FORECASTING AND PREDICTIVE MODELING

LECTURE 4

Lect. PhD. Oneț-Marian Zsuzsanna

Babeș - Bolyai University
Computer Science and Mathematics Faculty

2024 - 2025

In Lecture 3

- Time series visualizations
  - Time series plot
  - Trend, season and cycle in a time series.
  - Seasonal plot
  - Seasonal subseries plot
  - Scatterplot

Today

- Time series visualizations
  - Lag plots
  - Autocorrelation
- Stationary time series
- Differencing

- A few data sets that we have used in the previous lecture and will use today as well:
  - A10 - monthly drug prescription about antidiabetic drugs from Australia from 1991 July to 2008 June
  - A11 - monthly drug prescription about vitamins from Australia from 1991 July to 2008 June
  - vic_elec - half-hourly electricity demand for Victoria, Australia, between 2012-01-01 and 2014-12-31 (52608 observations). Contains information about the electricity demand and the temperature of Melbourne.
  - aus_production - quarterly data about the production of different Australian commodities. In Lecture 3 we used the data about brick production.
  - lynx - yearly data about lynx population

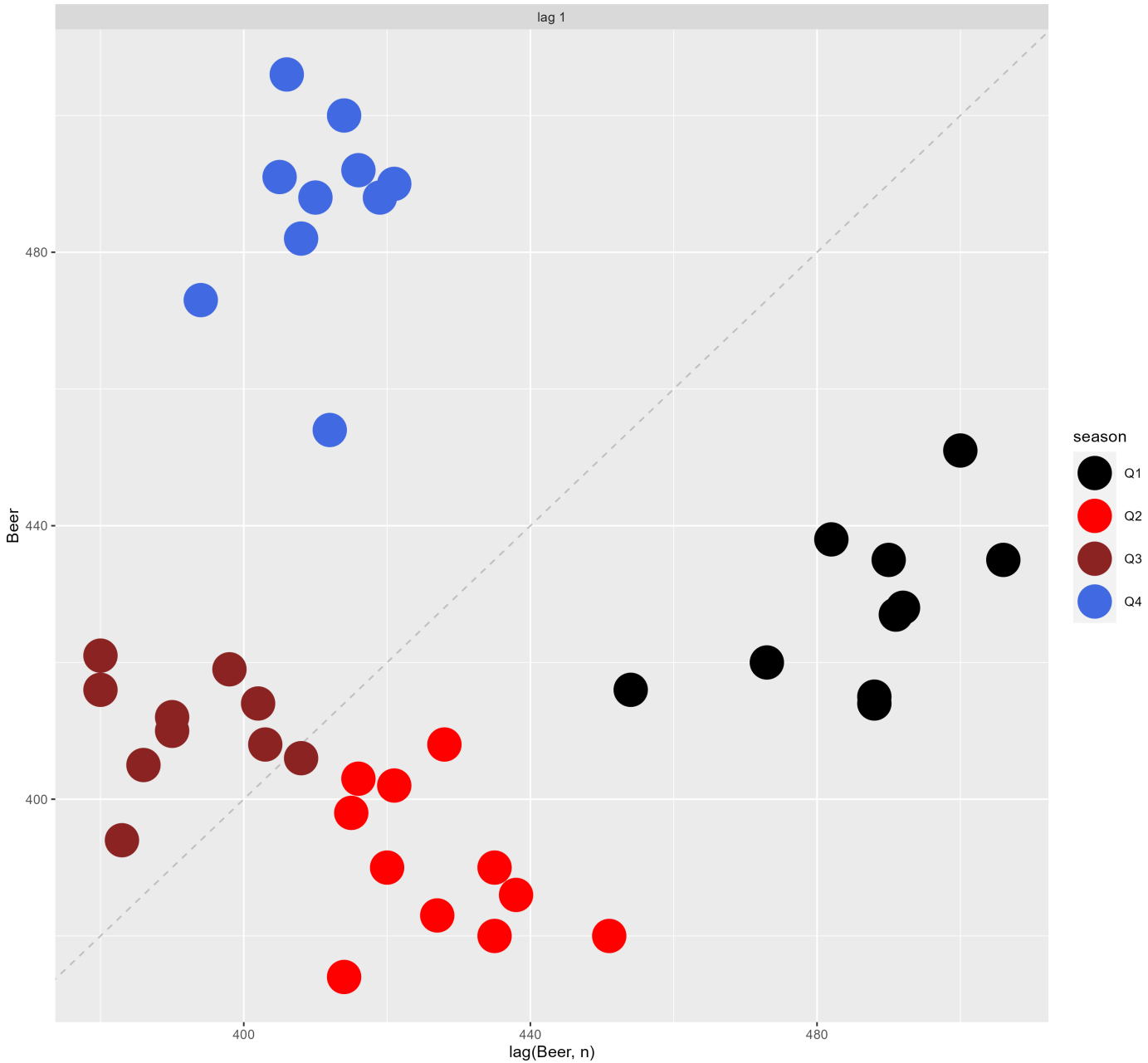
Lag plots

- *Lag plots* are scatter plots, where the time series y_t is plotted against y_{t-k} for different values of k.
- For example, for $k = 1$ we plot the value of y_t against y_{t-1}.
- In R a lag plot can be created using the function *gg_lag*

Lag plots

- Let us consider as example, a subset of the *aus_production* data set, which contains beer production after 2000.
- Let us create a scatterplot at a lag = 1.

```
recent_beer <- aus_production |>
  filter(year(Quarter) >= 2000) |>
  select(Quarter, Beer)

recent_beer
recent_beer |> autoplot()

recent_beer |> gg_lag(lags = 1, geom = "point") +
  scale_color_manual(values = c("black", "red", "brown4", "royalblue")) +
  geom_point(size = 10)
```

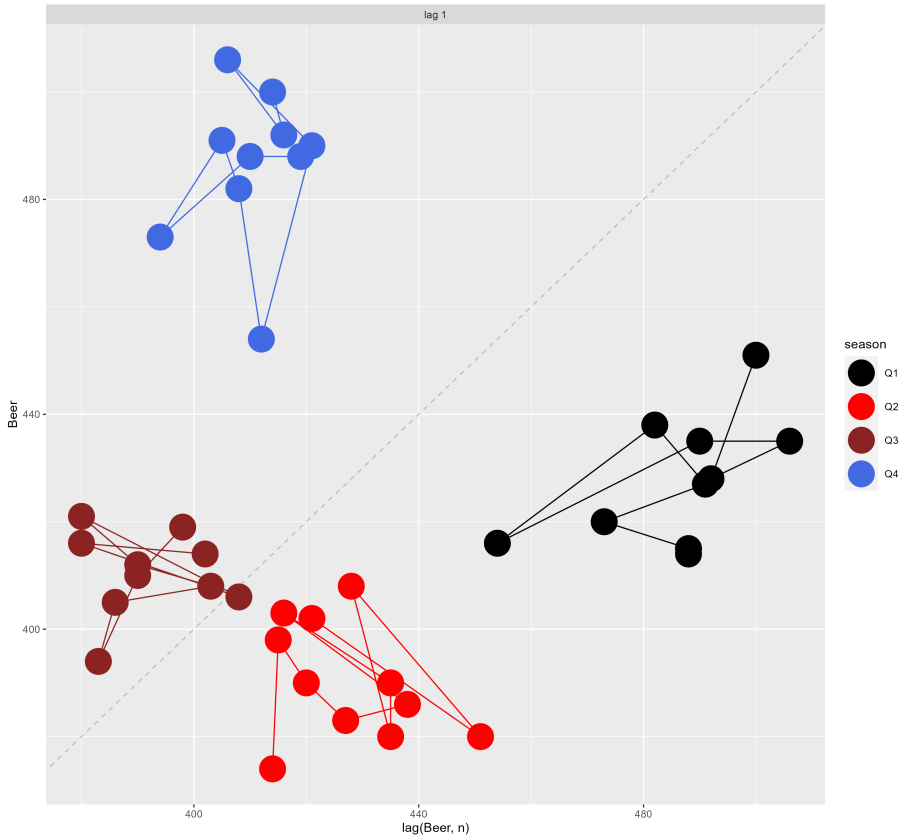

lag 1
Beer
480
440
400
400
440
480
lag(Beer, n)
season
Q1
Q2
Q3
Q4

Lag plot for k = 1

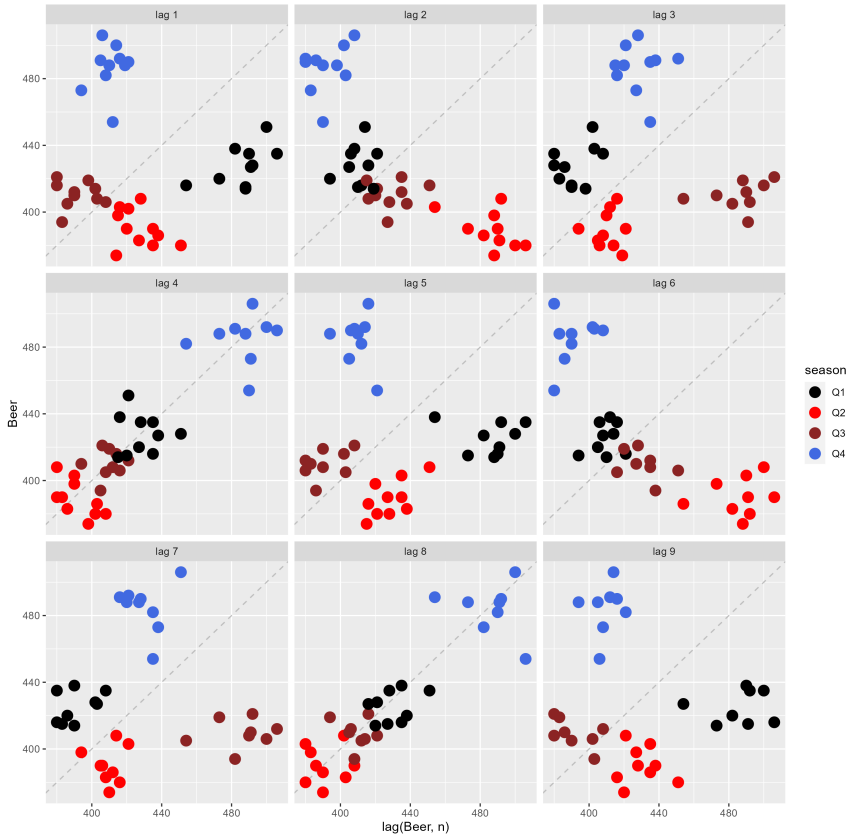
- On the previous slide, we have a plot of y_t (y-axis) over y_{t-1} (x-axis).
- The color of the dots denotes the quarter to which it belongs $\Rightarrow$ the blue dots in the upper left corner correspond to Q4 data, plot against Q3 data (for Q4 at lag 1 you find Q3 data).

Lag plot for k = 1

- If we remove the *geom* parameter from the *gg_lag* function (or set its value to *path*) points belonging to the same quarter will be connected chronologically.



- If we do not specify the lag for which we want the scatterplot, the *gg_lag* function will display by default plots up until lag 9.

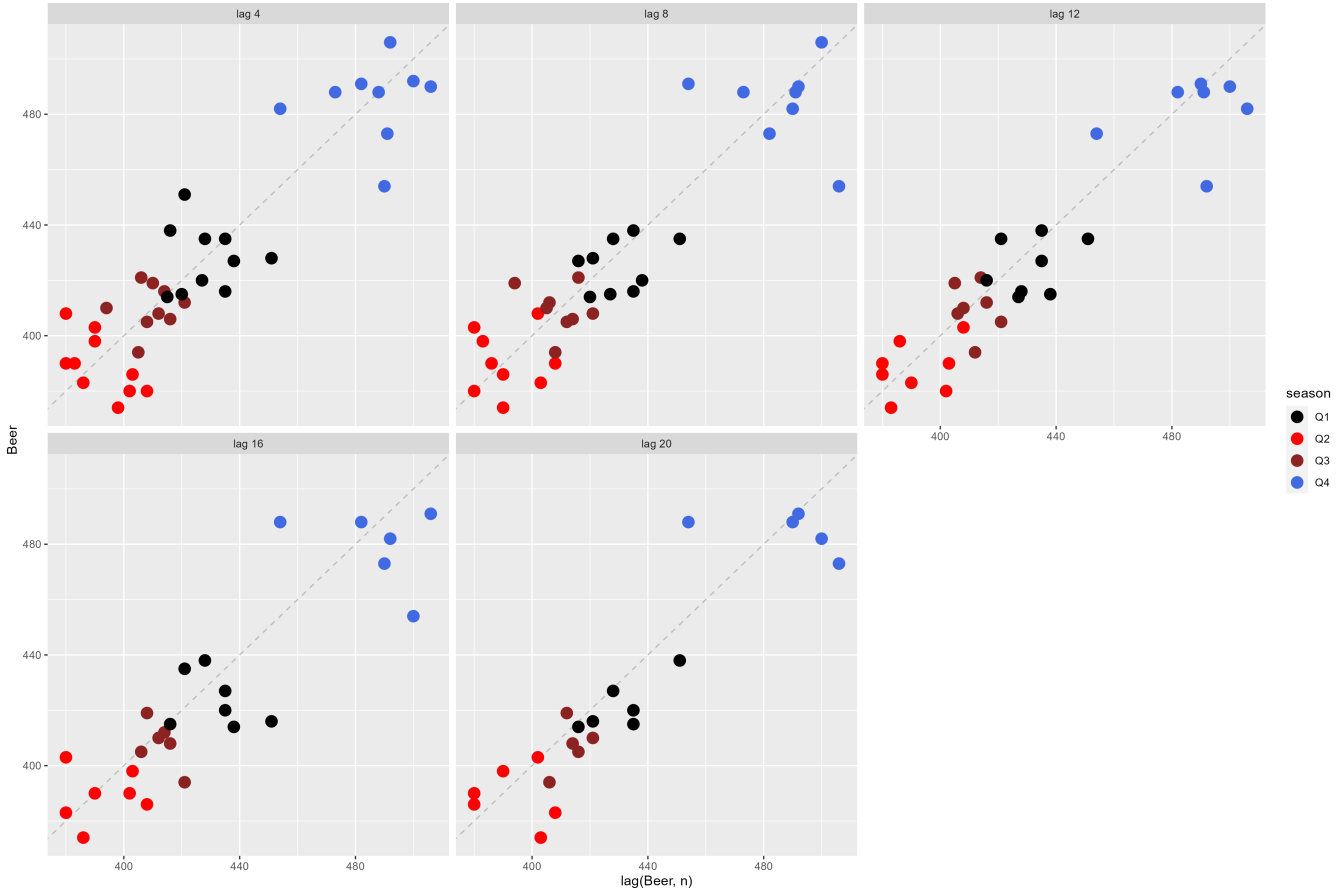


- Is there a lag for which there is a pattern in the data?

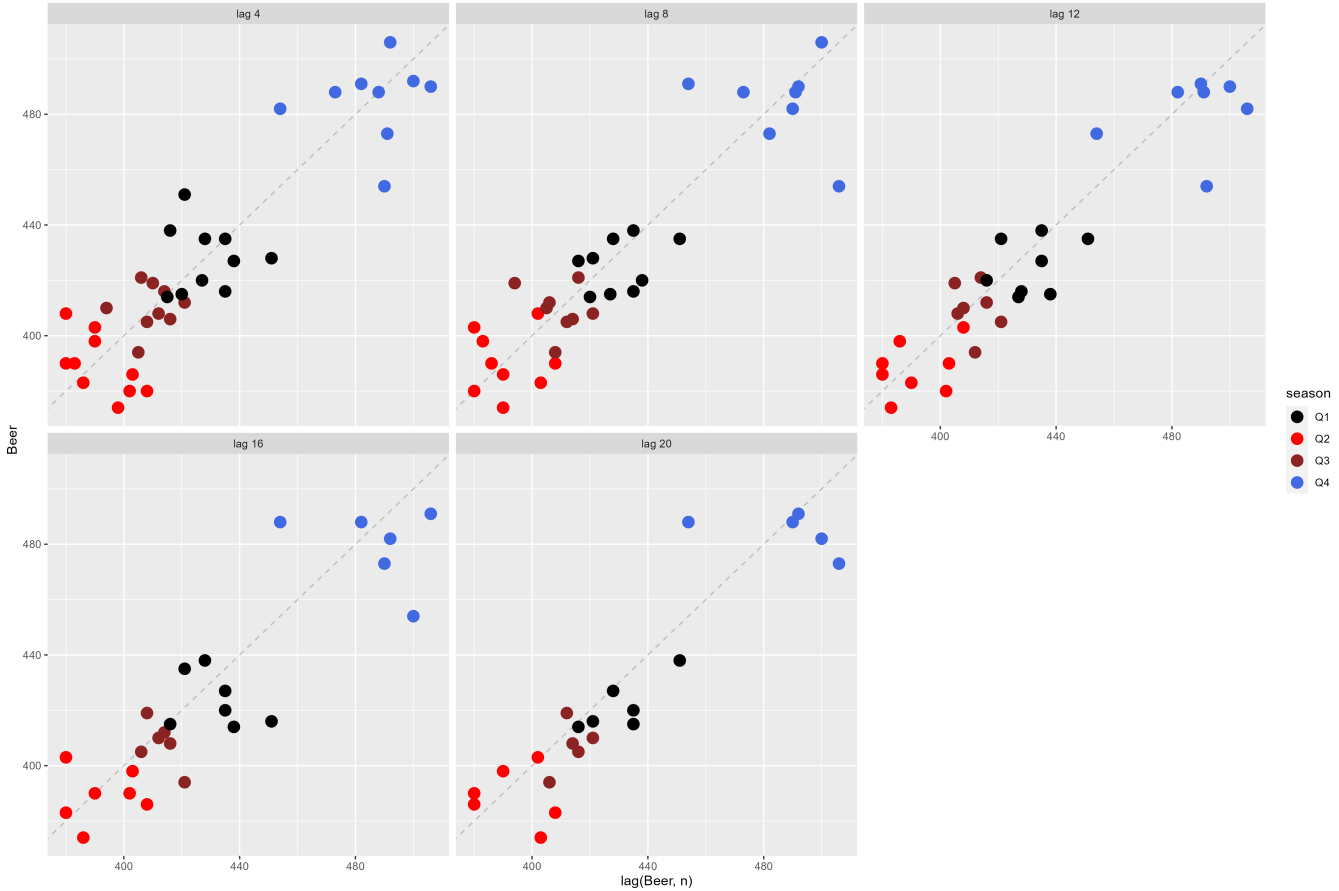
Selecting specific lags

- You can specify which lags to be used for the plot, using the *lag* parameter, whose value should be a vector (of lags). You can specify intervals (for ex: 2:5) or specific values.

```
recent_beer |> gg_lag(lags = c(4, 8, 12, 16, 20), geom = "point") +
  scale_color_manual(values = c("black", "red", "brown4", "royalblue")) +
  geom_point(size = 4)
```



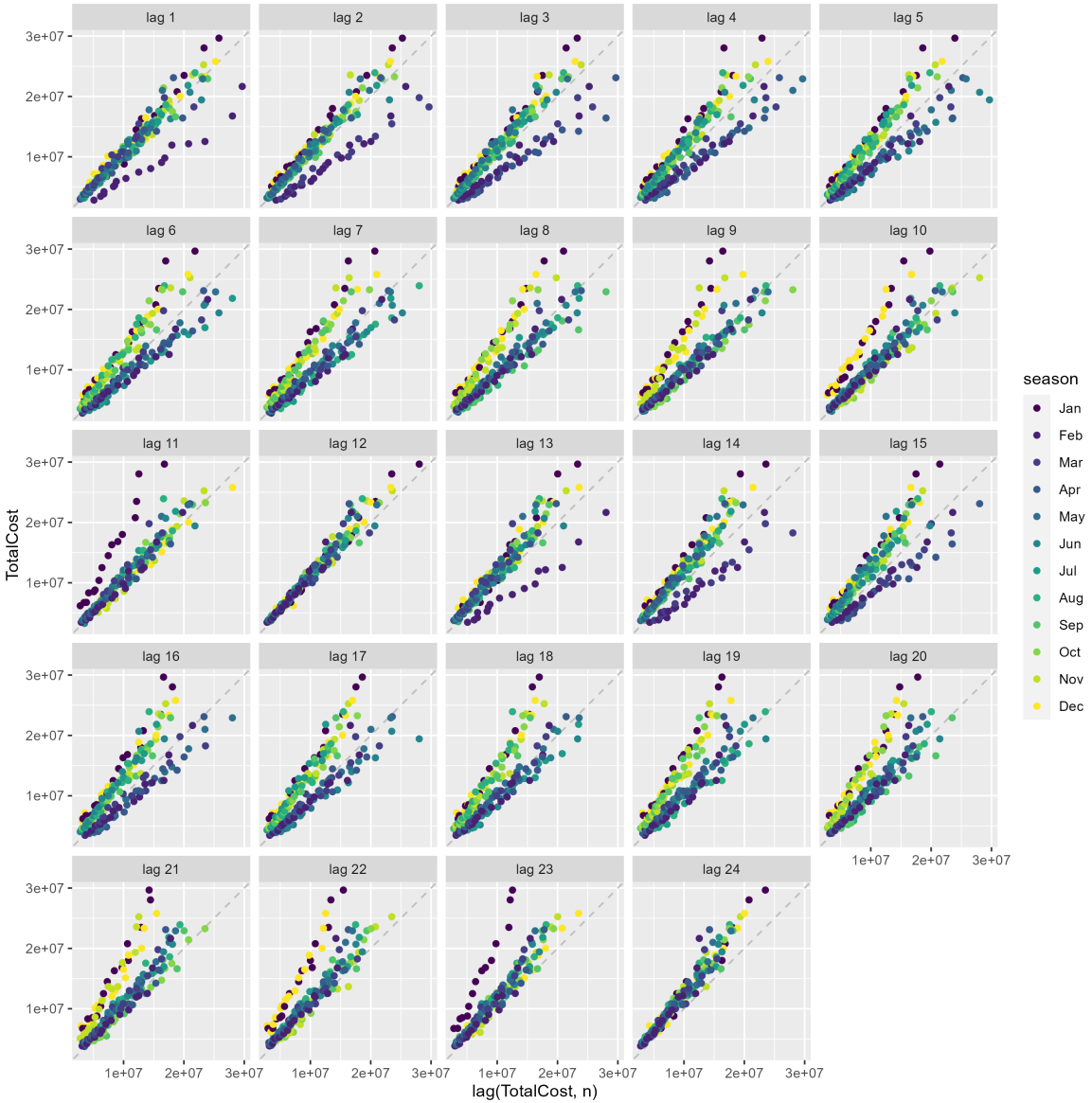
- We can see that there is a strong linear relationship between points and their lagged values at lags which are multiples of 4.
- What do you think is the reason for that?



- We can see that there is a strong linear relationship between points and their lagged values at lags which are multiples of 4.
- What do you think is the reason for that?
- Since data is seasonal, it makes sense to see a relation between points at a lag which is a multiple of 4.

- Let us see the lag plot of the A10 data set (monthly drug prescription for antidiabetic drugs)

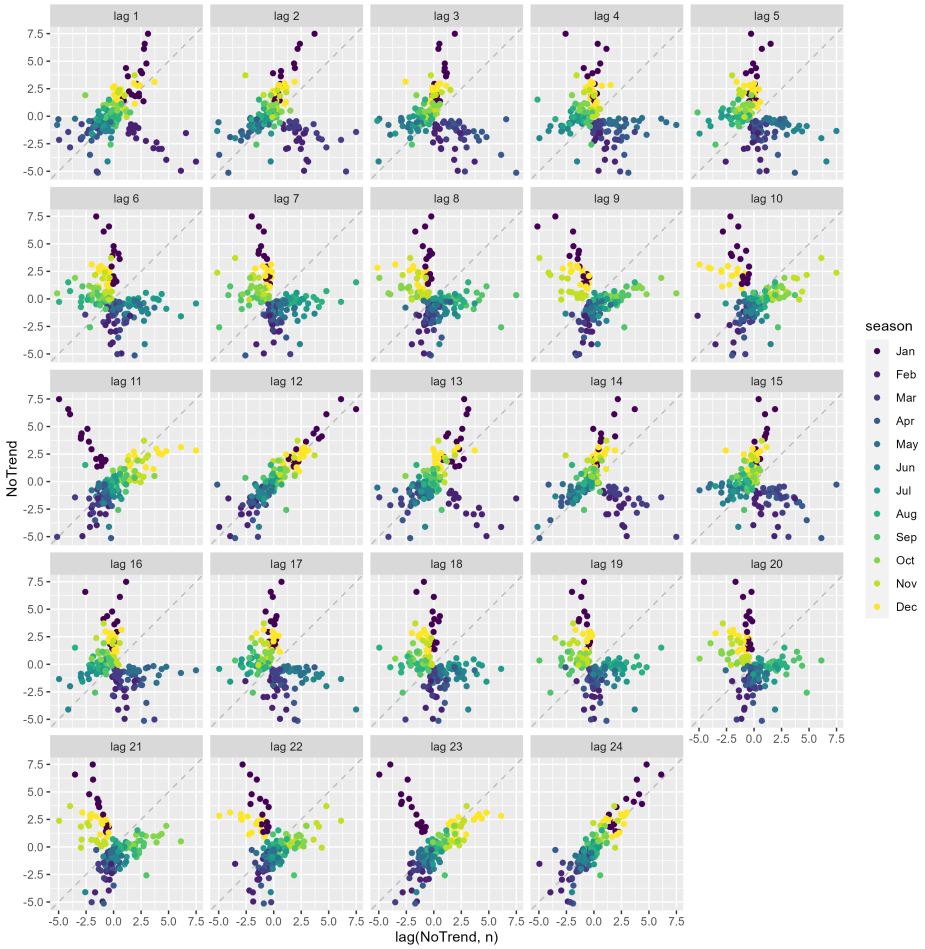
```
A10 <- PBS |>
  filter(ATC2 == "A10") |>
  select(Month, Concession, Type, Cost) |>
  summarize(TotalCost = sum(Cost)) |>
  mutate(Cost = TotalCost / 1000000)

A10 |> gg_lag(lags = 1:24, geom = "point")
```



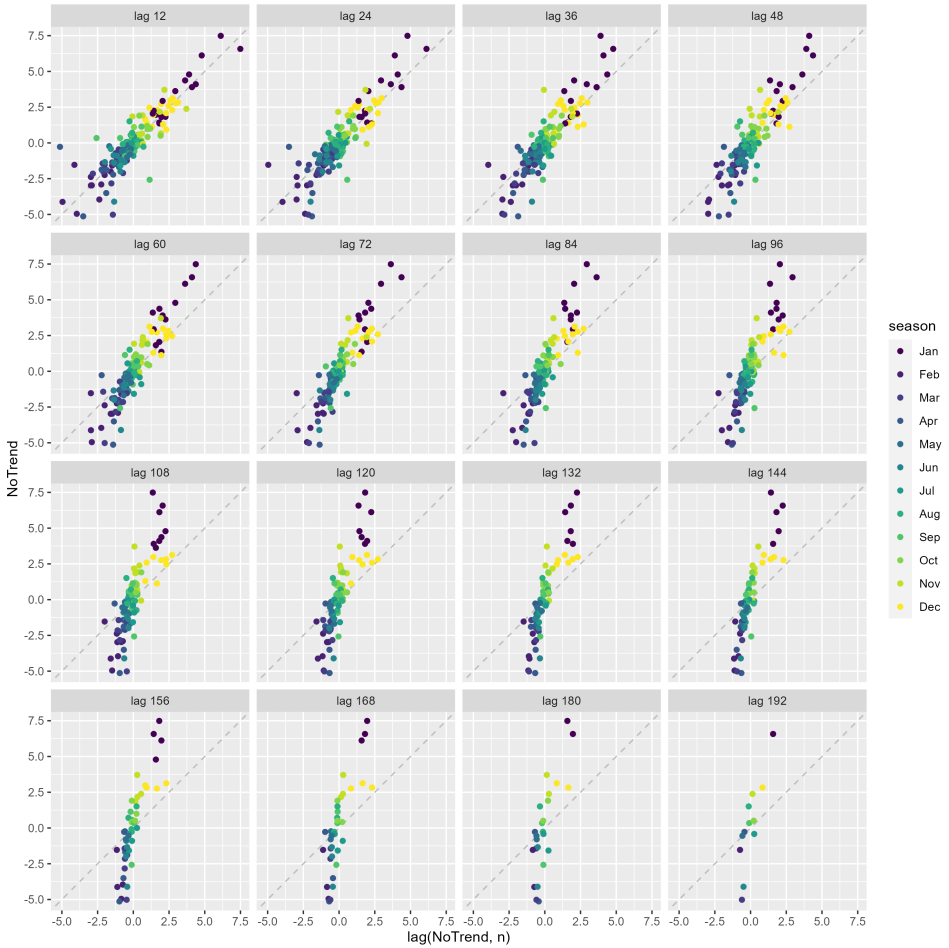
- Why is data correlated at all lags?

- Since the data has a trend, this is visible on the lag plots as well (just like in case of correlations, where trended data can lead to overly inflated correlations).
- Trend can also "hide" actual seasonality.
- If we eliminate the trend, we can see the actual seasonal pattern. (How to eliminate trend will be discussed later)

- After the elimination of trend

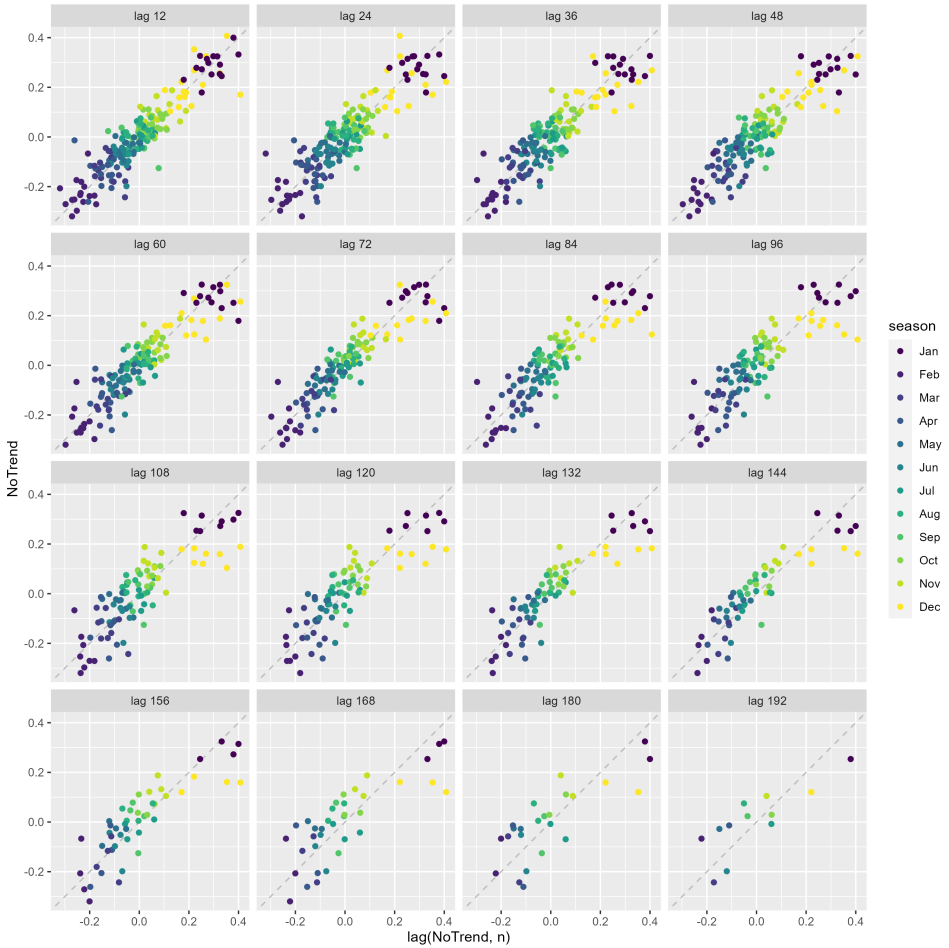


- If we only look at lags which are multiples of 12



- Why does the pattern in data change when the lag increases?

- After transforming data to eliminate multiplicative seasonality



Autocorrelation

- We can see that at some lags the data are strongly positively correlated. This shows the seasonality of the data.

- Correlation values can be computed for lagged values, they are called *autocorrelation*. You can have an autocorrelation coefficient for each possible value of k: for example, r_1 is the correlation between y_t and y_{t-1} (lag of 1).

- **Obs:** The actual way of computing the autocorrelation is the following (which is slightly different from how you would compute the Pearson correlation for the two attributes):

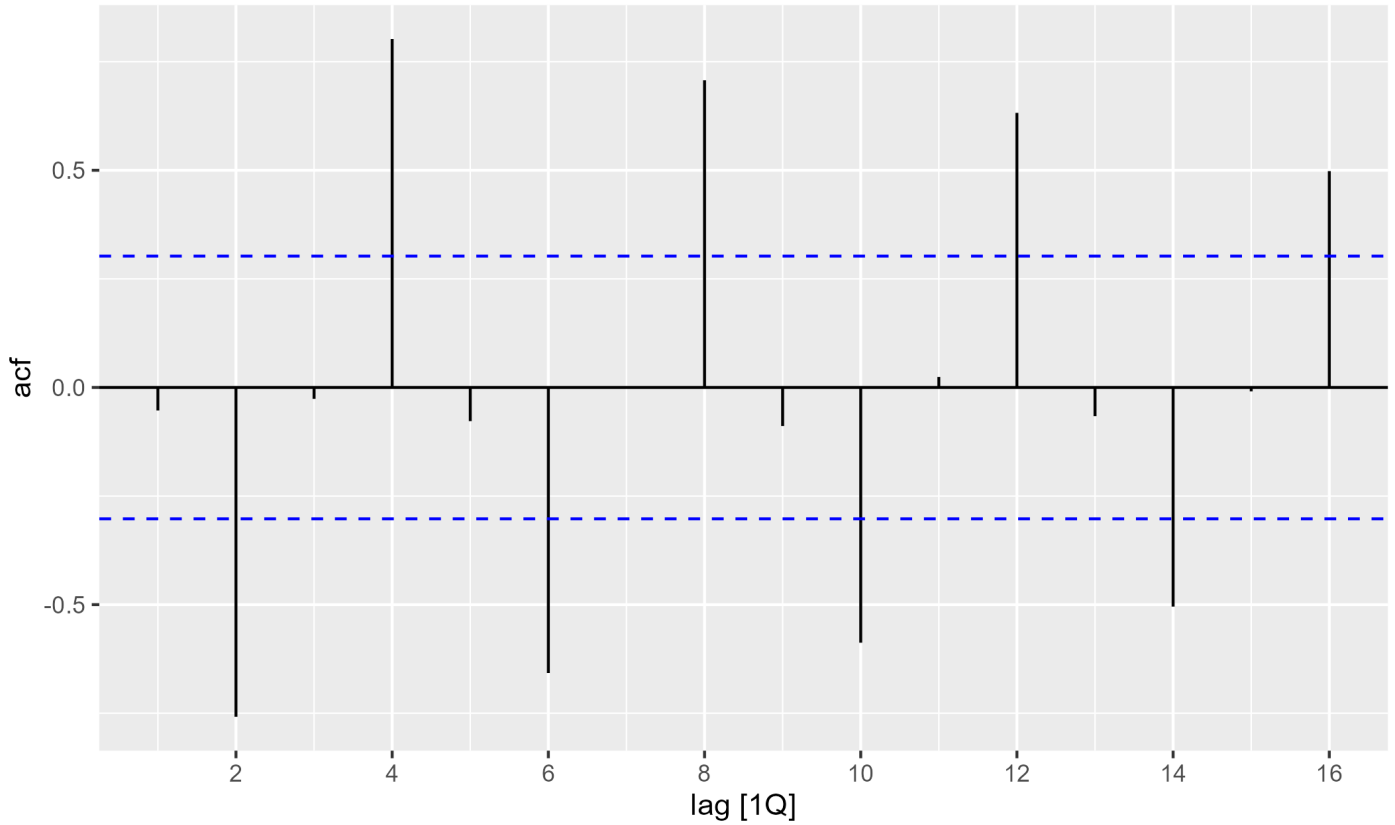
$$r_k = \frac{\sum_{t=k+1}^{T}(y_t - \bar{y})(y_{t-k} - \bar{y})}{\sum_{t=1}^{T}(y_t - \bar{y})^2}$$

Autocorrelation

- You can compute the autocorrelation values, using the *ACF* function. By default, the function computes the values up to a lag of 16, but this value can be set using the *lag_max* parameter.
- The result of *ACF* is a time series, which can be *autoplot* for a visual representation.

```
recent_beer |> ACF()
class(recent_beer |> ACF())
recent_beer |> ACF() |> as_tsibble(index = lag) |> autoplot(acf)
recent_beer |> ACF() |> autoplot()
recent_beer |> ACF() |> autoplot() + geom_point()
```

- **Obs.** When we look at the class of the result from the ACF function, its first type is *tbl_cf*, which is needed to make sure that *autoplot* plots it in a different ways, not like a regular time series.

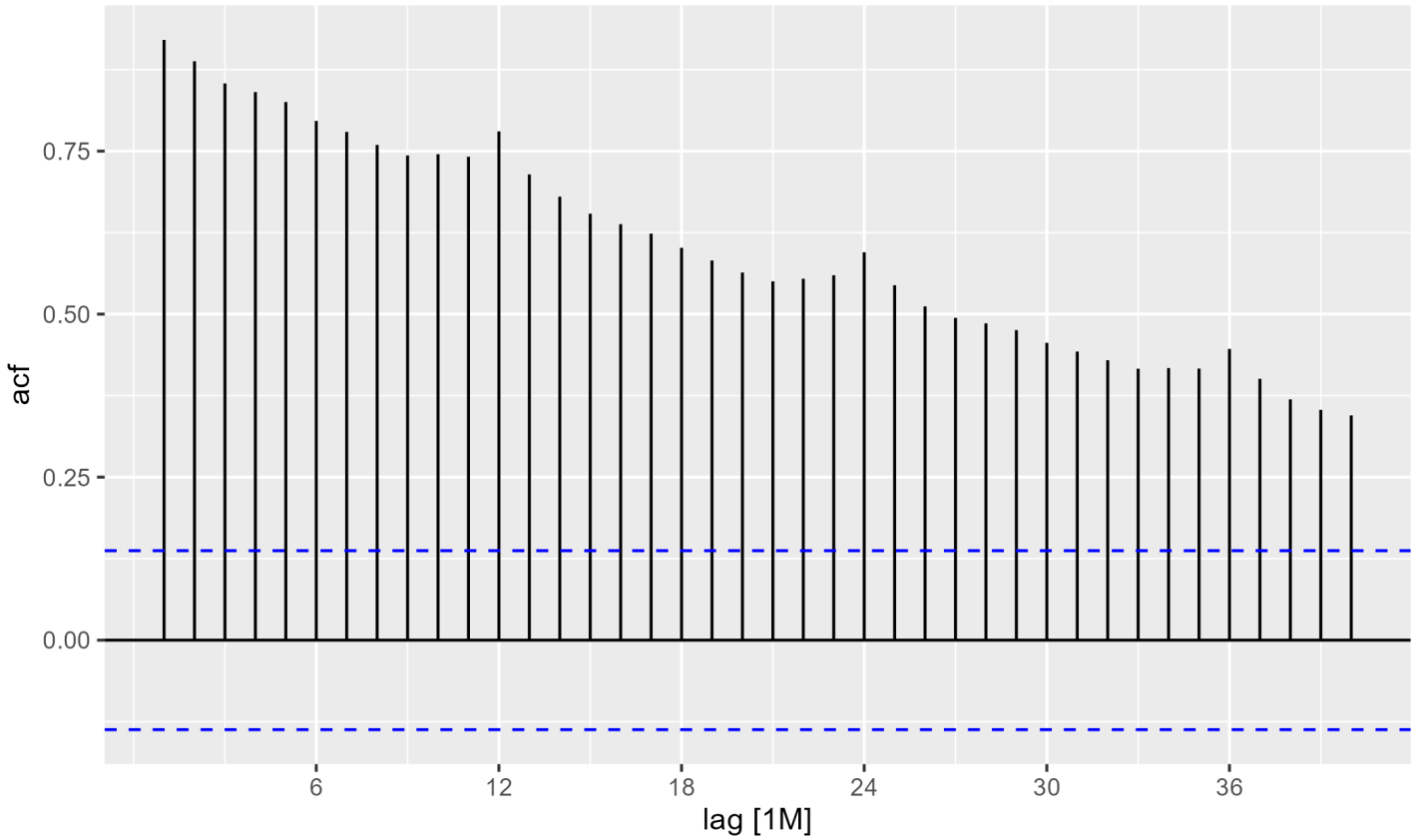

0.5
0.0
-0.5
acf
2
4
6
8
10
12
14
16
lag [1Q]

- We have high positive correlations for lags 4, 8, etc. because of the seasonality of the data.
- We have high negative correlation for lags 2, 6, 10, etc.
- The blue lines will be explained a little later.

Another example of autocorrelation

- Let us see the autocorrelation for the antidiabetic drug time series (A10).
- The beer timeseries contained quarterly data, A10 is monthly, so we might need more lags to see some pattern. This can be set with the *lag_max* parameter.

```
A10 |> ACF() |> autoplot()
A10 |> ACF(lag_max = 40) |> autoplot()
```

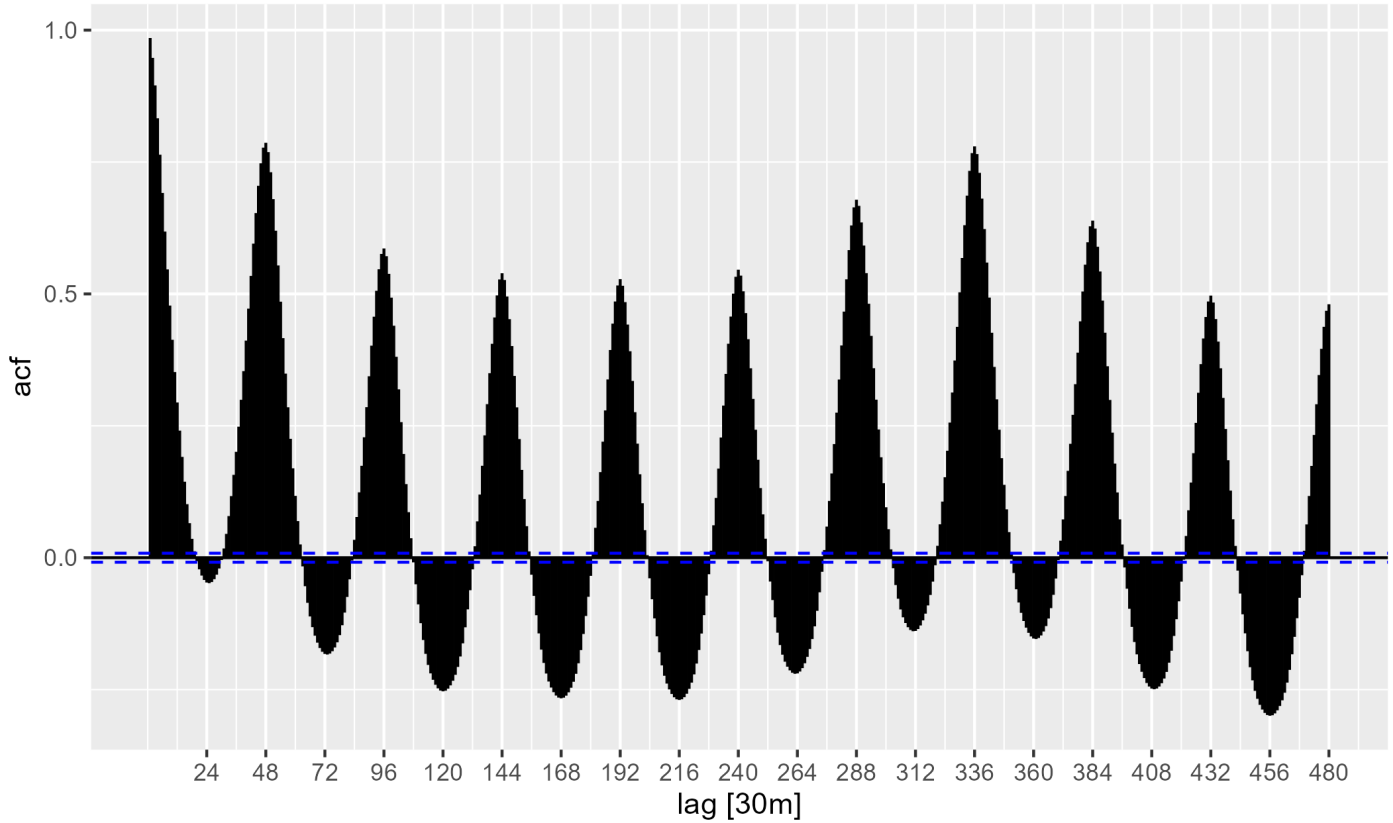


- What do you notice on this plot?

Autocorrelation for trends and seasonality

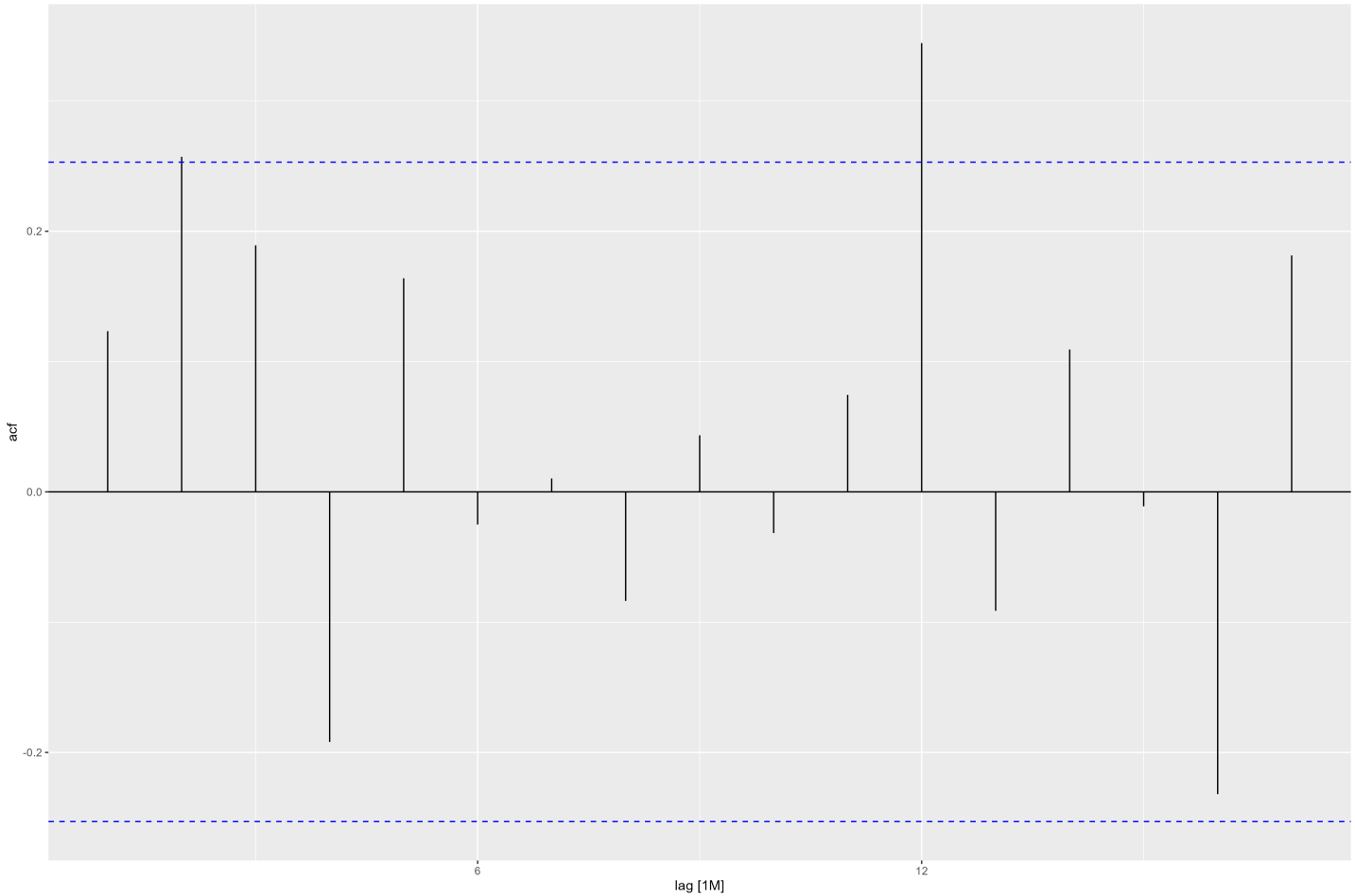
- In case of seasonal data, the autocorrelation will be higher for lags which are multiple of the frequency (see previous example with beer production).
- In case of trended data, the autocorrelation should be high for smaller lags (because data points close to each other have similar values). This autocorrelation decreases as the value of the lag increases.

```
vic_elec |> ACF(Demand, lag_max = 480) |> autoplot()
```



- Let us consider another example, a subset of the time series *aus_livestock*.

```
aus_livestock |>
  filter(Animal == "Pigs", State == "Victoria", year(Month) >= 2014) -> pigs

pigs |> autoplot() + geom_point()

pigs |> ACF() |> autoplot()
```



- What do you notice in the ACF plot?

White noise

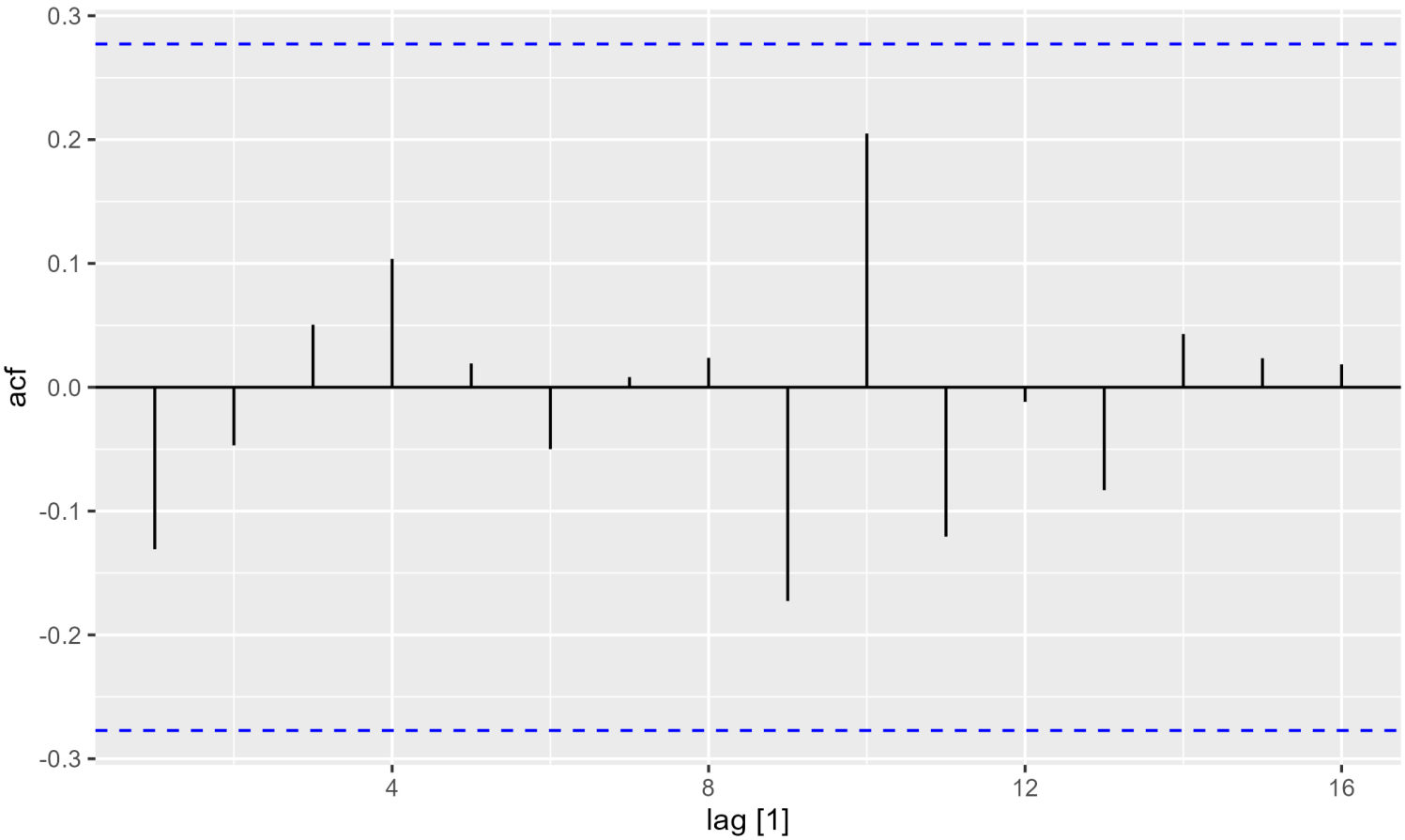
- A time series where the autocorrelation is close to 0 for most lag values is called *white noise*.
- More precisely, we expect 95% of autocorrelation values to be within the values $\pm 1.96 * \sqrt{T}$, where T is the number of observations.
- The values $\pm 1.96 * \sqrt{T}$ are plotted as the blue lines on the ACF plot.
- When all autocorrelation values are between the blue lines or at most 5% of the values are outside of them, the data is probably white noise.

- The *pigs* time series has 2 autocorrelations outside the blue lines: one at lag 12 and a very small one at lag 2.

- Since we only have 16 lags, this suggests that the data is not white noise. Moreover, since the largest lag is at 12, it might actually show some kind of seasonality.

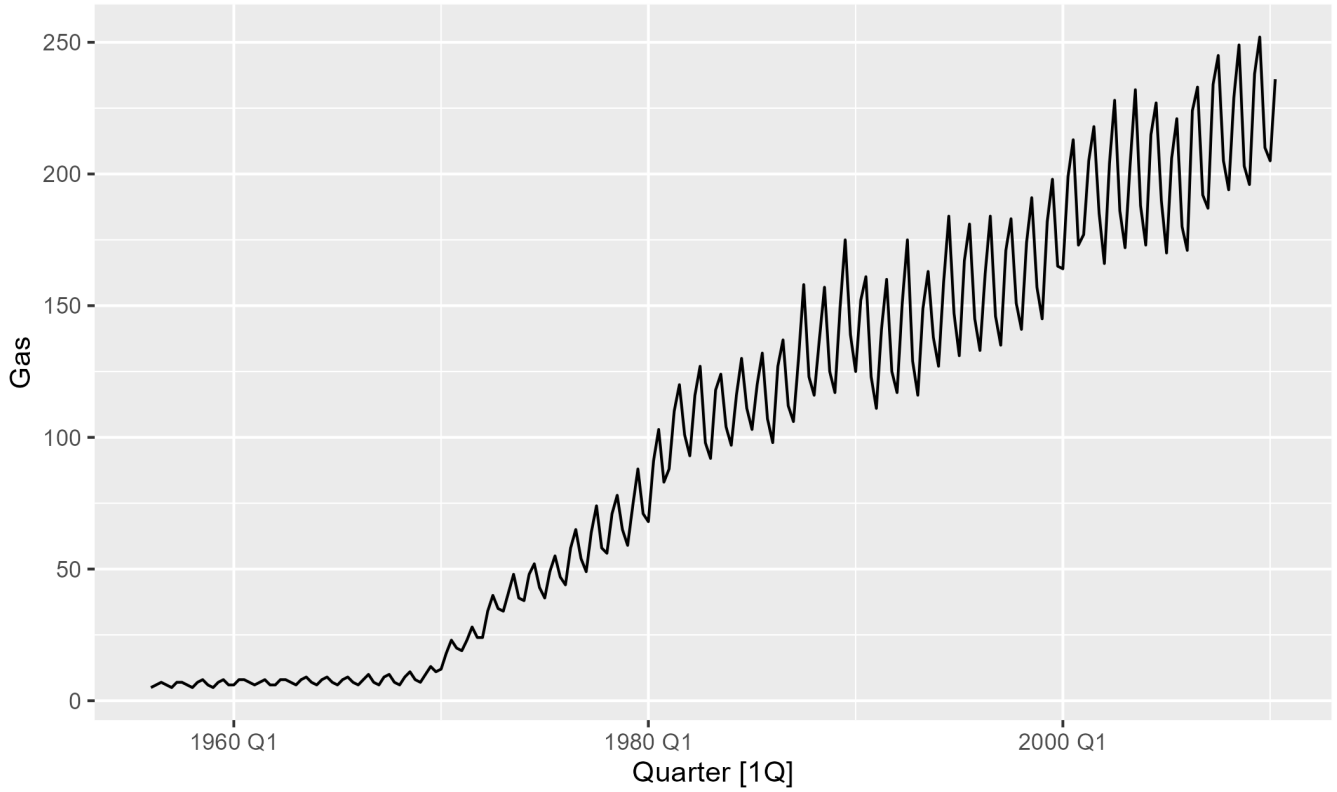
- Let us generate a time series with randomly chosen normal values.

```
set.seed(39)
wn <- tsibble(sample = 1:50, wn = rnorm(50), index = sample)
wn |> autoplot()
wn |> ACF() |> autoplot()
```


0.3
0.2
0.1
0.0
-0.1
-0.2
-0.3
acf
4
8
12
16
lag [1]

Stationary time series

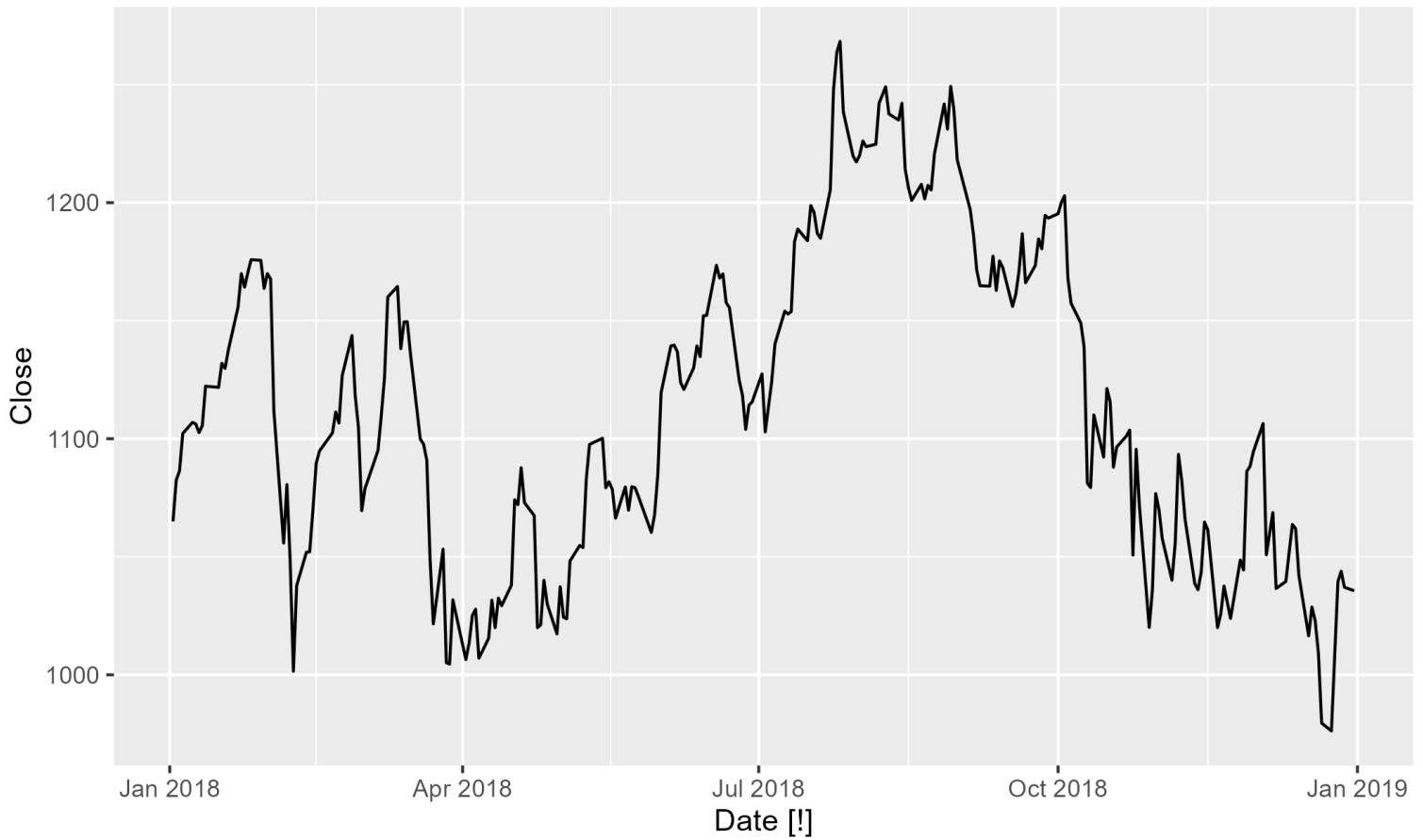
- A time series is *stationary* if its observations are not dependent on time and summary statistics computed for it (ex. mean and variance) are not dependent on time.
- In other words, a time series is stationary, if
  - it has no trend (it is roughly horizontal)
  - it has constant variance (it is homoscedastic)
  - it has no predictable patterns (no seasons).
- **Obs.** white noise is stationary.
- Some methods require time series to be stationary.

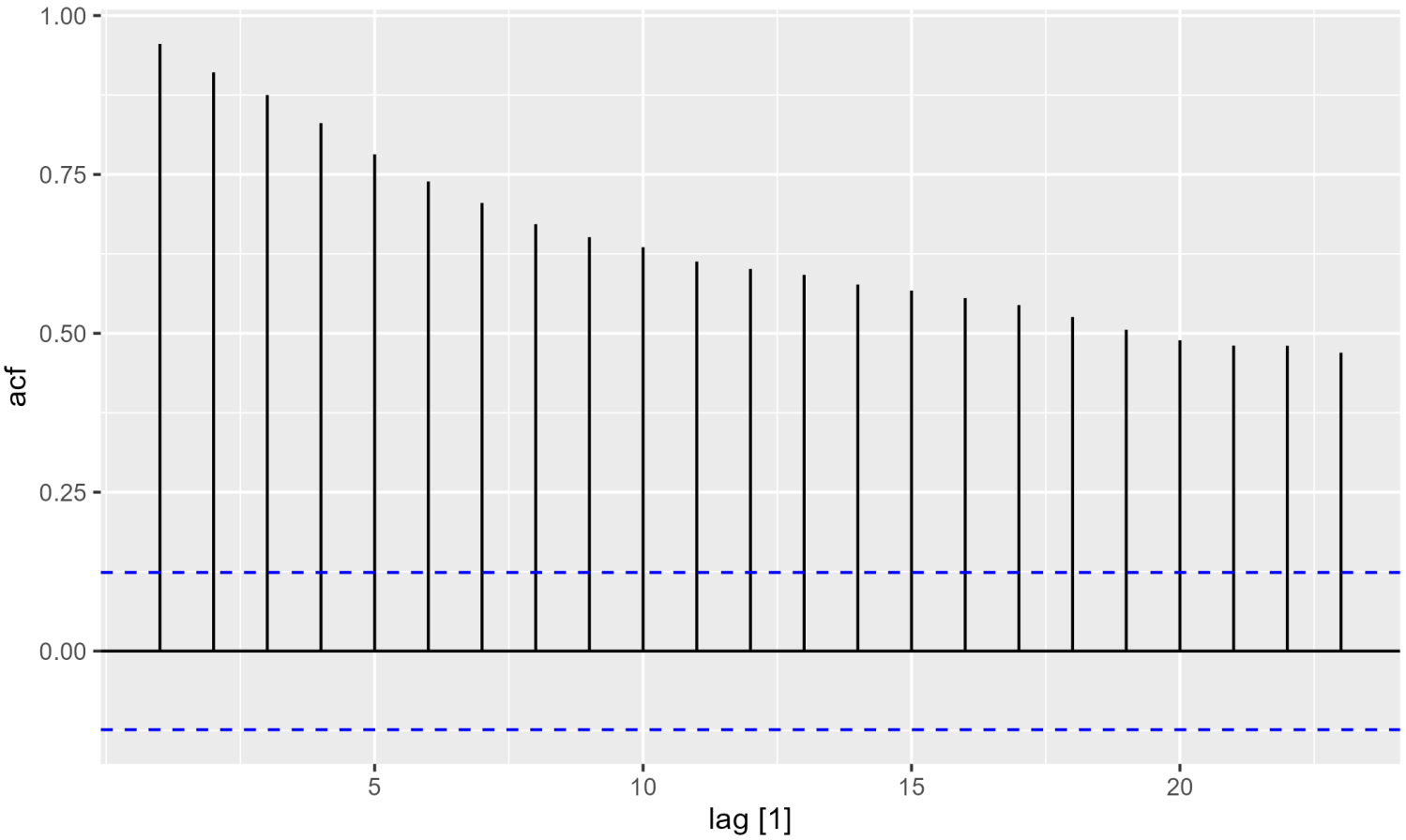


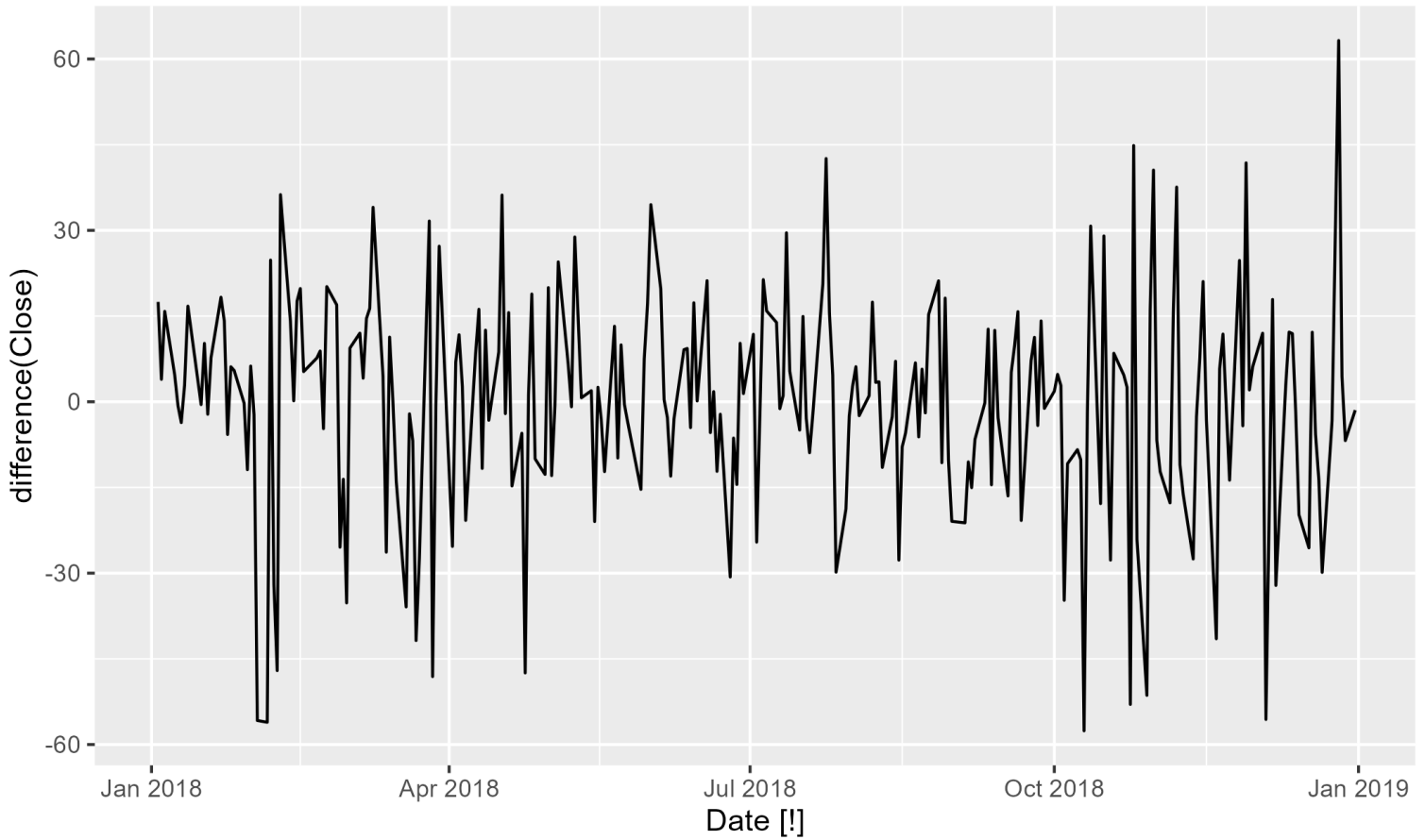
- Example of a time series about Australian gas production (which is as non-stationary as possible).

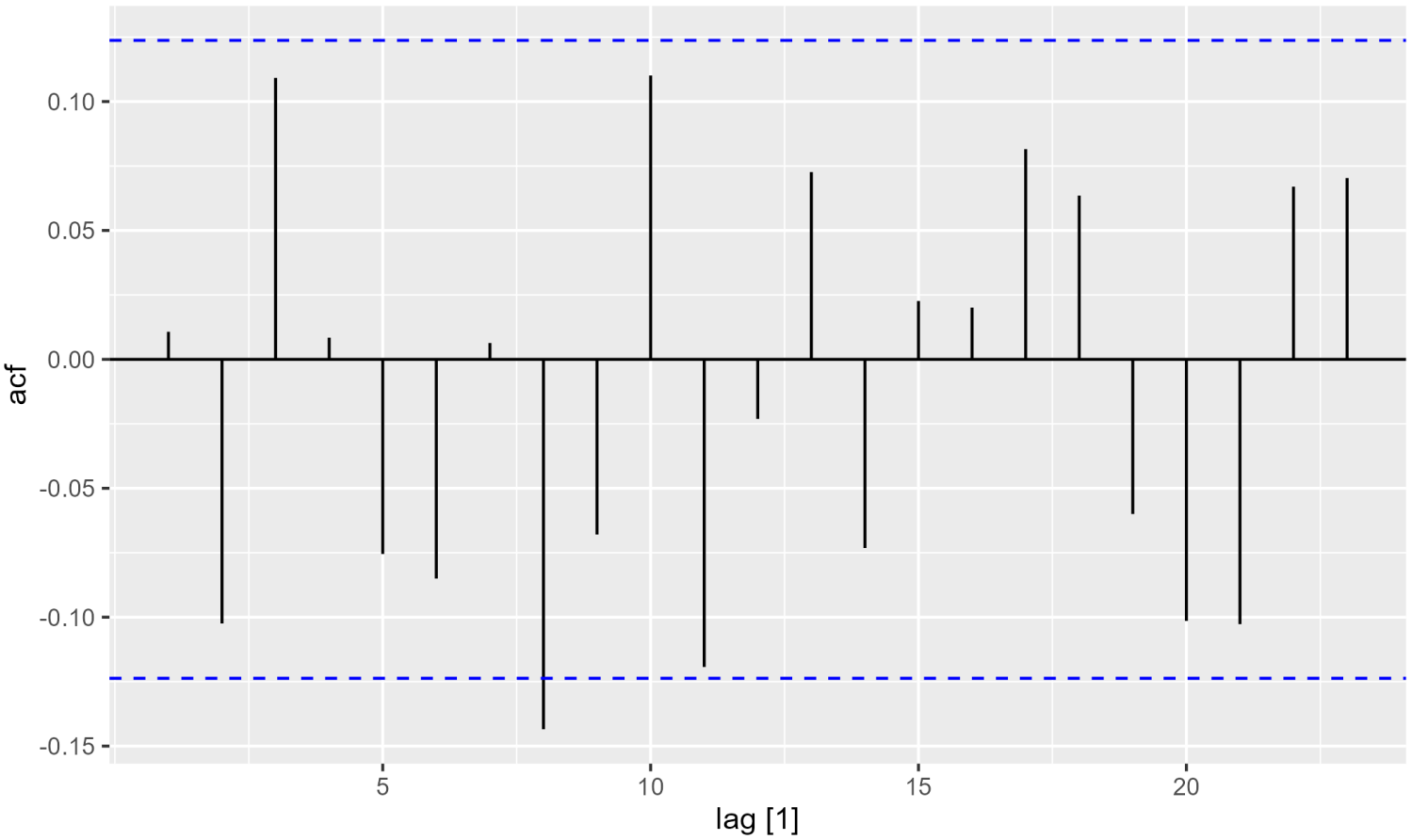
- We can look at the ACF plot as well, stationary time series have ACF plots which drop quickly to zero, while in case of non-stationary time series the decrease in general is gradual. For non-stationary data the first autocorrelation also tends to be a large, positive value.

```
gafa_stock
google <- gafa_stock |>
  filter(Symbol == "GOOG", year(Date) == 2018)

google |> autoplot()

google |> ACF() |>
    autoplot()
```


Close
1200
1100
1000
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Date [!]


1.00
0.75
0.50
0.25
0.00
acf
5
10
15
20
lag [1]


60
30
0
-30
-60
difference(Close)
Jan 2018
Apr 2018
Jul 2018
Oct 2018
Jan 2019
Date [!]


0.10
0.05
0.00
-0.05
-0.10
-0.15
acf
5
10
15
20
lag [1]

Statistical tests

- While in some cases it is very simple to identify whether you have trend and/or season from the plot of the data, sometimes it might be more complicated (or you just need a method that can be automated for a large number of time series).
- We can use statistical tests:
  - *Augmented Dickey-Fuller* test (H_0 : non-stationary)
  - *Kwiatkowski-Philips-Schmidt-Shin* (H_0 : stationary) - this is preferred in forecasting, since we only want to transform if there is strong evidence that we need to do it.

The Kwiatkowski-Philips-Schmidt-Shin (KPSS) test

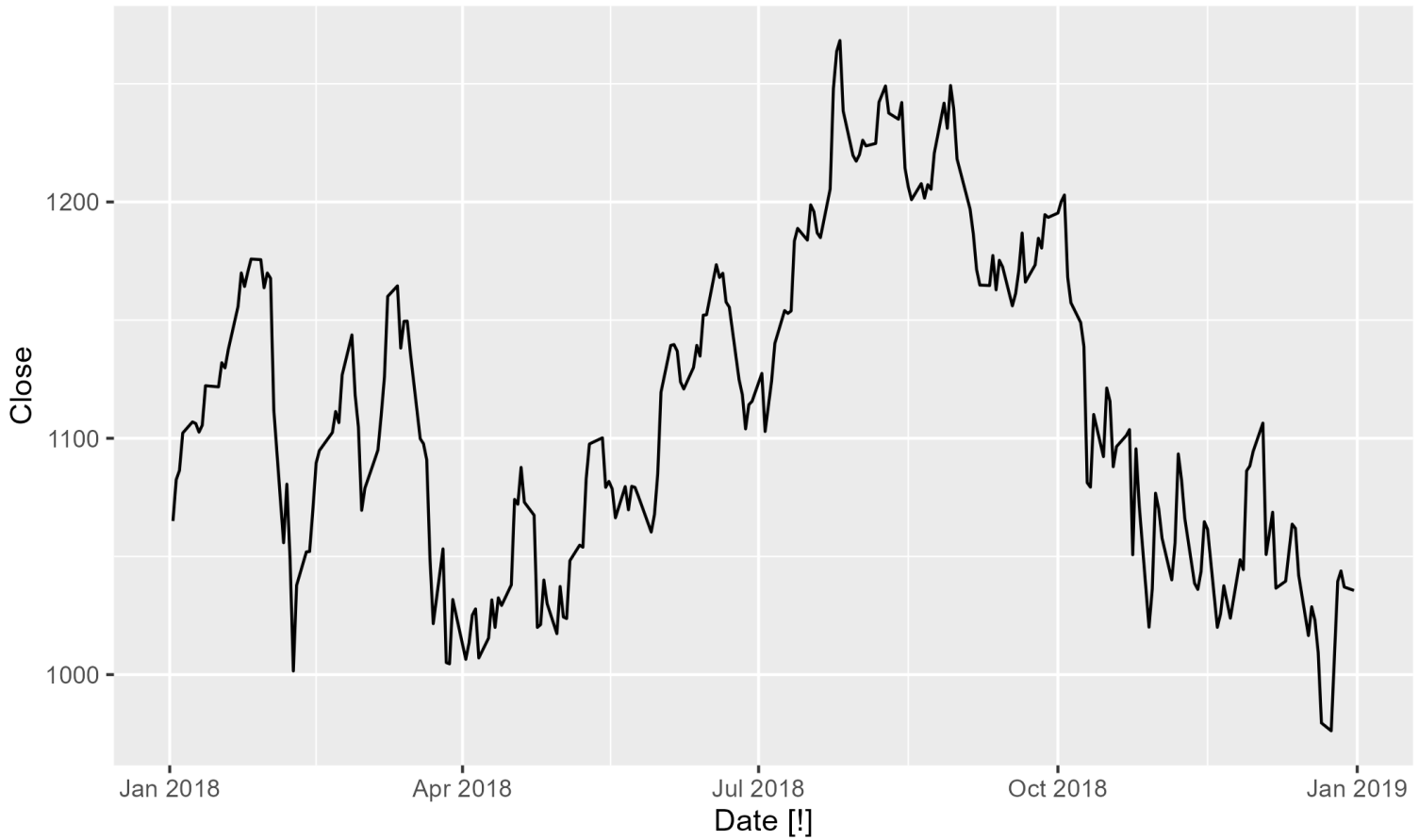
- The KPSS test is a *stationarity test*.
- **The null hypothesis:** The time series is stationary
- **The alternative hypothesis:** The time series is not stationary (it contains something called a *unit root*)
- Interpretation: If the p-value is less than the pre-defined significance level (in general 0.05) the null hypothesis is rejected and the time series is considered to be non-stationary.
- The KPSS test can be run in R using the *features* function and passing it as parameter that we want to compute the *unitroot_kpss* feature).

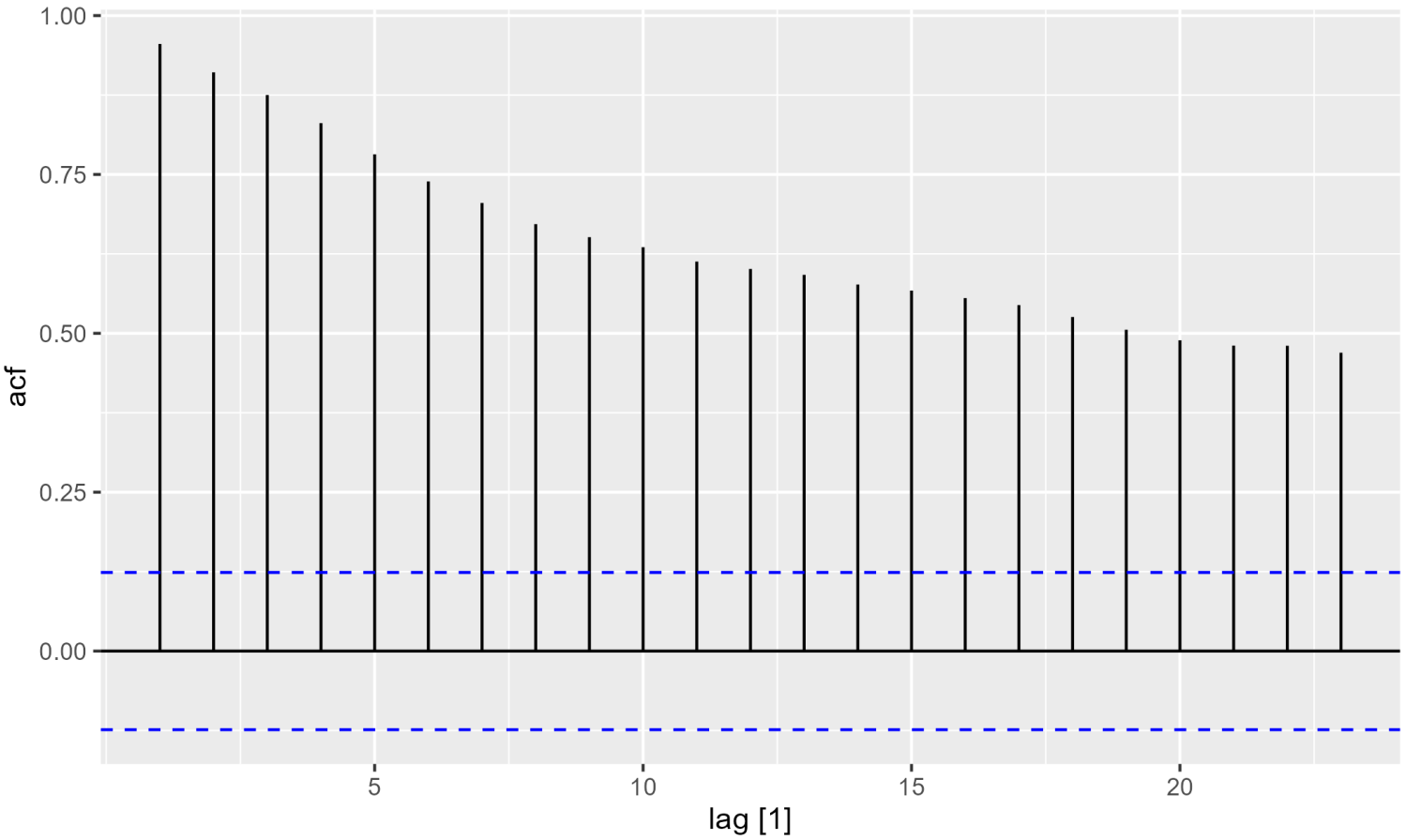
The Augmented Dickey-Fuller (ADF) test

- The ADF test is a *unit root* test.
- **The null hypothesis:** The time series is non-stationary (there is a unit root in it, which makes it non-stationary)
- **The alternative hypothesis:** The time series is stationary.
- Interpretation: If the p-value is less than the pre-defined significance level (in general 0.05) the null hypothesis is rejected and the time series is considered to be stationary.
- The ADF test can be run in R with the *adf.test* function from the *tseries* package.

Example 1

- Let us look again at the closing price for Google stocks in 2018




1.00
0.75
0.50
0.25
0.00
acf
5
10
15
20
lag [1]

KPSS test

```
# A tibble: 1 x 3
  Symbol kpss_stat kpss_pvalue
  <chr>       <dbl>       <dbl>
1 GOOG        0.573      0.0252
>
```

- The important value is the *kpss_pvalue* in the last column, which is 0.0252 $\Rightarrow$ we can reject the null hypothesis, the time series is not stationary.

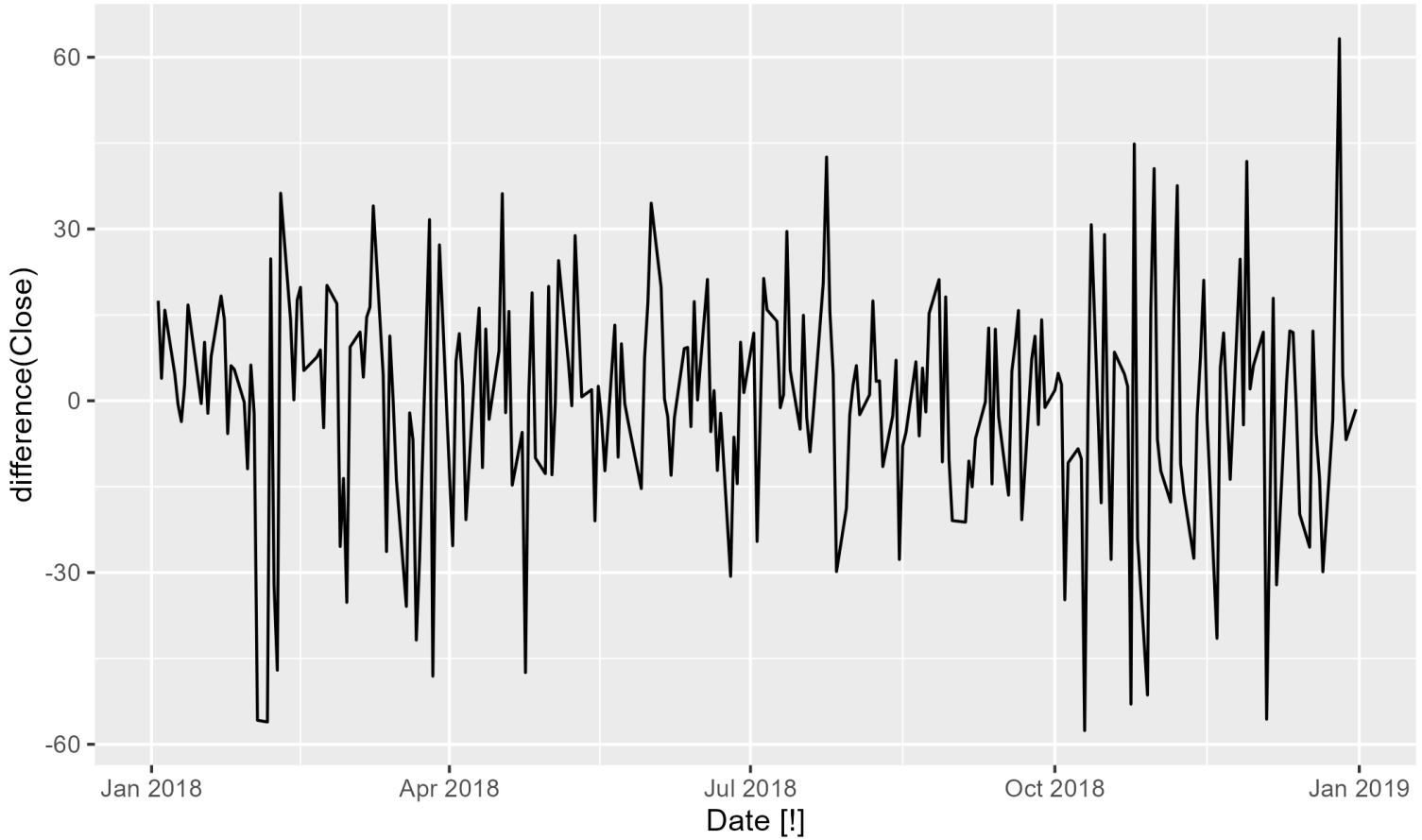
ADF test

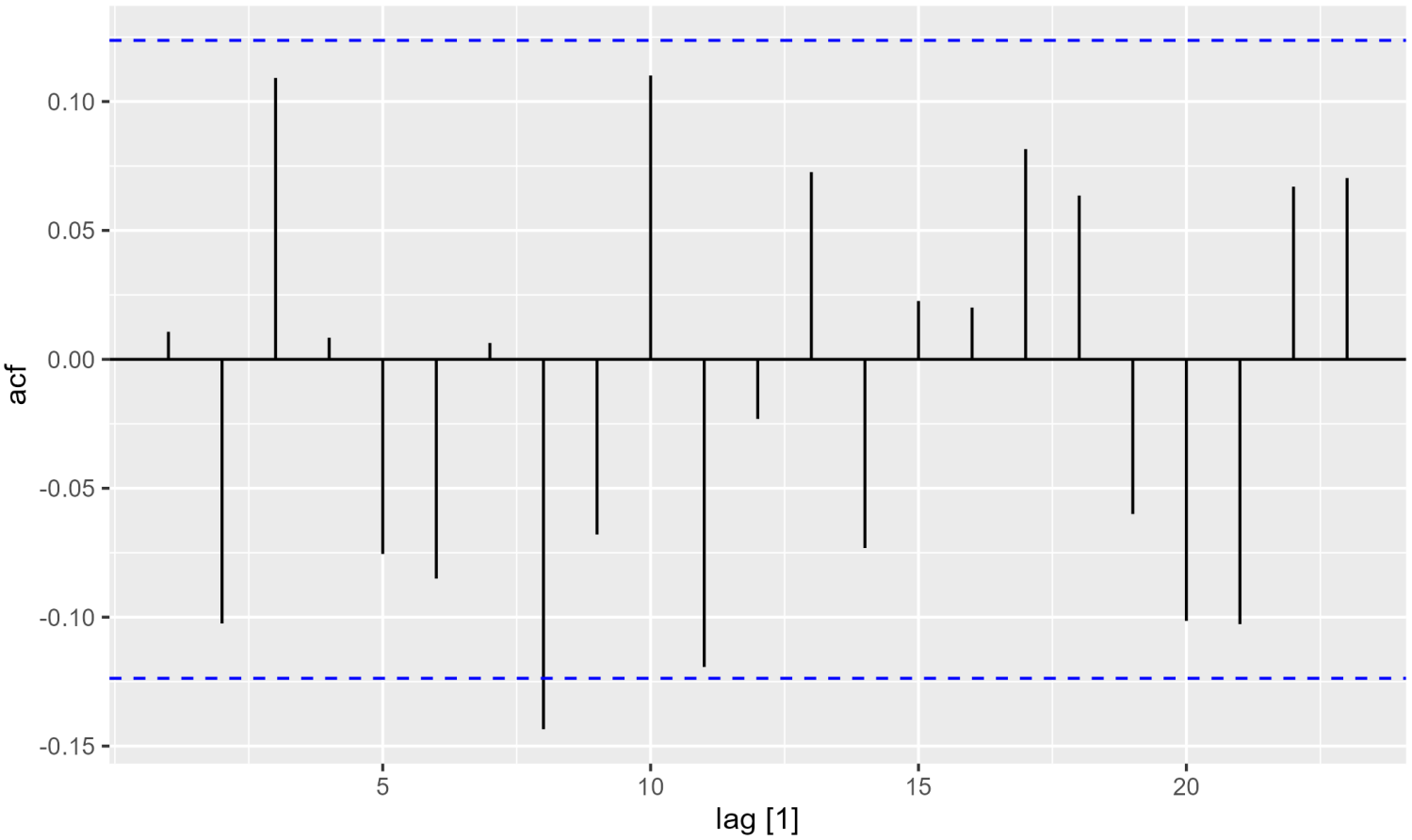
```
        Augmented Dickey-Fuller Test

data:  google$Close
Dickey-Fuller = -1.8265, Lag order = 6, p-value = 0.6489
alternative hypothesis: stationary
```

- The important value is the *p-value*, which is 0.6489 $\Rightarrow$ we cannot reject the null hypothesis, the time series is not stationary.
- For this example, both the visual inspection and the statistical tests conclude that the time series is not stationary.

Example 2

- Difference of Google closing stock prices for 2018.




0.10
0.05
0.00
-0.05
-0.10
-0.15
acf
5
10
15
20
lag [1]

KPSS test

```
# A tibble: 1 x 3
  Symbol kpss_stat kpss_pvalue
  <chr>       <dbl>       <dbl>
1 GOOG       0.0955         0.1
> |
```

- The important value is the *kpss_pvalue* in the last column, which is 0.1 $\Rightarrow$ we cannot reject the null hypothesis (it is greater than 0.05), the time series is stationary.

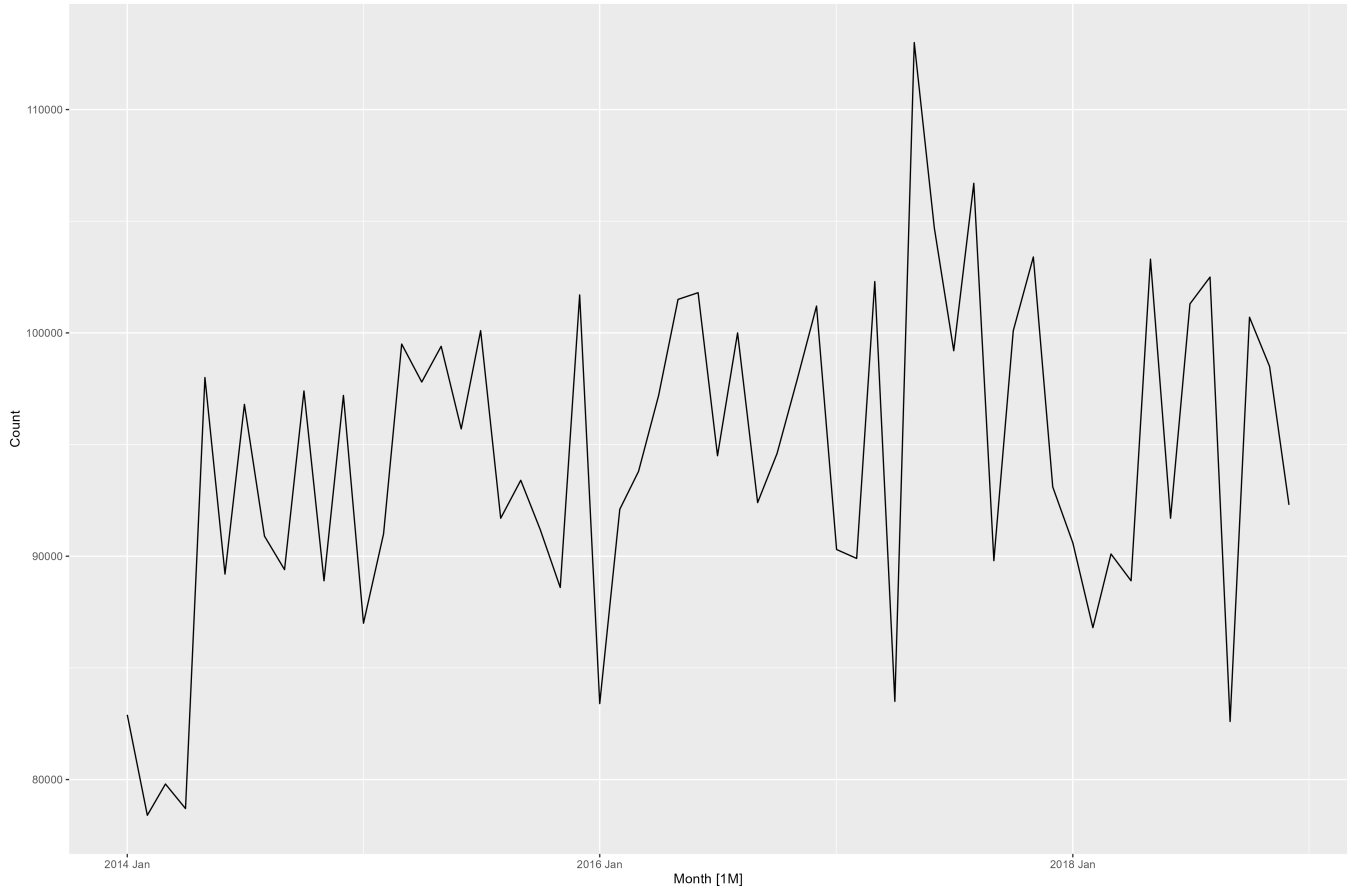
ADF test

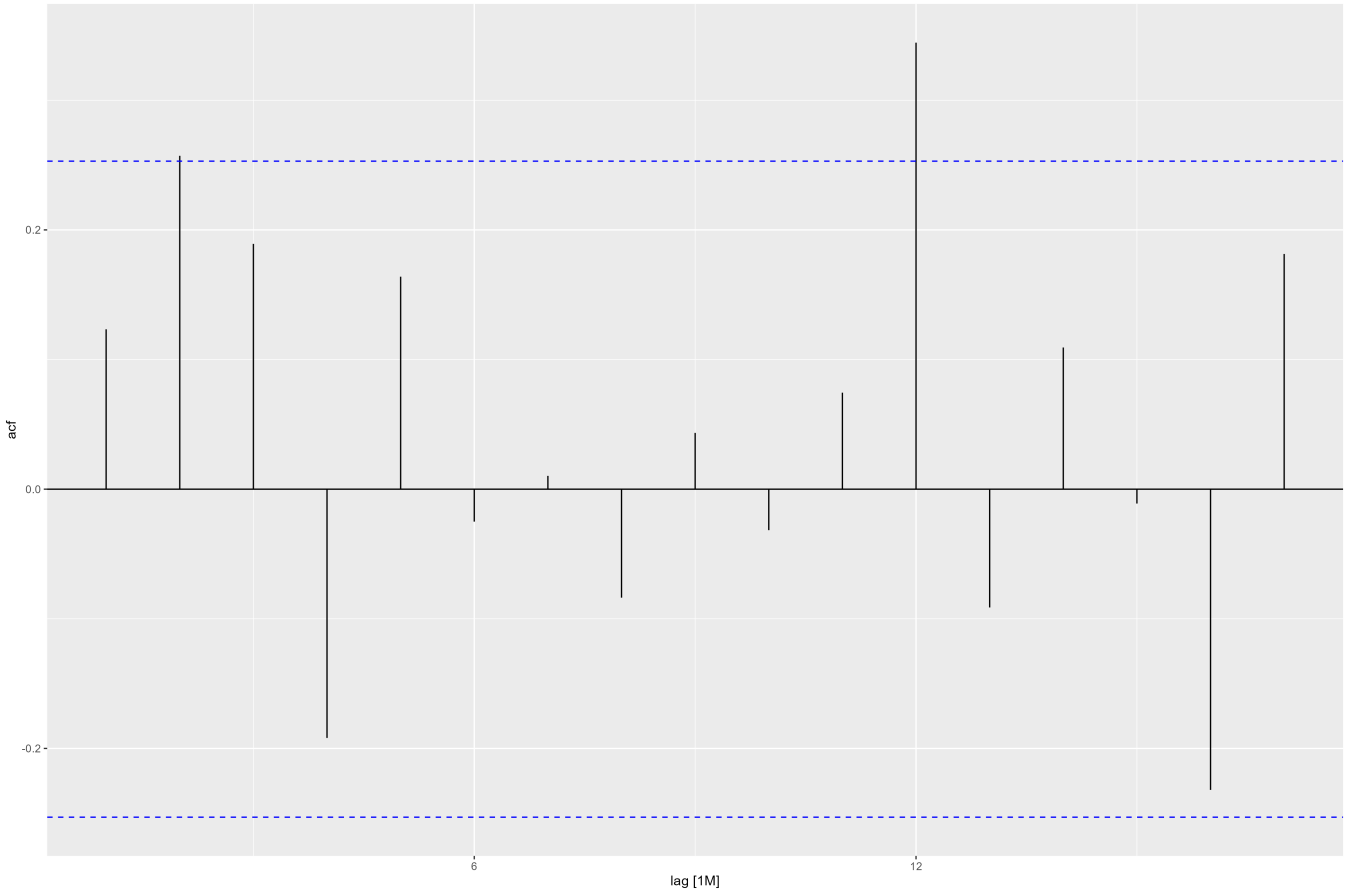
```
        Augmented Dickey-Fuller Test

data:  difference(google$Close)[2:251]
Dickey-Fuller = -6.4714, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(difference(google$Close)[2:251]) :
  p-value smaller than printed p-value
> |
```

- The important value is the *p-value*, which is 0.01 $\Rightarrow$ we can reject the null hypothesis, the time series is stationary.
- Actually, you can see a message that the real p-value might be smaller than the reported one, but this is not a problem.
- For this example, both the visual inspection and the statistical tests conclude that the time series is stationary.

Example 3

- Monthly total number of pigs slaughtered in Victoria, 2014-2018




0.2
0.0
-0.2
acf
6
12
lag [1M]

KPSS test

```
# A tibble: 1 × 4
  Animal State    kpss_stat kpss_pvalue
  <fct>  <fct>        <dbl>       <dbl>
1 Pigs   Victoria     0.491      0.0438
```

- The important value is the *kpss_pvalue* in the last column, which is 0.0438 $\Rightarrow$ we can reject the null hypothesis, the time series is non-stationary.

ADF test

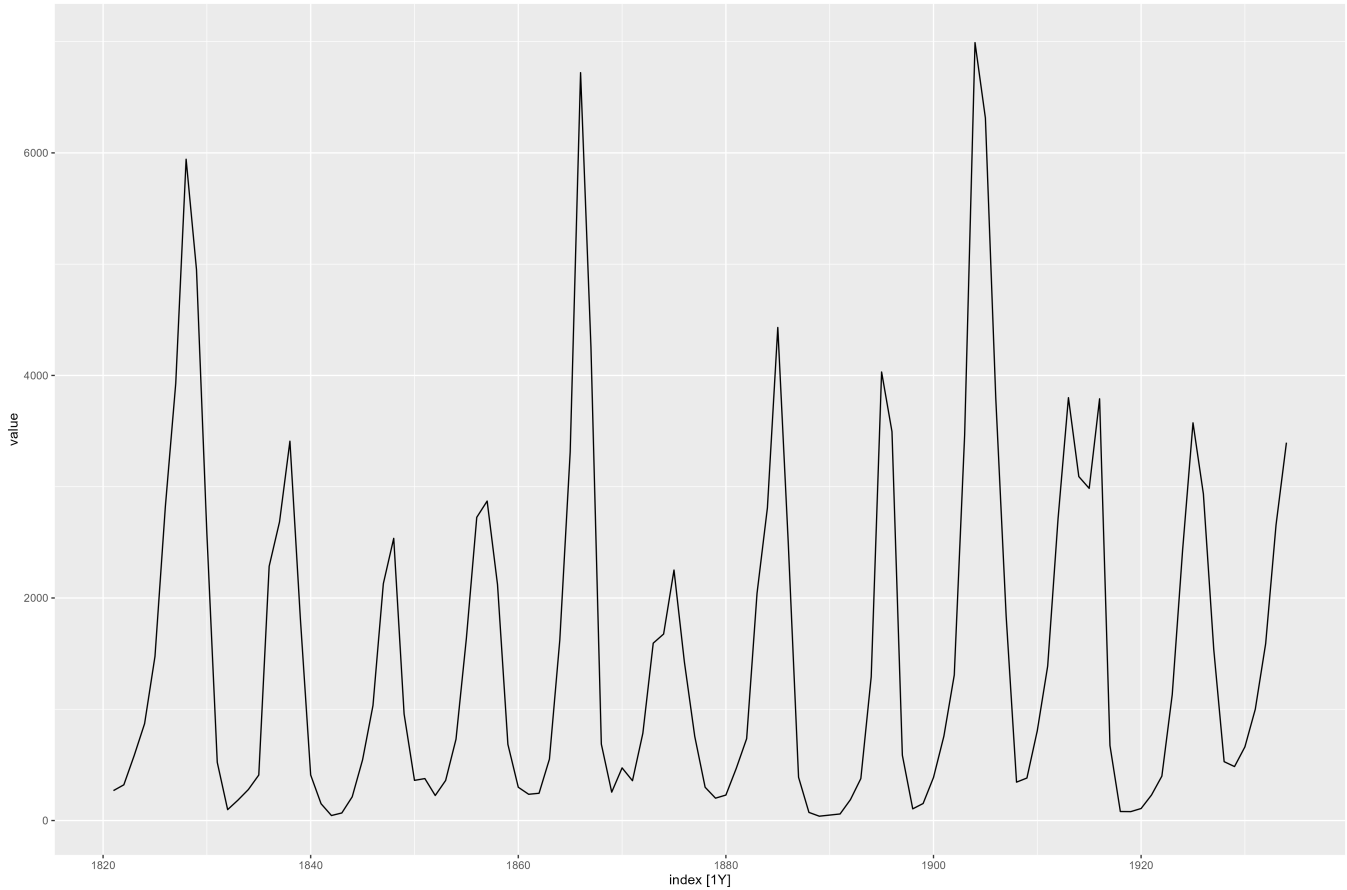
```
        Augmented Dickey-Fuller Test

data:  pigs$Count
Dickey-Fuller = -5.2471, Lag order = 3, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(pigs$Count) : p-value smaller than printed p-value
> |
```

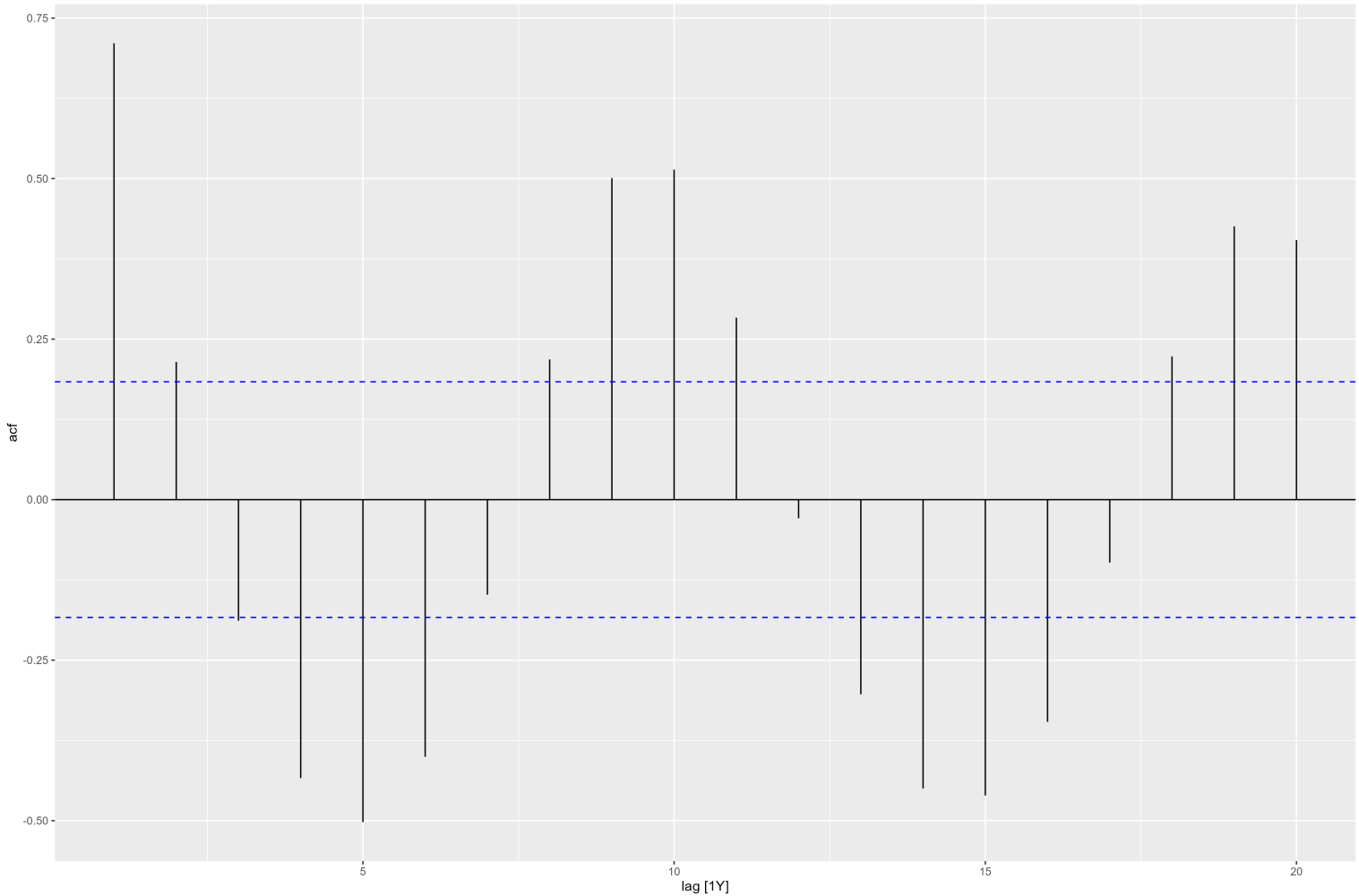
- The important value is the *p-value*, which is 0.01 $\Rightarrow$ we can reject the null hypothesis, the time series is stationary.

- When the two tests do not agree, we will consider the result of the KPSS test and we will consider that this time series is non-stationary.

Example 4

- Yearly lynx population




0.75
0.50
0.25
0.00
-0.25
-0.50
acf
5
10
15
20
lag [1Y]

KPSS test

```
# A tibble: 1 x 2
  kpss_stat kpss_pvalue
      <dbl>       <dbl>
1    0.0701         0.1
> |
```

- The important value is the *kpss_pvalue* in the last column, which is 0.1 $\Rightarrow$ we cannot reject the null hypothesis (it is greater than 0.05), the time series is stationary.

ADF test

```
        Augmented Dickey-Fuller Test

data:  lynxts$value
Dickey-Fuller = -6.3068, Lag order = 4, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(lynxts$value) : p-value smaller than printed p-value
>
```

- The important value is the *p-value*, which is 0.01 $\Rightarrow$ we can reject the null hypothesis, the time series is stationary.
- Both tests agree that the time series is stationary.
- **Obs:** If a time series is white noise (we can see that on the ACF plot) it is certainly stationary. But if it is stationary, it does not mean that it is white noise.

- If we have identified that our time series is not stationary, and we need a stationary time series, we can do the following:
  - Use a transformation to stabilize the variance (transformations will be discussed later).
  - Use differencing to stabilize the mean

Differencing

- Differencing is a transformation of the time series, we simply subtract from each observation the previous one:

$$y'_t = y_t - y_{t-1}$$

- Differencing can be repeated several times, we get a stationary time series. The number of repetitions is called the *difference order*.

- **Obs.** As in case of any transformation, if time series predictions are performed on transformed data, the result needs to be transformed back, in order to have values on the original scale.
- Assuming that you used a differenced time series and predicted y'_{t+1}, the actual prediction can be computed in the following way:

$$y'_{t+1} = y_{t+1} - y_t \quad \rightarrow \quad y_{t+1} = y'_{t+1} + y_t$$

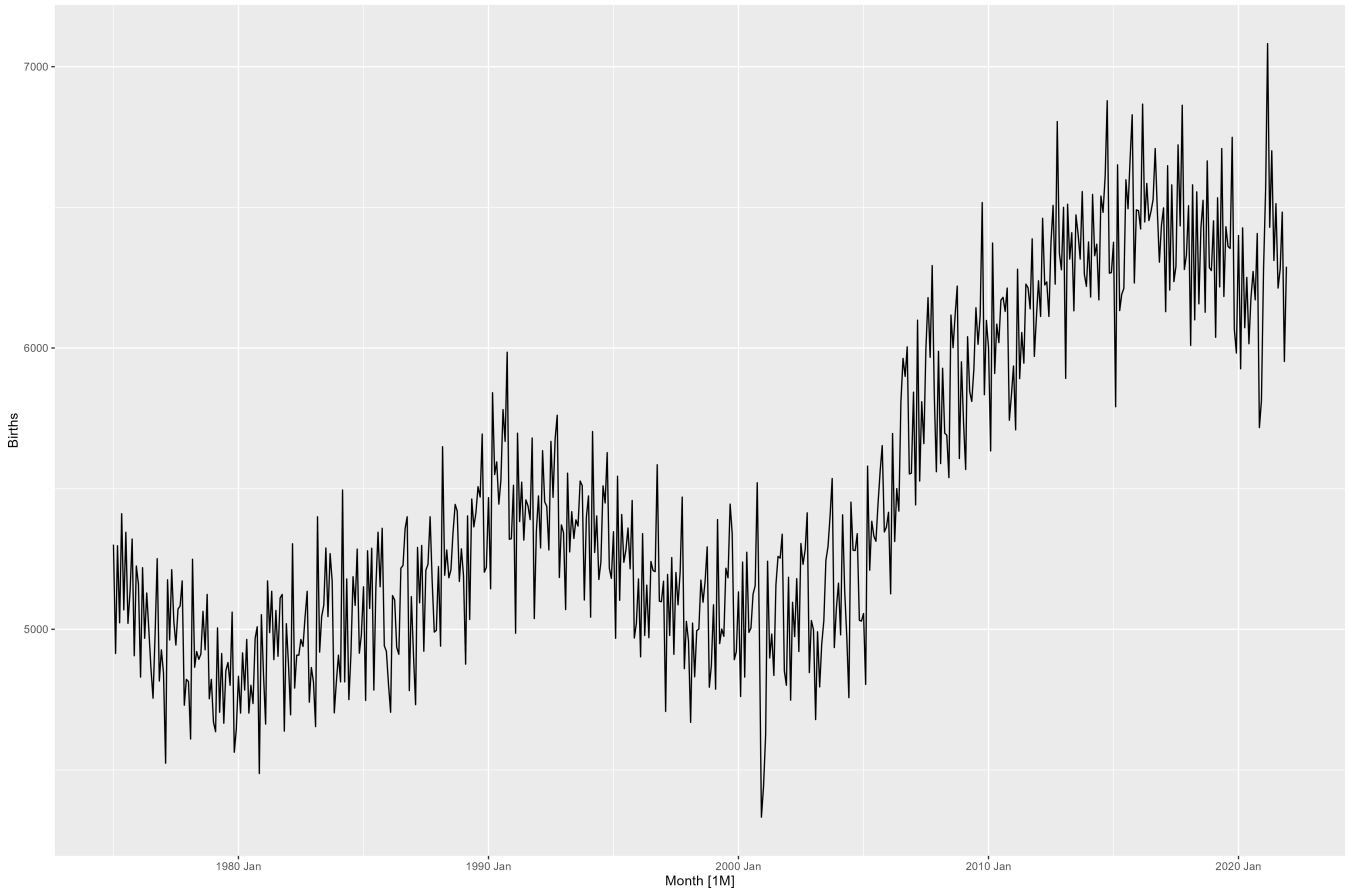
- So, the actual prediction is the last observation plus the predicted difference.
- If multiple future points were predicted, the process is performed for each point.

- In R we can compute the difference of a time series with the *difference* function.

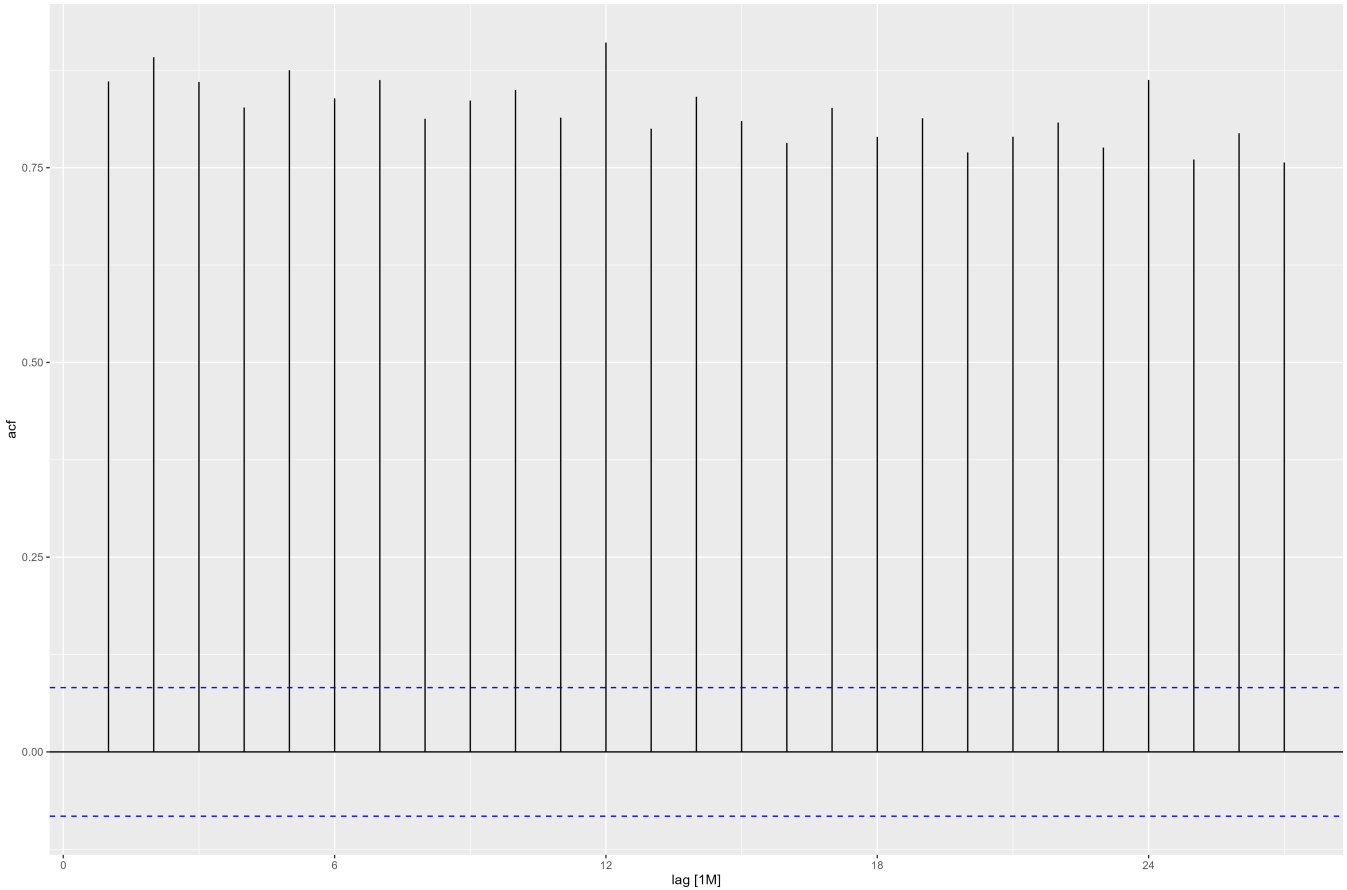
- Let us consider an example on the

```
?aus_births

aus_births |> filter(State == "VIC") -> vic_births

vic_births |> autoplot()
vic_births |> ACF() |> autoplot()

vic_births |> features(Births, unitroot_kpss)
adf.test(vic_births$Births)

vic_births |> mutate(BirthDiff = difference(Births)) -> vic_births2
vic_births2
vic_births2 |> autoplot(BirthDiff)
vic_births2 |> ACF(BirthDiff) |> autoplot()

vic_births2 |> features(BirthDiff, unitroot_kpss)
vic_births2 |> filter(is.na(BirthDiff) == FALSE) -> vic_births3
vic_births3
adf.test(vic_births3$BirthDiff)
```

- Initial data (Births in Victoria state in Australia)



- The corresponding ACF plot



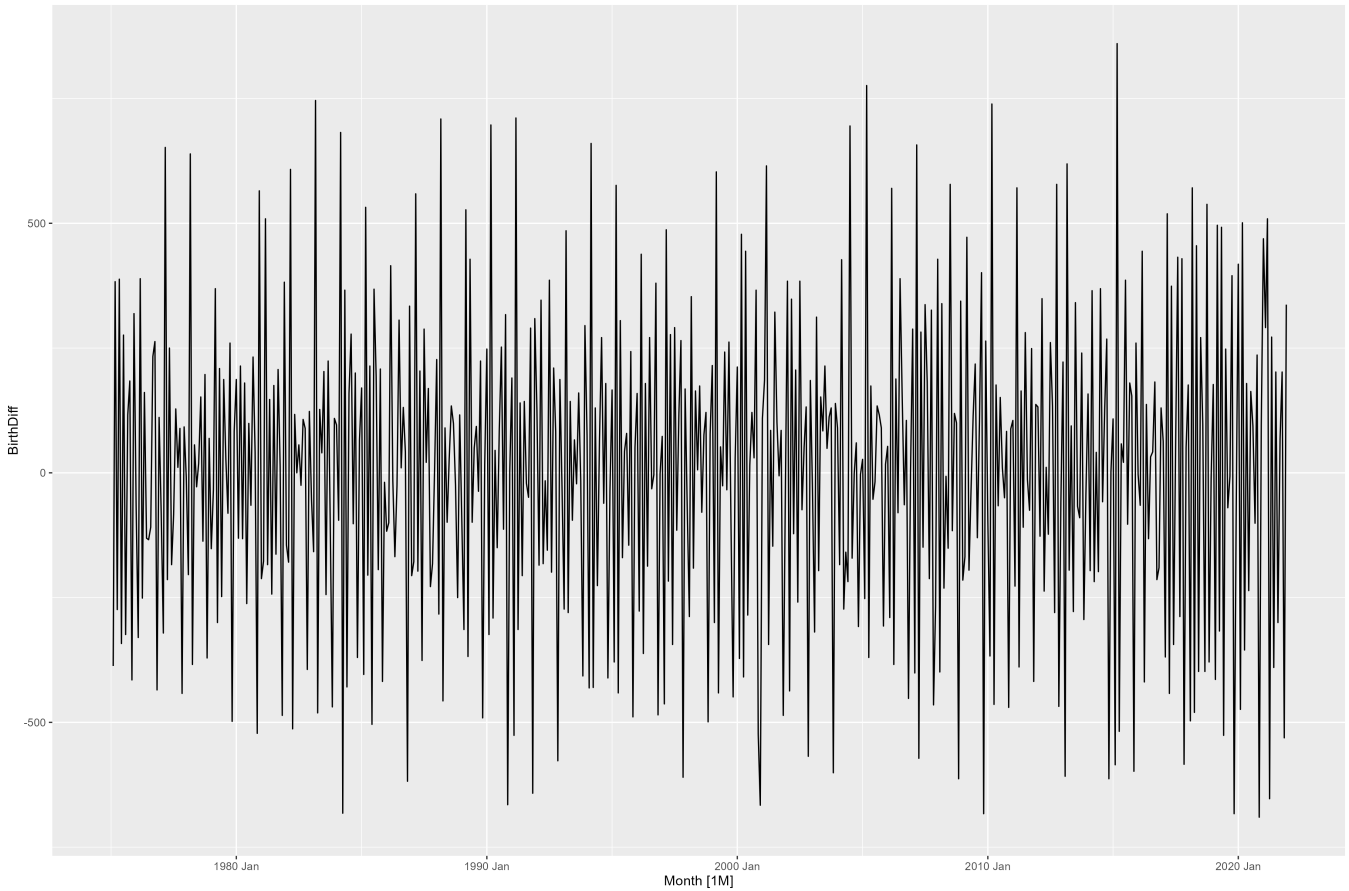
- And the statistical tests

```
# A tibble: 1 x 3
  State kpss_stat kpss_pvalue
  <chr>     <dbl>       <dbl>
1 VIC        6.12        0.01
> 
```

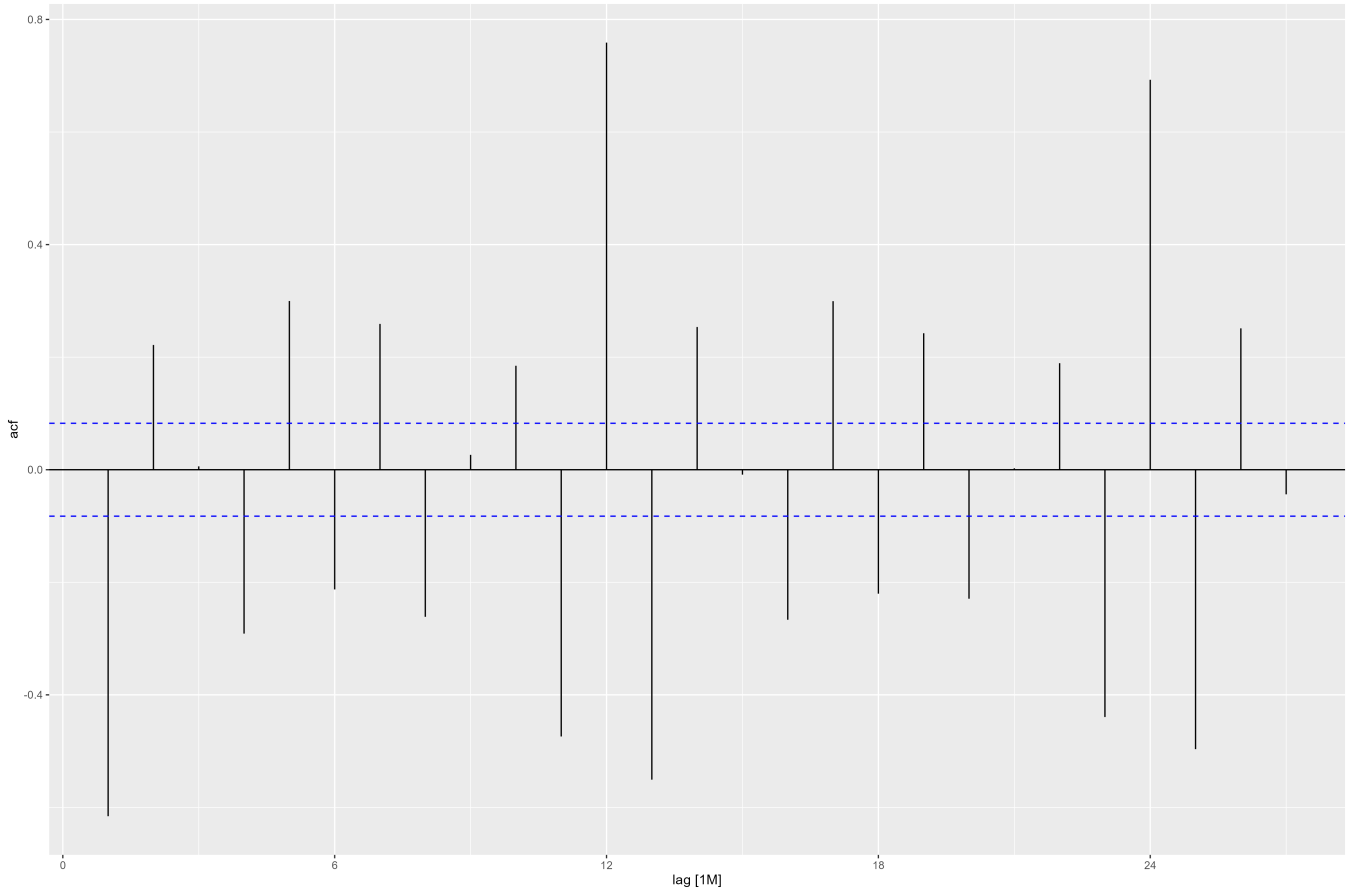
```
        Augmented Dickey-Fuller Test

data:  vic_births$Births
Dickey-Fuller = -2.7453, Lag order = 8, p-value = 0.2628
alternative hypothesis: stationary
```

- Everything shows that the data is non-stationary.

- The differenced data



- The corresponding ACF plot



- And the statistical tests

```
# A tibble: 1 x 3
  State kpss_stat kpss_pvalue
  <chr>      <dbl>       <dbl>
1 VIC       0.0367         0.1
>
```

```
        Augmented Dickey-Fuller Test

data:  vic_births3$BirthDiff
Dickey-Fuller = -16.104, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(vic_births3$BirthDiff) : p-value smaller than printed p-value
```

- Everything shows that the data is stationary (even if it is not white noise).

Seasonal differencing

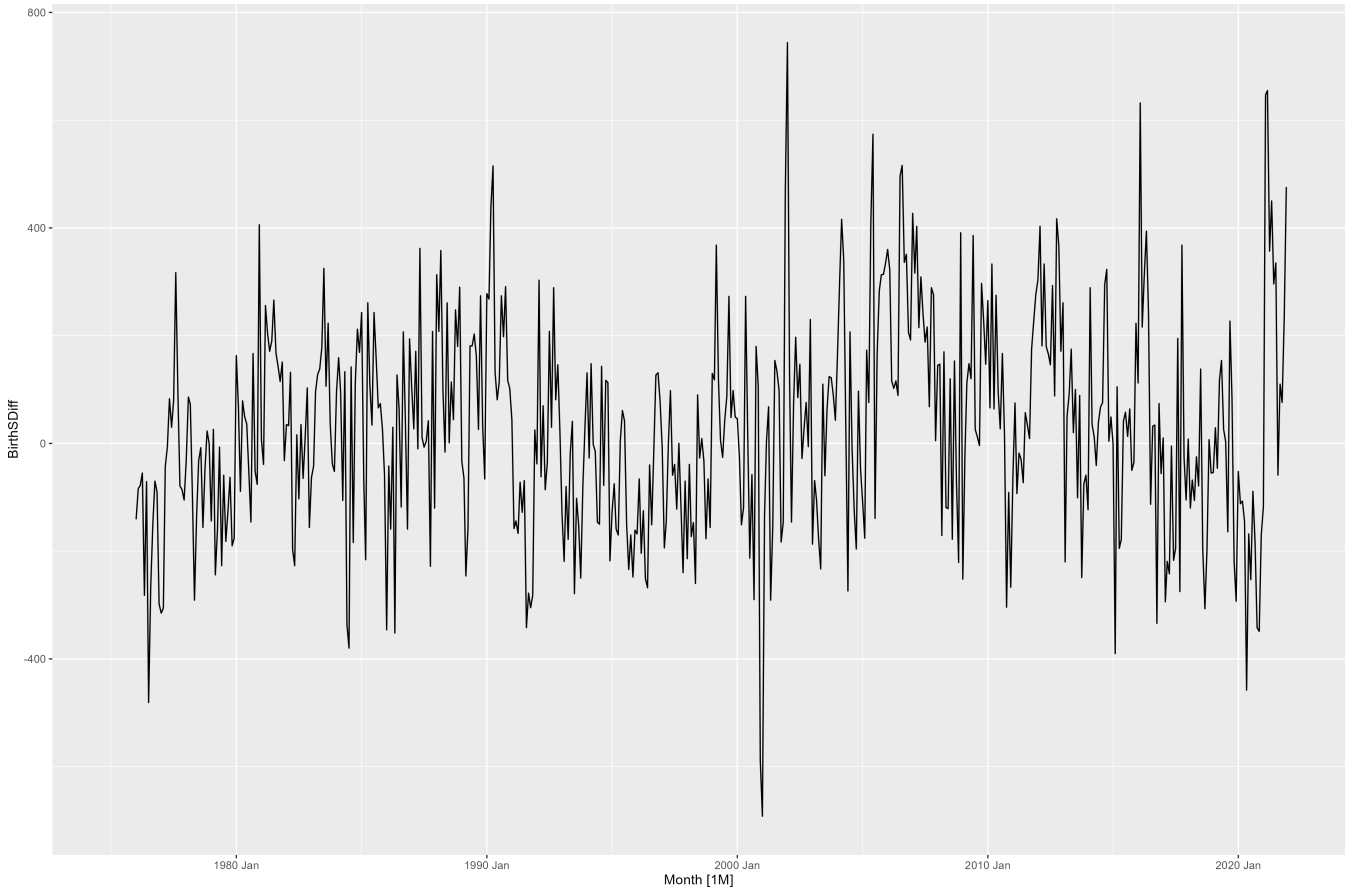
- A seasonal difference is the difference between an observation and the previous observation from the same season (so lag is equal to the length of a season, denoted by m):

$$y'_t = y_t - y_{t-m}$$

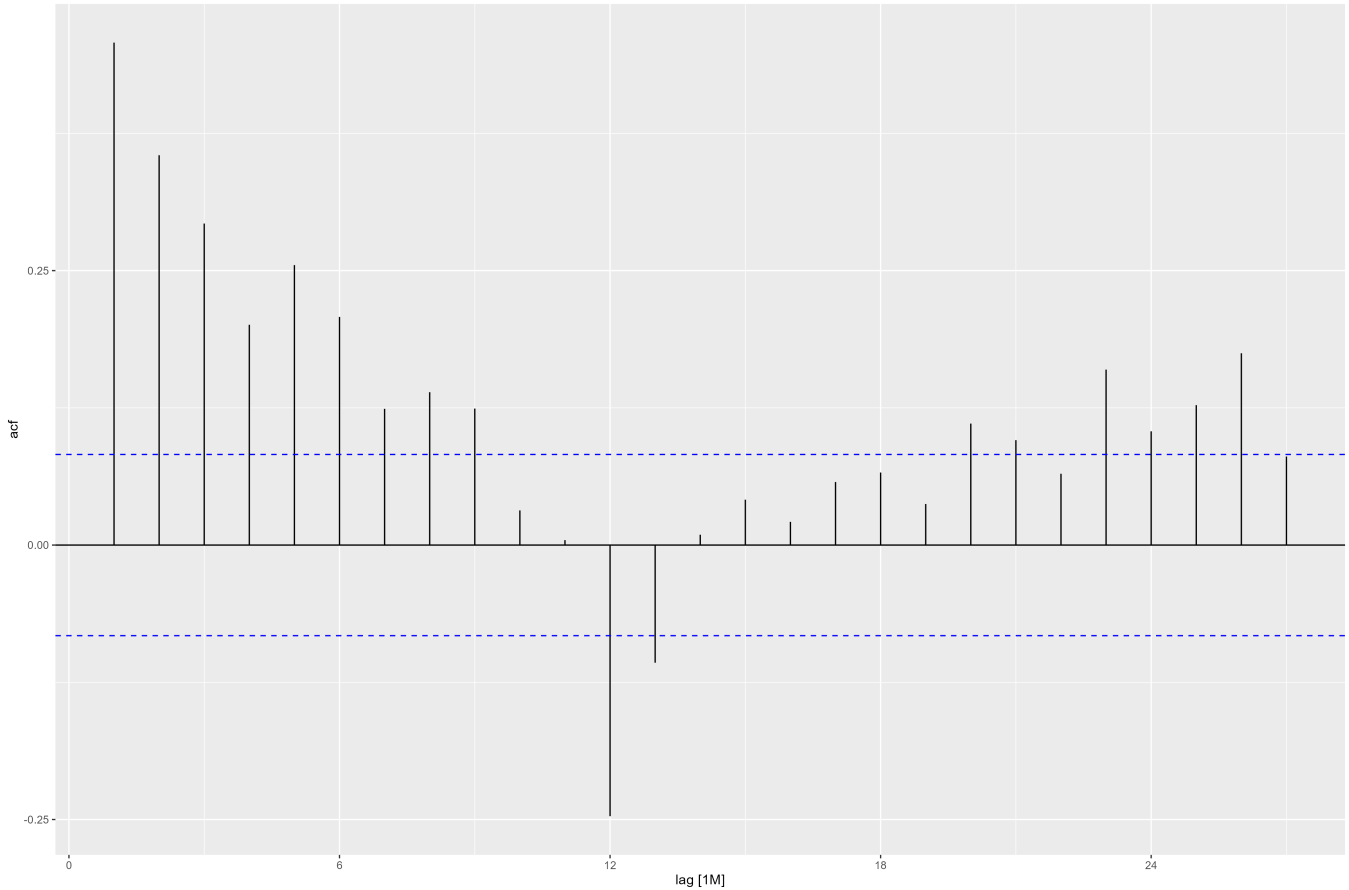
- We can do a seasonal difference with the same *difference* function in R, which takes as parameter the number of lags to consider for the difference.

```
vic_births4 <- vic_births |> mutate(BirthSDiff = difference(Births, 12))
vic_births4 |> head(n = 15)
vic_births4 |> autoplot(BirthSDiff)
vic_births4 |> ACF(BirthSDiff) |> autoplot()

vic_births4 |> features(BirthSDiff, unitroot_kpss)
vic_births5 <- vic_births4 |> filter(is.na(BirthSDiff) == FALSE)
vic_births5
adf.test(vic_births5$BirthSDiff)
```

- The seasonally differenced data



- The corresponding ACF plot



- And the statistical tests

```
# A tibble: 1 x 3
  State kpss_stat kpss_pvalue
  <chr>      <dbl>       <dbl>
1 VIC       0.0367         0.1
> |
```

```
        Augmented Dickey-Fuller Test

data:  vic_births5$BirthSDiff
Dickey-Fuller = -5.3991, Lag order = 8, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(vic_births5$BirthSDiff) : p-value smaller than printed p-value
```

- Everything shows that the data is stationary (even if it is not white noise).

Beer production

- Let us consider another example, the quarterly beer production in Australia (from now on we will only look at the KPSS test).

- Sometimes both a seasonal and a regular (first order) difference is needed to make the data stationary. Their order is not important, but it is suggested to start with the seasonal differencing, maybe the other one is not going to be needed after that.

- Since our previous example was not stationary after the seasonal difference, let's try a first order difference as well.

```
beer <- aus_production |> select(Beer)
beer
beer |> autoplot()
beer |> features(Beer, unitroot_kpss)

beerD1 <- beer |> mutate(BeerSD = difference(Beer, 4))
beerD1
beerD1 |> autoplot(BeerSD)

beerD1 |> features(BeerSD, unitroot_kpss)

beerD2 <- beerD1 |> mutate(BeerSDD = difference(BeerSD, 1))
beerD2
beerD2 |> autoplot(BeerSDD)
beerD2 |> features(BeerSDD, unitroot_kpss)
```

- For the previous examples, we followed a general flow: check the data for stationarity (for example with the KPSS test), difference if it is not stationary, check again, difference again, etc.

- This process can be simplified by the use of *unitroot_ndiffs* function, which will tell us how many times a time series should be differenced to become stationary.

- Similarly, the *unitroot_nsdiffs* function will tell us how many times do we need to perform a seasonal differencing on the data.

```
beer |> features(Beer, unitroot_ndiffs)
beer |> features(Beer, unitroot_nsdiffs)
beer |> features(Beer, c(unitroot_ndiffs, unitroot_nsdiffs))
```

- The results show that we need one seasonal and one first order differencing.

- The two functions (*unitroot_ndiffs* and *unitroot_nsdiffs* measure the required number of differencing independently of each other). Sometimes, it might happen that after performing one differencing (start with the seasonal), the second is no longer needed.

```
vic_births |> features(Births, c(unitroot_ndiffs, unitroot_nsdiffs))

vic_births |> features(difference(Births, 12), c(unitroot_ndiffs, unitroot_
    nsdiffs))

vic_births |> features(difference(Births, 1), c(unitroot_ndiffs, unitroot_
    nsdiffs))
```