Practica 3

Agustín Muñoz González

27/4/2020

Limpiamos registros.

```
rm(list=ls())
```

Tarea 1

Item 1

Queremos definir la función partida **funcion.h** dada por la fórmula. Dado que una función partida está dada por condiciones usaremos la estructura *if* para definirla

```
funcion.h=function(x){
  if(x<10){
    salida=0
  }else if(10<= x & x<=15){
    salida=(x-10)/(15-10)
  }else{salida=1}
  return(salida)
}
# No es necesario poner el return() al final, con poner la variable a mostrar estamos.
```

Item 2

Ahora que definimos la función, la graficamos en [5,20] y exportamos el gráfico a un pdf

```
pdf('Tarea 1-Item 2.pdf')
grilla=seq(5,20,0.01)
y=lapply(grilla,funcion.h)
plot(grilla,y,xlab='x',ylab='h(x)',type='l')
graphics.off()
```

Item 3

Por último queremos reemplazar los valores 10 y 15 por ciertos parámetros A<B a ingresar por el usuario. Con lo cual repetimos la definición anterior pero especificando que la función h requiere dos nuevos parámetros A y B

```
funcion.h.param=function(x,A,B){
  if(x<A){
    salida=0
  }else if(a<=x & x<=B){
    salida=(x-A)/(B-A)
  }else{salida=1}
  return(salida)
}
```

Tarea 2

Item 1

Queremos una función **suma.positivos** que tenga como input un vector v y devuelva el resultado de sumar todos los valores positivos de v. Usaremos por consiguiente las estructuras *for* (para recorrer todo el vector) e *if* para chequear que la componente i-ésima sea positiva.

```
suma.positivos=function(v){
  salida=0
  for (i in v){
    if(i>0){
      salida=salida+i
      }
  }
  return(salida)
}
```

Item 2

Tenemos que calcular la suma de los números naturales comprendidos entre (en mi caso) 868 y 3938 (inclusive ambos)

```
suma.positivos((868:3938))
```

```
## [1] 7379613
```

Item 3

Debemos definir **suma.si.hay.positivos** que tome como input un vector v y devuelva suma.positivos(v) si v tiene componentes positivas y un mensaje 'No hay positivos.' si v no tiene componentes positivas.

```
suma.si.hay.positivos=function(v){
  if(suma.positivos(v)==0){
    return(cat('No hay positivos.'))
  }else{return(suma.positivos(v))}
}
```

Tarea 3

Item 1

Definimos una función **suma.cubilete** que simule arrojar un cubilete y devuelva la suma de las 5 caras obtenidas. Para ello usaremos el comando *sample(vector,size,replace)* que devuelve un vector de tamaño size de elementos de vector elegidos al azar y sin repetición (replace=F) o con repeticion (replace=T), por defecto es sin repetición (i.e. si no ponemos aclaramos nada asume replace=T).

```
suma.cubilete=function(){
  cubilete=sample((1:6),5,replace = T)
  return(sum(cubilete))
}
# Otra posibilidad es sample(6,5). En general, sample(int,size) devuelve una
# muestra al azar de tamaño size del vector (1:int).
```

Item 2

Realizamos un ejecución de la función del item anterior

´´´{r} suma.cubilete() ´´´

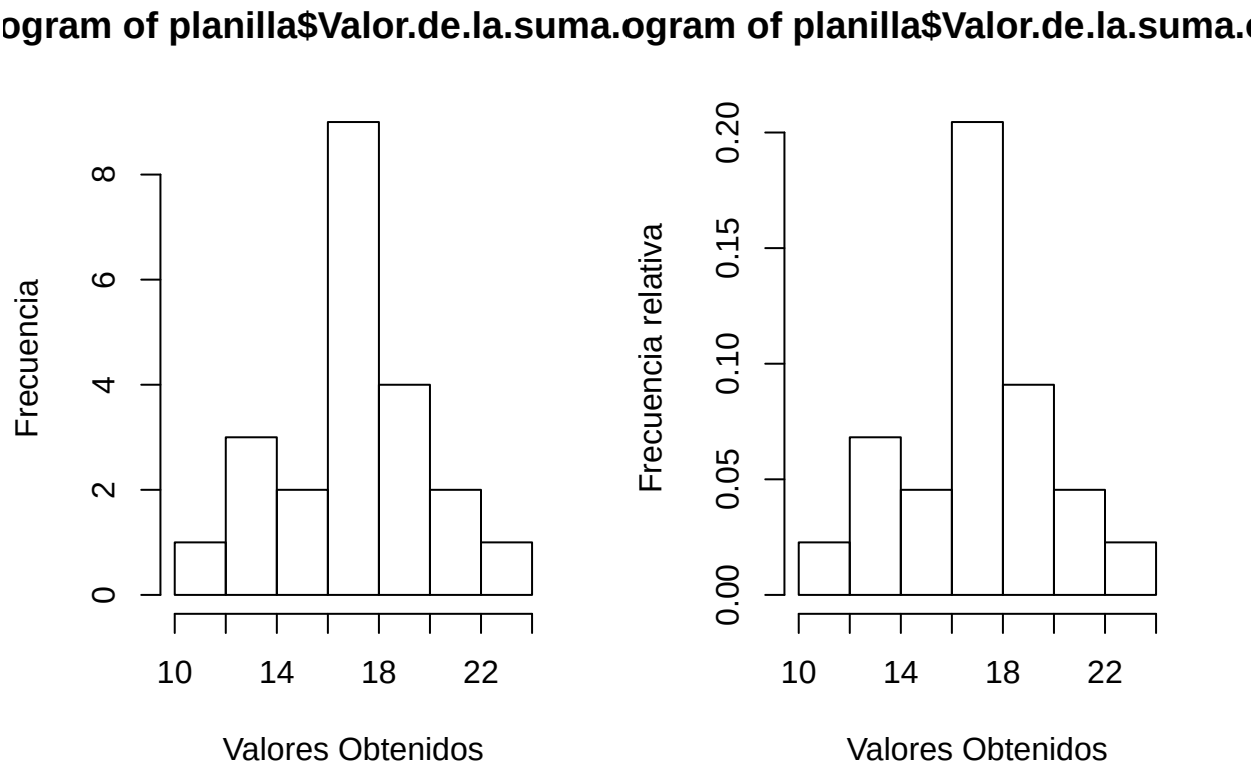
Item 3

Leemos la planilla

```
planilla=read.csv('Tareas_  primeros pasos en programación - Tarea 3_ suma cubilete.csv',header=T)
#cuando terminene de completar la planilla
```

Graficamos un histograma con los valores obtenidos por cada participante.

```
par(mfrow=c(1,2))
hist(planilla$Valor.de.la.suma.obtenido,
     xlab='Valores Obtenidos',
     ylab='Frecuencia')
hist(planilla$Valor.de.la.suma.obtenido,
     freq = F,
     xlab='Valores Obtenidos',
     ylab='Frecuencia relativa')
```



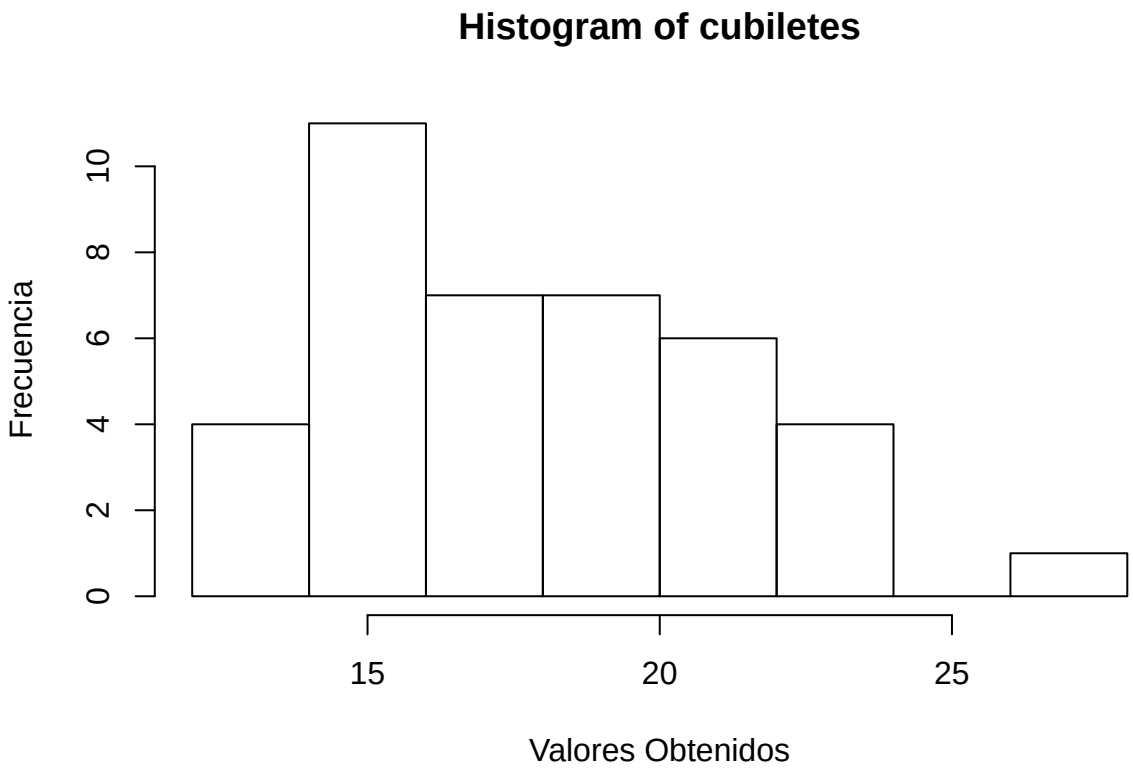

```
     #breaks=44)
# Completar breaks cuando tenga la planilla completa.
par(mfrow=c(1,1))
```

Notemos que tiene una distribución normal.

Item 4

Debemos simular 40 lanzadas de cubilete, calcular la suma y realizar el histograma.

```
i=1
cubiletes=c()
while (i<=40){
  cubiletes=c(cubiletes,suma.cubilete())
  i=i+1
}
hist(cubiletes, xlab='Valores Obtenidos',
     ylab='Frecuencia')
```



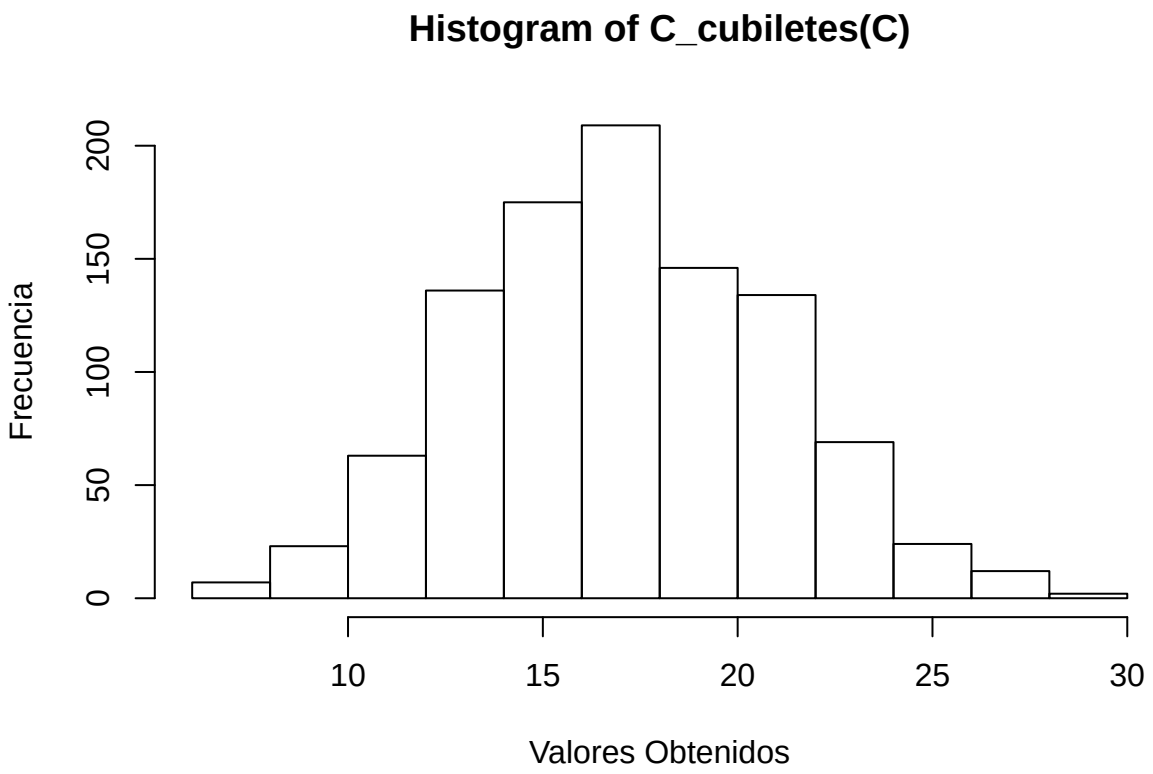
Item 5

En primer lugar vamos a definir una función que simule C lanzadas de cubilete y devuelva un vector formado por C resultados de las sumas de cada cubilete.

```
C_cubiletes=function(C){
  i=1
  salida=c()
  if(C<=0){return(cat('El número C debe ser positivo.'))}
  #else if(is.integer(C)==FALSE){return(cat('El número C debe ser entero.'))}
  else{
  while(i<=C){
    salida=c(salida,suma.cubilete())
    i=i+1
  }
  }
  return(salida)
}
```

Ahora realicemos el gráfico de la imágen de la función anterior para C=1000.

```
#input=readline(prompt = 'Ingrese la cantidad de cubiletes a lanzar: ')
#C=as.integer(input)
C=1000
hist(C_cubiletes(C),xlab='Valores Obtenidos',
     ylab='Frecuencia')
```




```
# Dejo comentado los comandos para leer el valor C como entrada del usuario.
```

Item 6

Debemos implementar una función **todos.seis** que cuente la cantidad necesaria de lanzamientos hasta obtener el valor seis en todas la caras.

Veamos una primera forma usando la función suma.cubilete()

```
todos.seis=function(){
  i=1
  while(suma.cubilete()!=30){
    i=i+1
  }
  return(i)
}
```

La forma de usar la función suma.cubiletes() es iterando esta función hasta que de como resultado 30, en cuyo caso significa que la última iteración dio como resultado el cubilete rep(6,5), como hicimos arriba.

Extra

Veamos otra forma sin usar suma.cubilete() pero usando comandos útiles como *all()* y viendo como se niega un statement

```
todos.seis_2=function(){
  cubilete=sample(6,5,replace = T)
  i=1
  while(!(all(cubilete==rep(6,5)))){
    cubilete=sample(6,5,replace = T)
    i=i+1
  }
  return(i)
}
```

Extra

Defino una función que devuelva la suma del cubilete y el propio cubilete que dio ese resultado. De paso muestro cómo se devuelve una lista.

```
lanzamiento.cubilete=function(){
  cubilete=sample((1:6),5,replace = T)
  return(list("Cubilete"=cubilete,"Suma"=suma.positivos(cubilete)))
}
```