Assignment 1

Applied Forecasting in Complex Systems 2021

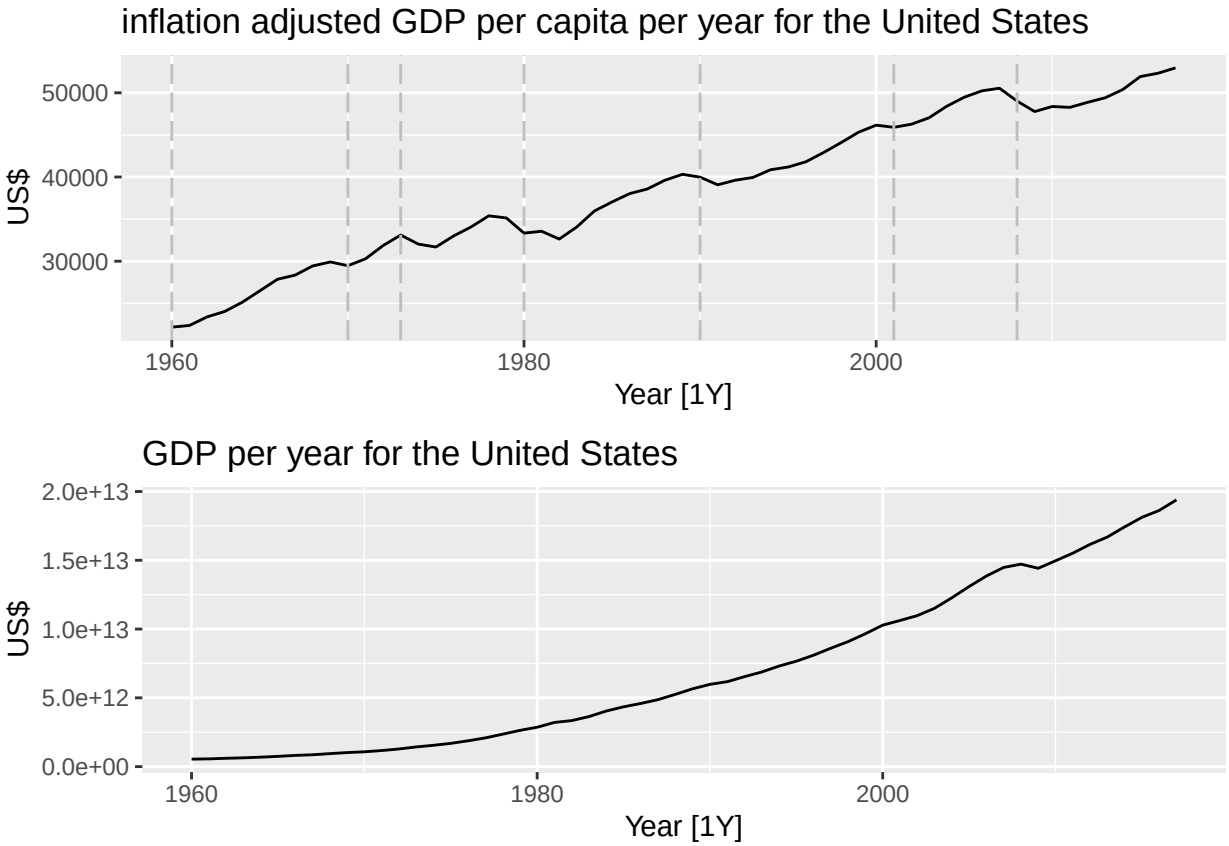
Emiel Steegh (14002558)

University of Amsterdam
November, 12, 2021

Note to the reader:
All of the code (and two additional plots) can be found in the appendix, starting on page 10

Exercise 1

1.1)

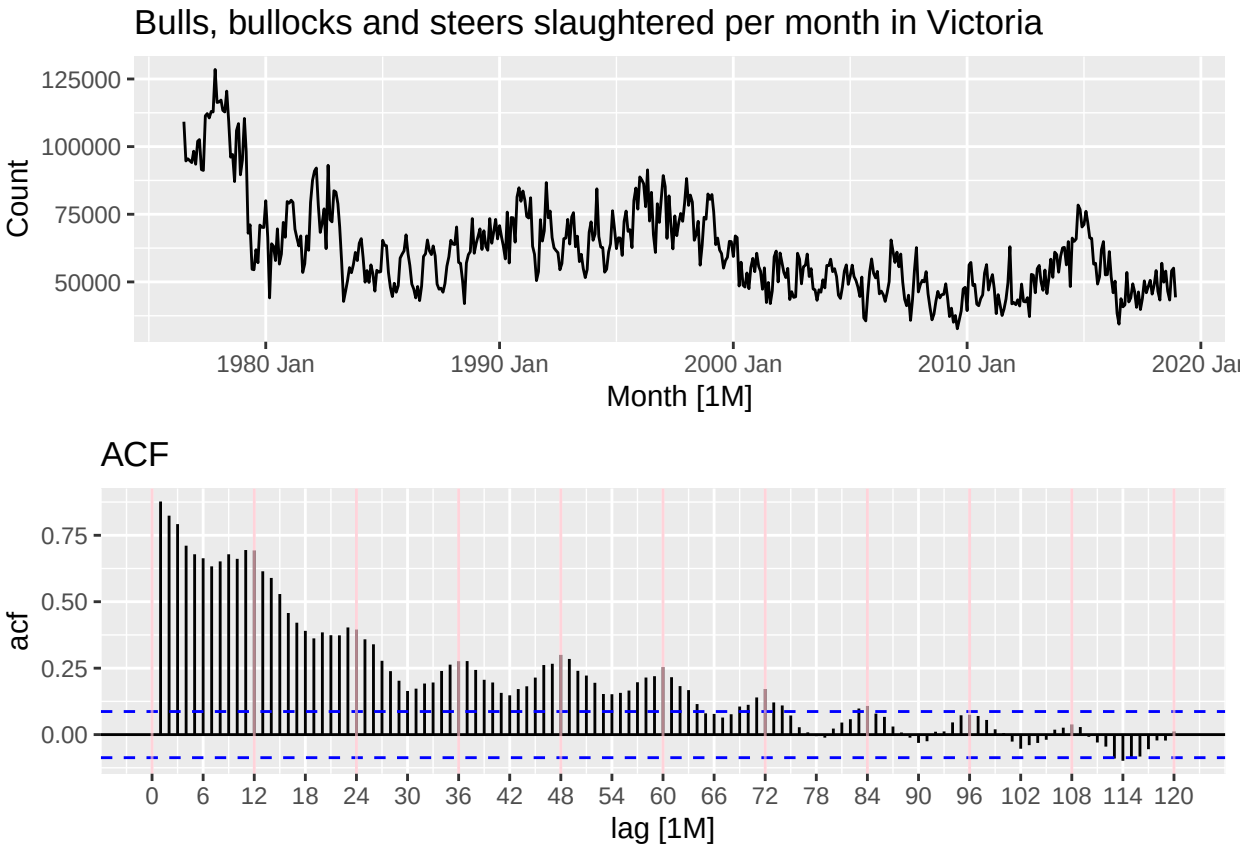


We transform the data from GDP per year to GDP *per capita* per year and then adjust for inflation (upper plot). By removing the effects population change and inflation have, the graph becomes much easier to interpret. The original data is visualised in the lower plot. There is no variance gain/loss or curvature in the data, so further transformations are not needed.

There is an upward trend and cyclicity appears when the inflation adjustment is applied.

In order to gain some insight, all of the starting years of major U.S. recessions are plotted as vertical dashed lines. The downward potions in the graph line up with the recession, which is not suprising as a recession is defined as a period of economic decline. What is perhaps suprising that some recessions "begin" *after* one or two years of decline.

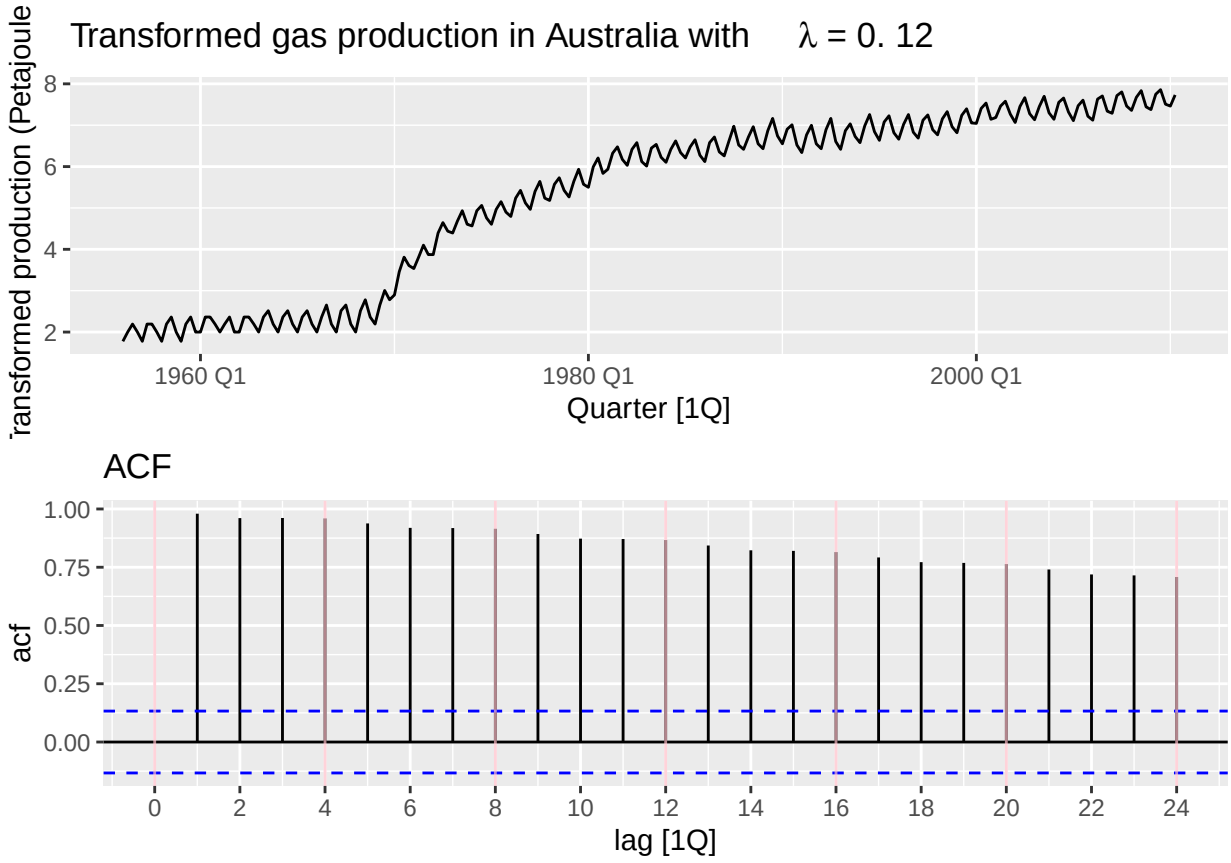
1.2)



Performing a guerrero test results in $\lambda = -0.072$ but doing a Box-Cox transformation with this lambda provides no noticeable improvement. Other attempted transformation methods provided no noticeable improvements either. So the data is left as is.

We observe a strong downward trend until about 1980, and what looks like a slight downward trend afterwards. There is definite cyclicity in the data as can be seen by the irregularly spaced peaks and troughs. Seasonality is difficult to see but there seems to be something going on. Making an auto correlation plot for 10 years worth of time shows seasonality with a one year period (as indicated by the pink lines).

To investigate the seasonality even further I generated a seasonal plot (Appendix 1.2) with a period of a year (the derived seasonal period from the ACF). In the seasonal plot we see seasonality across all data, not just the last 10 years. Most years start and end high with a dip in between although the pattern is noisy and a bit inconsistent, especially at the start of the sample. ## 1.3)



In the original data there was increasing variance so it has been transformed via the Box-Cox formula $\frac{y^{\lambda}-1}{\lambda}$ with $\lambda \approx 0.12$ as obtained from the Guerrero test (and trying some other values). This transformation (for a large part) stabilizes the variance in the data, making it easier to forecast.

We can spot an upward trend throughout the data, which can be divided into three approximate sections:
- 1956:1969 very slightly upward
- 1970:1980 strong upward trend that diminished with some curvature indicative of explosive growth
- 1981:2010 steady upward trend

We can also observe seasonality with a period of four quarters, indicating a yearly cycle. To inspect this assumption further I generated an auto correlation plot.
- A moderate downward trend of the correlation bars show an upward trend in the last six years of the data.
- The (very) slight drop after every multiple of four (indicated by the pink lines) reflect the observed yearly seasonality.
It must be noted that this seasonality is much more apparent without the Box-Cox transformation

In the seasonal plot (Appendix 1.3) with the observed period of 1 year we can see some clean but weak seasonality.

Exercise 2

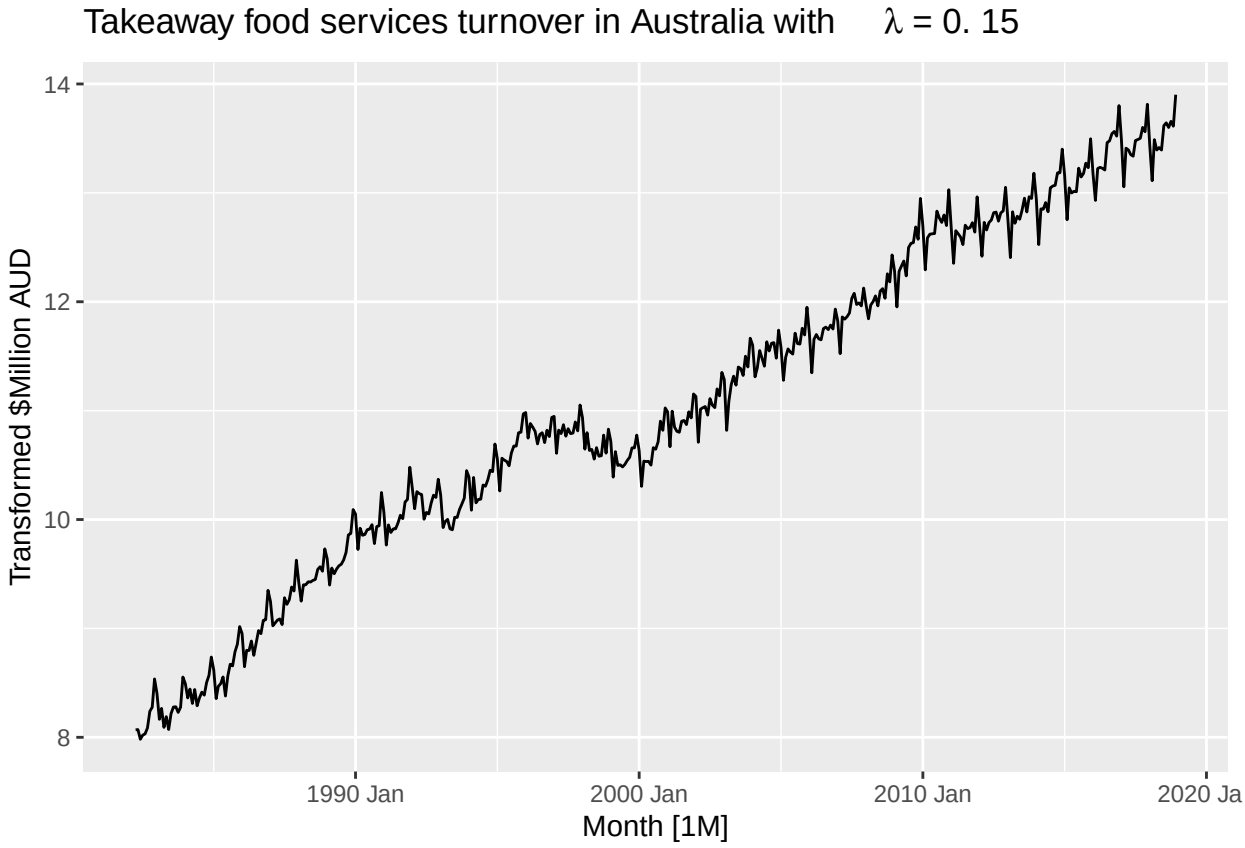
2.1)

Data preparation

I opted to sum the state data rather than taking the mean. When you take the mean a small state weighs as strongly as a large one. Since we are not looking for the mean consumption of all states, summing seems more apt.

The last four years of data are taken as a holdout set by splitting the data before / inclusive after January 2015

Visualisation



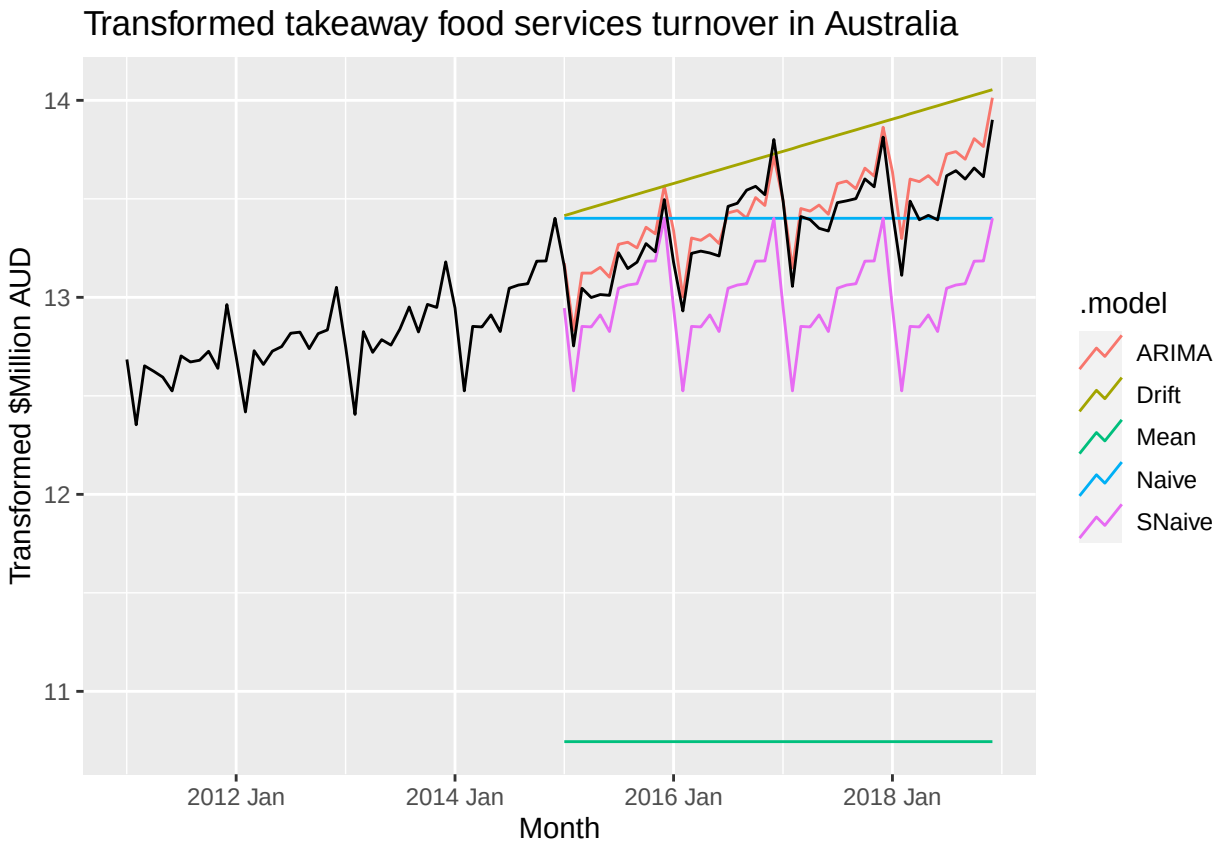
We can see an upward trend, seasonality with a yearly period, and some cyclic behaviour. The data has been transformed with the Box-Cox formula using $\lambda \approx 0.15$ to stabilize the variance

2.2)

Define the model

we are using the models *Seasonal naïve*, *Naïve*, *Drift* and *Mean*. The *ARIMA* model is included as a generally well performing benchmark.

Train the model



2.3)

Forecasting model accuracies

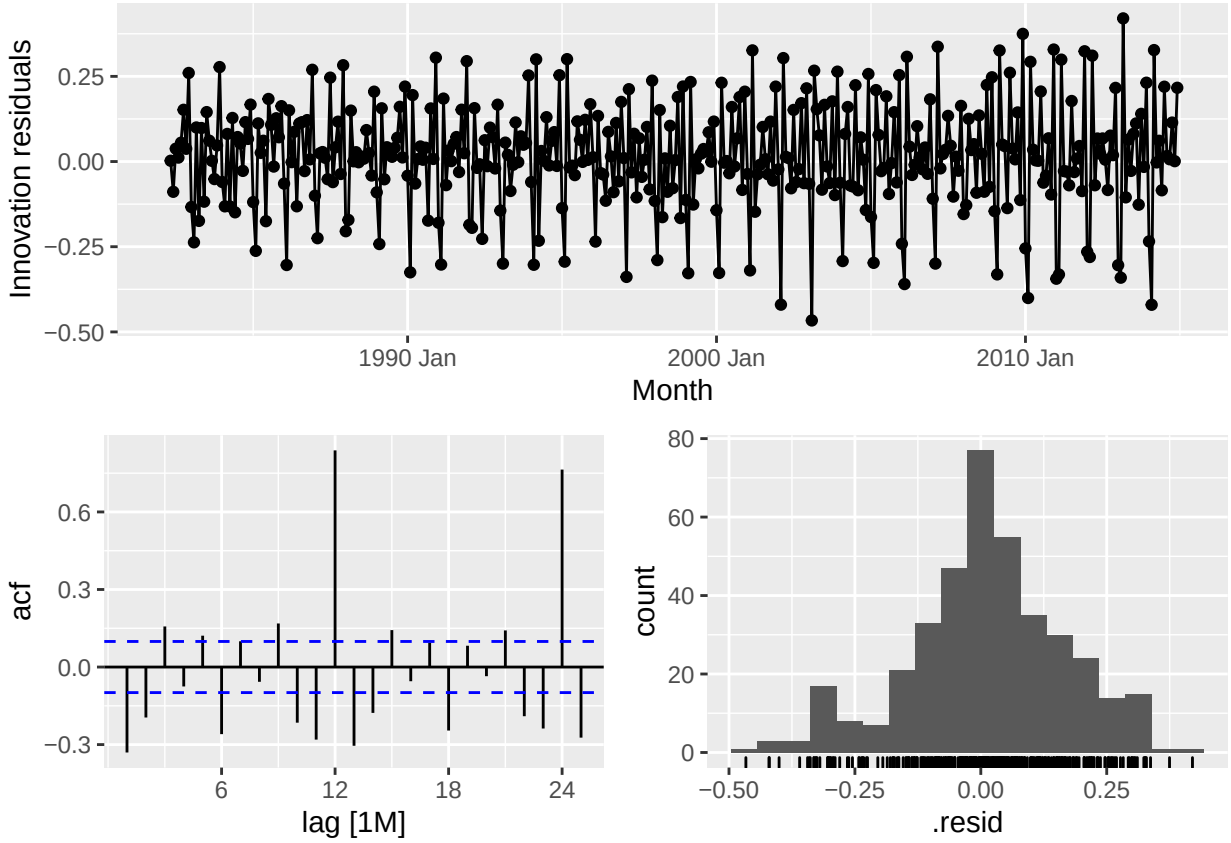
as fitted to the transformed *takeaway food services* turnover in Australia

.model	RMSE	MAE	MAPE	MASE
ARIMA	0.106	0.0945	0.707	0.517
Naive	0.245	0.1981	1.489	1.083
Drift	0.403	0.3698	2.781	2.022
SNaive	0.412	0.3796	2.830	2.075
Mean	2.634	2.6232	19.597	14.341

From the graph it becomes obvious that *Mean* performs very poorly and our accuracy table reflects this.

ARIMA out performs all the simpler benchmark methods by a stretch, following the true data quite closely.

Naive scores best compared to the simple benchmarks. This can be explained by looking at the graph: The forecasts start on a seasonal peak and the real data has an upward trend, thus the model's horizontal line crosses a lot of the data "accidentally". Because of this however, *Naive* is not likely to be an effective forecasting model for this data in the long term, nor have white noise like residuals because it neglects seasonality and trend.



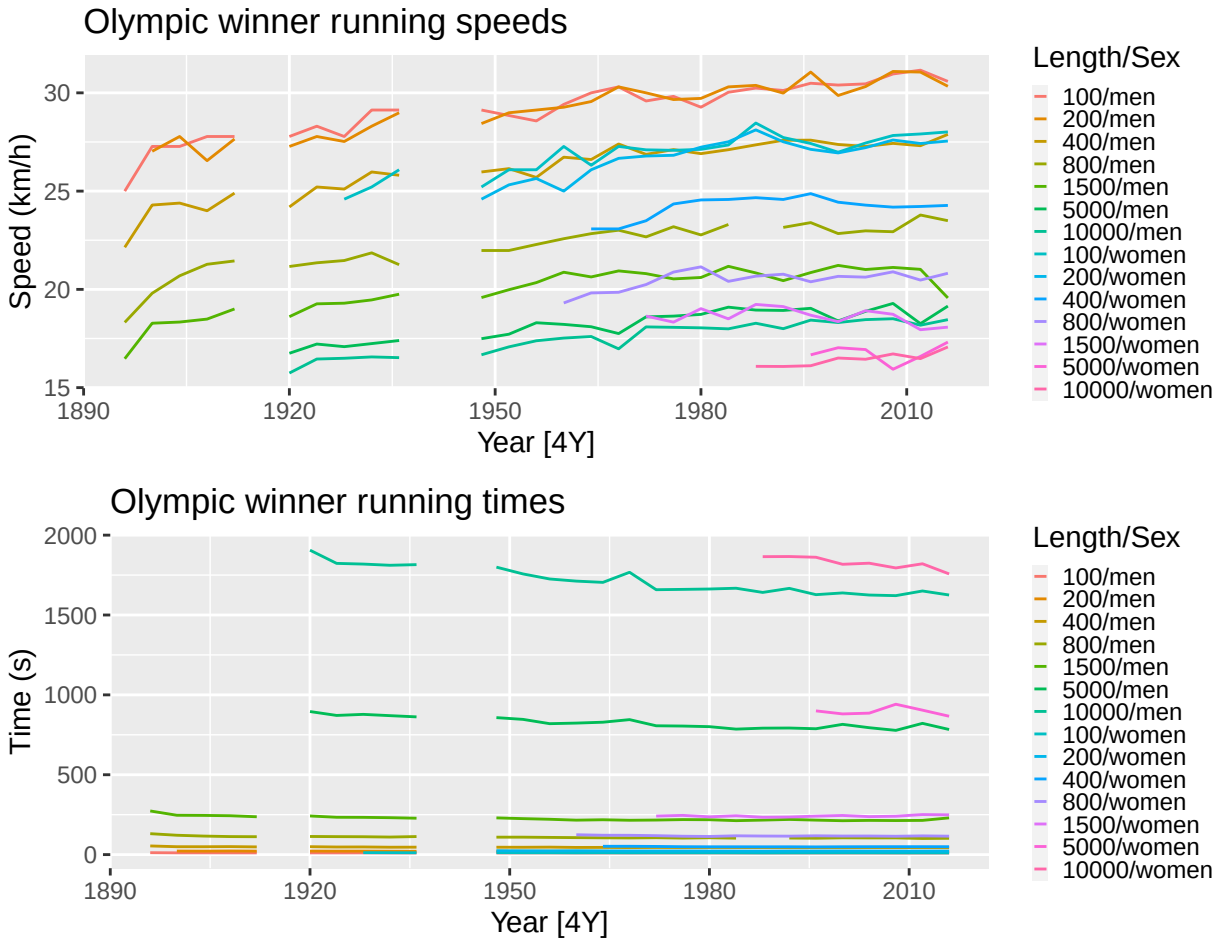
From the *Naive* model residuals plot, we can quickly confirm the previous intuition. The time plot shows a growing variance, the ACF has 18 out of 25 lags beyond the confidence interval, and it has a non-uniform looking distribution of residuals. This means that the *Naive* model has bias and does not behave like white noise.

The *ARIMA* model on the other hand (residual plots in Appendix 2.3), behaves more like white noise. The timeline has a stable variance, it has only two lags slightly outside of the interval, and the residuals look almost normal.

Of course, we should remember to back-transform if we were to use any of these models.

Exercise 3

3.1)



The data has been transformed from time (in seconds) to speed (in kilometer/hour). This normalizes the data and changes them to a better comparable domain by taking out the major differences between the the competitions caused by the length of a run. The transformation results in a more easily interpretable graph.

Almost all of the trends before 1920 seem steeper. Because the trends seem to be root like in nature, fitting a straight linear prediction model with a single will probably not work out.

We can also immediately notice that some points are missing. These are 1916, 1940 & 1944: No olympic games took place in these years. Other than that some races were added later. Men's 200, 5000 & 10000 were added between 1980 and 1920. All women's races were added between 1926 and 1996.

The women's 5000 & 10000 are 6 and 8 time steps long respectively, so they may not be the most insightful. An upward trend in speed (and to a lesser extent downward in time) can be seen in almost all races, with the relatively steady women's 1500 (and the short downward 5000) as an exception.

There is no seasonality present, cyclic behaviour may exist but it is more likely to be caused by

noise.

3.2)

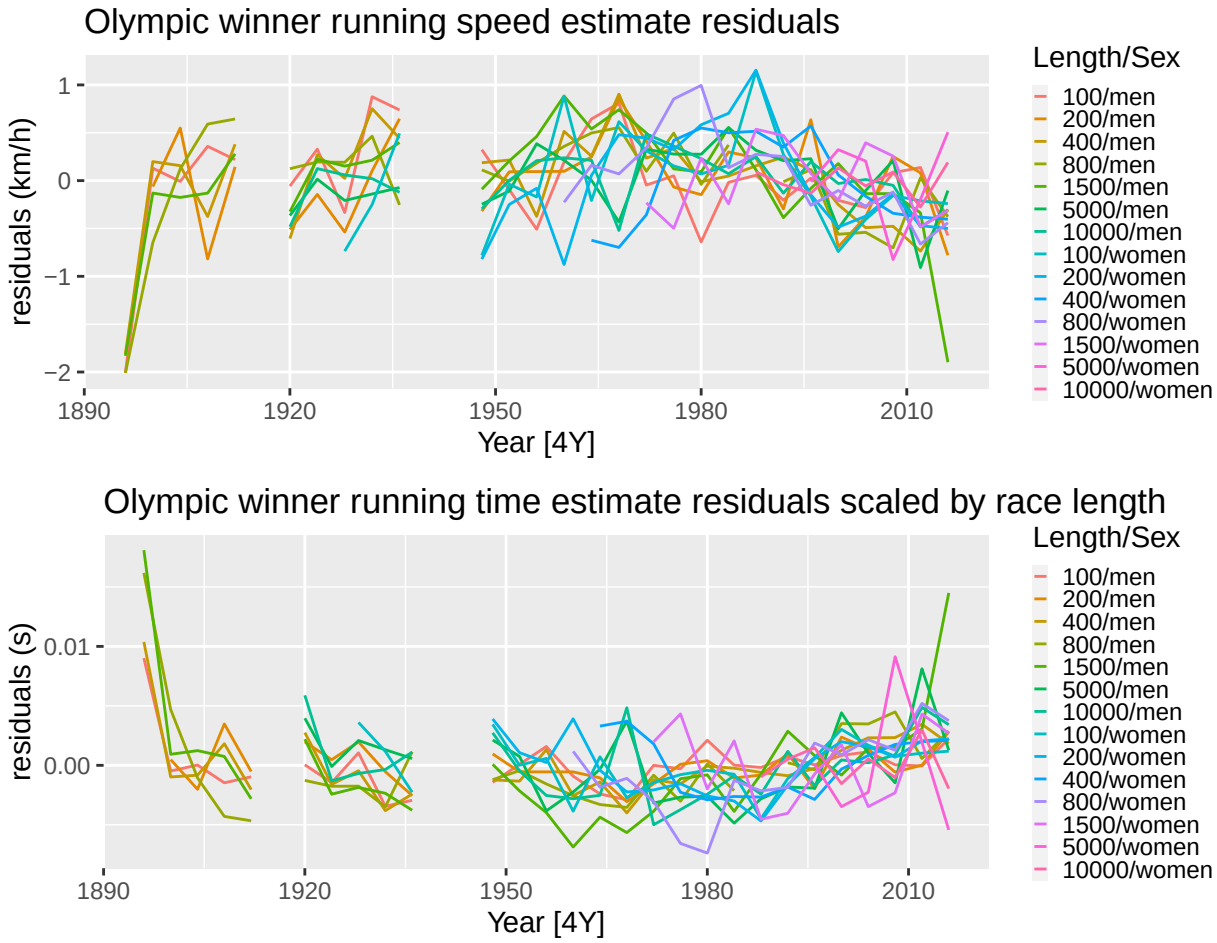
Note:
A linear downward trend in time will cross zero into the negative, which is technically impossible in running. A linear upward trend in speed will still hit impossibly fast running speeds, but will never become negative when back transformed to time.
However, since the exercise asks for time (even though speed makes more sense), and the conversion from δ speed to δ time is not linear, which makes answering the questions unnecessarily difficult, I will proceed with time and speed models both.

Estimated yearly running time and speed change
as obtained from linear models fitted to the olympic running data

Length	Sex	est. time change (s)	est. speed change (km/h)
100	men	-0.0126	0.03458
100	women	-0.0142	0.03323
200	men	-0.0249	0.03469
200	women	-0.0336	0.03879
400	men	-0.0646	0.03530
400	women	-0.0401	0.01874
800	men	-0.1518	0.02950
800	women	-0.1980	0.03079
1500	men	-0.3151	0.02634
1500	women	0.1468	-0.01107
5000	men	-1.0299	0.02229
5000	women	-0.3030	0.00654
10000	men	-2.6659	0.02650
10000	women	-3.4962	0.03168

According to our linear speed models the winning speeds seem to be getting faster by about 0.01 to 0.04 km/h per year and according to the linear time models, the winning times are getting shorter. The changes in time are much less consistent because they are dependent on length. The women's 1500 is the only exception, it has been getting slower over the years, but as discussed in *(3.1)* it has few entries so that model might not be accurate over a longer period

3.3)



From the plotted residuals we can see the steeper starts affect the estimates, the residuals are much larger there. There seems to be a downward trend (bias) in the residuals in the last 30 years, indicative that estimate will not hold up very well in the longer term because. This means that the model is not capturing something.

Because running time scales with track length, we scale the time residuals by the race length, and see that this graph tells the same story.

3.4)

2020 Olympic winning time and speed predictions

as obtained from linear models fitted to the olympic running data

		Time (s)			Speed (km/h)		
Length	Sex	Lower	Pred	Upper	Lower	Pred	Upper
100	men	9.05	9.53	9.05	30.1	31.3	32.5
100	women	10.06	10.53	10.06	27.3	28.4	29.5
200	men	18.42	19.12	18.42	30.3	31.3	32.2
200	women	20.11	21.20	20.11	27.0	28.2	29.5
400	men	39.67	42.04	39.67	27.2	28.3	29.5

400	women	46.12	48.44	46.12	23.6	24.7	25.9
800	men	92.14	99.24	92.14	22.8	24.0	25.2
800	women	104.11	111.48	104.11	20.2	21.4	22.5
1500	men	190.09	207.00	190.09	20.3	21.6	22.9
1500	women	233.79	245.46	233.79	17.4	18.3	19.2
5000	men	739.74	772.82	739.74	18.6	19.3	20.1
5000	women	814.66	892.11	814.66	15.4	16.8	18.3
10000	men	1519.30	1580.35	1519.30	18.3	18.9	19.5
10000	women	1717.90	1763.02	1717.90	16.6	17.0	17.4

The winning times (s) and speed (km/h) for the 2020 olympics as predicted by the linear models can be read from the table above.

To make these prediction with linear models the a number of assumption have been made:
- a linear relationship between the year and running speed exists. This is unlikely to actually be the case in the real world as humans have physical limits. Running times will never approach zero (or other very short times) nor are they technically able to become negative.
- The residuals of the model have a 0 mean. Judging by the plot in *(3.3)* this did not seem to be the case for the last 30 years.
- There is no autocorrelation or seasonality. This assumption was made explicit in *(3.1)*.

Appendix

Note: Due to an error in Knitr that I have not been able to fix, ggplot titles and labels are not being wrapped properly. However, you are able to read the contents in the graphs just fine.

Setup code

```
options(digits = 3)
library(fpp3)
library(latex2exp)
library(forecast)
library(formatR)
library(gridExtra)
library(gt)

knitr::opts_chunk$set(echo = FALSE, message = FALSE, warning = FALSE,
    cache = TRUE, dev.args = list(pointsize = 11))
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
```

1.1 code

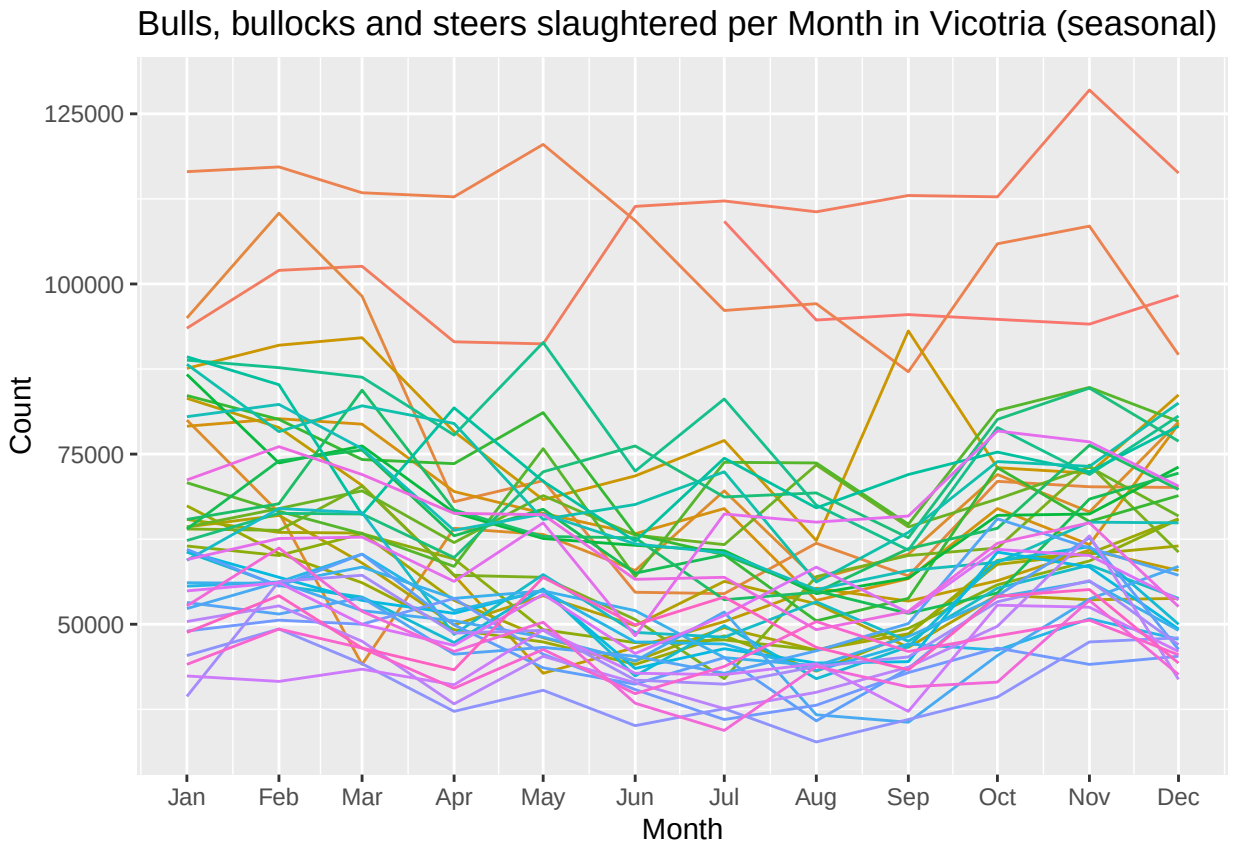
```
usecon <- global_economy %>%
    filter(Country == "United States")
us_recessions <- c(1960, 1970, 1973, 1980, 1990, 2001, 2008)  #https://www.thebalance.com/the-
```

```r
usgdp <- usecon %>%
    autoplot(GDP) + labs(y = "US$", title = "GDP per year for the United States")

usgdpcap <- usecon %>%
    autoplot((GDP/Population/CPI) * 100) + labs(y = "US$", title = "inflation adjusted GDP per
    geom_vline(xintercept = us_recessions, colour = "gray", linetype = "longdash")

grid.arrange(usgdpcap, usgdp, ncol = 1)
```

1.2 code

```r
bslaus <- aus_livestock %>%
    filter(Animal == "Bulls, bullocks and steers", State == "Victoria") %>%
    select(Count, Month)

lambda <- bslaus %>%
    features(Count, features = guerrero) %>%
    pull(lambda_guerrero)

# # Were not mutating because lambda = -0.072 does nothing
# bslaus <- bslaus %>% mutate(Count = BoxCox(Count,
# lambda))

data_p <- bslaus %>%
    autoplot(Count) + labs(y = "Count", title = "Bulls, bullocks and steers slaughtered per mo

acf_p <- bslaus %>%
    ACF(Count, lag_max = 12 * 10) %>%
    autoplot() + labs(title = "ACF") + geom_vline(xintercept = seq(0,
    120, 12), colour = "pink", alpha = 0.7)

grid.arrange(data_p, acf_p, ncol = 1)
```

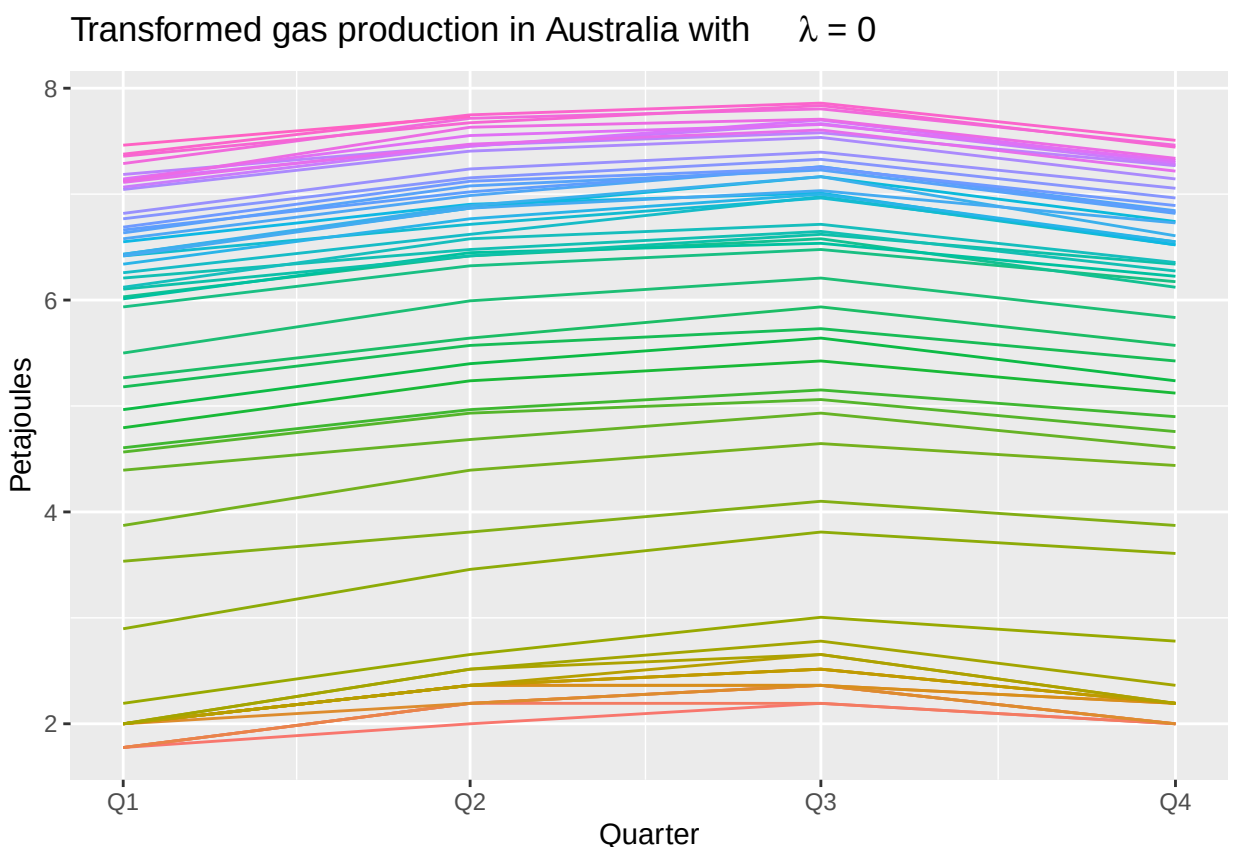
```r
season_p <- bslaus %>%
    gg_season(Count, period = "year") + theme(legend.position = "none") +
    labs(y = "Count", title = "Bulls, bullocks and steers slaughtered per Month in Vicotria (s
season_p
```



1.3 code

```
gasaus <- aus_production %>%
    select(Gas, Quarter)

lambda <- gasaus %>%
    features(Gas, features = guerrero) %>%
    pull(lambda_guerrero)

gasaus <- gasaus %>%
    mutate(Gas = BoxCox(Gas, lambda))

data_p <- gasaus %>%
    autoplot(Gas) + labs(y = "Transformed production (Petajoules)",
    title = latex2exp::TeX(paste0("Transformed gas production in Australia with $\\;\\;\\lambda
        round(lambda, 2))))

acf_p <- gasaus %>%
    ACF(lag_max = 4 * 6) %>%
    autoplot() + labs(title = "ACF") + geom_vline(xintercept = seq(0,
    24, 4), colour = "pink", alpha = 0.7)

grid.arrange(data_p, acf_p, ncol = 1)
```

```
season_p <- gasaus %>%
    gg_season(Gas, period = "year") + theme(legend.position = "none") +
    labs(y = "Petajoules", title = latex2exp::TeX(paste0("Transformed gas production in Austral
        round(lambda, lambda))))
season_p
```



2.1 code

```
foodaus <- aus_retail %>%
    filter(Industry == "Takeaway food services") %>%
    filter(!is.na(Turnover)) %>%
    summarise(Turnover = sum(Turnover))

lambda <- foodaus %>%
    features(Turnover, features = guerrero) %>%
    pull(lambda_guerrero)

foodaus <- foodaus %>%
    mutate(Turnover = BoxCox(Turnover, lambda))

foodaus_test <- foodaus %>%
```

```
    filter(year(Month) >= 2015)
foodaus_train <- foodaus %>%
    filter(year(Month) < 2015)
```

```
foodaus %>%
    autoplot(Turnover) + labs(y = "Transformed $Million AUD",
    title = latex2exp::TeX(paste0("Takeaway food services turnover in Australia with $\\;\\;\\
        round(lambda, 2))))
```

2.2 code

```
data_fit <- foodaus_train %>%
    model(SNaive = SNAIVE(Turnover), Naive = NAIVE(Turnover),
        Drift = RW(Turnover ~ drift()), Mean = MEAN(Turnover),
        ARIMA = ARIMA(Turnover))
```

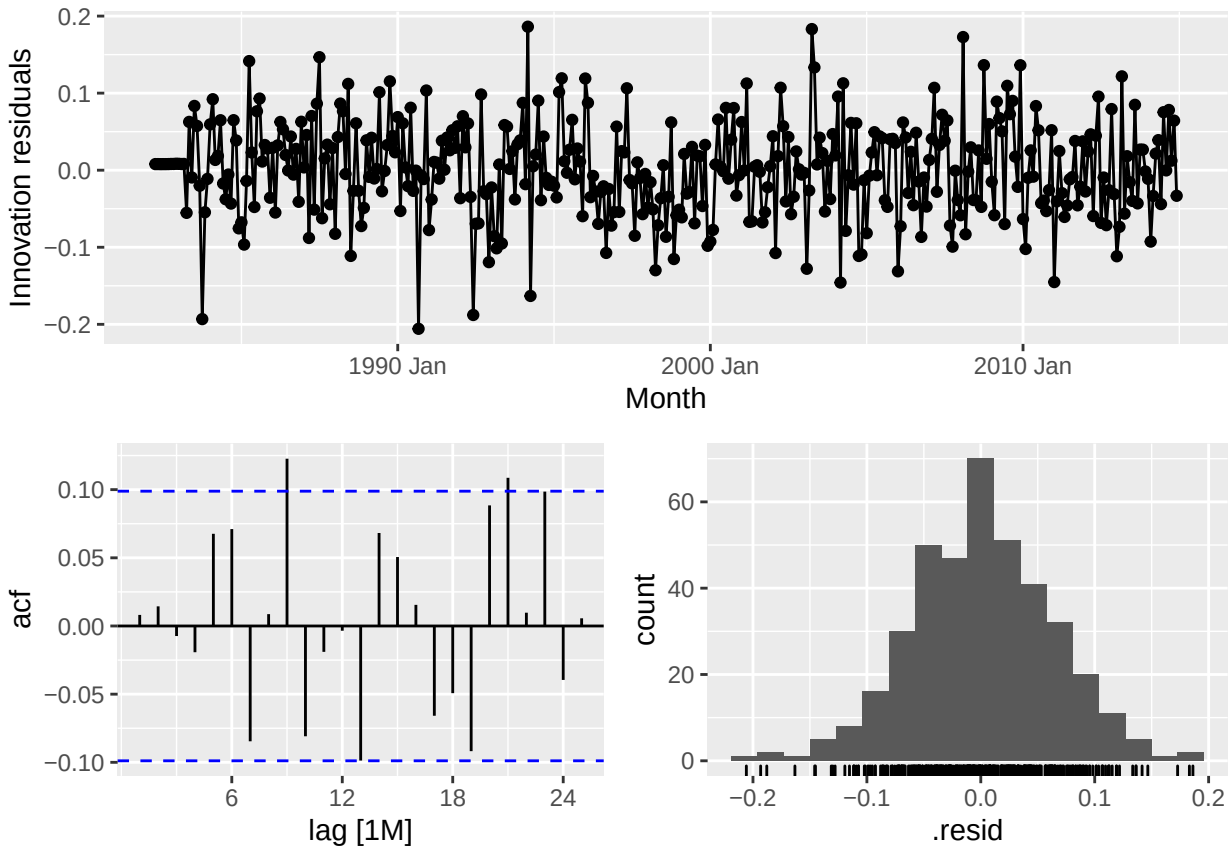
```
data_forecast <- data_fit %>%
    forecast(h = "4 years")

data_forecast %>%
    autoplot((foodaus %>%
        filter(year(Month) > 2010)), level = NULL) + labs(y = "Transformed $Million AUD",
    title = "Transformed takeaway food services turnover in Australia")
```

2.3 code

```
accuracy(data_forecast, foodaus) %>%
    select(c(".model", "RMSE", "MAE", "MAPE", "MASE")) %>%
    arrange(MASE) %>%
    gt() %>%
    tab_header(title = md("**Forecasting model accuracies**"),
        subtitle = md("as fitted to the transformed _takeaway food services_ turnover in Austra
    opt_align_table_header(align = "center")
```

```
gg_tsresiduals(data_fit %>%
    select(Naive))
```

```
gg_tsresiduals(data_fit %>%
    select(ARIMA))
```



3.1 code

```
olyrun <- olympic_running %>%
    mutate(Speed = (Length/Time) * 3, 6)

a <- olyrun %>%
    autoplot(Speed) + labs(y = "Speed (km/h)", title = "Olympic winner running speeds") +
    theme(legend.key.size = unit(0.2, "cm"))
b <- olyrun %>%
    autoplot(Time) + labs(y = "Time (s)", title = "Olympic winner running times") +
    theme(legend.key.size = unit(0.2, "cm"))

grid.arrange(a, b, ncol = 1)
```

3.2 code

```
models_v <- olyrun %>%
    model(TSLM(Speed ~ Year))

models_t <- olyrun %>%
    model(TSLM(Time ~ Year))
```

```
model_coefs <- coef(models_v) %>%
    filter(term == "Year") %>%
    rename(Speed = estimate) %>%
    select(-c(term, std.error, statistic, p.value, .model)) %>%
    full_join(coef(models_t) %>%
        filter(term == "Year") %>%
        rename(Time = estimate) %>%
        select(-c(term, std.error, statistic, p.value, .model)))

model_coefs %>%
    rename(`est. speed change (km/h)` = Speed, `est. time change (s)` = Time) %>%
    select(c("Length", "Sex", "est. time change (s)", "est. speed change (km/h)")) %>%
    gt() %>%
    tab_header(title = md("**Estimated yearly running time and speed change**"),
        subtitle = md("as obtained from linear models fitted to the olympic running data")) %>%
    opt_align_table_header(align = "center")
```

3.3 code

```
p1 <- models_v %>%
    augment() %>%
    select(!.model) %>%
    autoplot(.innov) + labs(y = "residuals (km/h)", title = "Olympic winner running speed estim
    theme(legend.key.size = unit(0.2, "cm"))

p2 <- models_t %>%
    augment() %>%
    select(!.model) %>%
    autoplot(.innov/Length) + labs(y = "residuals (s)", title = "Olympic winner running time es
    theme(legend.key.size = unit(0.2, "cm"))

grid.arrange(p1, p2, ncol = 1)
```

3.4 code

```
fc_olyrun_v <- forecast(models_v, h = 1, interval = "predict") %>%
    ungroup() %>%
    mutate(Lower = hilo(Speed, 95)$lower, Upper = hilo(Speed,
        95)$upper) %>%
    rename(Pred = .mean) %>%
    as_tibble() %>%
    select(-c(Year, .model))
fc_olyrun_t <- forecast(models_t, h = 1, interval = "predict") %>%
    ungroup() %>%
    mutate(`Lower ` = hilo(Time, 95)$lower, `Upper ` = hilo(Time,
        95)$lower, `   ` = "  ", `    ` = " ") %>%
```

```
    rename(`Pred ` = .mean) %>%
    as_tibble() %>%
    select(-c(Year, .model))

full_join(fc_olyrun_t, fc_olyrun_v) %>%
    select(c(Length, Sex, "   ", "Lower ", "Pred ", "    ", "Upper ",
        Lower, Pred, Upper)) %>%
    gt() %>%
    tab_spanner(label = "Time (s)", columns = c("Lower ", "Pred ",
        "Upper ")) %>%
    tab_spanner(label = "Speed (km/h)", columns = c(Lower, Pred,
        Upper)) %>%
    tab_header(title = md("**2020 Olympic winning time and speed predictions**"),
        subtitle = md("as obtained from linear models fitted to the olympic running data")) %>%
    opt_align_table_header(align = "center")
```