HW2

Lince Rumainum

January 31, 2019

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

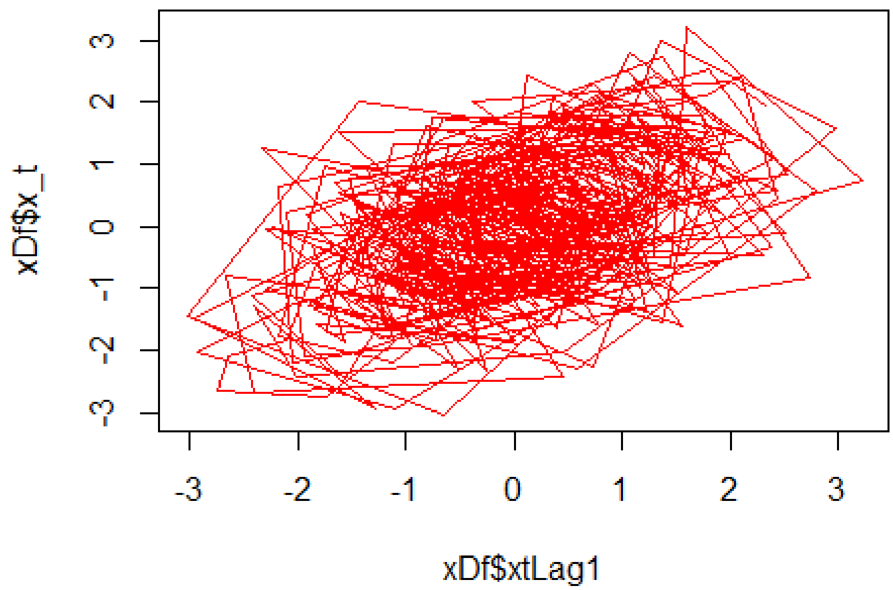
```r
# Lince Rumainum
# Time Series Analysis
# HW2

# libraries list
#install.packages("DataCombine")
#install.packages("nlme")
library(DataCombine) # for slide function, i.e.: x_t-1
library(mgcv)

## Loading required package: nlme

## This is mgcv 1.8-26. For overview type 'help("mgcv-package")'.

#############################################################################
##################################
# Problem 1
#############################################################################
##################################

# Create 100 samples of x_t ~ iid N(0,1)
sampleSize = 100

# Create a main data frame for problem 1 with i = 1 to 100
mainDf1 <- data.frame(i = seq(0, sampleSize, by = 1))

# Obtain 100 estimates of mean of x_i and mean of sigmasquare_i from 100
different x_t ~ iid N(0,1) values
for(k in 1:100){
  # Create 100 samples of x_t ~ iid N(0,1)
  x_i<- rnorm(sampleSize, mean = 0, sd = 1)
  # Calculate the mean and the mean variance of the samples
```

```
  mainDf1$meanVal[k] <- mean(x_i)
  mainDf1$meanVarVal[k] <- var(x_i)
}
mainDf1

##       i       meanVal meanVarVal
## 1     0  1.847232e-01  0.8434694
## 2     1  1.085858e-01  1.0148233
## 3     2  1.752538e-01  0.8825417
## 4     3 -4.575169e-02  0.7342363
## 5     4  2.061994e-01  0.7552576
## 6     5  8.767032e-02  0.8006301
## 7     6 -8.090654e-02  1.0131072
## 8     7 -9.805379e-02  0.9719662
## 9     8 -1.147912e-01  0.8730971
## 10    9  2.237196e-01  0.9589269
## 11   10 -1.318554e-02  1.2071031
## 12   11 -5.869274e-02  0.9402155
## 13   12 -5.281350e-02  0.8971385
## 14   13  1.293025e-03  1.1253025
## 15   14 -1.242682e-02  0.9755536
## 16   15 -5.771772e-02  0.9500835
## 17   16 -6.277000e-02  0.7687688
## 18   17  2.995520e-02  0.8561773
## 19   18  1.662792e-02  1.0409257
## 20   19  1.639619e-01  0.9610495
## 21   20 -5.190652e-02  0.9394073
## 22   21 -1.590549e-01  1.0282946
## 23   22 -5.361830e-02  0.8932469
## 24   23 -8.965351e-02  1.0620864
## 25   24  8.593970e-02  1.0631486
## 26   25  7.225832e-02  0.9280984
## 27   26 -1.277749e-02  0.8691008
## 28   27  8.370127e-04  0.8938746
## 29   28  3.645985e-02  1.0896055
## 30   29  1.730672e-02  0.9482932
## 31   30  1.494686e-01  0.9515380
## 32   31 -2.117684e-02  1.2323960
## 33   32 -2.349094e-02  1.0490644
## 34   33 -2.059478e-01  0.8526253
## 35   34 -2.368598e-01  0.8728439
## 36   35 -3.994069e-02  1.0792990
## 37   36 -1.661416e-01  0.8357619
## 38   37  2.081580e-02  1.2323106
## 39   38  1.398587e-02  0.8404066
## 40   39 -7.149687e-02  0.8983001
## 41   40  7.137427e-03  1.3897216
## 42   41 -2.361481e-04  1.0522781
## 43   42 -7.318671e-02  0.7553819
## 44   43  3.304496e-02  1.0599229
```

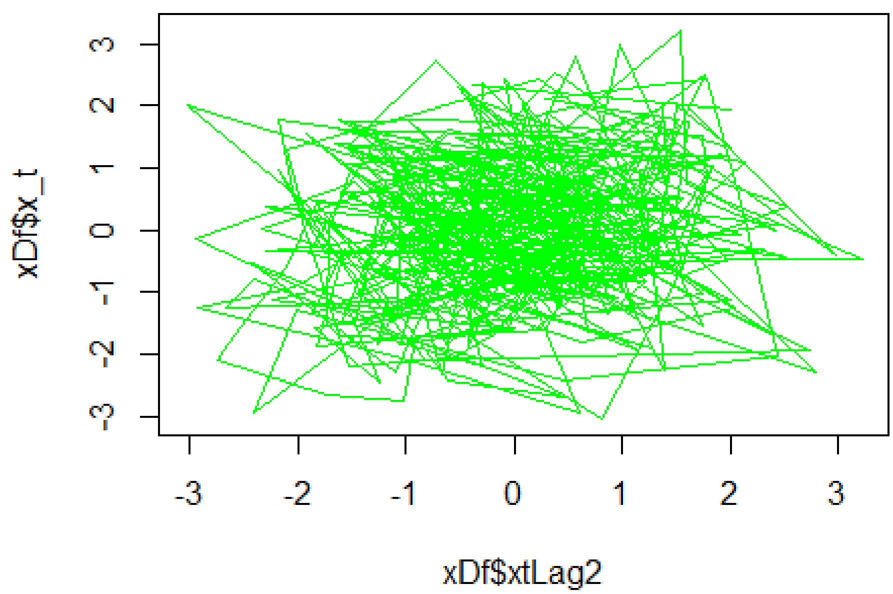
```
## 45   44 -1.065544e-03  0.9207083
## 46   45 -1.427195e-01  1.1001190
## 47   46  4.255228e-02  0.9371675
## 48   47 -3.254236e-02  0.7943418
## 49   48 -1.529341e-01  0.8578994
## 50   49 -1.391476e-01  0.9352398
## 51   50 -4.299177e-02  1.0224455
## 52   51 -2.113785e-01  1.1053180
## 53   52  1.287694e-01  0.9886688
## 54   53 -8.282806e-02  0.8925787
## 55   54 -7.167224e-02  1.0372734
## 56   55 -1.269445e-01  1.1944957
## 57   56  9.725911e-02  0.8889068
## 58   57  8.024062e-02  1.4675081
## 59   58  1.318075e-01  0.9179136
## 60   59 -6.838981e-02  1.2704530
## 61   60  2.919779e-02  0.9259549
## 62   61 -7.732603e-02  0.8534012
## 63   62 -2.060545e-02  0.9312502
## 64   63 -1.280870e-01  0.9430096
## 65   64 -1.164616e-01  0.8498317
## 66   65 -4.043469e-02  0.9899731
## 67   66 -6.300155e-02  0.9230435
## 68   67 -5.828497e-02  1.1922983
## 69   68 -5.838453e-02  0.9518576
## 70   69  6.297975e-02  0.9607049
## 71   70 -1.224205e-01  0.9602553
## 72   71 -8.838198e-02  0.9562378
## 73   72  6.279683e-03  0.9899707
## 74   73 -9.941813e-02  1.0165981
## 75   74  3.208054e-02  0.9833064
## 76   75 -5.956427e-04  1.1649728
## 77   76  1.017550e-01  0.9291089
## 78   77 -2.164778e-02  0.9613438
## 79   78 -1.998450e-01  0.6348057
## 80   79  4.998457e-02  1.0068329
## 81   80  5.593919e-02  1.1961713
## 82   81 -8.911976e-02  0.8903758
## 83   82 -9.231532e-02  1.0051179
## 84   83  1.661475e-02  0.8721023
## 85   84 -4.460432e-02  1.0083121
## 86   85 -1.553460e-01  1.2618727
## 87   86  1.220813e-02  0.8305050
## 88   87 -4.267206e-02  1.1144433
## 89   88  1.179391e-01  1.0136946
## 90   89 -3.680843e-03  0.8934856
## 91   90 -2.235756e-01  1.1842783
## 92   91 -4.725725e-03  0.9700473
## 93   92  5.447731e-02  1.1256367
## 94   93  7.173981e-02  0.7934488
```

```
## 95   94  7.112085e-05  1.0318848
## 96   95  8.149437e-02  1.0588587
## 97   96  1.335882e-02  0.7425467
## 98   97  1.246101e-01  1.0539260
## 99   98 -8.457408e-02  0.8900835
## 100  99 -2.562631e-01  1.1960794
## 101 100  1.847232e-01  0.8434694

# Compute the mean, variance of x_i bar for i = 1 to 100
meanX <- mean(mainDf1$meanVal)
meanX

## [1] -0.01817482

varX <- var(mainDf1$meanVal)
varX

## [1] 0.0101852

# Compute the mean, variance of variance_i bar for i = 1 to 100
meanVar <- mean(mainDf1$meanVarVal)
meanVar

## [1] 0.978411

varVar <- var(mainDf1$meanVarVal)
varVar

## [1] 0.02006139

# END OF PROBLEM #1

#############################################################################
##################################
# Problem 2 :
#############################################################################
##################################

# define: x_t = epsilon_t + 0.5*epsilon_t-1 called MA (1) process

rho_0 <- 1
rho_1 <- (0.5/(1+(0.5^2)))
rho_i <- 0 # for i > 1

# verified in module 2.3
sigma_ii = 1 + 2*(rho_1^2) # for all i > 1

# Create time sequences from 1 to 500
mainDf <- data.frame(t = seq(1, 500, by = 1))

# Create main data frame for iid epsilon_t values (500 samples)
```

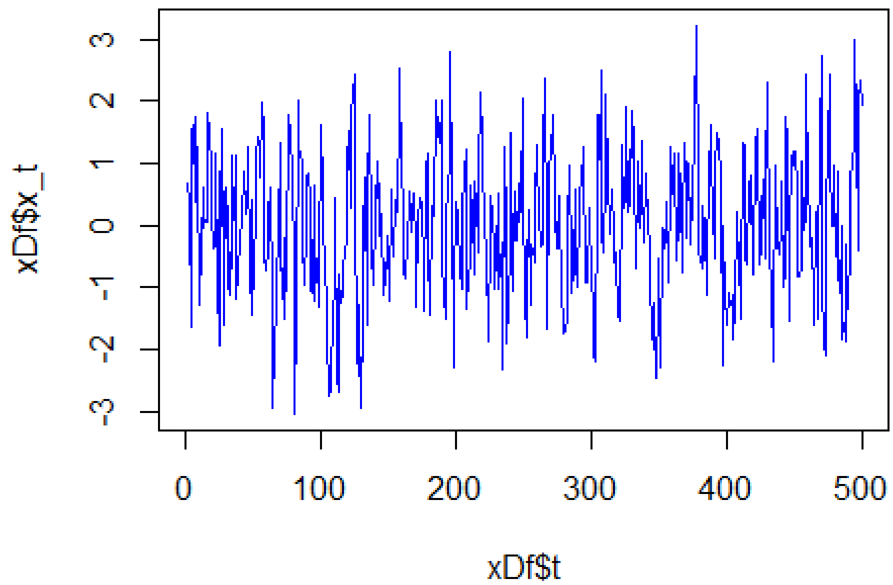
```r
mainDf$e_t <- rnorm(500, mean = 0, sd = 1)

# Create epsilon_t-1
mainDf <- slide(mainDf,"e_t", "t", NewVar="etLag1", slideBy = -1)

##
## Lagging e_t by 1 time units.

# Create time sequences from 0 to 500
xDf <- data.frame(t = seq(1, 500, by = 1))


# Create the rest of the table for x_t
for(i in 1:500){
  # Calculate data of x_t
  xDf$x_t[i] <- mainDf$e_t[i] + 0.5* mainDf$etLag1[i]
}

# Create x_t-1 and x_t-2
xDf <- slide(xDf,"x_t", "t", NewVar="xtLag1", slideBy = -1)

##
## Lagging x_t by 1 time units.

xDf <- slide(xDf,"x_t","t",NewVar="xtLag2", slideBy = -2)

##
## Lagging x_t by 2 time units.

# Line plot x_t vs x_t-1
plot(xDf$xtLag1, xDf$x_t, type = "l", col = "red")
```



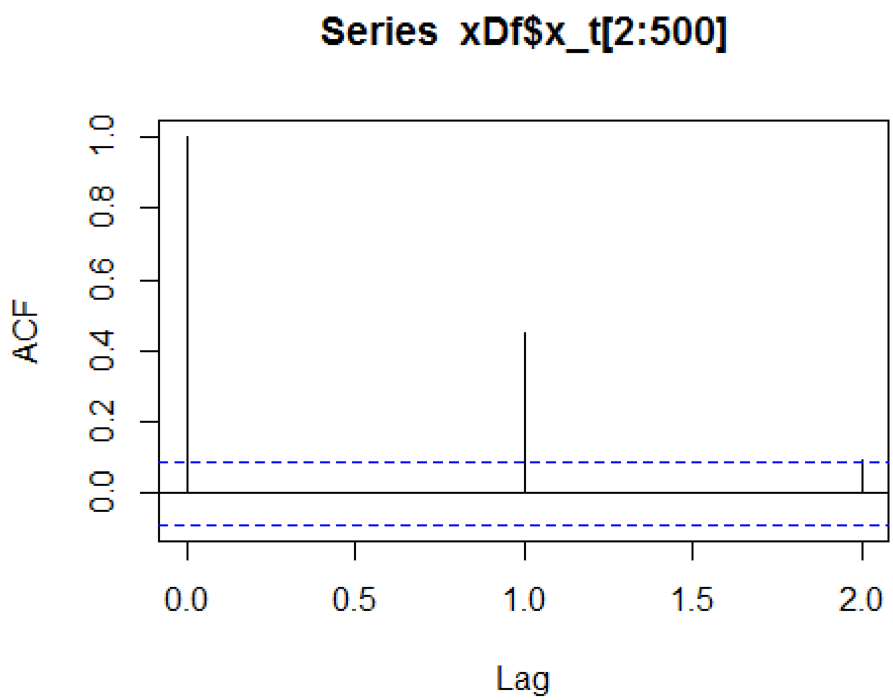

```
# Line plot x_t vs x_t-2
plot(xDf$xtLag2, xDf$x_t, type = "l", col = "green")
```



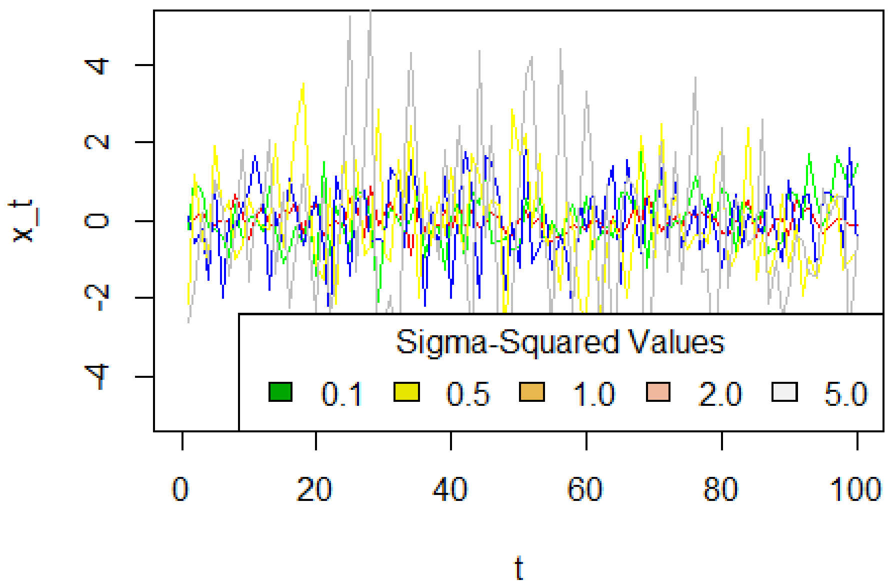
```r
# Line plot x_t vs t
plot(xDf$t, xDf$x_t, type = "l", col = "blue")
```




```r
#mean, variance, and autocorrelation
mean(xDf$x_t)

## [1] NA

var(xDf$x_t)

## [1] NA

acf (xDf$x_t[2:500], lag.max = 2, type=c("correlation"),plot=TRUE)
```




```
# END OF PROBLEM #2


########################################################################################################################
# Problem 3
########################################################################################################################

varArray <- c(sqrt(0.1), sqrt(0.5), sqrt(1.0), sqrt(2.0), sqrt(5.0))
colVar <- c("red","green","blue","yellow","gray")
t <- seq(1, 100, by = 1)

for (s in 1:length(varArray)){
  # Create 100 samples of x_t ~ iid N(0,1)
  x_t <- rnorm(t, mean = 0, sd = varArray[s])
  if (s == 1){
    plot(t, x_t, type = "l", col= colVar[s], xlim=c(0,length(t)), ylim=c(-
5,5))
  }
  else{
    lines(t, x_t, col= colVar[s], xlim=c(0,length(t)), ylim=c(-5,5))
  }
  legend("bottomright",title = "Sigma-Squared Values", c("0.1", "0.5", "1.0",
"2.0", "5.0"),fill=terrain.colors(5), horiz=TRUE)
}
```




```
# From the plot, it shows that for different value of sigma-squared, the plot
varies around its value.
# For example, for sigma-squared = 2.0 (blue line), the function flactuates
approximately between -2.0 and 2.0 while
# for sigma-squared = 0.1 (red line), it flactuates approximately between -
0.1 and 0.1.
# The plot also shows that each of the five different sigma-squared values,
they all still have approximately sample mean values of 0.

# END OF PROBLEM #3
################################################################################
###################################
```