Forecasting in R

Zahid Asghar and Tayyaba Batool,Capacity Analytics, Islamabad

26 September 2022

Table of contents

Forecasting in R

```r
library(readxl)
# Read the data from Excel into R
mydata <- read_excel("exercise1.xlsx")

# Look at the first few lines of mydata
head(mydata)

# Create a ts object called myts
myts <- ts(mydata[,2:4], start = c(1981, 1), frequency = 4)
```

- lot the data you stored as `myts` using `autoplot()` with facetting.
- Plot the same data without facetting by setting the appropriate argument to `FALSE`. What happens?
- Plot the `gold`, `woolyrnq`, and `gas` time series in separate plots.
- Use `which.max()` to spot the outlier in the `gold` series. Which observation was it?
- Apply the `frequency()` function to each commodity to get the number of observations per unit time. This would return 52 for weekly data, for example.

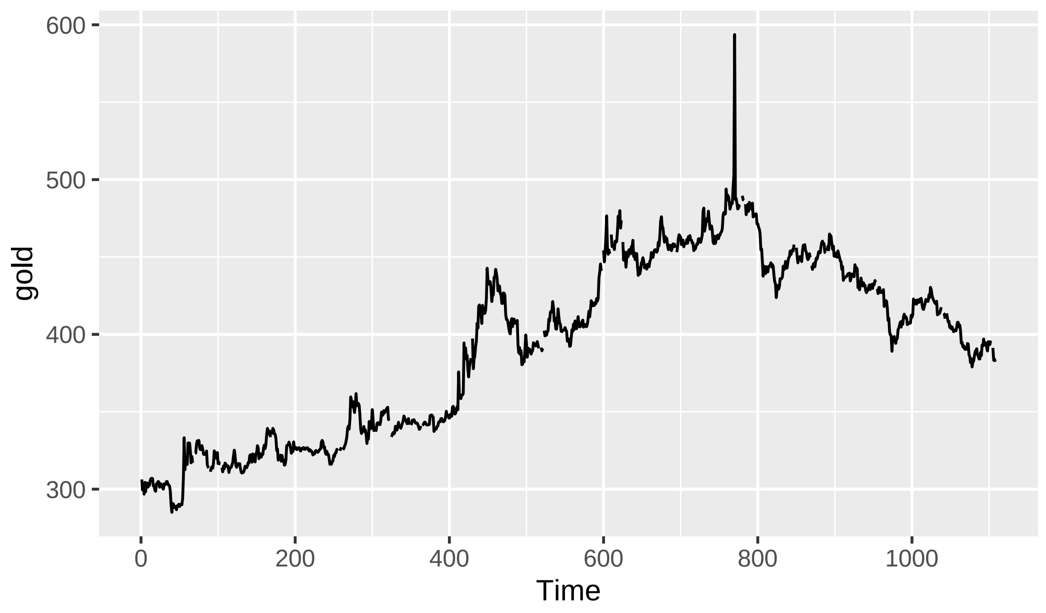
```r
library(fpp2)
```

```
Registered S3 method overwritten by 'quantmod':
  method            from
  as.zoo.data.frame zoo

-- Attaching packages ---------------------------------------------- fpp2 2.4 --

v ggplot2   3.3.6     v fma       2.4
v forecast  8.16      v expsmooth 2.3
```

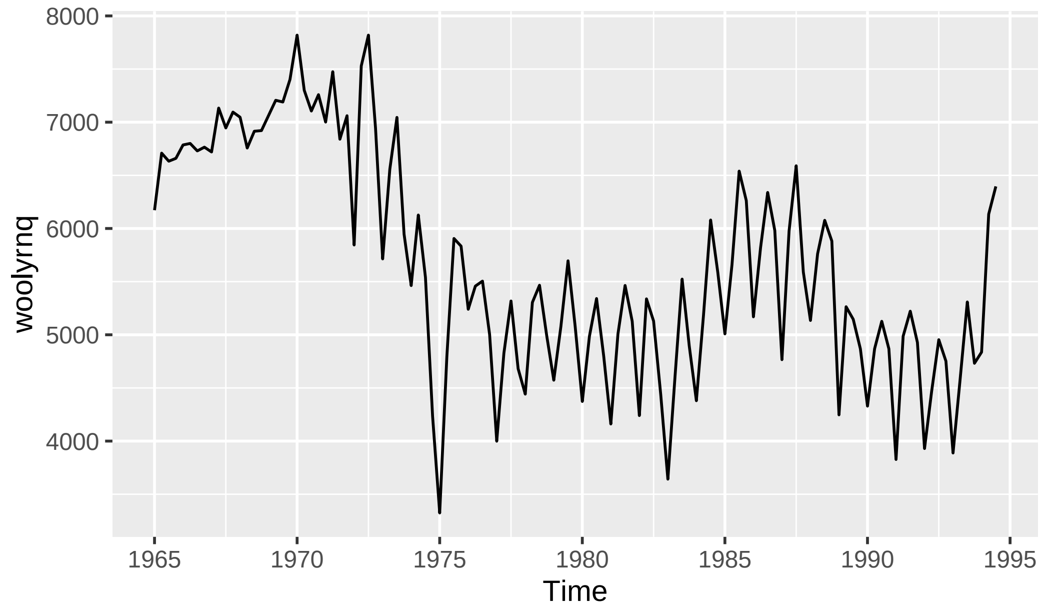
```r
library(fpp3)
```

```
-- Attaching packages -------------------------------------------- fpp3 0.4.0 --

v tibble      3.1.7     v tsibble     1.1.1
v dplyr       1.0.9     v tsibbledata 0.4.0
v tidyr       1.2.0     v feasts      0.2.2
v lubridate   1.8.0     v fable       0.3.1

-- Conflicts ------------------------------------------------- fpp3_conflicts --
x lubridate::date()      masks base::date()
x dplyr::filter()        masks stats::filter()
x fabletools::forecast() masks forecast::forecast()
x tsibble::intersect()   masks base::intersect()
x tsibble::interval()    masks lubridate::interval()
x dplyr::lag()           masks stats::lag()
x tsibble::setdiff()     masks base::setdiff()
x tsibble::union()       masks base::union()


Attaching package: 'fpp3'

The following object is masked from 'package:fpp2':

    insurance
```

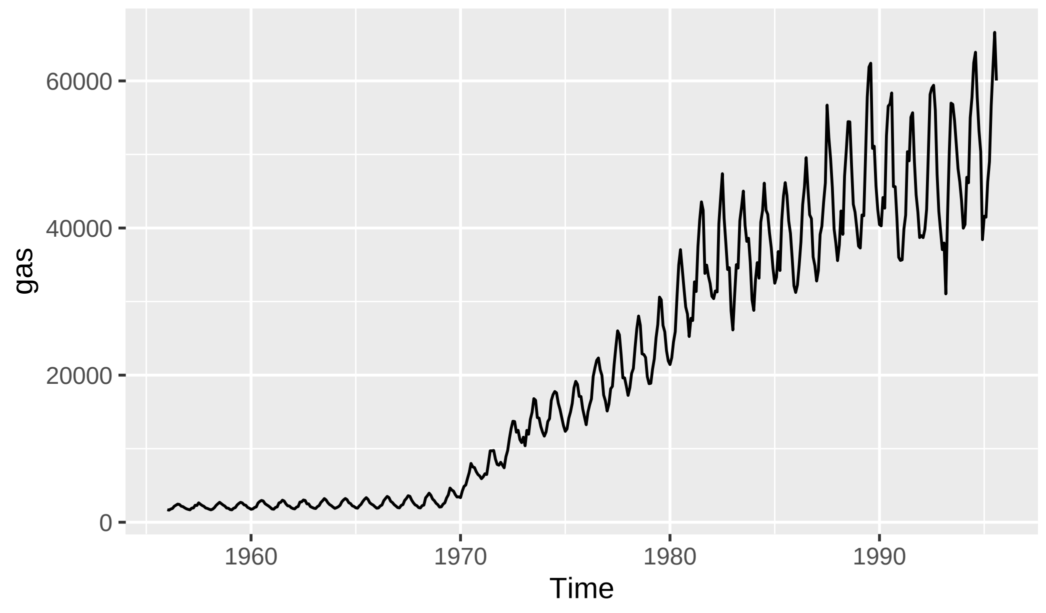
```
# Plot the data with facetting
#autoplot(myts, facets = TRUE)

# Plot the data without facetting
#autoplot(myts,facets=FALSE)

# Plot the three series
autoplot(gold)
```



```
autoplot(woolyrnq)
```



```r
autoplot(gas)
```



```r
# Find the outlier in the gold series
goldoutlier <- which.max(gold)

# Look at the seasonal frequencies of the three series
frequency(gold)
```

```
[1] 1
```

```r
frequency(woolyrnq)
```

```
[1] 4
```

```r
frequency(gas)
```

```
[1] 12
```

Seasonal plots

Along with time plots, there are other useful ways of plotting data to emphasize seasonal patterns and show changes in these patterns over time.

A seasonal plot is similar to a time plot except that the data are plotted against the individual "seasons" in which the data were observed. You can create one using the ggseasonplot() function the same way you do with autoplot(). An interesting variant of a season plot uses polar coordinates, where the time axis is circular rather than horizontal; to make one, simply add a polar argument and set it to TRUE. A subseries plot comprises mini time plots for each season. Here, the mean for each season is shown as a blue horizontal line. One way of splitting a time series is by using the window() function, which extracts a subset from the object x observed between the times start and end.

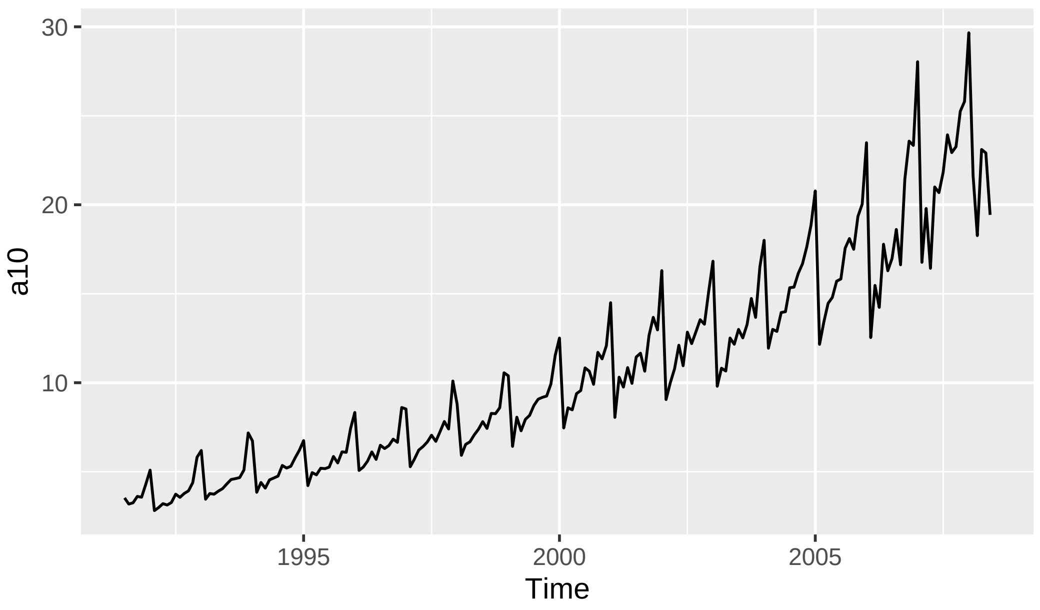
> window(x, start = NULL, end = NULL) In this exercise, you will load the fpp2 package and use two of its datasets:

a10 contains monthly sales volumes for anti-diabetic drugs in Australia. In the plots, can you see which month has the highest sales volume each year? What is unusual about the results in March and April 2008? ausbeer which contains quarterly beer production for Australia. What is happening to the beer production in Quarter 4? These examples will help you to visualize these plots and understand how they can be useful.

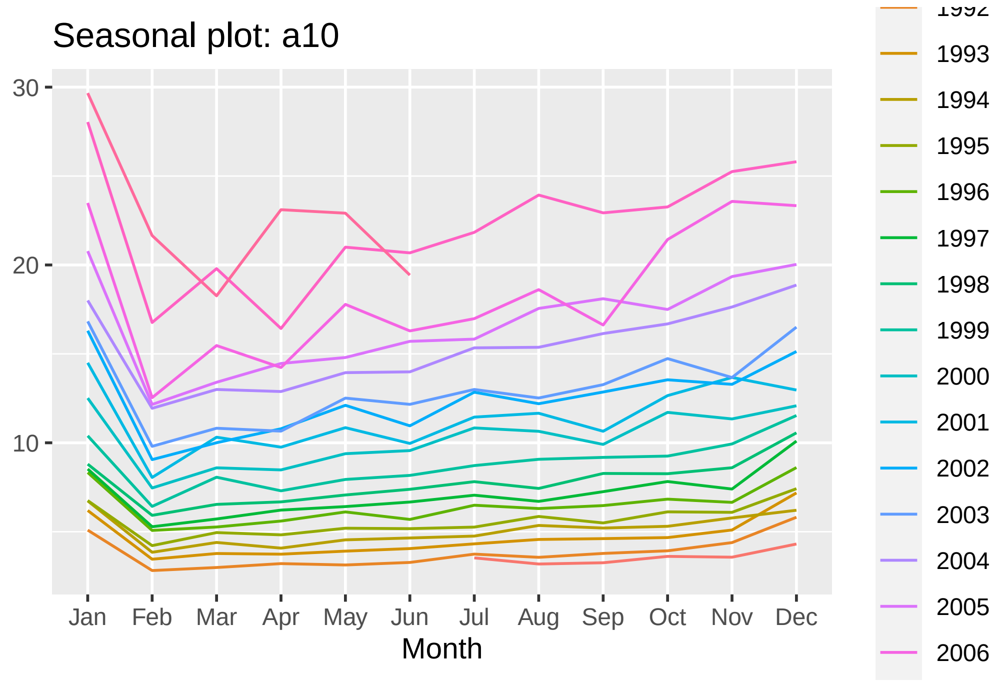
Instructions

Use library() to load the fpp2 package. Use autoplot() and ggseasonplot() to produce plots of the a10 data. Use the ggseasonplot() function and its polar argument to produce a polar coordinate plot for the a10 data. Use the window() function to consider only the ausbeer data starting from 1992. Finally, use autoplot() and ggsubseriesplot() to produce plots of the beer series.

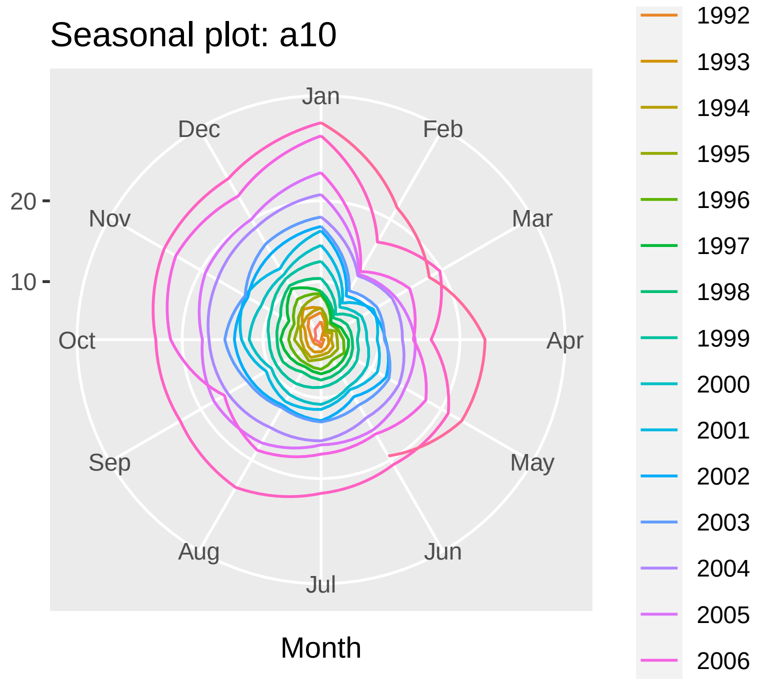
```r
# Load the fpp2 package
library(fpp2)

# Create plots of the a10 data
autoplot(a10)
```



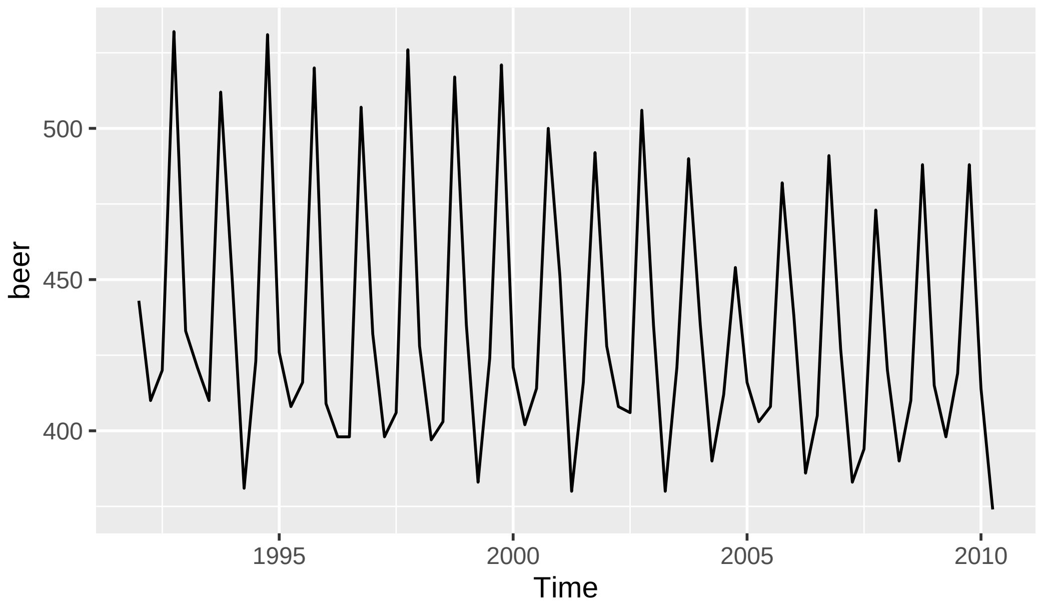
```r
ggseasonplot(a10)
```



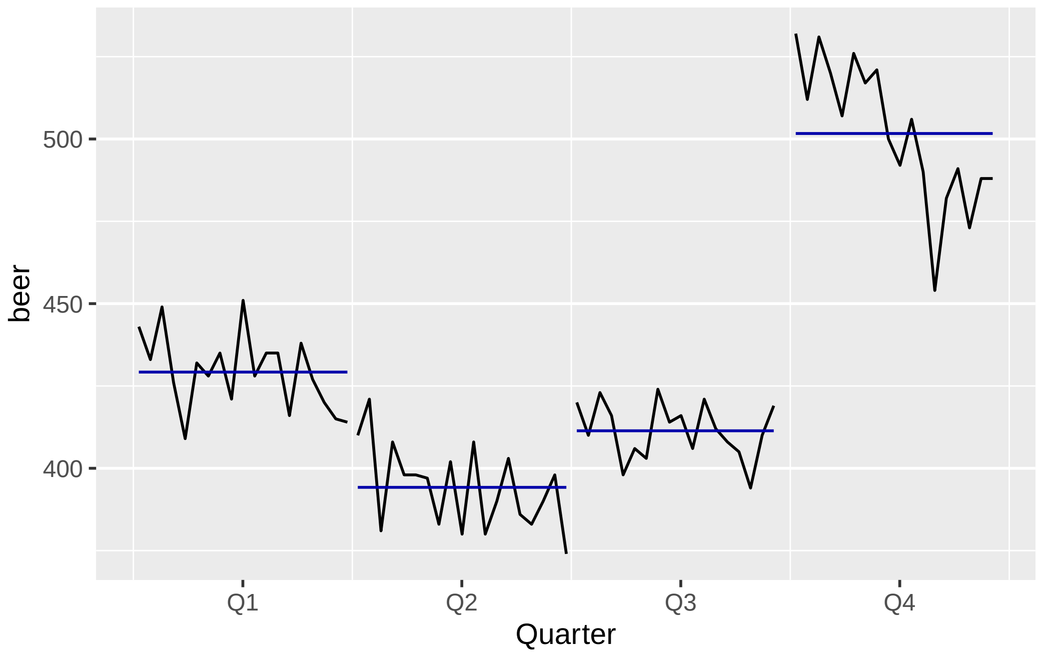
```r
# Produce a polar coordinate season plot for the a10 data
ggseasonplot(a10, polar = TRUE)
```



```
# Restrict the ausbeer data to start in 1992
beer <- window(ausbeer,start= 1992)

# Make plots of the beer data
autoplot(beer)
```



```
ggsubseriesplot(beer)
```



Autocorrelation of non-seasonal time series

Another way to look at time series data is to plot each observation against another observation that occurred some time previously by using gglagplot(). For example, you could plot against . This is called a lag plot because you are plotting the time series against lags of itself.

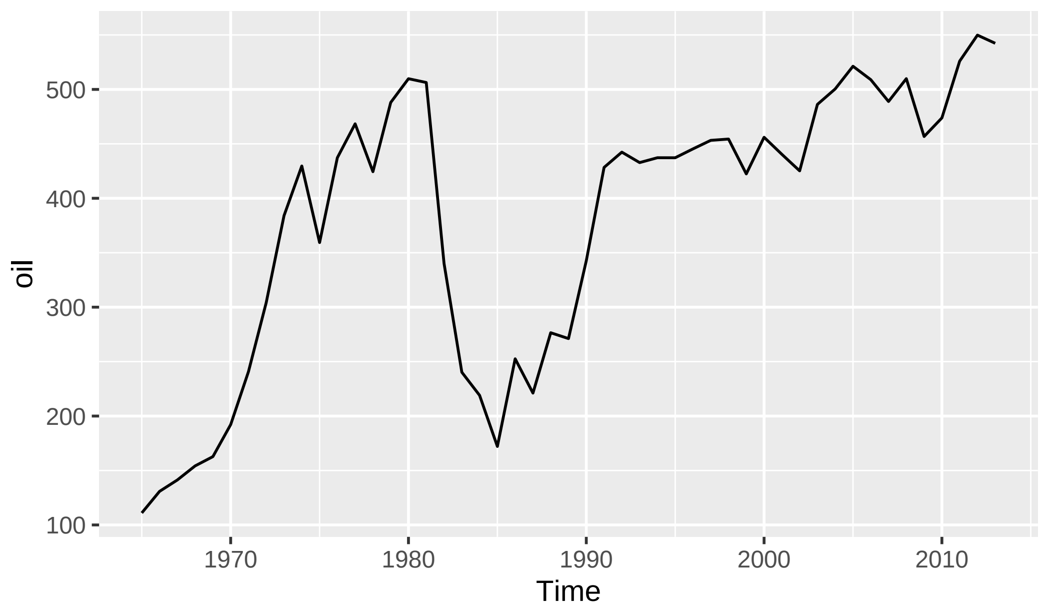
The correlations associated with the lag plots form what is called the autocorrelation function (ACF). The ggAcf() function produces ACF plots.

In this exercise, you will work with the pre-loaded oil data (available in the package fpp2), which contains the annual oil production in Saudi Arabia from 1965-2013 (measured in millions of tons).

Instructions

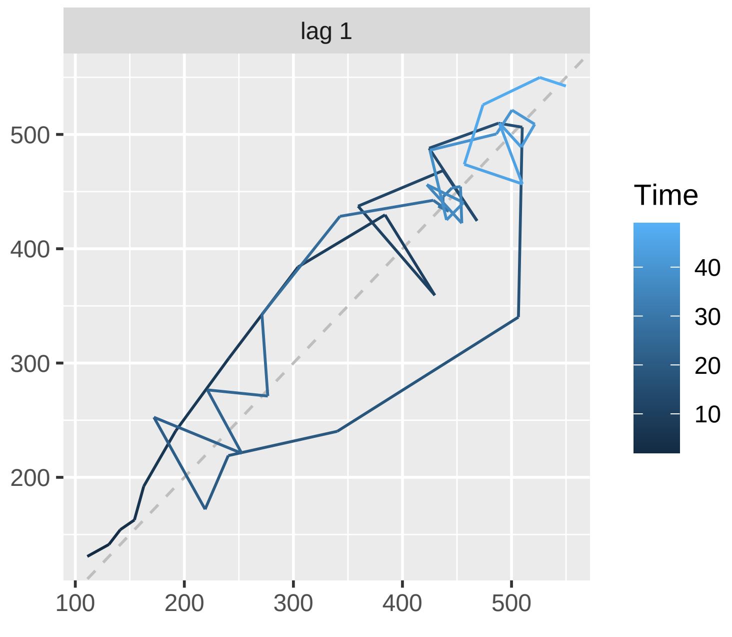
Use the autoplot() function to plot the oil data. For the oil data, plot the relationship between y_t and $y_{t-1}, k = 1, \dots, 9$ using one of the two functions introduced above. Look at how the relationships change as the lag increases. Likewise, plot the correlations associated with each of the lag plots using the other appropriate new function.

```
# Create an autoplot of the oil data
autoplot(oil)
```

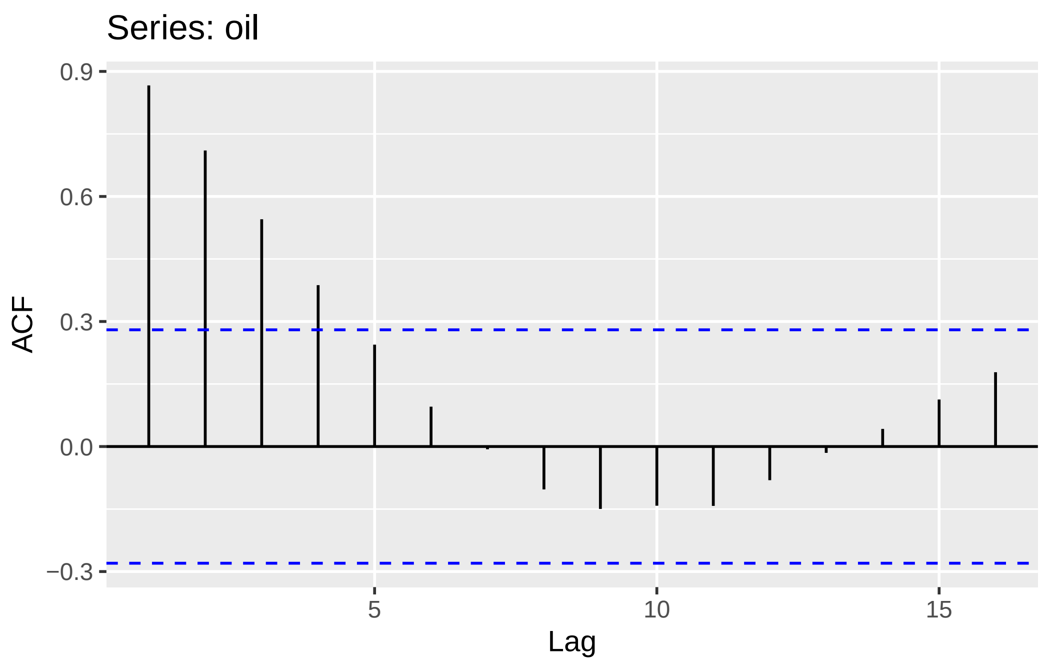


```
# Create a lag plot of the oil data
gglagplot(oil,lag=1:9)
```

```
Warning in 1:lags: numerical expression has 9 elements: only the first used
```



```
# Create an ACF plot of the oil data
ggAcf(oil)
```



Autocorrelation of seasonal and cyclic time series

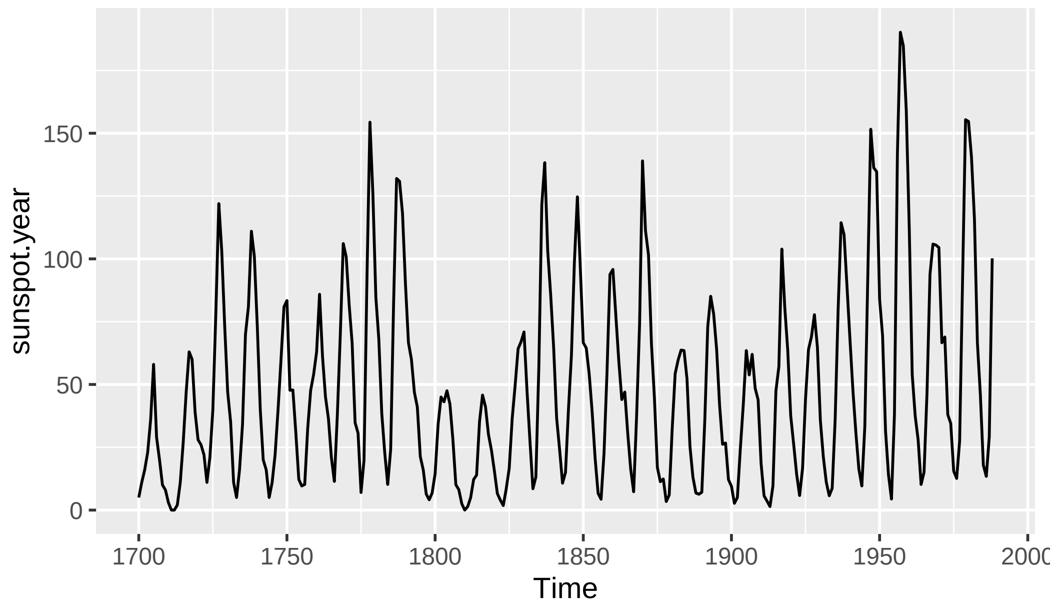
When data are either seasonal or cyclic, the ACF will peak around the seasonal lags or at the average cycle length.

You will investigate this phenomenon by plotting the annual sunspot series (which follows the solar cycle of approximately 10-11 years) in sunspot.year and the daily traffic to the Hyndsight blog (which follows a 7-day weekly pattern) in hyndsight. Both objects have been loaded into your workspace.

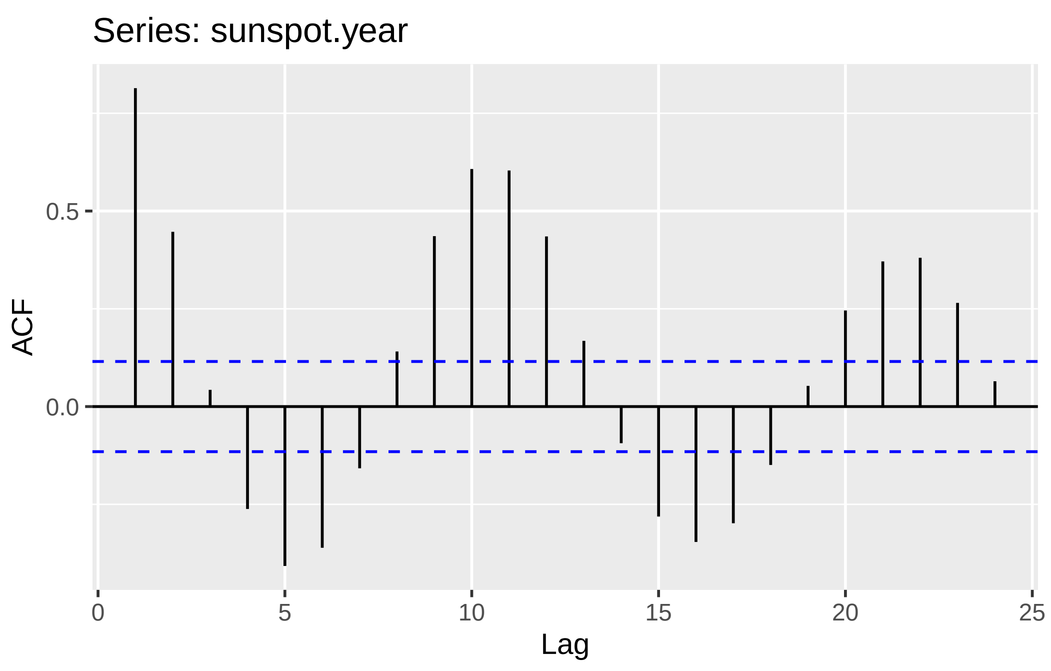
Instructions

Produce a time plot and ACF plot of sunspot.year. By observing the ACF plot, at which lag value (x) can you find the maximum autocorrelation (y)? Set this equal to maxlag_sunspot. Produce a time plot and ACF plot of hyndsight. By observing the ACF plot, at which lag value (x) can you find the maximum autocorrelation (y)? Set this equal to maxlag_hyndsight.

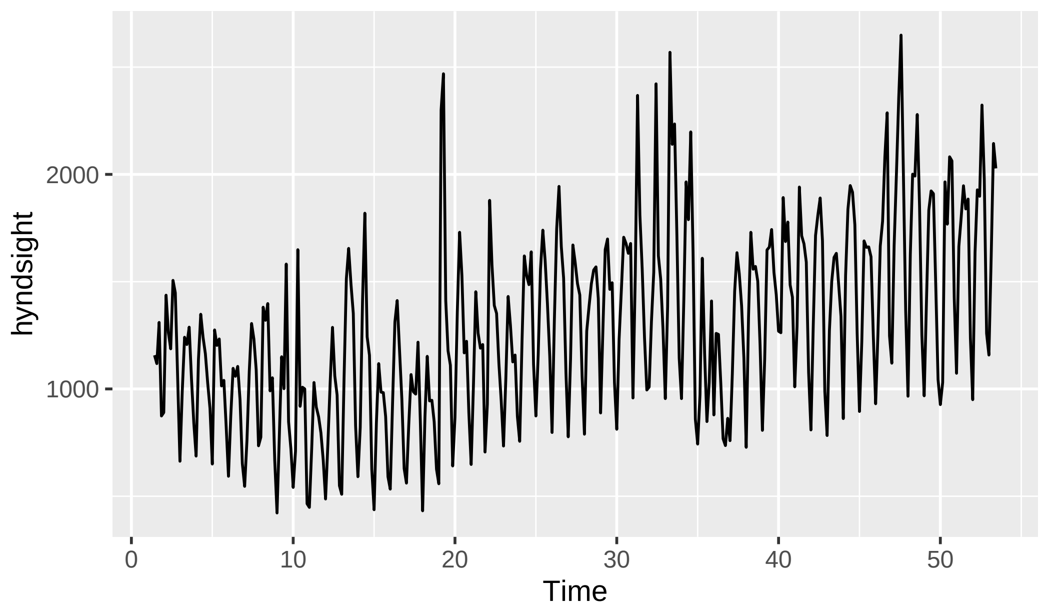
```
# Plot the annual sunspot numbers
autoplot(sunspot.year)
```



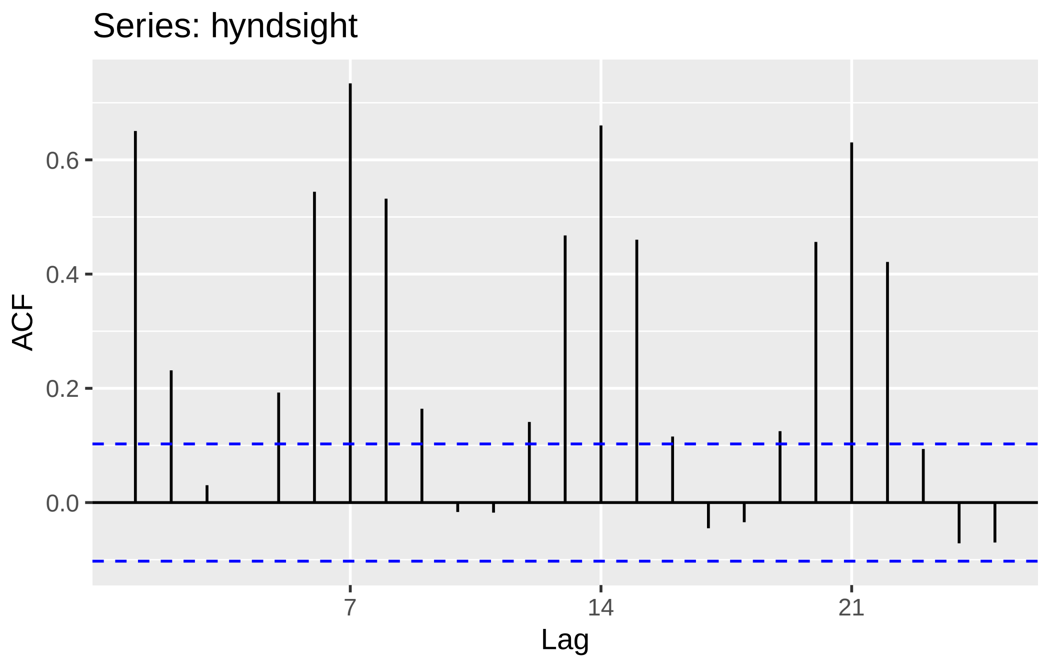
```
ggAcf(sunspot.year)
```



```
# Save the lag corresponding to maximum autocorrelation
maxlag_sunspot <- lag(1)

# Plot the traffic on the Hyndsight blog
autoplot(hyndsight)
```



```
ggAcf(hyndsight)
```


Series: hyndsight
ACF
Lag