The Report

Group 6

2024-05-11

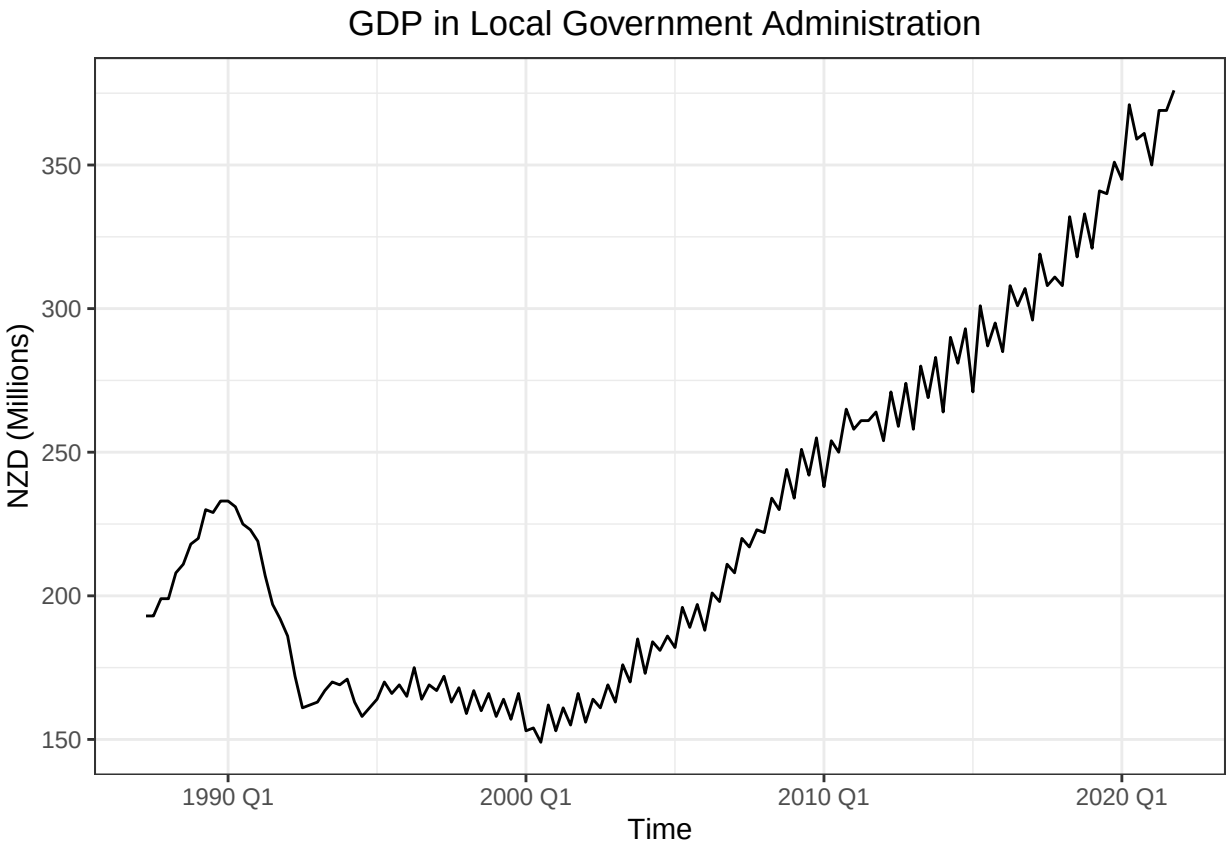
1. Exploratory data analysis

The data set contains information about the quarterly (chain volume) gross domestic product (QGDP) for all ANZSIC06 industry groups in New Zealand, measured in the prices from 2009/10 in NZD Millions from 1987 Q2 until 2021 Q4. Our task is exploring the quarterly GDP in Local Government Administration group.

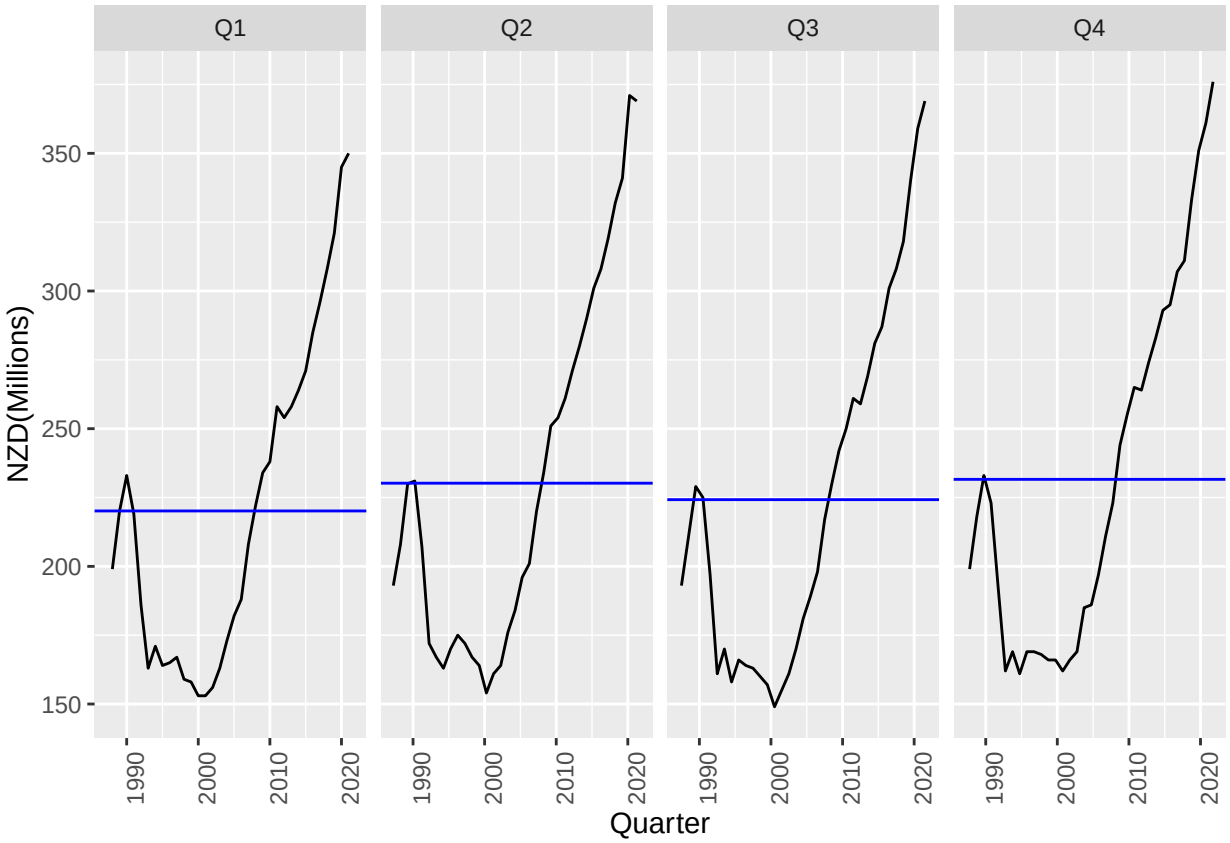
```
# read-in data
train <- read_csv("qgdp_training.csv",show_col_types = FALSE)
# convert data into tsibble
train <- train %>% mutate(Quarter = yearquarter(Date)) %>%
  select(Quarter,`Local Government Administration`) %>%
  as_tsibble(index = Quarter)
```

Firstly, we look at time plot and relevant plots.

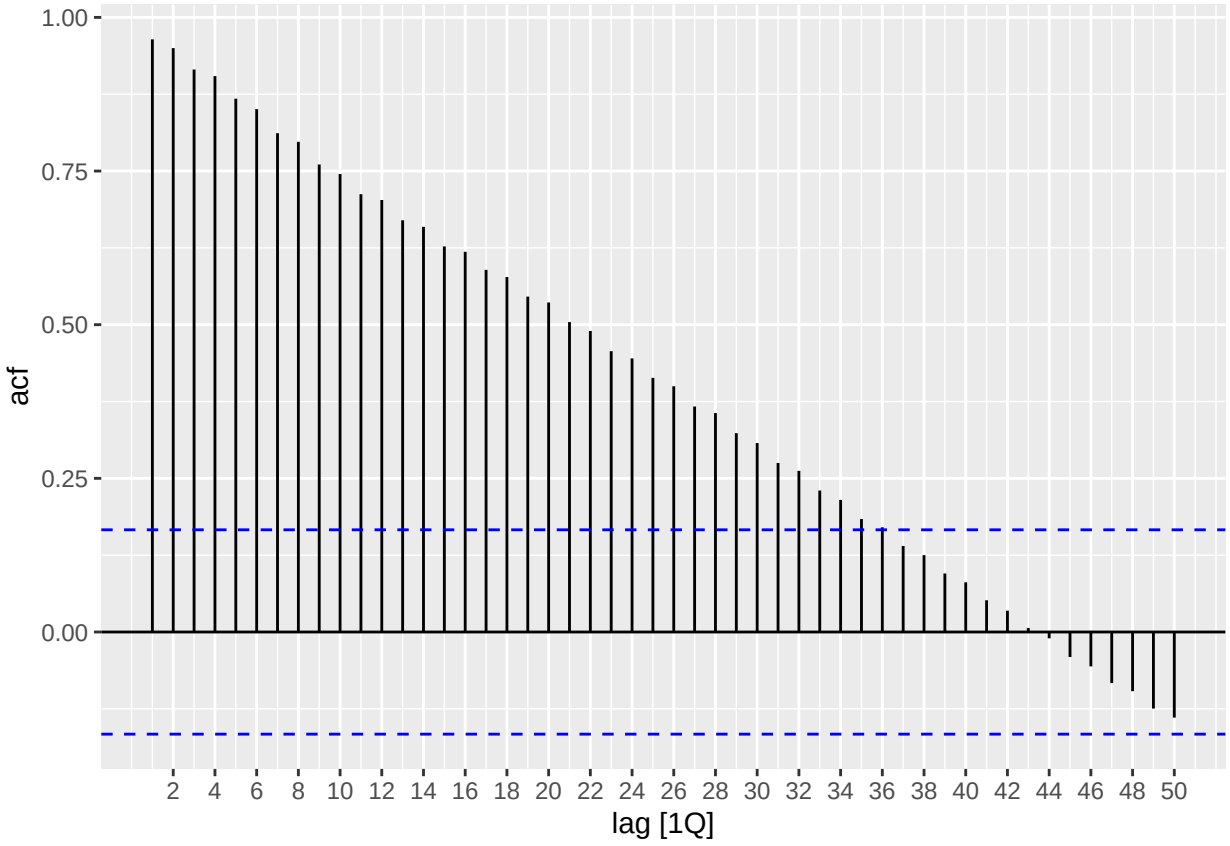
```
# time plot
train %>% autoplot(`Local Government Administration`) + ggtitle(" GDP in Local Government Administratio
  xlab("Time") + ylab("NZD (Millions)") +
  theme_bw()+
  theme(plot.title = element_text(hjust = 0.5))
```




```
# Seasonal subseries plot with gg_subseries
train %>%
  gg_subseries(`Local Government Administration`) +
  ylab("NZD(Millions)")
```



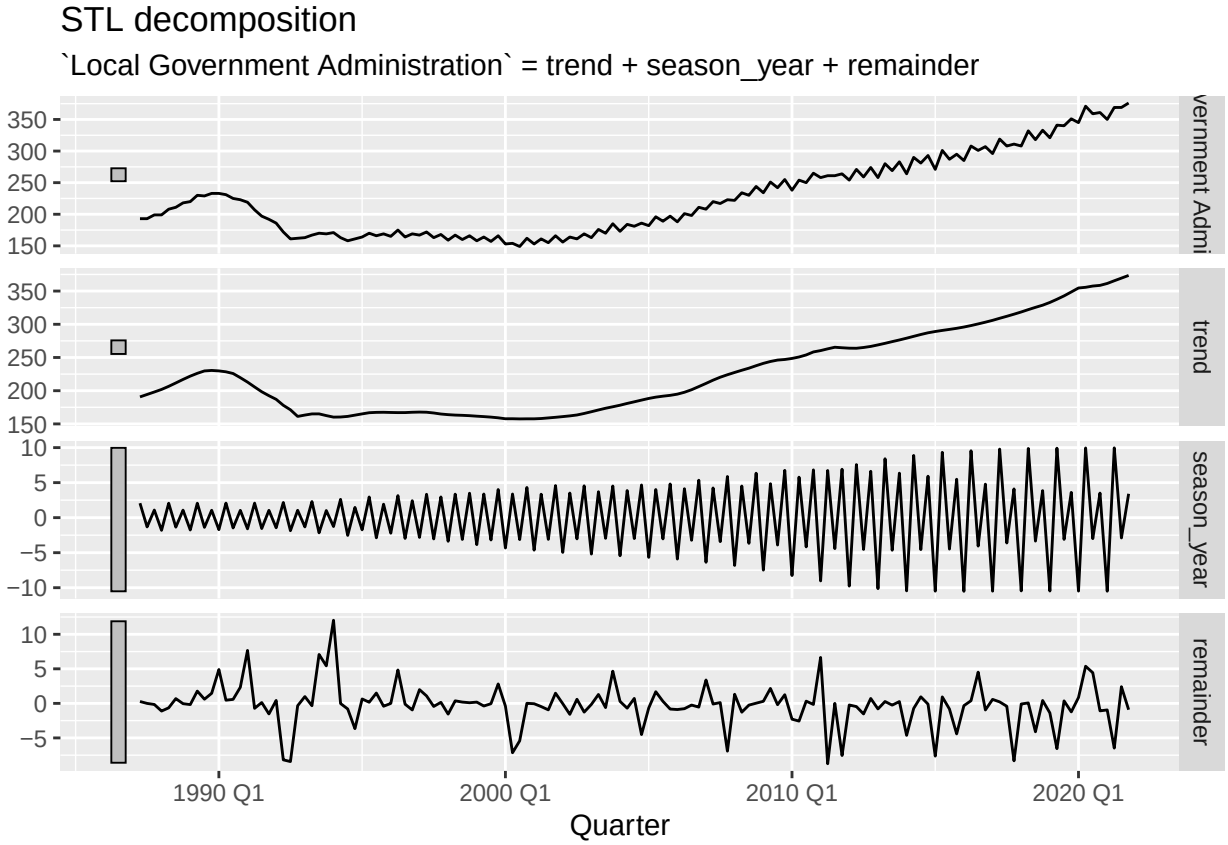

```
# correlogram
train %>%
  ACF(`Local Government Administration`, lag_max = 50) %>%
  autoplot()
```



Looking at the time plot, we observed that this is a trending with structure break and seasonality time series. This is also confirmed by seeing the autocorrelations for small lags tend to be large, positive and decays slowly as the lags increases and the reverse pattern appeared after lag 43, suggesting the structure break is in this trended time series, however the seasonal pattern is not clear in the correlogram. Furthermore, we can see the average GDP for Q4 was the highest value while the average GDP for Q1 was the smallest value based on the sub-series plot. We can also see an increasing year-on-year trend for each quarter in the sub-series plot and the GDP, on average, increases as the year increases.

Next, we have a look at the decomposition plot by using STL decomposition. We decide to use STL model because it allows the seasonal component change over time and we see the magnitude of variation around the trend-cycle does not really vary with the level, so we only consider additive decomposition.

```
# decomposition
stl.dcmp <- train %>%
  model(STL(`Local Government Administration`,robust = TRUE)) %>%
  components()
stl.dcmp%>%
  autoplot()
```



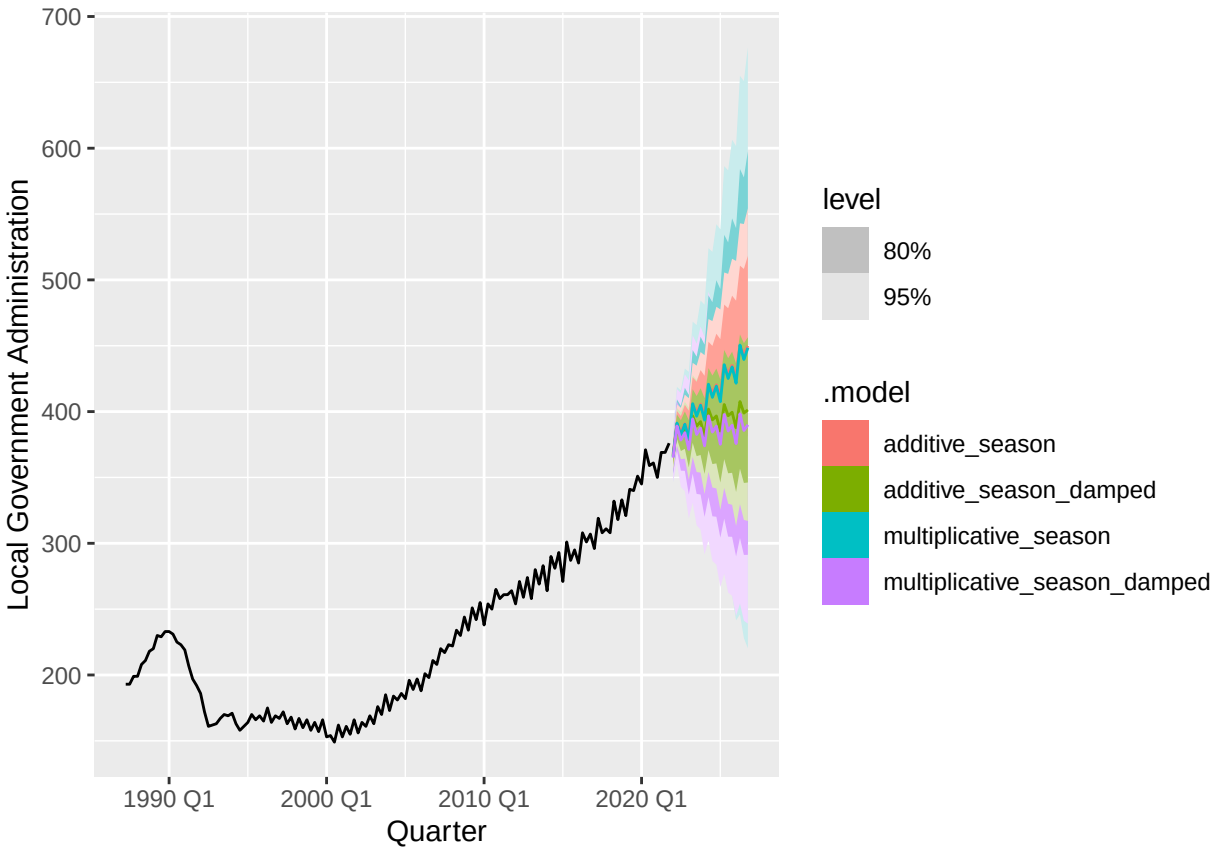
Based on the decomposition result, we can see the trend component dominates the time series, accounting for most of the variability. This is followed by the remainder component, and then the seasonal component. There is a period of time the trend is negative and then stable (from 1990 Q1 until around year of 2003). However, the long-term trend appears to be positive, indicating the GDP in Local Government Administration are increasing. The remainder component appears random, with no discernible patterns, and may be consistent with white noise but we would need further analysis test this. The seasonal component changes over time, where the magnitudes are significantly increasing and stabilizing after 2010.

2. ETS models

Based on the analysis in the first step, we can confirm that there are indeed trends and seasonal changes, but I am currently unable to determine whether to use multiplicative seasonality or additive seasonality to fit the ets model. And it is necessary to consider whether to adopt a pounded trend. So, I decided to manually try out the pros and cons of these four models first.

```
fit <- train %>%
       model(
         additive_season = ETS(`Local Government Administration` ~ error("A") + trend("A") + season("A")
         multiplicative_season = ETS(`Local Government Administration` ~ error("M") + trend("A") + seas
         additive_season_damped = ETS(`Local Government Administration` ~ error("A") + trend("Ad") + se
         multiplicative_season_damped = ETS(`Local Government Administration` ~ error("M") + trend("Ad"
       )

fc <- fit %>% forecast(h = "5 years")
fc %>% autoplot(train)
```




```
# Check model summary
report(fit)
```

```
## Warning in report.mdl_df(fit): Model reporting is only supported for individual
## models, so a glance will be shown. To see the report for a specific model, use
## 'select()' and 'filter()' to identify a single model.
```

```
## # A tibble: 4 x 9
##   .model                     sigma2 log_lik   AIC  AICc   BIC   MSE  AMSE    MAE
##   <chr>                       <dbl>   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
## 1 additive_season           3.08e+1   -577. 1172. 1173. 1198.  29.0  57.1 4.31
## 2 multiplicative_season     7.08e-4   -584. 1185. 1186. 1212.  28.8  56.6 0.0200
## 3 additive_season_damped    2.97e+1   -574. 1168. 1170. 1197.  27.8  52.9 4.33
## 4 multiplicative_season_da~ 6.63e-4   -578. 1176. 1178. 1206.  28.2  53.4 0.0198
```

```
fit_best <- train %>%
  model(ETS(`Local Government Administration`))
report(fit_best)
```

```
## Series: Local Government Administration
## Model: ETS(A,Ad,A)
##   Smoothing parameters:
##     alpha = 0.6961373
##     beta  = 0.1875109
```

```
##     gamma = 0.2832848
##     phi   = 0.89942
##
##   Initial states:
##      l[0]     b[0]      s[0]    s[-1]     s[-2]     s[-3]
##  192.8814 8.293304 -4.655678 4.227328 0.1403426 0.2880069
##
##   sigma^2:  29.7417
##
##      AIC     AICc      BIC
## 1168.152 1169.870 1197.496
```

We can clearly see here that among many aspects of error data, the method of adding seasonality and with Damped trend has the smallest errors, and my method of automatically obtaining the minimum AIC also told me that I should choose the method of adding seasonality and with Damped trend.

ETS Model Introduction

The ETS (Error, Trend, Seasonality) model we selected is specified as an additive error, damped additive trend, and additive seasonality model. Below are the equations representing this model setup:

Model Equations

The ETS model equations for an additive error, damped additive trend, and additive seasonality are as follows:

- **Forecast equation:**
$$\hat{y}_{t|t-1} = (l_{t-1} + \phi b_{t-1}) + s_{t-m}$$
- **Level equation:**
$$l_t = l_{t-1} + \phi b_{t-1} + \alpha e_t$$
- **Trend equation:**
$$b_t = \phi b_{t-1} + \beta e_t$$
- **Seasonal equation:**
$$s_t = s_{t-m} + \gamma e_t$$

Where:

- $\hat{y}_{t|t-1}$ is the one-step ahead forecast,
- l_t is the level component at time t,
- b_t is the trend component at time t,
- s_t is the seasonal component at time t,
- ϕ is the damping factor,
- α, β, γ are the smoothing parameters,
- e_t is the forecast error at time t.

3. ARIMA models

To fit an appropriate ARIMA model, we first need to test the stationary of the time series data. Because if the time series data is stationary, it means that its statistical characteristics are constant over time, allowing for better modeling and prediction. But here we first need to convert the Date column to date format, as it is quarterly data. And at the same time, it is necessary to convert the data into tsible format. Then, we first calculate the seasonal strength, F, on the training data.

```
# Seasonal Strength training data
train %>%
  features(`Local Government Administration`, feat_stl) %>%
  select(seasonal_strength_year)
```

```
## # A tibble: 1 x 1
##   seasonal_strength_year
##                    <dbl>
## 1                  0.858
```

We find that the seasonal strength on the training data is $F = 0.86 > 0.64$. This means that variability of the detrended component from an STL decompsition is much larger than the variability of the remainder component, suggesting data is strongly seasonal. This implies a seasonal-difference is required.

```
# Seasonal Strength on seasonally-differenced training data
train %>%
  features(difference(`Local Government Administration`, 4),feat_stl) %>%
  select(seasonal_strength_year)
```

```
## # A tibble: 1 x 1
##   seasonal_strength_year
##                    <dbl>
## 1                 0.0440
```

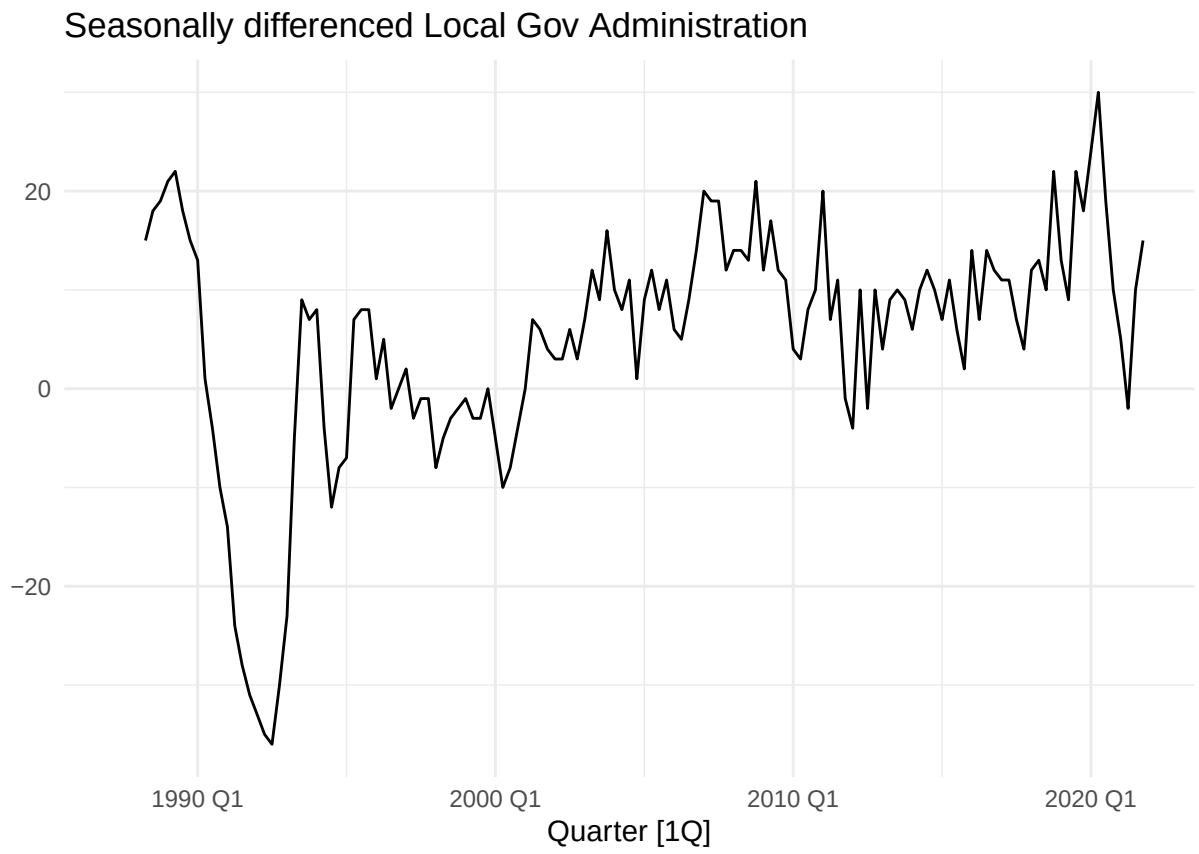
When applying a seasonal-difference and calculating the seasonal strength on this, we find $F = 0.04 < 0.64$, suggesting that we do not need any additional seasonal differencing. That is, $D = 1$. This is verified by the `unitroot_nsdiffs = 1` on the training data below.

```
# Verify with unitroot_nsdiffs feature
train %>%
  features(`Local Government Administration`, unitroot_nsdiffs)
```

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

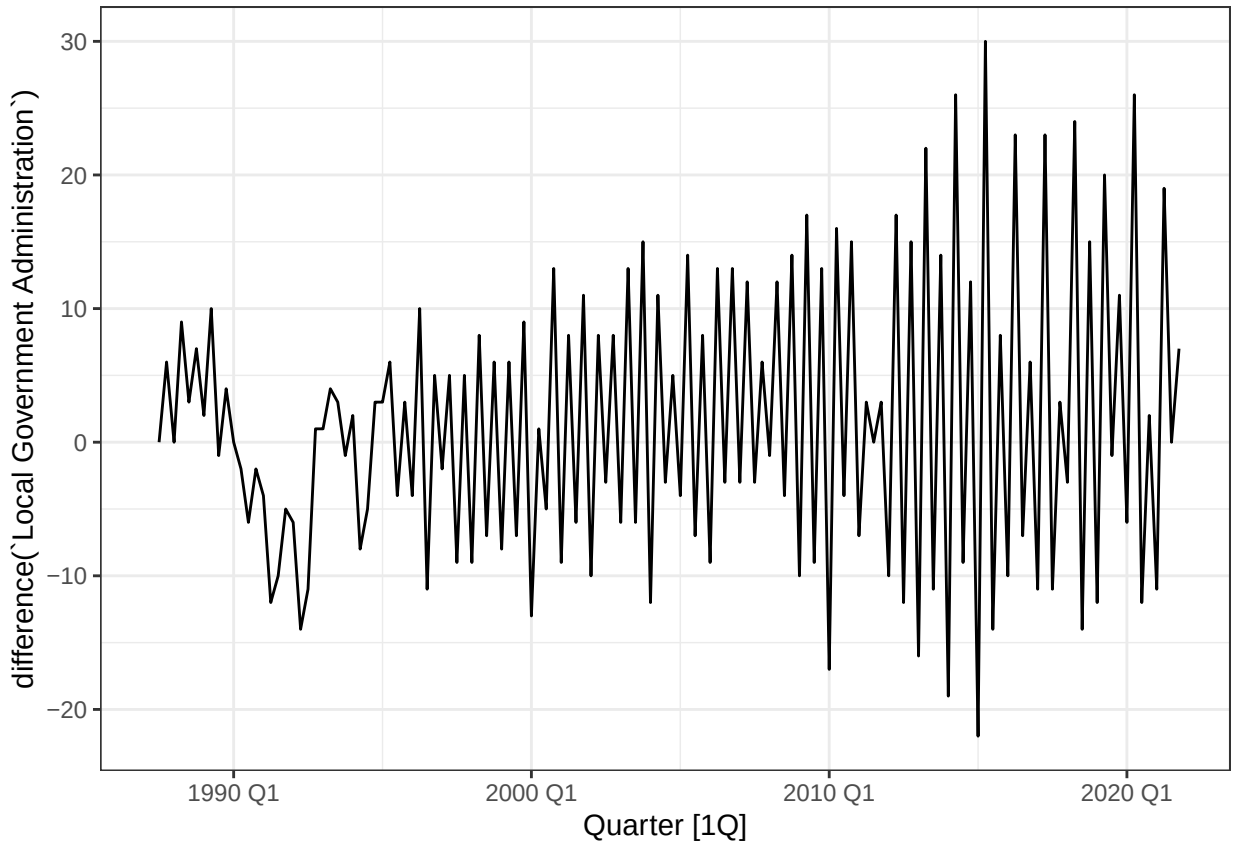
```
train %>%
  autoplot(difference(`Local Government Administration`, 4)) +
  theme_minimal() +
  labs(title= "Seasonally differenced Local Gov Administration", y = " ")
```

```
## Warning: Removed 4 rows containing missing values (‘geom_line()‘).
```




```
train %>%
  autoplot(difference(`Local Government Administration`)) +
  theme_bw()
```

```
## Warning: Removed 1 row containing missing values (‘geom_line()‘).
```



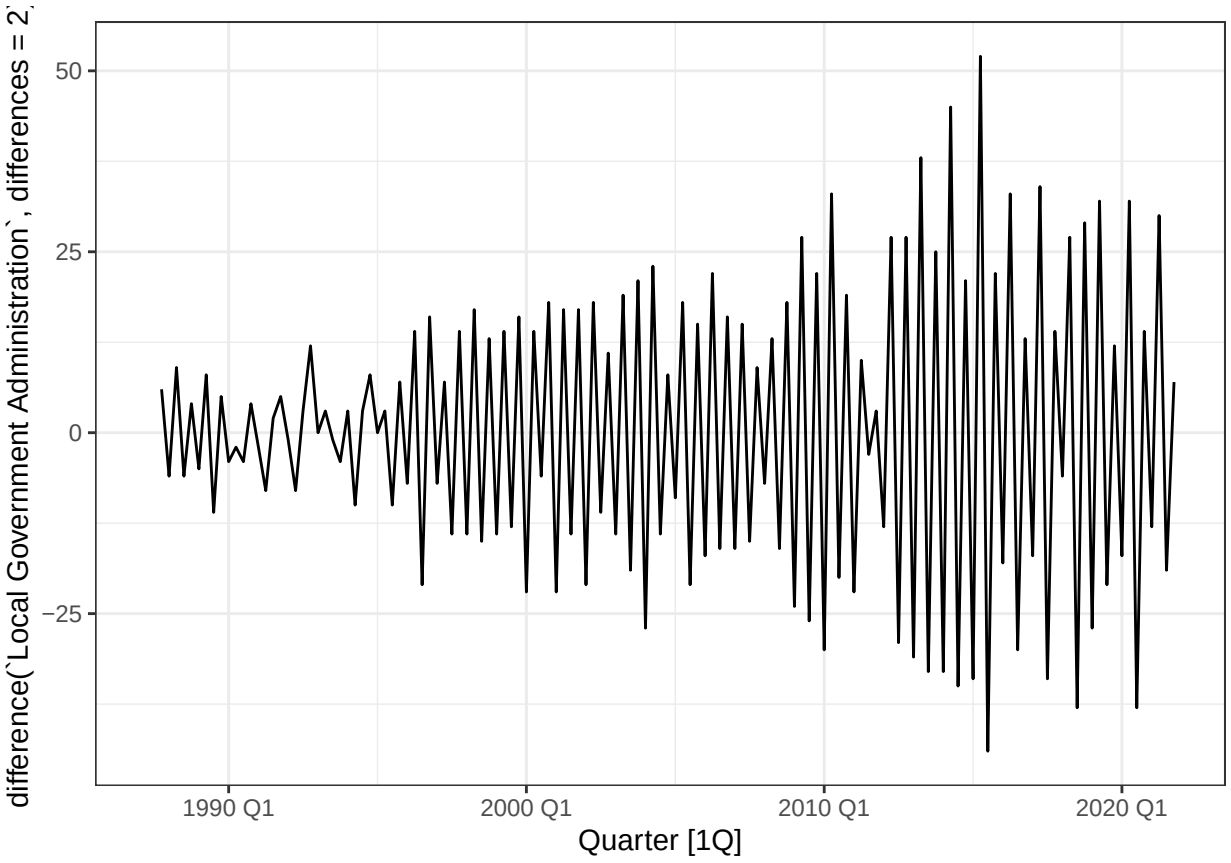

```
# Check the stationary of the original data.
train %>%
  features(difference(`Local Government Administration`), unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1     0.595      0.0231
```

From the test results, according to the KPSS test, the kpss_pvalue is very small, far below the commonly used significance level(0.05), so we reject the null hypothesis that the data is stationary. This means that the data is non-stationary. Therefore, we need to perform a first difference on the data.

```
train %>%
  autoplot(difference(`Local Government Administration`, differences = 2)) +
  theme_bw()
```

```
## Warning: Removed 2 rows containing missing values (‘geom_line()‘).
```




```
# Do the KPSS test.
train %>%
  features(difference(`Local Government Administration`, differences = 2), unitroot_kpss)
```

```
## # A tibble: 1 x 2
##   kpss_stat kpss_pvalue
##       <dbl>       <dbl>
## 1    0.0111         0.1
```

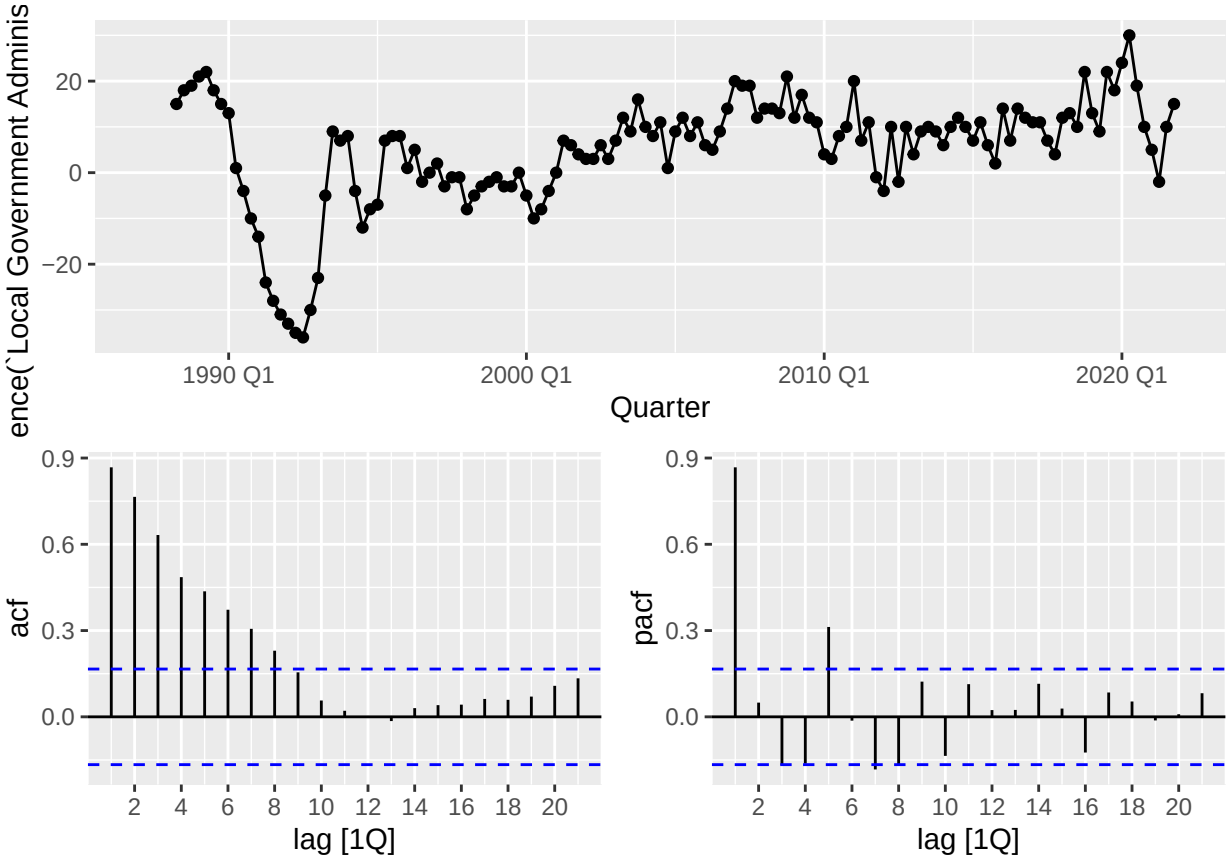
After performing a first difference on the data, we conducted a KPSS test again and found that kpss_pvalue was still below the significance level(0.05). So we still reject the null hypothesis that the data is stationary. This means that the data is still non-stationary. Therefore, we need to perform a second-order difference on the data.

From the results, we can see that the kpss_pvalue of the data after the second-order difference is greater than the significance level(0.05). There is no evidence against the null hypothesis that the twice differenced series is stationary (p-value = 0.1) in terms of the mean. The variance looks "okay" too because the large peaks are only transient and not persistent.

```
# Draw the ACF plot and PACF plot with the second-order difference data.
train %>%
  gg_tsdisplay(difference(`Local Government Administration`, 4),
               plot_type = "partial")
```

```
## Warning: Removed 4 rows containing missing values (‘geom_line()‘).
```

```
## Warning: Removed 4 rows containing missing values (`geom_point()`).
```



After applying one seasonal difference to the training set, we can be convinced the time series is now stationary. To determine candidate models we can look at the ACF and PACF plots. For a purely SAR(P) model, we would see exponential decay or damped sinusoidal behaviour in the ACF's seasonal frequencies (i.e., at 4, 8, 12, etc.), and significant peaks in the PACF at 4P (because it's a quarterly series) and no other seasonal frequencies after that. On the other hand, for a purely SMA(Q) model, we would see significant peaks in the ACF at 4Q and no other seasonal frequencies after that, and exponential decay or damped sinusoidal behaviour in the PACF's seasonal frequencies.

The ACF has a significant spike at lag 4 and 8 and none after that, and the PACF has no significant spike. This would suggest that either a model with P=0 seasonal AR terms, or Q=2 seasonal MA terms would be good candidates. To determine the non-seasonal AR and MA parts, we would restrict our attention to the lags before the first seasonal lag (i.e., 4). For a non-seasonal AR(p), we would see a significant spike at lag p and none after that in the PACF, as well as exponential decay or sinusoidal dampening in the ACF. For a non-seasonal MA(q), we would see a significant spike at lag q and none after that in the ACF, as well as exponential decay or sinusoidal dampening in the PACF. We see that there are 3 significant spikes in the ACF at lags 1, 2, and 3, and therefore we would choose p=0. There is 1 significant spike in the PACF at lag 1, thus q=0 in our candidate models.

Our two candidate models are therefore: ARIMA(1,2,0)(0,1,2)4 and ARIMA(0,2,3)(0,1,2)4.

```
# Fit the ARIMA model and store them in the shortlist.
fit= train %>%
  model(m1 = ARIMA(`Local Government Administration` ~ 1+ pdq(1, 2, 0) + PDQ(0,1,2)),
        m2 = ARIMA(`Local Government Administration` ~  1+ pdq(0, 2, 3) + PDQ(0,1,2)),
```

```
        stepwise = ARIMA(`Local Government Administration`),
        search = ARIMA(`Local Government Administration`, stepwise = FALSE))
```

```
## Warning: Having 3 or more differencing operations is not recommended. Please
## consider reducing the total number of differences.
```

```
## Warning: Model specification induces a quadratic or higher order polynomial trend.
## This is generally discouraged, consider removing the constant or reducing the number of differences.
```

```
## Warning: Having 3 or more differencing operations is not recommended. Please
## consider reducing the total number of differences.
```

```
## Warning: Model specification induces a quadratic or higher order polynomial trend.
## This is generally discouraged, consider removing the constant or reducing the number of differences.
```

```
# Report the summary of the models.
fit %>%
  glance() %>%
  select(.model, AICc)
```

```
## # A tibble: 4 x 2
##   .model    AICc
##   <chr>    <dbl>
## 1 m1        872.
## 2 m2        846.
## 3 stepwise  832.
## 4 search    832.
```

```
best_model <- fit %>%
  select(search) %>%
  report()
```

```
## Series: Local Government Administration
## Model: ARIMA(2,1,0)(0,1,1)[4]
##
## Coefficients:
##           ar1     ar2     sma1
##       -0.0764  0.2787  -0.6220
## s.e.   0.0833  0.0858   0.0868
##
## sigma^2 estimated as 27.54:  log likelihood=-411.73
## AIC=831.46   AICc=831.77   BIC=843.05
```

Based on the results of search model, we can write the fitted model equation in backshift notation:

$$(1 - 0.0764B - 0.2787B^2)(1 - B)^2 y_t = (1 - 0.6220B^4)\epsilon_t$$

4. Neural network autoregression (NNAR) models

Neural Network Autoregression (NNAR) model is a type of forecasting model that combines the principles of neural networks and autoregressive (AR) models. NNAR models are useful in time series forecasting, where past values of a series are used to predict future values.

Neural networks consist layers of interconnected nodes (neurons) that learn complex patterns in data through training. They are powerful for modelling non-linear relationships and also allow complex non-linear relationships between the response variable and its predictors.

The NNAR model combines these two approaches, which uses previous time series values (like the AR model) as input features for a neural network. The neural network then processes the inputs to predict future values.

For time series data, lagged values of the time series can be used as inputs to a neural network. This is the Neural Network Autoregression (NNAR) model.

Structure of the NNAR model:

1. **Input Layer**: Receives p lagged values of the time series (similar to AR model inputs).
2. **Hidden Layers**: One or more hidden layers , with neurons that apply non-linear transformations to the inputs. The number of neurons and hidden layers can vary based on the complexity required.
3. **Output Layer**: Typically a single neuron that outputs the predicted future value.

The notation NNAR(p, k) indicates there are p lagged inputs and k nodes in the hidden layer. For example, a NNAR(8,4) model is a neural network with the last eight observations $(y_{t-1}, y_{t-2}, \cdots, y_{t-8})$ used as inputs for forecasting the output y_t, and has four neurons in the hidden layer.

A NNAR(p,0) model is equivalent to an ARIMA(p,0,0) model, in terms of using p past observations for forecasting. But, note that there are no restrictions on the parameters to ensure stationary.

With seasonal data, we can also add the last observed values from the same season as inputs. NNAR(p, P, k)m model has inputs $(y_{t-1}, y_{t-2}, \cdots, y_{t-p}, y_{t-m}, y_{t-2m}, \cdots, y_{t-Pm})$ and k neurons in the hidden layer. P is the number of lagged seasonal observations used as inputs.

In R, the **NNETAR()** function fits an NNAR(p, P, k)m model. If the values of p and P are not specified, these will be selected automatically. The values selected is the optimal values.

For seasonal time series, the default values are $P = 1$ and p is chosen from the optimal linear model fitted to the seasonally adjusted data. If k is not specified, it is set to $k = (p + P + 1)/2$ (round to the nearest integer).

```
nnar_model <- train %>% model(NNETAR(`Local Government Administration`))
report(nnar_model)
```

```
## Series: Local Government Administration
## Model: NNAR(1,1,2)[4]
##
## Average of 20 networks, each of which is
## a 2-2-1 network with 9 weights
## options were - linear output units
##
## sigma^2 estimated as 46.7
```

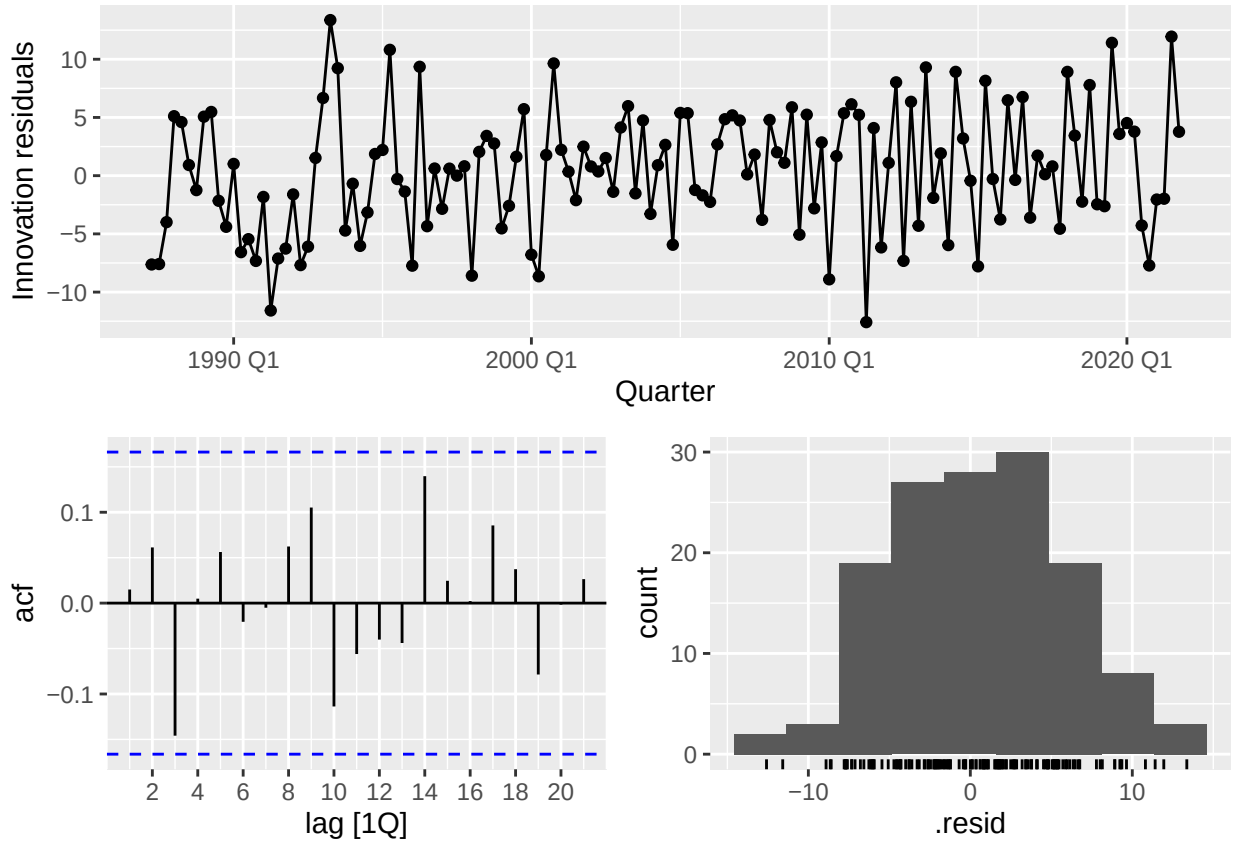
We will use the NNETAR() function. Since we are looking to fit an automatic NNAR model to our dataset, we do not manually specify the values for the parameters p (non-seasonal lags), P (seasonal lags), and k (number of neurons). This is because the NNETAR() function automatically selects the optimal values for the parameters based on the characteristics of the time series data.

From the output, `Model: NNAR(1,1,2)[4]` , this takes the format NNAR(p, P, k)m. This indicates that there is 1 lagged observation used as input (autoregressive term)- p . There is 1 lagged seasonal observation used as input - P. There are 2 nodes in the hidden layer - k. The number 4 means quarterly data. m is the seasonal frequency.

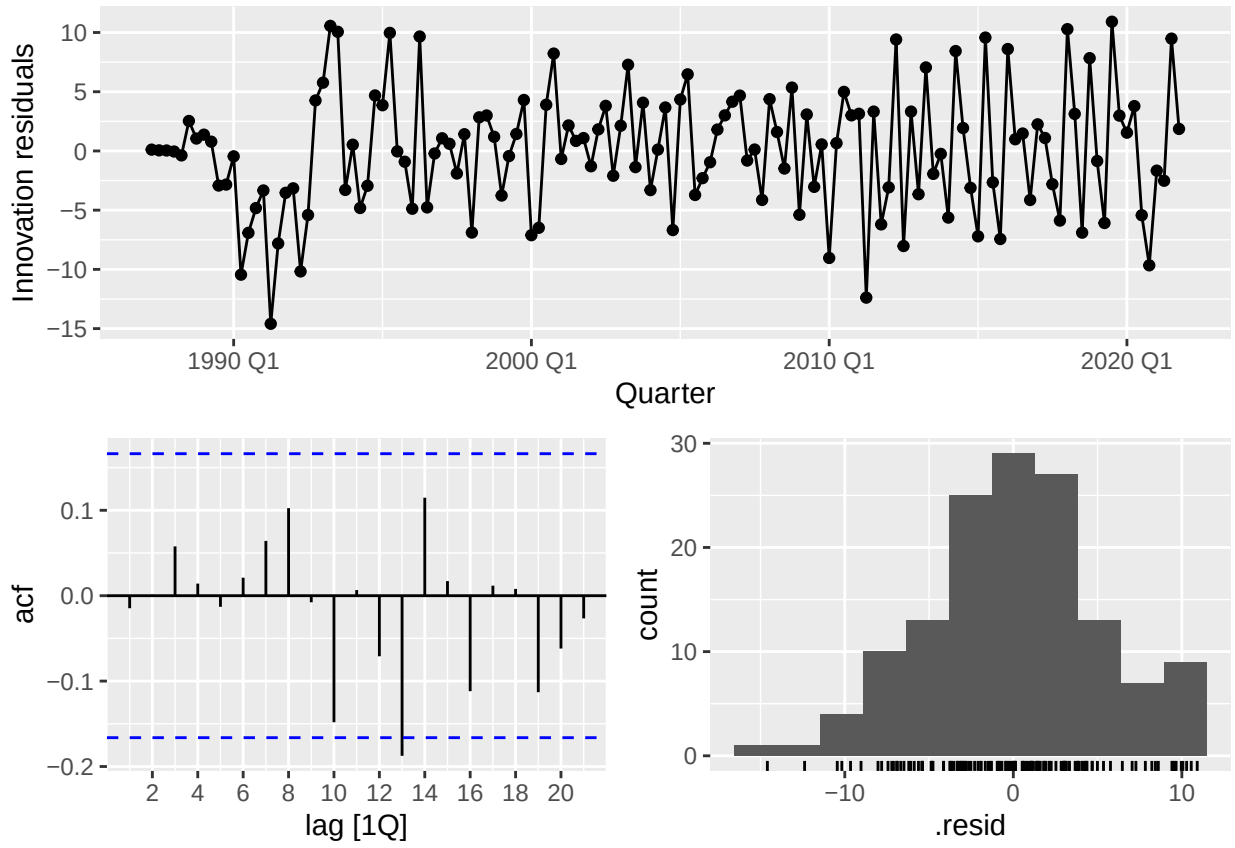
The numbers (2-2-1) indicates each network has the structure of 2 input nodes, 2 hidden nodes, and 1 output node. The 2 input nodes correspond to the number of autoregressive and seasonal terms (p+ P). 2 hidden nodes and 1 output node that produce the model's prediction.

5. Assumption checking

```
# residual diagnostic plots for chosen ETS model
fit_best %>% gg_tsresiduals()
```




```
# residual diagnostic plots for chosen ARIMA model
best_model %>% gg_tsresiduals()
```

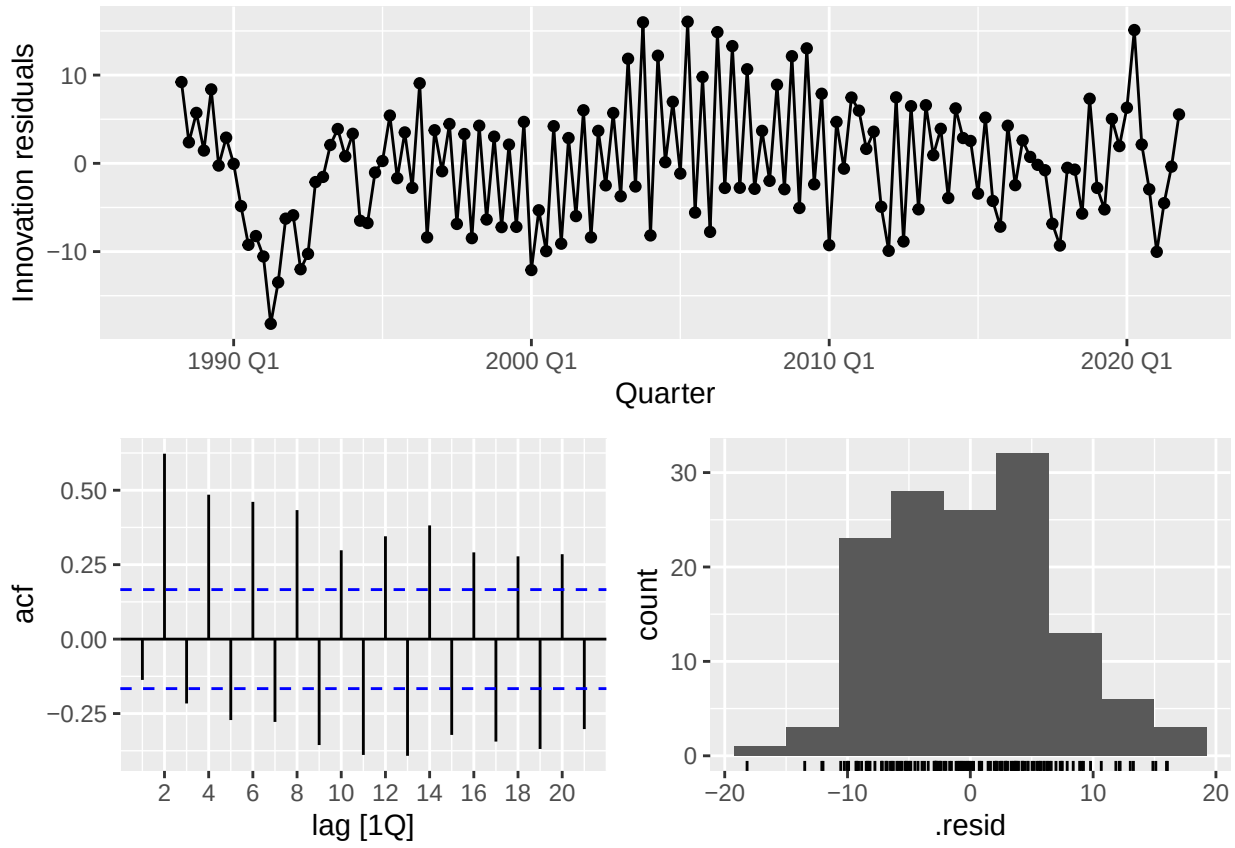



```
# residual diagnostic plots for chosen NNAR model
nnar_model %>% gg_tsresiduals()
```

```
## Warning: Removed 4 rows containing missing values (‘geom_line()‘).
```

```
## Warning: Removed 4 rows containing missing values (‘geom_point()‘).
```

```
## Warning: Removed 4 rows containing non-finite values (‘stat_bin()‘).
```



Surrogate data testing is a statistical proof by contradiction technique which is similar with permutation tests. The key idea is to generate surrogate datasets that preserve some aspects of the original data while randomizing others, and then compare the statistical properties of the original data to those of the surrogates. This helps to determine whether the observed properties in the original data are genuine or could have arisen by chance. Specifically, using Random Shuffle method creates new data simply by random permutations of the original series. This will destroy any temporal correlation that may have been in the original data but still guarantee the same amplitude distribution as the original series. This method is associated to the null hypothesis of the data being independent.

In `surrogate_test.R`, inputs are the time series that we want to expect the independence having one measure variable(if not, the function will stop), number of lags in portmanteau test statistic, number of permutations to perform(default = 1000), which type of Portmanteau test that we want(default = ljung-box) and outputs are Q values included Q observed and Q null hypothesis and p-value.

- Firstly, compute the observed Q statistic of the original time series corresponding with either ljung-box test or box-pierce test using all the autocorrelation from lag 1 to the given `lag` in input.
- Next, generate `N` samples by random permutations of the original time series, this will kill autocorrelation while maintaining amplitude. Then, computing Q statistics on each sample and comparing the original statistic to the distribution of surrogate statistics, which gives us p-value. If p-value is significant(smaller than 0.05, we reject the null hypothesis, indicating the time series is not independent otherwise our time series is independent.

```
source('surrogate_test.R')
# perform ljung-box test for ETS model
res1 <- fit_best %>% augment() %>% pull(.resid)
```

```
s1 <- surrogate.test(res1,lag=8)
s1$p.value # p-value
```

```
## [1] 0.775
```

```
# perform ljung-box test for ARIMA model
res2 <- best_model %>% augment() %>% pull(.resid)
s2 <- surrogate.test(res2,lag=8)
s2$p.value # p-value
```

```
## [1] 0.943
```

```
# perform ljungbox test for NNAR model
res3 <- nnar_model %>% augment() %>% pull(.resid) %>% na.omit() %>% as.vector()
s3 <- surrogate.test(res3,lag=8)
s3$p.value
```

```
## [1] 0
```

The time plot of residuals of ETS model show that the residuals appear to have zero mean and constant variance. The histogram of the residuals appears to be symmetric and normally distributed. The correlogram of the residuals shows insignificant autocorrelation and the p-value from surrogate test gives us no evidence against the null hypothesis, indicating that residuals are independent.

The time plot of residuals of ARIMA model show that the residuals appear to have zero mean and constant variance. The histogram of the residuals appears to be symmetric and normally distributed. The correlogram of the residuals only shows significant autocorrelation at lag 13 (it's okay) and the p-value from surrogate test gives us no evidence against the null hypothesis, indicating that residuals are independent.

However, interpreting residual diagnostics for the NNAR model is not very useful because there are not any assumptions about the residuals or the distribution of the forecasts.

6. Forecasting

```
ets_fit <- train %>%
  model(
    additive_season_damped = ETS(`Local Government Administration` ~ error("A") + trend("Ad") + season(
  )

fit_arima <- train %>%
  model(
    search = ARIMA(`Local Government Administration`, stepwise = FALSE)
  )

nnar_model <- train %>% model(NNETAR(`Local Government Administration`))

ets_forecast <- ets_fit %>% forecast(h = 8)
ets_forecast <- ets_forecast %>% hilo(level = 95) %>% as_tibble()
print(ets_forecast)
```

```
## # A tibble: 8 x 5
##   .model              Quarter Local Government Adm~1 .mean                 `95%`
##   <chr>                 <qtr>                 <dist> <dbl>                <hilo>
## 1 additive_season_d~ 2022 Q1             N(367, 30)  367. [355.8870, 377.2646]95
## 2 additive_season_d~ 2022 Q2             N(389, 52)  389. [374.4366, 402.6993]95
## 3 additive_season_d~ 2022 Q3             N(382, 83)  382. [363.9281, 399.5787]95
## 4 additive_season_d~ 2022 Q4            N(386, 122)  386. [364.2184, 407.5591]95
## 5 additive_season_d~ 2023 Q1            N(376, 195)  376. [348.2038, 402.8757]95
## 6 additive_season_d~ 2023 Q2            N(397, 252)  397. [365.5381, 427.7226]95
## 7 additive_season_d~ 2023 Q3            N(389, 317)  389. [354.0941, 423.9156]95
## 8 additive_season_d~ 2023 Q4            N(392, 391)  392. [353.6551, 431.1666]95
## # i abbreviated name: 1: `Local Government Administration`
```

```
arima_forecast <- fit_arima %>% forecast(h = 8)
arima_forecast <- arima_forecast %>% hilo(level = 95) %>% as_tibble()
print(arima_forecast)
```

```
## # A tibble: 8 x 5
##   .model Quarter `Local Government Administration` .mean                 `95%`
##   <chr>    <qtr>                             <dist> <dbl>                <hilo>
## 1 search 2022 Q1                          N(369, 28)  369. [358.7107, 379.2827]95
## 2 search 2022 Q2                          N(391, 51)  391. [376.6488, 404.6531]95
## 3 search 2022 Q3                          N(386, 91)  386. [366.9423, 404.3853]95
## 4 search 2022 Q4                         N(393, 129)  393. [370.3985, 414.8565]95
## 5 search 2023 Q1                         N(384, 201)  384. [356.4202, 412.0539]95
## 6 search 2023 Q2                         N(406, 270)  406. [373.7779, 438.1963]95
## 7 search 2023 Q3                         N(401, 351)  401. [363.9042, 437.3078]95
## 8 search 2023 Q4                         N(408, 429)  408. [367.0277, 448.2252]95
```

```
nnar_forecast <- nnar_model %>% forecast(h = 8)
nnar_forecast <- nnar_forecast %>% hilo(level = 95) %>% as_tibble()
print(nnar_forecast)
```

```
## # A tibble: 8 x 5
##   .model              Quarter Local Government Adm~1 .mean                 `95%`
##   <chr>                 <qtr>                 <dist> <dbl>                <hilo>
## 1 NNETAR(`Local Gov~ 2022 Q1          sample[5000]  366. [352.2809, 379.3765]95
## 2 NNETAR(`Local Gov~ 2022 Q2          sample[5000]  375. [361.1404, 387.8927]95
## 3 NNETAR(`Local Gov~ 2022 Q3          sample[5000]  375. [361.9943, 389.2390]95
## 4 NNETAR(`Local Gov~ 2022 Q4          sample[5000]  379. [365.3529, 392.2016]95
## 5 NNETAR(`Local Gov~ 2023 Q1          sample[5000]  374. [359.0139, 388.9493]95
## 6 NNETAR(`Local Gov~ 2023 Q2          sample[5000]  378. [362.7337, 392.9159]95
## 7 NNETAR(`Local Gov~ 2023 Q3          sample[5000]  379. [363.6803, 392.9764]95
## 8 NNETAR(`Local Gov~ 2023 Q4          sample[5000]  380. [365.7103, 394.5896]95
## # i abbreviated name: 1: `Local Government Administration`
```

ETS Prediction Interval Construction

The prediction intervals for ETS models can be constructed based on the error type (additive or multiplicative) and the specific form of the model. Specifically:

For additive error models (e.g., ETS(A,N,A)):

- The prediction distribution is Gaussian.
- Prediction intervals are calculated using a Gaussian approximation.
- For most ETS models, the following formula can be used:

$$\hat{y}_{T+h|T} \pm z_{\alpha/2}\hat{\sigma}_h$$

where $z_{\alpha/2}$ is the $\alpha/2$ quantile of the standard Gaussian distribution, and $\hat{\sigma}_h$ is an estimate of the forecast standard deviation.

Estimation of Forecast Standard Deviation

For additive error and damped trend ETS models (e.g., ETS(A,Ad,A)), the forecast standard deviation can be estimated using the following formula:

$$\sigma_h^2 = \sigma^2 \left[1 + \alpha^2(h-1) + \left(\frac{1-\phi^h}{1-\phi}\right)^2 (2\alpha(1-\phi) + \beta\phi)\right] + \gamma k(2\alpha+\gamma) + 2\left(\frac{\beta\gamma\phi}{1-\phi}\right) k(1-\phi^m) - \phi^m(1-\phi^{mk})$$

ARIMA Prediction Interval Construction

ARIMA models typically use prediction standard error to calculate prediction intervals. It calculates the residual sequence by fitting the ARIMA model and calculate its variance. It reflects the prediction error of the model. Due to the fact that the ARIMA model is based on a combination of autoregressive and moving average parts, its prediction error accumulates with the increase of prediction period. So the calculation of the prediction standard error can be expressed as:

$$\hat{\sigma}_h = \hat{\sigma}\sqrt{1 + \theta_1^2 + \theta_2^2 + \cdots + \theta_h^2}$$

where θ_i is a parameter of the MA part.

Finally, we can calculate the prediction interval:

$$\hat{y}_{T+h|T} \pm z_{\alpha/2} \cdot \hat{\sigma}_h$$

where $\hat{y}_{T+h|T}$ is the predicted value, $z_{\alpha/2}$ is the critical value of the standard normal distribution, $\hat{\sigma}_h$ is the prediction standard error.

NNAR Prediction Interval Construction

The neural network fitted to our data can be written in the form $(y_t = f(y_{t-1}) + \epsilon t)$. $(y_{t-1} = y_{t-1}, y_{t-2})$, is the vector that contains lagged values of the series. (f) is a neural network with 2 hidden nodes in a single layer. The error series (ϵt) is assumed to be homoscedasticity, which means error is constant, and normally distributed.

The prediction intervals for an NNAR model are typically calculated based on the residuals of the model and the assumption of normality. The forecast() package does the calculation.

1. **Residuals Calculation**: The residuals from the fitted NNAR model are computed. The residual is the differences between the observed values and the fitted values.
2. **Standard Error Estimation**: The standard error of the residuals is estimated. This is done by assuming that the residuals follow a normal distribution.

3. **Prediction Intervals**: Use the estimated standard error, prediction intervals are calculated by adding and subtracting a multiple of the standard error (based on the chosen confidence level, e.g, 1.96 for a 95% confidence interval) from the point forecasts.

```
test <- read_csv("qgdp_full.csv", show_col_types = FALSE)
# convert into tsibble
test <- test %>%
  mutate(Quarter = yearquarter(Date)) %>%
  select(Quarter, `Local Government Administration`) %>%
  as_tsibble(index = Quarter)
# model forecasts
ets_forecast <- ets_fit %>% forecast(h = 8)
arima_forecast <- fit_arima %>% forecast(h = 8)
nnar_forecast <- nnar_model %>% forecast(h = 8)
# measures of accuracy
ets_metrics <- accuracy(ets_forecast, test)
arima_metrics <- accuracy(arima_forecast, test)
nnar_metrics <- accuracy(nnar_forecast, test)
# show results
print(ets_metrics)
```

```
## # A tibble: 1 x 10
##   .model                 .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 additive_season_damped Test   13.0  17.0  13.7  3.16  3.37  1.30  1.30 0.373
```

```
print(arima_metrics)
```

```
## # A tibble: 1 x 10
##   .model .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>  <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 search Test   5.45  9.15  7.86  1.31  1.95 0.745 0.702 0.221
```

```
print(nnar_metrics)
```

```
## # A tibble: 1 x 10
##   .model                    .type    ME  RMSE   MAE   MPE  MAPE  MASE RMSSE  ACF1
##   <chr>                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 NNETAR(‘Local Governmen~ Test   21.8  25.4  22.2  5.35  5.48  2.11  1.95 0.385
```

Summary

The ARIMA model performs the best among all models, with the lowest RMSE, MAE, MASE and MAPE, indicating the highest prediction accuracy and the smallest error.

The predictive performance of the ETS model is second, although RMSE and MAE are relatively high, MAPE is relatively low, indicating that it performs well in terms of relative error.

The NNAR model performs the worst, with the highest RMSE, MAE, MASE and MAPE, indicating that it has the highest prediction error and the lowest accuracy.