1 El método

Sea X una variable aleatoria con función de densidad f definida en $[a,b]$, sea $c := sup\{f(x) : x \in [a,b]\}$
Podemos generar una realización de $X \sim f$ con los siguientes pasos:

1. Generar un número aleatorio X $\sim$ Uniforme(a,b)
2. Generar Y $\sim$ Uniforme(0,c)
3. Si Y $\leq$ f(X) regresa X, si no vuelve al paso 1.

2 Beta

La variable aleatoria Beta tiene la siguiente función de densidad:

$$f(x;\alpha,\beta) = \frac{x^{\alpha-1}(1-x)^{\beta-1}}{B(\alpha,\beta)} 1_{0\leq x\leq 1}$$

Donde 1 es la función indicadora y $B(\alpha,\beta)$ es la función Beta que tiene la siguiente propiedad:

$$B(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

3 Ejercicios

Para esta práctica deberán utilizar el método de aceptación rechazo para simular variables Beta(2,2) y Beta(2,5)

De la misma manera que su práctica 5 deberán agregar a su reporte un histograma y su función de densidad. Escojan un n y el parámetro *break* de manera adecuada(figura de la derecha).
Tambien deberán estimar el número de iteraciones promedio que hay que realizar para poder simular n variables Beta, esta información debe reportarse con una gráfica, en el eje x pongan el número de variables Beta que se deseaba simular y en el eje y, el número promedio de iteraciones.

Probar con 7 valores distintos de n, por ejemplo, 10,100,200,500,1000,2000,3000.
¿Cúal crees que sea la complejidad del Algoritmo?

Finalmente agregar un breve comentario que explique que restricciones hay que poner sobre α,β para que $f(x;\alpha,\beta)$ este bien definida. Y ¿A qué variable aleatoria se reduce la $Beta(\alpha,\beta)$ cuando $\alpha = 1$, $\beta = 1$?

Notas:
La función $beta(\alpha,\beta)$ ya esta implementada en R y pueden usarla, pero la densidad si tienen que programarla (aunque ya este implementada).

Para la parte de las iteraciones promedio durante la clase prueben con n pequeñas, pero para hacer el reporte usen n grandes.

Metodo de aceptacion rechazo

El metodo de aceptación rechazo consiste en generar un punto aleatorio distribuido uniformente en la distribución de la cual queremos generar variables aleatorias. Nuestra método esta acotado con el rango de la distribución beta.

```
#' Metodo aceptacin-rechazo
#'
#' @param a limite inferior
#' @param b limite superior
#' @param c numero que acota nuestra funcin en altura.
#' @param alfa parmetros de forma.
#' @param beta parmetros de forma.
#'
#' @return un vector con la va e iteraciones emitidas.
metodo_acep_rechazo= function(a,b,c, alfa, beta){
  iter <- 0
  while(1){
    iter <- iter +1
    varX <- runif(1,a,b)
    varY <- runif(1,0,c)
    if(varY <= densidad_beta(varX, alfa, beta)){
      return(c(varX,iter))
    }
  }
}
```

El algoritmo consiste en generar las coordenadas de un punto aleatoria y verificar si ese punto se encuentra adentro de la distribución, para eso hacemos una condicional y si cumple nuestra condicion de que este dentro, devolverá la variable aleatoria que corresponde con el valor de la coordenada x de nuestro punto generado anteriormente.

```
#'  Funcin  de densidad beta.
#'
#' Describe modelos los cuales la v.a representan proporciones o porcentajes.
#'
#' @param x - representa la variable aleatoria.
#' @param alfa - parmetros de forma.
#' @param beta - parmetros de forma.
#'
#' @return - la probabilidad de la v.a.
```

```
densidad_beta= function(x,alfa,beta){
  return((x^(alfa-1)*(1-x)^(beta-1))/beta(alfa,beta))
}
```

Implementamos tambien la funcion de densidad para simular nuestro método, simplemente es escribir la función en terminos de los parámetro de la distribucion.

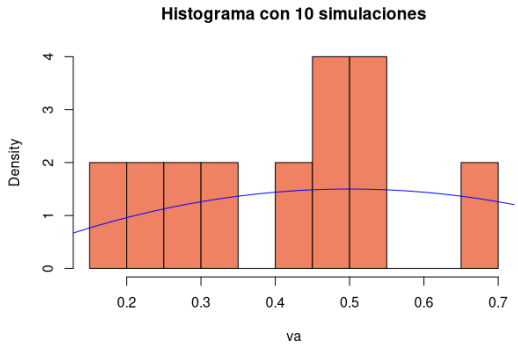
Para nuestra simulación:

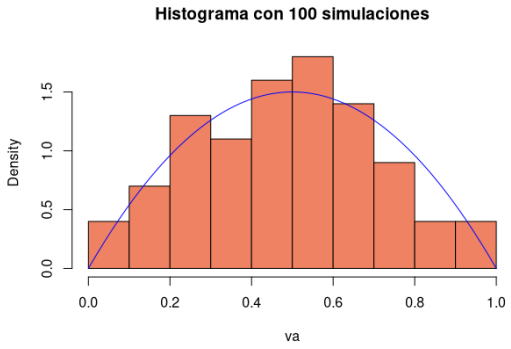
```
#' Simulacin para generar variables aleatorias de distribucion beta.
#'
#' @param n - un entero que indique el numero de variables a simular.
#' @param c - un real que acotara nuestra funcin en altura.
#' @param alfa - parmetros de forma.
#' @param beta - parmetros de forma.
#'
#' @return una lista con las variables aleatorias y numero de intentos.
simular <- function(n, c, alfa, beta){
  va <- c()
  iter <- c()
  c_1 <- densidad_beta(c, alfa, beta)
  for(i in 1:n){
    aux <- metodo_acep_rechazo(0,1, c_1, alfa, beta)
    va [i] <- aux[1]
    iter[i] <- aux[2]
  }
  return(list(va, iter))
}
```

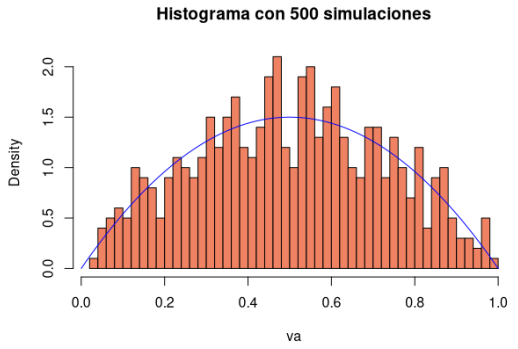
El método simular une todas nuestras funciones para generar n variables aleatorias de nuestra función de distribución beta, consiste en un 'for' de 1 hasta n donde guardaremos cada variable generada por nuestro método de aceptación rechazo al igual que el número de intentos o iteraciones que hubo al encontrar dicha variable.

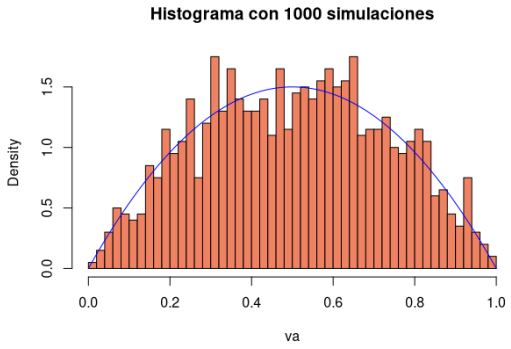
Para graficar el histograma y la función de distribución para distintas n utilizaremos este código que iremos cambiando la n correspondiente, para nuestros primeros histogramas para nuestra distribución beta con parámetos $beta(2,2)$ utilizamos una $c = 1/2$, encontramos este valor con el criterio de la segunda derivada, ya que necesitamos el valor máximo de nuestra función para poder acotar correctamente la altura, para que nuestro punto aleatorio no se aleje demasiado de nuestra distribución.

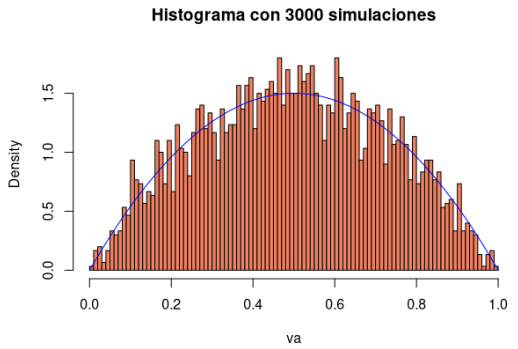
```
#n=10
aux <- simular(10, 1/2, 2, 2)
va <- unlist(aux[1])
hist(va ,probability=TRUE, breaks=10, main="Histograma con 10 simulaciones",col = "salmon2")
par(new=TRUE)
lines(graf_x, graf1_y, col="blue")
par(new=FALSE)
```

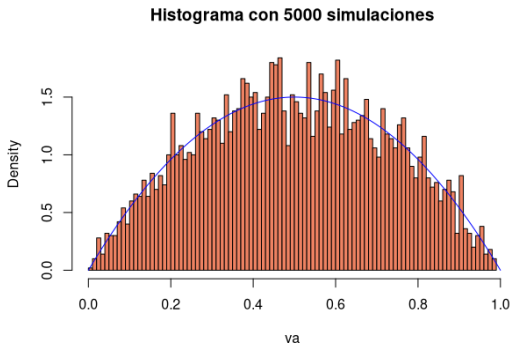

Histograma con 10 simulaciones
Density
va

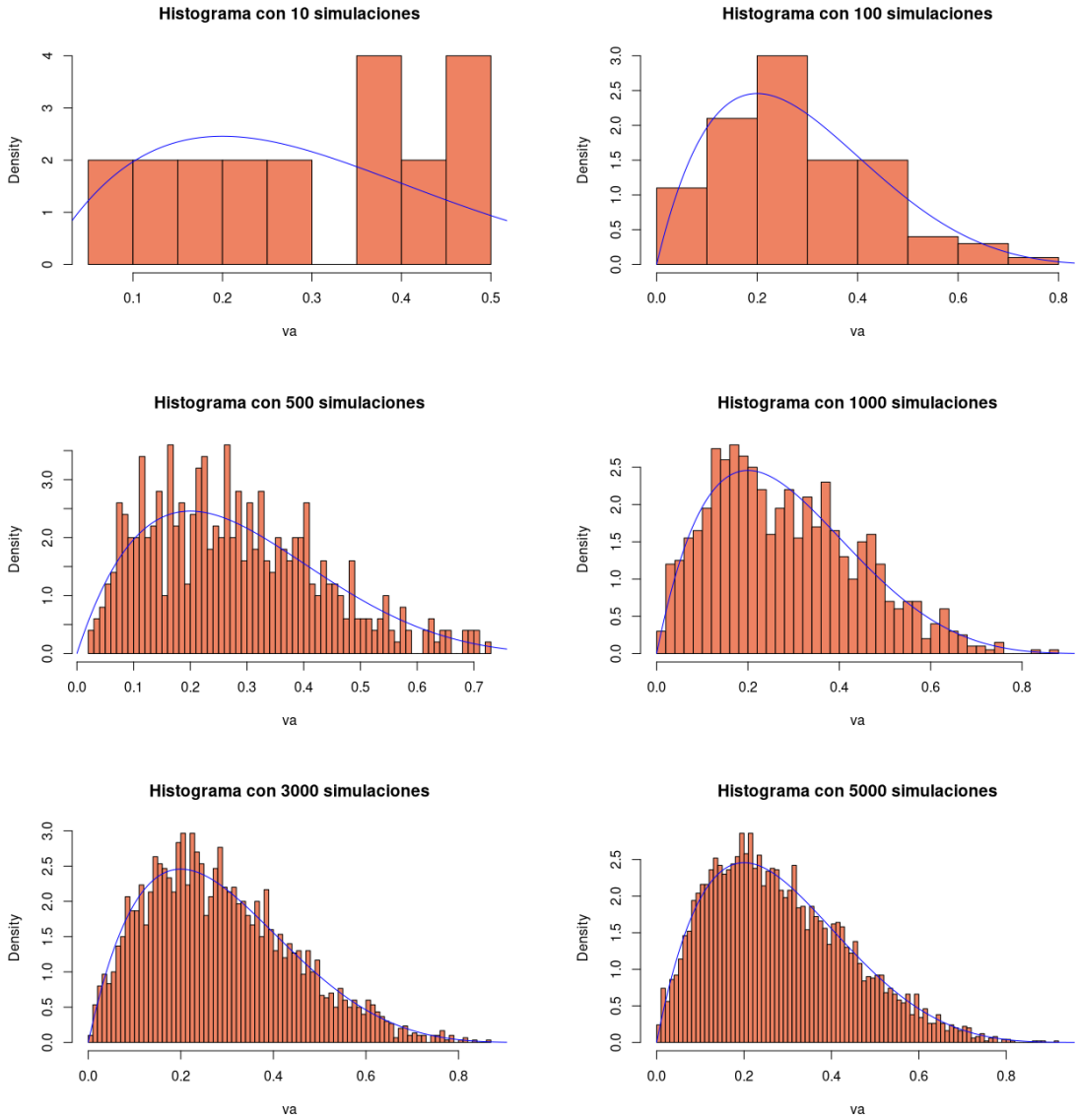

Histograma con 100 simulaciones
Density
va


Histograma con 500 simulaciones
Density
va


Histograma con 1000 simulaciones
Density
va


Histograma con 3000 simulaciones
Density
va


Histograma con 5000 simulaciones
Density
va



Podemos ver claramente que conforme más simulaciones hacemos los valores de nuestra variable aletoria van tomando la forma de nuestra función de distribución, con esto podemos concluir claramente que nuestra simulación de aceptación rechazo es correcta, debido a que la función de probabilidad nos indica la probabilidad de cierta variable aleatoria ocurra, por lo tanto al generar variables aleatorias de la distribución debe coincidir de igual forma con la frecuencia de aparición de cada experimento. Una cosa a recalcar es que si hacemos muchas simulaciones deberíamos estar cada vez más cerca de nuestra linea azul.

La complejidad de nuestro algoritmo del método aceptación-rechazo seria $O(n)$ debido a que solo utilizamos un loop, el cual se detendrá cuando la condición de la altura del punto este dentro del área de nuestra distribución,y en el mejor de los casos seria $O(1)$, esto es si en el primer intento cae dentro y como consecuencia termina nuestra función.

```
1  intentos=seq(1, 1000)
```

```
intentos=intentos
promedios1=c()
promedios2=c()
for(i in 1:1000){
  aux1=simular(i,1/2,2,2)
  intentos1=unlist(aux1[2])
  aux2=simular(i,1/5,2,5)
  intentos2=unlist(aux2[2])
  promedios1[i]=sum(intentos1) /length(intentos1)
  promedios2[i]=sum(intentos2) /length(intentos2)
}
plot(intentos, promedios1,
     type="l", col="red", main="Promedio de intentos al simular n valores",
     xlab="valores simulados", ylab= "promedio de intentos")
par=(new=FALSE)
lines(intentos, promedios2, col="blue")
legend("topleft",
       legend=c("Promedio de intentos para alfa=2,beta=2",
                "Promedio de intentos para alfa=2,beta=5"),col=c("red", "blue"),
       lty = 1:2, cex=0.8)
```

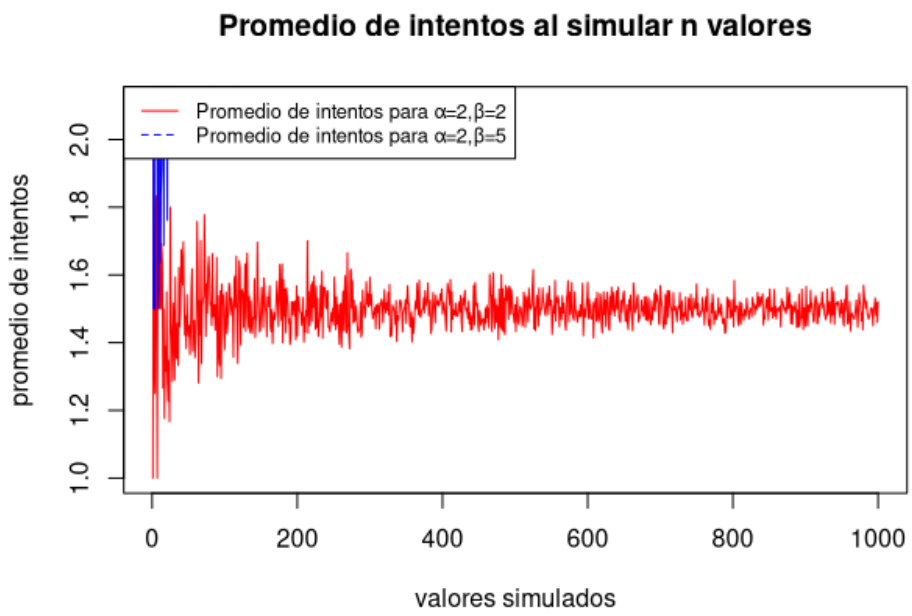


FIG. 1 – *Promedio de intentos al simular n=1000*

Para el promedio de iteraciones o intentos para simular n variables, hicimos un ciclo *for* de 1 hasta 1000, obteniendo por cada simulación el promedio, que seria sumando el numero de intentos que se hizo entre el numero de simulaciones, que corresponde al tamaño de nuestra variable 'intentos' y lo graficamos. Vemos que para los primeros valores el promedio aumenta y disminuye, pero a partir de la simulación con $n = 400$ llegaba con un promedio entre 1.4 y 1.6 de intentos y asi sucesivamente con los demás valores.

Cuando la variable aleatoria *beta* toma como parámetros $\lambda = 1$ y $\beta = 1$ podemos decir que se reduce a una distribución uniforme debido a que si sustituimos los valores, quedaría:

$$beta(x,1,1) = 1$$

La cual nos indica que para cualquier valor que tome la función de distribución su probabilidad sera siempre 1, por lo tanto para cada variable aleatoria tiene la misma probabilidad, gráficamente seria un linea recta que iría de 0 a 1.