**Práctico 1 Teoría de la información**

1- Se tienen 5 sobres: 3 temas fáciles, 2 temas complejos

Se extraen 2 sobres, con reposición(independencia)

Espacio muestral S = {FF,FC,CF,CC}

P(FF) = \* = ; P(FC) = P(CF) = \* =

P(CC) = \* =

-------------------------------------------------------------------------------

a) Ea = “Al menos un tema fácil” -> Ea = {FF,FC,CF}

P(Ea) = P(FF U FC U CF) = P(FF) + P(FC) + P(CF) = + +

= = 0,84 (Es muy probable ya que con un tema fácil ya se cumple)

-------------------------------------------------------------------------------

b) Eb = “sólo uno de los temas sea fácil” -> Eb = {FC,CF}

P(Eb) = P(FC U CF) = P(FC) + P(CF) = +

= = 0,48 (Tiende a la mitad, tiene un caso menos que Ea)

-------------------------------------------------------------------------------

c) Ec = “El segundo tema es fácil, si se sabe que el primero es fácil”

Como hay independencia, es lo mismo que decir “un tema fácil”

Ec = {F}

P(Ec) = P(F) = = 0,6

/////////////////////////////////////////////////////////////////

2- Ahora no hay independencia, debido a que no hay reposición

P(FF) = \* = ; P(FC) = \* = ;

P(CF) = \* = ; P(CC) = \* =

------------------------------------------------------------------------------

1. P(Ea) = P(FF U FC U CF) = P(FF) + P(FC) + P(CF) = + + = = 0,9

-----------------------------------------------------------------------------------------------

1. P(Eb) = P( FC U CF) = P(FC) + P(CF) = + = = 0,6

-----------------------------------------------------------------------------------------------

1. Ahora si hay dependencia, entonces P(Ec) = P(F/F) = P(FF) / P(F)

: = = 0,5

Aclaración para los 3 incisos: como ahora no hay reposición, las probabilidades son más altas.

///////////////////////////////////////////////////////////////

3- 10 billetes, 2 falsos, 8 reales ; Se extraen dos (sin reposición)

S= {FF,FR,RF,RR}

P(FF) = \* = ; P(FR) = \* =

P(RF) = \* = ; P(RR) = \* =

-----------------------------------------------------------------------------------------------

a) Ea = “Que ambos billetes sean falsos” -> Ea = {FF}

P(FF) = = 0,0(2)periódico (Es muy baja porque de 10 billetes hay 2 falsos, y justo sacar los 2 es muy poco probable).

----------------------------------------------------------------------------------------------

b) Eb = “Que ninguno de los billetes sean falsos” -> Eb = {RR}

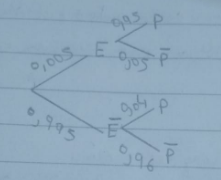
P(RR) = = 0,6(2)periódico (Bastante probable porque la mayoría son reales).

/////////////////////////////////////////////////////////////////

4- E = Persona enferma ; P = Resultado positivo

Datos: P(E) = 0,005 ; P(P/E)=0,95 ; P(~P/~E) = 0,96

Esto puede modelarse con un árbol:



De ahí se obtienen tres nuevos datos:

P(~E) = 0,995 ; P(~P/E) = 0,05 ; P(P/~E) = 0,04

------------------------------------------------------------------------------------------

1. P(~E/~P) ?? , es una probabilidad a posteriori, que se calcula con Bayes:

P(~E/~P) =(\*)

Pero antes deben calcularse las marginales P(P) y P(~P), donde salen de sumar las ramas del árbol para cada una:

P(P) = P(E) \* P(P/E) + P(~E) \* P(P/~E)

= 0,005 \* 0,95 + 0,995 \* 0,96 = 0,04455

P(~P) = P(E) \* P(~P/E) + P(~E) \* P(~P/~E)

= 0,005 \* 0,05 + 0,995 \* 0,04 = 0,95545

Efectivamente, P(P) + P(~P) = 1

Volviendo a (\*) : (0,995 \* 0,96) / 0,95545 = 0,999738

(Esto significa que el test es muy bueno, y falla aprox. en 1 de 1000 personas)

------------------------------------------------------------------------------------------------

1. P(E/P) = = (0,005 \* 0,95) / 0,04455 = 0,10662

(Sin embargo, el test no es bueno para detectar a enfermos)

////////////////////////////////////////////////////////////////

5- Se tiene un cubo Rubik, donde éste tiene 26 cubitos

El evento aleatorio es extraer un cubito

Se define una variable aleatoria c = # caras pintadas

En este caso el c puede tomar valores entre 1 y 3

La probabilidad se obtiene con la función distribución P(c):

|  |  |  |  |
| --- | --- | --- | --- |
| c | 1 | 2 | 3 |
| P(c) | 3/13 | 6/13 | 4/13 |

/////////////////////////////////////////////////////////////////

6- A partir del ejercicio 5- :

* Media <C> = = 1 \* 3/13 + 2 \* 6/13 + 3 \* 4/13 ~=2,077

(Se acerca a 2 ya que es el más probable que salga)

* Desvío estándar: σc  = , primero debo calcular σ2c (varianza)

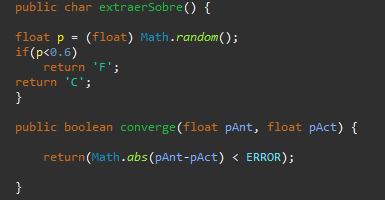
σ2c = <c2> - <C>2 = 4,846 - (2,077)2 = 0,532071

(\*)

(\*) <c2> = 12 \* 3/13 + 22 \* 6/13 + 32 \* 4/13 ~= 4,846

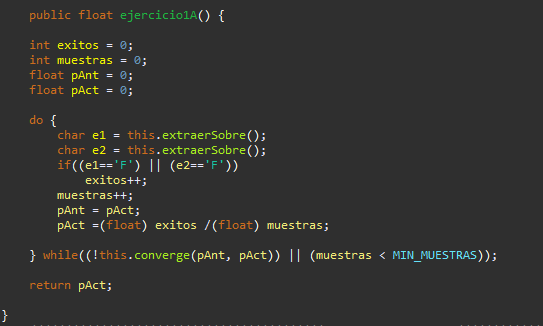
Entonces σc ~=0,73 (Es baja, ya que la mayoría de los datos tienden a la media)

7- 1) En este ejercicio hay dos extracciones con reposición, eso significa que cada extracción tiene el **mismo** panorama (extraerSobre() ). Se usará un mínimo de 100 repeticiones para no caer en convergencia temprana. Si sale ‘F’ el tema será fácil ([⅗ , 1]) . Supongo ε = 0,001



Estas dos funciones son fijas. Ahora depende del inciso el modelado del motor de Montecarlo:

1. Si al menos uno de los temas es fácil, para que haya éxito la primera ó segunda extracción deber ser fácil.



1. Si sólo uno de los temas deber ser ‘F’, se tiene la condición:

“El primero fácil y el segundo complejo **ó** el primero complejo y el segundo fácil”

En el código anterior, se cambia la consulta en la actualización del éxito:



1. Acá se había partido que P(F/F) = F por haber independencia, entonces para el cálculo, no es necesario e1:

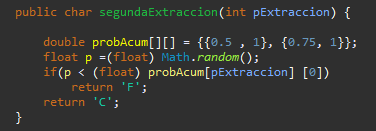


2) Ahora no hay reposición, y cambia el panorama en la 1ra y 2da extracción. La 2da tiene estas probabilidades acumuladas:

* Si salió ‘F’: {½ , 1}
* Si salió ‘C’: {¾ , 1}

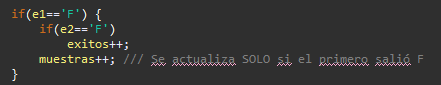
Renombro a la función extraerSobre() como primeraExtraccion()

Defino a segundaExtraccion:



1. Motor de Montecarlo:



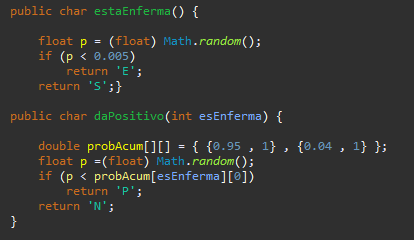
1. Idem 7) 1) b)
2. Debe redefinirse la condición para que actualice el éxito. Al haber la condicionalidad P(F/F) sería: 

4) En este problema hay dos eventos, cada uno con su distribución de probabilidad. El motor de Montecarlo funciona muy similar a los anteriores problemas: deben conocerse las probabilidades condicionales, y luego armar la condición para actualizar el éxito.

Volviendo al contexto del problema:

E = {0.005, 1} -> como hay un valor bajo de p, ahora cambio el MIN\_MUESTRAS a 1000.

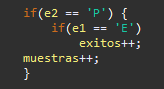
* Si salió ‘E’ , P = {0.95 , 1}
* Si salió ‘S’ , P = {0.04 , 1}



1. P(S/N) ?? Motor de Montecarlo asociado:



1. P(E/P)?? Cambia la condición, en este caso es:



8) Se define un evento aleatorio del cubo Rubik, donde se extraen dos cubitos al azar(con reposición). Se tienen las siguientes variables estocásticas:

A = “Suma de caras coloreadas” ; B = máximo de caras coloreadas

A = {2,3,4,5,6} y B={1,2,3}

1. P(A): se arma una función distribución para ver las probabilidades a los valores de A:

A= 2 -> {(1,1)} => P(2) = \* =

A= 3 -> {(1,2),(2,1)} => P(3) = \* + \* =

A= 4 -> {(1,3),(2,2),(3,1)} => P(4) = \* + \*

+ \* =

A= 5 -> {(3,2),(2,3)} => P(5) = \* + \* =

A= 6 -> {(3,3)} => P(6) = \* =

Entonces:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| A | 2 | 3 | 4 | 5 | 6 |
| P(A) |  |  |  |  |  |

P(B) : misma condición que A

B= 1 -> {(1,1)} => P(1) = \* =

B= 2 -> {(1,2),(2,1),(2,2)} => P(2) = \* + \*

+ \* =

B= 3 -> {(1,3),(3,1),(2,3),(3,2),(3,3)} =>

P(3) = \* + \* + \* + \* + \* =

Quedaría:

|  |  |  |  |
| --- | --- | --- | --- |
| B | 1 | 2 | 3 |
| P(B) |  |  |  |

P(A,B) : para cada valor (Ai , Bj) obtener la intersección y obtener la probabilidad asociada a ese par ordenado. La matriz de probabilidad conjunta queda:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| B/A | 2 | 3 | 4 | 5 | 6 | P(B) |
| 1 | 9/169 | 0 | 0 | 0 | 0 | 9/169 |
| 2 | 0 | 36/169 | 36/169 | 0 | 0 | 72/169 |
| 3 | 0 | 0 | 24/169 | 48/169 | 16/169 | 88/169 |
| P(A) | 9/169 | 36/169 | 60/169 | 48/148 | 16/169 |  |

P(B/A): conociendo la conjunta y la marginal:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| B/A | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1 | 3/5 | 0 | 0 |
| 3 | 0 | 0 | 2/5 | 1 | 1 |

P(A/B):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| B/A | 2 | 3 | 4 | 5 | 6 |
| 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 1/2 | 1/2 | 0 | 0 |
| 3 | 0 | 0 | 3/11 | 6/11 | 2/11 |

1. <A> = 2 \* + 3 \* + 4 \* +5 \* +6 \*

= 4,154 cubitos

<B> = 1 \* + 2 \* + 3 \* = 2,467 cubitos

<A2> = 22 \* + 32 \* + 42 \* +52 \* +62 \*

= 18,32 cubitos2

<B2> = 12 \* + 22 \* + 32 \* =6,444 cubitos2

σA = = 1,032 cubitos

σB = = 0,598 cubitos

<AB> = 2 \* 1 \* + 2 \* 3 \* + 2 \* 4 \* + 3\* 4 \*

+ 3 \* 5 \*+ 3 \* 6 \* = 10,757 cubitos2

Cov(A,B) <AB> - <A><B> = 0,509 cubitos2

Además, voy a calcular el rAB:

rAB = = 0,825 -> Tendencia a lineal ascendente

d) Estimación probabilística de B conocido A:

Si A =2: 1\*1 = 1 ; Si A =3: 2\*1 = 2

Si A = 4: 2 \* ⅗ + 3 \* ⅖ = 2,4 ; Si A =5: 3\*1 = 3

Si A = 6: 3\*1 = 3

e) Comparación con regresión lineal:

Se debe genera una recto del tipo B = c1A+c2 , donde:

C1 = ; C2 = <B> - C1<A>

C1= 0,478 y C2 = 0,481, entonces B = 0,478\*A+0,481

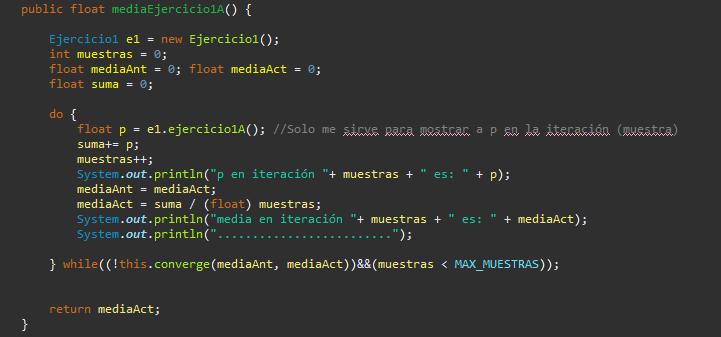
Comparando:

|  |  |  |  |
| --- | --- | --- | --- |
| A | B c/Estimación | B c/Regresión | Diferencia |
| 2 | 1 | 1,437 | 0,437 |
| 3 | 2 | 1,915 | 0,085 |
| 4 | 2,4 | 2,393 | 0,007 |
| 5 | 3 | 2,871 | 0,129 |
| 6 | 3 | 3,349 | 0,349 |

9- La implementación ya se mostró en el ejercicio 7). Se hace uso del lenguaje Java. Para hacer un análisis de cómo funciona el algoritmo entre la parte analítica y muestral, se hará una variación en el ERROR(0.1 , 0.001 , 0.00001 , 0.0000001). Además se realizará un testeo de cómo avanza la mediaMuestral. De cada uno se mostrará la cantidad de iteraciones(que hará las veces del tiempo) que llevó hacerlo. A la media muestral se le fijará un MAX\_MUESTRAS = 10. Es paralelo se usará un MIN\_MUESTRAS = 0 (Para el ejercicio b)) y un MIN\_MUESTRAS =100 (contemplando la convergencia temprana).

Aclaración: se usará el mismo error tanto para analizar la convergencia tanto en p como en la media.

Código para mediaMuestral:



Análisis empírico:

Con error = 0,1:

|  |  |  |  |
| --- | --- | --- | --- |
| iteración  mediaMuestral | p en iteración i | #iteraciones | media en iteración i |
| 1 | 0,87 | 100 | 0,87 |
| 2 | 0,87 | 100 | 0,86 (Corta) |

Iteraciones promedio = 100

Error = 0.001:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0.8138298 | 188 | 0.82138298 |
| 2 | 0.8427673 | 159 | 0.82829857 |
| 3 | 0.79802954 | 203 | 0.8182089 |
| 4 | 0.83 | 168 | 0.82198995 |
| 5 | 0.8427673 | 159 | 0.82614547 |
| 6 | 0.8138298 | 188 | 0.82409286 |
| 7 | 0.82758623 | 174 | 0.82459193 (Corta) |

Iteraciones promedio

Con error = 0,00001

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0.8427681 | 15722 | 0.847681 |
| 2 | 0.83833 | 16163 | 0.84055126 |
| 3 | 0.84349436 | 15648 | 0.8415323 |
| 4 | 0.83535916 | 16454 | 0.839989 |
| 5 | 0.8391634 | 16066 | 0.8398239 |
| 6 | 0.84324735 | 15668 | 0.8403945 |
| 7 | 0.83598655 | 16395 | 0.8397648 |
| 8 | 0.83619475 | 16367 | 0.8393185 |
| 9 | 0.8407847 | 15903 | 0.83931607 |
| 10 | 0.8378278 | 16205 | 0.83931607  (Máximo) |

Iteraciones promedio = 16059,1

Con error = 0,0000001

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 0.8397654 | 1345097 | 0.8397654 |
| 2 | 0.84047496 | 1339978 | 0.8401202 |
| 3 | 0.84349436 | 1339032 | 0.84024173 |
| 4 | 0.83535916 | 1343009 | 0.8402039 |
| 5 | 0.8391634 | 1343452 | 0.8401373 |
| 6 | 0.84324735 | 1341240 | 0.8401651 |
| 7 | 0.83598655 | 1345488 | 0.8401413 |
| 8 | 0.83619475 | 1342968 | 0.8401292 |
| 9 | 0.8407847 | 1345262 | 0.84007573 |
| 10 | 0.8378278 | 1344234 | 0.8400485  (Máximo) |

Iteraciones promedio = 1342976

Datos y consideraciones:

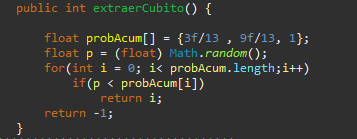
* Con un error bajo, se realiza pocas actualizaciones a la media. Eso es porque hay una baja tolerancia a la mejora. A mayor error alcanza las MAX\_MUESTRAS.
* El error controla que diferencia habrá entre cada una de las probabilidades que arroje la iteración. Una oscilación alta predice una tolerancia baja, y una baja muestra la alta tolerancia a la hora de la precisión.
* Misma explicación que el primer punto, viendo ahora las iteraciones para obtener cada una de las probabilidades.
* Con la media se ve más representativa la situación. Se puede obtener los siguientes valores.
  + Error = 0.1 -> |0.84 - 0.86| = 0.02
  + Error = 0.001 -> |0.84 - 0.82459195| = 0.01540805
  + Error = 0.00001 -> |0.84 - 0.83931607| = 0.00068393
  + Error = 0.1 -> |0.84 - 0.8400485| = 0.0000485
* Estos datos se obtuvieron ejecutando sólo una vez el programa

b) Cuando no hay control de la convergencia temprana, genera que la probabilidad sea poco representativa con la realidad, haciendo que el error casi sea independiente de la situación. Acá pierde totalmente sentido que el error sea inversamente proporcional con el tiempo de ejecución.

Aclaración: este análisis empírico es innecesario mostrarlo para los otros casos, ya que tiene mismo comportamiento. Vale aclarar que dicho análisis se abstrajo de la situación del problema 1) a)

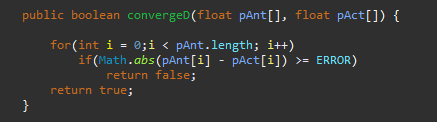
10) Antes de calcular las probabilidades cabe destacar que el evento aleatorio es extraer dos cubitos con reposición, o mejor dicho, hacer dos llamadas a una función extraerCubito(). Su distribución era:

{3/13 , 6/13, 4/13} -> Acumulada : {3/13 , 9/13, 1}



1. P(A)

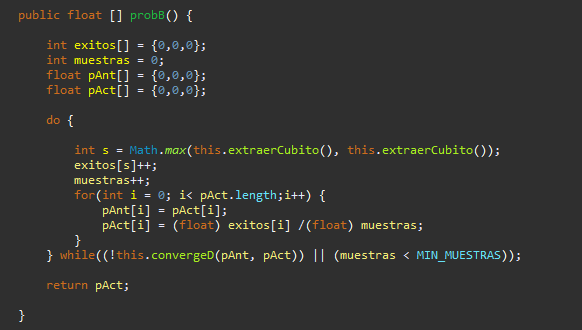
Cabe considerar que el esquema de convergencia y el motor de Montecarlo cambiará ya que ahora se busca una función asociada a una v.a



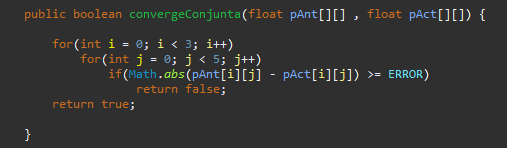
El motor de Montecarlo:

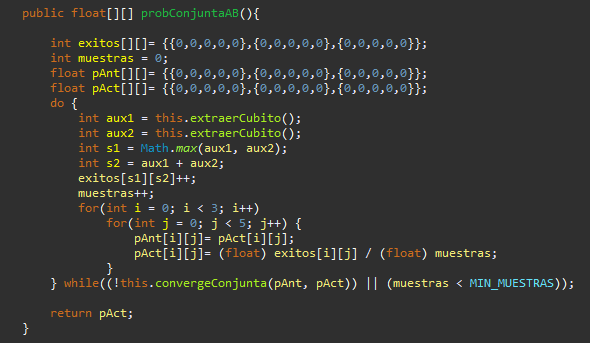


P(B) : esquema similar:



P(A,B): cuando hay probabilidades en las que intervienen 2 v.a , hay que trabajar con matrices. Nuevamente cambiará el converge y el motor:

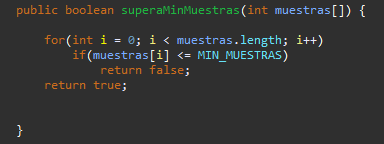




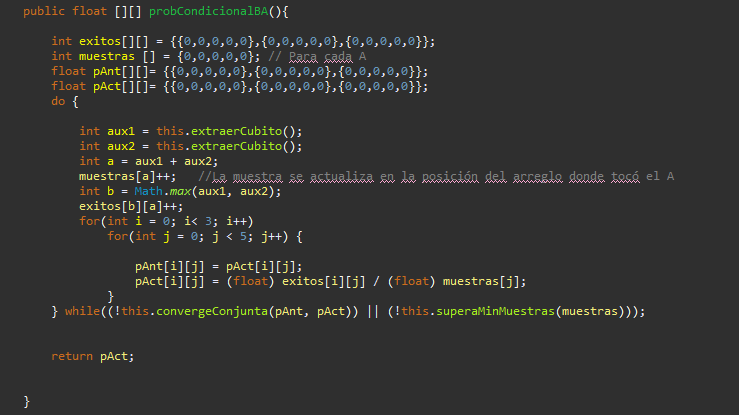
*Motor de Montecarlo asociado*

P(B/A): aquí varía un poco la situación. Si bien se continúa trabajando con matrices, al haber condicionales sólo interesa actualizar las muestras de la variable aleatoria dependiente ( para este caso A). Por eso se definirá un arreglo muestras [|A|].

Convención: se dirá que se superó el mínimo de muestras si todos los elementos de A lo hicieron. Código asociado:



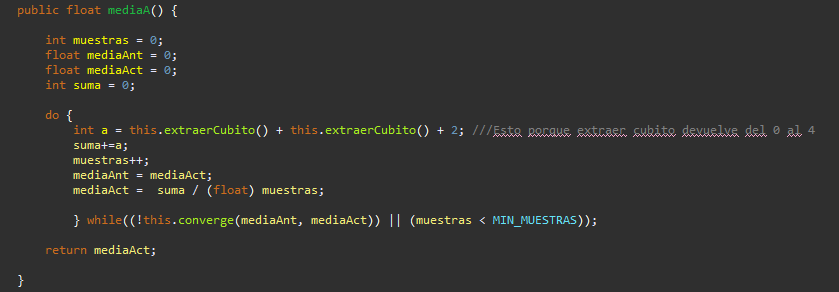
Montecarlo asociado:



P(A/B): mismo planteo, con pocas diferencias:

* Muestras se define con tres valores
* primero se inicializa el b
* En la actualización de pAct se divide por muestras[i]

1. <A>: acá el código será sencillo. La convergencia comparará dos valores(mediaAnt, mediaAct). Montecarlo extraerá dos cubitos y en una v.a se guardará la operación correspondiente ( en A es la suma) y en una variable se tendrá una suma acumulada de todas las v.a que salgan en cada iteración y se dividirá por el valor de la iteración(muestras).



<B>:se plantea igual que mediaA()

σA : el esquema funciona similar a la media, pero se deben guardar más datos antes de la actualización. Los pasos para obtener el desvío muestral:

* obtengo A ( variable aleatoria)
* acumulo los A de cada iteración
* obtengo el Error Medio Cuadrático haciendo el cuadrado de la diferencia entre A y el cociente entre la sumaAcum y las muestras.
* Acumulo el EMC ( sumaEMC)
* Actualizo el σANT y σACT como la raíz del cociente entre sumaEMC y las muestras.



σB: idem σA

11) Mostrado como código en lenguaje JAVA en el 10). En un pseudocódigo se mostraría menos detalles de implementación como:

* Casteos
* Llamados a librerías
* Inicializaciones instanteáneas de estructuras
* Índices más adaptados al problema analítico.
* pseudo-Java (no mostrar los public, referencias this, etc.)