CA1: Advanced analytics

Anastasia Kasara

15 March 2018

## Question 1

## (1a)

The probability model corresponding for each agent is shown in table

* Agent 1: Poisson, defined with one parameter (average number of customers ):
* Agent 2: Exponential, defined with one parameter (average service rate )
* Agent 3: Binomial, defined with 3 parametes: the sample size, the probability of succes and the number of outcomes.

## (1b)

The probability, that the third agent will report more than 2 customers purchased the product, is calculated as below. Since in the present case the probability model is binomial, with to buy the product and the sample size (number of customers in the sell zone), then to have 3 or more successes we need to sum up all the probabilities for :

where represents the binomial probability. For the summations the ‘dbinom’ function is utilized:

# (question 1b)  
p=.4 # probability for product purchase  
n=20 # number of customers  
# probability for more than 2 reports  
p2 = 1-pbinom(2,n,p) #

## (1c)

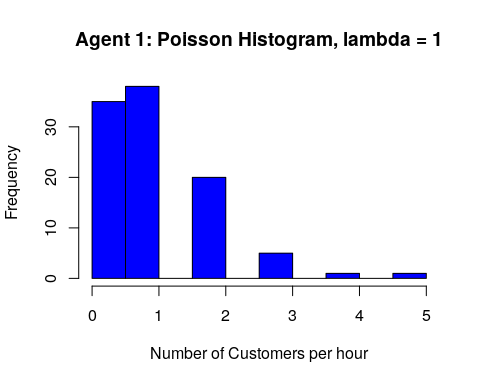
In the below 100 samples are generated from each of the above models (Poisson, Exponential, Binomial). We need to specify the corresponding parameters each time. So we set,

s.poisson <- rpois(n=100, lambda = 1) # 100 samples of a1 (Poisson)  
s.exp <- rexp(n=100, rate = 1) # 100 samples of a2 (exponential)  
# 100 samples of a3 (binom), number of customer = 30  
s.binom <-rbinom(n=100, size=30, p = 0.4)

## (1d)

– agent 1: Poisson, histogram plot and summary

hist(s.poisson, xlab="Number of Customers per hour", col = "blue", main="Agent 1: Poisson Histogram, lambda = 1" ) # histogram



summ.poisson <- summary(s.poisson) # summary  
var.poisson <- var(s.poisson) # variance  
print(summ.poisson)

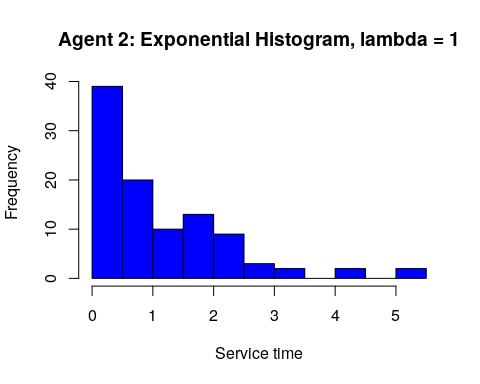
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 0.00 1.00 1.02 2.00 5.00

print(var.poisson)

## [1] 1.009697

– Agent 2: Exponential, histogram plot and summary

# agent 2: exponential, histogram, summary and var  
hist(s.exp, xlab="Service time", col = "blue", main="Agent 2: Exponential Histogram, lambda = 1") # histogram



summ.exp <- summary(s.exp) # summary  
var.exp <- var(s.exp) # variance  
print(summ.exp)

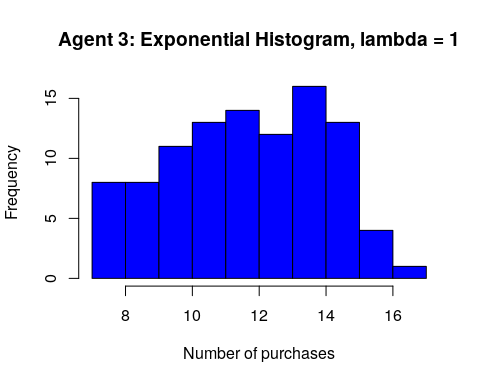
## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.000861 0.311425 0.761887 1.121638 1.764211 5.309253

print(var.exp)

## [1] 1.20768

– Agent 3: Binomial, histogram plot and summary

# agent 3: binom, histogram and summary  
hist(s.binom, xlab="Number of purchases", col = "blue", main="Agent 3: Exponential Histogram, lambda = 1") # histogram



summ.binom <- summary(s.binom) # summary  
print(summ.binom) # variance

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 7.00 10.00 12.00 12.11 14.00 17.00

## Question 2

## (2a)

Assuming that the sensor’s security probability model is the Beta function, the the expectation value, () and the variance, () are given by,

Next I take as probability failure for each individual sensor the expectation value, . For the given 4 sensors, complete failure exists only when the 3 or 4 sensors fail. So the probability for failures should be calculated. The probability for each sensor is binomial, therefore,

The last summation is performed by using ‘pbinom’ from the R. Below is the corresponding code.

##############################  
#   
# Beta distribution with a = 4, b=8, Beta(4,8)  
  
  
# question (2a)  
a<-4  
b<-8  
E<-a/(a+b) # E = 1/3  
varhat<-a\*b/((a+b)^2\*(a+b+1)) # V = 2/117  
print(E)

## [1] 0.3333333

print(varhat)

## [1] 0.01709402

#####  
# from the n = 4 sensors at least k = 2 are secured   
# probability for each individual sensor p = E  
n<-4 # number of sensors  
p<-1/3 # expectation of beta(4,8)  
y<-1-pbinom(2,n,p) # probability that at ;east 2 are secured  
print(y)

## [1] 0.1111111

## (2(b-c))

In the below 25 samples for each sensor are generated using the ‘rbinom’ function. After calculating the mean value of each of the random samples the preferred sensor in terms of security is the one with smaller variance. The behaviour of this sensor is more ‘predictable’ than the others as it will vary less around it’s mean value probability (being chosen to be ). The R code where the random samples are generated are shown below.

# question (2b)  
s <- 1  
n <- 25  
p<- 1/3   
  
  
b1 <-rbinom(n,s,p) # sensor 1  
b2 <-rbinom(n,s,p) # sensor 2  
b3 <-rbinom(n,s,p) # sensor 3  
b4 <-rbinom(n,s,p) # sensor 4  
  
mn1 <- mean(b1)   
vr1 <- var(b1)  
print(mn1)

## [1] 0.4

print(vr1)

## [1] 0.25

#  
mn2 <- mean(b2)  
vr2 <- var(b2)  
print(mn1)

## [1] 0.4

print(vr1)

## [1] 0.25

#  
mn3 <- mean(b3)  
vr3 <- var(b3)  
print(mn3)

## [1] 0.44

print(vr3)

## [1] 0.2566667

#  
mn4 <- mean(b4)  
vr4 <- var(b4)  
print(mn4)

## [1] 0.4

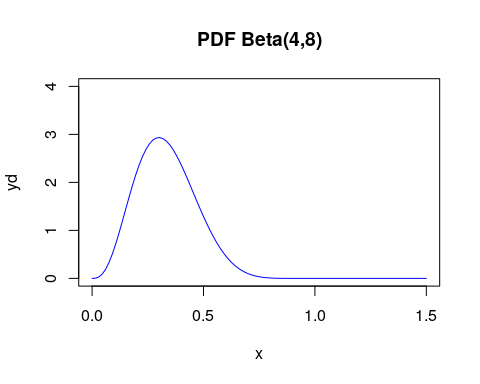
print(vr4)

## [1] 0.25

#  
  
  
  
# (d)  
par(mfrow=c(1,2))  
  
  
plot

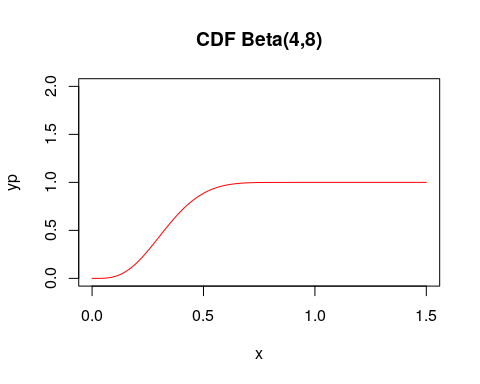
## function (x, y, ...)   
## UseMethod("plot")  
## <bytecode: 0x3bc1888>  
## <environment: namespace:graphics>

x=seq(0,1.5,length=100)  
yd=dbeta(x,4,8)  
plot(x,yd, type="l", col="blue",xlab="x", ylim=c(0,4), main="PDF Beta(4,8)" )



Below is the plot for the CDF of the beta distribution:

x=seq(0,1.5,length=100)  
yp=pbeta(x,4,8)  
plot(x,yp, type="l", col="red",xlab="x", ylim=c(0,2), main="CDF Beta(4,8)" )



## Question 3

Here we need to use the Normal distribution The below is known as the density of the normal distribution, . :

The probability is given by integrating the above probability density from to infinity:

This can be achieved by recalling the ‘pnorm’ R function as below:

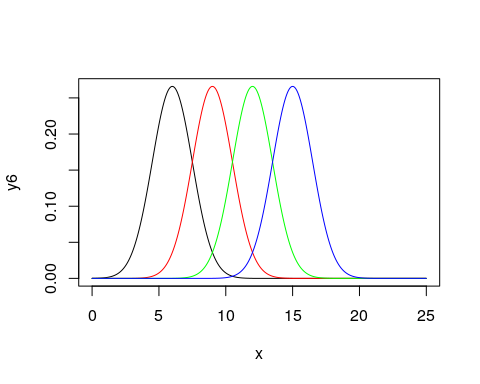
# Normal distribution N(527,112) mean = 527, std = 112  
mu<-527  
sigma<-112  
p1<-1-pnorm(500,mu,sigma) # probability to score above 500  
print(p1)

## [1] 0.5952501

## (3a)

Next we generate 4 plots with different means but same standard variance. The chosen means are and 15. As standard deviation the value was chosen.

# question (3a) plot  
  
x <- seq(0,25,length=2000)  
y6 <- dnorm(x,mean=6, sd=1.5)  
y9 <- dnorm(x,mean=9, sd=1.5)  
y12 <- dnorm(x,mean=12, sd=1.5)  
y15 <- dnorm(x,mean=15, sd=1.5)  
  
plot( x,y6,type="l", xlab="x", col="black")  
lines(x,y9,type="l", xlab="x",col="red" )  
lines(x,y12,type="l", xlab="x",col="green")  
lines(x,y15,type="l", xlab="x",col="blue" )



## (3b)

The probability for an individual scoring between and is given by,

Similarly, for the above the ‘pnorm’ function can be used as below.

# question (3b)   
# probability for GMAT score between 400 and 500  
p2 = pnorm(500,mu,sigma) - pnorm(400,mu,sigma)  
p2

## [1] 0.2763376

## Question 4

The proper probability model for the present case is the multinomial probability for 4 different outcomes of equal probability, so that . The expression for the multinomial probability is.

## (4a)

From a sample of size we require successes for the first feature (with probability ) and anything else. So, the other 3 features have probability . We can then use the binomial probability with , and , to obtain:

This is a very small probability. The relevant code is below:

p1<- 1/4  
n <- 10  
xi<- 9  
p1<-dbinom(xi,n,p1)  
print(p1)

## [1] 2.861023e-05

## (4b)

The correlation betweem the features ‘1’ and ‘2’ may be found from the following relation: The correlation between two random variables is defined as,

where are the corresponding and is the . But for the multinomial distribution:

The above formulas are calculated as below () and :

n<-4  
p<-1/4  
c12 <- -n\*p\*p  
v1 <- n\*p\*(1-p)  
v2 <- n\*p\*(1-p)  
r12 <-c12/(sqrt(v1\*v2))  
print(r12)

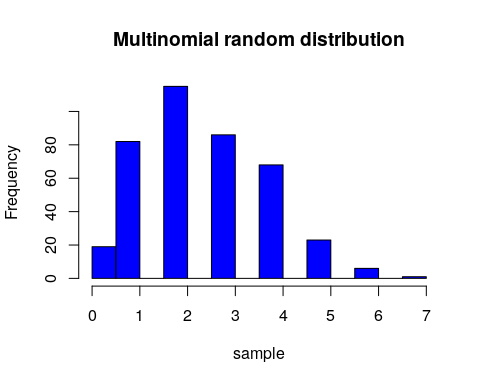
## [1] -0.3333333

## Simulation

Simulating 100 samples to find numerically the expected correlation. For this the use of ‘rmultinom’ provides the sample random vectors with probabilities distributed according the . The size of the sample is and . After we have generated the samples a matrix is created. The rows of length () contains the distributions over the 4 features while the columns of length () are the different samples. As it is more natural to have the samples as rows and the features as columns, I take the transpose of the generated matrix. To calculate the correlation between the features and the corresponding probabilities are needed ‘p1’ and ‘p2’. For this I first calculate the frequencies for each row as:

and then average over the samples to find the mean probabilities:

# simulation for 100 samples.  
  
pm = c(1/4,1/4,1/4,1/4)  
size = 10  
samples =100  
# assume the number of rows in each feature to be n=20  
s.multi = rmultinom(samples,size,pm)  
hist(s.multi, xlab="sample", col = "blue", main="Multinomial random distribution" ) # histogram



# set as rows the samples (100) and as column the features (4)  
ts.multi<-t(s.multi) # transpose  
#  
table(ts.multi)

## ts.multi  
## 0 1 2 3 4 5 6 7   
## 19 82 115 86 68 23 6 1

#View(ts.multi)  
#print(t)  
  
# First we find the frequencies for each sample (sample =100):  
f1 <- ts.multi[,1]/size # size of f1 should be 100  
f2 <- ts.multi[,2]/size # size of f2 should be 100  
# then we take the average over the samples (100)  
w1 <- mean(f1)  
w2 <- mean(f2)  
  
  
#pp<-c(0.1,0.6,0.3)  
#n<-10  
#k<-200  
#print(rmultinom(10,k,pp))

Having calculated the probabilities the calculation of the correlation of follows, according the expressions above in the text.

c12 <- -n\*w1\*w2 # 1-2   
  
v11 <- n\*w1\*(1-w1) # variance of 1  
v22 <- n\*w2\*(1-w2) # variance of 2  
r12 <- c12/sqrt(v11\*v22) # correlation of 1-2  
print(r12)

## [1] -0.3204547

Note that the model calculated value for the correlation coefficient was to be compared with the numerical above.