time_series

Contents

1 Read Time Series Date **1**

2 Plotting Time Series **2**

3 Decomposing Time Series **2**
3.1 Decomposing Non-Seasonal Date . . . 2
3.2 Decomposing Seasonal Date . . . 3
3.3 Seasonally Adjusting . . . 4

4 Forecasts using Exponential Smoothing **5**
4.1 Simple Exponential Smoothing . . . 5
4.2 Specifiy the inital value for the level in the HW model . . . 7
4.3 forecast.HoltWinters() . . . 7

1 Read Time Series Date

- scan() : read data into a vector or list from the console or file
- ts() : store the data into time series format

```r
# alt+ctrl+i = create a new code chunk
# import data
kings <- scan("http://robjhyndman.com/tsdldata/misc/kings.dat",skip=3)
kings
```

```
##  [1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48
## [26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56
```

```r
# store the data in a time series object
# ts(), frequency = 'the time interval data collected', such as 12 for montly, 4 for quarter
# start(1982,2) = the data start from 1982 second quarter
kingstimeseries <- ts(kings)
kingstimeseries
```

```
## Time Series:
## Start = 1
## End = 42
## Frequency = 1
##  [1] 60 43 67 50 56 42 50 65 68 43 65 34 47 34 49 41 13 35 53 56 16 43 69 59 48
## [26] 59 86 55 68 51 33 49 67 77 81 67 71 81 68 70 77 56
```

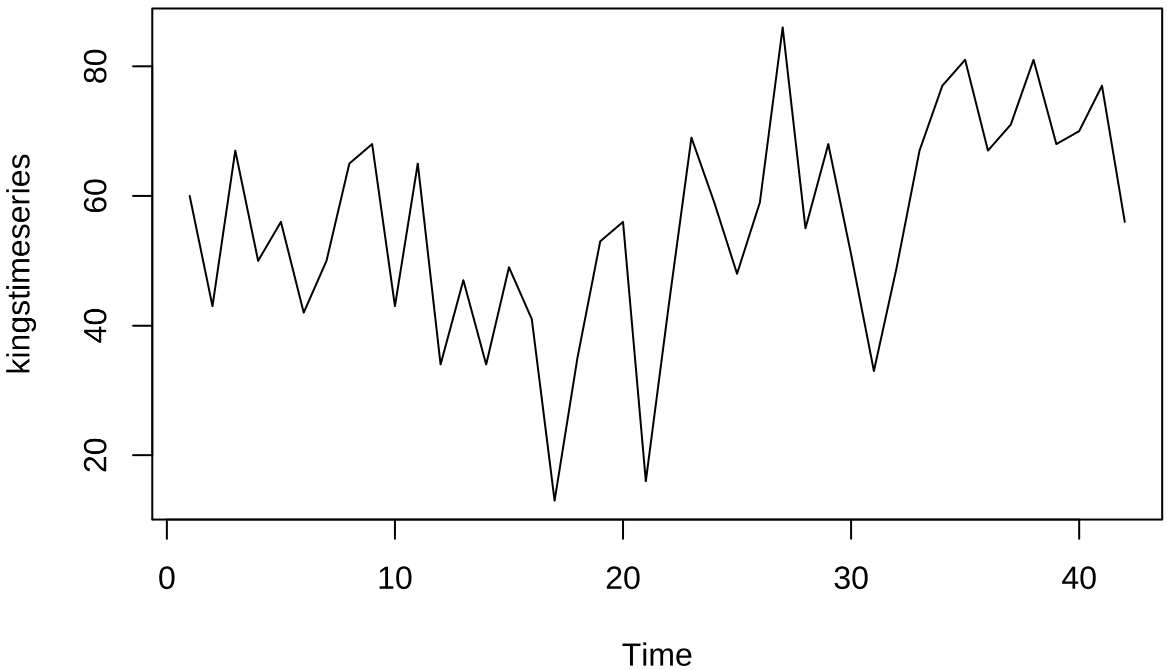
```r
# the example of advanced use
births <- scan("http://robjhyndman.com/tsdldata/data/nybirths.dat")
births.ts <- ts(births, frequency = 12,start = c(1946,1))
knitr::kable(head(births.ts),'markdown')
```

x
26.663
23.598
26.931
24.740
25.806
24.364

2 Plotting Time Series

- addictive model : be able to decribe seasonal data but trend
  - trend data can be transformed into log to decrease the trend effect.

```
plot.ts(kingstimeseries)
```



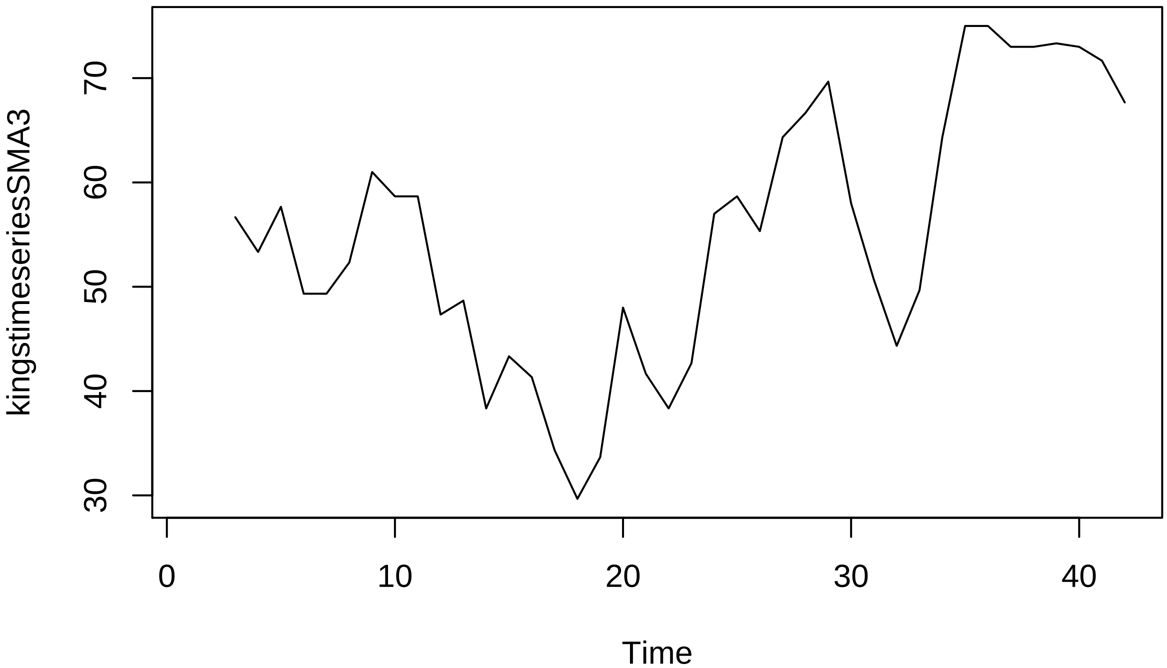
3 Decomposing Time Series

- to separat the data into its constituent components

3.1 Decomposing Non-Seasonal Date

- a non-seasonal time series consists of **a trend component** and **an irregular component**
- to estimate the trend component of a non-seasonal time series that can be described using an additive model : a smoothing mothod, such as MA.
- SMA() :
  - in "TTR" package
  - is used to smooth time series data, a simple moving average
  - specify the order of the simple moving average, n
    - to estimate the trend componet, might try smoothing the data with a simple moving average of a higher order
    - trial-and-error

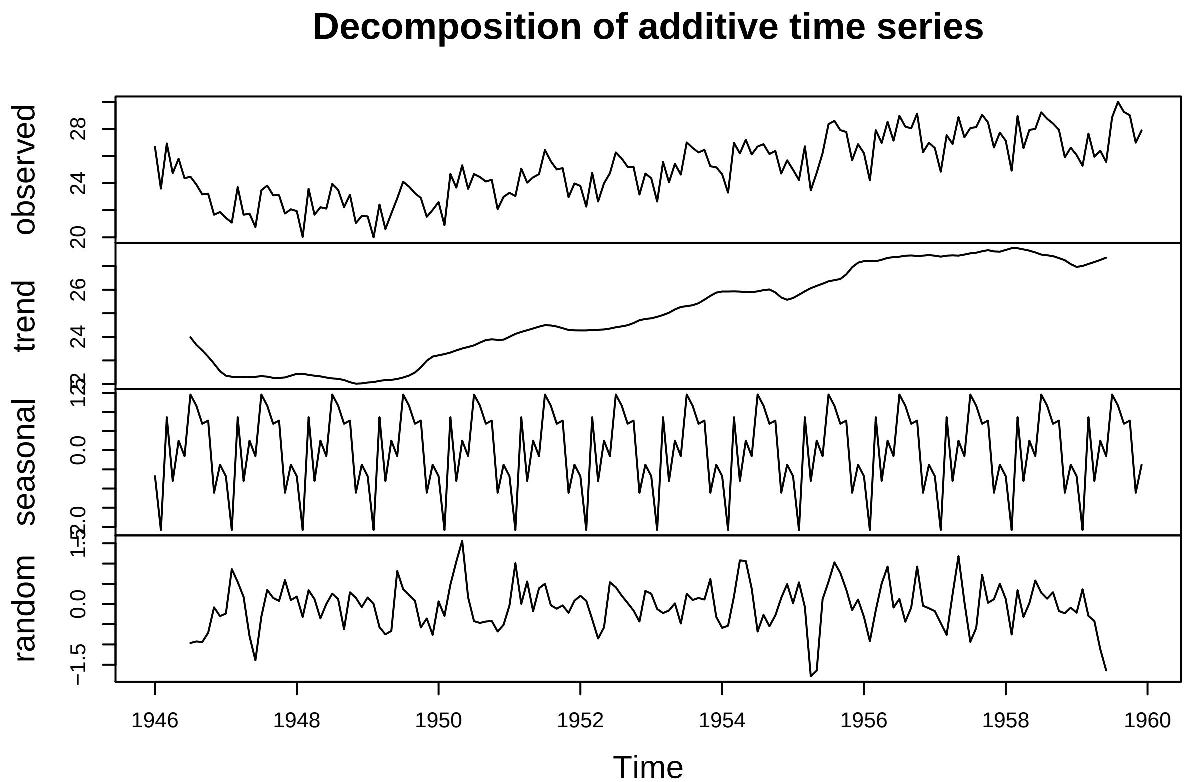
```r
library(TTR)
kingstimeseriesSMA3 <- SMA(kingstimeseries, n=3)
plot.ts(kingstimeseriesSMA3)
```



3.2 Decomposing Seasonal Date

- a seasonal data consists of **a trend component**, **a seasonal component** and **an irregular component**.
- to estimate the trend component and seasonal component, using additive model
  - decompose() : decompose a time series into **seasonal**, **trend** and **irregular components**, using moving averages.
    - return a list object

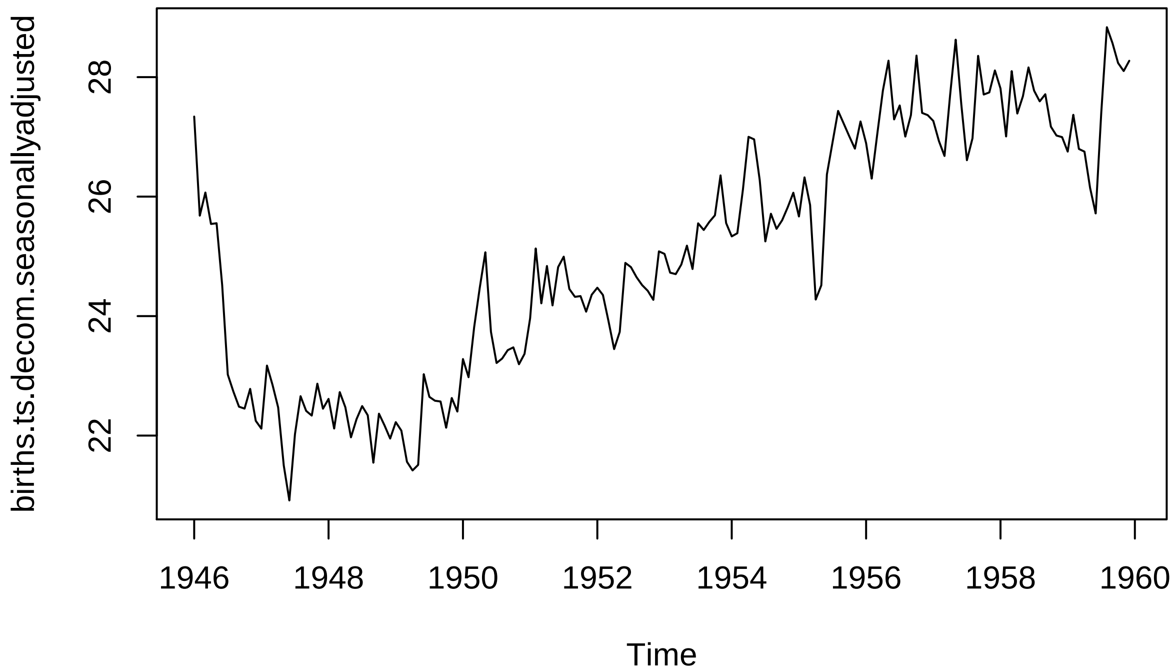
```r
births.ts.decom <- decompose(births.ts)
plot(births.ts.decom)
```



3.3 Seasonally Adjusting

- if the seasonal time series can be described by an additive model, the data can be seasonally adjusted by estimating the seasonal component and substracting the estmated seasonal component from the original time series.
- the seasonally adjusted result just contains **the trend component** and **an irregular component*
- decompose()

```
births.ts.decom.seasonallyadjusted <- births.ts - births.ts.decom$seasonal
plot(births.ts.decom.seasonallyadjusted)
```



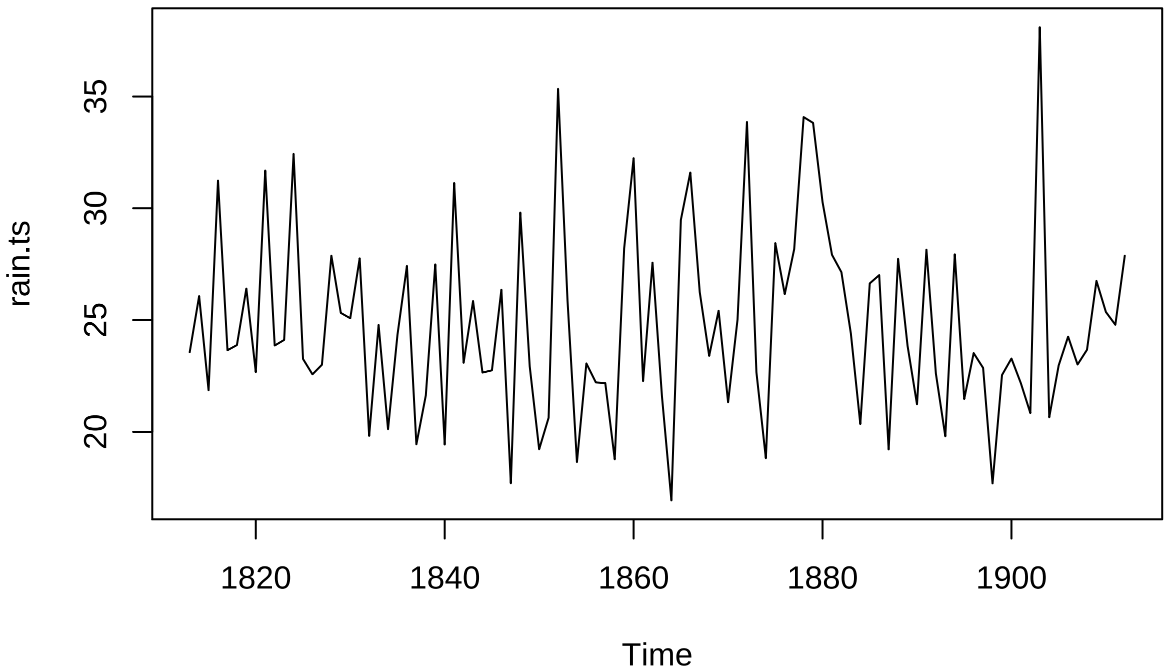
4 Forecasts using Exponential Smoothing

- being used to make short-time forecasts for time series data

4.1 Simple Exponential Smoothing

- if the data can be described using an additice model with **constant level** and **no seasonality**
- using simple exponential smoothing to make short-time forcasts
- alpha : lies between 0 and 1
  - close to 0 : meaning little weight is placed on the most recent observations when making forecasts of future values
- HoltWinters() : simple exponential smoothing
  - beta = FALSE ; gamma = FALSE : for Holt's exponential smoothing or Holt-Winters exponential smoothing
  - **fitted** : the forcasts made by HoltWinters is stored in fitted

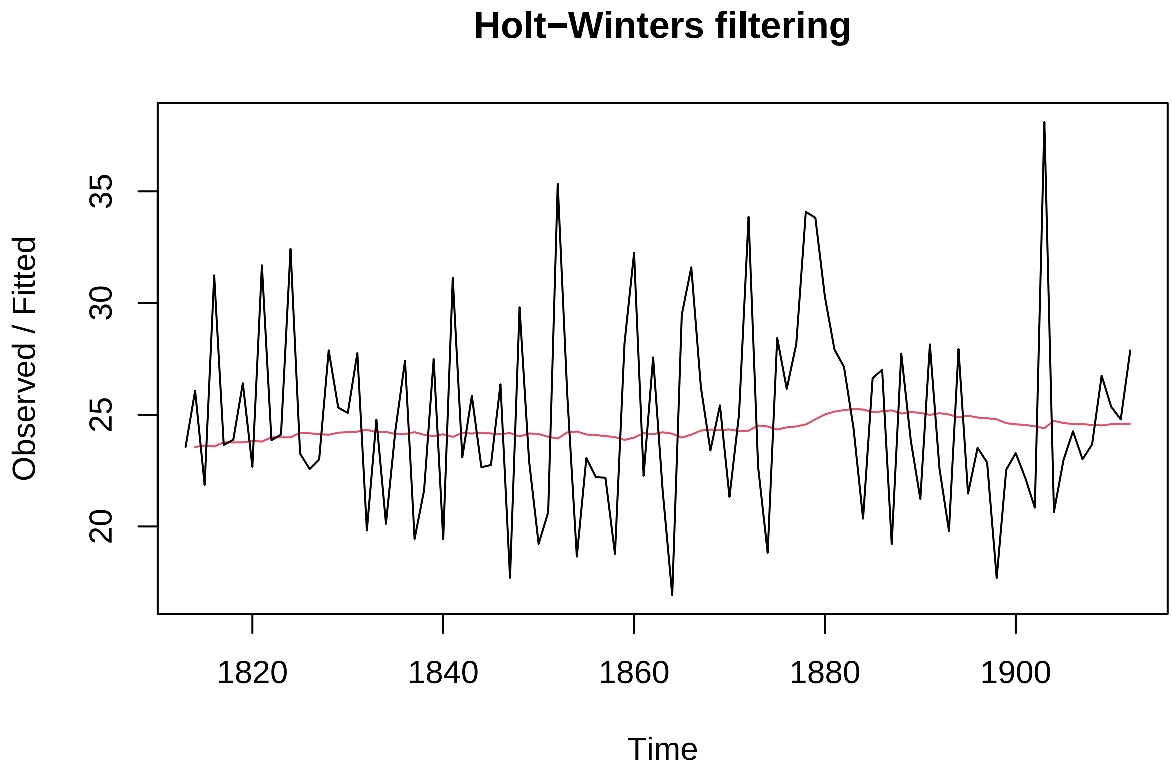
```
rain <- scan("http://robjhyndman.com/tsdldata/hurst/precip1.dat",skip=1)
rain.ts <- ts(rain,start = 1813)
plot.ts(rain.ts)
```




```r
rain.ts.forcast <- HoltWinters(rain.ts,beta=FALSE,gamma=FALSE)
knitr::kable(head(rain.ts.forcast$fitted))
```

xhat	level
23.56000	23.56000
23.62054	23.62054
23.57808	23.57808
23.76290	23.76290
23.76017	23.76017
23.76306	23.76306

```r
plot(rain.ts.forcast)
```



- the accuracy offorcasting : the sum of squared errors
  - $SSE

4.2 Specifiy the inital value for the level in the HW model

- l.start

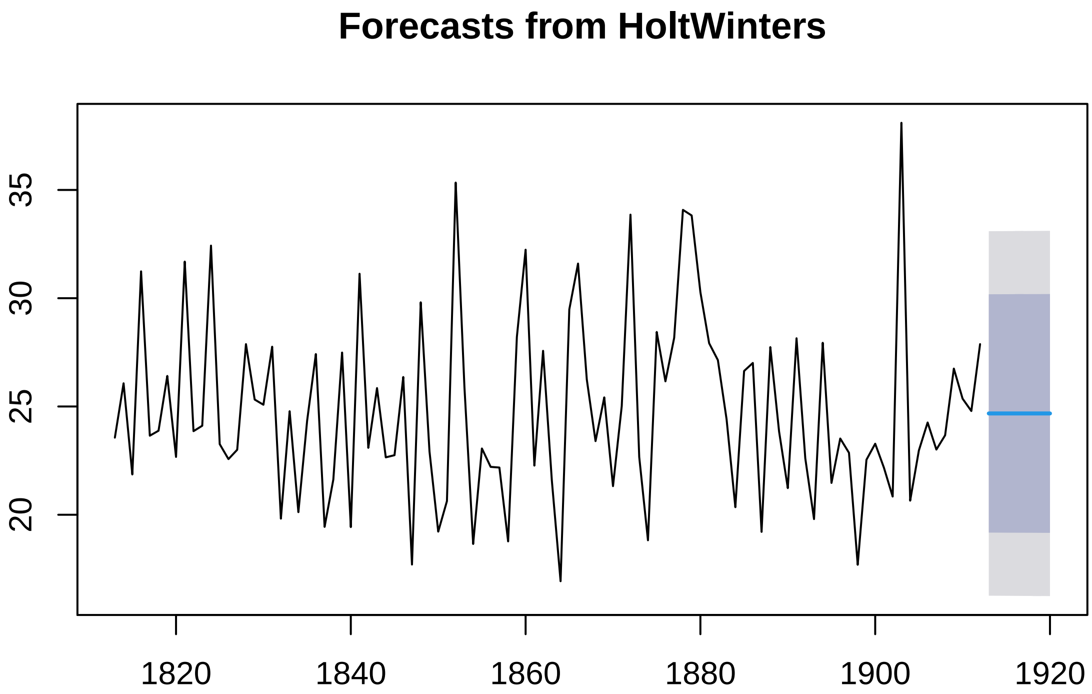
4.3 forecast.HoltWinters()

- make forecast for further time point
- *forecast* package
- h : how many further time points going to make forecast **forecast.HoltWinter() could not find in the package stats::forecast.HoltWinters() is not availibale for R 4.0 forecast::forecast() has the same function**
- the predicted area, 80% and 95% prediction interval

```
require('forecast')
rain.ts.forcast2 <- forecast::forecast(rain.ts.forcast, h=8)
knitr::kable(rain.ts.forcast2,'markdown')
```

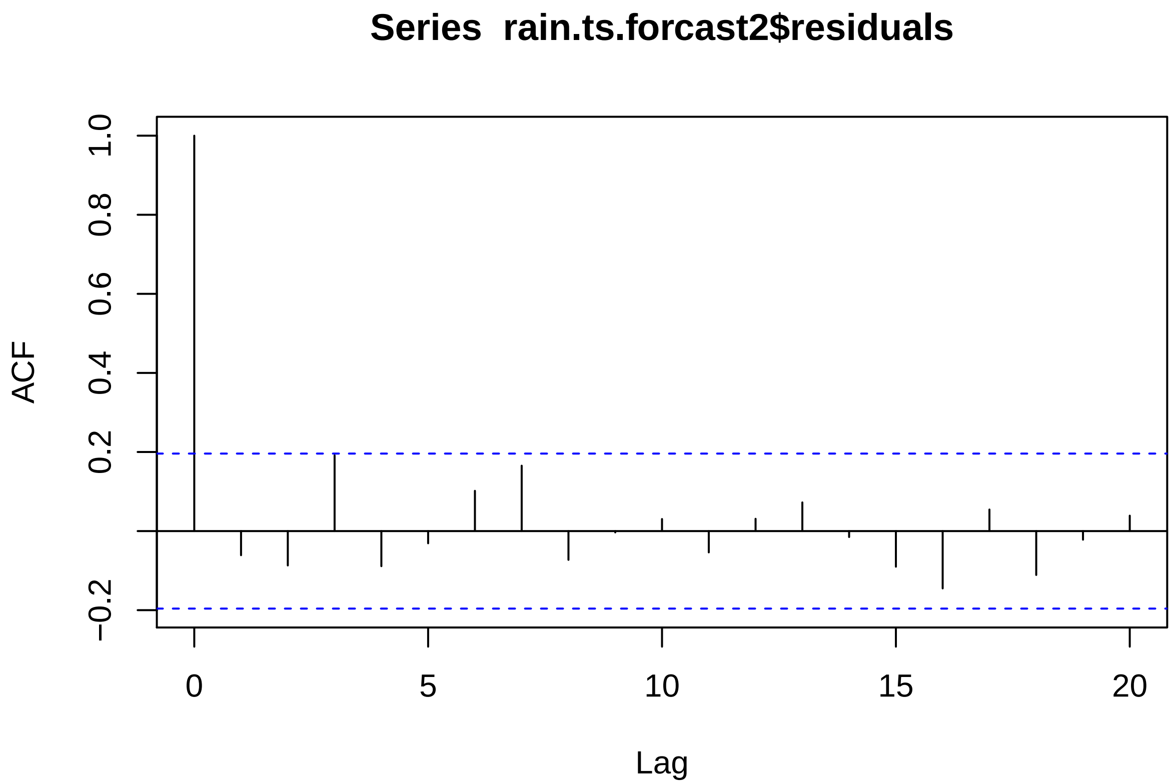
	Point Forecast	Lo 80	Hi 80	Lo 95	Hi 95
1913	24.67819	19.17493	30.18145	16.26169	33.09470
1914	24.67819	19.17333	30.18305	16.25924	33.09715
1915	24.67819	19.17173	30.18465	16.25679	33.09960
1916	24.67819	19.17013	30.18625	16.25434	33.10204
1917	24.67819	19.16853	30.18785	16.25190	33.10449
1918	24.67819	19.16694	30.18945	16.24945	33.10694
1919	24.67819	19.16534	30.19105	16.24701	33.10938
1920	24.67819	19.16374	30.19265	16.24456	33.11182

```r
plot(rain.ts.forcast2)
```



- if there are correlations between **forecast errors** for **successive predictions**, it is likely that the simple exponential smoothing forecasts could be improved upon by another forecasting technique.
- forecast error : a measure of accurancy of the predictive model is SSE, sum of squared error for the **in-sample** forecast errors. it is stored in **residual** of the forecast of HoltWinters.
- if the SSE could be improved, the predictive model should be improved as well.
- acf() : showing a correlogram of the **in-sample** forecast errors, to specifiy the max lag

```r
acf(rain.ts.forcast2$residuals,lag.max = 20, na.action=na.pass)
```




```
#na.action=na.pass  to pass the NA data
```

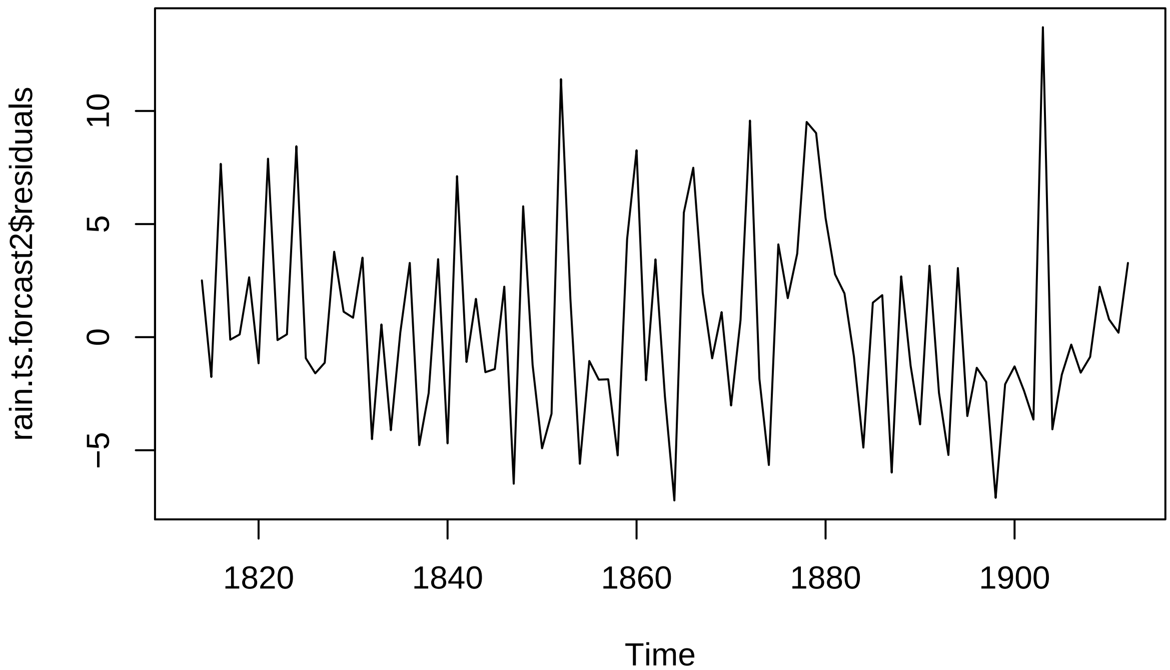
- the lag3 is just touching the significance bounds
- **Ljung-Box text** : to test whether there is significant evidence for non-zero correlations at lags 1-20.
- Box.test() : Ljung-Box test

```
Box.test(rain.ts.forcast2$residuals,lag = 20,type = 'Ljung-Box')
```

```
##
##  Box-Ljung test
##
## data:  rain.ts.forcast2$residuals
## X-squared = 17.401, df = 20, p-value = 0.6268
```

- the p-value is 0.6, so there is a little evidence of non-zero autocorrelation in the in-sample forecast error at lags1.20.
- checking the normal distribution with mean zero and constant variance : to be sure that the predictive model cannot be improved upon.

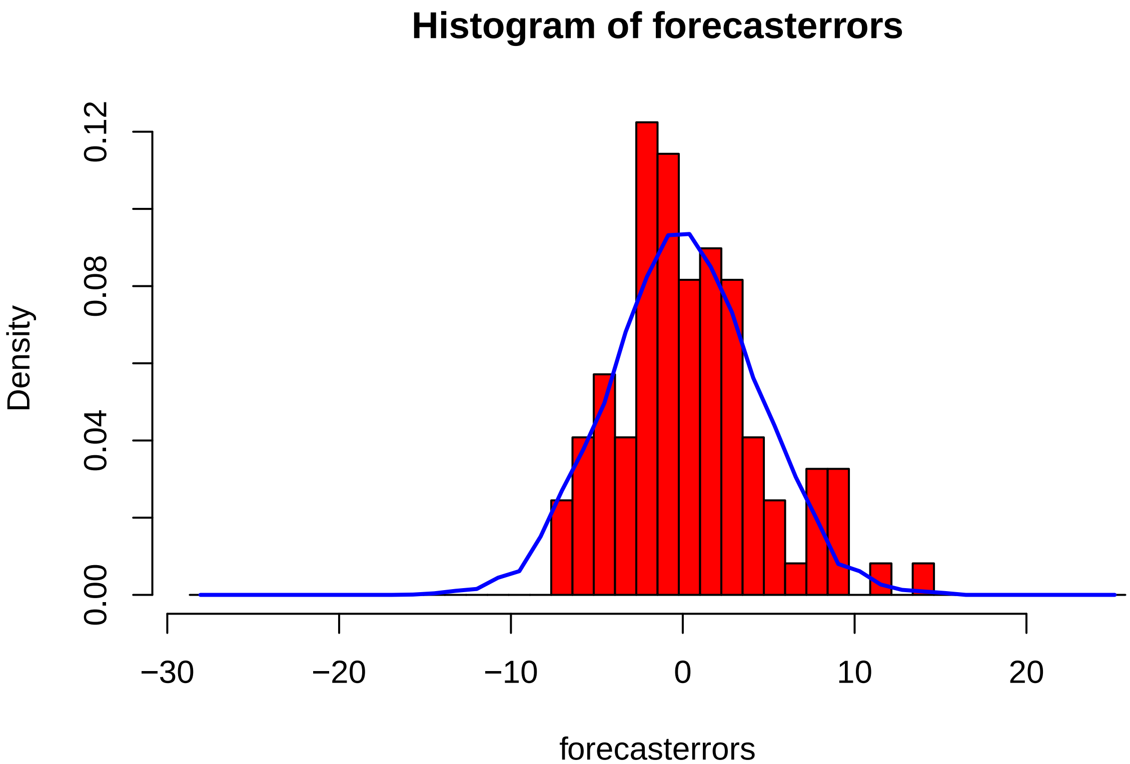
```
plot.ts(rain.ts.forcast2$residuals)
```



*

histogtam plotting : to check whether the forecast errors are normally distributed with mean zero.

```
plotForecastError <- function(forecasterrors){
     forecasterrors <- na.omit(forecasterrors)
  # make a histogram of the forecast errors:
     mybinsize <- IQR(forecasterrors)/4
     mysd   <- sd(forecasterrors)
     mymin  <- min(forecasterrors) - mysd*5
     mymax  <- max(forecasterrors) + mysd*3
     # generate normally distributed data with mean 0 and standard deviation mysd
     mynorm <- rnorm(10000, mean=0, sd=mysd)
     mymin2 <- min(mynorm)
     mymax2 <- max(mynorm)
     if (mymin2 < mymin) { mymin <- mymin2 }
     if (mymax2 > mymax) { mymax <- mymax2 }
     # make a red histogram of the forecast errors, with the normally distributed data overlaid:
     mybins <- seq(mymin, mymax, mybinsize)
     hist(forecasterrors, col="red", freq=FALSE, breaks=mybins)
     # freq=FALSE ensures the area under the histogram = 1
     # generate normally distributed data with mean 0 and standard deviation mysd
     myhist <- hist(mynorm, plot=FALSE, breaks=mybins)
     # plot the normal curve as a blue line on top of the histogram of forecast errors:
     points(myhist$mids, myhist$density, type="l", col="blue", lwd=2)
}
plotForecastError(rain.ts.forcast2$residuals)
```



- the Ljung-Box : little evidence of non-zero autocorrelation
- the normal distribution with mean zero
- conclusion : the simple exponential smoothing method provides an adequate predictive model.