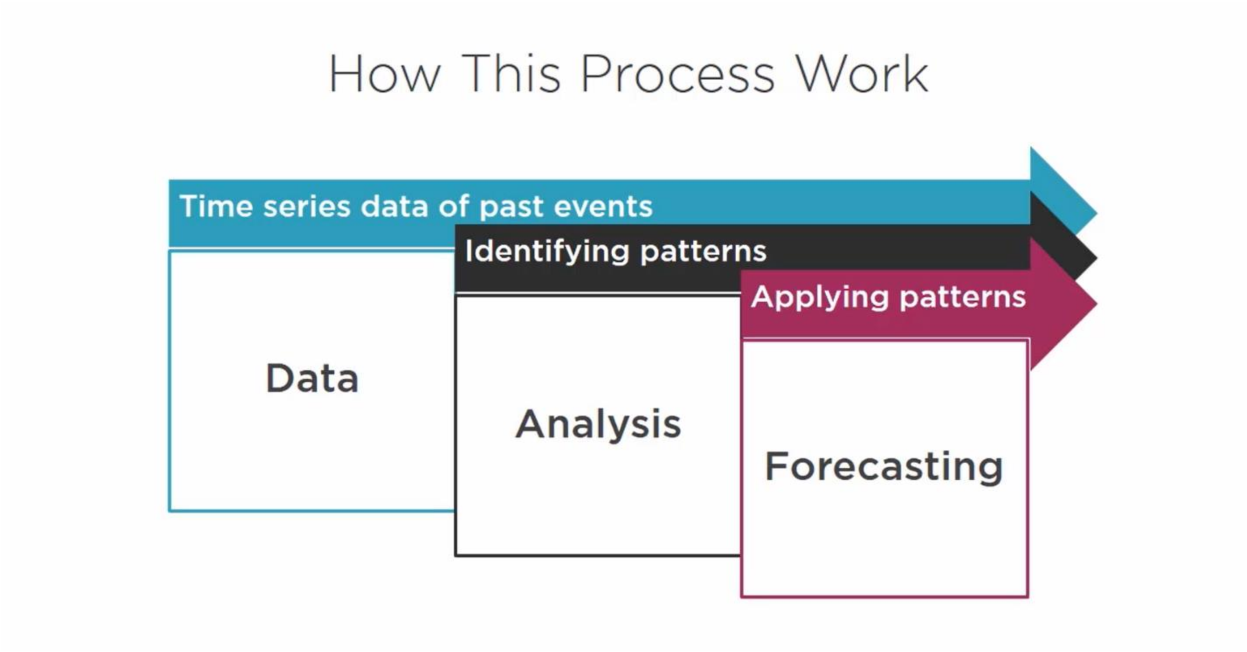
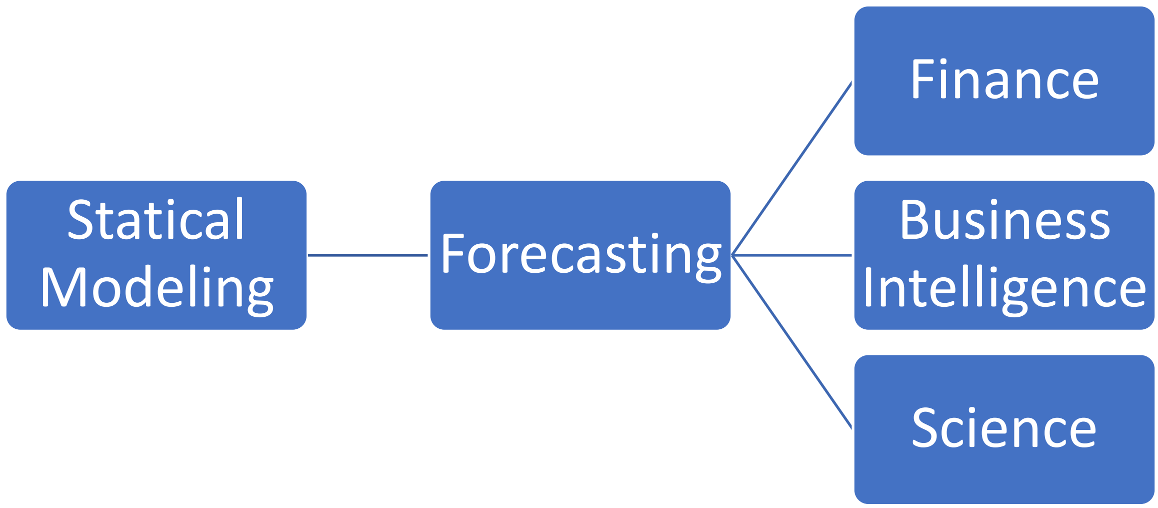
Time Series Analysis

Source: https://app.pluralsight.com/library/courses/r-time-series-analysis-forecasting/table-of-contents

Time Series Analysis Background:





Univariate Time Series:

One variable attached to a time stamp.

1. Linear
    - ARIMA
    - Exponential Smoothing
    - Simple Methods

2. Non-linear
    - K Nearest Neighbors
    - Clustering
    - Neural Nets
    - Support Vector Machines
    - Q Learning
    - Decision Trees

Multivariate Time Series:

Two or more variables attached to a time stamp.

Datasets:

1. Lynx trapping in Canada.
2. Temperature measurements in Nottingham.
3. Randomly generated series.

Time Series	Vector
The time stamp specifies a successive order for the values.	A unique ID does not necessarily provide a specific order to the data.

Converting Vectors to Time Series:

1. Functions ts()
    - Attaches a time stamp to a vector
    - Converts the class to ‘ts’
    - Use it to build time series from scratch
2. Library (xts)
    - Importing time series data into R

Lag:

A gap between two or more observations.

Y_t: An observation of the time series

Y_{114} = 3396 (The last observation of 'lynx')

Lag of 1 = $Y_t - Y_{t-1} = Y_{114} - Y_{113}$ = 3396-2657

Lag of 2 = $Y_t - Y_{t-2} = Y_{114} - Y_{112}$ = 3396 - 1590

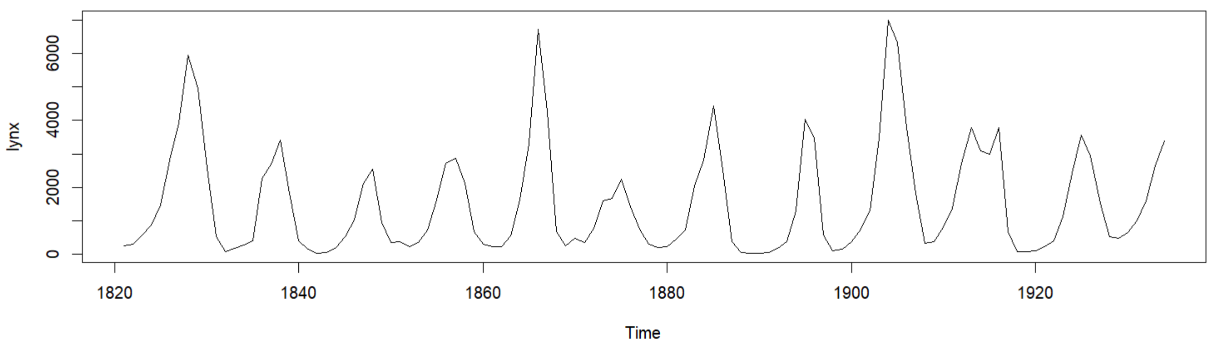
Lynx Dataset:

- mean()
- median()
- plot()
- sort()
- quantile()

Results:

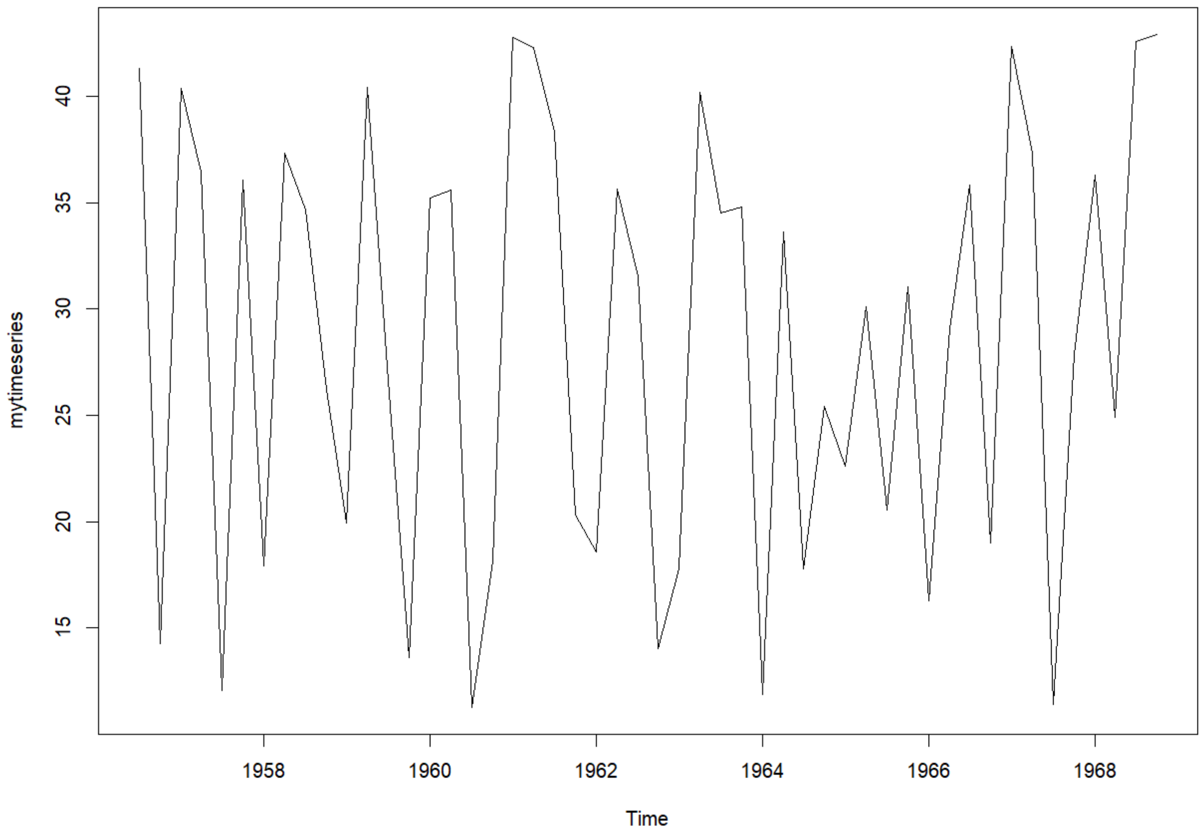
```
> head(lynx)
[1]  269  321  585  871 1475 2821
> time(lynx)
Time Series:
Start = 1821
End = 1934
Frequency = 1
  [1] 1821 1822 1823 1824 1825 1826 1827 1828 1829 1830 1831 1832 1833 1834 1835 1836 1837 1838 1839 1840 1841 1842 1843
 [24] 1844 1845 1846 1847 1848 1849 1850 1851 1852 1853 1854 1855 1856 1857 1858 1859 1860 1861 1862 1863 1864 1865 1866
 [47] 1867 1868 1869 1870 1871 1872 1873 1874 1875 1876 1877 1878 1879 1880 1881 1882 1883 1884 1885 1886 1887 1888 1889
 [70] 1890 1891 1892 1893 1894 1895 1896 1897 1898 1899 1900 1901 1902 1903 1904 1905 1906 1907 1908 1909 1910 1911 1912
 [93] 1913 1914 1915 1916 1917 1918 1919 1920 1921 1922 1923 1924 1925 1926 1927 1928 1929 1930 1931 1932 1933 1934
> length(lynx)
[1] 114
> mean(lynx); median(lynx)
[1] 1538.018
[1] 771
> plot(lynx)
> sort(lynx)
  [1]   39   45   49   59   68   73   80   81   98  105  108  151  153  184  188  201  213  225
 [19]  229  229  236  245  255  269  279  299  299  321  345  358  360  361  377  377  382  387
 [37]  389  399  409  409  469  473  485  523  529  546  552  585  587  662  674  684  687  731
 [55]  736  756  758  784  808  871  957 1000 1033 1132 1292 1307 1388 1426 1475 1537 1590 1594
 [73] 1623 1638 1676 1824 1836 2042 2119 2129 2251 2285 2432 2511 2536 2577 2657 2685 2713 2725
 [91] 2811 2821 2871 2935 2985 3091 3311 3396 3409 3465 3495 3574 3790 3794 3800 3928 4031 4254
[109] 4431 4950 5943 6313 6721 6991
> quantile(lynx)
     0%     25%     50%     75%    100%
  39.00  348.25  771.00 2566.75 6991.00
> quantile(lynx, prob = seq(0, 1, length = 11), type = 5)
    0%    10%    20%    30%    40%    50%    60%    70%    80%    90%   100%
  39.0  146.7  259.2  380.5  546.6  771.0 1470.1 2165.6 2818.0 3790.4 6991.0
>
```

<u>Plot:</u>



<u>ts and mts:</u>

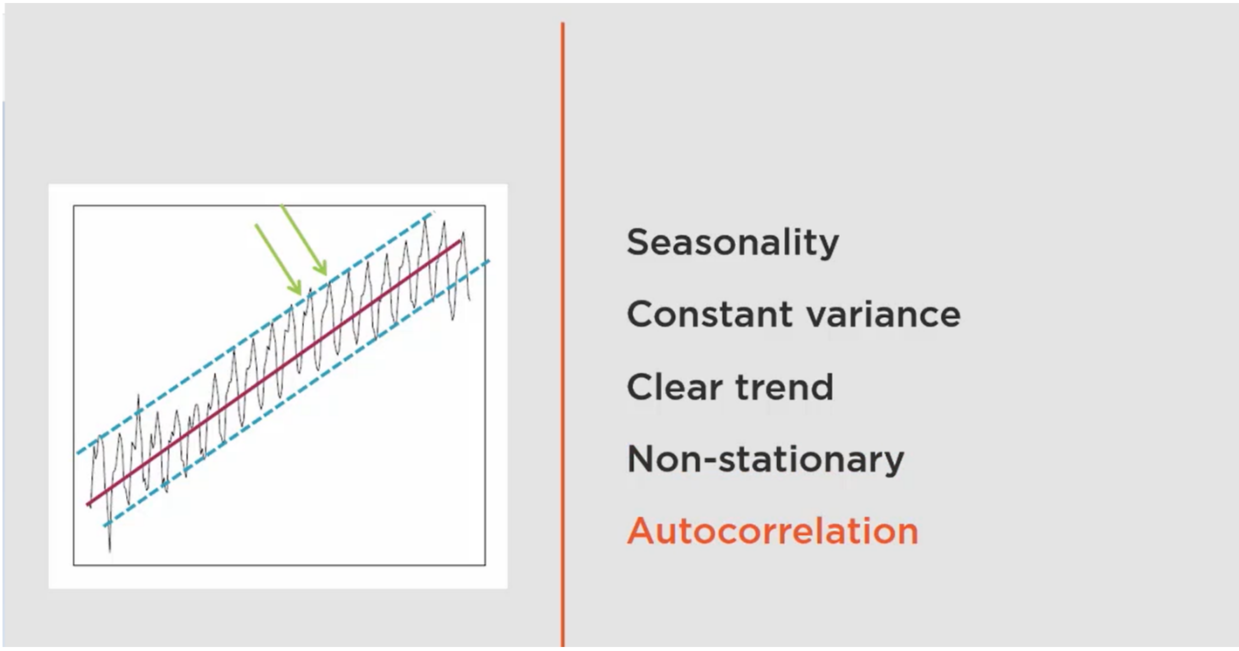
```
Console   Terminal ×   Background Jobs ×
R 4.2.1 · ~/
> #random uniform data between 10 to 45
> mydata = runif(n = 50, min = 10, max = 45)
> #packing into a quarterly time series
> mytimeseries = ts(data = mydata, start = c(1956,3), frequency = 4)
> plot(mytimeseries)
> class(mydata)
[1] "numeric"
> class(mytimeseries)
[1] "ts"
> class(lynx)
[1] "ts"
> #a typical mts data set
> class(EuStockMarkets); head(EuStockMarkets)
[1] "mts"    "ts"     "matrix"
         DAX    SMI    CAC   FTSE
[1,] 1628.75 1678.1 1772.8 2443.6
[2,] 1613.63 1688.5 1750.5 2460.2
[3,] 1606.51 1678.6 1718.0 2448.2
[4,] 1621.04 1684.1 1708.1 2470.4
[5,] 1618.16 1686.6 1723.1 2484.7
[6,] 1610.61 1671.6 1714.3 2466.8
> 
```

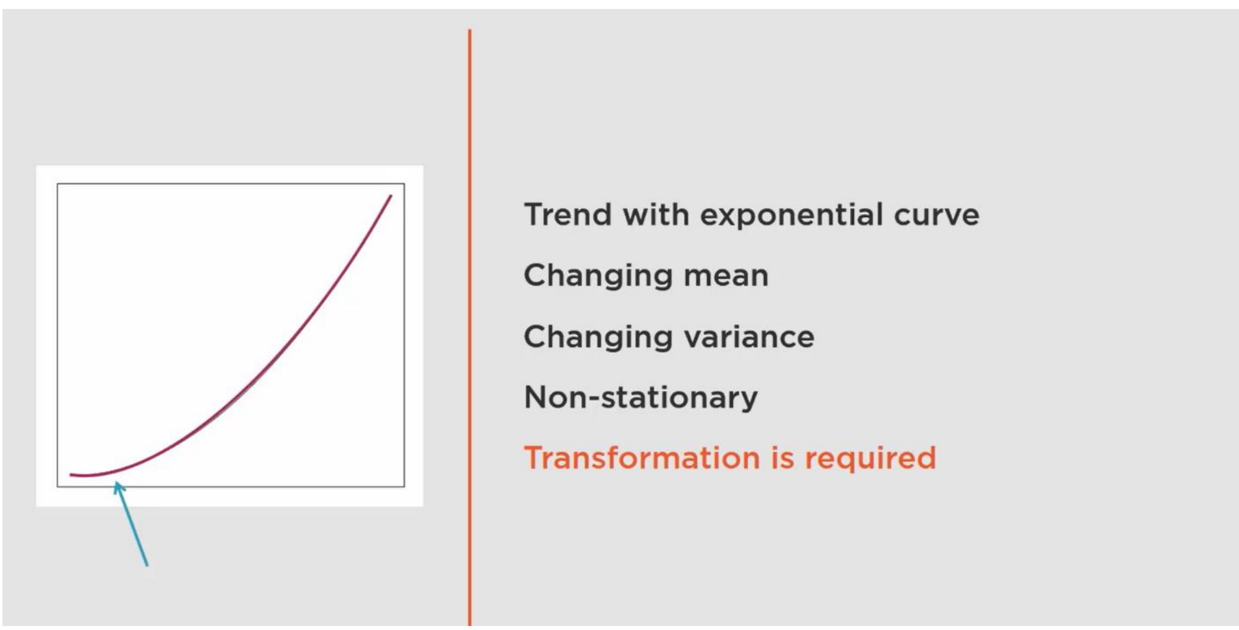


Pattern to Identify:

1. Trend: Dataset moving towards a direction.
2. Seasonality: Repeated pattern over a fixed interval.
3. Mean: The average of the dataset.
4. Variance: Indicator of variability.
5. Stationarity: Constant mean and variance.

Understanding Pattern Using Graph:



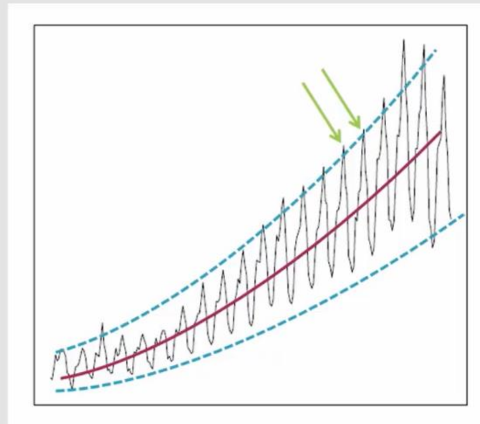


Exponential seasonality

Exponential trend

Changing variance

Non-stationary



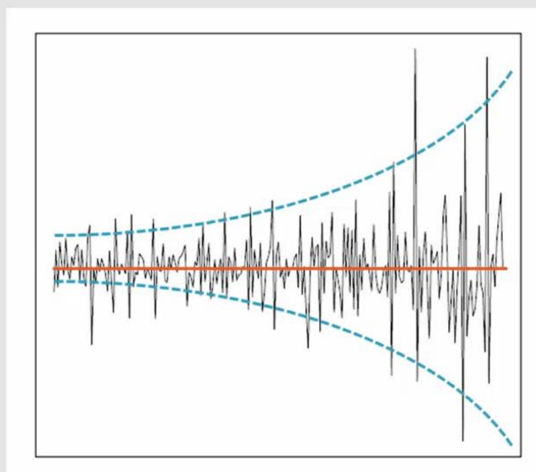
Constant mean

Changing variance

Heteroscedastic dataset

Non-stationary

Preprocessing prior analysis

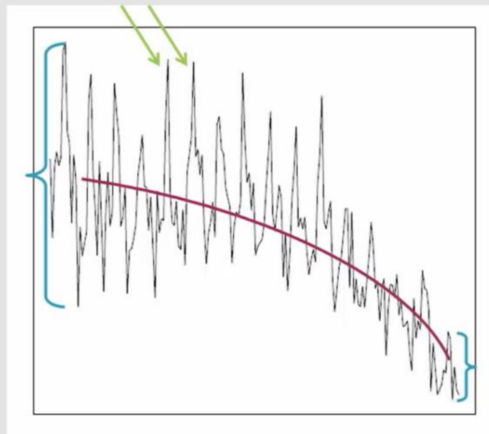


Seasonality

Changing variance

Trend

Non-stationary



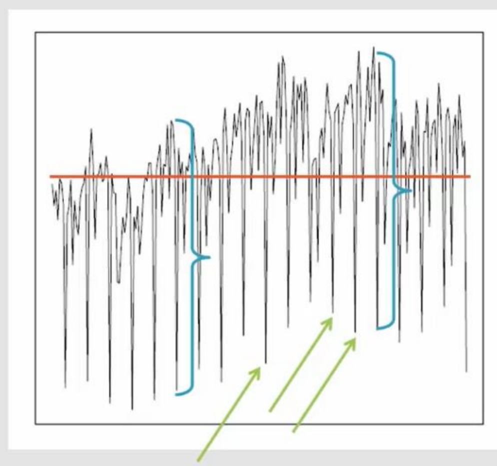
Seasonality

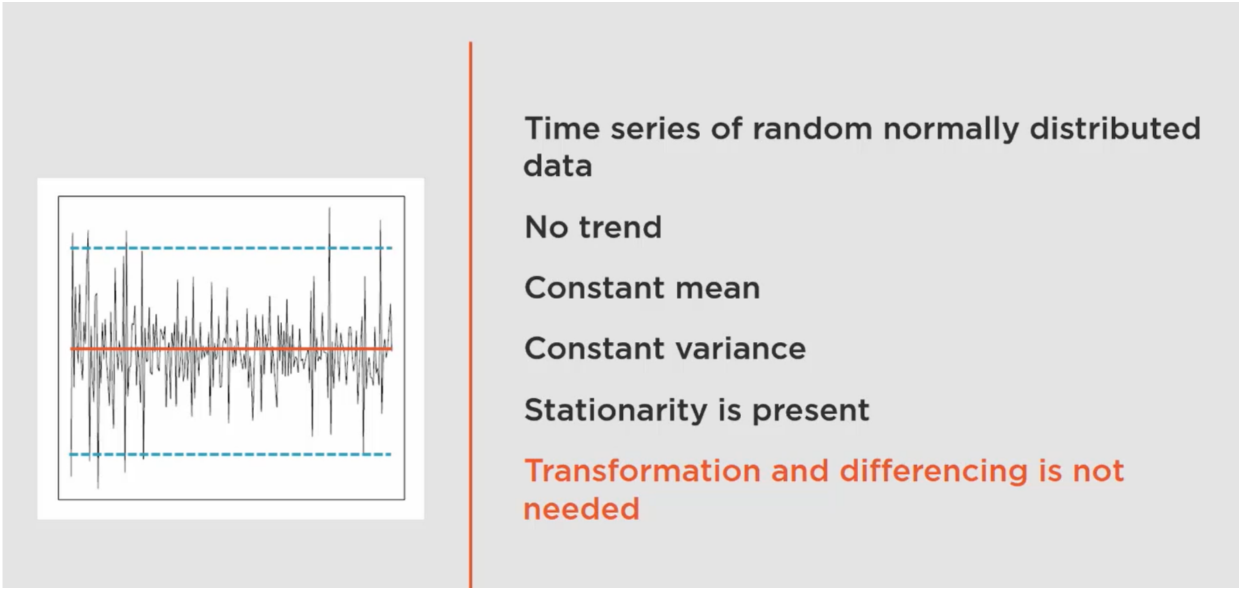
Constant mean

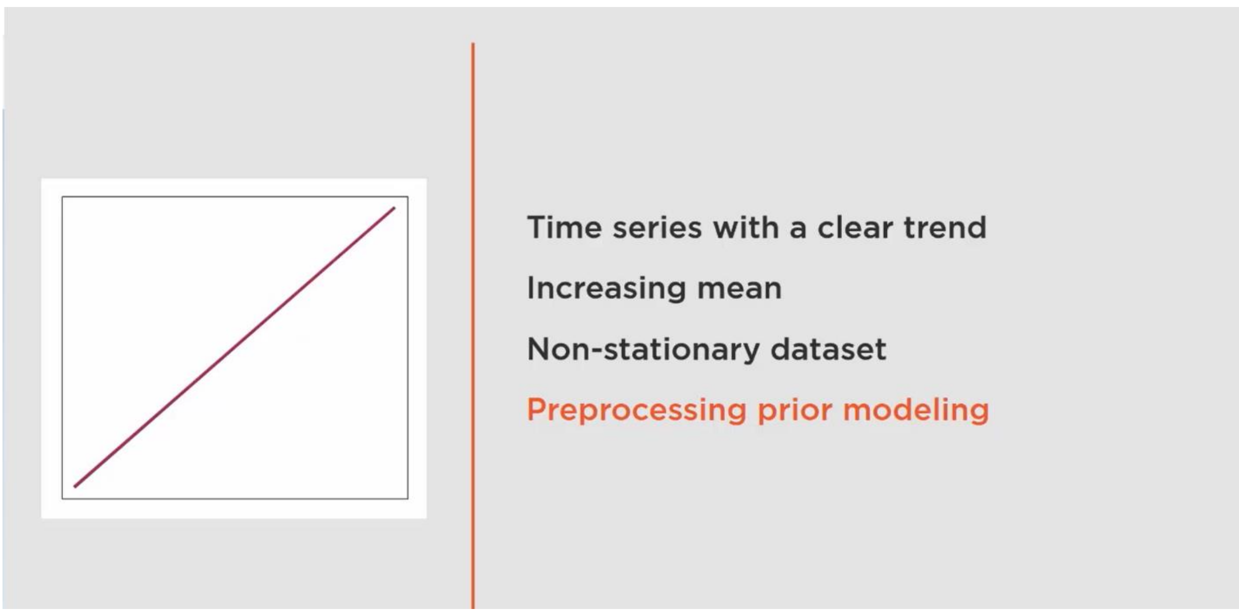
Constant variance

Non-stationary

Autocorrelation is present

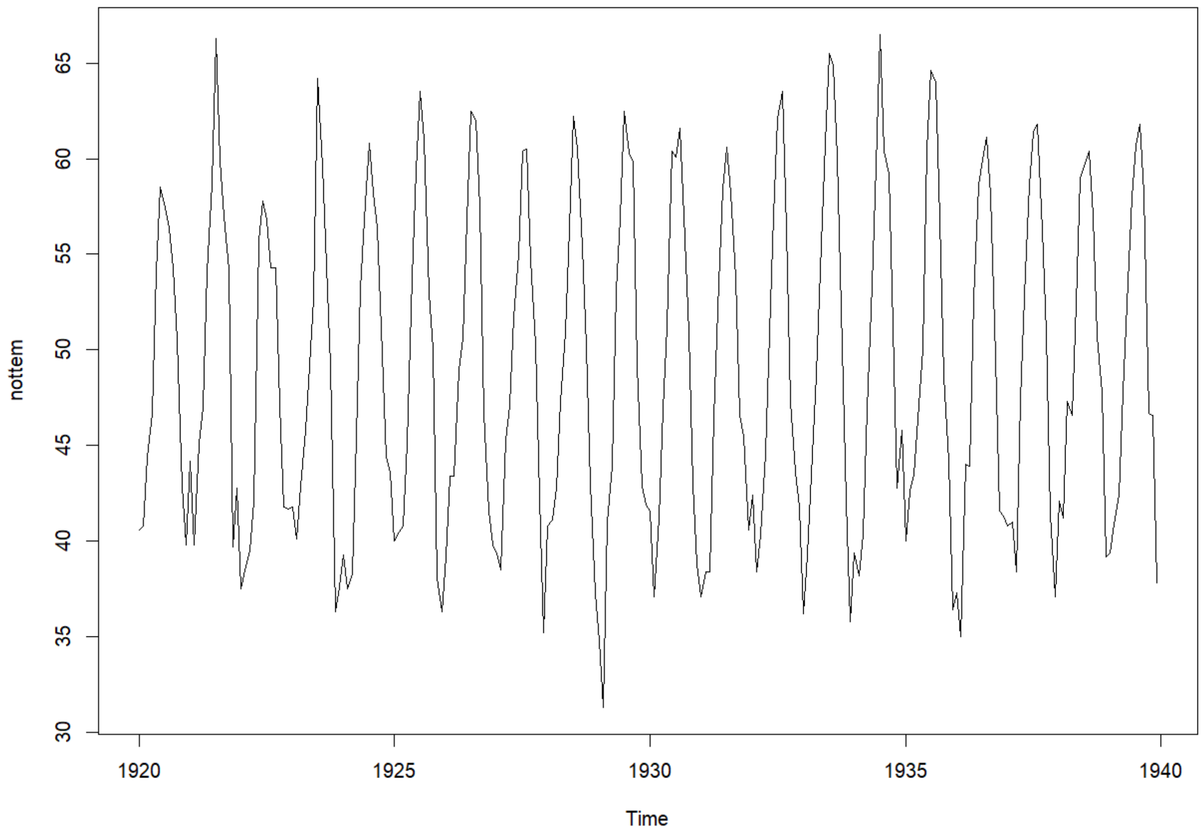




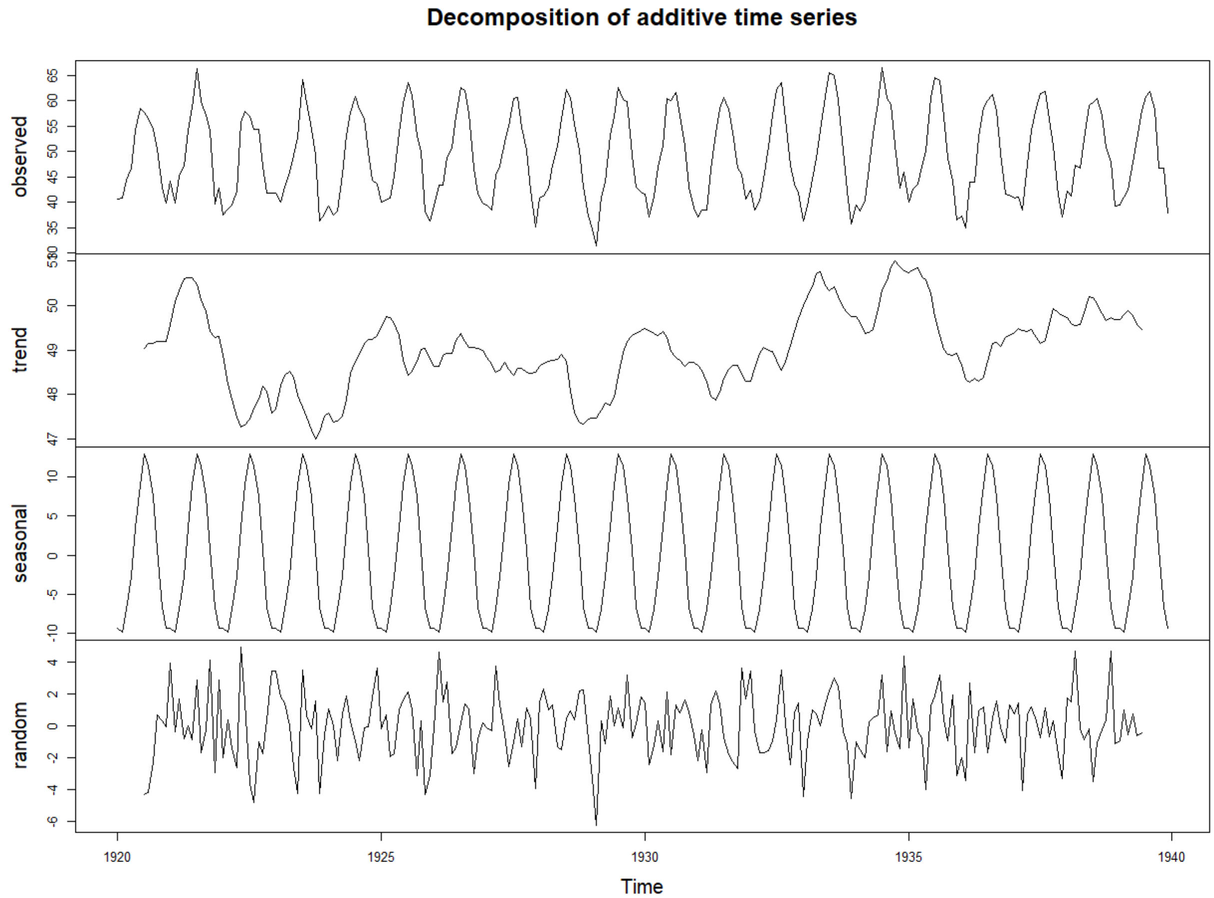


Nottem Dataset:

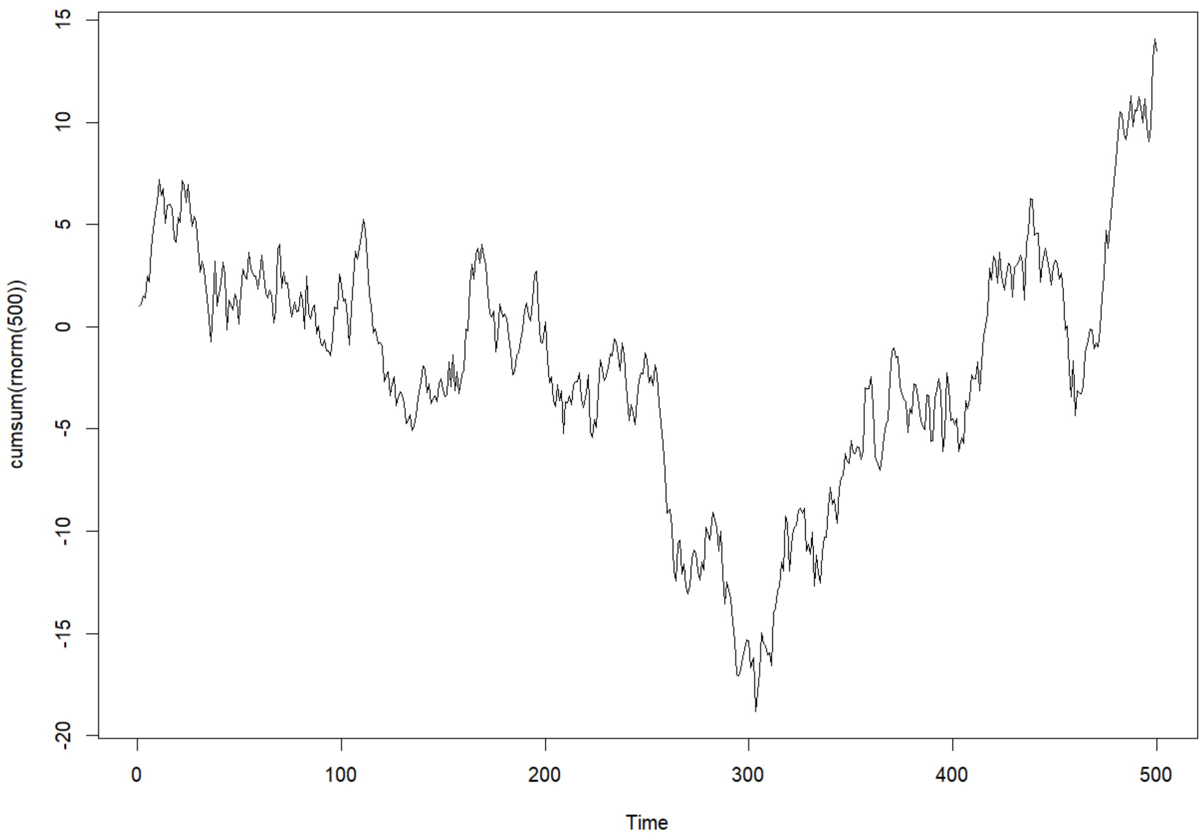
- plot()



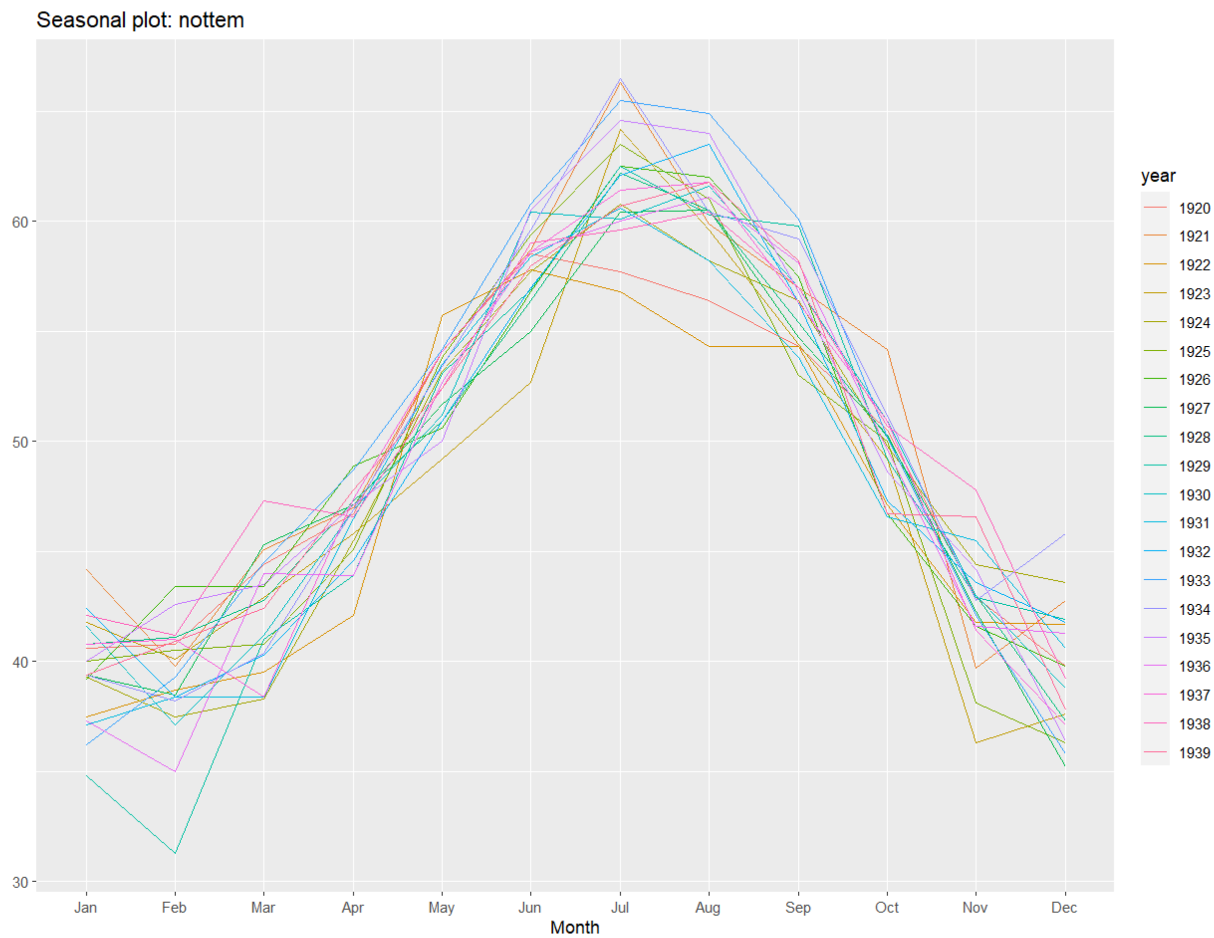
- decompose()


Decomposition of additive time series
observed
trend
seasonal
random
Time
1920
1925
1930
1935
1940

- plot.ts()

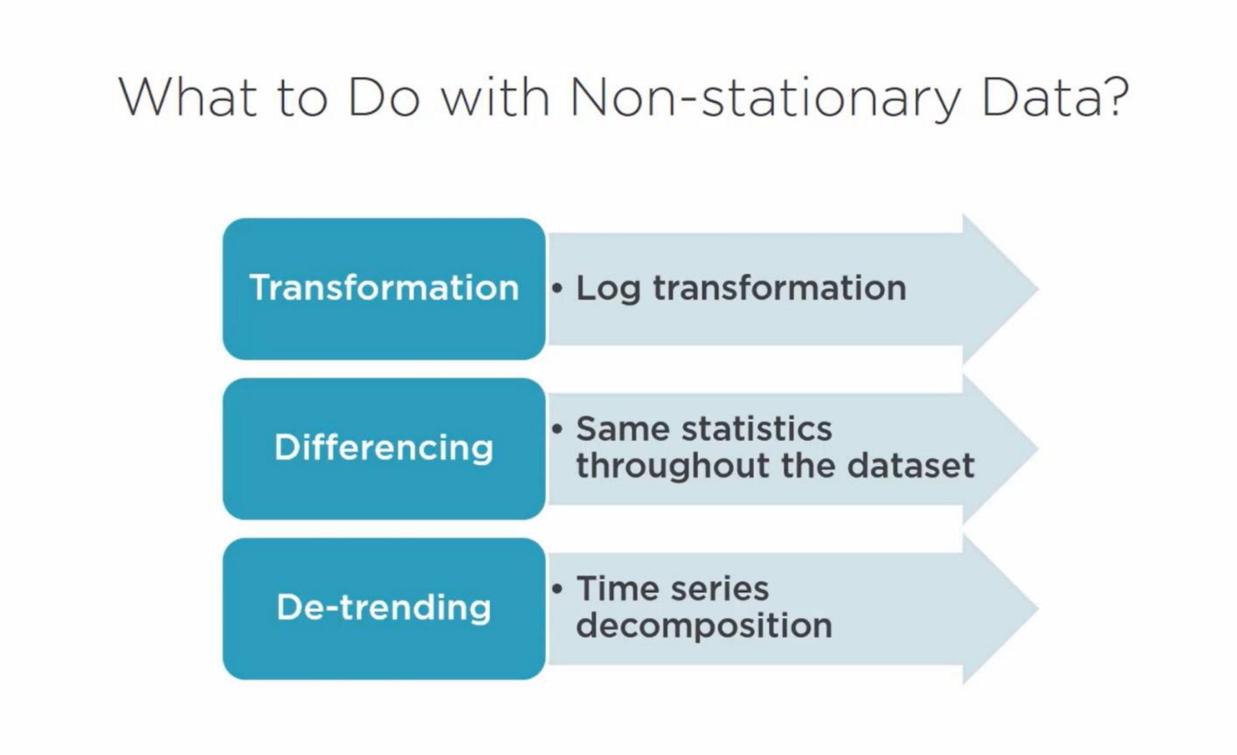


- ggseasonplot()



Seasonal plot: nottem
60
50
40
30
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sep
Oct
Nov
Dec
Month
year
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939

Stationarity:

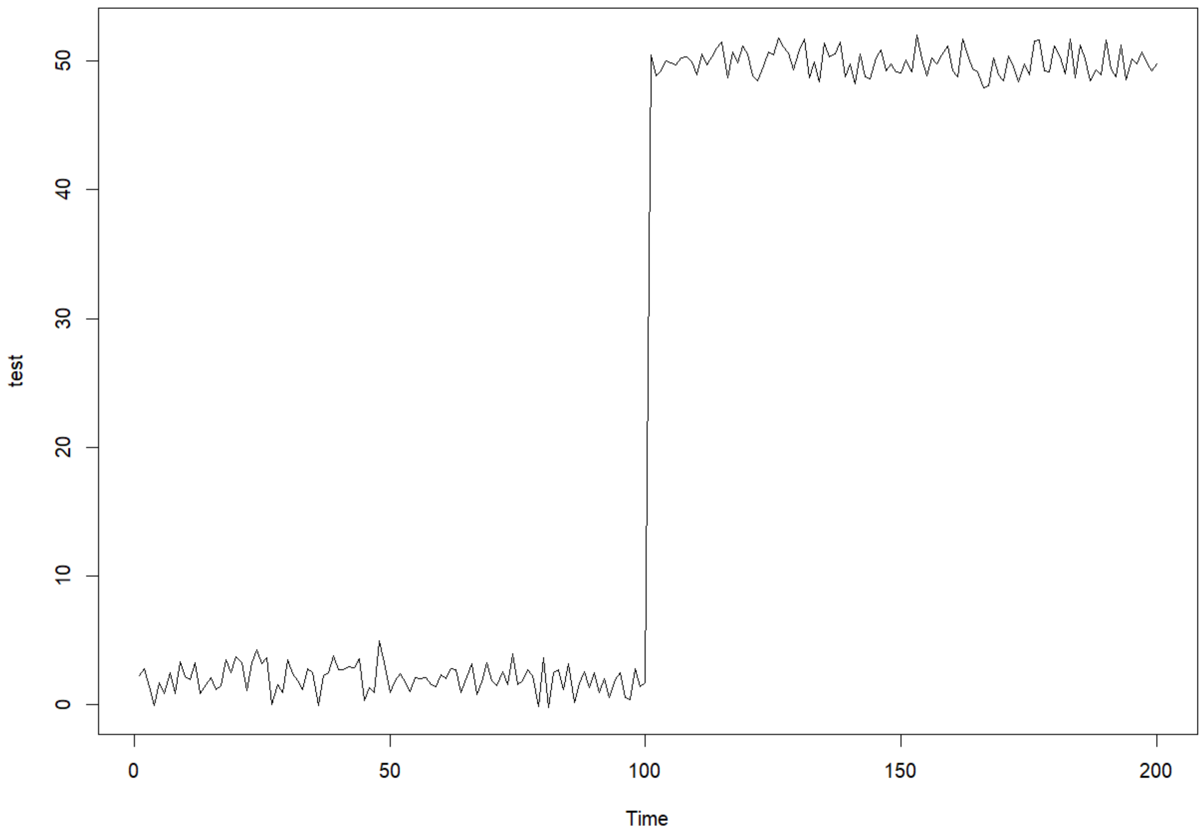


Console Terminal Background Jobs

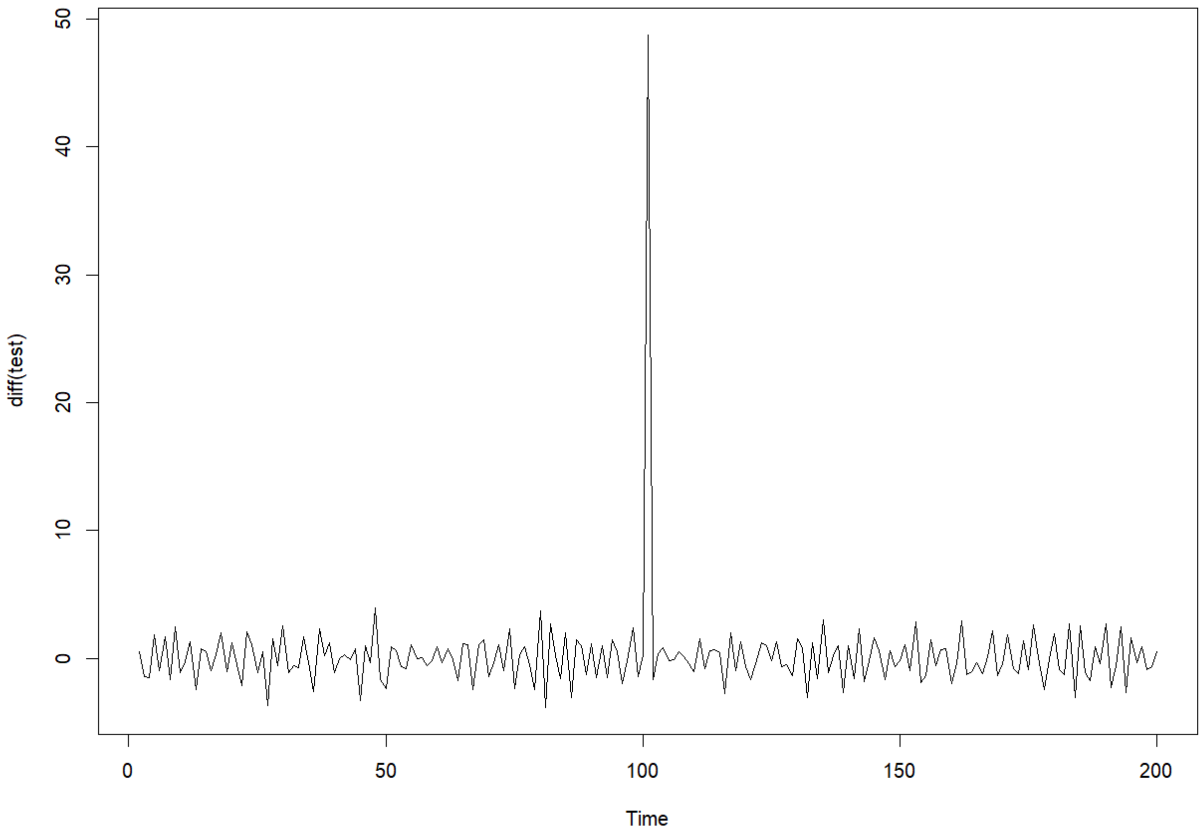
R 4.2.1 · ~/

```
> #Stationary
> test = ts(c(rnorm(100,2,1), rnorm(100,50,1)),start = 1)
> plot(test)
> plot(diff(test))
> #Unit Root Tests
> x = rnorm(1000) #random normal data
> library(tseries)
> adf.test(x)

        Augmented Dickey-Fuller Test

data:  x
Dickey-Fuller = -10.397, Lag order = 9, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(x) : p-value smaller than printed p-value
> plot(nottem) #Seasonal data
> adf.test(nottem)

        Augmented Dickey-Fuller Test

data:  nottem
Dickey-Fuller = -12.998, Lag order = 6, p-value = 0.01
alternative hypothesis: stationary

Warning message:
In adf.test(nottem) : p-value smaller than printed p-value
> 
```

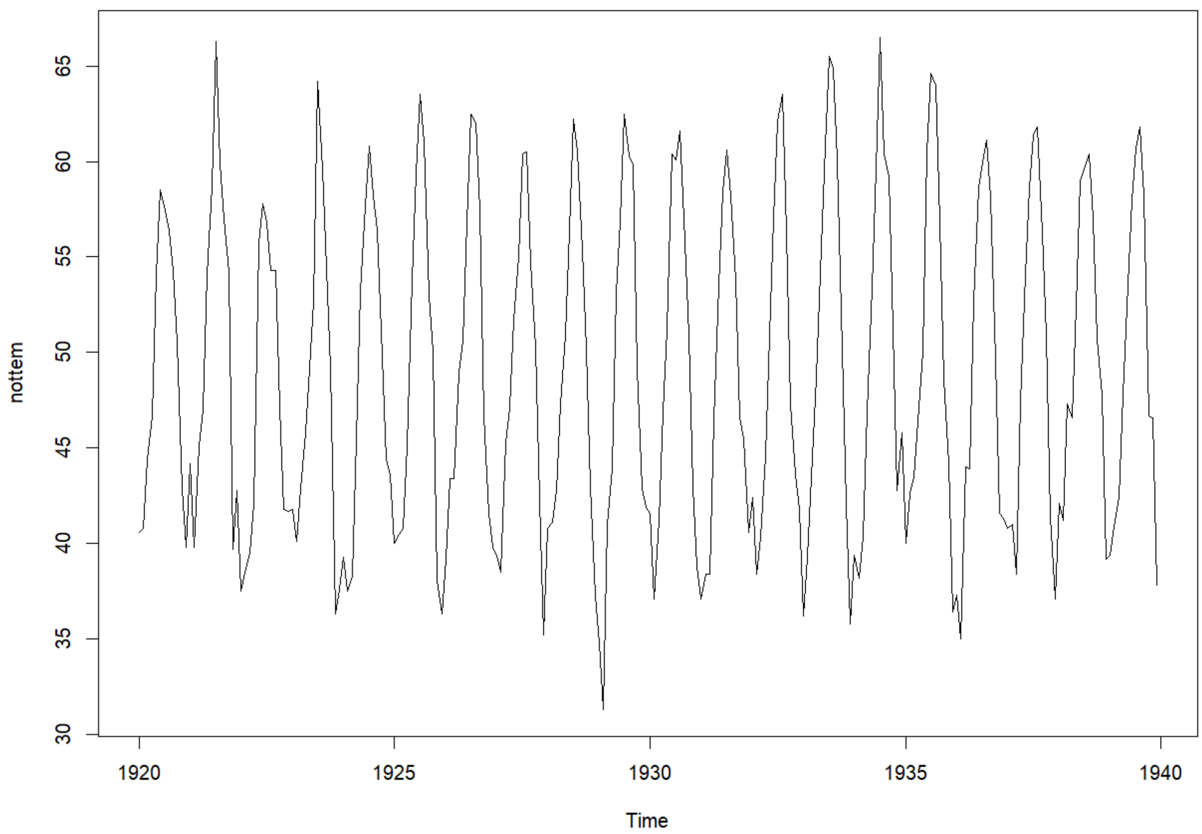
Test plot:



Diff(test) Plot:



Nottem Plot:



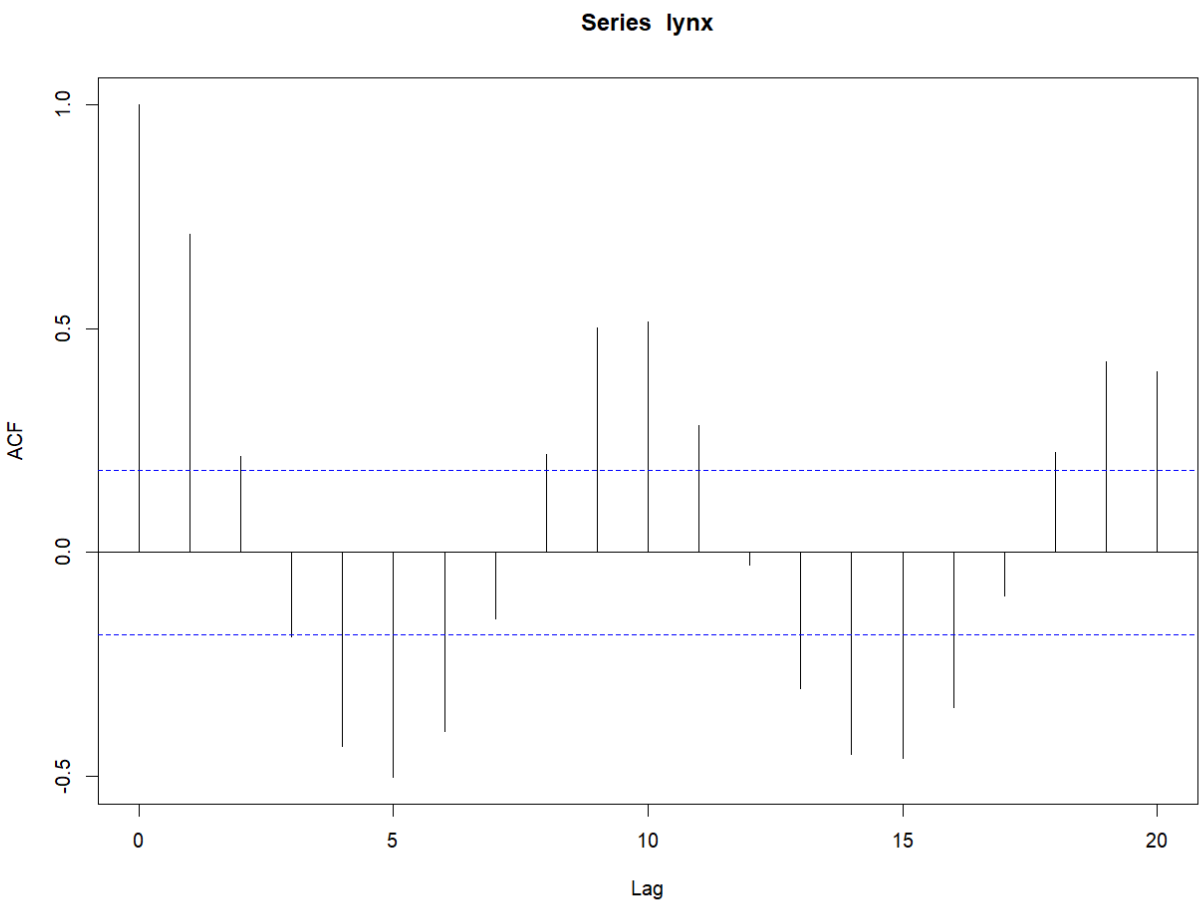
Auto-Correlation:

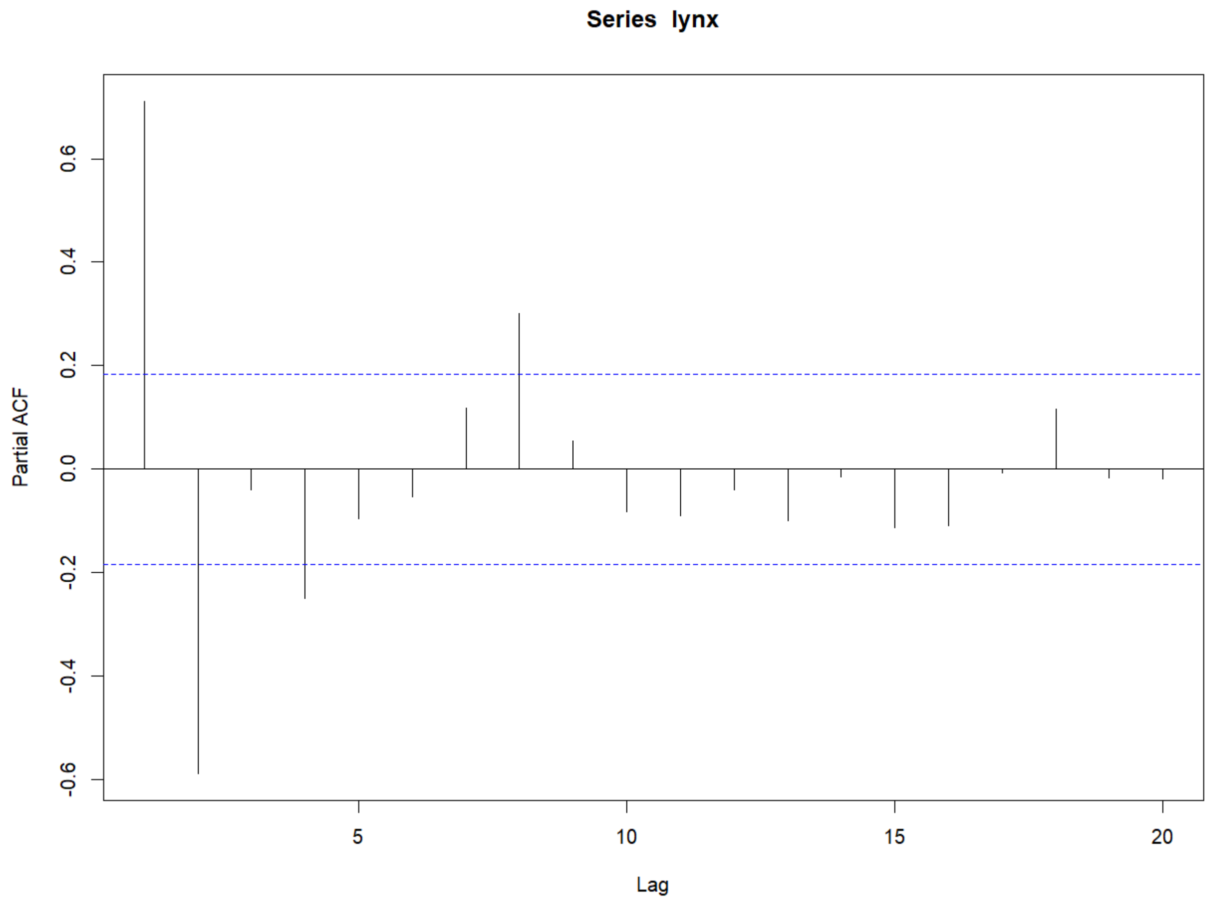
- acf() Shows the autocorrelation.
- pacf() Shows the partial autocorrelation.

Autocorrelation	Partial autocorrelation
The correlation coefficient between lags of the time series.	The correlation coefficient adjusted for shorter lags.

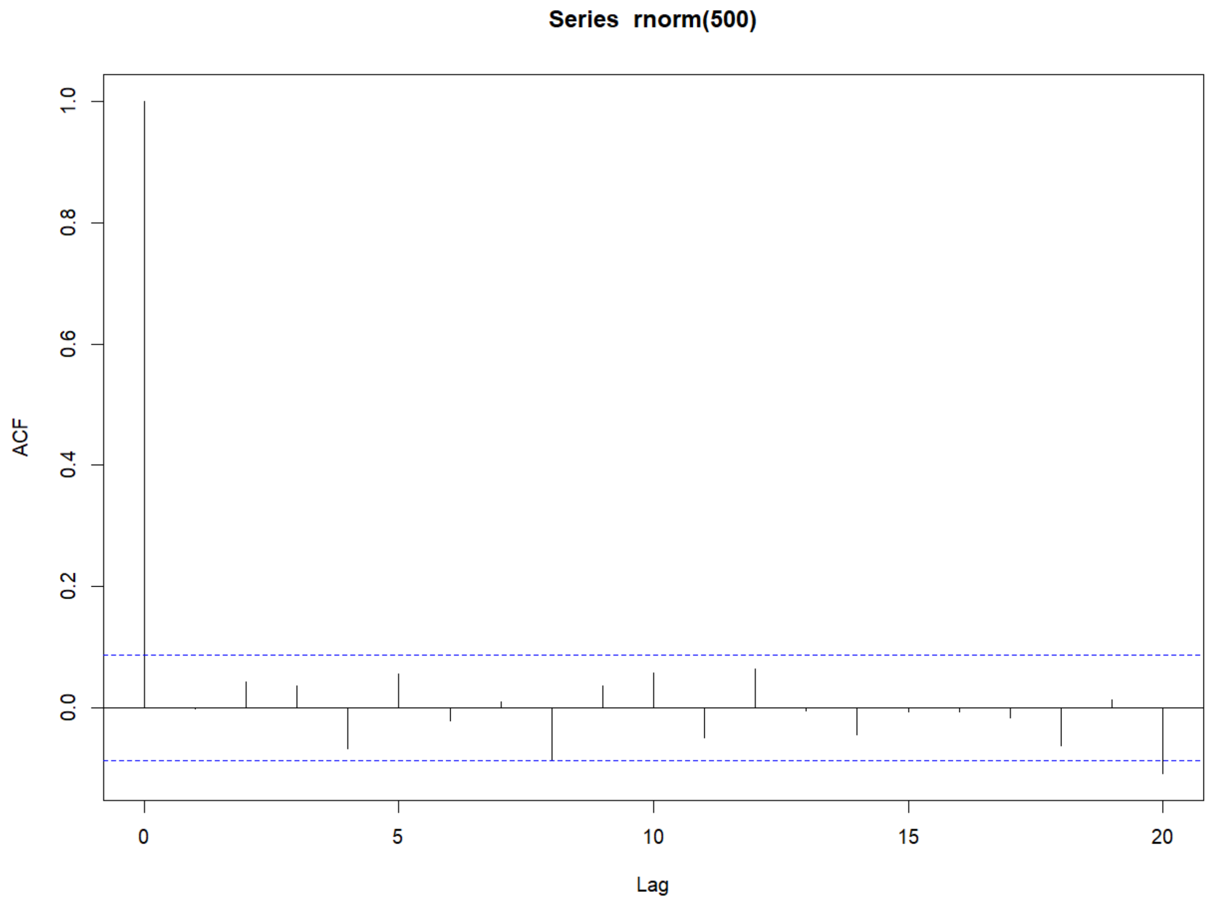
acf Vs pacf:

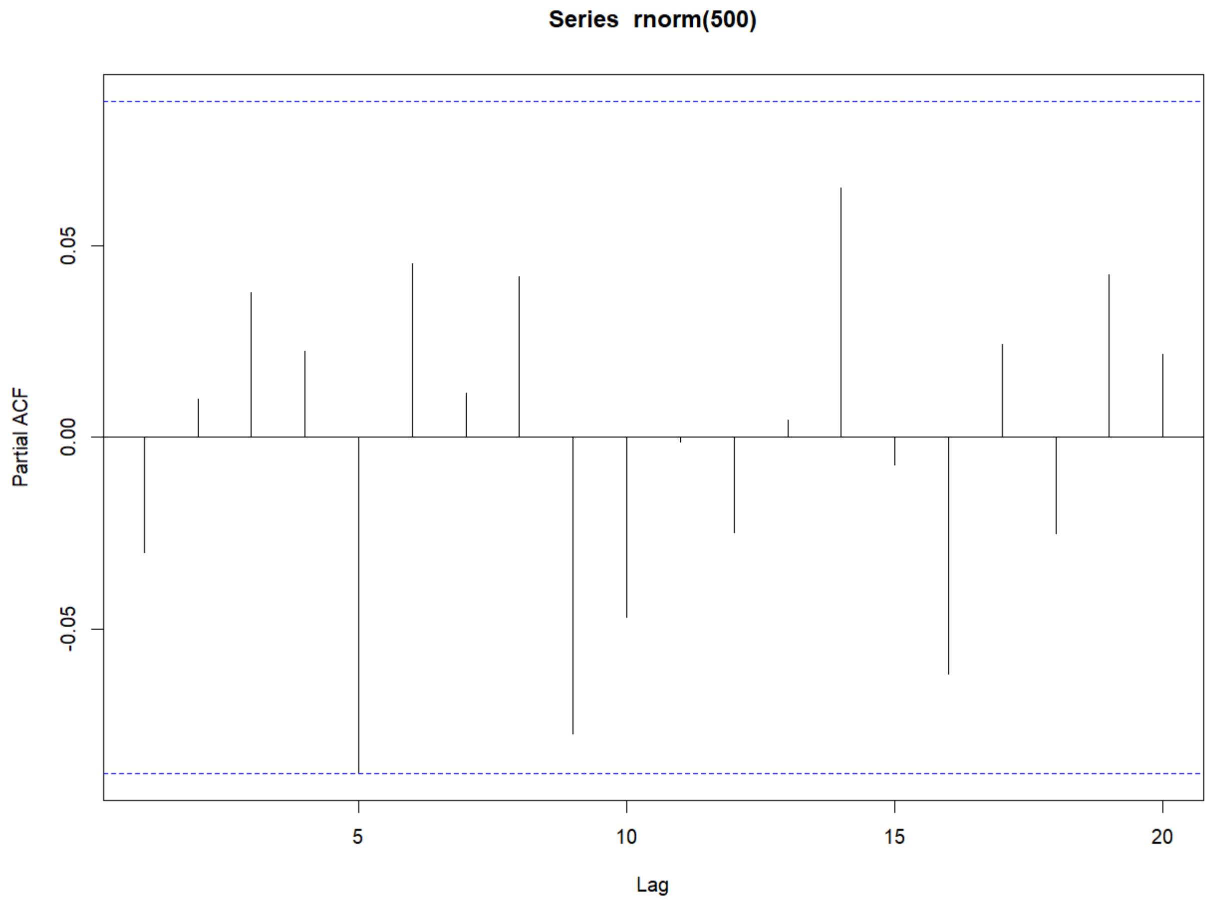
- Using lynx





- Using rnorm


Series rnorm(500)
ACF
0.0
0.2
0.4
0.6
0.8
1.0
0
5
10
15
20
Lag



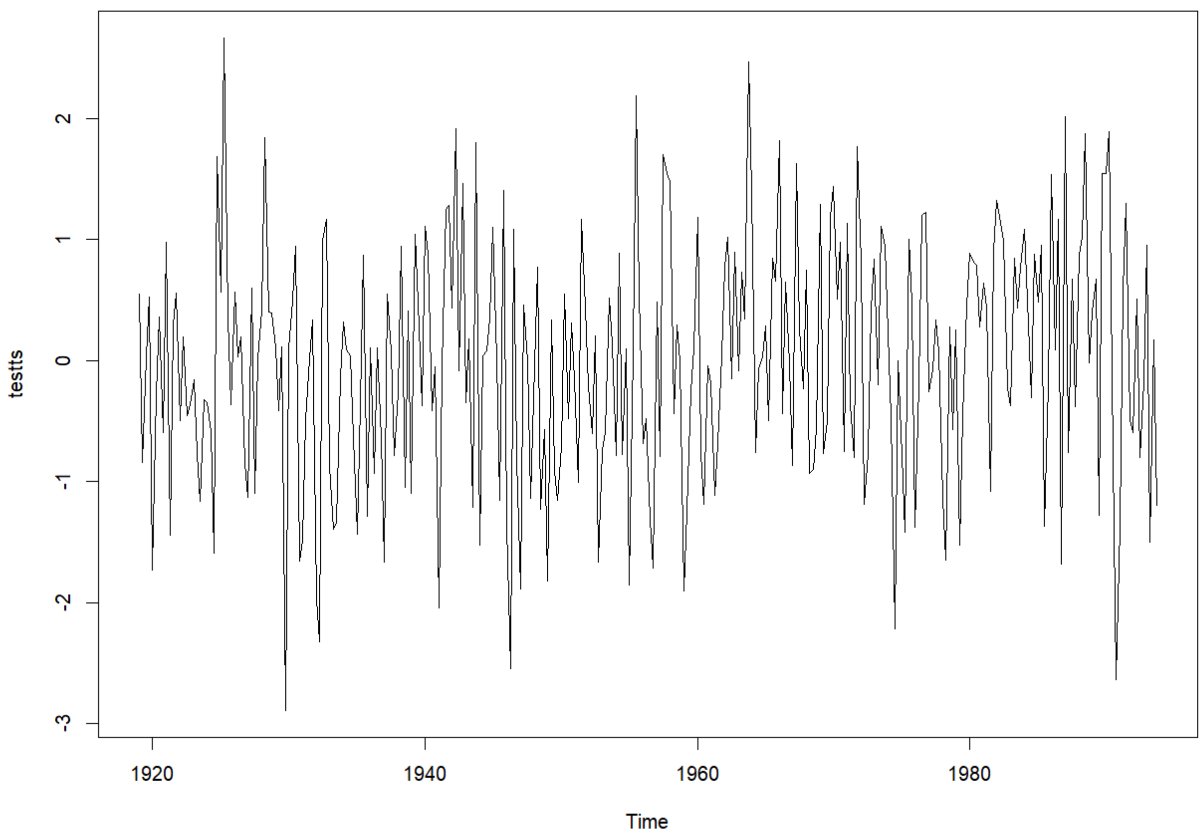
Methods with Library Forecast:

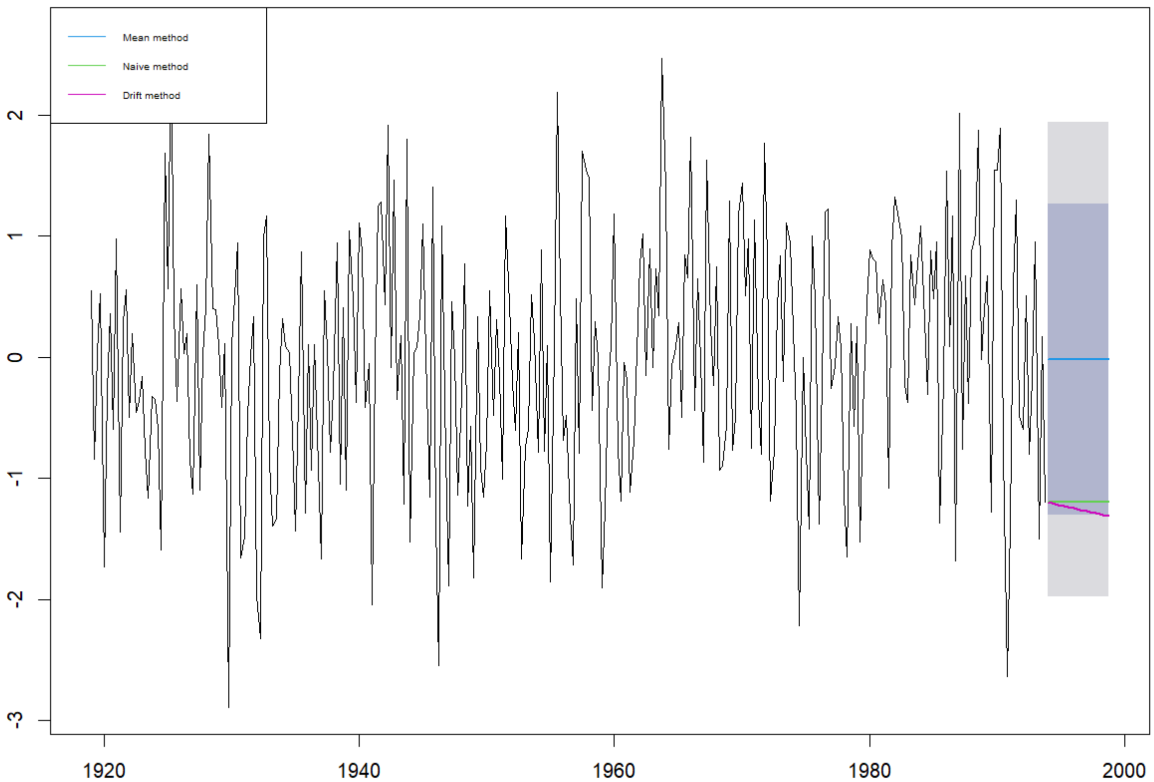
Naïve Method	Seasonal naïve method	Mean method	Drift method
Returns the last observation as forecast value – naive()	Returns the last observation of the seasonal stage – snaive()	Returns the mean as forecast value – meanf()	Carries the change over first to last observation into the future – rwf()

Console Terminal Background Jobs

R 4.2.1 · ~/

```
> set.seed(50)
> testts <- ts(rnorm(300), start = c(1919,1), frequency = 4)
> plot(testts)
> library(forecast)
> meanmodel <- meanf(testts, h = 20)
> naivemodel <- naive(testts, h = 20)
> driftmodel <- rwf(testts, h = 20, drift = T)
> plot(meanmodel, main = "")
> lines(naivemodel$mean, col = 123, lwd = 2)
> lines(driftmodel$mean, col = 22, lwd = 2)
> legend("topleft", lty = 1, cex = 0.5, col = c(4, 123, 22), legend = c("Mean method","Naive method","Drift method"))
> 
```

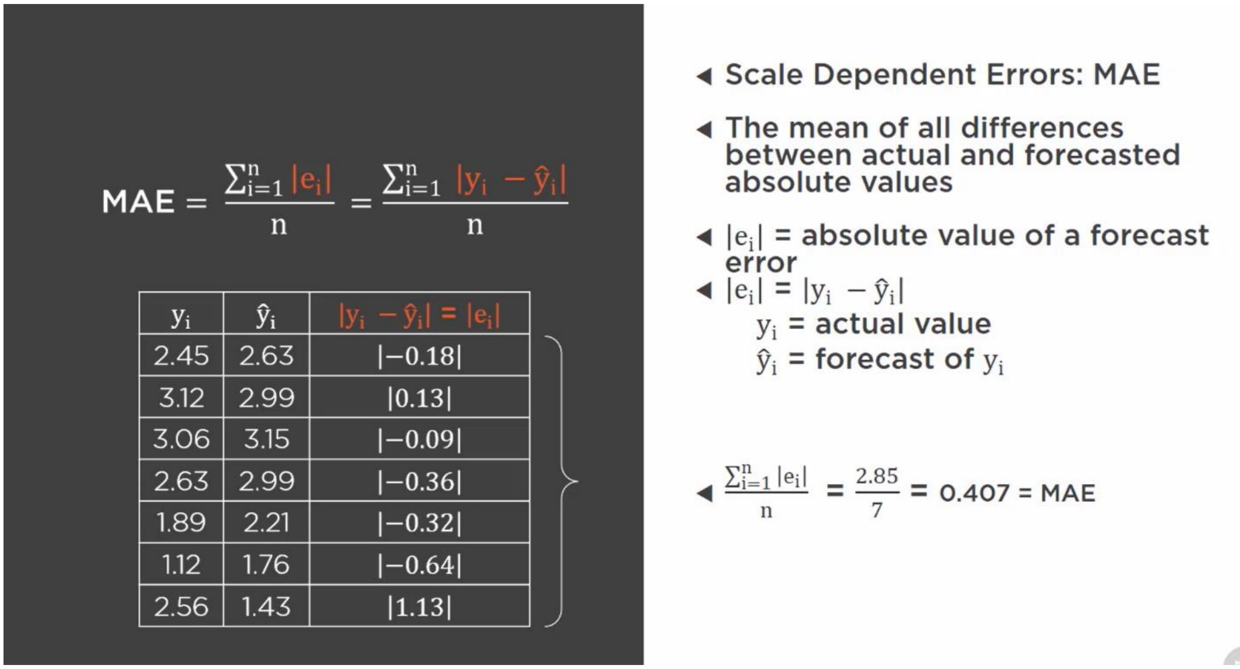




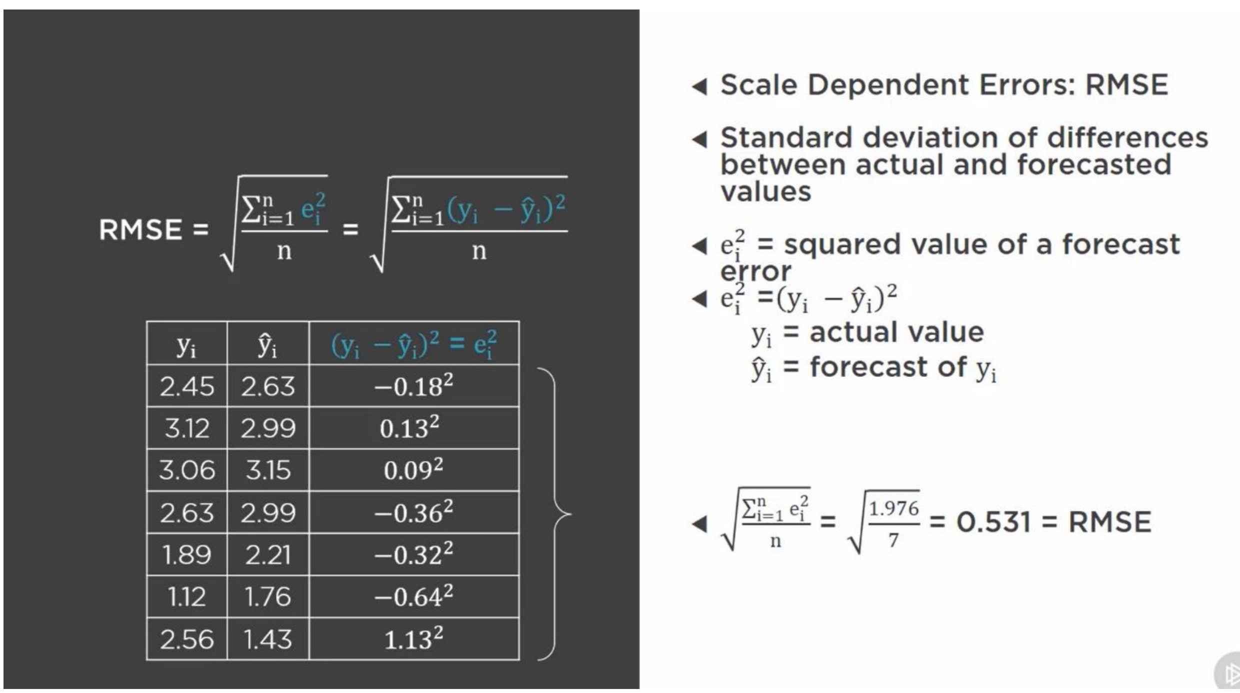
Error Indicators:

- Mean Absolute Error- MAE
- Root Mean Squared Error- RMSE
- Mean Absolute Scale Error- MASE
- Mean Absolute Percentage Error- MAPE

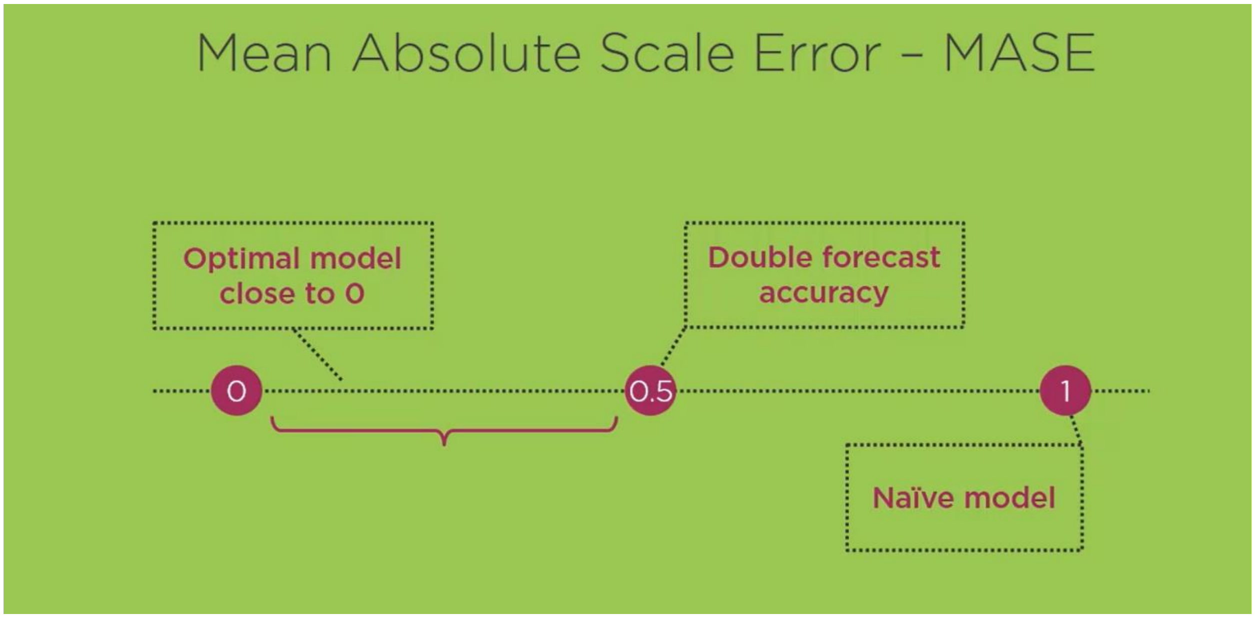
MAE:


$MAE = \frac{\sum_{i=1}^{n} |e_i|}{n} = \frac{\sum_{i=1}^{n} |y_i - \hat{y}_i|}{n}$
y_i | $\hat{y}_i$ | $|y_i - \hat{y}_i| = |e_i|$
2.45 | 2.63 | $|-0.18|$
3.12 | 2.99 | $|0.13|$
3.06 | 3.15 | $|-0.09|$
2.63 | 2.99 | $|-0.36|$
1.89 | 2.21 | $|-0.32|$
1.12 | 1.76 | $|-0.64|$
2.56 | 1.43 | $|1.13|$
Scale Dependent Errors: MAE
The mean of all differences between actual and forecasted absolute values
$|e_i|$ = absolute value of a forecast error
$|e_i| = |y_i - \hat{y}_i|$
y_i = actual value
$\hat{y}_i$ = forecast of y_i
$\frac{\sum_{i=1}^{n} |e_i|}{n} = \frac{2.85}{7}$ = 0.407 = MAE

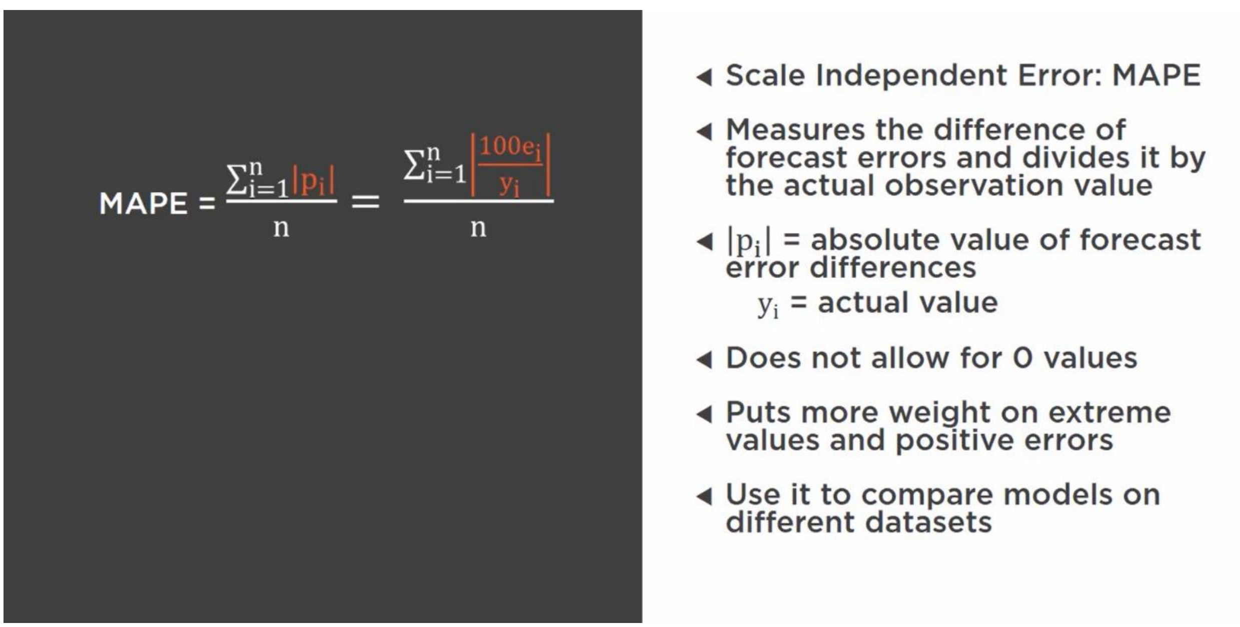
RMSE:


$RMSE = \sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$
y_i | $\hat{y}_i$ | $(y_i - \hat{y}_i)^2 = e_i^2$
2.45 | 2.63 | -0.18^2
3.12 | 2.99 | 0.13^2
3.06 | 3.15 | 0.09^2
2.63 | 2.99 | -0.36^2
1.89 | 2.21 | -0.32^2
1.12 | 1.76 | -0.64^2
2.56 | 1.43 | 1.13^2
Scale Dependent Errors: RMSE
Standard deviation of differences between actual and forecasted values
e_i^2 = squared value of a forecast error
$e_i^2 = (y_i - \hat{y}_i)^2$
y_i = actual value
$\hat{y}_i$ = forecast of y_i
$\sqrt{\frac{\sum_{i=1}^{n} e_i^2}{n}} = \sqrt{\frac{1.976}{7}}$ = 0.531 = RMSE

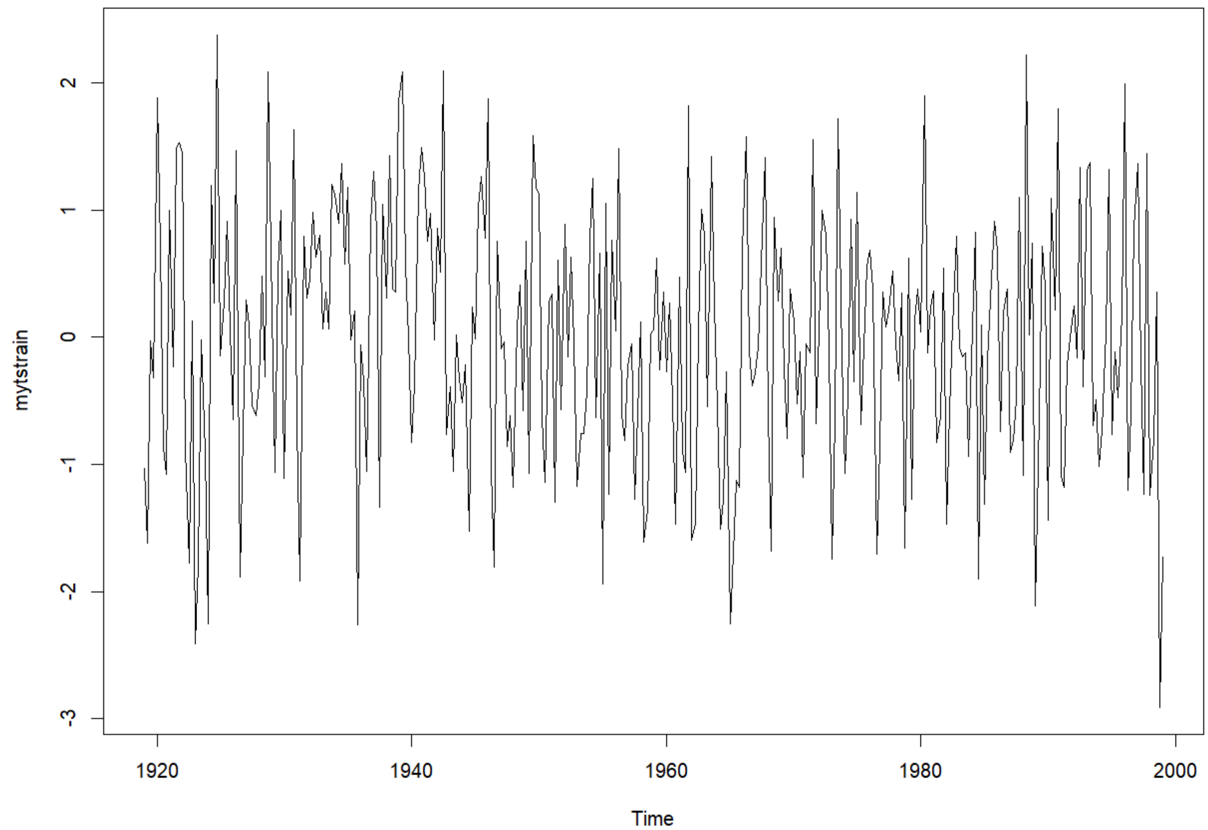
MASE:

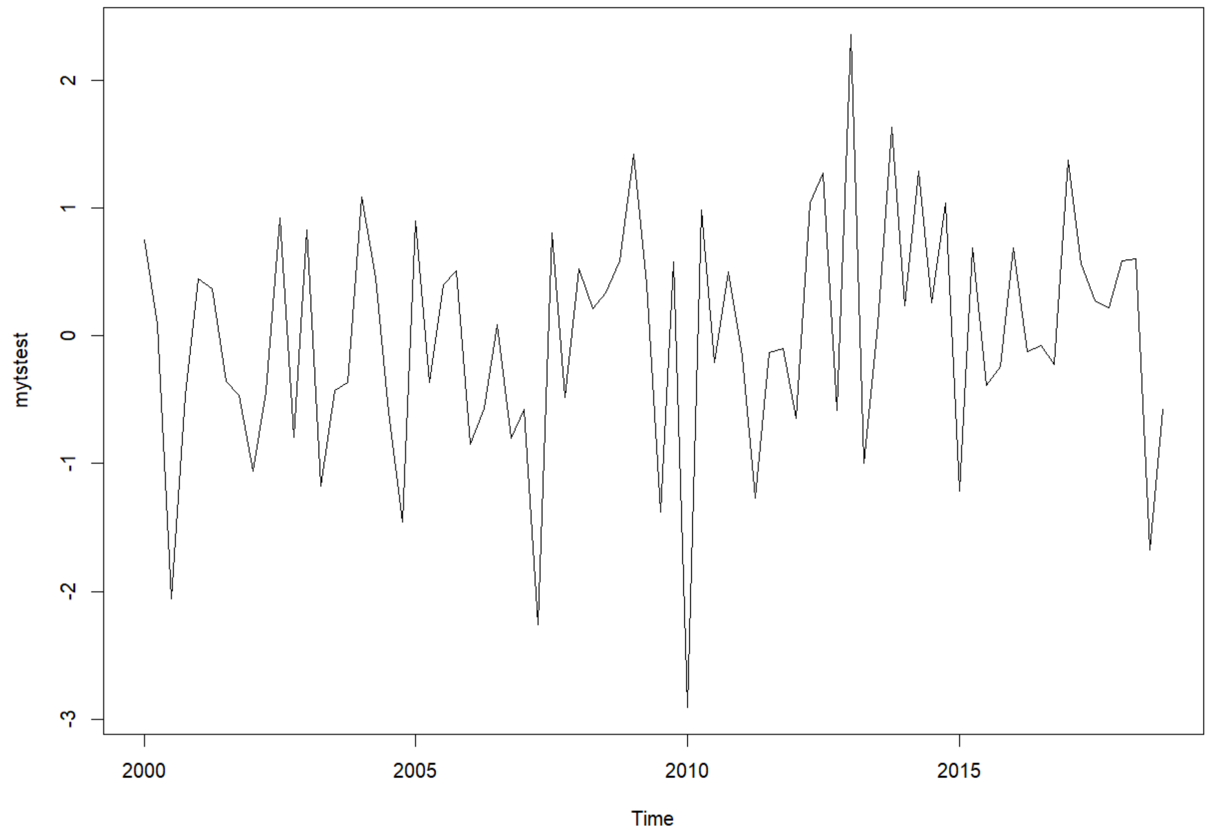

Mean Absolute Scale Error – MASE
Optimal model close to 0
Double forecast accuracy
0
0.5
1
Naïve model

MAPE:


$\text{MAPE} = \frac{\sum_{i=1}^{n} |p_i|}{n} = \frac{\sum_{i=1}^{n} \left|\frac{100e_i}{y_i}\right|}{n}$
Scale Independent Error: MAPE
Measures the difference of forecast errors and divides it by the actual observation value
$|p_i|$ = absolute value of forecast error differences
y_i = actual value
Does not allow for 0 values
Puts more weight on extreme values and positive errors
Use it to compare models on different datasets

Comparison:





Console Terminal Background Jobs

R 4.2.1 · ~/

```
> accuracy(meanmodel, mytstest)
                        ME      RMSE       MAE       MPE      MAPE      MASE        ACF1 Theil's U
Training set  4.146651e-18 0.9922591 0.7999341 103.27287 103.27287 0.7247526  0.06882037        NA
Test set     -5.947488e-05 0.9250726 0.7350389  99.97963  99.97963 0.6659566 -0.15127726 0.9658077
> accuracy(naivemodel, mytstest)
                       ME     RMSE      MAE       MPE     MAPE      MASE       ACF1 Theil's U
Training set -0.002185465 1.351649 1.086030 -32.96660 564.0232 0.9839598 -0.4830987        NA
Test set      1.714805941 1.948414 1.768427  97.41156 466.8504 1.6022227 -0.1512773  1.675834
> accuracy(driftmodel, mytstest)
                        ME     RMSE      MAE       MPE     MAPE      MASE       ACF1 Theil's U
Training set -1.014813e-17 1.351647 1.086125 -32.43850 563.4943 0.9840464 -0.4830987        NA
Test set      1.805503e+00 2.032968 1.854351  95.38722 492.5072 1.6800705 -0.1268056  1.767978
> 
```

Residuals:

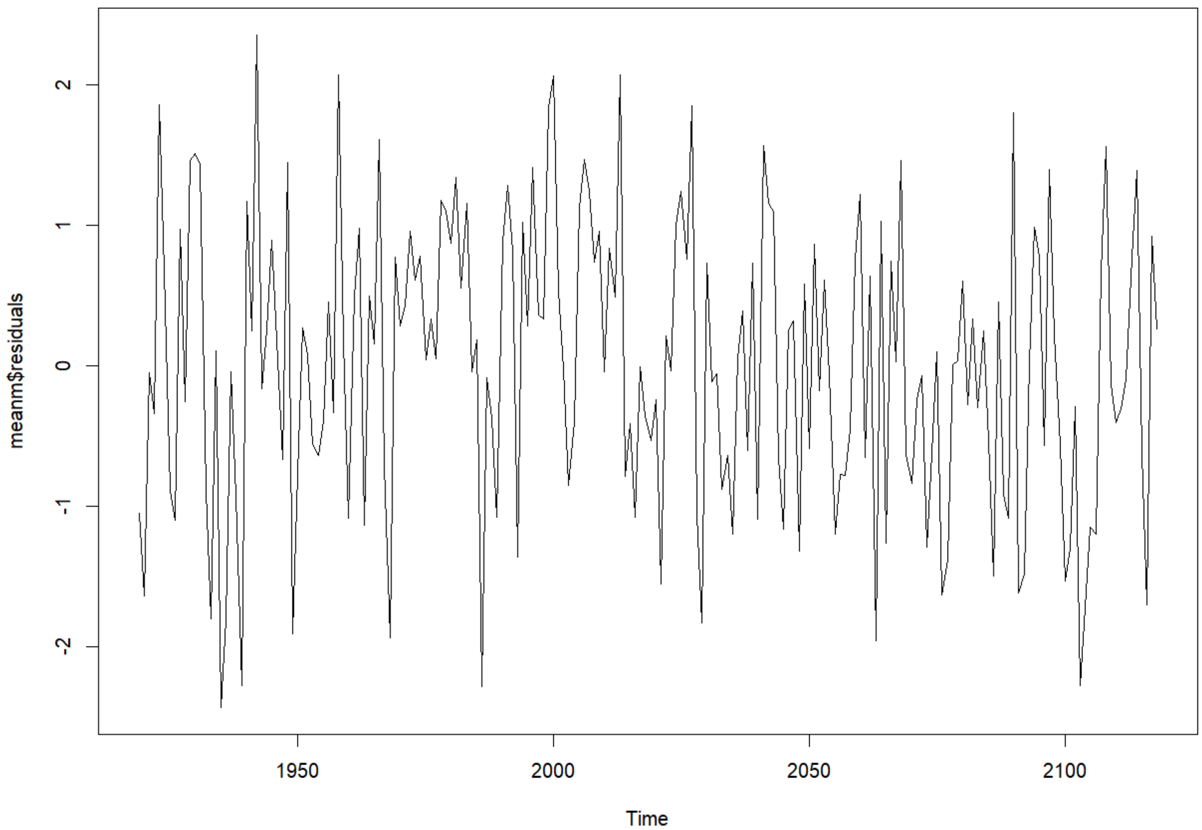
Ideal Model

Zero mean	Constant variance	Correlated residuals	Normal distribution
Fix: addition or subtraction	Fix: transformation – not always possible.	Fix: differencing	Fix: transformation – not always possible

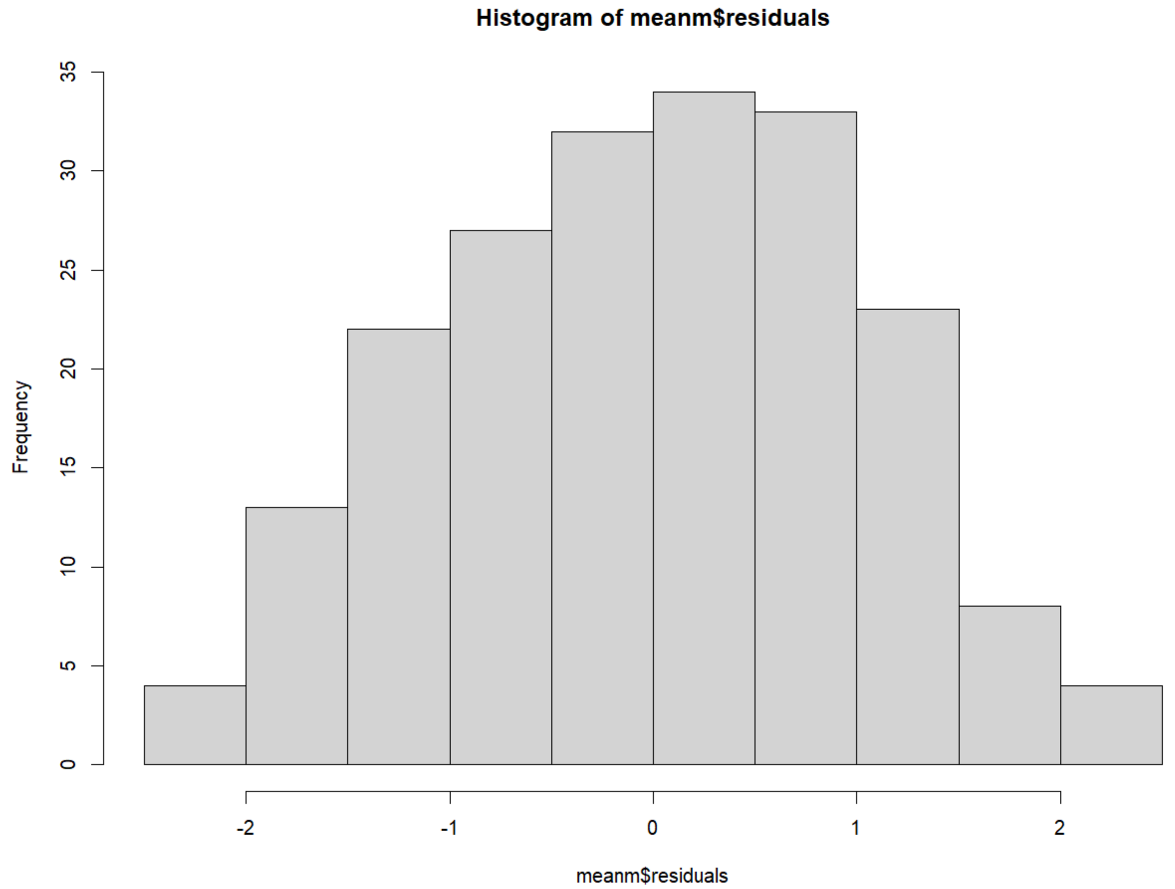
Console Terminal Background Jobs

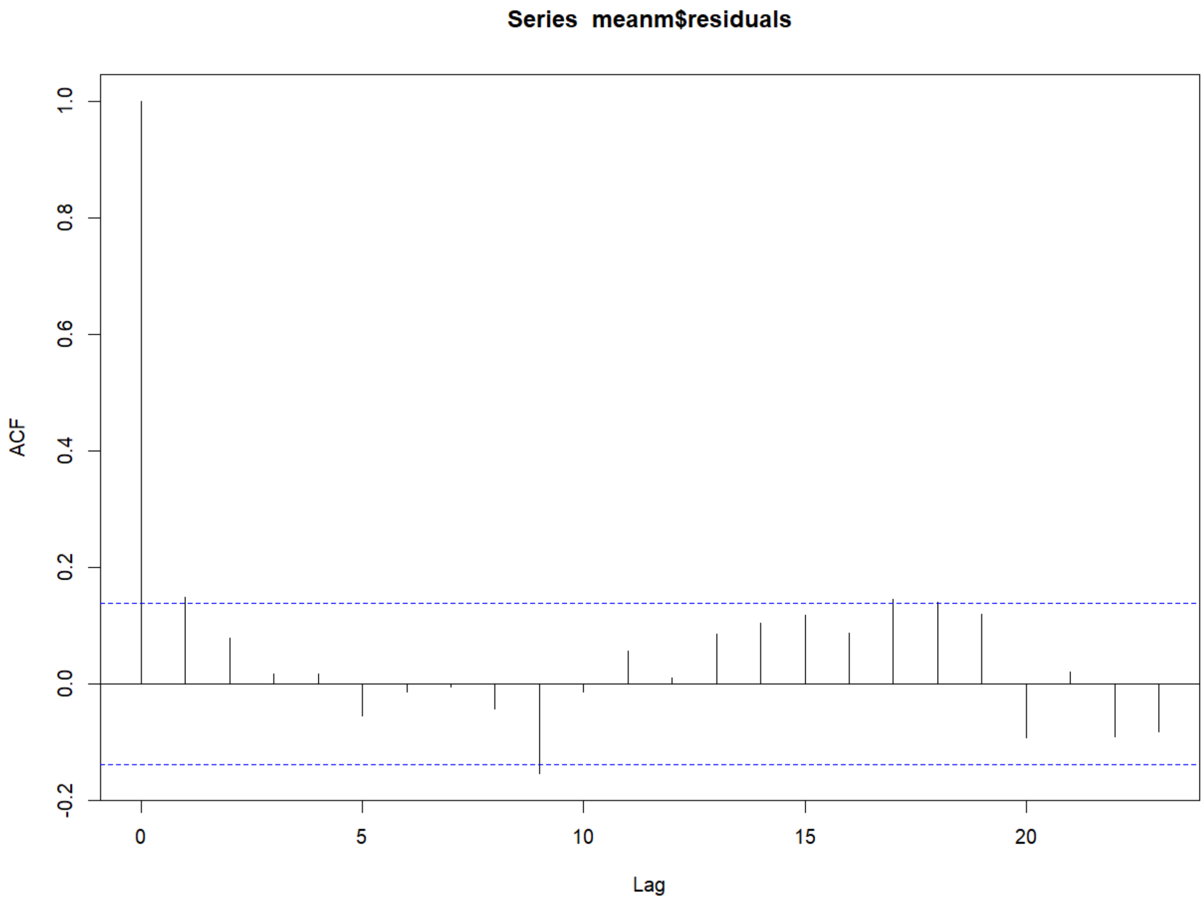
R 4.2.1 · ~/

```
> #Data set
> set.seed(95)
> myts <- ts(rnorm(200), start = (1919))
> #Setting up simple models
> library(forecast)
> meanm <- meanf(myts, h = 20)
> naivem <- naive(myts, h = 20)
> driftm <- rwf(myts, h = 20, drift = T)
> #Variance and mean of the mean model
> var(meanm$residuals)
[1] 1.053807
> plot(meanm$residuals)
> mean(meanm$residuals)
[1] -5.95498e-18
> #Deleting the NA at the front of the vector
> naivwithoutNA <- naivem$residuals
> naivwithoutNA <- naivwithoutNA[2:200]
> var(naivwithoutNA)
[1] 1.798592
> mean(naivwithoutNA)
[1] 0.006605028
> driftwithoutNA <- driftm$residuals
> driftwithoutNA <- driftwithoutNA[2:200]
> var(driftwithoutNA)
[1] 1.798592
> mean(driftwithoutNA)
[1] -4.502054e-17
> # Histogram of distribution
> hist(meanm$residuals)
> #Autocorrelation
> acf(meanm$residuals)
> 
```



2
1
0
-1
-2
meanm$residuals
1950
2000
2050
2100
Time


Histogram of meanm$residuals
Frequency
meanm$residuals
0
5
10
15
20
25
30
35
-2
-1
0
1
2



The Mean Model on Random Data:

Zero mean	Normal distribution	Equal variance	No autocorrelation
mean()	hist()	var() and plot()	acf()

ARIMA:

ARIMA => Univariate => Non-seasonal

AR	• Autoregressive term – "p"
I	• Integration / differencing – "d"
MA	• Moving average – "q"

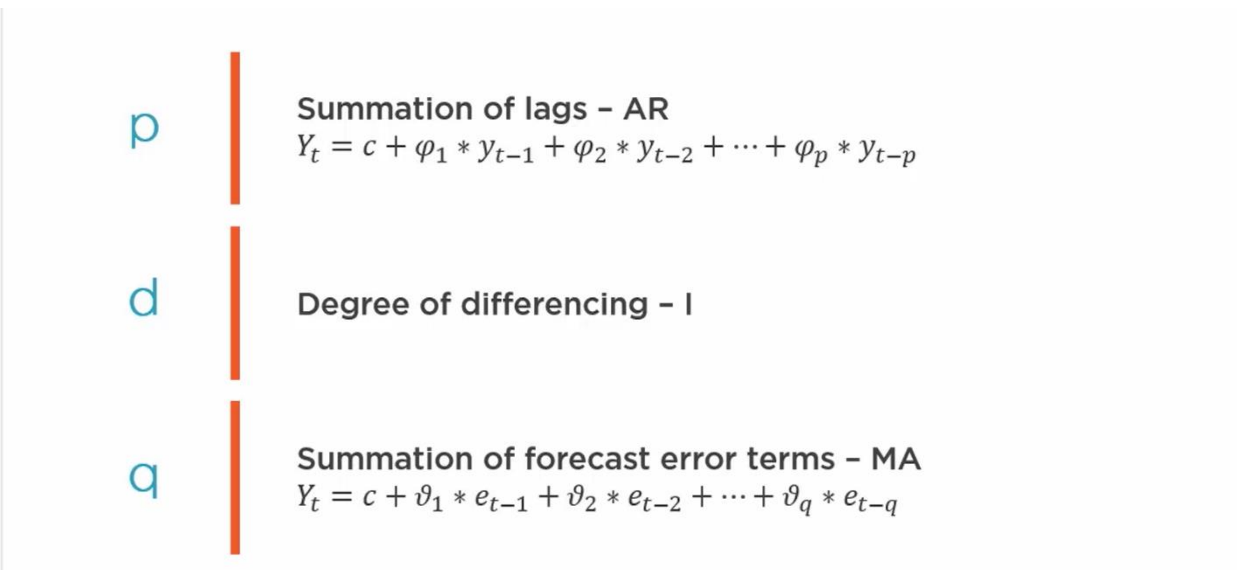
Stationarity:

ARIMA(p, d, q)	ARMA(p, d)
Non-stationary time series gets differenced (“d”) before “p” and “q” get specified.	With stationary time series the autoregressive (“p”) and moving average (“q”) terms get ordered without differencing.

Variations of the Model:

AR(1) – ARIMA(1,0,0)	MA(1) – ARIMA (0,0,1)
Autoregressive model (“p” only)	Moving average model (“q” only)

What do the Parameters do?



How to Calculate an AR Model

Coefficients:

	ar1	mean
	1.1246	1547.3859
s.e.	0.0903	136.8501

$$Y_t = c + \varphi_1 * y_{t-1}$$

Coefficients:

	ar1	ar2	ar3	ar4	mean
	1.1246	-0.7174	0.2634	-0.2543	1547.3859
s.e.	0.0903	0.1367	0.1361	0.0897	136.8501

$$Y_t = c + \varphi_1 * y_{t-1} + \varphi_2 * y_{t-2} + \\ + \varphi_3 * y_{t-3} + \varphi_4 * y_{t-4}$$

How to Calculate an ARMA Model

Coefficients:

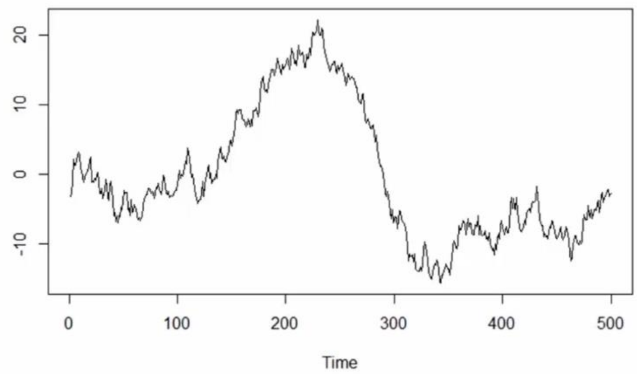
	ar1	ma1	mean
	1.3421	-0.2027	1544.4039
s.e.	0.0984	0.1261	131.9242

$$Y_t = c + \varphi_1 * y_{t-1} + \vartheta_1 * e_{t-1}$$

ARIMA(0, 1, 0)

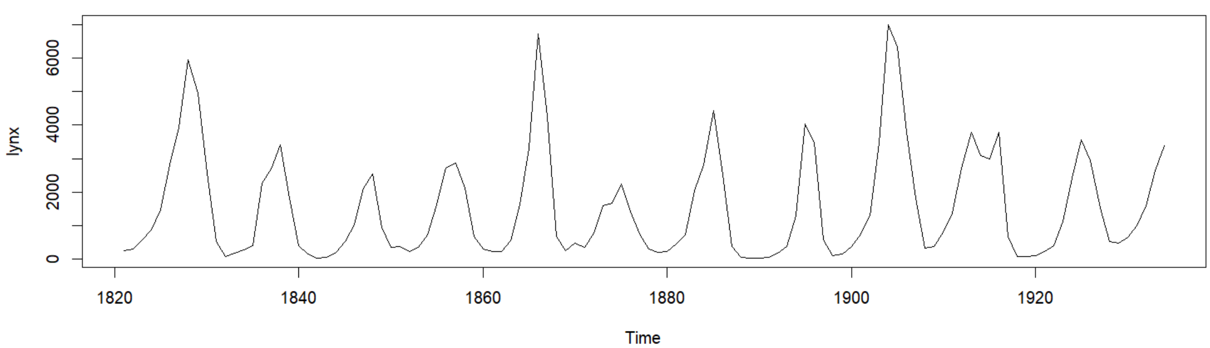
Drift: $c = Y_t - Y_{t-1}$

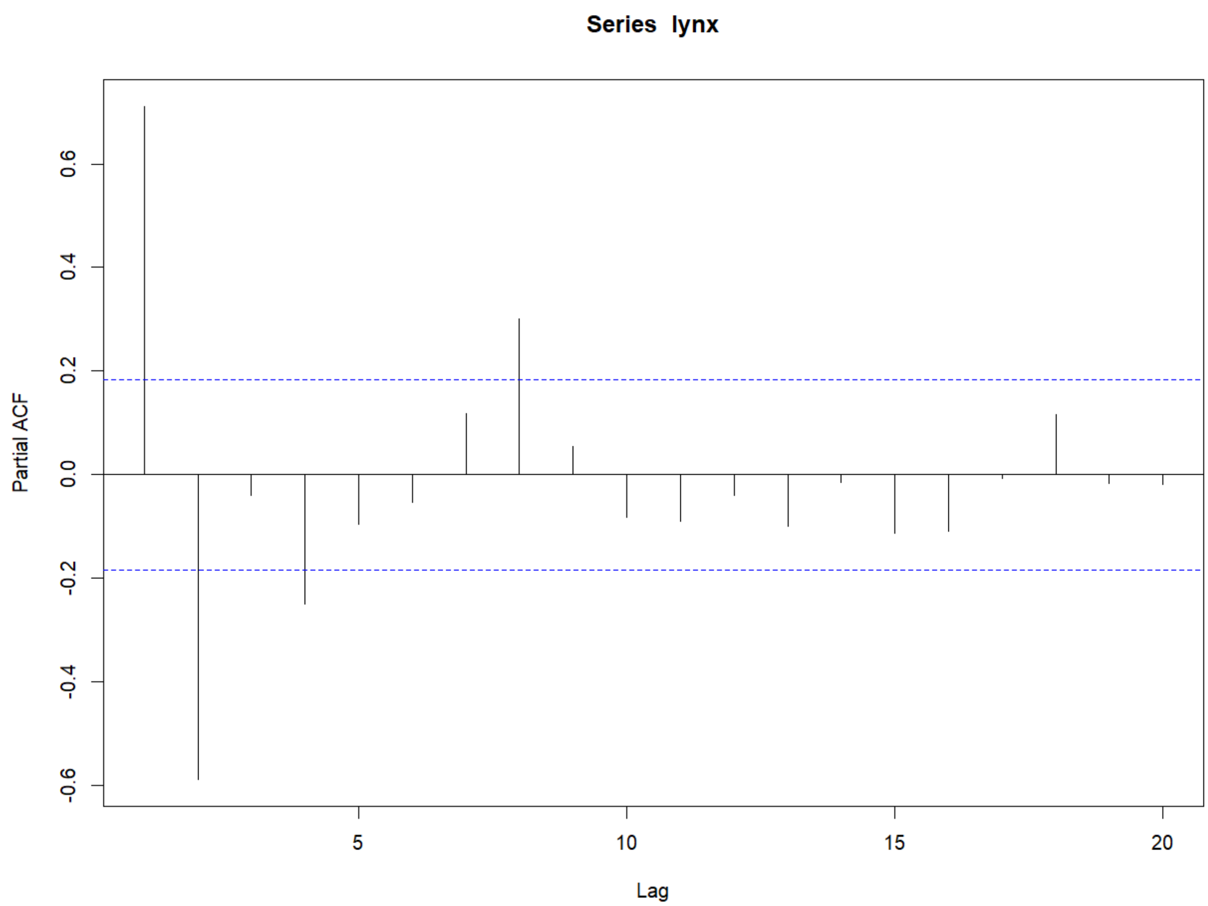
No drift: $Y_t = Y_{t-1}$



ARIMA Model Practice:

- p	Autocorrelation is clear
- d	It might be stationary
- q	There might be forecasting errors

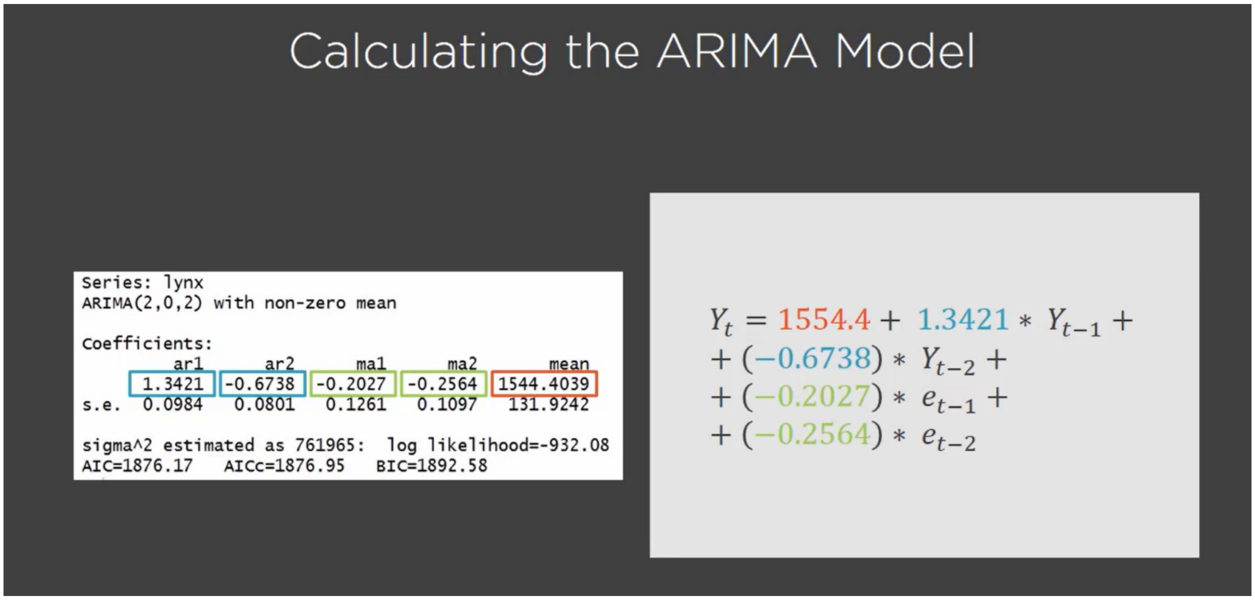


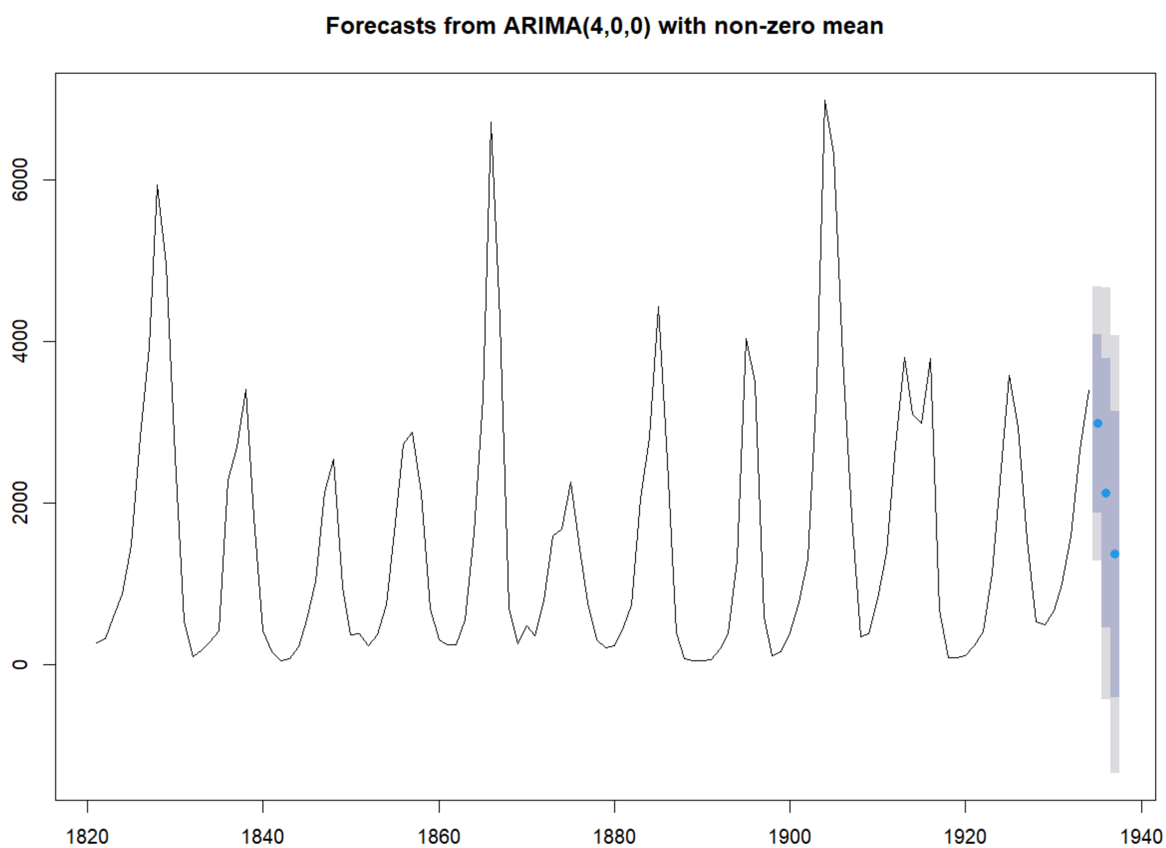

Series lynx
Partial ACF
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
5
10
15
20
Lag

Console | Terminal | Background Jobs

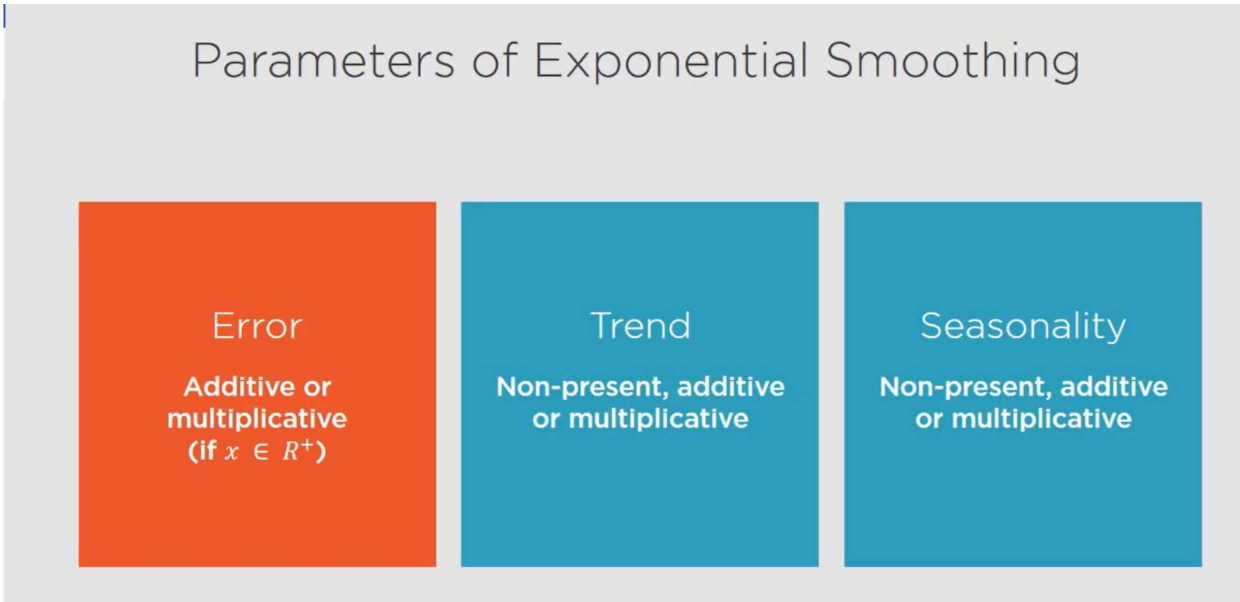
R 4.2.1 · ~/

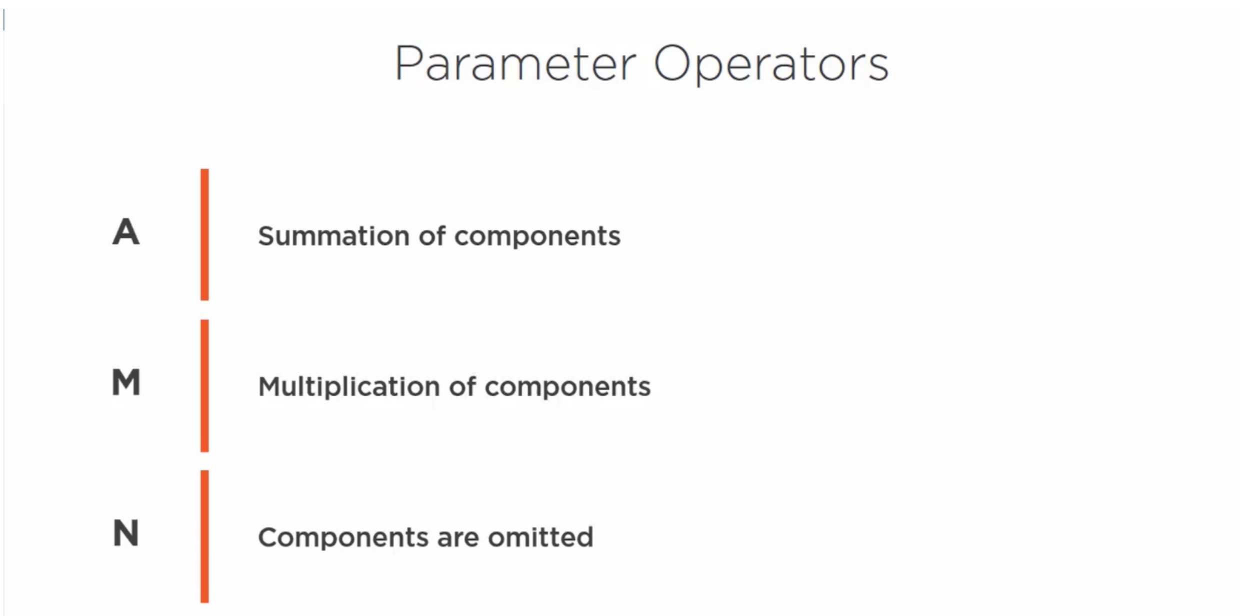
```
> plot(lynx)
> acf(lynx) ; pacf(lynx)
> auto.arima(lynx)
Series: lynx
ARIMA(2,0,2) with non-zero mean

Coefficients:
         ar1      ar2      ma1      ma2       mean
      1.3421  -0.6738  -0.2027  -0.2564  1544.4039
s.e.  0.0984   0.0801   0.1261   0.1097   131.9242

sigma^2 = 761965:  log likelihood = -932.08
AIC=1876.17   AICc=1876.95   BIC=1892.58
> auto.arima(lynx, trace = T)

 ARIMA(2,0,2) with non-zero mean : 1876.952
 ARIMA(0,0,0) with non-zero mean : 2006.724
 ARIMA(1,0,0) with non-zero mean : 1927.209
 ARIMA(0,0,1) with non-zero mean : 1918.165
 ARIMA(0,0,0) with zero mean     : 2080.721
 ARIMA(1,0,2) with non-zero mean : 1888.757
 ARIMA(2,0,1) with non-zero mean : 1880.014
 ARIMA(3,0,2) with non-zero mean : 1878.603
 ARIMA(2,0,3) with non-zero mean : Inf
 ARIMA(1,0,1) with non-zero mean : 1891.442
 ARIMA(1,0,3) with non-zero mean : 1890.03
 ARIMA(3,0,1) with non-zero mean : 1881.962
 ARIMA(3,0,3) with non-zero mean : Inf
 ARIMA(2,0,2) with zero mean     : 1905.595

 Best model: ARIMA(2,0,2) with non-zero mean

Series: lynx
ARIMA(2,0,2) with non-zero mean

Coefficients:
         ar1      ar2      ma1      ma2       mean
      1.3421  -0.6738  -0.2027  -0.2564  1544.4039
s.e.  0.0984   0.0801   0.1261   0.1097   131.9242

sigma^2 = 761965:  log likelihood = -932.08
AIC=1876.17   AICc=1876.95   BIC=1892.58
> myar = auto.arima(lynx, stepwise = F, approximation = F)
> myar
Series: lynx
ARIMA(4,0,0) with non-zero mean

Coefficients:
         ar1      ar2     ar3      ar4       mean
      1.1246  -0.7174  0.2634  -0.2543  1547.3859
s.e.  0.0903   0.1367  0.1361   0.0897   136.8501

sigma^2 = 748457:  log likelihood = -931.11
AIC=1874.22   AICc=1875.01   BIC=1890.64
> plot(forecast(myar, h = 3))
> 
```


Calculating the ARIMA Model
Series: lynx
ARIMA(2,0,2) with non-zero mean
Coefficients:
ar1 ar2 ma1 ma2 mean
1.3421 -0.6738 -0.2027 -0.2564 1544.4039
s.e. 0.0984 0.0801 0.1261 0.1097 131.9242
sigma^2 estimated as 761965: log likelihood=-932.08
AIC=1876.17 AICc=1876.95 BIC=1892.58
$Y_t = 1554.4 + 1.3421 * Y_{t-1} +$
$+ (-0.6738) * Y_{t-2} +$
$+ (-0.2027) * e_{t-1} +$
$+ (-0.2564) * e_{t-2}$


Forecasts from ARIMA(4,0,0) with non-zero mean
6000
4000
2000
0
1820
1840
1860
1880
1900
1920
1940

Exponential Smoothing





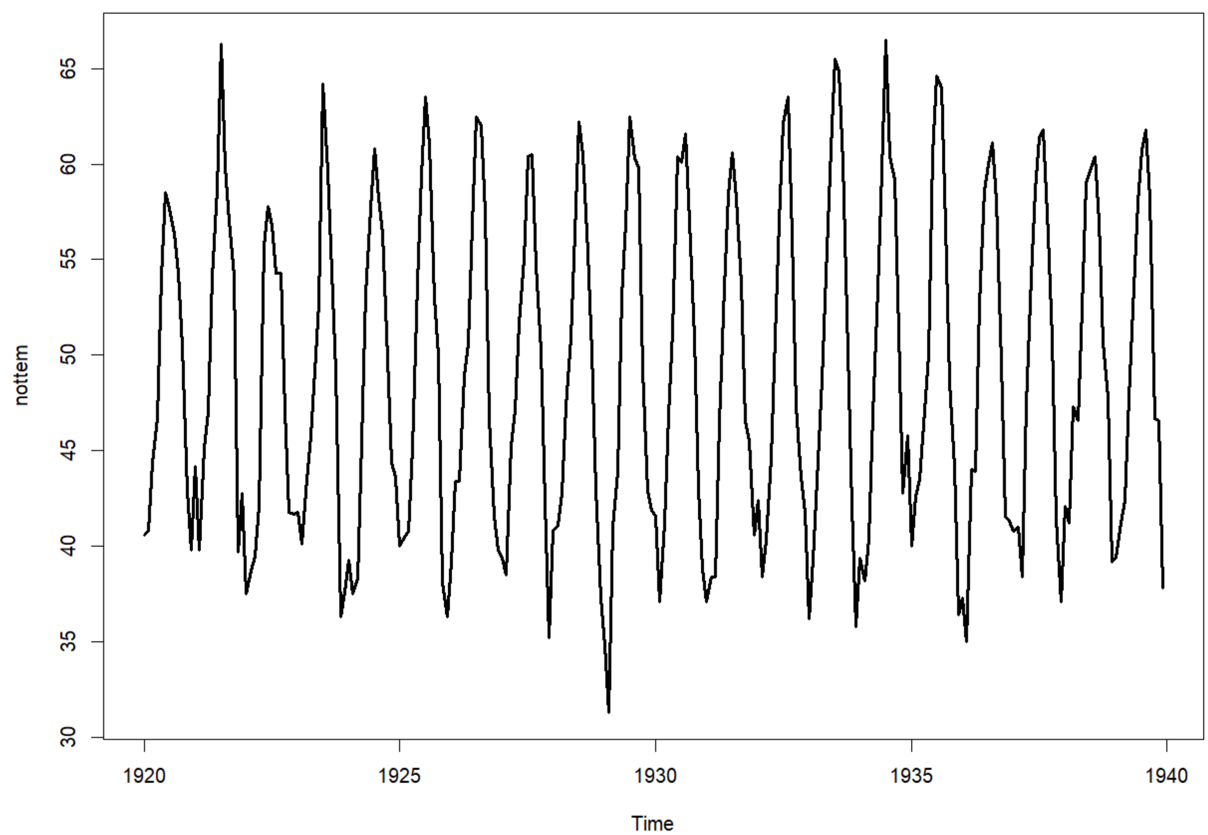
<u>Exponential Smoothing Function:</u>

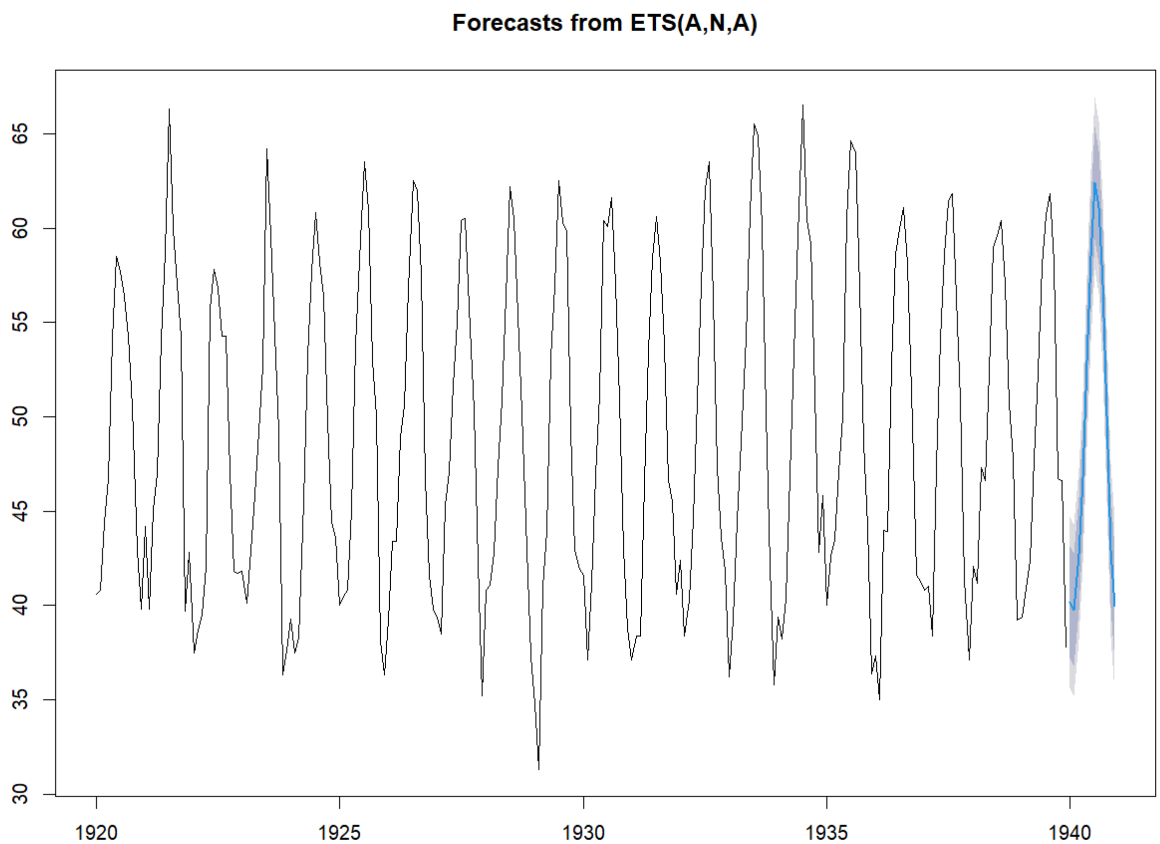
Function ses() Simple exponential smoothing	Function holt() Trend methods
Function hw() Holt-Winter seasonal method	Function ets() Selects the optimal model

ETS:

The ‘model=’ Argument of ‘ets()’

- Z Auto Selection
- A Additive model
- M Multiplicative model
- N Non-present (except error)

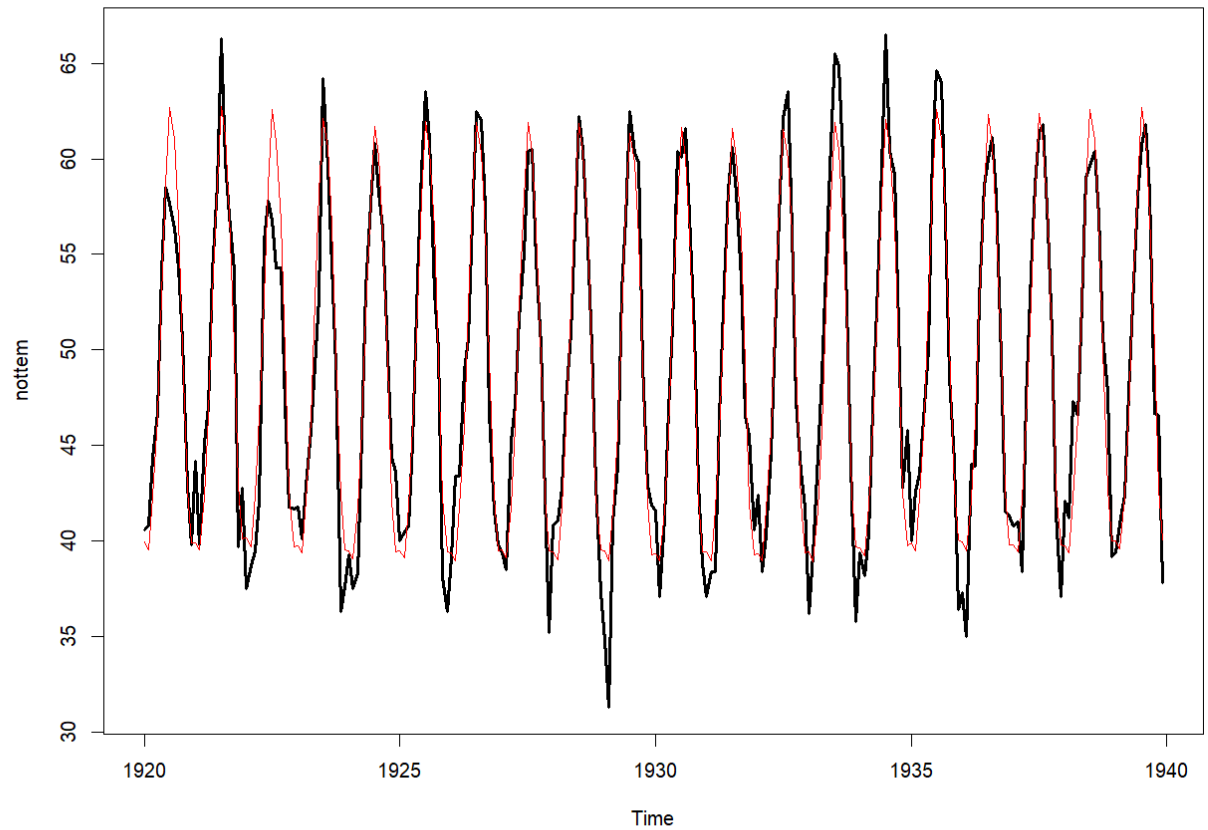



Forecasts from ETS(A,N,A)
30
35
40
45
50
55
60
65
1920
1925
1930
1935
1940

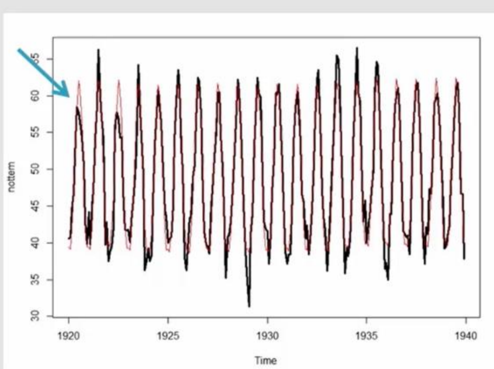
Console Terminal Background Jobs

R 4.2.1 · ~/

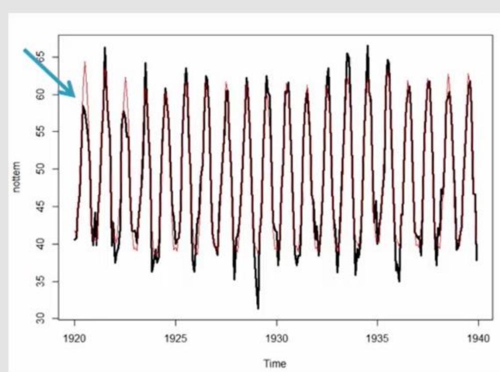
```
> # ets
> library(forecast)
> #using function ets
> etsmodel = ets(nottem) ; etsmodel
ETS(A,N,A)

Call:
 ets(y = nottem)

  Smoothing parameters:
    alpha = 0.0392
    gamma = 1e-04

  Initial states:
    l = 49.4597
    s = -9.5635 -6.6186 0.5447 7.4811 11.5783 12.8567
           8.9762 3.4198 -2.7516 -6.8093 -9.7583 -9.3556

  sigma:  2.3203

     AIC     AICc      BIC
1734.944 1737.087 1787.154
> #Plotting the model vs original
> plot(nottem, lwd = 3)
> lines(etsmodel$fitted, col = "red")
> #Plotting the forecast
> plot(forecast(etsmodel, h = 12))
> #Changing the prediction interval
> plot(forecast(etsmodel, h = 12, level = 95))
> #Menually setting the ets model
> etsmodmult = ets(nottem, model = "MZM")
> etsmodmult
ETS(M,N,M)

Call:
 ets(y = nottem, model = "MZM")

  Smoothing parameters:
    alpha = 0.0214
    gamma = 1e-04

  Initial states:
    l = 49.3793
    s = 0.8089 0.8647 1.0132 1.1523 1.2348 1.2666
           1.1852 1.0684 0.9405 0.8561 0.8005 0.8088

  sigma:  0.0508

     AIC     AICc      BIC
1761.911 1764.054 1814.121
> #Plot as comparison
> plot(nottem, lwd = 3)
> lines(etsmodmult$fitted, col = "red")
> 
```



Comparing the Models



Model 'ANA'



Model 'MNM'