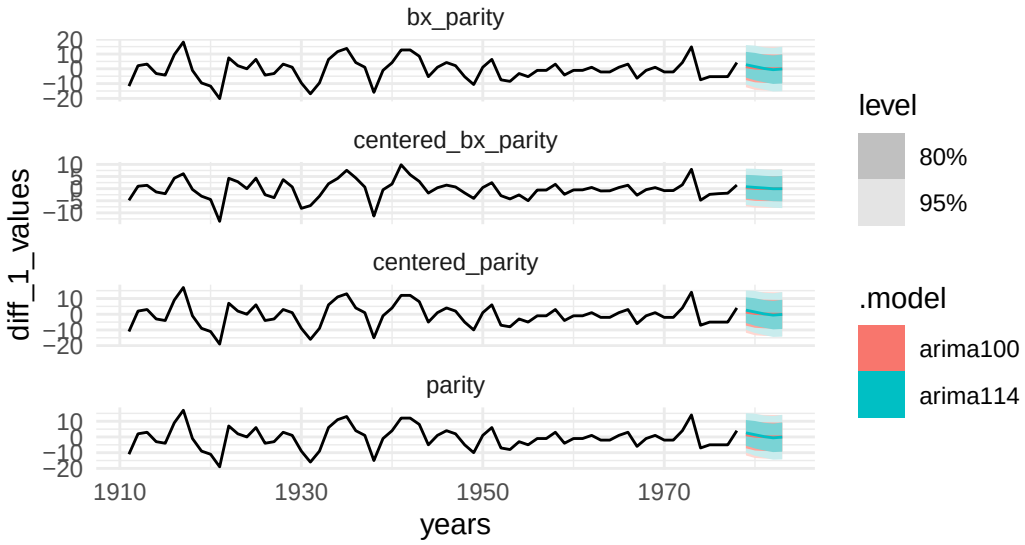
```
# Create the differenced series
parity_tsibble_diff_1 <- parity_tsibble |>
  group_by(transform_type) |>
  mutate(diff_1_values = difference(values)) |>
  filter(!is.na(diff_1_values)) |>
  select(-values)

parity_tsibble_diff_1
```

```
## # A tsibble: 272 x 3 [1Y]
## # Key:       transform_type [4]
## # Groups:    transform_type [4]
##    years transform_type diff_1_values
##    <dbl> <chr>                  <dbl>
##  1  1911 bx_parity             -11.7
##  2  1912 bx_parity               2.12
##  3  1913 bx_parity               3.19
```

```
##  4  1914 bx_parity              -3.19
##  5  1915 bx_parity              -4.25
##  6  1916 bx_parity               9.56
##  7  1917 bx_parity              18.1
##  8  1918 bx_parity              -1.07
##  9  1919 bx_parity              -9.58
## 10  1920 bx_parity             -11.7
## # i 262 more rows
```

```
arima_models <- parity_tsibble_diff_1 |>
  model(
    arima114 = ARIMA(diff_1_values ~ pdq(1,0,4)),
    arima100 = ARIMA(diff_1_values ~ pdq(1,0,0))
  )

forecasts_arima <- arima_models |>
  forecast(h = 5)
forecasts_arima |>
  autoplot(parity_tsibble_diff_1) +
  theme_minimal()
```

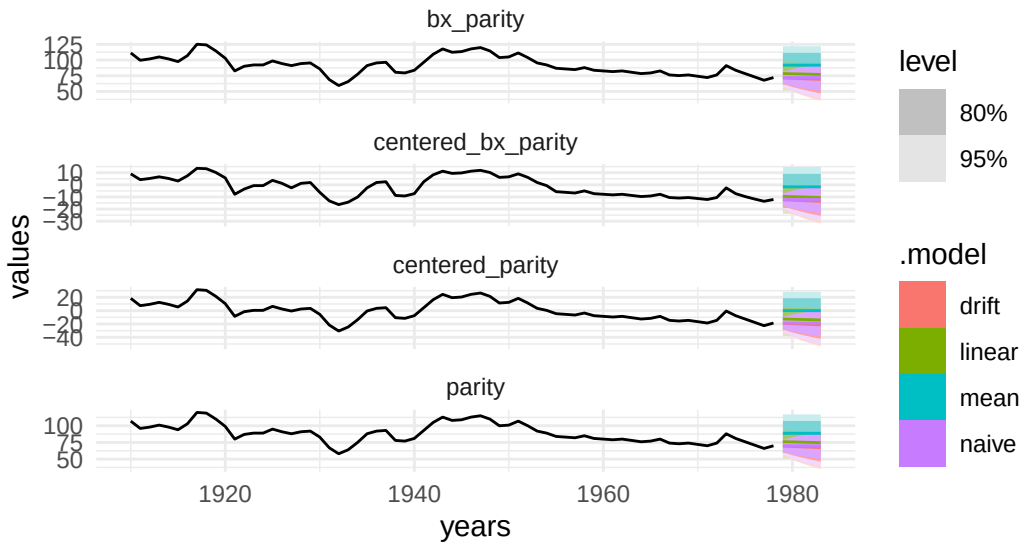
```
## `mutate_if()` ignored the following grouping variables:
## * Column `transform_type`
```



Now, build models on the original data with 4 benchmarks.

```
bench_models <- parity_tsibble |>
  model(
    mean = MEAN(values),
    naive = NAIVE(values),
    drift = RW(values ~ drift()),
    linear = TSLM(values ~ years)
  )
```

```
bench_models |>
  forecast(h = 5) |>
  autoplot(parity_tsibble) +
  theme_minimal()
```



Check some statistics for these models, choose the best one from them:

```
arima_models |> glance() |> select(transform_type, .model, AICc, BIC, sigma2)
```

```
## # A tibble: 8 x 5
##   transform_type     .model    AICc   BIC sigma2
##   <chr>              <chr>    <dbl> <dbl>  <dbl>
## 1 bx_parity          arima114  462.  474.   46.1
## 2 bx_parity          arima100  463.  467.   50.4
## 3 centered_bx_parity arima114  384.  396.   14.6
## 4 centered_bx_parity arima100  381.  386.   15.2
## 5 centered_parity    arima114  454.  466.   40.9
## 6 centered_parity    arima100  455.  459.   44.8
## 7 parity             arima114  454.  466.   40.9
## 8 parity             arima100  455.  459.   44.8
```

```
bench_models |> glance()
```

```
## # A tibble: 16 x 16
##    transform_type .model sigma2 r_squared adj_r_squared statistic  p_value    df
##    <chr>          <chr>   <dbl>     <dbl>         <dbl>     <dbl>    <dbl> <int>
##  1 bx_parity      mean    234.     NA            NA          NA   NA          NA
##  2 bx_parity      naive    56.3    NA            NA          NA   NA          NA
##  3 bx_parity      drift    56.3    NA            NA          NA   NA          NA
##  4 bx_parity      linear  178.      0.247         0.236      22.0  1.38e-5     2
##  5 centered_bx_p~ mean     69.1    NA            NA          NA   NA          NA
##  6 centered_bx_p~ naive    16.2    NA            NA          NA   NA          NA
```

```
##  7 centered_bx_p~ drift    16.2    NA           NA          NA   NA          NA
##  8 centered_bx_p~ linear   49.6     0.293        0.283      27.8  1.56e-6     2
##  9 centered_pari~ mean    207.     NA           NA          NA   NA          NA
## 10 centered_pari~ naive    50.0    NA           NA          NA   NA          NA
## 11 centered_pari~ drift    50.0    NA           NA          NA   NA          NA
## 12 centered_pari~ linear  158.      0.247        0.236      22.0  1.38e-5     2
## 13 parity         mean    207.     NA           NA          NA   NA          NA
## 14 parity         naive    50.0    NA           NA          NA   NA          NA
## 15 parity         drift    50.0    NA           NA          NA   NA          NA
## 16 parity         linear  158.      0.247        0.236      22.0  1.38e-5     2
## # i 8 more variables: log_lik <dbl>, AIC <dbl>, AICc <dbl>, BIC <dbl>,
## #   CV <dbl>, deviance <dbl>, df.residual <int>, rank <int>
```

Overall, ARIMA(1,1,0) built on the centered-transformed data got the best performance, we use this model to forecast the future 5 values and get the reversed original values.

However, the centered box cox transformed data got negative values after the transformation, which made it hard to transform them back to the original positive data.

I can not handle this problem with my current capability, so i choose using ARIMA(1,1,4) on the centered differed parity data to forecast the future differences, and accummulate them on top of the last observation to get forecasts.

```
# take the best to do forecast
arima114 <- arima_models |>
  filter(transform_type == "centered_parity") |>
  select(arima114)

# forecast and convert it to the original values
cumusum_diff_forecasts <- arima114 |>
  forecast(h = 5) |>
  pull(.mean) |>
  cumsum()


last_observation <- parity_tsibble |>
  filter(transform_type == "centered_parity" & years == 1978)

original_forecasts <- cumusum_diff_forecasts +
  last_observation$values +
  mean(parity_tsibble |>
         filter(transform_type == "parity") |>
         pull(values))

original_forecasts
```

```
## [1] 72.60811 74.01537 74.23888 73.68626 73.59427
```

0.4 Question 4

0.4.1 (a) Create a function to generate different AR(1) series

Create the ar1 function:

```
ar1 <- function(phi, n = 100L){
  y <- numeric(n)
  e <- rnorm(n)
  for (i in 2:n){
    y[i] <- phi * y[i - 1] + e[i]
  }
   as_tsibble(
    tibble(step = seq(1, n), y = y),
    index = step
  )
}
```

Now i want to create a function to quickly pivot a tsibble with many column series into a longer tsibble, prepared for the further ploting work.

Now, create a tsibble with different series that come from AR(1) models having different ϕ_{11} parameters.

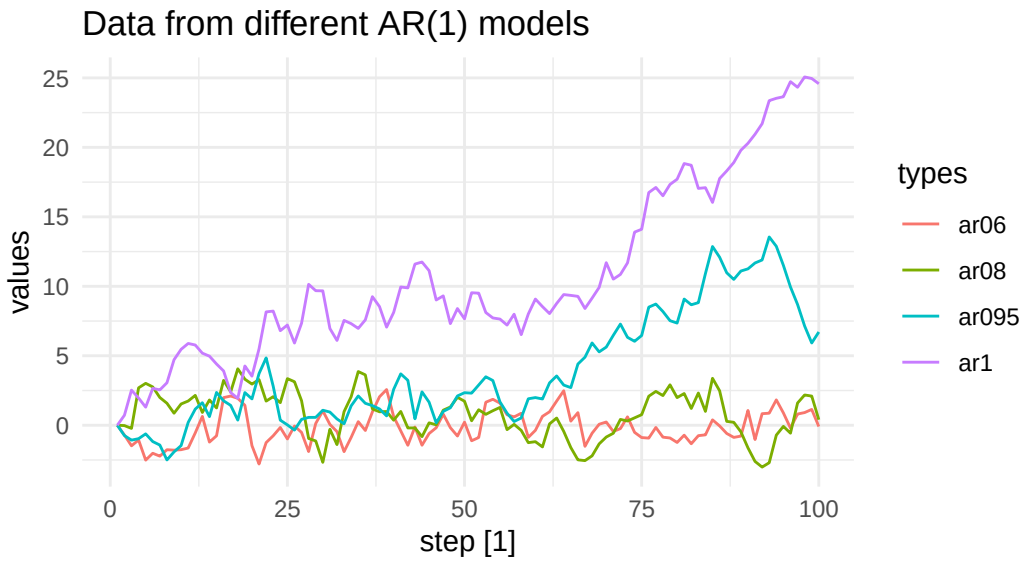
```
# create different series
ar1_values <- tibble(
  step = seq(1:100),
  ar06 = ar1(.6)$y,
  ar08 = ar1(.8)$y,
  ar095 = ar1(.95)$y,
  ar1 = ar1(1)$y,
  ar1.1 = ar1(1.1)$y
) |> as_tsibble(index = step)

# convert the tibble into a longer tsibble
ar1_values <- pivot_longer_series(ar1_values, "step")

# take out the ar1(.6) data
ar1_values |> filter(types == "ar06")
```

```
## # A tsibble: 100 x 3 [1]
## # Key:       types [1]
##     step types values
##    <int> <chr>  <dbl>
##  1     1 ar06   0
##  2     2 ar06  -0.703
##  3     3 ar06  -1.48
##  4     4 ar06  -1.06
##  5     5 ar06  -2.51
##  6     6 ar06  -2.02
##  7     7 ar06  -2.22
##  8     8 ar06  -1.76
##  9     9 ar06  -1.79
## 10    10 ar06  -1.76
## # i 90 more rows
```

###(b) Produce the plots for series that come from different ϕ_{11}

```r
# remove the non-stationary one to make the plot clear
ar1_values |> filter(types != "ar1.1") |>
  autoplot() +
  labs(title = "Data from different AR(1) models") +
  theme_minimal()
```

```
## Plot variable not specified, automatically selected `.vars = values`
```



0.4.2 (c) Write code to generate data from an MA(1) model

Define such a function:

```r
ma1 <- function(theta, n = 100L){
  y <- numeric(n)
  e <- rnorm(n)
  for (i in 2:n){
    y[i] = theta * e[i - 1] + e[i]
  }
  as_tsibble(
    tibble(step = seq(1, n), y = y),
    index = step
  )
}
```

Create a series from MA(1) with $\theta_{11} = .6$.

```r
# create different series
ma1_values <- tibble(
  step = seq(1:100),
  ma06 = ma1(.6)$y,
  ma08 = ma1(.8)$y,
  ma095 = ma1(.95)$y,
```

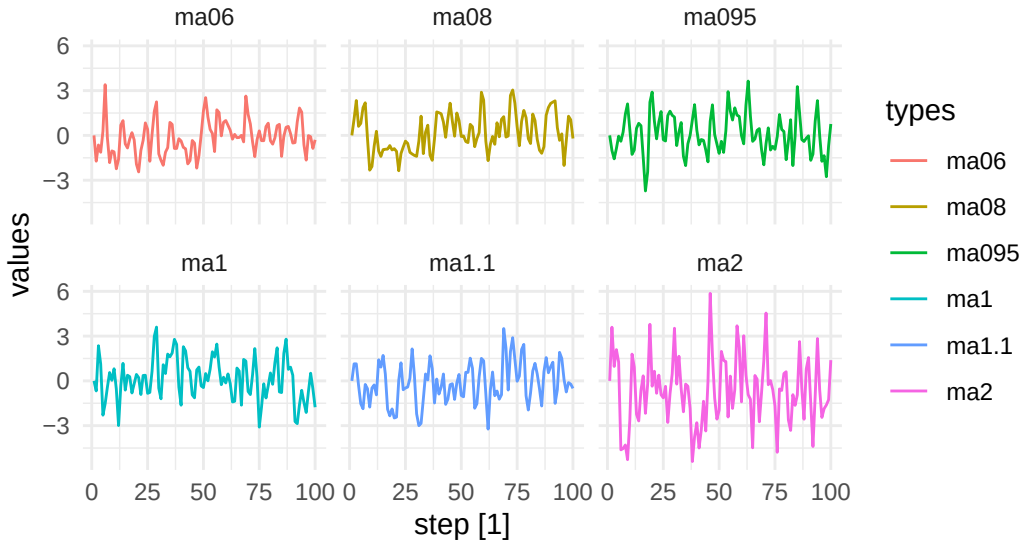
```
  ma1 = ma1(1)$y,
  ma1.1 = ma1(1.1)$y,
  ma2 = ma1(2)$y
) |> as_tsibble(index = step)

# convert the tibble into a longer tsibble
ma1_values <- pivot_longer_series(ma1_values, "step")

# take out the ma1(.6) data
ma1_values |> filter(types == "ma06")
```

```
## # A tsibble: 100 x 3 [1]
## # Key:       types [1]
##     step types values
##    <int> <chr>  <dbl>
##  1     1 ma06   0
##  2     2 ma06  -1.73
##  3     3 ma06  -0.640
##  4     4 ma06  -1.13
##  5     5 ma06   0.276
##  6     6 ma06   3.40
##  7     7 ma06  -0.169
##  8     8 ma06  -1.82
##  9     9 ma06  -1.04
## 10    10 ma06  -1.10
## # i 90 more rows
```

0.4.3 (d) Produce the time plots for different MA(1) models

```
ma1_values |> autoplot() +
  facet_wrap(vars(types)) +
  theme(legend.position = "none") +
  theme_minimal()
```

```
## Plot variable not specified, automatically selected `.vars = values`
```



From the plots we found that even the θ_{11} is greater than 1, the series is still stationary but has a larger variation and is not able to be expressed as an AR model.

0.4.4 (e) Generate data from an ARIMA(1,0,1)

Define a simple function to generate an ARIMA(1,0,1) model:

```
arima101 <- function(phi, theta, n = 100L){
  y <- numeric(n)
  e <- rnorm(n)
  for (i in 2:n) {
    y[i] = phi * y[i - 1] + theta * e[i - 1] + e[i]
  }
  as_tsibble(
    tibble(step = seq(1,n), y = y),
    index = step
  )
}
```

Generate data from an ARIMA(1,0,1) with $\phi = 0.6$ and $\theta = 0.6$.

```
arima101_vlaues <- arima101(.6, .6)
arima101_vlaues
```

```
## # A tsibble: 100 x 2 [1]
##     step      y
##    <int>  <dbl>
##  1     1  0
##  2     2 -0.673
##  3     3  0.109
##  4     4  0.517
##  5     5  0.962
##  6     6  0.473
```

```
##  7     7 -0.792
##  8     8 -2.40
##  9     9 -2.62
## 10    10 -1.82
## # i 90 more rows
```

0.4.5 (f) Generate data from an AR(2) model

Define the simple function that creates an AR(2) model:

```
ar2 <- function(phi1, phi2, n = 100L){
  y <- numeric(n)
  e <- rnorm(n)
  for (i in 3:n) {
    y[i] = phi1 * y[i - 1] + phi2 * y[i - 2] + e[i]
  }
  as_tsibble(
    tibble(step = seq(1,n), y = y),
    index = step
  )
}
```

Generate the data from an AR(2) with $\phi_{21} = -0.8$ and $\phi_{22} = 0.3$:

```
ar2_values <- ar2(-.8, .3)
ar2_values
```

```
## # A tsibble: 100 x 2 [1]
##     step      y
##    <int>  <dbl>
##  1     1  0
##  2     2  0
##  3     3  0.196
##  4     4  0.619
##  5     5 -0.237
##  6     6  2.43
##  7     7 -1.59
##  8     8  1.23
##  9     9 -1.23
## 10    10  2.58
## # i 90 more rows
```

0.4.6 (g) Graph the latter two series and compare them

Let us combine them so that we can plot them easily and then visualize them directly.

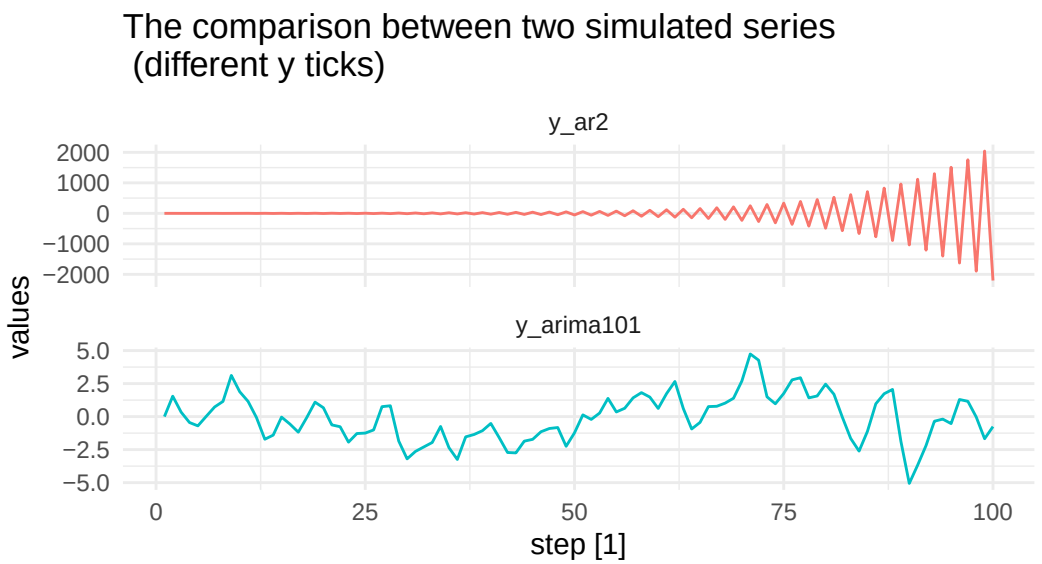
```
simul_series <- arima101(.6, .6) |>
  left_join(
    ar2(-.8, .3),
    by = c("step"),
    suffix = c("_arima101", "_ar2")) |>
```

```
  pivot_longer_series("step")

simul_series
```

```
## # A tsibble: 200 x 3 [1]
## # Key:       types [2]
##     step types      values
##    <int> <chr>       <dbl>
##  1     1 y_arima101  0
##  2     1 y_ar2       0
##  3     2 y_arima101  1.54
##  4     2 y_ar2       0
##  5     3 y_arima101  0.339
##  6     3 y_ar2      -0.195
##  7     4 y_arima101 -0.449
##  8     4 y_ar2      -0.613
##  9     5 y_arima101 -0.701
## 10     5 y_ar2      -0.890
## # i 190 more rows
```

```
simul_series |>
  autoplot() +
  facet_wrap(
    facets = "types",
    scales = "free_y",
    ncol = 1) +
  labs(title = "The comparison between two simulated series \n (different y ticks)") +
  theme_minimal() +
  theme(legend.position = "none")
```

```
## Plot variable not specified, automatically selected `.vars = values`
```



From the two plots we found that the AR(2) model is not stationary because of the $\phi_{21} = -0.8$ and $\phi_{22} = 0.3$, which does not meet the restriction: $|\phi_{22} - \phi_{21}| < 1$. The ARIMA(1,0,1) still meets the restriction and shows stationary over all steps.

0.5 Question5

0.5.1 (a) Find the mean and variance of the smoothed statistics S_t

Because of $S_t = Z_t = \mu + \epsilon_t$, where $\epsilon_t = a_t - \theta a_{t-1}$, we can express S_t as:

$$S_t = \mu + a_t - \theta a_{t-1}$$

where, all $a_t's$ are independently identical distribution, assuming they all follow: $a_t's \sim N(0, \sigma_a^2)$.

Hence, the mean of S_t would be:

$$\begin{aligned} E(S_t) &= E(\mu + a_t - \theta a_{t-1}) \\ &= \mu + E(a_t - \theta a_{t-1}) \\ &= \mu \end{aligned}$$

variance would be:

$$\begin{aligned} Var(S_t) &= Var(\mu + a_t - \theta a_{t-1}) \\ &= Var(a_t) + \theta^2 Var(a_{t+1}) - 2\theta Cov(a_t, a_{t+1}) \\ &= (1 + \theta^2)\sigma^2 \end{aligned}$$

0.5.2 (b) Find the mean and variance of the $\bar{Z}_t^N$

Because the ϵ_t comes from a MA(1) model, which determines that the $Z_t's$ are not independent with each other.

First, calculate the mean of the $\bar{Z}_t^N$:

$$\begin{aligned} E(\bar{Z}_t^N) &= \frac{1}{N} E(\sum_{t=1}^{N} Z_t) \\ &= \frac{1}{N} \sum_{t=1}^{N} E(Z_t) \\ &= \mu \end{aligned}$$

Then, the variance would be:

$$\begin{aligned} Var(\bar{Z}_t^N) &= \frac{1}{N^2} Var(\sum_{t=1}^{N} Z_t) \\ &= \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=1}^{N} Cov(Z_i, Z_j) \end{aligned}$$

Because the ϵ_t in the Z_t term has influence on the $\epsilon_t + 1$ that consists the Z_{t+1} term, so $Cov(Z_t, Z_s) \neq 0$, where $|t - s| = 1$.

By calculating, we have:

$$\sum_{i=1}^{N}\sum_{j=1}^{N} Cov(Z_i, Z_j) = \begin{cases} (1+\theta^2)\sigma^2, & \text{if } i = j \\ -\theta\sigma^2, & \text{if } |i-j| = 1 \\ 0, & \text{else} \end{cases}$$

By take replace of these values into the $Var(\bar{Z}_t^N)$, we have the value of the variance of $\bar{Z}_t^N$:

$$Var(\bar{Z}_t^N) = \frac{1}{N}(1+\theta^2)\sigma^2 + \frac{2(N-1)}{N^2}(-\theta\sigma^2)$$

0.6 Question6

0.6.1 (a) The least squares estimate of β

To a pre-set model:

$$y_t = \beta x_t + \eta_t \eta_t = \epsilon_t - \theta_1 \epsilon_{t-1} - \theta_2 \epsilon_{t-2}$$

where the residuals' distribution are known:

$$\epsilon_t \sim N(0, \sigma^2)$$

We can find the estimate of β by minimizing the summation of square of errors in our model.

$$SSE = \sum_{t=1}^{T}(y_t - \hat{y}_t)^2 = \sum_{t=1}^{T}(y_t - \beta x_t)^2 \sum_{t=1}^{T}(\eta_t)^2 = \sum_{t=1}^{T}(y_t - \beta x_t)^2$$

The β^* that minimizes SSE is the estimate of β from this sample, and we denote this estimate of β as $\hat{\beta}$.

Hence, let us take the derivative of this equation with the respective of β, making it equal to 0 to find the $\hat{\beta}$.

$$\partial \sum_{t=1}^{T}(\eta_t)^2 / \partial\beta = 2\sum_{t=1}^{T} x_t y_t + 2\beta \sum_{t=1}^{T} x_t^2 = 0$$

hence:

$$\beta = \frac{\sum_{t=1}^{T} x_t y_t}{\sum_{t=1}^{T} x_t^2}$$

If we have a sample that has a size of n, we will get an estimate from this sample :

$$\hat{\beta} = \frac{\sum_{t=1}^{n} x_t y_t}{\sum_{t=1}^{n} x_t^2}$$

0.6.2 (b) Find the variance of the estimate.

Now, we continue the inferring under the assumption that we got an $\hat{\beta}$ from a sample, let us simplify this $\hat{\beta}$.

$$\hat{\beta} = \frac{\sum_{t=1}^{n} x_t(\hat{y}_t + \eta_t)}{\sum_{t=1}^{n} x_i^2}$$

in which, the $x_t's$ are exogenous, so we can treat them as an unknown constant.

Here by, we can take the variance of $\hat{\beta}$.

$$\begin{aligned}
&Var(\hat{\beta}) \\
&= \frac{1}{(\sum_{t=1}^{n} x_i^2)^2} Var(\sum_{t=1}^{n} x_t\hat{y}_t + \sum_{t=1}^{n} x_t\eta_t) \\
&= \frac{1}{(\sum_{t=1}^{n} x_i^2)^2} Var(\sum_{t=1}^{n} x_t\eta_t) \\
&= \frac{1}{(\sum_{t=1}^{n} x_i^2)^2} \sum_{i=1}^{n}\sum_{j=1}^{n} Cov(x_i\eta_i, x_j\eta_j) \\
&= \frac{1}{(\sum_{t=1}^{n} x_i^2)^2} \sum_{i=1}^{n}\sum_{j=1}^{n} x_i x_j Cov(\eta_i, \eta_j)
\end{aligned}$$

To this step, we need to dive into the $Cov(\eta_i, \eta_j)$ part because of its special structure, which is not a simple "Independently identical distributed" random variable.

$$\begin{aligned}
&Cov(\eta_i, \eta_j) \\
&= Cov(\epsilon_i - \theta_1\epsilon_{i-1} - \theta_2\epsilon_{i-2}, \epsilon_j - \theta_1\epsilon_{j-1} - \theta_2\epsilon_{j-2}) \\
&= Cov \begin{bmatrix} \epsilon_i\epsilon_{i-j} & -\theta_1\epsilon_i\epsilon_{i-1-j} & -\theta_2\epsilon_i\epsilon_{i-2-j} \\ -\theta_1\epsilon_{i-1}\epsilon_{i-j} & -\theta_1^2\epsilon_{i-1}\epsilon_{i-1-j} & \theta_1\theta_2\epsilon_{i-1}\epsilon_{i-2-j} \\ -\theta_2\epsilon_{i-2}\epsilon_{i-j} & \theta_1\theta_2\epsilon_{i-2}\epsilon_{i-1-j} & \theta_2^2\epsilon_{i-2}\epsilon_{i-2-j} \end{bmatrix} \\
&= \begin{cases} (1 + \theta_1^2 + \theta_2^2)\sigma^2, & \text{if } i = j \\ \theta_1(\theta_2 - 1)\sigma^2, & \text{if } |i-j| = 1 \\ -\theta_2\sigma^2, & \text{if } |i-j| = 2 \\ 0, & \text{otherwise} \end{cases}
\end{aligned}$$

After carefully counting the number of $i = j, |i-j| = 1, |i-j| = 2$, we can write the Variance of $\hat{\beta}$ as:

$$Var(\hat{\beta}) = \frac{1}{\sum_{i=1}^{n} x_i^2}[\sigma^2(1 + \theta_1^2 + \theta_2^2)\sum_{t=1}^{n} x_i^2 + \sigma^2\theta_1(\theta_2 - 1)2\sum_{i=1}^{n}\sum_{j=i+1}^{n} x_i x_j - \sigma^2\theta_2 2\sum_{i=1}^{n}\sum_{j=i+2}^{n} x_i x_j]$$

0.6.3 (c) Commend on the validity of the usualy t-test for siginificance of β

The validity of the usual t-test for significance of β would be influenced because the t-test comes from a structure in which a normal distribution divided by the summation of a series squares of normal random variable. For example:

$$t \sim \frac{\bar{X} - \mu}{\sigma_x/\sqrt{n}}$$

The denominator here actually comes from the inference for $Var(\bar{X})$ that mainly depends on the "independent and identical" assumption.

However, our $y_t's$ here do not follow these assumptions, because there are autocorrelation in the η_t structure that influences the y_t and $\hat{b}eta$.

Hence, the usual t-test is not invalid for β.

0.7 Question7

0.7.1 Prepare the data set

Because the initial data set consists of every 30-minute electricity demand and the corresponding temperature from 2012-01-01 to 2014-10-31.

```
head(vic_elec)
```

```
## # A tsibble: 6 x 5 [30m] <Australia/Melbourne>
##   Time                Demand Temperature Date       Holiday
##   <dttm>               <dbl>       <dbl> <date>     <lgl>
## 1 2012-01-01 00:00:00  4383.        21.4 2012-01-01 TRUE
## 2 2012-01-01 00:30:00  4263.        21.0 2012-01-01 TRUE
## 3 2012-01-01 01:00:00  4049.        20.7 2012-01-01 TRUE
## 4 2012-01-01 01:30:00  3878.        20.6 2012-01-01 TRUE
## 5 2012-01-01 02:00:00  4036.        20.4 2012-01-01 TRUE
## 6 2012-01-01 02:30:00  3866.        20.2 2012-01-01 TRUE
```

To get the daily electricity demand and the maximum temperature for that day, we need to group these rows by the 'day' component from the 'Time' attribute.

```
# process the data set
processed_vic_elec <- vic_elec |>
  as_tibble() |>  # remove the root index of tsibble
  group_by(Date) |>
  summarise(
    maxi_temp = max(Temperature),
    elec_demand = sum(Demand),
    holiday = max(Holiday) # boolean type: 0, 1
    ) |>
    mutate(maxi_temp_2 = maxi_temp**2)

head(processed_vic_elec)
```

```
## # A tibble: 6 x 5
##   Date       maxi_temp elec_demand holiday maxi_temp_2
##   <date>         <dbl>       <dbl>   <int>       <dbl>
## 1 2012-01-01      32.7     222438.       1       1069.
## 2 2012-01-02      39.6     257965.       1       1568.
## 3 2012-01-03      31.8     267099.       0       1011.
## 4 2012-01-04      25.1     222742.       0        630.
## 5 2012-01-05      21.2     210585.       0        449.
## 6 2012-01-06      23.6     210247.       0        557.
```

Now, split the data set into train and test set.

```
train_vic_elec <- processed_vic_elec |>
  filter(
    (yearmonth(Date)) <= yearmonth('2014-10')) |>
  as_tsibble()
```

```
## Using 'Date' as index variable.
```

```
test_vic_elec <- processed_vic_elec |>
  filter(
    (yearmonth(Date)) > yearmonth('2014-10')) |>
  as_tsibble()
```

```
## Using 'Date' as index variable.
```

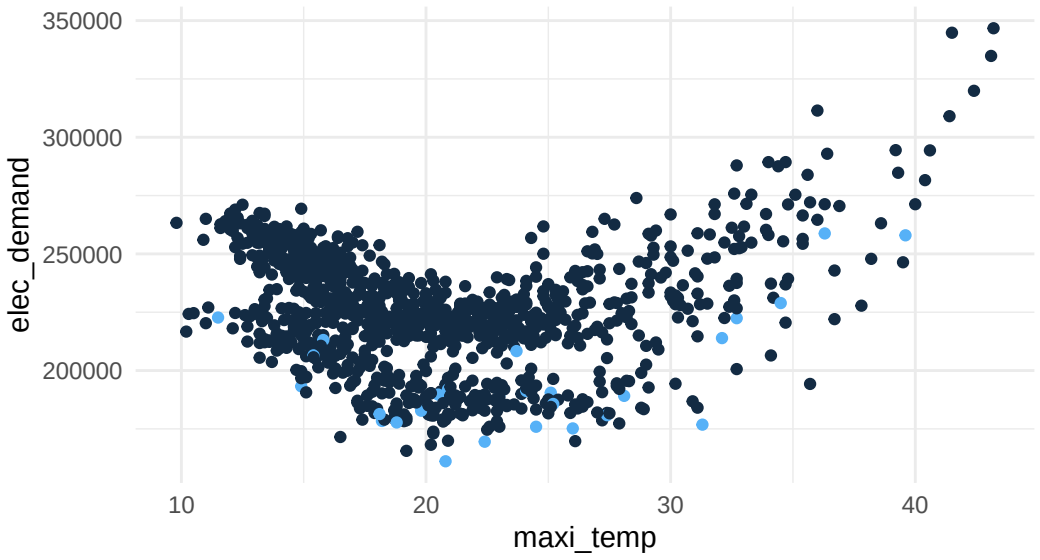
```
head(train_vic_elec)
```

```
## # A tsibble: 6 x 5 [1D]
##   Date       maxi_temp elec_demand holiday maxi_temp_2
##   <date>         <dbl>       <dbl>   <int>       <dbl>
## 1 2012-01-01      32.7     222438.       1       1069.
## 2 2012-01-02      39.6     257965.       1       1568.
## 3 2012-01-03      31.8     267099.       0       1011.
## 4 2012-01-04      25.1     222742.       0        630.
## 5 2012-01-05      21.2     210585.       0        449.
## 6 2012-01-06      23.6     210247.       0        557.
```

0.7.2 (a) Fit a regression model with SARIMA error process and compare it with simple regression model.

Have a look at the relationship between temperature and electricity demand to check if they are in linear relation.

```
train_vic_elec |>
  ggplot(
    aes(x=maxi_temp,
        y = elec_demand,
        color = holiday)
    ) +
  geom_point() +
  theme_minimal() +
    theme(legend.position = "none")
```



After checking the relationship between them, i choose to build a linear model with the quadratic term of temperature in it.

$$y_t = \beta_0 + \beta_1 X_t + \beta_2 X_t^2 + \beta_3 H + \eta_t$$

In which, the H is a categorical variable:

$$H = \begin{cases} 1, \text{if the day is holidy} \\ 0, \text{if the day is not holidy} \end{cases}$$

and the η_t follows a SARIMA process that we will build in the following.

$$\Phi_P(B^s)\phi_p(B)(1-B)^d(1-B^s)^D\eta_t = \Theta_Q(B^s)\theta_q(B)\epsilon_t\epsilon_t \sim N(0,\sigma^2) \quad \text{to all t's}$$

In order to compare this SARIMA error process model with the standard regression model, we build the standard model at the same time.

$$y_t = \beta_0 + \beta_1 X_t + \beta_2 X_t^2 + \beta_3 H + \epsilon_t\epsilon_t \sim N(0,\sigma^2) \quad \text{to all t's}$$

(*Note: the two* ϵ here can be two different random variables.)

Now, create 4 models in which there are 3 different ARIMA/SARIMA models and one standard linear model:

```
elec_models <- train_vic_elec |>
    model(
      arima_default = ARIMA(
        formula = elec_demand ~
          maxi_temp + maxi_temp_2 + holiday
      ),

      sarima_intercept = ARIMA(
          formula = elec_demand ~
        1 + maxi_temp + maxi_temp_2 +holiday +
```

```
      pdq(
          p = 0:3, d = 0:1, q = 0:3,
          p_init = 1, q_init = 1, fixed = list()
          ) +
      PDQ(
          P = 0:2, D = 0:1, Q = 0:2, period = NULL,
          P_init = 1, Q_init = 1, fixed = list()
    )
),

      sarima = ARIMA(
        formula = elec_demand ~
        maxi_temp + maxi_temp_2 +holiday +
        pdq(
            p = 0:3, d = 0:1, q = 0:3,
            p_init = 1, q_init = 1, fixed = list()
        ) +
        PDQ(
            P = 0:2, D = 0:1, Q = 0:2, period = NULL,
            P_init = 1, Q_init = 1, fixed = list()
        )

        ),
      linear = TSLM(
        elec_demand ~ 1 + maxi_temp + maxi_temp_2 + holiday)
    )
```

Let us check on our model performance, through σ^2, AIC , AICc, and BIC:

```
glance(elec_models) |>
  select(.model, sigma2, AICc, BIC)
```

```
## # A tibble: 4 x 4
##   .model               sigma2   AICc    BIC
##   <chr>                 <dbl>  <dbl>  <dbl>
## 1 arima_default     58967630. 21324. 21363.
## 2 sarima_intercept  59020802. 21326. 21370.
## 3 sarima            58967630. 21324. 21363.
## 4 linear           387874072. 20474. 20499.
```

Based on the criteria, choose the model 'arima_default' (or 'sarima' equivalent) as the best model with SARIMA error process.

Now, we can compare this best SARIMA-improved model with the standard linear model, check how difference their parameters are.

```
# pull the models out
best_arima <- elec_models[1][1]
sd_linear <- elec_models[4][1]

print(best_arima)
```

```
## # A mable: 1 x 1
```

```
##                            arima_default
##                                  <model>
## 1 <LM w/ ARIMA(2,0,0)(2,1,0)[7] errors>
```

Check the parameters for the SARIMA-improved model.

```
# report the model parameters
print(coef(best_arima))
```

```
## # A tibble: 7 x 6
##   .model        term           estimate std.error statistic   p.value
##   <chr>         <chr>             <dbl>     <dbl>     <dbl>     <dbl>
## 1 arima_default ar1               0.749     0.0379    19.8   5.99e- 74
## 2 arima_default ar2               0.0108    0.0383     0.282 7.78e-  1
## 3 arima_default sar1             -0.651     0.0311   -20.9   2.06e- 81
## 4 arima_default sar2             -0.270     0.0305    -8.86  3.33e- 18
## 5 arima_default maxi_temp    -7388.      293.       -25.2   2.01e-109
## 6 arima_default maxi_temp_2    175.        5.86      29.8   4.47e-141
## 7 arima_default holiday     -29890.     1092.       -27.4   2.09e-124
```

Check the parameters for the standard linear model.

```
print(coef(sd_linear))
```

```
## # A tibble: 4 x 6
##   .model term        estimate std.error statistic   p.value
##   <chr>  <chr>          <dbl>     <dbl>     <dbl>     <dbl>
## 1 linear (Intercept)  377764.    6622.      57.0  3.17e-321
## 2 linear maxi_temp    -14325.     587.     -24.4  4.00e-104
## 3 linear maxi_temp_2     310.      12.3     25.3  4.68e-110
## 4 linear holiday      -30004.    3787.      -7.92 5.95e- 15
```

From their estimates of common parameters, there are actually some difference on their values.

To make the difference look more clear, we can calculate the difference between them:

```
# calculate the difference between the coefficients
com_coef_linear <- coef(sd_linear) |>
          filter((term == "maxi_temp") |
              (term == "maxi_temp_2") |
              (term == "holiday"))

com_coef_sarima <- coef(best_arima) |>
          filter((term == "maxi_temp") |
              (term == "maxi_temp_2") |
              (term == "holiday"))

diff_coeffs <- com_coef_sarima$estimate -       com_coef_linear$estimate

diff_coeffs
```

```
## [1] 6936.6870 -135.4569  113.8567
```

So, we see that the SARIMA-improved model is 6936.6870 higher than linear model in “maximum temperature” term; 135.4569 lower in ” maximum temperature” term; 113.8567 higher in “holiday” term.

0.7.3 (b) Write down the model and plot the forecasts.

Now, i can write down the model with the specific formulas.

To the SARIMA-improved model:

$$Y_t^{'} = -7387.82X_t + 174.53X_t^2 - 29890.15H + 0.7495Y_{t-1}^{'} + 0.0108Y_{t-2}^{'} + \ - 0.6509Y_{t-7}^{'} - 0.2701Y_{t-14}^{'} + \epsilon_t$$

where, the $Y_t^{'}$ is:

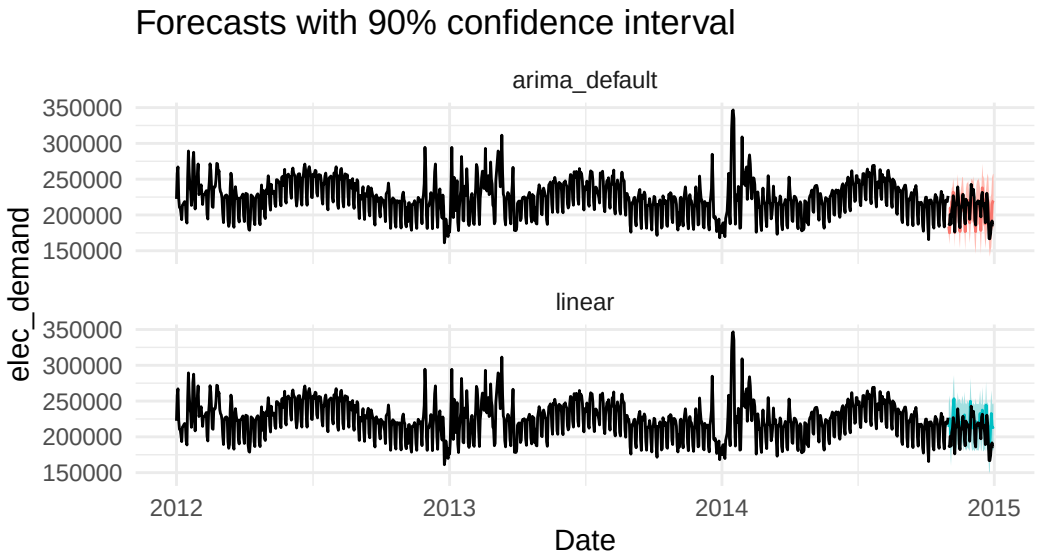
$$Y_t^{'} = Y_t - Y_{t-7}$$

To the standard linear model:

$$Y_t = 377763.84 - 14324.51X_t + 309.99X_t^2 - 30004.01H + \epsilon_t$$

Now, let us do forecasting on the test data and plot them.

```
sarima_linear_models <- elec_models |>
  select(linear, arima_default)

forecasts <- sarima_linear_models |>
  forecast(test_vic_elec)

forecasts|>
  autoplot(train_vic_elec, level = 90) +
  geom_line(
    data = test_vic_elec,
    mapping = aes(x = Date, y = elec_demand)) +
  facet_wrap(vars(.model), ncol = 1) +
  theme_minimal() +
  labs(title = "Forecasts with 90% confidence interval") +
  theme(legend.position = 'none')
```



Now, let us check and calculate the difference in their forecasts.

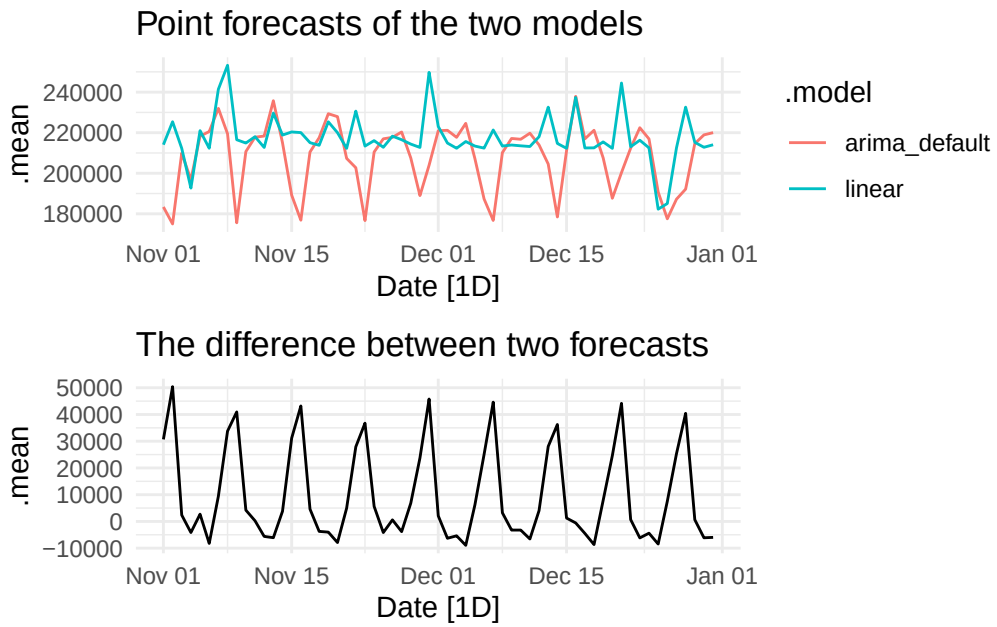
```
plot1 <- forecasts |>
  as_tsibble() |>
  autoplot(.mean) +
  labs(title = "Point forecasts of the two models") +
  theme_minimal()

points_fore_sarima <- forecasts |>
  filter(.model == 'arima_default')|>
  as_tsibble() |>
  select(.mean)

points_fore_linear <- forecasts |>
  filter(.model == "linear") |>
  as_tsibble() |>
  select(.mean)

# make difference between point forecasts
points_difference <- points_fore_linear - points_fore_sarima
# get the Date variable back
points_difference <- points_difference |>
  mutate(Date = points_fore_sarima$Date) |>
  as_tsibble()
```

```
## Using ‘Date‘ as index variable.
```

```
plot2 <- points_difference |>
  autoplot() +
  labs(title = "The difference between two forecasts") +
  theme_minimal()
```

```
## Plot variable not specified, automatically selected ‘.vars = .mean‘
```

```
plot1 / plot2
```



From the point forecasts and difference plots, we found that these two models behave very differently in forecasting the future value(the test data). Most time when the SARIMA-improved forecasts a trough, the linear model always gives a slight peak or normal value.

Further more, we can measure their performance by checking the forecasts with the true values. Here, i prefer using MAE, RMSE and MAPE, no other reasons, just intuitive :).

```
accuracy(forecasts, test_vic_elec) |> select(.model, MAE, RMSE, MAPE)
```

```
## # A tibble: 2 x 4
##   .model           MAE   RMSE  MAPE
##   <chr>          <dbl>  <dbl> <dbl>
## 1 arima_default  8804. 11601.  4.35
## 2 linear        16159. 20847.  8.17
```

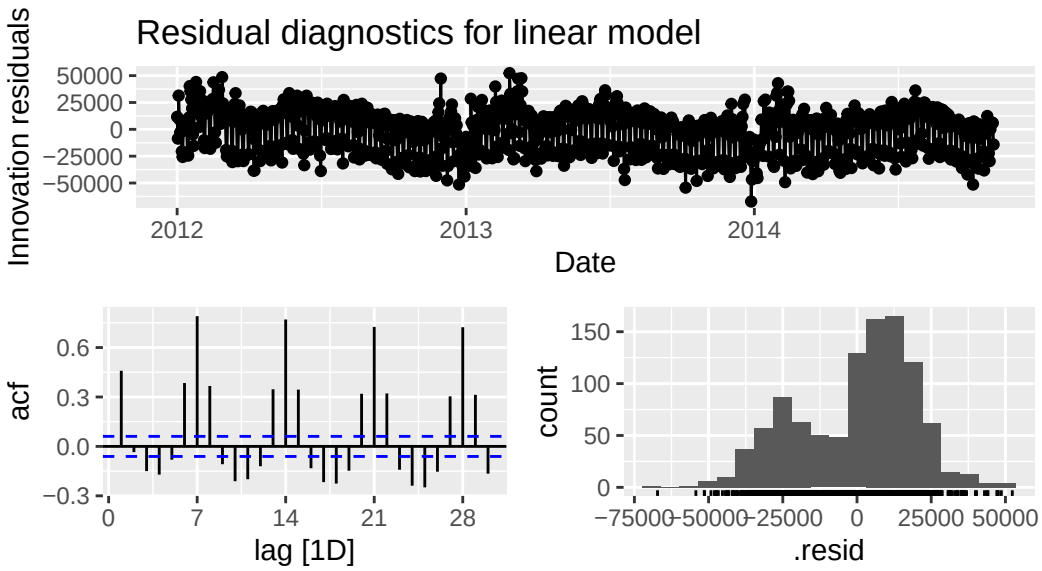
Based on these criterion values, we found the arima_default model(<LM w/ ARIMA(2,0,0)(2,1,0)[7] errors>) perform better than the linear model in forecasting the future (test data).

Thanks god that the AICc and BIC not telling our the whole story about the models!

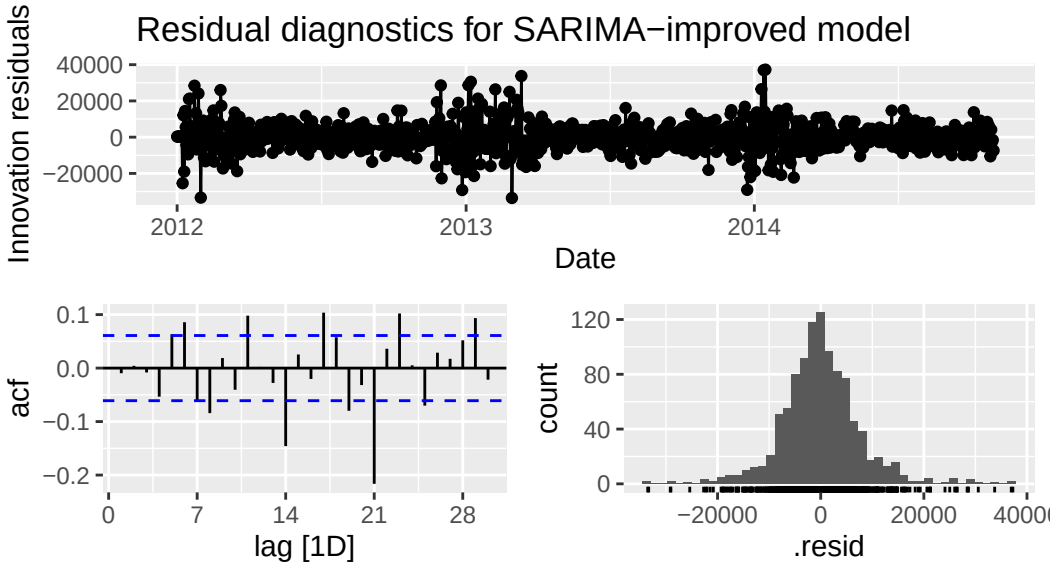
0.7.4 (c) Check the residuals of models to ensure the SARIMA-improved model has addressed the autocorrelations seen in the linear model.

Let draw the residual diagnostics plots first to have a direct and intuitive view about their residuals.

```
# plot the residual for the linear model
sd_linear |> gg_tsresiduals() +
  labs(title = "Residual diagnostics for linear model")
```




```
best_arima |>
  gg_tsresiduals() +
  labs(title = "Residual diagnostics for SARIMA-improved model")
```



From the two plots, we can find that the residuals from the SARIMA-improved follow a distribution that looks more like a normal distribution. And the ACF values in the SARIMA one are much lower than the values from the linear one, indicating there are less possible autocorrelation in the SARIMA residuals.

However, the ACF of the SARIMA still shows some correlation, especially in the lag 8, 14, and 21, which also indicates there are still some seasonal components. So the SARIMA one can still be improved.

Further more, we can do some statistics tests on the two models to check features about their residuals. Here let us do one test from the Portmanteau tests: Ljung Box test.

Because the two models have different amount of parameters, so their "dof" parameter should be differnt.

```
# for the SARIMA one
print(augment(best_arima) |>
        features(.innov, ljung_box, dof = 7, lag = 15)
      )
```

```
## # A tibble: 1 x 3
##   .model        lb_stat lb_pvalue
##   <chr>           <dbl>     <dbl>
## 1 arima_default    62.3  1.64e-10
```

```
# for the linear one
augment(sd_linear) |> features(.innov, ljung_box, dof = 4, lag = 15)
```

```
## # A tibble: 1 x 3
##   .model lb_stat lb_pvalue
##   <chr>    <dbl>     <dbl>
## 1 linear   2218.         0
```

From the two tests, we found that both of them do not pass the test, so we have to reject the null hypothesis for both models - H_0: the residuals are independent with each other. It means both models can be improved by capturing the autocorrelation from the residuals.

However, generally speaking, the SARIMA-improved model considers more autocorrelation in the initial residuals than the linear model, so the former one behaves better in forecasting.

0.8 Question8

Let us read the data into our current working environment first.

```
getSymbols("^VIX", src = "yahoo", from = "2015-1-1", to = "2019-12-31")
```

```
## Warning: ^VIX contains missing values. Some functions will not work if objects
## contain missing values in the middle of the series. Consider using na.omit(),
## na.approx(), na.fill(), etc to remove or replace them.
```

```
## [1] "VIX"
```

```
# Convert the VIX variable into a tsibble object
vix_index <- zoo::fortify.zoo(VIX)
```

Transform it into a tsibble object, creating a consecutive trading day series as a suitable index.

```
# cleaning the original
vix_index <- vix_index |>
  rename(
  "Close" = "VIX.Close",
  "Date" = "Index") |>
  filter(!is.na(Close)) |> # remove the rows where the Close is NA
  mutate(Days = row_number()) |>
  as_tsibble(index = Days) |>
```

```
  relocate(Days, Close, .before = VIX.Open)

# split the data into train and text data sets
train_vix_index <- vix_index |>
  filter(
    (Date >= as.Date("2017-02-02")) & #
    (Date <= as.Date("2019-10-31"))
  )

test_vix_index <- vix_index |>
  filter(
    Date > as.Date("2019-10-31")
    )

head(test_vix_index)
```

```
## # A tsibble: 6 x 8 [1]
##   Date        Days Close VIX.Open VIX.High VIX.Low VIX.Volume VIX.Adjusted
##   <date>     <int> <dbl>    <dbl>    <dbl>   <dbl>      <dbl>        <dbl>
## 1 2019-11-01  1218  12.3     12.5     12.6    12.3          0         12.3
## 2 2019-11-04  1219  12.8     12.7     13.1    12.4          0         12.8
## 3 2019-11-05  1220  13.1     12.9     13.3    12.2          0         13.1
## 4 2019-11-06  1221  12.6     13.2     13.4    12.6          0         12.6
## 5 2019-11-07  1222  12.7     12.6     12.9    12.3          0         12.7
## 6 2019-11-08  1223  12.1     13.0     13.1    12            0         12.1
```

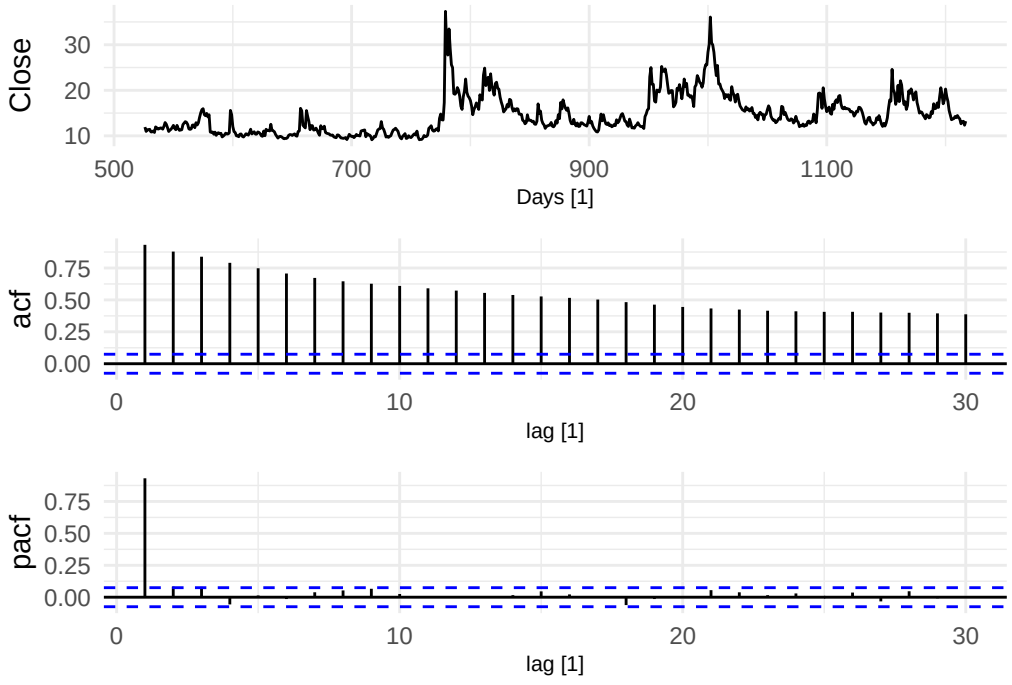
Now, we can start our model fitting work.

0.8.1 (a) Fit the benchmarks and a suitable ARIMA model for closing price and check the performance by cross-validation.

Before we fit these models on the Close variable, let us glance the trend of the Close first.

```
plot1 <- train_vix_index |>
  autoplot(Close) +
  # labs(title = "Trend of Close in training set") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 8)
  )

plot2 <- train_vix_index |>
  ACF(Close, lag_max = 30) |>
  autoplot() +
  # labs(title = "ACF of the Close variabe from training set") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 8)
  )

plot3 <- train_vix_index |>
  PACF(Close, lag_max = 30) |>
```

```
  autoplot() +
  # labs(title = "ACF of the Close variabe from training set") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 8)
  )

plot1/plot2/plot3
```



The trend and ACF plot shows there is a significant autocorrelation in the Close variable, maybe a Move Averaging process is a good choice to model the autocorrelation in the Close variable.

Let us do some tests on it for ARIMA model, deciding whether we need do some transformations on it to make it stationary.

```
train_vix_index |>
  features(
    Close,
   c(unitroot_kpss, # check if the series is stationary
     unitroot_ndiffs, # check the difference for normal
     unitroot_nsdiffs, # check the difference number for season
     box_pierce,
     ljung_box,
     guerrero # check for the good lambda value for box-cox transformation
  )
  ) |>
  pivot_longer(cols = !starts_with("_")) # make it look clear
```

```
## # A tibble: 9 x 2
```

```
##   name              value
##   <chr>             <dbl>
## 1 kpss_stat         2.77
## 2 kpss_pvalue       0.01
## 3 ndiffs            1
## 4 nsdiffs           0
## 5 bp_stat         600.
## 6 bp_pvalue         0
## 7 lb_stat         603.
## 8 lb_pvalue         0
## 9 lambda_guerrero  -0.900
```

According to these tests, the Close variable is not stationary and needs one-order difference and maybe need do a Box-Cox transformation to keep it having good statistical properties.

Let us do a first-order difference on it and tests it again.

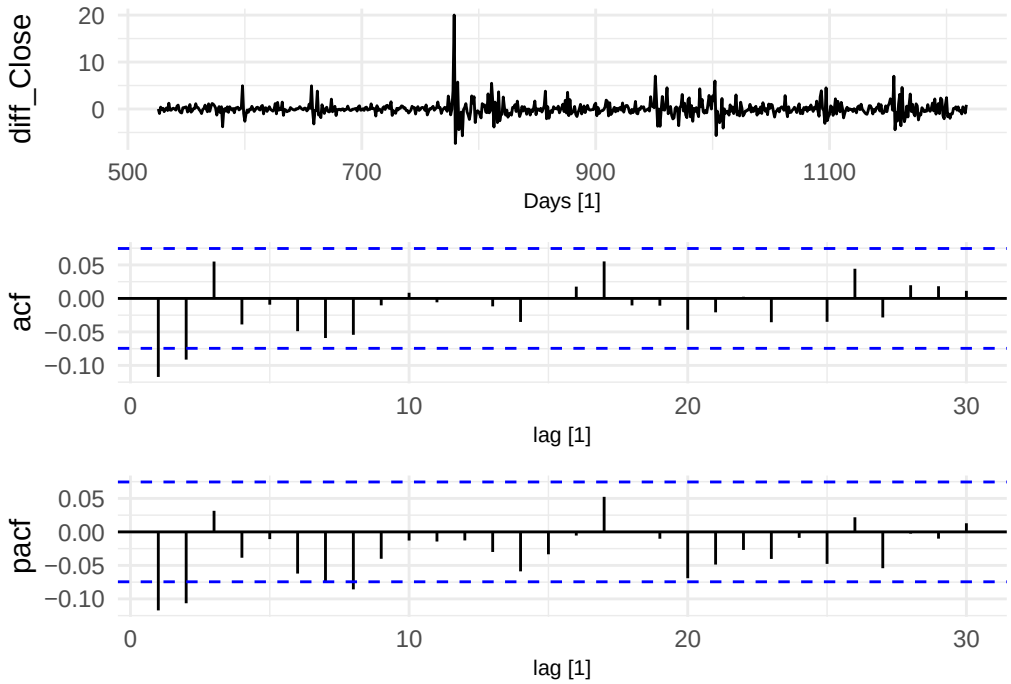
```
train_vix_index <- train_vix_index|>
  mutate(diff_Close = c(0, diff(Close)))

# update the test data set simultaneously
test_vix_index <- test_vix_index|>
  mutate(diff_Close = c(0, diff(Close)))

train_vix_index |>
  features(
    diff_Close,
   c(unitroot_kpss, # check if the series is stationary
     unitroot_ndiffs, # check the difference for normal
     unitroot_nsdiffs, # check the difference number for season
     box_pierce,
     ljung_box,
     guerrero # check for the good lambda value for box-cox transformation
  )
  ) |>
  pivot_longer(cols = !starts_with("_")) # make it look clear
```

```
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf replaced by maximum positive value
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf replaced by maximum positive value
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf replaced by maximum positive value
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf replaced by maximum positive value
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf replaced by maximum positive value
## Warning in optimise(lambda_coef_var, c(lower, upper), x = x, .period =
## max(.period, : NA/Inf replaced by maximum positive value
```

```
## # A tibble: 9 x 2
##   name              value
##   <chr>             <dbl>
## 1 kpss_stat       0.0177
```

```
## 2 kpss_pvalue     0.1
## 3 ndiffs          0
## 4 nsdiffs         0
## 5 bp_stat         9.50
## 6 bp_pvalue       0.00205
## 7 lb_stat         9.54
## 8 lb_pvalue       0.00201
## 9 lambda_guerrero 1.00
```

```
plot1 <- train_vix_index |>
  autoplot(diff_Close) +
  # labs(title = "Trend of Close in training set") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 8)
  )

plot2 <- train_vix_index |>
  ACF(diff_Close, lag_max = 30) |>
  autoplot() +
  # labs(title = "ACF of the Close variabe from training set") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 8)
  )

plot3 <- train_vix_index |>
  PACF(diff_Close, lag_max = 30) |>
  autoplot() +
  # labs(title = "ACF of the Close variable from training set") +
  theme_minimal() +
  theme(
    axis.title.x = element_text(size = 8)
  )

plot1/plot2/plot3
```



I think the first-order differed data is good enough for us to start the following modeling work, no need to take the Box-Cox transformation here.

Now, let us try fit the benchmark models and a suitable ARIMA model here. *(To trading days, it is hard and unrealistic to find a proper seasonal component from it, so SARIMA model will not be considered here.)*

```
vix_models <- train_vix_index |>
  model(
    naive = NAIVE(diff_Close),
    mean = MEAN(diff_Close),
    drift = RW(diff_Close ~ drift()),
    arima = ARIMA(diff_Close),
    arima_intercept = ARIMA(diff_Close ~ 1)
  )

# compare the models here, choose the suitable ARIMA model
glance(vix_models) |> select(.model, AICc, BIC, log_lik, sigma2)
```

```
## # A tibble: 5 x 5
##   .model           AICc   BIC log_lik sigma2
##   <chr>           <dbl> <dbl>   <dbl>  <dbl>
## 1 naive             NA    NA      NA    5.57
## 2 mean              NA    NA      NA    2.49
## 3 drift             NA    NA      NA    5.57
## 4 arima           2575. 2589.  -1285.   2.40
## 5 arima_intercept 2586. 2604.  -1289.   2.44
```

According the AICc and BIC criteria, we think the ARIMA model(ARIMA(1,0,1)) with no intercept is proper for this forecasting work.

Here, we use cross-validation method to evaluate the model performance on the training data set. It is worth to notice that the test set split in the cross-validation is from the training data, not the test data that includes data after "2019-10-31". Cross-validation only offers a more accurate measure on the model performance.

To split the training data set into many accumulative "training" sets, we use the "stretch_tsibble()" function and make the amount of set as 20 to save our time and computation.

```
amount_train_set <- 14
initial_size <- 300
each_step <- ceiling(
  (dim(train_vix_index)[1] - 300) /
    amount_train_set
  ) # take the upper integer for each step, make the total amount of set is 14

train_cross_vix_index <- train_vix_index |>
  stretch_tsibble(.init = 300, .step = each_step) |>
  relocate(.id, .before = Date) # move the identifier .id to the first

head(train_cross_vix_index)
```

```
## # A tsibble: 6 x 10 [1]
## # Key:       .id [1]
##     .id Date        Days Close VIX.Open VIX.High VIX.Low VIX.Volume VIX.Adjusted
##   <int> <date>     <int> <dbl>    <dbl>    <dbl>   <dbl>      <dbl>        <dbl>
## 1     1 2017-02-02   526  11.9     12.4     12.5    11.6          0         11.9
## 2     1 2017-02-03   527  11.0     11.8     11.8    10.7          0         11.0
## 3     1 2017-02-06   528  11.4     11.4     11.8    11.1          0         11.4
## 4     1 2017-02-07   529  11.3     11.4     11.7    11.1          0         11.3
## 5     1 2017-02-08   530  11.4     11.2     11.8    11.1          0         11.4
## 6     1 2017-02-09   531  10.9     11.4     11.5    10.7          0         10.9
## # i 1 more variable: diff_Close <dbl>
```

Now, let us fit the models on these sub-training sets.

```
vix_models_from_cv <-  train_cross_vix_index |> model(
  naive = NAIVE(diff_Close),
    mean = MEAN(diff_Close),
    drift = RW(diff_Close ~ drift()),
    arima = ARIMA(diff_Close),
    arima_intercept = ARIMA(diff_Close ~ 1)
  )

vix_models_from_cv|>
  forecast(h = 1) |>
  accuracy(train_vix_index) |>
  select(.model, .type, RMSE, MAE, MASE, MAPE)
```

```
## Warning: The future dataset is incomplete, incomplete out-of-sample data will be treated as missing.
## 1 observation is missing at 1218
```

```
## # A tibble: 5 x 6
##   .model          .type  RMSE   MAE  MASE  MAPE
```

```
##   <chr>           <chr> <dbl> <dbl> <dbl> <dbl>
## 1 arima           Test   1.63  1.16 0.838  98.2
## 2 arima_intercept Test   1.64  1.16 0.841  96.1
## 3 drift           Test   2.55  1.58 1.15  152.
## 4 mean            Test   1.64  1.21 0.877 101.
## 5 naive           Test   2.54  1.58 1.15  152.
```

From the RMSE, MAE and MASE criteria, the ARIMA(1,0,1) is still the best one to do forecasting because it has all smallest values in these three criteria, which indicates it fits the data better to produce a more accurate forecast.

However, not all criteria support the ARIMA(1,0,1), for example, the MAPE supports SARIMA(0,0,2) to be a more suitable model.

Actually, if we peek the models that were trained on these sub-training sets, it is easy to find that the parameters about these models are changing with the training size.

If we consider all models equal in weight and choose one representative model, then the parameter of (p,d,q) from the "mode" is the best choice for the model from where it comes.

Similarly, this reason also supports ARIMA(1,0,1) as the representative one for ARIMA model.

```
vix_models_from_cv
```

```
## # A mable: 15 x 6
## # Key:     .id [15]
##      .id   naive    mean          drift           arima        arima_intercept
##    <int> <model> <model>       <model>         <model>                <model>
##  1     1 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,4)> <ARIMA(1,0,4) w/ mean>
##  2     2 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,4)> <ARIMA(0,0,3) w/ mean>
##  3     3 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(3,0,2)> <ARIMA(2,0,1) w/ mean>
##  4     4 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(1,0,1) w/ mean>
##  5     5 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(1,0,1) w/ mean>
##  6     6 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,4)> <ARIMA(1,0,4) w/ mean>
##  7     7 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(1,0,1) w/ mean>
##  8     8 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,4)> <ARIMA(0,0,3) w/ mean>
##  9     9 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(0,0,2) w/ mean>
## 10    10 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(1,0,1) w/ mean>
## 11    11 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(1,0,1) w/ mean>
## 12    12 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(0,0,2) w/ mean>
## 13    13 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(1,0,1) w/ mean>
## 14    14 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(0,0,2) w/ mean>
## 15    15 <NAIVE>  <MEAN> <RW w/ drift> <ARIMA(1,0,1)> <ARIMA(0,0,2) w/ mean>
```

And if we look into the specific parameters, we would found even some models have the same (p,d,q), they have different specific estimates. These details well represents the randomness from sampling and we should prefer models from larger samples.

Generally speaking, we choose ARIMA(1,0,1) as our suitable model to do the forecasts because it performs general good.

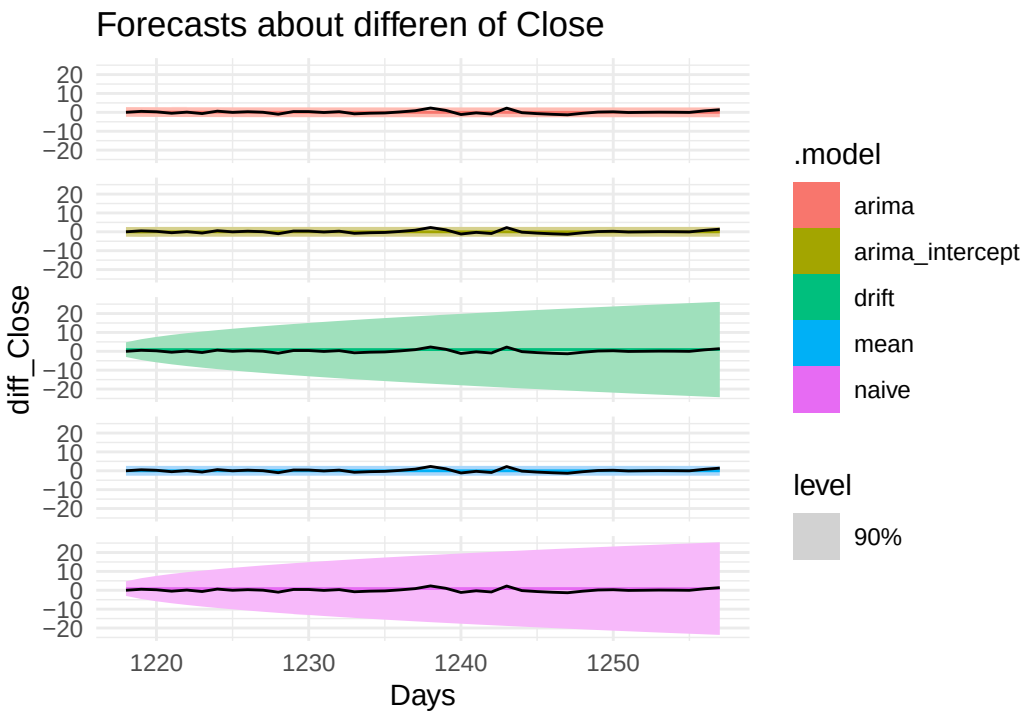
0.8.2 (b) Produce forecasts of the best fitted model and plot for the test sample.

```r
# take the best fitted model
best_arima <- vix_models |>  # same to the ARIMA trained on sub-training data
  select(arima)

best_arima |> report()
```

```
## Series: diff_Close
## Model: ARIMA(1,0,1)
##
## Coefficients:
##          ar1      ma1
##       0.8426  -0.9461
## s.e.  0.0541   0.0353
##
## sigma^2 estimated as 2.405:  log likelihood=-1284.66
## AIC=2575.32   AICc=2575.36   BIC=2588.94
```

Do forecasting with this model and plot the forecasts.

```r
# do forecasting together with other models and true data
forecast_values_all <- vix_models |>
  forecast(test_vix_index)

forecast_values_all |>
  autoplot(test_vix_index, level = 90) +
  facet_wrap(facet = ".model",
             ncol = 1) +
  labs(title = 'Forecasts about differen of Close') +
  theme_minimal() +
  theme(
    strip.text = element_blank()  # Removes facet titles
  )
```



From my view, the forecasts from ARIMA is reasonable, because the 90% confidence interval is informative with the margin or error less than 3 or even 2. The true data is also falling within that informative interval.

But these forecasts are the difference of Close price, to get the real Close price forecasts, we need to accumulate these differences and add them in vector on top of the last observation.

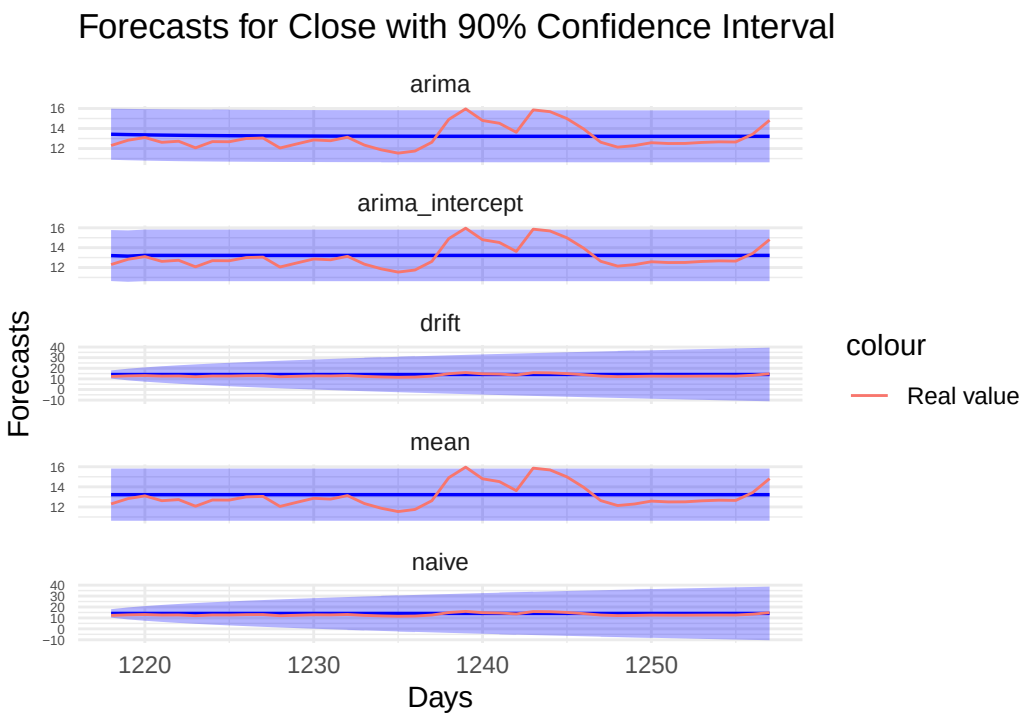
```
# accumulate the difference

# take the value of the last observation
last_value <-  train_vix_index |>
  filter(Date == as.Date("2019-10-31")) |>
  select(Close) |>
  pull(var = 1)

# return the true forecasts and their confidence interval ends
forecast_values_all <- forecast_values_all |>
  mutate(
    Forecasts_mean = mean(diff_Close) + last_value, # extract the mean of each distribution
    Forecasts_lower = quantile(diff_Close + last_value, 0.05), # lower 5% for 90% CI
    Forecasts_upper = quantile(diff_Close + last_value, 0.95)
    ) # upper 5% for 90% CI

forecast_values_all
```

```
## # A fable: 200 x 14 [1]
## # Key:     .model [5]
##    .model  Days   diff_Close .mean Date       Close VIX.Open VIX.High VIX.Low
##    <chr>  <int>       <dist> <dbl> <date>     <dbl>    <dbl>    <dbl>   <dbl>
##  1 naive   1218 N(0.89, 5.6) 0.890 2019-11-01  12.3     12.5     12.6    12.3
##  2 naive   1219  N(0.89, 11) 0.890 2019-11-04  12.8     12.7     13.1    12.4
```

```
##  3 naive   1220  N(0.89, 17) 0.890 2019-11-05  13.1     12.9     13.3    12.2
##  4 naive   1221  N(0.89, 22) 0.890 2019-11-06  12.6     13.2     13.4    12.6
##  5 naive   1222  N(0.89, 28) 0.890 2019-11-07  12.7     12.6     12.9    12.3
##  6 naive   1223  N(0.89, 33) 0.890 2019-11-08  12.1     13.0     13.1    12
##  7 naive   1224  N(0.89, 39) 0.890 2019-11-11  12.7     13.1     13.5    12.7
##  8 naive   1225  N(0.89, 45) 0.890 2019-11-12  12.7     12.6     13.1    12.4
##  9 naive   1226  N(0.89, 50) 0.890 2019-11-13  13       12.9     13.9    12.9
## 10 naive   1227  N(0.89, 56) 0.890 2019-11-14  13.1     13.2     13.8    12.9
## # i 190 more rows
## # i 5 more variables: VIX.Volume <dbl>, VIX.Adjusted <dbl>,
## #   Forecasts_mean <dbl>, Forecasts_lower <dbl>, Forecasts_upper <dbl>
```

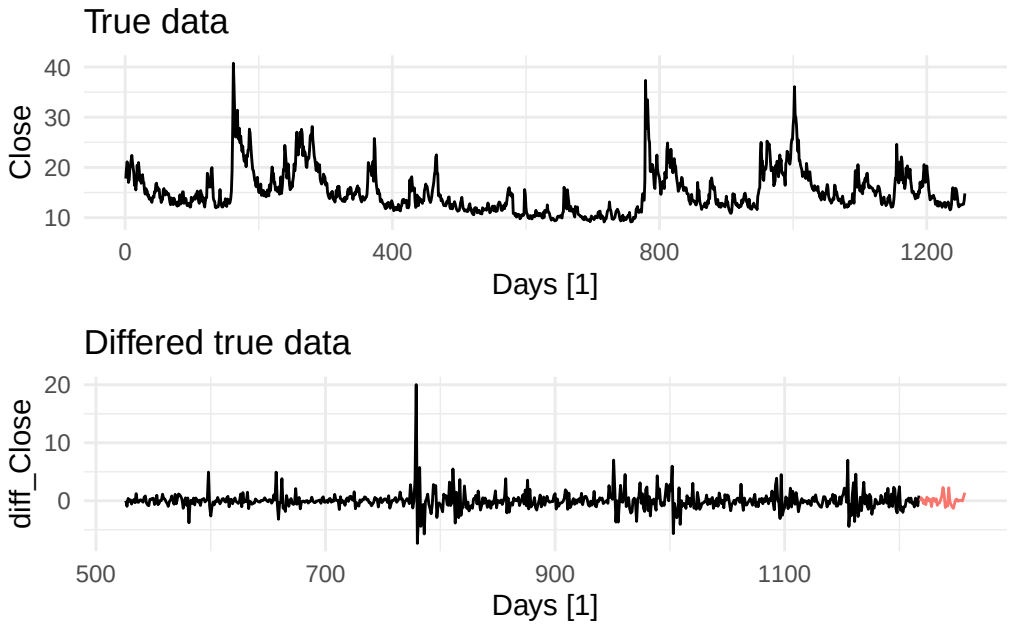
```r
# plot the forecasts in the same scale
# Create the plot
 ggplot(forecast_values_all, aes(x = Days)) +
    geom_line(
      aes(y = Forecasts_mean),
      color = "blue",
      size = .6
      ) + # Line for the mean
    geom_ribbon(
      aes(ymin = Forecasts_lower, ymax = Forecasts_upper),
      alpha = 0.3, fill = "blue"
      ) + # Ribbon for CI
    labs(
      title = "Forecasts for Close with 90% Confidence Interval",
      x = "Days",
      y = "Forecasts"
      ) +
    geom_line(aes(y = Close, color = "Real value")) +
    facet_wrap(
      ".model",
      scales = "free_y",
      ncol = 1) +
    theme_minimal() +
    theme(
      axis.text.y = element_text(size = 5) # Makes y-axis ticks smaller
    )
```

```
## Warning: Using ‘size‘ aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use ‘linewidth‘ instead.
## This warning is displayed once every 8 hours.
## Call ‘lifecycle::last_lifecycle_warnings()‘ to see where this warning was
## generated.
```



The forecasts from ARIMA model look reasonable, the 90% CI covers all true values within it and the errors are relative small. Actually, the two ARIMA models and the MEAN model all perform well, it is might due to *the stable property of the differed true value*, which makes it easy to forecast.

```
plot1 <- vix_index |>
  autoplot(Close) +
  labs(title = "True data") +
  theme_minimal()


plot2 <- train_vix_index |>
  autoplot(diff_Close) +
  geom_line(
    data = test_vix_index,
    mapping = aes(y = diff_Close, color = "red")
  ) +
  labs(title = "Differed true data") +
  theme_minimal() +
  theme(legend.position = 'none')

plot1 / plot2
```


True data
Close
40
30
20
10
0
400
800
1200
Days [1]
Differed true data
diff_Close
20
10
0
500
700
900
1100
Days [1]