Estadística Computacional: Primer Exámen Parcial

Edgar Granados

September 30, 2018

1. Manipulación y Visualización de Datos en R

a. Cumple la base de datos (iris) con el principio de datos limpios?

Respuesta

Examinando la base de datos, se puede observar que la base de datos cumple con los dos primeros principios de datos limpios. En cuanto al tercer principio, la base de datos presenta el problema de tener filas que son iguales: no se puede diferenciar entre filas repetidas. Debido a esto, conviene agregar una columna adicional "id" que permita diferenciar entre distintas filas.

b. En caso de que no cumpla con el principio de datos limpios, limpe los datos. Imprima las primeras 6 líneas de los datos limpios. (si ya estaban limpios, entonces imprima las primeras 6 líneas de los datos originales)

Respuesta

```
iris <- mutate(iris, id = 1:150)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species id
## 1          5.1         3.5          1.4         0.2  setosa  1
## 2          4.9         3.0          1.4         0.2  setosa  2
## 3          4.7         3.2          1.3         0.2  setosa  3
## 4          4.6         3.1          1.5         0.2  setosa  4
## 5          5.0         3.6          1.4         0.2  setosa  5
## 6          5.4         3.9          1.7         0.4  setosa  6
```

c. Cuántas observaciones y cuántas variables tiene la base de datos?

Respuesta

Utilizando la función "summary" es posible responder ésta pregunta facilmente. La base de datos contiene 6 variables. Se tienen 150 observaciones (50 de cada una de las tres especies de flor).

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
```

```
##        Species         id
##  setosa    :50   Min.   :  1.00
##  versicolor:50   1st Qu.: 38.25
##  virginica :50   Median : 75.50
##                  Mean   : 75.50
##                  3rd Qu.:112.75
##                  Max.   :150.00
```

d. Cuál es la clase atómica de cada una de las variables?

Respuesta

Para obtener la clase atómica de cada variable, se utiliza la función "class". En el caso de la variable "Species", se utiliza la función "typeof" a la salida de "class" debido a que se trata de una estructura "factor". Con esto se obtiene que, todas las variables son de tipo "numeric" excepto por la variable "Species" que es de tipo "character".

```
class(iris$Species)
```

```
## [1] "factor"
```

```
typeof(class(iris$Species))
```

```
## [1] "character"
```

```
class(iris$Sepal.Length)
```

```
## [1] "numeric"
```

```
class(iris$Sepal.Width)
```

```
## [1] "numeric"
```

```
class(iris$Petal.Length)
```

```
## [1] "numeric"
```

```
class(iris$Petal.Width)
```

```
## [1] "numeric"
```

```
class(iris$id)
```

```
## [1] "integer"
```

e. Filtre las flores de la especia Setosa e imprima las primeras 6 observaciones

Respuesta

Se utiliza la función "filter" para obtener todas las ocurrencias de "setosa" y ésta salida se pasa a la función "head".

```
head(filter(iris, Species == "setosa"))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species id
## 1          5.1         3.5          1.4         0.2  setosa  1
## 2          4.9         3.0          1.4         0.2  setosa  2
## 3          4.7         3.2          1.3         0.2  setosa  3
```

```
## 4          4.6         3.1          1.5         0.2  setosa  4
## 5          5.0         3.6          1.4         0.2  setosa  5
## 6          5.4         3.9          1.7         0.4  setosa  6
```

f. Ordene la base de datos de manera descendente con respecto a la variable Petal.Length e imprima las primeras 6 observaciones.

Respuesta

Para ordenar una base de datos, se utiliza la función "arrange" definiendo las variables que se desea ordenera. En particular, a la variable "Petal.Length" se le aplica la función "desc" para que se ordene de manera descendiente. A la salida, se aplica la función "head" para obtener las primeras 6 observaciones.

```
head(arrange(iris, desc(Petal.Length)))
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width   Species  id
## 1          7.7         2.6          6.9         2.3 virginica 119
## 2          7.7         3.8          6.7         2.2 virginica 118
## 3          7.7         2.8          6.7         2.0 virginica 123
## 4          7.6         3.0          6.6         2.1 virginica 106
## 5          7.9         3.8          6.4         2.0 virginica 132
## 6          7.3         2.9          6.3         1.8 virginica 108
```

g. Cree una nueva variable en donde se muestre el atributo Sepal.Length en milímetros e imprima las 6 primeras observaciones.

Respuesta

Se utiliza la función "mutate" para agregar una nueva variable (columna). Para los valores de ésta nueva columna, se toman los valores de la columna "Sepal.Length" y se multiplican por 10.

```
iris <- mutate(iris, Sepal.Length.cm = iris$Sepal.Length* 10)
head(iris)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species id
## 1          5.1         3.5          1.4         0.2  setosa  1
## 2          4.9         3.0          1.4         0.2  setosa  2
## 3          4.7         3.2          1.3         0.2  setosa  3
## 4          4.6         3.1          1.5         0.2  setosa  4
## 5          5.0         3.6          1.4         0.2  setosa  5
## 6          5.4         3.9          1.7         0.4  setosa  6
##   Sepal.Length.cm
## 1              51
## 2              49
## 3              47
## 4              46
## 5              50
## 6              54
```

h. Elimine las observaciones con valores faltantes en la variable Sepal.Width e indique el número de observaciones de la nueva base de datos.

Respuesta

Examinando la base de datos original, no se encontraron valores faltantes para ninguna variable.

i. Cuál es la media de la variable Petal.Width para cada una de las especies de flores?

Respuesta

Las medias son las siguientes: setosa - 0.2 versicolor - 1.3 virginica - 2.0 Se utiliza la función "filter" para filtrar los datos requeridos y a la salida, solamente se pasa la variable Petal.Width a la función summary para obtener la media. A continuación se muestran los comandos para obtener las medias:

```
summary(filter(iris, Species == "setosa")$Petal.Width)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   0.100   0.200   0.200   0.246   0.300   0.600
```

```
summary(filter(iris, Species == "versicolor")$Petal.Width)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   1.200   1.300   1.326   1.500   1.800
```

```
summary(filter(iris, Species == "virginica")$Petal.Width)
```

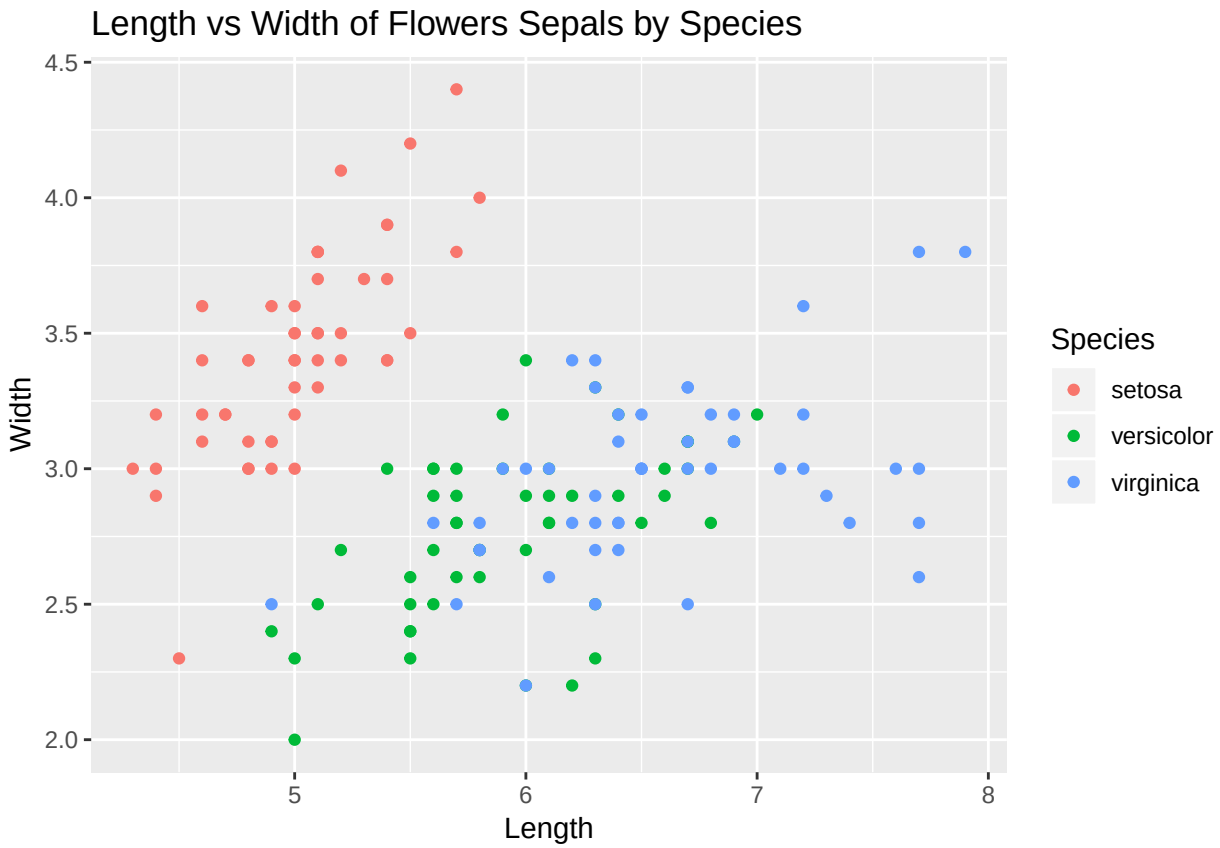
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.400   1.800   2.000   2.026   2.300   2.500
```

j. Realice una gráfica de dispersión de las variables x=Sepal.Length contra y=Sepal.Width en la que se distingan las diferentes esoecies por color o por forma de los puntos. La gráfica debe incluir título y nombres de los ejes horizontal y vertical.

Respuesta

Se utilizó la librería ggplot para realizar la gráfica que se muestra a continuación:

```
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) +
  geom_point() +
  ggtitle("Length vs Width of Flowers Sepals by Species") +
  xlab("Length") +
  ylab("Width")
```

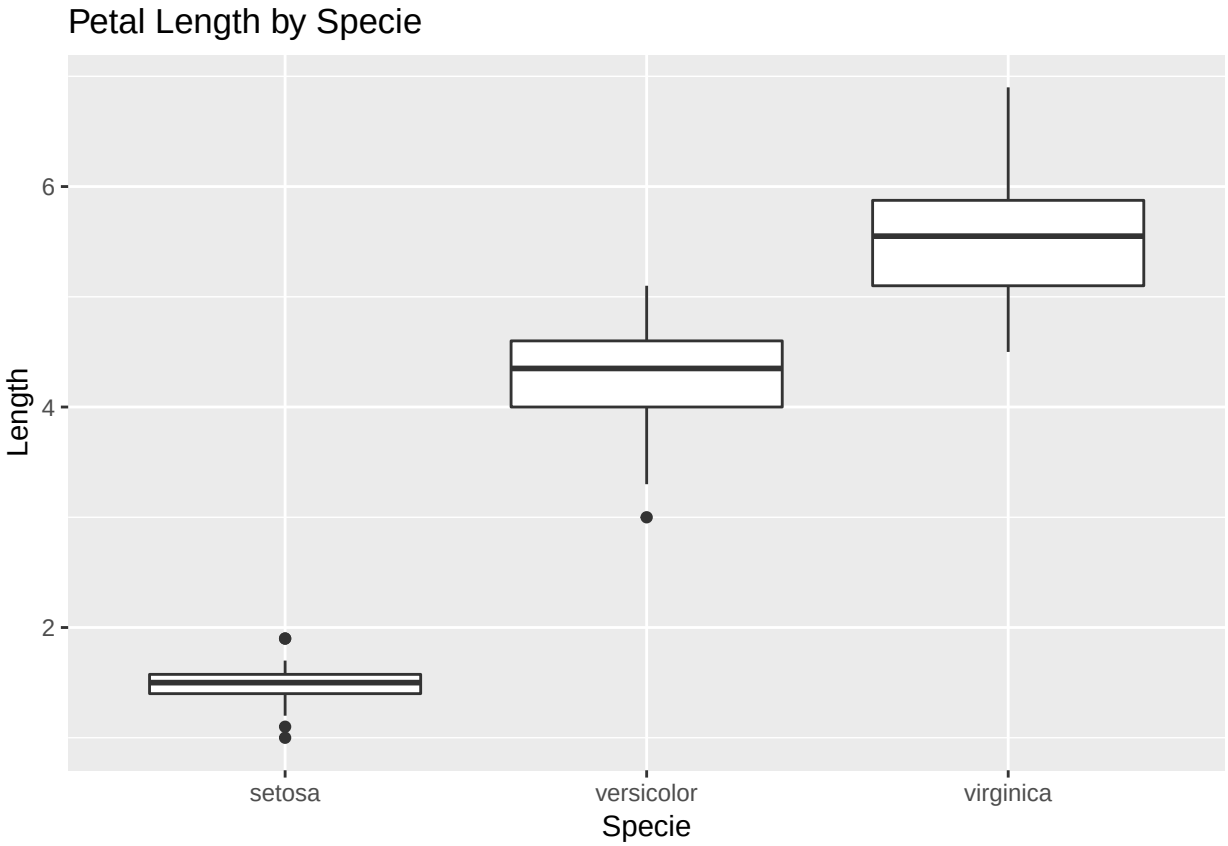


k. Realice una gráfica de cajas de la variable Petal.Length en la que se distingan las diferentes especies

Respuesta

La gráfica de cajas se realizó con ggplot, se muestra a continuación:

```
ggplot(iris, aes(Species, Petal.Length)) +
  geom_boxplot() +
  ggtitle("Petal Length by Specie") +
  xlab("Specie") +
  ylab("Length")
```



2. Espacio de Probabilidad y Variables Aleatorias

Considere un experimento que consiste en una carrera de caballos con tres caballos numerados del 1 al 3. Si no está permitido que dos o más caballos lleguen a la meta en la misma posición:

a. Cuál es el espacio de resultados Ω del experimento?

Respuesta

Como cada caballo puede quedar en cualquiera de los tres lugares pero solamente uno puede quedar en cada lugar, el espacio de resultados del experimento tiene un total de $3! = 6$ combinaciones.

$$\Omega = \{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2),(3,2,1)\}$$

b. Asumiendo que todos los elementos del espacio de resultados de $\omega \in \Omega$ tienen la misma probabilidad $P(\omega)$ de ocurrir: Cuál es esta probabilidad $P(\omega)$?

Respuesta

Al cada elemento del espacio de resultados tiene la misma probabilidad, es:

$$P(\omega) = \frac{1}{3!} = \frac{1}{6}$$

c. Si A denota el evento en el que el caballo número 1 llega a la meta dentro de las primeras dos posiciones y B denota el evento en el que el caballo número 3 llega a la meta en la segunda posición. Cuáles son los elementos de los eventos A y B respectivamente?

Respuesta

$A = \{(1,2,3),(1,3,2),(2,1,3),(3,1,2)\}.\ B = \{(1,3,2),(2,3,1)\}$

d. Cuáles son los elementos del evento $A \cap B$?

Respuesta

$$A \cap B = \{(1,3,2)\}$$

e. Cuáles son los elementos del evento $B \cup A$

Respuesta

$$A \cup B = \{(1,2,3),(1,3,2),(2,1,3),(2,3,1),(3,1,2)\}$$

f. Cuál es la probabilidad P(B) de que ocurra el evento B?

Respuesta

$$P(B) = \frac{2}{6} = \frac{1}{3}$$

g. Sea $X : \Omega \to \mathbb{R}$ una variable aleatoria que describe la posición en la que llegó a la meta el caballo número 2. Liste los valores $X(\omega)$ que toma la variable X para cada uno de los elementos $\omega \in \Omega$.

Respuesta

$$\begin{aligned}
X(\omega_0) &= X((1,2,3)) = 2 \\
X(\omega_1) &= X((1,3,2)) = 3 \\
X(\omega_2) &= X((2,1,3)) = 1 \\
X(\omega_3) &= X((2,3,1)) = 1 \\
X(\omega_4) &= X((3,1,2)) = 3 \\
X(\omega_5) &= X((3,2,1)) = 2
\end{aligned}$$

h. Cuál es la probabilidad $P(X = 1)$?

$$P(X = 1) = \frac{2}{6} = \frac{1}{3}$$

3. Probabilidad Condicional

Considere 3 urnas. La urna A contiene 2 pelotas blancas y 4 pelotas negras, la urna B contiene 8 pelotas blancas y 4 negras, la urna C contiene 1 pelota blanca y 3 negras. Si se selecciona 1 pelota de cada urna: Cuál es la probabilidad de que la pelota seleccionada de la urna A sea blanca dado que exactamente 2 de las 3 pelotas seleccionadas son blancas?

Respuesta.

Sea el evento X ~ La pelota de la urna A es blanca

Sea el evento Y ~ 2 de las 3 pelotas obtenidas fueron blancas

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)}$$

El espacio de resultados del evento X es:

$$\Omega_X = \{(b, b, n), (b, n, b)\}$$

El espacio de resultados del evento Y es:

$$\Omega_Y = \{(b, b, n), (b, n, b), (n, b, b)\}$$

y la probabilidad de cada elemento de Ω_B, dadas las probabilidades de cada urna son:

$$P((b, b, n)) = \frac{2}{6} \cdot \frac{8}{12} \cdot \frac{3}{4} = \frac{1}{6}$$
$$P((b, n, b)) = \frac{2}{6} \cdot \frac{4}{12} \cdot \frac{1}{4} = \frac{1}{36}$$
$$P((n, b, b)) = \frac{4}{6} \cdot \frac{8}{12} \cdot \frac{1}{4} = \frac{1}{9}$$

Entonces,

$$P(Y) = \frac{1}{6} + \frac{1}{36} + \frac{1}{9} = \frac{11}{36} \approx 30.6\%$$

Para la intersección de X con Y, es importante notar que el espacio de resultados de X está contenido en Y, por lo que:

$$P(X \cap Y) = P((b, b, n)) + P((b, n, b)) = \frac{1}{6} + \frac{1}{36} = \frac{7}{36} \approx 19.44\%$$

Por lo que:

$$P(X|Y) = \frac{P(X \cap Y)}{P(Y)} = \frac{\frac{7}{36}}{\frac{11}{36}} = \frac{7}{11} \approx 63.64\%$$

4. Bootstrap

Se desea simular muestras de tamaño 20 de una distribución exponencial con tasa $\lambda = \frac{1}{\beta} = 1$. El estadístico de interés es $\hat{\theta}$ el estimador de la media $\theta = \beta$ Siga el siguiente procedimiento: i. Utilice la función rexp() (y la semilla 261285) para generar una muestra aleatoria de tamaño 20 de una distribución exponencial con $\beta = 1$. ii. Genere 2,000 muestras bootstrap y calcule intervalos de confianza con coeficiente de confianza de 95% para $\hat{\theta}$ usando 1) el método normal, 2) percentiles y 3) BC_α iii. Revise si el intervalo de confianza contiene el verdadero valor del parámetro θ; en caso de que no lo contenga registre si falló por la izquierda o falló por la derecha.

a. Repita 500 veces el procedimiento descrito y llena la tabla (utilice una sola vez la semilla)

Respuesta

El procedimiento descrito se realiza en una función llamada "proc_examen". EL procedimiento se repite 500 veces usando la función rerun y se guarda el resultado en un arreglo. Se implementó una función para pasar de dicho arreglo a un "data frame" para poder filtrar los resultados de acuerdo a si el parámetro se encuentra o no en el intervalo (usando la función "filter"). Por último se obtienen los resultados como porcentaje y se imprimen.

```
proc_examen <- function()
{
  beta <- 1
    mediaBoot <- function(x)
    {
        muestra_boot <- sample(x, size = length(x), replace = TRUE)
        mean(muestra_boot)
    }
    muestra <- rexp(20, 1/beta)
    thetas_boot <- rerun(2000, mediaBoot(muestra)) %>% flatten_dbl()
    head(thetas_boot)
    theta_hat <-mean(muestra)

    li_normal <- round(theta_hat + qnorm(0.025) * sd(thetas_boot), 2)
    ls_normal <- round(theta_hat + qnorm(0.975) * sd(thetas_boot), 2)

    ls_per <- round(quantile(thetas_boot, prob = 0.975), 2)
    li_per <- round(quantile(thetas_boot, prob = 0.025), 2)

    b_var <- rerun(2000, mean(sample(muestra, size = length(muestra),
                replace = TRUE))) %>% flatten_dbl()
    ls_bc <- round(quantile(b_var, prob = 0.975), 2)
    li_bc <- round(quantile(b_var, prob = 0.025), 2)

    return(c(li_normal, ls_normal, li_per, ls_per, li_bc, ls_bc))
}

get_df_from_res <- function(arr)
{
    df <- tribble(
        ~prueba, ~li_normal, ~ls_normal, ~li_per, ~ls_per, ~li_bc, ~ls_bc
    )
```

```
    cont <- 1
    for (i in 1:length(arr))
    {
      switch(((i-1) %% 6)+1,
        li_normal <- arr[i],
        ls_normal <- arr[i],
        li_per <- arr[i],
        ls_per <- arr[i],
        {
            li_bc <- arr[i]
        },
        {
            ls_bc <- arr[i]
            df <- add_row(df, prueba = cont,
                        li_normal = li_normal,
                        ls_normal = ls_normal,
                        li_per = li_per,
                        ls_per = ls_per,
                        li_bc = li_bc,
                        ls_bc = ls_bc
                )
            cont <- cont + 1
        }
        )

    }
    return(df)
}
set.seed(261285)
res_aux <- rerun(500, proc_examen()) %>% flatten_dbl()
df_res_aux <- get_df_from_res(res_aux)

norm_izq <- nrow(filter(df_res_aux, li_normal > 1))
norm_der <- nrow(filter(df_res_aux, ls_normal < 1))
per_izq <- nrow(filter(df_res_aux, li_per > 1))
per_der <- nrow(filter(df_res_aux, ls_per < 1))
bc_izq <- nrow(filter(df_res_aux, li_bc > 1))
bc_der <- nrow(filter(df_res_aux, ls_bc < 1))

norm_cob <- nrow(filter(df_res_aux, li_normal <= 1, 1 <= ls_normal))
per_cob <- nrow(filter(df_res_aux, li_per <= 1, 1 <= ls_per))
bc_cob <- nrow(filter(df_res_aux, li_bc <= 1, 1 <= ls_bc))

div_var <- 5

df <- tribble(
        ~Metodo, ~"% Fallo Izquierda", ~"% Fallo Derecha", ~"% Cobertura (Simulaciones)",
        "Normal", toString(round(norm_izq/div_var, 2)),
          toString(round(norm_der/div_var, 2)), toString(round(norm_cob/div_var, 2)),
        "Percentiles",toString(round(per_izq/div_var, 2)),
          toString(round(per_der/div_var, 2)), toString(round(per_cob/div_var, 2)),
        "BC_a", toString(round(bc_izq/div_var, 2)),
          toString(round(bc_der/div_var, 2)), toString(round(bc_cob/div_var, 2))
```

```
    )

df %>%
  kable() %>%
  kable_styling()
```

Metodo	% Fallo Izquierda	% Fallo Derecha	% Cobertura (Simulaciones)
Normal	0.8	9.6	89.6
Percentiles	1.4	8.2	90.4
BC_a	1.4	8	90.6

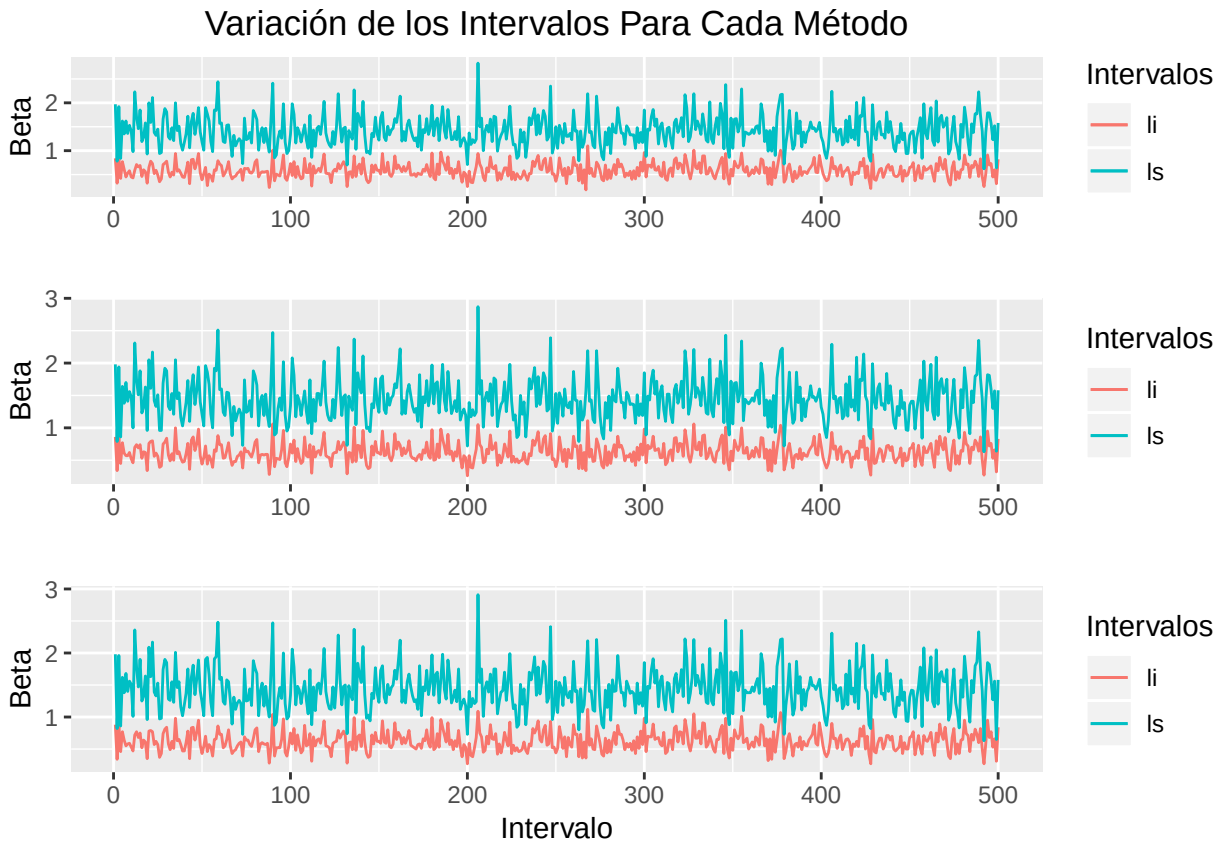
b. Realice una gráfica de paneles: en cada panel mostrará los resultados de uno de los métodos (normal, percentil y), en el eje horizontal graficará el número de intervalos de confianza (1,2,. . . ,500) y en el eje vertical graficará los límites de los intervalos, es decir, graficará 2 líneas (use la función geom_line): una corresponderá a los límites inferiores de los intervalos y la otra a los superiores.

Respuesta

Se realiza una gráfica para cada uno de los métodos y después se realiza una sola gráfica (multiplot) utilizando un script del libro "Cookbook for R".

```
p_norm <- ggplot() +
    geom_line(data = df_res_aux, aes(x = prueba, y = li_normal, color = "blue")) +
    geom_line(data = df_res_aux, aes(x = prueba, y = ls_normal, color = "red")) +
    xlab("") +
  ylab("Beta") +
  scale_color_discrete(name = "Intervalos", labels = c("li", "ls")) +
    ggtitle("Variación de los Intervalos Para Cada Método") +
  theme(plot.title = element_text(hjust = 0.5))
p_per <- ggplot() +
    geom_line(data = df_res_aux, aes(x = prueba, y = li_per, color = "blue")) +
    geom_line(data = df_res_aux, aes(x = prueba, y = ls_per, color = "red")) +
    xlab("") +
  ylab("Beta") +
  scale_color_discrete(name = "Intervalos", labels = c("li", "ls"))

p_bc <- ggplot() +
    geom_line(data = df_res_aux, aes(x = prueba, y = li_bc, color = "blue")) +
    geom_line(data = df_res_aux, aes(x = prueba, y = ls_bc, color = "red")) +
    xlab("Intervalo") +
  ylab("Beta") +
  scale_color_discrete(name = "Intervalos", labels = c("li", "ls"))
multiplot(p_norm, p_per, p_bc, cols = 1)
```



5. Simulación de Variables Aleatorias

Una variable aleatoria X tiene una distribución binomial con parámetros n y p, esto es, X~Binomial(n, p), si su función de masa de probabilidad es:

$$p_i = P\{X = i\} = \left(\begin{array}{c} n \\ i \end{array} \right) \cdot p^i \cdot (1-p)^{n-i}, i \in \{0, 1, \ldots, n\}$$

El objetivo es generar X a partir de una variable aleatoria U con distribución uniforme continua en el itervalo (0,1) utilizando el método de Transformación Inversa Discreta. La clave para utilizar este método en el presente caso es seguir el procedimiento indicado.

a. Encuentre la relación de recurrencia entre p_{i+1} y p_i para i > 0.

Respuesta

Para encontrar la relación de recurrencia entre p_{i+1} y p_i, se empieza sustituyendo en la función de masa de probabilidad para cuando $i = 0$, obteniendo:

$$P_0 = (1-p)^n$$

A partir de esto, es posible ir construyendo p_{i+1} para $i >= 1$. Dado que la función de masa de probabilidad tiene un término $(1-p)^(n-i)$, basta con dividir entre $(1-p)$ para cada nueva i. Similarmente, a cada nueva

i se multiplica por p para que al i-esimo término se tenga p^i. Para obtener la parte de las combinaciones es importante recordar que $\binom{n}{i} = \frac{n!}{i!\cdot(n-i)!}$, obteniendo que para el i-esimo término, se tiene que:

$$\frac{1\cdot 2\cdot \ldots \cdot (n-i)\cdot (n-i+1)\cdot \ldots \cdot n}{(1\cdot 2\cdot \ldots \cdot i)(1\cdot 2\cdot \ldots \cdot (n-i))} = \frac{(n-i+1)\cdot \ldots \cdot n}{(1\cdot 2\cdot \ldots \cdot i)}$$

Con lo que se tiene que $(1\cdot 2\cdot \ldots \cdot x)$ equivale a i para el i-ésimo término. Similarmente, $((n-i+1)\cdot \ldots \cdot n)$ equivale a $(n-i+1)$ para el i-ésimo término.

Entonces, juntando todos los términos que se construyeron previamente, se tiene que:

$$P_{i+1} = \frac{P_i(n-i+1)p}{i(1-p)} \quad \forall i \in \{1,\ldots,n\}$$

En palabras, empezando con P_0 como n veces $(1-p)$, a cada nueva P_i se le *agrega* p y se le *quitan* tanto i como $(1-p)$, esto es a cada iteración se reduce en uno el exponente de $(1-p)$ mientras que la divisón entre i sirve para formar el factorial.

b. Utilizando la relación de recurrencia del inciso anterior, escribir un algoritmo de 5 pasos que genere una variable aleatoria binomial con parámetros n y p mediante el método de Transformación Inversa Discreta.

Respuesta

1. Generar un número aleatorio a partir de una distribución Uniforme(0,1)
2. Inicializar $i = 1$, $P = (1-p)^n$, $F = p$
3. Si $U < F$ regresar $i-1$, en otro caso seguir a 4
4. $p_i = \frac{P_{i-1}(n-i+1)p}{i(1-p)}$, $F = F + p_i$, $i = i+1$
5. Volver a 3

c. Escriba en R una función que implemente el algoritmo del inciso b para n = 10 y p = 0.3

Respuesta

A continuación se presenta la función en R que implementa el algoritmo descrito anteriormente. La función recibe dos parámetros n y p. En particular, la función se prueba con $n = 10$ y $p = 0.3$.

```
set.seed(221285)
rBinomialI <- function(n, p)
  {
  U <- runif(1)
  i <- 1
  p_i <- (1 - p) ^ n
  Fu <- p_i
  while(U >= Fu){

    p_i <- (p_i / (1 - p) ) * (n - i + 1) * p / i
    Fu <- Fu + p_i
    i <- i + 1
  }
```

```
  i - 1
}
rBinomialI(10, 0.3)
```

```
## [1] 6
```

d. Realice 10,000 simulaciones utilizando la semilla 221285 y reporte las primeras 5 simulaciones obtenidas

Respuesta

En el siguiente código de R, primero se establece la semilla. Despues se realizan las 10,000 simulaciones utilizando la función *rerun*. Para reportar las primeras 5 simulaciones se utiliza el operados "[]" para obtener los primeros 5 valores.

```
set.seed(221285)
binomial_aux <- rerun(10000, rBinomialI(10, 0.3)) %>% flatten_dbl()
binomial_aux[1:5]
```

```
## [1] 6 2 4 1 4
```

e. Genere un histograma con las 10,000 simulaciones anteriores y compárelo con una distribución construida utilizando la función dbinom de R.

Respuesta

A continuación se muestra el código completo para generar 10,000 simulaciones utilizando el algoritmo implmentado anterioremente así como utilizando la función nativa de R. En la gráfica se presenta la frecuencia obtenida para cada valor que se obtuvo en las 10,000 simulaciones. En ésta gráfica se puede ver que hay diferencias mínimas entre ambos métodos, en general la *forma* de la distribución coincide con la esperada para una variable aleatoria binomial. Al incrementar el número de simulaciones, deberían tender a ser iguales.

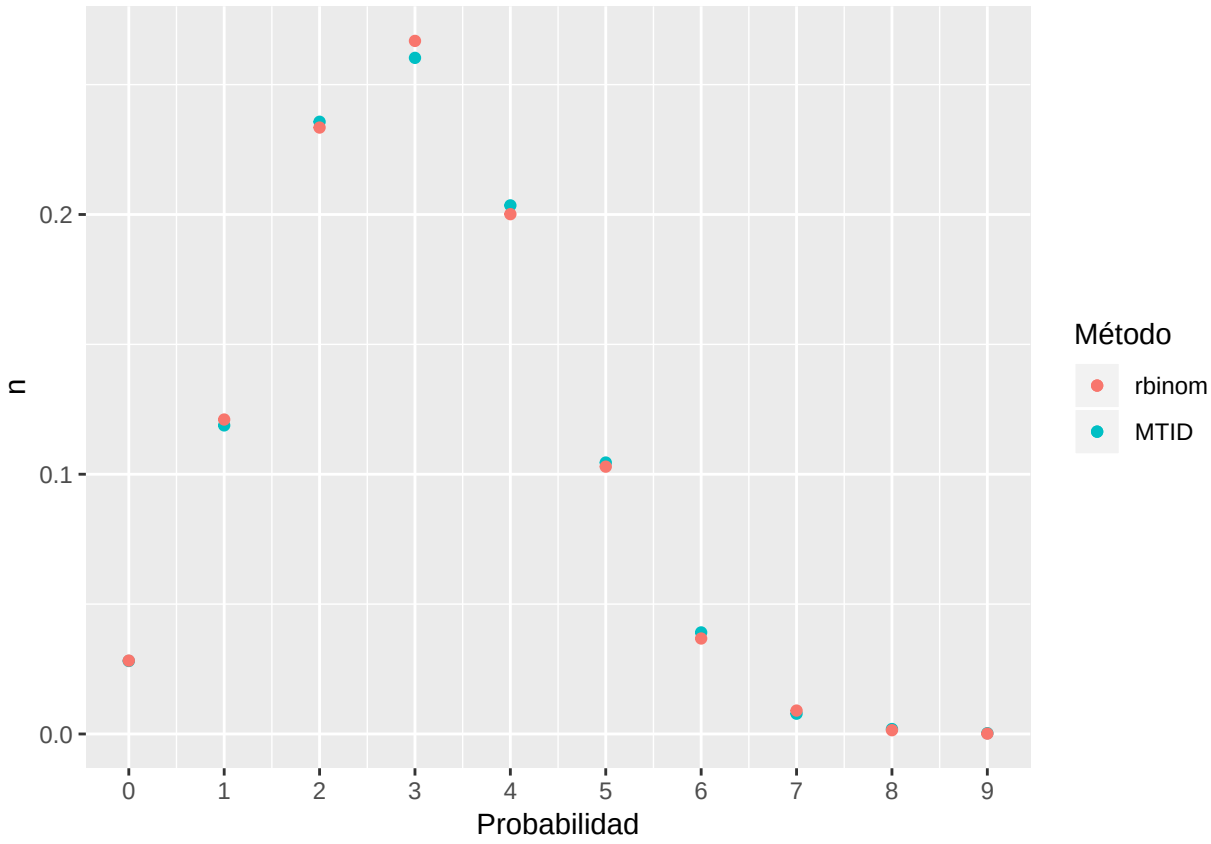
```
set.seed(221285)
binomial_aux <- rerun(10000, rBinomialI(10, 0.3)) %>% flatten_dbl()
df_binomial_aux <- as.data.frame(0:9)

df_binomial_aux <- mutate(df_binomial_aux, MTID = table(binomial_aux))
df_binomial_aux$MTID <- df_binomial_aux$MTID / rep(10000, 10)

rbinom_v <- dbinom(0:9, 10, 0.3)
df_binomial_aux <- mutate(df_binomial_aux, rbin = rbinom_v)
names(df_binomial_aux) <- c("i", "MTID", "rbin")

ggplot() +
    geom_point(data = df_binomial_aux,
        aes(x = i, y = MTID, color = "red")) +
    geom_point(data = df_binomial_aux,
        aes(x = i, y = rbin, color = "blue")) +
    xlab("Probabilidad") + ylab("n") +
    scale_color_discrete(name = "Método", labels = c("rbinom", "MTID")) +
  scale_x_continuous(breaks = 0:9)
```

```
## Don't know how to automatically pick scale for object of type table. Defaulting to continuous.
```


0.2
0.1
0.0
n
0
1
2
3
4
5
6
7
8
9
Probabilidad
Método
rbinom
MTID