Trabajo Final
Investigación Operativa

Indicadores - Simulación Montercarlo y las Vegas

Integrantes del Grupo:

Canalejo, Leonardo Ceferino
leocanalejo@gmail.com
Leguizamon, Francisco Nicolas
fnleguizamon@gmail.com
Lopez, José Martín
martinlopez89.quequen@gmail.com
Auzmendi, Tobias Adriel
toba.neco@gmail.com

Tutor del trabajo:

Gustavo Illescas

1. Resumen

En el presente trabajo se estudiará una serie de Indicadores mediante la simulación de montecarlo, la cual será implementada para trabajar con la base de datos que se le brindó a los estudiantes.

Dicha base de datos pertenece a los datos de alumnos recibidos correspondiente a la carrera de ingeniería de sistemas, la cual tiene dos tipos de indicadores, la cantidad de alumnos recibidos por fecha y el tiempo de carrera que le llevó a los alumnos.

El objetivo es analizar cómo se comporta el simulador teniendo la posibilidad de simular datos nuevos para poder estimar valores futuros y también simular fechas ya existentes en donde se podrá comparar el dato del simulador con el dato real.

2. Palabras Clave

- Montecarlo: (Simulador) Es un método no determinista o estadístico numérico, usado para aproximar expresiones matemáticas complejas y costosas de evaluar con exactitud.
- Las vegas: es un algoritmos de computación de carácter aleatorio que no es apropiado: da el resultado correcto o informa que ha fallado.

3. Abstract and keywords

- Plugin: es una noción que no forma parte del diccionario de la Real Academia Española (RAE). Se trata de un concepto de la lengua inglesa que puede entenderse como “inserción”y que se emplea en el campo de la informática.
- Java: Lenguaje de programación.

4. Introducción

En el presente trabajo se estudiará una serie de indicadores mediante la simulación de montecarlo, la cual será implementada para trabajar con la base de datos que se le brindó a los estudiantes.

Para este trabajo se desarrollará un programa el cual podrá manipular una bases de datos. Dicha base de datos corresponde a los alumnos recibidos en la carrera de ingeniería de sistemas y tiene dos tipos de indicadores, la cantidad de alumnos recibidos por fecha y el tiempo que le llevó recibirse a cada alumno, estos datos datan desde el 9 de marzo de 1995 hasta 23 de agosto de 2013. En el programa se podrá trabajar con fechas correspondientes entre las mencionadas anteriormente y una vez que se eligen el programa detalla los datos.

El objetivo del trabajo es implementar el algoritmo de montecarlo y el algoritmos de las vegas para poder simular diferentes intervalos entre las fechas seleccionadas y poder comparar el resultado de las fechas reales con las obtenidas con el simulador, siempre en cada simulación se trabajará con todos los datos anteriores a la fecha que se pretende simular.

Para la implementación se utilizaron apuntes correspondientes a la materia teoría de la información correspondientes a la unidad de Resolución de forma analitica y por muestreo computacional. La documentación leída es la siguiente:

http://www.exa.unicen.edu.ar/catedras/teoinfo/files/2014/teoria01_2014.pdf

http://www.exa.unicen.edu.ar/catedras/teoinfo/files/2014/teoria02_2014.pdf

El hecho de que se halla utilizado estos apuntes es que los alumnos integrantes del grupo cursaron dicha materia.

Para realizar la modificaciones y adiciones al programa brindado se usó la documentación oficial de jiglu que es el plugin que se utilizó para realizar interfaz gráfica en lenguaje java.

http://www.jiglu.com/

5. Desarrollo

5.1 Simulador Montecarlo

El método Montecarlo consiste en crear un modelo matemático del sistema que se desea analizar, haciendo uso de la estadística, proporcionando posibles escenarios del problema.

5.1.1 Detalles de funcionamiento.

Una vez abierta la aplicación y la Base de datos, se debe seleccionar el indicador que se quiere estudiar. En pantalla aparecerá la fecha mínima y máxima de los datos almacenados, pudiendo modificar éstas para simular un rango a elección.

Al ejecutar el algoritmo, se obtiene la lista de todas las fechas con sus respectivos valores en el rango indicado, y se comienza a simular los diferentes intervalos tomando un mínimo inicial de 5 fechas, para devolver el posible escenario de la fecha siguiente. Así en cada iteración, se vuelven a tomar todos los datos desde el mínimo hasta la última fecha que se simuló.

Una vez definido cada unos de los intervalos, se calcula la distribución de probabilidades de los valores intervinientes, para facilitar la creación del vector acumulado a ser utilizado por el motor. El paso por seguir es realizar los cálculos necesarios: Cada uno de los ensayos en la simulación involucra generar valores al azar para las entradas probabilísticas (Valor de tiempo en años de carrera, y cantidad de egresados de Ing. De Sistemas) y después calcular la utilidad. La corrida de simulación queda concluida cuando el valor de la media converge con un error de 0.02.

La aplicación mostrará: los datos originales que se tomaron, una tabla con los resultados de la simulación y los histogramas correspondientes (fig. Vista aplicación). Además cuenta con la opción de guardar dichos resultados en la base de datos, para luego poder ser consultada.

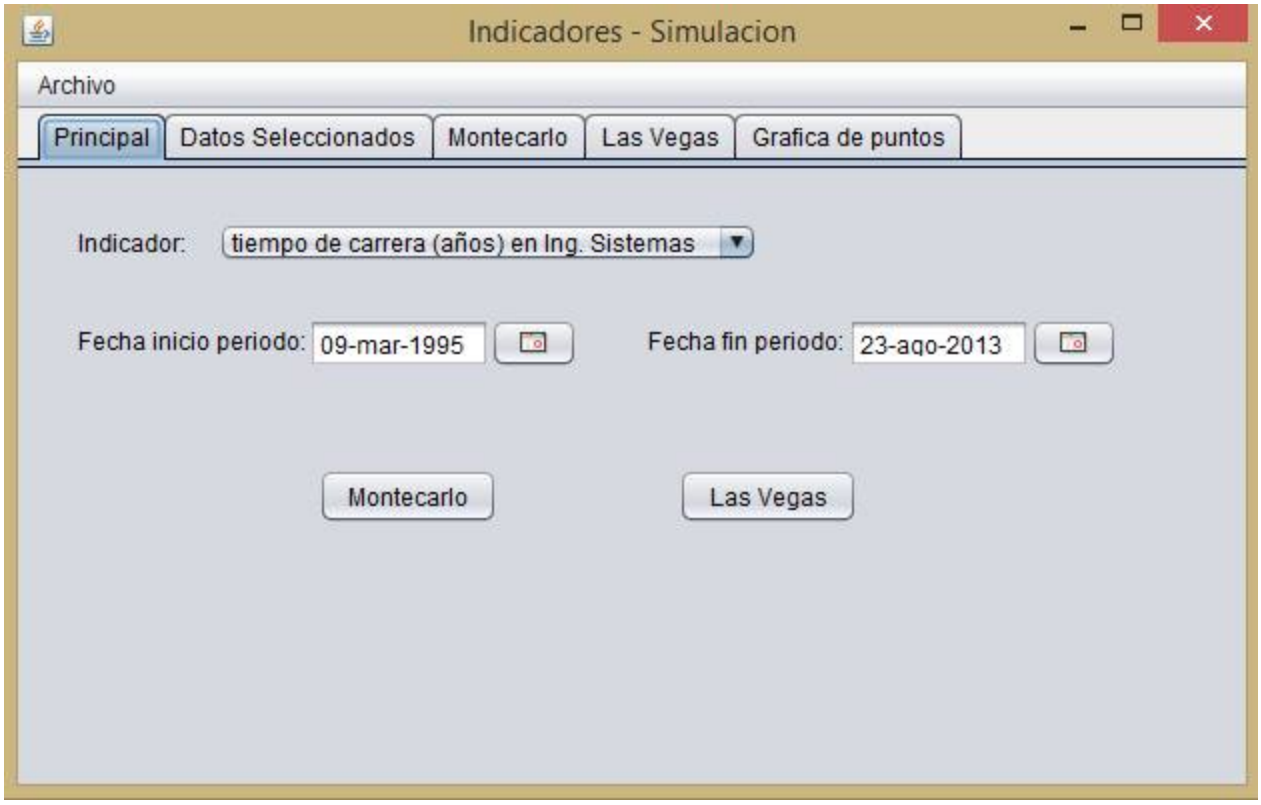


fig. Vista aplicación

5.1.2 Implementación

En esta sección se explicará brevemente algunos aspectos importantes de la implementación y en algunos casos detallaremos un pseudocódigo de cada método pero no se detalla ningun codigo, para ello se encuentran todas las fuentes del proyecto disponibles para su observación.

get_vec_acum: este metodo esta en la clase Montecarlo.java modifica la lista generada por getRowSim, de manera que cada en vez de dejar la repetición por cada fecha, la modifica obteniendo la probabilidad que salga cada fecha y además acumula estas probabilidades ya que así se la necesita para poder utilizar el simulador.

sacarDato: este método está en la clase Montecarlo.java y la cantidad correspondiente a la probabilidad que genera la una función random y así generar aleatoriamente la cantidad a profesar por el simulador.

```
float sacarDato ( listaAcumulada ) {
        random = random();
        for ( i = 0 to listaAcumulada[N] ) {
               if ( random < listaAcumulada[N])
                      return listaAcumulada[i].valor;
        }
 }
```

converge: este método está en la clase Montecarlo.java, y se encarga de terminar la simulación esto se produce cuando el error generado entre la media actual y la anterior es menor al establecido, así de esta forma se puede asegurar que se llegó a un resultado con la probabilidad de error que se ha establecido, y de esta forma se decida que no se deberá seguir simulando.

```
boolean converge (media , mediaAnt) {
        if (|media - mediaAnt| < Ɛ ) //Siendo Ɛ una constante pequeña
               return true ;
        else
               return false ;
 }
```

simulacion: este metodo esta en la clase Montecarlo.java y se encarga de simular utilizando los métodos anteriores, a continuación se detalla el pseudocódigo y se explicará brevemente cómo funciona.

```
float simulator (datosFiltrados) {
        float media = 0 ;
        float mediaAnt = 1 ;
        int tiradas = 0 ;
        list listaAcumulada = gen_vec_acum (datosFiltrados);
        float acum = 0 ;
        while (!converge(media,mediaAnt) || tiradas < 100) {
               tiradas++;
               float dato = sacarDato (listaAcumulada);
               acum = acum + dato ;
               mediaAnt = media ;
               media = acum / tiradas ;
        }
        return media ;
 }
```

El código inicia declarando las variables media, mediaAnt, tiradas, acum, y listaAcumulada. La variable media almacena la media que se va calculando, mediaAnt almacena la media anterior para poder ser comparada en el método converge, acum va

acumulando los datos que devuelve sacarDato, tiradas va contando el número de simulaciones y listaAcumulada es la lista que genera el método gen_vec_acum.

Posteriormente empieza la simulación, primero se estableció un mínimo de 100 tiradas así el algoritmo no converge rápidamente en caso de salir en las primeras dos simulaciones el mismo valor. Una vez que pase las 100 tiradas dependerá del método converge el cual se explico anteriormente. En cada paso se aumentará el número de tiradas, se sacara el dato nuevo que se acumula en acum, y con esto dos valores se calculará la media, antes de modificar el valor media se almacena en mediaAnt para poder compararlos en el método converge.

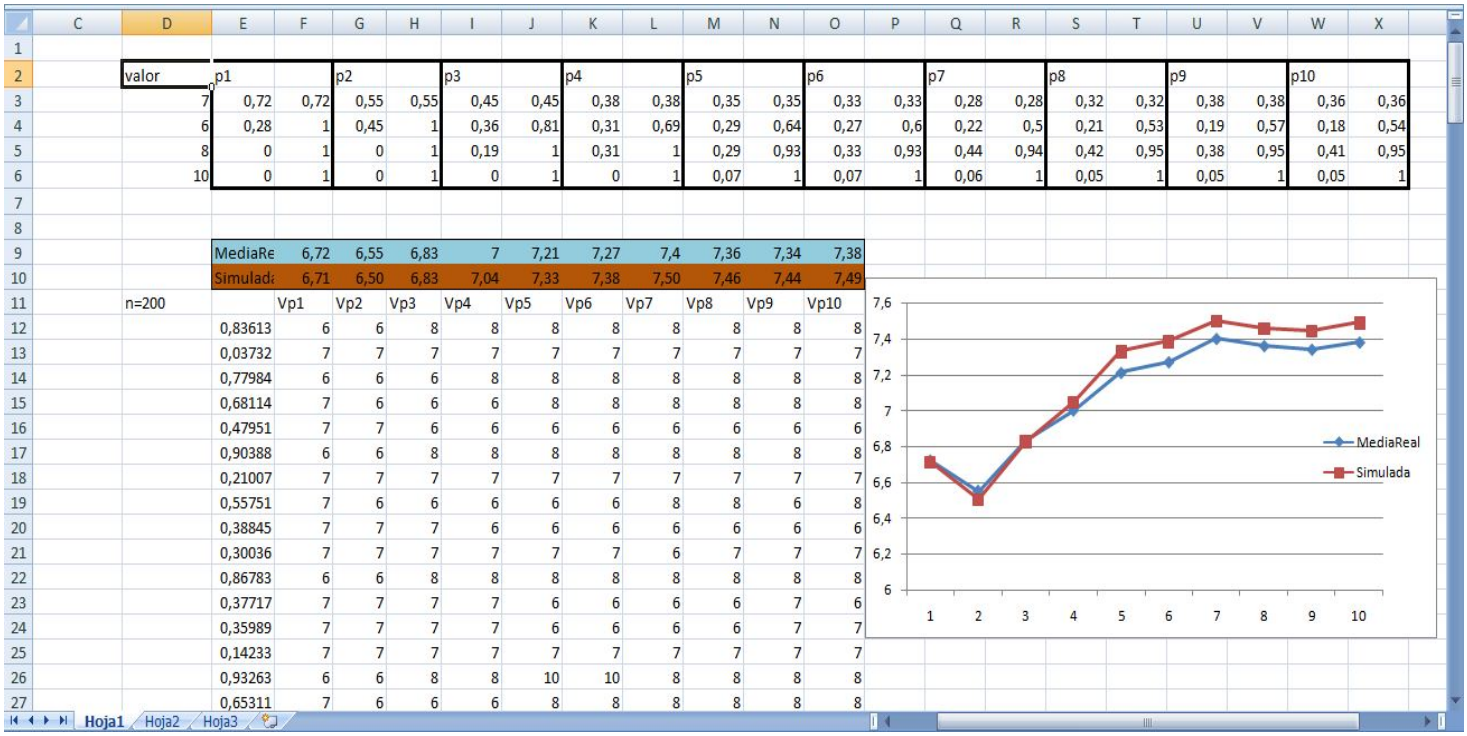
Una vez que el algoritmo converge se devuelve el valor media que es el resultado final de la simulación.

5.2 Comparación del simulador con otro sistema

5.2.1 Resumen

Para Comprar y poder verificar el buen funcionamiento del simulador, se ha utilizado Microsoft Office Excel. Se tomó un intervalo de fechas (fig, Resultados de la simulación en la aplicación), el cual se simuló en la aplicación y en software propiamente dicho, como se puede ver en la imagen (fig, resultado de la simulación en Excel) los resultados de las simulaciones se asimilan a los del simulador.También se ha calculado la media real para dichos valores en cada uno de los intervalos tomados (recurriendo a la teoría de probabilidades) para un análisis más certero en la comparación.

5.2.2 Comparación



valor	p1		p2		p3		p4		p5		p6		p7		p8		p9		p10	
7	0,72	0,72	0,55	0,55	0,45	0,45	0,38	0,38	0,35	0,35	0,33	0,33	0,28	0,28	0,32	0,32	0,38	0,38	0,36	0,36
6	0,28	1	0,45	1	0,36	0,81	0,31	0,69	0,29	0,64	0,27	0,6	0,22	0,5	0,21	0,53	0,19	0,57	0,18	0,54
8	0	1	0	1	0,19	1	0,31	1	0,29	0,93	0,33	0,93	0,44	0,94	0,42	0,95	0,38	0,95	0,41	0,95
10	0	1	0	1	0	1	0	1	0,07	1	0,07	1	0,06	1	0,05	1	0,05	1	0,05	1

MediaRe	6,72	6,55	6,83	7	7,21	7,27	7,4	7,36	7,34	7,38
Simulada	6,71	6,50	6,83	7,04	7,33	7,38	7,50	7,46	7,44	7,49

n=200	Vp1	Vp2	Vp3	Vp4	Vp5	Vp6	Vp7	Vp8	Vp9	Vp10
0,83613	6	6	8	8	8	8	8	8	8	8
0,03732	7	7	7	7	7	7	7	7	7	7
0,77984	6	6	6	8	8	8	8	8	8	8
0,68114	7	6	6	6	8	8	8	8	8	8
0,47951	7	7	6	6	6	6	6	6	6	6
0,90388	6	6	8	8	8	8	8	8	8	8
0,21007	7	7	7	7	7	7	7	7	7	7
0,55751	7	6	6	6	6	6	8	8	6	8
0,38845	7	7	7	6	6	6	6	6	6	6
0,30036	7	7	7	7	7	7	6	7	7	7
0,86783	6	6	8	8	8	8	8	8	8	8
0,37717	7	7	7	7	6	6	6	6	7	6
0,35989	7	7	7	7	6	6	6	6	7	7
0,14233	7	7	7	7	7	7	7	7	7	7
0,93263	6	6	8	8	10	10	8	8	8	8
0,65311	7	6	6	6	8	8	8	8	8	8

fig, resultado de la simulación en Excel

Para el experimento se simuló entre las fechas 09-03-1995 y 26-08-1996, con una muestra aleatoria de n= 200 (cantidad de repeticiones) se observó los resultados de los MC que se asemeja a la Media real y lo indicado por el simulador.

Id Indicador	Periodo	Valor	Fecha Simulada
1	09-03-1995 / 18-08-1995	6.67	13-10-1995
1	09-03-1995 / 13-10-1995	6.57	27-10-1995
1	09-03-1995 / 27-10-1995	6.87	07-12-1995
1	09-03-1995 / 07-12-1995	7.01	22-12-1995
1	09-03-1995 / 22-12-1995	7.15	13-04-1996
1	09-03-1995 / 13-04-1996	7.13	19-04-1996
1	09-03-1995 / 19-04-1996	7.26	24-05-1996
1	09-03-1995 / 24-05-1996	7.34	21-06-1996
1	09-03-1995 / 21-06-1996	7.4	02-08-1996
1	09-03-1995 / 02-08-1996	7.42	30-08-1996

fig, Resultados de la simulación en la aplicación

5.3 Simulación Las Vegas

5.3.1 Resumen

Un algoritmo tipo Las Vegas es un algoritmo de computación de carácter aleatorio que no es aproximado: es decir, da el resultado correcto o informa que ha fallado.

Un algoritmo de este tipo no especula con el resultado sino que especula con los recursos a utilizar en la computación del mismo.

De la misma manera que el método Montecarlo, la probabilidad de encontrar una solución correcta aumenta con el tiempo empleado en obtenerla y el número de muestreos utilizado. Un algoritmo tipo Las Vegas se utiliza sobre todo en problemas NP-Completos, que serían intratables con métodos determinísticos.

Existe un riesgo de no encontrar solución debido a que se hacen elecciones de rutas aleatorias que pueden no llevar a ningún sitio. El objetivo es minimizar la probabilidad de no encontrar la solución, tomando decisiones aleatorias con inteligencia, pero minimizando también el tiempo de ejecución al aplicarse sobre el espacio de información aleatoria.

La decisión que se tomó ante esta problemática fue que la solución correcta en este caso iba a ser retornar una media igual a la media de la lista "*datosFiltrados*". El algoritmo, para obtenerla, genera una lista de datos aleatoria del tamaño de la lista de entrada y verifica que tenga la misma media. No obstante, para quitarle trivialidad a los resultados que arroje la aplicación se le introdujo un error con el cual el valor de retorno va a cambiar en cada ejecución pero nunca se va a alejar de un cierto valor a la media de la lista entrante. Para optimización del algoritmo se le agregó una poda (si va por la mitad del llenado de la lista y la media es menor que el cuartil Q1 de la lista que entró como parámetro o si es mayor que el Q3 va a ser un mal caso para seguir generando datos por lo tanto vacía la lista y empieza a generar datos aleatorios de vuelta).

5.3.2 Implementación

Se detalla a continuación una breve descripción de los algoritmos principales que se implementaron para el desarrollo de la resolución. Cabe destacar que lo que posteriormente será llamado como “*datosFiltrados*” es una lista de *SimEntry* la cual por cada elemento posee su valor y su cantidad de ocurrencias.

obtenerDatoLV: Este algoritmo permite obtener un valor aleatorio de entre los valores que componen a la lista *datosFiltrados.*

agregarDatoALista: Este método permite agregar un nuevo valor a una lista de *SimEntry*. En caso de que el valor ya se encuentre en ella se incrementará en 1 su cantidad.

cargarCuartiles: Este algoritmo genera los cuartiles Q1 y Q3 que son utilizados para la poda. Estos son calculados en base a *datosFiltrados.*

converge: Este método determina si el error entre los dos valores de entrada es menor a una constante. Se utiliza para comprobar si los datos de una simulación se aproximan a los datos de entrada comparando la media de ambos.

obtenerMedia: Determina la media muestral de una lista de *SimEntry.*

LasVegas: Es el algoritmo principal de la clase LasVegas.java y utiliza los métodos anteriormente descritos para efectuar la simulación. En el siguiente pseudocódigo queda detallado una idea conceptual del mismo:

```
public float lasVegas (List datosFiltrados){
              ordenar(datosFiltrados);
              int longitud=datosFiltrados.size();
              if(longitud>4){//Se debe simular con una lista cuyo tamaño sea mayor a 4
                                  //debido a las limitaciones de la funcion
cargarCuartiles
                     int mitad=longitud/2;
                     float mediaStandar=obtenerMedia(datosFiltrados);
                     int cuartil1=cargarCuartiles(datosFiltrados).getC1();
                     int cuartil3=cargarCuartiles(datosFiltrados).getC3();
                     boolean poda;
                     float media=0;
                     while ( !converge(media,mediaStandar)) {
                            List aux;
                            poda=false;
                            for(int i=0;i<longitud and !poda;i++){
                                   if((i=mitad) and
                            ( (obtenerMedia(aux)<cuartil1) or (obtenerMedia(aux)>cuartil3)
                            ))
                                          poda=true;
                                   else{
                                          float dato=obtenerDatoLV(datosFiltrados);
                                          agregarDatoAlista(aux,dato);
                                   }}
                            if(!poda)
                                   media=obtenerMedia(aux);
                                   //el else es innecesario porque ya va a tener
//cargada la media anterior que va a volver a fallar y arranca así un nuevo intento
                     }
                     return media;
              } else
                     System.out.println("La lista tiene menos de 5 elementos");

              return 0;}
```

5.3.3 Comparación

Para concluir se encontró que este algoritmo presenta grandes ventajas basándose entre otras cosas en la forma particular en la que se manipulan los datos de entrada. Nos da como herramienta la posibilidad de cambiar el camino de la búsqueda hacia la solución en caso de que el escenario de simulación sea desfavorable, y es de apreciar que en búsquedas muy costosas esta característica es realmente favorable en términos de eficiencia.

En comparación, el algoritmo de Montecarlo ocasionalmente comete un error, pero encuentra la solución correcta con una probabilidad alta. Y es posible generalmente reducir la probabilidad de error a costa de aumentar el tiempo de cálculo. Pero tiene la gran desventaja de que en caso de que se cometa un error no da aviso. En caso de necesitar exactitud en el resultado, sin margen al error en los datos que arroje una simulación, es cuando podemos utilizar Las Vegas que siempre encuentra la solución correcta y en caso de no hacerlo nos informa al respecto.

6. Conclusiones

6.1 Interpretación de los resultados de la simulación

Para la interpretación se simularon valores de la cantidad de años de duración de la carrera Ing. de Sistemas entre las fechas 09-03-1995 y 26-08-1996, teniendo en cuenta el histórico.

Se observa la realización de un histograma para facilitar la comprensión de los resultados, como muestra la figura (fig. histograma Simulación) se realizaron dos funciones, una corresponde a la media computacional, es decir, a la simulación por montecarlo y la otra a la media real.

Se puede concluir que la simulación por montecarlo ofrece mucha más información, en un escenario más real del problema y más en el programa realizado para este trabajo ya que la simulación se realiza hasta que el resultado converge, en cambio en la comparación con excel se realizaron 200 iteraciones.

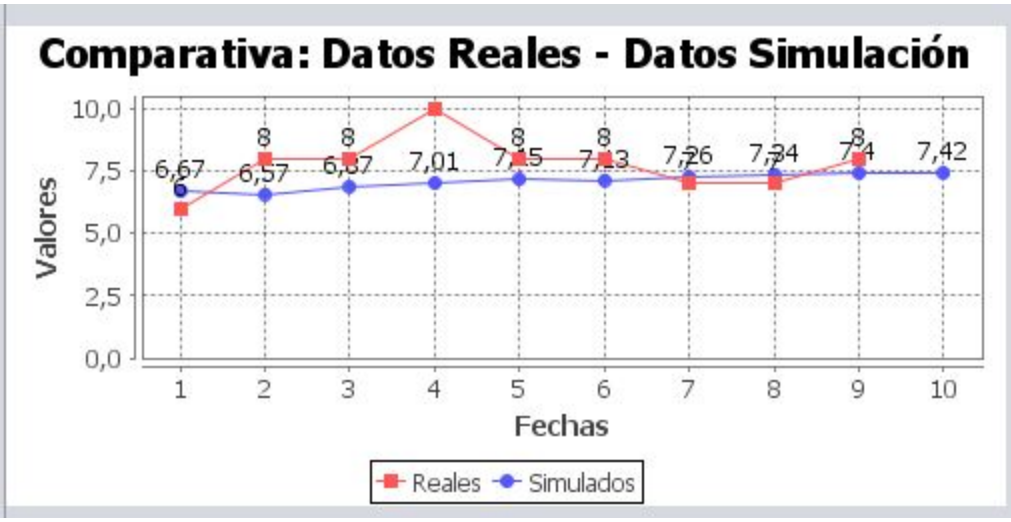


fig. histograma Simulación

En la figura anterior se observa un ejemplo de cómo el programa muestra los histogramas de los datos filtrados que el usuario elige,en la línea roja se ven los datos reales mientras en la simulada, se obtienen los datos simulados teniendo en cuenta todos los datos anteriores, el gráfico empieza a mostrar los resultados a partir de la fecha 5 es decir que la fecha uno en el histograma corresponde a la quinta fecha, y el valor del histograma correspondiente a los datos simulados obtiene ese resultado dependiendo de todas fechas anteriores lo cual eso se ve reflejado en el resultado final.

Lo explicado anteriormente es muy importante para considerar simulaciones con datos más antiguos, los cuales serían más erróneos que si se toman datos más actuales.

7. Referencias

[1]TEORÍA DE LA INFORMACIÓN (2014): “Repaso de probabilidad y estadística”. Facultad de ciencias exactas-UNCPBA
http://www.exa.unicen.edu.ar/catedras/teoinfo/files/2014/teoria01_2014.pdf
[2] TEORÍA DE INFORMACiÓN (2014): SIN TÍTULO.
Facultad de ciencias exactas-UNCPBA
http://www.exa.unicen.edu.ar/catedras/teoinfo/files/2014/teoria02_2014.pdf
[3]TAHA H. (2004): Investigación de operaciones 7a. edición. Pearson educación.
University of Arkansas, Fayetteville