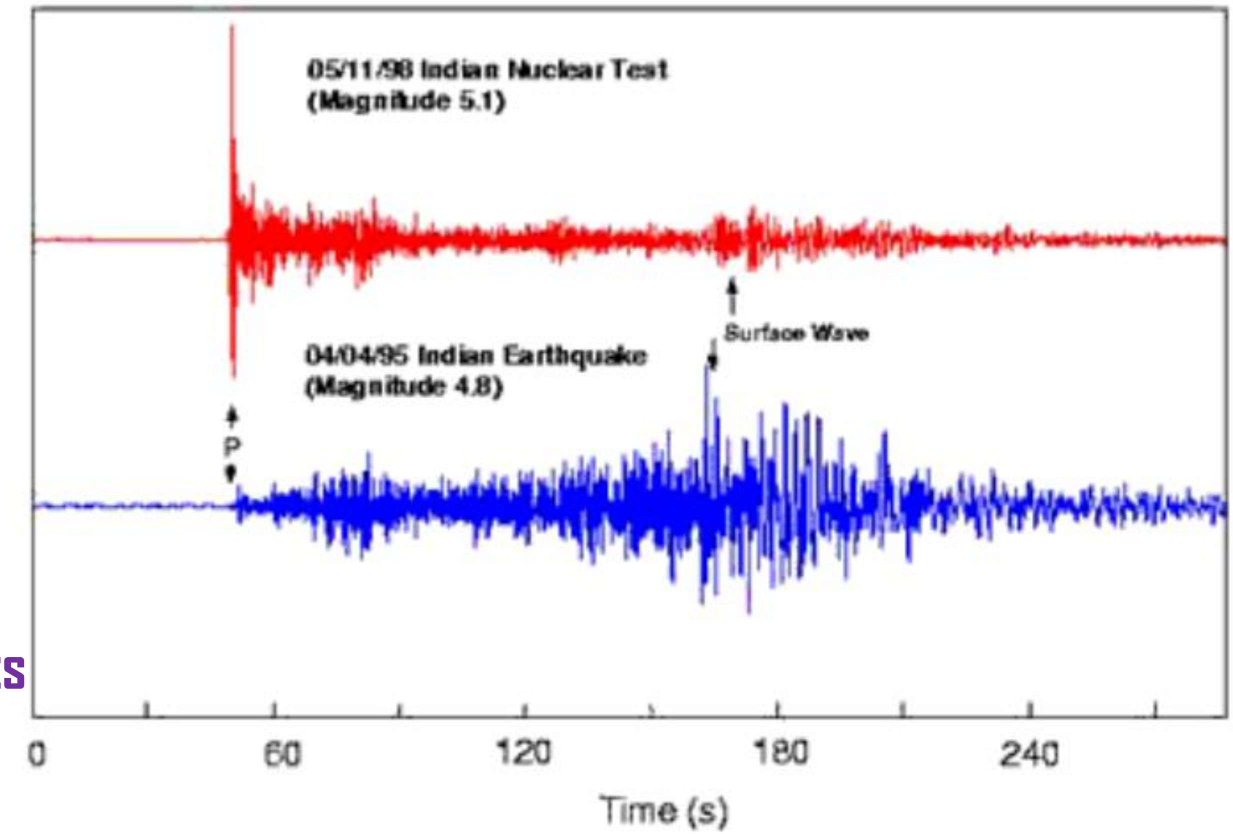
Time Series Analysis

References:
[1] F. Anderson, Time Series Analysis by Example : Hands on approach using R
[2] M. O. Adennomon, An Introduction to Univariate and Multiple Time Series Analysis with R (Lambert Academic Pub, Saarbrücken, 2015)

Indentifying Explosions and Earthquakes

Data recorded at Nilore, Pakistan



1. Introduction to Time Series
2. Creating Time Series Objects
3. Time Series Visualization
4. Time Series Stationarity
5. Time Series and Forecasting
6. Financial Time Series

http://www.lanl.gov/orgs/ees/ees11/geophysics/gnem/expseis.shtm

1. Introduction to Time Series

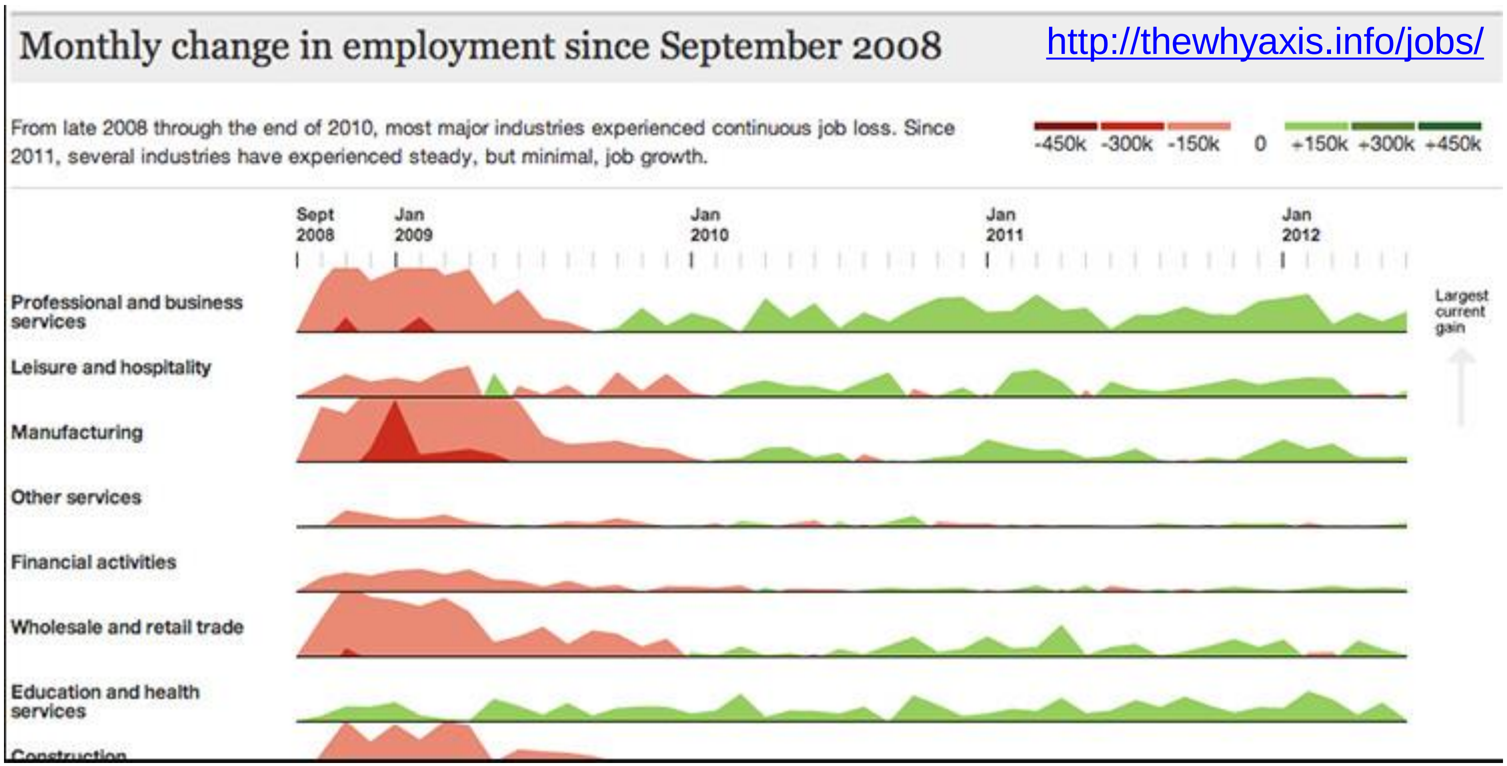
► A **time series** is a sequence of data points ordered in time.

A time series is defined by the values $y_1, y_2, \cdots, y_n$ of variable Y at times $t_1, t_2, \cdots, t_n$, thus y is a function of t: $y = f(t)$.

► **Measurement of water level in Nile River**

► **Time series analysis aims to draw inferences, forecast future values, and control the series.**

- **Identifying the patterns of trends and seasonal variation in correlated data**
- **Understanding and modeling the data for explanation**
- **Predicting short-term trends based on previous patterns**



►What are components of time series data?

1. Trend (T_t)

- A trend exists when there is a long-term increase or decrease in the data.

2. Seasonal variation (S_t)

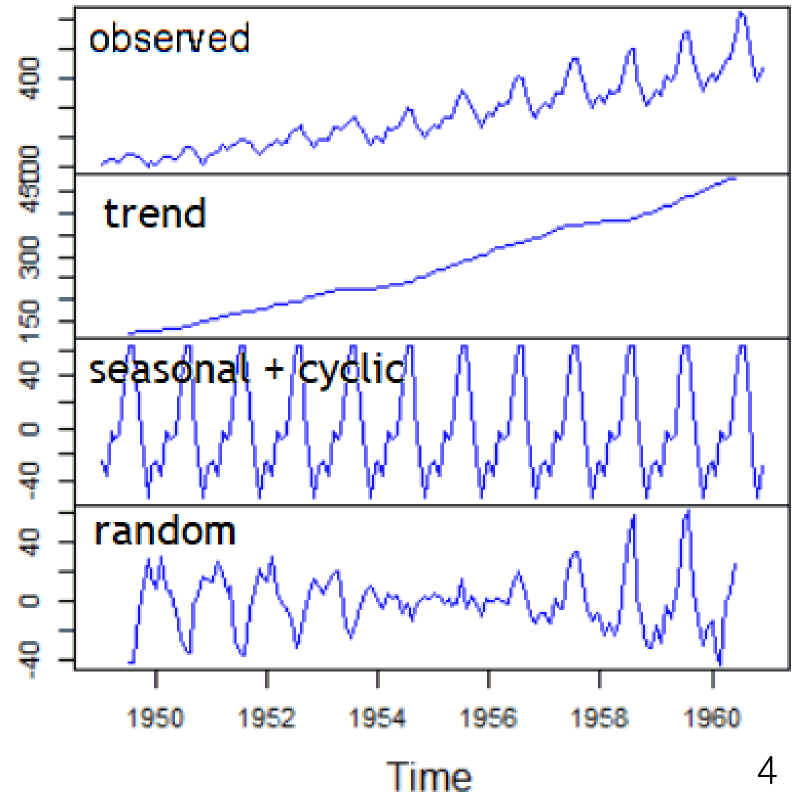
- A seasonal pattern exists when a series is influenced by seasonal factors (ex: quarterly, monthly, half-yearly).

3. Cyclical variation (C_t)

- A cyclic pattern exists when data exhibit rises and falls that are not of fixed period.

4. Irregular or random variation (I_t)

- The variation of observations in a time series which is unusual or unexpected. Ex: Floods, fires, earthquakes, revolutions, epidemics, strikes, ...



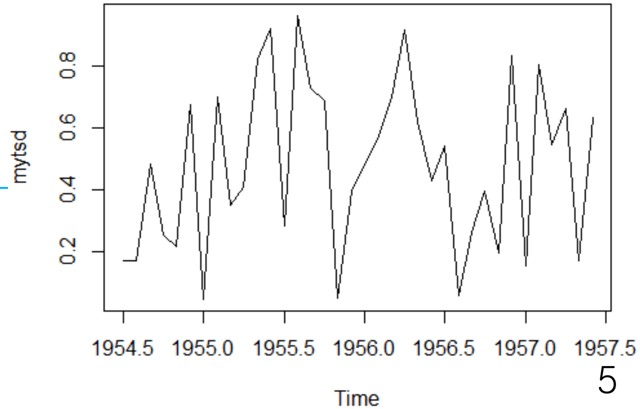
2. Creating Time Series Objects

(1) Creating a simple time series

The **ts()** function will convert a numeric vector into an R time series object

ts(data, start, end, frequency, …) {stats}
The function ts is used to create time-series objects.
frequency: the number of observations per unit time
(1=annual, 4=quartly, 12=monthly, etc.)

```
> ## Using May 1954 as start date:
> x <- 1:36; td <- runif(x)
> mytsd <- ts(td, start = c(1954,7), frequency = 12)
> mytsd
            Jan        Feb        Mar        Apr        May        Jun        Jul
1954                                                                   0.17064524
1955 0.04766363 0.70085309 0.35188864 0.40894400 0.82095132 0.91885735 0.28252833
1956 0.47784538 0.56025326 0.69826159 0.91568354 0.61835123 0.42842151 0.54208037
1957 0.15288722 0.80341854 0.54682616 0.66231764 0.17169849 0.63305536
            Aug        Sep        Oct        Nov        Dec
1954 0.17217175 0.48204261 0.25296493 0.21625479 0.67437639
1955 0.96110479 0.72839443 0.68637508 0.05284394 0.39522013
1956 0.05847849 0.26085686 0.39715195 0.19774474 0.83192756
1957
> plot(mytsd) # using 'plot.ts' for time-series plot
```



(2) Sample Dataset: hflights

flights {flights} Houston flights data

This dataset contains all flights departing from Houston airports IAH and HOU. Data from: http://www.transtats.bts.gov/DatabaseInfo.asp?DB_ID=120&Link=0


```
> #(1) Reading time series data
> library(hflights)
> head(hflights,2)
     Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier
5424 2011     1          1         6    1400    1500            AA
5425 2011     1          2         7    1401    1501            AA
     FlightNum TailNum ActualElapsedTime AirTime ArrDelay DepDelay Origin
5424       428  N576AA                60      40      -10        0    IAH
5425       428  N557AA                60      45       -9        1    IAH
     Dest Distance TaxiIn TaxiOut Cancelled CancellationCode Diverted
5424  DFW      224      7      13         0                         0
5425  DFW      224      6       9         0                         0
```

Let's transform the data.frame to data.table for easy aggregation.
Then we create a **date** variable from the provided Year, Month, and DayofMonth columns.

ISOdate(year, month, day, ...) {base}

Date-time conversion function from numeric representations

```
> library(data.table)
> dt <- data.table(hflights)
> dt[, date := ISOdate(Year, Month, DayofMonth)]
> head(dt,2)
   Year Month DayofMonth DayOfWeek DepTime ArrTime UniqueCarrier FlightNum TailNum
1: 2011     1          1         6    1400    1500            AA       428  N576AA
2: 2011     1          2         7    1401    1501            AA       428  N557AA
   ActualElapsedTime AirTime ArrDelay DepDelay Origin Dest Distance TaxiIn TaxiOut
1:                60      40      -10        0    IAH  DFW      224      7      13
2:                60      45       -9        1    IAH  DFW      224      6       9
   Cancelled CancellationCode Diverted                date
1:         0                         0 2011-01-01 12:00:00
2:         0                         0 2011-01-02 12:00:00
```

date column was created!

We can compute **the total number of flights**, **the overall sum of arrival delays**, **the number of cancelled flights, and the average distance of the related flights** for each day in 2011.

```
> daily <- dt[, list( N = .N,
+   Delays    = sum(ArrDelay, na.rm=TRUE),
+   Cancelled = sum(Cancelled),
+   Distance  = mean(Distance) ), by=date]
> head(daily)
                 date   N Delays Cancelled Distance
1: 2011-01-01 12:00:00 552   5507         4 827.0761
2: 2011-01-02 12:00:00 678   7010        11 786.7788
3: 2011-01-03 12:00:00 702   4221         2 772.3276
4: 2011-01-04 12:00:00 583   4631         2 754.5523
5: 2011-01-05 12:00:00 590   2441         3 759.5441
6: 2011-01-06 12:00:00 660   3994         0 755.5955
```

Group by ‘date’ and count instances (.N)

Ref: G. Daroczi, Mastering Data Analysis with R (Packt Pub., Birmingham, 2015) Chap. 12.

3. Time Series Visualization

▪A time series plot is a graph that you can use to evaluate patterns and behavior in data over time.

▪A time series plot displays observations on the y-axis against equally spaced time intervals on the x-axis.

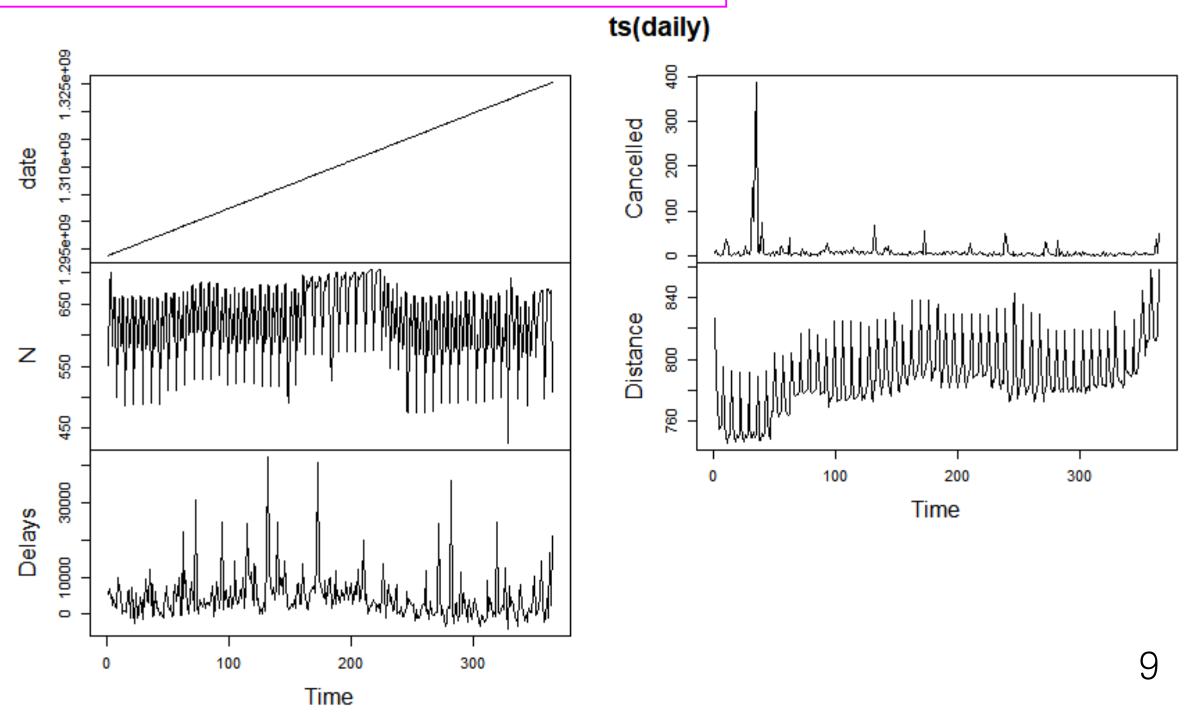
setorder(x, …) {data.table}

Fast row reordering of a data.table by reference

ts(data, start, end, frequency, deltat, …) {stats}

The function ts is used to create time-series objects.

```
#sorting daily data by date
setorder(daily,date)
plot(ts(daily))
```



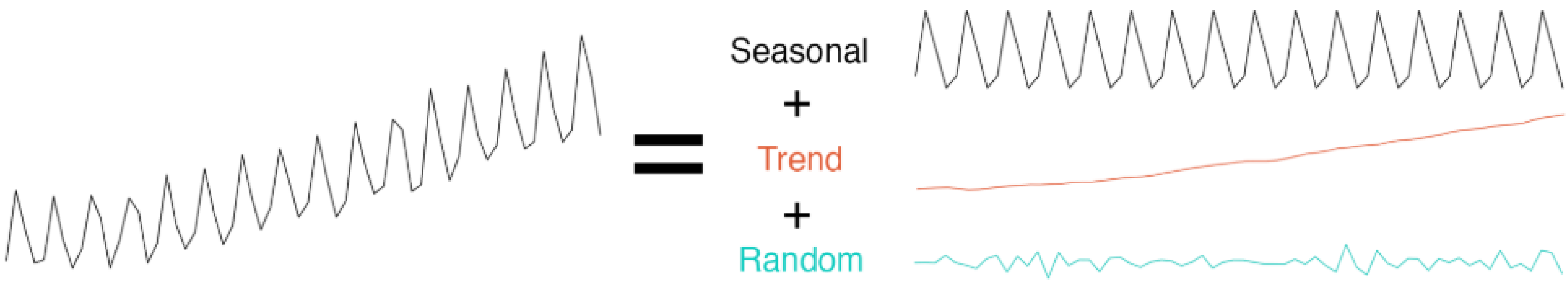
The x axis is indexed from 1 to 365 and the date transformed to timestamps on the y axis.

•Decomposition of Time Series Data

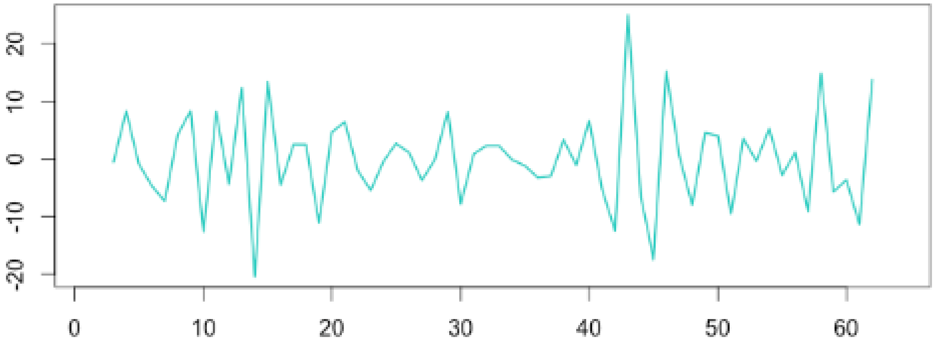
Additive Model: $$Y_t = T_t + C_t + S_t + \epsilon_t$$

Multiplicative Model: $$Y_t = T_t \times C_t \times S_t \times \epsilon_t$$

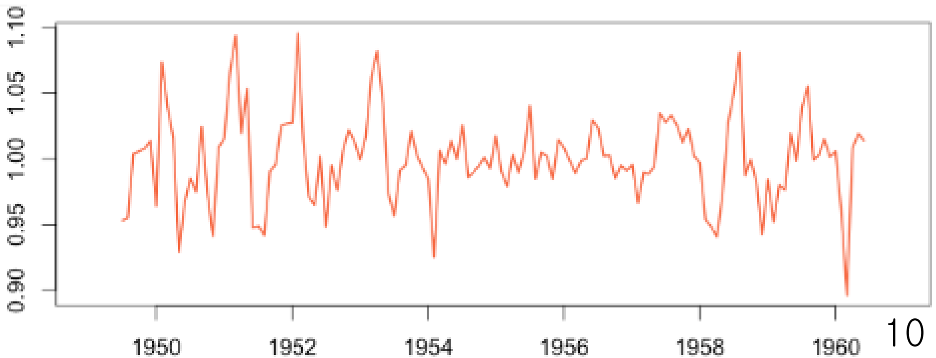
decompose(x, type=c("additive", "multiplicative"), filter=NULL) {stats}
Decompose a time series into seasonal, trend and irregular components using moving averages. Deals with additive or multiplicative seasonal component.



type = "additive"
Time series = Seasonal + Trend + Random

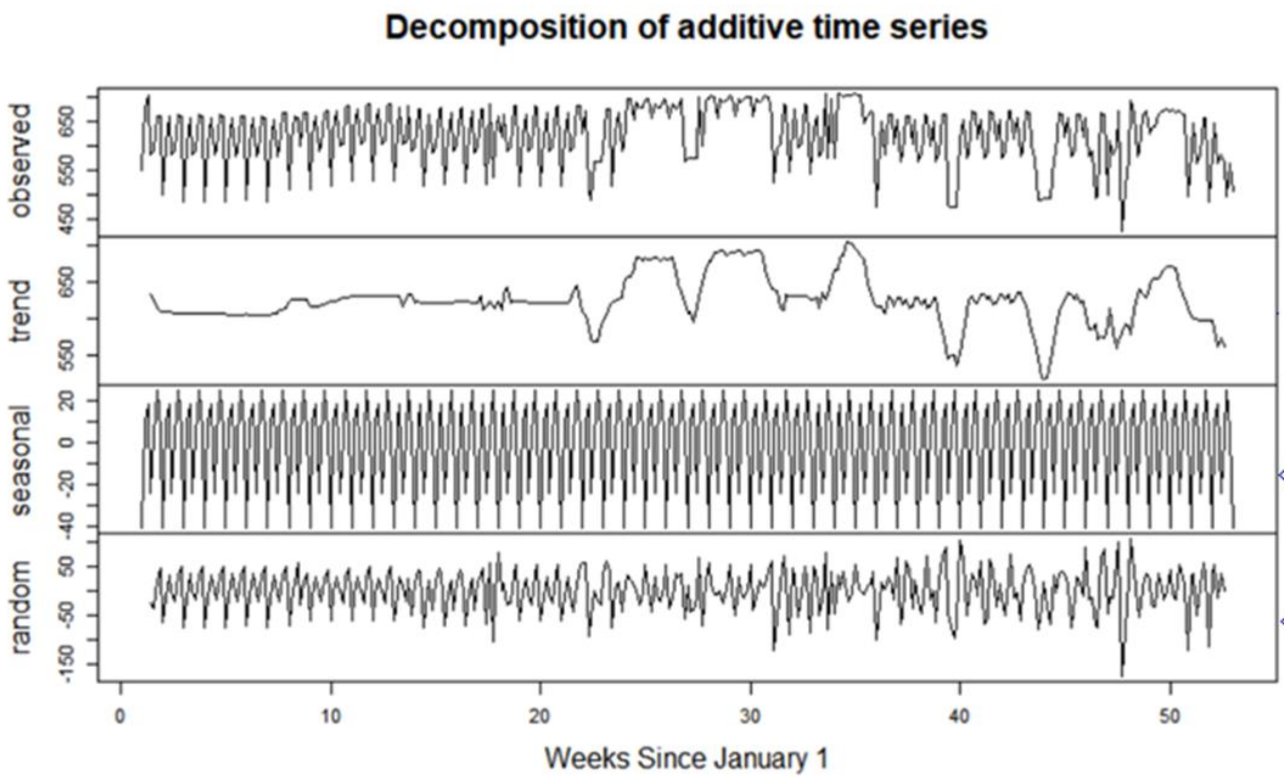


type = "multiplicative"
Time series = Seasonal * Trend * Random



•Decomposition of Time Series Data

```r
daily7 <- decompose(ts(daily$N, frequency=7))
plot(daily7)
```



effect on time series periodically

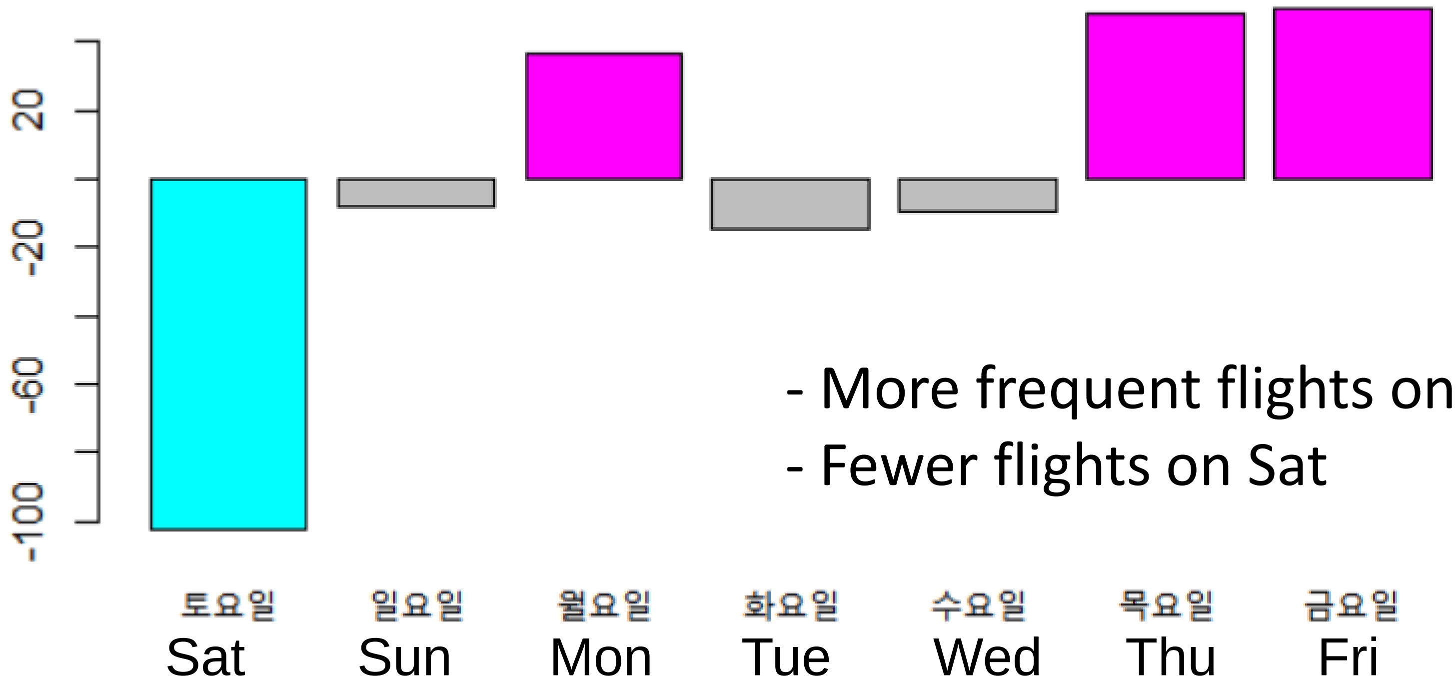
white noice on the data

In the **trend** plot: Peak interval between 25 and 35 weeks (summertime)
Lowest number of flights on 46th week (probably due to Thanksgiving Day)

Weekly Seasonality

```
> weeks = setNames(daily7$figure, weekdays(daily$date[1:7]))
> weeks
   Saturday      Sunday      Monday   Tuesday  Wednesday
     토요일        일요일        월요일      화요일        수요일
-102.171776   -8.051328   36.595731 -14.928941   -9.483886
   Thursday      Friday
     목요일        금요일
  48.335226   49.704974
```

```
barplot(weeks,col=c('cyan','gray','magenta',
        'gray','gray','magenta','magenta'))
```



- More frequent flights on Mon, Thu, Fri
- Fewer flights on Sat

4. Time Series Stationarity

- Stationarity is prerequisite for time series modeling.
- Data in stationarity time series do not depend on time.
- So it does not have a trend or seasonality.

Three conditions of time series stationarity:

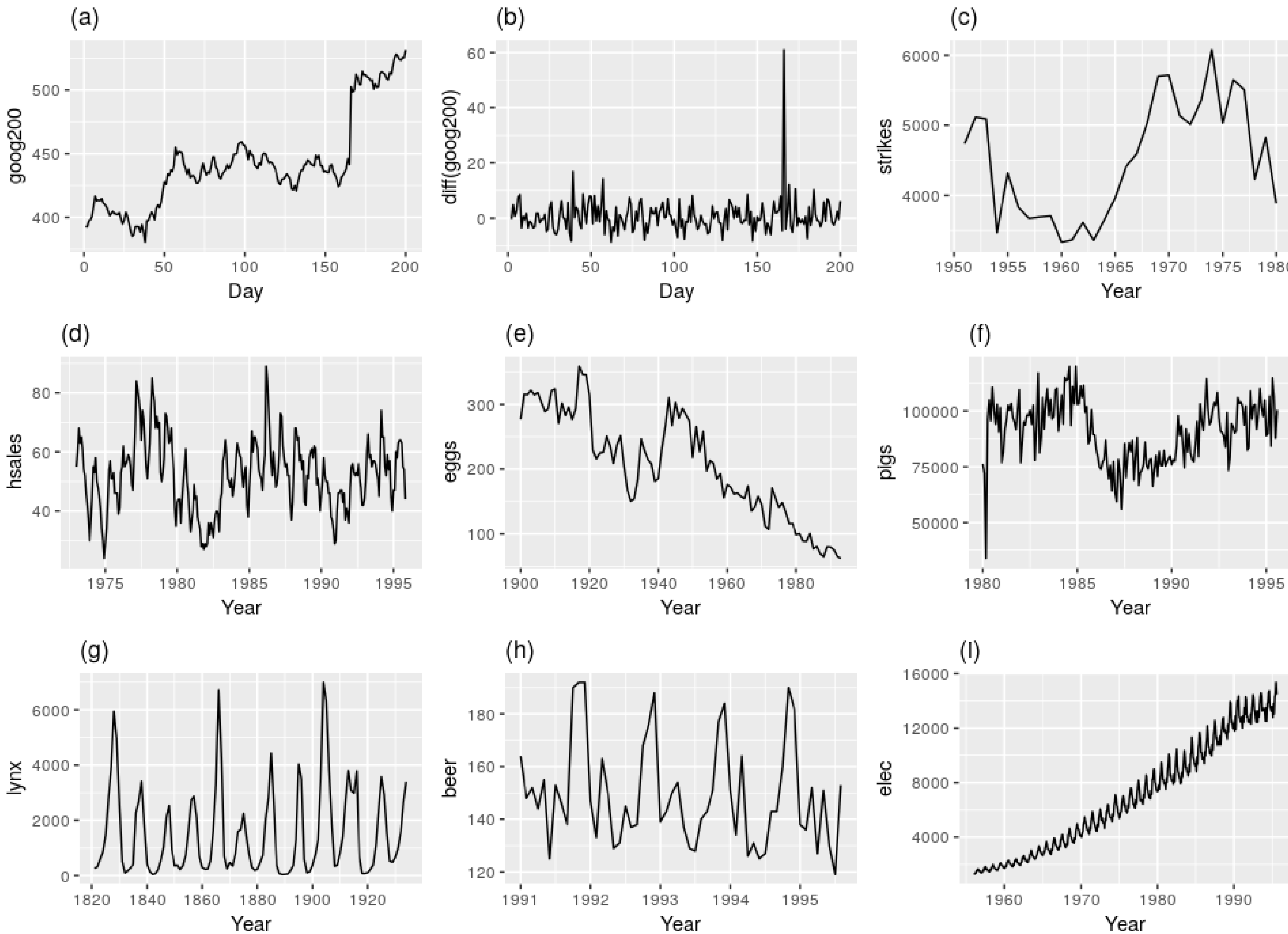
(1) **The mean value of time-series is constant over time,** which implies, the trend component is nullified.

(2) **The variance does not increase over time.**

(3) **Seasonality effect is minimal.**

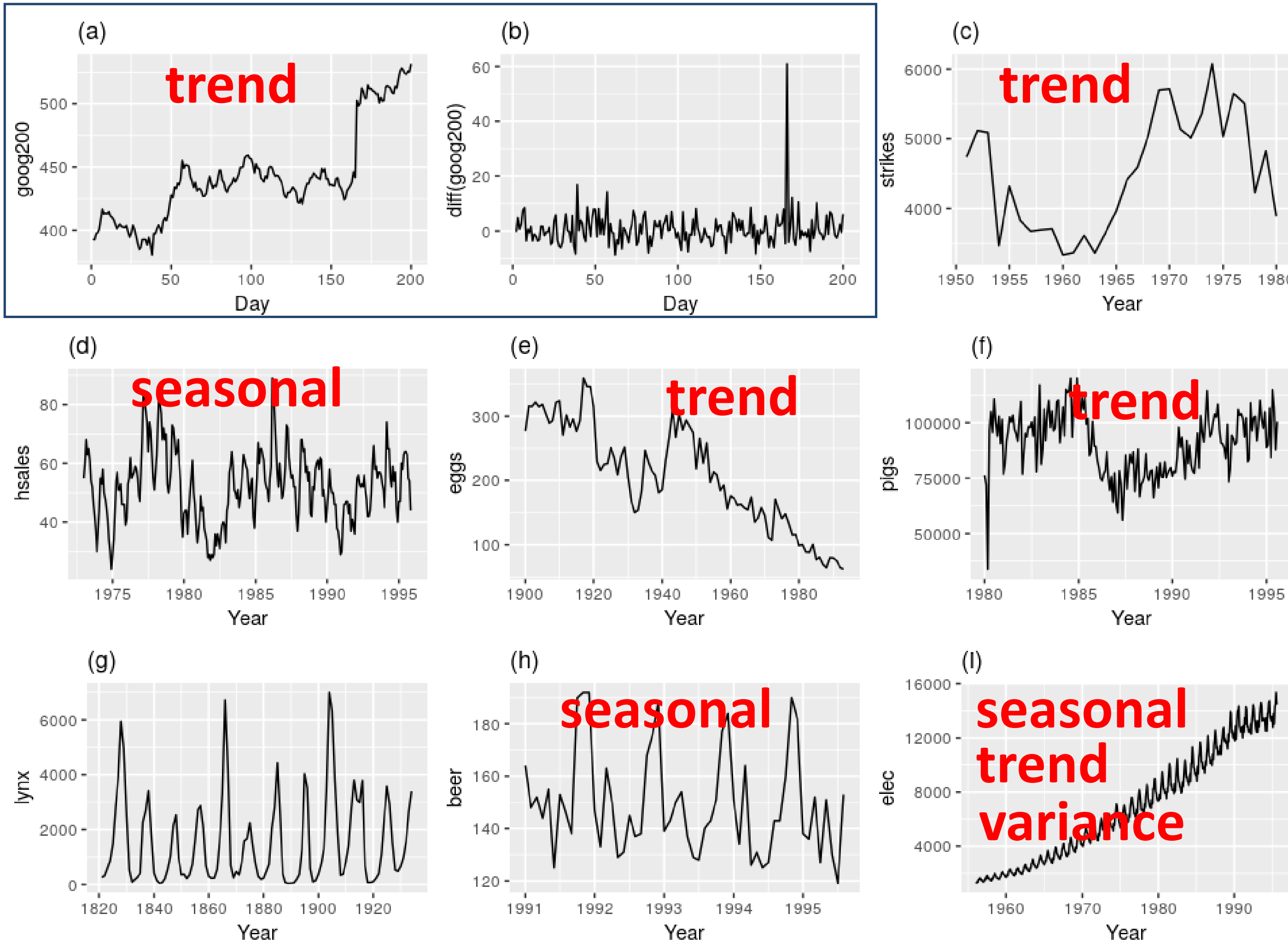
[Reference]
http://r-statistics.co/Time-Series-Analysis-With-R.html

Which of these series are stationary?

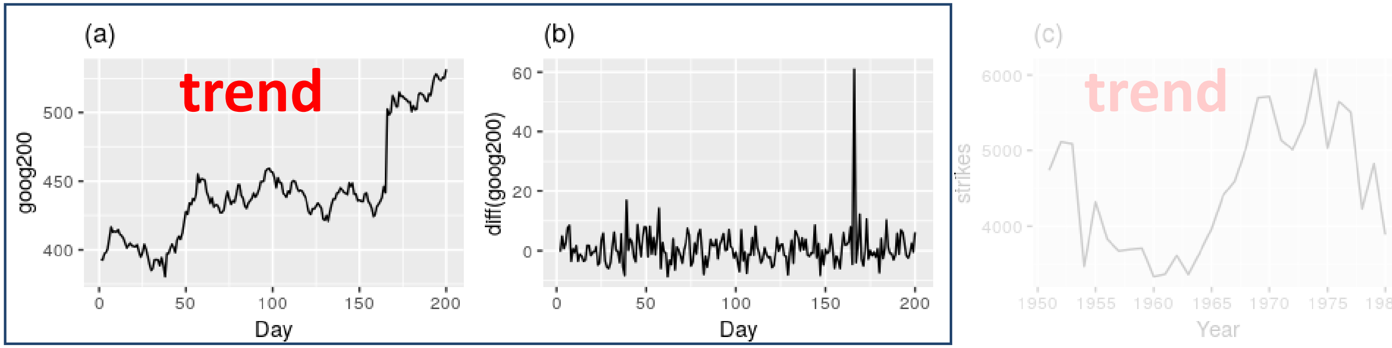


Stationary time series

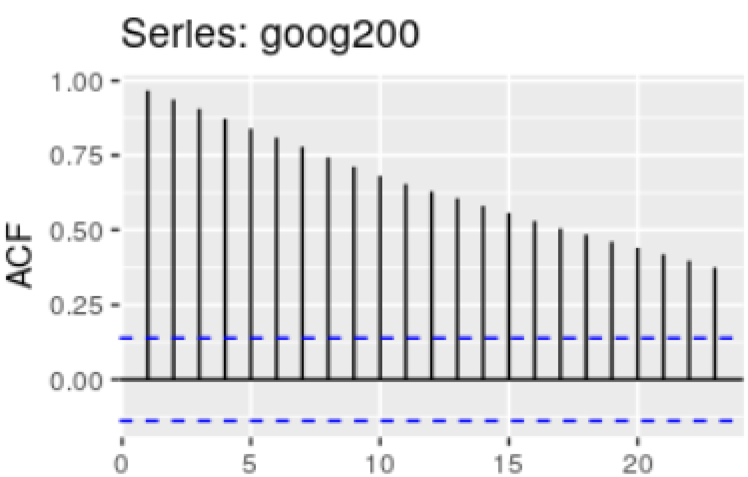
Which of these series are stationary?

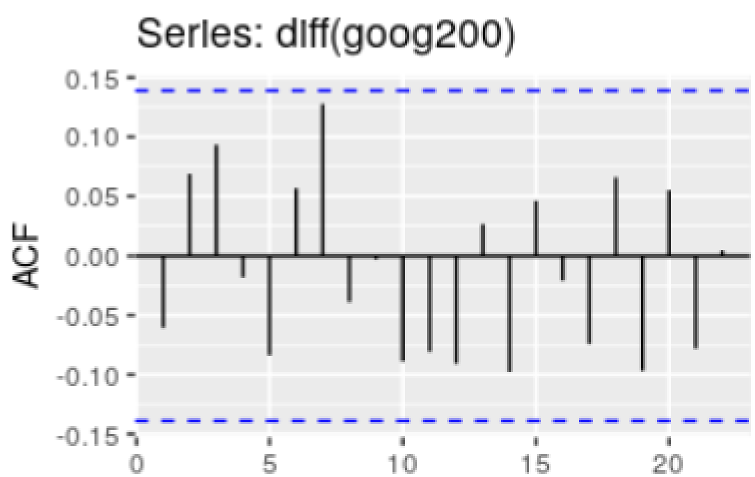


Which of these series are stationary?



The raw stock price data in (a) is non-stationary, yet its daily price difference shown in (b) is stationary. This process is called **differencing**.





ACF (autocorrelation function) plots can be used to visualize this.

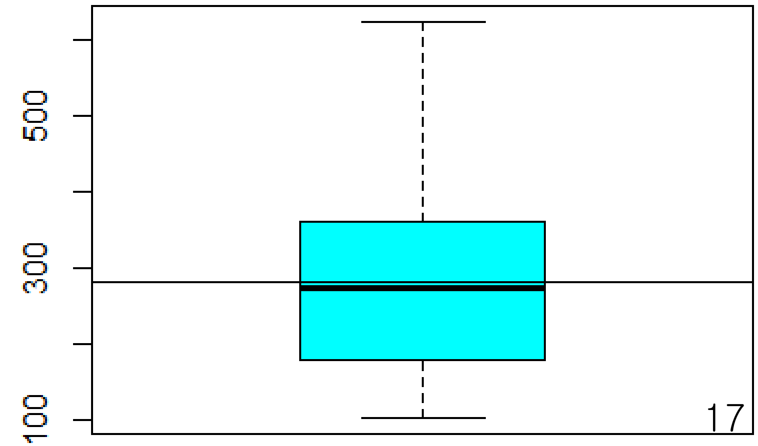
Sample Dataset: AirPassengers

AirPassengers {datasets}

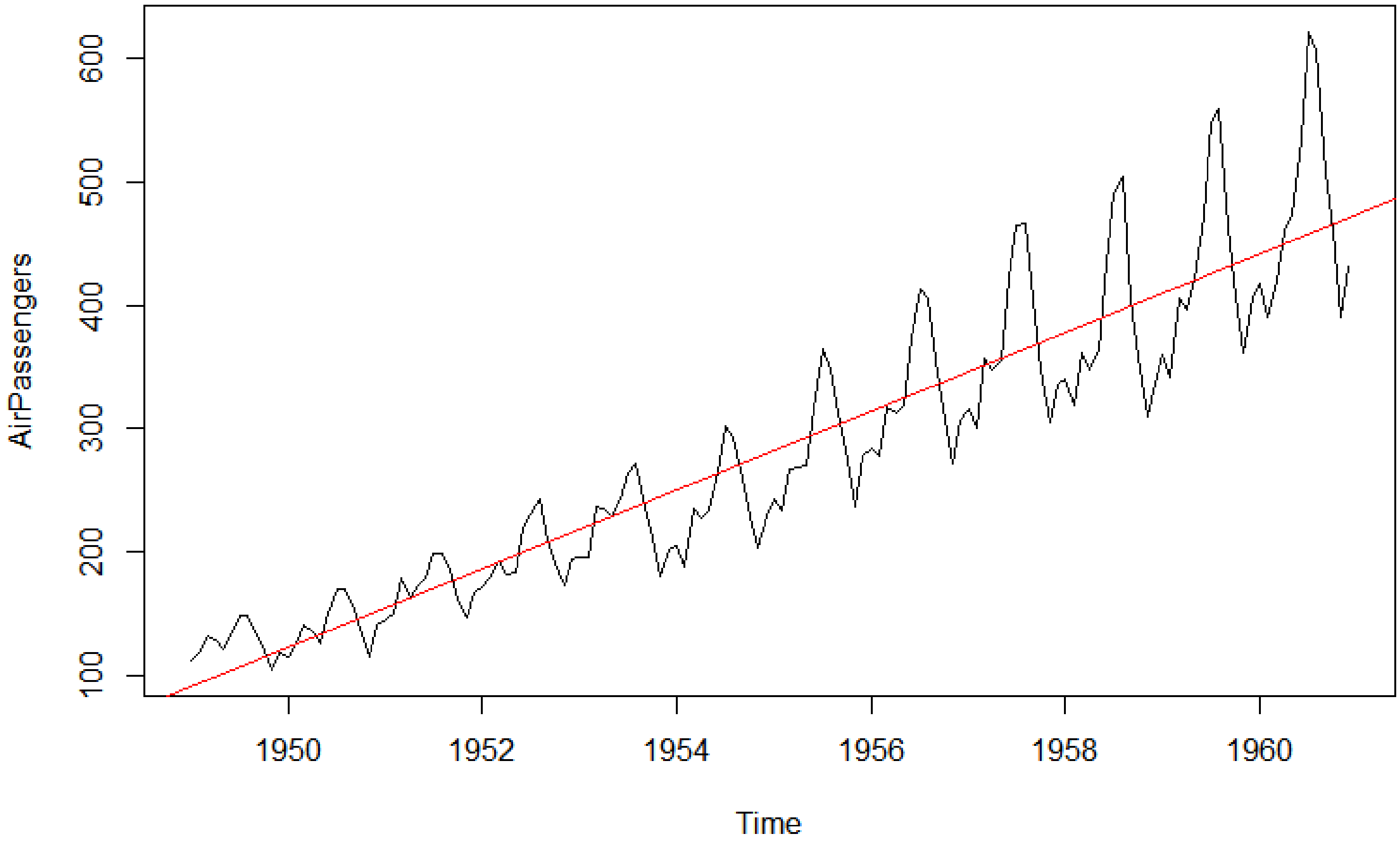
Monthly Airline Passenger Numbers 1949-1960

The classic Box & Jenkins airline data. Monthly totals of international airline passengers, 1949 to 1960.

```
> class(AirPassengers)
[1] "ts"
> start(AirPassengers)
[1] 1949    1
> end(AirPassengers)
[1] 1960   12
> sm <- summary(AirPassengers); sm
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  104.0   180.0   265.5   280.3   360.5   622.0 
> boxplot(sm, col='cyan')
> abline(h=sm[4])
```



```
plot(AirPassengers)
abline(reg=lm(AirPassengers~time(AirPassengers)),
         col='red')
```



Plotting the data shows: (1) the numbers are increasing and (2) there exists a seasonality pattern that repeats every year.

- Autocorrelation Function (ACF) plot shows the *correlation* of the series with *itself* at different lags. Autocorrelation of Y at lag *h* is the correlation between *Y* and *LAG(Y,h)*

$$\gamma_X(t+h,t) = \mathrm{Cov}(X_{t+h}, X_t)$$

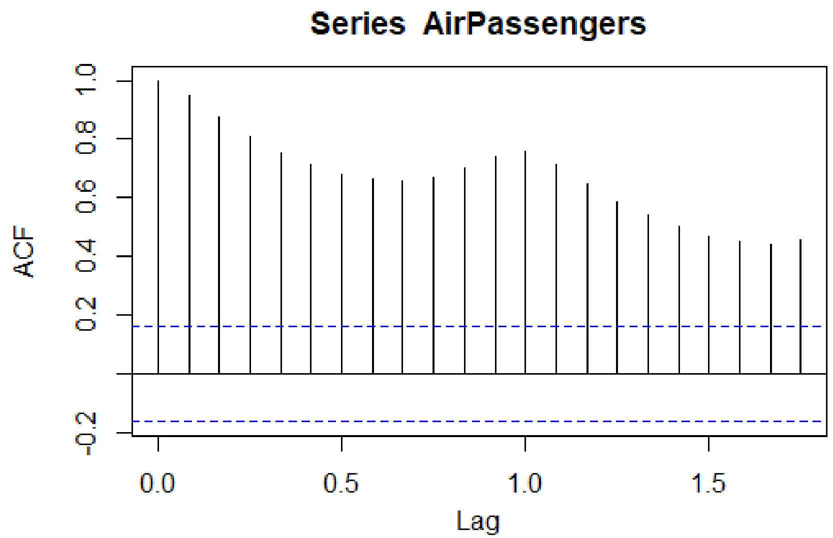
$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = \mathrm{Corr}(X_{t+h}, X_t)$$

- Partial autocorrelation function (PACF) – measures correlation between (time series) observations that are k time periods apart after controlling for correlations at intermediate lags (i.e., lags less than *k*). In other words, it is the correlation between Y_t and Y_{t-k} after removing the effects of intermediate *Y*'s.

Autocorrelation Function

Autocorrelation function (ACF) is the correlation between series values that are *k* intervals apart at lag k.

```
# autocorrelation
acf(AirPassengers)
```



In R acf starts with lag 0, that is the correlation of a value with itself.

The blue dashed lines represent the confidence limits (insignificant zone)

Identifying non-stationary series

- The ACF of stationary data drops to zero relatively quickly.
- The ACF of non-stationary data decreases slowly.

Unit root tests

kpss.test(x, null = c("Level", "Trend") {tseries}
KPSS Test for Stationarity
Computes the Kwiatkowski-Phillips-Schmidt-Shin (KPSS) test for the null hypothesis that x is level or trend stationary.

```
> kpss.test(AirPassengers)

        KPSS Test for Level Stationarity

data:  AirPassengers
KPSS Level = 2.7395, Truncation lag parameter = 4, p-value = 0.01
```

Interpretation: AirPassengers dataset is *not stationary* because it has both a trend and seasonality. The p-value for the test is less than 0.05. Thus we reject the null hypothesis of stationary.

Differencing

- Differencing can help stabilize the mean of a time series by removing changes in the level of a time series, and therefore reducing trend and seasonality.
- The diff() function can make a non-stationary time series stationary.

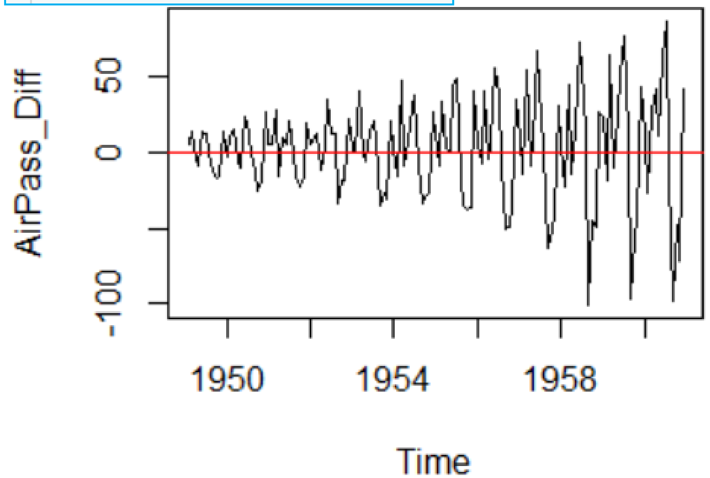
diff(x, ...) {base}
Returns suitably lagged and iterated differences.

```
> ## Differencing
> AirPass_Diff <- diff(AirPassengers)
> kpss.test(AirPass_Diff)

        KPSS Test for Level Stationarity

data:  AirPass_Diff
KPSS Level = 0.014626, Truncation lag parameter = 4,
p-value = 0.1
```

```
plot(AirPass_Diff)
abline(h=0,col=2)
```



This result shows stationary.

- p-value = 0.1
- The mean and std variations have small variations with time.

(1) Reading Time Series Data

co2 {datasets}

Mauna Loa Atmospheric CO2 Concentration

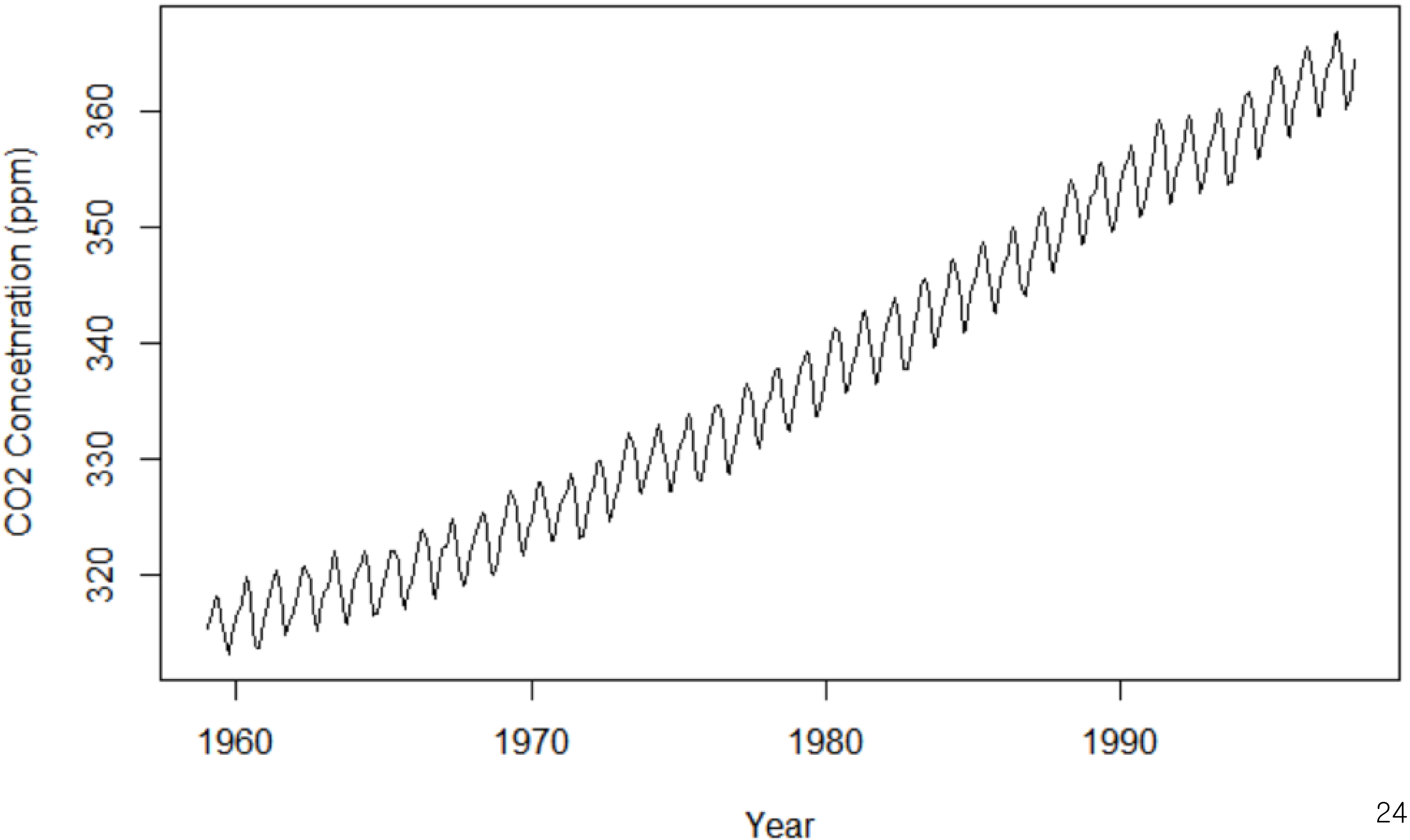
A time series of 468 observations; monthly from 1959 to 1997.

Atmospheric concentrations of CO2 are expressed in parts per million (ppm).

Source: C. D. Keeling and T. P. Whorf, Scripps Institution of Oceanography (SIO), University of California, La Jolla, California USA 92093-0220.

```
> co2 #time series data
        Jan    Feb    Mar    Apr    May    Jun    Jul    Aug    Sep    Oct    Nov    Dec
1959 315.42 316.31 316.50 317.56 318.13 318.00 316.39 314.65 313.68 313.18 314.66 315.43
1960 316.27 316.81 317.42 318.87 319.87 319.43 318.01 315.74 314.00 313.68 314.84 316.03
1961 316.73 317.54 318.38 319.31 320.42 319.61 318.42 316.63 314.83 315.16 315.94 316.85
1962 317.78 318.40 319.53 320.42 320.85 320.45 319.45 317.25 316.11 315.27 316.53 317.53
1963 318.58 318.92 319.70 321.22 322.08 321.31 319.58 317.61 316.05 315.83 316.91 318.20
1964 319.41 320.07 320.74 321.40 322.06 321.73 320.27 318.54 316.54 316.71 317.53 318.55
...
1996 362.09 363.29 364.06 364.76 365.45 365.01 363.70 361.54 359.51 359.65 360.80 362.38
1997 363.23 364.06 364.61 366.40 366.84 365.68 364.52 362.57 360.24 360.83 362.49 364.34
```

```r
# plot co2 time series
plot(co2, xlab='Year', ylab='CO2 Concetnration (ppm)')
```



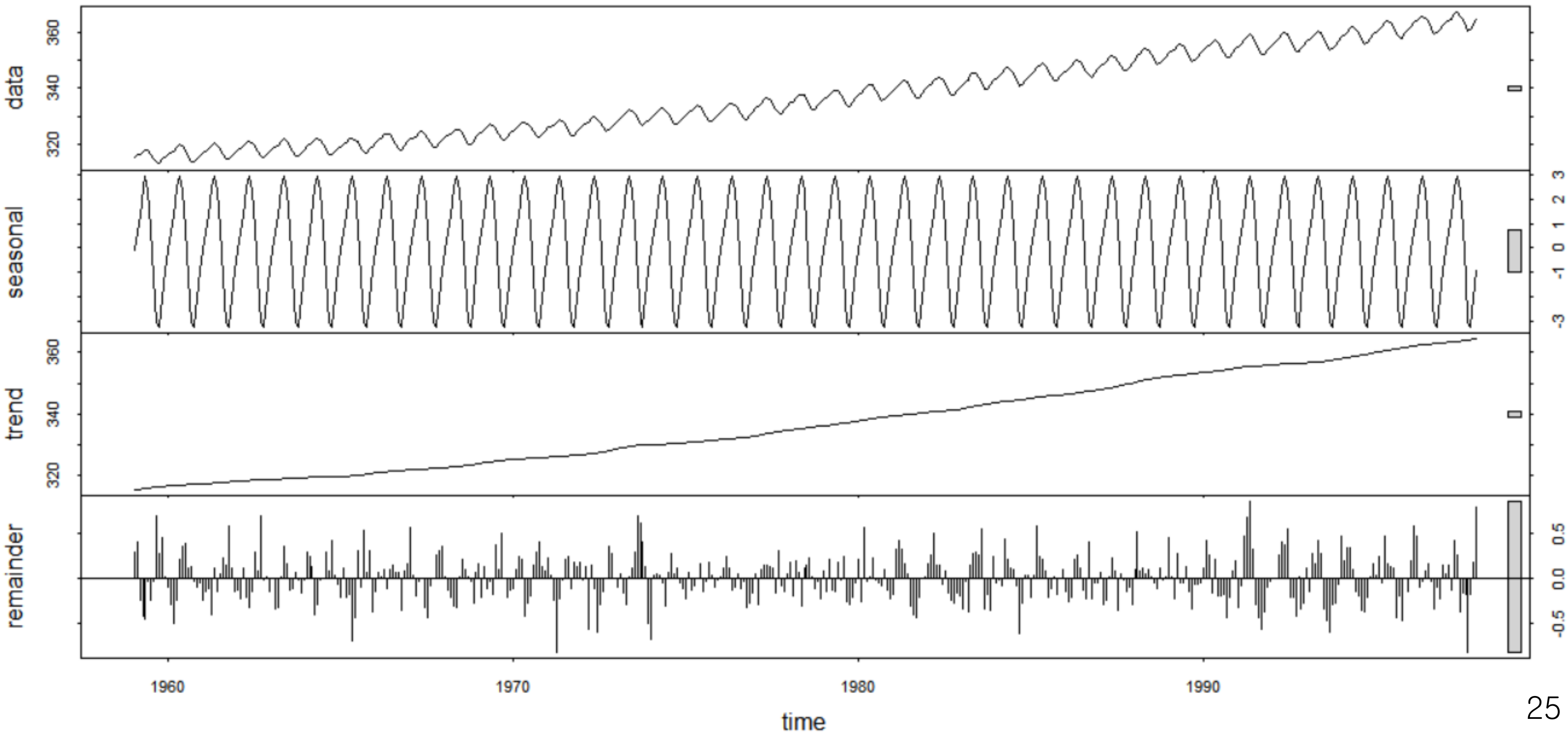
(2) Seasonal Decomposition

stl(x, s.window, ...) {stats}

Seasonal Decomposition of Time Series by Loess

Decompose a time series into seasonal, trend and irregular components using loess (locally weighted smoothing).

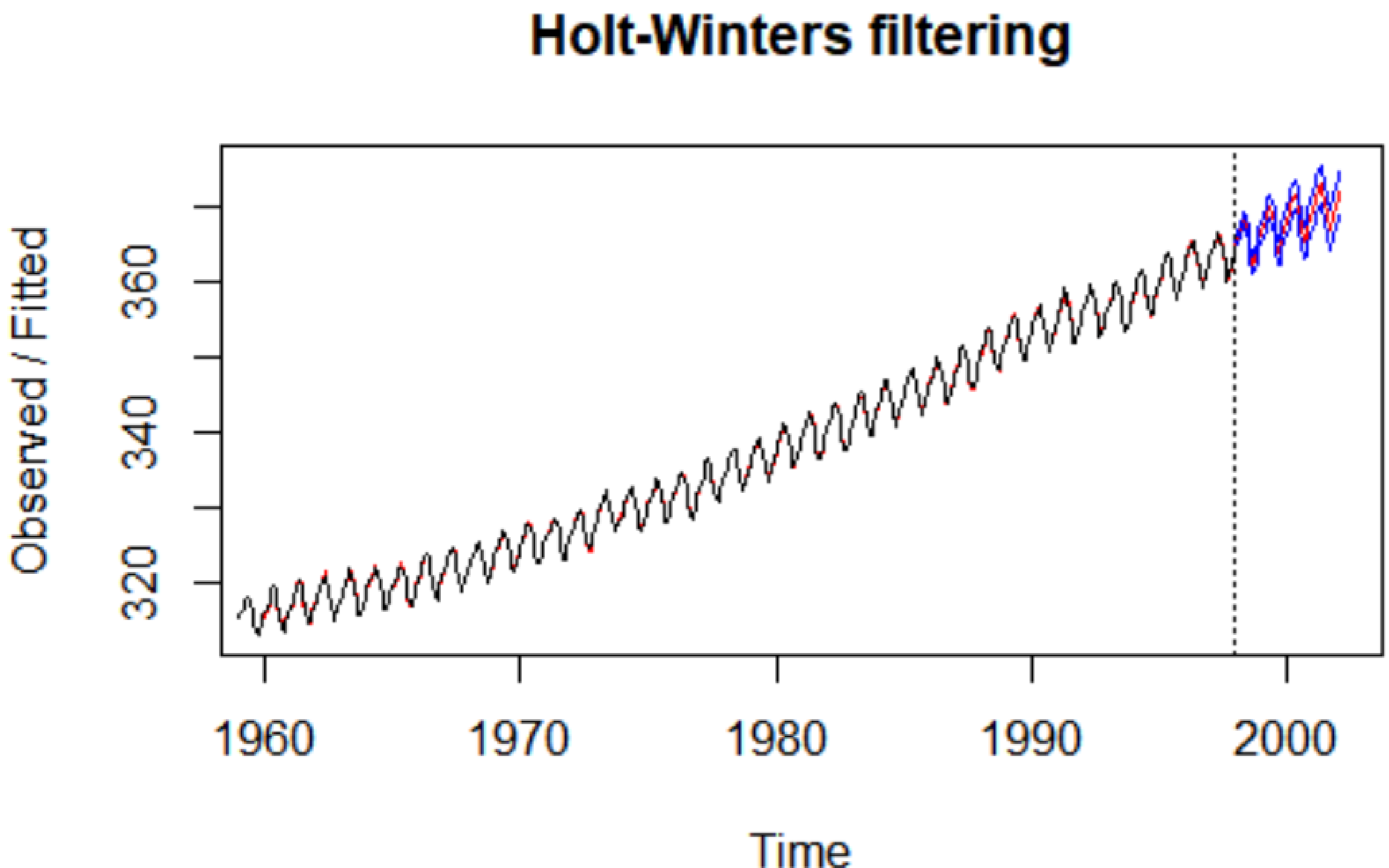
```
fit <- stl(co2, s.window="period")
plot(fit)
```



(3) Forecasting Time Series

We can now forecast future values based on the smoothing model.

```
library(forecast)
m <- HoltWinters(co2)
p <- predict(m, 50, prediction.interval = TRUE)
plot(m, p)
```



(1) Reading Stock Dataset

The getSymbols function from the quantmod package is an easy and convenient way to bring historical stock prices into your R environment.

getSymbols(Symbols, src, auto.assign, …) {quantmod}
Load and Manage Data from Multiple Sources

```
## TimeSeriesAnalysis5_financial.R ##
library(quantmod)
#Microsoft Corporation (Nasdaq, USA)
Microsoft <- getSymbols("MSFT", src="yahoo", auto.assign=F,
                    from='2015-10-01', to='2017-09-30')
tail(Microsoft,2)
```

```
           MSFT.Open MSFT.High MSFT.Low MSFT.Close MSFT.Volume MSFT.Adjusted
2017-09-28     73.54     73.97    73.31      73.87    10883800      72.84149
2017-09-29     73.94     74.54    73.88      74.49    17079100      73.45286
```

```
#Facebook, Inc. (Nasdaq, USA)
Facebook <- getSymbols("FB", src="yahoo", auto.assign=F,
                    from='2015-10-01', to='2017-09-30')
tail(Facebook,2)
```

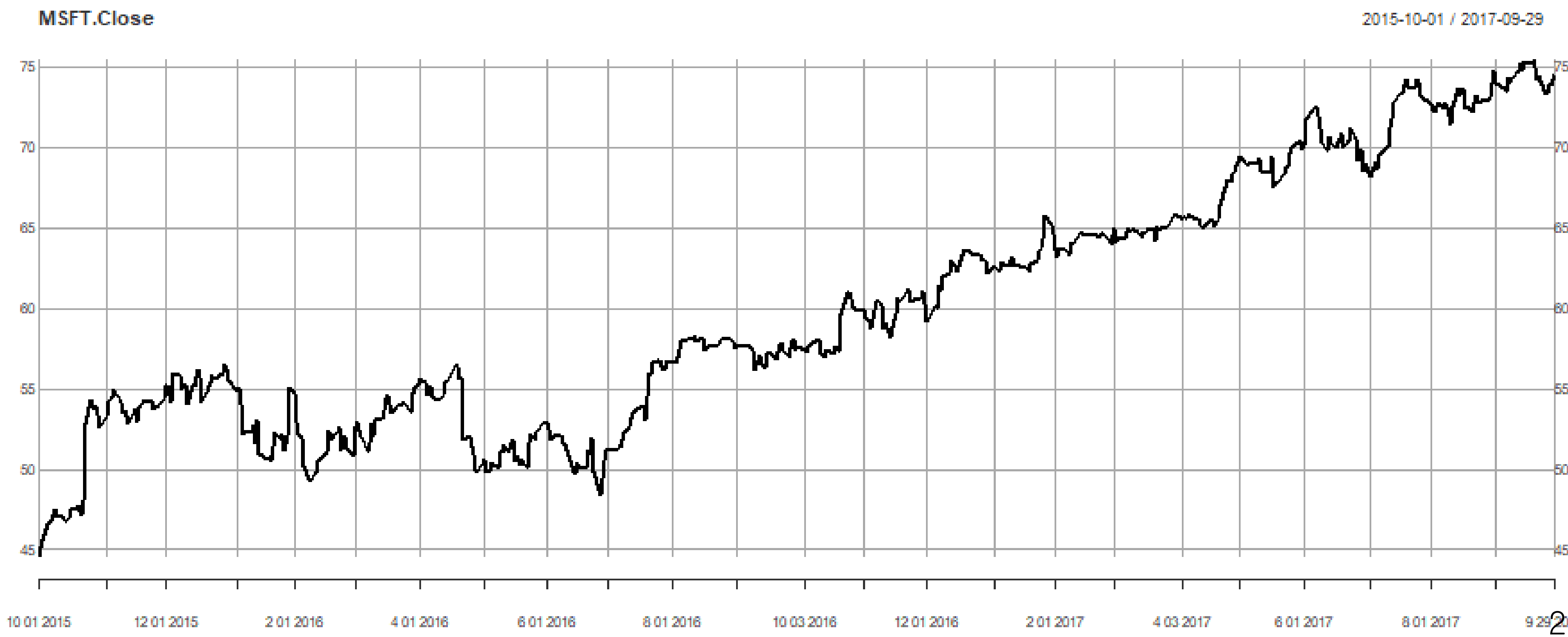
```
           FB.Open FB.High FB.Low FB.Close FB.Volume FB.Adjusted
2017-09-28  167.94  169.07 167.16   168.73  12178700      168.73
2017-09-29  168.83  171.66 168.81   170.87  15340400      170.87
```

```
> #SKhynix (KOSPI, Republic of Korea)
> SKhynix <- getSymbols("000660.KS", src="yahoo", auto.assign=F,
+                        from='2017-12-11', to='2018-12-10')
> class(SKhynix)
[1] "xts" "zoo"
> tail(SKhynix)
           000660.KS.Open 000660.KS.High 000660.KS.Low
2018-12-03          71000          71100         69800
2018-12-04          69900          70400         68400
2018-12-05          67200          68800         67200
2018-12-06          67800          68200         65700
2018-12-07          66700          67400         66600
2018-12-10          65900          65900         64700
           000660.KS.Close 000660.KS.Volume 000660.KS.Adjusted
2018-12-03           70500          2760290              70500
2018-12-04           69000          3222060              69000
2018-12-05           68200          3008950              68200
2018-12-06           66000          3902275              66000
2018-12-07           66800          2590733              66800
2018-12-10           65500          2206084              65500
```

(2) Time Series Plot

A time series plot is a graph that you can use to evaluate patterns and behavior in data over time.
A time series plot displays observations on the y-axis against equally spaced time intervals on the x-axis.

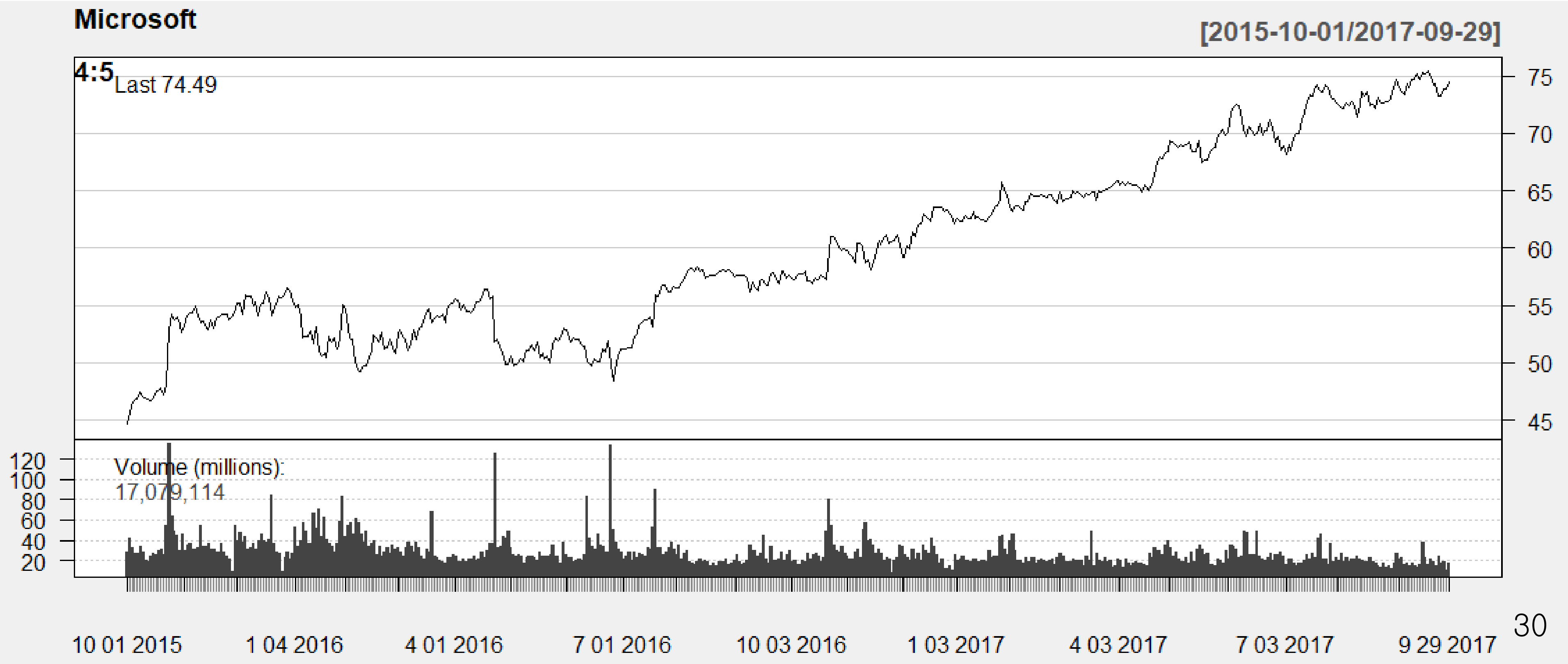
```
MSFT.Close = Microsoft[,4]
plot(MSFT.Close) #Close price
```



chartSeries(x, subset, theme, up.col, ...) {quantmod}

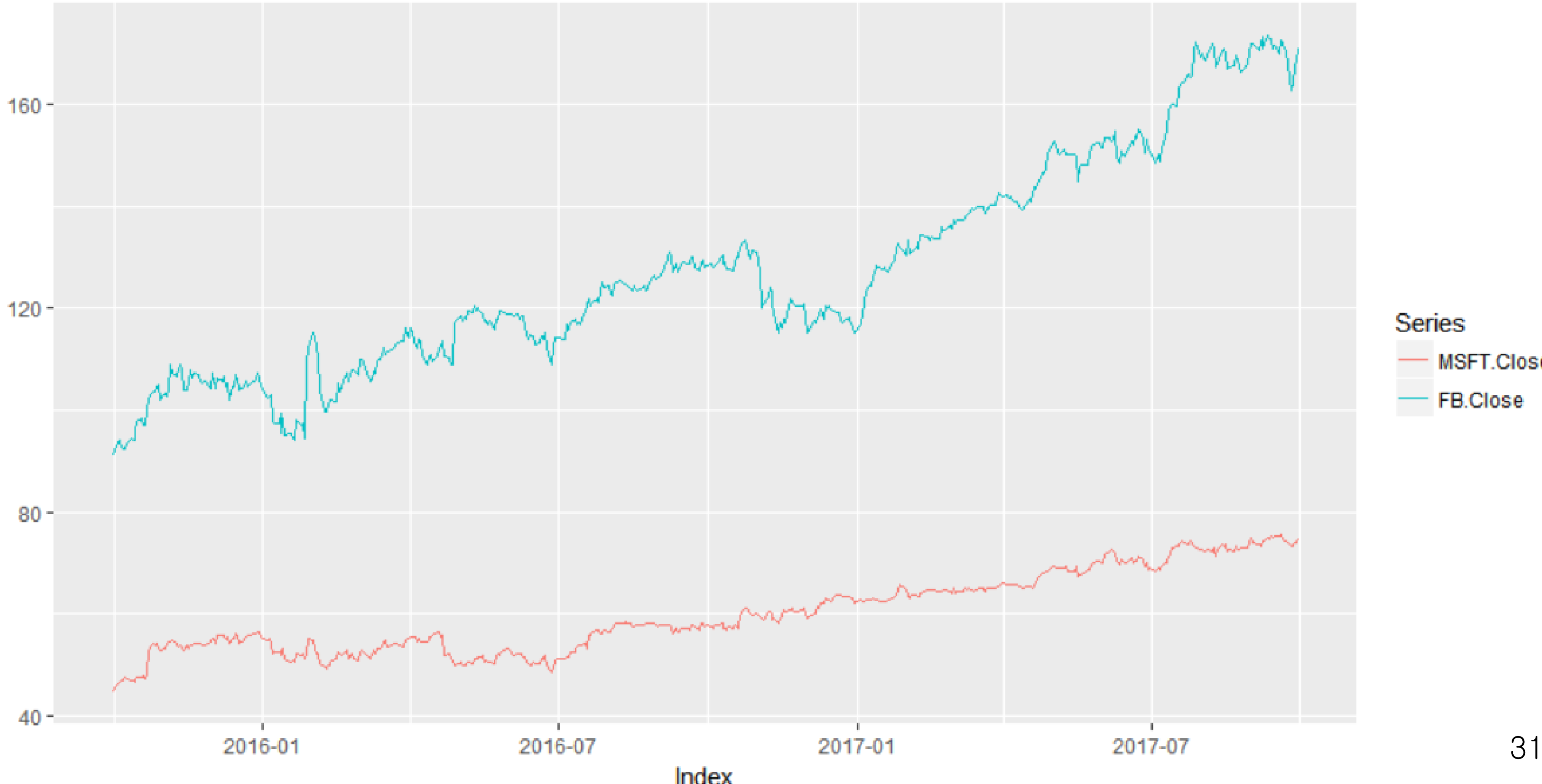
Charting tool to create standard financial charts given a time series like object. Possible chart styles include candles, matches, bars, and lines.

```
#4=close 5=volume
chartSeries(Microsoft[,4:5],subset="2015-10-01::2017-09-30",
            theme='white', up.col='black')
```



If you already have the download Microsoft and Facebook, then you can merge the resulting closing prices.

```
#Comparing two stock prices
z <- merge(Microsoft[,4], Facebook[,4])
library("ggplot2")
autoplot(z, facets = NULL)
```



(3) Forecasting Time Series

auto.arima(y, test, ic, ...) {forecast}
Fit best ARIMA model to univariate time series.

```
library(forecast)
fit = auto.arima(Microsoft[,4], test='adf', ic='bic')
plot(forecast(fit, h=48), main="Forecast by ARIMA of Microsoft")
```