Time Series Analysis

UC Berkeley, School of Information: MIDS w271

2023-12-02

Contents

Problem Set 11 **5**

1 (10 points) ARIMA model **7**

1.1 Time series plot 7

1.2 Fit a model 9

1.3 Is this model appropriate? 9

1.4 Forecasts, by hand! 13

1.5 Interpret roots 14

2 (10 points) Seasonal ARIMA model **17**

2.1 Time series plot 20

2.2 Check for Stationary 21

2.3 Model identification and estimation 31

2.4 Model diagnostic 32

2.5 Forecasting 37

3 (10 points) Time Series Linear Model and Cointegration **43**

3.1 Plot electricity 44

3.2 Cointegration test 46

3.3 Fit Model 48

3.4 Residuals Plot 49

3.5 Forecasting model 50

4 (12 points) Vector autoregression **53**

4.1 Time series plot 53

4.2 Check for the unit root 57

4.3 Determine VAR model 61

4.4 Estimation 61

4.5 Model diagnostic 64

4.6 Forecasting 65

Problem Set 11

This is the final problem set that you will have to work on for this class. Congratulations! (Although there is still a group lab that will be the final assignment in the course.)

You will start with some guided work, and then proceed into less structured work that will let you stretch and demonstrate what you have learned to date.

Notice, in particular, that the last few questions are asking you essentially to "produce a model" using a method. At this point in the course, you should be familiar with many of the model forms that you *might* fit; and, you are familiar with methods that you can use to evaluate models' performances. In these questions, we are asking you to, essentially, fit a good model with a method and then to evaluate how a good model with "this" method is doing compared to another good model with "that" method.

In several of these questions, there isn't a correct answer, *per se*. Instead, there is the process that you will undertake and record as you are producing your argument for the model that you think is best meeting your objectives. This is a **very** applied task that we anticipate you will see many times in your work.

```
knitr::opts_chunk$set(echo=TRUE)
```

We are providing you with an additional challenge, but one that is also very evocative of work that you're likely to come across. This is a well-built repository, that uses a well-documented framework to produce reports, namely `bookdown`.

Once you have done your work, you can render the entire book using the following call in your console:

```
> bookdown::render_book()
```

This will ingest each of the files `01-time_series_...`, `02-cross_validation.Rmd`, `03-ARIMA_model.Rmd` and so on ... and produce a PDF that is stored in `./_book/_main.pdf`. If you would like to read more about this framework, you can do so at the following website: https://bookdown.org/yihui/bookdown/ .

Chapter 1

(10 points) ARIMA model

Consider `fma::sheep`, the sheep population of England and Wales from 1867–1939. :sheep:

```
head(fma::sheep)
```

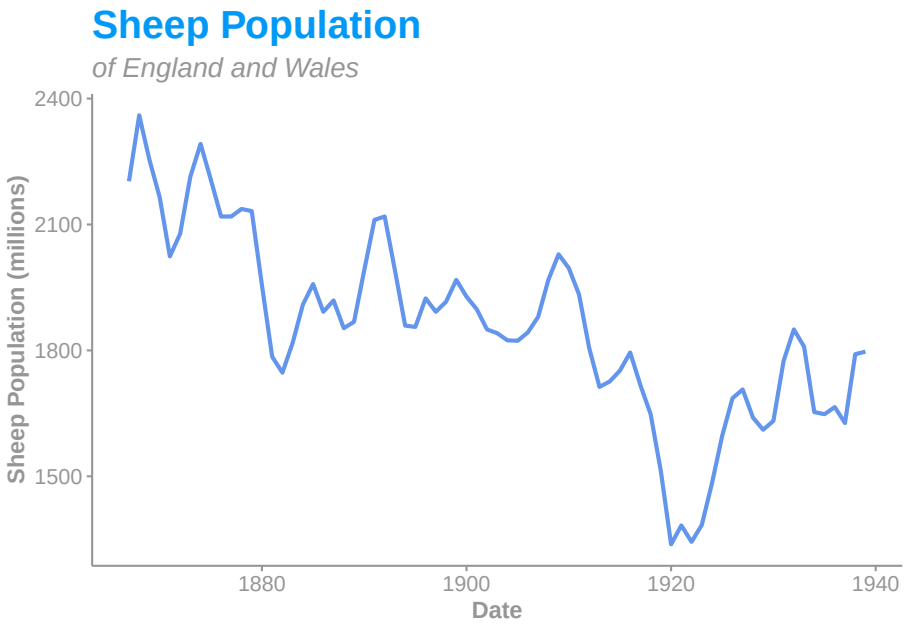
```
## Time Series:
## Start = 1867
## End = 1872
## Frequency = 1
## [1] 2203 2360 2254 2165 2024 2078
```

1.1 Time series plot

Produce a time plot of the time series, and comment on what you observe.

```
# Sheep TS plot
sheep_plot <- autoplot(
  fma::sheep,
  colour="cornflowerblue",
  size = 1
  ) +
  labs(
    title = "Sheep Population",
    subtitle = "of England and Wales",
    x = "Date",
    y = "Sheep Population (millions)"
       ) +
```

```
    theme_classic() +
    theme(
    plot.title = element_text(color = "#0099F8",
                              size = 20,
                              face = "bold"),
    plot.subtitle = element_text(color="#969696",
                                 size = 14,
                                 face = "italic"),
    axis.title = element_text(color = "#969696",
                              size = 12,
                              face = "bold"),
    axis.text = element_text(color = "#969696", size = 11),
    axis.line = element_line(color = "#969696"),
    axis.ticks = element_line(color = "#969696"),
    )
sheep_plot
```



We can observe there's a negative trend (sheep population decreased as time passed), we have a couple of peaks which might mean certain seasonality or perhaps specific events that ocurred during those years.

1.2 Fit a model

Assume you decide to fit the following model:

$$y_t = y_{t-1} + \phi_1(y_{t-1} - y_{t-2}) + \phi_2(y_{t-2} - y_{t-3}) + \phi_3(y_{t-3} - y_{t-4}) + \epsilon_t,$$

where ϵ_t is a white noise series.

1.2.1 Model type

What sort of ARIMA model is this (i.e., what are p, d, and q)?

> We're looking at an ARIMA(3,1,0) model. Meaning $p = 3$, $d = 1$, and $q = 0$.

1.2.2 Back to the future

Express this ARIMA model using backshift operator notation.

> We have the following model:
>
> $$y_t = y_{t-1} + \phi_1(y_{t-1} - y_{t-2}) + \phi_2(y_{t-2} - y_{t-3}) + \phi_3(y_{t-3} - y_{t-4}) + \epsilon_t$$
>
> $$\begin{aligned} \iff \epsilon_t &= y_t - y_{t-1} - \phi_1(y_{t-1} - y_{t-2}) - \phi_2(y_{t-2} - y_{t-3}) - \phi_3(y_{t-3} - y_{t-4}) \\ &= (1 - \mathbb{B})y_t - \phi_1(1 - \mathbb{B})y_{t-1} - \phi_2(1 - \mathbb{B})y_{t-2} - \phi_3(1 - \mathbb{B})y_{t-3} \\ &= (1 - \mathbb{B})(y_t - \phi_1 y_{t-1} - \phi_2 y_{t-2} - \phi_3 y_{t-3} \\ &= (1 - \mathbb{B})(1 - \phi_1\mathbb{B} - \phi_2\mathbb{B}^2 - \phi_3\mathbb{B}^3)y_t \end{aligned}$$
>
> And so we have a model $\theta_3(\mathbb{B})(1 - \mathbb{B})^1 y_t = \epsilon_t$ which is consistent with the p, d, and q previously stated.

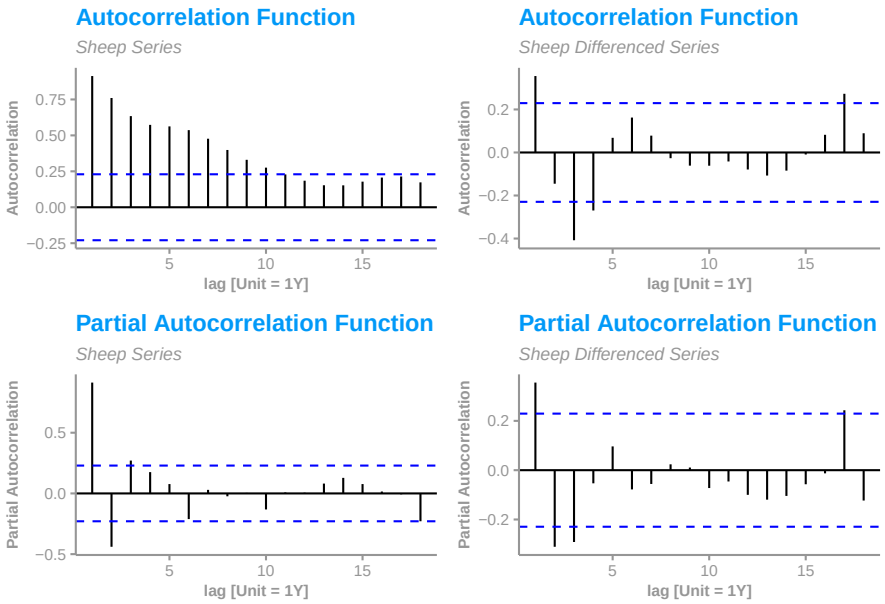
1.3 Is this model appropriate?

Examine the ACF and PACF of the differenced data. Evaluate whether this model is appropriate.

```
# Getting a tsibble with index from the original ts
sheep.ts <- data.frame(
  date = index(fma::sheep),
  sheep = melt(fma::sheep)$value
  )

sheep.ts <- sheep.ts %>%
  mutate(time_index = year(make_datetime(date))) %>%
  as_tsibble(index = time_index)

# ACF plot - Original Series
sheep.acf <- ACF(
sheep.ts,
sheep
) %>% autoplot() +
  labs(
      title = "Autocorrelation Function",
      subtitle = "Sheep Series",
      x = "lag [Unit = 1Y]",
      y = "Autocorrelation"
    ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 12,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 9,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 8,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 8),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  )

# PACF plot - Original Series
sheep.pacf <- PACF(
sheep.ts,
sheep
) %>% autoplot() +
  labs(
      title = "Partial Autocorrelation Function",
      subtitle = "Sheep Series",
```

```
      x = "lag [Unit = 1Y]",
      y = "Partial Autocorrelation"
    ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 12,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 9,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 8,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 8),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  )

# ACF plot - Differenced Series
diff.acf <- ACF(
sheep.ts,
diff(sheep)
) %>% autoplot() +
  labs(
      title = "Autocorrelation Function",
      subtitle = "Sheep Differenced Series",
      x = "lag [Unit = 1Y]",
      y = "Autocorrelation"
    ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 12,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 9,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 8,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 8),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  )
```

```
# PACF plot - Differenced Series
diff.pacf <- PACF(
sheep.ts,
diff(sheep)
) %>% autoplot() +
  labs(
      title = "Partial Autocorrelation Function",
      subtitle = "Sheep Differenced Series",
      x = "lag [Unit = 1Y]",
      y = "Partial Autocorrelation"
    ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 12,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 9,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 8,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 8),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  )

# 2x2 grid to compare original VS differenced
(sheep.acf | diff.acf) /(sheep.pacf | diff.pacf)
```



By looking at the original series' graphs (left) and comparing them to the differenced series (right) we can see that differencing the series removed the trend as expected and stabilized the PACF values, so it seems like there's indeed an unit root and this might be a good step to take, although we still don't have a stationary process quite yet, perhaps due to an AR process as well as a seasonal component which would explain the fluctuations.

1.4 Forecasts, by hand!

The last five values of the series are given below:

Year	1935	1936	1937	1938	1939
Millions of sheep	1648	1665	1627	1791	1797

The estimated parameters are:

- $\phi_1 = 0.42$;
- $\phi_2 = -0.20$; and,
- $\phi_3 = -0.30$.

Without using the forecast function, calculate forecasts for the next three years (1940–1942).

```
## Using R as a calculator here
y_35 <- 1648
y_36 <- 1665
y_37 <- 1627
y_38 <- 1791
y_39 <- 1797

y_40 <- y_39 +
  0.42 * (y_39-y_38) -
  0.20 * (y_38-y_37) -
  0.30 * (y_37-y_36)

y_41 <- y_40 +
  0.42 * (y_40-y_39) -
  0.20 * (y_39-y_38) -
  0.30 * (y_38-y_37)

y_42 <- y_41 +
  0.42 * (y_41-y_40) -
  0.20 * (y_40-y_39) -
  0.30 * (y_39-y_38)
```

We have:

$$\hat{y}_{1940} = y_{1939} + \phi_1(y_{1939} - y_{1938}) + \phi_2(y_{1938} - y_{1937}) + \phi_3(y_{1937} - y_{1936}) = 1778.12$$

$$\hat{y}_{1941} = y_{1940} + \phi_1(y_{1940} - y_{1939}) + \phi_2(y_{1939} - y_{1938}) + \phi_3(y_{1938} - y_{1937}) = 1719.79$$

$$\hat{y}_{1942} = y_{1941} + \phi_1(y_{1941} - y_{1940}) + \phi_2(y_{1940} - y_{1939}) + \phi_3(y_{1939} - y_{1938}) = 1697.27$$

1.5 Interpret roots

Find the roots of your model's characteristic equation. Is this process stationary?.

```
roots <- polyroot(c(1, -0.42, 0.20, 0.30))
roots
```

```
## [1]  0.714019+1.039948i -2.094704+0.000000i  0.714019-1.039948i
```

> Since we have $d = 1$ we would say that the process is non-stationary to begin with, but applying this first difference, we have that the characteristic equation is $(1 - \phi_1\mathbb{B} - \phi_2\mathbb{B}^2 - \phi_3\mathbb{B}^3)$.

The roots will be:

- $\mathbb{B}_1$ = 0.71+1.04i
- $\mathbb{B}_2$ = -2.09+0i
- $\mathbb{B}_3$ = 0.71-1.04i

And their absolute value would be:

- $|\mathbb{B}_1|$ = 1.261
- $|\mathbb{B}_2|$ = 2.095
- $|\mathbb{B}_3|$ = 1.261

Since all of them, in absolute value, exceed the unity we would say the process is stationary.

Chapter 2

(10 points) Seasonal ARIMA model

Download the series of E-Commerce Retail Sales as a Percent of Total Sales here.

(Feel free to explore the `fredr` package and API if interested.)

Our goal is to Build a Seasonal ARIMA model, following all appropriate steps for a univariate time series model.

Separate the data set into training and test data. The training data is used to estimate model parameters, and it is for 10/1999-12/2020. The test data is used to evaluate its accuracy, and it is for 01/2021-01/22.

```
# Downloading dataset from FREDR data
ecom <- fredr(
  series_id = "ECOMPCTNSA"
  )


ecom.ts <- ecom %>%
  mutate(time_index = yearquarter(date)) %>%
  as_tsibble(index = time_index)

ecom.train <- ecom.ts %>%
  filter_index(~ "2020-12-31")

ecom.test <- ecom.ts %>%
  filter_index("2021-01-01" ~.)
```

```
# Graph functions to easily repeat plots

# TS line plot
gglineplot <- function(ts = NULL,       # Time Series to be used
                       var = NULL,      # Variable to be plotted
                       graph_title = "",    # Title for the graph
                       graph_subtitle = "", # Subtitle for the graph
                       x_label = "",        # Custom x axis label
                       y_label = ""         # Custom x axis label
                       ) {

  graph <- ggplot(
    ts,
    var
    ) +
    geom_line(
      size = 0.8,
      color = "cornflowerblue"
      ) +
    labs(
      title = graph_title,
      subtitle = graph_subtitle,
      x = x_label,
      y = y_label
      ) +
      theme_classic() +
      theme(
      plot.title = element_text(color = "#0099F8",
                                size = 20,
                                face = "bold"),
      plot.subtitle = element_text(color="#969696",
                                   size = 14,
                                   face = "italic"),
      axis.title = element_text(color = "#969696",
                                size = 12,
                                face = "bold"),
      axis.text = element_text(color = "#969696", size = 11),
      axis.line = element_line(color = "#969696"),
      axis.ticks = element_line(color = "#969696"),
      ) + scale_y_continuous(labels = scales::percent)

  return (graph)
}

# TS ACF plot
```

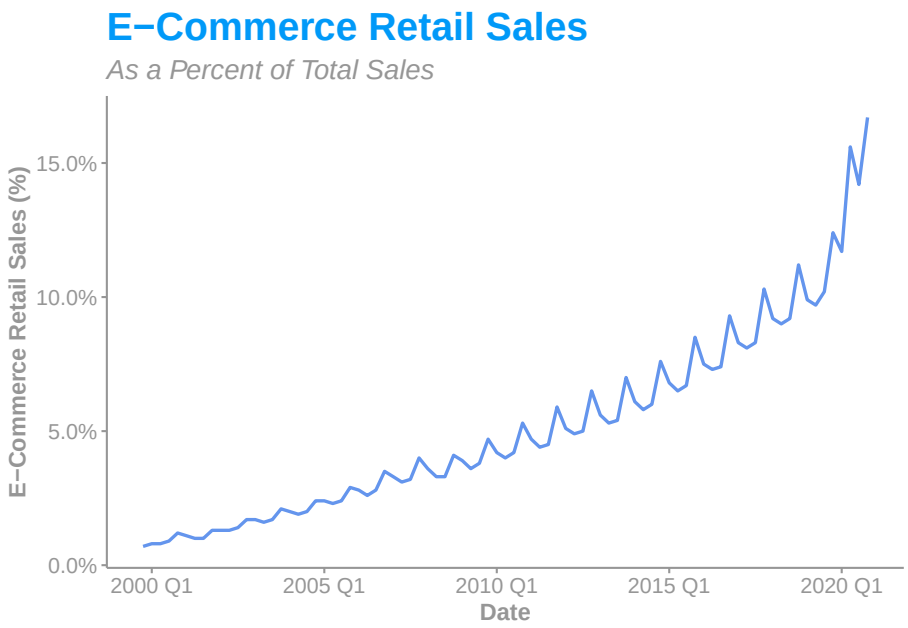
```
ggacfplot <- function(ts = NULL,           # Time Series to be used
                      var = NULL,          # Variable to be plotted
                      graph_subtitle = "", # Subtitle for the graph
                      x_label = "",        # Custom x axis label
                      y_label = ""         # Custom x axis label
                      ) {

  acfgraph <- ACF(
    ts,
    var
    ) %>% autoplot() +
      labs(
          title = "Autocorrelation Function",
          subtitle = graph_subtitle,
          x = x_label,
          y = "Autocorrelation"
        ) +
      theme_classic() +
      theme(
      plot.title = element_text(color = "#0099F8",
                                size = 40,
                                face = "bold"),
      plot.subtitle = element_text(color="#969696",
                                   size = 32,
                                   face = "italic"),
      axis.title = element_text(color = "#969696",
                                size = 28,
                                face = "bold"),
      axis.text = element_text(color = "#969696", size = 28),
      axis.line = element_line(color = "#969696"),
      axis.ticks = element_line(color = "#969696")
      ) + scale_x_continuous(breaks = c(0,4,8,12,16,20))

  return (acfgraph)
}

#TS PACF plot
ggpacfplot <- function(ts = NULL,           # Time Series to be used
                      var = NULL,          # Variable to be plotted
                      graph_subtitle = "", # Subtitle for the graph
                      x_label = "",        # Custom x axis label
                      y_label = ""         # Custom x axis label
                      ) {

  pacfgraph <- PACF(
```

```
    ts,
    var
    ) %>% autoplot() +
      labs(
          title = "Partial Autocorrelation Function",
          subtitle = graph_subtitle,
          x = x_label,
          y = "Partial Autocorrelation"
        ) +
      theme_classic() +
      theme(
      plot.title = element_text(color = "#0099F8",
                                size = 40,
                                face = "bold"),
      plot.subtitle = element_text(color="#969696",
                                   size = 32,
                                   face = "italic"),
      axis.title = element_text(color = "#969696",
                                size = 28,
                                face = "bold"),
      axis.text = element_text(color = "#969696", size = 28),
      axis.line = element_line(color = "#969696"),
      axis.ticks = element_line(color = "#969696")
      ) + scale_x_continuous(breaks = c(0,4,8,12,16,20))

  return (pacfgraph)
}
```

2.1 Time series plot

Plot training data set of Retail Sales. What do you notice? Is there any transformation necessary?

```
gglineplot(
  ts = ecom.train,
  var = aes(x = time_index, y = value / 100),
  graph_title = "E-Commerce Retail Sales",
  graph_subtitle = "As a Percent of Total Sales",
  x_label = "Date",
  y_label = "E-Commerce Retail Sales (%)"
)
```



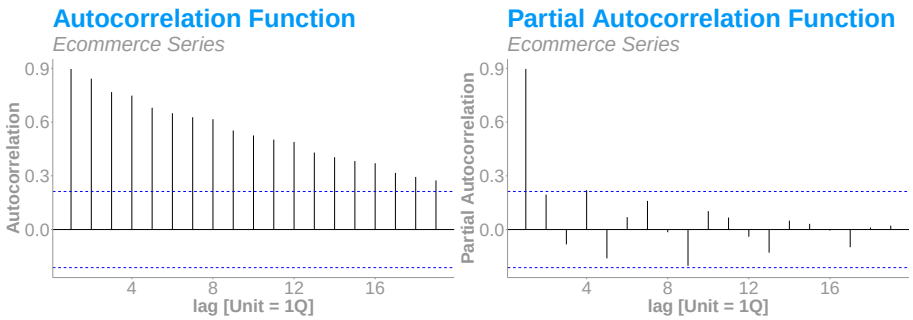
Data has a pronounced positive trend and also the variance seems to increase as time passes. We'll likely need to difference the data in order to make it stationary.

2.2 Check for Stationary

Use ACF/PACF and a unit root test to check if Retail Sales is stationary. If data is not stationary, difference the data, and apply the test again until it becomes stationary? How many differences are needed to make data stationary?

```
# ACF - Original series
ecom.acf <- ggacfplot(
  ts = ecom.train,
  var = ecom.train$value,
  graph_subtitle = "Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

# PACF - Original series
ecom.pacf <- ggpacfplot(
  ts = ecom.train,
  var = ecom.train$value,
  graph_subtitle = "Ecommerce Series",
```

```
  x_label = "lag [Unit = 1Q]",
)

ecom.acf | ecom.pacf
```




```
# Unit Root Tests
PP.test(ecom.train$value)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  ecom.train$value
## Dickey-Fuller = -0.11819, Truncation lag parameter = 3, p-value = 0.99
```

```
adf.test(ecom.train$value, alternative = "stationary", k = 1)
```

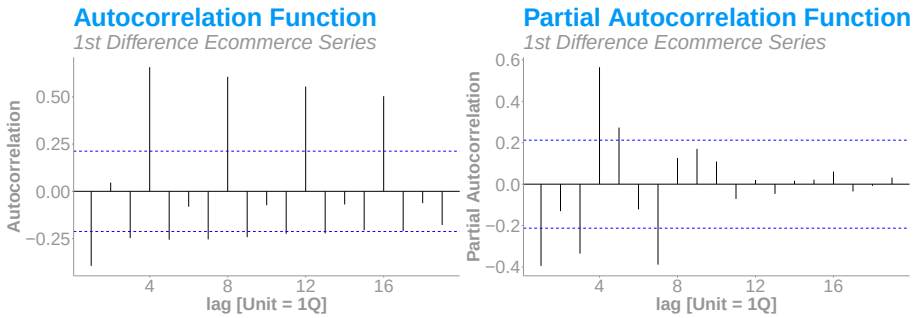
```
##
##  Augmented Dickey-Fuller Test
##
## data:  ecom.train$value
## Dickey-Fuller = 0.31652, Lag order = 1, p-value = 0.99
## alternative hypothesis: stationary
```

```
adf.test(ecom.train$value, alternative = "stationary", k = 2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  ecom.train$value
## Dickey-Fuller = 1.7123, Lag order = 2, p-value = 0.99
## alternative hypothesis: stationary
```

When applying the PP test and the ADF test (up to 2 lags) to the original series we fail to reject the null hypothesis meaning that we might still have a unit root and the series is non stationary. Which is also consistent when looking at the ACF and PACF plot (which might also suggest at least an AR(1) since the PACF for lag 1 is really high).

```
# ACF - 1st Difference Series
ecom.diff.acf <- ggacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 1),
  graph_subtitle = "1st Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

# PACF - 1st Difference Series
ecom.diff.pacf <- ggpacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 1),
  graph_subtitle = "1st Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

ecom.diff.acf | ecom.diff.pacf
```




```
# Unit Root Tests
PP.test(diff(ecom.train$value, differences = 1))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff(ecom.train$value, differences = 1)
## Dickey-Fuller = -17.19, Truncation lag parameter = 3, p-value = 0.01
```

```
adf.test(diff(ecom.train$value, differences = 1), alternative = "stationary", k = 1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 1)
## Dickey-Fuller = -9.0593, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 1), alternative = "stationary", k = 2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 1)
## Dickey-Fuller = -11.813, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 1), alternative = "stationary", k = 3)
```

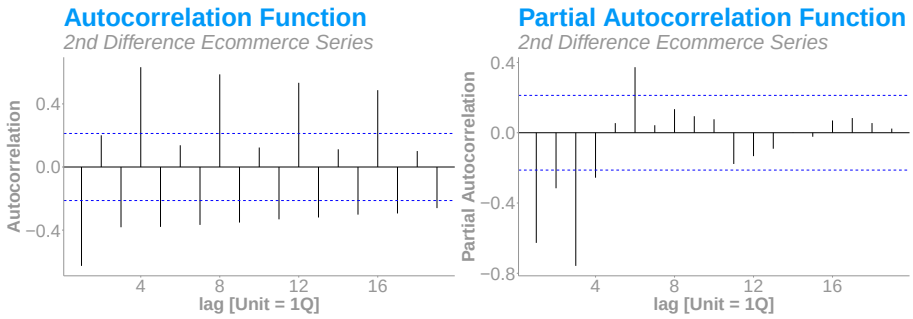
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 1)
## Dickey-Fuller = -2.051, Lag order = 3, p-value = 0.555
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 1), alternative = "stationary", k = 4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 1)
## Dickey-Fuller = -1.674, Lag order = 4, p-value = 0.7096
## alternative hypothesis: stationary
```

When applying the PP test and the ADF test (up to 2 lags) and to the 1st differenced series we reject the null hypothesis (H_0 : not stationary) meaning that this series might be stationary, although when trying for lags 3 and 4 for the ADF test we fail to reject the null and by looking at the ACF and PACF graphs it doesn't seem stationary yet

```
# ACF - 2nd Difference Series
ecom.diff2.acf <- ggacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 2),
  graph_subtitle = "2nd Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

# PACF - 2nd Difference Series
ecom.diff2.pacf <- ggpacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 2),
  graph_subtitle = "2nd Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

ecom.diff2.acf | ecom.diff2.pacf
```




```
# Unit Root Tests
PP.test(diff(ecom.train$value, differences = 2))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff(ecom.train$value, differences = 2)
## Dickey-Fuller = -60.161, Truncation lag parameter = 3, p-value = 0.01
```

```
adf.test(diff(ecom.train$value, differences = 2), alternative = "stationary", k = 1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 2)
```

```
## Dickey-Fuller = -10.569, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 2), alternative = "stationary", k = 2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 2)
## Dickey-Fuller = -28.083, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 2), alternative = "stationary", k = 3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 2)
## Dickey-Fuller = -9.1315, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

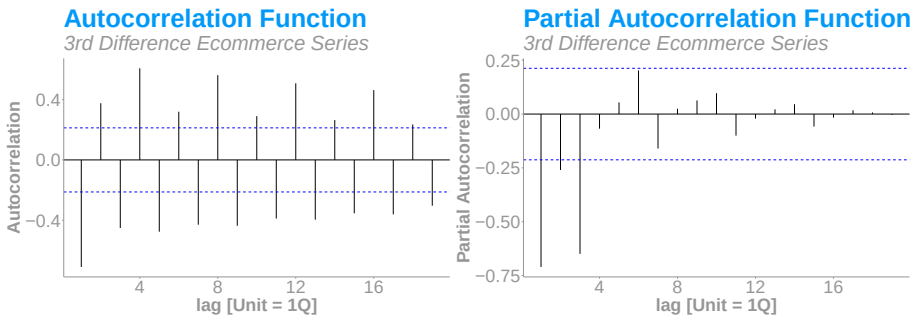
```
adf.test(diff(ecom.train$value, differences = 2), alternative = "stationary", k = 4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 2)
## Dickey-Fuller = -7.2975, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

When applying the PP test and ADF test for the 2nd differenced series, we reject the null hypothesis (H_0 : not stationary) meaning that this series might be stationary as well even for higher lags (up to 4), but the the ACF and PACF graphs still have many significant values which says again that perhaps we're dealing with an AR model and a seasonal component as well.

```
# ACF - 3rd Difference Series
ecom.diff3.acf <- ggacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 3),
  graph_subtitle = "3rd Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
```

```
)

# PACF - 3rd Difference Series
ecom.diff3.pacf <- ggpacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 3),
  graph_subtitle = "3rd Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

ecom.diff3.acf | ecom.diff3.pacf
```




```
# Unit Root Tests
PP.test(diff(ecom.train$value, differences = 3))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -93.631, Truncation lag parameter = 3, p-value = 0.01
```

```
adf.test(diff(ecom.train$value, differences = 3), alternative = "stationary", k = 1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -10.247, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 3), alternative = "stationary", k = 2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -29.154, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 3), alternative = "stationary", k = 3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -13.515, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 3), alternative = "stationary", k = 4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -12.227, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 3), alternative = "stationary", k = 5)
```

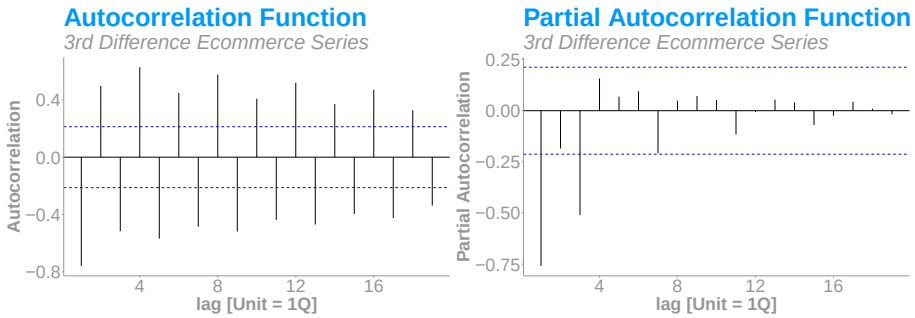
```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -6.9286, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 3), alternative = "stationary", k = 6)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 3)
## Dickey-Fuller = -4.1016, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

When applying the PP test and ADF test (up to 6 lags) we fail to reject the null meaning that this series might also be stationary, but again the the ACF and PACF graphs have many significant values and it seems like we're really not getting significantly better results . We'll still try another iteration (since lag 4 would mean 1 year back).

```
# ACF - 4th Difference Series
ecom.diff4.acf <- ggacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 4),
  graph_subtitle = "3rd Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

# PACF - 4th Difference Series
ecom.diff4.pacf <- ggpacfplot(
  ts = ecom.train,
  var = diff(ecom.train$value, differences = 4),
  graph_subtitle = "3rd Difference Ecommerce Series",
  x_label = "lag [Unit = 1Q]",
)

ecom.diff4.acf | ecom.diff4.pacf
```




```
# Unit Root Tests
PP.test(diff(ecom.train$value, differences = 4))
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -124.72, Truncation lag parameter = 3, p-value = 0.01
```

```
adf.test(diff(ecom.train$value, differences = 4), alternative = "stationary", k = 1)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -10.157, Lag order = 1, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 4), alternative = "stationary", k = 2)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -26.016, Lag order = 2, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 4), alternative = "stationary", k = 3)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -18.149, Lag order = 3, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 4), alternative = "stationary", k = 4)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -12.983, Lag order = 4, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 4), alternative = "stationary", k = 5)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -10.775, Lag order = 5, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(ecom.train$value, differences = 4), alternative = "stationary", k = 6)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(ecom.train$value, differences = 4)
## Dickey-Fuller = -9.0808, Lag order = 6, p-value = 0.01
## alternative hypothesis: stationary
```

> We're getting similar results, so we'll stop there. We would say that we can get a "stationary" series even from the first difference, but we would still need to combine it with an AR model and perhaps a seasonal component.

2.3 Model identification and estimation

Use ACF/PACF to identify an appropriate SARIMA model. Estimate both select model and model chosen by ARIMA()

```
# Human (myself) Selected Model
mod.select <- ecom.train %>%
  model(
    ARIMA(value ~ 1 + pdq(3,1,0) + PDQ(0,0,0),
          ic="bic",
          stepwise=F,
          greedy=F)
    )

mod.select %>% report()
```

```
## Series: value
## Model: ARIMA(3,1,0) w/ drift
##
## Coefficients:
##           ar1      ar2      ar3  constant
##       -0.5977  -0.3433  -0.5475    0.4287
## s.e.   0.0978   0.1071   0.1082    0.0771
##
## sigma^2 estimated as 0.5084:  log likelihood=-89.46
## AIC=188.92   AICc=189.69   BIC=201.07
```

```
# ARIMA automatic selection
mod.arima <- ecom.train %>%
  model(
    ARIMA(value ~ 1 + pdq(0:10,0:10,0:10) + PDQ(0:10,0:10,0:10),
          ic="bic",
          stepwise=F,
          greedy=F)
    )

mod.arima %>% report()
```

```
## Series: value
## Model: ARIMA(1,0,0)(0,1,0)[4] w/ drift
##
## Coefficients:
##          ar1  constant
##       0.8826    0.1114
## s.e.  0.0682    0.0554
##
## sigma^2 estimated as 0.2715:  log likelihood=-61.87
## AIC=129.74   AICc=130.05   BIC=136.92
```

The model I've picked is an ARIMA(3,1,0) due to the plots and analysis made before. We get a stationary series for the first difference but the PACF still has significant values for several lags, although I stopped at 3 lags for the AR process since adding one more would just mean going one year back.

Although when letting the function pick a model automatically it selects an ARIMA(1,0,0)(0,1,0)[4] which says that indeed we had an AR component but the difference is better applied in the seasonal component (specifically for lag 4 which is exactly one year before). The criterions confirm that indeed this seems like a better fitted model than the one I've picked.

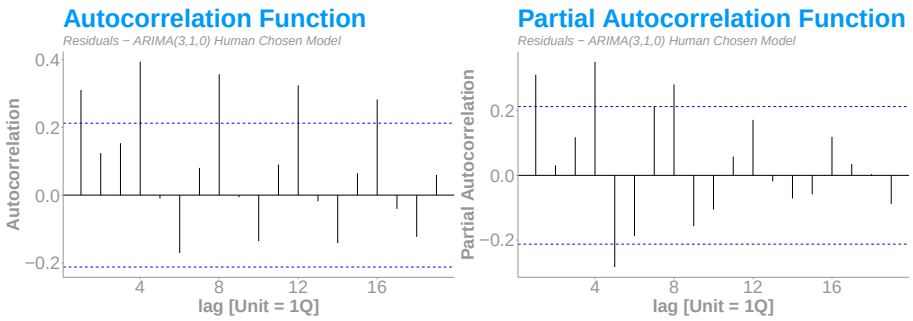
2.4 Model diagnostic

Do residual diagnostic checking of both models. Are the residuals white noise? Use the Ljung-box test to check if the residuals are white noise.

```
# Residual Analysis for Human Selected Model
resid.select <- mod.select %>%
  augment() %>%
```

```
  dplyr::select(.resid) %>%
  as.ts()

select.acf <- mod.select %>%
  augment() %>%
  ACF(.resid) %>%
  autoplot()+ labs(
    x = "lag [Unit = 1Q]",
    y = "Autocorrelation",
    title = "Autocorrelation Function",
    subtitle = "Residuals - ARIMA(3,1,0) Human Chosen Model"
  ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 40,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 20,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 28,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 28),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  ) + scale_x_continuous(breaks = c(0,4,8,12,16,20))

select.pacf <- mod.select %>%
  augment() %>%
  PACF(.resid) %>%
  autoplot()+ labs(
    x = "lag [Unit = 1Q]",
    y = "Partial Autocorrelation",
    title = "Partial Autocorrelation Function",
    subtitle = "Residuals - ARIMA(3,1,0) Human Chosen Model"
  ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 40,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 20,
                               face = "italic"),
```

```
  axis.title = element_text(color = "#969696",
                            size = 28,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 28),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  ) + scale_x_continuous(breaks = c(0,4,8,12,16,20))

select.acf | select.pacf
```




```
Box.test(resid.select, lag=1, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.select
## X-squared = 8.5045, df = 1, p-value = 0.003543
```

```
Box.test(resid.select, lag=2, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.select
## X-squared = 9.8828, df = 2, p-value = 0.007145
```

```
Box.test(resid.select, lag=3, type="Ljung-Box")
```

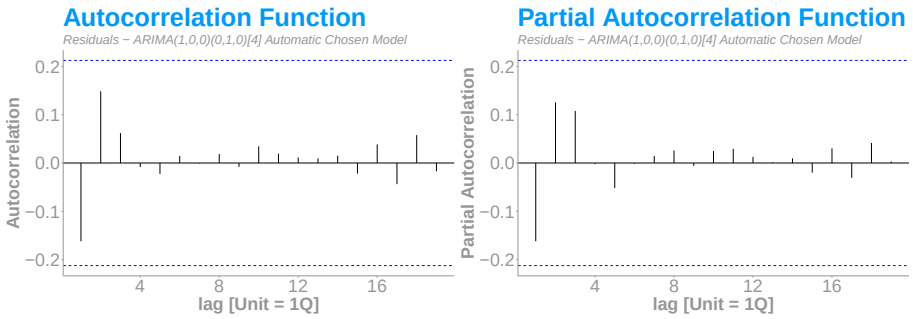
```
##
##  Box-Ljung test
##
## data:  resid.select
## X-squared = 12.005, df = 3, p-value = 0.007365
```

```
Box.test(resid.select, lag=4, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.select
## X-squared = 26.172, df = 4, p-value = 2.922e-05
```

```
# Residual Analysis for ARIMA Selected Model
resid.arima <- mod.arima %>%
  augment() %>%
  dplyr::select(.resid) %>%
  as.ts()

arima.acf <- mod.arima %>%
  augment() %>%
  ACF(.resid) %>%
  autoplot()+ labs(
    x = "lag [Unit = 1Q]",
    y = "Autocorrelation",
    title = "Autocorrelation Function",
    subtitle = "Residuals - ARIMA(1,0,0)(0,1,0)[4] Automatic Chosen Model"
  ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 40,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 20,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 28,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 28),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  ) + scale_x_continuous(breaks = c(0,4,8,12,16,20))

arima.pacf <- mod.arima %>%
  augment() %>%
  PACF(.resid) %>%
  autoplot()+ labs(
    x = "lag [Unit = 1Q]",
    y = "Partial Autocorrelation",
```

```
    title = "Partial Autocorrelation Function",
    subtitle = "Residuals - ARIMA(1,0,0)(0,1,0)[4] Automatic Chosen Model"
  ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 40,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 20,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 28,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 28),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  ) + scale_x_continuous(breaks = c(0,4,8,12,16,20))

arima.acf | arima.pacf
```




```
Box.test(resid.arima, lag=1, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.arima
## X-squared = 2.3155, df = 1, p-value = 0.1281
```

```
Box.test(resid.arima, lag=2, type="Ljung-Box")
```

```
##
```

```
##  Box-Ljung test
##
## data:  resid.arima
## X-squared = 4.2809, df = 2, p-value = 0.1176
```

```
Box.test(resid.arima, lag=3, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.arima
## X-squared = 4.6274, df = 3, p-value = 0.2012
```

```
Box.test(resid.arima, lag=4, type="Ljung-Box")
```

```
##
##  Box-Ljung test
##
## data:  resid.arima
## X-squared = 4.6339, df = 4, p-value = 0.327
```

For the model I've chosen - ARIMA(3,1,0) - the ACF and PACF don't show white noise at all and when performing the Ljung Box Tests we reject the null (H_0 : data are independently distributed) up to 4 lags.

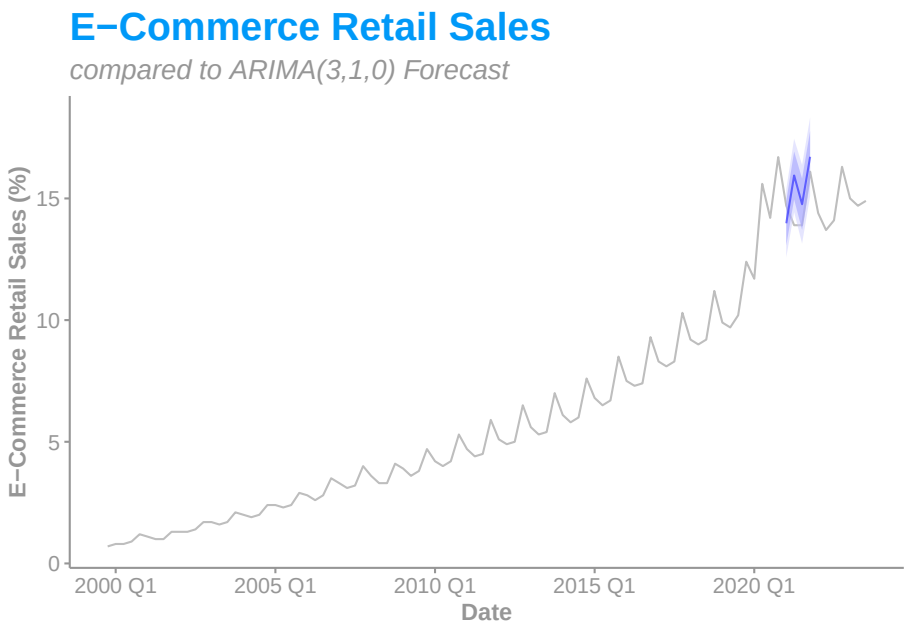
On the other hand when looking at the automatically chosen model - ARIMA(1,0,0)(0,1,0)[4] even when it doesn't look completely as white noise there are no significant lags in either the ACF nor PACF, and when performing the Ljung Box Tests we fail to reject the null for every one of the first 4 lags, meaning these residuals might resemble a white noise process.

2.5 Forecasting

Use the both models to forecast the next 12 months and evaluate the forecast accuracy of these models.

```
# Forecasting 4 quarters ahead (12 Months)
per = 4

# Forecast Plot - Human Selected Model
```

```
sel.fore <- forecast(mod.select, h = per)
autoplot(
  ecom.ts,
  .vars = value,
  colour = "gray"
  ) +
  autolayer(sel.fore, colour="blue", alpha = 0.5) +
  labs(
      title = "E-Commerce Retail Sales",
      subtitle = "compared to ARIMA(3,1,0) Forecast",
      x = "Date",
      y = "E-Commerce Retail Sales (%)"
    ) +
    theme_classic() +
    theme(
    plot.title = element_text(color = "#0099F8",
                              size = 20,
                              face = "bold"),
    plot.subtitle = element_text(color="#969696",
                                 size = 14,
                                 face = "italic"),
    axis.title = element_text(color = "#969696",
                              size = 12,
                              face = "bold"),
    axis.text = element_text(color = "#969696", size = 11),
    axis.line = element_line(color = "#969696"),
    axis.ticks = element_line(color = "#969696"),
    legend.position = "none"
    )
```

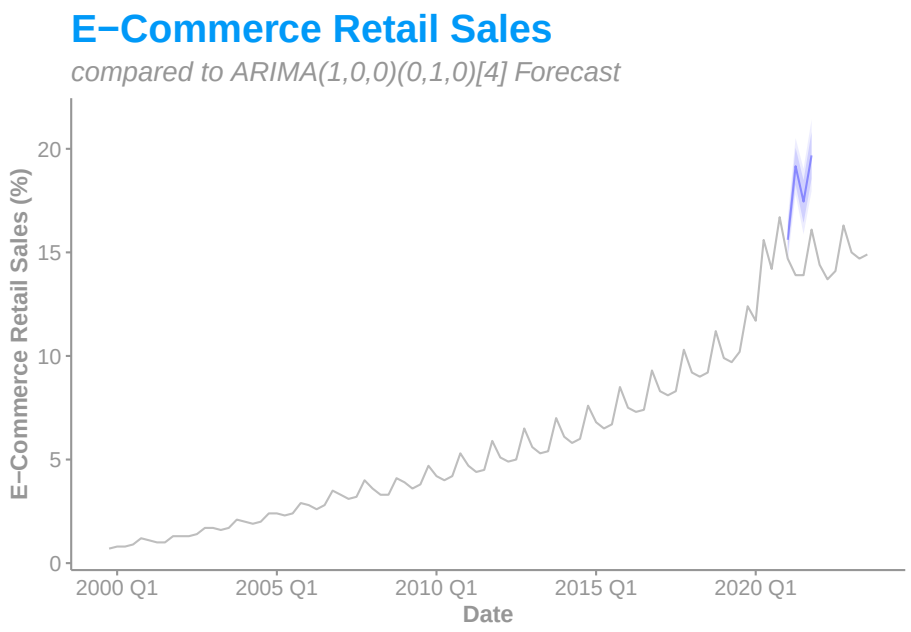



```
# Accuracy Metrics - Human Selection
sel.eval <- accuracy(sel.fore, ecom.test)
sel.eval
```

```
## # A tibble: 1 x 10
##   .model                                  .type     ME  RMSE   MAE   MPE  MAPE  MASE RMSSE   A
##   <chr>                                   <chr>  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <d
## 1 "ARIMA(value ~ 1 + pdq(3, 1, 0) + PDQ(~ Test  -0.700  1.21  1.06 -4.95  7.40   NaN   NaN -0.
```

```
# Forecast Plot - ARIMA Selected Model
ari.fore <- forecast(mod.arima, h = per)
autoplot(
  ecom.ts,
  .vars = value,
  colour = "gray"
  ) +
  autolayer(ari.fore, colour="blue", alpha = 0.35) +
  labs(
      title = "E-Commerce Retail Sales",
      subtitle = "compared to ARIMA(1,0,0)(0,1,0)[4] Forecast",
      x = "Date",
      y = "E-Commerce Retail Sales (%)"
  ) +
  theme_classic() +
```

```
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 20,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 14,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 12,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 11),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696"),
  legend.position = "none"
  )
```




```
# Accuracy Metrics - ARIMA selection
ari.eval <- accuracy(ari.fore, ecom.test)
ari.eval
```

```
## # A tibble: 1 x 10
##   .model                                    .type    ME  RMSE   MAE   MPE  MAPE  MASE
##   <chr>                                     <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 "ARIMA(value ~ 1 + pdq(0:10, 0:10, 0:10~ Test  -3.32  3.67  3.32 -23.0  23.0   Na
```

Metric	Human Chosen Model	Automatically Chosen Model
RMSE	1.21	3.67
MAE	1.06	3.32
MAPE	7.4	22.95

Even when the automatically chosen model (ARIMA(1,0,0)(0,1,0)[4]) performed better in the training data, by comparing the forecasts and the evaluation metrics, the model I've picked (ARIMA(3,1,0)) proved a better performance in test data. Although this might seem unfair because we can notice in the graph that the forecasted year didn't really follow the pattern we were observing before, so it wasn't really a good predictor, it was actually just luck.

Chapter 3

(10 points) Time Series Linear Model and Cointegration

Daily electricity demand and temperature (in degrees Celsius) is recorded in `./data/temperature_demand.csv`. Please work through the following questions to build a time series linear model against this data.

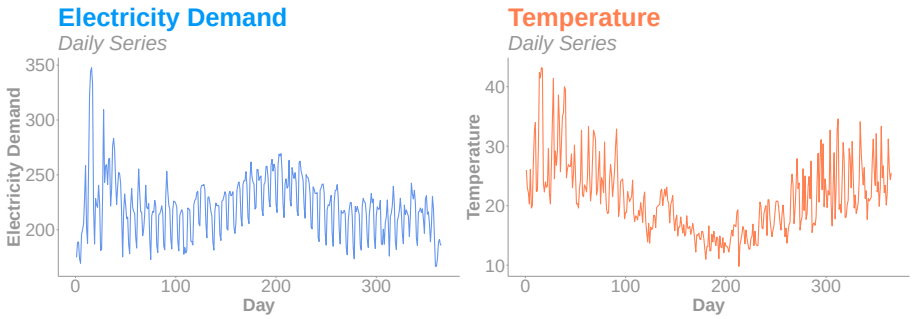
```
# Loading Data
temperature <- read_csv('./data/temperature_demand.csv') %>%
  rename(
    c(
     'index' = '...1',
     'demand' = 'Demand',
     'work_day'= 'WorkDay',
     'temperature' = 'Temperature'
    )
  )
# glimpse(temperature)

# Creating tsibble
temp.ts <- temperature %>%
  mutate(time_index = ...1) %>%
  as_tsibble(index = time_index)
```

3.1 Plot electricity

Plot electricity demand and temperature as time series. Is there any correlation between these to variables? If yes, Do you think is it a spurious correlation?

```
# Electricity Demand TS Plot
elec.plot <- ggplot(
    temp.ts,
    aes(
      x = time_index,
      y = Demand
      )
    ) +
    geom_line(
      size = 0.8,
      color = "cornflowerblue"
      ) +
    labs(
      title = "Electricity Demand",
      subtitle = "Daily Series",
      x = "Day",
      y = "Electricity Demand"
      ) +
      theme_classic() +
      theme(
      plot.title = element_text(color = "#0099F8",
                                size = 40,
                                face = "bold"),
      plot.subtitle = element_text(color="#969696",
                                size = 32,
                                face = "italic"),
      axis.title = element_text(color = "#969696",
                                size = 28,
                                face = "bold"),
      axis.text = element_text(color = "#969696", size = 28),
      axis.line = element_line(color = "#969696"),
      axis.ticks = element_line(color = "#969696"),
      )

# Temperature TS Plot
temp.plot <- ggplot(
    temp.ts,
    aes(
      x = time_index,
      y = Temperature
```

```
      )
    ) +
    geom_line(
      size = 0.8,
      color = "coral"
      ) +
    labs(
      title = "Temperature",
      subtitle = "Daily Series",
      x = "Day",
      y = "Temperature"
      ) +
      theme_classic() +
      theme(
      plot.title = element_text(color = "coral",
                                size = 40,
                                face = "bold"),
      plot.subtitle = element_text(color="#969696",
                                size = 32,
                                face = "italic"),
      axis.title = element_text(color = "#969696",
                                size = 28,
                                face = "bold"),
      axis.text = element_text(color = "#969696", size = 28),
      axis.line = element_line(color = "#969696"),
      axis.ticks = element_line(color = "#969696"),
      )

elec.plot | temp.plot
```



There seems to be certain correlation. This could arise indeed a spurious regression since these variables may not be necessarily dependant on each other (for instance they both could be related to time of the day instead of being directly related). Although, since

our measurement unit is "day" it could be that indeed we have at least a cointegration relationship or even a dependant variable on each other.

3.2 Cointegration test

Use the Engle-Granger test to check for cointegration. What do you conclude?

In order to perform this cointegration test we will carry an ADF test first to determine whether both series are integrated in the same order.

```
adf.test(temp.ts$Demand, alternative = "stationary", k = 30)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temp.ts$Demand
## Dickey-Fuller = -2.0998, Lag order = 30, p-value = 0.5348
## alternative hypothesis: stationary
```

```
adf.test(temp.ts$Temperature, alternative = "stationary", k = 30)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  temp.ts$Temperature
## Dickey-Fuller = -0.89843, Lag order = 30, p-value = 0.9523
## alternative hypothesis: stationary
```

```
adf.test(diff(temp.ts$Demand), alternative = "stationary", k = 30)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(temp.ts$Demand)
## Dickey-Fuller = -4.0524, Lag order = 30, p-value = 0.01
## alternative hypothesis: stationary
```

```
adf.test(diff(temp.ts$Temperature), alternative = "stationary", k = 30)
```

```
##
##  Augmented Dickey-Fuller Test
##
## data:  diff(temp.ts$Temperature)
## Dickey-Fuller = -5.0041, Lag order = 30, p-value = 0.01
## alternative hypothesis: stationary
```

For the original series both p-values are much higher than 5% so we fail to reject the null hypothesis of a non stationary series (meaning we could have an unit root), but for their differences p-values are less than 5% so we reject the null hypothesis and we can proceed with the test since both are I(1).

```
coint_reg <- lm(data = temp.ts, formula = Demand ~ Temperature)
coef(summary(coint_reg))
```

```
##                Estimate Std. Error  t value      Pr(>|t|)
## (Intercept) 212.3855624  5.0079889 42.40935 1.057811e-142
## Temperature   0.4182455  0.2263087  1.84812  6.539825e-02
```

```
coint_res <- ts(coint_reg$res)
pp.test(coint_res)
```

```
## Warning in pp.test(coint_res): p-value smaller than printed p-value
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  coint_res
## Dickey-Fuller Z(alpha) = -96.401, Truncation lag parameter = 5, p-value = 0.01
## alternative hypothesis: stationary
```

To perform the Engle-Granger we first run a regression of Electricity Demand on Temperature and then carry out a unit root test on the residuals. Based on the Phillips-Perron Unit Root Test, the p-value is less than 5%, and we reject the null hypothesis of unit root for the residuals, so the test provides evidence that the Electricity Demand and Temperature are cointegrated since the residuals are stationary.

3.3 Fit Model

Based on cointegration test, fit a regression model for demand with temperature as an explanatory variable(or their first difference).

Since they're cointegrated we can perform a regression of Demand on Temperature. An also becauseboth of them are I(1) it makes sense to use their first difference as follows:

```
# Creating a differences df
diff.df <- data.frame(
  Elec.diff = diff(temp.ts$Demand),
  Temp.diff = diff(temp.ts$Temperature)
  )

# Regression on differenced series
elec.model <- lm(data = diff.df, formula = Elec.diff ~ Temp.diff)
summary(elec.model)
```

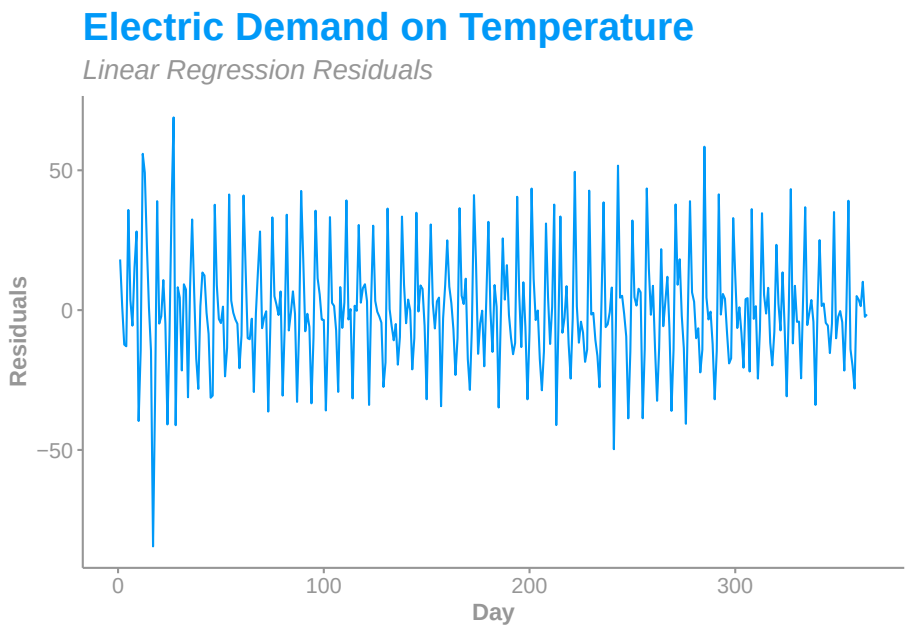
```
##
## Call:
## lm(formula = Elec.diff ~ Temp.diff, data = diff.df)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -84.488 -10.465  -1.310   8.001  68.896
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  0.03356    1.08112   0.031    0.975
## Temp.diff    1.48143    0.24585   6.026 4.15e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 20.63 on 362 degrees of freedom
## Multiple R-squared:  0.09116,    Adjusted R-squared:  0.08865
## F-statistic: 36.31 on 1 and 362 DF,  p-value: 4.146e-09
```

Which gives us a positive and significant coefficient for temperature difference as an explanatory variable for the electricity demand difference.

3.4 Residuals Plot

Produce a residual plot of the estimated model in previous part. Is the model adequate? Describe any outliers or influential observations, and discuss how the model could be improved.

```
# Plotting Residuals
elec.resid.plot <- elec.model %>%
  augment() %>%
  dplyr::select(.resid) %>%
  as.ts() %>%
  autoplot(colour = "#0099F8") + labs(
    x = "Day",
    y = "Residuals",
    title = "Electric Demand on Temperature",
    subtitle = "Linear Regression Residuals"
  ) +
  theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 20,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 14,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 12,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 11),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696")
  )
elec.resid.plot
```



This model seems adequate since residuals appear stationary although we still have certain outliers at the beginning of the year. In order to improve the model we could add an autoregressive component on top of this regression result, or add other variables to the model that could explain these outliers.

3.5 Forecasting model

Use a model to forecast the electricity demand (with **prediction** intervals) that you would expect for the next day if the maximum temperature was 15°. Compare this with the forecast if the with maximum temperature was 35°. Do you believe these forecasts? Why or why not?

```
# Last values
last.temp <- tail(temp.ts, 1)$Temperature
last.elec <- tail(temp.ts, 1)$Demand

# New assigned temperature differences
new.temp = data.frame(Temp.diff = c(15 - last.temp, 35 - last.temp))

# Predictions
diff.pred <- predict(elec.model, newdata = new.temp)
elec.pred <- diff.pred + last.elec
```

```
# Confidence intervals
conf.diff <- confint(object = elec.model, level = 0.95)
conf.15 <- last.elec +
  c((15 - last.temp)*conf.diff[2], (15 - last.temp)*conf.diff[4])
conf.35 <- last.elec +
  c((35 - last.temp)*conf.diff[2], (35 - last.temp)*conf.diff[4])
```

Since we're using first differences we will need the last known values for Temperature (25.5°) and for demand (186.37). So, for the case where maximum temperature is 15°, the difference would be -10.5° and the predicted difference in electricity demand is -15.52, so the electricity demand point estimate would be 170.85, with a 95% confidence interval of (165.74, 175.89).

And for the case where maximum temperature is 35°, the difference would be 9.5° and the predicted difference in electricity demand predicted is 14.11, so the electricity demand point estimate would be 200.48, with a 95% confidence interval of (195.85, 205.04). Because all the process we've followed, and the results we've obtained of cointegration and residual analysis these seem like good forecasts.

Chapter 4

(12 points) Vector autoregression

Annual values for real mortgage credit (RMC), real consumer credit (RCC) and real disposable personal income (RDPI) for the period 1946-2006 are recorded in `./data/mortgage_credit.csv`.

All of the observations are measured in billions of dollars, after adjustment by the Consumer Price Index (CPI).

Our goal is to develop a VAR model for these data for the period 1946-2003, and then forecast the last three years, 2004-2006.

```
# Creating tsibble
credit <- read_csv('./data/mortgage_credit.csv')
credit.ts <- credit %>%
 as_tsibble(index = Year)

# Splitting train/test datasets
credit.train <- credit.ts %>%
  filter_index(~ "2003")
credit.test <- credit.ts %>%
  filter_index("2004" ~.)
```

4.1 Time series plot

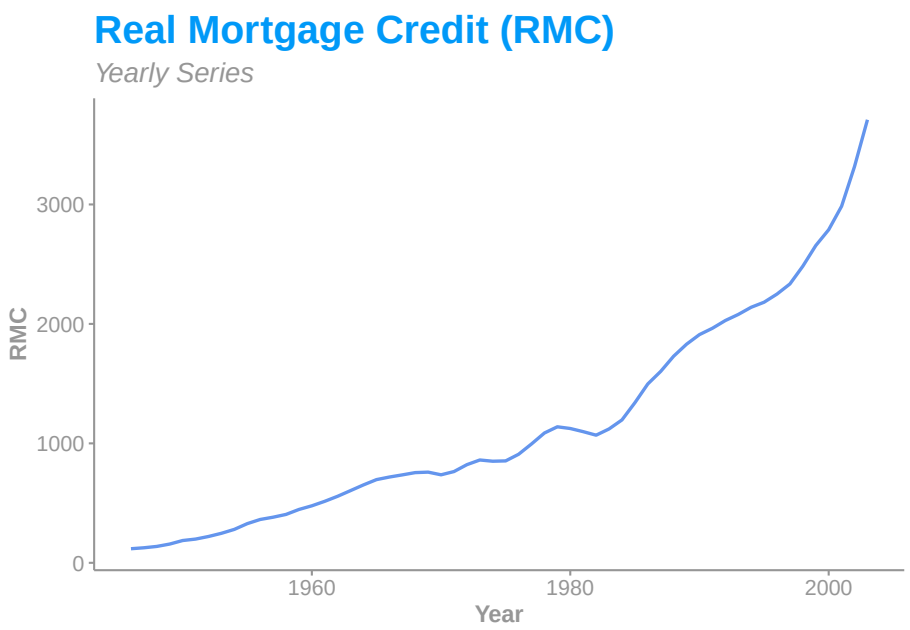
Plot the time-series of real mortgage credit (RMC), real consumer credit (RCC) and real disposable personal income (RDPI)? Do they look stationary?

```
My_Theme = theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 20,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                               size = 14,
                               face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 12,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 11),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696"),
  legend.position = "none"
  )

Double_Theme = theme_classic() +
  theme(
  plot.title = element_text(color = "#0099F8",
                            size = 40,
                            face = "bold"),
  plot.subtitle = element_text(color="#969696",
                            size = 32,
                            face = "italic"),
  axis.title = element_text(color = "#969696",
                            size = 28,
                            face = "bold"),
  axis.text = element_text(color = "#969696", size = 28),
  axis.line = element_line(color = "#969696"),
  axis.ticks = element_line(color = "#969696"),
  )
```

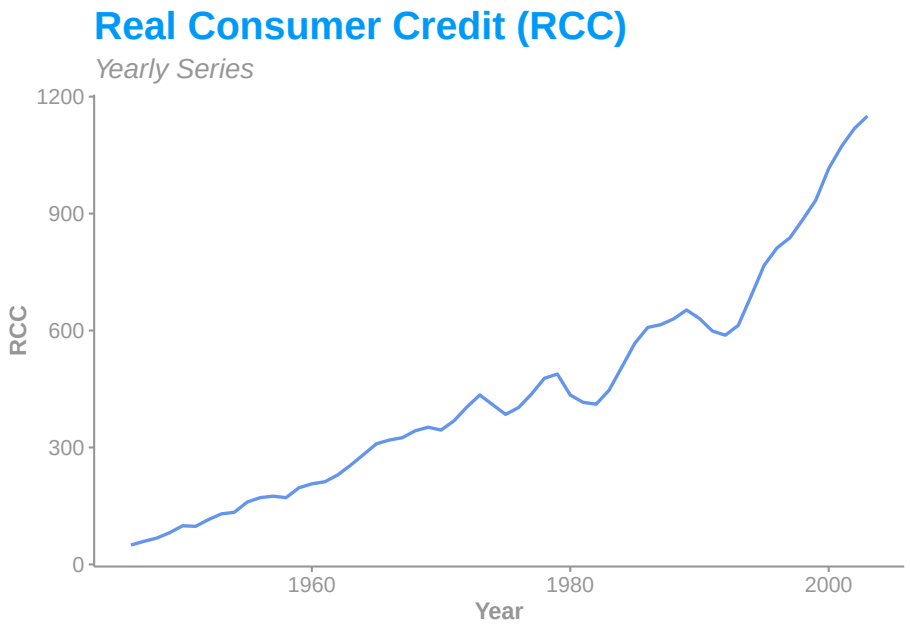
```
rmc.plot <- ggplot(
  credit.train,
  aes(
    x = Year,
    y = RMC
    )
  ) +
  geom_line(
    size = 0.8,
    color = "cornflowerblue"
    ) +
  labs(
```

```
    title = "Real Mortgage Credit (RMC)",
    subtitle = "Yearly Series",
    x = "Year",
    y = "RMC"
    ) +
  My_Theme

rcc.plot <- ggplot(
  credit.train,
  aes(
    x = Year,
    y = RCC
    )
  ) +
  geom_line(
    size = 0.8,
    color = "cornflowerblue"
    ) +
  labs(
    title = "Real Consumer Credit (RCC)",
    subtitle = "Yearly Series",
    x = "Year",
    y = "RCC"
    ) +
  My_Theme

rdp.plot <- ggplot(
  credit.train,
  aes(
    x = Year,
    y = RDPI
    )
  ) +
  geom_line(
    size = 0.8,
    color = "cornflowerblue"
    ) +
  labs(
    title = "Real Disposable Personal Income (RDPI)",
    subtitle = "Yearly Series",
    x = "Year",
    y = "RDPI"
    ) +
  My_Theme
```

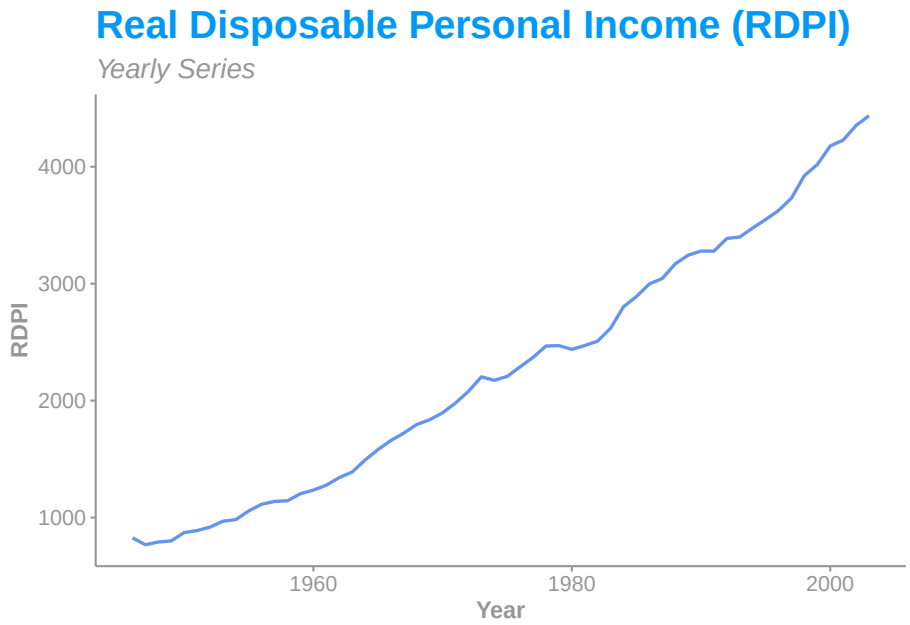
```
rmc.plot
```




```
rcc.plot
```



```
rdp.plot
```



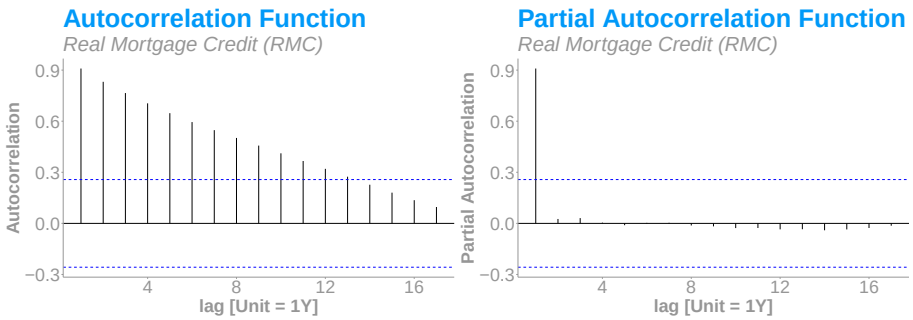
By just taking a quick look at the 3 plots we can notice that there seems to be a clear trend in them so they don't look stationary and it is likely that we'll need to difference these series to obtain stationary processes.

4.2 Check for the unit root

Plot ACF/PACF and Perform the unit root test on these variables and report the results. Do you reject the null of unit root for them? Is the first differencing necessary?

```
# ACF for RMC Original Series
rmc.acf <- ACF(
      credit.train,
      RMC
    ) %>% autoplot() +
    labs(
        title = "Autocorrelation Function",
        subtitle = "Real Mortgage Credit (RMC)",
        x = "lag [Unit = 1Y]",
```

```
        y = "Autocorrelation"
    ) + Double_Theme

# PACF for RMC Original Series
rmc.pacf <- PACF(
      credit.train,
      RMC
    ) %>% autoplot() +
    labs(
        title = "Partial Autocorrelation Function",
        subtitle = "Real Mortgage Credit (RMC)",
        x = "lag [Unit = 1Y]",
        y = "Partial Autocorrelation"
    ) + Double_Theme

rmc.acf | rmc.pacf
```

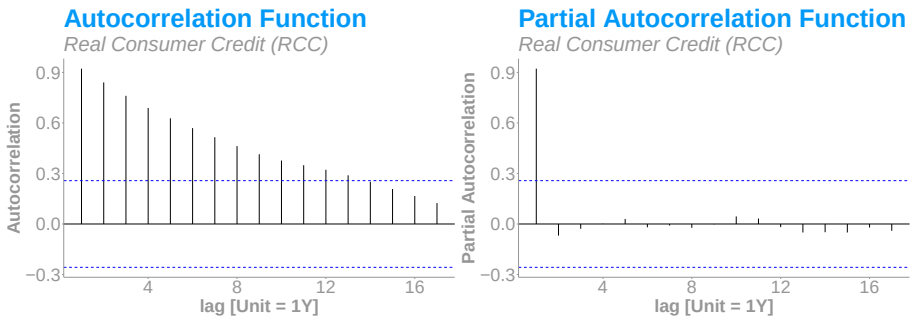



```
# Unit Root Test RMC
PP.test(credit.train$RMC)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  credit.train$RMC
## Dickey-Fuller = 3.3488, Truncation lag parameter = 3, p-value = 0.99
```

```
# ACF for RCC Original Series
rcc.acf <- ACF(
      credit.train,
      RCC
    ) %>% autoplot() +
    labs(
        title = "Autocorrelation Function",
```

```
        subtitle = "Real Consumer Credit (RCC)",
        x = "lag [Unit = 1Y]",
        y = "Autocorrelation"
    ) + Double_Theme

# PACF for RCC Original Series
rcc.pacf <- PACF(
      credit.train,
      RCC
    ) %>% autoplot() +
    labs(
        title = "Partial Autocorrelation Function",
        subtitle = "Real Consumer Credit (RCC)",
        x = "lag [Unit = 1Y]",
        y = "Partial Autocorrelation"
    ) + Double_Theme

rcc.acf | rcc.pacf
```

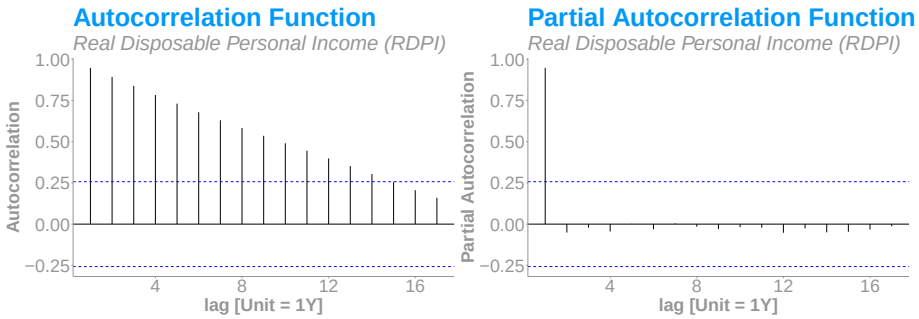



```
# Unit Root Test RCC
PP.test(credit.train$RCC)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  credit.train$RCC
## Dickey-Fuller = 0.13432, Truncation lag parameter = 3, p-value = 0.99
```

```
# ACF for RDPI Original Series
rdp.acf <- ACF(
      credit.train,
      RDPI
    ) %>% autoplot() +
```

```
    labs(
        title = "Autocorrelation Function",
        subtitle = "Real Disposable Personal Income (RDPI)",
        x = "lag [Unit = 1Y]",
        y = "Autocorrelation"
    ) + Double_Theme

# PACF for RDPI Original Series
rdp.pacf <- PACF(
      credit.train,
      RDPI
    ) %>% autoplot() +
    labs(
        title = "Partial Autocorrelation Function",
        subtitle = "Real Disposable Personal Income (RDPI)",
        x = "lag [Unit = 1Y]",
        y = "Partial Autocorrelation"
    ) + Double_Theme

rdp.acf | rdp.pacf
```




```
# Unit Root Test RDPI
PP.test(credit.train$RDPI)
```

```
##
##  Phillips-Perron Unit Root Test
##
## data:  credit.train$RDPI
## Dickey-Fuller = -1.7776, Truncation lag parameter = 3, p-value = 0.6645
```

For all 3 we fail to reject the null (H_0 : non stationary) and by looking at the ACFs we can notice that we need to remove the trend, so differencing is indeed necessary. By looking at the PACFs the

significant lag is always 1 so First-differencing (meaning lag 1) seems appropriate.

4.3 Determine VAR model

Based on the unit root results transform the variables and determine the lag length of the VAR using the information criteria.

```
# Creating a ts for differenced variables
diff.df <- credit.train %>%
  mutate(
   RMC.diff = difference(RMC),
   RCC.diff = difference(RCC),
   RDPI.diff = difference(RDPI)
   ) %>%
  filter(Year > "1946") %>%
  dplyr::select(RMC.diff,RCC.diff, RDPI.diff)

# Testing for Lag Length
VARselect(diff.df, lag.max = 4, type="none")
```

```
## $selection
## AIC(n)  HQ(n)  SC(n) FPE(n)
##      2      2      2      2
##
## $criteria
##                  1             2             3             4
## AIC(n)    10.18215 -3.501512e+01 -3.467725e+01 -3.444329e+01
## HQ(n)     10.41088 -3.455766e+01 -3.399105e+01 -3.352836e+01
## SC(n)     10.77695 -3.382551e+01 -3.289284e+01 -3.206407e+01
## FPE(n) 26457.55347  6.268489e-16  8.989438e-16  1.189039e-15
```

We'll be using the differenced series, and when evaluating the optimal number of lags, all criteria suggest two lags. So we'll be estimating a VAR(2) model.

4.4 Estimation

Estimate the selected VAR in previous part and comment on the results.

```r
# Estimating a VAR(2) model
var_diff <- vars::VAR(as.ts(diff.df), p = 2, type = "none")
summary(var_diff)
```

```
##
## VAR Estimation Results:
## ========================
## Endogenous variables: RMC.diff, RCC.diff, RDPI.diff
## Deterministic variables: none
## Sample size: 55
## Log Likelihood: -772.156
## Roots of the characteristic polynomial:
## 0.9667 0.9667 0.6519 0.6519 0.5756 0.3971
## Call:
## vars::VAR(y = as.ts(diff.df), p = 2, type = "none")
##
##
## Estimation results for equation RMC.diff:
## =========================================
## RMC.diff = RMC.diff.l1 + RCC.diff.l1 + RDPI.diff.l1 + RMC.diff.l2 + RCC.diff.l2 + RI
##
##              Estimate Std. Error t value Pr(>|t|)
## RMC.diff.l1   1.34498    0.16545   8.129 1.21e-10 ***
## RCC.diff.l1  -0.08113    0.30650  -0.265   0.7924
## RDPI.diff.l1  0.01613    0.12596   0.128   0.8986
## RMC.diff.l2  -0.39512    0.20062  -1.970   0.0546 .
## RCC.diff.l2   0.16934    0.29973   0.565   0.5747
## RDPI.diff.l2  0.04287    0.12440   0.345   0.7319
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 35.25 on 49 degrees of freedom
## Multiple R-Squared: 0.8888,  Adjusted R-squared: 0.8752
## F-statistic: 65.29 on 6 and 49 DF,  p-value: < 2.2e-16
##
##
## Estimation results for equation RCC.diff:
## =========================================
## RCC.diff = RMC.diff.l1 + RCC.diff.l1 + RDPI.diff.l1 + RMC.diff.l2 + RCC.diff.l2 + RI
##
##              Estimate Std. Error t value Pr(>|t|)
## RMC.diff.l1   0.15180    0.09641   1.575 0.121788
## RCC.diff.l1   0.69925    0.17860   3.915 0.000279 ***
## RDPI.diff.l1  0.05067    0.07340   0.690 0.493241
```

```
## RMC.diff.l2  -0.15295    0.11690  -1.308 0.196864
## RCC.diff.l2  -0.43142    0.17466  -2.470 0.017036 *
## RDPI.diff.l2  0.12331    0.07249   1.701 0.095277 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 20.54 on 49 degrees of freedom
## Multiple R-Squared: 0.6644,  Adjusted R-squared: 0.6233
## F-statistic: 16.17 on 6 and 49 DF,  p-value: 3.739e-10
##
##
## Estimation results for equation RDPI.diff:
## ==========================================
## RDPI.diff = RMC.diff.l1 + RCC.diff.l1 + RDPI.diff.l1 + RMC.diff.l2 + RCC.diff.l2 + RDPI.diff.l
##
##               Estimate Std. Error t value Pr(>|t|)
## RMC.diff.l1    0.4477     0.2269   1.973  0.05416 .
## RCC.diff.l1   -0.3083     0.4204  -0.733  0.46685
## RDPI.diff.l1   0.4343     0.1728   2.514  0.01529 *
## RMC.diff.l2   -0.5242     0.2752  -1.905  0.06269 .
## RCC.diff.l2   -0.1890     0.4111  -0.460  0.64774
## RDPI.diff.l2   0.6079     0.1706   3.563  0.00083 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## Residual standard error: 48.35 on 49 degrees of freedom
## Multiple R-Squared: 0.6795,  Adjusted R-squared: 0.6403
## F-statistic: 17.31 on 6 and 49 DF,  p-value: 1.264e-10
##
##
##
## Covariance matrix of residuals:
##           RMC.diff RCC.diff RDPI.diff
## RMC.diff    1237.2    368.0     741.1
## RCC.diff     368.0    415.1     668.1
## RDPI.diff    741.1    668.1    2197.9
##
## Correlation matrix of residuals:
##           RMC.diff RCC.diff RDPI.diff
## RMC.diff    1.0000   0.5136    0.4494
## RCC.diff    0.5136   1.0000    0.6995
## RDPI.diff   0.4494   0.6995    1.0000
```

As previously said we estimated a VAR(2) model, obtaining the

following results:

When estimating differenced RMC, none of the coefficients are statistically significant at even 10% but the ones corresponding to the same variable's lags. Although this is enough to have a high R^2 and explain about 88% of the variation in RMC differences.

For the second equation (estimating differenced RCC), once again its own lags have significant coefficients, and we also find that only RDPI's second lag has marginal significance at 10%. We have a moderately good R^2 and explain about 66% of the variation in RCC differences.

Lastly, when estimating differenced RDPI, we find that its own lags have very significant coefficients, but also RMCs both lags have marginal significance at 10%. We also have a moderately good R^2 and we're able to explain about 68% of the variation in RDPI differences.

We basically can conclude that RMC and RCC are mainly explained by their own lags, but RMCs past lags also have some explaining power to predict RDPI.

4.5 Model diagnostic

Do diagnostic checking of the VAR model.

```
# Granger-causality
vars::causality(var_diff,cause="RMC.diff")$Granger
```

```
##
##  Granger causality H0: RMC.diff do not Granger-cause RCC.diff RDPI.diff
##
## data:  VAR object var_diff
## F-Test = 1.1221, df1 = 4, df2 = 147, p-value = 0.3484
```

```
vars::causality(var_diff,cause="RCC.diff")$Granger
```

```
##
##  Granger causality H0: RCC.diff do not Granger-cause RMC.diff RDPI.diff
##
## data:  VAR object var_diff
## F-Test = 0.6962, df1 = 4, df2 = 147, p-value = 0.5957
```

```
vars::causality(var_diff,cause="RDPI.diff")$Granger
```

```
##
##  Granger causality H0: RDPI.diff do not Granger-cause RMC.diff RCC.diff
##
## data:  VAR object var_diff
## F-Test = 1.3629, df1 = 4, df2 = 147, p-value = 0.2496
```

```
# Check stability
eigen <- roots(var_diff)
eigen
```

```
## [1] 0.9666608 0.9666608 0.6518943 0.6518943 0.5755576 0.3971109
```

```
# No serial correlation
var_diff_test<- serial.test(var_diff, lags.pt = 10)
var_diff_test
```

```
##
##  Portmanteau Test (asymptotic)
##
## data:  Residuals of VAR object var_diff
## Chi-squared = 77.672, df = 72, p-value = 0.3029
```

To diagnose our model we first conduct a test for Granger-causality, and we find that for all 3 variables we fail to reject the null hypothesis H_0 : variable does not Granger-cause the other two analyzed variables, which seems consistent with the previous result.

Then we find the eigenvalues of the companion matrix. Since all of them are less than one we can conclude that this VAR(2) process is stable.

Finally we check for autocorrelation in residuals by computing the Portmanteau test for serial correlation where we fail to reject the null hypothesis H_0 : no autocorrelation.

4.6 Forecasting

forecast the last three years, 2004-2006.

```r
# Using the VAR model to predict 3 periods ahead
pred <- predict(var_diff, n.ahead = 3)
pred
```

```
## $RMC.diff
##          fcst    lower    upper        CI
## [1,] 407.4484 338.3579 476.5389  69.09049
## [2,] 401.2265 286.2089 516.2442 115.01764
## [3,] 386.7609 234.1237 539.3981 152.63724
##
## $RCC.diff
##          fcst      lower    upper       CI
## [1,] 30.59177  -9.667935 70.85147 40.25970
## [2,] 24.82891 -30.734279 80.39210 55.56319
## [3,] 17.21150 -44.303548 78.72656 61.51505
##
## $RDPI.diff
##          fcst       lower    upper        CI
## [1,] 95.13613   0.3648468 189.9074  94.77128
## [2,] 53.84724 -54.8683289 162.5628 108.71557
## [3,] 33.85573 -94.2871350 161.9986 128.14287
```

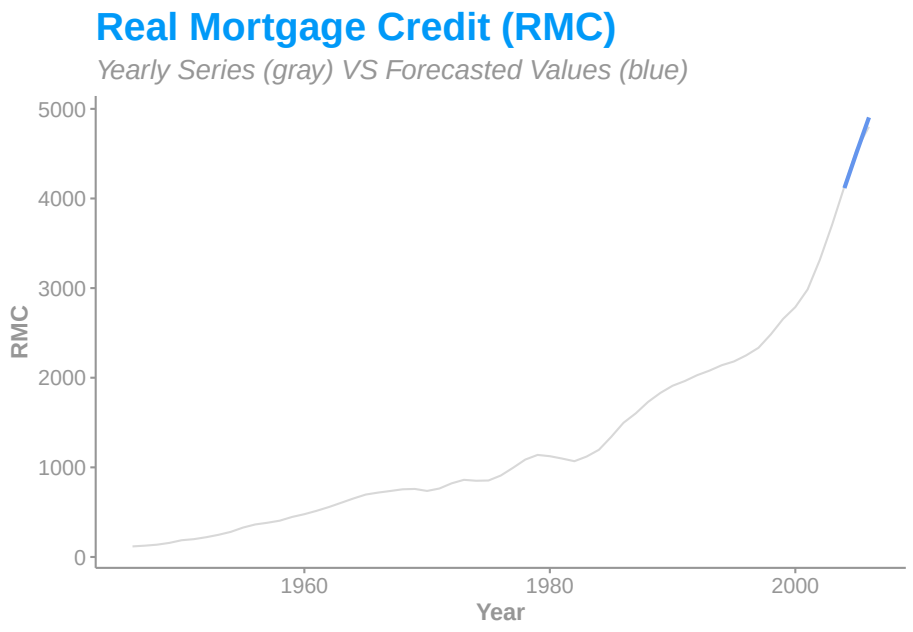
```r
# Last data point in the training dataset
last.data <- data.frame(tail(credit.train,1))

# Creating a Dataframe with the predicted differences
diff.pred <- data.frame(
  RMC = pred$fcst$RMC.diff[,1],
  RCC = pred$fcst$RCC.diff[,1],
  RDPI = pred$fcst$RDPI.diff[,1],
  Year = 2004:2006
  )
diff.pred <- rbind(last.data, diff.pred)

# Summing to the last value
# To find point estimators for the variables
# Instead of just the differences
final.pred <- data.frame(
  RMC = cumsum(diff.pred$RMC),
  RCC = cumsum(diff.pred$RCC),
  RDPI = cumsum(diff.pred$RDPI),
  Year = diff.pred$Year
)

# Keeping only 2004 - 2006
```

```
final.pred <- final.pred %>%
  filter(Year > "2003") %>%
  as_tsibble(index = Year)
final.pred
```

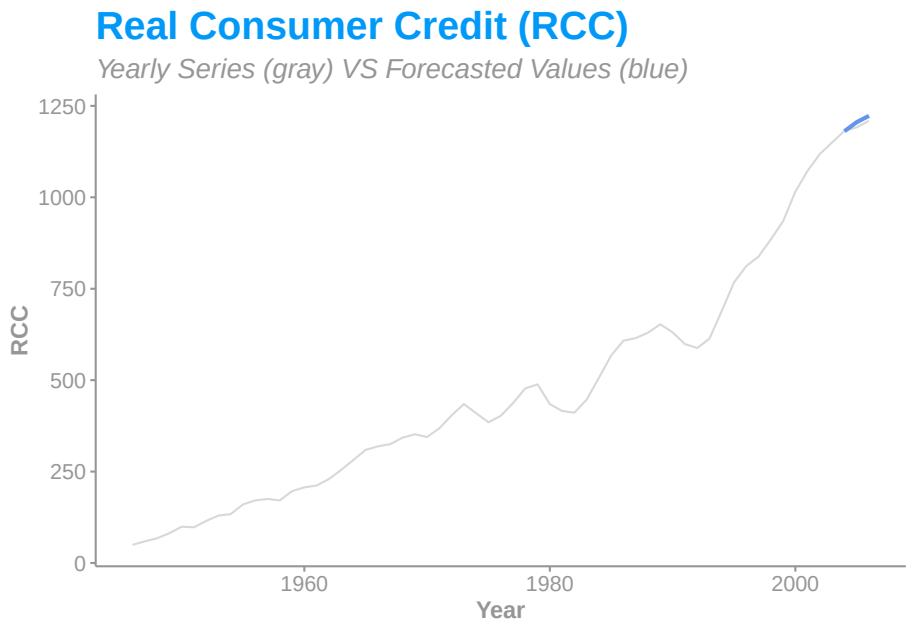
```
## # A tsibble: 3 x 4 [1Y]
##     RMC   RCC  RDPI  Year
##   <dbl> <dbl> <dbl> <dbl>
## 1 4116. 1181. 4531.  2004
## 2 4518. 1206. 4585.  2005
## 3 4904. 1223. 4619.  2006
```

```
# RMC Forecast Plot
rmc.for.plot <- ggplot(
  credit.ts,
  aes(
    x = Year,
    y = RMC
    )
  ) +
  geom_line(
    size = 0.5,
    color = "gray",
    alpha = 0.6
    ) +
  labs(
    title = "Real Mortgage Credit (RMC)",
    subtitle = "Yearly Series (gray) VS Forecasted Values (blue)",
    x = "Year",
    y = "RMC"
    ) + autolayer(
      final.pred,
      .vars = RMC,
      colour="cornflowerblue",
      size = 1) +
  My_Theme

# RCC Forecast Plot
rcc.for.plot <- ggplot(
  credit.ts,
  aes(
    x = Year,
    y = RCC
    )
  ) +
```

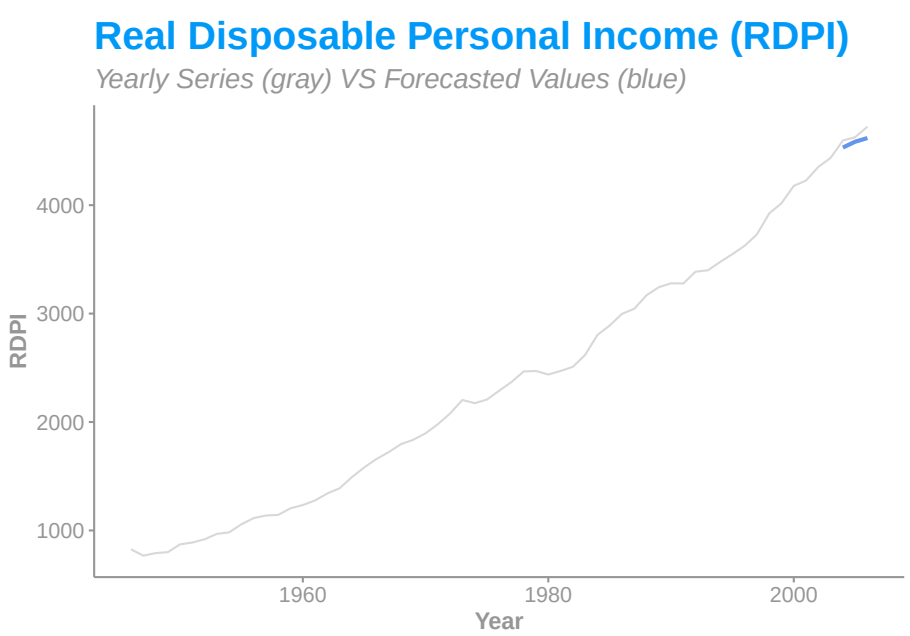
```
  geom_line(
    size = 0.5,
    color = "gray",
    alpha = 0.6
    ) +
  labs(
    title = "Real Consumer Credit (RCC)",
    subtitle = "Yearly Series (gray) VS Forecasted Values (blue)",
    x = "Year",
    y = "RCC"
    ) + autolayer(
      final.pred,
      .vars = RCC,
      colour="cornflowerblue",
      size = 1) +
  My_Theme

# RDPI Forecast Plot
rdp.for.plot <- ggplot(
  credit.ts,
  aes(
    x = Year,
    y = RDPI
    )
  ) +
  geom_line(
    size = 0.5,
    color = "gray",
    alpha = 0.6
    ) +
  labs(
    title = "Real Disposable Personal Income (RDPI)",
    subtitle = "Yearly Series (gray) VS Forecasted Values (blue)",
    x = "Year",
    y = "RDPI"
    ) + autolayer(
      final.pred,
      .vars = RDPI,
      colour="cornflowerblue",
      size = 1) +
  My_Theme

rmc.for.plot
```


Real Mortgage Credit (RMC)
Yearly Series (gray) VS Forecasted Values (blue)
5000
4000
3000
2000
1000
0
RMC
1960
1980
2000
Year

```
rcc.for.plot
```


Real Consumer Credit (RCC)
Yearly Series (gray) VS Forecasted Values (blue)
1250
1000
750
500
250
0
RCC
1960
1980
2000
Year

```
rdp.for.plot
```




```
rmc.ev <- accuracy(final.pred$RMC, credit.test$RMC)
rcc.ev <- accuracy(final.pred$RCC, credit.test$RCC)
rdp.ev <- accuracy(final.pred$RDPI, credit.test$RDPI)
```

In order to forecast the next 3 years, we proceed to predict the differences for those 3 years and then summing them to the last data point in the training dataset to compute the predicton of the actual values. The results are plotted for all 3 variables and we can see that even when we're mainly using previous lags of the same variable the predicted values make sense with what actually happened. We also have the following accuracy metrics, although we would need more models to compare the model's true efficiency:

Metric	RMC	RCC	RDPI
ME	-18.99	-8.95	70.41
RMSE	63.92	11.39	75.09
MAE	50.97	9.62	70.41
MPE	-0.36	-0.75	1.51
MAPE	1.09	0.8	1.51