M12: Scenario Generation

Luana Lima

04/01/2021

Setting R code chunk options

First R code chunk is used for setting the options for all R code chunks. The choice echo=TRUE means both code and output will appear on report, include = FALSE neither code nor output is printed.

Loading packages and initializing

Second R code chunk is for loading packages. By setting message = FALSE and warning = FALSE, the code will appear but it will node include messages and warnings.

```
library(lubridate)
library(ggplot2)
library(forecast)
library(Kendall)
library(tseries)
library(outliers)
library(tidyverse)
library(smooth)
library(zoo)
library(kableExtra)
```

Cholesky decomposition in R

Here is an R code chunk that corresponds to the file Choleskydecomp.R that I discussed on video 2.

Recall we have 3 variables that follow a $N(0,1)$ distribution and are highly correlated. We want to make sure that the realizations for all three variable on each scenario makes sense. In other words if they are highly positively correlated, higher values for one leads to higher values for the other two.

```
nvar=3
nscen=1000

#Generate 1000 normal random variates for each variable
X=array(0,c(nvar,nscen))
for(i in 1:nvar){
  X[i,]=rnorm(nscen,mean=0,sd=1)
}

# Alternatively, if you are not a big fun of loops you could do this three times.
#X[1,]=rnorm(nscen,mean=0,sd=1)
#X[2,]=rnorm(nscen,mean=0,sd=1)
#X[3,]=rnorm(nscen,mean=0,sd=1)

#Calculating correlation matrix R
```

```r
Xcor=cor(t(X))

# Note: the t(X) will return the transpose of matrix X.
# We need to transpose so that cor() function finds the correlations
# among the three variable and not among the 1000 scenarios


# Calculating correlation matrix R and Cholesky decomposition R
# Here I am just defining a fictional correlation matrix
# But usually you will get correlation matrix from historial data.

# creating an identity matrix (1 in the principal diagonal, 0 o.w.)
# and order nvar x nvar
R=diag(nvar)
R[1,2]=0.8   #define correlation between variables 1 and 2
R[2,1]=0.8
R[1,3]=0.9
R[3,1]=0.9
R[2,3]=0.85
R[3,2]=0.85

# Get Cholesky decomposition, chol() will give upper triangular matrix
U=chol(R)

# Transpose U to get lower triangular matrix just
L=t(U)

#Passing the correlation matrix R to the the scenarios in matrix X

Y=L%*%X   # the symbol %*% is for matrix multiplication

# Checking if the correlation of generated scenarios matches matrix R
Ycor=cor(t(Y))
print(Ycor) #compare Ycor with R and you will see it worked.
```

```
##           [,1]      [,2]      [,3]
## [1,] 1.0000000 0.7884963 0.9035818
## [2,] 0.7884963 1.0000000 0.8438647
## [3,] 0.9035818 0.8438647 1.0000000
```

Generating scenarios with Arima

The next Rcode chunk corresponds to the file ScenarioGeneration.R from video 3.

```r
#Importing data set with inflow for 15 hydro power plants (HPP)
data=read.table("./Data/inflowtimeseries.txt",header=FALSE,skip=0)

nhydro=3   #choosing to work with only the first three HPP
nobs=nrow(data)

#Creating time series object, note that I will only consider 10 year of data for the analysis just to s
#But you can try to run the code with the full data set
t=829:960 #choosing observations that correspond to Jan 2000 through Dec 2010
inflow_data=ts(log(data[t,3:(nhydro+2)]),start=c(2000,1),end=c(2010,12),frequency=12)
```

```r
#Calculating correlation matrix R and Cholesky decomposition R
R = cor(inflow_data)
U=chol(R) #that will give upper triangular matrix for Cholesky decomposition
L=t(U) #to get lower triangular matrix you need to transpose U, that is what the t() function is doing

#fit the seasonal ARIMA to the each basin
horizon=24  #we want to forecast two years ahead in monthly steps
nscen=10    #number of scenarios to be generated

X=array(0,c(nhydro,horizon,nscen)) #initial array with independently generated scenarios

# Need to do a loop over all HPP under analysis or repeat process 3 times
for(i in 1:nhydro){

  # Fit a SARIMA model
  # Note I am fixing a few parameters regarding the order of the model
  # just to help auto.arima() converge faster

  fit_SARIMA=auto.arima(inflow_data[,i],max.d=1,max.D=1,max.p=1,max.P=1,max.Q=1)

  for_SARIMA=forecast(fit_SARIMA, h=horizon)   #forecast using the fitted SARIMA

  #Generating scenarios
  for(t in 1:horizon){
    # Forecast function does not directly output the standard error other will
    # So I will use the following expression to manually compute sd
    sd=(for_SARIMA$upper[t,1] - for_SARIMA$lower[t,1]) / (2 * qnorm(.5 + for_SARIMA$level[1] / 200))

    # Now that I have mean and standard deviation for time t
    # I can draw scenarios using the rnorm() function
    X[i,t,]=rnorm(nscen,mean=for_SARIMA$mean[t],sd=sd)

    #note this is done in a loop for all the 24 steps we are forecasting
    #and this loop is inside a loop over all HPP

  } # end t loop

  # remove models just to make sure we start from scratch for the next HPP
  # remember we are still inside the HPP loop
  rm(fit_SARIMA, for_SARIMA)

}#end HPP loop

#Creating array Y where we will store correlated scenarios
Y=array(0,c(nhydro,horizon,nscen))

# Need to use another loop structure to make sure spatial correlation among HPP is present in all scena
for(s in 1:nscen){
  aux=exp(X[,,s]) #creating aux variable simple because X is not a 2x2 matrix,
                  #but an array of 3 dimension and we cannot do matrix multiplication with arrays

  Y[,,s]=L%*%aux  #recall L is the Cholesky decomposition of our correlation matrix R computed from wit
```

```r
}#end scenario loop

#Just to illustrate what we have done let's plot the scenarios for the first HPP
i=1
t=1:horizon

#getting min and max values of Y to make sure all scenarios will be within the plot limits
ymax=max(Y[i,,])
ymin=min(Y[i,,])
plot(Y[i,t,1],col="gray",type="l",ylim=c(ymin,ymax),xaxt='n',xlab="") #plotting first scenario
axis(1,at=c(1,13),labels=c("2011","2012"))
for(s in 2:nscen){
  lines(Y[i,,s],col="gray")   #adding lines to the plot corresponding to all scenarios
}
```