Time Series – Group Project || Prof. Francesca Lipari

José Ignacio Díez Ruiz	–	100487766
Carlos Roldán Piñero	–	100484904
Pablo Vidal Fernández	–	100483812
Jialian Zhou He	–	100407485

February 2023

Exercise 1

Use the appropiate graphics functions, explore features from the following time series: **bicoal**, **bricksq**, **hsales**, **ibmclose**, **Internet**, **writing**.

- Can you spot any seasonality, cyclicity and trend?
- What do you learn about the time series?
- Justify the choice of the graphic function.

Bicoal TS

According to the documentation, this time series contains the annual bituminous coal production in the USA between 1920 and 1968.

We check that the frequency of the time series is annual:

```
frequency(bicoal)
```

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

Plotting the series and the first differences:

```
p1 <- autoplot(bicoal, main = "Bituminous coal production \nin the USA",
               xlab = "Year",
               ylab = "Bicoal")

p2 <- autoplot(diff(bicoal), main = "Bituminous coal
               production  \nin the USA",
               xlab = "Year",
               ylab = "First difference")
```

```
ggarrange(p1, p2, ncol = 2)
```

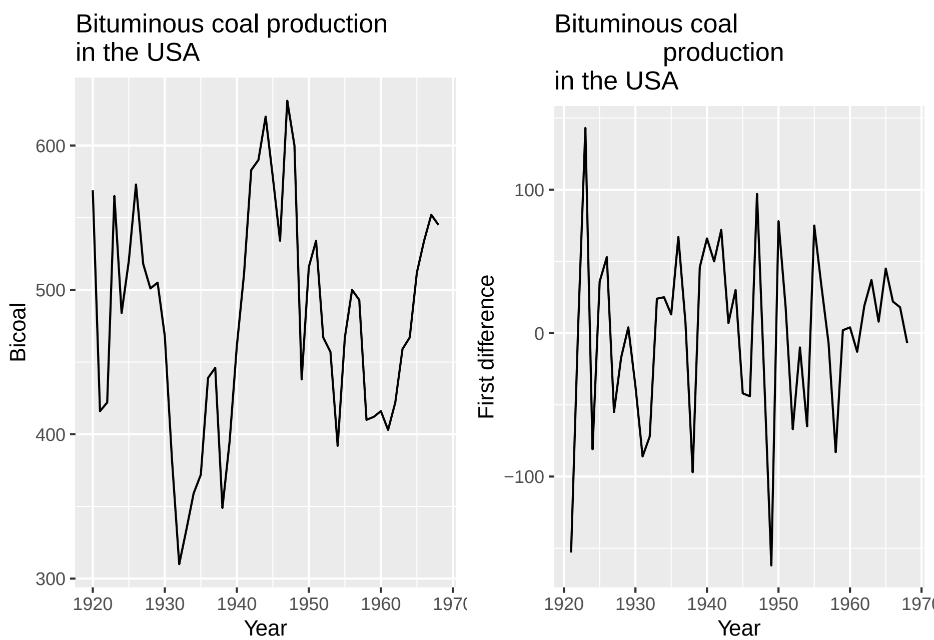


Figure 1: Bicoal

It is annual data, therefore it cannot be seasonal.

We can see that there is no overall trend, and there seems to be a cyclic behaviour every 10 years. The plot for the differences resembles white noise. At this point, one could suspect stationarity. For that, we will use the correlogram:

```
ggAcf(bicoal, lag.max = 45) + ggtitle("ACF Bituminous coal production")
```

We can observe a sinusoidal pattern, and it goes towards zero. It seems that this time series is stationary, with a cyclic component.

Bricksq TS

According to the documentation, this time series contains the Australian quarterly clay brick production between 1956 and 1994.

We check that the frequency of the time series is quarterly.

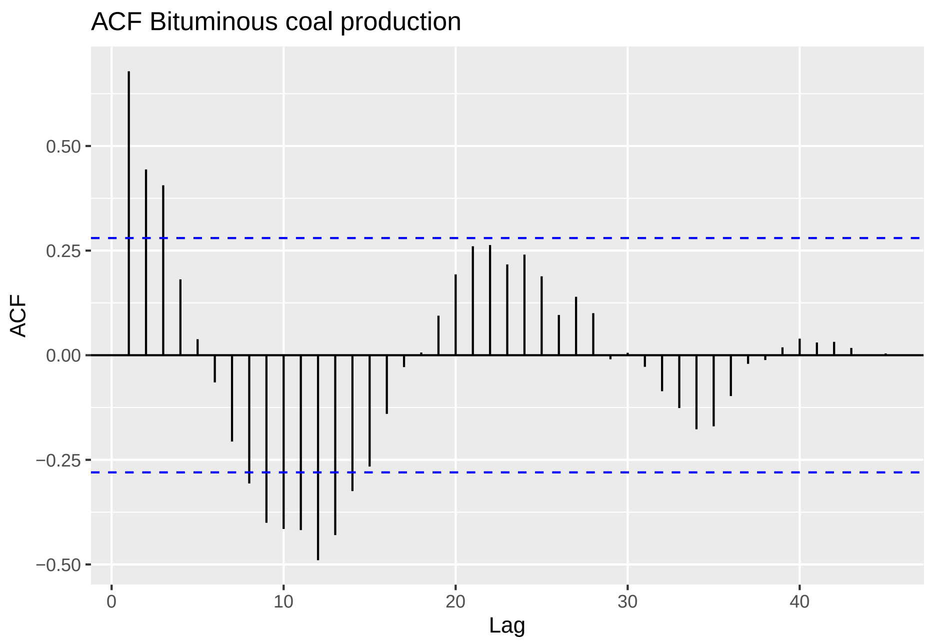


Figure 2: Bicoal ACF

```
frequency(bricksq)
```

```
## [1] 4
```

Plotting the series:

```
par(mfrow=c(1,2))
p3 <- autoplot(bricksq, main = "Australian clay \nbrick production",
               xlab = "Year",
               ylab = "Bricksq")
p4 <- autoplot(diff(bricksq, lag = 1),
               main = "Australian clay \nbrick production",
               xlab = "Year",
               ylab = "First differences")
ggarrange(p3, p4, ncol = 2)
```

Looking at the plot, we can see that there might be a seasonal component and a trend. The series doesn't appear to be stationary. It also seems that there is heterocedasticity.

```
ggAcf(bricksq) + ggtitle("ACF Australian clay brick production")
```

As the autocorrelations for the first lags are large and they slowly decrease, we can say that there is a trend. Moreover, we can see that each 4 lags the autocorrelation is higher, a sign of seasonality.

```
p5 <- ggseasonplot(bricksq) +
    ggtitle("Australian clay brick production")
p5
```

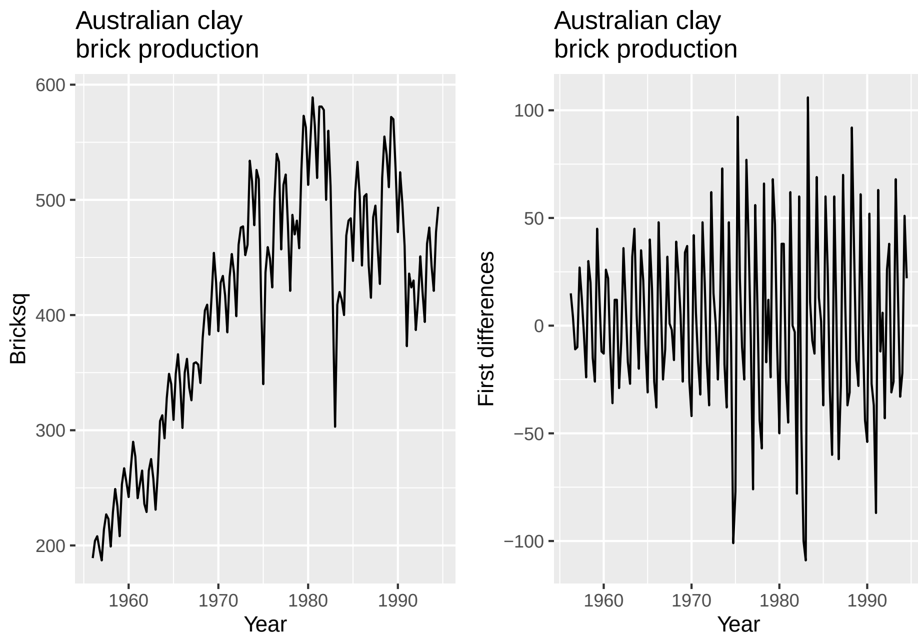


Figure 3: Bricksq plot

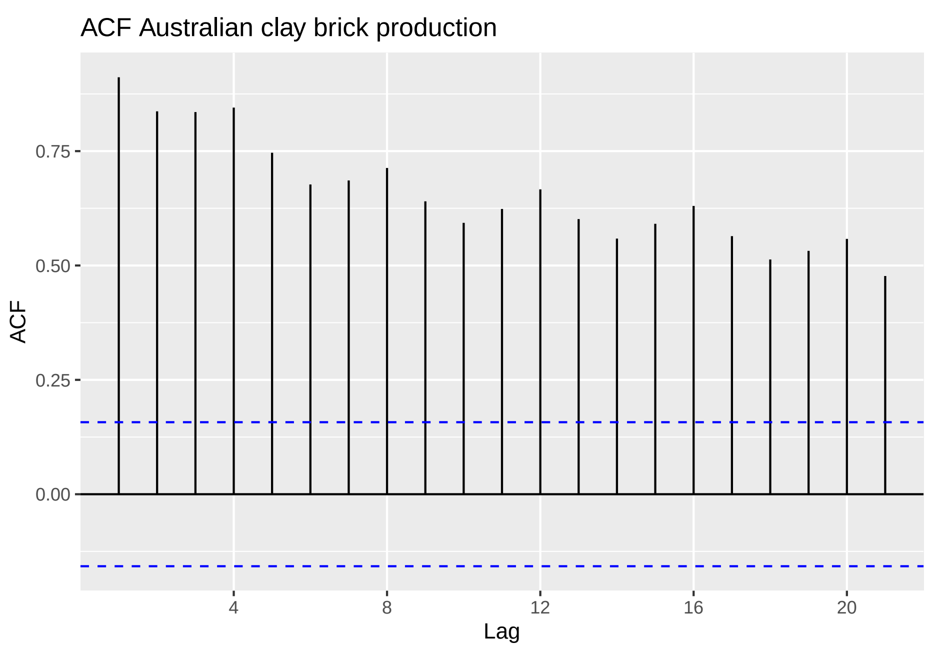


Figure 4: ACF bricksq

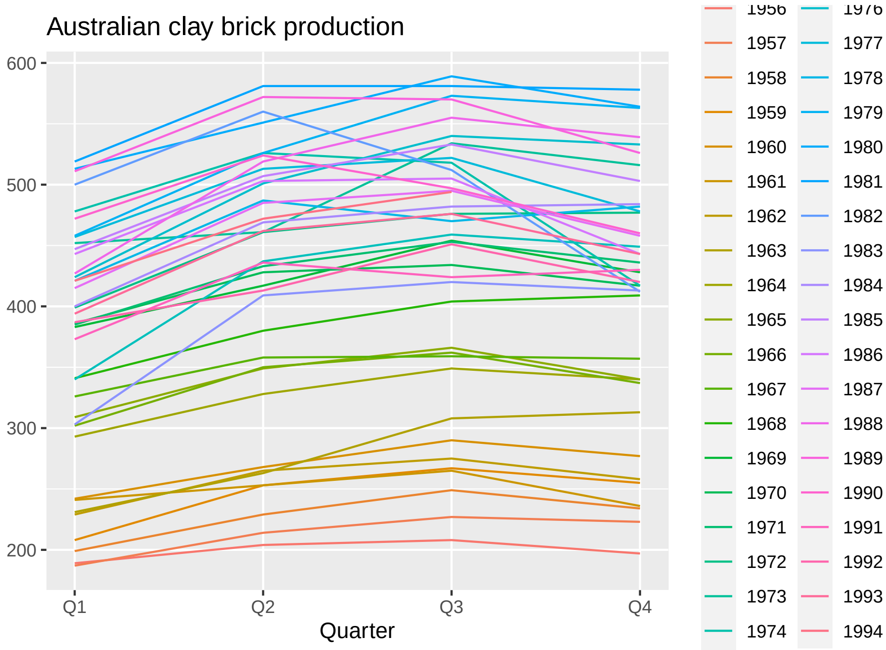


Figure 5: Bricksq seasonal plot

There are a few outliers in the Q4 for some years, and all years seem to present a very similar pattern: Q2 and Q3 are higher than Q1 and Q4.

We can appreciate this more clearly in the subseries plot:

```
p6 <- ggsubseriesplot(bricksq, ylab = "",
                      main = "Australian clay brick production")
p6
```

Hsales TS

According to the documentation, this time series contains the monthly sales of new one-family houses sold in the USA since 1973.

We check that the frequently is monthly:

```
frequency(hsales)
```

```
## [1] 12
```

We have monthly data, thus a seasonal component might exist.

```
p7 <- autoplot(hsales, main = "Sales of new one-family \nhouses")
p8 <- autoplot(diff(hsales, lag = 1),
               main = "Sales of new one-family \nhouses",
               xlab = "Year",
               ylab = "First differences")

ggarrange(p7, p8, ncol = 2)
```

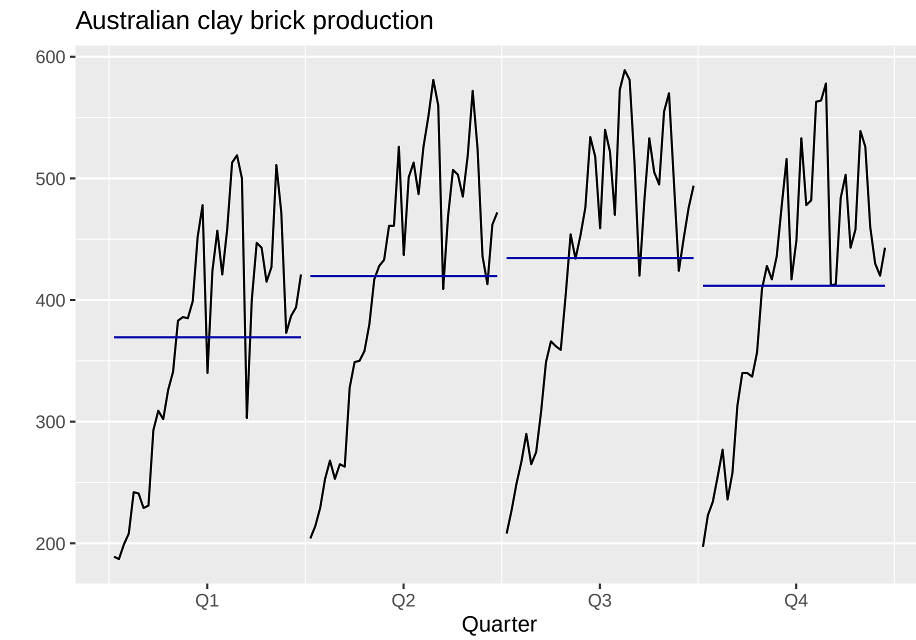


Figure 6: Bricksq subseries plot

Figure 7: Plot for hsales

At a first glance, we can see a cyclical component in the series. There doesn't appear to be any trend. Moreover, a seasonal component can be appreciated within each year.

```
p9 <- ggseasonplot(hsales,
                   main = "Sales of new one-family \nhouses")
p9
```

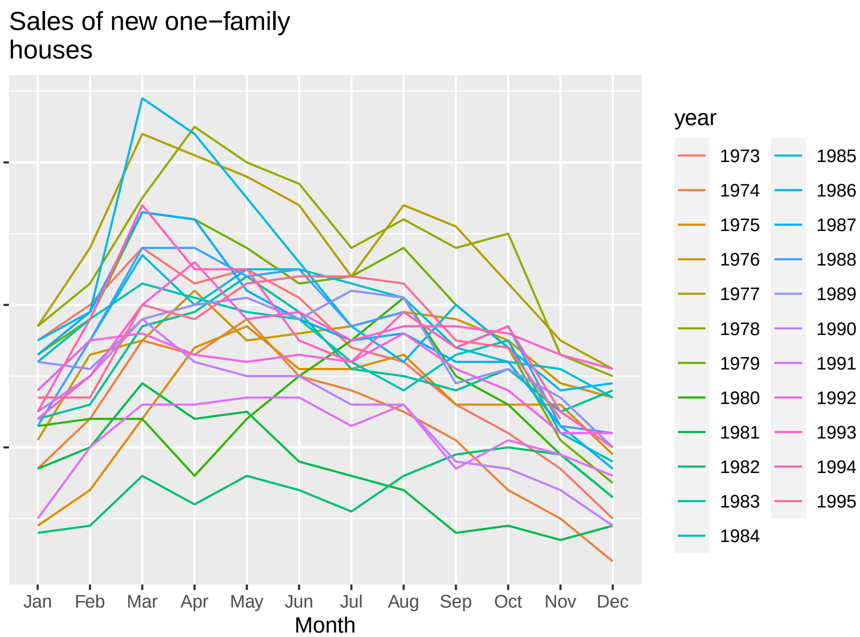


Figure 8: Hsales season plot

We can see that March and April tend to be the months with higher volume, whereas the last months of the year come with a decrease. In August, there seems to be a recuperation with respect to July.

```
p10 <- ggAcf(hsales) +
    ggtitle("ACF Sales of new one-family \nhouses")
p10
```

There are peaks at 12 and 24 in the correlogram, confirming seasonality. There is no evidence for a trend.

Ibmclose TS

According to the documentation, the series contains the daily closing IBM stock price.

We check that the frequency is daily:

```
frequency(ibmclose)
```

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

Plotting the series:

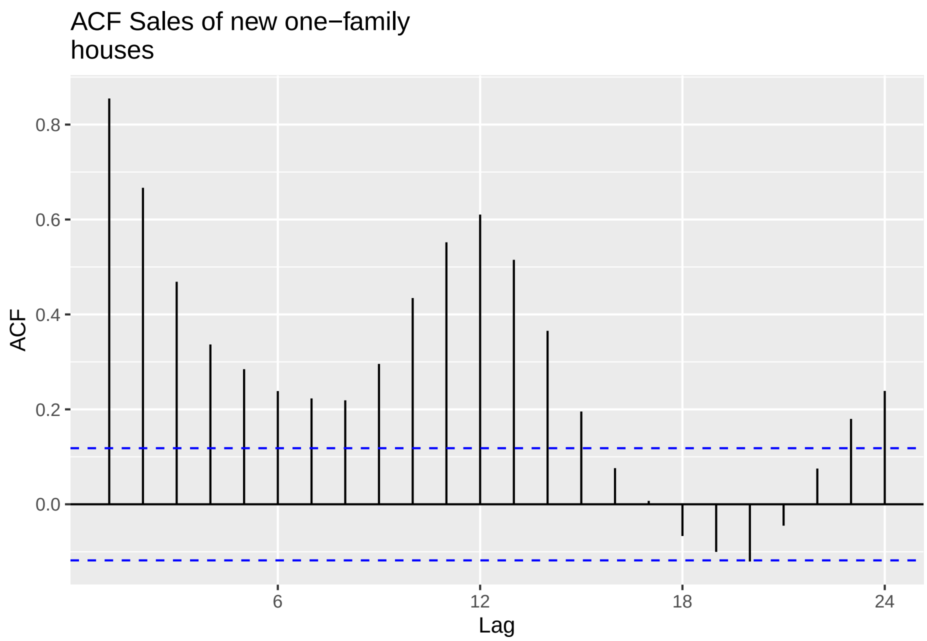


Figure 9: ACF Hsales

```
autoplot(ibmclose, main = "IBM stock price")
```

The data is daily, so there can't be a seasonal nor cyclic component. It looks like there is a negative trend, although it doesn't seem to be linear.

```
ggAcf(ibmclose, lag.max = 50) + ggtitle("ACF IBM stock price")
```

The series is definitely not stationary. Correlations are very high and don't go rapidly towards zero, confirming that there is a trend.

Internet TS

Using *help*, we can see that the dataset contains the number of user logged on to an internet server each minute over a 100-minute period. Consequently, there won't be a cyclic or seasonal component.

```
frequency(internet)
```

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

```
autoplot(internet,
         xlab = "Number of logged users",
         main = "Internet")
```

It doesn't look like there is a trend. The series goes up, then goes down and finally it starts rising again.

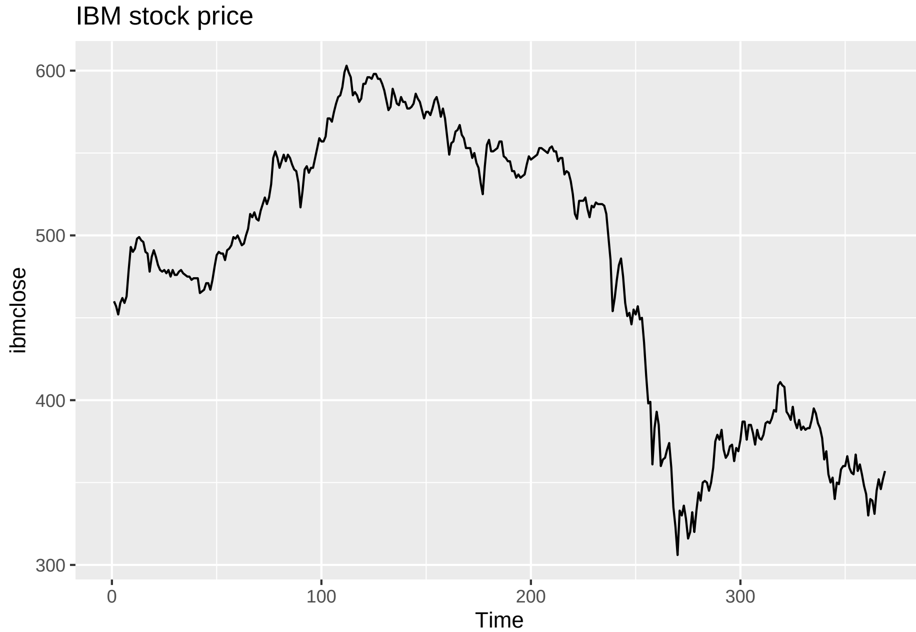


Figure 10: Plot for ibmclose

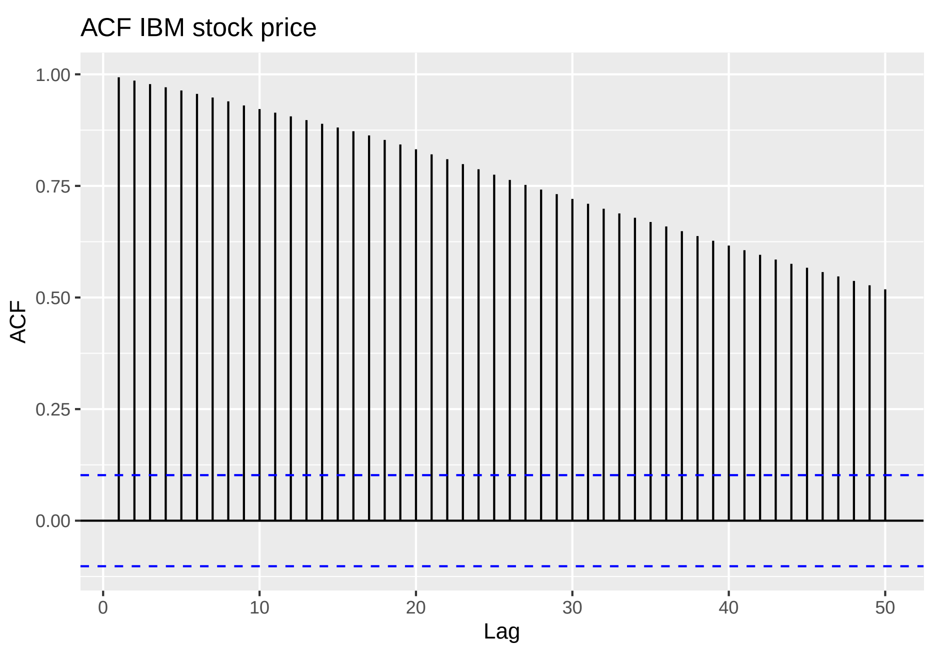


Figure 11: ACF ibmclose

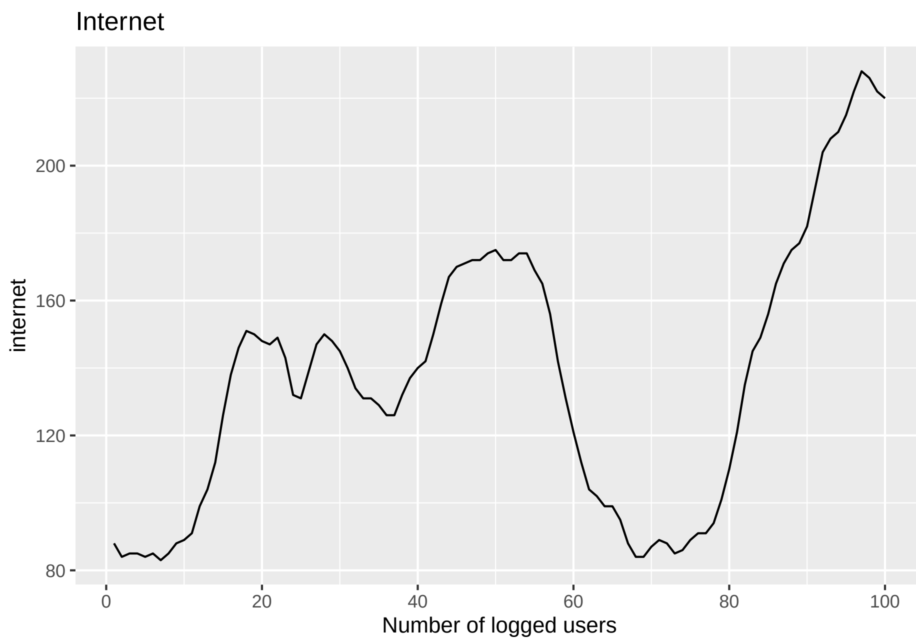


Figure 12: Plot for internet

```
ggAcf(internet, lag.max = 90) +
    ggtitle("ACF Internet")
```

We can see oscillations between positive and negative autocorrelations, consistent with the ups and downs in the series. It doesn't go rapidly towards zero, so the data is not stationary.

Writing TS

According to the documentation, the series contains the industry sales for printing and writing paper in France from Jan 193 to Dec 1972.

Checking that the frequency is monthly:

```
frequency(writing)
```

```
## [1] 12
```

Plotting the data:

```
autoplot(writing,
         main = "Industry sales",
         xlab = "French francs")
```

We can see a positive linear trend, a strong seasonality and there doesn't seem to be a cyclic component.

```
ggseasonplot(writing,
             main = "Industry sales",
             xlab = "French francs")
```

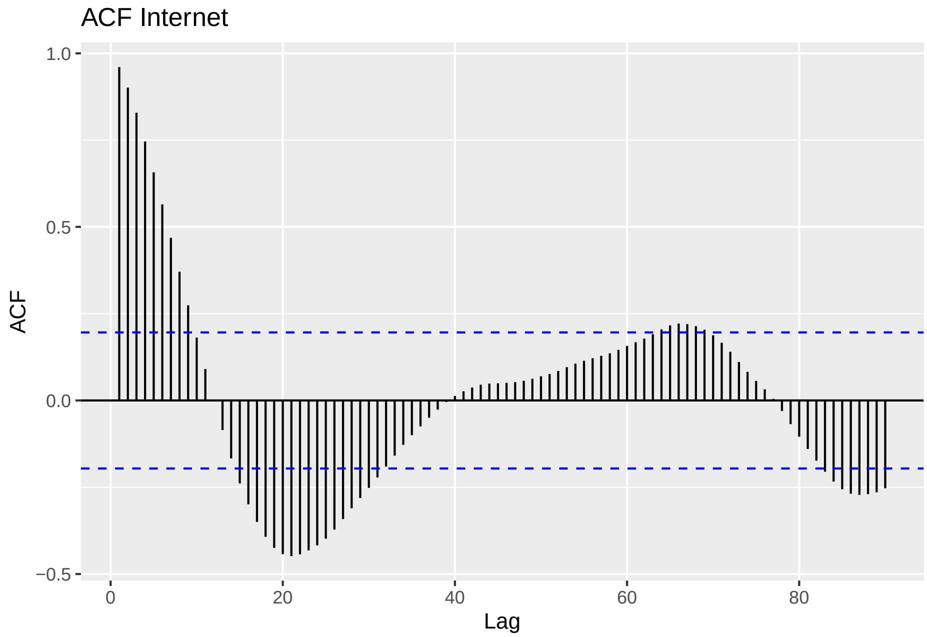


Figure 13: ACF internet

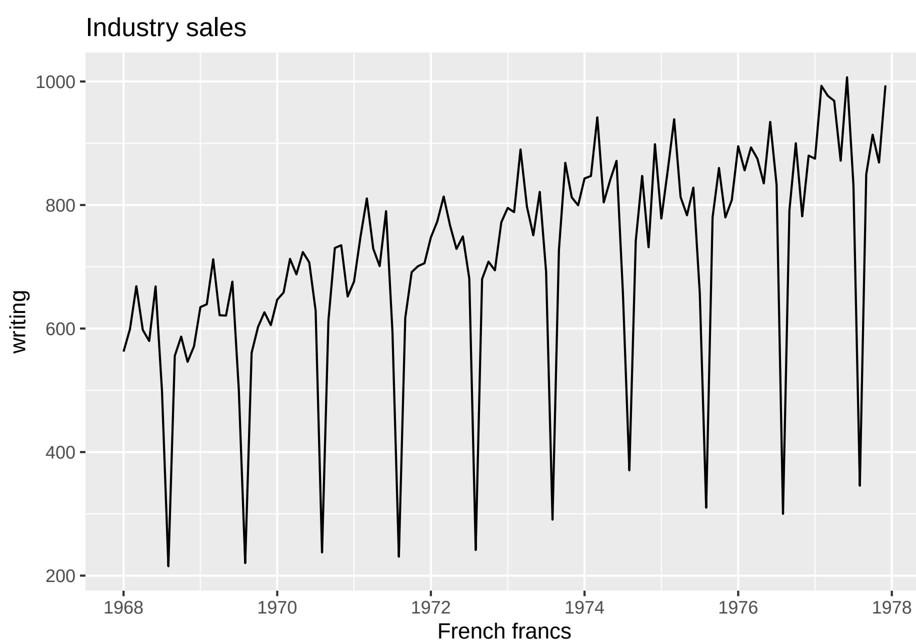


Figure 14: Plot for writing

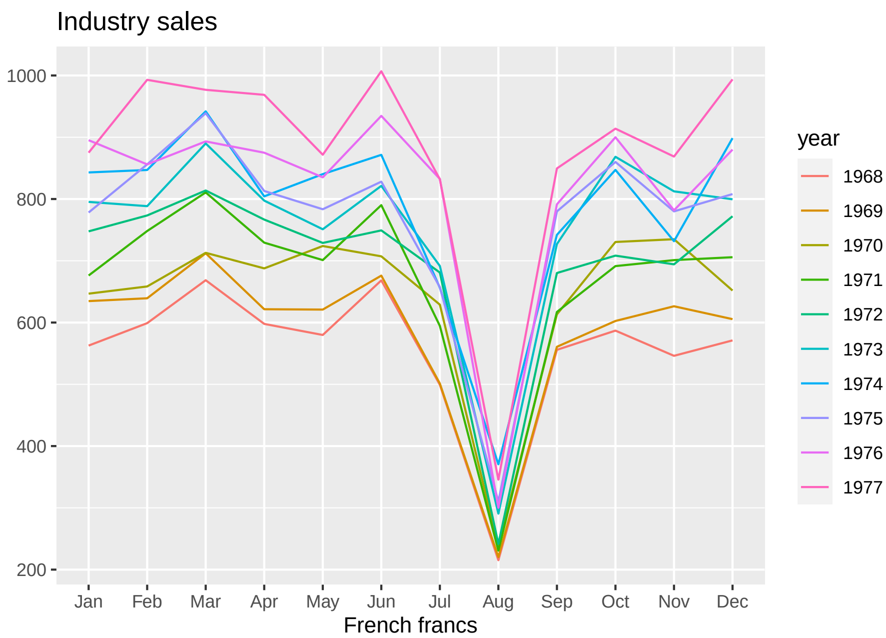


Figure 15: Writing seasonplot

In this plot, we can appreciate that each year, values keep getting higher, so there is a positive trend. Furthermore, there is always a big drop in August, possibly related to the vacation period of workers.

```
ggsubseriesplot(writing,
                main = "Industry sales",
                xlab = "French francs")
```

We can see the same information in the subseries plot.

```
autoplot(diff(writing),
         main = "Industry sales",
         xlab = "First differences")
```

We can see in the first differences series that variability also grows as time passes, hence we have heterocedasticity.

The correlogram:

```
ggAcf(writing) +
    ggtitle("ACF Industry sales") +
    xlab("French francs")
```

Peaks at 12 and 24 confirm the seasonality.

Exercise 2

The following time plots and ACF plots correspond to four different time series. Your task is to match each time plot in the first row with one of the ACF plots in the second row.

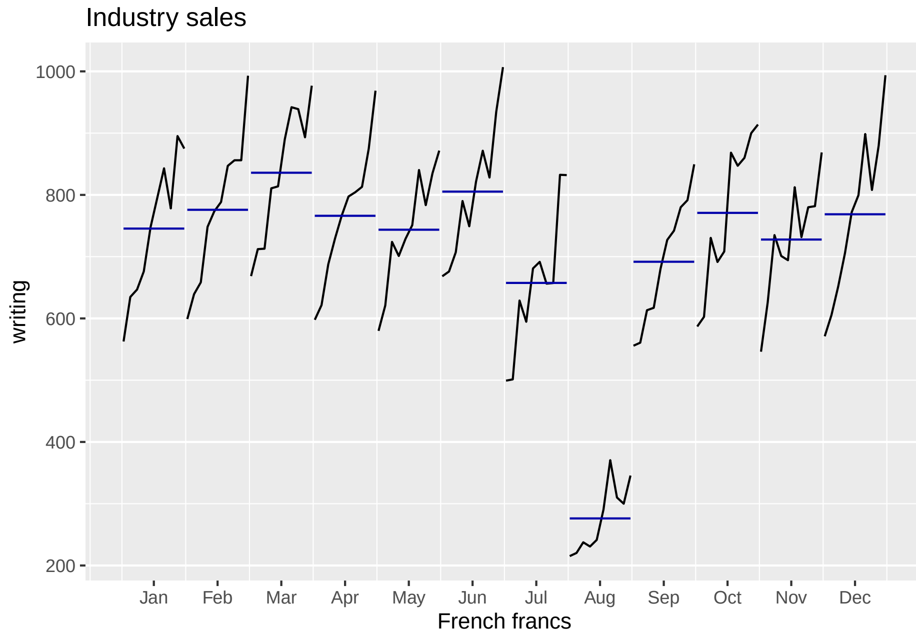


Figure 16: Writing subseries plot

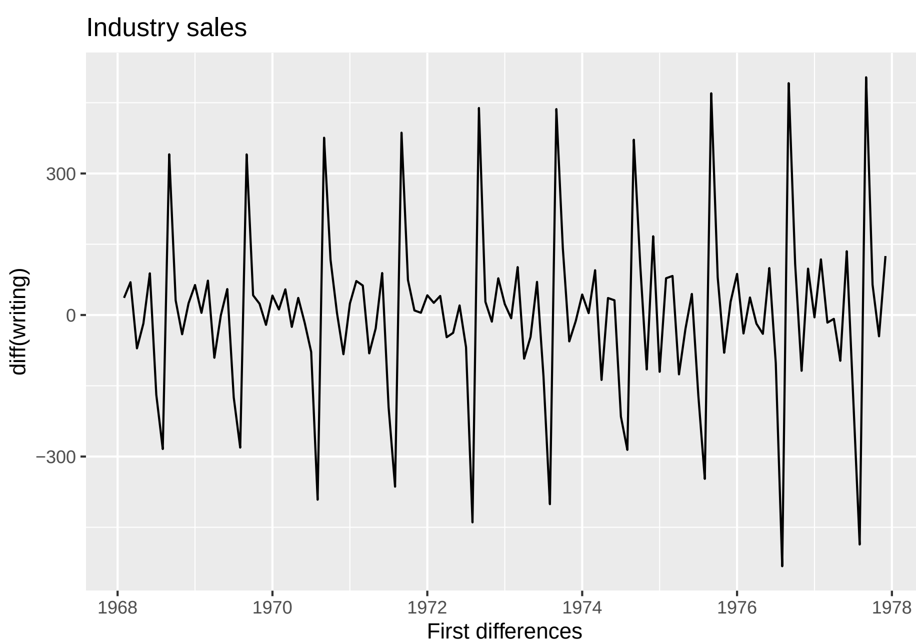


Figure 17: Writing first differences

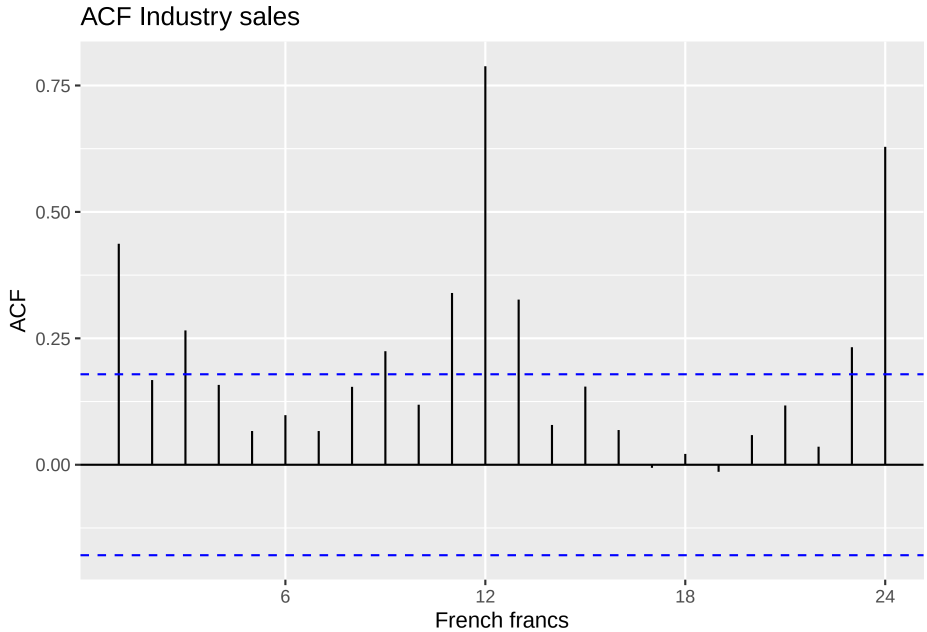


Figure 18: ACF writing

We will approach this exercise by first giving the chosen match and then justifying it:

- `1 - C`. The first time series exhibits seasonal pattern which does not appear to have a predominant period. This is why we are looking at a corrplot which shows fluctuations in the magnitudes of the peaks with yet a repeatable pattern. This is precisely what may be found in `C`.
- `2 - B`. This one exhibits a slight positive tendency. Nonetheless, the main property of this plot is its heavy seasonal peaks which should in turn correspond to the heaviest peaks on the ACF plots. This is why we select `B` as the match, having the most prominent peaks with a period of 12 months.
- `3 - D`. Once again we have a a slight positive tendency with prominent seasonal peaks. Nonetheless, this time the tendency appears to be flatter and the peaks are less strong overall with respect to `2`. As such, we select `D` which haves faster decreasing aurocorrelations than `C` while keeping a clear 12 months seasonality.
- `4 - A`. For the last one, we see tendency on the time series and an lack of any kind of seasonality. Hence the lag plot should follow a slowly decrease without periodic peaks, just what `A` shows.

Exercise 3

For each of the following series, make a graph of the data with forecasts using the most appropriate of the four benchmark methods: mean, naive, seasonal naive or drift.

(a) Monthly total of people on unemployed benefits in Australia (January 1956 - July 1992). Data set **dole**.

(b) Annual Canadian lynx trappings (1821 - 1934). Data set **lynx**.

In each case, do you think the forecasts are reasonable? If not, how could they be improved?

(a)

We are going to start by plotting the data, using time, seasonal and subseries plots.

```
autoplot(dole, main = "Time plot: Monthly total of people on
         unemployment benefits in Australia",
        xlab = "Time",
        ylab = "Number of people") + theme_bw()
```

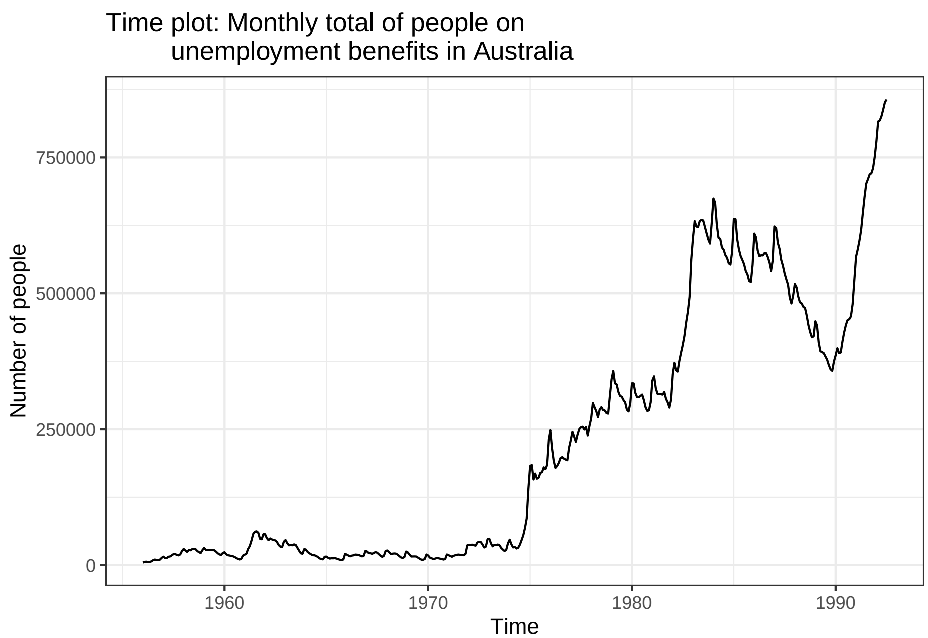


Figure 19: Time plot for the monthly total of people on unemployment benefits in Australia.

```
dole_seasonal  <- ggseasonplot(dole,
        main = "Seasonal plot: Monthly total of people\non
        unemployment benefits in Australia",
        xlab = "Time",
        ylab = "Number of people") + theme_bw()
dole_pseasonal <- ggseasonplot(dole,
        main = "Seasonal plot (polar): Monthly total
        of people \non unemployment benefits in Australia",
        xlab = "Time",
        ylab = "Number of people",
        polar = T) + theme_bw()
```

```
plot(dole_seasonal)
```

```
plot(dole_pseasonal)
```

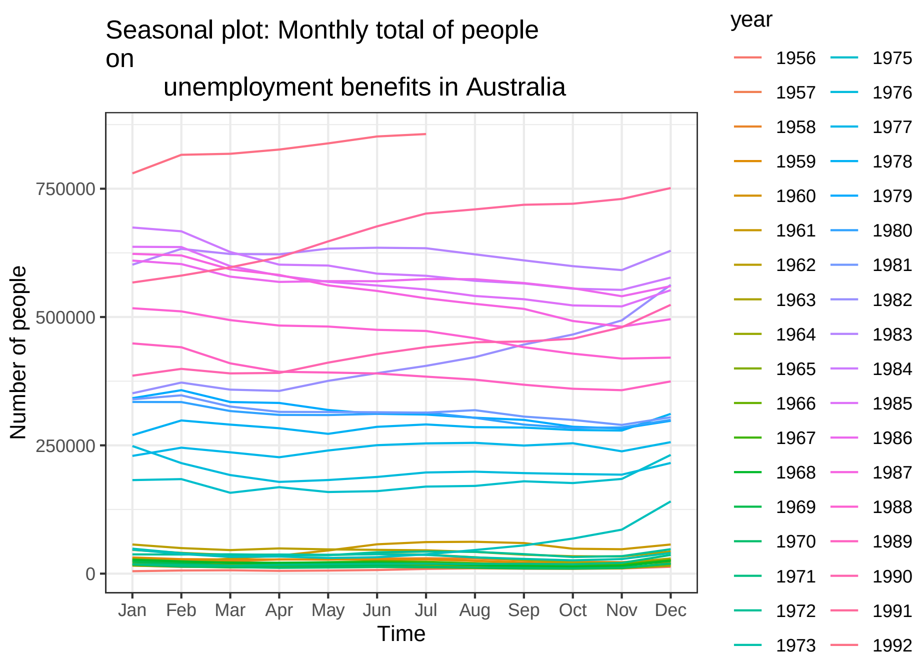


Figure 20: Seasonal plot of monthly total of people on unemployment benefits in Australia

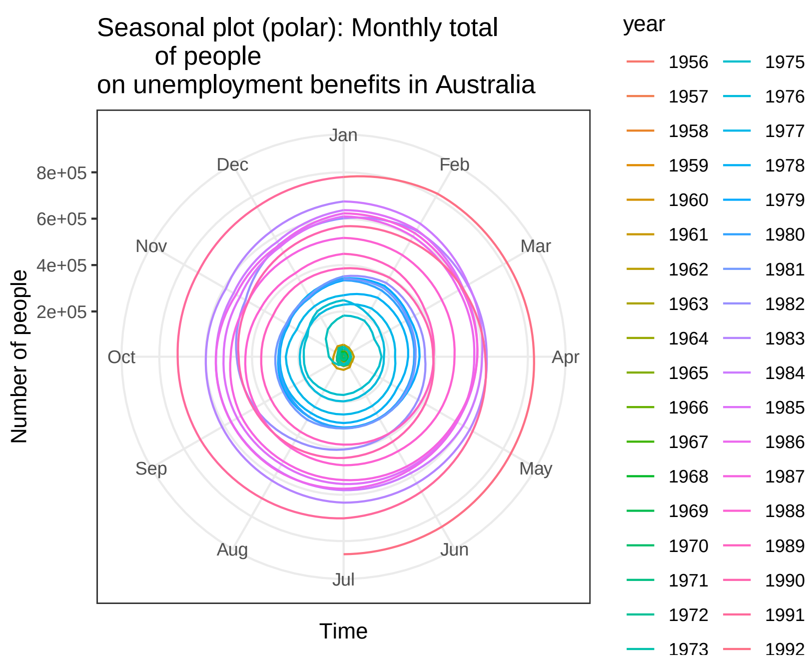


Figure 21: Seasonal polar plot of monthly total of people on unemployment benefits in Australia

```r
plot(ggsubseriesplot(dole,
        main = "Subseries plot: Monthly total of people
        on unemployment benefits in Australia",
        xlab = "Month",
        ylab = "Number of people",) + theme_bw())
```

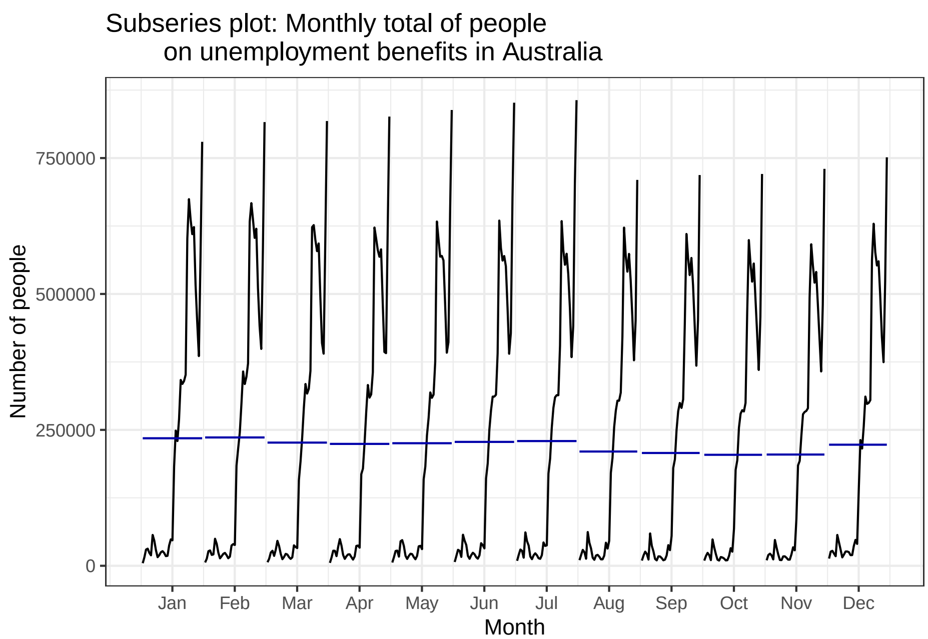


Figure 22: Subseries plot for the monthly total of people on unemployment benefits in Australia

We can observe a clearly positive trend over time, as well as an apparent lack of seasonality. Also, in the subseries, we do not see such high peaks in the months from August to December, unlike the rest of the months. However, this is because we only have data up to July 1992, which seems to coincide with a period of significant rise in people on unemployment benefits.

It may also be worth noting that there is usually an increase in the month of December. This can also be seen in the plot subseries, where, although the month of December does not have the data corresponding to the last year (which probably increased its average value), it has an average value similar to those of the months from January to July, which do have the data for the last year. If we remove the data from the last year, it is more clear.

```r
plot(ggsubseriesplot(window(dole, 1956, c(1991,12)), # not selecting the last year
        main = "Subseries plot: Monthly total
        of people on unemployment benefits in Australia",
        xlab = "Month",
        ylab = "Number of people",) + theme_bw())
```

Next, we will look the ACF:

```r
ggAcf(dole) + ggtitle("ACF of unemployment benefits
                      in Australia") +  theme_bw()
```

The ACF confirms the positive trend as, when data have a trend, the autocorrelations for small lags tend to be large and positive (due to observations nearby in time are also nearby in size). Also, the ACF of trended time series tend to have positive values that slowly decrease as the lags increase.

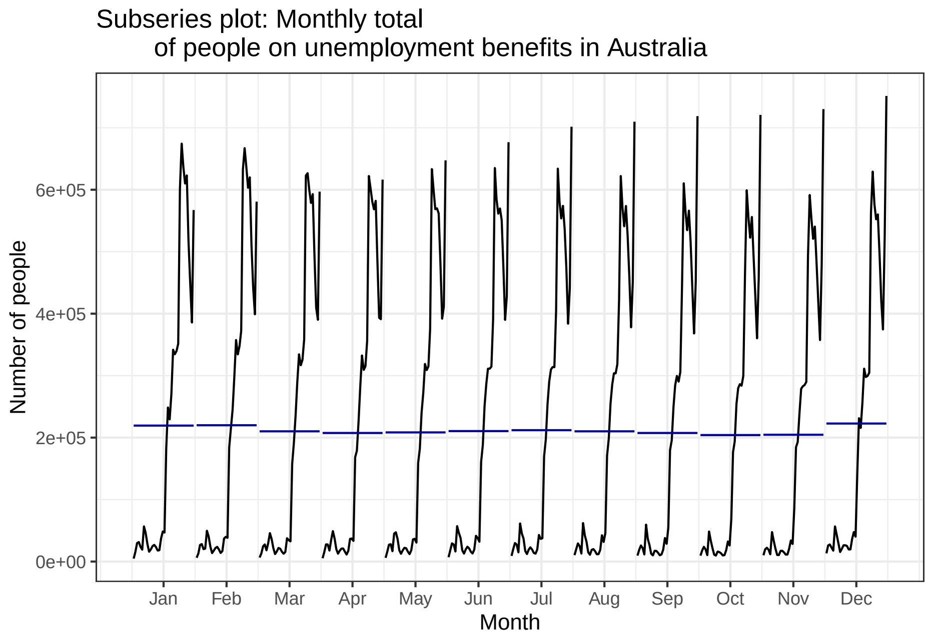


Figure 23: Subseries plot for the monthly total of people on unemployment benefits in Australia (from 1956 to the end of 1991)

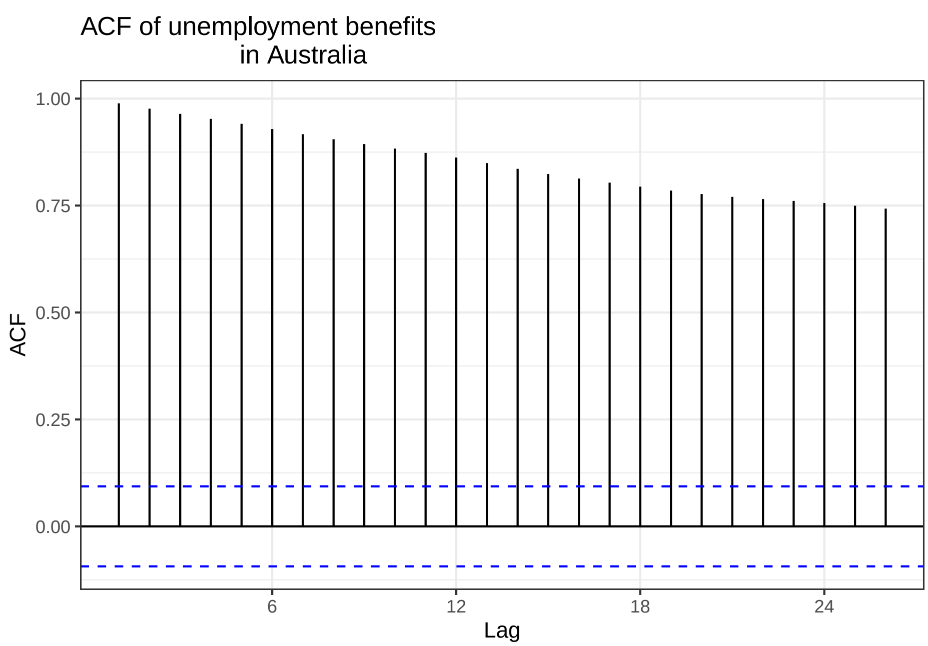


Figure 24: ACF of unemployment benefits in Australia

Based on this plot, we may think that the most appropiate methods will be the naive methods.

Now, we are going to do the predictions for the next 24 months using the four methods:

```
dole_mean   <- meanf(dole,h=24)
dole_naive  <- naive(dole,h=24)
dole_snaive <- snaive(dole,h=24)
dole_drift  <- rwf(dole,h=24,drift=TRUE)

plot(dole_mean, PI=FALSE, main="Forecasts for monthly total of people
     on unemployment benefits in Australia")
lines(dole_naive$mean, col=2)
lines(dole_snaive$mean, col=3)
lines(dole_drift$mean, col=5)
legend("topleft", lty=1, cex = .8, col=c(4,2,3,5),
       legend=c("Mean method","Naive method",
                "Seasonal naive method","Drift method"))
```

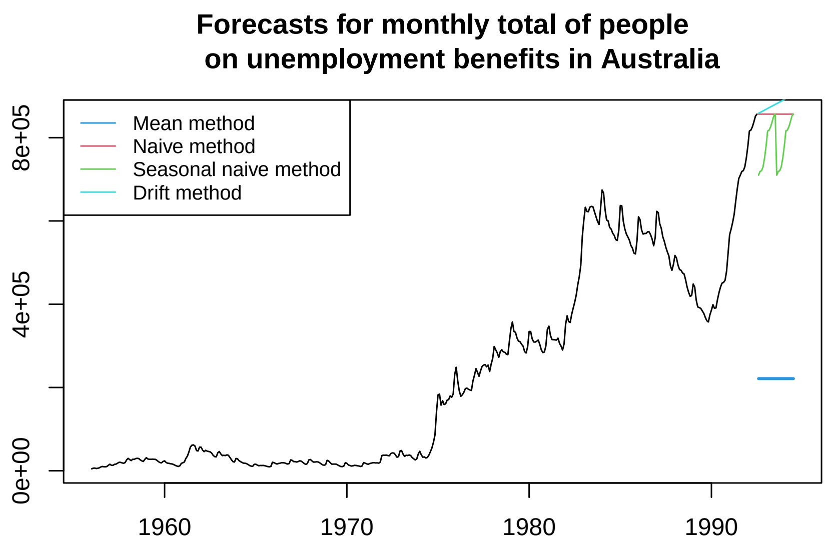


Figure 25: Forecast for monthly total of people on unemployment benefits in Australia

At first glance, none of the methods seem particularly good, with the mean method being perhaps the worst. Based on the graph, it seems unlikely that any of the predictions will be fulfilled, with the drift method being the one that seems closest to what could happen.

We now check the residuals.

```
summary(residuals(dole_mean))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -216496 -200838 -164483       0  171383  635267
```

```r
checkresiduals(dole_mean)
```

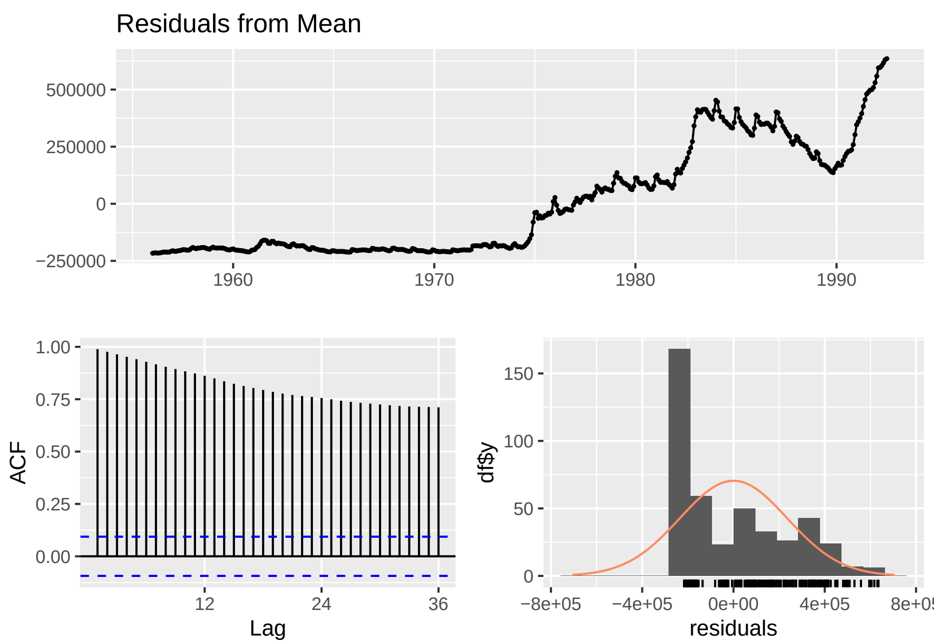


Figure 26: Residuals obtained with the mean method in the dole dataset

```
##
##  Ljung-Box test
##
## data:  Residuals from Mean
## Q* = 8081.1, df = 23, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 24
```

```r
summary(residuals(dole_naive))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -40406.0  -3115.2   -422.5   1944.7   4315.8  69146.0        1
```

```r
checkresiduals(dole_naive)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Naive method
## Q* = 725.9, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

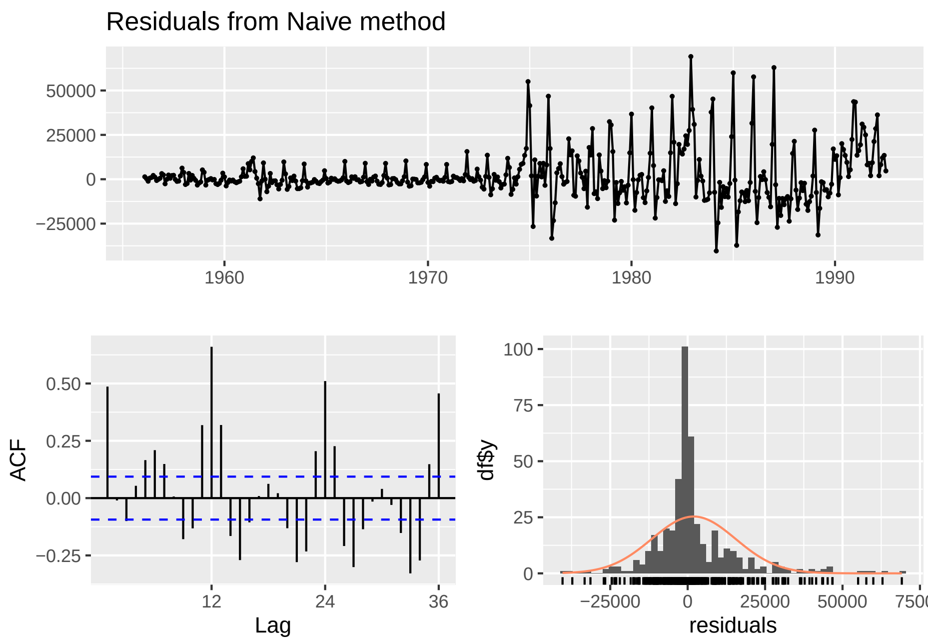


Figure 27: Residuals obtained with the naive method in the dole dataset

```
summary(residuals(dole_snaive))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
## -108955   -9246    3751   21827   28572  266444      12
```

```
checkresiduals(dole_snaive)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Seasonal naive method
## Q* = 2733.7, df = 24, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 24
```

```
summary(residuals(dole_drift))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##  -42351   -5060   -2367       0    2371   67201       1
```

```
checkresiduals(dole_drift)
```

```
##
##  Ljung-Box test
##
```

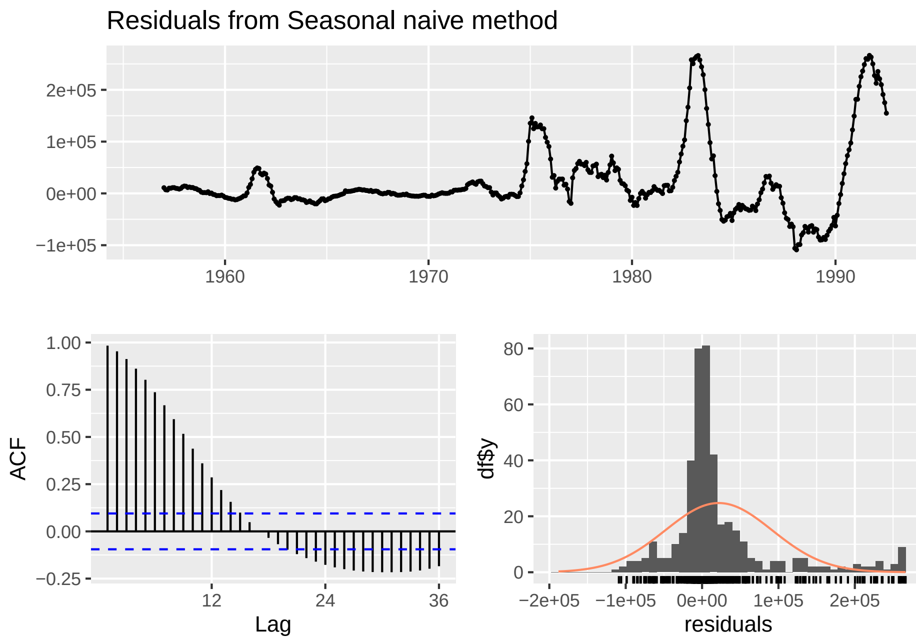


Figure 28: Residuals obtained with the seasonal naive method in the dole dataset

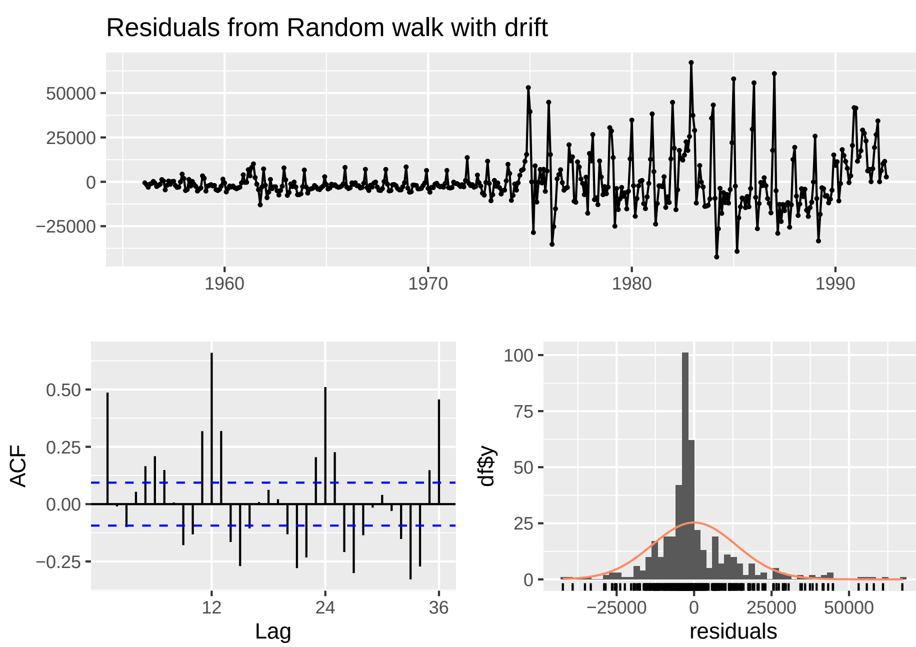


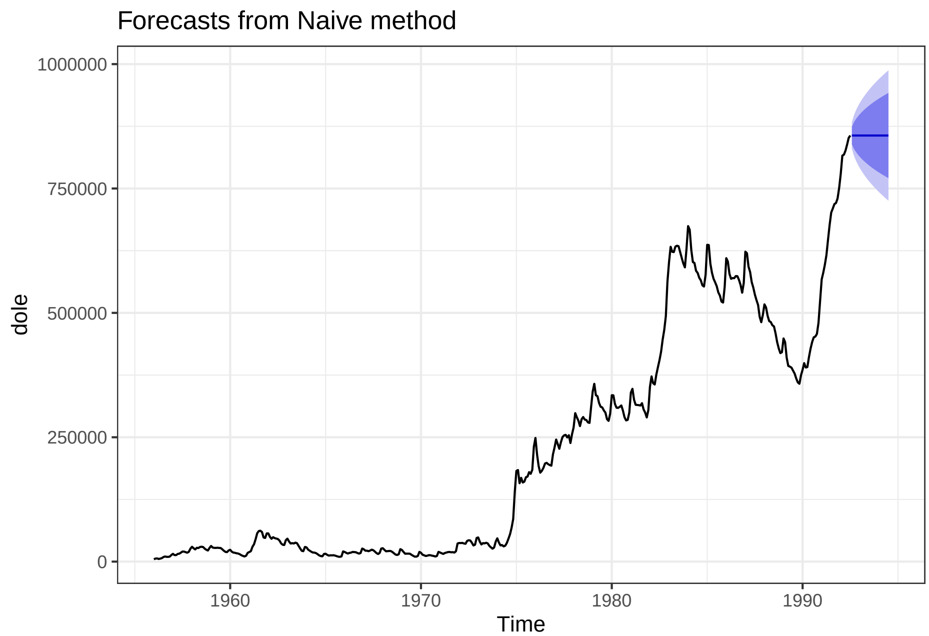
Figure 29: Residuals obtained with the drift method in the dole dataset

```
## data:  Residuals from Random walk with drift
## Q* = 725.9, df = 23, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 24
```

Again, none of the methods seems particularly correct, the naive seems the most correct among them. Now, we will do what the exercise asks for (*"graph with forecasts using the most appropriate of the four benchmark methods"*):

```
autoplot(dole_naive) + theme_bw()
```

\begin{figure}



{

}

\caption{Forecasts obtained from the naive method in the dole dataset with confidence intervals (80% and 95%).} \end{figure}

It can be seen that the confidence intervals are wide.

Forecasts do not seem reasonable for this data set using these methods. To improve them, we see two possible options:

1. Transform the data. We must take into account that Australia's population has grown from 11.4M inhabitants to 17.5M (according to the data provided by google). This is something to take into account and it would be good to make better comparisons. In addition, there seems to be heterocedasticity in the data, so a mathematical transformation would be convenient.
2. Use more complex models. The models we have used to make predictions are really simple, and we believe that a more complex model would yield better results.

(b)

We start again plotting the data. As the data is not seasonal, we will only use the time plot and we wont use any seasonal method.

```
autoplot(lynx, main = "Time plot: Annual Canadian Lynx trappings",
        xlab = "Time",
        ylab = "Number of lynx") + theme_bw()
```

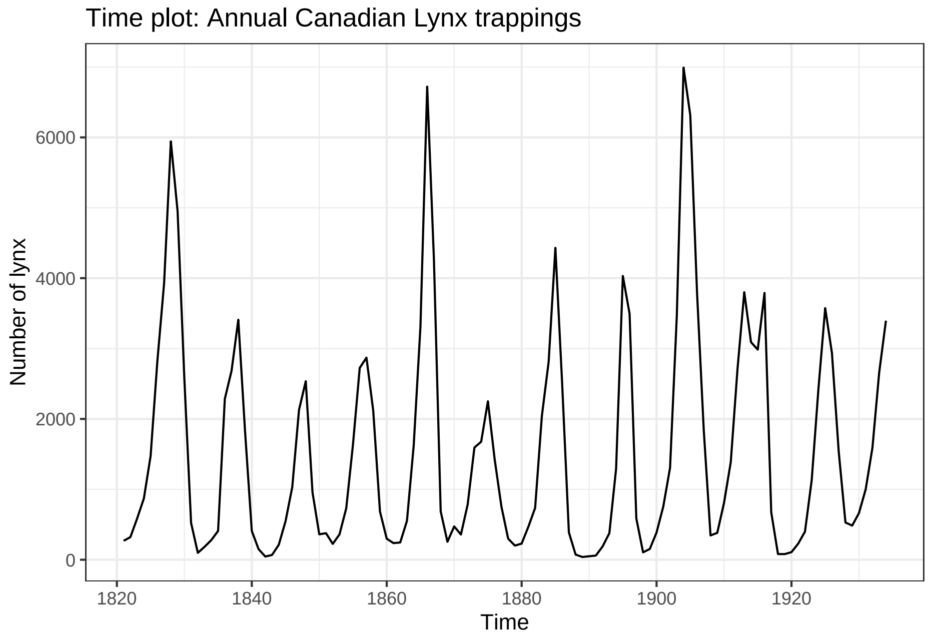


Figure 30: Time plot for the Annual Canadian Lynx trappings.

As stated in Lab1b_Transformation (seem in class), this time series is stationary, although it may not appear so due to cycles, which occur when there are too many lynx for the available feed.

Next, we compute forecast with mean, naive and drift methods and plot them all

```
lynx_mean   <- meanf(lynx,h=24)
lynx_naive  <- naive(lynx,h=24)
lynx_drift  <- rwf(lynx,h=24,drift=TRUE)

plot(lynx_mean, PI=FALSE, main="Forecasts for
     Annual Canadian Lynx trappings",
    xlab = "Time",
    ylab = "Number of lynx")
lines(lynx_naive$mean, col=2)
lines(lynx_drift$mean, col=5)
legend("topleft", lty=1, cex = .7, col=c(4,2,5),
       legend=c("Mean method","Naive method","Drift method"))
```

None of the three methods seems to be doing great. Based on the above data and looking at the cycles, we would expect either a continuation of the rise or a fairly steep decline, but none of the methods make forecasts that indicate that. Let's check the residuals.

```
summary(residuals(lynx_mean))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   -1499   -1190    -767       0    1029    5453
```

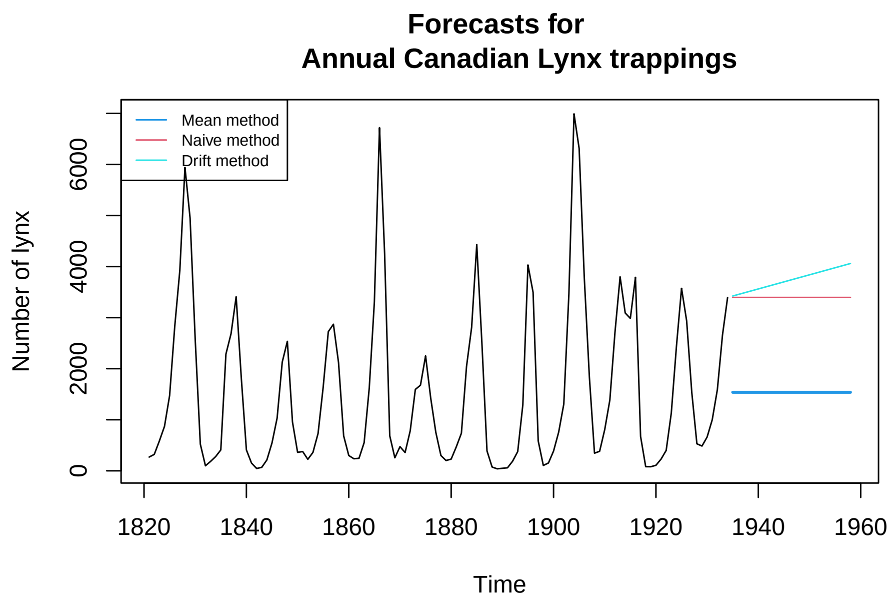


Figure 31: Forecasts for the Annual Canadian Lynx trappings with the mean, naive and drift methods.

```
checkresiduals(lynx_mean)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Mean
## Q* = 215.45, df = 9, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 10
```

```
summary(residuals(lynx_naive))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -3567.00  -457.00   121.00    27.67   590.00  3526.00        1
```

```
checkresiduals(lynx_naive)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Naive method
## Q* = 113.73, df = 10, p-value < 2.2e-16
##
## Model df: 0.   Total lags used: 10
```

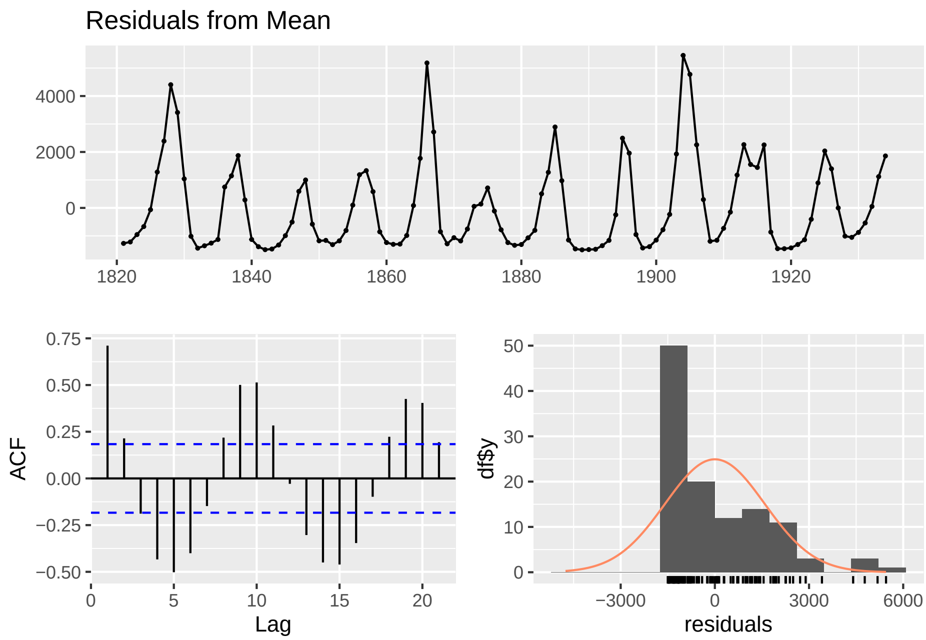


Figure 32: Residuals obtained with the mean method in the lynx dataset.

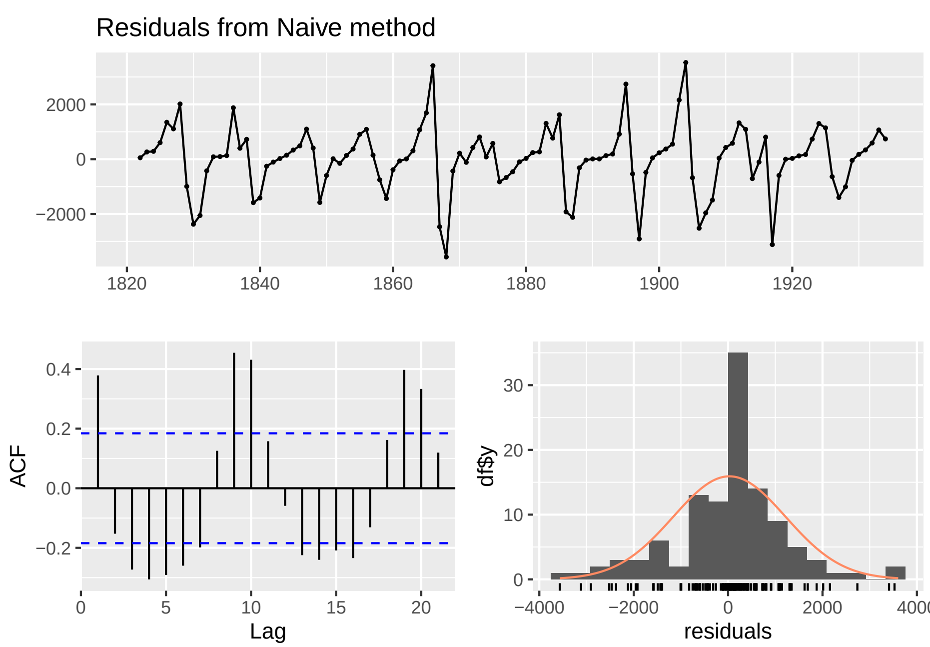


Figure 33: Residuals obtained with the naive method in the lynx dataset.

```
summary(residuals(lynx_drift))
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -3594.67  -484.67    93.33     0.00   562.33  3498.33        1
```

```
checkresiduals(lynx_drift)
```

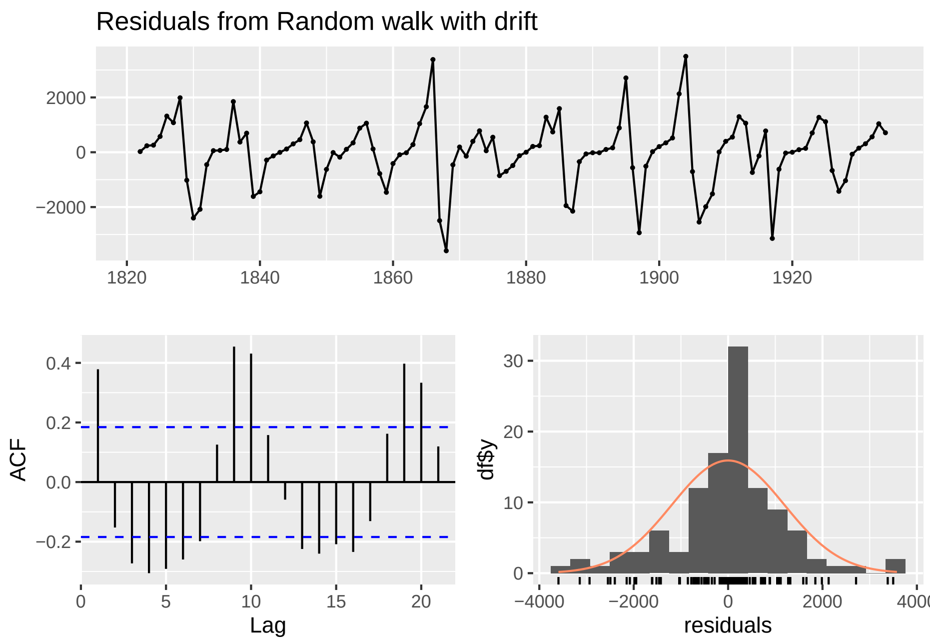


Figure 34: Residuals obtained with the drift method in the lynx dataset.

```
##
##  Ljung-Box test
##
## data:  Residuals from Random walk with drift
## Q* = 113.73, df = 9, p-value < 2.2e-16
##
## Model df: 1.   Total lags used: 10
```

The residuals from none of the methods look remotely good. While the exercise statement calls for *"graph with forecasts using the most appropriate of the four benchmark methods"*, none of them seem remotely appropriate for these data, so we will not graph any of them individually.

Regarding the way to improve the forecast, using more complex methods is the way to go. None of the simple methods used is able to capture the particularities of this time series.

Exercise 4

Consider the daily IBM stock prices (data set ibmclose).

(a) Produce some plots of the data in order to become familiar with it.

(b) Split the data into a training set of 300 observations and a test set of 69 observations.

(c) Try various benchmark methods to forecast the training set and compare the results on the test set. Which method did best?

(d) For the best method, compute the residuals and plot them. What do the plots tell you?

```
#  Daily IBM stock prices

ibmclose <- window(ibmclose)
head(ibmclose, 24)
```

```
## Time Series:
## Start = 1
## End = 24
## Frequency = 1
##  [1] 460 457 452 459 462 459 463 479 493 490 492 498 499 497 496 490 489 478 487
## [20] 491 487 482 479 478
```

(a)

The data set contains the daily IBM stock's prices at closure time.

```
ibmclose %>%
autoplot() + xlab("Time")+ ylab("IBM stock prices")+ ggtitle("Daily IBM stock prices")
```

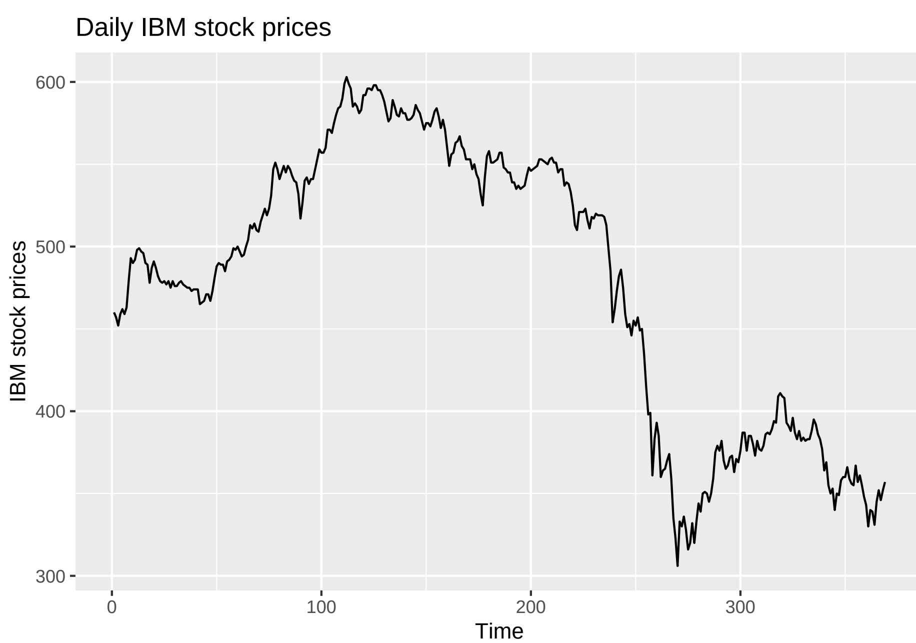


Figure 35: Daily IBM stock prices.

The following conclusions can be drawn from this first plot:

- There is not a clear trend: the series has some ups and downs, which makes the series unpredictable.

- There is a big drop in the prices right after the 200 day mark.
- Not seasonal pattern can be detected as the frequency of the data is daily and only for a year.
- The mean method is not going to be the best option due to the heterogeneity and the naïve method will give the most conservative forecast.

The autocorrelation plot:

```
ggAcf(ibmclose) + ggtitle("ACF of Daily IBM stock prices")
```

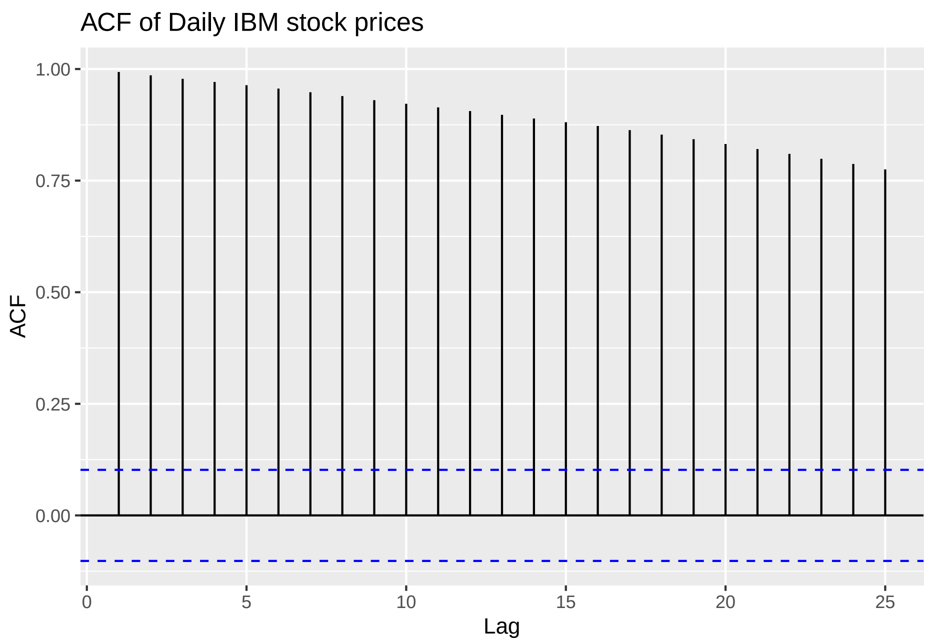


Figure 36: ACF of Daily IBM stock prices.

In the autocorrrelation plot we can conclude that the IBM stock prices are highly correlated with each other. That is to say, when the stock price rises, it tends to continue this way and when it falls, it keeps going downwards.

(b)

```
# Split
train <- subset(ibmclose, end = 300)
test  <- subset(ibmclose, start = 301, end = length(ibmclose))

# Plot of the split
plot(ibmclose, main = "Plot of the split")
lines(train,col="red")
lines(test, col="blue")
```

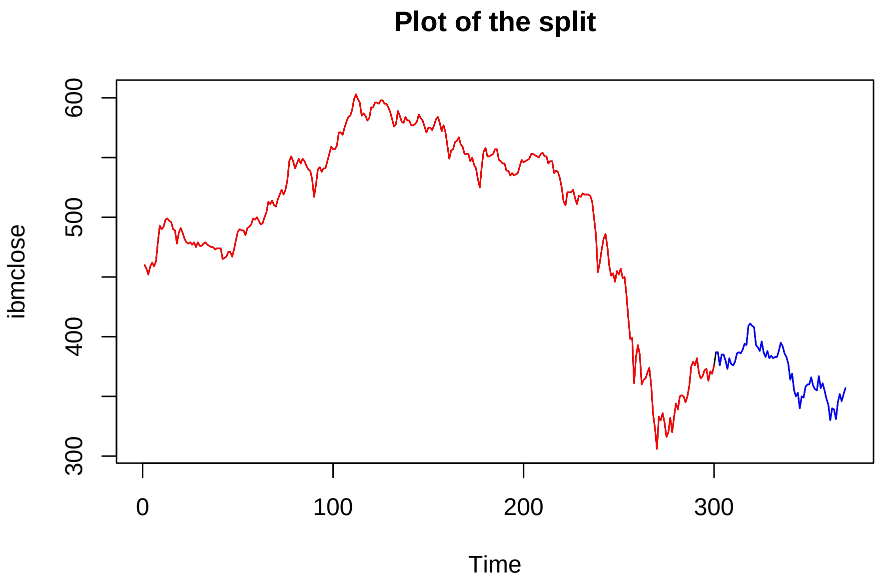


Figure 37: Plot of the split.

(c)

```
ibmclosefit1 <- meanf(train,h=69)
ibmclosefit2 <- rwf(train,h=69)
ibmclosefit3 <- rwf(train, drift=TRUE,h=69)

autoplot(train) +
  autolayer(ibmclosefit1, series="Mean", PI=FALSE) +
  autolayer(ibmclosefit2, series="Naïve", PI=FALSE) +
  autolayer(ibmclosefit3, series="Drift", PI=FALSE) +
  xlab("Day") + ylab("Prices") +
  ggtitle("Forecasts for IBM stock prices") +
  guides(colour=guide_legend(title="Forecast"))
```

The mean method seems to perform the worst out of the three methods, which is to expected as the series is not stationary. And, as it has been stated before, we will be sticking to the naïve method as it gives the most reasonable forecast.

Now, we will be comparing the results with the real data.

```
autoplot(window(ibmclose)) +
  autolayer(ibmclosefit1, series="Mean", PI=FALSE) +
  autolayer(ibmclosefit2, series="Naïve", PI=FALSE) +
  autolayer(ibmclosefit3, series="Drift", PI=FALSE) +
  xlab("Day") + ylab("Prices") +
  ggtitle("Forecasts for IBM stock prices") +
  guides(colour=guide_legend(title="Forecast"))
```

In order to evaluate the predictive performance, the **accuracy** been calculated based on the test set for each forecasting method.

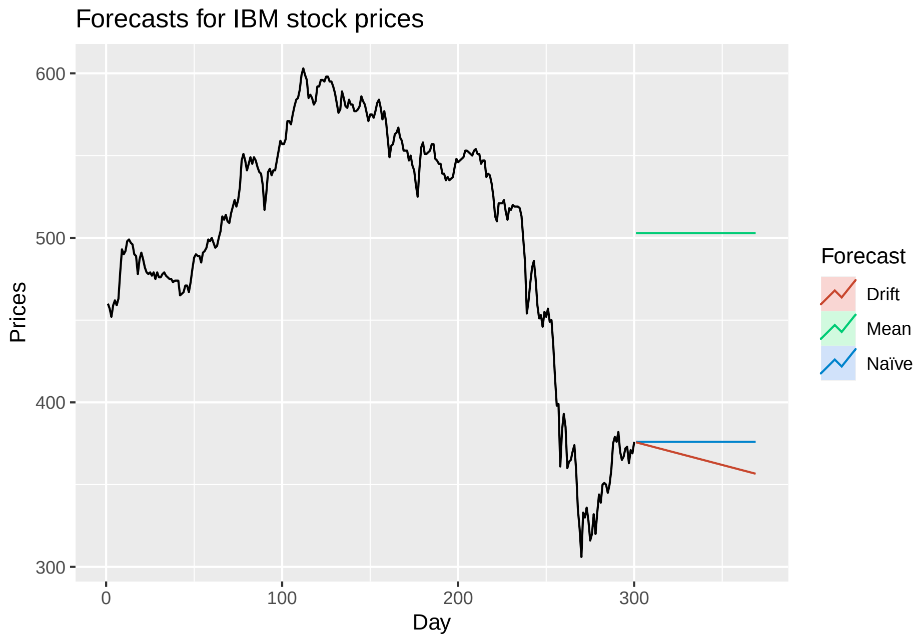


Figure 38: Forecasts for IBM stock prices with drift, mean and naïve methods.

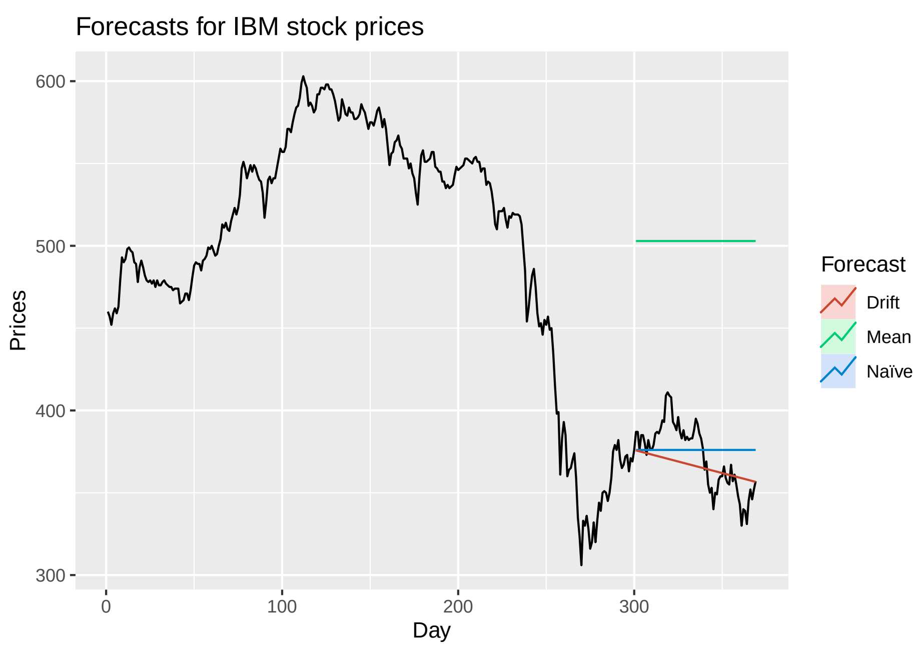


Figure 39: Comparison of the forecasts for IBM stock prices with drift, mean and naïve methods against the real data.

```r
forecast::accuracy(ibmclosefit1, test)
```

```
##                         ME      RMSE       MAE         MPE     MAPE     MASE
## Training set  1.660438e-14  73.61532  58.72231  -2.642058 13.03019 11.52098
## Test set     -1.306180e+02 132.12557 130.61797 -35.478819 35.47882 25.62649
##                   ACF1 Theil's U
## Training set 0.9895779        NA
## Test set     0.9314689  19.05515
```

```r
forecast::accuracy(ibmclosefit2, test)
```

```
##                      ME      RMSE      MAE         MPE     MAPE     MASE
## Training set -0.2809365  7.302815  5.09699 -0.08262872 1.115844 1.000000
## Test set     -3.7246377 20.248099 17.02899 -1.29391743 4.668186 3.340989
##                   ACF1 Theil's U
## Training set 0.1351052        NA
## Test set     0.9314689  2.973486
```

```r
forecast::accuracy(ibmclosefit3, test)
```

```
##                        ME      RMSE       MAE         MPE     MAPE     MASE
## Training set 2.870480e-14  7.297409  5.127996 -0.02530123 1.121650 1.006083
## Test set     6.108138e+00 17.066963 13.974747  1.41920066 3.707888 2.741765
##                   ACF1 Theil's U
## Training set 0.1351052        NA
## Test set     0.9045875  2.361092
```

The best results are obtained with the **drift method** for this test set, but the difference with the **naïve method** (the second best) is minimum. However, it cannot be assured that the data will follow the descent trend which is forecast by the drift method.

(d)

After the previous section, there are still be some doubts about which is the best method, the naïve or the drift. Therefore, we have computed the residuals for both methods.

```r
# Naïve
train_naive <- naive(train,h=69)
res_naive   <- residuals(train_naive)
summary(res_naive)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's 
## -38.0000  -4.0000   0.0000  -0.2809   4.0000  27.0000        1
```

```r
checkresiduals(train_naive)
```

```
## 
##  Ljung-Box test
```

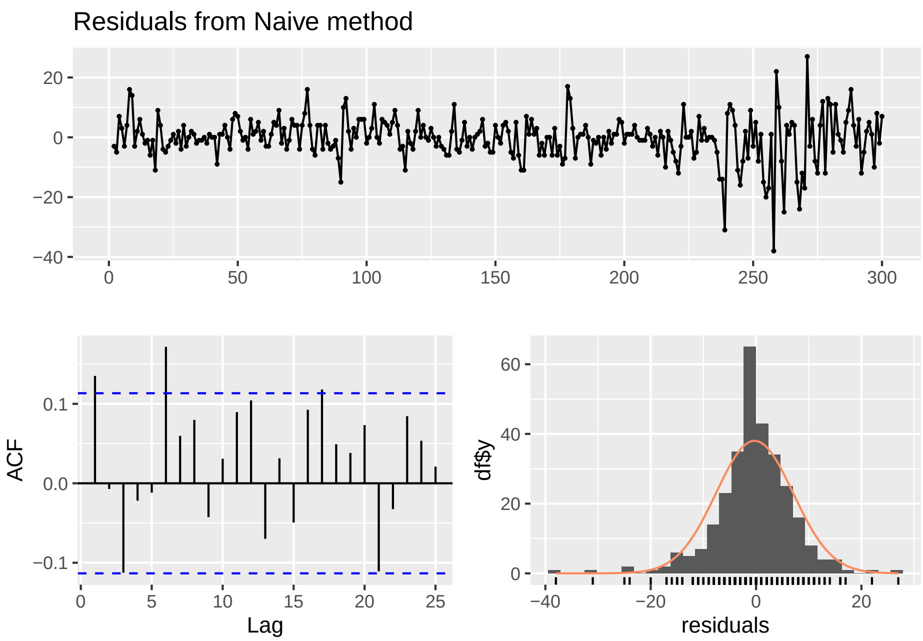


Figure 40: Residuals from the naive method.

```
##
## data:  Residuals from Naive method
## Q* = 22.555, df = 10, p-value = 0.01251
##
## Model df: 0.   Total lags used: 10
```

```
# Drift method
train_drift <- rwf(train, drift=TRUE,h=69)
res_drift   <- residuals(train_drift)
summary(res_drift)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.     NA's
## -37.7191  -3.7191   0.2809   0.0000   4.2809  27.2809        1
```

```
checkresiduals(train_drift)
```

```
##
##  Ljung-Box test
##
## data:  Residuals from Random walk with drift
## Q* = 22.555, df = 9, p-value = 0.007278
##
## Model df: 1.   Total lags used: 10
```

Although both methods have some lags which exceed the 95% confidence interval, the residual plots do not show any kind of pattern. Therefore, they could behave as white noise.

However, the p-values obtained in the Ljung-Box test for both methods are smaller than 0.05, so the null hypothesis can be rejected: we cannot conclude that the data are not independently distributed.

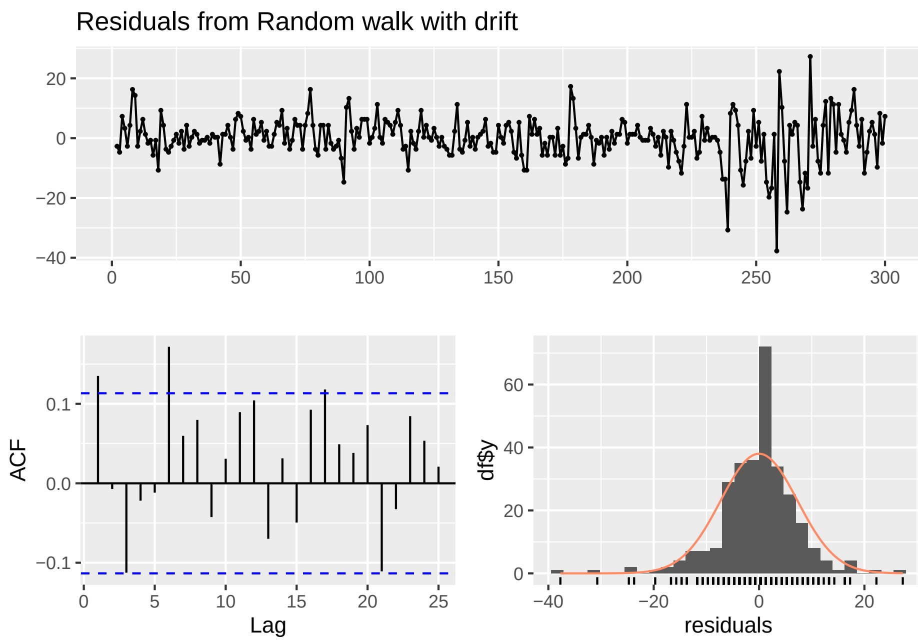


Figure 41: Residuals from Random walk with drift.

$$H_0 = \textit{The data are independently distributed}$$
$$H_1 = \textit{The data are not independently distributed}$$

Finally, looking at the residuals and the normal distribution, we can see that the residuals obtained with the **naïve method** are better adjusted to the gaussian distribution. Hence, the **naïve method** was the correct one in this case.

Exercise 5

The data below represent the monthly sales (in thousands) of product A for a plastics manufacturer for years 1 through 5 (data set plastics).

(a) Plot the time series of sales of product A. Can you identify any key feature? Explain what you see.

(b) Use an STL decomposition to calculate the trend-cycle and seasonal indices. (Experiment with having fixed or changing seasonality). Do you see any difference?

(c) Please, discuss whether the results support the graphical interpretation form part (a).

(d) Compute and plot the seasonally adjusted data.

(e) Use a random walk to produce forecasts of the seasonally adjusted data.

(f) Re-seasonilize the results to give forecasts on the original scale.

(a)

```
autoplot(plastics)
```

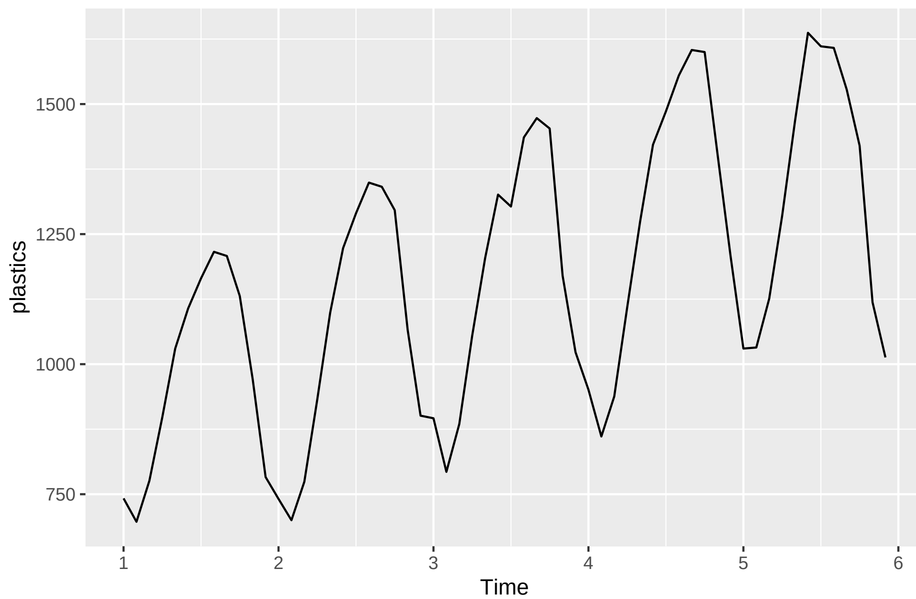


Figure 42: Plot for plastics

We can see a strong seasonality component, with an incremental production up to the summer, and then a decrease; and also a positive and linear trend. However, it seems that the decrease of the final cycle is greater than in the rest of them.

(b)

With fixed seasonality and a window for the trend of 5:

```
fit <- stl(plastics, t.window = 5, s.window = "periodic", robust = T)
plot(fit)
```

With changing seasonality (s.window = 13):

```
fit2 <- stl(plastics, t.window = 5, s.window = 13, robust = T)
plot(fit2)
```

There are no changes in the seasonality. This must be caused due to the fact that seasonality is indeed fixed. With regard to the trend, as the series ends in the lower part of the cycle, the first estimation uses a greater window than the second and thus it is smoother and with a final negative trend.

(c)

We can see a strong seasonality component and a positive trend. As explained before, both decompositions have a trend with a negative slope at the end, but this is due to the abrupt end of the series and should not be considered true.

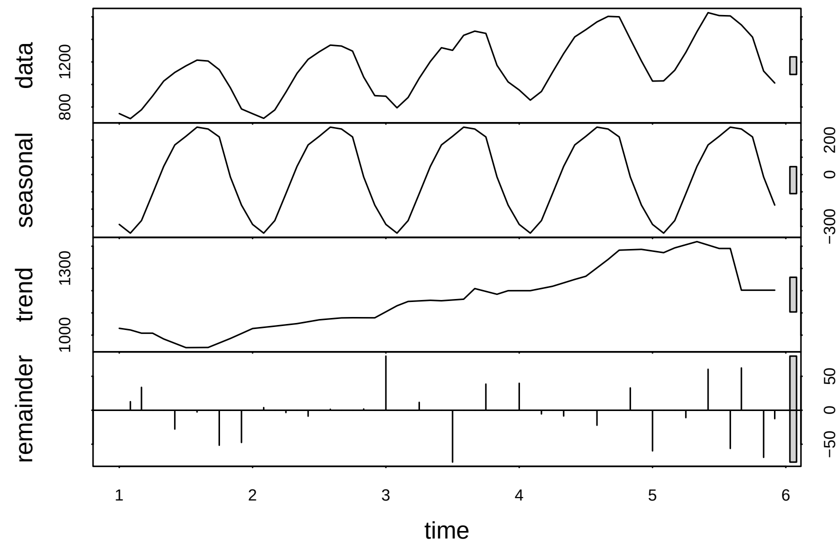


Figure 43: Plastics STL decomposition

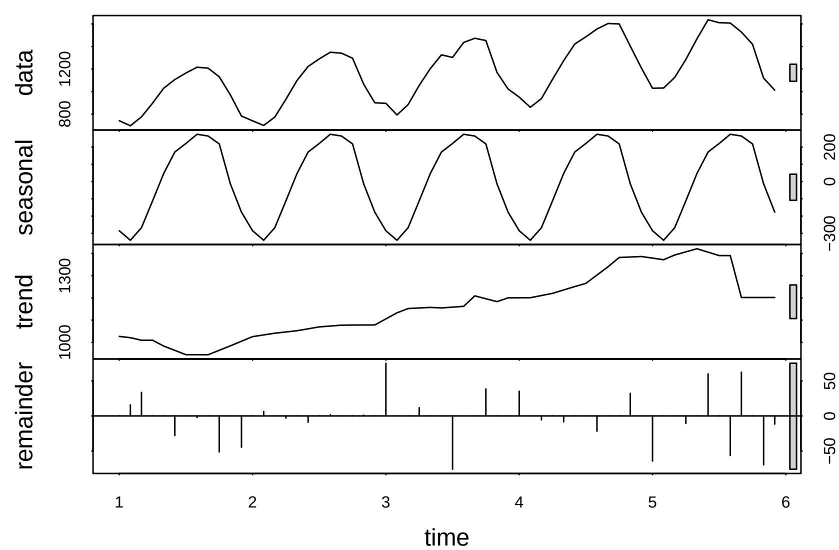
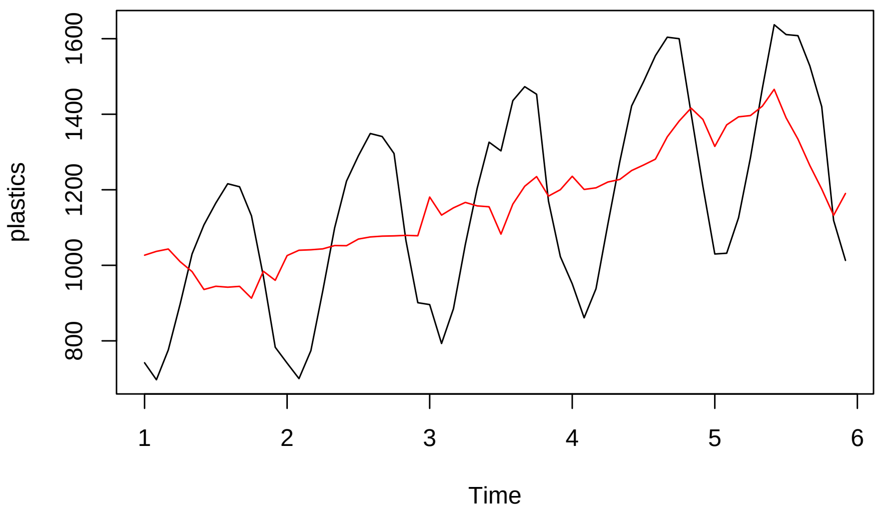


Figure 44: Plastics STL decomposition 2

(d)

```
seadj <- seasadj(fit2)
plot(plastics)
lines(seadj, col = "red")
```



We can see that the data has been successfully deseasonalized.

(e)

```
autoplot(seadj) +
    autolayer(rwf(seadj), PI = TRUE) +
    xlab("Time") + ylab("Plastics") + ggtitle("rfw() forecasting") + theme_bw()
```

(f)

The forecast on a decomposed time series, we forecast the seasonal component and the seasonally adjusted component separately.

```
fcast <- forecast::forecast(fit2, method = "naive")
plot(fcast)
```

Exercise 6

Use the monthly Australian short-term overseas visitors data, May 1985-April 2005 (Data set visitors).

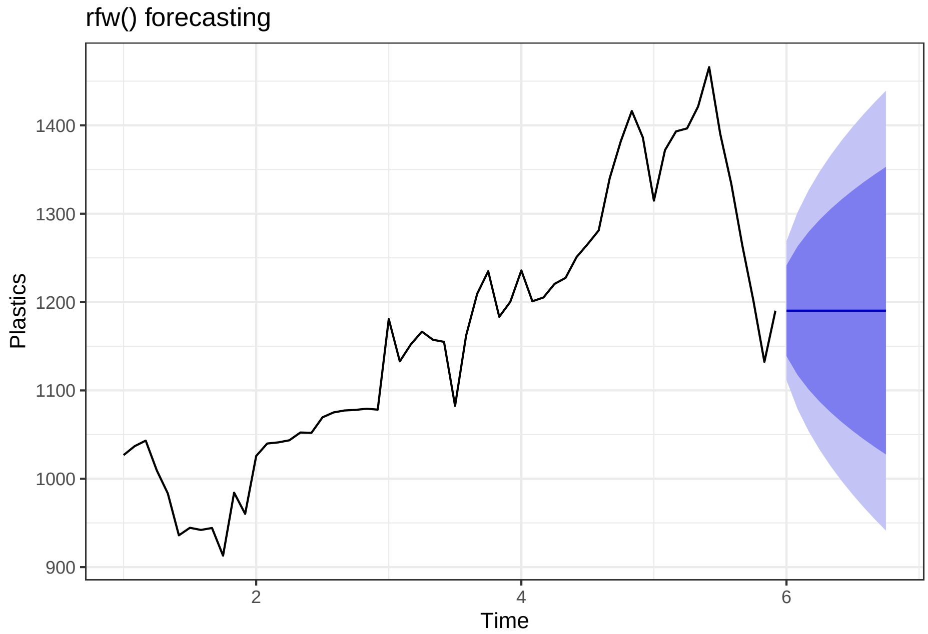


Figure 45: Forecast with seasonally adjusted data

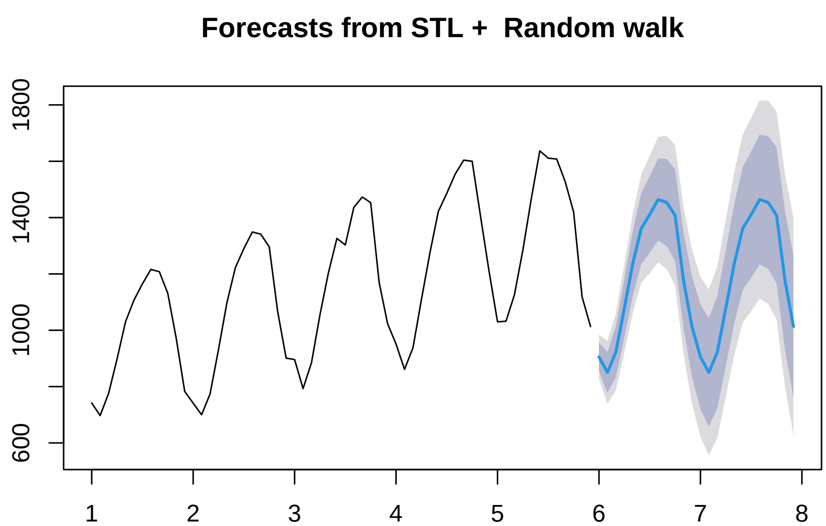


Figure 46: Forecast on the original scale

(a) With the help of the appropriate graphical representation, please describe the main features of the series.

(b) Forecast the next two years using Holt-Winters method according to the features you found in the previous point. Justify your choice.

(c) Experiment with making the trend exponential and/or damped. Do you see any difference? Justify your answer.

(d) Now fit each of the following models to the same data:

(1) an ETS model
(2) an additive ETS model applied to a Box-Cox transformed series
(3) an STL decomposition applied to the Box-Cox transformed data followed by an ETS model applied to the seasonally adjusted (transformed) data. Plot all the forecasts together.

(e) For each model, look at the residual diagnostics and compare the forecasts for the next two years. Which do you prefer?

We take the data `visitors` from the `fpp2` package. It exhibits the following form:

```
head(visitors, 5 * 12)
```

```
##        Jan   Feb   Mar   Apr   May   Jun   Jul   Aug   Sep   Oct   Nov   Dec
## 1985                            75.7  75.4  83.1  82.9  77.3 105.7 121.9 150.0
## 1986  98.0 118.0 129.5 110.6  91.7  94.8 109.5 105.1  95.0 130.3 156.7 190.1
## 1987 139.7 147.8 145.2 132.7 120.7 116.5 142.0 140.4 128.0 165.7 183.1 222.8
## 1988 161.3 180.4 185.2 160.5 157.1 163.8 203.3 196.9 179.6 207.3 208.0 245.8
## 1989 168.9 191.1 180.0 160.1 136.6 142.7 175.4 161.4 149.9 174.1 192.7 247.4
## 1990 176.2 192.8 189.1 181.1
```

```
autoplot(visitors, ylab = "visitors", xlab = "Year",
         main="Australian short-term overseas visitors") + theme_bw()
```

(a)

We note a strong seasonality component with no apparent cyclic behavior which can be more prominently shown in a seasonal plot or a polar plot. In it we note that there is a yearly period with a clear anual pattern. There seems to be also a positive tendency, apparently linear.

```
ggseasonplot(visitors, ylab = "visitors", xlab = "Year",
             main = "Seasonal plot") + theme_bw()
```

```
ggseasonplot(visitors, ylab = "visitors", xlab = "Year",
             polar = TRUE, main = "Polar plot") + theme_bw()
```

Moreover, in the corresponding ACF plot we observe a slowly decreasing autocorrelation, hinting tendency, combined with periodicity in the peaks, signing mark of seasonality with a 12 month period.

```
ggAcf(visitors) + ggtitle("ACF plot") + theme_bw()
```

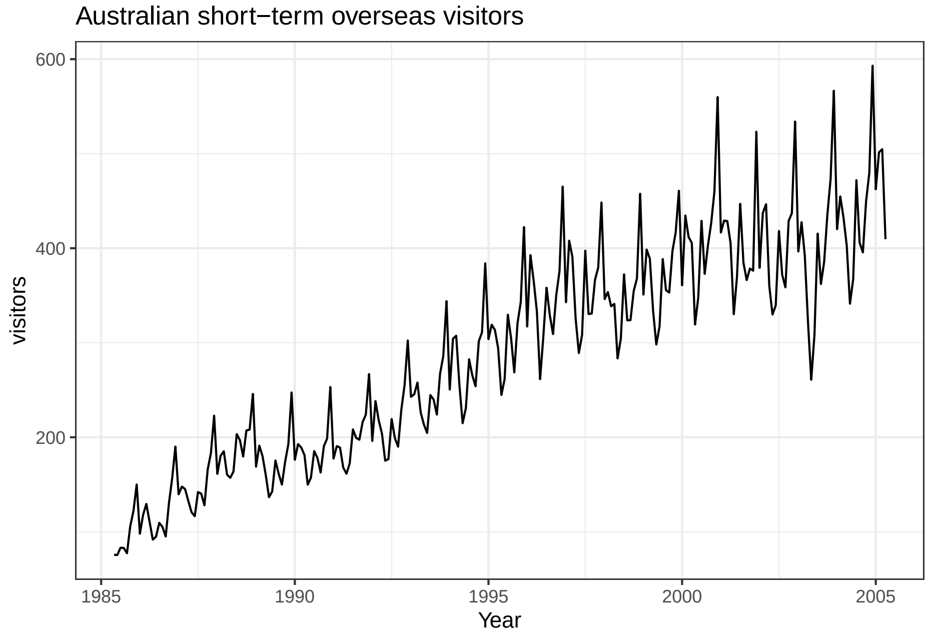


Figure 47: Time series plot of the ‘visitors‘ dataset.

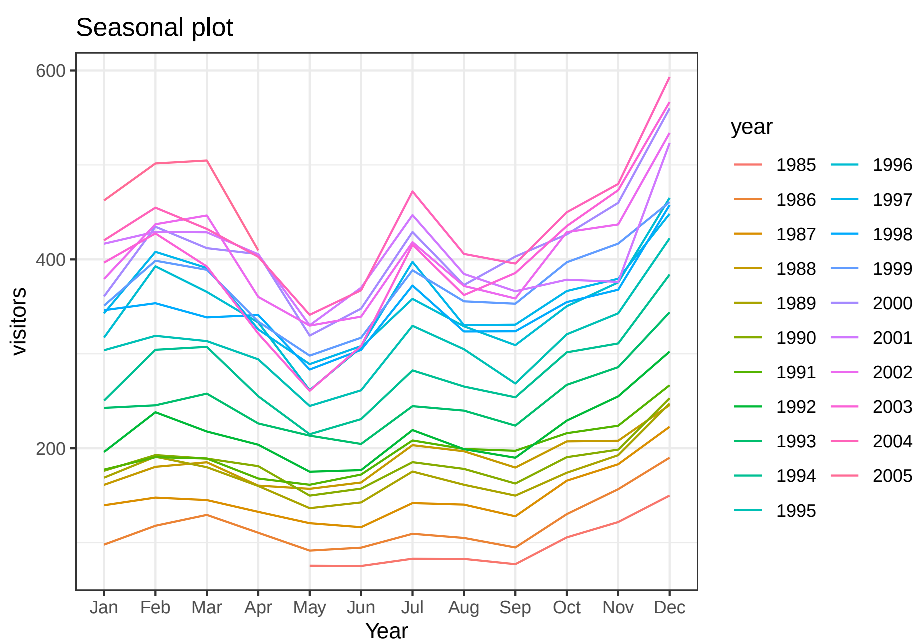


Figure 48: Seasonal plot for the ‘visitors‘ dataset.

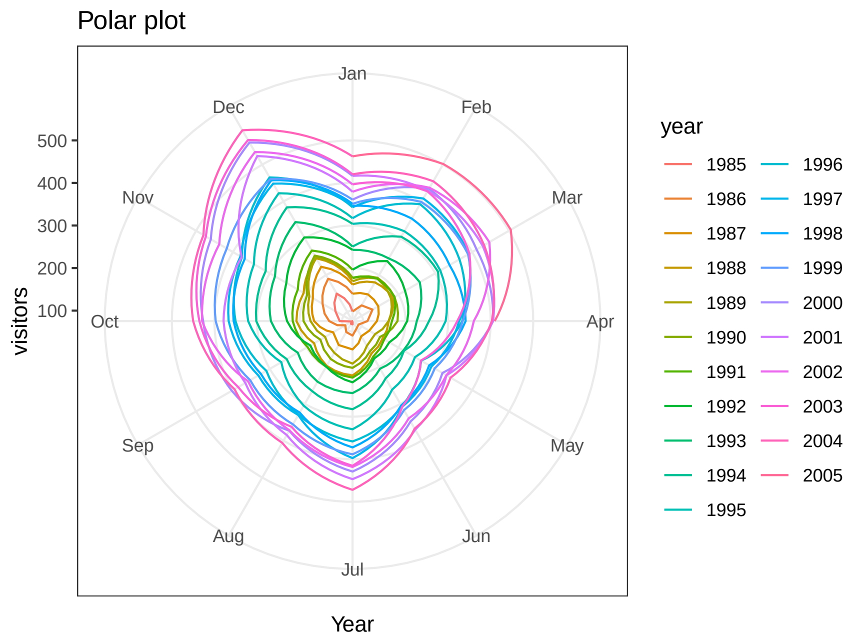


Figure 49: Seasonal polar plot for the ‘visitors‘ dataset.

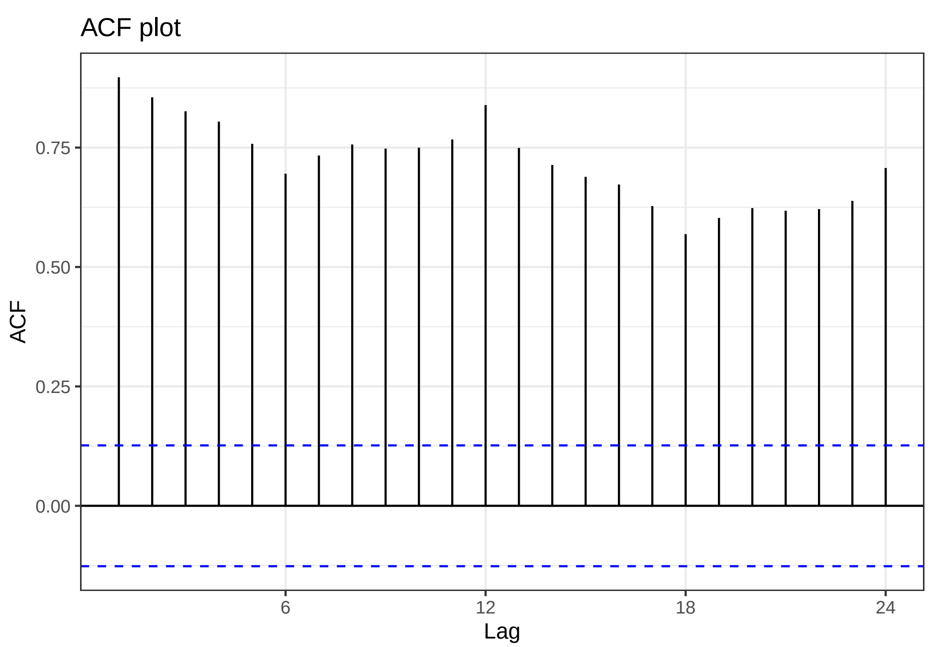


Figure 50: Autocorrelations of the ‘visitor‘ dataset. Lags are in months.

Let us plot two decomposition graphs, one additive and another multiplicative, and then decide which of them better fits our data.

We see the additive one leads to a distribution of the remainder which does appear to have heteroskedasticity. More precisely, at the later years there are quite prominent peaks compared to other time ranges. In the center there is kind of a dip.

```
autoplot(decompose(visitors, "additive")) + theme_bw()
```

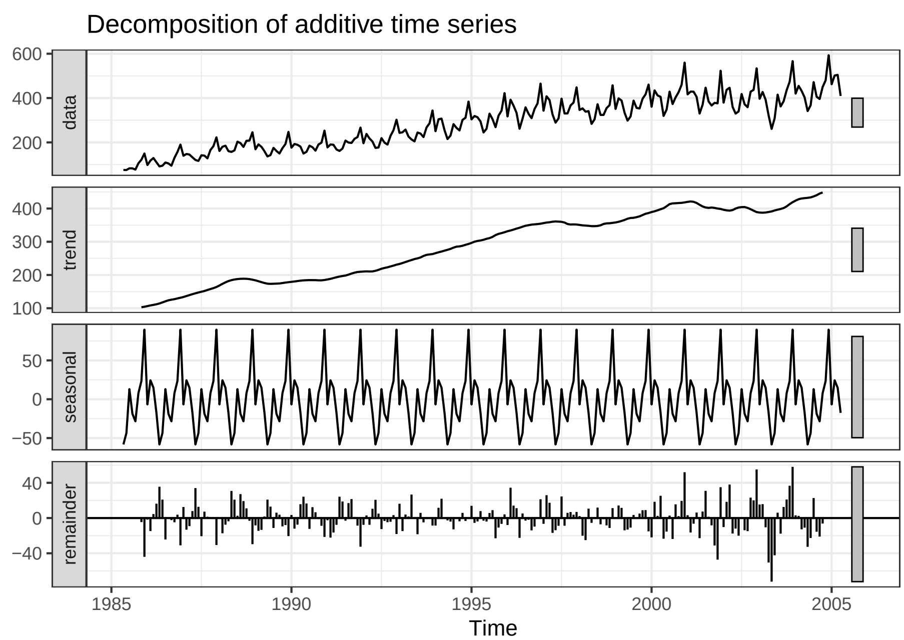


Figure 51: Classical decomposition of the 'visitors' data set assuming the underlaying model is additive on the tendency and errors.

On the other hand, with the multiplicative decomposition we observe a more even distribution of remainders, which do not look distributed following any pattern, without clear differences between areas.

```
autoplot(decompose(visitors, "multiplicative")) + theme_bw()
```

With these facts in mind, we decide the latter, the multiplicative model, better fits our data.

(b)

As discussed before, we will use a multiplicative seasonal decomposition when performing a Holter-Winters fit. We also decide that, for both, performance of the forecast and numerical cost, we will limit the data used for forecasting from the year 2000 onwards.

```
vis_cut <- window(visitors, start = 2000)
fit_hw  <- hw(vis_cut, seasonal = "multiplicative", h = 2 * 12)

autoplot(window(vis_cut, start = 1995)) +
    autolayer(fit_hw, PI = FALSE, col = "deepskyblue4") +
    xlab("Year") + ylab("Visitors") + ggtitle("Forecasting 2 years") + theme_bw()
```

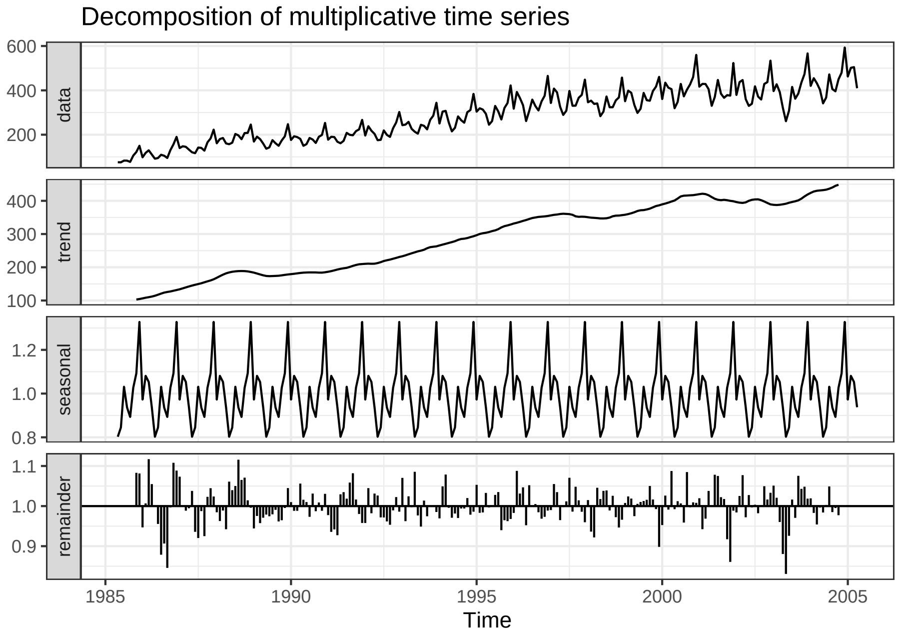


Figure 52: Classical decomposition of the 'visitors' data set assuming the underlaying model is multiplicative on the tendency and errors.

```
## Warning in window.default(x, ...): 'start' value not changed
```

We see that the previously made decision does indeed seem grounded as the forecast, at least on the naked eye, does look as a good candidate.

(c)

We are now tasked with testing the forecast with different options. Taking the same Holter-Winters multiplicative model, we experiment by choosing exponential trend, damped trend and both of them at once. If we plot them together we see that all of them do *a priori* a good job in forecasting the following two years. In this regards, the exponential trend offers more volatile extrema, while the damped offers the smallest. The curve with both of them does seem like a middle compromise between the exponential and damped models.

```
fit_exp  <- hw(vis_cut, seasonal = "multiplicative", h = 2 * 12, exponential = TRUE)
fit_dam  <- hw(vis_cut, seasonal = "multiplicative", h = 2 * 12, damped = TRUE)
fit_both <- hw(vis_cut, seasonal = "multiplicative", h = 2 * 12,
               exponential = TRUE, damped = TRUE)

autoplot(window(vis_cut, start = 2000)) +
    autolayer(fit_exp,  PI = FALSE, series = "Exponential") +
    autolayer(fit_dam,  PI = FALSE, series = "Damped") +
    autolayer(fit_both, PI = FALSE, series = "Exp + Damp") +
    xlab("Year") + ylab("Visitors") + ggtitle("Testing HW hyperparameters") +
    guides(colour = guide_legend(title ="Model")) + theme_bw()
```

It seems clear that by eyeballing we will not be able to properly select one of these models as the most appropiate. Hence we resort to metrics such as AIC, AICc and BIC. Looking at them it seems that the

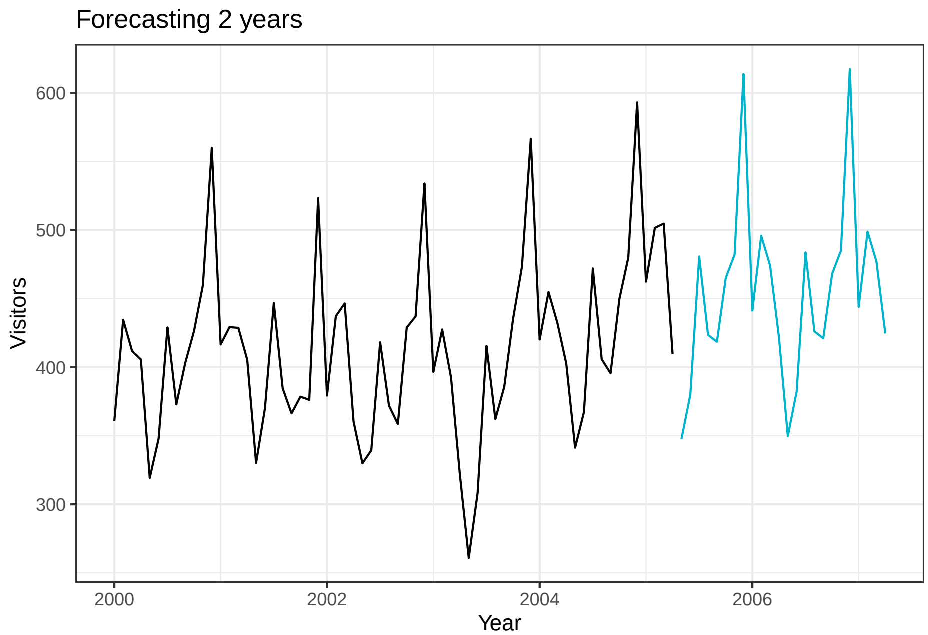


Figure 53: Forecast of two years using multiplicative Holter-Winters as the the forecasting model.

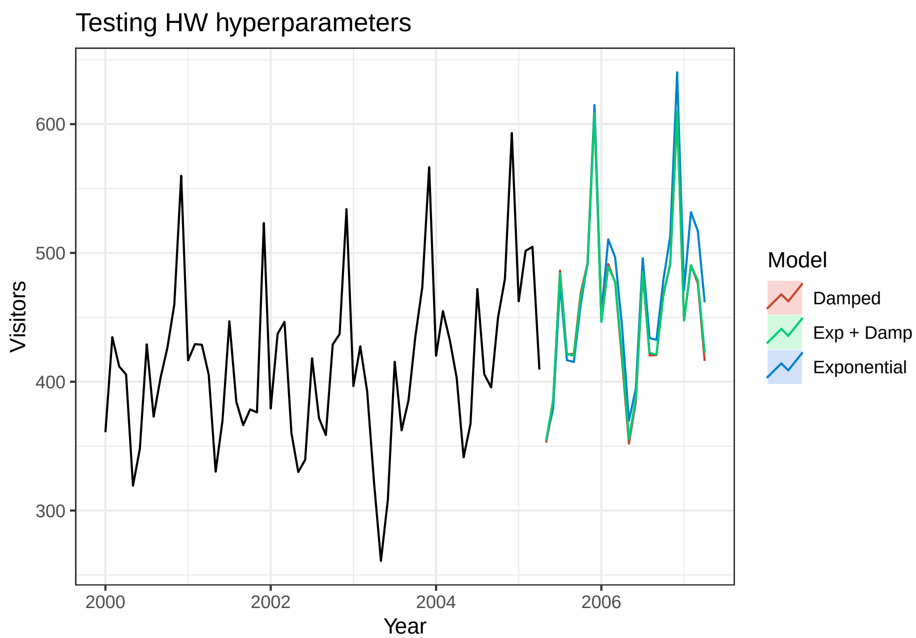


Figure 54: Forecast of two years using multiplicative Holter-Winters as the the forecasting model with different modifications to the tendency component: exponential, damped and both combined.

previous modifications do indeed improve on the forecasting. Among them, the exponential and damped takes the edge over the just damped one by a slight margin, giving us an indication that reduced extrema better forecasts our time series.

```
knitr::kable(data.frame(
    "Default"  = c(fit_hw$model$aic,   fit_hw$model$aicc,   fit_hw$model$bic),
    "Exp"      = c(fit_exp$model$aic,  fit_exp$model$aicc,  fit_exp$model$bic),
    "Damped"   = c(fit_dam$model$aic,  fit_dam$model$aicc,  fit_dam$model$bic),
    "Exp.Damp" = c(fit_both$model$aic, fit_both$model$aicc, fit_both$model$bic),
    row.names = c("AIC", "AICc", "BIC")
))
```

	Default	Exp	Damped	Exp.Damp
AIC	700.4753	697.5382	686.5541	686.0368
AICc	713.7797	710.8426	701.7541	701.2368
BIC	737.1763	734.2392	725.4140	724.8967

(d)

This is a just computational part in which are tasked with fitting three models:

1. An ETS model. Automatically this chooses the model for the error, trend and seasonality that better suits our data. Note that it chose the same that we used for our Holter-Winter model, giving us confidence on our previously extracted conclusions.

```
fit_ets <- ets(vis_cut)
summary(fit_ets)
```

```
## ETS(A,N,A)
##
## Call:
##  ets(y = vis_cut)
##
##   Smoothing parameters:
##     alpha = 0.7188
##     gamma = 1e-04
##
##   Initial states:
##     l = 409.9759
##     s = 139.4822 30.3792 10.6118 -31.1065 -32.0805 26.2524
##            -63.0939 -92.4909 -30.097 19.5845 33.0686 -10.5098
##
##   sigma:  21.1111
##
##      AIC     AICc      BIC
## 670.7438 680.7438 703.1270
##
## Training set error measures:
##                        ME     RMSE      MAE        MPE     MAPE      MASE
## Training set 0.8927493 18.65976 15.50572 0.07119266 3.850555 0.4952687
##                        ACF1
## Training set -0.02160021
```

2. An additive ETS model with Box-Cox transformed data. Here we impose the additivity of the model but let the algorithm to determine the correct λ needed for the transformation.

```r
fit_bc  <- ets(vis_cut, model = "AAA", lambda = "auto")
summary(fit_bc)
```

```
## ETS(A,A,A)
##
## Call:
##  ets(y = vis_cut, model = "AAA", lambda = "auto")
##
##   Box-Cox transformation: lambda= -0.2557
##
##   Smoothing parameters:
##     alpha = 0.7475
##     beta  = 0.0066
##     gamma = 2e-04
##
##   Initial states:
##     l = 3.0674
##     b = 5e-04
##     s = 0.0601 0.0148 0.0073 -0.0116 -0.0117 0.0174
##            -0.0316 -0.0537 -0.016 0.0111 0.0174 -0.0035
##
##   sigma:  0.0124
##
##       AIC      AICc       BIC
## -279.9891 -266.6848 -243.2881
##
## Training set error measures:
##                     ME     RMSE     MAE        MPE    MAPE     MASE        ACF1
## Training set 0.2552246 19.94666 16.4712 -0.1392087 4.07376 0.526107 -0.04972495
```

3. An additive ETS model applied to the STL-seasonality-freed Box-Cox transformed time series. Again, we let the algorithm decide the appropriate transformation parameter. Nonetheless, we fix the seasonality period as 12 months.

```r
fit_stl <- stlm(vis_cut, s.window = 12, method = "ets", lambda = "auto")
summary(fit_stl)
```

```
##               Length Class  Mode
## stl           256    mstl   numeric
## model          19    ets    list
## modelfunction   1    -none- function
## lambda          1    -none- numeric
## x              64    ts     numeric
## series          1    -none- character
## m               1    -none- numeric
## fitted         64    ts     numeric
## residuals      64    ts     numeric
```

(e)

At last, we are tasked with comparing these last three approaches to see which of them performs better. To do so, we first plot all of them together and see whether there is some significant difference between them visible by the naked eye.

If we look at the forecast, we note that the additive ETS applied to the Box-Cox transformed data has in general higher values. Then, when using the seasonality adjust forecast, there is a shift to lower values while keeping a similar shape. Lastly, the plain ETS prediction seems to have smaller values in general, with more prominent minima.

```
autoplot(vis_cut) +
    autolayer(forecast::forecast(fit_ets, h = 2 * 12),
              PI = FALSE, series = "ETS") +
    autolayer(forecast::forecast(fit_bc,  h = 2 * 12),
              PI = FALSE, series = "Box-Cox") +
    autolayer(forecast::forecast(fit_stl, h = 2 * 12),
              PI = FALSE, series = "BC-Seas") +
    xlab("Year") + ylab("Visitors") + ggtitle("Three model comparison") +
    guides(colour = guide_legend(title ="Model")) + theme_bw()
```

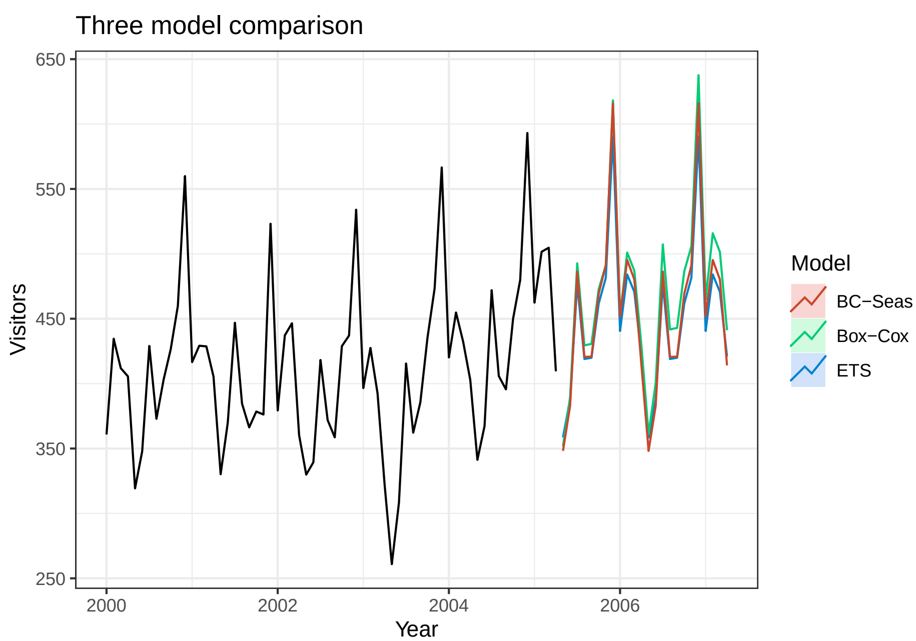


Figure 55: Forecast of two years with three different models: ETS (the algorithm chose 'MAM' as the most appropiated decomposition), additive ETS applied to the Box-Cox transformed data, and additive ETS applied to the seasonally adjusted (via STL decomposition) Box-Cox transformed data.

Again, just by looking at the graph we cannot tell which of them better describe our time series. Thus we resort to accuracy metrics. More exactly, we will be using the `tsCV` function which performs cross-validations and return a vector of residuals. We then sum the square residuals and declare the one with the smallest sum as the most suitable for our case. We also plot them to have visual feedback on the magnitude and whether they resemble white noise, which all of them do. This happens for the last of the models, which performed slightly better than the rest, highlighting the vital importance of taking the necessary preprocessing steps before fitting and forecasting.

```r
f1 <- function(y, h) { forecast::forecast(ets(y, model = "MAM"), h = h) }
f2 <- function(y, h) { forecast::forecast(ets(y, model = "AAA",
                                              lambda = "auto"), h = h) }
f3 <- function(y, h) { forecast::forecast(
  stlm(y, s.window = 12, method = "ets", lambda = "auto"), h = h) }
e1 <- tsCV(vis_cut, f1, h = 1)
e2 <- tsCV(vis_cut, f2, h = 1)
e3 <- tsCV(vis_cut, f3, h = 1)

# e3 starts at 2002, we cut them
e1 <- window(e1, start = 2002)
e2 <- window(e2, start = 2002)
e3 <- window(e3, start = 2002)

c(
    "ETS"     = sqrt(mean(e1 ^ 2, na.rm = TRUE)),
    "Box-Cox" = sqrt(mean(e2 ^ 2, na.rm = TRUE)),
    "BC-Seas" = sqrt(mean(e3 ^ 2, na.rm = TRUE))
)
```

```
##      ETS  Box-Cox  BC-Seas
## 27.66942 25.85527 23.92951
```

```r
autoplot(e1, series = "ETS") +
    autolayer(e2, series = "Box-Cox") +
    autolayer(e3, series = "BC-Seas") +
    xlab("Year") + ylab("Visitors") + ggtitle("Residuals") +
    guides(colour = guide_legend(title ="Model")) + theme_bw()
```

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

Exercise 7

Data set **books** contains the daily sales of paperback and hardcover books at the same store. The task is to forecast the next four days sales for paperback books (data set **books**).

(a) Plot the series and discuss the main features of the data.

(b) Use simple exponential smoothing with the **ses** function (setting `initial="simple"`) and explore different values of α for the paperback series. Record the within-sample SSE for the one-step forecasts. Plot SSE against α and find which values of α works best. What is the effect of α on the forecasts?

(c) Now let **ses** select the optimal value of α. Use this value to generate forecasts for the next four days. Compare your results with (b).

(a)

As said in the problem statement, the data set *books* contains the daily sales of paperback and hardcover books:

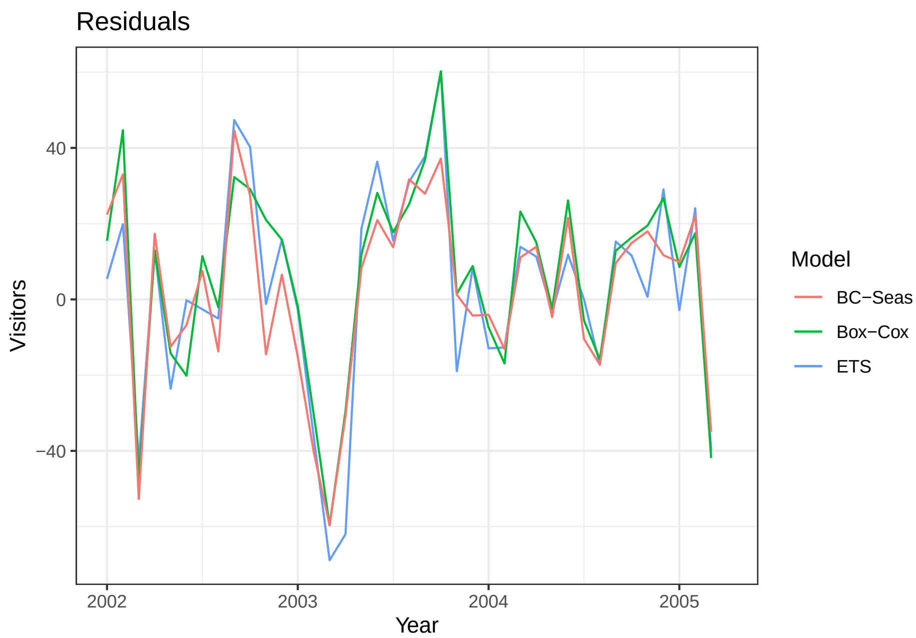


Figure 56: Cross-Validation computed residuals on the 'visitors' dataset from year 2002 onwards with three different models: ETS (the algorithm chose 'MAM' as the most appropiated decomposition), additive ETS applied to the Box-Cox transformed data, and additive ETS applied to the seasonally adjusted (via STL decomposition) Box-Cox transformed data.

```r
head(books, 3)
```

```
## Time Series:
## Start = 1
## End = 3
## Frequency = 1
##   Paperback Hardcover
## 1       199       139
## 2       172       128
## 3       111       172
```

We can see that the frequency is 1, so there's no seasonality in the data.

The interest lies in the paperback books so we're going to save the data related to the paperback books sales in a new variable called `paperback`.

```r
paperback <- books[,1]
```

Now, we do a time plot for the data:

```r
autoplot(paperback, main = "Time plot: Daily sales of paperback books",
        xlab = "Time",
        ylab = "Number of books sold") + theme_bw()
```

At first glance we can see a possible positive trend. We could also talk about the possible existence of cycles but, as it is said in the theory of the subject: *"The duration of a cycle extends over longer period of*

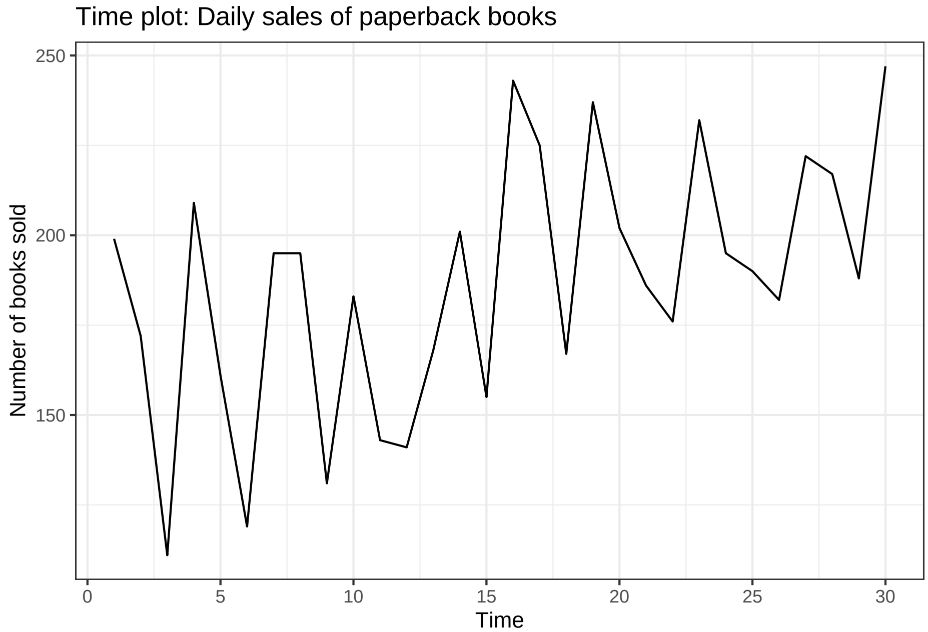


Figure 57: Time plot for the daily sales of paperback books.

time, usually two or more years. two or more years.”, and, in this case, we would be talking about cycles of only a few days, so we may have doubts as to whether this is really a cyclical behavior.

Next, we’re going to plot the ACF:

```
ggAcf(paperback) + ggtitle("ACF of daily sales of paperback books") + theme_bw()
```

The ACF does not seem to show the presence of a positive trend in the data. The high value for r~3 may indicate cycles of length 3, but it is really hard to tell.

We have a little theory about this: maybe the days are not necessarily from Monday to Sunday, and it is from Monday to Saturday (1 -> Monday, 6 -> Saturday, 7 -> Monday . . .) and that both Wednesdays and Saturdays there are offers, which makes sales on those days go up. This is probably not the case, but it is a silly explanation we found that may explain these “cycles”.

Now, looking to the lagplots:

```
par(mfrow= c(1,2))
gglagplot(paperback) + ggtitle("Lagged scatterplots for
                               daily sales of paperback books") +
  theme_bw()
```

We cannot see the presence of cycles.

(b)

We start by generating 101 alphas (from 0 to 1 with a step of 0.01) and computing the within-sample SSE for the one-step forecasts for every of these alphas.

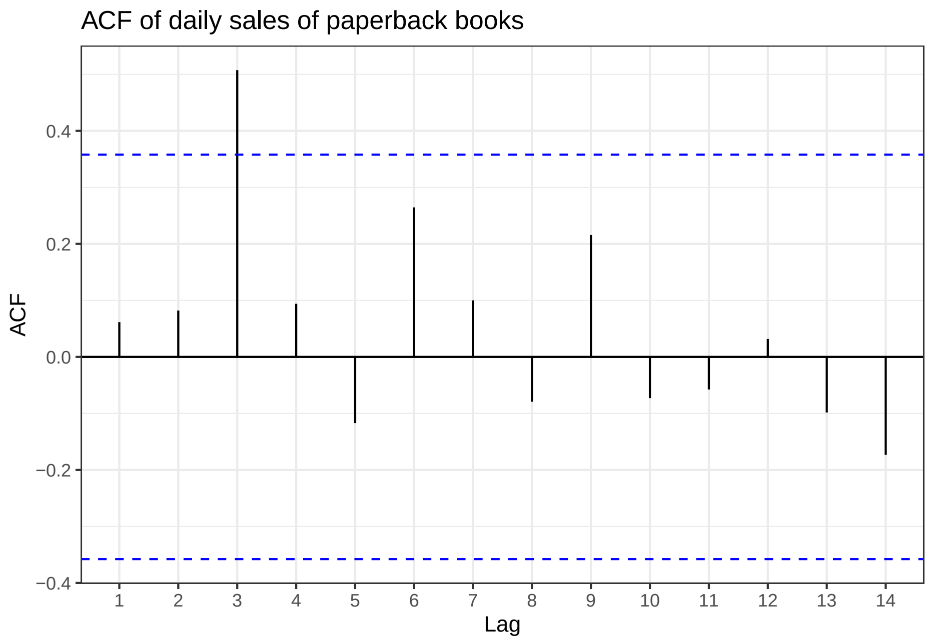


Figure 58: ACF of daily sales of paperback books.

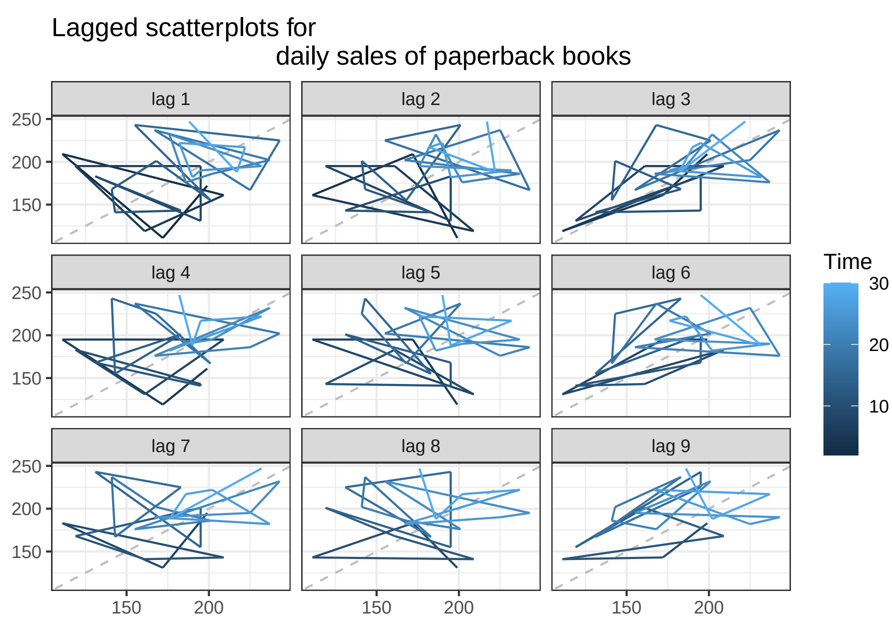


Figure 59: Lagged scatterplots for daily sales of paperback books.

```r
alphas <- seq(0,1,0.01)
fits   <- lapply(alphas, function(alpha) {ses(paperback, initial = "simple",
                                              h = 1, alpha = alpha)})
SSEs   <- lapply(fits, function(fit) {sum(fit$residuals^2)})
```

Now, we plot each alpha value with its corresponding SSE:

```r
df                <- data.frame(alphas=alphas, SSEs=unlist(SSEs))
df$label          <- paste("a = ",alphas)
best_found_alpha <- df[which.min(df$SSEs),1]

ggplot(data = df, aes(x = alphas, y = SSEs)) +
    geom_point(color="black", size=2) +
    geom_point(data = df[which.min(df$SSEs),], color="green",
               size=3) +
    geom_point(data = df[which.max(df$SSEs),], color="red",
               size=3) +
    geom_text(data = rbind(df[which.min(df$SSEs), ], df[which.max(df$SSEs),]),
              aes(alphas,SSEs+1500, label=label)) +
    xlab("alpha") +
    ylab("SSE") +
    ggtitle("SSE vs alpha for each value of alpha") +
    theme_bw()
```

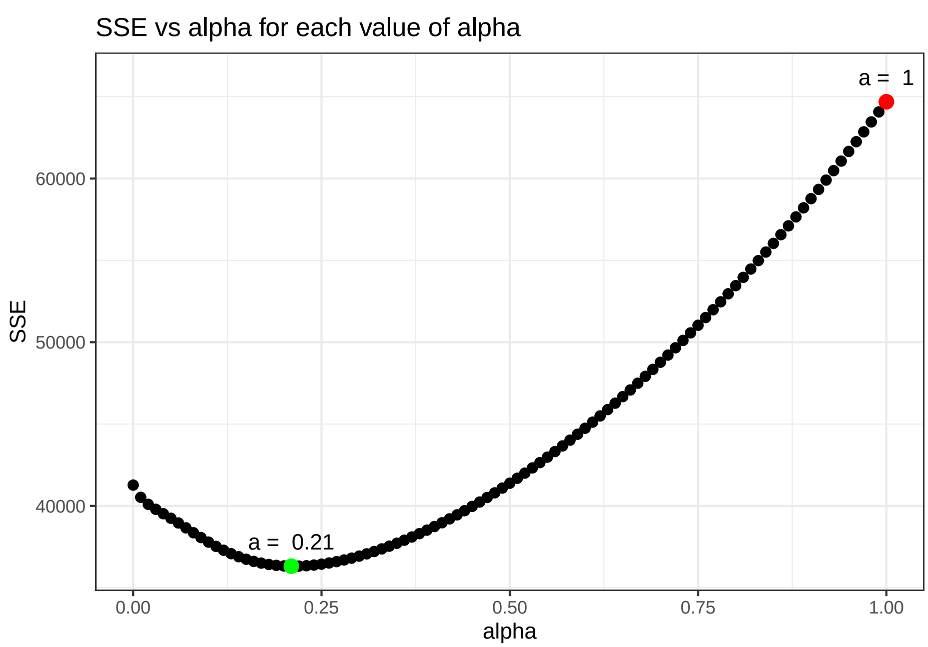


Figure 60: SSE vs alpha for each value of alpha.

According to the graph, the best α is 0.21 and the worst is 1. This makes sense as the SES model with an $\alpha = 1$ is no more than a naive model.

Answering to *"What is the effect of α on the forecasts?"*, α is changing the weight attached to the observations, with an small α (close to 0) giving more weight to the observations from the more distant past and a big α giving more weight to recent observations. As the model is exponential, we can say that

$\alpha = 0.21$ (the best α found according to the criteria of minimizing the SSE among all the α tested) is giving importance both to instances from the near and for the far past.

(c)

To let the `ses` function select the optimum value for α by itself, we need to leave the alpha parameter empty.

```
fit_auto_alpha <- ses(paperback, initial = "simple", h = 4, alpha = NULL)
fit_auto_alpha$model
```

```
## Simple exponential smoothing
##
## Call:
##  ses(y = paperback, h = 4, initial = "simple", alpha = NULL)
##
##   Smoothing parameters:
##     alpha = 0.2125
##
##   Initial states:
##     l = 199
##
##   sigma:  34.7918
```

The value found by the function is $\alpha = 0.2125$, really close to the value found by us in the previous section.

We plot the forecast with this new α:

```
plot_auto_alpha <- autoplot(fit_auto_alpha,
        main = "Forecasts from SES method for paperback (alpha = 0.2125)",
        xlab = "Time",
        ylab = "Daily sales of paperback") + theme_bw()
plot_auto_alpha
```

Comparing the forecasting with both alphas we can see that they are pretty much the same, being able to observe minimal differences.

```
plot_found_alpha <- autoplot(ses(paperback, initial = "simple", h = 4,
                                 alpha = best_found_alpha),
        main = "Forecasts from SES method for paperback (alpha = 0.21)",
        xlab = "Time",
        ylab = "Daily sales of paperback") + theme_bw()
plot_found_alpha
```

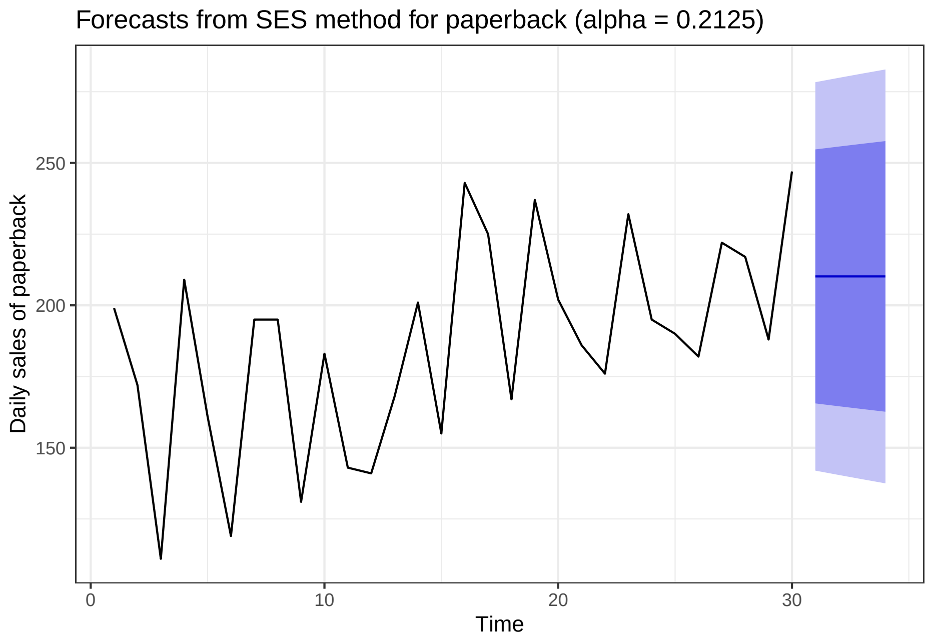


Figure 61: Forecasts from SES method for paperback (alpha = 0.2125)

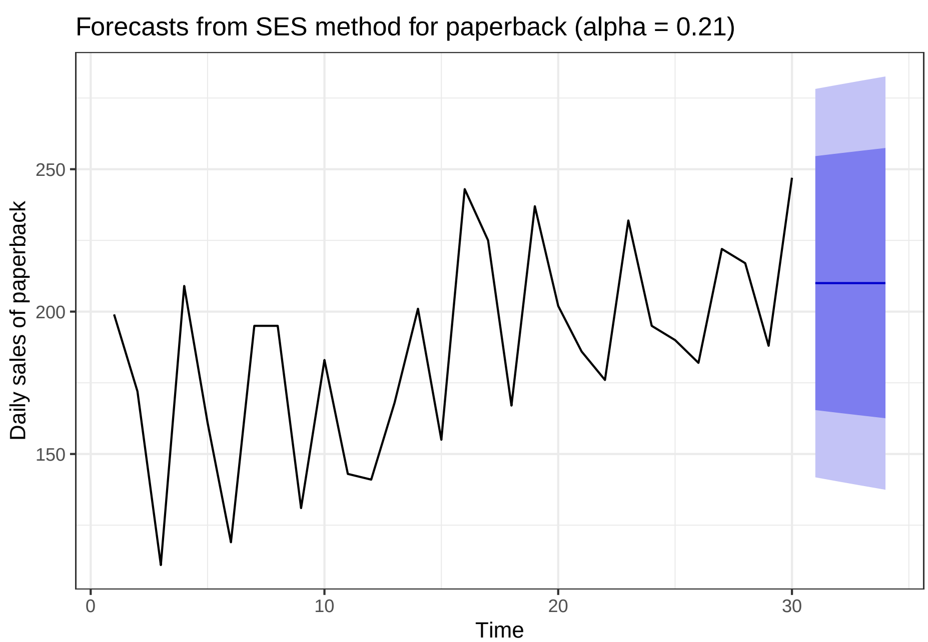


Figure 62: Forecasts from SES method for paperback (alpha = 0.21)