Trabajo Práctico Especial

Taller de Matemática Computacional – TUDAI
Cursada 2017

Facultad de Ciencias Exactas, UNCPBA

Hora de ensuciarse con Octave!

Operadores y Variables

```
n = 25;
n = 25.5;
```

- No hay declaraciones de variables.
- Los tipos de las variables cambian dinámicamente!

```
s = 'Hola mundo!'
s = "Hola mundo!"
s = 'Hola mundo!';
```

- Comilla simple o doble sirven como delimitadores de string.
- Si no se quiere imprimir en pantalla algo, poner punto y coma.

Operadores y Variables

- **Operaciones básicas**

```
6 * 5
#ans = 30
```

  - Los resultados se guardan en una variable **ans** si no se especifica la variable en la que guardar el resultado.

- **Constantes**

```
number = 25 + pi
```

  - Ya existen precargadas varias constantes (pi, e, etc…)

Vectores (o arreglos)

- **Vectores**

```
vector = [1 2 3];
```

  - Los elementos se separan con comas o espacios.

IMPORTANTE: en Octave, los vectores no son únicamente estructuras de datos... **son también un concepto matemático!!**

MÁS IMPORTANTE TODAVÍA!!
En Octave los **vectores** se indexan a partir de 1!!!!
(No del 0, como en Java/C/C++)

Algunas pruebas con índices

- **¿Qué pasa en cada línea de este código?**

```
A = [5, 3, 4, 1, 2]
A(2)
A(end)
A(2:4)
A(end-2:end)
```

```
A = [5, 3, 4, 1, 2]
A(2)                    % Devuelve un 3
A(end)                  % Devuelve un 2 (el último valor)
A(2:4)                  % Devuelve el arreglo [3,4,1]
A(end-2:end)            % Devuelve el arreglo [4,1,2]
```

Funciones predefinidas

- **Muchísimas! Completar qué hace cada función:**

```
A = [ 1, 2, 4, -5 ]
length(A)           % Longitud del arreglo
size(A)             % Tamaño del arreglo (1 x 4)
sum(A(:))           % Suma de todos los elementos
min(A(:))           % Mínimo de todos los elementos
max(A(:))           % Máximo elemento del arreglo
abs(-3)             % Devuelve el valor absoluto (3)
...
```

IMPORTANTE
Las funciones predefinidas de Octave son mucho más eficientes de lo que podríamos implementarlas nosotros.

Así que usemos siempre que podamos las funciones predefinidas!!

Medir tiempos en Octave

- **En Octave es muy simple medir el tiempo que tarda en ejecutarse un conjunto de sentencias**

```
tic       % Empieza a cronometrar el tiempo
...
...       % Pongo todo aquello a lo que le quiero medir el tiempo
...
...
tiempo = toc; % Corta el cronómetro y guarda el tiempo total en tiempo
```

Estadística en Octave

- **Existen funciones nativas para el cálculo de estadísticas**

```
X = [ 1 1 1 2 1 1 ]

[ valor, pos ] = min(X(:)) % Devuelve el valor mínimo
[ valor, pos ] = max(X(:)) % Devuelve el valor máximo

Y = mean(X(:))             % Devuelve el valor promedio
Y = median(X(:))           % Devuelve la mediana
Y = mode(X(:))             % Devuelve la moda

Y = std(X(:))              % Devuelve el desvío estándar
Y = var(X(:))              % Devuelve la varianza
```

Gráficos en Octave

- **Gráfico de frecuencias**

```
figure, plot(X);
ylim([0 1]);
grid on;
legend('Probabilidades parciales');
xlabel('Iteraciones');
ylabel('Probabilidad');
```

Gráficos en Octave

- **Gráfico de frecuencias**

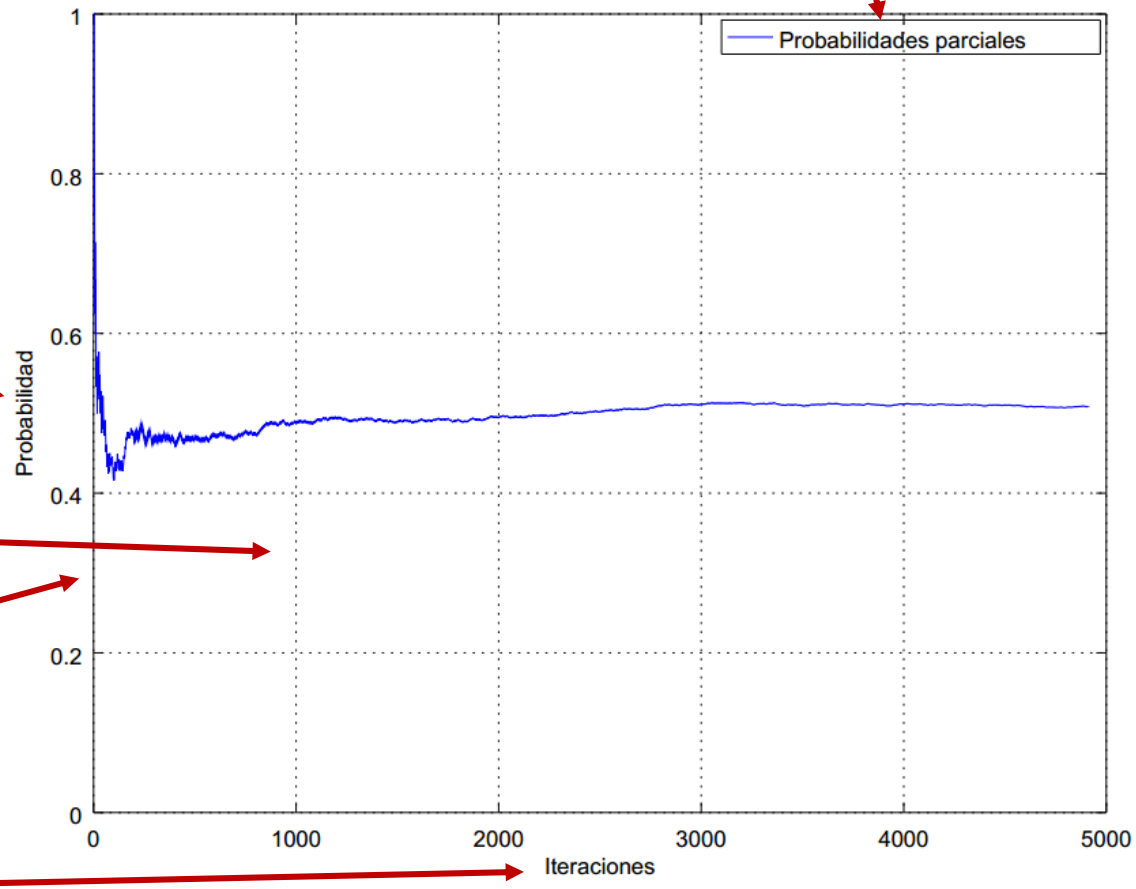
```
legend('Probabilidades parciales')
```

```
ylabel('Probabilidad')
```

```
grid on
```

```
ylim([0 1])
```

```
xlabel('Iteraciones')
```



Trabajo Práctico!

Enunciado

- **La nave espacial con la que estamos trabajando utiliza un sistema de autentificación bastante obsoleto, que autoriza a realizar disparos láser de acuerdo al DNI introducido.**
- El mismo es simulado con Octave mediante llamadas a la función `my_mex_service(dni)`, que recibe como parámetro un número entero dni que representa el número de documento del alumno, y devuelve un 1 en caso de autorizar el disparo y un 0 en caso de no autorizarlo.

Enunciado

`dni`

`puedo_tirar`

Un 1 si puedo
Un 0 si no puedo

Ejercicios entregables: Programación

Entregar un repositorio en Github con la implementación de todas las funciones y scripts necesarios para resolver los siguientes ejercicios:

Programación. Ejercicio 1

1. Implementar una función que, dado tu DNI y un cierto valor de epsilon, calcule la probabilidad de que ocurran dos fallos sucesivos de la función `my_mex_service`. Utilizar para ello el método de Montecarlo, y devolver tanto la probabilidad estimada como un arreglo de las probabilidades parciales obtenidas en cada iteración.

Programación. Ejercicio 1

`dni`

`epsilon`

`probabilidad_estimada`

`probabilidades_parciales`

Programación. Ejercicio 2

2. Modificar el script `script_trabajo_especial` para que repita llamados a la función programada en el inciso anterior, utilizando los siguientes valores de epsilon:

   - epsilon = 0.1
   - epsilon = 0.01
   - epsilon = 0.001

Programación. Ejercicio 2

script_trabajo_especial

epsilon = 0.1;
(llamado a la función)

dni, epsilon

probabilidad_estimada,
probabilidades_parciales

Programación. Ejercicio 2

script_trabajo_especial

epsilon = 0.1;
(llamado a la función)

dni, epsilon

probabilidad_estimada,
probabilidades_parciales

epsilon = 0.01;
(llamado a la función)

dni, epsilon

probabilidad_estimada,
probabilidades_parciales

...

Programación. Ejercicio 3

3. Modificar el script `script_trabajo_especial` para que imprima por pantalla los distintos valores de probabilidad obtenidos para cada valor de epsilon, y para que muestre las gráficas con las probabilidades parciales asociadas a cada uno de estos experimentos.

Programación. Ejercicio 3

script_trabajo_especial

epsilon = 0.1;
(llamado a la función)

dni, epsilon

probabilidad_estimada, probabilidades_parciales

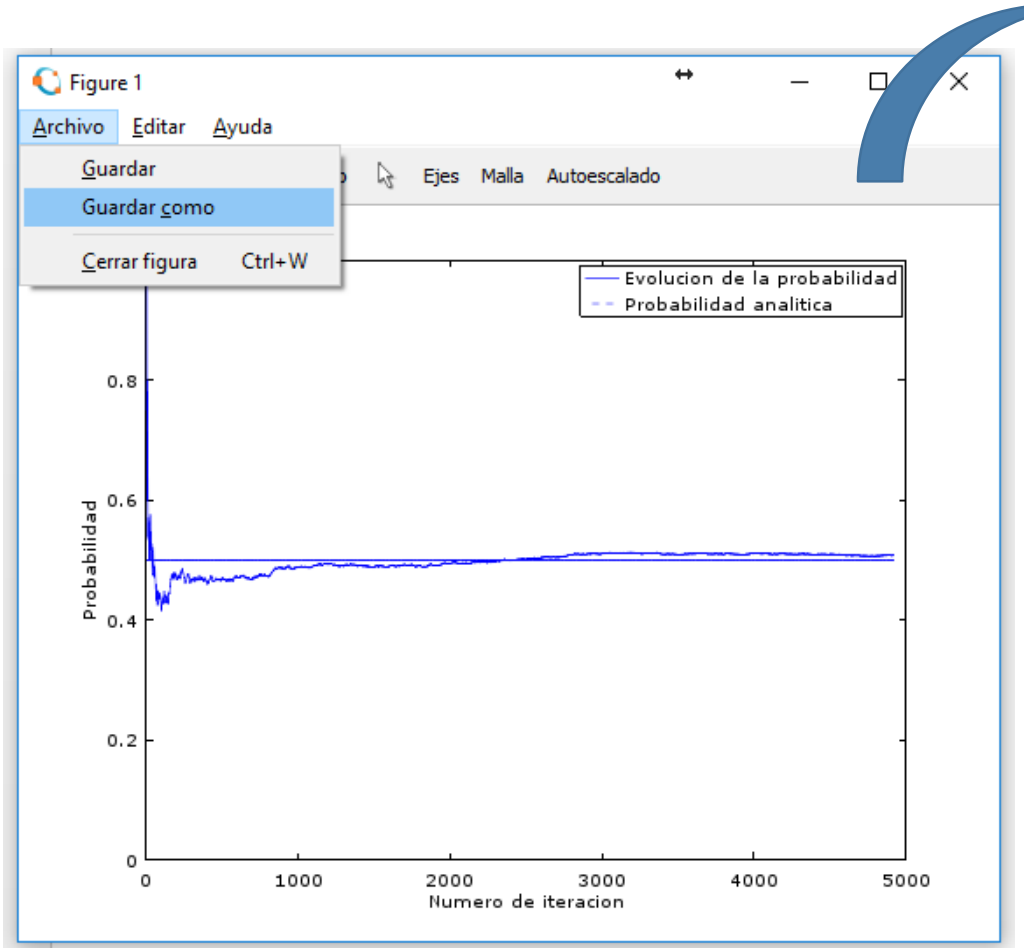
fprintf('Probabilidad estimada: %f', <<variable con la probabilidad>>);

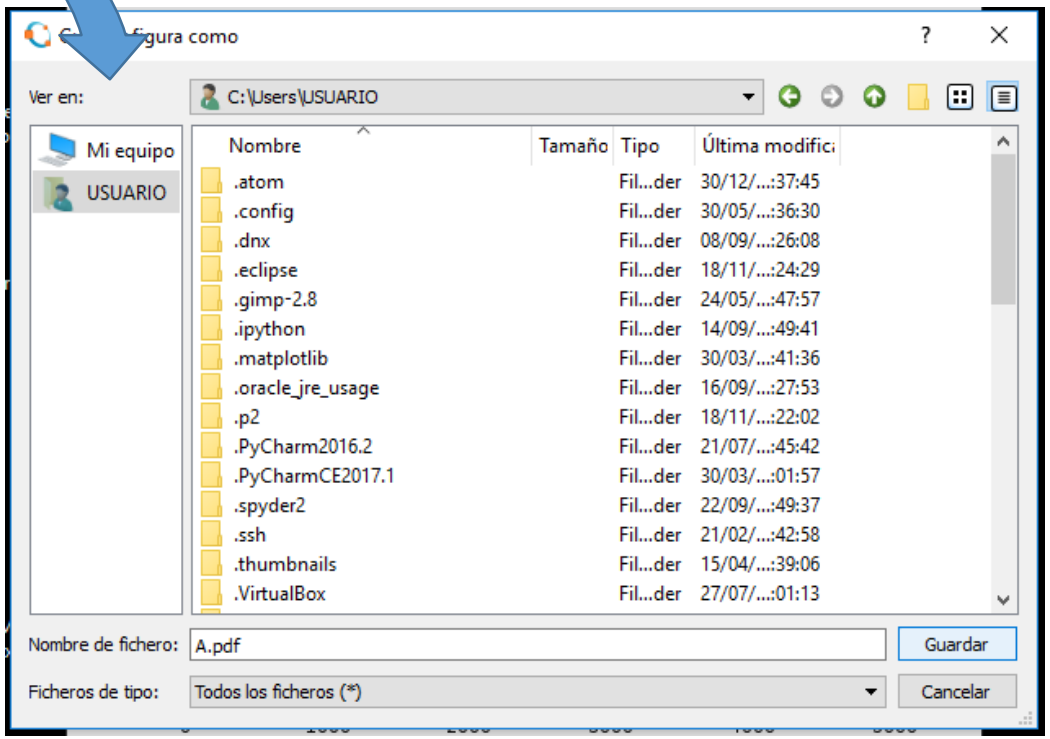
Con esta sentencia se puede imprimir por pantalla un número decimal por pantalla

figure, plot(<<variable con las probabilidades parciales>>);

Con esta sentencia se imprime por pantalla la figura de las probabilidades

Programación. Ejercicio 3



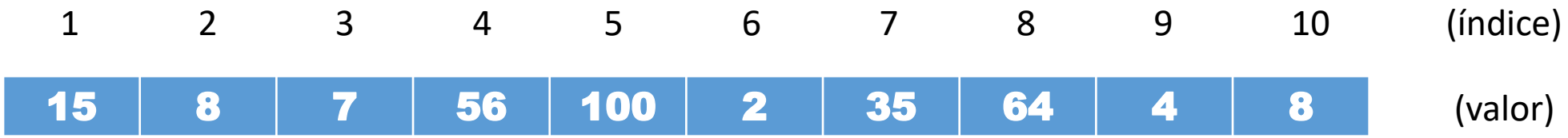


Programación. Ejercicio 4

4. Modificar el script `script_trabajo_especial` para que calcule e imprima por pantalla los desvíos estándar de las probabilidades parciales en las primeras 20 y las últimas 20 iteraciones de cada experimento anterior.

Programación. Ejercicio 4

- Si yo tengo un arreglo A…



- … puedo recuperar valores en una posición dada:

```
A(1) es 15          A(9) es 4          A(end) es 8
```

- … puedo recuperar valores en una posición dada:
  - `A(1:3)` devuelve un arreglo con los primeros 3 elementos.
  - `A(4:6)` devuelve un arreglo con los elementos desde el 4to al 6to lugar
  - `A(end-5:end)` devuelve los últimos 6 elementos (del 5 al 10)

Programación. Ejercicio 5

5. Incorporar al script las sentencias necesarias para medir el tiempo que tarda cada corrida del algoritmo.

¿Cómo apruebo?

Condiciones de aprobación

- **El trabajo (código + informe) deberá ser entregado a través del repositorio personal de Github, antes de las 0:00 hs. del martes 13 de junio de 2017**. Se evaluará únicamente la versión existente en el repositorio antes de la fecha límite. Versiones posteriores no serán tenidas en cuenta al momento de asignar la nota.
- Para aprobar el trabajo, el mismo debe cumplir todos los siguientes requisitos:
  - Haber sido entregado en tiempo y forma (es decir, antes de las 0:00 hs del martes 13 de junio de 2017) en el repositorio personal que fue declarado al momento de inscribirse al trabajo.
  - El código debe poder ejecutarse sin inconvenientes. Si el mismo presenta errores de sintaxis o no realiza lo solicitado, será considerado como desaprobado.
  - El informe debe respetar los límites de páginas establecidos y la organización prevista.
- Se considerarán únicamente como A- aquellos trabajos que, funcionando correctamente y con informes correctamente estructurados, hagan un análisis parcialmente erróneo de los resultados obtenidos.

Flujo de Git

Flujo de Git: Por única vez

1. **`git clone https://github.com/usuario/tmc-tp-especial.git`**
2. `cd tmc-tp-especial`
3. Pegar en la carpeta la versión de la funci
   `my_mex_service` acorde a tu sistema o
   descargada de la página de la materia.

Esto clona el repositorio en el que van a entregar el trabajo especial.

IMPORTANTE
Cambiar "usuario" por el usuario de Github que se hicieron en el laboratorio.

Flujo de Git: Por única vez

1. `git clone https://github.com/usuario/tmc-tp-especial.git`
2. **`cd tmc-tp-especial`**
3. Pegar en la carpeta la versión de la funci… `my_mex_service` acorde a tu sistema operativo, descargada de la página de la materia.

Con esto se mueven hasta la carpeta donde se descargó el repositorio.

Flujo de Git: Por única vez

1. `git clone https://github.com/usuario/tmc-tp-especial.git`
2. `cd tmc-tp-especial`
3. **Pegar en la carpeta la versión de la función `my_mex_service` acorde a tu sistema operativo, descargada de la página de la materia.**

IMPORTANTE HACER ESTO BIEN o no va a andar!!!

Flujo de Git: Cuando terminaste

1. `git add -A`
2. `git commit -m 'Mensaje'`
3. `git push origin master`

Esto registra los cambios que hayas realizado en el repositorio, pero en el repositorio local.

IMPORTANTE

Haciendo esto no queda actualizado el repositorio remoto, se actualiza **SOLAMENTE EL LOCAL!**

Cada vez que hagas un cambio, necesitás hacer esto.

Flujo de Git: Cuando terminaste

1. `git add -A`
2. `git commit -m 'Mensaje'`
3. **`git push origin master`**

Haciendo esto se actualiza el repositorio remoto a la versión del último commit.

IMPORTANTE
Haciendo esto **SÍ SE ACTUALIZA EL REPOSITORIO REMOTO.**

IMPORTANTÍSIMO

Podés repetir los pasos 1 y 2 todas las veces que quieras.

Solamente se va a tener en cuenta el último push anterior a la fecha de entrega al momento de corregir.

Los push posteriores a la fecha de entrega no se van a tener en cuenta en la evaluación del trabajo.

Si hacés el trabajo en el laboratorio, siempre que llegues tenés que hacer el clone, y antes de irte hacé un commit, un push y borrá la carpeta.

¿Preguntas?

Taller de Matemática Computacional – TUDAI
Cursada 2017

Facultad de Ciencias Exactas, UNCPBA