EST-24107: Simulación

Profesor: Alfredo Garbuno Iñigo — Primavera, 2022 — Números no uniformes.
Objetivo: En esta sección del curso veremos métodos para generar números aleatorios de distribuciones un poco mas generales que una distribución uniforme. En particular veremos algunos ejemplos de distribuciones continuas, discretas y mezclas. Además veremos una prueba de uniformidad adicional a la prueba Kolmogorov-Smirnov.
Lectura recomendada: Capítulo 2 de [2].

1. GENERACIÓN DE VARIABLES NO UNIFORMES

R, por ejemplo, tiene distintos generadores de variables aleatorias. Un catálogo nos muestra que podemos generar mucho mas variables que solamente una variable uniforme.

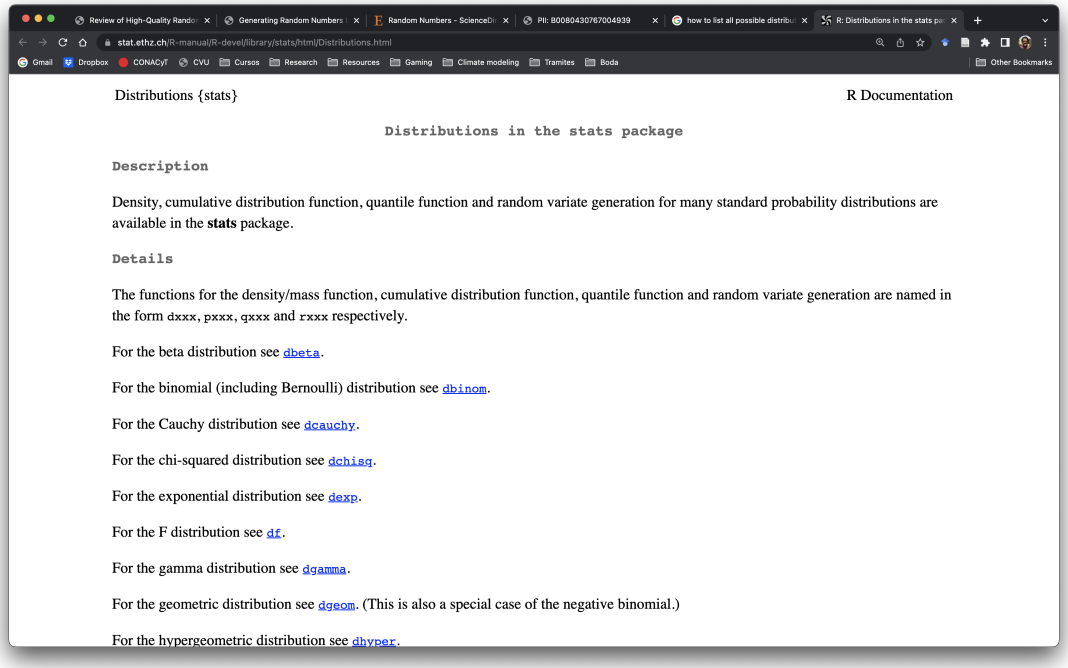


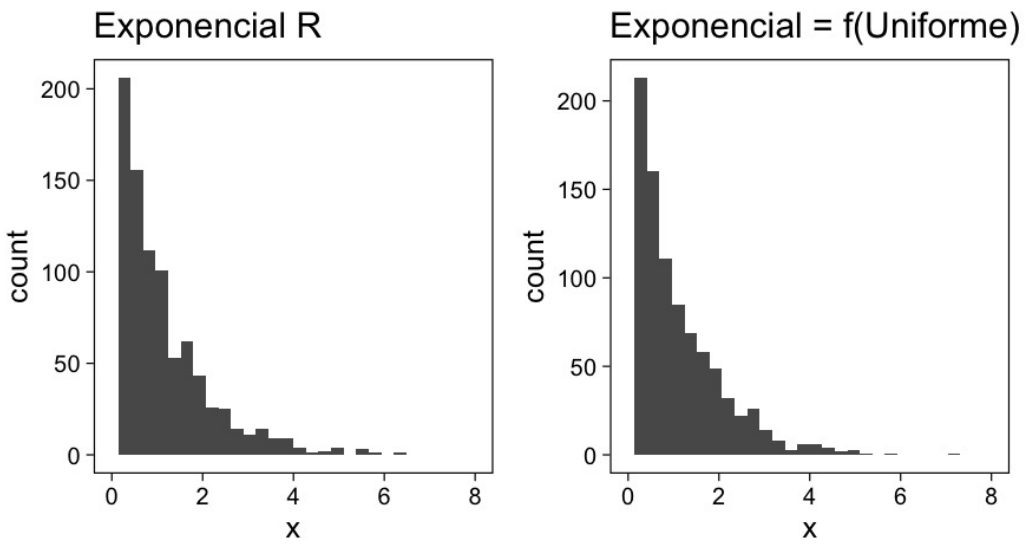
FIGURA 1. *Catálogo de distribuciones en R.*

En esta sección exploraremos los mecanismos tradicionales para generar números pseudo-aleatorios de distribuciones un poco mas generales que una distribución uniforme.

1.1. Método de la transformada inversa

De su curso de cálculo de probabilidades se acordarán que si tenemos una variable aleatoria X y una función de acumulación $\mathbb{P}$. Si utilizamos $U = \mathbb{P}(X)$, con U una variable aleatoria con distribución $\mathsf{U}(0,1)$. Entonces, $X \sim \mathbb{P}$.

1.1.1. Ejercicio: Considera $X \sim \mathsf{Exp}(1)$, de tal forma que $\mathbb{P}(x) = 1 - ^{-x}$. Si sólo sabemos generar números uniformes, ¿cómo generarías números de X?



1.1.2. Tarea: Considera las siguientes distribuciones:

1. Logística con μ, β y función de acumulación

$$\mathbb{P}(x) = \frac{1}{1 + e^{-(x-\mu)/\beta}}\,. \tag{1}$$

2. Cauchy con parámetros μ, β y función de acumulación

$$\mathbb{P}(x) = \frac{1}{2} + \frac{1}{\pi}\mathsf{arctan}((x-\mu)/\beta)\,. \tag{2}$$

Genera números aleatorios con valores $\mu = 1$, $\beta = 2$ y comenta el rol de cada parámetro en cada distribución.

1.2. Generalización

En el caso anterior asumimos que la función inversa de la función de acumulación existe. Sin embargo, no siempre es el caso. Necesitamos definir lo siguiente.

1.2.1. ***Definición*** *[Función inversa generalizada]:* Sea X una variable aleatoria con función de acumulación $\mathbb{P}$. La función inversa generalizada está definida como

$$\mathbb{P}^{-1}(u) = \text{ínf}\{x|F(x) \geq u\}\,. \tag{3}$$

1.2.2. ***Teorema*** *[de la función inversa generalizada]:* Sea $\mathbb{P}$ una función de distribución. Sea $\mathbb{P}^{-1}(\cdot)$ la función inversa generalizada de $\mathbb{P}$ y $U \sim \mathsf{U}(0,1)$. Entonces, la variable aleatoria $X = \mathbb{P}^{-1}(U)$ tiene distribución $\mathbb{P}$.

2. DISTRIBUCIONES CONTINUAS

Hay situaciones donde generar números aleatorios puede ser extremadamente sencillo pues existe una relación entre la distribución que queremos generar y ciertos componentes fáciles de simular.

2.1. Variables χ^2

Recordarán de su curso de cálculo de probabilidades que si $X_i \sim \mathsf{Exp}(1)$ entonces

$$Y = 2\sum_{j=1}^{\nu} X_j \sim \chi^2_{2\nu}\,, \tag{4}$$

con $\nu \in \mathbb{N}$.

2.2. Variables gamma

También recordarán que si $X_i \sim \mathsf{Exp}(1)$ entonces

$$Y = \beta\sum_{j=1}^{\alpha} X_j \sim \mathsf{Gamma}(\alpha, \beta)\,, \tag{5}$$

con $\alpha \in \mathbb{N}$.

2.3. Variables beta

También recordarán que si $X_i \sim \mathsf{Exp}(1)$ entonces

$$Y = \frac{\sum_{j=1}^{\alpha} X_j}{\sum_{j=1}^{\alpha+\beta} X_j} \sim \mathsf{Beta}(\alpha, \beta)\,, \tag{6}$$

con $\alpha, \beta \in \mathbb{N}$.

2.3.1. Tarea: Prueba las igualdades anteriores.

2.3.2. Código: Por ejemplo, podemos utilizar el siguiente código para generar variables de una χ^2_6 .

```
set.seed(108)
U ← runif(3 * 10^4)       # Genera uniformes
U ← matrix(U, nrow = 3)   # Transforma a matriz
X ← -log(U)               # Transforma a exponenciales
X ← 2 * apply(X, 2, sum) # Suma los tres renglones
summary(X)
```

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.151   3.505   5.420   6.064   7.926  27.904
```

A partir de la versión 4.1.1 `R` cuenta con un operador especial (`|>`) llamado `pipe` el cual permite *anidar* ciertas funciones y evitar la asignación repetitiva de variables.

```
set.seed(108)
runif(3 * 10^4) ▷         # Genera uniformes
  matrix(nrow = 3) ▷      # Transforma a matriz
  log() ▷                 # Calcula logaritmos
  apply(2, function(x){-2 * sum(x)} ) ▷
  summary()
```

```
 Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
0.151   3.505   5.420   6.064   7.926  27.904
```

2.4. Variables Gaussianas correlacionadas

Supongamos que queremos generar un par de variables $X \in \mathbb{R}^2$ de tal forma que

$$X \sim \mathsf{N}(0, \Sigma) \,, \tag{7}$$

donde $\Sigma_{ii} = 1$ para $i \in \{1, 2\}$ y $\Sigma_{ij} = \rho$ con $i \neq j$. Supongamos que sólo sabemos generar números aleatorios $\mathsf{N}(0, 1)$.

¿Cómo generaríamos los vectores aleatorios que necesitamos?

¿Qué saben de propiedades matriciales de vectores aleatorios?

```
set.seed(108)
Sigma ← diag(2); Sigma[1,2] ← .75; Sigma[2,1] ← .75;
L ← chol(Sigma)

Z ← rnorm(2 * 10^4)       # Generamos vectores estandar
Z ← matrix(Z, nrow = 2)   # Reacomodamos en matriz
X ← t(L) %*% Z            # Transformacion lineal
cov(t(X))
```

```
       [,1]   [,2]
[1,] 1.0173 0.7772
[2,] 0.7772 1.0312
```

> El operador `%*%` ejemplifica uno de las limitantes por diseño de `R`. Pues no está hecho para realizar operaciones vectoriales de manera nativa. Por ejemplo, en `Matlab` las operaciones son nativas y en `python` a través de `numpy` las operaciones matriciales también (y parte de los métodos).

3. DISTRIBUCIONES DISCRETAS

Ahora, veremos algunas técnicas generales para distribuciones discretas. O mejor dicho, para generar números aleatorios con soporte en los enteros.

Supongamos que nuestro objetivo es poder generar de una $X \sim \mathbb{P}_\theta$ donde $X \in \mathbb{Z}$. La estrategia es `guardar todas las probabilidades del soporte`. Es decir, calcular

$$p_0 = \mathbb{P}_\theta(X \leq 0) \,, \quad p_1 = \mathbb{P}_\theta(X \leq 1) \,, \ldots \,, \tag{8}$$

generar $U \sim \mathsf{U}(0, 1)$ y establecer

$$X = k \text{ si } \; p_{k-1} < U < p_k \,. \tag{9}$$

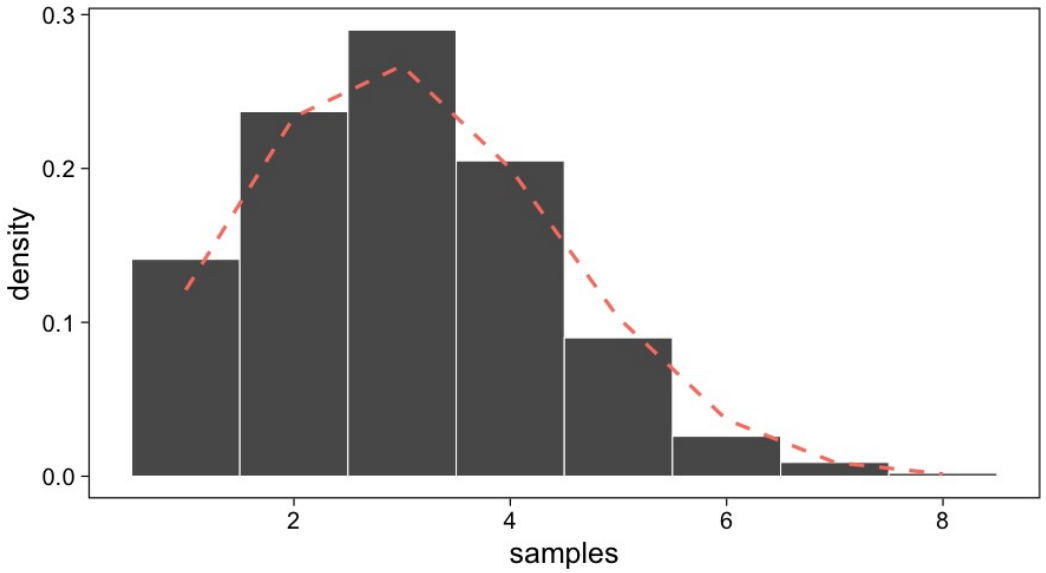
3.1. Binomial

Supongamos que nos interesa $X \sim \mathsf{Bin}(10, 0.3)$, el vector de probabilidades lo podemos calcular con la función `pbinom(k, 10, .3)`.

```
k ← 1:10
pbinom(k, 10, .3)
```

```
[1] 0.1493 0.3828 0.6496 0.8497 0.9527 0.9894 0.9984 0.9999 1.0000 1.0000
```

```
rbinomial ← function(nsamples, size, theta){
  probs ← pbinom(1:10, size, theta)
  x ← c()
  for (jj in 1:nsamples){
    u ← runif(1)
    x[jj] ← which(probs > u)[1]
  }
  return(x)
}
```



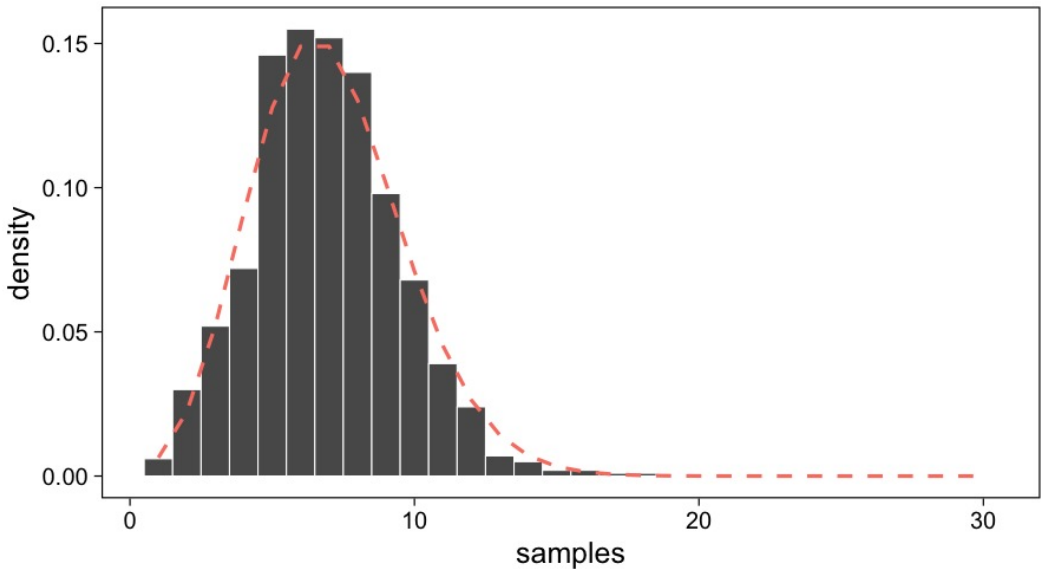
3.2. Poisson

Ahora supongamos que nos interesa simular de una Poisson con parámetro $\lambda = 7$. ¿Cuál es el soporte de una Bin$(10, .3)$? ¿Cuál es el soporte de una Poisson(7)? Tenemos que guardar las probabilidades

```
k ← 1:24
ppois(k, 7)
```

```
 [1] 0.007295 0.029636 0.081765 0.172992 0.300708 0.449711 0.598714 0.729091
 [9] 0.830496 0.901479 0.946650 0.973000 0.987189 0.994283 0.997593 0.999042
[17] 0.999638 0.999870 0.999956 0.999986 0.999995 0.999999 1.000000 1.000000
```

```
rpoisson ← function(nsamples, lambda){
  probs ← ppois(1:30, lambda)
  x ← c()
  for (jj in 1:nsamples){
    u ← runif(1)
    x[jj] ← which(probs > u)[1]
  }
  return(x)
}
```



El problema de generar números aleatorios de la manera anterior es la necesidad de *guardar* el vector de probabilidades. Por ejemplo, una Poisson(100). El intervalo $\lambda \pm 3\sqrt{\lambda}$ es $(70, 130)$.

3.2.1. **Propiedad** *[Regla Empírica o regla de Pukelsheim [1]]:* Si X es una variable aleatoria con media y varianza finitas. Entonces, la probabilidad de que una realización de X se encuentre a mas de 3 desviaciones estándar de la media es a lo mas 5 %.

4. MEZCLAS

Otra familia de distribuciones que es muy interesante de simular son las mezclas. Es decir, cuando podemos escribir

$$\pi(x) = \int_{\mathcal{Y}} \pi(x|y)\,\pi(y)\,\mathrm{d}y\,, \quad \text{o} \quad \mathbb{P}(x) = \sum_{i\in\mathcal{Y}} \mathbb{P}(x|Y=i)\,\mathbb{P}(Y=i)\,, \tag{10}$$

siempre y cuando sea sencillo generar números aleatorios de las marginales.

Por ejemplo, para generar números aleatorios de una t Student con ν grados de libertad. Podemos usar la representación

$$X|y \sim \mathsf{N}(0, \nu/y)\,, \quad Y \sim \chi^2_\nu\,. \tag{11}$$

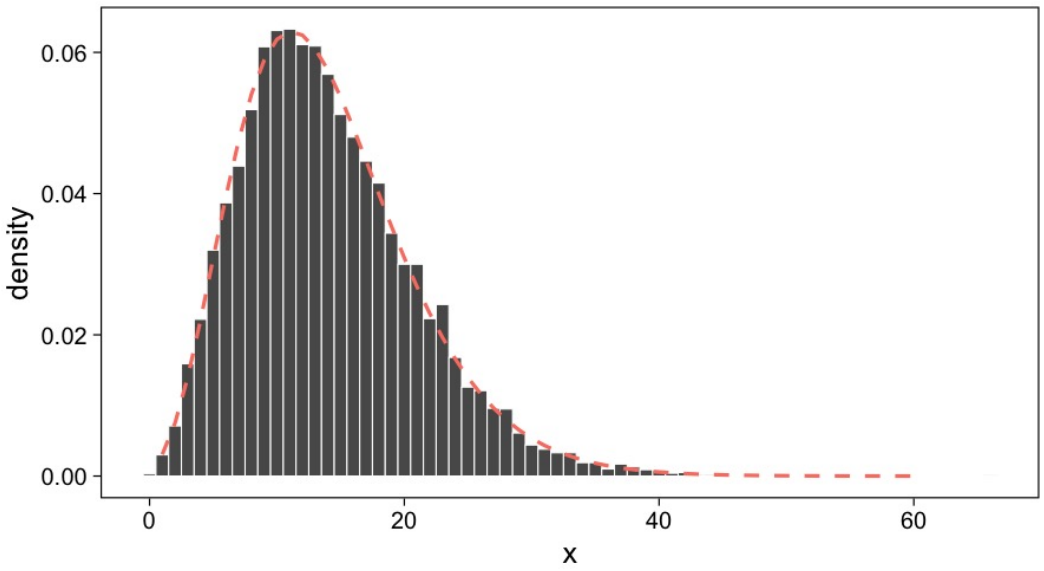
4.1. Binomial negativa

La variable aleatoria $X \sim \mathsf{BinNeg}(n, \theta)$ tiene una representación

$$X|y \sim \mathsf{Poisson}(y)\,, \quad Y \sim \mathsf{Gamma}(n, \beta)\,, \tag{12}$$

donde $\beta = (1-\theta)/\theta$.

```
nsamples ← 10^4
n ← 6; theta ← .3
y ← rgamma(nsamples, n, rate = theta/(1-theta))
x ← rpois(nsamples, y)
```



4.1.1. Tarea Prueba que una binomial negativa puede ser expresada como una mezcla Poisson-Gamma.

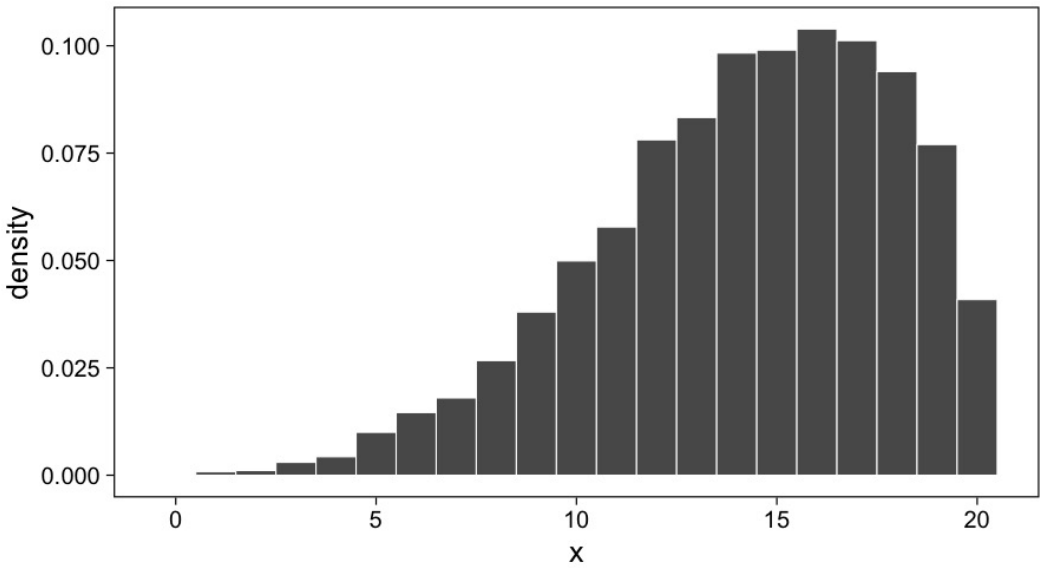
4.2. Beta Binomial

Una distribución bastante conocida es la distribución Beta-Binomial la cual, como su nombre indica, está conformada por una mezcla de una $\mathsf{Bin}(n, \theta)$ y una $\mathsf{Beta}(\alpha, \beta)$. Es decir, $x \sim \mathsf{BetaBin}(n, \alpha, \beta)$ si su función de masa de probabilidad está dada por

$$\mathbb{P}(x) = \binom{n}{x} \frac{B(x+\alpha, n-x+\beta)}{B(\alpha, \beta)}, \tag{13}$$

donde $B(\alpha, \beta)$ se conoce como la función Beta $B(\alpha, \beta) = (\Gamma(\alpha)\Gamma(\beta))/\Gamma(\alpha+\beta)$.

```
nsamples ← 10^4
n ← 20; a ← 5; b ← 2
theta ← rbeta(nsamples, a, b)
x ← rbinom(nsamples, n, theta)
```

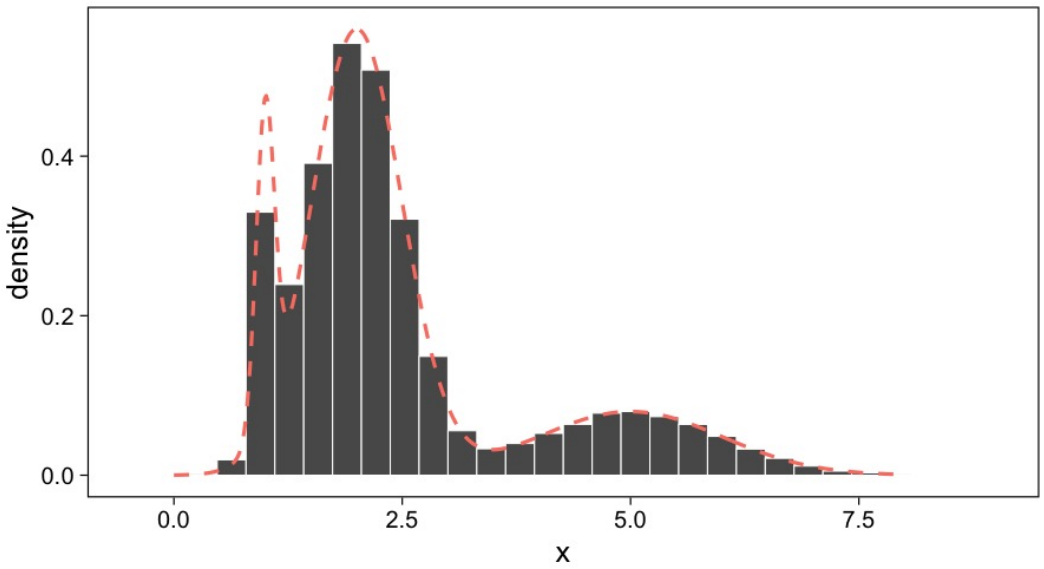


4.3. Mezclas Gaussianas

Otro modelo *famoso* es el de una mezcla de Gaussianas. Donde tenemos k posibles poblaciones cada una con proporción $\pi_k \in (0,1)$ y $\sum \pi_k = 1$. Y además, cada población tiene comportamiento distinto

$$\pi(x|k) = \mathsf{N}(x|\mu_k, \sigma_k)\,. \tag{14}$$

```
nsamples ← 10^5
y ← sample(1:3, size = nsamples, prob = c(.1, .7, .2), replace = TRUE)
x ← rnorm(nsamples,
          mean = ifelse(y==1, 1, ifelse(y==2, 2, 5)),
          sd = ifelse(y==1, 0.1, ifelse(y==2, 0.5, 1)))
```



5. PRUEBA χ^2

Podemos usar otro mecanismo para probar estadísticamente si nuestros números pseudo aleatorios siguen la distribución que deseamos.

Podemos pensar en esta alternativa como la versión `discreta` de la prueba `KS`.

Lo que estamos poniendo a prueba es

$$H_0 : \mathbb{P}(x) = \mathbb{P}_0(x)\ \forall x \quad \text{contra} \quad H_1 : \mathbb{P}(x) \neq \mathbb{P}_0(x) \text{ para alguna } x\,. \tag{15}$$

5.1. Procedimiento de la prueba χ^2

1. Hacemos una partición del rango de la distribución supuesta en k subintervalos con límites $\{a_0, a_1, \ldots, a_k\}$, y definimos N_j como el número de observaciones (de nuestro generador de pseudo-aleatorios) en cada subintervalo.
2. Calculamos la proporción esperada de observaciones en el intervalo $(a_{j-1}, a_j]$ como

$$p_j = \int_{a_{j-1}}^{a_j} \mathrm{d}\mathbb{P}(x)\,. \tag{16}$$

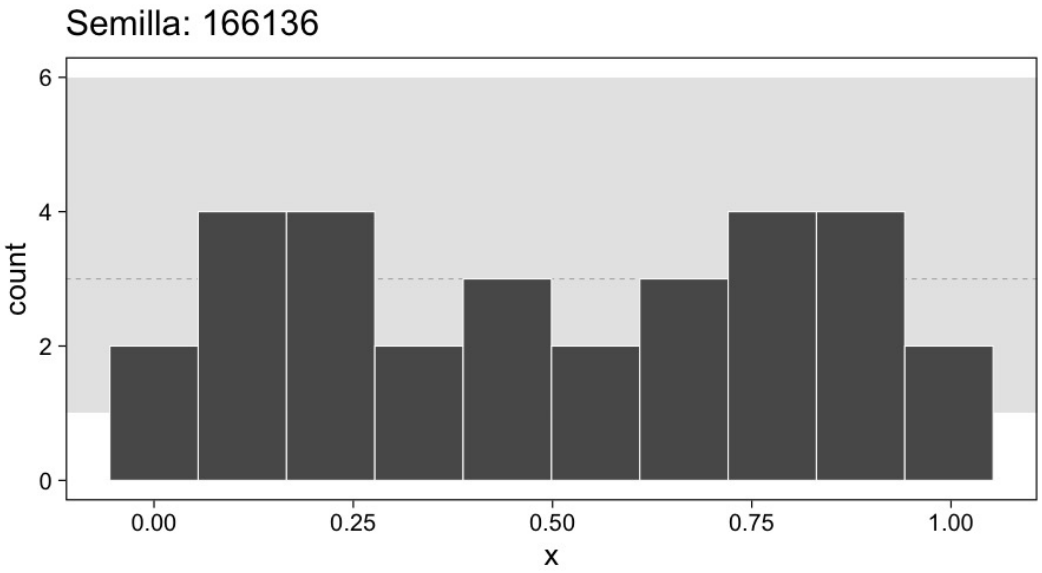
3. La estadística de prueba es

$$\chi^2 = \sum_{j=1}^{k} \frac{(N_j - np_j)^2}{np_j}. \tag{17}$$

> Nota que estamos comparando dos histogramas. El histograma observado que construimos a partir de nuestros números pseudo-aleatorios contra el histograma que esperaríamos de la distribución. ¿Puedes pensar en algún problema con esta prueba?

La visualización correspondiente sería lo siguiente. Utilizamos nuestro generador para obtener muestras.

```
## Esto es para poner a prueba un pseudo generador
rpseudo.uniform ← function(nsamples, seed = 108727){
  x0 ← seed; a ← 7**5; m ← (2**31)-1;
  x  ← x0;
  for (jj in 2:nsamples){
    x[jj] ← (a * x[jj-1]) %% m
  }
  x/m
}
```



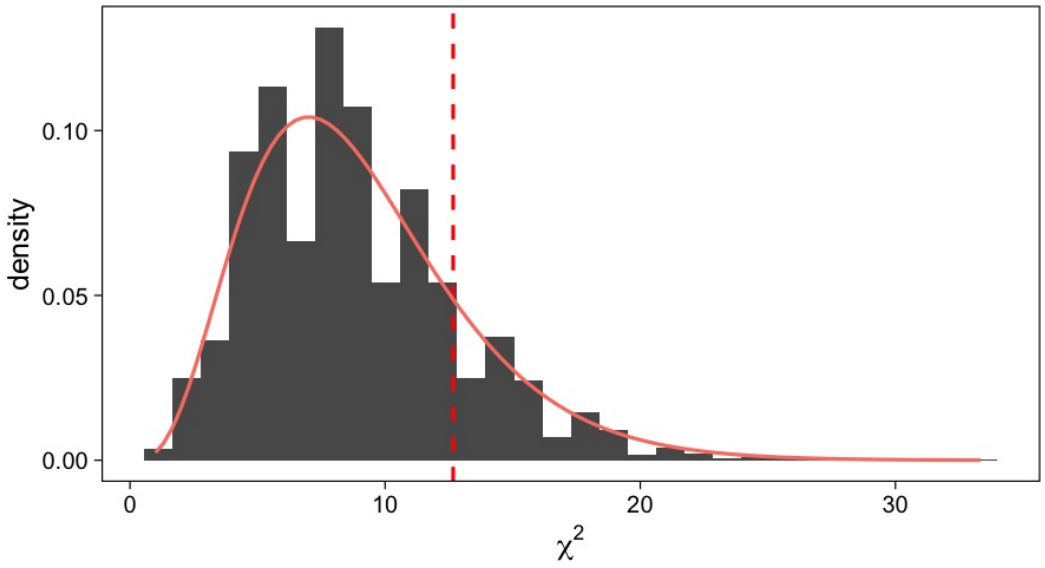
5.1.1. Pregunta: ¿Qué esperaríamos de nuestro estadístico χ^2 si nuestro generador de pseudo-aleatorios es incorrecto?

5.2. Aplicación de la prueba

```
nsamples ← 30; nbreaks ← 10
samples ← data.frame(x = rpseudo.uniform(nsamples))

Fn ← hist(samples$x, breaks = nbreaks, plot = FALSE)$counts/nsamples
F0 ← 1/nbreaks

X2.obs ← (nsamples*nbreaks)*sum((Fn - F0)**2)
```

```r
## Esto es para generar datos observados de la distribucion que queremos
experiment ← function(nsamples){
  nbreaks ← 10
  samples ← data.frame(x = runif(nsamples))
  Fn ← hist(samples$x, breaks = nbreaks, plot = FALSE)$counts/nsamples
  F0 ← 1/nbreaks
  X2 ← (nsamples*nbreaks)*sum((Fn - F0)**2)
  return(X2)
}

X2 ← c()
for (jj in 1:5000){
  X2[jj] ← experiment(nsamples)
}
```

En la Section 5.2 se muestra el histograma de las réplicas del estadístico χ^2 bajo el generador uniforme (lo tomamos como la distribución de la hipótesis nula) y comparamos contra el observado (línea punteada). Adicional, se incorpora la densidad de una χ^2_{k-1} (leáse ji-cuadrada con $k-1$ grados de libertad) que es la distribución asintótica del estadístico.



Por lo tanto, la probabilidad de haber observador una estadístico χ^2 tan extremo como el que observamos si el generador hubiera sido el que suponemos es:

```
[1] "Estadistico: 12.6667, Probabilidad: 0.1694"
```

Que podemos comparar contra el que obtenemos de una prueba "tradicional":

```r
counts.obs ← Fn*nsamples
chisq.test(counts.obs, p = rep(1, nbreaks)/nbreaks, simulate.p.value = TRUE)
```

```

        Chi-squared test for given probabilities with simulated p-value (based
        on 2000 replicates)

data:  counts.obs
X-squared = 13, df = NA, p-value = 0.2
```

- La prueba χ^2 pues usualmente no es buena cuando el número de observaciones es menor a 50.
- La prueba `KS` tiene mejor potencia que la prueba χ^2:

```
ks.test(samples$x, "punif")
```

```

        Exact one-sample Kolmogorov-Smirnov test

data:  samples$x
D = 0.16, p-value = 0.4
alternative hypothesis: two-sided
```

5.3. Aplicación de pruebas

En la práctica se utiliza una colección de pruebas pues cada una es sensible a cierto tipo de desviaciones. La bateria de pruebas mas utilizada es la colección de pruebas `DieHARD` que desarrolló George Marsaglia y que se ha ido complementando con los años.

REFERENCIAS

[1] F. Pukelsheim. The three sigma rule. *The American Statistician*, 48(2):88–91, 1994. 6

[2] C. Robert and G. Casella. *Introducing Monte Carlo Methods with R*. Springer New York, New York, NY, 2010. 1