Lecture1 practical exercises

In the first lecture, you have seen a few examples about the famous Logistic map. In this excercise, we will use RStudio to make some of those figures presented, which also helps you to get familar with RStudio.

The mathematical form of the Logistic map is

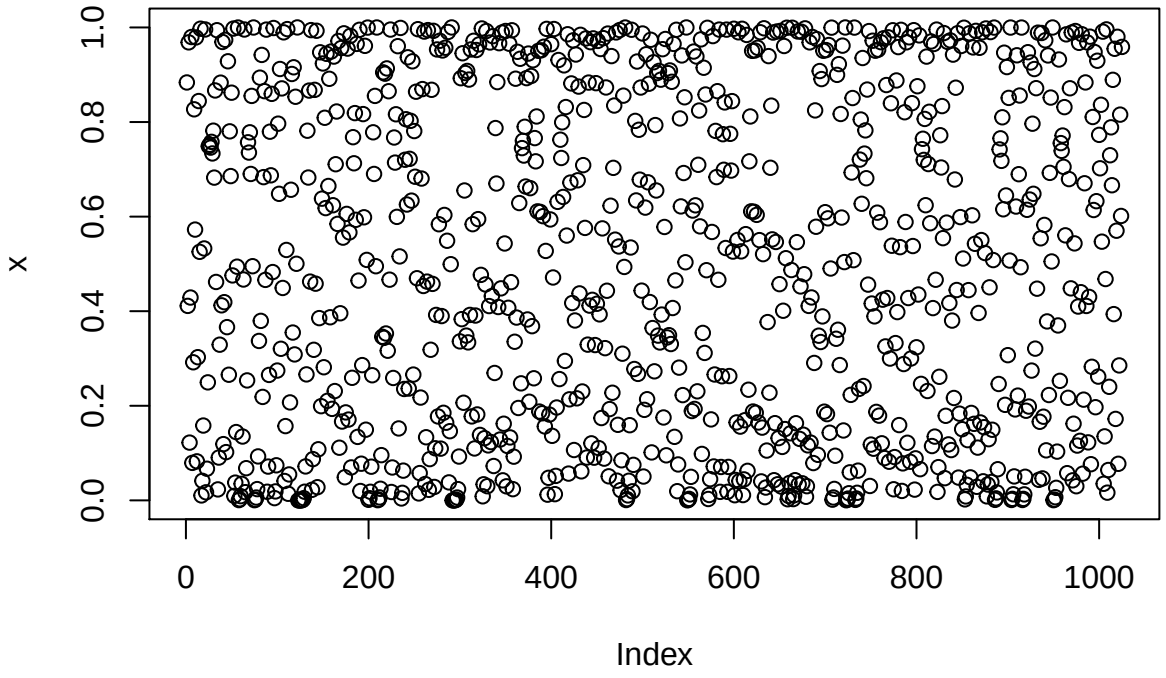
$$x_{i+1} = a x_i (1 - x_i).$$

Let's first create a time series of Logistic map, you may use the logistic.R function file. To compile the function you need to first set the working directory to the one contains the function codes. For example:

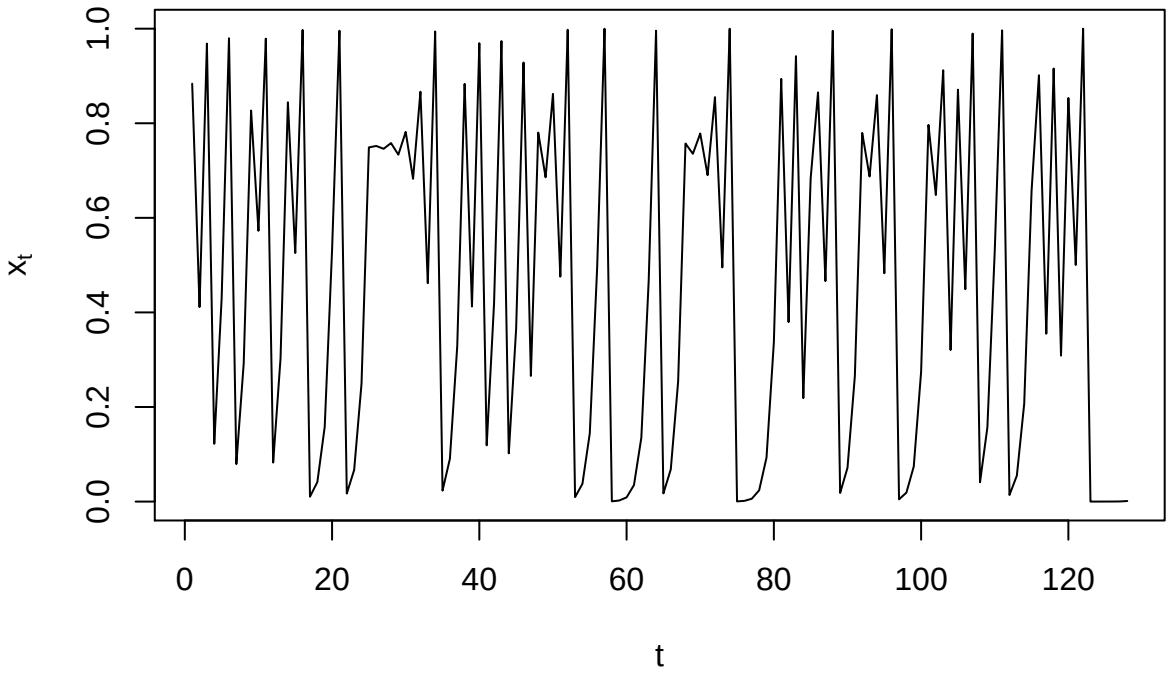
```
setwd("/Users/hailiangdu/HDU/Teaching/NanJing_SummerSchool/workshops")
source('logistic.R')
```

Now open logistic.R function code and take a look at the comments. Create a time series of Logistic map using $a = 4$ and $x_0 \in (0, 1)$ (you can use the runif function to sample a point uniformly between 0 and 1). And plot the time series.

```
a=4
x=logistic(runif(1),a,1024)
plot(x)
```

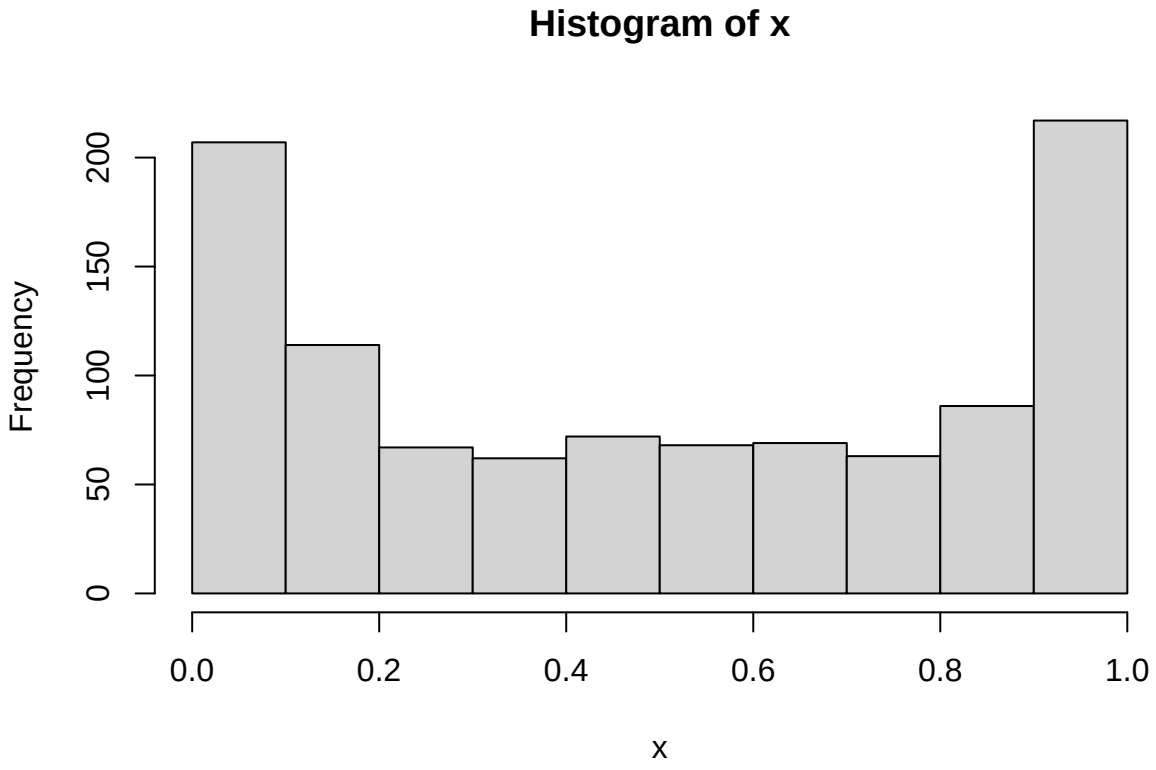



```
plot(x[1:128],type="l",ylab=expression(x[t]), xlab="t") #zoom in plot
```



It is often useful to look at the histogram of a time series, which reflects the unconditional distribution (climatology). To do so, simply use this hist function.

```
hist(x)
```

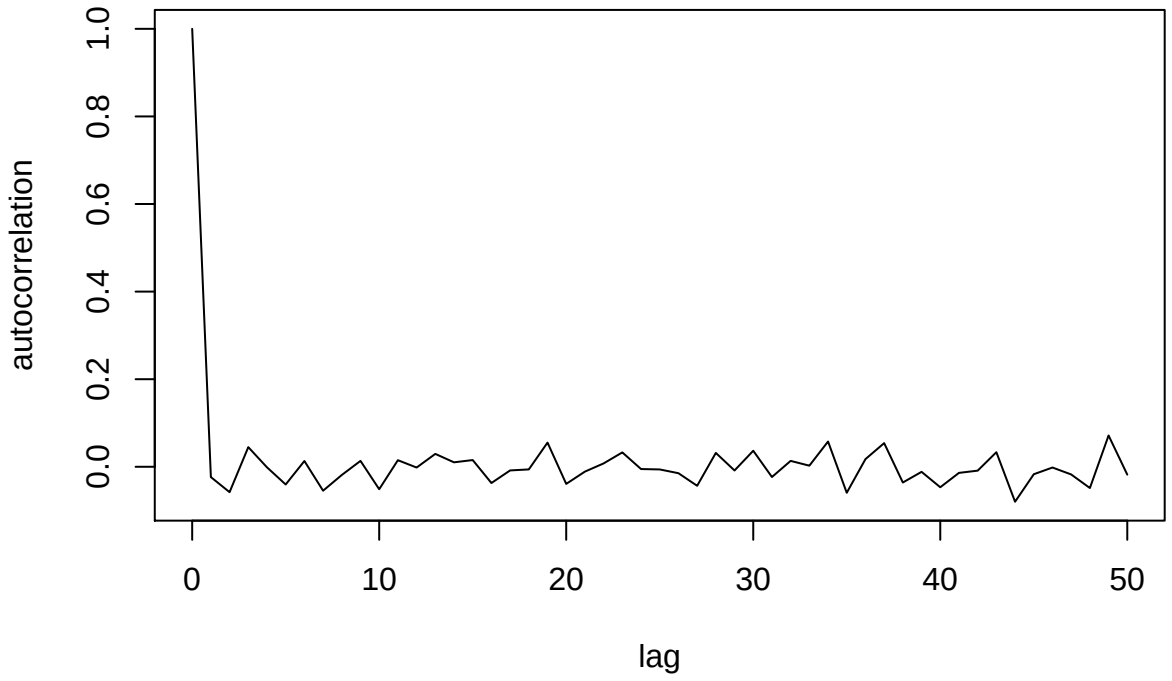


Now find the autocorrelation functions up to lag 50, using the source function code autoCC.R. Note the autocorrelation functions only measures linear relationship.

```r
source('autoCC.R')

# find the autocorrelation up to lag 50
autoCC(x,50)
```

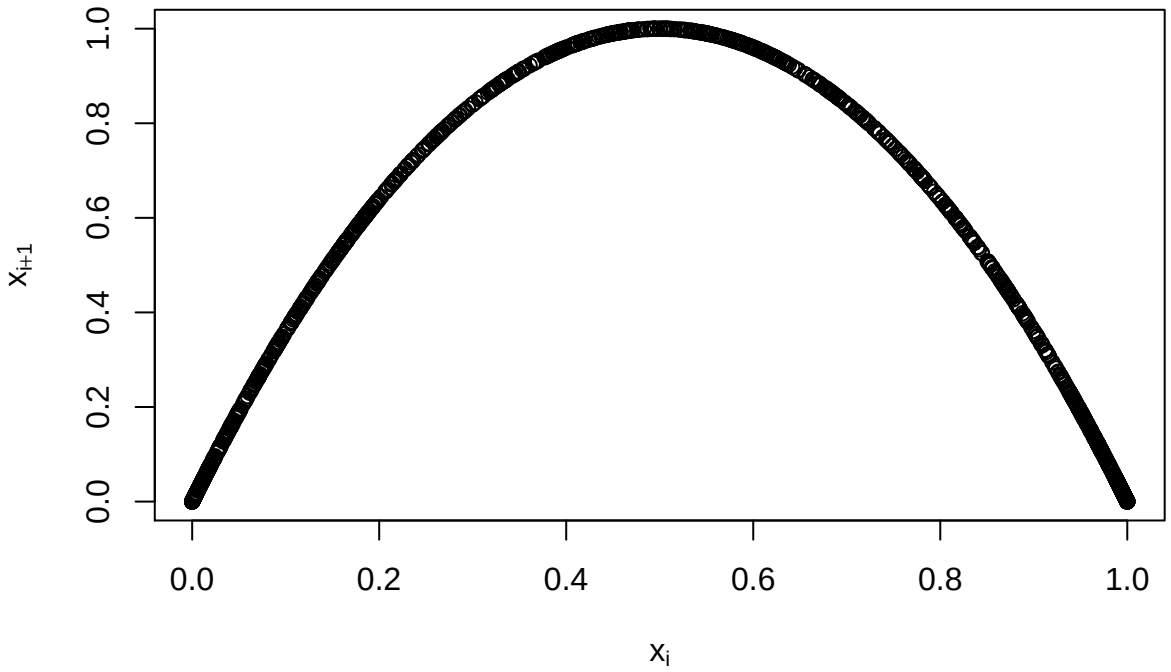
```
##  [1]  1.000000000 -0.023558889 -0.057986560  0.044844955 -0.001241100
##  [6] -0.040446109  0.013168319 -0.054589776 -0.018735476  0.013452815
## [11] -0.051267169  0.015046272 -0.001726744  0.029392584  0.010205357
## [16]  0.015325375 -0.037047615 -0.008322391 -0.006028730  0.055257141
## [21] -0.039066987 -0.011035631  0.007709730  0.032846122 -0.005147854
## [26] -0.006104249 -0.014680028 -0.043432958  0.031685140 -0.008459401
## [31]  0.036602648 -0.023380849  0.013490697  0.002527920  0.057728847
## [36] -0.059348268  0.017509155  0.054007748 -0.035908303 -0.011547913
## [41] -0.046754217 -0.013944105 -0.008828940  0.033323068 -0.079797197
## [46] -0.017033263 -0.001678727 -0.017515775 -0.048527718  0.071436822
## [51] -0.017863778
```

```r
#plot the autocorrelation up to lag 50
plot(0:50,autoCC(x,50),type="l",ylab='autocorrelation', xlab="lag")
```

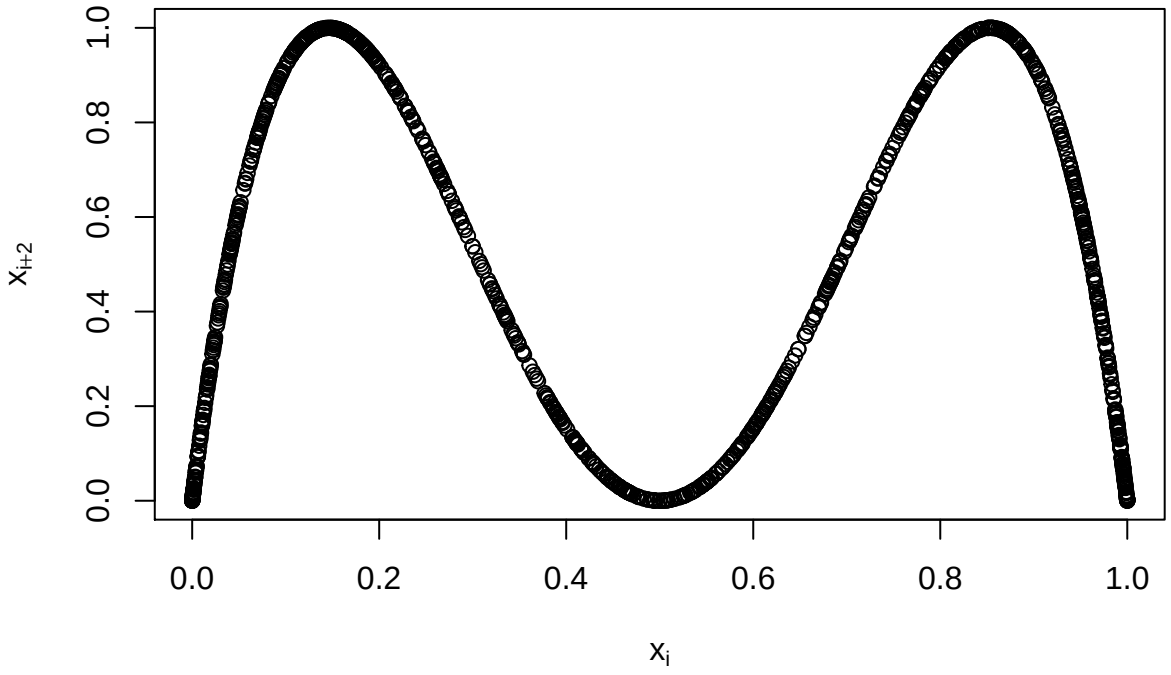


To visualize the relationship between the current state and its lags, we can use delay plots. Make delay plots with lag 1 and lag 2, using the source function code delayplot.R.

```r
source('delayplot.R')

delayplot(x,1) # delay plot with lag 1
```

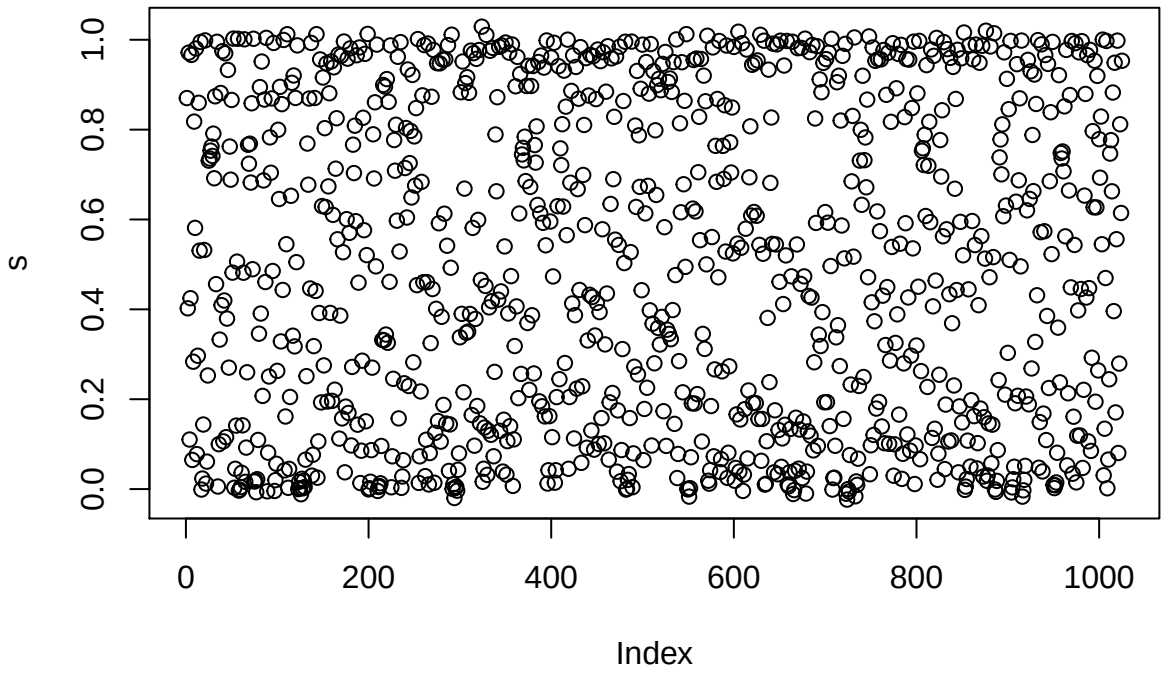



```
delayplot(x,2) # delay plot with lag 1
```

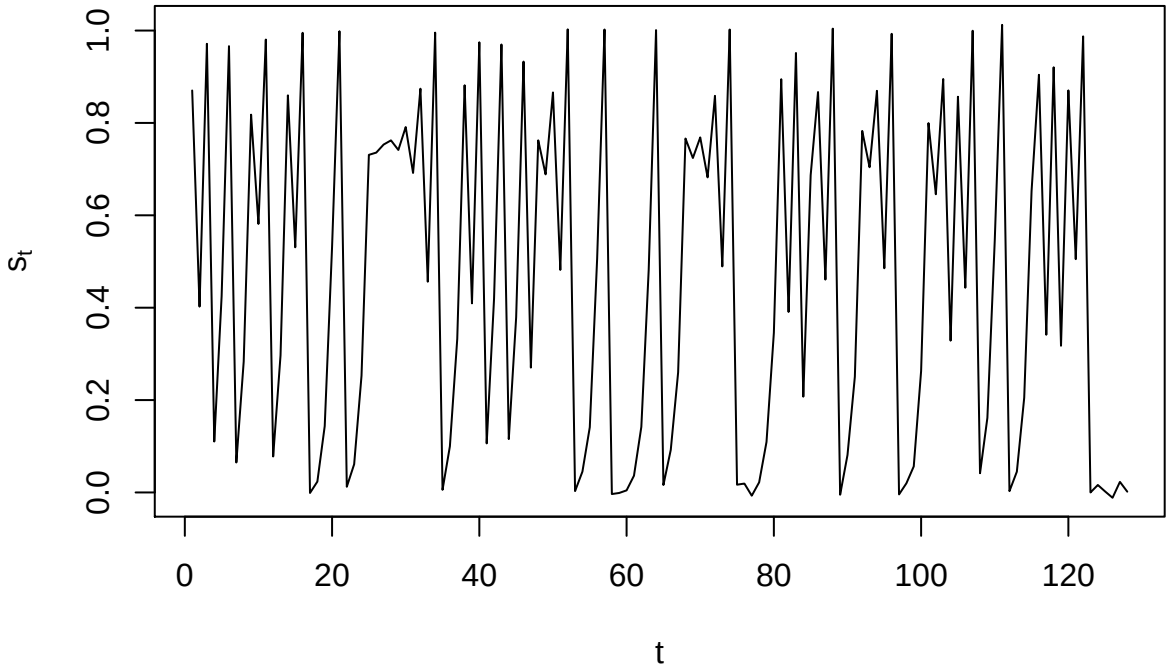


In practice, a real time series almost always comes with observational noise. Let's add IID Gaussian observational noise ($N(0, 0.01^2)$) onto the time series of Logistic map created above (use rnorm to create Gaussian samples), and redo the plots based on the observations instead.

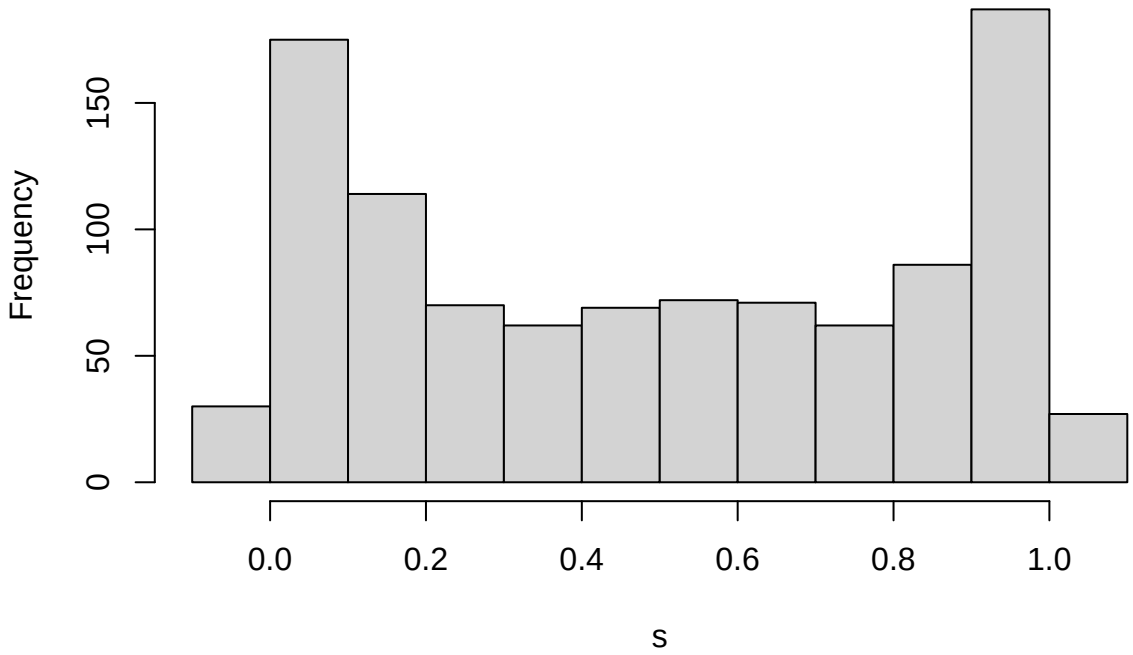
```
#create observations using Gaussian IID observational noise
s=x+rnorm(length(x),0,0.01)

plot(s)
```




```
plot(s[1:128],type="l",ylab=expression(s[t]), xlab="t") #zoom in plot
```



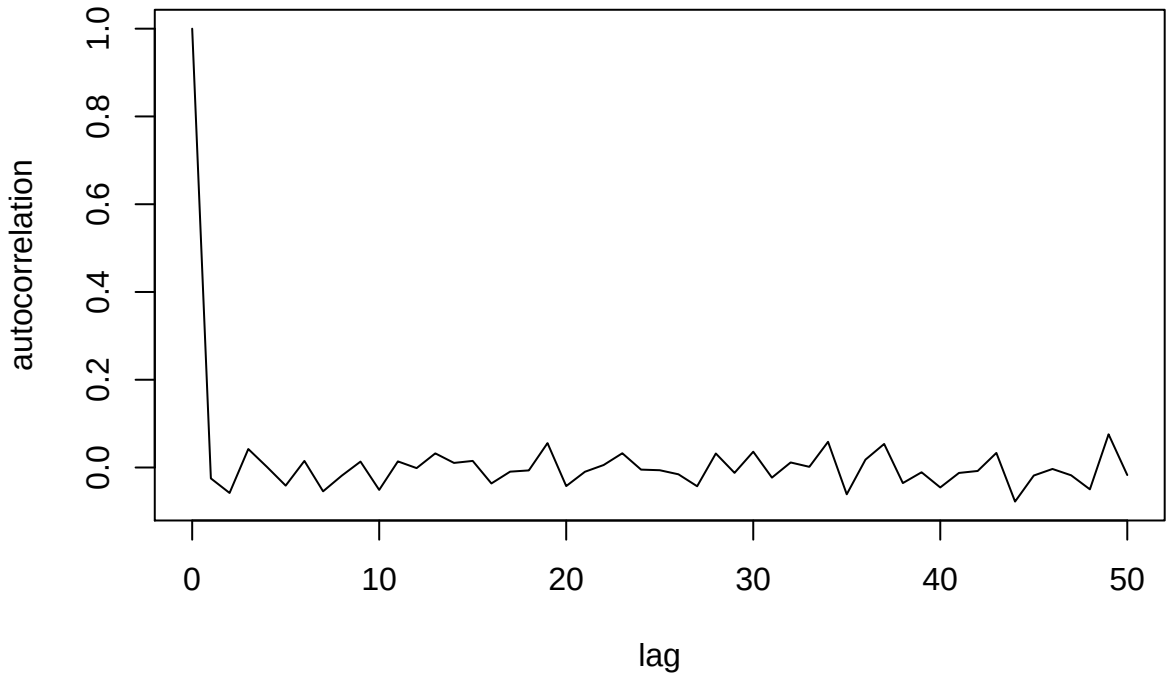

```
hist(s)
```



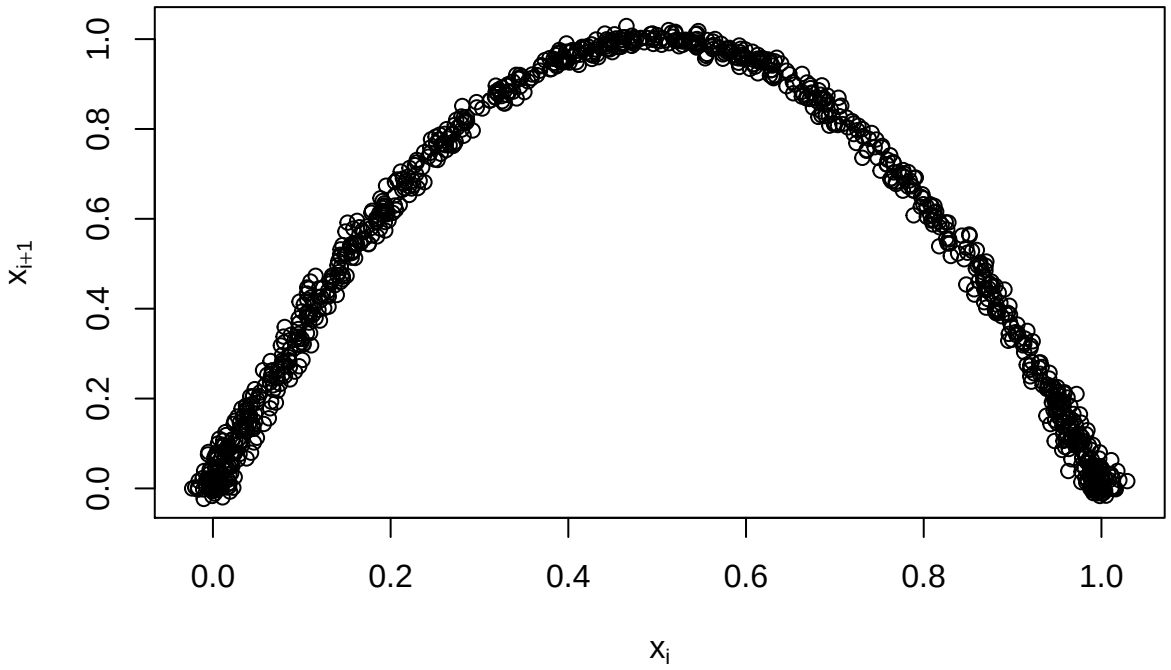

```
autoCC(s,50)
```

```
##  [1]  1.000000000 -0.024864849 -0.058016233  0.042104235  0.001257268
##  [6] -0.041280828  0.015014626 -0.054278001 -0.018602801  0.013398146
## [11] -0.051054107  0.013926847 -0.001402573  0.032207278  0.010451285
## [16]  0.015024491 -0.036364460 -0.009494846 -0.006612621  0.055565520
## [21] -0.042393177 -0.009732092  0.005613143  0.032423817 -0.004828486
## [26] -0.006196911 -0.015512533 -0.042722304  0.031912391 -0.012202376
## [31]  0.035972019 -0.023012130  0.011486744  0.001501272  0.058595035
## [36] -0.061059407  0.017935797  0.053622033 -0.035529663 -0.010875113
## [41] -0.045477697 -0.012413139 -0.007923464  0.033260126 -0.077653685
## [46] -0.018306590 -0.003340817 -0.017818431 -0.049798930  0.075656910
## [51] -0.017178208
```

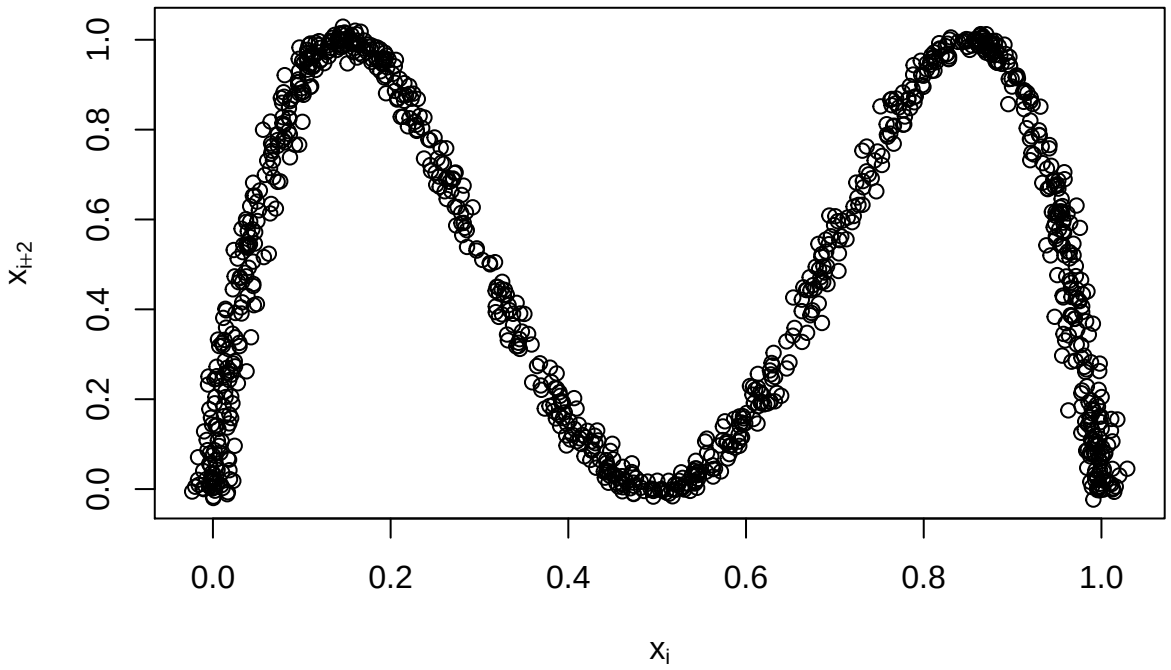
```
plot(0:50,autoCC(s,50),type="l",ylab='autocorrelation', xlab="lag")
```




```
delayplot(s,1)
```




```
delayplot(s,2)
```


0.0
0.2
0.4
0.6
0.8
1.0
x_{i+2}
0.0
0.2
0.4
0.6
0.8
1.0
x_i