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:

```
# 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)
```

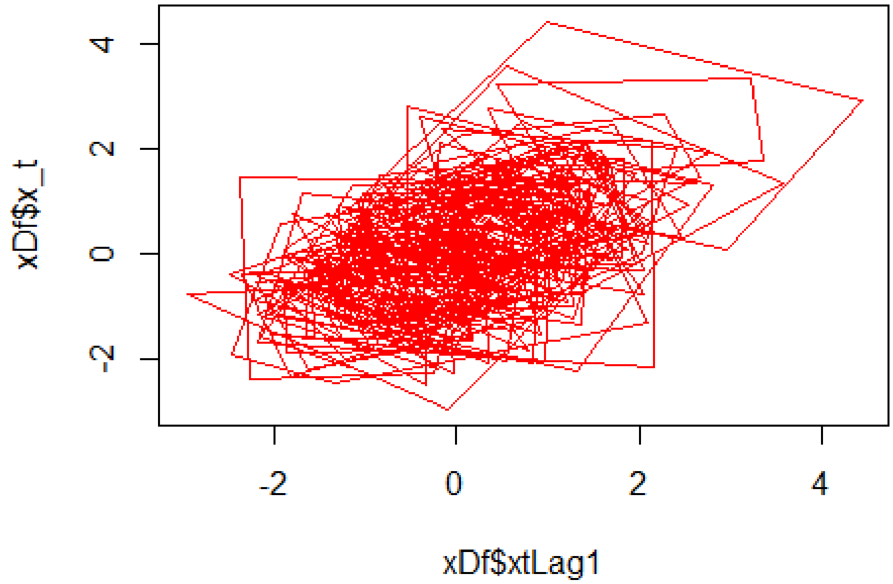
```
}
mainDf1
```

```
##       i      meanVal meanVarVal
## 1     0  0.057754055  1.1070505
## 2     1  0.155325577  0.9626471
## 3     2  0.022703393  1.0741841
## 4     3 -0.144675605  0.7839156
## 5     4  0.161763196  0.8832970
## 6     5 -0.046984702  1.0395357
## 7     6 -0.034939808  1.1101418
## 8     7  0.028686753  0.8344021
## 9     8  0.033805024  0.9117767
## 10    9 -0.015181535  1.0980395
## 11   10  0.092896626  0.8422795
## 12   11 -0.159154176  1.4088661
## 13   12 -0.035480443  0.8638764
## 14   13 -0.107722980  1.0950996
## 15   14  0.086024013  0.8924161
## 16   15 -0.052273723  1.1243079
## 17   16 -0.080227929  0.9019362
## 18   17  0.032259851  0.8730117
## 19   18  0.164819718  0.9754592
## 20   19  0.027753697  1.1917852
## 21   20 -0.033019707  0.8423774
## 22   21  0.146275891  1.0788675
## 23   22 -0.056834052  0.9440617
## 24   23  0.005074917  0.8779730
## 25   24 -0.064529129  0.8839993
## 26   25 -0.054551471  1.0996838
## 27   26 -0.047285267  1.2481339
## 28   27  0.105278954  0.9094382
## 29   28  0.139146828  1.1354261
## 30   29  0.139890813  0.8768656
## 31   30  0.070513889  1.0650882
## 32   31  0.039138955  0.9301918
## 33   32 -0.041679275  1.1284790
## 34   33 -0.034956677  0.6628663
## 35   34  0.064679043  1.0942413
## 36   35  0.055652149  1.1735840
## 37   36  0.005539468  1.1560392
## 38   37 -0.129487327  1.1679604
## 39   38 -0.044557317  0.7918385
## 40   39 -0.141210965  0.9418879
## 41   40  0.005611925  0.9046201
## 42   41  0.071771405  1.1515589
## 43   42 -0.116244327  0.9487240
## 44   43  0.175840603  1.1604107
## 45   44  0.020724872  1.2416833
## 46   45 -0.077539463  0.8851223
```

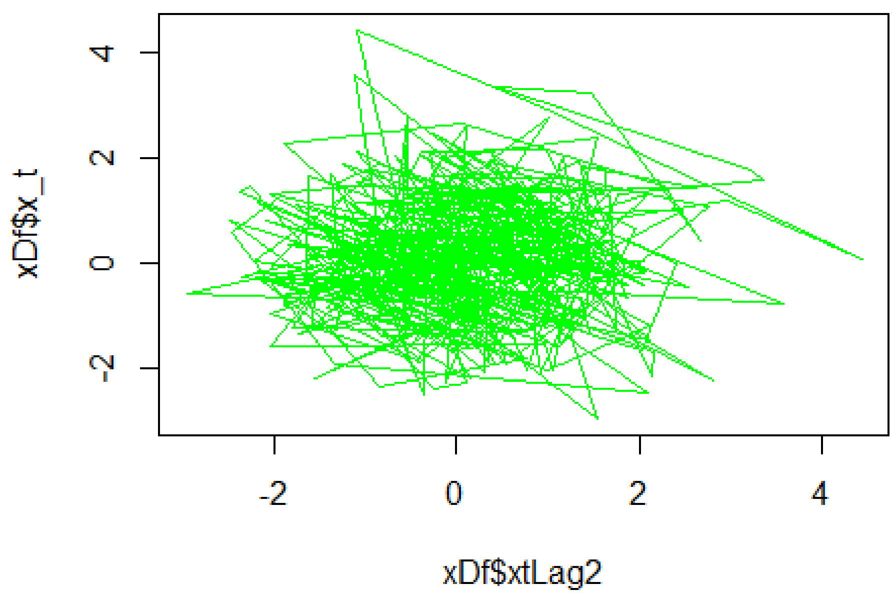
```
## 47   46 -0.038221744  1.0742136
## 48   47 -0.014860019  0.9702879
## 49   48 -0.120173449  0.9820175
## 50   49 -0.012715913  0.9809343
## 51   50  0.117150335  0.9150204
## 52   51  0.034979757  1.0547695
## 53   52  0.123663073  1.0614515
## 54   53  0.006999810  1.0569681
## 55   54  0.166663398  0.8890923
## 56   55 -0.046258171  1.2641023
## 57   56  0.132002608  0.9765072
## 58   57 -0.042695067  0.9896023
## 59   58  0.077862037  1.0857133
## 60   59 -0.034605222  1.1738903
## 61   60  0.145277186  1.2793864
## 62   61  0.033707405  1.1090482
## 63   62  0.195509850  1.0974766
## 64   63 -0.126678605  0.9167719
## 65   64 -0.055785210  1.0206415
## 66   65  0.055825499  0.8452546
## 67   66 -0.085329348  0.9183706
## 68   67 -0.014753657  0.9528680
## 69   68 -0.072452662  1.0147472
## 70   69 -0.098291081  0.8130631
## 71   70  0.074399497  1.2805482
## 72   71 -0.104855506  0.9856815
## 73   72 -0.084257588  1.1838375
## 74   73 -0.119737529  1.1870214
## 75   74  0.186774582  0.8001315
## 76   75 -0.118403197  0.8299422
## 77   76 -0.057542726  1.2306106
## 78   77  0.088560783  1.4038370
## 79   78  0.115438209  1.1593604
## 80   79  0.106934181  0.7840869
## 81   80 -0.045269243  0.9464801
## 82   81 -0.150411454  0.9587518
## 83   82  0.020223791  0.9795977
## 84   83 -0.054724588  1.2870111
## 85   84 -0.067259279  0.9840246
## 86   85 -0.212485035  1.3954972
## 87   86  0.139824869  1.2021613
## 88   87  0.120044159  0.8053154
## 89   88 -0.061009633  0.9568978
## 90   89 -0.014523167  1.0756734
## 91   90 -0.098252315  1.2823730
## 92   91 -0.039016455  0.8560265
## 93   92  0.038444447  1.1096606
## 94   93 -0.137374953  1.1939411
## 95   94 -0.112546637  0.9600341
## 96   95  0.033969113  0.9968530
```

```
## 97   96 -0.034213487  1.1787048
## 98   97  0.092634243  1.0904117
## 99   98 -0.011250825  1.2542322
## 100  99 -0.010116018  0.9480818
## 101 100  0.057754055  1.1070505

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

## [1] 0.002564048

varX <- var(mainDf1$meanVal)
varX

## [1] 0.00874305

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

## [1] 1.031002

varVar <- var(mainDf1$meanVarVal)
varVar

## [1] 0.02360465

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

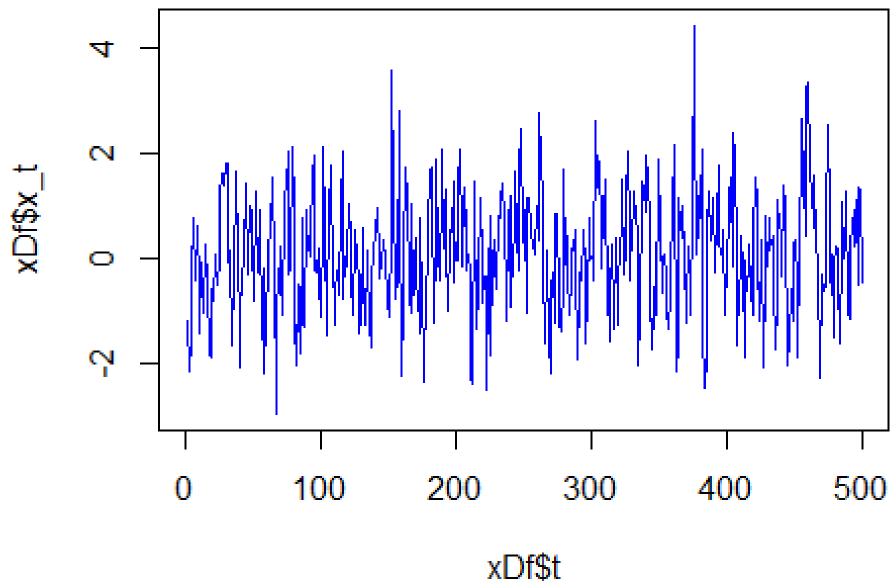
```
# Create epsilon_t-1
mainDf <- slide(mainDf,"e_t", "t", NewVar="etLag1", slideBy = -1)

##
## 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")
```



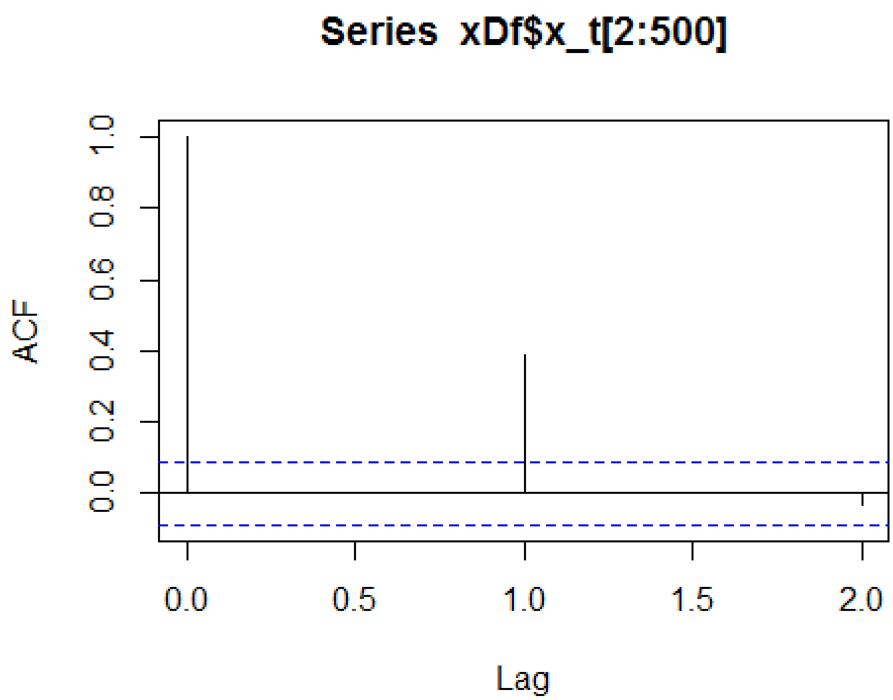

```r
# Line plot x_t vs x_t-2
plot(xDf$xtLag2, xDf$x_t, type = "l", col = "green")
```



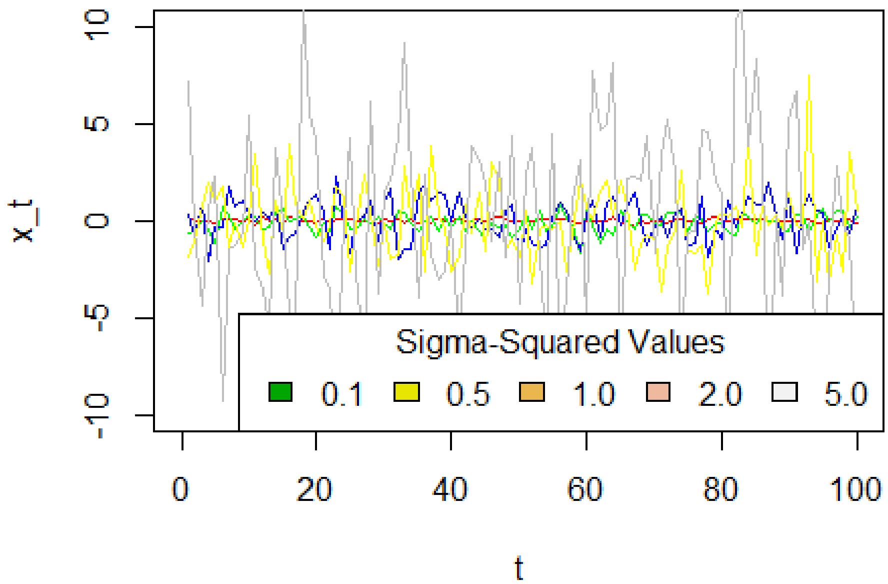
```r
# Line plot x_t vs t
plot(xDf$t, xDf$x_t, type = "l", col = "blue")
```




```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(0.1, 0.5, 1.0, 2.0, 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(-
10,10))
  }
  else{
    lines(t, x_t, col= colVar[s], xlim=c(0,length(t)), ylim=c(-10,10))
  }
  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
##########################################################################
#################################
```