

1. Stopping generating new simulation data

Write a program to generate standard normal random variables until you have generated n of them, where $n \geq 100$ is such that $S/\sqrt{n} < 0,01$, where S is the sample standard deviation of the n data values. Note that this is the “Method for Determining When to Stop Generating New Data”. Also, answer the following questions:

Debido a que se va a generar variables aleatorias **normales estándar** X_i , es decir, $X_i \sim \mathcal{N}(0, 1)$ y se debe parar de generar cuando $\frac{S}{\sqrt{n}} < 0,01$.

```
x <- numeric()
s <- 1
n <- 0

while(n < 100 || s / sqrt(n) >= 0.01) {
  x <- c(x, rnorm(1))
  n <- length(x)
  s <- sd(x)
}

cat("Cantidad de Xi generados", n, "\n")

## Cantidad de Xi generados 10195

cat("Desviación estándar", round(s, 5), "\n")

## Desviación estándar 1.00968

cat("Error estándar S/sqrt(n)", round(s/sqrt(n), 5), "\n")

## Error estándar S/sqrt(n) 0.01
```

En el código se declara una variable s inicializada en 1 para que luego sea nuevamente computada a la desviación estándar de x que es un vector que va a almacenar todas las variables normales aleatorias generadas. Según el enunciado tenemos una condición de que al menos deben haber 100 variables aleatorias generadas ($n \geq 100$) y que el error estándar sea menor a 0.01 ($S/\sqrt{n} < 0,01$). En el ciclo **while**, por lo tanto, tiene sentido que el ciclo continúe si alguna de las afirmaciones anteriores son falsas, por eso queda la condición de esa manera. Se hace uso de la función **rnorm(1)** para que genere una variable aleatoria normal con los valores por defecto (promedio 0 y desviación 1).

1.1. How many normals do you think will be generated? Give an analytic estimate.

1.1.1. Respuesta

Aunque ya se tengan los resultados de la simulación, se puede hacer un estimado analítico con la condición $S/\sqrt{n} < 0,01$ donde se puede despejar n para saber cuántas variables se necesitan para parar el criterio.

Se tiene en primer lugar la inecuación:

$$\frac{S}{\sqrt{n}} < 0,01$$

Se eleva ambas partes con menos 1:

$$\left(\frac{S}{\sqrt{n}}\right)^{-1} > 0,01^{-1}$$

$$\frac{\sqrt{n}}{S} > 100$$

Multiplicando ambas partes por S :

$$\cancel{S} \times \frac{\sqrt{n}}{\cancel{S}} > 100 \times S$$

$$\sqrt{n} > 100 \times S$$

Ahora se cancela la raíz elevando ambas partes al cuadrado:

$$(\sqrt{n})^2 > (100 \times S)^2$$

Quedando entonces:

$$n > 10000 \times S^2$$

Para hacer el ejercicio se utilizó en R la función `rnorm(1, mean = 0, sd = 1)`, de esta manera genera únicamente 1 valor con media 0 y desviación 1, por lo tanto, podemos hacer $S = 1$ para estimar cuántos n necesitamos para que se cumpla condición y deje de generar variables aleatorias, por lo tanto:

$$n > 10000 \times (1)^2$$

$$n > 10000$$

Este valor se acerca bastante al que se imprime en la simulación.

1.2. How many normals did you generate?

1.2.1. Respuesta

Se han generado $n=10195$ normales.

1.3. What is the sample mean of all the normals generated?

1.3.1. Respuesta

Utilizando el comando `mean()` al vector `x` se obtiene $-0,0025131$.

1.4. What is the sample variance?

1.4.1. Respuesta

La varianza muestral se calcula como:

$$S^2 = \frac{\sum_{i=1}^n (X_i - \bar{X})^2}{n - 1}$$

Esto se puede sacar con la función `var()` de R, por lo tanto, aplicando eso al vector `x` se obtiene $1,0194571$.

1.5. Comment on the results of (1.3) and (1.4). Were they surprising?

1.5.1. Respuesta

El resultado de 1.3 es la media de `x` que es $-0,0025131$ y el de 1.4 es la varianza muestral de `x` dando $1,0194571$. Estos valores muy sorprendentes no fueron... Se espera que al simular muchas veces una distribución normal, esta nos dé los valores de la media y la desviación estándar al cuadrado. Pero sí es interesante que dado este algoritmo para detenerse después de cruzar un S/\sqrt{n} nos dé valores muy cercanos a una distribución normal con $\mu = 0$ y $\sigma = 1$.

2. Gaining confidence with confidence intervals

We know that the $\mathcal{U}(-1, 1)$ r.v. has mean 0. Use a sample of size 1000 to estimate the mean and give a 95 % confidence interval (CI). Does the CI contain 0? Repeat the above a large number of times (≥ 100). What percentage of time does the CI contain 0? Write your code so that it produces output similar to the following:

```
Number of trials: 10

Sample mean  lower bound  upper bound  contains mean?

-0.0733      -0.1888      0.0422         1
-0.0267      -0.1335      0.0801         1
-0.0063      -0.1143      0.1017         1
-0.0820      -0.1869      0.0230         1
-0.0354      -0.1478      0.0771         1
-0.0751      -0.1863      0.0362         1
-0.0742      -0.1923      0.0440         1
 0.0071      -0.1011      0.1153         1
 0.0772      -0.0322      0.1867         1
-0.0243      -0.1370      0.0885         1

100 percent of CI's contained the mean
```

2.1. Respuesta

Se sabe que para tener un intervalo de confianza de 95 %, el límite inferior y superior serán, respectivamente:

$$\left(\bar{X} - 1,96 \frac{S}{\sqrt{n}}, \bar{X} + 1,96 \frac{S}{\sqrt{n}} \right)$$

El siguiente código permite sacar el intervalo de confianza del 95 % y también el estimado de la media.

```
n <- 1000
x <- runif(n, -1, 1)
xmean <- mean(x)
S <- sd(x)
L <- xmean - 1.96 * S/sqrt(n)
U <- xmean + 1.96 * S/sqrt(n)
cat("El estimado es", xmean, "\n")

## El estimado es -0.005261789

cat("95% está entre (", L, ", ", U, ") \n", sep="")

## 95% está entre (-0.04109755, 0.03057397)
```

Aquí se estimó la media de la distribución uniforme, dando un resultado de $-0,0052618$, muy cercano a 0. El intervalo de confianza está entre $-0,0410976$ y $0,030574$, en este caso sí contiene a 0.¹ Ahora, se va a realizar el proceso unas 200 veces para conocer el porcentaje de intervalos de confianza que contienen 0.

```
trials <- 200
n <- 1000
true_mean <- 0

results <- data.frame(
  sample_mean = numeric(trials),
  lower_bound = numeric(trials),
  upper_bound = numeric(trials),
  contains_mean = integer(trials)
)

for (i in 1:trials){
  x <- runif(n, -1, 1)
  xmean <- mean(x)
  S <- sd(x)
  L <- xmean - 1.96 * S/sqrt(n)
  U <- xmean + 1.96 * S/sqrt(n)
  contains <- as.integer(L <= true_mean & true_mean <= U)

  results[i, ] <- c(xmean, L, U, contains)
}

cat("Número de intentos:", trials, "\n")

## Número de intentos: 200

print(results, digits= 4)

##      sample_mean lower_bound upper_bound contains_mean
## 1      0.0190973  -0.0165813    0.054776             1
## 2      0.0006915  -0.0352762    0.036659             1
## 3     -0.0172115  -0.0530536    0.018631             1
## 4      0.0050162  -0.0310239    0.041056             1
## 5      0.0009417  -0.0342249    0.036108             1
## 6      0.0167117  -0.0195190    0.052942             1
## 7      0.0197878  -0.0159063    0.055482             1
## 8     -0.0190615  -0.0540806    0.015958             1
```

¹Aquí hay que hacer una aclaración, los autores utilizaron knitr, una herramienta para la generación de reportes dinámicos en R, entonces cada vez que se compile este archivo .Rnw va a salir una salida diferente y puede ser que ya el intervalo no contenga 0.

## 9	-0.0009254	-0.0374129	0.035562	1
## 10	-0.0049668	-0.0397895	0.029856	1
## 11	-0.0156264	-0.0510312	0.019778	1
## 12	0.0357229	-0.0001320	0.071578	1
## 13	0.0106253	-0.0263896	0.047640	1
## 14	0.0033504	-0.0320010	0.038702	1
## 15	-0.0327945	-0.0684779	0.002889	1
## 16	0.0147764	-0.0200589	0.049612	1
## 17	0.0146584	-0.0203104	0.049627	1
## 18	0.0198254	-0.0163845	0.056035	1
## 19	0.0064255	-0.0299121	0.042763	1
## 20	-0.0361921	-0.0721422	-0.000242	0
## 21	0.0148459	-0.0216631	0.051355	1
## 22	-0.0170990	-0.0527240	0.018526	1
## 23	-0.0179286	-0.0534877	0.017631	1
## 24	0.0157888	-0.0209565	0.052534	1
## 25	0.0117639	-0.0240530	0.047581	1
## 26	0.0404388	0.0040027	0.076875	0
## 27	-0.0090648	-0.0441665	0.026037	1
## 28	-0.0155558	-0.0520774	0.020966	1
## 29	0.0296623	-0.0057954	0.065120	1
## 30	-0.0151422	-0.0503697	0.020085	1
## 31	-0.0168390	-0.0529613	0.019283	1
## 32	0.0009831	-0.0349295	0.036896	1
## 33	-0.0249255	-0.0609619	0.011111	1
## 34	0.0277939	-0.0084169	0.064005	1
## 35	-0.0217032	-0.0579757	0.014569	1
## 36	-0.0232368	-0.0582778	0.011804	1
## 37	0.0180046	-0.0174665	0.053476	1
## 38	0.0200399	-0.0156350	0.055715	1
## 39	0.0200478	-0.0156946	0.055790	1
## 40	0.0007808	-0.0351802	0.036742	1
## 41	-0.0193217	-0.0544670	0.015824	1
## 42	-0.0178301	-0.0540222	0.018362	1
## 43	0.0212798	-0.0147037	0.057263	1
## 44	-0.0108521	-0.0472193	0.025515	1
## 45	0.0042346	-0.0314267	0.039896	1
## 46	0.0347303	-0.0009823	0.070443	1
## 47	-0.0063150	-0.0425929	0.029963	1
## 48	-0.0035331	-0.0393528	0.032287	1
## 49	0.0099811	-0.0259275	0.045890	1
## 50	0.0210413	-0.0150911	0.057174	1
## 51	0.0119365	-0.0235484	0.047421	1
## 52	0.0012126	-0.0347209	0.037146	1

## 53	0.0059158	-0.0292288	0.041060	1
## 54	-0.0067596	-0.0427808	0.029262	1
## 55	0.0034415	-0.0317771	0.038660	1
## 56	0.0033653	-0.0325309	0.039262	1
## 57	-0.0411725	-0.0764317	-0.005913	0
## 58	-0.0113363	-0.0473355	0.024663	1
## 59	0.0077015	-0.0274624	0.042865	1
## 60	0.0096264	-0.0266963	0.045949	1
## 61	0.0156618	-0.0202477	0.051571	1
## 62	0.0039682	-0.0314281	0.039365	1
## 63	0.0243906	-0.0109005	0.059682	1
## 64	-0.0010009	-0.0370900	0.035088	1
## 65	0.0009671	-0.0342610	0.036195	1
## 66	0.0184944	-0.0161350	0.053124	1
## 67	0.0055586	-0.0304921	0.041609	1
## 68	0.0006418	-0.0348899	0.036174	1
## 69	0.0013723	-0.0339824	0.036727	1
## 70	-0.0171070	-0.0537283	0.019514	1
## 71	0.0130293	-0.0230269	0.049085	1
## 72	0.0238591	-0.0113132	0.059031	1
## 73	-0.0118539	-0.0471194	0.023412	1
## 74	0.0116290	-0.0245629	0.047821	1
## 75	-0.0152639	-0.0515358	0.021008	1
## 76	-0.0125347	-0.0486270	0.023558	1
## 77	0.0005846	-0.0353099	0.036479	1
## 78	-0.0028849	-0.0393244	0.033555	1
## 79	0.0062766	-0.0294273	0.041980	1
## 80	0.0145287	-0.0208303	0.049888	1
## 81	0.0157828	-0.0195164	0.051082	1
## 82	-0.0265286	-0.0624618	0.009405	1
## 83	-0.0044438	-0.0399240	0.031036	1
## 84	0.0251749	-0.0111674	0.061517	1
## 85	0.0215557	-0.0133675	0.056479	1
## 86	0.0223580	-0.0135729	0.058289	1
## 87	0.0221922	-0.0137225	0.058107	1
## 88	-0.0235150	-0.0599032	0.012873	1
## 89	-0.0052702	-0.0416449	0.031105	1
## 90	0.0004221	-0.0361149	0.036959	1
## 91	0.0154551	-0.0220895	0.053000	1
## 92	-0.0217820	-0.0580672	0.014503	1
## 93	-0.0177621	-0.0533681	0.017844	1
## 94	-0.0267001	-0.0624480	0.009048	1
## 95	0.0016699	-0.0336313	0.036971	1
## 96	0.0035390	-0.0328130	0.039891	1

## 97	0.0196651	-0.0166677	0.055998	1
## 98	0.0256741	-0.0102678	0.061616	1
## 99	-0.0316338	-0.0672525	0.003985	1
## 100	-0.0218883	-0.0576997	0.013923	1
## 101	0.0242256	-0.0108991	0.059350	1
## 102	-0.0268067	-0.0620269	0.008413	1
## 103	0.0231499	-0.0122020	0.058502	1
## 104	0.0142955	-0.0220525	0.050643	1
## 105	-0.0008219	-0.0371832	0.035539	1
## 106	0.0110842	-0.0248332	0.047002	1
## 107	-0.0193765	-0.0553223	0.016569	1
## 108	0.0019574	-0.0342139	0.038129	1
## 109	-0.0161749	-0.0523819	0.020032	1
## 110	0.0113998	-0.0246569	0.047457	1
## 111	-0.0135126	-0.0486284	0.021603	1
## 112	-0.0177022	-0.0534907	0.018086	1
## 113	-0.0187500	-0.0551701	0.017670	1
## 114	-0.0420652	-0.0776491	-0.006481	0
## 115	-0.0127181	-0.0485226	0.023086	1
## 116	0.0061241	-0.0293559	0.041604	1
## 117	0.0009940	-0.0350835	0.037071	1
## 118	0.0075902	-0.0271586	0.042339	1
## 119	-0.0080311	-0.0443227	0.028260	1
## 120	0.0105961	-0.0253383	0.046531	1
## 121	0.0031948	-0.0325070	0.038897	1
## 122	-0.0021946	-0.0383281	0.033939	1
## 123	-0.0148536	-0.0508382	0.021131	1
## 124	0.0166196	-0.0190386	0.052278	1
## 125	0.0217711	-0.0140582	0.057600	1
## 126	-0.0290650	-0.0647404	0.006610	1
## 127	0.0044614	-0.0318589	0.040782	1
## 128	-0.0097796	-0.0463608	0.026802	1
## 129	0.0286095	-0.0068284	0.064047	1
## 130	0.0172588	-0.0180861	0.052604	1
## 131	0.0104759	-0.0255872	0.046539	1
## 132	0.0196068	-0.0152294	0.054443	1
## 133	0.0400061	0.0041707	0.075841	0
## 134	0.0028546	-0.0335591	0.039268	1
## 135	0.0065896	-0.0297413	0.042921	1
## 136	0.0016451	-0.0340673	0.037357	1
## 137	0.0012847	-0.0344430	0.037012	1
## 138	0.0349889	-0.0009886	0.070966	1
## 139	-0.0054164	-0.0400443	0.029212	1
## 140	0.0042175	-0.0313427	0.039778	1

## 141	-0.0127629	-0.0481735	0.022648	1
## 142	-0.0019549	-0.0371562	0.033246	1
## 143	-0.0237352	-0.0596701	0.012200	1
## 144	-0.0107051	-0.0467634	0.025353	1
## 145	-0.0076831	-0.0439149	0.028549	1
## 146	0.0123975	-0.0237456	0.048541	1
## 147	0.0029451	-0.0330856	0.038976	1
## 148	-0.0156805	-0.0520000	0.020639	1
## 149	0.0177160	-0.0184124	0.053844	1
## 150	-0.0093149	-0.0452167	0.026587	1
## 151	-0.0260725	-0.0612304	0.009085	1
## 152	0.0092927	-0.0261540	0.044739	1
## 153	0.0110982	-0.0245151	0.046712	1
## 154	0.0090059	-0.0267029	0.044715	1
## 155	-0.0030013	-0.0393594	0.033357	1
## 156	-0.0004926	-0.0367019	0.035717	1
## 157	0.0106008	-0.0253863	0.046588	1
## 158	0.0138385	-0.0219629	0.049640	1
## 159	-0.0211853	-0.0558648	0.013494	1
## 160	-0.0286571	-0.0638217	0.006507	1
## 161	-0.0082433	-0.0433348	0.026848	1
## 162	0.0269112	-0.0092410	0.063063	1
## 163	0.0009838	-0.0352696	0.037237	1
## 164	-0.0009192	-0.0365940	0.034756	1
## 165	-0.0179742	-0.0537261	0.017778	1
## 166	0.0266639	-0.0084088	0.061737	1
## 167	0.0066789	-0.0287529	0.042111	1
## 168	0.0121641	-0.0237917	0.048120	1
## 169	0.0213762	-0.0151332	0.057886	1
## 170	-0.0120802	-0.0479467	0.023786	1
## 171	-0.0259919	-0.0613285	0.009345	1
## 172	0.0153246	-0.0210069	0.051656	1
## 173	0.0239552	-0.0118530	0.059763	1
## 174	-0.0051102	-0.0411904	0.030970	1
## 175	-0.0033103	-0.0390094	0.032389	1
## 176	0.0070863	-0.0288949	0.043067	1
## 177	0.0364751	0.0013864	0.071564	0
## 178	-0.0066137	-0.0423129	0.029086	1
## 179	0.0318241	-0.0046502	0.068299	1
## 180	-0.0054969	-0.0408790	0.029885	1
## 181	0.0077178	-0.0269740	0.042410	1
## 182	0.0034179	-0.0324980	0.039334	1
## 183	0.0469500	0.0114522	0.082448	0
## 184	0.0157090	-0.0210366	0.052455	1

```
## 185 -0.0061476 -0.0417655 0.029470 1
## 186 0.0278131 -0.0080892 0.063715 1
## 187 0.0166306 -0.0190230 0.052284 1
## 188 0.0081387 -0.0278542 0.044132 1
## 189 0.0092457 -0.0265717 0.045063 1
## 190 0.0151626 -0.0209521 0.051277 1
## 191 -0.0150389 -0.0506305 0.020553 1
## 192 0.0117003 -0.0237745 0.047175 1
## 193 -0.0168972 -0.0527803 0.018986 1
## 194 -0.0162271 -0.0514073 0.018953 1
## 195 -0.0195571 -0.0556126 0.016498 1
## 196 0.0309929 -0.0043813 0.066367 1
## 197 -0.0113080 -0.0470059 0.024390 1
## 198 -0.0128589 -0.0494615 0.023744 1
## 199 0.0066233 -0.0291551 0.042402 1
## 200 0.0051234 -0.0311658 0.041413 1

porcentaje <- mean(results$contains_mean) * 100

cat("\n", porcentaje, "% de los intervalos de",
    "confianza contienen la media real\n")

##
## 96.5 % de los intervalos de confianza contienen la media real
```

En conclusión, el 96,5 % de los intervalos generados incluyen al 0.

3. Standard deviation of a proportion

Assume a manager is using the sample proportion \hat{p} to estimate the proportion p of a new shipment of computer chips that are defective. He doesn't know p for this shipment, but in previous shipments it has been close to 0,01, that is 1 % of chips have been defective.

3.1. If the manager wants the standard deviation of \hat{p} to be about 0,02, how large a sample should she take based on the assumption that the rate of defectives has not changed dramatically?

En clase se vieron los estimadores de probabilidad, donde se quiere estimar

$$p = (PX \in A)$$

Donde A es el subconjunto del espacio de estados de Ω de X . Es decir, para nuestro

problema, este subconjunto de espacios en la muestra que tomó la administradora del cargamento de chips defectuosos. Se puede definir la variable indicadora Z como:

$$Z = \begin{cases} 1, & X \in A \\ 0, & X \notin A \end{cases}$$

1 significa que sí está defectuoso y 0 que no lo está. Se puede escribir el estimador como:

$$p = E[Z]$$

En clase se realizó una demostración de como el valor esperado de Z se le puede asignar a p . Se define la varianza de Z como:

$$\begin{aligned} \text{Var}(Z) &= E[Z^2] - E[Z]^2 \\ &= (1^2 \times P(X \in A) + 0^2 \times P(X \notin A)) - (P(X \in A))^2 \\ &= p - p^2 \end{aligned}$$

Aplicando factorización:

$$\text{Var}(Z) = p(1 - p)$$

Estimar p se puede realizar mediante el promedio muestral de Z , es decir, se puede estimar la proporción \hat{p} con el promedio de los chips que son defectuosos:

$$\hat{p} = \frac{1}{n} \sum_{i=1}^n Z_i$$

Se puede calcular entonces, la varianza del estimador:

$$\text{Var}(\hat{p}) = \frac{1}{n} \text{Var} \left(\sum_{i=1}^n Z_i \right)$$

Por independencia de los valores del cargamento de chips Z_i se puede meter la varianza de estos en la suma:

$$\begin{aligned} \text{Var}(\hat{p}) &= \frac{1}{n^2} \sum_{i=1}^n \text{Var}(Z_i) \\ &= \frac{1}{n^2} \sum_{i=1}^n p(1 - p) = \frac{1}{n} p(1 - p) \end{aligned}$$

Por lo tanto, una aproximación es:

$$\text{Var}(\hat{p}) \approx \frac{1}{n} \hat{p}(1 - \hat{p})$$

Teniendo en cuenta que la desviación estándar es el cuadrado de la varianza, se puede despejar:

$$\sigma_{\hat{p}} \approx \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}}$$

El problema nos dice que la desviación estándar de \hat{p} ($\sigma_{\hat{p}}$) debe ser sobre 0.02 y nos piden hallar cuántas muestras debe tomar la administradora, por lo tanto, en la parte izquierda de la ecuación reemplazamos por 0,02, se despeja para n y la proporción de defectuosos se toma como 0,01 debido a que en el problema se nos dice que no ha cambiado tanto.

$$\begin{aligned}\sigma_{\hat{p}} &\approx \sqrt{\frac{\hat{p}(1 - \hat{p})}{n}} \\ 0,02 &\approx \sqrt{\frac{0,1(1 - 0,1)}{n}} \\ 0,02 &\approx \sqrt{\frac{0,0099}{n}} \\ (0,02)^2 &\approx \left(\sqrt{\frac{0,0099}{n}} \right)^2 \\ 0,0004 &\approx \frac{0,0099}{n} \\ n &\approx \frac{0,0099}{0,0004} = 24,75\end{aligned}$$

Por lo tanto, la administradora debe tomar una muestra de por lo menos 25 chips para tener una desviación estándar de al menos 0,02