Universitat Autònoma de Barcelona

Facultat de Ciències

Pràctica 2

Autors:

Gerard Lahuerta & Ona Sánchez

1601350 — 1601181

29 de Abril del 2022

Índex

1 Presentació de la llibreria i informació d'interés

La llibreria de funcions ha estat programada en llenguatge C i consta de tres fitxers, un fitxer capcelera i dos amb el codi de les funcions.

Aquests dos fitxers amb codi: un que precisa de la llibreria gsl_rng.h per al seu funcionament, i un altre que usa math.h, limits.h, stdlib.h.

El fitxer capcelera, per contra, es comú als dos fitxers.

També, els fitxers capcelera tenen definides les constants:

- E_INTER: missatge d'error per valor inferior del interval major al superior.
- E_SIGMA: missatge d'error per $\sigma^2 \leq 0$.

La llibreria té programades tres funcions, on dues permeten generar nombres aleatoris (cadascuna amb una distribució diferent). Ambdues retornen un sol valor aleatori, que és de tipus **double**. La tercera funció permet inicialitzar la llavor de generació dels nombres aleatoris.

Recalcar a més, que cada funció conté el seu *CDocs* propi que conté la informació més rellevant de la funció de manera resumida: *input, output, descripció* i *informació rellevant.*

Per finalitzar, comentar que s'ha decidit crear un segon fitxer per a la utilització de la llibreria gsl_rng.h.

Cal mencionar que el programa *gauss.c* ha sigut modificat, de manera que fa ús exclusiu de la funció **Simpson** per calcular $P(|X| > |x|) \leq p$.

2 Presentació de les funcions

2.1 Fitxer aleatori.c

2.1.1 init_seed

- Entrada: Llavor, **double** seed.
- Sortida: No retorna cap valor, **void**.
- Funcionament: Inicialitza els mètodes de generació aleatoris de la llibreria stdlib.h mitjançant la funció **srand**, introduïnt-li per argument la llavor que rep el mètode.

2.1.2 Uniforme

- Entrada:
  - Valor inferior de l'interval: **double** a.
  - Valor superior de l'interval: **double** b.
- Sortida:
  - Nombre aleatori dins l'interval $[a, b]$: **double**.
- Funcionament: Un cop cridada la funció amb els paràmetres corresponents, s'usa la funció **rand** per generar un nombre aleatori a l'interval $[0, 1]$ i el següent canvi de variable per retornar un nombre aleatori dins l'interval desitjat: $rnd \cdot (b - a) + a$, on rnd és un nombre aleatori de l'interval $[0, 1]$.

2.1.3 Normal

- Entrada:
  - Valor de l'esperança de la normal: **double** mu.
  - Valor de la variància de la normal: **double** sigma.
- Sortida:
  - Nombre aleatori que segueix una distribució $\mathcal{N}(\mu, \sigma^2)$: **double**.
- Funcionament: Un cop cridada la funció amb els paràmetres corresponents, es crida la funció **Uniforme**, amb els paràmetres -1 i 1, per generar una parella de nombres aleatoris u, v uniformes a $[-1, 1]$ tals que $s = u^2 + v^2 < 1$. De manera aleatòria es selecciona **un** dels següents nombres aleatoris que segueixen una $\mathcal{N}(0, 1)$:

$$n_1 = u\sqrt{\frac{-2\log(s)}{s}}$$
$$n_2 = v\sqrt{\frac{-2\log(s)}{s}}$$

Se li aplica al valor seleccionat el següent canvi de variable: $n_i \cdot s \cdot sigma + mu$, de manera que es retorna un valor aleatori que segueix una $\mathcal{N}(\mu, \sigma^2)$.

2.2 Fitxer aleatori_gsl.c

Mencionar que el fitxer *aleatori_gsl.c* conté declarades les variables globals *T* i *rng*.

2.2.1 init_seed

- Entrada: Llavor, **double** seed.
- Sortida: No retorna cap valor, **void**.
- Funcionament: Crea l'estructura *rng* de la llibreria gsl_rng.h encarregada de generar els nombres aleatoris mitjançant la llavor introduïda com a paràmetre.

2.2.2 Uniforme

- Entrada:
  - Valor inferior de l'interval: **double** a.
  - Valor superior de l'interval: **double** b.
- Sortida:
  - Nombre aleatori dins l'interval [a,b]: **double**.
- Funcionament: Usa la funció **gsl_rng_uniform**(*rng*) per generar un nombre aleatori a l'interval $[0, 1]$ i el següent canvi de variable per retornar un nombre aleatori dins l'interval desitjat: $rnd \cdot (b-a)+a$ on rnd és el valor retornat per per la funció anteriorment mencionada.

2.2.3 Normal

- Entrada:
  - Valor de l'esperança de la normal: **double** mu.
  - Valor de la variància de la normal: **double** sigma.
- Sortida:
  - Nombre aleatori que segueix una distribució $\mathcal{N}(\mu, \sigma^2)$: **double**.
- Funcionament: Segueix el procediment explicat anteriorment a l'apartat 2.1.3. Es fan dues crides a la funció **Uniforme** que usa el mètode de la gsl per retornar un nombre aleatori.

3 Control dels errors

3.1 Llibreries: *aleatori.c* i *aleatori_gsl.c*

Per tal d'assegurar el correcte funcionament dels mètodes implementats a les llibreries a inicis de la funció (abans de començar a fer cap càlcul) es confirmarà la correcta introducció dels paràmetres. Es comprovarà:

1. Que el valor inferior de l'interval sigui menor al superior (en la funció Uniforme).
2. Que el valor σ^2 introduït sigui estrictament positiu (en la funció Normal).

En cas d'inclompir alguna d'aquestes condicions el programa actuarà de la següent manera:

- Es mostrarà per pantalla el missatge d'error corresponent, guardat a la variable E_ITER o E_SIGMA (explicades anteriorment a la secció **1**).

Recalcar que en cas d'inclomplir alguna condició que impedexi fer els càlculs (cas que no es compleixin les condicions 1 o 2) es retornarà el valor $LLONG\_MAX$.

3.2 Programa *generador.c*

Per tal d'assegurar el correcte funcionament del programa es confirmarà la correcta introducció dels paràmetres de la següent forma:

1. Es comprova que s'hagin introduït els dos paràmetres necessaris (μ, σ^2). És retornarà el valor -1 en cas que no es compleixi la condició.

2. Es comprova que el valor de sigma sigui positiu. Es retornarà el valor -2 en cas que no es compleixi la condició.

3. Si s'introdueix un valor més, es guarda a la variable n (nombre de números aleatoris a generar), sempre i quan sigui major a 0. Si s'introdueix un valor decimal, es guardarà el valor enter més proper arrodonint cap avall.

4. Si s'introdueix un valor més, es guarda a la variable *seed*.

5. Tota la resta d'arguments entrats seran obviats.

6. En cas de no haver-se produït cap error, es retornarà un 0.

3.3 Programa *comprovador.sh*

Per tal d'assegurar el correcte funcionament del programa a l'inici de la funció (abans de començar a fer cap càlcul) es confirmarà la correcta introducció dels paràmetres de la següent forma:

1. Es comprova que s'hagi introduït el paràmetre necessari p i que aquest pertanyi a l'interval $[0, 1]$.

2. Si s'introdueixen més paràmetres, es guarden a les variables μ, σ^2, n i *seed* respectivament.

3. Si s'afegeix un sisè paràmetre (nombre d'intervals pel mètode de Simpson), es comprova que el valor sigui positiu i diferent de 0, en cas contrari, no es guardarà i quedarà el valor per defecte, 300.

4 Compilació i execució

4.1 Compilació del Programa

4.1.1 Makefile

Per facilitar la compilació dels programes s'ha creat un fitxer $makefile$ que inclou tant les comandes per crear els executables com altres comandes associades (que explicarem més endavant) per la correcta gestió dels fitxers que s'obtenen de l'execució del makefile.

Per poder executar el fitxer $makefile$ cal abans modificar el fitxer d'opcions amb els *path* correctes d'on es troba la carpeta *gsl* que conté totes les seves llibreries.
El fitxer d'opcions s'anomena *options.mk* i s'han de modificar les variables INCLUDEPATH i LIBPATH amb els path correctes (path on es troba la carpeta *gsl* i path on es troba la carpeta *libgsl.so.xx* on *xx* és la versió de la llibreria *gsl* respectivament).

Mencionar que no es poden utilitzar dues comandes "*clean*" de manera seguida ja que donarà error.

Important: en cas de no utilitzar el sistema operatiu *Linux* (o semblants) o *IOS* cal modificar la variable DELETE de l'arxiu *options.mk* per a poder utilitzar-lo (substituir per la comanda *del* en cas d'utilitzar Windows).

Per compilar el programa, generar els executables i gestionar els fitxers creats, hem creat les següents rutines, podem executar les comandes:

Comanda	**Descripció**
$make	Compila el programa i genera els executables *gauss*, *generador* i *comprovador.sh*.
$make gauss	Compila el programa i genera l'executable corresponent (*gauss*).
$make generador	Compila el programa i genera l'executable corresponent (*generador*) mitjançant el fitxer *aleatori.c*.
$make generador_gsl	Compila el programa i genera l'executable corresponent (*generador*) utilitzant el fitxer *aleatori_gsl.c*
$make comprovador.sh	Canvia els permisos del fitxer *comprovador.sh* per a que es pugui executar.
$make clean	Elimina tots el fitxers amb terminació *.o* del directori.
$make clean_gauss	Elimina l'executable *gauss* i els fitxers generats amb terminació *.o*.
$make clean_generador	Elimina tant l'executable *generador* com els fitxers generats amb terminació *.o*.
$make clean_all	Elimina tant tots els executables com els fitxers amb terminació *.o*.

Important: En executar la comanda *make* es compilarà l'executable *generador* utilitzant el fitxer *aleatori.c*.

4.1.2 Compilació manual

Recomanem utilitzar en ordre les comandes que mostrarem al terminal (una vegada ubicat al directori on es troben els fitxers).

- **Compilació del programa** *gauss*

  Recomanem llegir l'apartat de compilació manual de la pràctica 1 (5.1.2).

- **Compilació del programa** *generador*

  – Compilació amb *aleatori.c*

  ```
  gcc -c aleatori.c aleatori.h -lm
  gcc -c generador.c  aleatori.c -lm
  gcc generador.o  aleatori.o -lm -o generador
  ```

  – Compilació amb *aleatori_gsl.c*

  ```
  gcc -c aleatori_gsl.c aleatori.h -I$(INCLUDEPATH) -lm
  gcc -c generador.c  aleatori_gsl.c -I$(INCLUDEPATH) -lm
  gcc generador.o aleatori_gsl.o -L$(LIBPATH) -Wl,-R$(LIBPATH) -lgsl
                                  -lgslcblas -lm -o generador
  ```

- **Modificació dels permisos del shell script** *comprovador.sh*

  ```
  chmod 744 comprovador.sh
  ```

4.2 Execució del Programa

Per a una millor utilització i gestió dels programes, hem fet tots els nostres programes suficientment flexibles com per a que s'adaptin a les necessitats que es poden tenir sense accedir i/o modificar el codi. Per això es poden executar de diverses maneres; mostrem ara una taula amb totes les possibles execucions dels nostres programes.

4.2.1 Execució del programa *generador*

Comanda	**Descripció**
`./generador` μ σ^2	Genera 2 milions de nombres aleatoris que segueixen una $\mathcal{N}(\mu, \sigma^2)$, usant una llavor aleatòria.
`./generador` μ σ^2 `n`	Genera n nombres aleatoris que segueixen una $\mathcal{N}(\mu, \sigma^2)$, usant una llavor aleatòria.
`./generador` μ σ^2 `n seed`	Genera n nombres aleatoris que segueixen una $\mathcal{N}(\mu, \sigma^2)$, usant la llavor seed.

Qualsevol argument de més que s'introdueixi al programa serà omés.

Recalcar que tant si és compilat amb el fitxer *aleatori.c* com amb el fitxer *aleatori_gsl.c* el funcionament i execució del programa generador és el mateix; només varia la forma de generar nombres pseudo-aleatoris.

IMPORTANT: Si a l'hora d'executar el programa *generador* compilat amb el fitxer *aleatori_gsl.c* apareix l'error: `error while loading shared libraries libgsl.so.xx` on xx és la versió del *gsl* o algun error semblant, pot ser degut a un d'aquest dos casos:

1. Les variables INCLUDEPATH i/o LIBPATH no contenen el *path* correcte. Cal revisar el *path* que guarden.

2. L'executable no accedeix correctament a la llibreria *gsl*. Cal executar al terminal la comanda següent: $export PATH, on PATH és una variable que conté el mateix path que LIBPATH i ha sigut declarada al terminal, prèviament a la recompilació i reexecució del programa.

5 Comprovacions de funcionament

5.1 Programa *generador.c*

Recordem que el programa *generador.c* és l'encarregat de generar una mostra de nombres aleatoris mitjançant la llibreria aleatori.h.
Recomanem haver llegit amb anterioritat les seccions 2.1 i 4.2.1 (per a les diverses formes de generació de nombres aleatoris, *aleatori.c* i *aleatori_gsl.c*, i les diverses formes d'executar el programa). Fem énfasis en que cal llegir l'aparat 4.1 en cas de tenir algun dubte sobre com compilar degudament el programa.

Per tal d'assegurar el correte funcionament del programa, adjuntem aquí un seguit d'instruccions per comprovar que actui de la manera desitjada.

Recalcar que s'ha utilitzat el fitxer *aleatori.c* per generar la mostra de nombres aleatoris, les excepcions estan indicades i quan s'ha utilitzat el fitxer *aleatori_gsl.c* s'ha fet mitjançant la versió 2.7 de la llibreria de funcions *gsl*.

- Execució per la comanda `./generador` μ σ^2

Comanda	**Output**
`./generador`	`./generador mu sigma`
`./generador` 0 -1	`The value of sigma squared has to be positive (more than 0)`

`./generador` -1 1	`mu sigma x`
	`-1.0000 1.0000 0.281100657152`
	`-1.0000 1.0000 0.799279379685`
	`-1.0000 1.0000 -1.165641526839`
	`-1.0000 1.0000 -0.609642695197`
	$\vdots$ $\vdots$ $\vdots$

`./generador` 0 1	`mu sigma x`
	`0.0000 1.0000 -0.873712043554`
	`0.0000 1.0000 0.378018969177`
	`0.0000 1.0000 -0.681342010392`
	`0.0000 1.0000 -1.982028443212`
	$\vdots$ $\vdots$ $\vdots$

Fem énfasi que els valors que s'obtenen de les comandes superiors estan generats per una llavor aleatòria pel que si es repeteix la comanda és possible que no s'obtinguin els mateixos valors.

Si es volen comparar els valors que s'obtenen amb alguna de les mostres cal introduir la mateixa llavor (cas que es mostrarà amb posterioritat).

- Execució per la comanda `./generador` μ σ^2 n

Comanda	**Output**

`./generador` 0 1 0	`mu sigma x`
	`0.0000 1.0000 -0.725798830034`
	`0.0000 1.0000 0.945648247014`
	⋮ ⋮ ⋮

`./generador` 0 1 3	`The value of numbers requested has changed to 3`
	`mu sigma x`
	`0.0000 1.0000 -0.886783121527`
	`0.0000 1.0000 0.807136605579`
	`0.0000 1.0000 1.354447153111`

`./generador` 0 1 −3	`mu sigma x`
	`0.0000 1.0000 1.515762223841`
	`0.0000 1.0000 2.500994297887`
	⋮ ⋮ ⋮

`./generador` 0 1 3.1	`The value of numbers requested has changed to 3`
	`mu sigma x`
	`0.0000 1.0000 -0.418507079666`
	`0.0000 1.0000 1.128969639183`
	`0.0000 1.0000 0.045605789271`

Com ja s'ha explicat, en cas d'introduir un valor per al paràmatre $n \in (-\infty, 0]$ el programa ignorarà el valor i assignarà a n el valor 2000000.

En cas de dubtes sobre els controls d'errors del programa *generador.c* cal llegir la secció 3.2

- Execució per la comanda `./generador` μ σ^2 n *seed*

Comanda	**Output**

`./generador` 0 1 5 6	`The value of numbers requested`
	`has changed to 5`
	`The seed for random generation`
	`has been changed by 6.000000`
	`mu sigma x`
	`0.0000 1.0000 -0.418507079666`
	`0.0000 1.0000 1.128969639183`
	`0.0000 1.0000 0.045605789271`
	`0.0000 1.0000 -0.422646654265`
	`0.0000 1.0000 0.067429724966`

`./generador` 0 1 5 6 *others*	`The value of numbers requested`
	`has changed to 5`
	`The seed for random generation`
	`has been changed by 6.000000`
	`mu sigma x`
	`0.0000 1.0000 -0.418507079666`
	`0.0000 1.0000 1.128969639183`
	`0.0000 1.0000 0.045605789271`
	`0.0000 1.0000 -0.422646654265`
	`0.0000 1.0000 0.067429724966`

Podem observar que, com cal esperar, en introduir el mateix valor a la variable *seed* (6, en el cas de la taula mostrada) s'obtenen resultats idèntics.

També es pot veure que en introduir més paràmetres (*others*), el programa els ignora i actua com si no hi fossin.

- Execució amb la llibreria *gsl_rng*

Comanda	**Output**

`./generador 0 1`	`mu sigma x`
	`0.0000 1.0000 -0.710010616221`
	`0.0000 1.0000 -0.492539236371`
	⋮ ⋮ ⋮

`./generador 0 1 3`	`The value of numbers requested has changed to 3`
	`mu sigma x`
	`0.0000 1.0000 0.178299690868`
	`0.0000 1.0000 0.611195213845`
	`0.0000 1.0000 0.739190745288`

`./generador 0 1 3 6`	`The value of numbers requested has changed to 3`
	`The seed for random generation has been changed by 6.000000`
	`mu sigma x`
	`0.0000 1.0000 0.931214928923`
	`0.0000 1.0000 0.413912021687`
	`0.0000 1.0000 0.527323084343`

Com es pot observar, en introduir els valors μ, σ^2 es genera un fitxer (que ocupa 200000 de linies), del qual es mostren les primeres files.

En afegir el valor de la n, es genera un fitxer de n linies.

Destacar que els valors obtinguts han estat generats per una llavor aleatòria pel que en executar aquestes mateixes comandes, s'obtindran resultats diferents.

A la tercera comanda, en canvi, la llavor ha sigut introduïda per paràmetre (6), de manera que si s'executa aquesta mateixa comanda un altre cop, donarà resultats idèntics.

També es veu com (mitjançant la mateixa llavor que anteriorment) es generen nombres aleatoris diferents, ja que el mètode de generació es diferent amb la llibreria *gsl* que amb els mètodes implementats per nosaltres (resultats de la taula de la pàgina 17).

La informació en detall es mostra a les seccions: 2.1, 2.2 i 4.1

5.2 Shell script *comprovador.sh*

Recordem que el programa *comprovador.sh* és l'encarregat de crear un fitxer (que s'anomena automàticament *random_samples.txt*), que guarda a cada linia els valors μ, σ^2 i x, on x és el nombre aleatori que segueix una distribució $\mathcal{N}(\mu, \sigma^2)$, resultant d'una crida al programa *generador.c* explicat anteriorment.
Per a l'execució i introducció de les dades es recomana llegir l'apartat anterior **??**.
Per a la correcta compilació del programa recomanem llegir l'apartat 4.1.

Per tal d'assegurar el correte funcionament del programa, adjuntem aquí un seguit d'instruccions per tal de comprovar que el programa actui de la manera desitjada.

- Execució per la comanda `./comprovador` p

Comanda	**Output**	**Fragment** *random_samples.txt*
`./comprovador.sh`	Argument of the probability p is missing	
`./comprovador.sh 2`	Argument of the probability has to be between 0 and 1	
`./comprovador.sh 0.3`	The number of samples x with probability P(|X| > |x|) ≤ p is 600437	0.0000 1.0000 -0.666410103803 0.0000 1.0000 0.543107721219 ⋮ ⋮ ⋮
`./comprovador.sh -0.3`	Argument of the probability has to be between 0 and 1	

Com es pot veure a les taules, si es produeix algun error en la introducció de les dades es mostra el missatge corresponent, i no es genera el fitxer *random_samples.txt*.
Si, en canvi, els paràmetres s'han introduït correctament, l'ouput mostra quantes de les mostres aleatòries x generades compleixen $P(|X| > |x|) \leq p$ i es mostra un fragment del fitxer generat.

- Execució per la comanda `./comprovador` p μ σ^2 n *seed* *inter*

Comanda	**Output**	**random_samples.txt**
`./comprovador.sh 0.9 0 1 10000 1 1`	⋯ The number of samples x with probability P(|X| > |x|) ≤ p is 9037	0.0000 1.0000 1.1122716⋯ 0.0000 1.0000 -0.584169⋯ 0.0000 1.0000 0.4174655⋯
`./comprovador.sh 0.9 0 1 10000 1 0`	⋯ The number of samples x with probability P(|X| > |x|) ≤ p is 9013	0.0000 1.0000 1.3691849⋯ 0.0000 1.0000 1.3871392⋯ ⋮ ⋮ ⋮

Com es pot veure a la taula mostrada, la diferència entre les comandes és el nombre d'intervals introduït amb els que es calcularà la probabilitat desitjada.

En el primer cas, es farà Simpson amb 1 interval.

En el segon cas, com el nombre introduït és 0, s'usarà el valor per defecte, 300.

La part omesa del output són missatges on s'informa a l'usuari que els valors de n, *seed* i *inter* s'han guardat correctament.

En diferir en el nombre d'intervals, la quantitat de nombres aleatoris que compleixen $P(|X| > |x|) \leq p$ serà diferent, però en tenir els mateixos μ, σ^2, n i *seed* el fitxer *random_samples.txt* generat serà el mateix.

6 Conclusions

Finalitzem l'informe confirmant (a partir de les comprovacions de funcionament dutes a terme a l'apartat 5) que tant el programa *comprovador.sh* com el *generador.c* (en les seves dues versions, amb i sense ús de la llibreria gsl_rng.h) funcionen de manera correcta, proporcionant mostres de nombres aleatoris que segueixen una distribució $\mathcal{N}(\mu, \sigma^2)$. Per tant, podem afirmar també que la funció **Uniforme** funciona correctament, ja que és utilitzada a la funció **Normal** a l'hora de generar els nombres aleatoris.

S'informa també que, encara que no ha estat explicat anteriorment, s'ha comprovat al programa *comprovador.sh* que en la generació del fitxer *random_samples.txt* no es forma cap cicle per valors de n menors o iguals a 2000000. És a dir, els valors aleatoris generats són diferents i no segueixen cap patró.

6.1 Informació adicional

- Si comparem els dos fitxers de aleatori.h: *generador* i *generador_gsl*, observem que l'única diferència notable és en la mostra de números aleatoris que ens dona en usar una mateixa llavor; la resta de funcionament és idèntic, s'executen de la mateixa manera i ambdós donen resultats correctes. Pel que recomenem utilitzar qualsevol de les dues implementacions de la mateixa manera. Llegir la secció 4.1 per saber com implementar un fitxer o l'altre.

- Responem a la pregunta: **Com es relacionen els valors x amb p, $P(|X| > |x|) \leq p$?**
  Com es pot observar a la imatge adjuntada, la distribució normal es centra al valor μ.
  La probabilitat $P(|X| > |x|) \leq p$ es pot escriure també com: $P(|x - \mu| \leq |x|) < p$.
  A partir d'això, es dedueix que com més petita sigui la probabilitat p introduïda, més centrats estaràn els valors x generats que compleixin $P(|X| > |x|) \leq p$, seran més propers a μ. En canvi, com més gran sigui la p introduïda, més distants estaran els valors de x respecte μ que compleixen aquesta condició. Com més gran sigui la p, més valors inclourà.

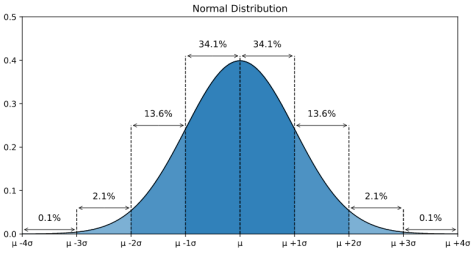

Figura 1: Exemplificació d'una distribució normal.

- La complexitat dels mètodes són d'ordre lineal; és a dir, $O(n)$ (on n és el nombre de mostres que volem que generi el programa).
- Aconsellem llegir el document d'utilització de la llibreria integracio.h i del fitxer *gauss.c* per informació complementaria sobre l'ús i les propietats del programa ja que són utilitzats durant l'execució del shell script *comprovador.sh*.
- Recalcar que al tractar-se el fitxer *comprovador.sh* d'un arxiu *bash*, l'execució del mateix (a l'hora de fer tots els càlculs i crides als subprogrames amb una n prou gran) sol trigar una quantitat de temps significativa.