DS 116 - Data Visualization

Time series visualization

Habet Madoyan

American University of Armenia

Time Series

Time Series

- Series of data observed over time
- Anything that is observed over time are time series
- Eg.: Daily IBM stock prices, monthly rainfall in London, Annual GDP of Armenia

Time Series

Time Series:

- A sequence of data in chronological order
- Data is commonly recorded sequentially, over time
- Time series data is everywhere

Time Series

Why to study time series separately?

- Sequence of the data does matter
- Look for the pattern in sequence
- Usually autoregressive in nature (value in time t depends on the value in time $t-1$)

Time Series

```r
rates <- read.csv('Data/exchange_rates.csv')
head(rates)
```

```
##         Date    EUR    USD
## 1 2005-01-04 661.22 486.05
## 2 2005-01-05 649.65 486.05
## 3 2005-01-07 646.32 488.75
## 4 2005-01-08 651.25 493.37
## 5 2005-01-10 653.41 495.01
## 6 2005-01-11 651.73 497.39
```

Time Series

```r
str(rates)
```

```
## 'data.frame':    3926 obs. of  3 variables:
##  $ Date: chr  "2005-01-04" "2005-01-05" "2005-01-07" "2005-01-08" ...
##  $ EUR : num  661 650 646 651 653 ...
##  $ USD : num  486 486 489 493 495 ...
```

Time Series

Symbol	Meaning	Example
%d	day as a number (0-31)	31-Jan
%a	abbreviated weekday	Mon
%A	unabbreviated weekday	Monday
%m	month (00-12)	00-12
%b	abbreviated month	Jan
%B	unabbreviated month	January
%y	2-digit year	7
%Y	4-digit year	2007

Time Series

We have

- 2-digit year (%Y)
- month - (%m)
- day (%d)

values are separated by $-$

Time Series

Change the format to date

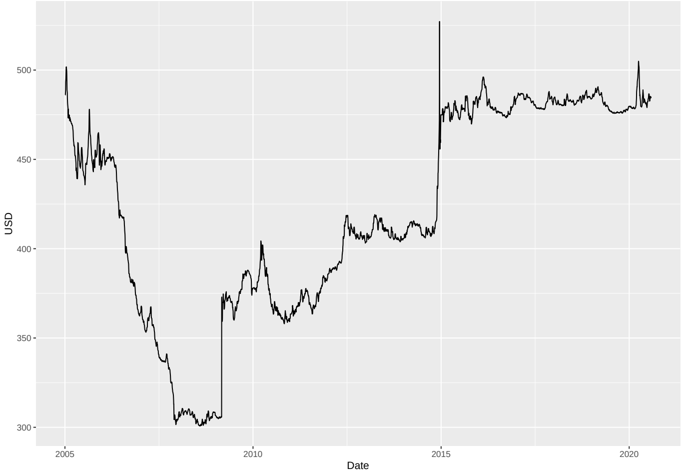
```
rates$Date <- as.Date(rates$Date, format = '%Y-%m-%d')
str(rates)
```

```
## 'data.frame':    3926 obs. of  3 variables:
##  $ Date: Date, format: "2005-01-04" "2005-01-05" ...
##  $ EUR : num  661 650 646 651 653 ...
##  $ USD : num  486 486 489 493 495 ...
```

Time Series

Now ggplot understands X axis as a *date*

```
ggplot(rates, aes(x = Date, y  = USD)) + geom_line()
```



Time Series

Customize the plot: breaks

```r
ggplot(rates, aes(x = Date, y  = USD)) + geom_line() +
  scale_x_date(breaks = 'year')
```

Time Series

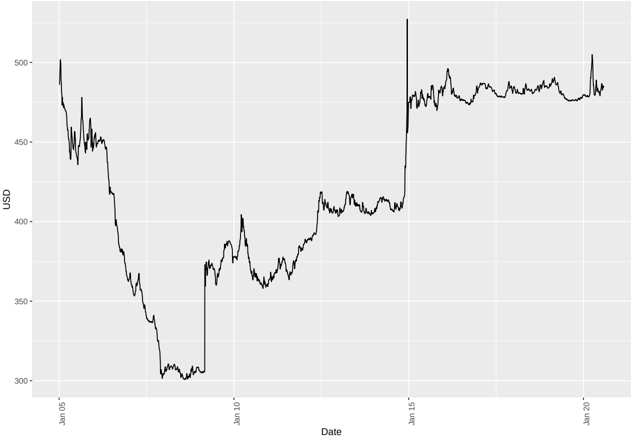
Fix the angle

```
ggplot(rates, aes(x = Date, y  = USD)) + geom_line() +
  scale_x_date(breaks = 'year') +
  theme(axis.text.x = element_text(angle = 90))
```

Time Series

Change the format of data label

```
ggplot(rates, aes(x = Date, y  = USD)) + geom_line() +
  scale_x_date(date_labels = '%b %y') +
  theme(axis.text.x = element_text(angle = 90))
```

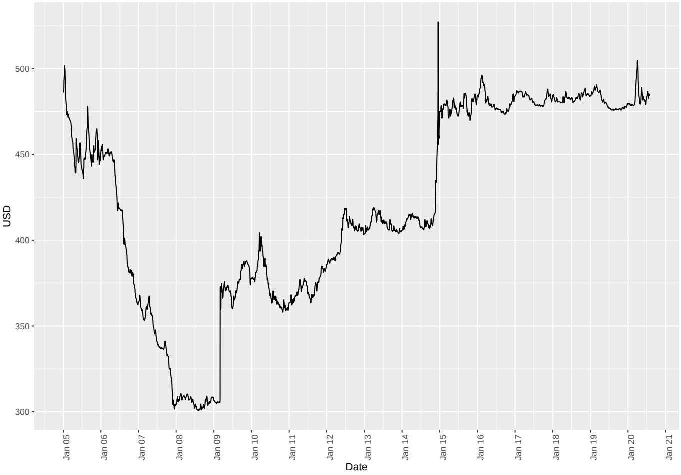


Time Series

Breaks and data labels together

```
ggplot(rates, aes(x = Date, y  = USD)) + geom_line() +
  scale_x_date(breaks = 'year', date_labels = '%b %y') +
  theme(axis.text.x = element_text(angle = 90))
```

Time Series



USD
500
450
400
350
300
Jan 05
Jan 06
Jan 07
Jan 08
Jan 09
Jan 10
Jan 11
Jan 12
Jan 13
Jan 14
Jan 15
Jan 16
Jan 17
Jan 18
Jan 19
Jan 20
Jan 21
Date

Time Series

R has several specific classes for time series, the basic is ts()

```
t1 <- ts(c(103,44,78,52,90), start=2012)
class(t1)
```

```
## [1] "ts"
```

Time Series

```r
t1
```

```
## Time Series:
## Start = 2012
## End = 2016
## Frequency = 1
## [1] 103  44  78  52  90
```

Time Series

- When creating a ts object, it is important to determine the “frequency” of the series.
- The frequency is the number of observations before the seasonal pattern repeats.
- Or otherwise it is showing how many times during the time period the observation was taken

Time Series

Frequency when the time period is a year

Data	Frequency
Annual	1
Quarterly	4
Monthly	12
Weekly	52

Time Series

Air Passengers data
Monthly totals of international airline passengers, 1949 to 1960, thousands.

```
data("AirPassengers")
start(AirPassengers)
```

```
## [1] 1949    1
```

```
end(AirPassengers)
```

```
## [1] 1960   12
```

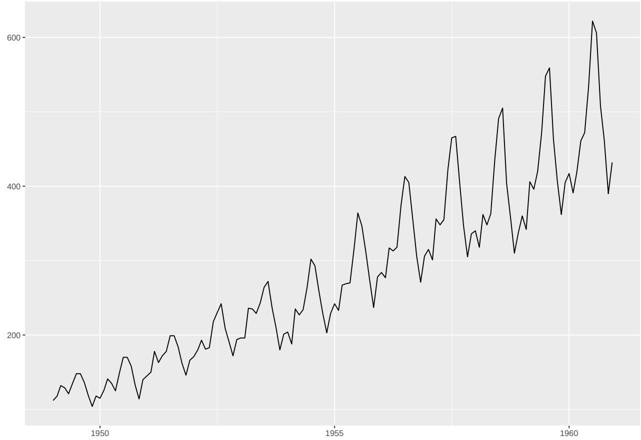
```
frequency(AirPassengers)
```

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

Time Series

autplot() is a generic method from ggplot2 package that draws a plot for an object of a particular class in a single command.

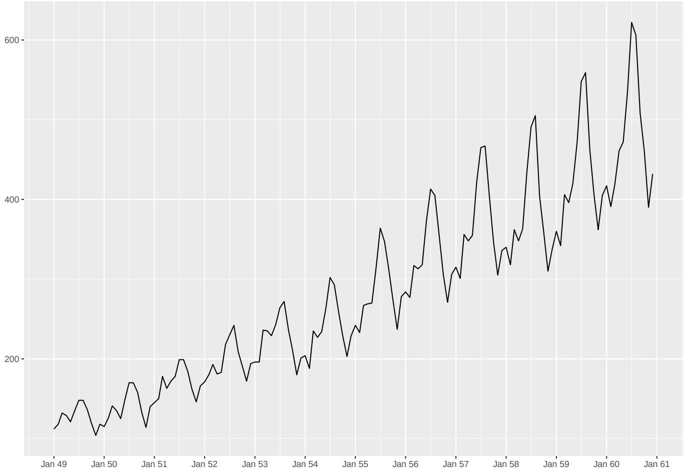
```
autoplot(AirPassengers)
```



Time Series

Autoplot is a ggplot object and is used as a ggplot object!

```
autoplot(AirPassengers) +
  scale_x_date(breaks = 'year', date_labels = '%b %y')
```



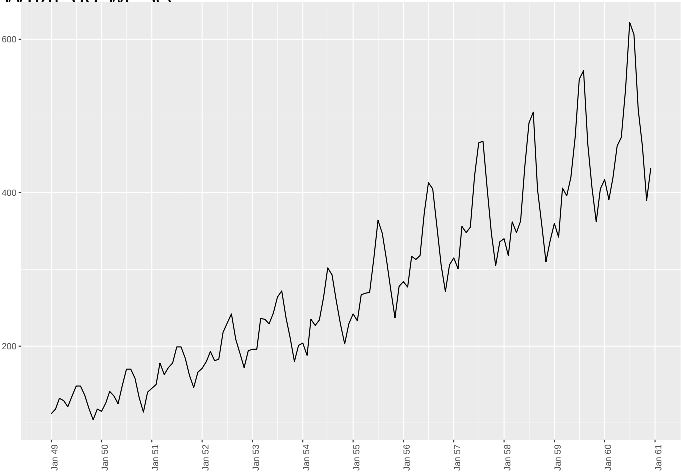
Time Series

We visualize time series to see the patterns:
There are three main patterns in time series

- **Trend** pattern exists when there is a long-term increase or decrease in the series. It can be linear, exponential, or different one and can change direction during time.
- **Seasonality** exists when data is influenced by seasonal factors, such as a day of the week, a month, and one-quarter of the year. A seasonal pattern exists for a fixed known period.
- **Cyclic** pattern occurs when data rise and fall, but this does not happen within the fixed time and the duration of these fluctuations is usually at least 2 years.

Time Series

What do we see ?



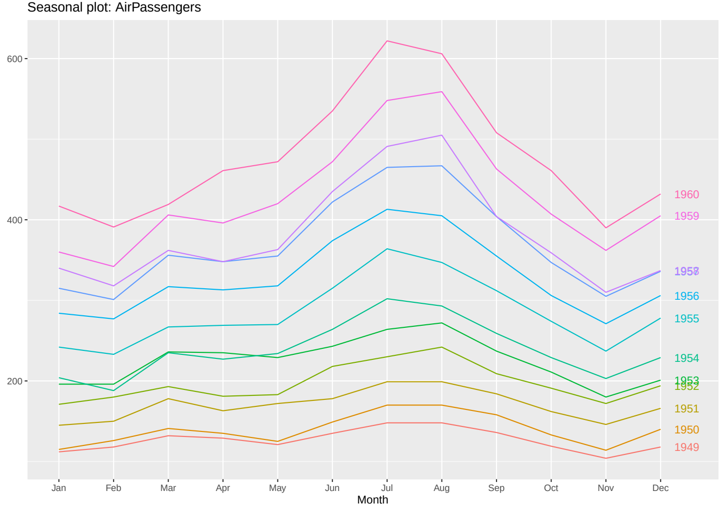
Time Series

Few other ways of visualization to spot patterns:

- 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.
- An example is given for the Air Passengers data.

Time Series

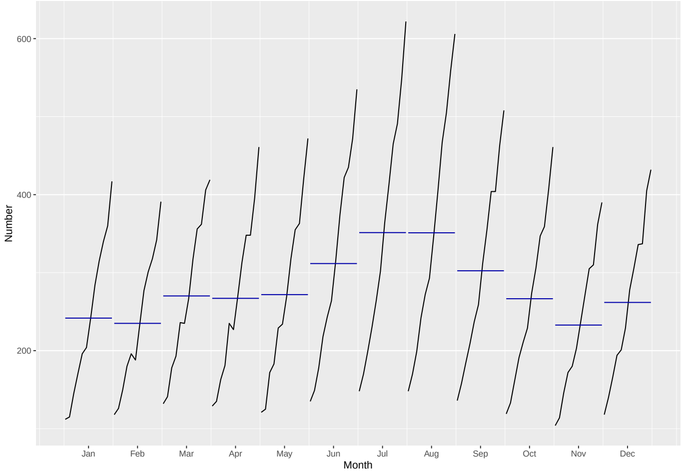
```r
ggseasonplot(AirPassengers, year.labels = T)
```



Time Series

Subseries plot

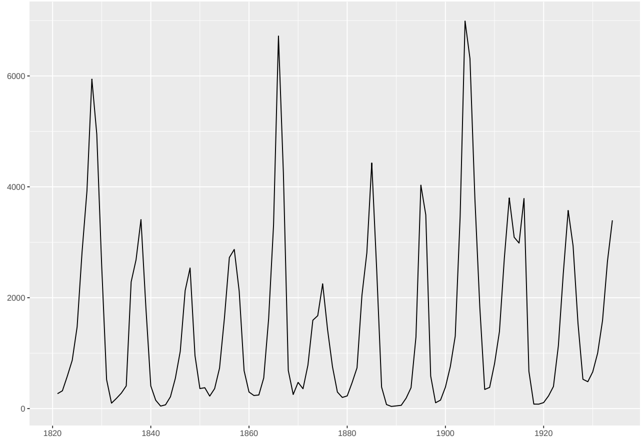
```r
ggsubseriesplot(AirPassengers) + ylab("Number")
```



Time Series

The plot shows the famous Canadian lynx data – the number of lynx trapped each year in the McKenzie river district of northwest Canada (1821-1934)

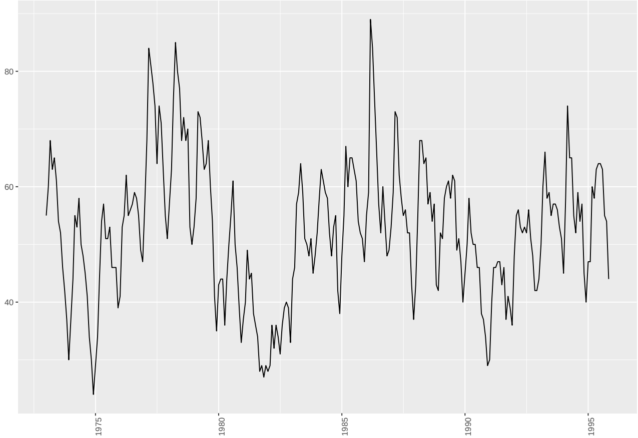
```
autoplot(lynx)
```



Time Series

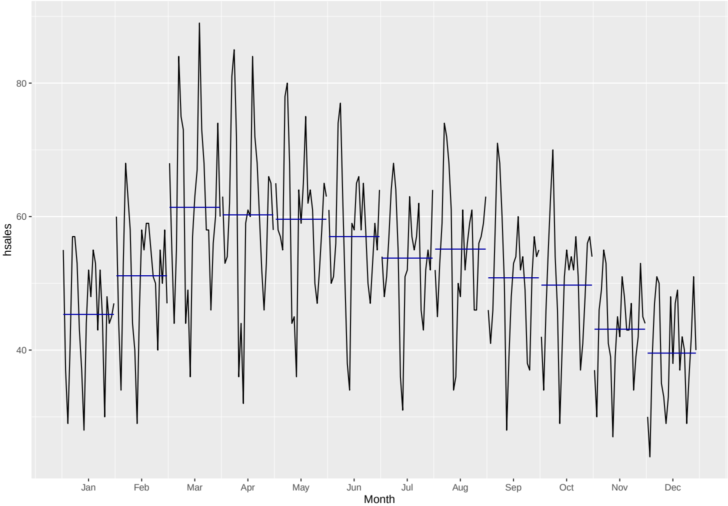
Monthly sales of new one-family houses sold in the USA (1973-1995)

```
autoplot(hsales) + theme(axis.text.x = element_text(angle = 90))
```



Time Series

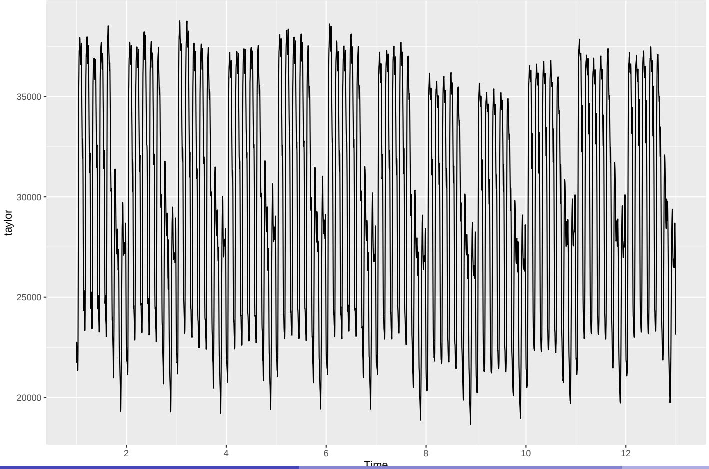
```
ggsubseriesplot(hsales)
```



Time Series

Half-hourly electricity demand in England and Wales from Monday 5 June 2000 to Sunday 27 August 2000

```r
autoplot(taylor)
```



Time Series

Time series decomposition

Time series data can exhibit a variety of patterns, and it is often helpful to split a time series into several components, each representing an underlying pattern category.

We have discussed three types of time series patterns: trend, seasonality and cycles. When we decompose a time series into the following components:

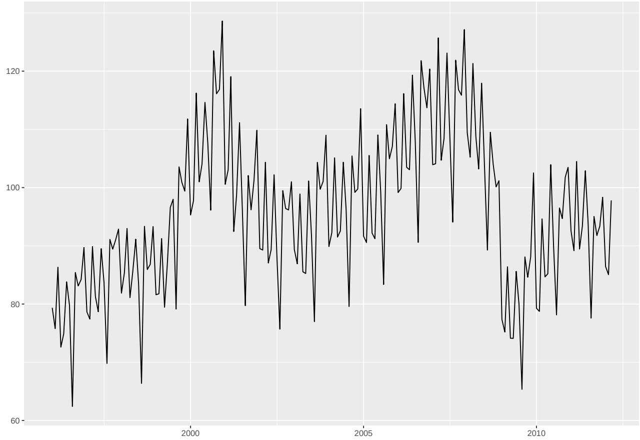
- Combine the trend and cycle into a single trend-cycle component (sometimes called the trend for simplicity).
- Seasonal Component
- A remainder component (containing anything else in the time series).

More on the mathematical functions for the decomposition can be see seen here

Time Series

Monthly manufacture of electrical equipment: computer, electronic and optical products. January 1996 - March 2012.

```
autoplot(elecequip)
```



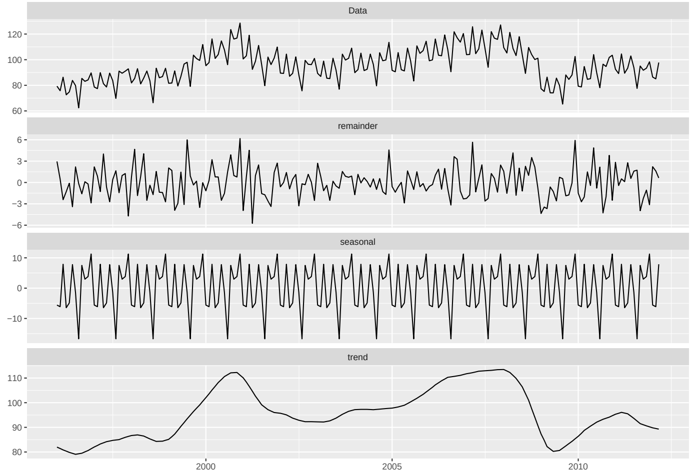
Time Series

Decomposing time series:

- t.window is the number of consecutive observations to be used when estimating the trend-cycle;
- s.window is the number of consecutive years to be used in estimating each value in the seasonal component.

Time Series

```
decomposed <- stl(elecequip, s.window = 'periodic', t.window = 13)
autoplot(decomposed)
```



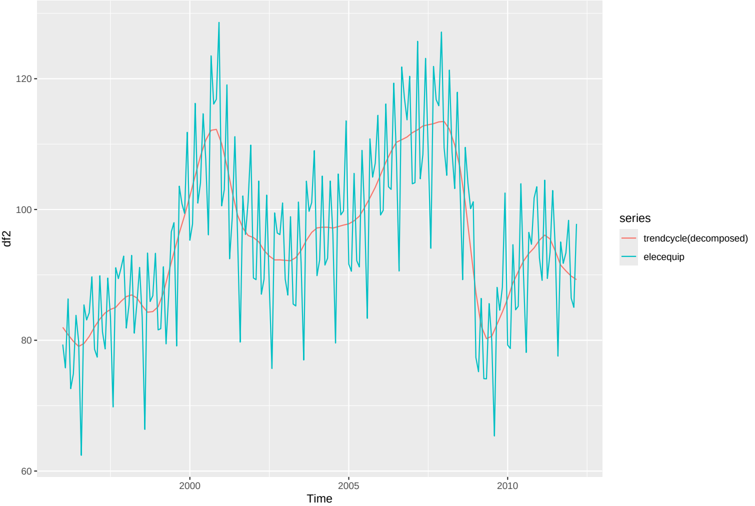
Time Series

```r
head(decomposed$time.series)
```

```
##           seasonal    trend  remainder
## Jan 1996 -5.585463 81.97999  2.9554746
## Feb 1996 -6.114891 81.41464  0.4802485
## Mar 1996  7.885094 80.84930 -2.4143901
## Apr 1996 -6.456929 80.34599 -1.2890627
## May 1996 -4.874037 79.84269 -0.1086499
## Jun 1996  7.766608 79.45116 -3.4077635
```

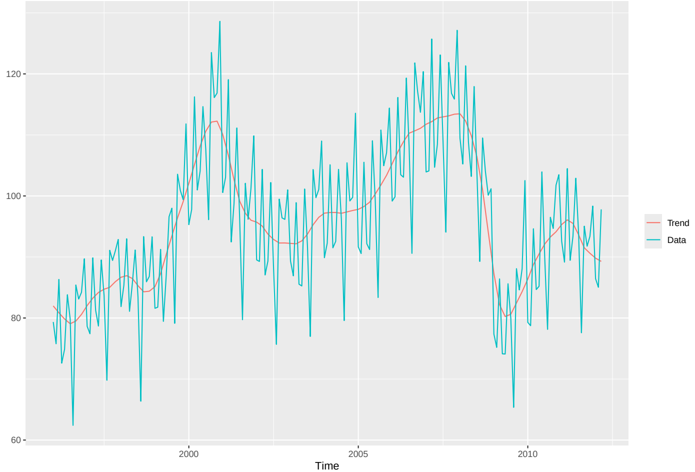
Time Series

```
df2 <- cbind(trendcycle (decomposed), elecequip)
autoplot(df2)
```



Time Series

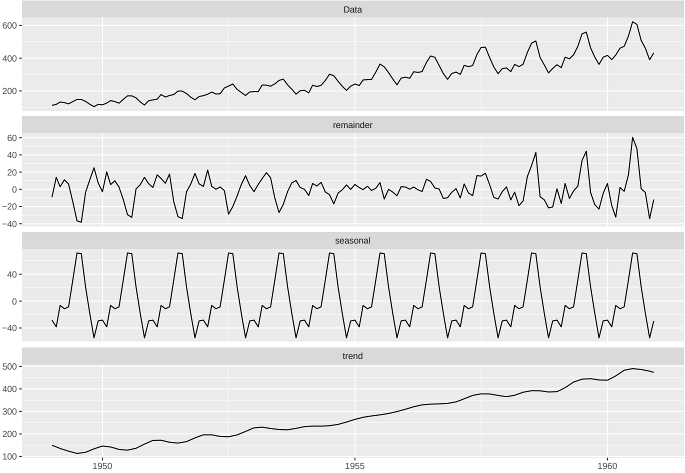
```
autoplot(df2) +
  scale_color_discrete(labels = c('Trend', 'Data'), name = '') +
  labs(y = "")
```



Time Series

Decomposing AirPassangers data

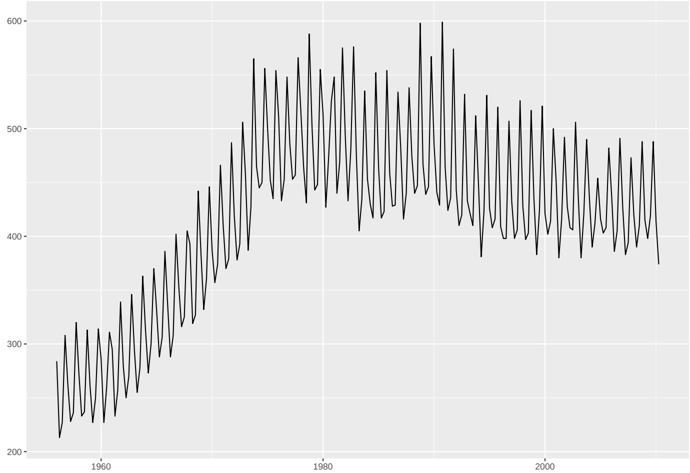
```
air_dec <- stl(AirPassengers,s.window = 'periodic', t.window = 13)
autoplot(air_dec)
```



Time Series

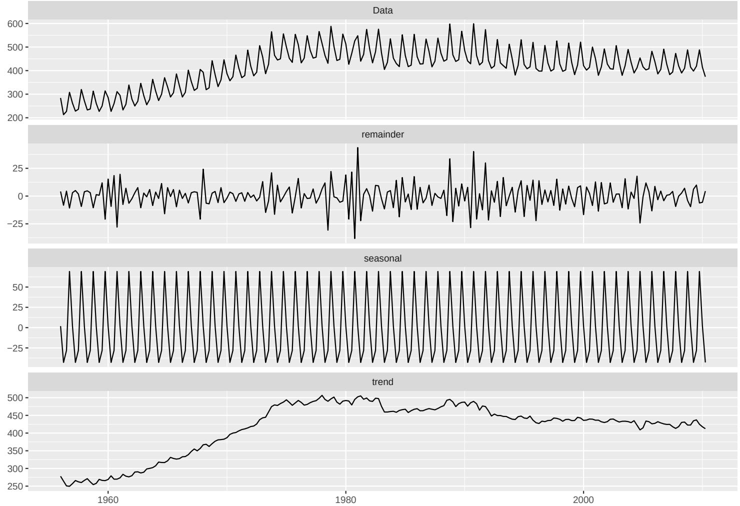
Total quarterly beer production in Australia (in megalitres) from 1956:Q1 to 2010:Q2.

```r
autoplot(ausbeer)
```



Time Series

```r
dec <- stl(ausbeer, s.window = 'periodic', t.window = 4)
autoplot(dec)
```



Time Series

STL is using LOESS for the decomposition of time series, however there are other methods as well. decompose() function does additive or multiplicative decomposition - works with moving average

- S_t is the seasonal component of the time series at the time period t
- T_t is the trend-cycle period
- R_t is the random component

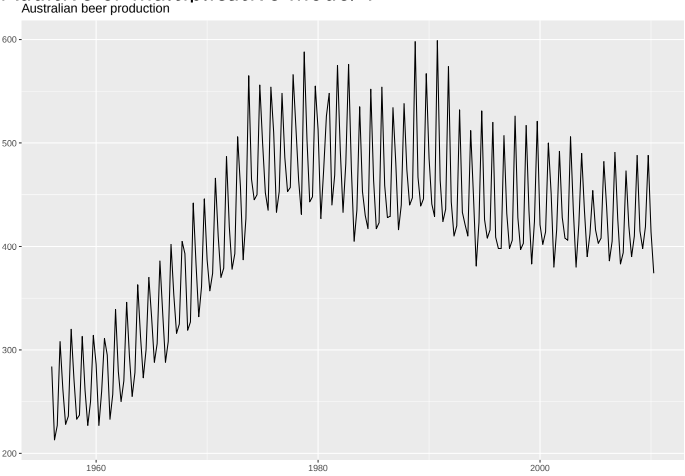
Additive decomposition
$y_t = S_t + T_t + R_t$
Multiplicative decomposition
$y_t = S_t * T_t * R_t$

Time Series

- Additive decomposition is used when variation in seasonal component is pretty stable
- Multiplicative decomposition is used when variation in seasonal pattern is increasing over time

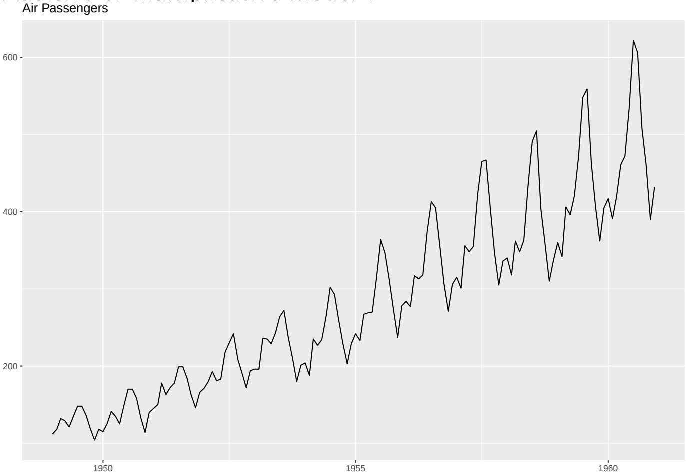
Time Series

Additive or multiplicative model ?



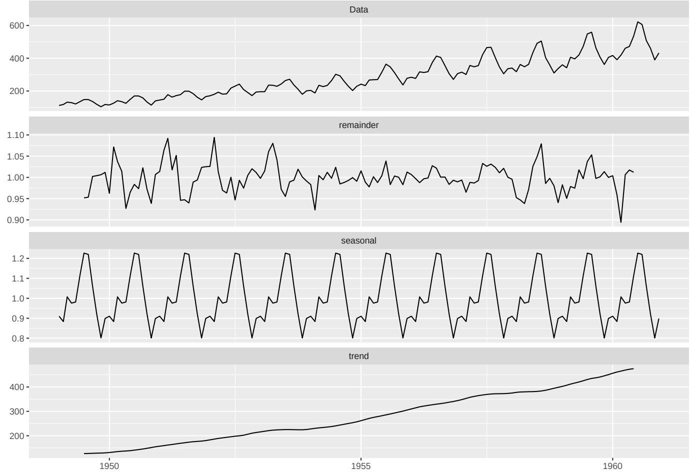
Time Series

Additive or multiplicative model ?



Time Series

```
dec <- decompose(AirPassengers, type = 'multiplicative')
autoplot(dec)
```



Time Series: autocorrelation

- Correlation measures the extent of a linear relationship between two variables,
- Autocorrelation measures the linear relationship between lagged values of a time series.

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

Time Series: autocorrelation

There are several autocorrelation coefficients, corresponding to each lag.

- r_1 correspondents to the correlation between y_t and y_{t-1}
- r_2 correspondents to the correlation between y_t and y_{t-2}

Time Series: autocorrelation

Autocorrelation plot helps to:

- Understand the trend and seasonality in the data
- Choose appropriate forecasting method

Time Series: autocorrelation

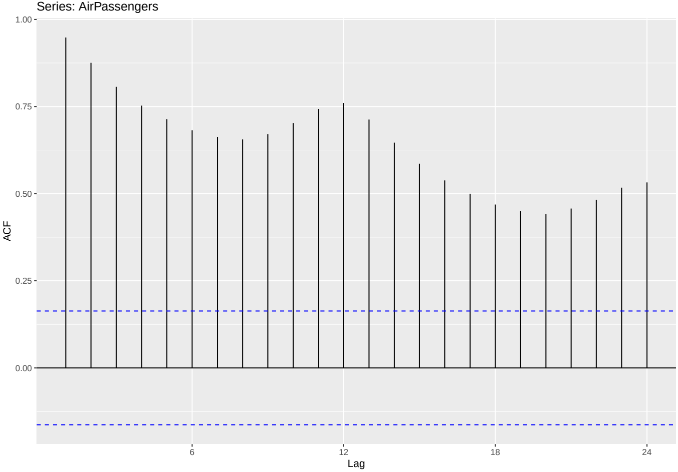
lag explained

```r
b <- as.vector(AirPassengers)
Lag1 <- Hmisc::Lag(b,1)
Lag2 <- Hmisc::Lag(b,2)
df <-data.frame(b,Lag1, Lag2)
head(df)
```

```
##     b Lag1 Lag2
## 1 112   NA   NA
## 2 118  112   NA
## 3 132  118  112
## 4 129  132  118
## 5 121  129  132
## 6 135  121  129
```

Time Series: autocorrelation

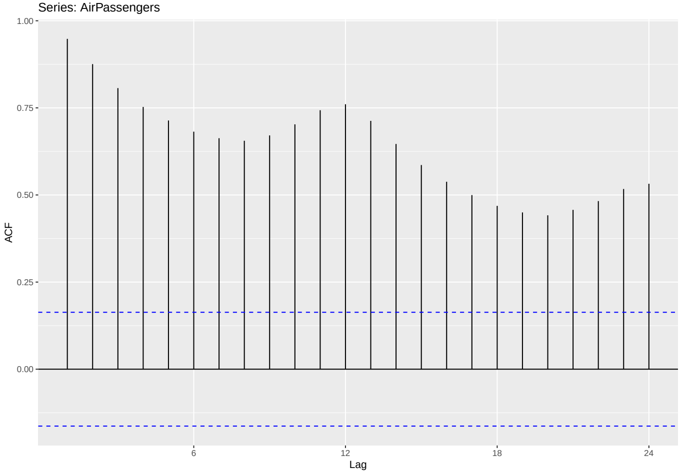
```r
ggAcf(AirPassengers)
```



Time Series: autocorrelation

Trend
When data have a trend, the autocorrelations for small lags tend to be large and positive because observations nearby in time are also nearby in size. So the ACF of trended time series tend to have positive values that slowly decrease as the lags increase.

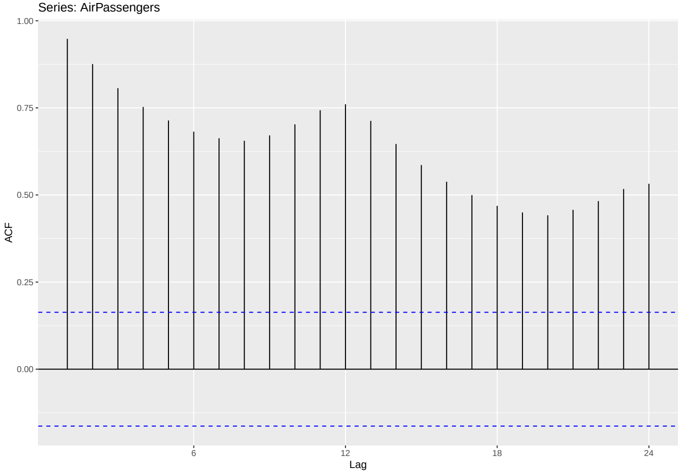
Time Series: autocorrelation



Time Series: autocorrelation

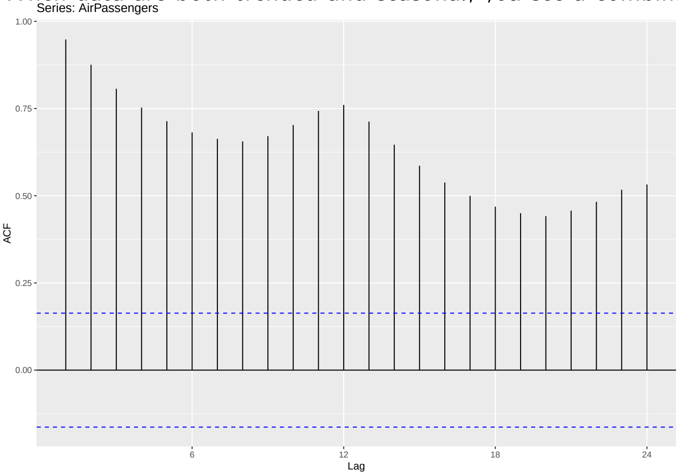
Seasonality
When data are seasonal, the autocorrelations will be larger for the seasonal lags (at multiples of the seasonal frequency) than for other lags.

Time Series: autocorrelation



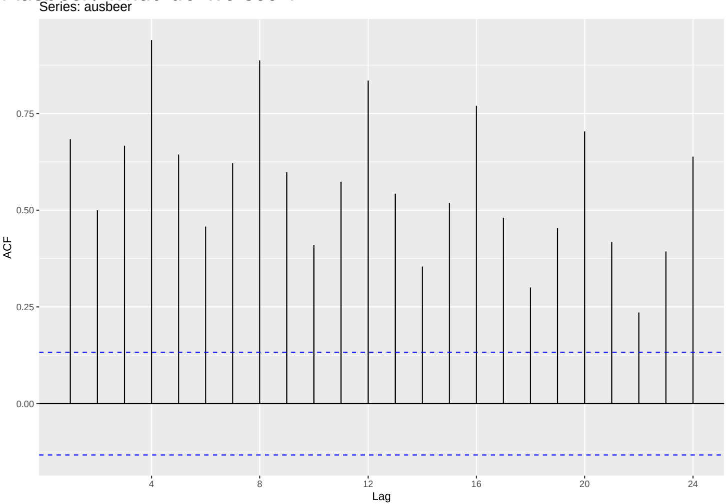
Time Series: autocorrelation

When data are both trended and seasonal, you see a combination of these effects.

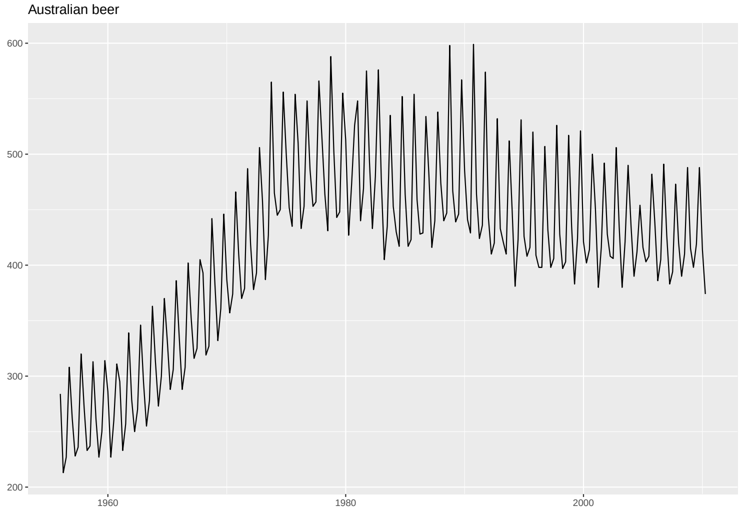


Time Series: autocorrelation

Ausbeer. what do we see ?



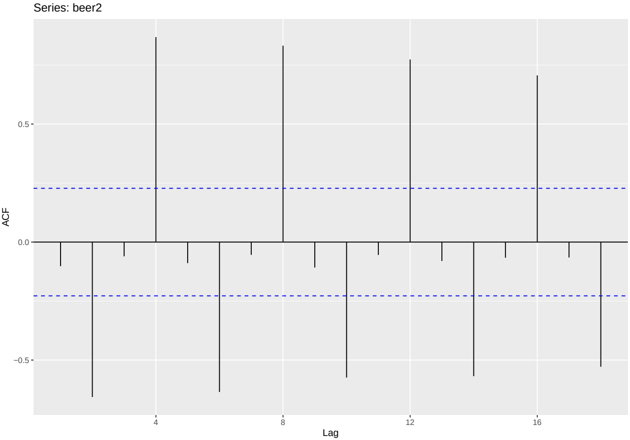
Time Series: autocorrelation



Time Series: autocorrelation

Lets look at the second part of the plot only, starting from 1992, use function window()

```
beer2 <- window(ausbeer, start=1992)
ggAcf(beer2)
```



Case: California energy consumption

california.rda contains data on hourly energy consumption in California for the period of 2016-01-01 to 2019-12-31.
The sampling frequency is an **hour**
You can get more energy related data using *library(eia)* or from the website of US Energy Information Agency

Case: California energy consumption

value shows energy demand per hour in megawatt

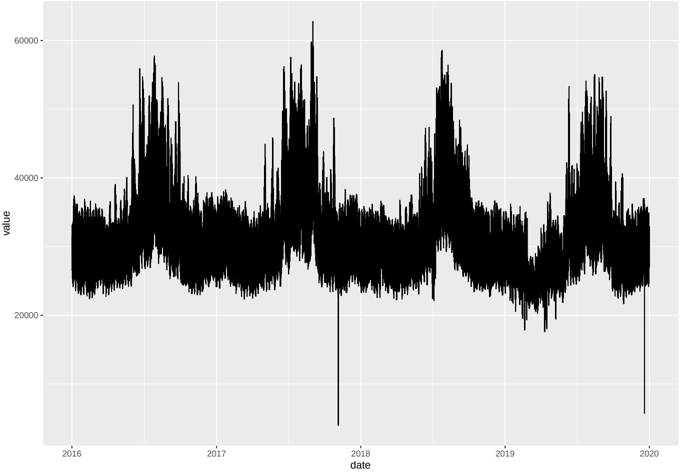
```
load('Data/california.rda')
summary(california)
```

```
##       date                          value
##  Min.   :2016-01-01 00:00:00   Min.   : 3964
##  1st Qu.:2016-12-31 05:45:00   1st Qu.:26991
##  Median :2017-12-31 11:30:00   Median :30584
##  Mean   :2017-12-31 11:30:00   Mean   :31603
##  3rd Qu.:2018-12-31 17:15:00   3rd Qu.:34084
##  Max.   :2019-12-31 23:00:00   Max.   :62787
```

Case: California energy consumption

plot the demand

```
ggplot(california, aes(date, value)) + geom_line()
```

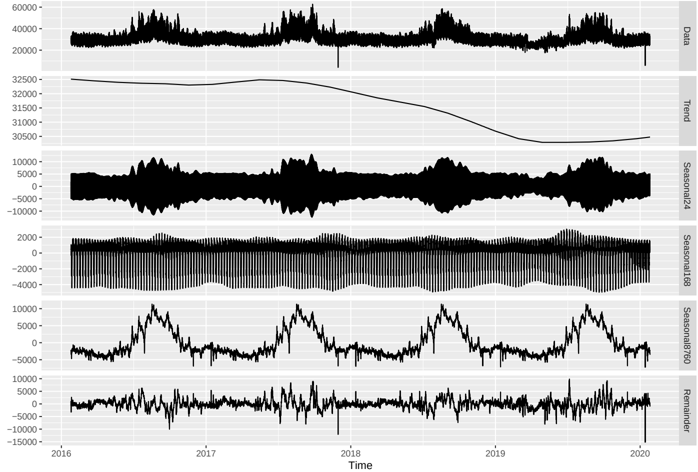


Case: California energy consumption

Take a moment to think what are the seasonal periods in this data ?

Case: California energy consumption

```r
cali_ts <- msts(california$value, seasonal.periods=c(24,24*7,24*365),
                start = 2016 + 1/365*24)
mstl(cali_ts) %>% autoplot()
```



Case: California energy consumption

Autocorrelation plot

```
ggAcf(cali_ts, lag.max = 48)
```