EYP1113 - Probabilidad y Estadística

Laboratorio 4

Profesoras: Natalia Venegas y Pilar Tello

Facultad de Matemáticas
Departamento de Estadística
Pontificia Universidad Católica de Chile

Segundo Semestre 2019

Contenido I

Modelos de Probabilidad

Modelos de Probabilidad

Existen diversos modelos de probabilidad para estudiar. Los modelos más utilizados son

Modelo	Comando
Binomial	`_binom()`
Poisson	`_pois()`
Uniforme	`_unif()`
Normal	`_norm()`
Exponencial	`_exp()`
Gamma	`_gamma()`
Chi Cuadrado	`_chisq()`
t-Student	`_t()`
Fisher	`_f()`

Cada comando puede ser utilizado de 4 formas diferentes:

- dDISTR(x,...). Entrega $P(X = x)$.
- pDISTR(x,...). Entrega $P(X \leq x)$.
- qDISTR(q,...). Entrega el valor de x tal que $P(X \leq x) = q$.
- rDISTR(n,...). Genera una muestra proveniente de un modelo de distribución.

Distribución Exponencial

La distribución Exponencial

Si $X \sim \text{Exp}(\lambda), x > 0$, entonces:

- Para obtener la densidad $f_X(x)$ en el punto x se usa el comando `dexp(x,rate=`λ`)`.
- Para calcular la probabilidad acumulada $P(X < q)$ usamos el comando `pexp(q,rate=`λ`)`.
- Para generar n variables aleatorias provenientes de la distribución se utiliza el comando `rexp(n,rate=`λ`)`.
- Para calcular cuantiles de la distribución se usa el comando `qexp(p,rate=`λ`)`.

Ejemplos:

Si $X \sim \text{Exp}(\lambda = 3)$:

- Calcule $f_X(1)$.
  `dexp(1,rate=3)`
- Calcule $P(X < 1.5) = F_X(1.5)$
  `pexp(1.5,rate=3)`
- Genere una muestra de tamaño $n = 1000$.
  `rexp(1000,rate=3)`
- Si $P(X < k) = 0.5$, obtenga el valor de k.
  `qexp(0.5,rate=3)`

Distribución Uniforme

La distribución Uniforme

Si $X \sim \text{Unif}(a, b), a < x < b$, entonces:

- Para obtener la densidad $f_X(x)$ en el punto x se usa el comando `dunif(x,min=a,max=b)`.
- Para calcular la probabilidad acumulada $P(X < q)$ usamos el comando `punif(q,min=a,max=b)`.
- Para generar n variables aleatorias provenientes de la distribución se utiliza el comando `runif(n,min=a,max=b)`.
- Para calcular cuantiles de la distribución se usa el comando `qunif(p,min=a,max=b)`.

Ejercicios

Invente tres situaciones similares al ejemplo anterior, para la distribución uniforme, tomando en cuenta los siguientes casos:

- $a = 0, b = 1$
- $a = -10, b = 10$
- $a = 50, b = 100$
- $a = -15, b = 4$

Distribución Normal

La distribución Normal

Si $X \sim \text{Normal}(\mu, \sigma^2), x \in \mathbb{R}$, entonces:

- Para obtener la densidad $f_X(x)$ en el punto x se usa el comando `dnorm(x,mean=`μ`,sd=`σ`)`.
- Para calcular la probabilidad acumulada $P(X < q)$ usamos el comando `pnorm(q,mean=`μ`,sd=`σ`)`.
- Para generar n variables aleatorias provenientes de la distribución se utiliza el comando `rnorm(n,mean=`μ`,sd=`σ`)`.
- Para calcular cuantiles de la distribución se usa el comando `qnorm(p,mean=`μ`,sd=`σ`)`.

Distribución Log-Normal

La distribución Log-Normal

Si $X \sim$ Log-Normal$(\mu, \sigma^2), x > 0$, entonces:

- Para obtener la densidad $f_X(x)$ en el punto x se usa el comando `dlnorm(x,meanlog=`μ`,sdlog=`σ`)`.
- Para calcular la probabilidad acumulada $P(X \leq q)$ usamos el comando `plnorm(q,meanlog=`μ`,sdlog=`σ`)`.
- Para generar n variables aleatorias provenientes de la distribución se utiliza el comando `rlnorm(n,meanlog=`μ`,sdlog=`σ`)`.
- Para calcular cuantiles de la distribución se usa el comando `qnorm(p,meanlog=`μ`,sdlog=`σ`)`.

Otras distribuciones

La distribución Gamma

Si $X \sim \text{Gamma}(\lambda, \nu), x > 0$, entonces:

- Para obtener la densidad $f_X(x)$ en el punto x se usa el comando `dgamma(x,shape=`ν`,scale=`λ`)`.
- Para calcular la probabilidad acumulada $P(X < q)$ usamos el comando `pgamma(q,shape=`ν`,scale=`λ`)`.
- Para generar n variables aleatorias provenientes de la distribución se utiliza el comando `rgamma(n,shape=`ν`,scale=`λ`)`.
- Para calcular cuantiles de la distribución se usa el comando `qgamma(p,shape=`ν`,scale=`λ`)`.

La distribución *Chi*-cuadrado

Si $X \sim \chi^2_n, x > 0$, entonces:

- Para obtener la densidad $f_X(x)$ en el punto x se usa el comando `dchisq(x,df=n)`.
- Para calcular la probabilidad acumulada $P(X < q)$ usamos el comando `pchisq(q,df=n)`.
- Para generar n variables aleatorias provenientes de la distribución se utiliza el comando `rchisq(n,df=n)`.
- Para calcular cuantiles de la distribución se usa el comando `qchisq(p,df=n)`.

Gráficos de distribuciones continuas

El comando curve( )

Podemos usar el comando curve( ) para graficar la función de densidad o de distribución de variables aleatorias continuas. Para una variable aleatoria X el uso es el siguiente:

curve(ddist(x,...),from=,to=,...) o

curve(pdist(x,...),from=,to=,...)

donde ddist(x,...) puede ser cualquier función de densidad vistas anteriormente (dnorm, dunif, dexp, etc.) y pdist(x,...) puede ser cualquier función de distribución vistas anteriormente (pnorm, punif, pexp, etc.). Esta función debe quedar expresada en términos de x.

Ejemplos:

Sea $X \sim \text{Normal}(\mu = 6, \sigma^2 = 36)$

- Grafique la función de densidad de X, cuando $x \in [-6, 16]$
  `curve(dnorm(x,mean=6,sd=6),from=-6,to=16)`

- Grafique la función de distribución de X, cuando $x \in [-6, 16]$
  `curve(pnorm(x,mean=6,sd=6),from=-6,to=16)`

Cálculo de integrales en R

Intregración en R

Para calcular integrales en R basta con definir la función a integrar y luego utilizar el comando `integrate()`. Por ejemplo, si queremos calcular el área bajo la curva de una densidad normal estándar, entre 1.96 e infinito, entonces usamos:

```
integrate(dnorm, -1.96, Inf)
```

Lo cual arroja el resultado requerido con cierto error de aproximación. Si por ejemplo, queremos calcular la siguiente integral:

$$\int_0^{\infty} \frac{1}{(1+x)\sqrt{x}} dx$$

entonces usamos:

```
integrand <- function(x){1/((x+1)*sqrt(x))}
integrate(integrand, lower = 0, upper = Inf)
```

Ejercicios

Ejercicios

1. Simule 100 variables aleatorias provenientes de una distribución exponencial con tasa 2. Grafique su función de densidad empírica junto con la curva de densidad teórica.
2. Simule 100 variables aleatorias provenientes de una distribución log normal de parámetros 0.5 y 0.02. Grafique su función de densidad empírica junto con la curva de densidad teórica.
3. Estudios realizados recientemente por una AFP han revelado que el tiempo de duración de sus cuentas de ahorro voluntario de afiliados (lapso en el que las cuentas de ahorro voluntario permanecen abiertas) sigue un comportamiento normal con una media de 24 meses y una desviación estándar de 10.2 meses.

Ejercicios

a) Si un afiliado abre una cuenta de ahorro voluntario en la AFP, ¿cuál es la probabilidad de que esa cuenta haya aún dinero después de 32 meses?

b) Si una cuenta lleva abierta 18 meses, ¿cuál es la probabilidad que dure entre 20 y 32 meses?

c) Si se escogen 5 personas afiliadas con cuenta de ahorro voluntario, ¿cuál es la probabilidad de que a lo más una de ellas mantenga la cuenta por menos de 24 meses?

Programación básica

Programación básica

- Un programa es una lista de comandos que son ejecutados uno tras otro.
- Típicamente un programa tiene tres partes: input, cálculos y output.
- El siguiente es un programa que calcula la raíz real de la ecuación $ax^2 + bx + c = 0$:

```
# input
a<- 1; b<-4; c<- 2
# calculos
raiz1 <- (-b-sqrt(b^2-4*a*c))/(2*a)
raiz2 <- (-b+sqrt(b^2-4*a*c))/(2*a)
# output
c(raiz1, raiz2)
```

Programación condicional, comando `if( )`

- La ejecución o no de alguna parte de un programa puede depender de si una condición se cumple o no. En este contexto usamos el comando `if` de la siguiente forma

```
if(expresión lógica){
expresión 1
 ....}
```

- Si la condición tiene dos posibilidades, podemos usar el comando `if()...else...` de la siguiente forma:

```
if(expresión lógica){
expresión 1
...} else{
expresión 2
...}
```

Ejemplo

- Vamos a calcular la raíz real de la ecuación $ax^2 + bx + c = 0$ dependiendo del valor del discriminante.

```
discriminante <- b^2-4*a*c
if(discriminante > 0){
raiz <- c((-b-sqrt(b^2-4*a*c))/(2*a),
  (-b+sqrt(b^2-4*a*c))/(2*a))
} else {
if(discriminante == 0){
raiz <- -b/(2*a)
} else{
raiz <- c()
}
}
raiz
a<- 1; b<- 4; c<- 2
a<- 1; b<- 2; c<- 1
a<- 2; b<- 2; c<- 1
```

Programación con loops, comando `for( )`

- Sea `x` una variable simple y `vector` un vector. Entonces el comando `for` se usa como

```
for(x in vector){
expresión 1
...
}
```

- Este comando ejecuta las instrucciones que están dentro de `{...}` para cada elemento del objeto `vector`. Podemos usar `x` dentro del loop.

Ejemplo

- Definir un objeto `(1, 4, 7, 10, 13, 16, 19)` y calcular $\prod_{i=1}^{n} x_i$, $n = 1, 2, \ldots, 7$. A continuación se muestran dos formas de hacerlo

```
x.seq <- seq(1, 19, 3)
producto <- 1
for(x in x.seq){
producto <- producto*x
print(producto)
}
# otra forma de hacer lo mismo
for(i in 1:length(x.seq)){
producto <- prod(x.seq[1:i])
 print(producto)
}
```

Qué hace el comando `print()`?

Programación con loops, comando `while()`

- A veces queremos hacer un loop pero no sabemos cuantas veces, sino que sabemos hasta cuando queremos hacerlo. O sea, cada vez que hacemos el loop chequeamos alguna condición para ver si hemos terminado o seguimos.

- En este contexto usamo el loop `while`

```
while(expresión lógica){
expresión 1
...
}
```

- En un `while` la expresión lógica se chequea antes de empezar. Si la expresión lógica se cumple, se ejecutan los comandos dentro de `{...}`, si no se cumple se detiene. La expresión lógica debe ser falsa en algún instante.

Ejemplo

- Definir un objeto `(1, 4, 7, 10, 13, 16, 19)` y calcular $\prod_{i=1}^{n} x_i$, siempre y cuando $\prod_{i=1}^{n} x_i < 4000$

```
x.seq <- seq(1, 19, 3)
producto <- 1
i <- 1
while(producto < 4000){
producto <- producto*x.seq[i]
print(c(i, producto))
i <- i+1
}
```

Ejemplo

- Calcular $\sum_{i=1}^{100} i^2$.

  Usando loop:

```
suma2 <- 0
for(i in 1:100){
suma2 <- suma2 + i^2
}
suma2
```

  Usando vectores:

```
sum((1:100)^2)
```

Comando `ifelse`

- Podemos aplicar la misma lógica usada con el loop `if()...else...` pero ahora usando vectores.
- El comando a usar es `ifelse(test, A, B)` donde `test` es una expresión lógica, A es lo que se ejecuta si la expresión lógica es verdadera y B es lo que se ejecuta si la expresión lógica es falsa.
- Veamos el siguiente ejemplo

```
x <- c(-2, -1, 1, 2)
ifelse(x > 0, "Positivo", "Negativo")
```

Creando nuevas funciones en R, comando `function`

- En R podemos crear nuevas funciones usando el comando
```
function(argumento1, argumento2, ...){
expresión 1
...
return()/list()
}
```
- La función recibe los argumento `argumento1`, `argumento2`, etc los que utiliza durante las instrucciones secuenciales `expresión 1`, ... Finalmente los resultados de la función se entregan en una lista, con el comando `list` o como un valor, con el comando `return`.

Ejemplo

- Ejemplo: Crear una función que reciba como argumento un vector de datos y entregue una análisis descriptivo completo (lo que hemos visto hasta ahora) dependiendo del tipo de variable (numérica o categórica).

```
AED <- function(x) {
if(is.numeric(x)==TRUE){
prom <- mean(x); prom.rec <- mean(x, trim=0.1)
med <- median(x); cuar <- quantile(x)
mi <- min(x); ma <- max(x)
va <- var(x); RIC <- IQR(x)
par(mfrow=c(2, 1))
hist(x, freq=F)
boxplot(x,horizontal=TRUE)
list(promedio=prom, promedio.recortado=prom.rec, mediana=med, cuartiles=cuar,
minimo=mi, maximo=ma, varianza=va, RIC=RIC)
}else{conteo <- table(x)
par(mfrow=c(1,2))
barplot(conteo)
pie(conteo)
return(Tabla.conteo=conteo)
}
}
```

Ejemplo

- Pruebe la función recién creada con los siguientes ejemplos:

```
muestra3=rnorm(n=1000,mean=100,sd=20)
AED(muestra3)
muestra4=c("a","a","a","b","b")
AED(muestra4)
```

Comandos pmin y pmax

- Con la función `pmin(`x_1`,...,` x_n`, na.rm=F)` podemos calcular el mínimo de cada vector x_i, $i = 1, 2, ..., n$, por lo tanto, el resultado es un vector de mínimos. El argumento `na.rm` indica si hay observaciones “en blanco" o no en los vectores x_1`,...,` x_n.

- Análogamente, con la función `pmax(`x_1`,...,` x_n`, na.rm=F)` podemos calcular el máximo de cada vector x_i, $i = 1, 2, ..., n$. El resultado es un vector de máximos.

Comandos `sapply`, `apply` y `tapply`

- La familia de funciones `apply` de R permiten aplicar funciones a vectores o matrices.
- El comando `sapply(X, FUN, ...)` calcula para cada elemento del vector X la función FUN. Si la función FUN tiene más de un argumento, éstos los podemos agregar en los argumentos `...` de la función `sapply`.
- Ejemplo: Genera un vector de 10 observaciones de una distribución $N(0, 1)$ y redondea cada valor del vector. Usa la función `round` para redondear.

```
x1 <- rnorm(10); x1
sapply(x1, round)
```

Repite lo anterior, pero que ahora cada valor redondeado tenga 3 decimales.

```
sapply(x1, round, digits=3)
```

Comandos `sapply`, `apply` y `tapply`

- La función `apply(X, MARGIN, FUN,...)` calcula para cada fila o columna de la matriz X la función `FUN`. Si ésta tiene más de un argumento, los agregamos en el argumento `...` de la función `apply`. Para determinar si el cálculo se hace por filas, usamos el argumento `MARGIN=1`, si es por columnas, usamos el argumento `MARGIN=2`

- Ejemplo: Define una matriz de (10×3) de valores de una distribución $N(20, 9)$ y calcula el promedio de cada columna.
  `X <- matrix(rnorm(30, 20, 3), ncol=3, nrow=10); X`
  `apply(X, 2, mean)` Repite lo anterior pero calculando la media recortada al 20% para cada columna.
  `apply(X, 2, mean, trim=0.1)`

Comandos sapply, apply y tapply

- La función `tapply(X, INDEX, FUN , ...)` calcula la función FUN al vector X dependiendo de los valores del argumento INDEX que por defecto se asume categórico y es un vector del mismo largo que X. Si la función FUN tiene más de un argumento, los agregamos en los argumentos `...` de la función `tapply`.

- Ejemplo: Definir una variable edad de tamaño 30 con distribución $N(40, 10)$ y otra variable genero de tamaño 30 con distribución $\text{Bernoulli}(0.3)$. Calcular la edad promedio de hombres y mujeres.

```
edad <- rnorm(30, 40, sqrt(10));edad
genero <- rbinom(30, 1, 0.3); genero
tapply(edad, genero, mean)
```

Repetir lo anterior, pero calculando la media recortada al 20%.

```
tapply(edad, genero, mean, trim=0.1)
```