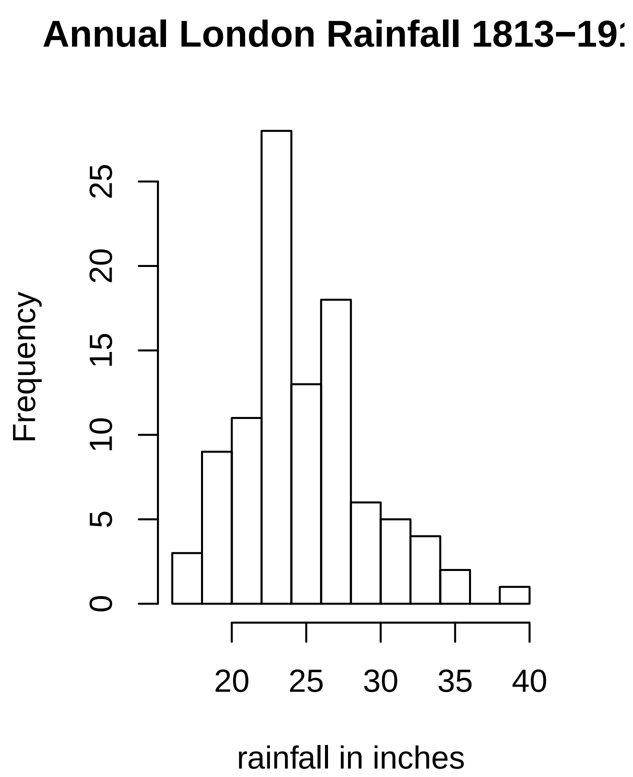
W6.SES Forecast

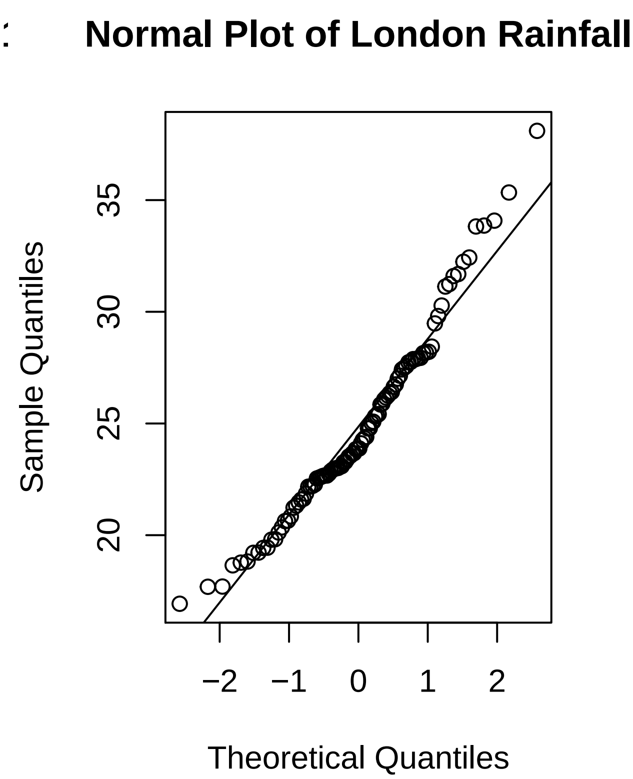
Yansong Liu

5/28/2020

Dataset source:https://robjhyndman.com Dataset description:https://pkg.robjhyndman.com/fma/reference/index.html Simple Exponential Smoothing -dataset:total annual rainfall recorded-during the years 1813-1912-in inches-for London,England -scan:go to website-grab data-store in array

```
rm(list = ls(all=TRUE))
rain.data=scan('https://robjhyndman.com/tsdldata/hurst/precip1.dat',skip = 1)
rain.ts=ts(rain.data,start = c(1813))
#rain.ts

#plot-summarize data nature
par(mfrow=c(1,2))
hist(rain.data,main = 'Annual London Rainfall 1813-1912',xlab = 'rainfall in inches')
#histogram:mound shaped-not quite symmetrical
qqnorm(rain.data,main='Normal Plot of London Rainfall')
qqline(rain.data)
```

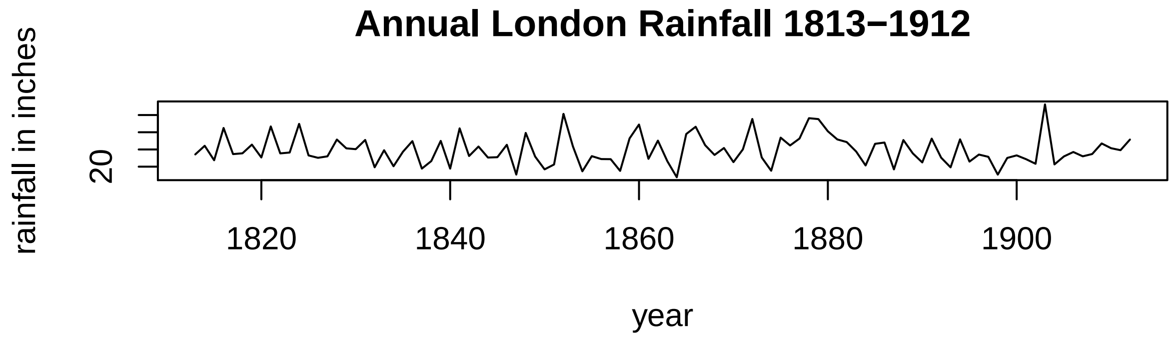


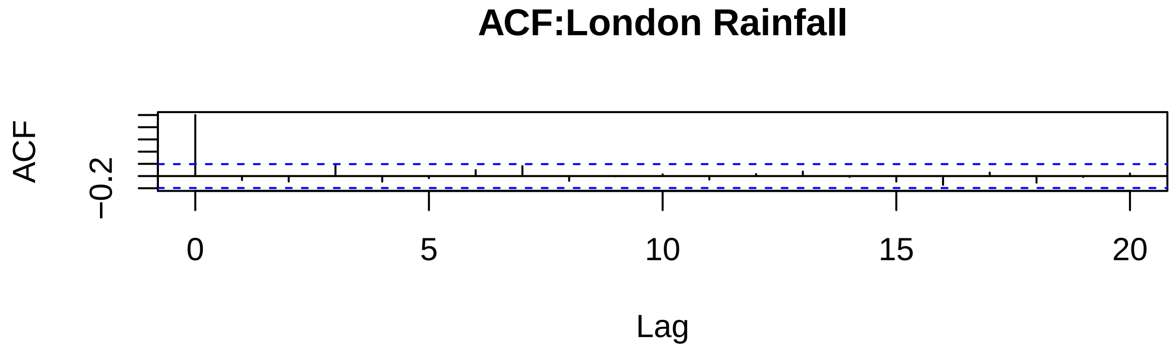



```
#normal distributed test
#skew-not quite normal-systematic departure from normality
```

```r
#close-not extreme

#plot-data, ACF
par(mfrow=c(2,1))
plot.ts(rain.ts,main='Annual London Rainfall 1813-1912',
        xlab='year',ylab='rainfall in inches')
acf(rain.ts,main='ACF:London Rainfall')
#time series->sequence-see ACF
#dataset-noise-no structure
#autocorrelations week-cannot fit obvious model

library(forecast)
```

```
## Registered S3 method overwritten by 'xts':
##   method     from
##   as.zoo.xts zoo

## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo

## Registered S3 methods overwritten by 'forecast':
##   method             from
##   fitted.fracdiff    fracdiff
##   residuals.fracdiff fracdiff
```





```r
auto.arima(rain.ts)
```

```
## Series: rain.ts
## ARIMA(0,0,0) with non-zero mean
##
## Coefficients:
##          mean
##       24.8239
## s.e.   0.4193
##
## sigma^2 estimated as 17.76:  log likelihood=-285.25
## AIC=574.49   AICc=574.61   BIC=579.7
```

```r
# no fit model-no coef
```

Forecast Naive method -data:previous year's=current year's-last few year's annual rainfall -forecast-predict future value:current time n_th -h lags away-future time:n+h -x_n_n+h -x_n_n+1=x_n

seasonal naive method -seasonality-predict future value-value previous season -n predict next period n+1 -1 season ago-week:s=7 -x_n_n+1=x_n+1-s

Average method-Simple Moving Average -average all past values, weighted equally-predict future value

```r
mean(rain.ts)
```

```
## [1] 24.8239
```

```r
#forecast:rainfall in 1913
```

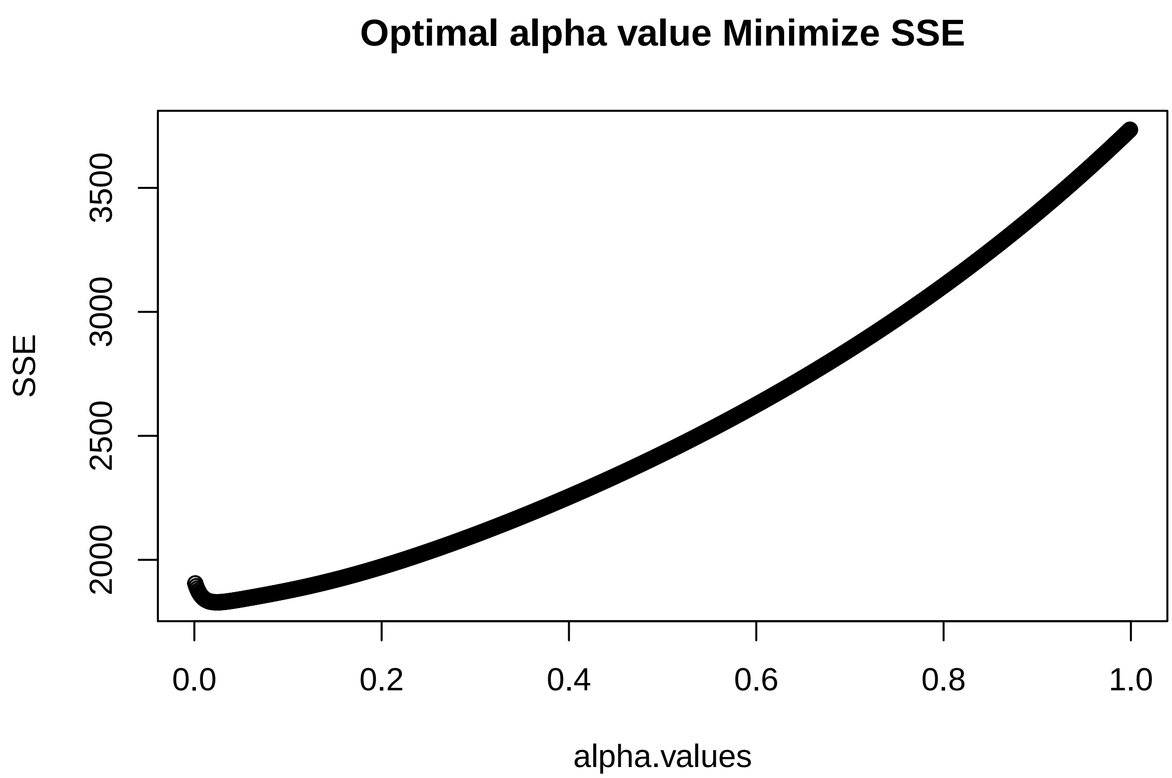
Simple Exponential Smoothing(SES) greater weight->closer values-decaying sequence of weights-lesser weight to further past -geometric series-decrease at a constant ratio -x_n_n+1=alpha*x_n+(1-alpha)*x_n-1_n -alpha=1:naive forecast -alpha->1:series decay rapidly, emphasis on 'near' observations -alpha->0:series decay slowly, emphasis on futher past points

```r
alpha=0.2  #increase alpha-more rapid decay
forecast.values=NULL  #establish array-forecast values

n=length(rain.data)

#naive first forecast
forecast.values[1]=rain.data[1]

#loop to creat all forecaset values
for (i in 1:n) {
  forecast.values[i+1]=alpha*rain.data[i]+(1-alpha)*forecast.values[i]
}
paste('forecast for time',n+1,'=',forecast.values[n+1])
```

```
## [1] "forecast for time 101 = 25.3094062064236"
```

Moving Towards a Least Squares Approach-find level:alpha

```r
SSE= NULL
alpha.values=seq(0.001,0.999,by=0.001)
number.alphas=length(alpha.values)

for (k in 1:number.alphas) {
    forecast.values=NULL
    forecast.values[1]=rain.data[1]
    n=length(rain.data)
    alpha=alpha.values[k]

    for (i in 1:n) {
       forecast.values[i+1]=alpha*rain.data[i]+(1-alpha)*forecast.values[i]
    }
    SSE[k]=sum((rain.data-forecast.values[1:n])^2)
}
plot(SSE~alpha.values,main='Optimal alpha value Minimize SSE')
```




```r
#retrieve SSE position, alpha
index.of.smallest.SSE=which.min(SSE)
index.of.smallest.SSE  #return position 24
```

```
## [1] 24
```

```r
alpha.values[which.min(SSE)]   #return alpha=0.024
```

```
## [1] 0.024
```

```r
alpha=0.024
forecast.values=NULL
forecast.values[1]=rain.data[1]
n=length(rain.data)

for (i in 1:n) {
  forecast.values[i+1]=alpha*rain.data[i]+(1-alpha)*forecast.values[i]
}
paste('forecast at time:',n+1,'=',forecast.values[n+1])
```

```
## [1] "forecast at time: 101 = 24.6771392918524"
```

```r
# HoltWinters for SES-turn off seasonal,trend effects
HoltWinters(rain.ts,beta = FALSE, gamma = FALSE)
```

```
## Holt-Winters exponential smoothing without trend and without seasonal component.
##
## Call:
## HoltWinters(x = rain.ts, beta = FALSE, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.02412151
##  beta : FALSE
##  gamma: FALSE
##
## Coefficients:
##       [,1]
## a 24.67819
```

Holt-Winters for level,Trend-alpha,beta -ACF:slow decay, PACF:cut off sharply-accommodate trend -non-seasonal data-level,trend -SES:forecast value at step,n+1-smoothed value at step n

```r
#set up transformed data,smoothing parameters
data=rain.data
N=length(data)
alpha=0.7
beta=0.5

#prepare empty array-store values
forecast=NULL
level=NULL
trend=NULL
#initialize level,trend in a very simple way
level[1]=data[1]
trend[1]=data[2]-data[1]
#initialize forecast to get started
forecast[1]=data[1]
```

```r
forecast[2]=data[2]

#loop to build forecast
for (n in 2:N) {
  level[n]=alpha*data[n]+(1-alpha)*(level[n-1]+trend[n-1])
  trend[n]=beta*(level[n]-level[n-1])+(1-beta)*trend[n-1]
  forecast[n+1]=level[n]+trend[n]
}

#display calculated forecast value
forecast[3:N]
```

```
##  [1] 28.58000 24.03400 31.75830 25.92469 23.61996 25.67606 22.62277
##  [8] 31.19431 25.71777 23.68708 31.96190 24.97968 21.55863 21.33779
## [15] 26.97731 26.29711 25.49906 27.92697 20.25991 23.01382 19.56516
## [22] 23.15575 27.88142 20.75862 20.45976 26.93269 20.60663 30.58199
## [29] 25.32440 25.86308 22.66011 21.80068 25.66561 17.97513 28.28719
## [36] 24.68979 19.09914 18.94475 34.93377 29.95016 19.43202 20.63337
## [43] 20.95060 21.45506 18.27962 27.41063 34.66121 25.52046 27.20557
## [50] 21.55966 14.98350 26.86943 33.57491 29.27783 23.93647 24.26730
## [57] 20.46499 23.50855 34.23263 25.56993 17.91365 26.03498 26.91914
## [64] 29.02919 35.56699 36.73488 32.34804 27.83017 25.68725 22.67584
## [71] 18.12337 24.14145 27.20987 19.87044 26.39396 24.73763 21.17906
## [78] 27.39532 23.70733 19.26637 26.66785 22.54005 23.07969 22.70269
## [85] 17.21615 20.82853 23.28826 22.85779 21.09142 38.59651 25.35176
## [92] 22.16872 22.84875 22.23420 23.01436 26.71189 26.37498 25.32017
```

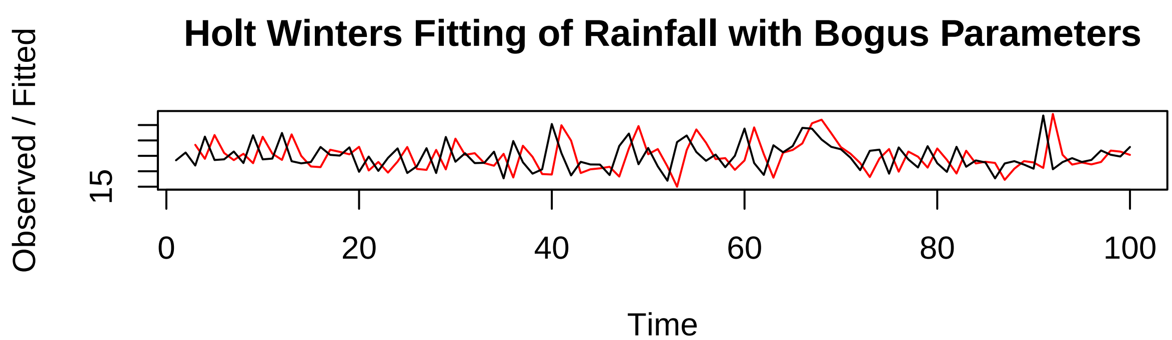
```r
#verify with HaltWinters() output
par(mfrow=c(2,1))
m=HoltWinters(data,alpha=0.7,beta=0.5,gamma=FALSE)
m$fitted[,1]
```

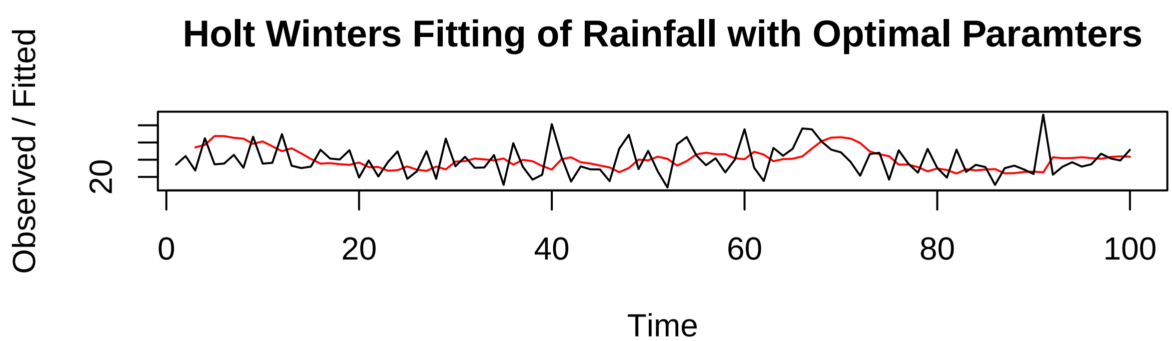
```
## Time Series:
## Start = 3
## End = 100
## Frequency = 1
##  [1] 28.58000 24.03400 31.75830 25.92469 23.61996 25.67606 22.62277
##  [8] 31.19431 25.71777 23.68708 31.96190 24.97968 21.55863 21.33779
## [15] 26.97731 26.29711 25.49906 27.92697 20.25991 23.01382 19.56516
## [22] 23.15575 27.88142 20.75862 20.45976 26.93269 20.60663 30.58199
## [29] 25.32440 25.86308 22.66011 21.80068 25.66561 17.97513 28.28719
## [36] 24.68979 19.09914 18.94475 34.93377 29.95016 19.43202 20.63337
## [43] 20.95060 21.45506 18.27962 27.41063 34.66121 25.52046 27.20557
## [50] 21.55966 14.98350 26.86943 33.57491 29.27783 23.93647 24.26730
## [57] 20.46499 23.50855 34.23263 25.56993 17.91365 26.03498 26.91914
## [64] 29.02919 35.56699 36.73488 32.34804 27.83017 25.68725 22.67584
## [71] 18.12337 24.14145 27.20987 19.87044 26.39396 24.73763 21.17906
## [78] 27.39532 23.70733 19.26637 26.66785 22.54005 23.07969 22.70269
## [85] 17.21615 20.82853 23.28826 22.85779 21.09142 38.59651 25.35176
## [92] 22.16872 22.84875 22.23420 23.01436 26.71189 26.37498 25.32017
```

```r
plot(m,main = 'Holt Winters Fitting of Rainfall with Bogus Parameters')
m2=HoltWinters(data,gamma = FALSE)
m2
```

```
## Holt-Winters exponential smoothing with trend and without seasonal component.
##
## Call:
## HoltWinters(x = data, gamma = FALSE)
##
## Smoothing parameters:
##  alpha: 0.1973985
##  beta : 0.3469287
##  gamma: FALSE
##
## Coefficients:
##         [,1]
## a 26.2701189
## b  0.2851515
```

```r
plot(m2,main = 'Holt Winters Fitting of Rainfall with Optimal Paramters')
```





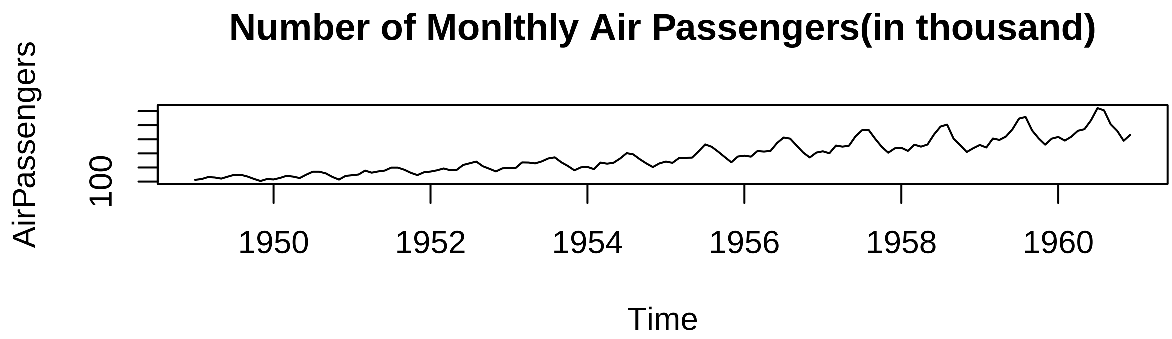
Airline Dataset -Monthly Airline Passenger Number 1949-1960

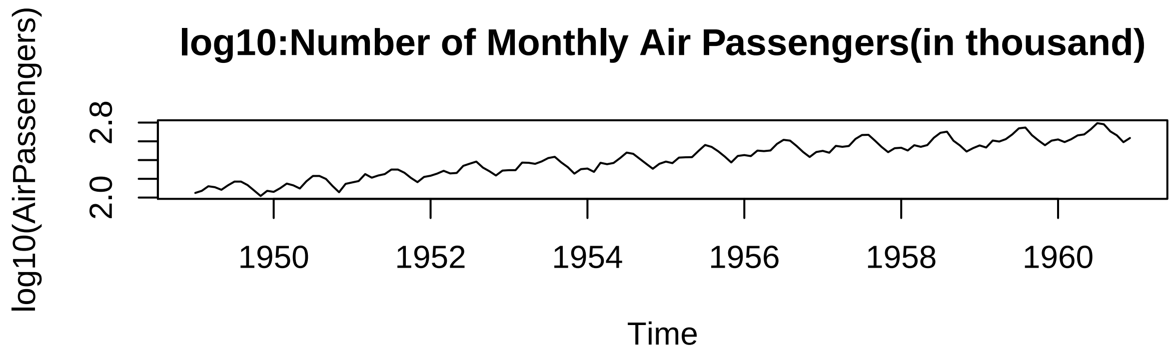
```r
#AirPassengers
```

```r
par(mfrow=c(2,1))
plot.ts(AirPassengers,main='Number of Monlthly Air Passengers(in thousand)')
#strong seasonality-12 month
#strong trend-go up
#increasing variability in later years

#tranform-log10-stablize variation
log10(AirPassengers)
```

```
##           Jan      Feb      Mar      Apr      May      Jun      Jul
## 1949 2.049218 2.071882 2.120574 2.110590 2.082785 2.130334 2.170262
## 1950 2.060698 2.100371 2.149219 2.130334 2.096910 2.173186 2.230449
## 1951 2.161368 2.176091 2.250420 2.212188 2.235528 2.250420 2.298853
## 1952 2.232996 2.255273 2.285557 2.257679 2.262451 2.338456 2.361728
## 1953 2.292256 2.292256 2.372912 2.371068 2.359835 2.385606 2.421604
## 1954 2.309630 2.274158 2.371068 2.356026 2.369216 2.421604 2.480007
## 1955 2.383815 2.367356 2.426511 2.429752 2.431364 2.498311 2.561101
## 1956 2.453318 2.442480 2.501059 2.495544 2.502427 2.572872 2.615950
## 1957 2.498311 2.478566 2.551450 2.541579 2.550228 2.625312 2.667453
## 1958 2.531479 2.502427 2.558709 2.541579 2.559907 2.638489 2.691081
## 1959 2.556303 2.534026 2.608526 2.597695 2.623249 2.673942 2.738781
## 1960 2.620136 2.592177 2.622214 2.663701 2.673942 2.728354 2.793790
##           Aug      Sep      Oct      Nov      Dec
## 1949 2.170262 2.133539 2.075547 2.017033 2.071882
## 1950 2.230449 2.198657 2.123852 2.056905 2.146128
## 1951 2.298853 2.264818 2.209515 2.164353 2.220108
## 1952 2.383815 2.320146 2.281033 2.235528 2.287802
## 1953 2.434569 2.374748 2.324282 2.255273 2.303196
## 1954 2.466868 2.413300 2.359835 2.307496 2.359835
## 1955 2.540329 2.494155 2.437751 2.374748 2.444045
## 1956 2.607455 2.550228 2.485721 2.432969 2.485721
## 1957 2.669317 2.606381 2.540329 2.484300 2.526339
## 1958 2.703291 2.606381 2.555094 2.491362 2.527630
## 1959 2.747412 2.665581 2.609594 2.558709 2.607455
## 1960 2.782473 2.705864 2.663701 2.591065 2.635484
```

```r
plot(log10(AirPassengers),main='log10:Number of Monthly Air Passengers(in thousand)')
```





Exponential Smoothing with Trend, Seasonality seasonal differences -additive:Jan-add a constant amount-June projection -multiplicative-multiply a constant amount

```
#transform data-SES
Airpassengers.SES=HoltWinters(x=log10(AirPassengers),beta=FALSE,gamma = FALSE)
Airpassengers.SES$SSE
```

```
## [1] 0.3065102
```

```
#additive seasonality
Airpassengers.HW=HoltWinters(log10(AirPassengers))
Airpassengers.HW$SSE
```

```
## [1] 0.0383026
```

```
#smoothing parameters
Airpassengers.HW$alpha
```

```
##    alpha
## 0.326612
```

```
Airpassengers.HW$beta
```

```
##        beta
## 0.005744246
```

```r
Airpassengers.HW$gamma
```

```
##     gamma
## 0.8207255
```

```r
Airpassengers.HW$coefficients
```

```
##            a            b           s1           s2           s3
##  2.680598830  0.003900787 -0.031790733 -0.061224237 -0.015941495
##           s4           s5           s6           s7           s8
##  0.006307818  0.014138008  0.067260071  0.127820295  0.119893006
##           s9          s10          s11          s12
##  0.038321663 -0.014181699 -0.085995400 -0.044672707
```

```r
rm(list = ls(all=TRUE))
library('forecast')
Airpassengers.HW=HoltWinters(log10(AirPassengers))
AirPassengers.forecast=forecast(Airpassengers.HW)
plot(AirPassengers.forecast)
```