3. Para el siguiente ejercicio es necesario usar R.

a) Considere una moneda desequilibrada que tiene probabilidad p de obtener águila. Usando el comando sample, escriba una función que simule N veces lanzamientos de esta moneda hasta obtener un águila. La función deberá recibir como parámetros a la probabilidad p de obtener águila y al número N de veces que se repite el experimento; y tendrá que regresar un vector de longitud N que contenga el número de lanzamientos hasta obtener un águila en cada uno de los N experimentos.

RESPUESTA
Si X es el número de lanzamientos de la modena hasta obtener un águila, con probabilidad p de obtener águila en un lanzamiento. Entonces, $X \sim Geo(p)$. Por lo que la función que solicitan sería la simulación de X N veces. Ocupando la siguiente notación de 1:águila y 0:sol:

```r
moneda_geometrica <- function(p, N){ # p: probabilidad de aguila, N # repeticiones.
  resultados <- c() # Inicializamos un vector.
  for (i in 1:N) { # Repetimos el experimentos N veces.
    contador <- 0 # Inicializamos el número de lanzamientos.
    while(sample(x=c(1,0), size=1, prob=c(p,1-p))!=1){ # si ya se obtuvo águila deterner.
      contador <- contador + 1
    }
    resultados[i] <- contador
  }
  resultados # regresamos los resultados.
}
```

Observamos que en los incisos siguientes se ocupa esta funcipon para N un poco grandes, por lo que vectorizo la función anterior para tener lo mismo en un tiempo más corto. La diferencia entre estas dos funciones radica basicamente en el *sample*, ya que nosotros simularemos por bloques, es decir, como si estuvieramos muchas modenas lanzandose al mismo tiempo.

```r
moneda_geometrica_optimizada <- function(p, N, potencia){ # potencia: tamaño del bloques.
  resultados <- c() # Inicializamos un vector.
  while(length(resultados)<N) { # Repetimos el hasta tener N resultados
    contador <- 0 # Inicializamos el número de lanzamientos.
    resultados_preliminar <- c() # Inicializamos los resultados por bloques.
    while(length(resultados_preliminar)<potencia){
      contador_s<- sum(sample(x=c(1,0), size=potencia-length(resultados_preliminar),
                              prob=c(p,1-p), replace=TRUE))
      contador <- contador + 1
     resultados_preliminar<- c(resultados_preliminar, rep(contador, contador_s)) # Concatenamos l
    }
    resultados <- c(resultados, resultados_preliminar) # Concatenamos los resultados por bloques
  }
  resultados # regresamos los resultados.
}
```

Donde el parámetro potencia representa el tamaño del bloque, es decir, cuantas monedas se lanzarán al mismo tiempo. Algo curioso de este parámetro por intución entre más grande sea más rápido será, pero no es así aunque no estoy muy seguro por que sucede.

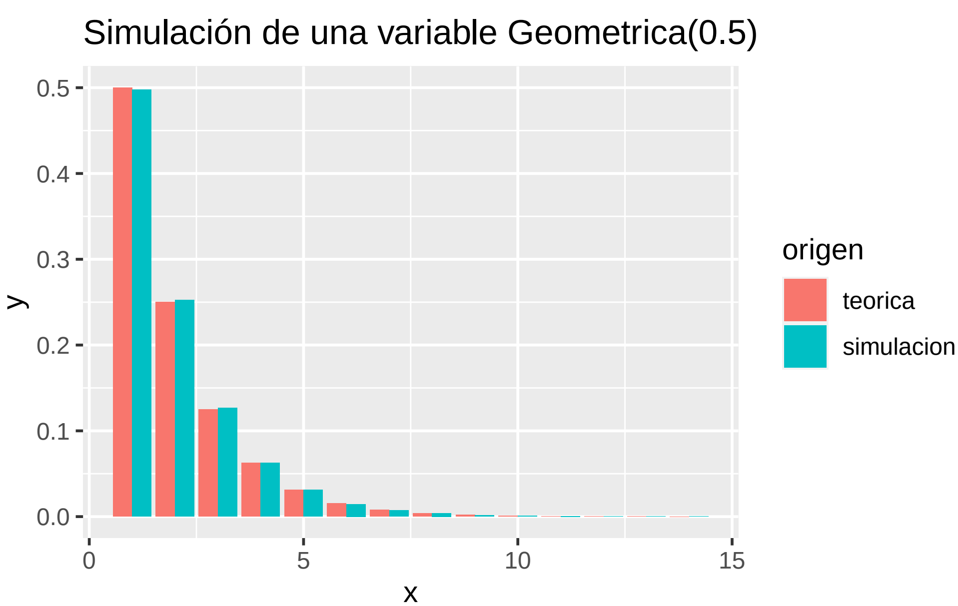
b) Usando la función anterior simule $N = 10^4$ veces una variable aleatoria Geom(p) para $p = 0.5, 0.1, 0.01$. Grafique las frecuencias normalizadas en color azul. Sobre está última figura empalme en rojo la gráfica de la función de masa correspondiente. ¿Qué observa?

RESPUESTA
Creemos otra función que utilice la función del inciso a) y que grafique las frecuencias normalizadas en azul y en rojo las frecuencias obtenidas de función de distribución de un variable Geometrica.

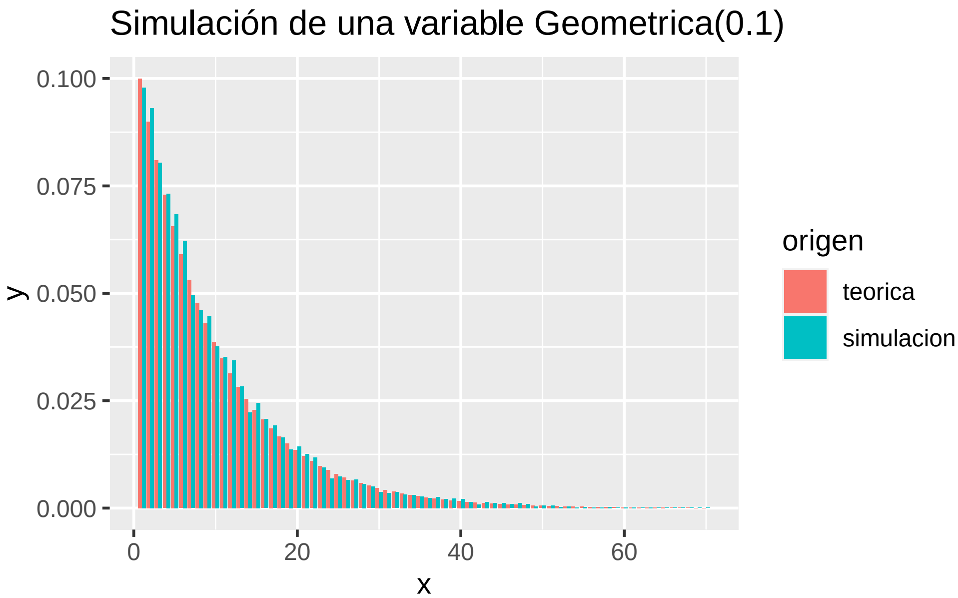
```r
library(tidyverse) # ggplot and dplyr
geometric_graph_simula_and_teoric <- function(p, N, potencia, titulo){
  # Utilizamos la opción del inciso a).
  simular_geometrica <- data.frame(resultado=moneda_geometrica_optimizada(p, N, potencia))
  # Generamos las frecuenciass normalizadas.
  simular_geometrica <- data.frame(table(simular_geometrica)/N)
  names(simular_geometrica) <- c("x", "y")
  simular_geometrica$x <- as.numeric(simular_geometrica$x)
  # Variable auxiliar.
  simular_geometrica$origen <- "simulacion"
  max_resul <- max(simular_geometrica$x)
  # Función de distribución utilizando la formula.
  teoric_geometrica <- data.frame(x=seq(1,max_resul,1),
                                  y=dgeom(x=seq(0,(max_resul-1),1),

  # Concatenamos las frecuencias obtenidas.
  geometrica <- rbind(teoric_geometrica, simular_geometrica)
  # Graficamos
  g <- ggplot(geometrica, mapping=aes(x,y,fill=origen))+
    geom_histogram(position="dodge", stat="identity", bins = max_resul)+
    labs(title=titulo)
  return(g)
}
```

Por lo que las gráficas variando el parámetro p son

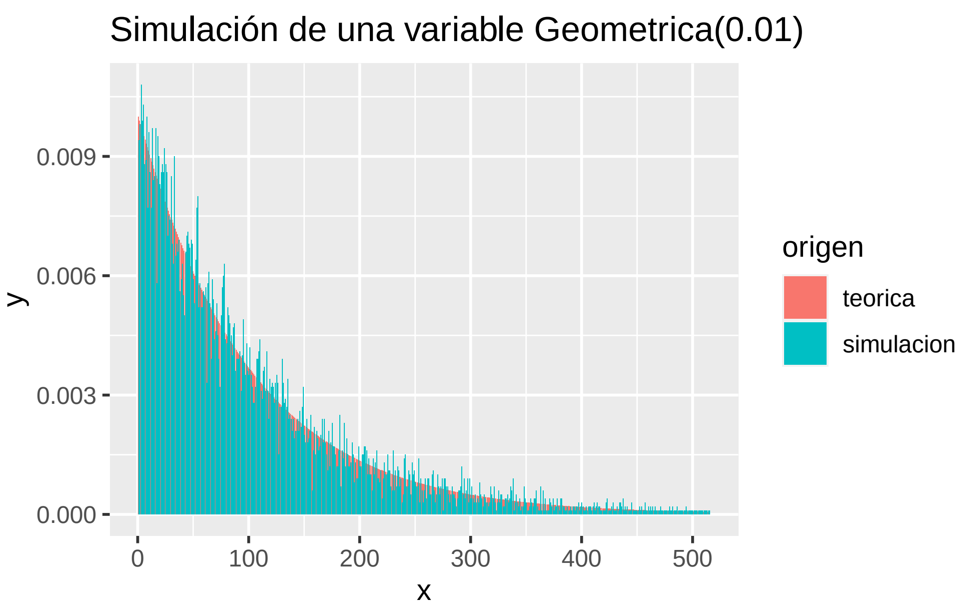
```r
set.seed(08081997)
geometric_graph_simula_and_teoric(0.5, 10^4, 10^4,
                                  "Simulación de una variable Geometrica(0.5)")
```



```r
geometric_graph_simula_and_teoric(0.1, 10^4, 10^4,
                                  "Simulación de una variable Geometrica(0.1)")
```



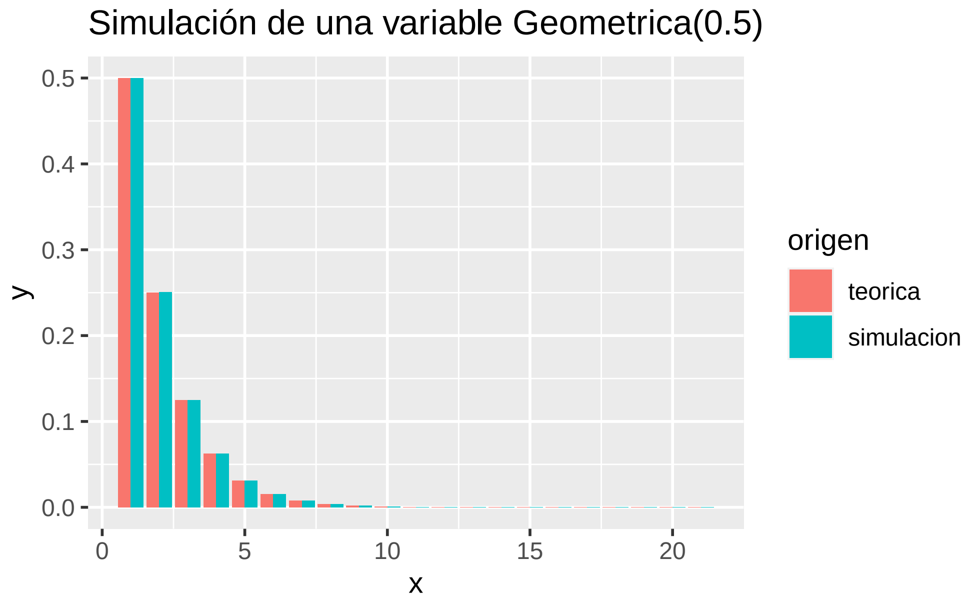

```r
geometric_graph_simula_and_teoric(0.01, 10^4, 10^4,
                                  "Simulación de una variable Geometrica(0.01)")
```



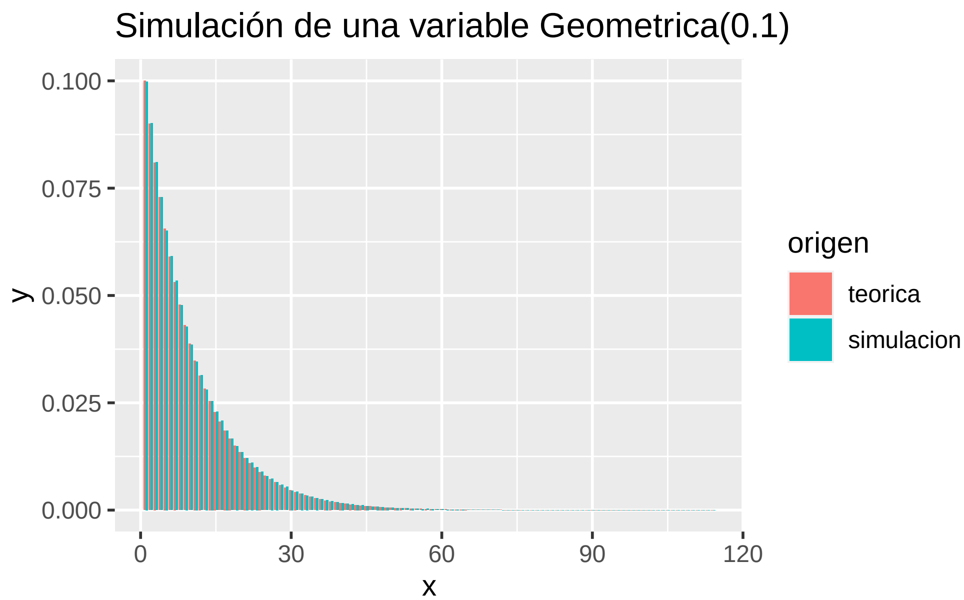
Observemos que si comparamos las frecuencias de las simulaciones y las frecuencias obtenidas de la función de probabilidad de una geometrica se ven muy cercanas. Pero conforme p se acerca a 0 la comparaciones entre estas frecuencias son más notorias. Esto se puede explicar debido a que cuando p es más chico la $\mathbb{P}(X = x)$ se va hacieno más pequeña, por lo que x toma un rango más amplio de valores posibles. No hay que confundirse por el hecho de que como la función de distribución de una variable aleatoria geometrica esta defina en todos los naturales. Ya que si p es cercano a 1, las probabilidades convergen más rapido a 0, y viceversa, si p es cercano a 0 las probabilidad convergen más lento a 0.

c) Repita el inciso anterior para $N = 10^6$. Además calcule el promedio y la desviación estándar de las simulaciones que realizó ¿Qué observa?

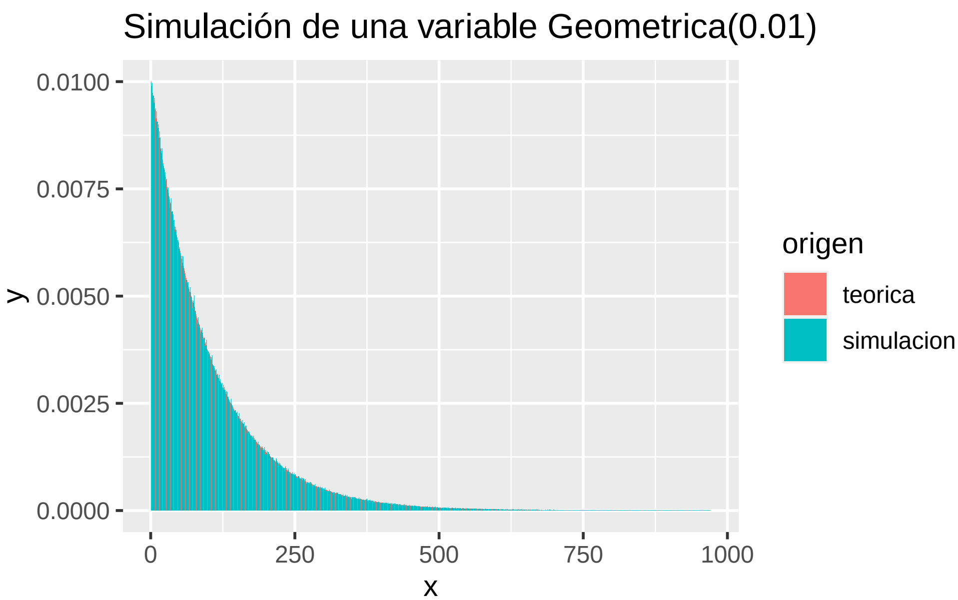
```r
set.seed(08081997)
geometric_graph_simula_and_teoric(0.5, 10^6, 10^5,
"Simulación de una variable Geometrica(0.5)")
```




```r
geometric_graph_simula_and_teoric(0.1, 10^6, 10^5,
"Simulación de una variable Geometrica(0.1)")
```




```r
geometric_graph_simula_and_teoric(0.01, 10^6, 10^5,
"Simulación de una variable Geometrica(0.01)")
```



Cómo el número de simulaciones son mayores que el inciso anterior, observamos que las diferencias se entre las frecuencias simuladas y frecuencias calculados son muy cercanas "casi nulas". Y esto incita a concluir que la distribución Geometrica modela bien este esperimento de lanzamiento de monedas. ■.

4. Usando las ideas del inciso anterior escriba una función en R que simule N veces los lanzamientos de moneda hasta obtener r águilas. La función deberá recibir como parámetros a la probabilidad p de obtener águila, al número r de águilas a observar antes de detener el experimento y al número N de veces que se repite el experimento; y tendrá que regresar un vector de longitud N que contenga el número de lanzamientos hasta obtener las r águilas en cada uno de los N experimentos. Grafique las frecuencias normalizadas de los experimentos para $N = 10^6$, $p = 0.2, 0.1$ y $r = 2, 7$ y compárelos contra la función de masa de la distribución más adecuada para modelar este tipo de experimentos.

RESPUESTA
Sea X el número de lanzamientos hasta obtener r aguilas.

```
moneda_desequilibrada_r_exitos <- function(r, p, N){
  resultados <- c()
  for(i in 1:N){
    contador <- 0
    lanzamiento <- ""
    num_aguilas <- 0
    while(num_aguilas<r){
      lanzamiento <- sample(x=c("aguila", "sol"), size=1, prob=c(p,1-p))
      contador <- contador + 1
      if(lanzamiento=="aguila"){
        num_aguilas<-num_aguilas+1
      }
    }
    resultados[i] <- contador
  }
  resultados
}
```

```
moneda_desequilibrada_r_exitos_optimizada <- function(r, p, N, potencia){
  resultados <- c()
  while(length(resultados)<N) { # Repetimos el experimentos N veces.
    contador <- 0 # Inicializamos el número de lanzamientos.
    resultados_preliminar <- c()
    inicial <- rep(0, potencia)
    while(length(resultados_preliminar)<potencia){ # si ya se obtuvo águila deterner.
      inicial <- inicial + sample(x=c(1,0), size=potencia-length(resultados_preliminar), prob=c(p
      contador_s <- sum(inicial==r)
      contador <- contador + 1
     resultados_preliminar<- c(resultados_preliminar, rep(contador, contador_s))
     inicial <- inicial[inicial<r]
    }
    resultados <- c(resultados, resultados_preliminar)
  }
  resultados # regresamos los resultados.
}
```

7. Escriba una función en R que simule una aproximación al proceso Poisson a partir de las 5 hipótesis que usamos en clase para construir tal proceso. Usando esta función, simule tres trayectorias de un proceso Poisson $\lambda = 2$ sobre el intervalo $[0, 10]$ y grafíquelas. Además simule 10^4 veces un proceso de Poisson N con $\lambda 1/2$ y hasta el tiempo $t = 1$. Haga un histograma de $N(1)$ en su simulación anterior y compare contra la distribución de Poisson correspondiente.

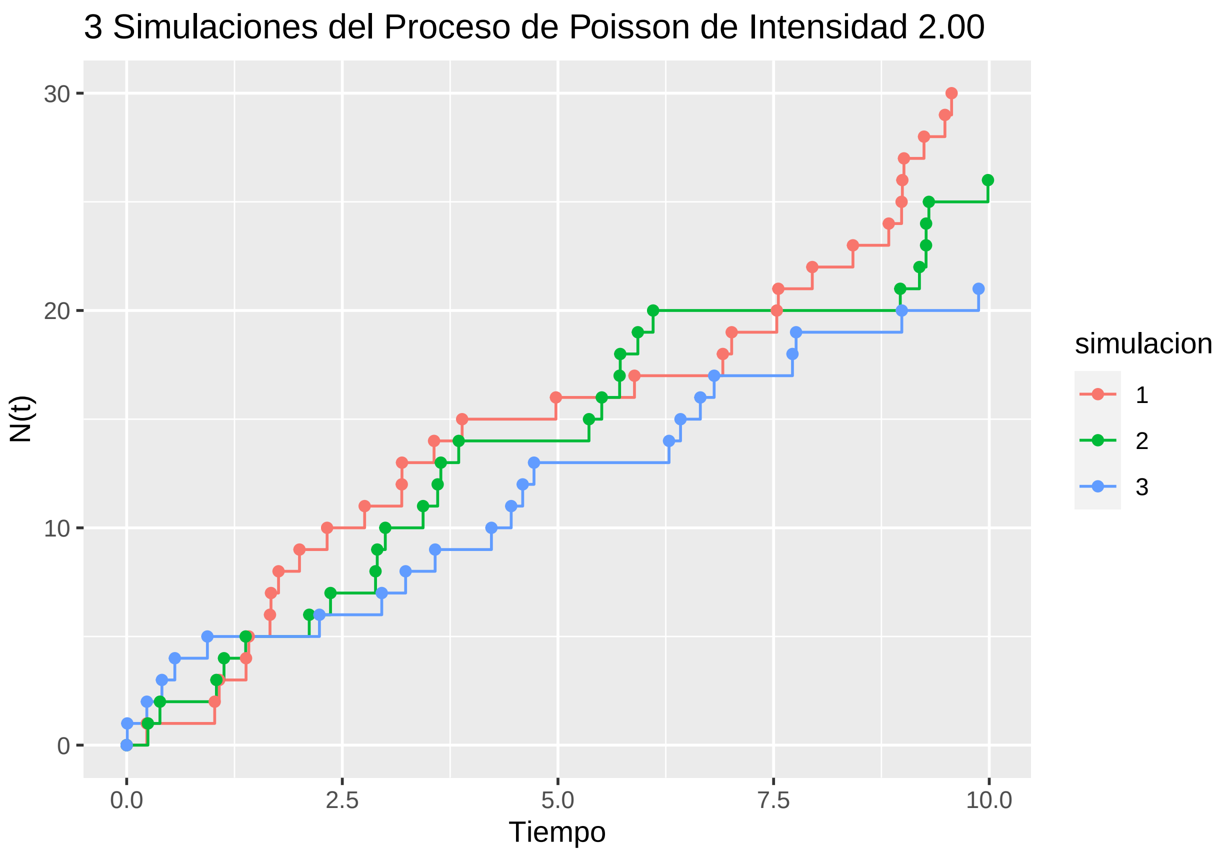
```
ProcesoPois<- function(t,lambda){
  N<- rpois(1,t*lambda) #Paso 1
  C<- sort(runif(N,0,t)) #Paso 2 y 3
  data.frame(x=c(0,0,C),y=c(0,0:N))
}
```

```
library(tidyverse)
library(plyr)
NPois<-function(n,t,rate){
  C<- lapply(1:n, function(n) data.frame(ProcesoPois(t,rate),simulacion=n)) #Genera N dataframes
  C<-ldply(C, data.frame) #Une en una sola dataframe
  C$simulacion<-factor(C$simulacion) #Convierte en factores
  C
}
```

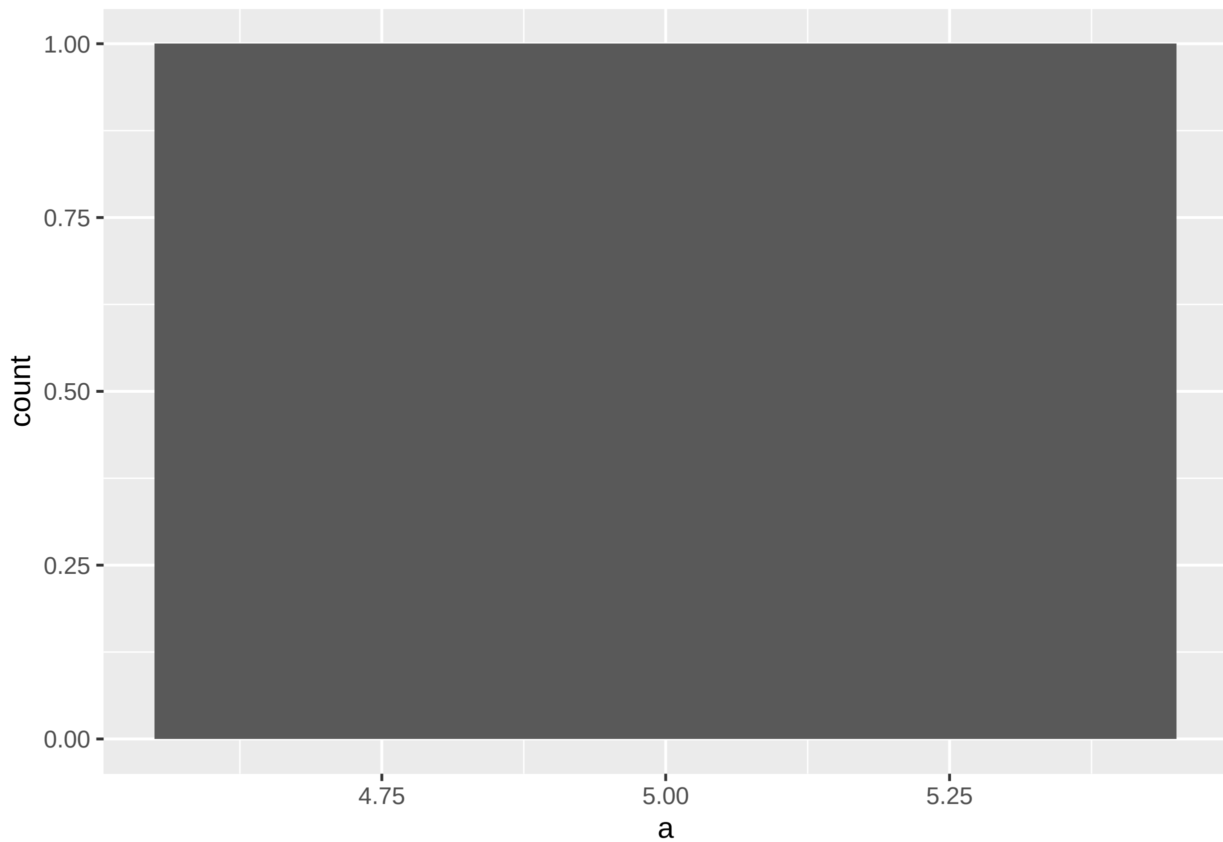
```
simulacion_process_a <- NPois(3,10,2)
head(simulacion_process_a)
```

```
##           x y simulacion
## 1 0.0000000 0          1
## 2 0.0000000 0          1
## 3 0.2331624 1          1
## 4 1.0204764 2          1
## 5 1.0724719 3          1
## 6 1.3837753 4          1
```

```
qplot(x,y,data=simulacion_process_a,geom=c("step","point"),color=simulacion,xlab="Tiempo",ylab="N
```




```r
set.seed(13)
prueba <- NPois(10^4, 1,0.5)

prueba %>% group_by(simulacion) %>% summarise(a=max(y)) %>%
  ggplot(aes(x=a))+geom_bar()
```




```r
dpois(x = 0,lambda = 0.5)
```

```
## [1] 0.6065307
```

```r
dpois(x = 1,lambda = 0.5)
```

```
## [1] 0.3032653
```

```r
dpois(x = 2,lambda = 0.5)
```

```
## [1] 0.07581633
```

```r
dpois(x = 3,lambda = 0.5)
```

```
## [1] 0.01263606
```

```r
dpois(x = 4,lambda = 0.5)
```

```
## [1] 0.001579507
```

```r
dpois(x = 5,lambda = 0.5)
```

```
## [1] 0.0001579507
```

10. **Este es un problema al que se recurrirá en el futuro,** su intención es que empiecen a jugar con datos reales. El archivo Delitos.csv contiene información sobre los delitos denunciados en la ciudad de Aguascalientes, para el período comprendido entre enero de 2011 a junio del 2016. Dicho archivo contiene 5 columnas: la primera columna contiene la fecha de denuncia del delito; la columna TIPO muestra una descripción del tipo de delito; la columna CONCATENAD presenta un descripción más amplia del delito; la columna SEMANA contiene la semana del año a la que corresponde la fecha de denuncia; y la columna SEMANA_COMPLETAS indica la semana a lo largo del estudio en la

cual se presentó la denuncia. A través de métodos gráficos (e.g. boxplots) traten de determinar el comportamiento semanal de los delitos y discutan alternativas de modelos para describir los delitos cometidos en forma relativamente apropiada.

```
# Cargamos las librerias a ocupar.
library(tidyverse)
library(lubridate)

# Leamos los datos.
df_delitos <- read.csv(file = "Delitos.csv")
```

Conozcamos un poco los datos.

```
names(df_delitos)
```

```
## [1] "FECHA"            "TIPO"             "CONCATENAD"       "SEMANA"
## [5] "SEMANA_COMPLETAS"
```

```
head(df_delitos,3)
```

```
##        FECHA      TIPO                           CONCATENAD SEMANA
## 1 2011-01-01 COMERCIAL COMERCIAL/EMPRESA/INDUSTRIA/FARDERO      1
## 2 2011-01-04 COMERCIAL COMERCIAL/EMPRESA/INDUSTRIA/FARDERO      1
## 3 2011-01-16 COMERCIAL COMERCIAL/EMPRESA/INDUSTRIA/FARDERO      3
##   SEMANA_COMPLETAS
## 1                1
## 2                1
## 3                3
```

```
str(df_delitos)
```

```
## 'data.frame':    44212 obs. of  5 variables:
##  $ FECHA           : Factor w/ 1988 levels "2011-01-01","2011-01-02",..: 1 4 16 21 21 23 25 25
##  $ TIPO            : Factor w/ 23 levels "BICICLETA","COMERCIAL",..: 2 2 2 2 2 2 17 3 11 2 ...
##  $ CONCATENAD      : Factor w/ 305 levels "BICICLETA/PERSONA/ASALTO",..: 44 44 44 44 44 44 223
##  $ SEMANA          : int  1 1 3 3 3 4 4 4 5 6 ...
##  $ SEMANA_COMPLETAS: int  1 1 3 3 3 4 4 4 5 6 ...
```

```
unique(df_delitos$TIPO)
```

```
##  [1] COMERCIAL                          TRANSEUNTE
##  [3] CRISTAL                            MOTOCICLETA
##  [5] VEHICULO                           TRANSEUNTE EN VEHICULO
##  [7] BICICLETA                          TRANSPORTE DE PASAJEROS CIUDAD
##  [9] DOMICILIARIO                       INSTITUCIONES PUBLICAS
## [11] INSTITUCION POLITICA               REMOLQUE/PLATAFORMA
## [13] INSTITUCION FINANCIERA             OTRO
## [15] TARJETA BANCARIA/COMERCIAL         TRANSPORTE DE CARGA CIUDAD
## [17] MAQUINARIA PESADA                  TRANSPORTE DE CARGA CARRETERA
## [19] GANADO                             INSTITUCION BANCARIA
## [21] TRANSPORTE DE PASAJEROS CARRETERA No Capturado
## [23] TRACTOR AGRICOLA
## 23 Levels: BICICLETA COMERCIAL CRISTAL DOMICILIARIO ... VEHICULO
```

```r
#df_delitos %>% group_by(TIPO) %>%
#  count() %>% arrange(desc(n)) %>% head()
```

Esto puede deberse a que no todos los delitos se reportan, probablemente exista un sesgo cuando las perdidas son mayores.

```r
#df_delitos %>% group_by(TIPO,SEMANA) %>%
#  count() %>% group_by(TIPO,SEMANA) %>% arrange(desc(n)) %>% head(4)
```

Si observamos el calendario, probablemente se daba a las vacaciones de semana santas.

```r
ggplot(data=df_delitos, aes(x=SEMANA))+
  geom_density()
```