```
#################################################
#####   CREATE RANDOM VARIABLES EXERCISES    #####
#####                SOLUTIONS               #####
#################################################
```

 # Generate a binomial $\mathcal{B}in(n,p)$ random variable with
 # n = 25 and p = 0.2. Plot histogram for a simulated
 # sample and compare with the binomial mass function
 # using dbinom() function in R.

 # Try the R code

```
nsim<-5000
n=25;p=.2;
cp=pbinom(c(0:n),n,p)
X=array(0,c(nsim,1))
for(i in 1:nsim){
   u=runif(1)
   X[i]=sum(cp<u)
   }
hist(X,freq=F)
lines(1:n,dbinom(1:n,n,p),lwd=2)
```

 # Use the system.time()
 # function in R to compare your generator with the R
 # binomial generator.

 # To check timing, create the function

```
MYbinom<-function(s0,n0,p0){
  cp=pbinom(c(0:n0),n0,p0)
  X=array(0,c(s0,1))
  for (i in 1:s0){
    u=runif(1)
    X[i]=sum(cp<u)
  }
  return(X)
}
```

```
# and use
system.time(rbinom(500000,25,.2))

#  user  system elapsed
#  0.08    0.00    0.08

# and
system.time(MYbinom(500000,25,.2))

# user  system elapsed
# 3.68    0.00    3.70

# to see how much faster R is.

# For α in the set [0,1] show that the following R
# code produces a random variable U from U([0,α]).

U=runif(1)
while(u > alpha) u=runif(1)
U=u

# Compare it with the transform αU, U ~ U(0,1), for
# values of α close to 0 and close to 1, and with
# runif(1,max=alpha).

# Create the R functions Wait and Trans:

Wait<-function(s0,alpha){
  U=array(0,c(s0,1))
  for (i in 1:s0){
    u=runif(1)
    while (u > alpha) u=runif(1)
    U[i]=u
    }
  return(U)
}

Trans<-function(s0,alpha){
  U=array(0,c(s0,1))
  for (i in 1:s0) U[i]=alpha*runif(1)
```

```
  return(U)
}

# Use
par(mfrow = c(1, 2))
hist(Wait(1000,.5))

# and
hist(Trans(1000,.5))
par(mfrow = c(1,1))

# to see the corresponding histograms.

# Vary n and α. Use the system.time()
# function to see the timing. In particular,
# Wait() is very bad if α is small.

system.time(Wait(10000,.005))
system.time(Trans(10000,.005))
```