9th Russian Summer School in Information Retrieval Big Data Analytics with R

Introduction to Time Series with R

A. Karakitsiou A. Migdalas

Industrial Logistics, ETS Institute
Luleå University of Technology
971 87 Luleå, Sweden
{athkar, athmig}@ltu.se

Time Series Analysis I

- A time series is a collection of observations of a variable obtained through repeated measurements over time.
- The pattern of the data is an important factor in understanding how the time series has behaved in the past.
- If such behavior can be expected to continue in the future, we can use it to guide us in selecting an appropriate forecasting method.
- Let us review some of the common types of data patterns that can be identified when examining a time series plot.

Time Series Analysis II

Trend A trend exists when there is a long-term increase or decrease in the data. It does not have to be linear. Sometimes we will refer to a trend "changing direction" when it might go from an increasing trend to a decreasing trend. Trend is usually the result of factors such as population increases or decreases, changing demographic characteristics of the population, technology, and/or consumer preferences.

Seasonal A seasonal pattern exists when a series is influenced by seasonal factors (e.g., the quarter of the year, the month, or day of the week). Seasonality is always of a fixed and known period.

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

Time Series Analysis III

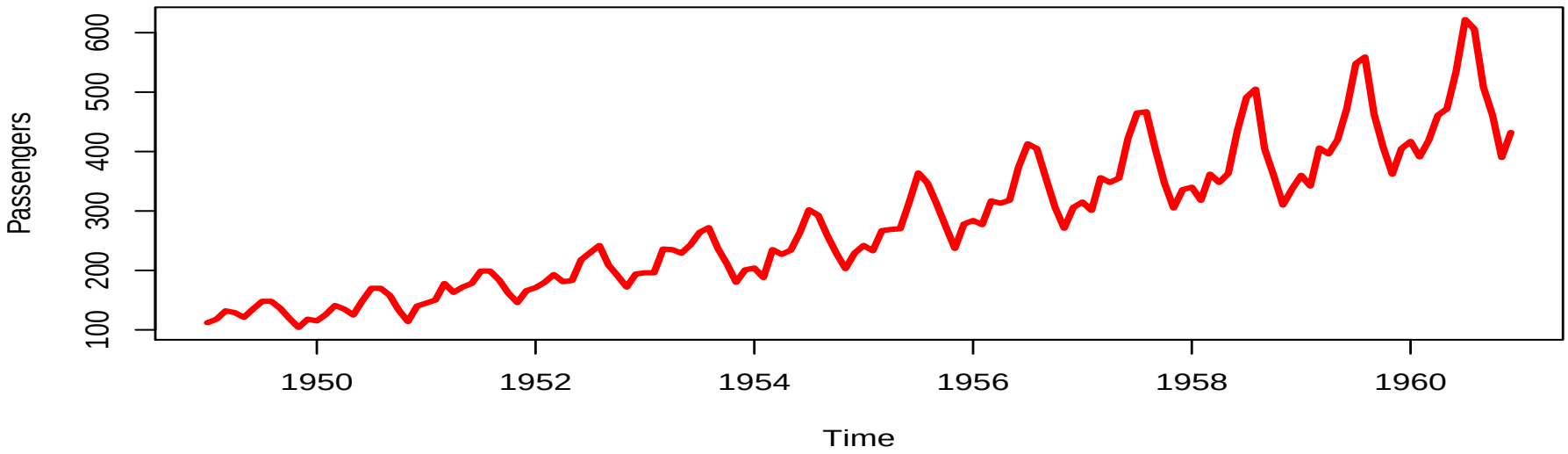
Irregular It corresponds to the high frequency fluctuations of the series. The irregular component results from short term fluctuations in a series which are not systematic and in some instances not predictable, e.g. uncharacteristic weather patterns.

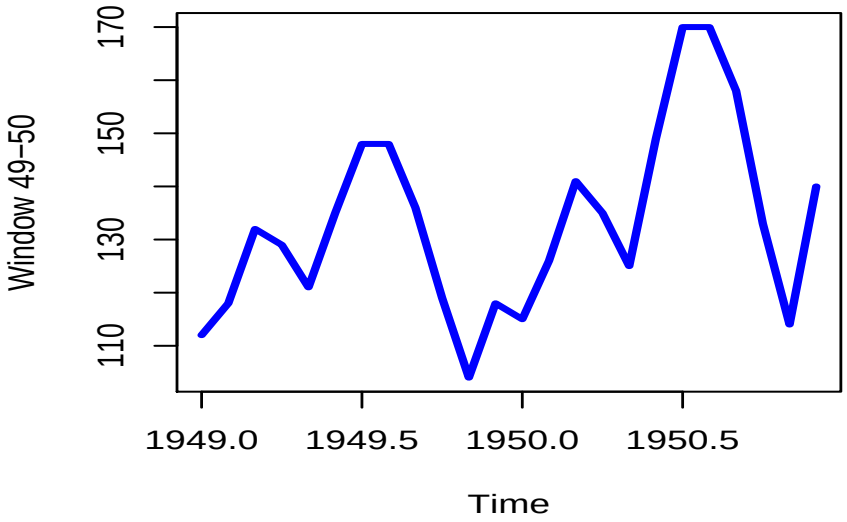
- Another important feature of most time series is that observations closer together in time tend to be correlated.
- Much of the methodology in a time series analysis is aimed at explaining this correlation and the main featured in the data using appropriate statistical models.
- Once a good model is found and fitted to data it can be used to forecast future values or generate simulation
- To identify the underlying pattern in the data, a useful first step is to construct a time series plot.

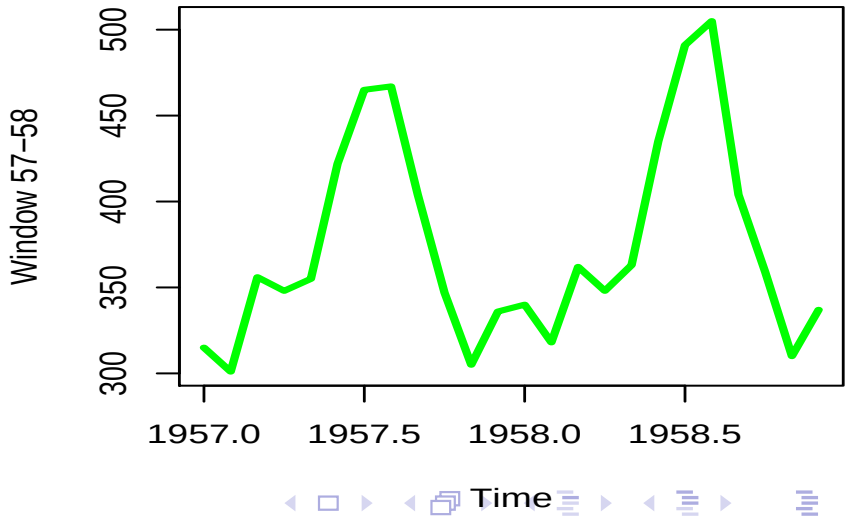
Plot Time Series

Example 1

```
> data(AirPassengers)
> Pass<-AirPassengers
> class(Pass)
[1] "ts"
> start(Pass)
[1] 1949    1
> end(Pass)
[1] 1960   12
> frequency(Pass)
[1] 12
#extract specific time window
> Pass.49.50<-window(Pass, start=c(1949,1), end=c(1950,12))
> Pass.57.58<-window(Pass, start=c(1957,1), end=c(1958,12))
#save plot in a pdf file
> pdf("PassPlots2.pdf")
#arrange the window to put plots
> layout(matrix(c(1,1,2,3),2,2, byrow=TRUE))
> plot(Pass, ylab="Passengers", xlab="Time", type="l", col="red", lwd=3)
> plot(Pass.49.50, ylab="Window 49-50", xlab="Time", type="l", col="blue", lwd=3)
> plot(Pass.57.58, ylab="Window 57-58", xlab="Time", type="l", col="green", lwd=3)
>layout(1)
>#return to the console
>dev.off()
```


Passengers
100
200
300
400
500
600
1950
1952
1954
1956
1958
1960
Time

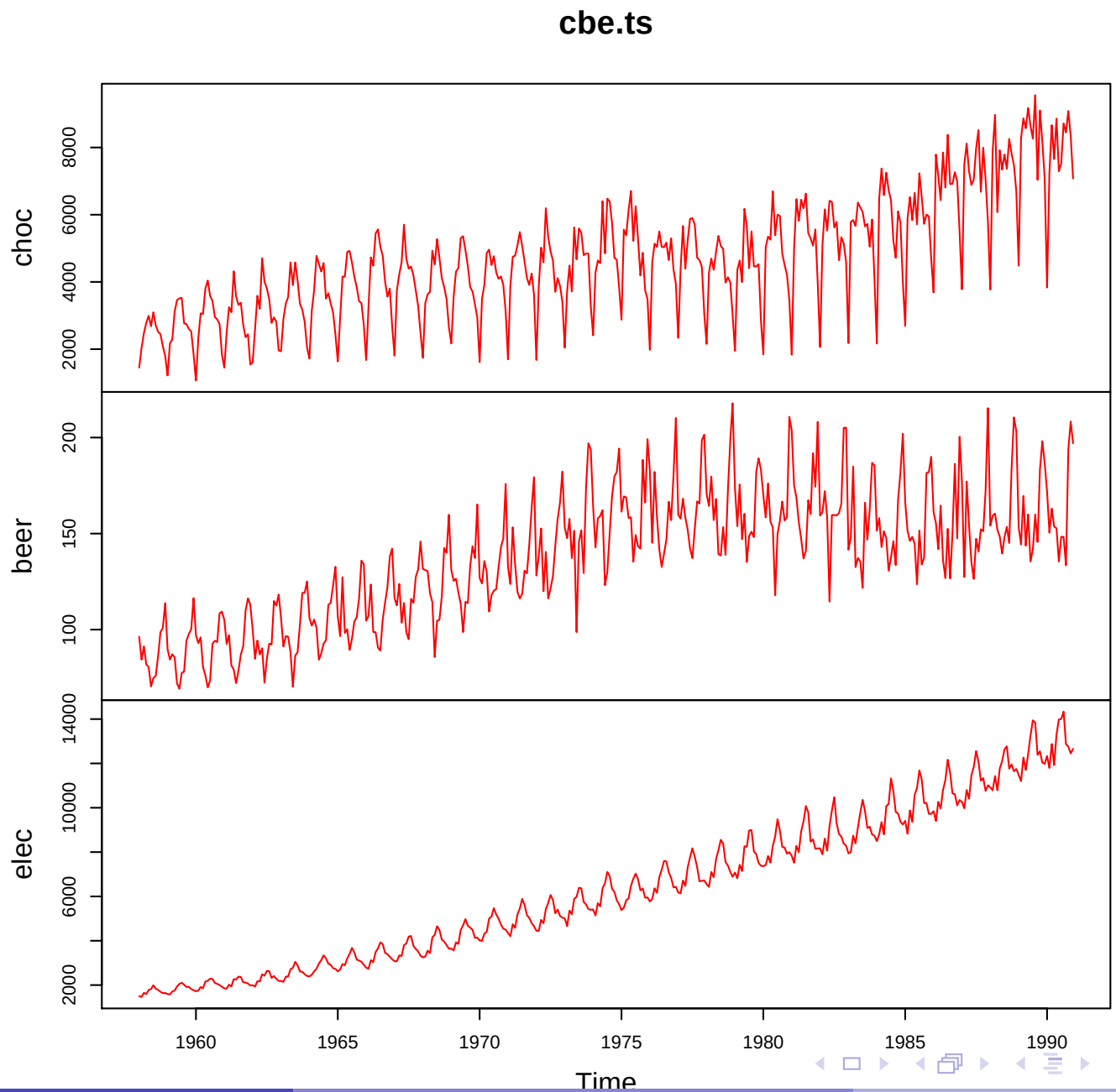

Window 49–50
110
130
150
170
1949.0
1949.5
1950.0
1950.5
Time


Window 57–58
300
350
400
450
500
1957.0
1957.5
1958.0
1958.5
Time

Working with Multiple Time Series

Example 2

```
#data obtained from http://elena.aut.ac.nz/~pcowpert/ts/
> cbe<-read.csv("cbe.csv", header=TRUE, sep=",")
> class(cbe)
[1] "data.frame"
#convert to time series
> cbe.ts<-ts(cbe, start=c(1958,1), freq=12)
> cbe.ts[1:4,]
     choc beer elec
[1,] 1451 96.3 1497
[2,] 2037 84.4 1463
[3,] 2477 91.2 1648
[4,] 2785 81.9 1595
> cbe.elec<-ts(cbe.ts[,3], start=c(1958,1), freq=12)
> cbe.beer<-ts(cbe.ts[,2], start=c(1958,1), freq=12)
> cbe.choc<-ts(cbe.ts[,1], start=c(1958,1), freq=12)
>plot.ts(cbe.ts, col="red", type="l", lwd=3)
```


cbe.ts
choc
beer
elec
2000
4000
6000
8000
100
150
200
2000
6000
10000
14000
1960
1965
1970
1975
1980
1985
1990
Time

Time series decomposition I

- We shall think of the time series Y_t as comprising three components: a seasonal component, a trend-cycle component (containing both trend and cycle), and a irregular component (containing anything else in the time series).
- For example, if we assume an additive model, then we can write

$$Y_t = S_t + T_t + E_t,$$

where y_t is the data at period t, S_t is the seasonal component at period t, T_t is the trend-cycle component at period t and E_t is the remainder (or irregular or error) component at period t.

- The additive model is most appropriate if the magnitude of the seasonal fluctuations or the variation around the trend-cycle does not vary with the level of the time series.

Time series decomposition II

- When the variation in the seasonal pattern, or the variation around the trend-cycle, appears to be proportional to the level of the time series, then a multiplicative model is more appropriate.

$$Y_t = S_t \times T_t + E_t.$$

- If the random variation is modeled by a multiplicative factor and the variable is positive, an additive decomposition model for $\log(Y_t)$ can be used:

$$\log(x_t) = S_t + T_t + E_t$$

Estimating trends and seasonal effects I

- There are various ways to estimate the trend $\hat{T}_t$. A relative simple procedure which is available in R is to calculate a *moving average* centered on Y_t.
- The idea behind the moving averages is that observations which are nearby in time are also likely to be close in value.
- The average of the points near an observation will provide a reasonable estimate of the trend-cycle at that observation.
- The average eliminate some of the randomness in the data, and leaves a smooth trend-cycle component.
- The term moving average is used because each average is computed by dropping the oldest observation and including the next observation.

Estimating trends and seasonal effects II

Simple moving average

- The simple moving average technique can take different forms.
- We begin by choosing an order for the moving average k where k is an odd integer.
- Then the $\mathrm{MA(k)}$ moving average is

$$MA_t(k) = \frac{1}{k} \sum_{j=-m}^{m} Y_{t-j}$$

where $m = \frac{k-1}{2}$

Estimating trends and seasonal effects III

Centered moving average

- While the definition of the $MA(k)$ works for odd values of k it is less satisfactory for even values of k.
- It is possible to apply a moving average to a moving average.
- One reason for doing this is to make an even-order moving average symmetric.
- Hence we define the centered $2MA_t(k)$ smoother to be

$$2MA_t(k) = \frac{MA_t(k^L) + MA_t(k^R)}{2}$$

Estimating trends and seasonal effects IV

Seasonal Effects

- If a additive model is assumed, an estimate of the seasonal effect at time t can be obtained by subtracting $\hat{T}_t$

$$\hat{S} = Y_t - \hat{T}_t$$

- By averaging these estimate of the monthly effect for each month we obtain a single estimate of the effect for each month.
- If a multiplicative model is considered then the seasonal effect estimates are given by

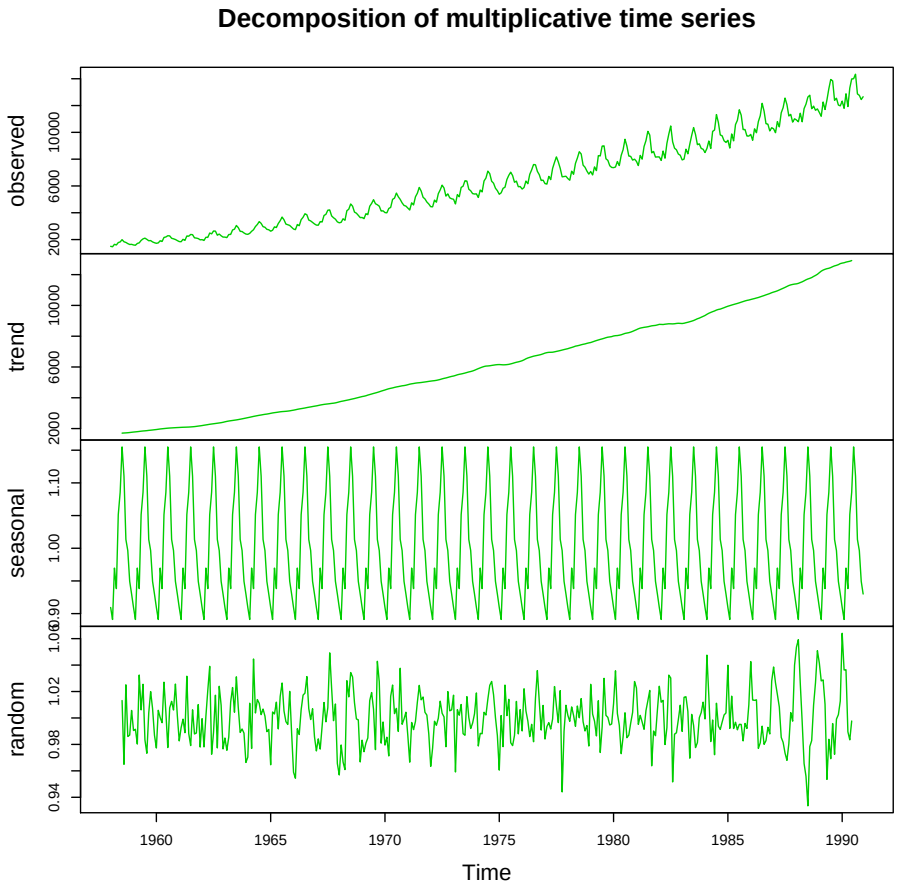
$$\hat{S}_t = \frac{Y_t}{\hat{T}_t}$$

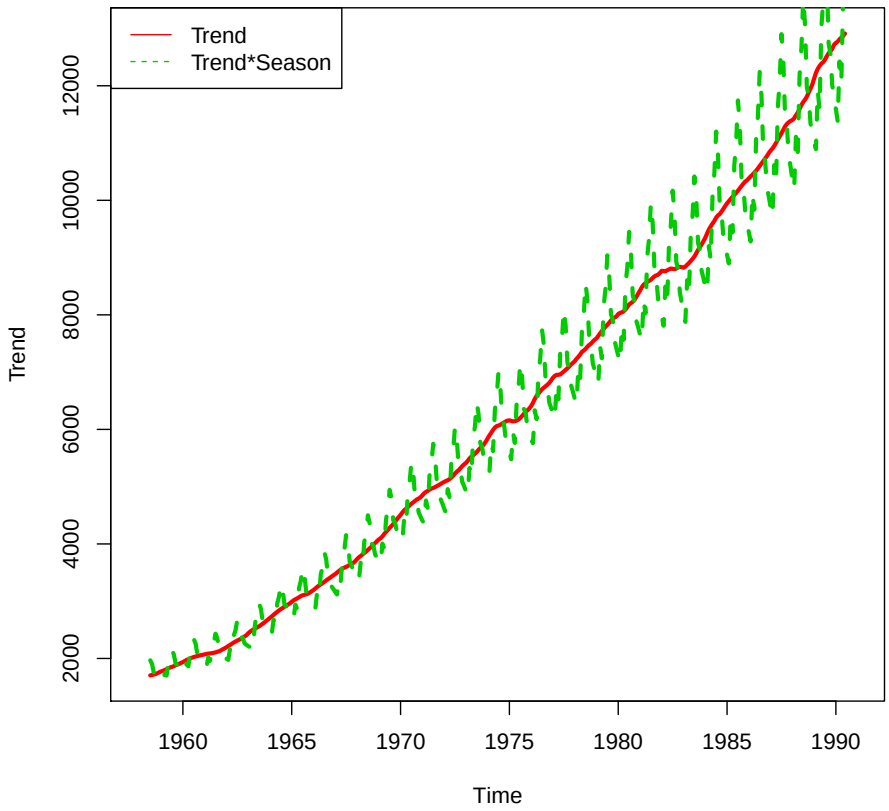
- A seasonally adjusted times series is obtained by
  1. $Y_t - \hat{S}_t$ if the seasonal effect is additive
  2. $\frac{Y_t}{\hat{S}_t}$ if the seasonal effect is multiplicative.

Decomposition in R

Example 3

```
#decompose()is used
#assume multiplicatice model sinces variance and trend
#increase over time
> cbe.elec.dec<-decompose(cbe.elec, type="mult")
#plot the components
> plot(cbe.elec.dec, col=5)
>Trend<-cbe.elec,dec$trend
>TrendxSeason<-cbe.elec.dec$season
> plot(Trend, type="l", lty=1, lwd=3, col=2)
> lines(TrendxSeason, type="l", lty=2, lwd=3, col=3)
> legend("topleft", c("Trend","Trend*Season"), lty=1:2, col=2
```


Decomposition of multiplicative time series
observed
trend
seasonal
random
Time
1960
1965
1970
1975
1980
1985
1990


Trend
Trend*Season
Trend
Time
2000
4000
6000
8000
10000
12000
1960
1965
1970
1975
1980
1985
1990

Forecasting with decomposition I

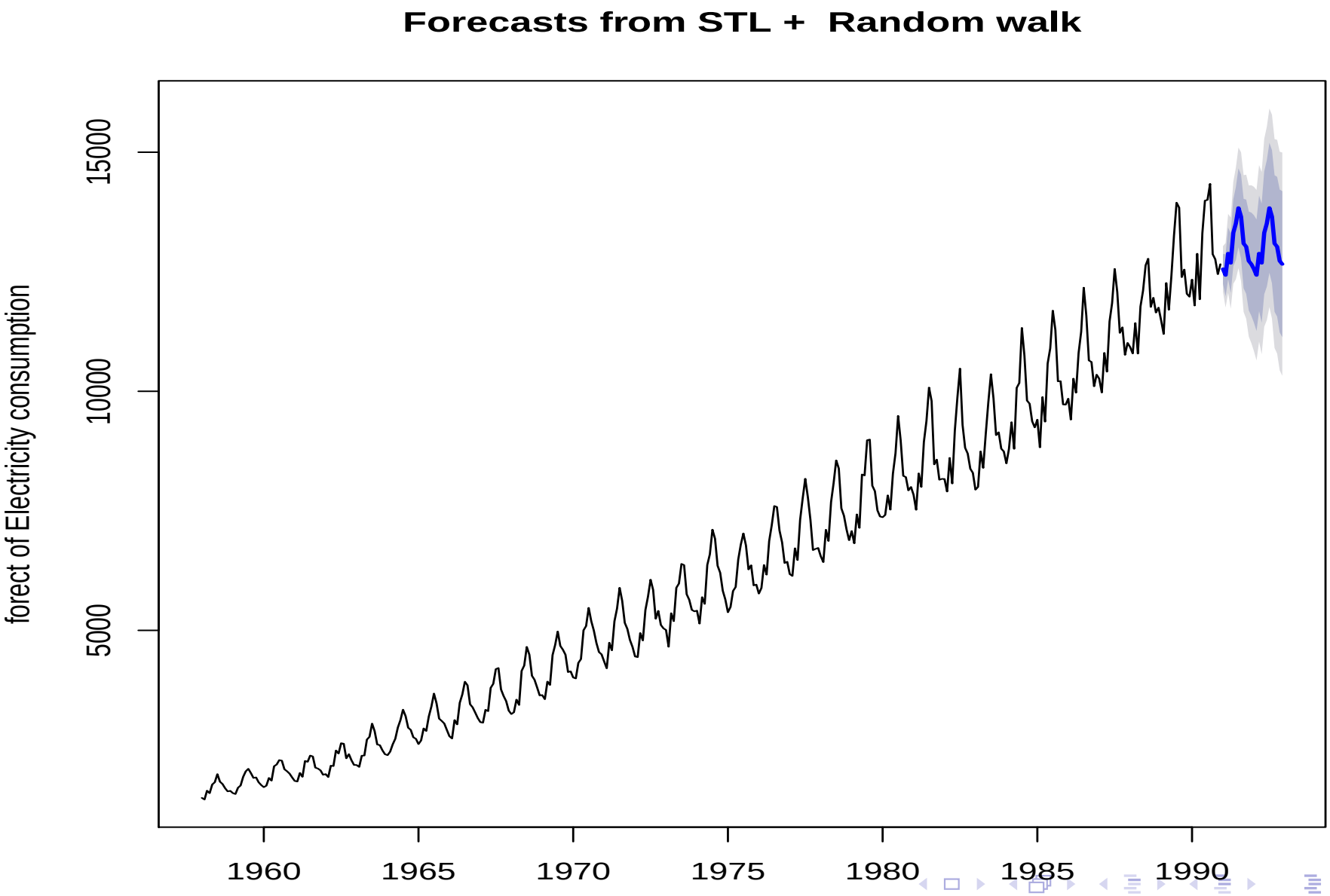
- To forecast the seasonally adjusted component, any non-seasonal forecasting method may be used.
- For example, a random walk with drift model, or Holt's method r), or a non-seasonal ARIMA model (discussed later), may be used.

Forecasting with decomposition II

Example 4

```
#decompose using stl() function
#"Seasonal and Trend decomposition using Loess"
#The two main parameters to be chosen when using STL are
#the trend window (t.window) and
#seasonal window (s.window).
#These control how rapidly the trend and seasonal components
#can change. Small values allow more rapid change.
#Setting the seasonal window to be infinite is
#equivalent to forcing the seasonal component to be
#periodic (i.e., identical across years).
>install.packages("forecast", dependencies = TRUE)
>library("forecast")
>cbe.elec.fit<- stl(cbe.elec, t.window=15,
                   s.window="periodic", robust=TRUE)
#adjust seasonality
>cbe.elec.adj <- seasadj(cbe.elec.fit)
>cbe.elec.fcast <- forecast(cbe.elec. fit, method="naive")
>plot(cbe.ele.fcast, ylab="forect of Electricity consumption")
#naive method =All forecasts are simply set to be the
#value of the last observation
```

Forecasting with decomposition III



Stationality I

- A stationary time series is one whose properties do not depend on the time at which the series is observed
- So time series with trends, or with seasonality, are not stationary-the trend and seasonality will affect the value of the time series at different times.
- The mean function of a time series model is

$$\mu(t) = E(Y_t)$$

which is, in general, a function of time t.

- If the mean function is constant, i.e.

$$\mu(t) = \mu$$

, we say that the time series model is stationary in the mean.

Stationality II

- If we assume that a sufficiently long time series characterizes the hypothetical model, then the sample estimate of the population mean μ is the sample mean $\bar{Y}$

$$\bar{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t$$

and such models are known as *ergodic*.

- The variance function of a time series model that is stationary in the mean is

$$\sigma_t^2 = E[(Y_t - \mu)^2]$$

- In principal, σ_t can take a different value at every time t. In other words, being stationary in mean does not necessarily mean being stationary in variance.

Stationality III

- Unfortunately, we cannot estimate a different variance at each time point from a single time series. To progress, we must make some simplifying assumption.
- If we assume that the time series model is stationary in variance, then this constant population variance σ^2 can be estimated from the sample variance

$$\text{Var}(Y) = \frac{1}{n-1}\sum_{t=1}^{n}(Y_t - \bar{Y})^2$$

- In a time series analysis, sequential observations may be correlated. If the correlation is positive, Var(Y) will tend to underestimate the population variance in a short series because successive observations tend to be relatively similar. In most cases, this does not present a problem since the bias decreases rapidly as the length n of the series increases.

Autocorrelation I

- Consider a time series that is stationary in both the mean and the variance.
- The variables may be serially correlated, and the model is *second-order stationary* if the serial correlation between variables depends only on the number of the steps between them.
- The number of the steps between them is know as *lag*.
- For a second-order stationary time series, we can define the autocovariance and autocorrelation functions of lag k as

$$\text{acvf:} \quad \gamma_k = E[(Y_t - \mu)(Y_{t+k} - \mu) \tag{1}$$

$$\text{acf:} \quad \rho_k = \frac{\gamma_k}{\sigma^2} \tag{2}$$

Autocorrelation II

- The autocovariance and autocorrelation functions can be estimated from a time series by their sample equivalents

$$\text{acvf:} \quad c_k = \frac{1}{n}\sum_{t=1}^{n-k}(Y_t - \bar{Y})(Y_{t+k} - \bar{Y}) \tag{3}$$

$$\text{acf:} \quad r_k = \frac{c_k}{c_0} \tag{4}$$

- $-1 \leq r_k < 1$

Tests for autocolleration I

ACF() function in R

- The function **acf()** computes (and by default plots) estimates of the autocovariance or autocorrelation function.

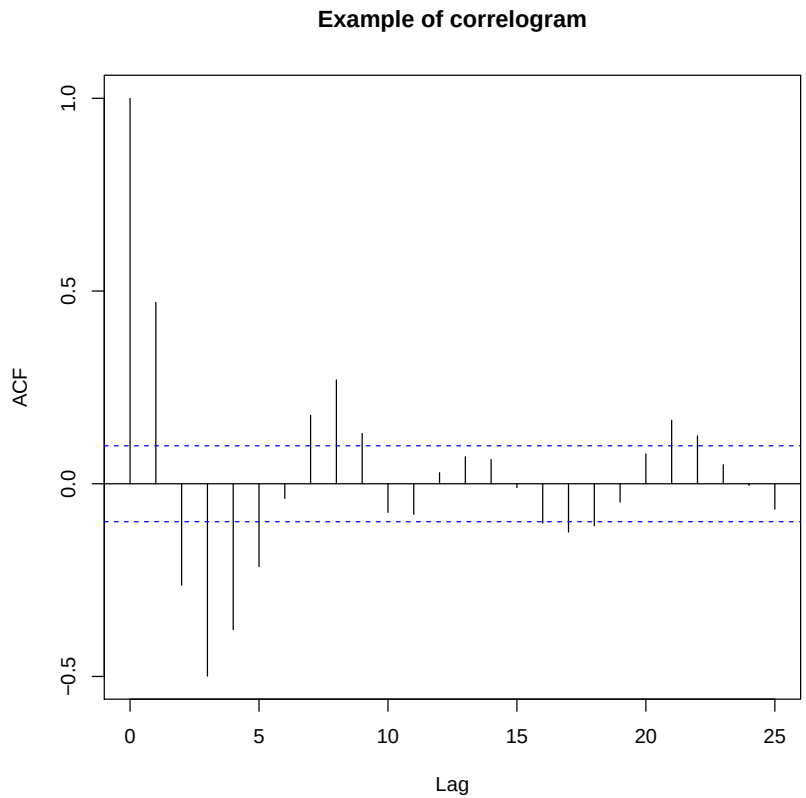
acf()

```
acf(x, type = c("correlation", "covariance", "partial"), plot = TRUE,.)
# x time series under consideration
#type of acf to be computed. Allowed values are correlation (the default),
#covariance or partial.
#plot. If TRUE (the default) the collegramm is plotted.
```

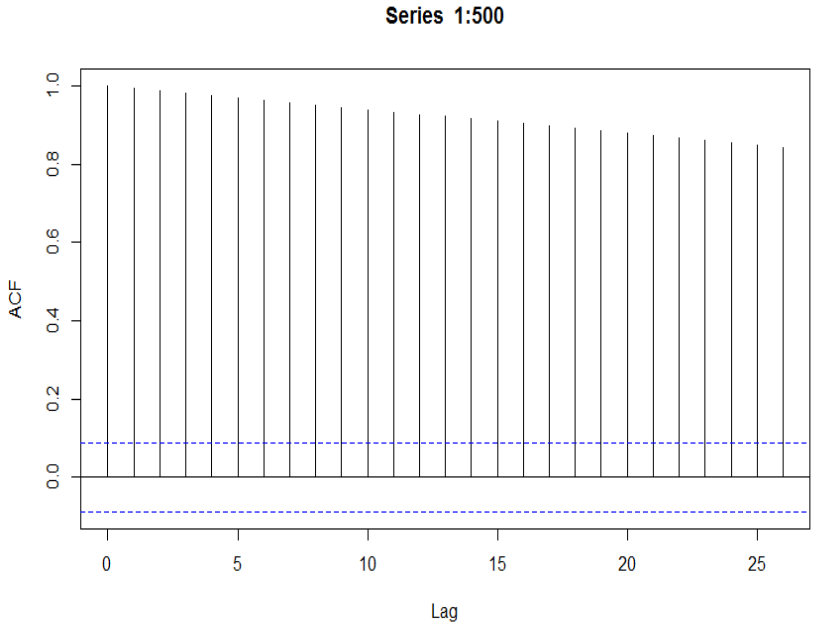
- The lag 0 autocorrelation is fixed at 1 by convention.)
- The autocorrelations of x are stored in the vector **acf(x)$acf**, with the lag k autocorrelation located in **acf(x)$acf[k+1]**.

Example 5

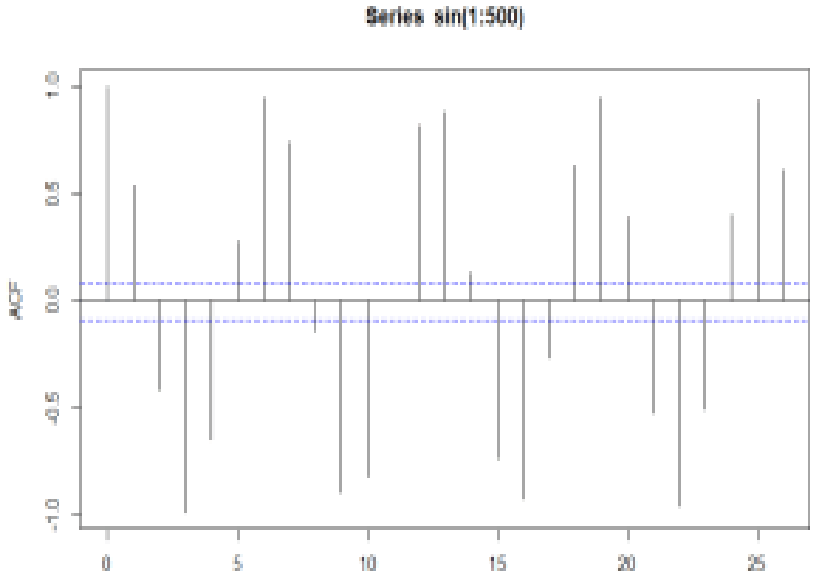
```
#initial data obtained from
#http://elena.aut.ac.nz/~pcowpert/ts/wave.dat
>cgramm.exam<-read.csv("autoc_data.csv", header=TRUE,
                       sep=";")
> attach(cgramm.exam)
#plot correlogramm
>acf(ts(autoc), main="Example of correlogram")
#autocorrelation at lag 0
> acf(autoc)$acf[1]
[1] 1
##autocorrelation at lag 1
> acf(autoc)$acf[2]
[1] 0.4702564
#autocorrelation at lag 9
> acf(autoc)$acf[10]
[1] 0.1303853
```



- The unit of the x-axis, the lags, is the sampling interval. The y-axix is dimensionless.
- If ρ_k=0, the sampling distribution of ρ_ks estimate, r_k, is approximately $\sim N(-\frac{1}{n}, \frac{1}{n})$. Therefore, on the correlogram, the dotted lines are drawn at $\frac{1}{n} \pm \frac{2}{\sqrt{n}}$ such that if r_k falls outside these lines, we have evidence at the 5% level to reject the null hypothesis that $\rho_k = 0$.
- The r_k are correlated, so if one falls outside the lines, the neighboring ones are more likely to be statistically significant.
- If ρ_k does equal 0 at all lags k, we expect 5% of the estimates, r_k, to fall outside the $-\frac{1}{n} \pm \frac{2}{\sqrt{n}}$lines.



•Usually a trend in the data will show in the correlogram as a slow decay in the autocorrelations, which are large and positive due to similar values in the series occurring close together in time.



•If there are seasonal variation, season spikes will be superimposed on the pattern

Simple Exponential Smoothing I

- Given a past history $\{Y_1, Y_2, \cdots, Y_n\}$, we want to predict some future value Y_{n+k}.
- We assume that:
  1. We assume there is no systematic trend or seasonal effects in the process, or that these have been identified and removed.
  2. The mean of the process can change from one time step to the next, but we have no information about the likely direction of these changes.
- The exponential smoothing model follows:

$$F_t = \alpha Y_t + (1 - \alpha) F_{t-1}$$

  α is the smoothing constant $0 \leq \alpha \leq 1$.
- Equation shows that the forecast for period F_t is a weighted average of the actual value in period t and the forecast for period $t - 1$.

Simple Exponential Smoothing II

- The weight given to the actual value in period t is the smoothing constant α and the weight given to the forecast in period t is $1 - \alpha$
- It turns out that the exponential smoothing forecast for any period is actually a weighted average of all the previous actual values of the time series.

Holt's linear trend method

- Holt (1957) extended simple exponential smoothing to allow forecasting of data with a trend. This method involves a forecast equation and two smoothing equations (one for the level and one for the trend):

$$\begin{aligned} \text{Forecast equation} \quad & y_{t+h} = \ell_t + hb_t \\ \text{Level equation} \quad & \ell_t = \alpha y_t + (1-\alpha)(\ell_{t-1} + b_{t-1}) \\ \text{Trend equation} \quad & b_t = \beta^*(\ell_t - \ell_{t-1}) + (1-\beta^*)b_{t-1} \end{aligned}$$

where ℓ_t denotes an estimate of the level of the series at time t, b_t denotes an estimate of the trend (slope) of the series at time t, α is the smoothing parameter for the level, $0 \leq \alpha \leq 1$ and β^* is the smoothing parameter for the trend, $0 \leq \beta^* \leq 1$.

Holt-Winters seasonal method

- Holt (1957) and Winters (1960) extended Holt's method to capture seasonality. The Holt-Winters seasonal method comprises the forecast equation and three smoothing equations
  1. one for the level ℓ_t,
  2. one for trend b_t, and
  3. one for the seasonal component denoted by s_t,
- with smoothing parameters α, β, γ.
- We use m to denote the period of the seasonality, i.e., the number of seasons in a year.

The HoltsWinters() Function I

- To make forecasts using simple exponential smoothing in R, we can fit a simple exponential smoothing predictive model using the **HoltWinters()** function in R.

test

```
HoltWinters(x, alpha = NULL, beta = NULL, gamma = NULL,
            seasonal = c("additive", "multiplicative"),
            start.periods = 2, l.start = NULL, b.start = NULL,
            s.start = NULL,
            optim.start = c(alpha = 0.3, beta = 0.1, gamma = 0.1),
            optim.control = list())
```

- The HoltWinters() function returns a list variable, that contains several named elements.
- By default **HoltWinters()** just makes forecasts for the time period covered by the original data

The HoltsWinters() Function II

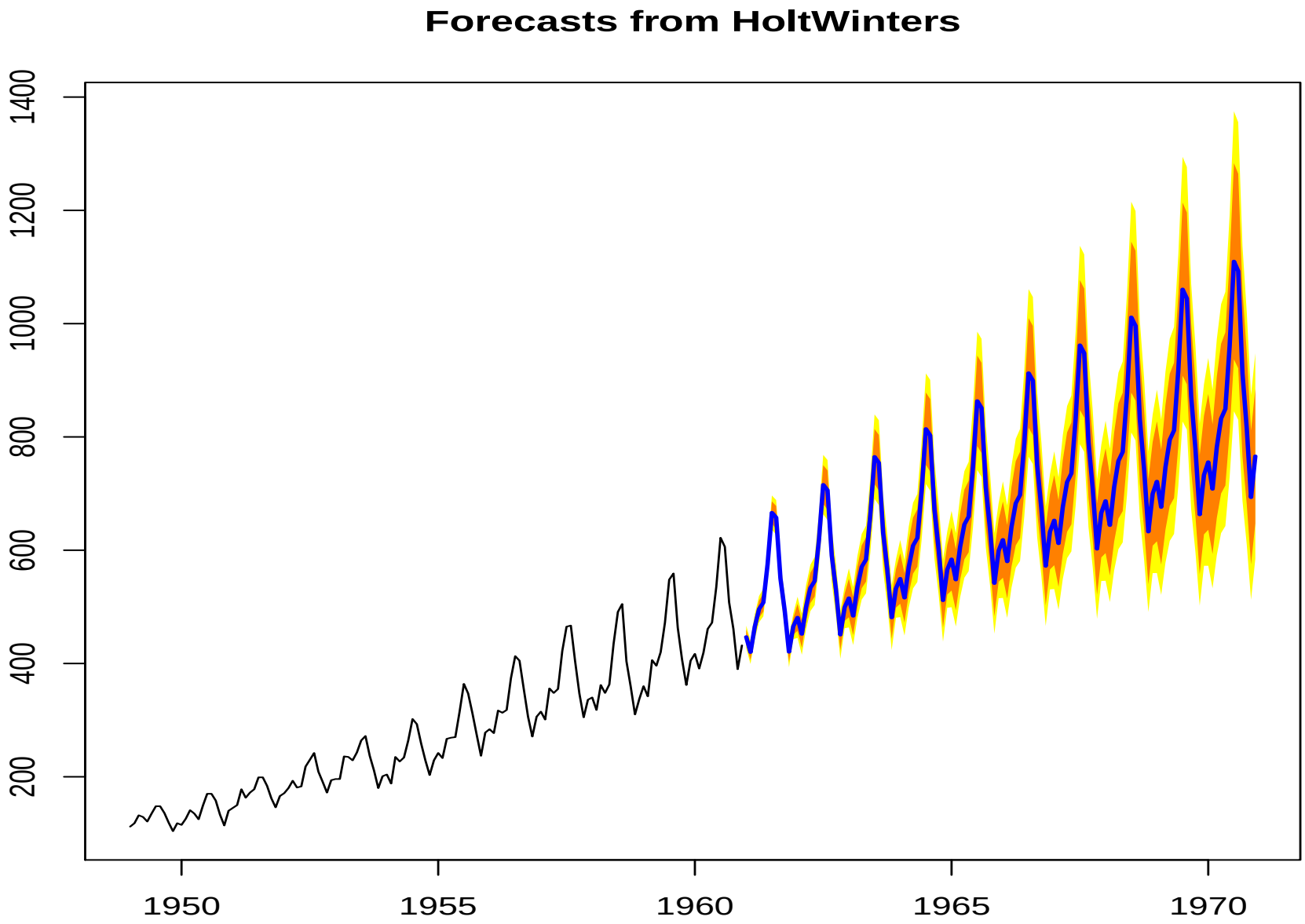
- We can make forecasts for further time points by using the **forecast.HoltWinters()** function in the R "forecast" package.
- When using the **forecast.HoltWinters()** function, as its first argument (input), you pass it the predictive model that you have already fitted using the HoltWinters() function
- You specify how many further time points you want to make forecasts for by using the **h** parameter.
- The forecast.HoltWinters() function gives you the forecast for time period, a 80% prediction interval for the forecast, and a 95% prediction interval for the forecast

Example 6

```
>library("forecast")
>Pass.fit<-HoltWinters(Pass.ts, seasonal="multi")
> Pass.fit
Holt-Winters exponential smoothing with trend and multiplicative seasonal component

Call:
  HoltWinters(x = Pass.ts, seasonal = "multi")

Smoothing parameters:
  alpha: 0.2755925
beta : 0.03269295
gamma: 0.8707292

Coefficients:
  [,1]
a   469.3232206
b     3.0215391
s1    0.9464611
s2    0.8829239
s3    0.9717369
s4    1.0304825
s5    1.0476884
s6    1.1805272
s7    1.3590778
```

Example 7

```
#Holts-winter forcast
Pass.for<-forecast.HoltWinters(Pass.fit, h=120)

         Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
Jan 1961       447.0559 434.1422  459.9696 427.3061  466.8057
Feb 1961       419.7123 406.2559  433.1686 399.1326  440.2920
Mar 1961       464.8671 450.5448  479.1895 442.9630  486.7712
Apr 1961       496.0839 480.8809  511.2870 472.8329  519.3350
   . . . .
   . . . .
Sep 1970       911.9885 764.9479 1059.0292 687.1093 1136.8678
Oct 1970       815.0306 680.7768  949.2843 609.7072 1020.3540
Nov 1970       693.0595 574.9614  811.1576 512.4440  873.6750
Dec 1970       766.1769 647.4488  884.9051 584.5979  947.7559
>plot.forecast(Pass.for)
```


Forecasts from HoltWinters
1400
1200
1000
800
600
400
200
1950
1955
1960
1965
1970

Box-Jenkins Methodology I

- The Box-Jenkins methodology refers to a set of procedures for identifying and estimating time series models within the class of **autoregressive integrated moving average (ARIMA)** models.
- ARIMA models are regression models that use lagged values of the dependent variable and/or random disturbance term as explanatory variables.
- ARIMA models rely heavily on the autocorrelation pattern in the data
- Three basic ARIMA models for a stationary time series Y_t :

Box-Jenkins Methodology II

[1] Autoregressive model of order p (AR(p))

- In an autoregression model, we forecast the variable of interest using a linear combination of past values of the variable.
- The term autoregression indicates that it is a regression of the variable against itself.
- Thus an autoregressive model of order p can be written as

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \varepsilon_t,$$

  where c is a constant and ε_t is white noise
- Y_t depends on its p previous values

Box-Jenkins Methodology III

[2] Moving Average model of order q (MA(q))

- Rather than use past values of the forecast variable in a regression, a moving average model uses past forecast errors in a regression-like model.

$$y_t = c + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q},$$

  ε_t is white noise

- Y_t depends on q previous random error terms

Box-Jenkins Methodology IV

[3] Autoregressive-moving average model of order p and q (ARMA(p,q))

$$\begin{aligned} Y_t &= c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + \cdots + \phi_p Y_{t-p} + \\ &+ \varepsilon_t + \theta_1 \varepsilon_{t-1} + \theta_2 \varepsilon_{t-2} + \cdots + \theta_q \varepsilon_{t-q} \end{aligned}$$

Y_t depends on its p previous values and q previous random error terms

Autoregressive Integrated Moving Average ARIMA(p,d,q)

- ARMA models are defined for stationary time series.
- If you start off with a non-stationary time series, you will first need to "difference" the time series until you obtain a stationary time series.
- Because the need to make a time series stationary is common, the differentiating can be intergraded into ARMA model by defining the ARIMA(p,d,q) model. d denotes the differencing time

Box-Jenkins Methodology V

- There are five stages in building a Box Jenkins time series model:
  1. Stationarity Checking
  2. Model Identification
  3. Parameter Estimation
  4. Diagnostic Checking
  5. Forecasting

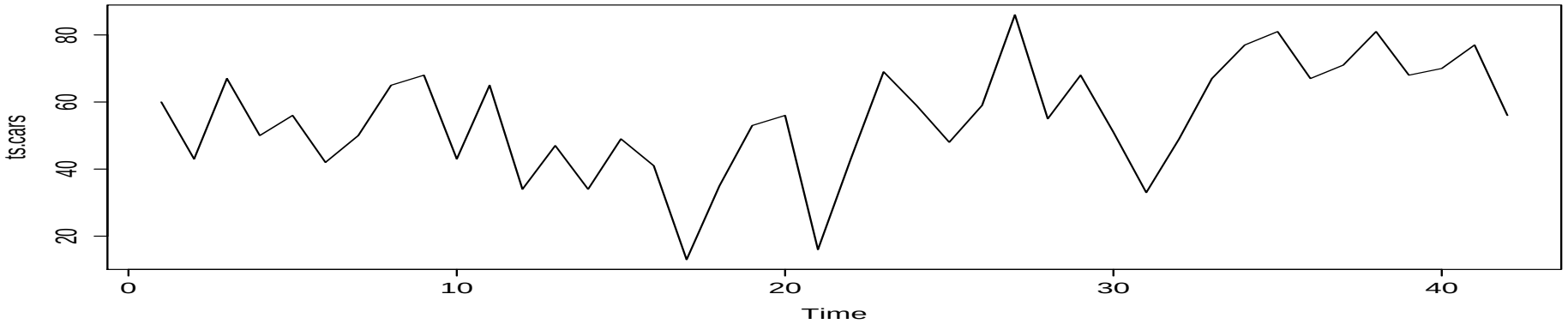
Stationary Checking

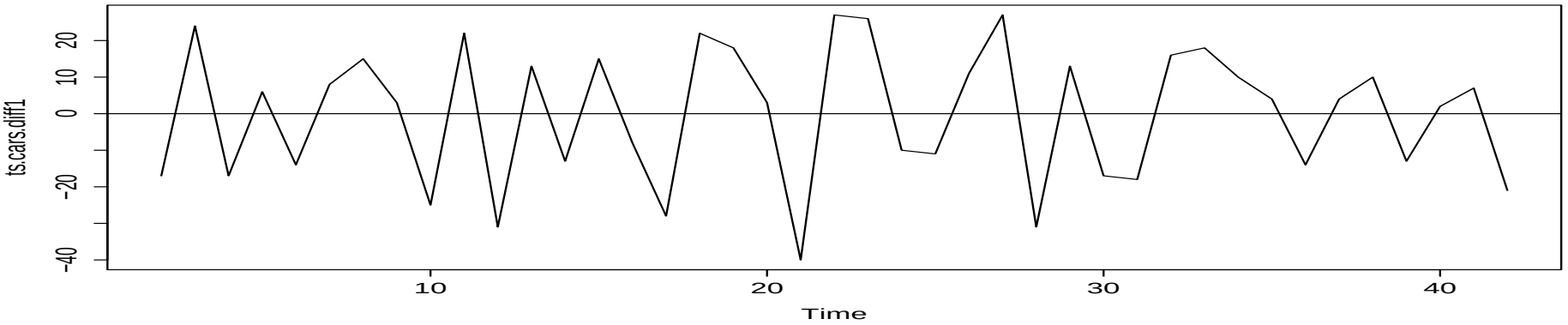
Differencing a Time Series

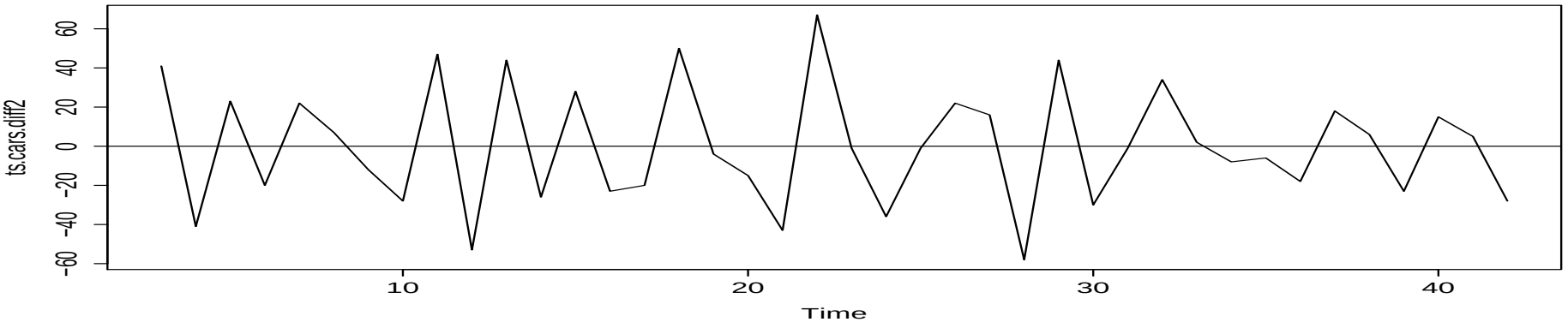
- To apply ARMA model the dataset needs to be a stationary time series.
- As a first step plot your time series
- If there is an obvious upward or downward linear trend, differencing difference may be needed.
- You can difference a time series using the **diff()** function in R.

Example 8

```
#data set carsdemand, demand for cars in a region
>cars<-read.csv("carsdemand.csv", header=TRUE)
>ts.cars<-ts(cars, freq=1)
>plot(ts.cars, ylab="ts.cars"
>ts.cars.diff1<-diff(ts.cars, differences = 1)
>plot(ts.cars.diff1, ylab="ts.cars.diff1")
>abline(0,0)
>ts.cars.diff2<-diff(ts.cars, differences = 2)
>plot(ts.cars.diff1, ylab="ts.cars.diff1")
>abline(0,0)
```

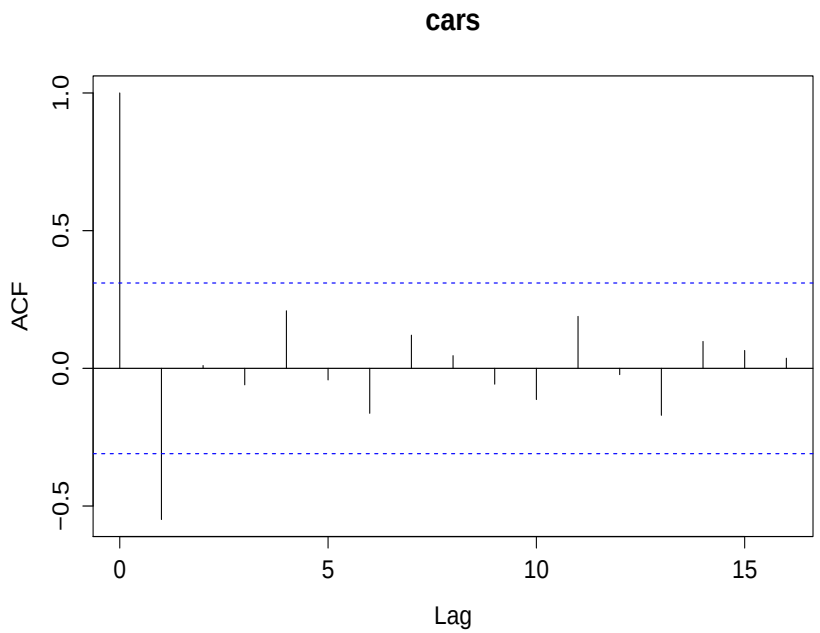

ts.cars
20
40
60
80
0
10
20
30
40
Time

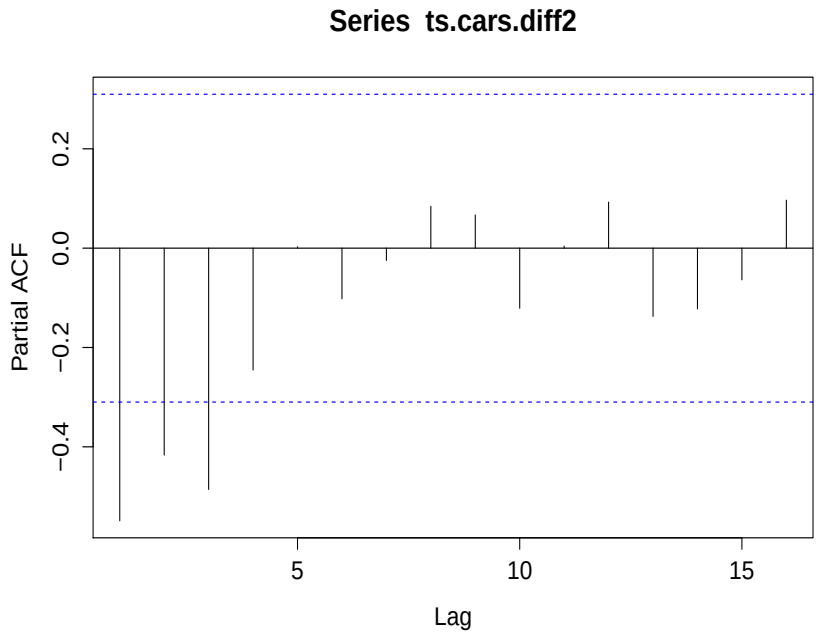

ts.cars.diff1
-40
-20
0
10
20
10
20
30
40
Time


ts.cars.diff2
-60
-40
-20
0
20
40
60
10
20
30
40
Time

Model Identification

- After a time series has been stationarized by differencing, the next step in fitting an ARIMA model is to determine whether AR or MA terms are needed to correct any autocorrelation that remains in the differenced series
- By looking at the autocorrelation function (ACF) and partial autocorrelation (PACF) plots of the differenced series, you can tentatively identify the numbers of AR and/or MA terms that are needed.
- A pure AR(p) will have a cut off at lag p in the PACF
- A pure MA(q) will have a cut off at lag q in the ACF.
- AR((I)MA(p,q) will (eventually) have a decay in both





- the following ARIMA (autoregressive moving average) models are possible for the time series of first differences:
- an ARIMA(3,2,0) model, since the partial autocorrelogram is zero after lag 3, and the autocorrelogram tails off to zero
- An ARIMA(0,2,1) model, since the autocorrelogram is zero after lag 1 and the partial autocorrelogram tails off to zero
- an ARMA(p,2,q)

Parameter Estimation

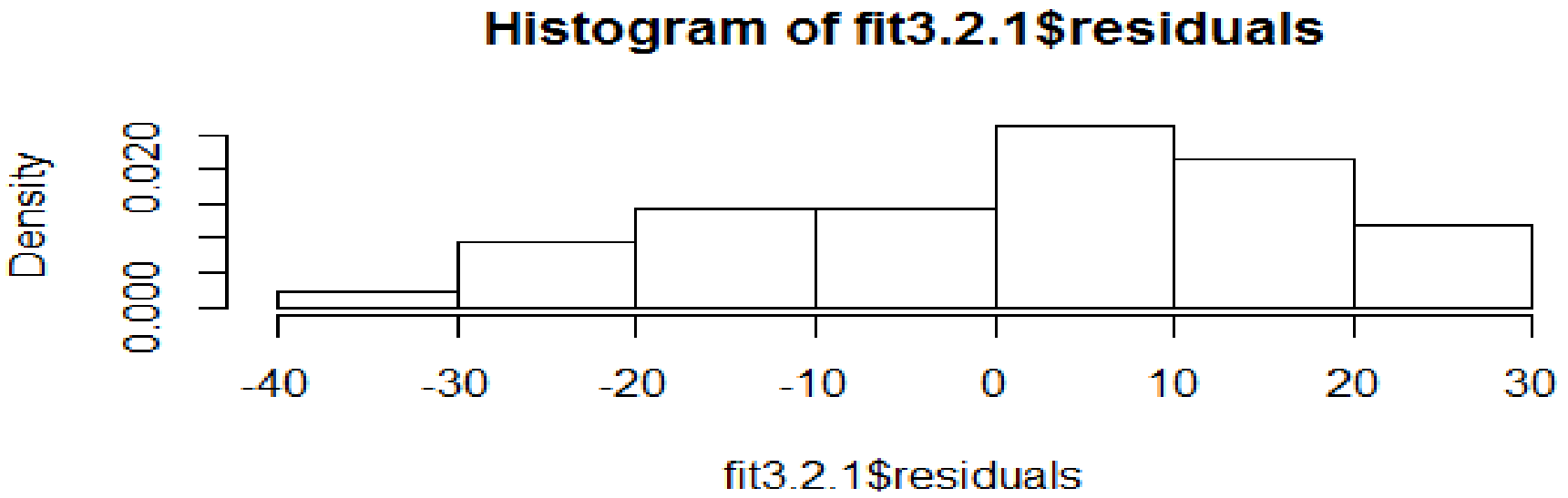
- Once the model order has been identified (i.e., the values of p, d and q), we need to estimate the parameters $c, \phi_1, \ldots, \phi_p, \theta_1, \ldots, \theta_q$
- When R estimates the ARIMA model, it uses maximum likelihood estimation (MLE). This technique finds the values of the parameters which maximize the probability of obtaining the data that we have observed
- Akaike's Information Criterion is a measure of the relative quality of statistical models
- To apply AIC in practice, we start with a set of candidate models, and then find the models' corresponding AIC values.
- Good models are obtained by minimizing AIC.

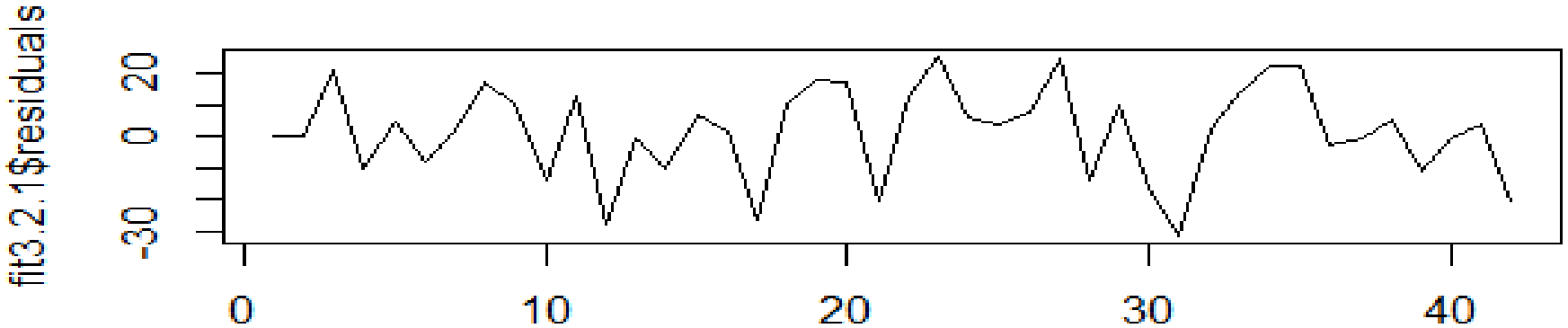
Estimation with R

- R provides the **arima()** and **Arima()** which differ in the way they handle the constant terms
- There exists also the **auto.arima()** function which can be used to find the appropriate ARIMA model

```
#library(forecast)
> fit3.2.1<-arima(ts.cars, order=c(3,2,1))
> fit3.2.1

Call:
  arima(x = ts.cars, order = c(3, 2, 1))

Coefficients:
         ar1      ar2      ar3      ma1
      -0.5912  -0.4729  -0.3133  -1.0000
s.e.   0.1522   0.1585   0.1509   0.0965

sigma^2 estimated as 227.6:  log likelihood = -168.34,  aic = 346.69


> fit0.2.1<-arima(ts.cars, order=c(0,2,1))
> fit0.2.1

Call:
  arima(x = ts.cars, order = c(0, 2, 1))

Coefficients:
         ma1
      -1.0000
s.e.   0.0641

sigma^2 estimated as 336.8:  log likelihood = -175,  aic = 354.01

#the arima(3 2 1) has the minimum aic
```

Diagnostics Checking



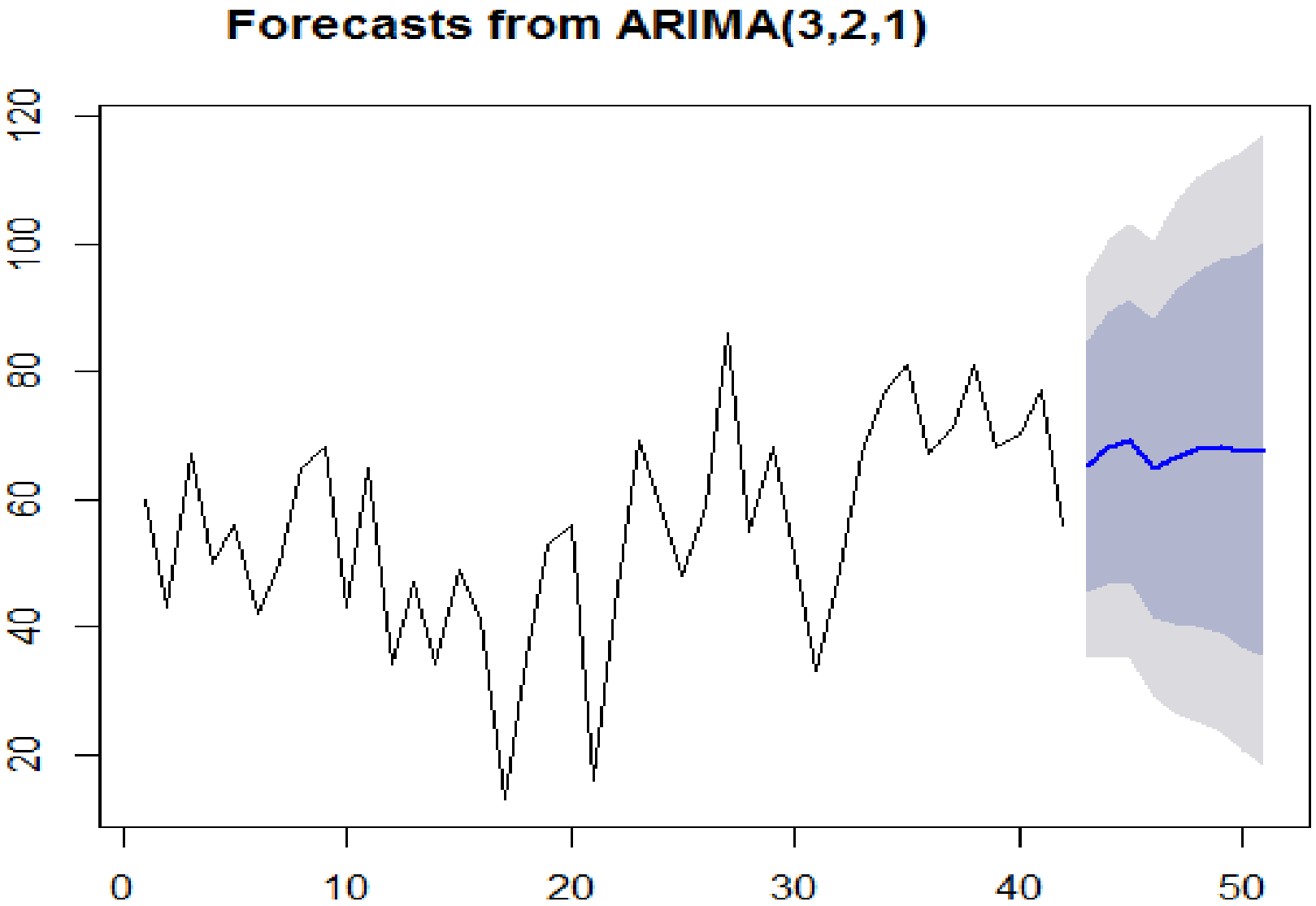


forecasting

We can then use the ARIMA model to make forecasts for future values of the time series, using the **forecast.Arima** function

Example 9

```
> cars.forecast<-forecast.Arima(fit3.2.1, h=9)
> cars.forecast
Point Forecast    Lo 80     Hi 80    Lo 95     Hi 95
43       65.07456 45.49324  84.65589 35.12751  95.02162
44       68.04095 46.70211  89.37979 35.40601 100.67589
45       69.17146 46.93815  91.40477 35.16855 103.17438
46       64.85213 41.50081  88.20345 29.13937 100.56489
47       66.53699 40.45408  92.61991 26.64662 106.42737
48       67.82415 39.99911  95.64919 25.26942 110.37888
49       68.21501 39.01372  97.41629 23.55550 112.87452
50       67.44250 36.80105  98.08395 20.58045 114.30455
51       67.90625 35.52907 100.28343 18.38963 117.42288
```


Forecasts from ARIMA(3,2,1)
120
100
80
60
40
20
0
10
20
30
40
50