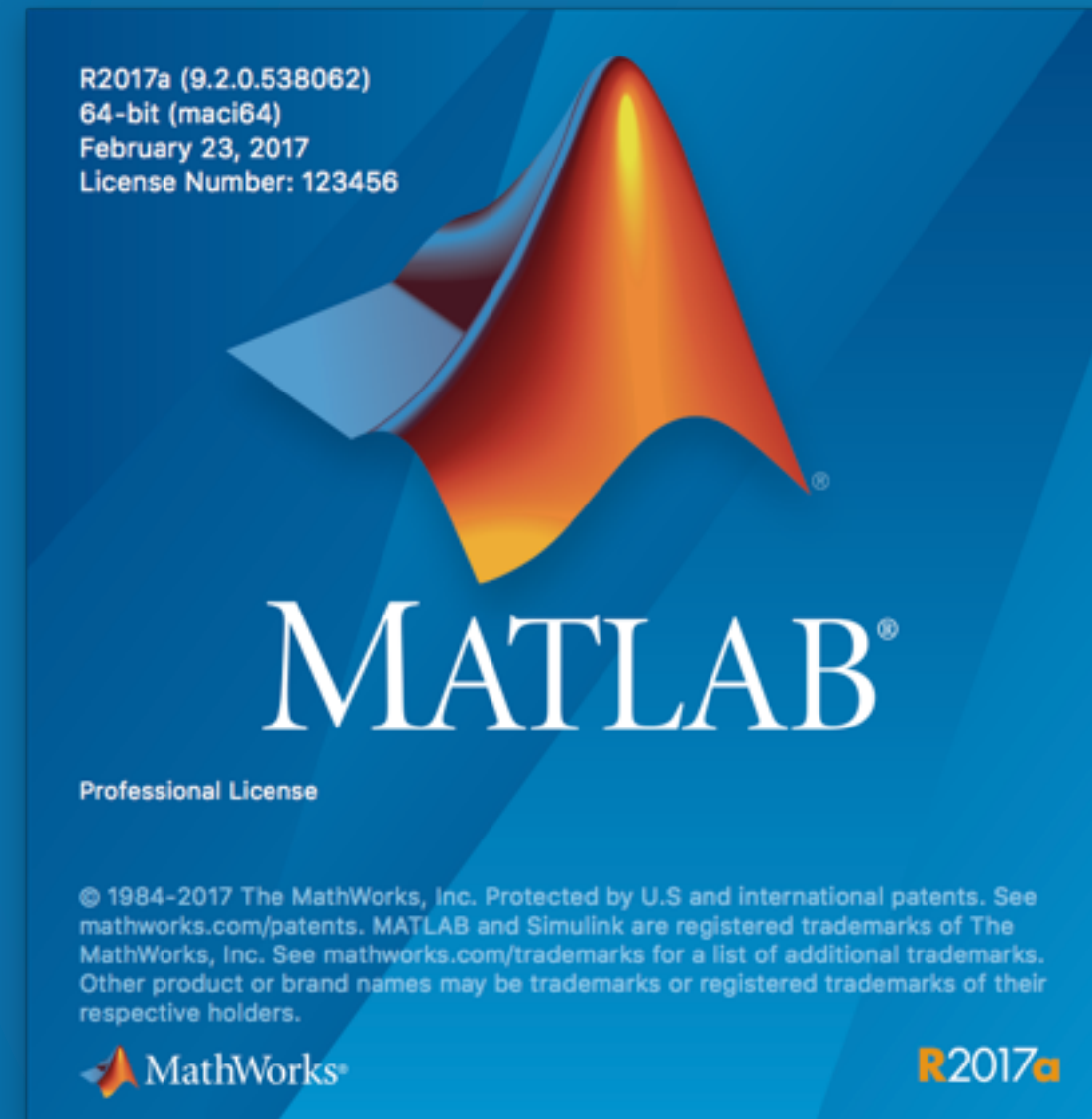


MATLAB® & SIMULINK®

- Curso de introducción 2025 -

Presentación por: **David López García**

Entorno de trabajo en Matlab



QUÉ ES MATLAB

ENTORNO DE TRABAJO

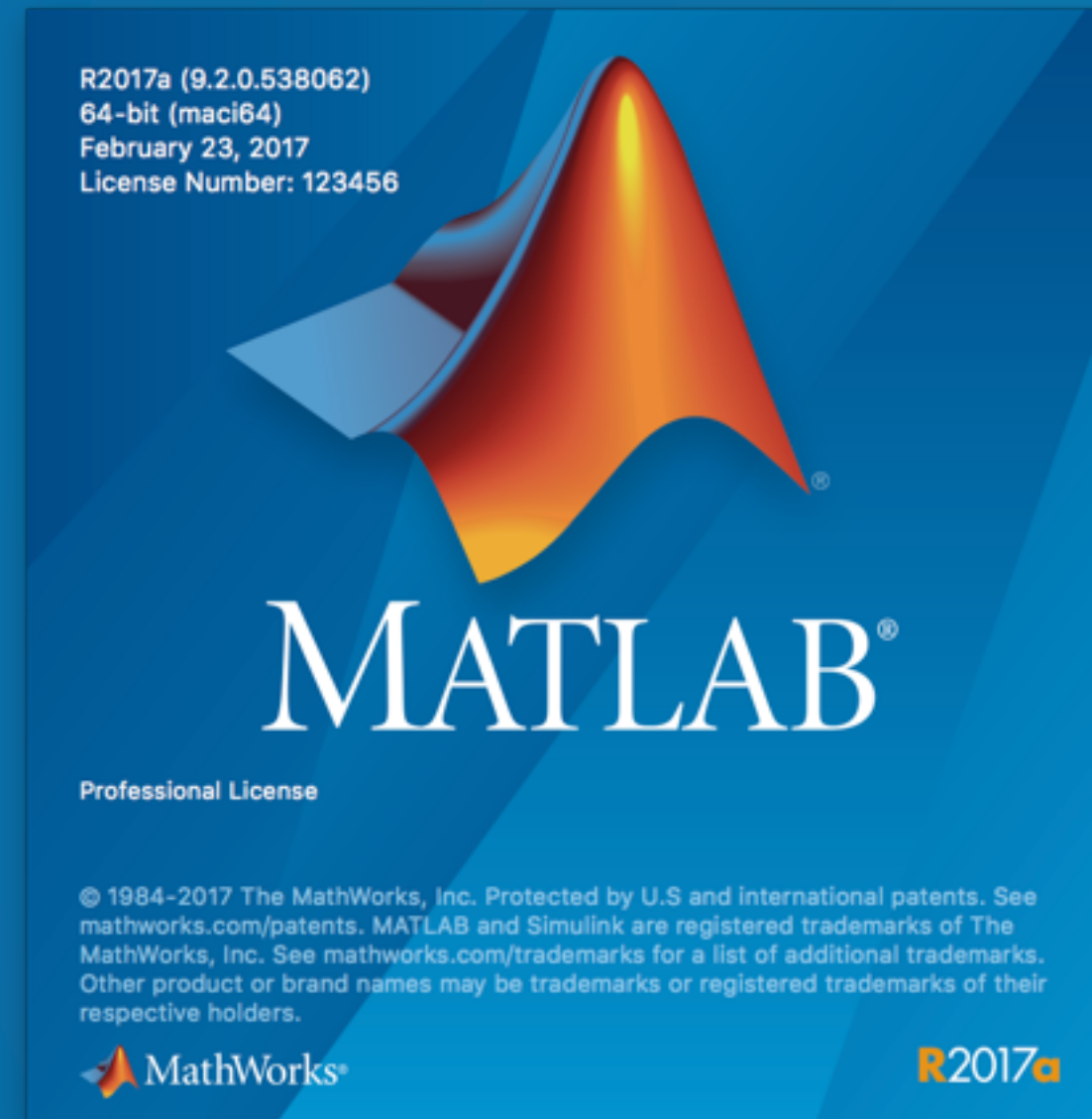
- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB

Entorno de trabajo en Matlab



QUÉ ES MATLAB

ENTORNO DE TRABAJO

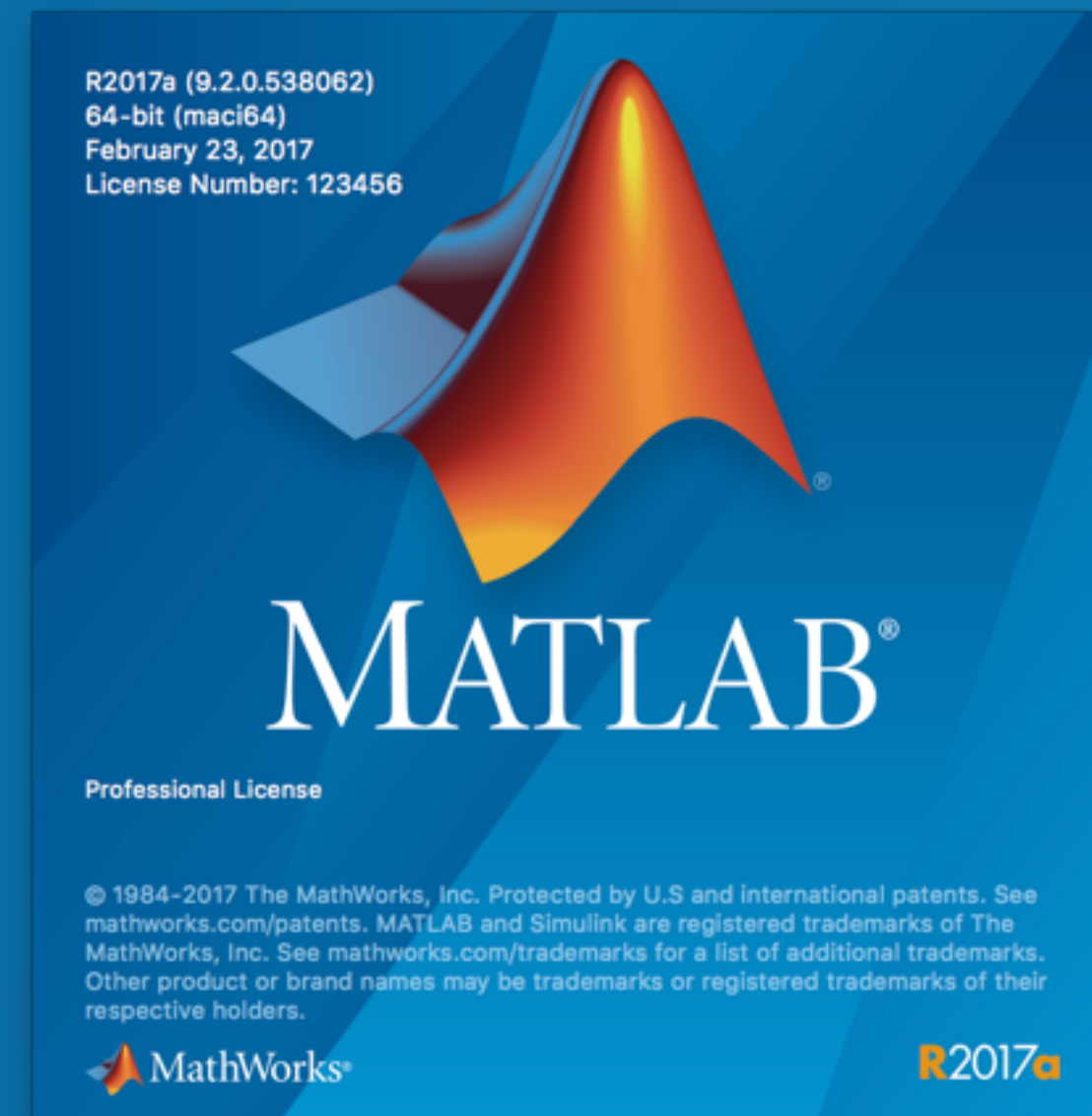
- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB

Entorno de trabajo en Matlab



QUÉ ES MATLAB

ENTORNO DE TRABAJO

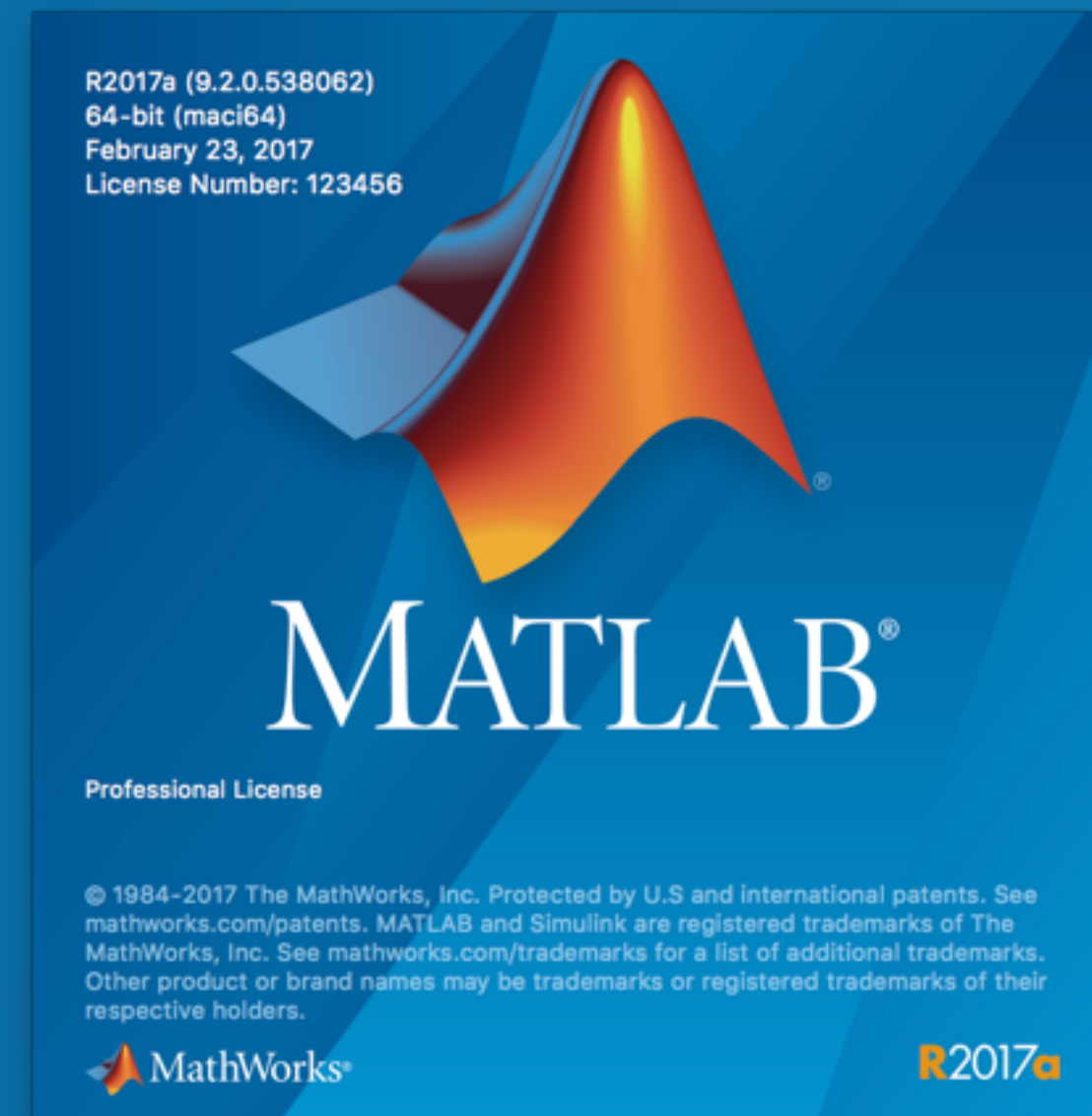
- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB

Entorno de trabajo en Matlab



QUÉ ES MATLAB

ENTORNO DE TRABAJO

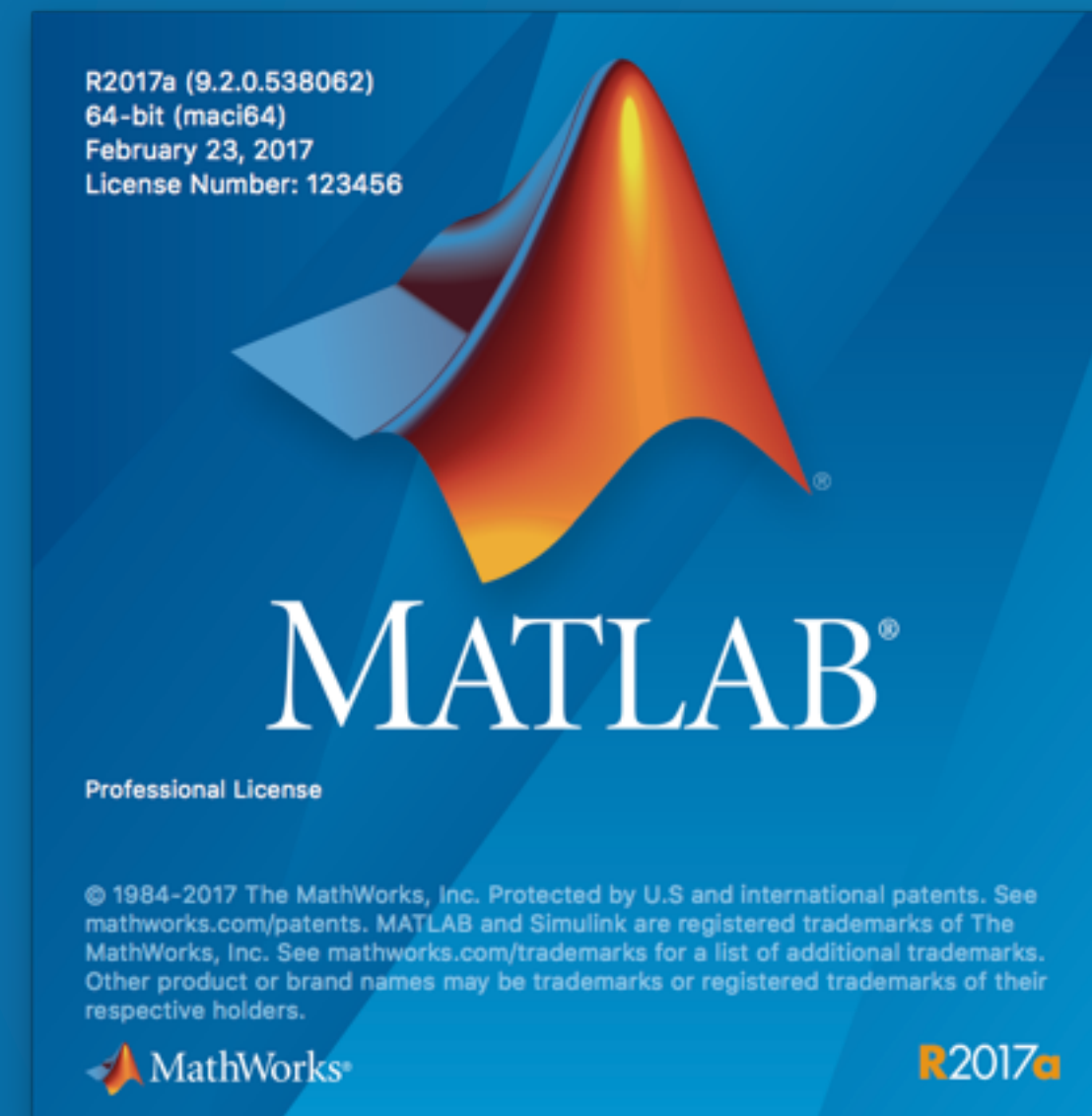
- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB

Entorno de trabajo en Matlab



QUÉ ES MATLAB

ENTORNO DE TRABAJO

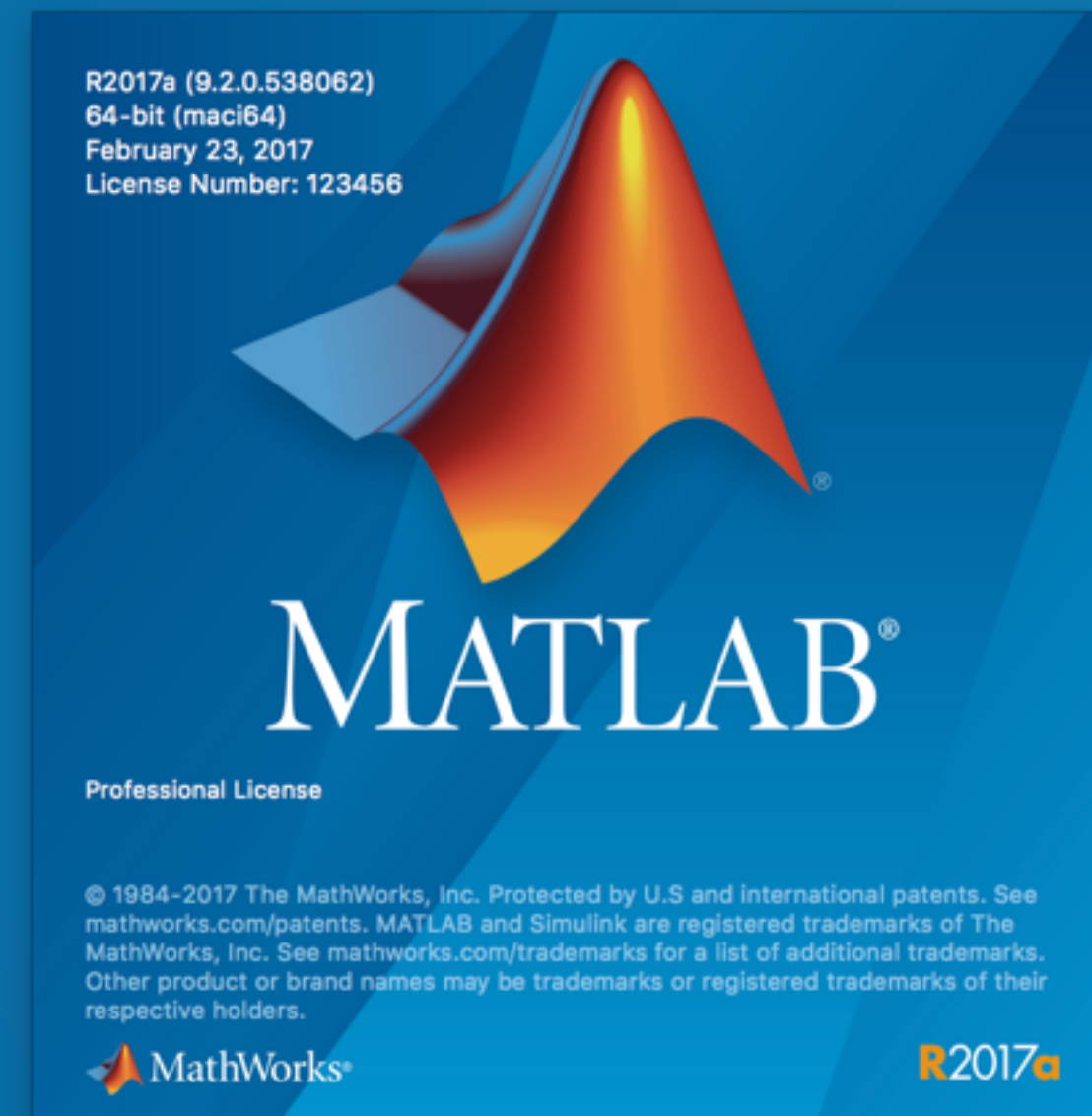
- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB

Entorno de trabajo en Matlab



QUÉ ES MATLAB

ENTORNO DE TRABAJO

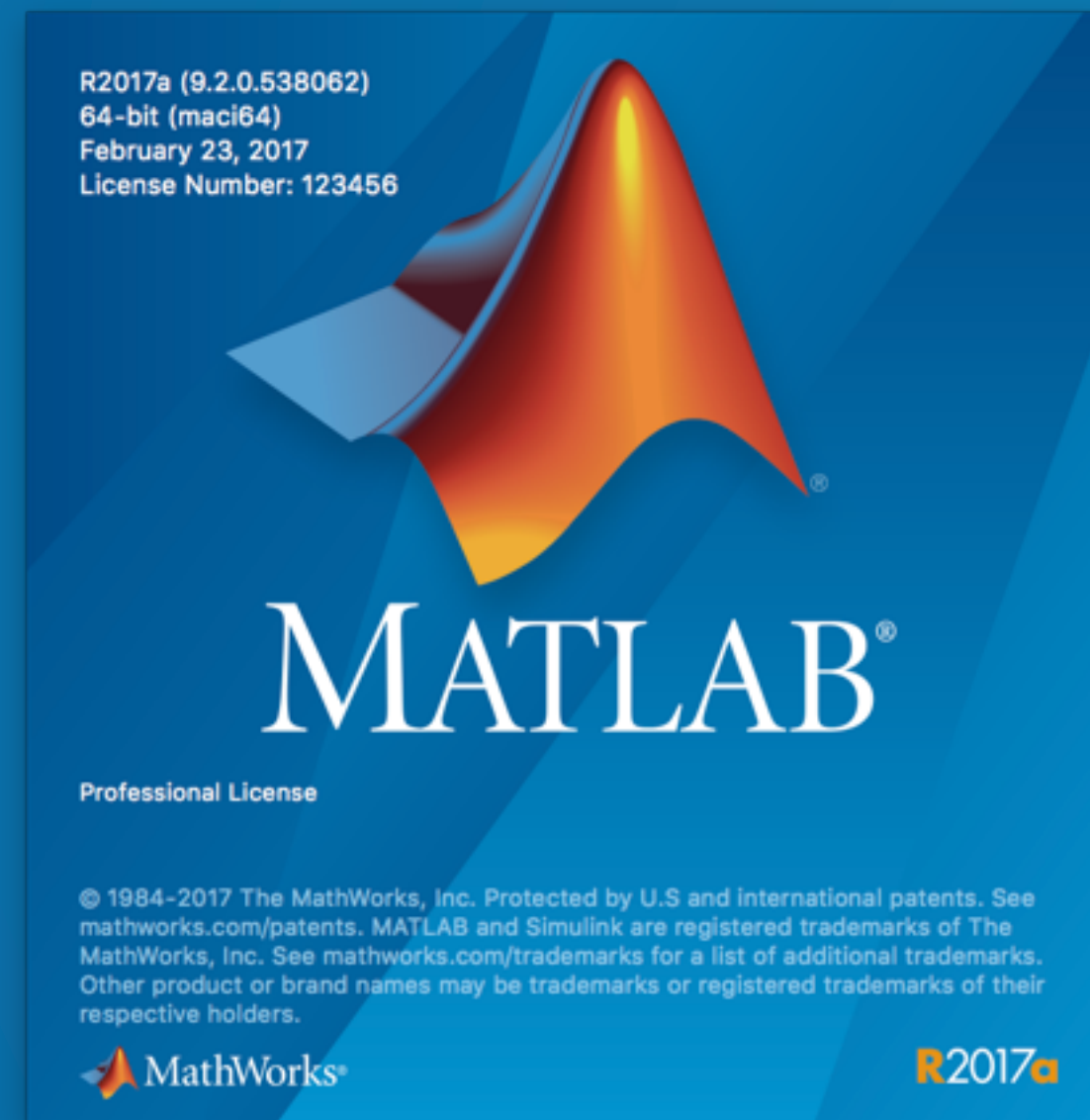
- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB

Entorno de trabajo en Matlab



QUÉ ES MATLAB

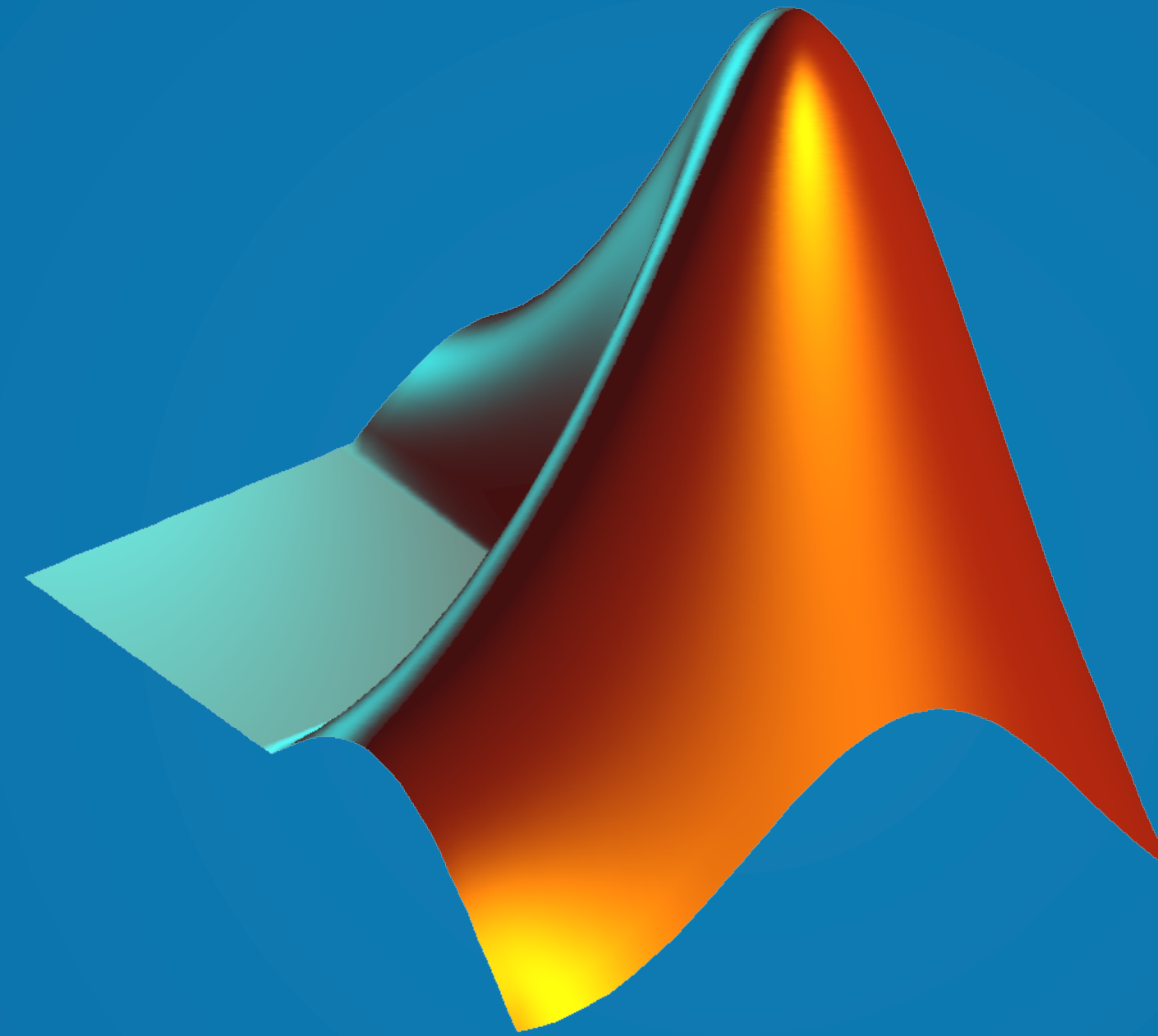
ENTORNO DE TRABAJO

- Path
- Command window
- Workspace
- Inspector de variables
- Editor de script y funciones

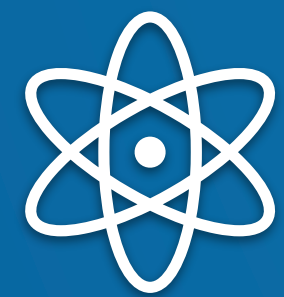
FUNCIONES PROPIAS DE MATLAB

BUENAS PRÁCTICAS

DEPURACIÓN EN MATLAB



¿Qué es Matlab?



FÍSICA



TELECO



GENET



NEURO



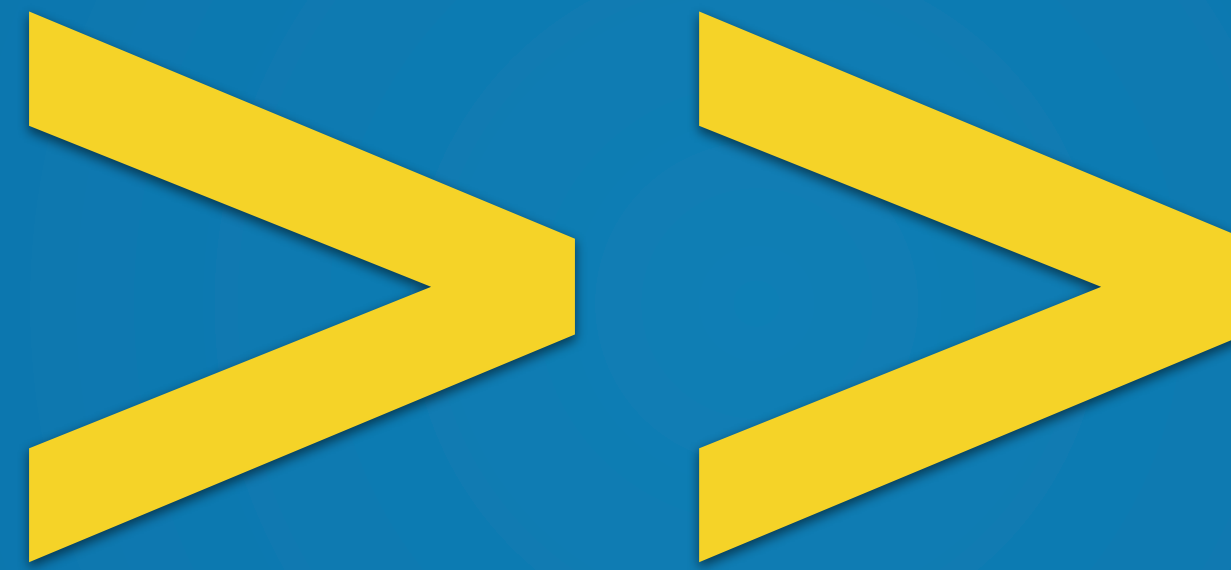
AERO



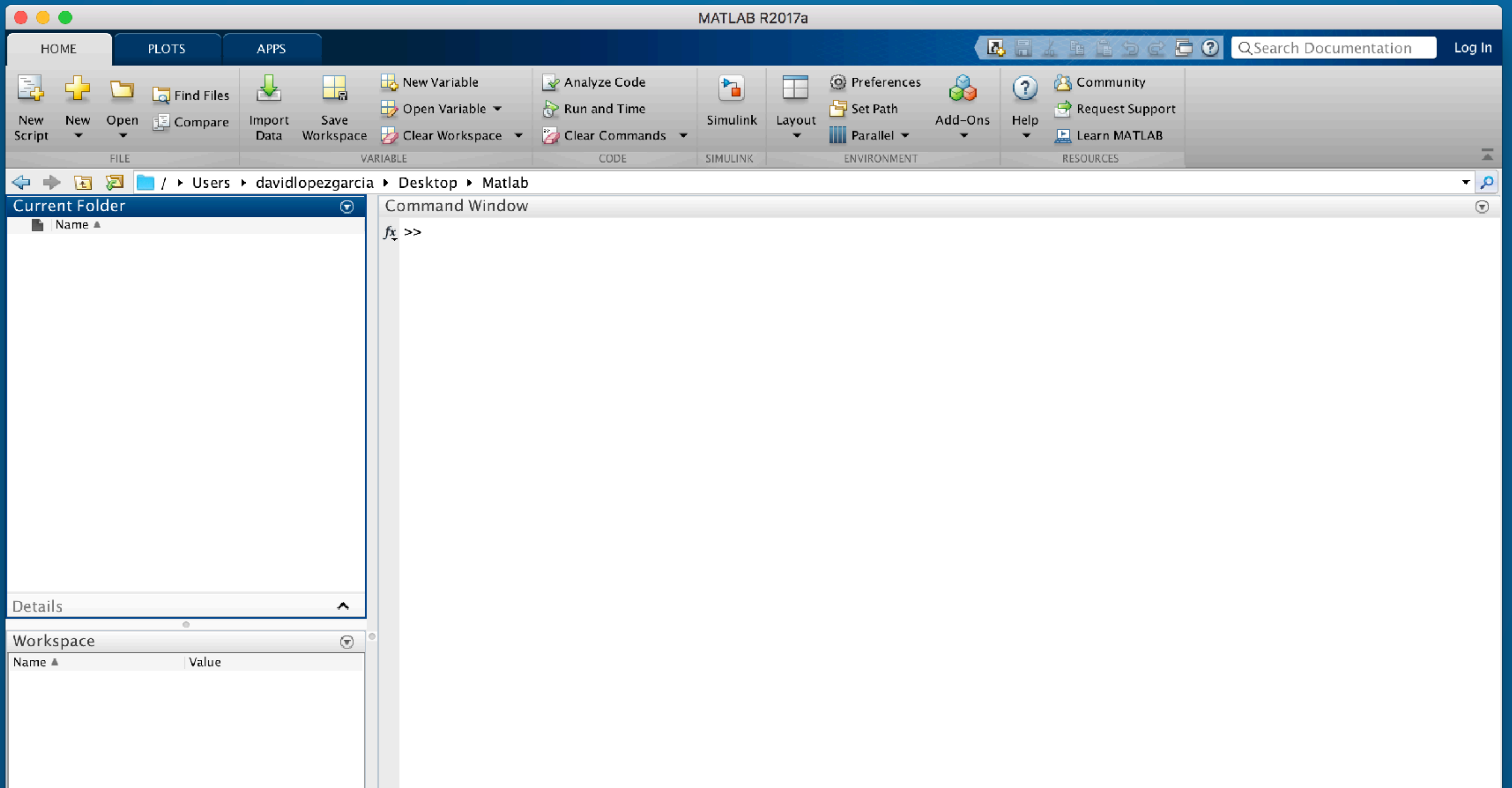
QUÍMICA

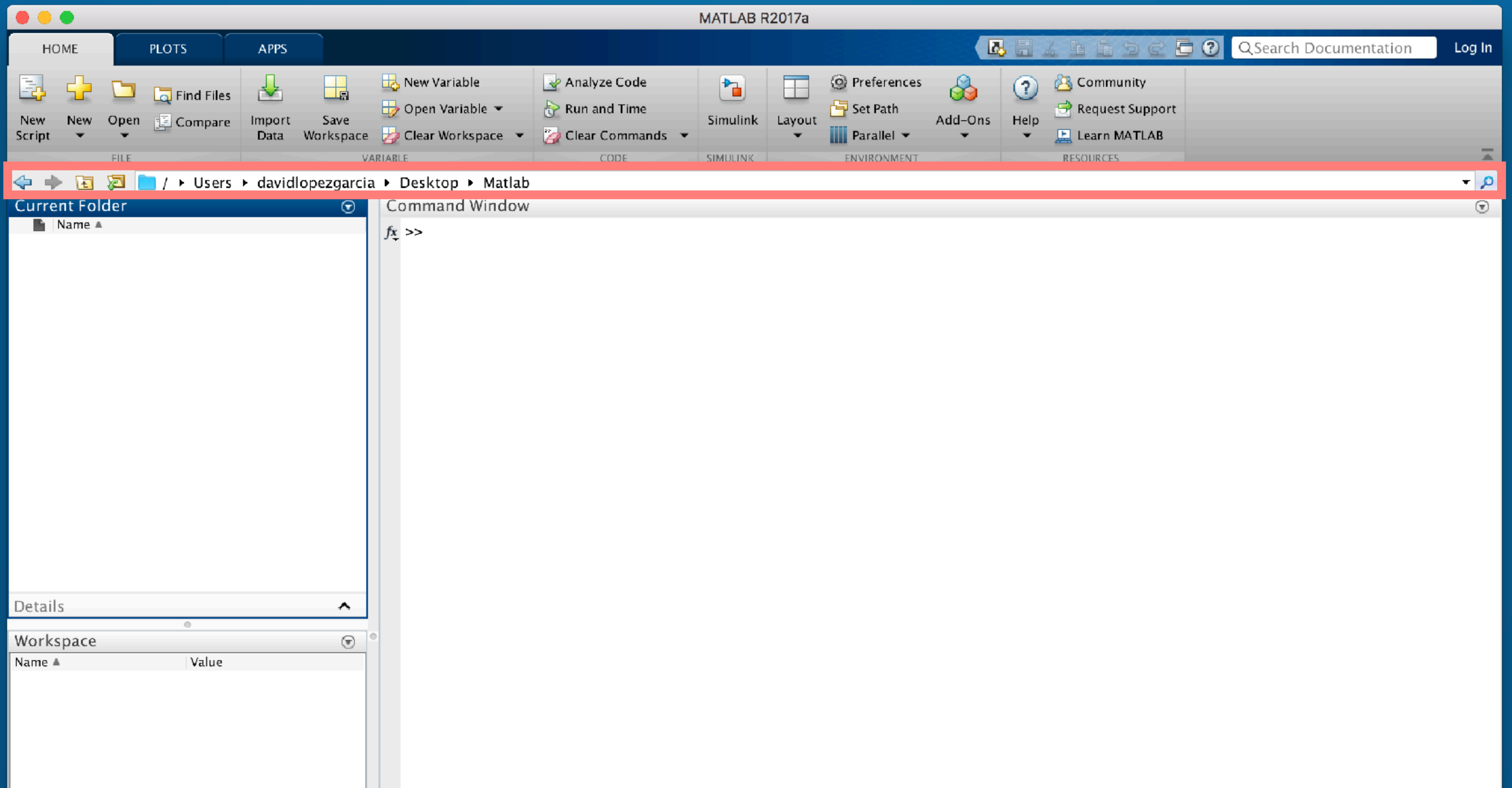


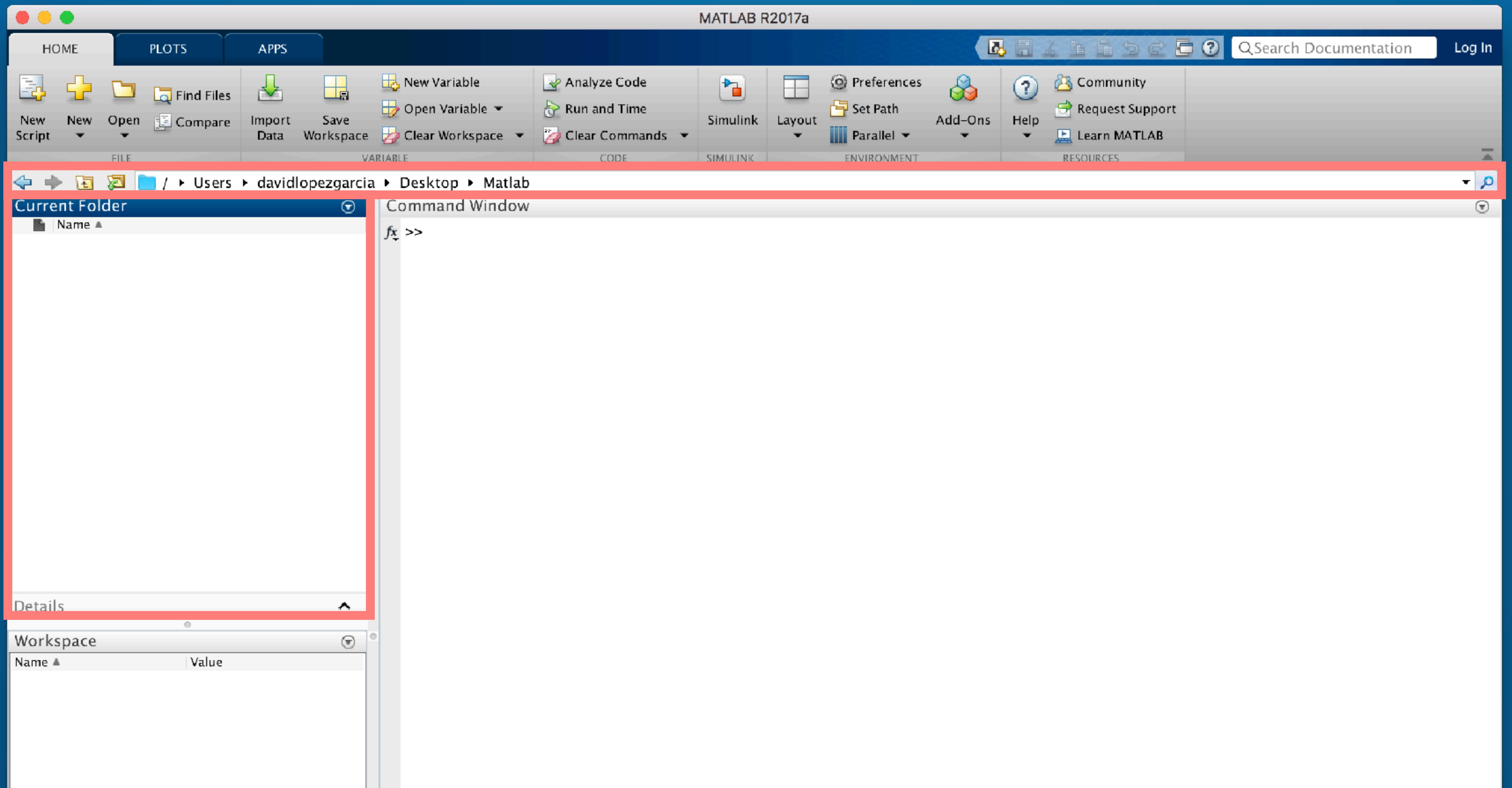
BIO

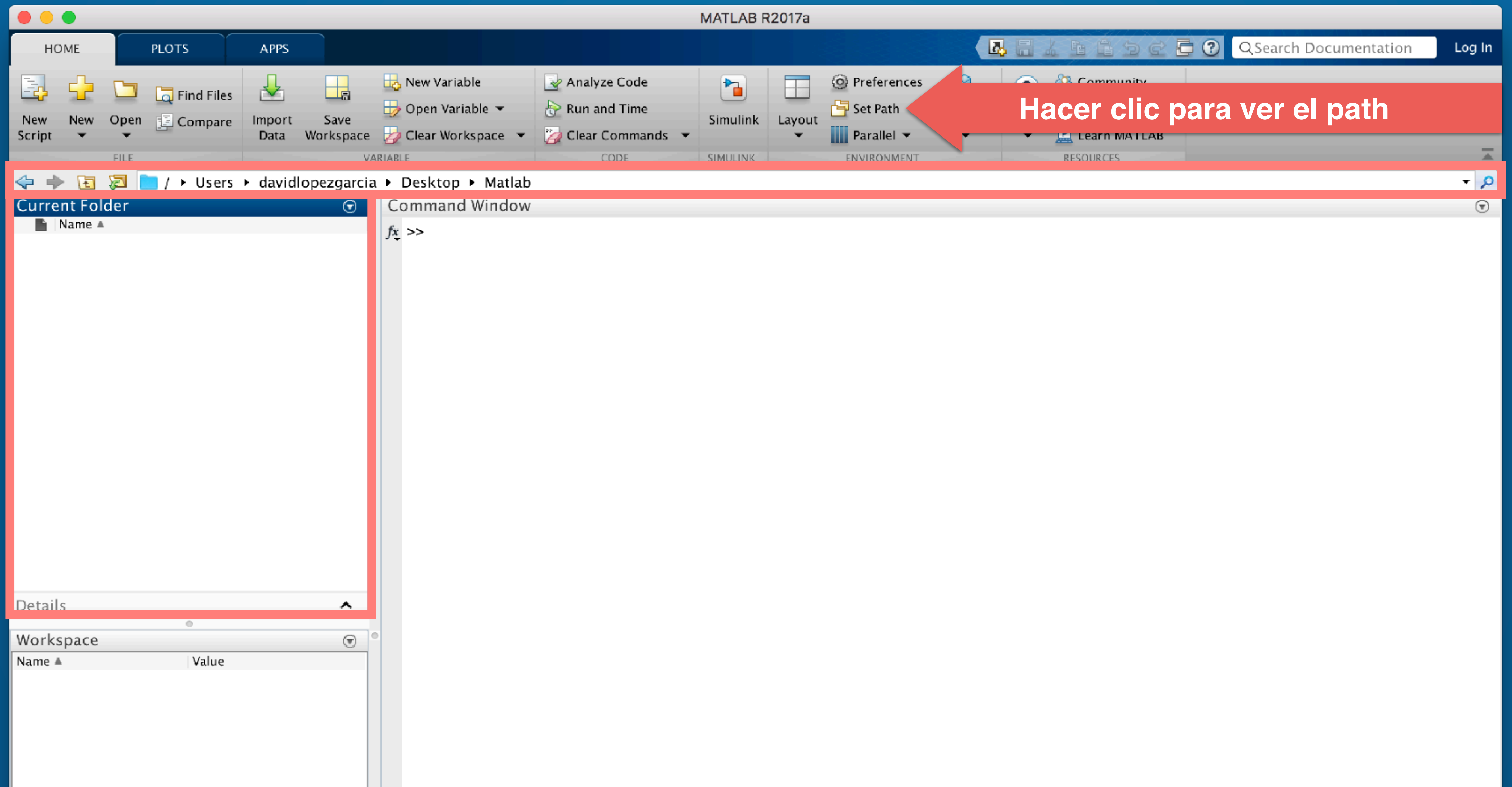


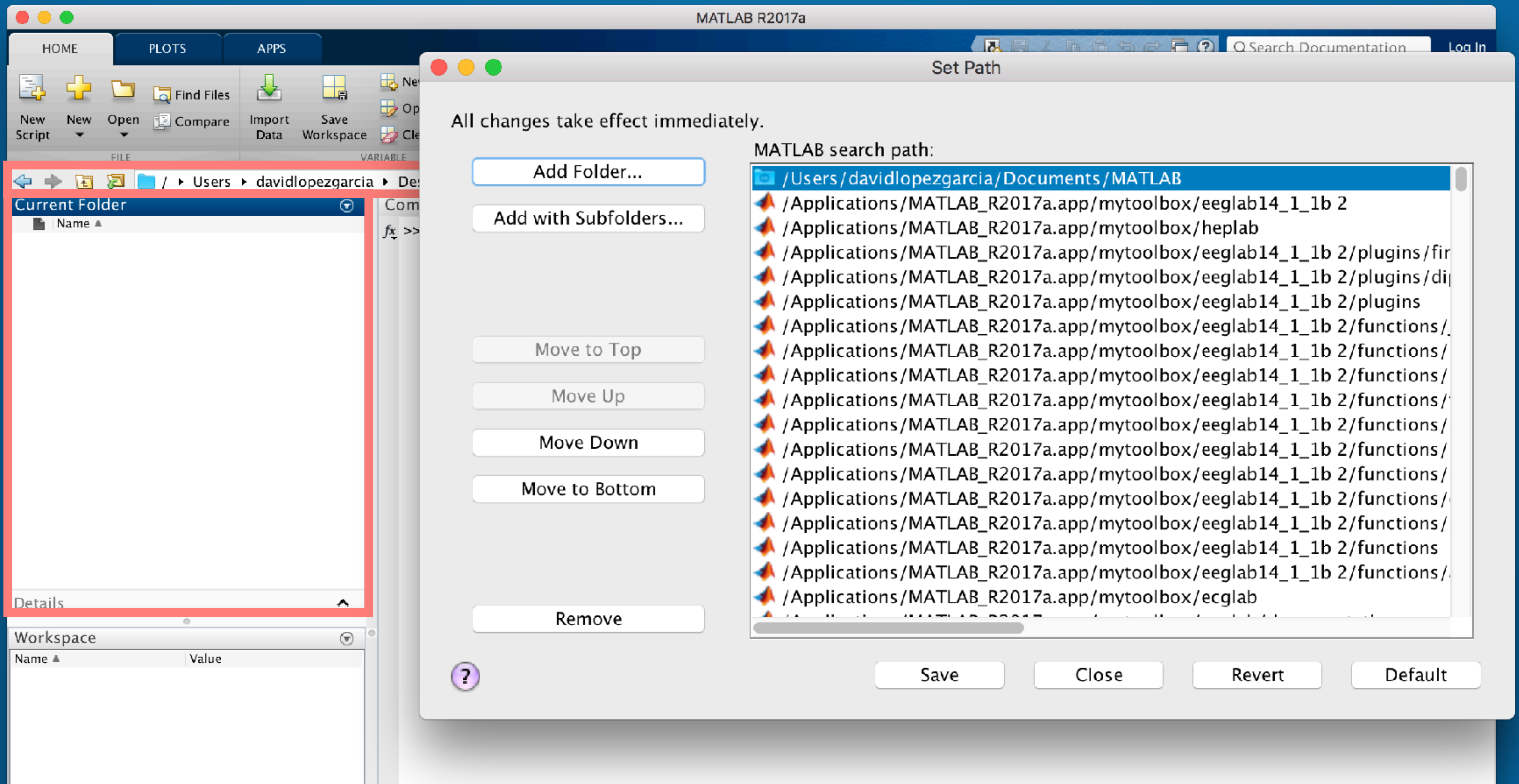
Path, command line, workspace and editor

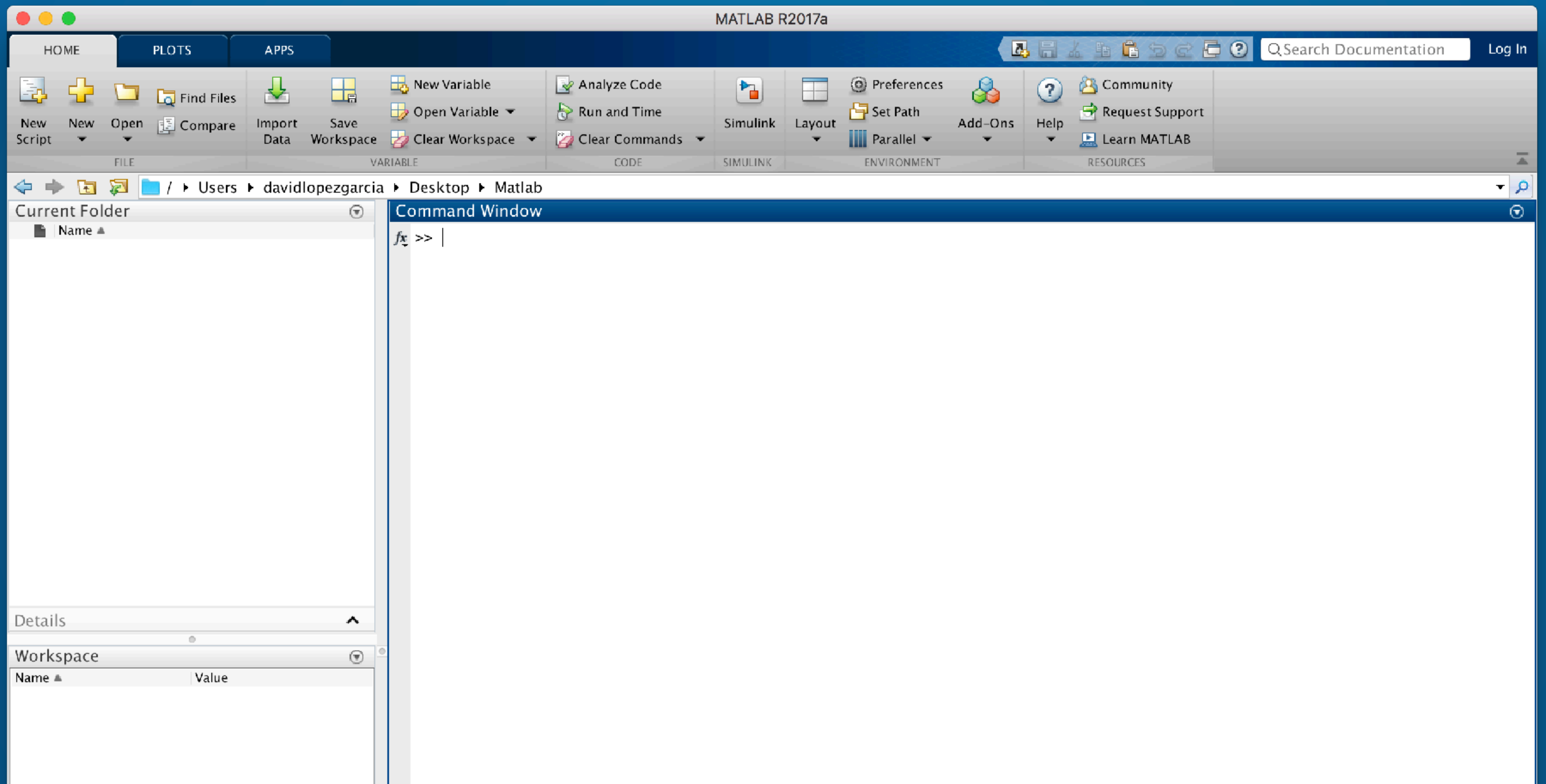


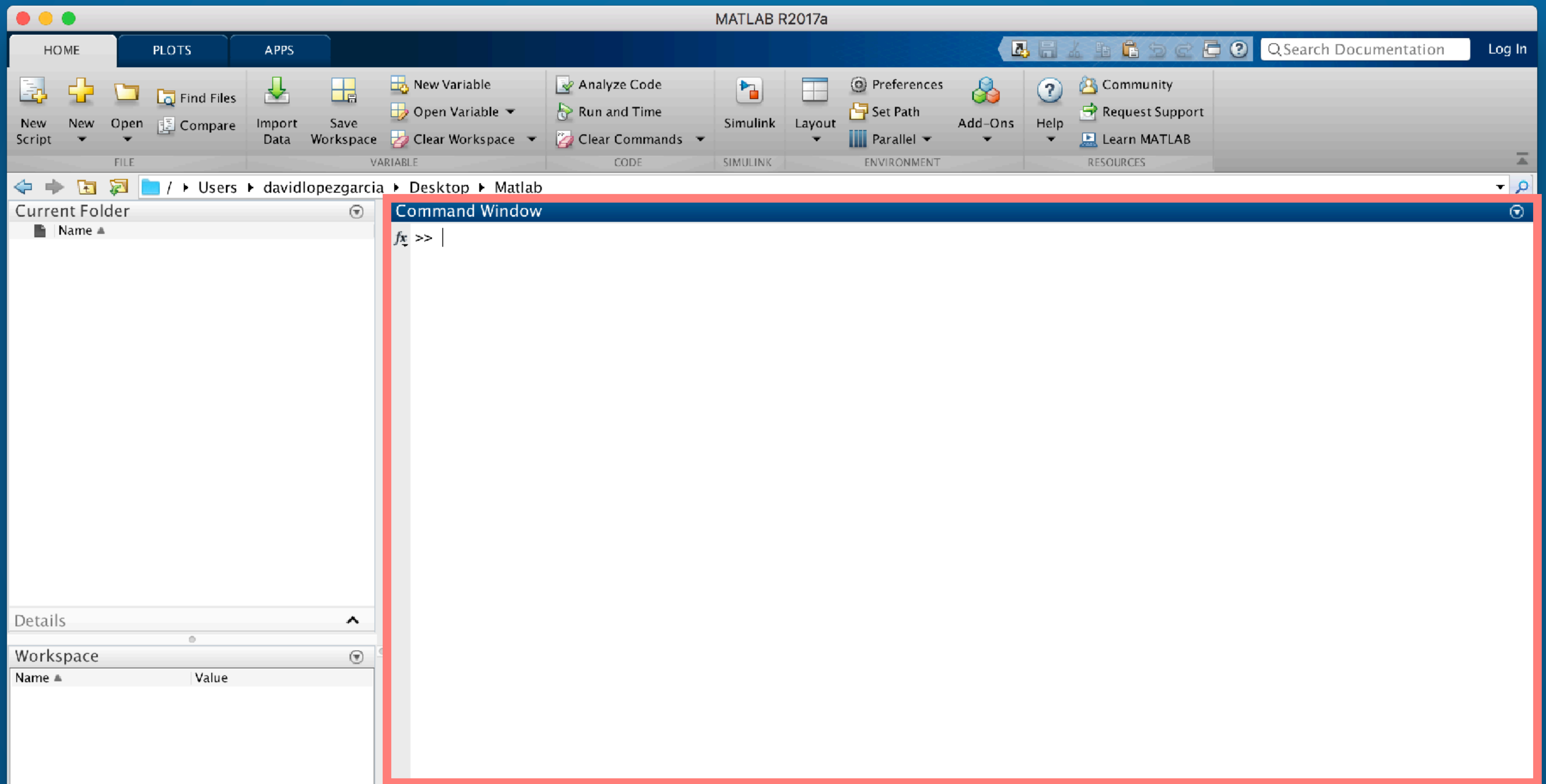


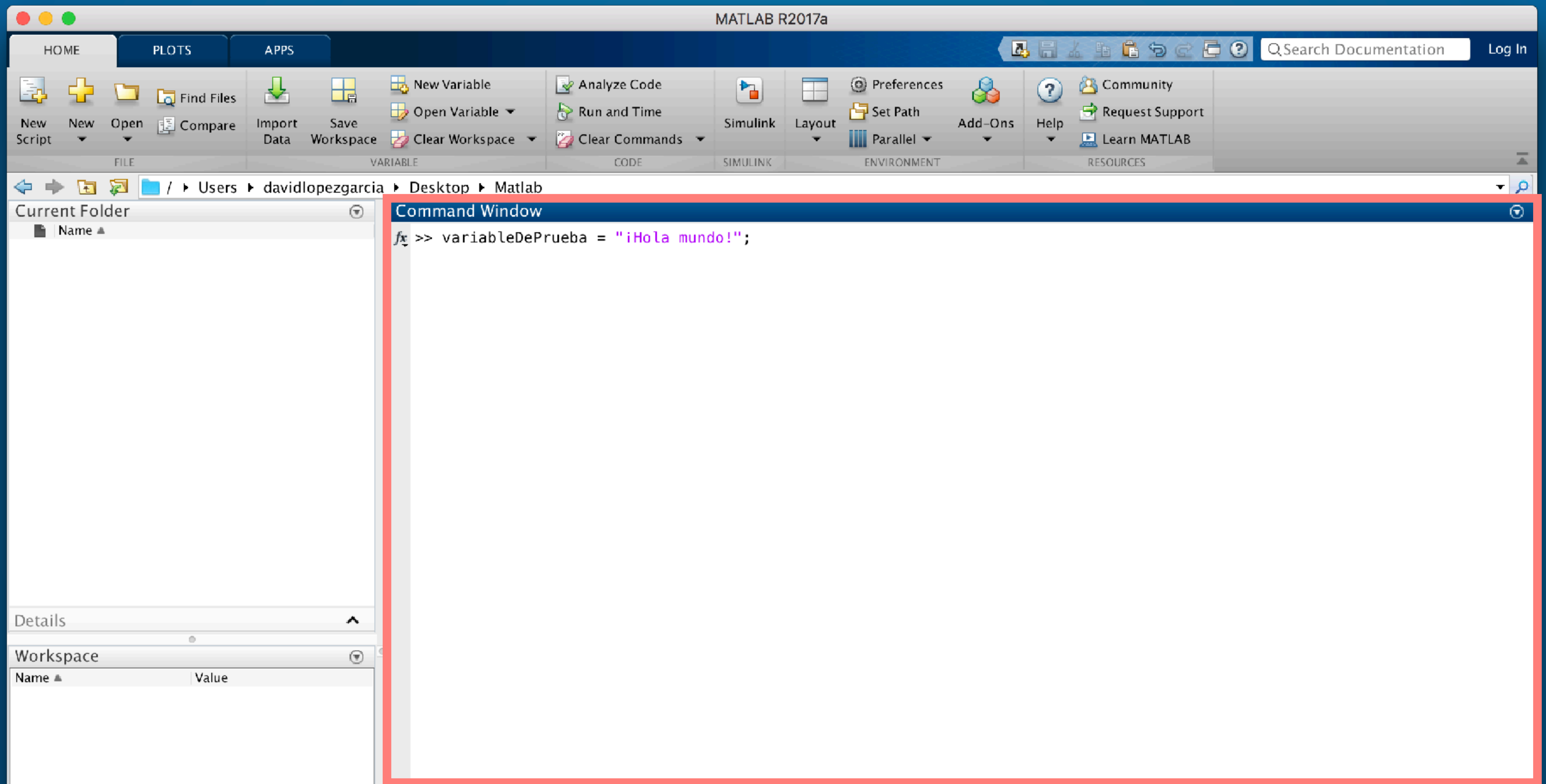












MATLAB R2017a

HOME PLOTS APPS

Search Documentation Log In

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder: / Users davidlopezgarcia Desktop Matlab

Details

Workspace

Name	Value
variableDePrueba	"¡Hola mundo!"

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
fx >> |
```

Script

Data

Workspace

Clear Workspace

Clear Commands

Parallel

Learn MATLAB

FILEVARIABLECODESIMULINKENVIRONMENTRESOURCES

←→

📁

📄

🔍

/ ▸ Users ▸ davidlopezgarcia ▸ Desktop ▸ Matlab

Current Folder

📁

Name ▴

Details

^

Workspace

📄

🔍

Name ▴	Value
str variableDePrueba	"¡Hola mundo!"

Command Window

🔍

```
>> variableDePrueba = "¡Hola mundo!";  
fx >> |
```

Script

Data

Workspace

Clear Workspace

Clear Commands

Parallel

Learn MATLAB

FILEVARIABLECODESIMULINKENVIRONMENTRESOURCES

←→

📁

📄

🔍

/ > Users > davidlopezgarcia > Desktop > Matlab

Current Folder

📁

Name ▲

Details

▲

Workspace

📁

Name ▲	Value
📄 variableDePrueba	"¡Hola mundo!"

Command Window

📄

>> variableDePrueba = "¡Hola mundo!";
fx >> |

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Details

Workspace

Name	Value
variableDePrueba	"¡Hola mundo!"

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
fx>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];|
```

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
fx >> |
```

Details

Workspace

Name	Value
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
variableDePrueba	"¡Hola mundo!"

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
fx>> structDePrueba.data = 3;
```

Details

Workspace

Name	Value
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
variableDePrueba	"¡Hola mundo!"

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
>> structDePrueba.data = 3;  
fx >>
```

Details

Workspace

Name	Value
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
>> structDePrueba.data = 3;  
fx>> condicionDePrueba = false;
```

Details

Workspace

Name	Value
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
>> structDePrueba.data = 3;  
>> condicionDePrueba = false;  
fx >> |
```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

Script Data Workspace Clear Workspace Clear Commands Parallel Learn MATLAB

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Current Folder / Users davidlopezgarcia Desktop Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
>> structDePrueba.data = 3;  
>> condicionDePrueba = false;  
fx >> |
```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3; 4 5 6; 7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

Hacer doble clic para ver el inspector de variables

MATLAB R2017a

HOME PLOTS APPS VARIABLE VIEW

Open Print Rows Columns
 1 1
 Insert Delete Transpose Sort

Current Folder: / Users davidlopezgarcia Desktop Matlab

Variables – matrixDePrueba

matrixDePrueba 3x3 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3										
2	4	5	6										
3	7	8	9										
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3; 4 5 6; 7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

HOME

PLOTS

APPS

VARIABLE

VIEW

New from Selection

Open

Print

Rows

Columns

1

1

Insert

Delete

Transpose

Sort

/

Users

▸

davidlopezgarcia

▸

Desktop

▸

Matlab

Current Folder

Name

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

Variables – matrixDePrueba

matrixDePrueba

3x3 double

	1	2	3	4	5	6	7	8	9	10	11	12	13
1	1	2	3										
2	4	5	6										
3	7	8	9										
4													
5													
6													
7													
8													
9													
10													
11													
12													
13													
14													
15													
16													
17													
18													
19													
20													
21													

MATLAB R2017a

HOME PLOTS APPS

Find Files New Variable Analyze Code Simulink Layout Preferences Set Path Add-Ons Help Community

New Script New Open Compare Import Data Save Workspace Open Variable Run and Time Clear Commands Clear Commands Parallel Parallel Add-Ons Help Community

FILE VARIABLE CODE SIMULINK ENVIRONMENT RESOURCES

Search Documentation Log In

Current Folder

/ > Users > davidlopezgarcia > Desktop > Matlab

Command Window

```
>> variableDePrueba = "¡Hola mundo!";
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];
>> structDePrueba.data = 3;
>> condicionDePrueba = false;
>> |
```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3; 4 5 6; 7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

MATLAB R2017a

HOME PLOTS APPS

Search Documentation Log In

Hacer doble clic para crear un nuevo script

FILE EDIT VIEW TOOLBOX WINDOW HELP

Current Folder: / Users davidlopezgarcia Desktop Matlab

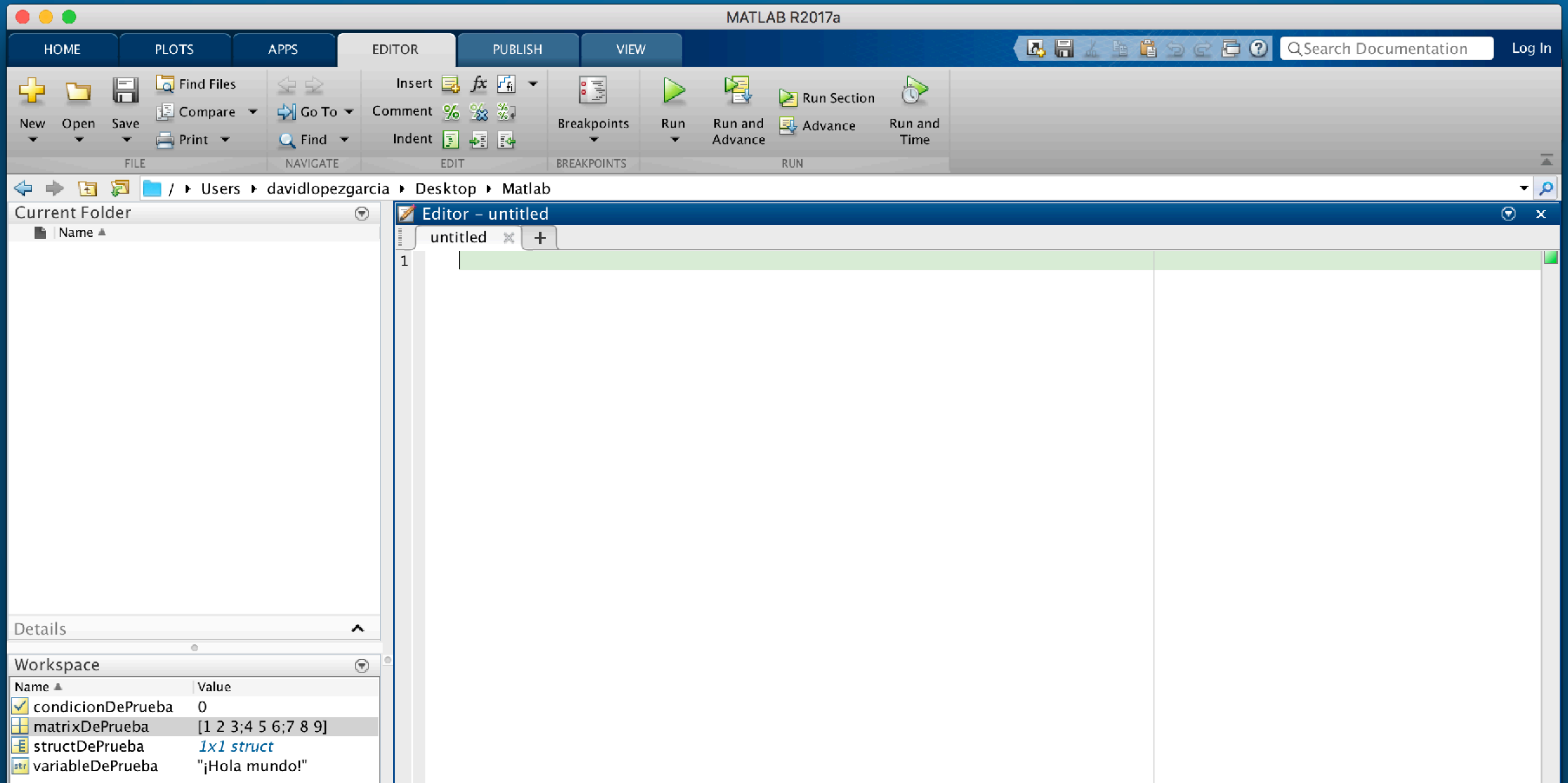
Command Window

```
>> variableDePrueba = "¡Hola mundo!";  
>> matrixDePrueba = [1 2 3 ; 4 5 6 ; 7 8 9];  
>> structDePrueba.data = 3;  
>> condicionDePrueba = false;  
>> |
```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3; 4 5 6; 7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"



MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Current Folder: / Users davidlopezgarcia Desktop Matlab

Editor - untitled*

```

1  %% EVENT-RELATED POTENTIALS TEST (main.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  [~,struc] = fileattrib;
9  PathCurrent = struc.Name;
10
11  path(path,[PathCurrent '/src/func']);
12  path(path,[PathCurrent '/src/mexf/io64']);
13
14  clear all;
15  clc;
16
17  % 1. EXPERIMENT INITIALIZATION
18  % -----
19  run src/session.m           % Initial subject and session data input
20  run src/conf.m              % Experiment configuration
21  run src/ctb.m               % Counterbalance configuration
22  run src/init.m              % Experiment initialization
23  run src/ptb.m               % Psychtoolbox configuration
24
25  % 2. INSTRUCTIONS
26  % -----
27  run src/instructions.m      % Initial instructions

```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation Log In

New Open Save Print

FILE NAVIGATE EDIT BREAKPOINTS RUN

Users davidlopezgarcia Desktop Matlab

Current Folder

Name

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3; 4 5 6; 7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

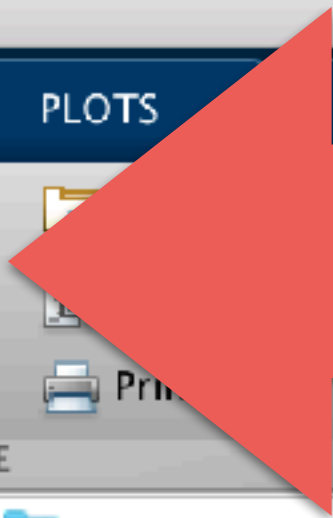
Editor - untitled*

untitled* +

```

1  %% EVENT-RELATED POTENTIALS TEST (main.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  [~,struc] = fileattrib;
9  PathCurrent = struc.Name;
10
11  path(path,[PathCurrent '/src/func']);
12  path(path,[PathCurrent '/src/mexf/io64']);
13
14  clear all;
15  clc;
16
17  % 1. EXPERIMENT INITIALIZATION
18  % -----
19  run src/session.m      % Initial subject and session data input
20  run src/conf.m         % Experiment configuration
21  run src/ctb.m          % Counterbalance configuration
22  run src/init.m         % Experiment initialization
23  run src/ptb.m          % Psychtoolbox configuration
24
25  % 2. INSTRUCTIONS
26  % -----
27  run src/instructions.m % Initial instructions

```



Hacer clic para guardar el script

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation Log In

New Open Save Print

FILE NAVIGATE EDIT BREAKPOINTS RUN

Users davidlopezgarcia Desktop Matlab

Current Folder

Name

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

Editor - untitled*

untitled* +

```
1 %% EVENT-RELATED POTENTIALS TEST (main.m)
2 % -----
3 % David López García
4 % dlope
5 % CIMCYC
6 % -----
7
8 [~, structDePrueba] = structDePrueba(
9 PathCurrentFolder);
10
11 path(pathDePrueba, structDePrueba);
12 path(pathDePrueba, structDePrueba);
13
14 clear all;
15 clc;
16
17 % 1. EXPERIMENTAL DATA
18 % -----
19 run src/experimental_data.m;
20 run src/experimental_data.m;
21 run src/experimental_data.m;
22 run src/experimental_data.m;
23 run src/ptb.m; % Psychtoolbox configuration
24
25 % 2. INSTRUCTIONS
26 % -----
27 run src/instructions.m; % Initial instructions
```

Save As: scriptDePrueba

Tags:

Where: Matlab

Format: MATLAB Code files (*.m)

Cancel Save

Hacer clic para guardar el script

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Run Run and Advance Run Section Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Current Folder: /Users/davidlopezgarcia/Desktop/Matlab

scriptDePrueba.m

Editor - /Users/davidlopezgarcia/Desktop/Matlab/scriptDePrueba.m

```

1  %% EVENT-RELATED POTENTIALS TEST (main.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  [~,struc] = fileattrib;
9  PathCurrent = struc.Name;
10
11 path(path,[PathCurrent '/src/func']);
12 path(path,[PathCurrent '/src/mexf/io64']);
13
14 clear all;
15 clc;
16
17 % 1. EXPERIMENT INITIALIZATION
18 % -----
19 run src/session.m      % Initial subject and session data input
20 run src/conf.m         % Experiment configuration
21 run src/ctb.m          % Counterbalance configuration
22 run src/init.m         % Experiment initialization
23 run src/ptb.m          % Psychtoolbox configuration
24
25 % 2. INSTRUCTIONS
26 % -----
27 run src/instructions.m % Initial instructions

```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Search Documentation Log In

Hacer clic para ejecutar el script

Run Run and Advance Run Section Advance Run and Time

FILE NAVIGATE EDIT BREAKPOINTS RUN

Current Folder

/ Users davidlopezgarcia Desktop Matlab

scriptDePrueba.m

Editor - /Users/davidlopezgarcia/Desktop/Matlab/scriptDePrueba.m

```

1  %% EVENT-RELATED POTENTIALS TEST (main.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  [~,struc] = fileattrib;
9  PathCurrent = struc.Name;
10
11 path(path,[PathCurrent '/src/func']);
12 path(path,[PathCurrent '/src/mexf/io64']);
13
14 clear all;
15 clc;
16
17 % 1. EXPERIMENT INITIALIZATION
18 % -----
19 run src/session.m      % Initial subject and session data input
20 run src/conf.m         % Experiment configuration
21 run src/ctb.m          % Counterbalance configuration
22 run src/init.m         % Experiment initialization
23 run src/ptb.m          % Psychtoolbox configuration
24
25 % 2. INSTRUCTIONS
26 % -----
27 run src/instructions.m % Initial instructions

```

Details

Workspace

Name	Value
condicionDePrueba	0
matrixDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	"¡Hola mundo!"



Funciones y comandos útiles de Matlab

► Funciones propias de Matlab ► Borrar el workspace



Workspace	
Name ▲	Value
<input checked="" type="checkbox"/> binDePrueba	0
<input type="checkbox"/> matrizDePrueba	[1 2 3 4 5;6 7 8 9 10]
<input type="checkbox"/> stringDePrueba	"¡Hola mundo!"
<input type="checkbox"/> variableDePrueba	1
<input type="checkbox"/> vectorDePrueba	[1 2 3 4 5]



Workspace	
Name ▲	Value

► Funciones propias de Matlab ► Borrar el workspace



Workspace	
Name ▲	Value
<input checked="" type="checkbox"/> binDePrueba	0
<input type="checkbox"/> matrizDePrueba	[1 2 3 4 5;6 7 8 9 10]
<input type="checkbox"/> stringDePrueba	"¡Hola mundo!"
<input type="checkbox"/> variableDePrueba	1
<input type="checkbox"/> vectorDePrueba	[1 2 3 4 5]



Workspace	
Name ▲	Value

clear all



Borra todas la variables definidas en el workspace y libera memoria.

Workspace	
Name ▲	Value
<input checked="" type="checkbox"/> binDePrueba	0
<input type="checkbox"/> matrizDePrueba	[1 2 3 4 5;6 7 8 9 10]
<input type="checkbox"/> stringDePrueba	"¡Hola mundo!"
<input type="checkbox"/> variableDePrueba	1
<input type="checkbox"/> vectorDePrueba	[1 2 3 4 5]



Workspace	
Name ▲	Value

clear all

Borra todas la variables definidas en el workspace y libera memoria.

EJEMPLO DE USO:

```
clear all;
```

Workspace	
Name ▲	Value
<input checked="" type="checkbox"/> binDePrueba	0
<input type="checkbox"/> matrizDePrueba	[1 2 3 4 5;6 7 8 9 10]
<input type="checkbox"/> stringDePrueba	"¡Hola mundo!"
<input type="checkbox"/> variableDePrueba	1
<input type="checkbox"/> vectorDePrueba	[1 2 3 4 5]



Workspace	
Name ▲	Value

clear all

Borra todas la variables definidas en el workspace y libera memoria.

EJEMPLO DE USO:

```
clear all;
```

Workspace	
Name ▲	Value
<input checked="" type="checkbox"/> binDePrueba	0
<input type="checkbox"/> matrizDePrueba	[1 2 3 4 5;6 7 8 9 10]
<input type="checkbox"/> stringDePrueba	"¡Hola mundo!"
<input type="checkbox"/> variableDePrueba	1
<input type="checkbox"/> vectorDePrueba	[1 2 3 4 5]



Workspace	
Name ▲	Value

► Funciones propias de Matlab ► Borrar la línea de comandos



```
Command Window
>> variableDePrueba = 1;
>> vectorDePrueba = [1 2 3 4 5];
>> matrizDePrueba = [1 2 3 4 5 ; 6 7 8 9 10];
>> binDePrueba = false;
>> stringDePrueba = "¡Hola mundo!";
fx >>
```



```
Command Window
fx >>
```

► Funciones propias de Matlab ► Borrar la línea de comandos



```
Command Window
>> variableDePrueba = 1;
>> vectorDePrueba = [1 2 3 4 5];
>> matrizDePrueba = [1 2 3 4 5 ; 6 7 8 9 10];
>> binDePrueba = false;
>> stringDePrueba = "¡Hola mundo!";
fx >>
```



```
Command Window
fx >>
```

clc



Limpia por completo la línea de comandos.

```
Command Window
>> variableDePrueba = 1;
>> vectorDePrueba = [1 2 3 4 5];
>> matrizDePrueba = [1 2 3 4 5 ; 6 7 8 9 10];
>> binDePrueba = false;
>> stringDePrueba = "¡Hola mundo!";
fx >>
```



```
Command Window
fx >>
```


clc



Limpia por completo la línea de comandos.

EJEMPLO DE USO:

clc;

```
Command Window
>> variableDePrueba = 1;
>> vectorDePrueba = [1 2 3 4 5];
>> matrizDePrueba = [1 2 3 4 5 ; 6 7 8 9 10];
>> binDePrueba = false;
>> stringDePrueba = "¡Hola mundo!";
fx >>
```



```
Command Window
fx >>
```

clc



Limpia por completo la línea de comandos.

EJEMPLO DE USO:

clc;

```
Command Window
>> variableDePrueba = 1;
>> vectorDePrueba = [1 2 3 4 5];
>> matrizDePrueba = [1 2 3 4 5 ; 6 7 8 9 10];
>> binDePrueba = false;
>> stringDePrueba = "¡Hola mundo!";
fx >>
```



```
Command Window
fx >>
```

► Funciones propias de Matlab ► Función para mostrar texto por pantalla



Command Window

```
>> disp('¡Hola mundo!')  
¡Hola mundo!  
fx >> |
```

disp([string]) ➡

Command Window

```
>> disp('¡Hola mundo!')  
¡Hola mundo!  
fx >> |
```

disp([string]) → Muestra en la línea de comandos la cadena pasada como argumento.

Command Window

```
>> disp('¡Hola mundo!')  
¡Hola mundo!  
fx >> |
```


disp([string]) ➔ Muestra en la línea de comandos la cadena pasada como argumento.

EJEMPLO DE USO:

disp('¡Hola mundo!');

```
Command Window
>> disp('¡Hola mundo!')
¡Hola mundo!
fx >> |
```




sum([var]) →



sum([var]) → Devuelve el valor correspondiente a la suma de todos los valores de [arg]




sum([var])  Devuelve el valor correspondiente a la suma de todos los valores de [arg]

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];
```



sum([var])  Devuelve el valor correspondiente a la suma de todos los valores de [arg]

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];
```

```
suma = sum(vector);
```



sum([var]) → Devuelve el valor correspondiente a la suma de todos los valores de [arg]

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];
```

```
suma = sum(vector);
```

suma → ans = **55**



mean([var]) →



mean([var]) → Devuelve el valor correspondiente a la media de todos los valores de [arg]



mean([var]) → Devuelve el valor correspondiente a la media de todos los valores de [arg]

EJEMPLO DE USO:


```
vector = [1,2,3,4,5,6,7,8,9,10];
```



mean([var]) → Devuelve el valor correspondiente a la media de todos los valores de [arg]

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];  
media = mean(vector);
```



mean([var]) → Devuelve el valor correspondiente a la media de todos los valores de [arg]

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];
```

```
media = mean(vector);
```

```
media → ans = 5.5
```



length([var]) →



length([var]) → Devuelve el número total de elementos de la variable pasada como argumento.



length([var]) → Devuelve el número total de elementos de la variable pasada como argumento.

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];
```



length([var]) → Devuelve el número total de elementos de la variable pasada como argumento.

EJEMPLO DE USO:

```
vector = [1,2,3,4,5,6,7,8,9,10];  
longitud = length(vector);
```

length([var]) → Devuelve el número total de elementos de la variable pasada como argumento.

EJEMPLO DE USO:

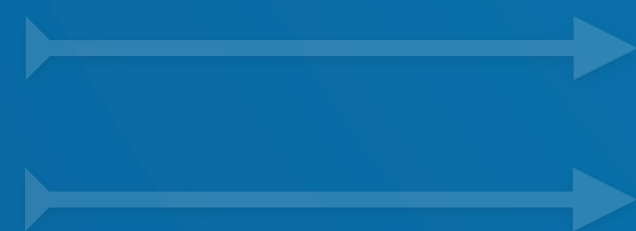
```
vector = [1,2,3,4,5,6,7,8,9,10];
```

```
longitud = length(vector);
```

```
longitud → ans = 10
```

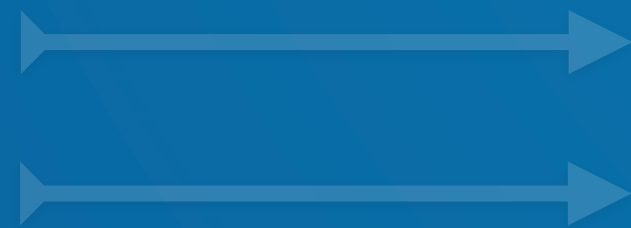


size([var]) →



size([var]) →

Devuelve el número total de filas y columnas de la variable pasada como argumento.

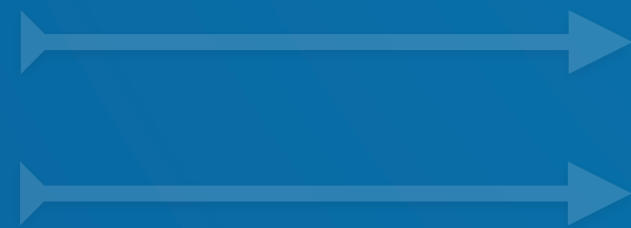


size([var]) →

Devuelve el número total de filas y columnas de la variable pasada como argumento.

EJEMPLO DE USO:

```
matrix = [1,2,3 ; 4,5,6 ; 7,8,9];
```

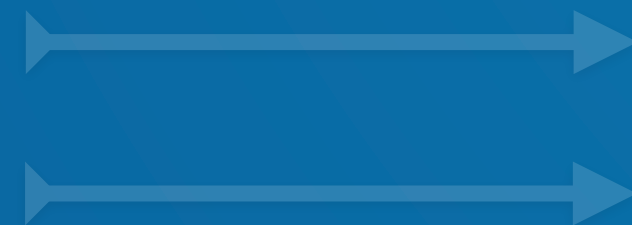


size([var]) →

Devuelve el número total de filas y columnas de la variable pasada como argumento.

EJEMPLO DE USO:

```
matrix = [1,2,3 ; 4,5,6 ; 7,8,9];  
[row, col] = size(matrix);
```



size([var]) →

Devuelve el número total de filas y columnas de la variable pasada como argumento.

EJEMPLO DE USO:

```
matrix = [1,2,3 ; 4,5,6 ; 7,8,9];
```

```
[row, col] = size(matrix);
```

row → ans = 3

→

size([var]) →

Devuelve el número total de filas y columnas de la variable pasada como argumento.

EJEMPLO DE USO:

```
matrix = [1,2,3 ; 4,5,6 ; 7,8,9];
```

```
[row, col] = size(matrix);
```

row → ans = 3

col → ans = 3



zeros([m],[n]) →



zeros([m],[n]) → Devuelve una matriz *mxn* con todos los valores a 0.



zeros([m],[n]) → Devuelve una matriz *mxn* con todos los valores a 0.

EJEMPLO DE USO:

```
matrix = zeros(3,10);
```



zeros([m],[n]) → Devuelve una matriz $m \times n$ con todos los valores a 0.

EJEMPLO DE USO:

```
matrix = zeros(3,10);
```

```
matrix → ans =
```

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



ones([m],[n]) →



ones([m],[n]) → Devuelve una matriz *mxn* con todos los valores a 1.



ones([m],[n]) → Devuelve una matriz *mxn* con todos los valores a 1.

EJEMPLO DE USO:

```
matrix = ones(3,3);
```



ones([m],[n]) → Devuelve una matriz *mxn* con todos los valores a 1.

EJEMPLO DE USO:

```
matrix = ones(3,3);
```

```
matrix → ans = 1 1 1  
              1 1 1  
              1 1 1
```



NaN([m],[n]) →



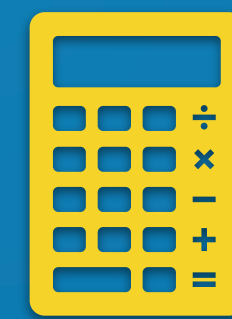
NaN([m],[n]) → Devuelve una matriz *mxn* con todos los valores a NaN.



NaN([m],[n])



Devuelve una matriz *mxn* con todos los valores a NaN.



$$0 \div 0 = ?$$



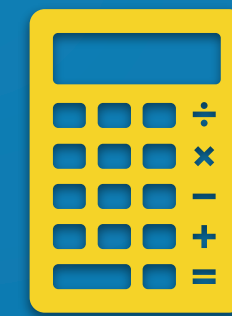
NaN([m],[n])



Devuelve una matriz *mxn* con todos los valores a NaN.

EJEMPLO DE USO:

```
matrix = NaN(3,3);
```



$0 \div 0 = ?$

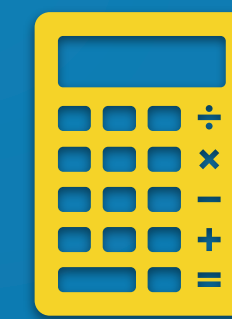


NaN([m],[n]) → Devuelve una matriz *mxn* con todos los valores a NaN.

EJEMPLO DE USO:

```
matrix = NaN(3,3);
```

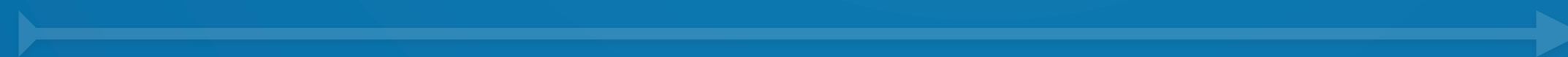
```
matrix → ans = NaN NaN NaN  
          NaN NaN NaN  
          NaN NaN NaN
```



→ $0 \div 0 = ?$



rand()



rand()



Devuelve un valor aleatorio en el intervalo [0,1]



rand()  Devuelve un valor aleatorio en el intervalo [0,1]

EJEMPLO DE USO:

```
aleatorio = rand();
```



rand() → Devuelve un valor aleatorio en el intervalo [0,1]

EJEMPLO DE USO:

```
aleatorio = rand();
```

aleatorio → ans = 0.8763



rand([m],[n]) →



rand([m],[n]) → Devuelve una matriz *mxn* de números aleatorios en el intervalo [0,1]



rand([m],[n]) → Devuelve una matriz *mxn* de números aleatorios en el intervalo [0,1]

EJEMPLO DE USO:

```
rndMatrix = rand(1,3) ;
```



rand([m],[n]) → Devuelve una matriz *mxn* de números aleatorios en el intervalo [0,1]

EJEMPLO DE USO:

```
rndMatrix = rand(1,3);
```

```
rndMatrix → ans = 0.8763 0.2345 0.9854
```




randn([m],[n]) →



randn([m],[n]) → Devuelve una matriz *mxn* de números aleatorios con distribución normal.



randn([m],[n]) → Devuelve una matriz *mxn* de números aleatorios con distribución normal.

EJEMPLO DE USO:

```
rndMatrix = randn(1,3);
```



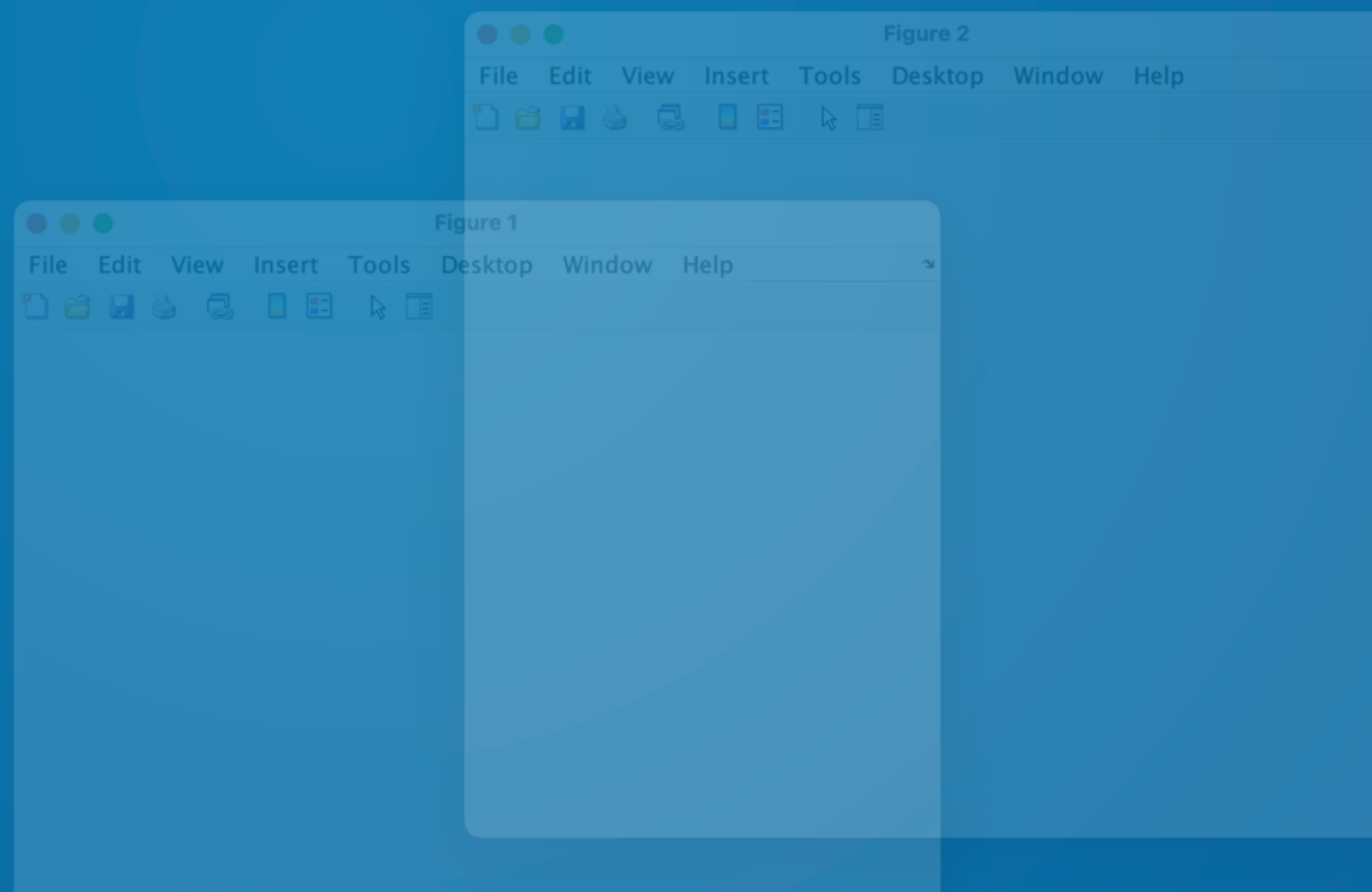
randn([m],[n]) → Devuelve una matriz *mxn* de números aleatorios con distribución normal.

EJEMPLO DE USO:

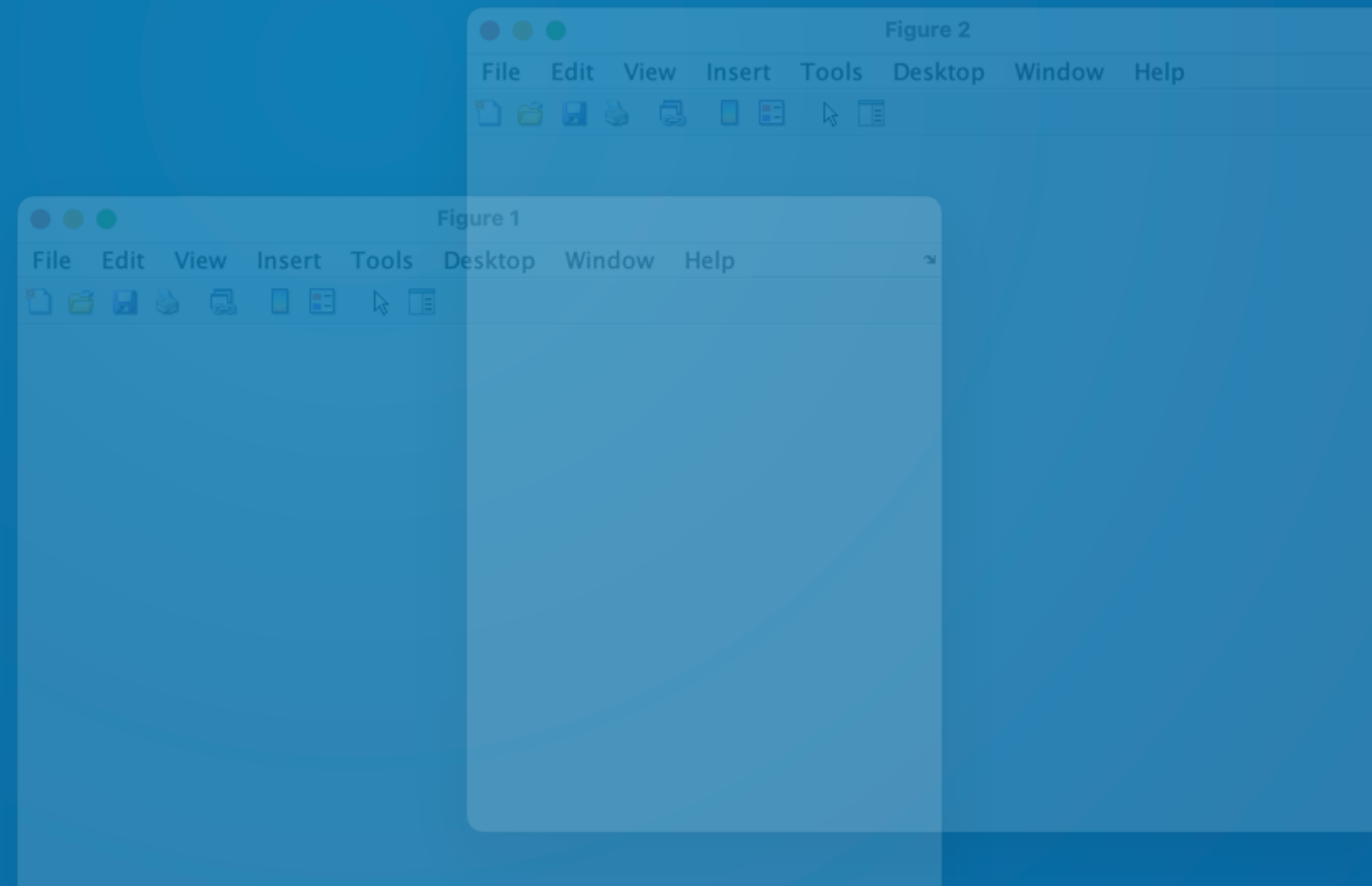
```
rndMatrix = randn(1,3);
```

```
rndMatrix → ans = -1.3077   -0.4336   0.3426
```


► Funciones propias de Matlab ► Funciones para gráficos

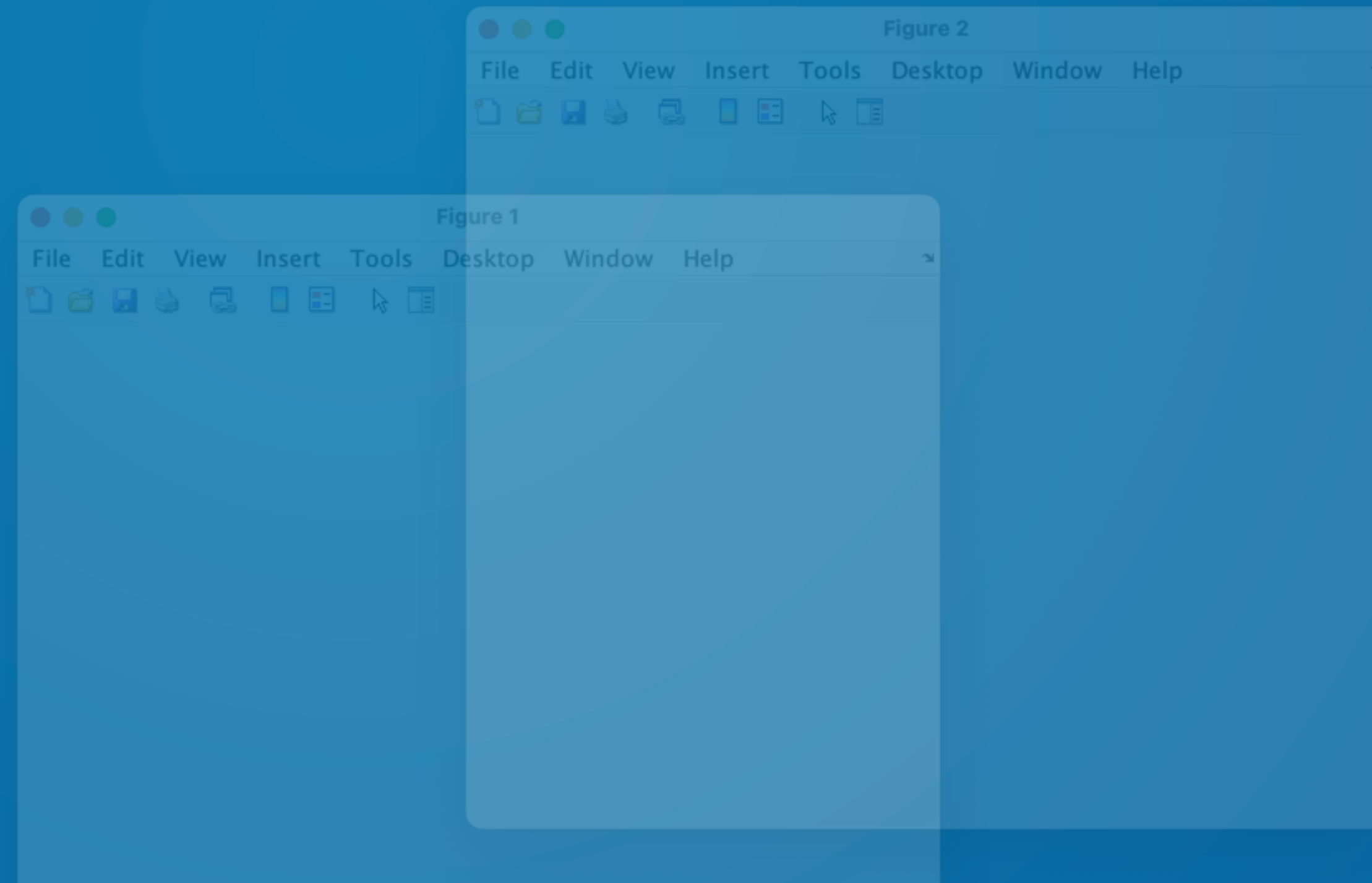


figure([int]) →



figure([int])

Genera una figura vacía en la que podemos representar gráficos, imágenes, etc.

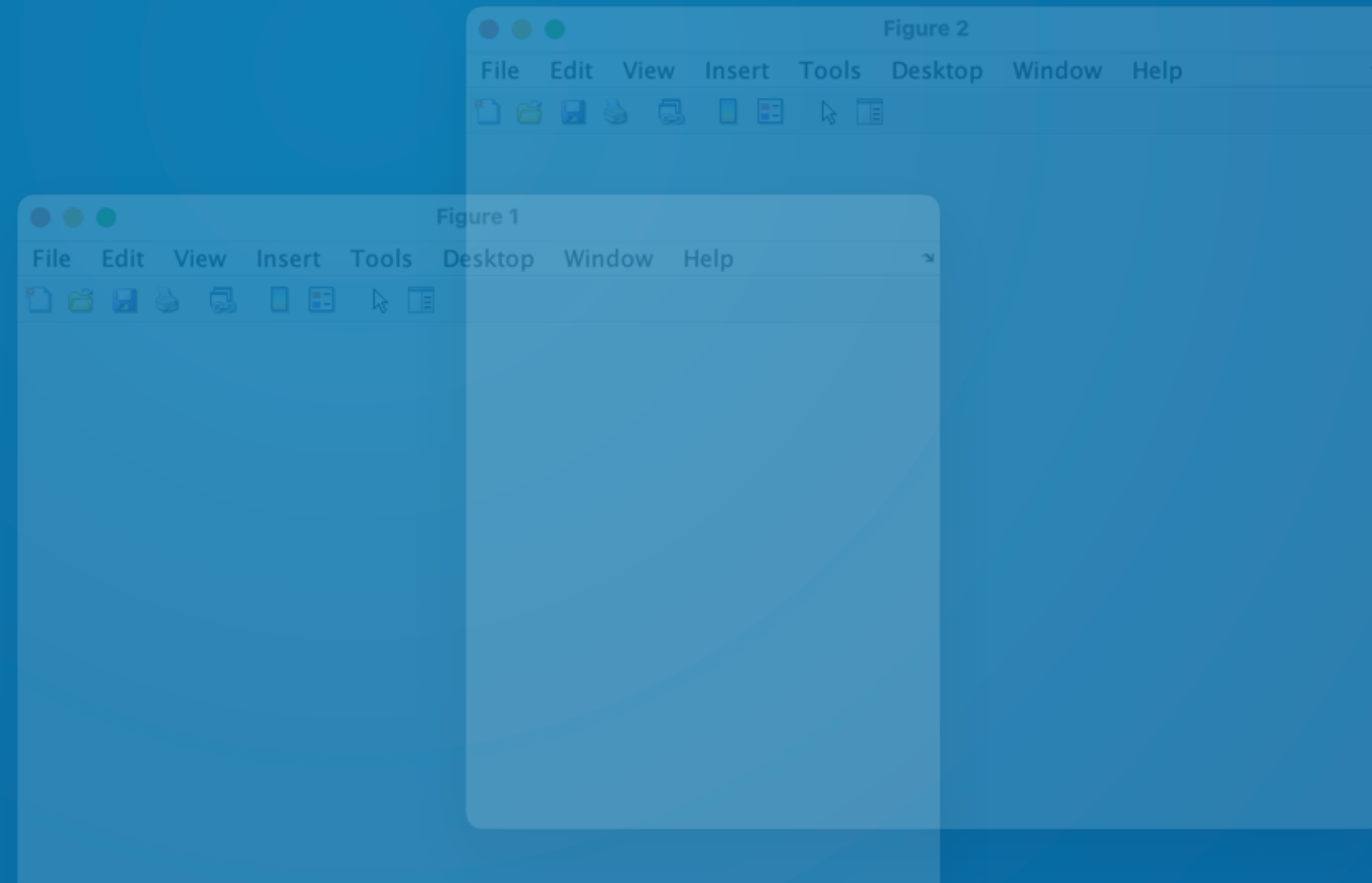


figure([int])

Genera una figura vacía en la que podemos representar gráficos, imágenes, etc.

EJEMPLO DE USO:

```
figure(1);
```



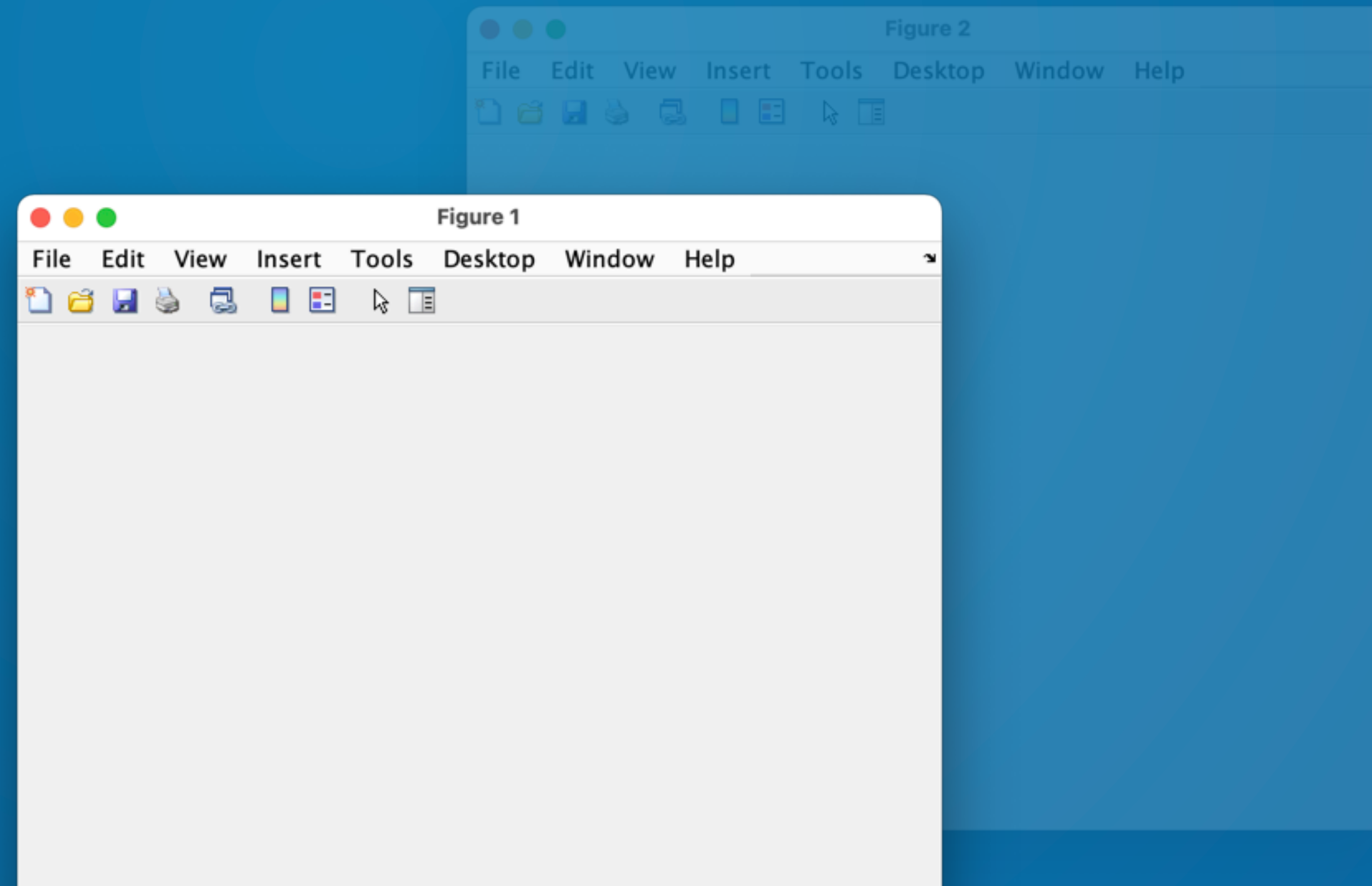
figure([int])



Genera una figura vacía en la que podemos representar gráficos, imágenes, etc.

EJEMPLO DE USO:

```
figure(1);
```



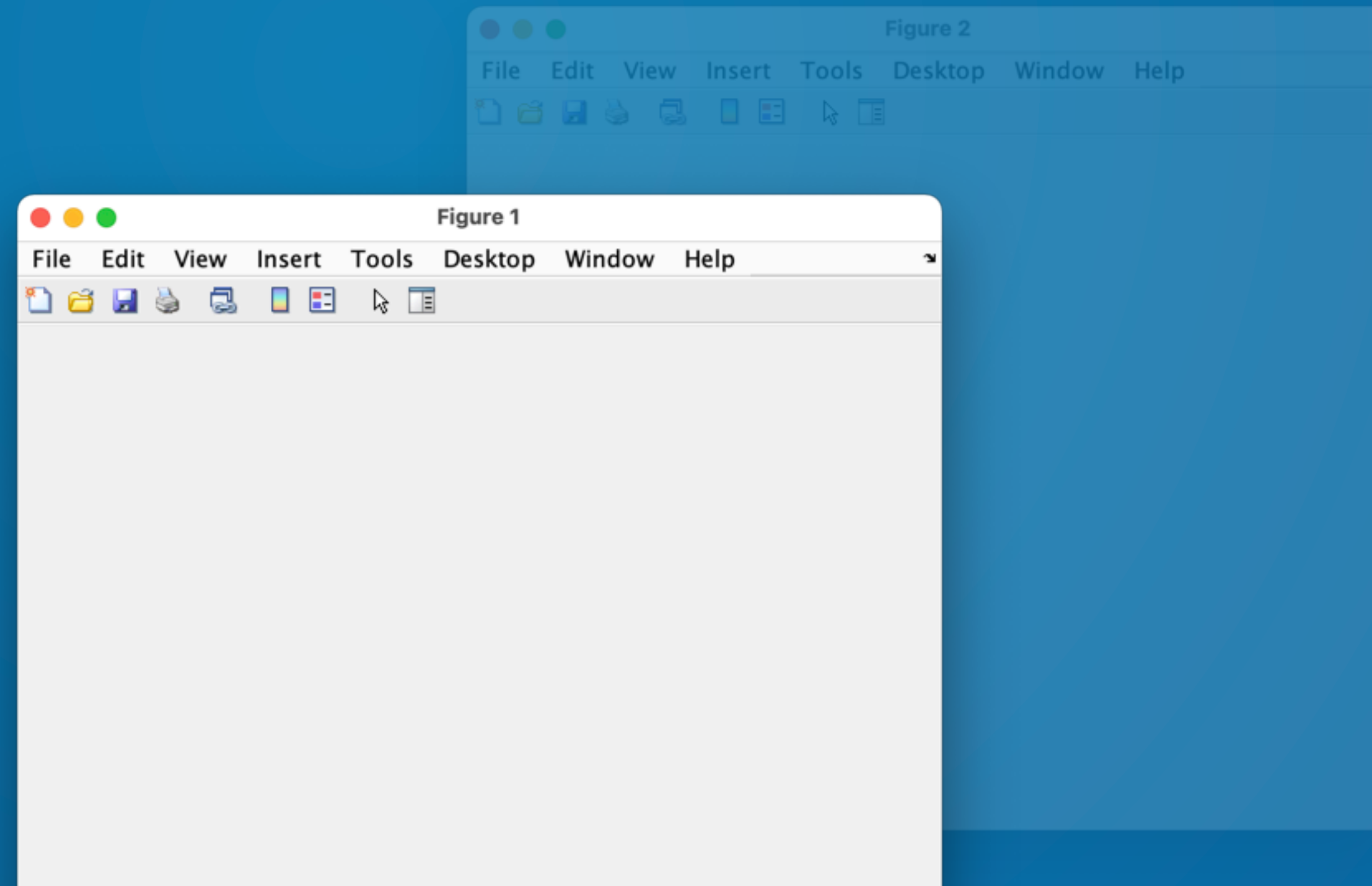
figure([int])



Genera una figura vacía en la que podemos representar gráficos, imágenes, etc.

EJEMPLO DE USO:

```
figure(1);  
figure(2);
```

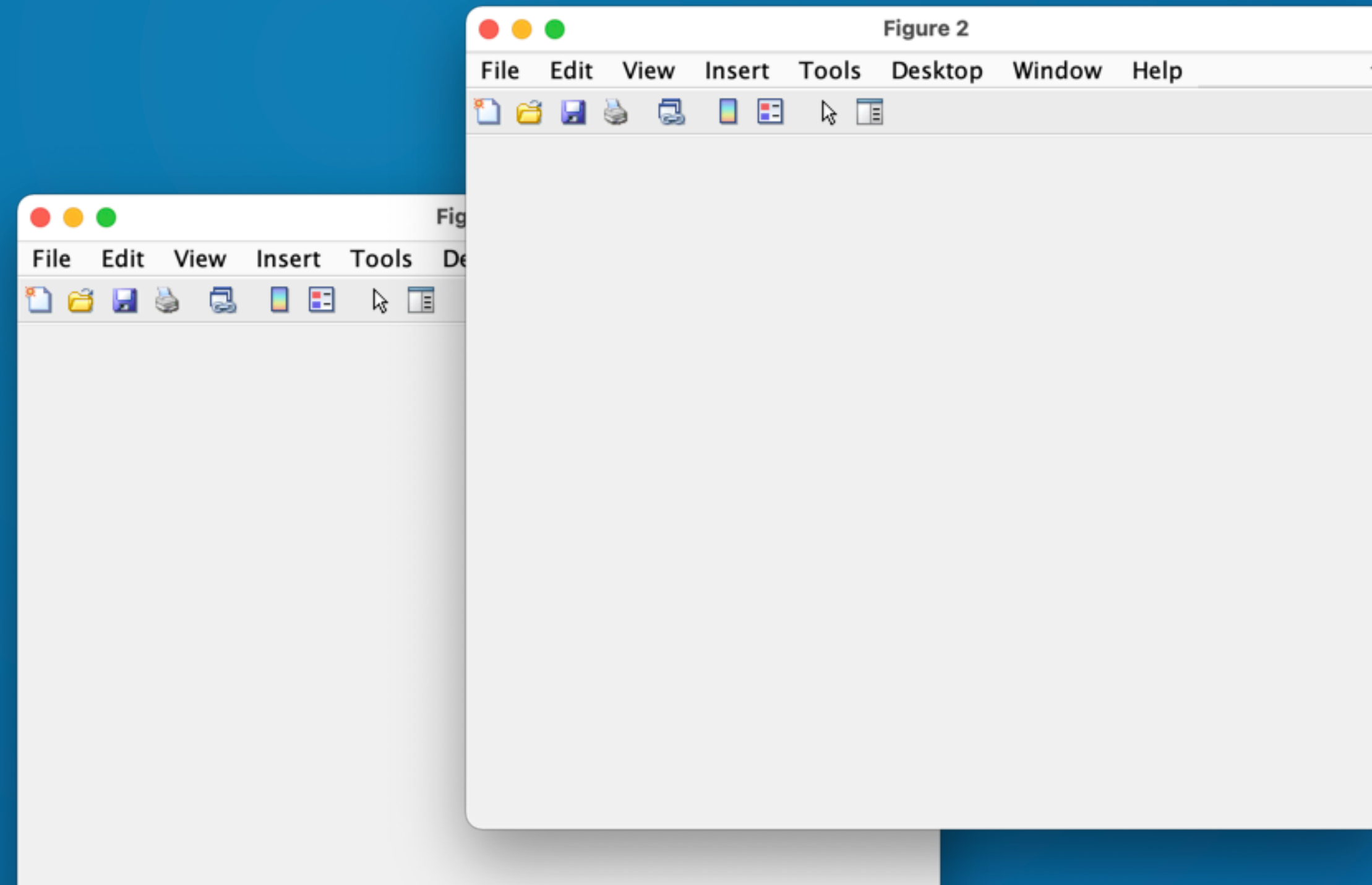


figure([int])

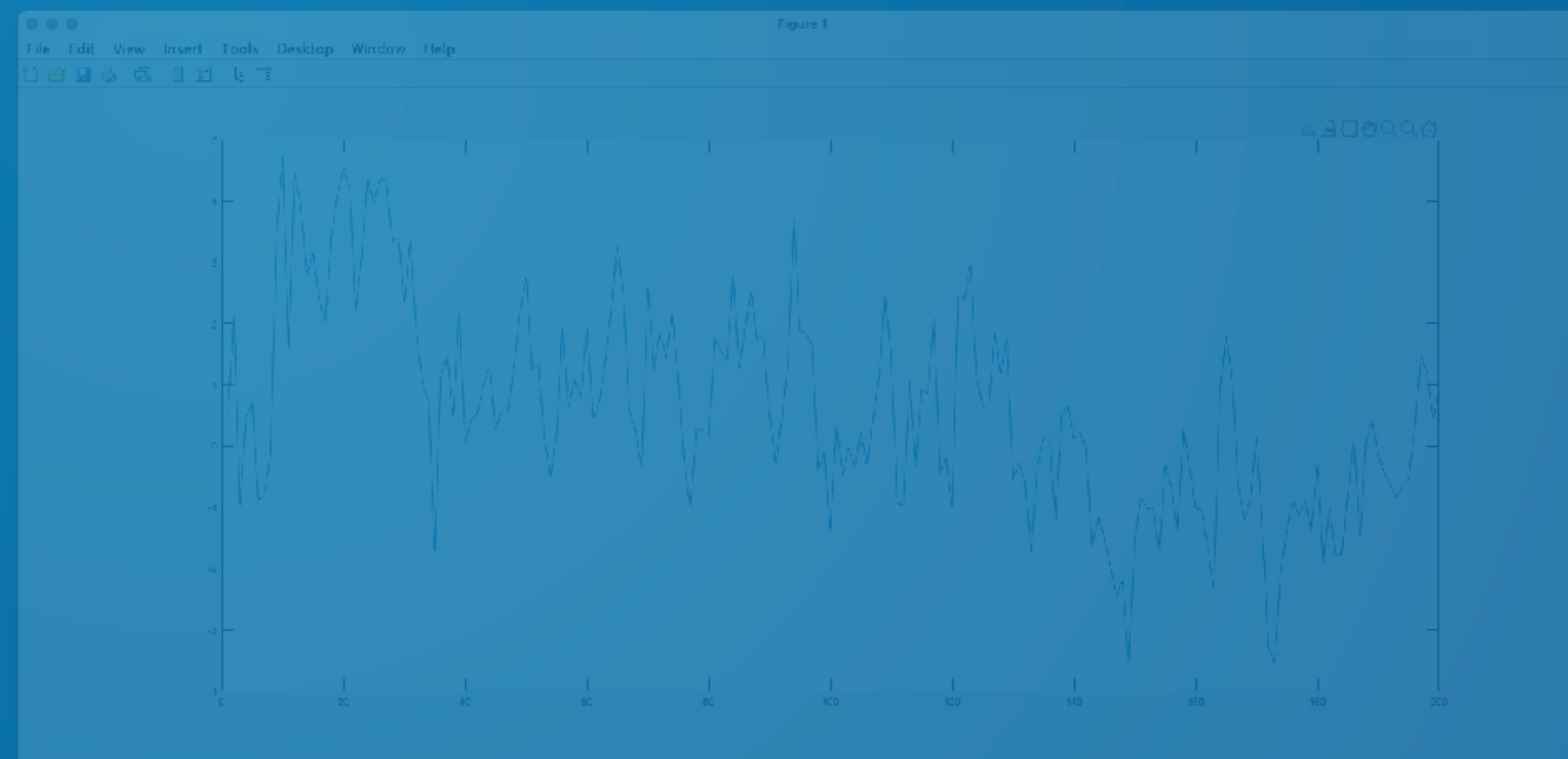
Genera una figura vacía en la que podemos representar gráficos, imágenes, etc.

EJEMPLO DE USO:

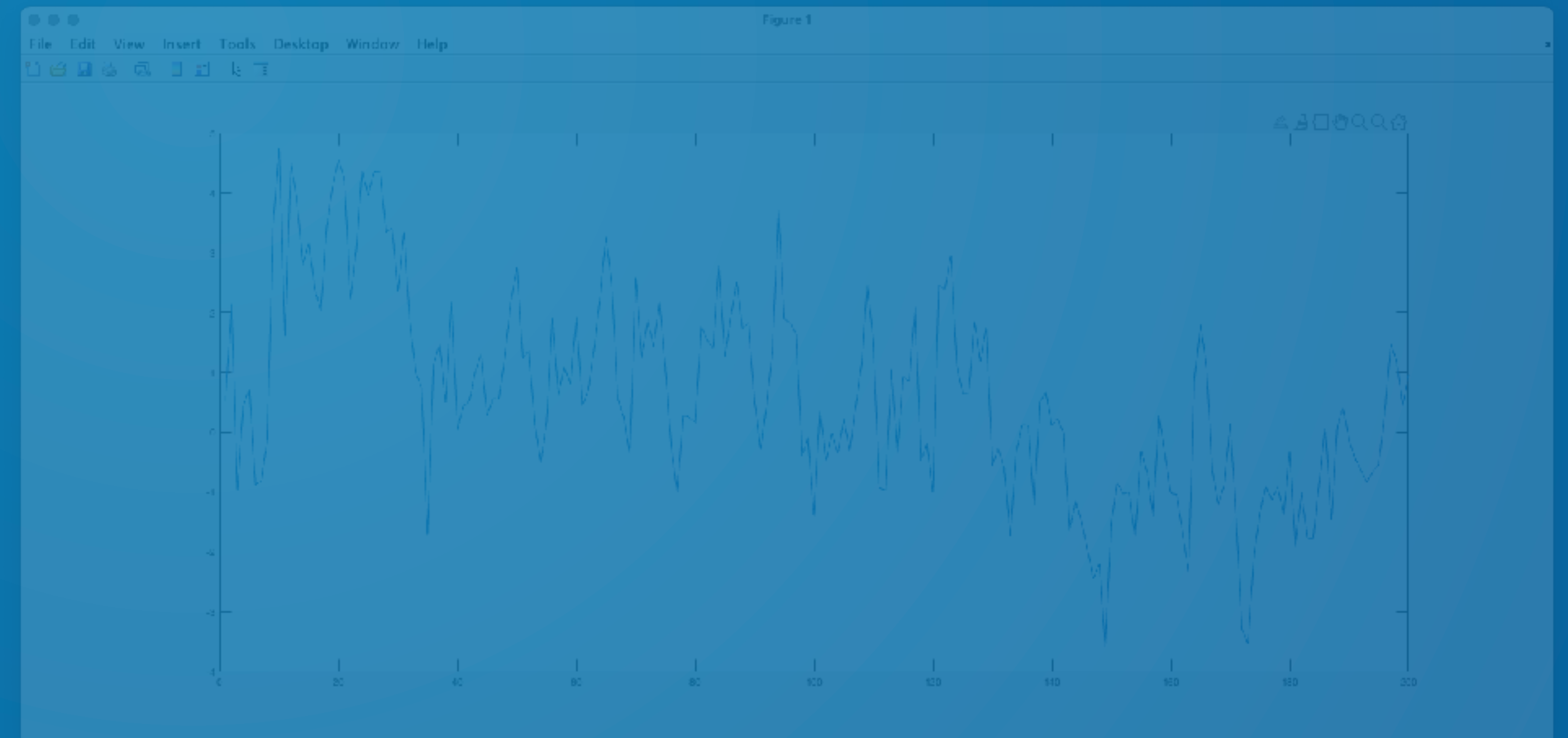
```
figure(1);  
figure(2);
```



► Funciones propias de Matlab ► Funciones para gráficos

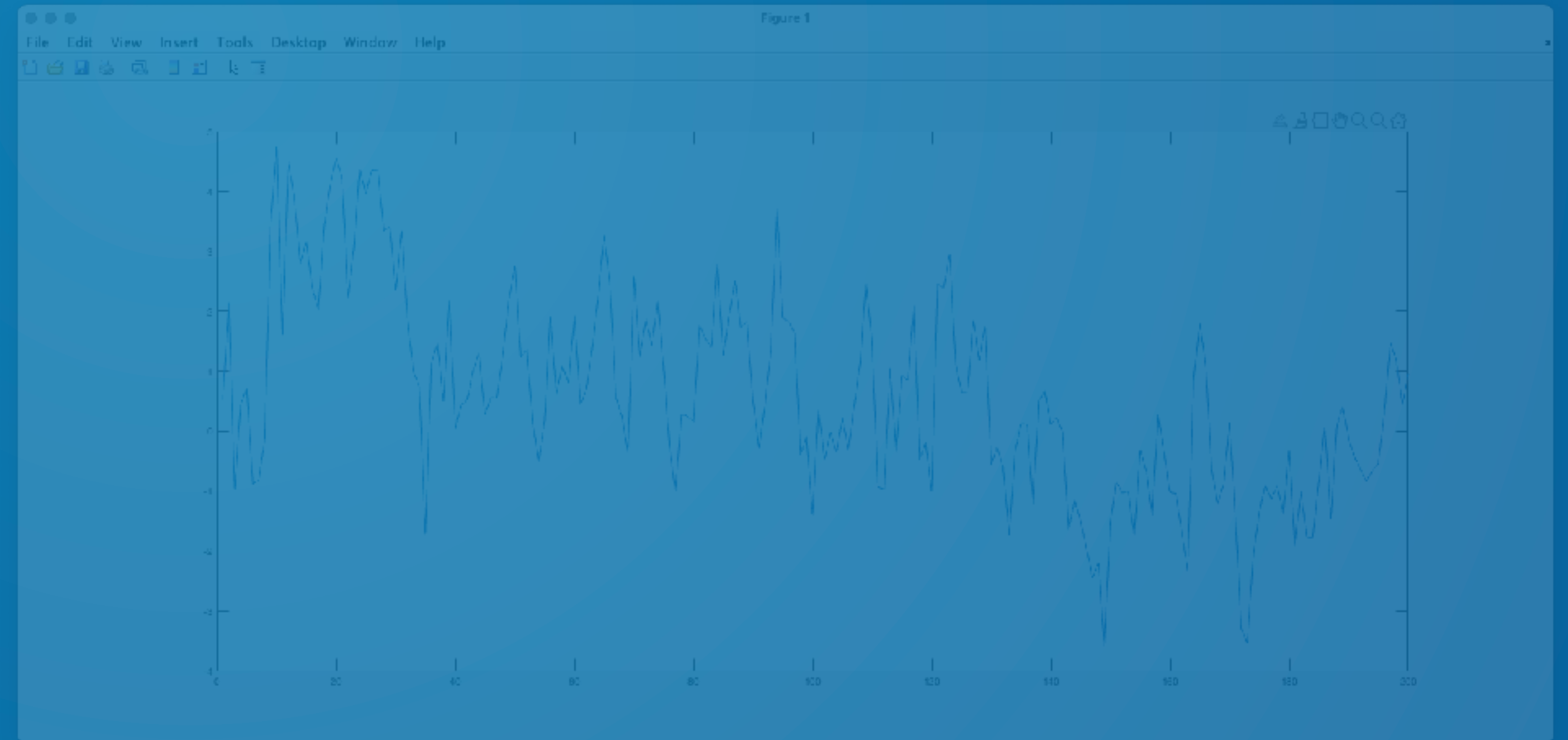


plot([var]) →



plot([var]) →

Genera un gráfico de líneas a partir de los datos contenidos en [var]

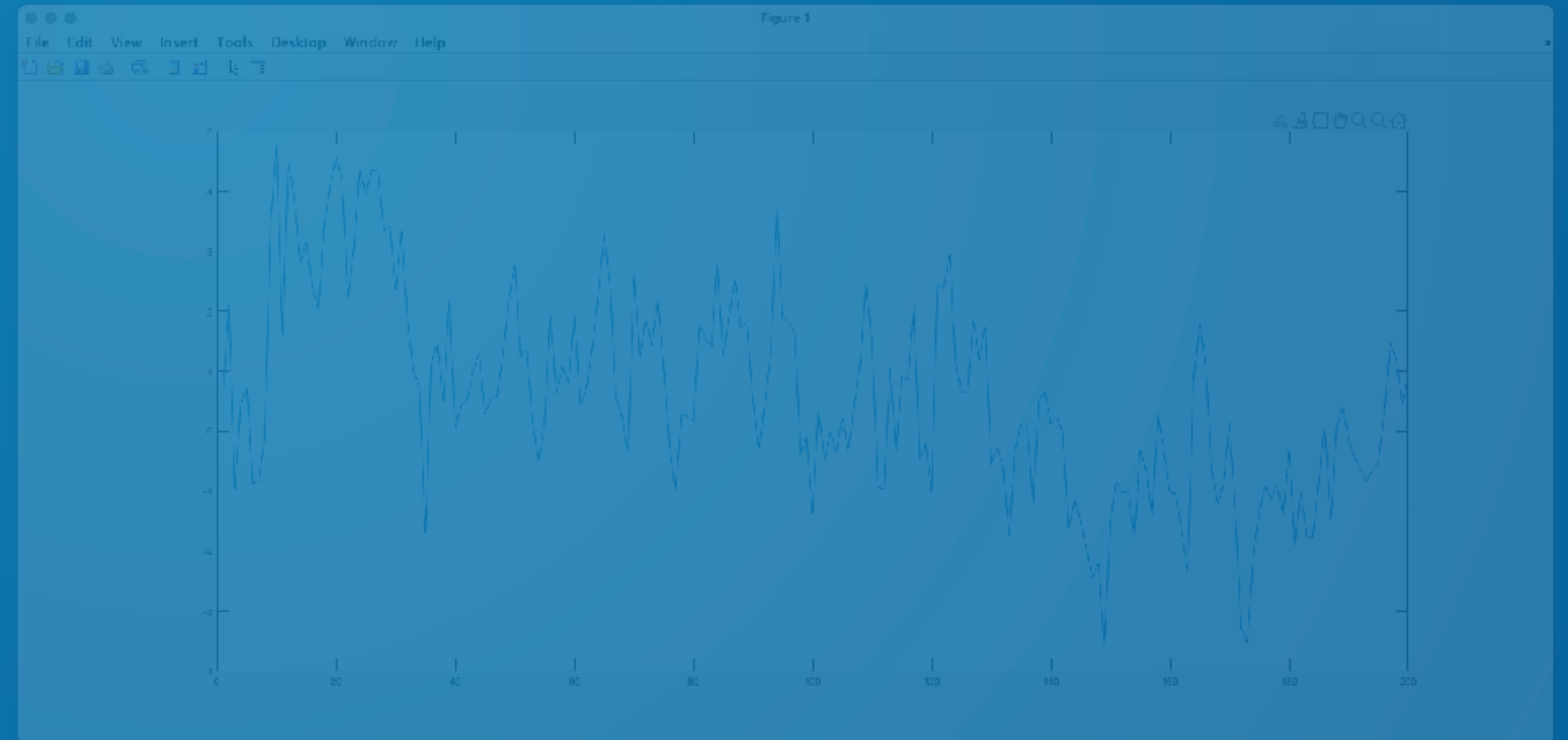


plot([var]) →

Genera un gráfico de líneas a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
eeg = [0.53, 2.15, ...
```

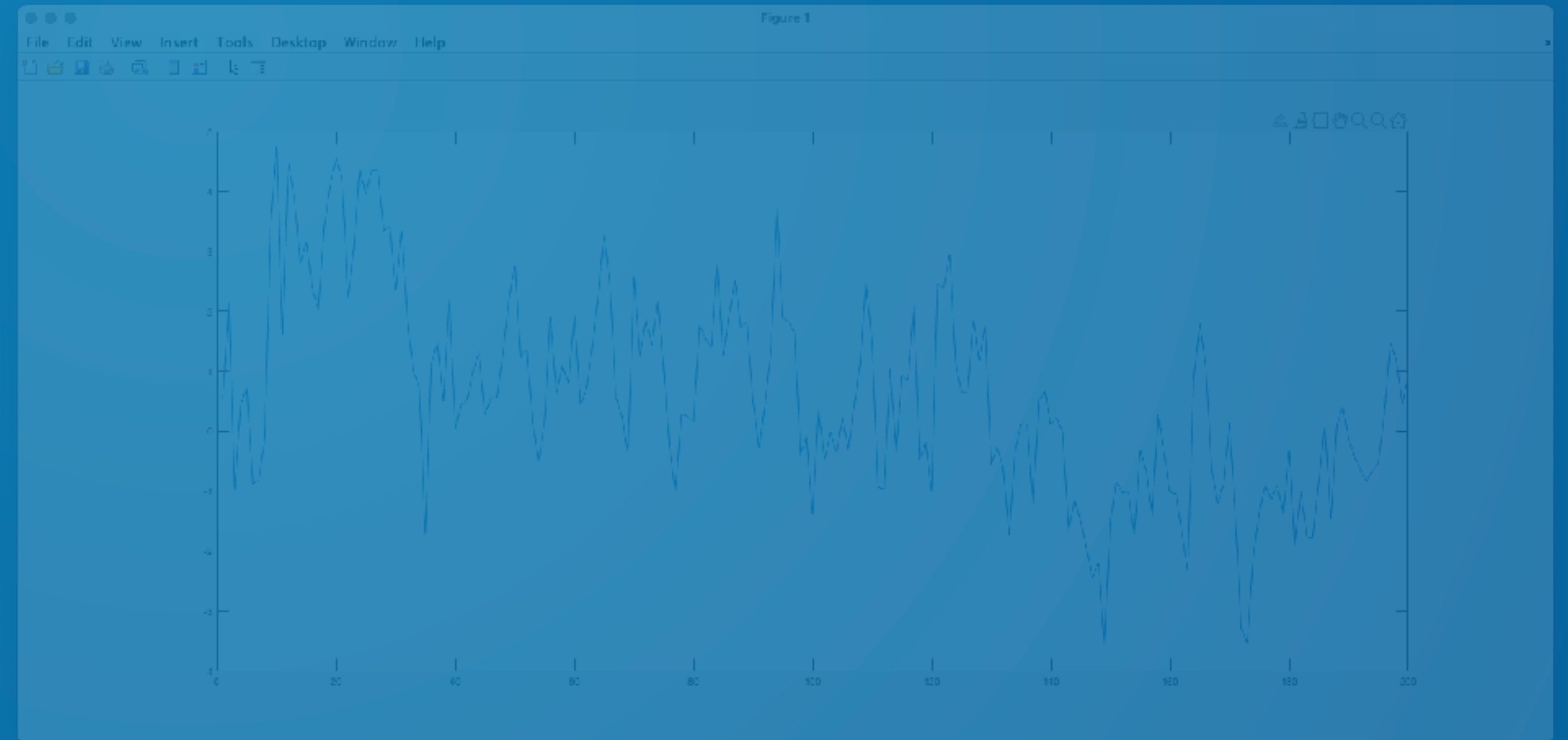


plot([var]) →

Genera un gráfico de líneas a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
eeg = [0.53, 2.15, ...  
plot(eeg);
```

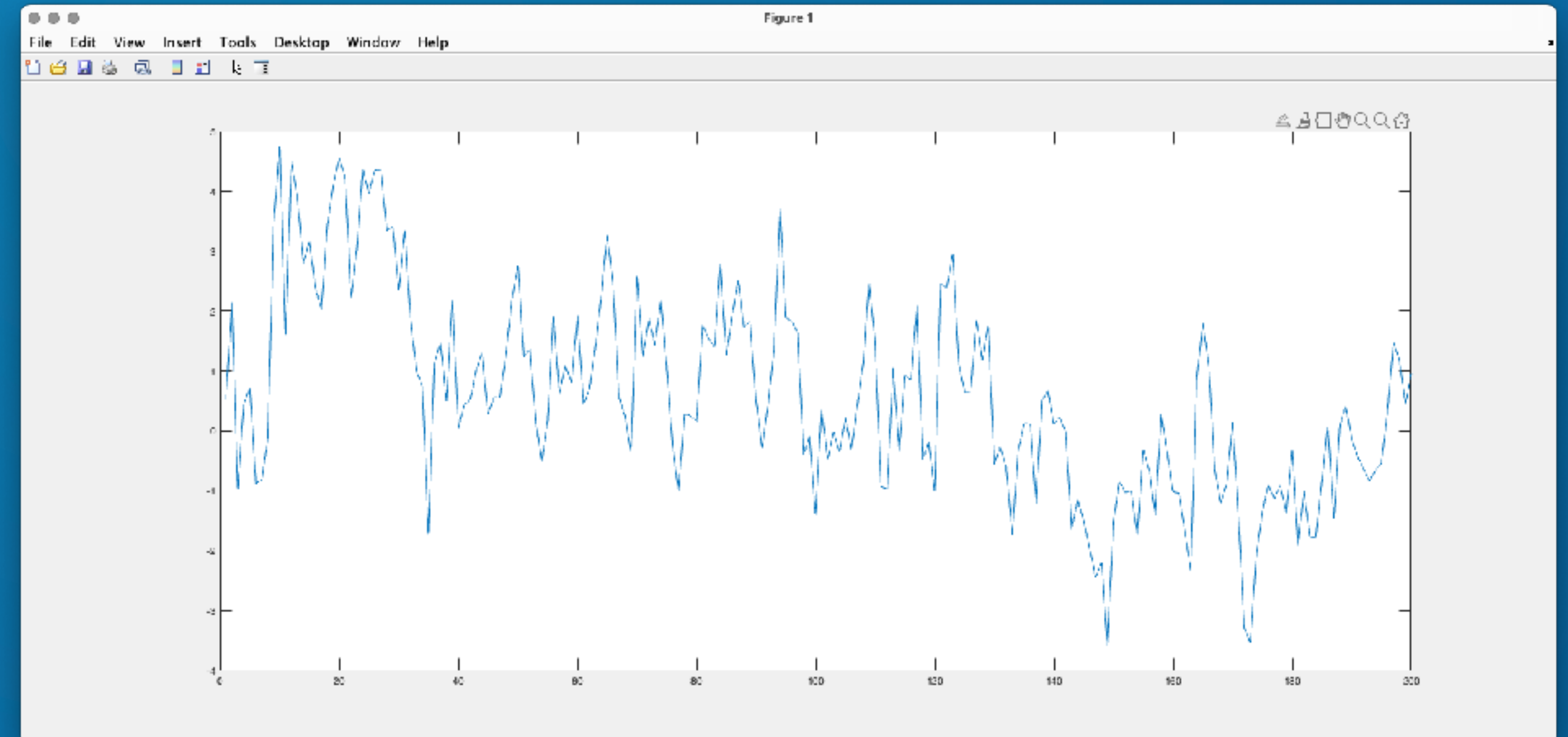


plot([var])

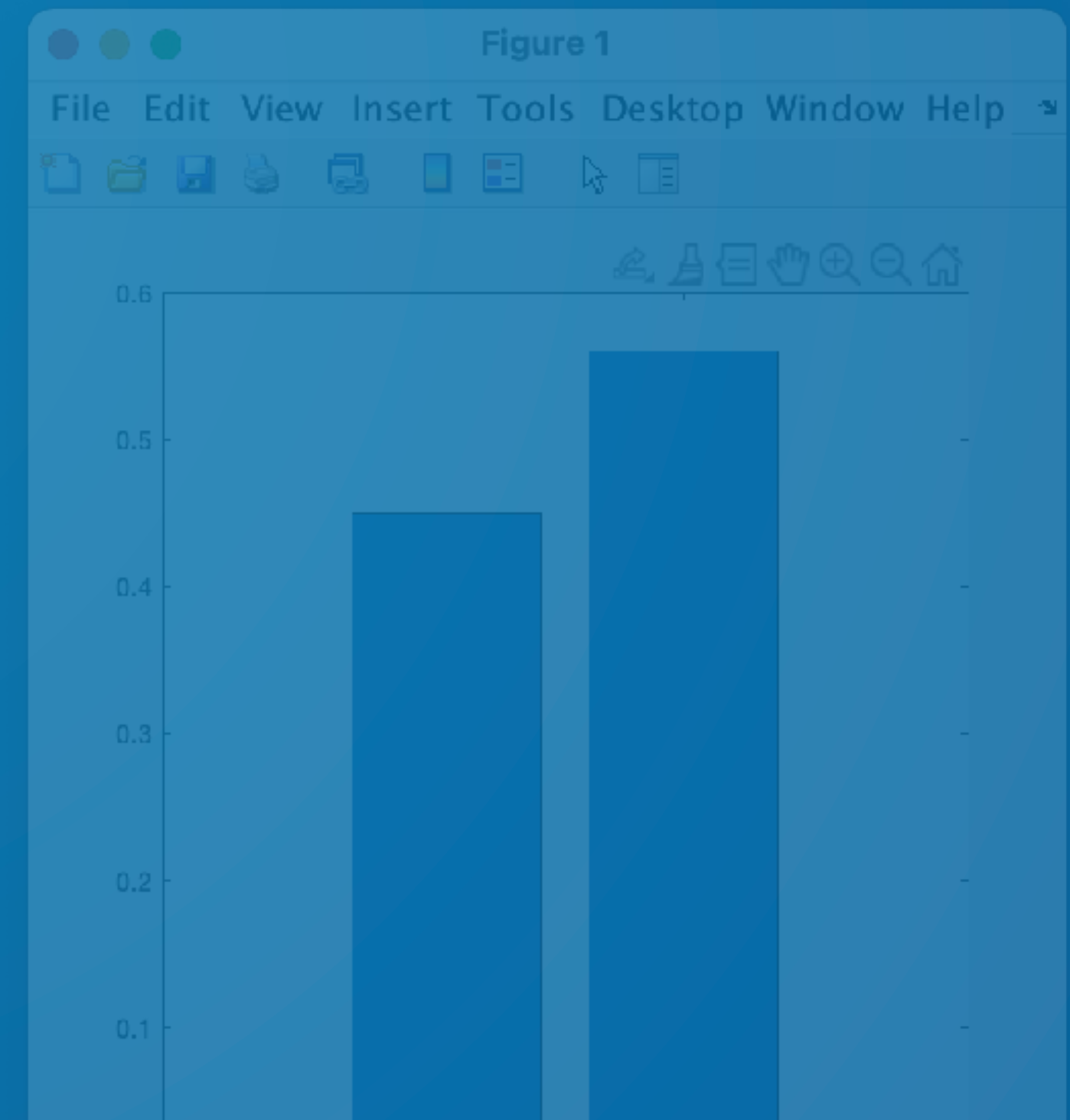
Genera un gráfico de líneas a partir de los datos contenidos en [var]

EJEMPLO DE USO:

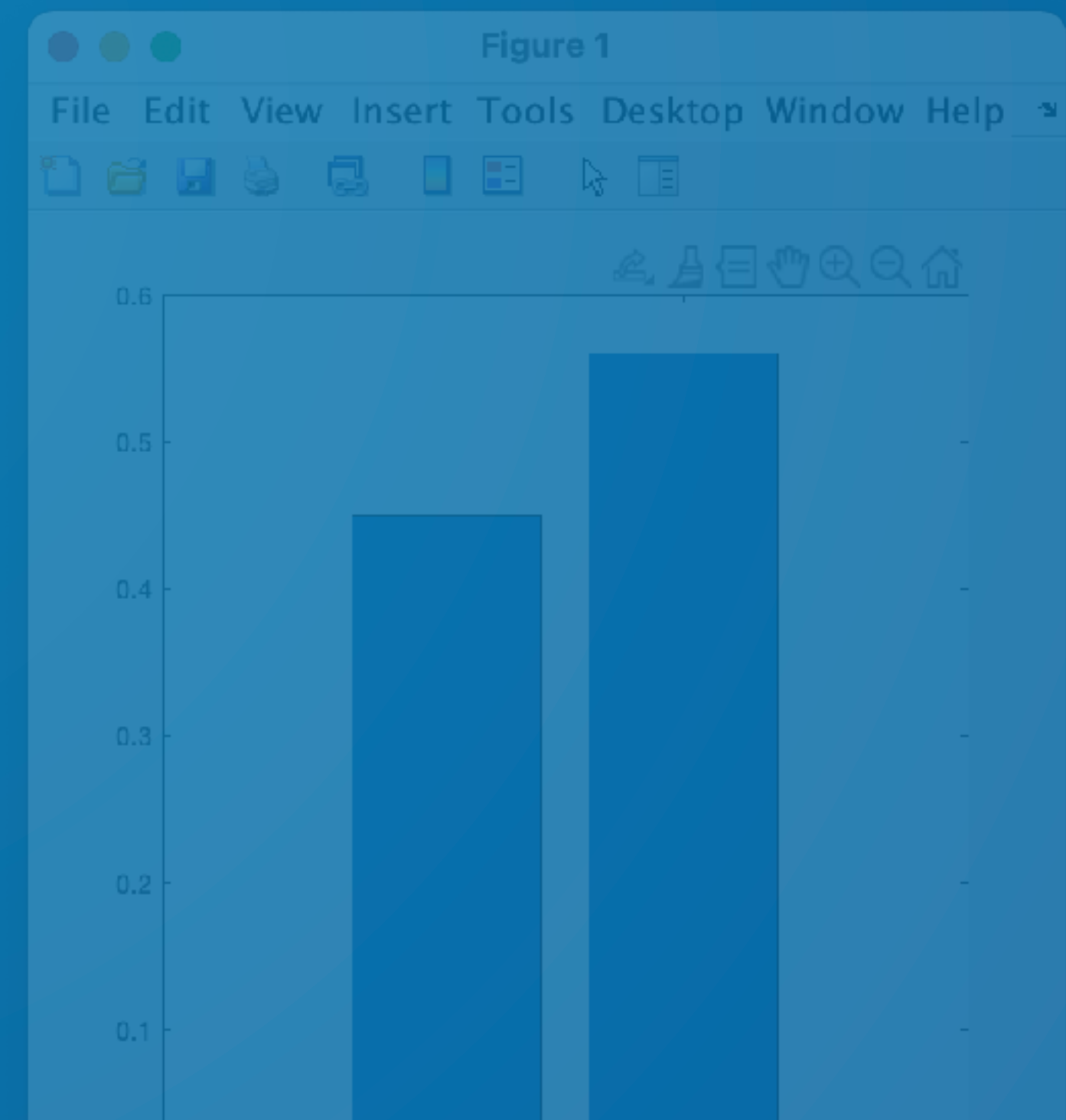
```
eeg = [0.53, 2.15, ...  
plot(eeg);
```



► Funciones propias de Matlab ► Funciones para gráficos

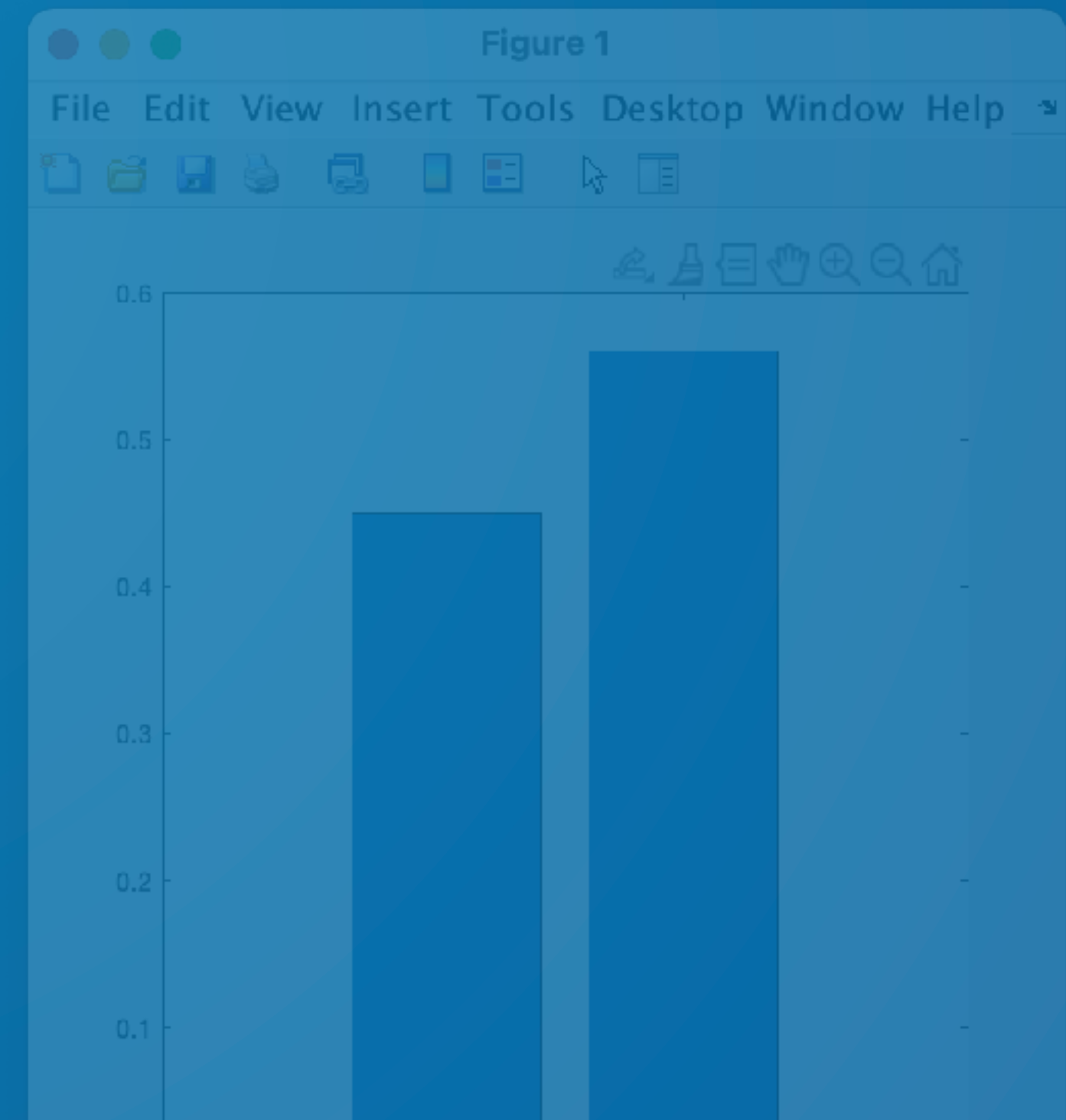


bar([var]) →



bar([var]) →

Genera un gráfico de barras a partir de los datos contenidos en [var]

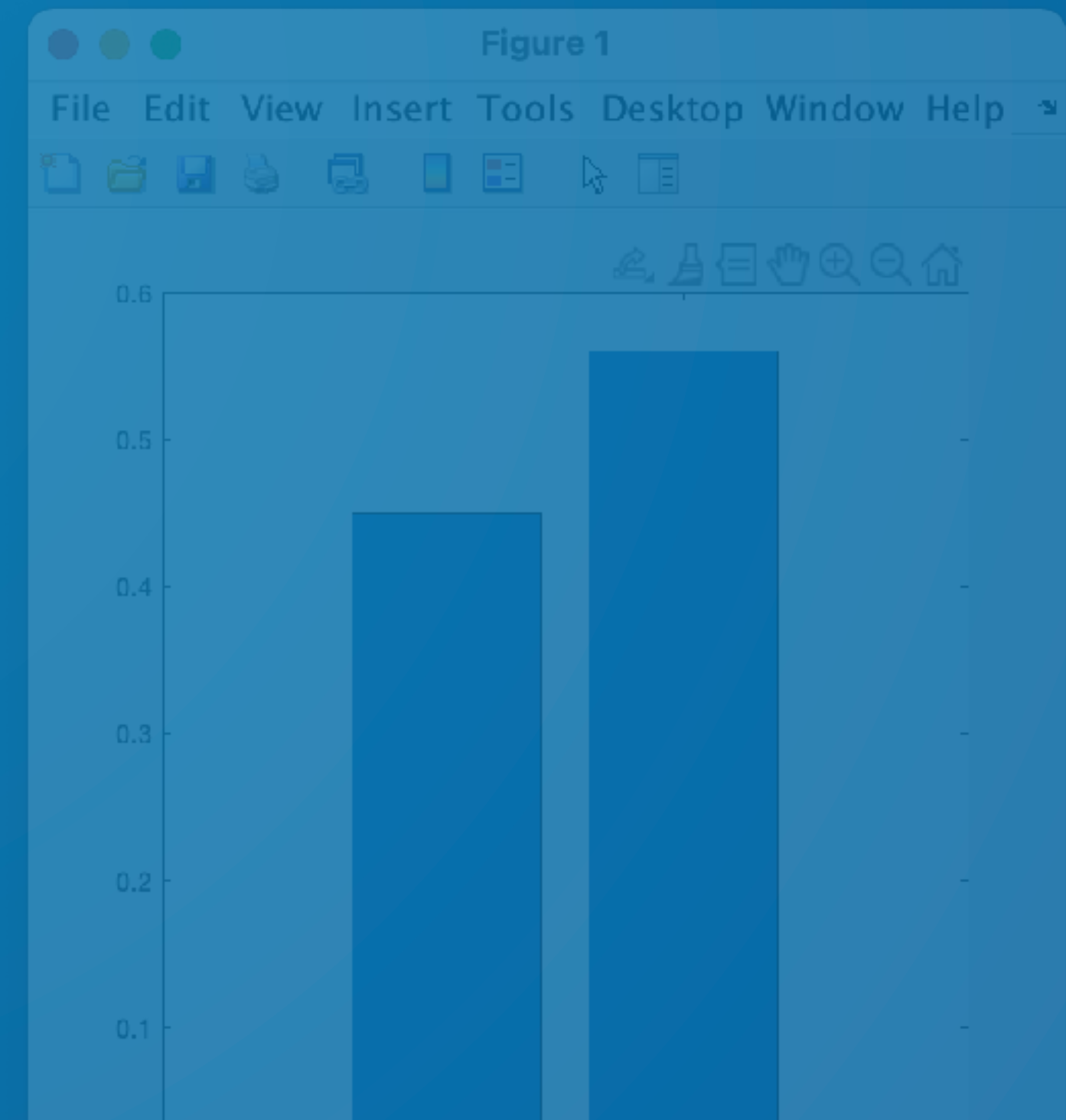


bar([var]) →

Genera un gráfico de barras a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
reactionTimes = [0.45, 0.56];
```

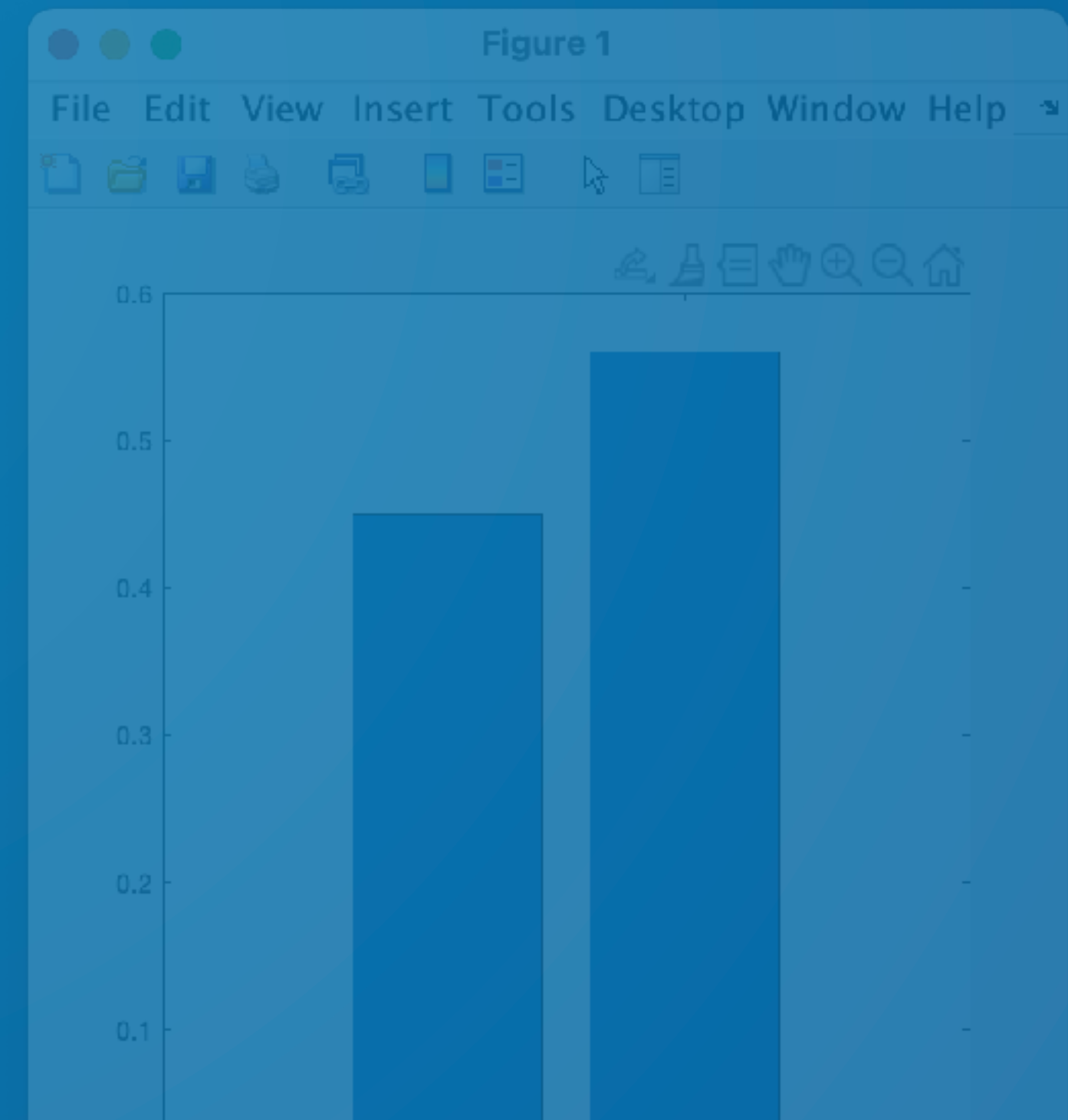


bar([var]) →

Genera un gráfico de barras a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
reactionTimes = [0.45, 0.56];  
bar(reactionTimes);
```

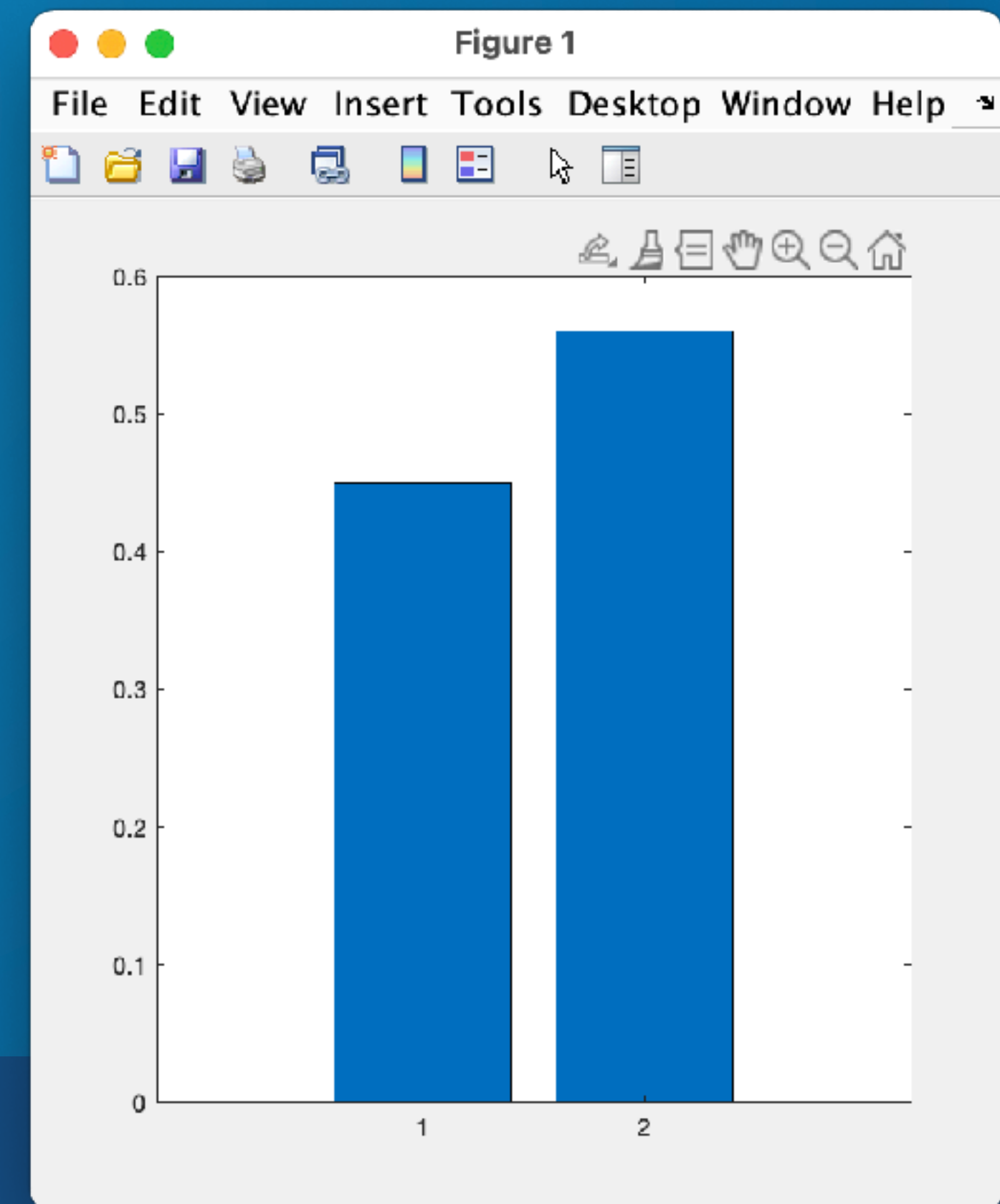


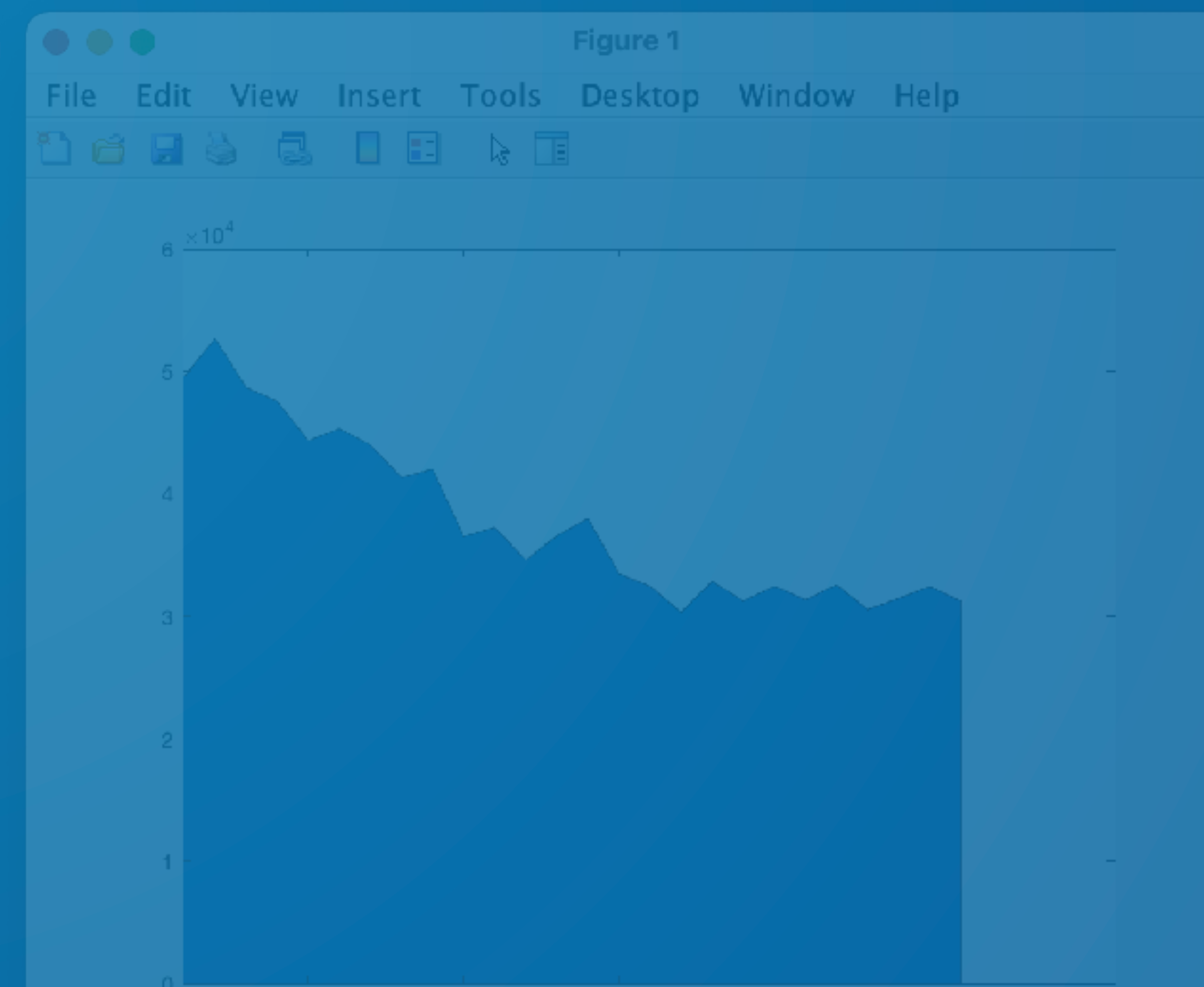
bar([var])

Genera un gráfico de barras a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
reactionTimes = [0.45, 0.56];  
bar(reactionTimes);
```





area([var]) →



area([var]) →

Genera un gráfico de área a partir de los datos contenidos en [var]



area([var]) →

Genera un gráfico de área a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
bitcoin = [49567, 52657, ...
```



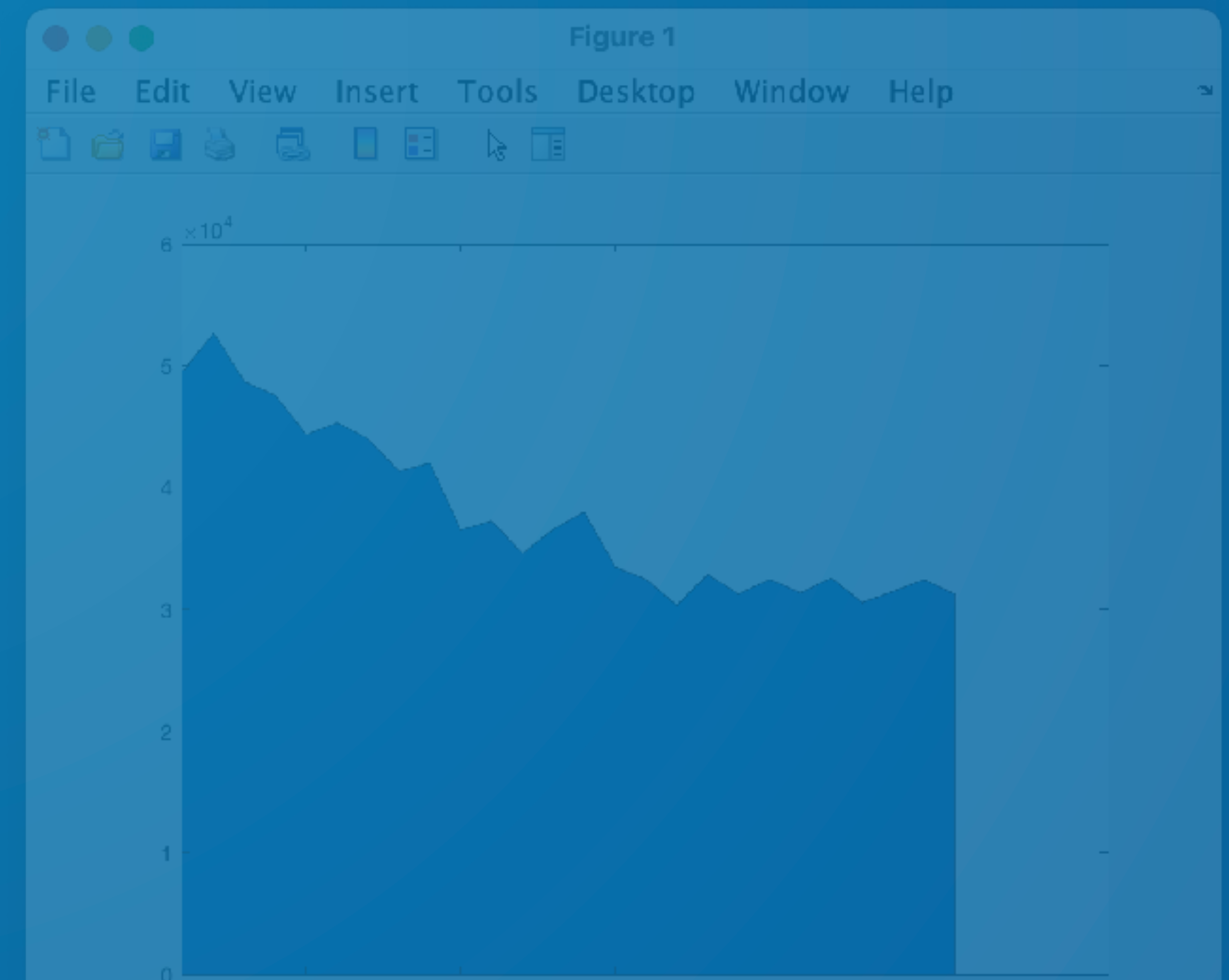
area([var]) →

Genera un gráfico de área a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
bitcoin = [49567, 52657, ...
```

```
area(values);
```

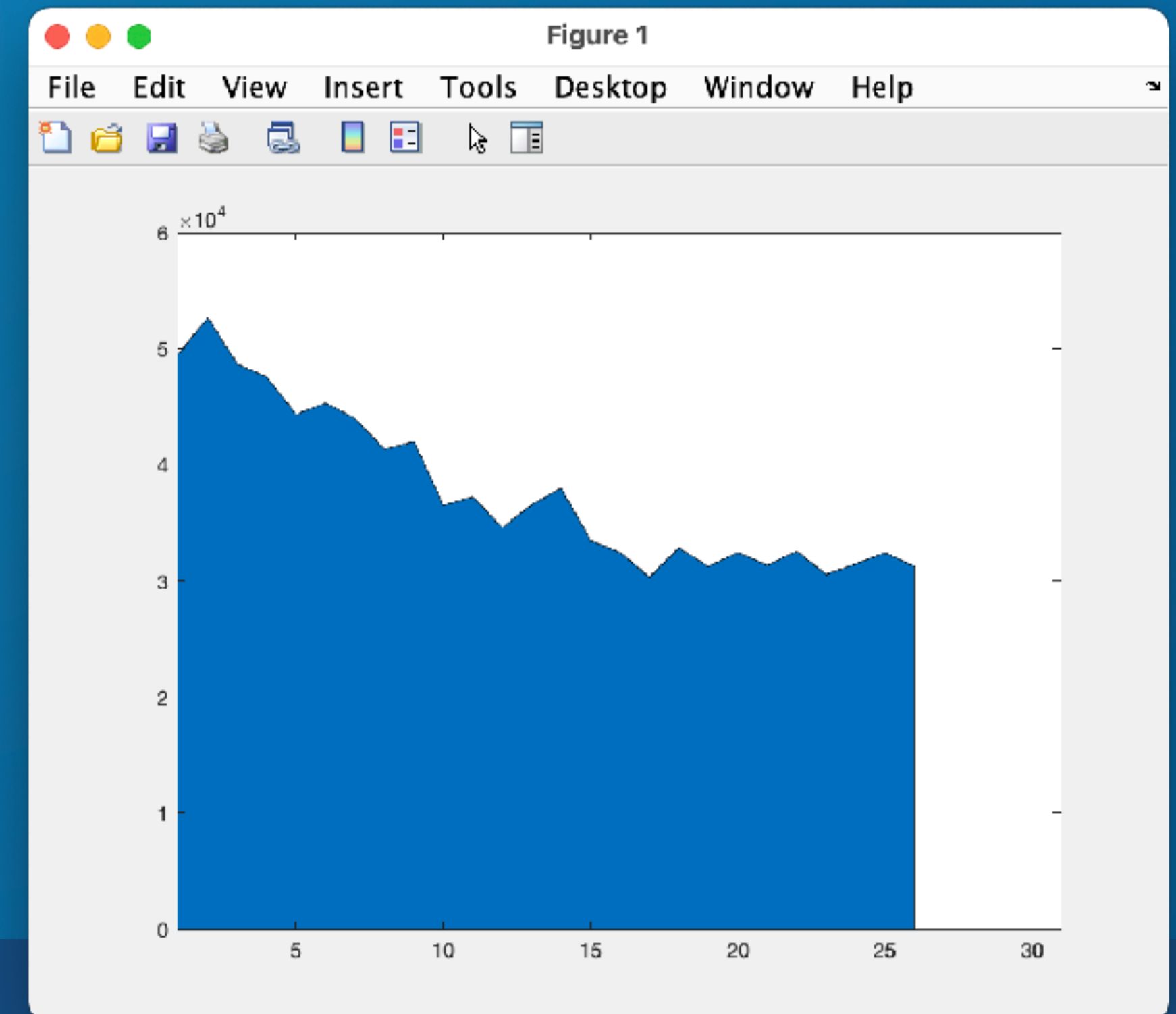


area([var]) →

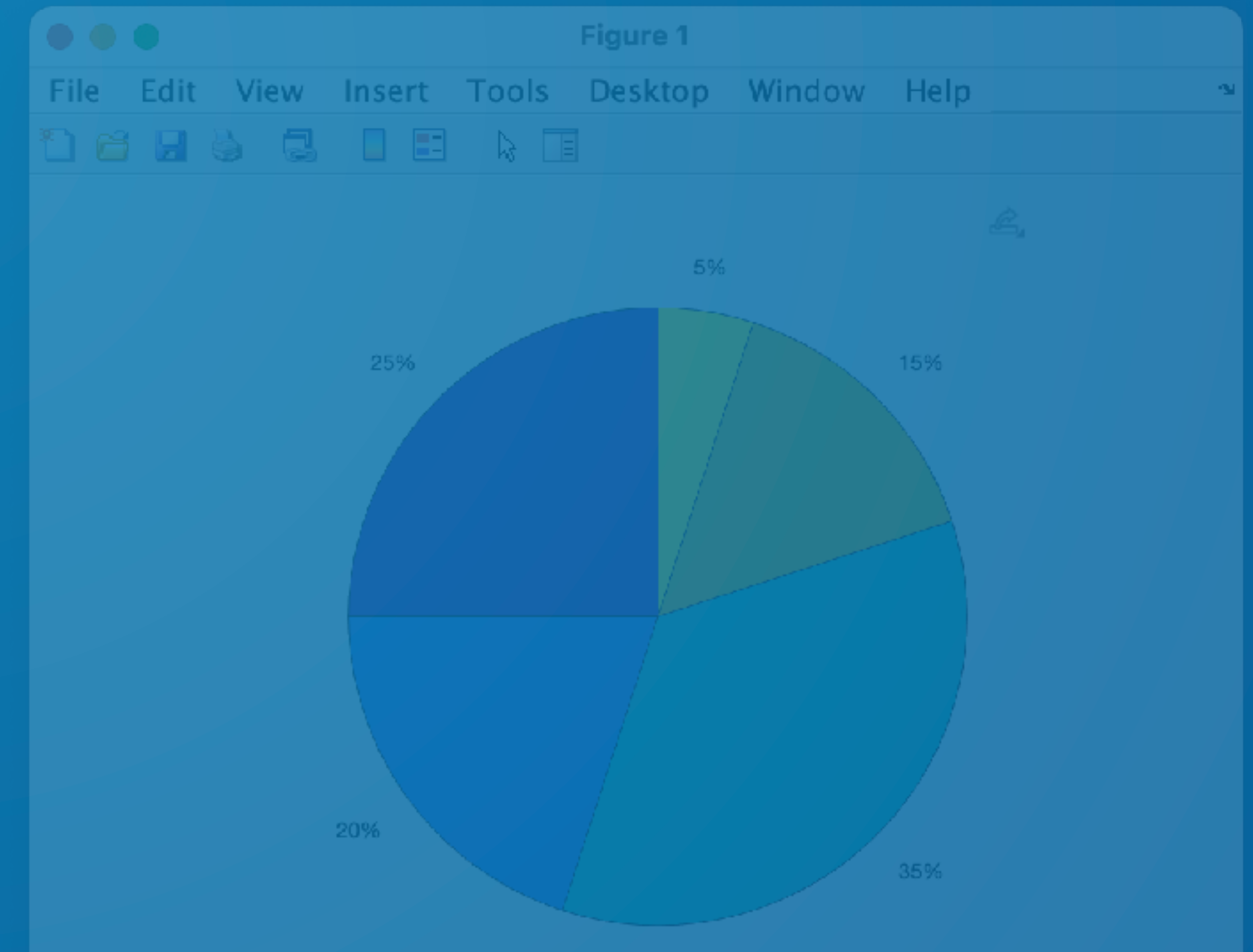
Genera un gráfico de área a partir de los datos contenidos en [var]

EJEMPLO DE USO:

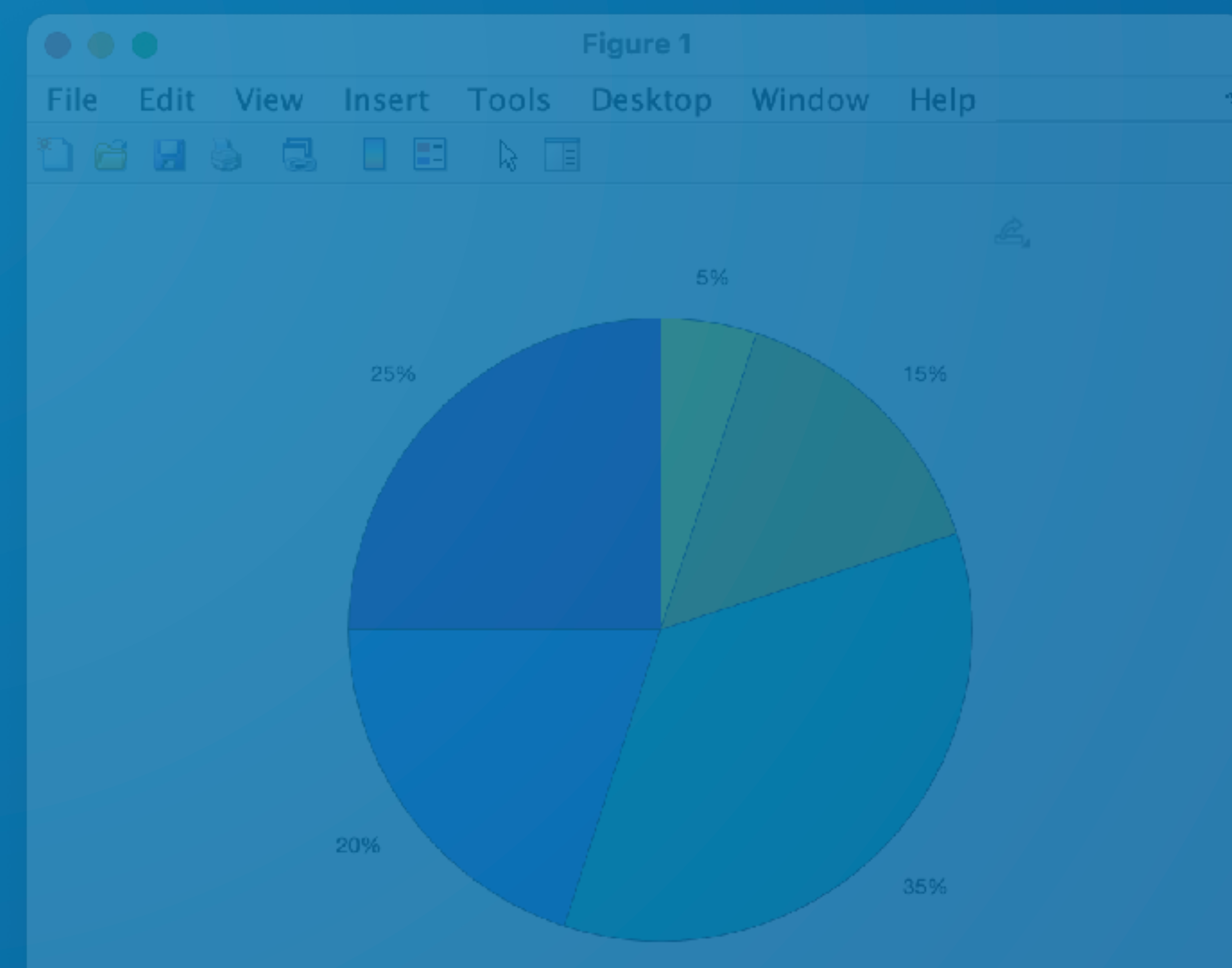
```
bitcoin = [49567, 52657, ...  
area(values);
```



► Funciones propias de Matlab ► Funciones para gráficos

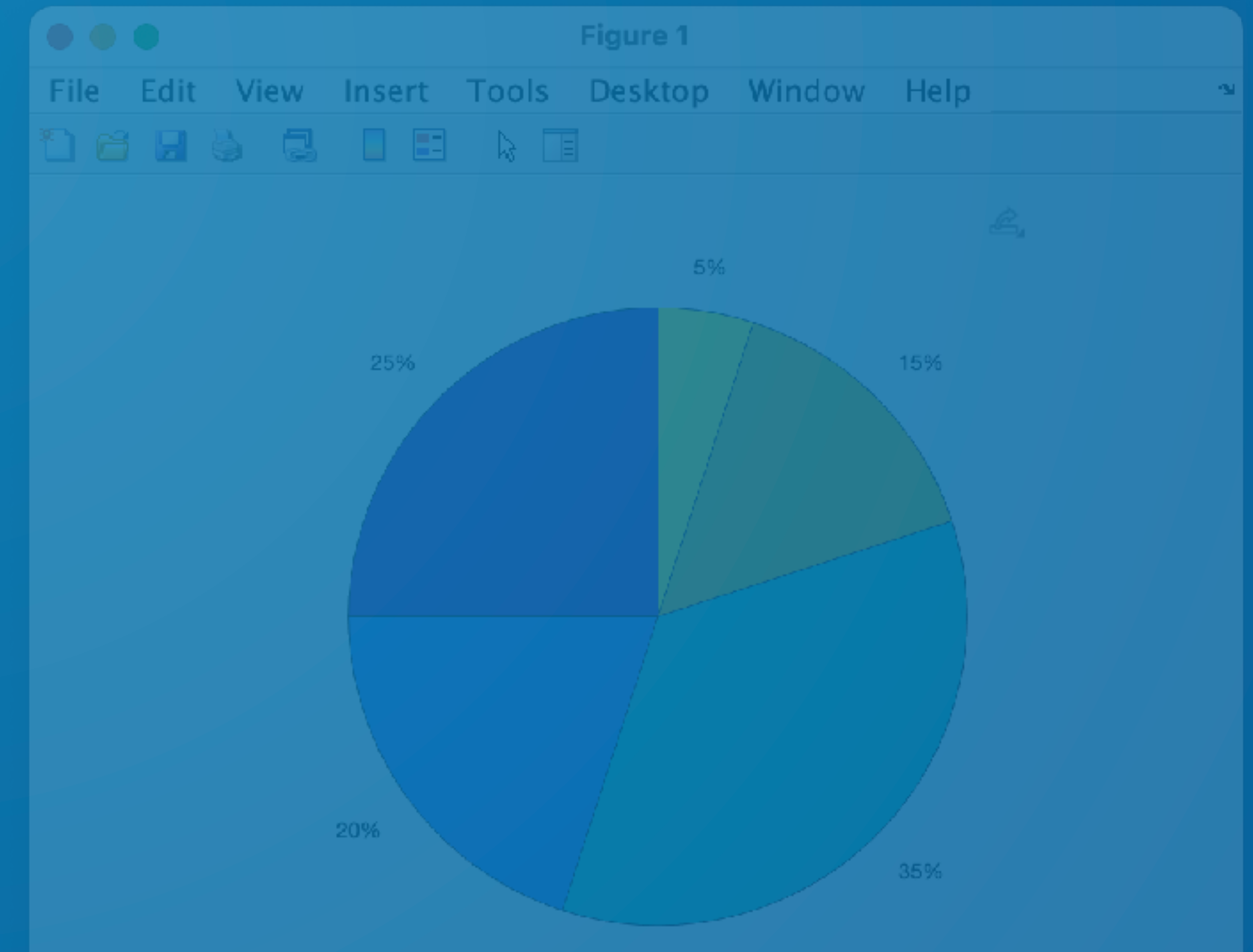


pie([var]) →



pie([var]) →

Genera un gráfico de porcentajes a partir de los datos contenidos en [var]

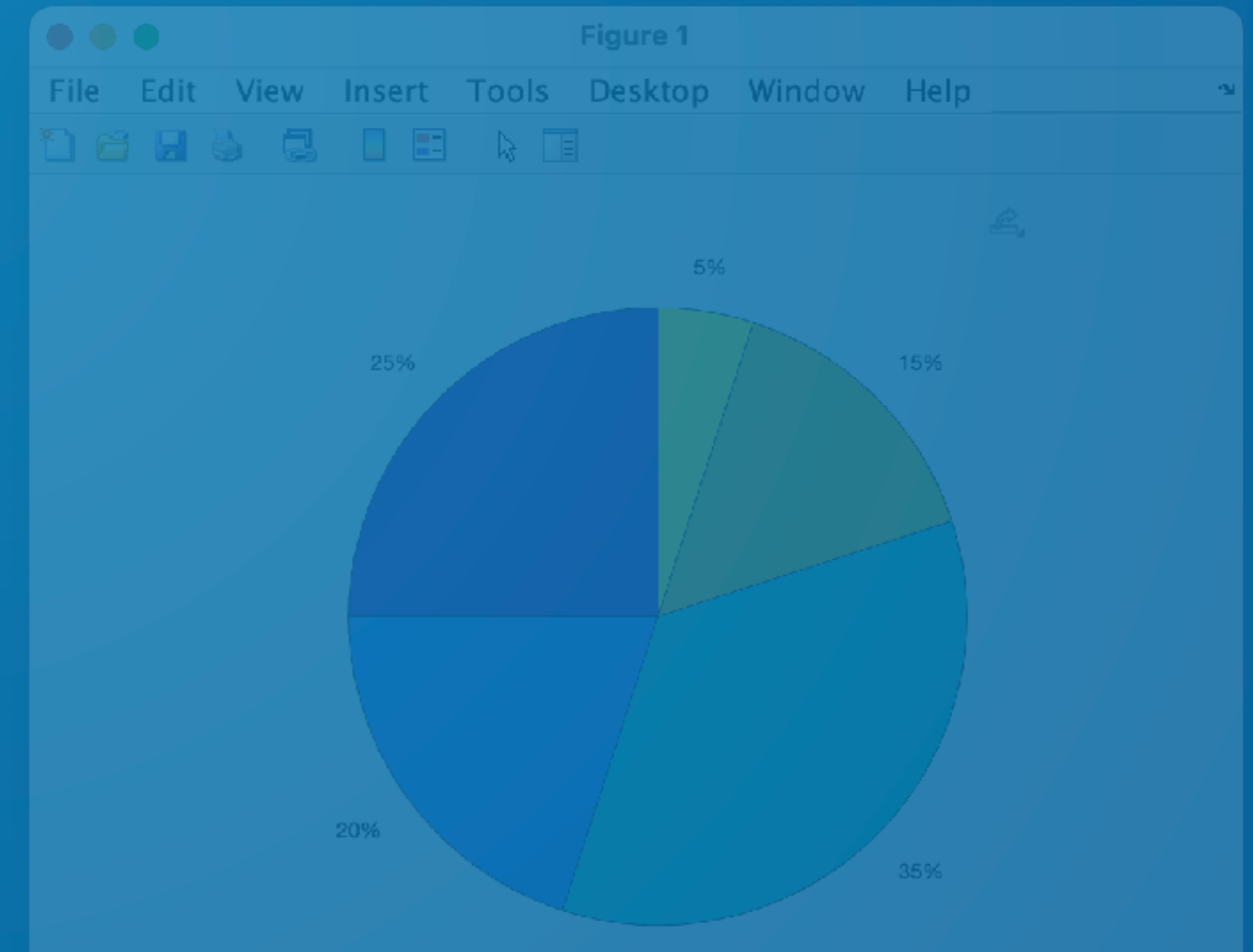


pie([var]) →

Genera un gráfico de porcentajes a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
marks = [25,20,35,15,5];
```



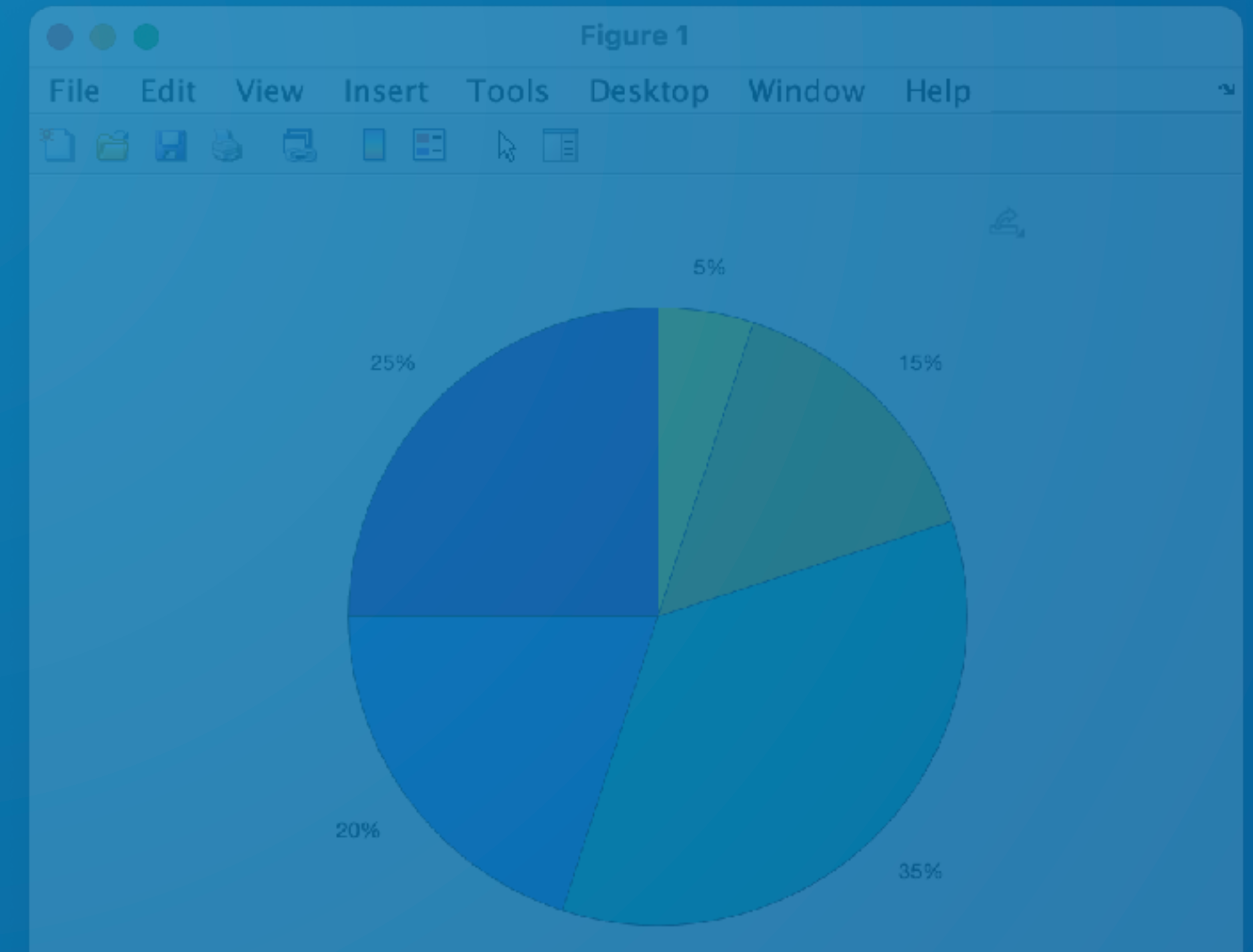
pie([var]) →

Genera un gráfico de porcentajes a partir de los datos contenidos en [var]

EJEMPLO DE USO:

```
marks = [25,20,35,15,5];
```

```
pie(marks);
```



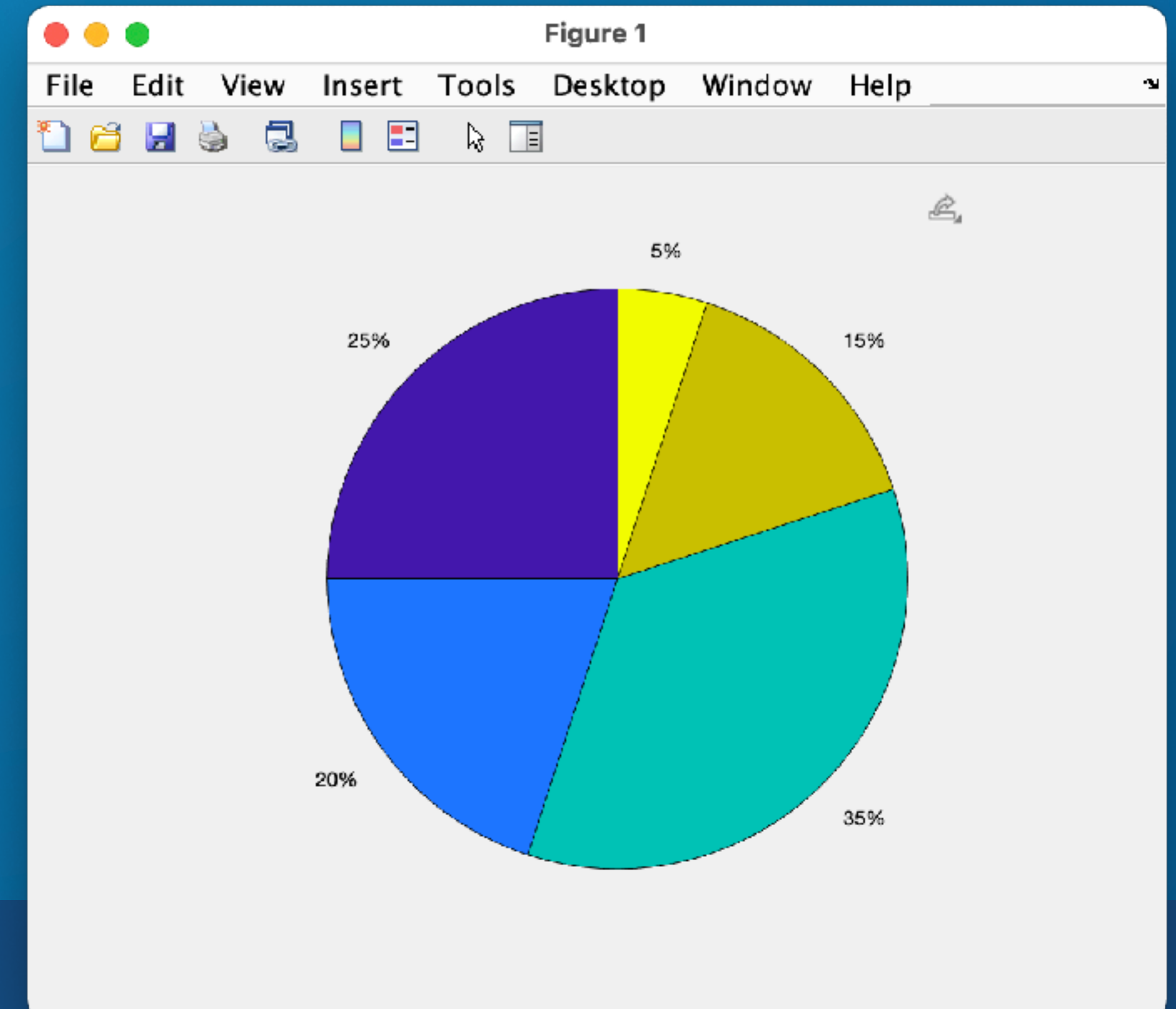
pie([var]) →

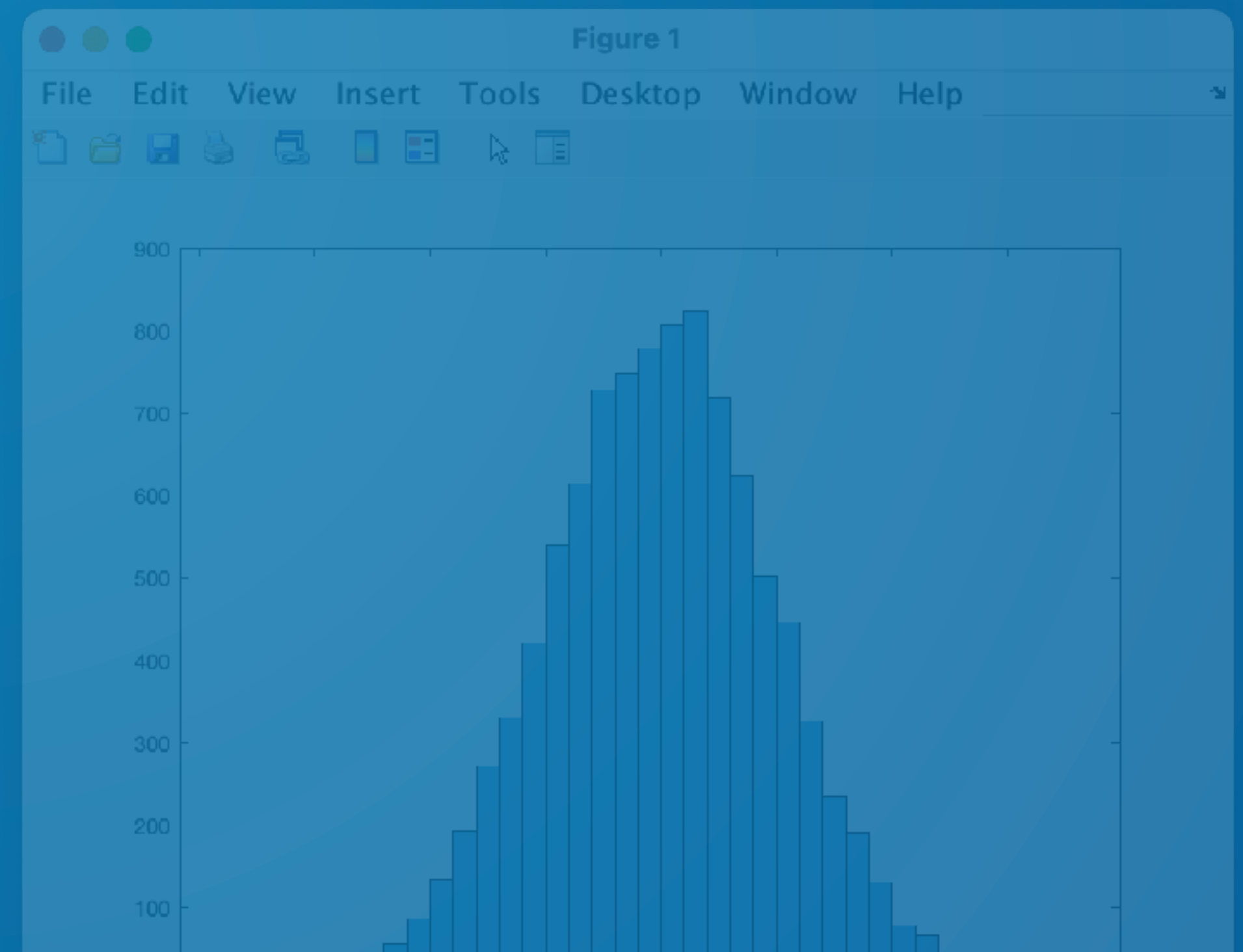
Genera un gráfico de porcentajes a partir de los datos contenidos en [var]

EJEMPLO DE USO:

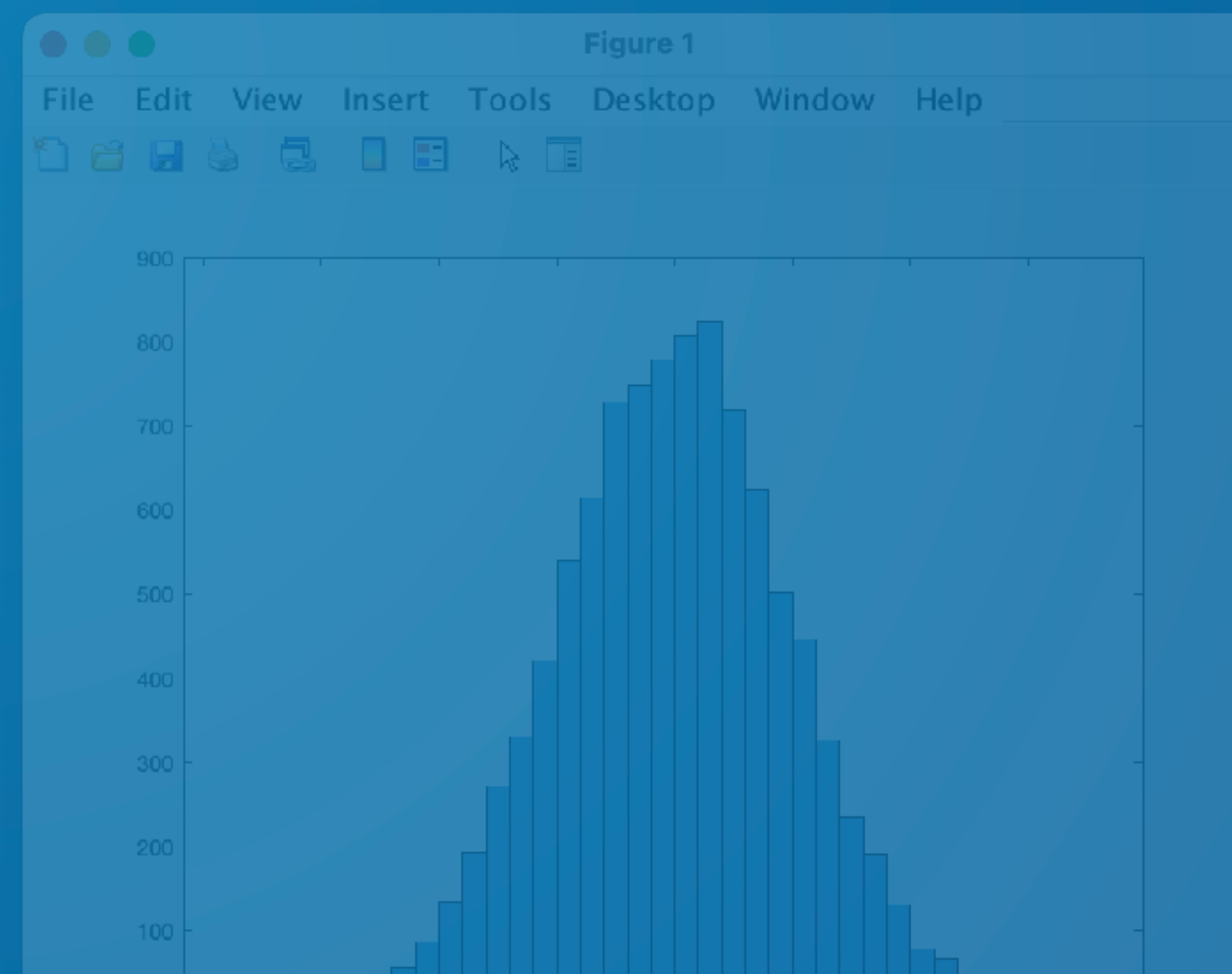
```
marks = [25,20,35,15,5];
```

```
pie(marks);
```



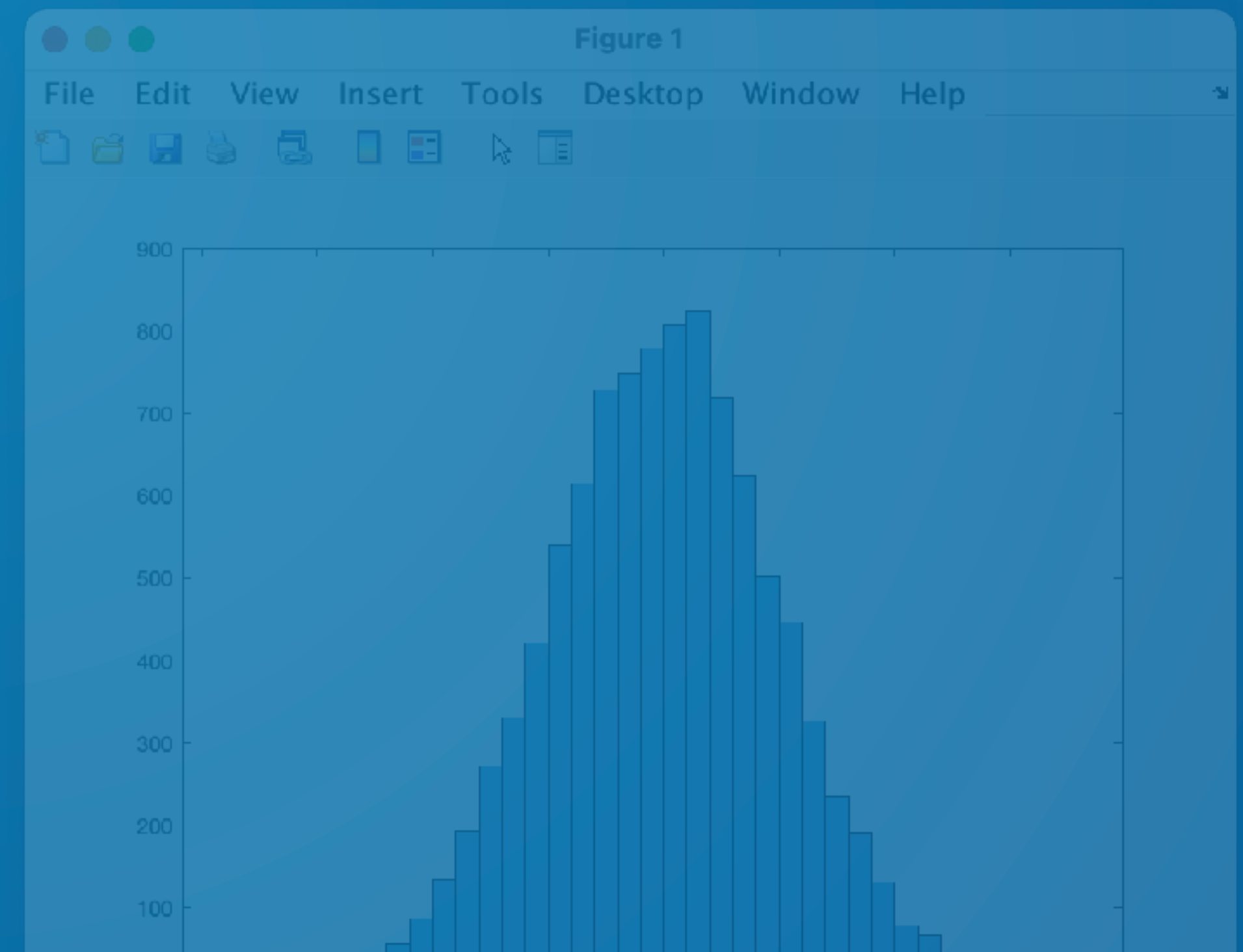


hist([var]) →



hist([var]) →

Genera el histograma de un conjunto de datos pasados como argumento.

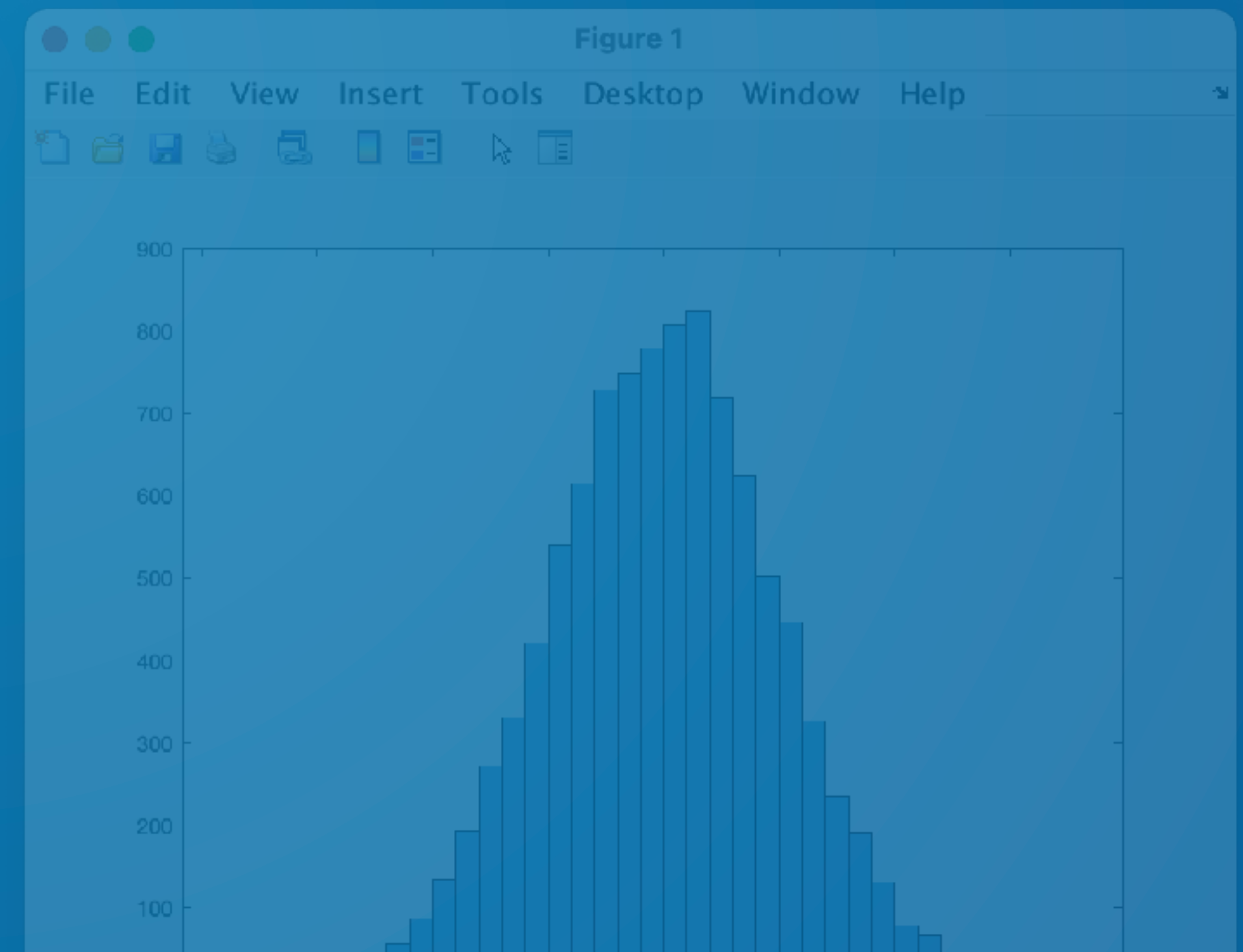


hist([var]) →

Genera el histograma de un conjunto de datos pasados como argumento.

EJEMPLO DE USO:

```
values = randn(1,10000);  
histogram(values);
```

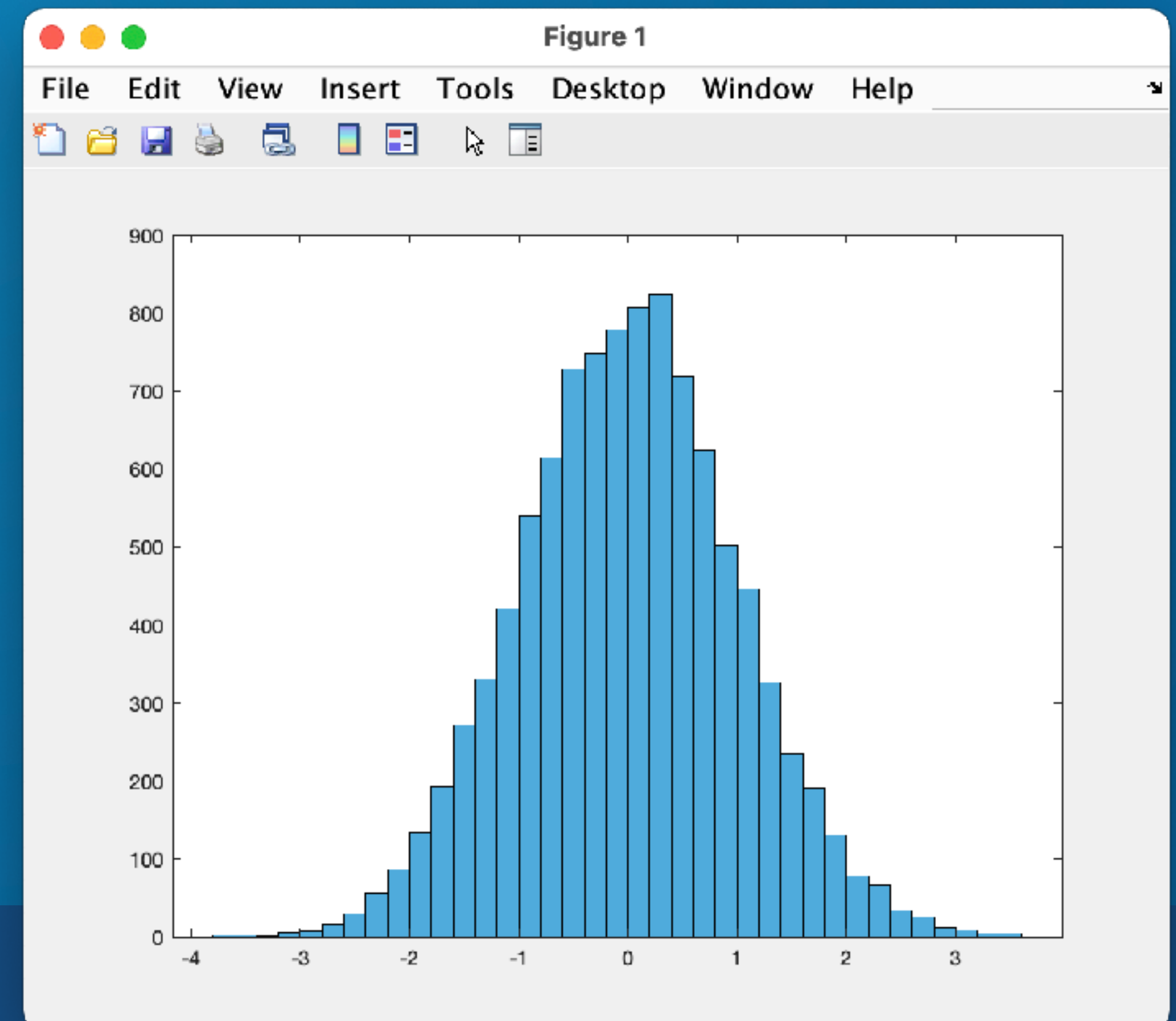


hist([var]) →

Genera el histograma de un conjunto de datos pasados como argumento.

EJEMPLO DE USO:

```
values = randn(1,10000);  
histogram(values);
```

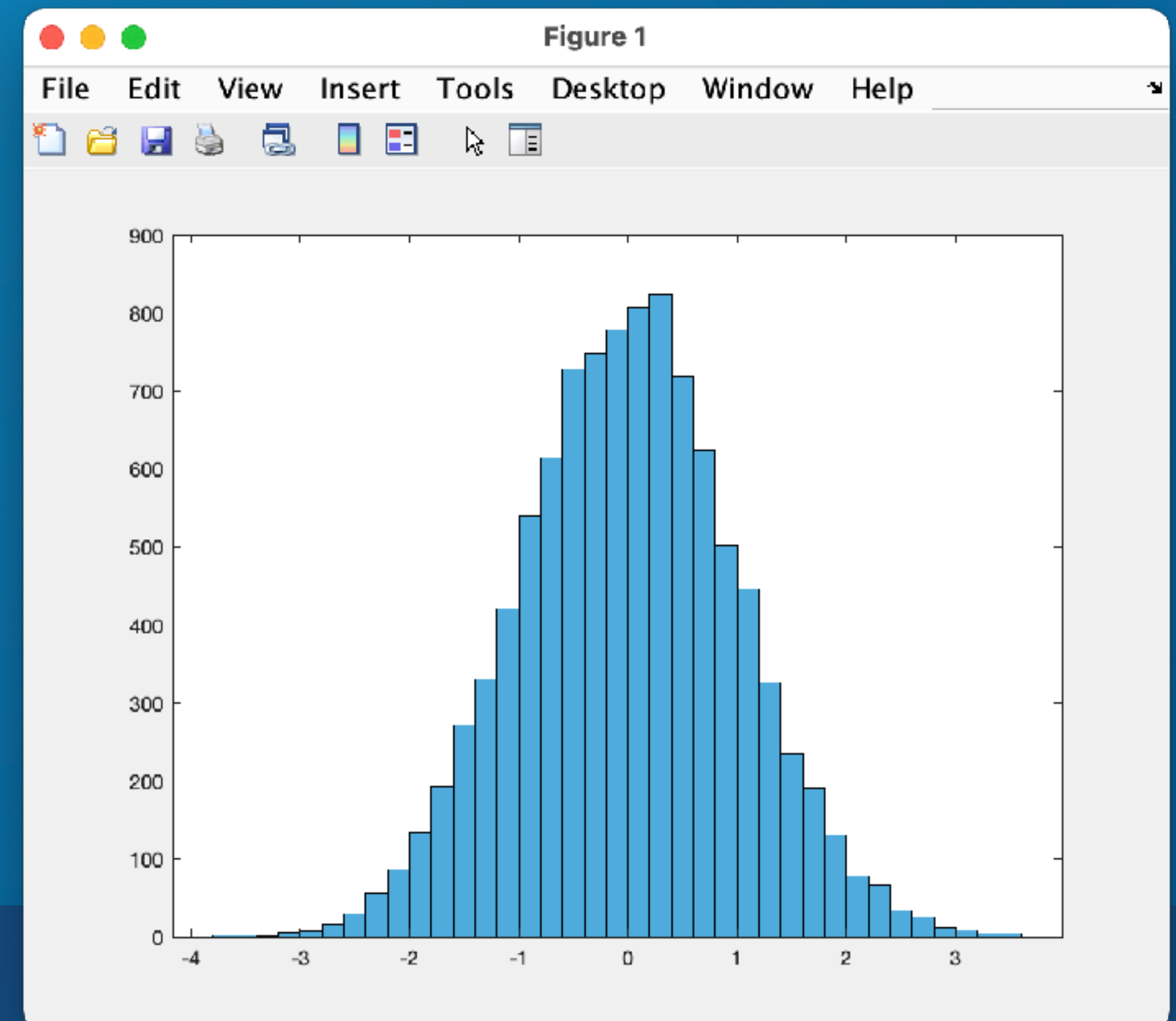


hist([var]) →

Genera el histograma de un conjunto de datos pasados como argumento.

EJEMPLO DE USO:

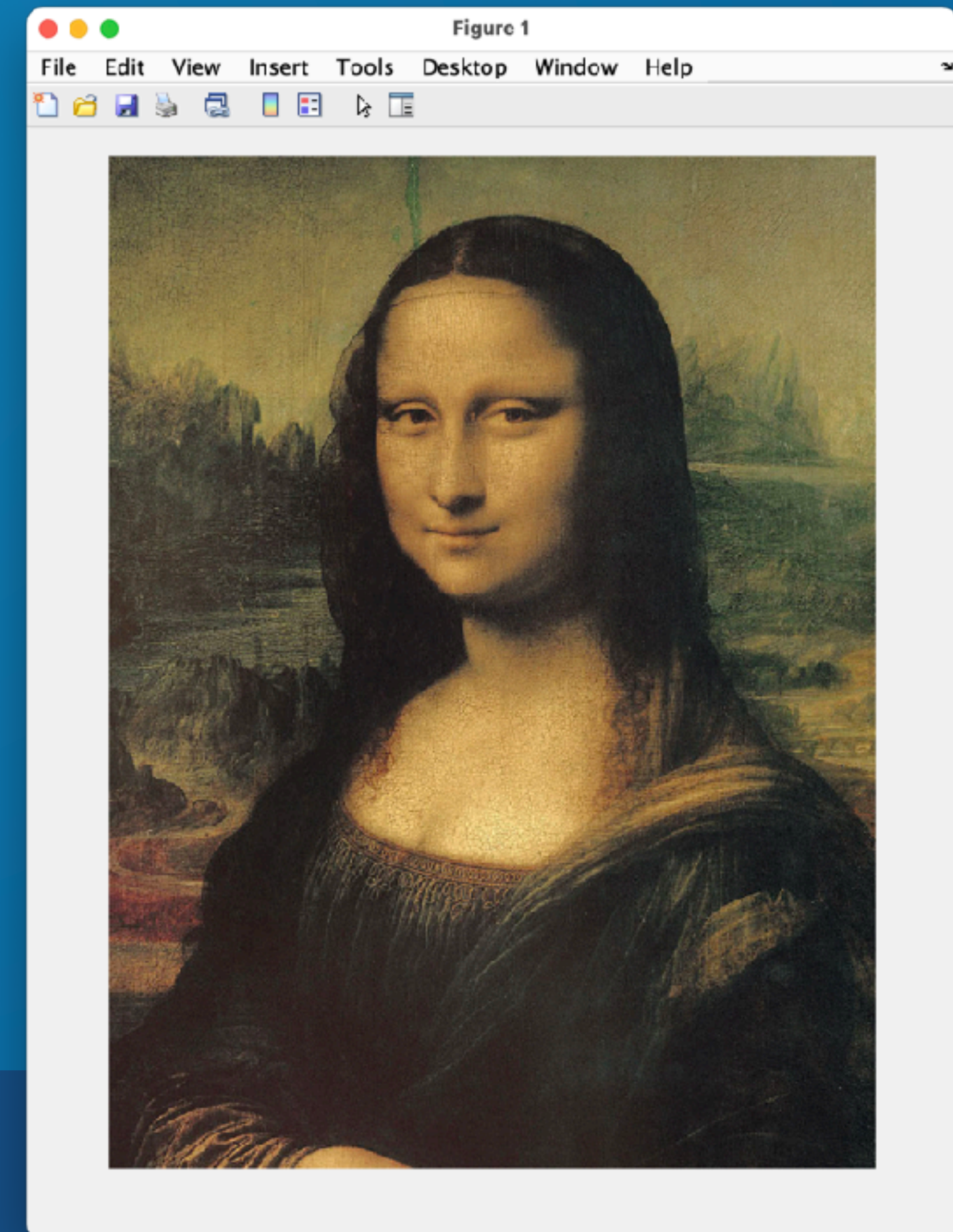
```
values = randn(1,10000);  
histogram(values);
```

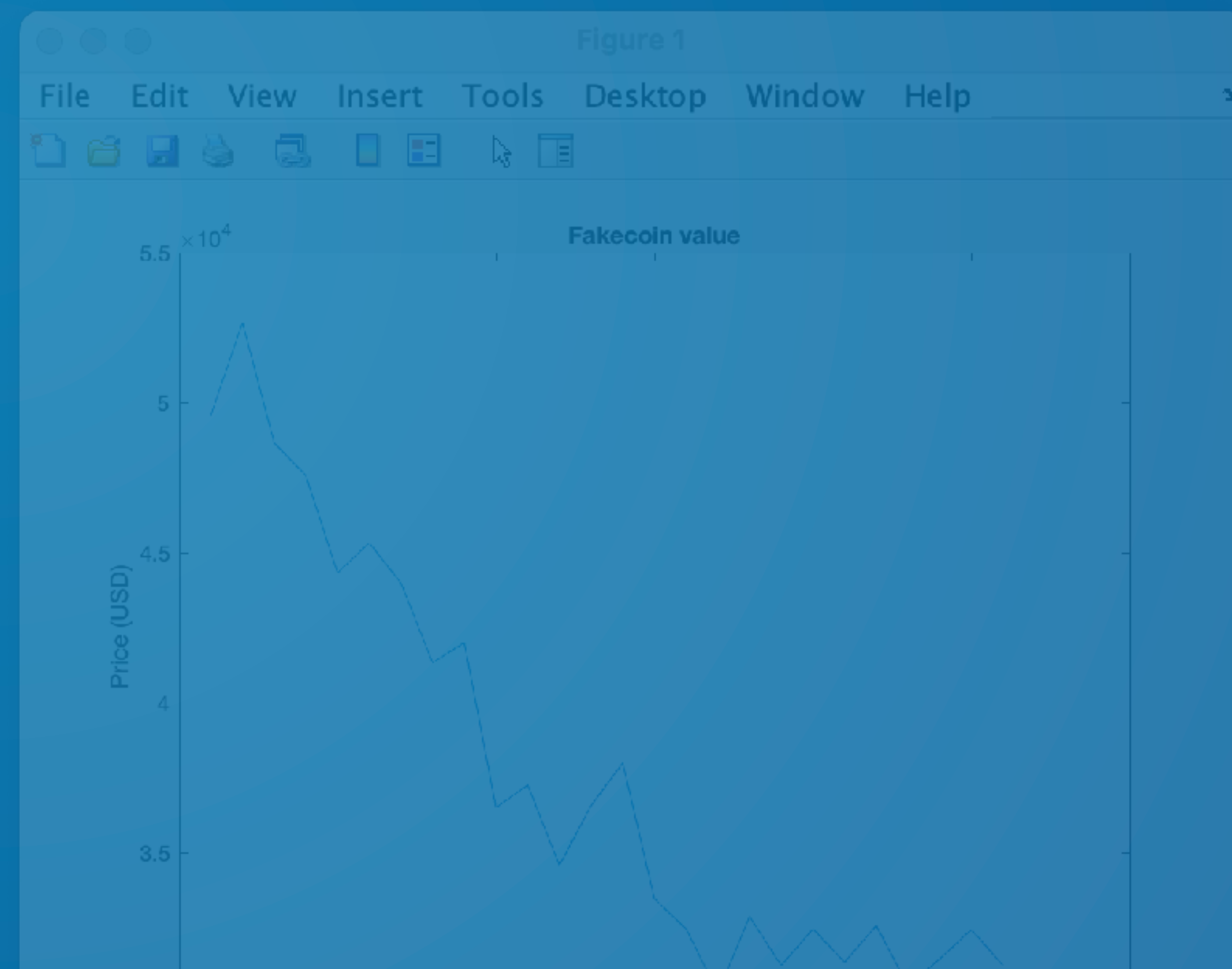


imshow([var]) → Representa gráficamente una imagen que le pasemos como argumento.

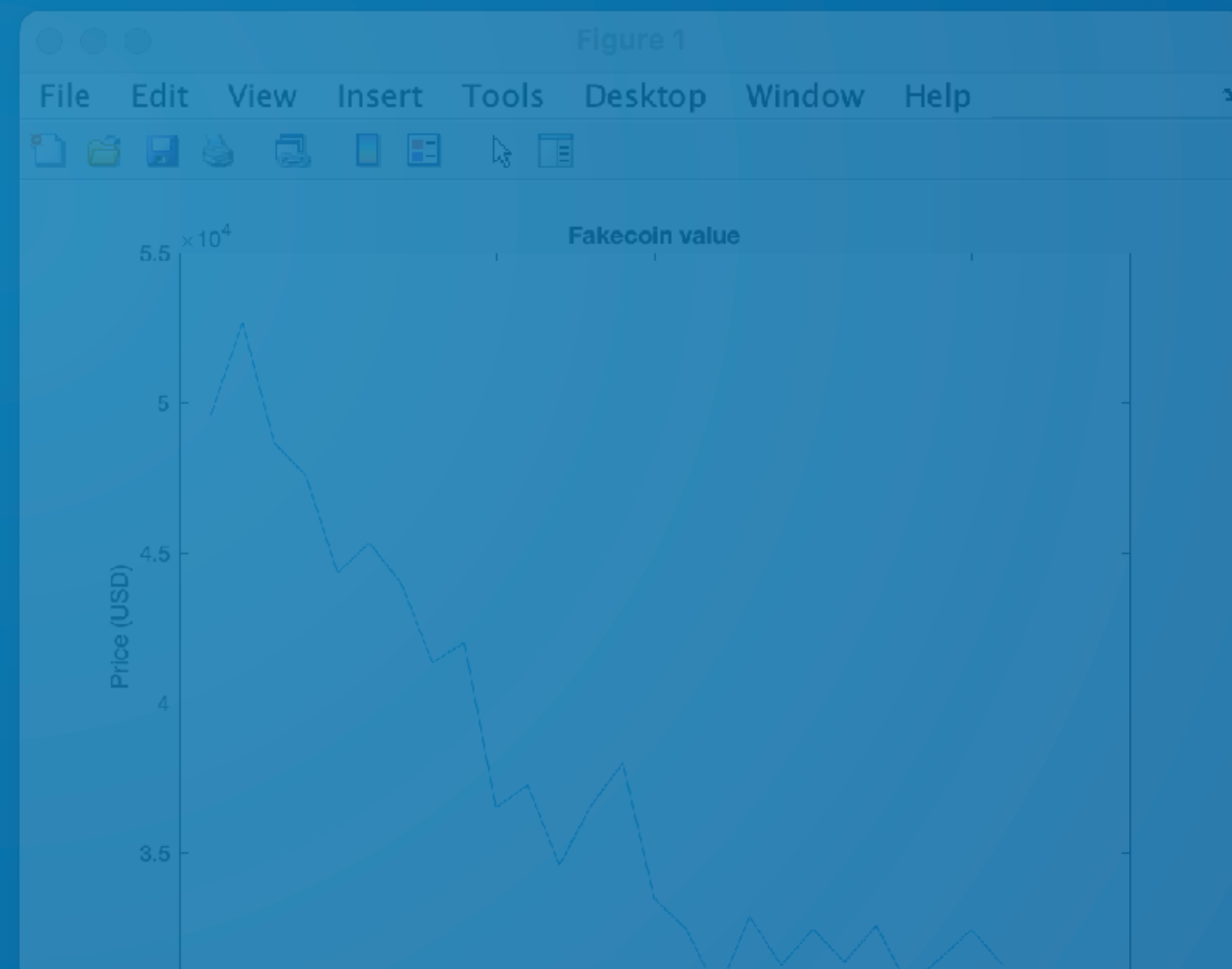
EJEMPLO DE USO:

```
figure(1);  
imshow(gioconda);
```

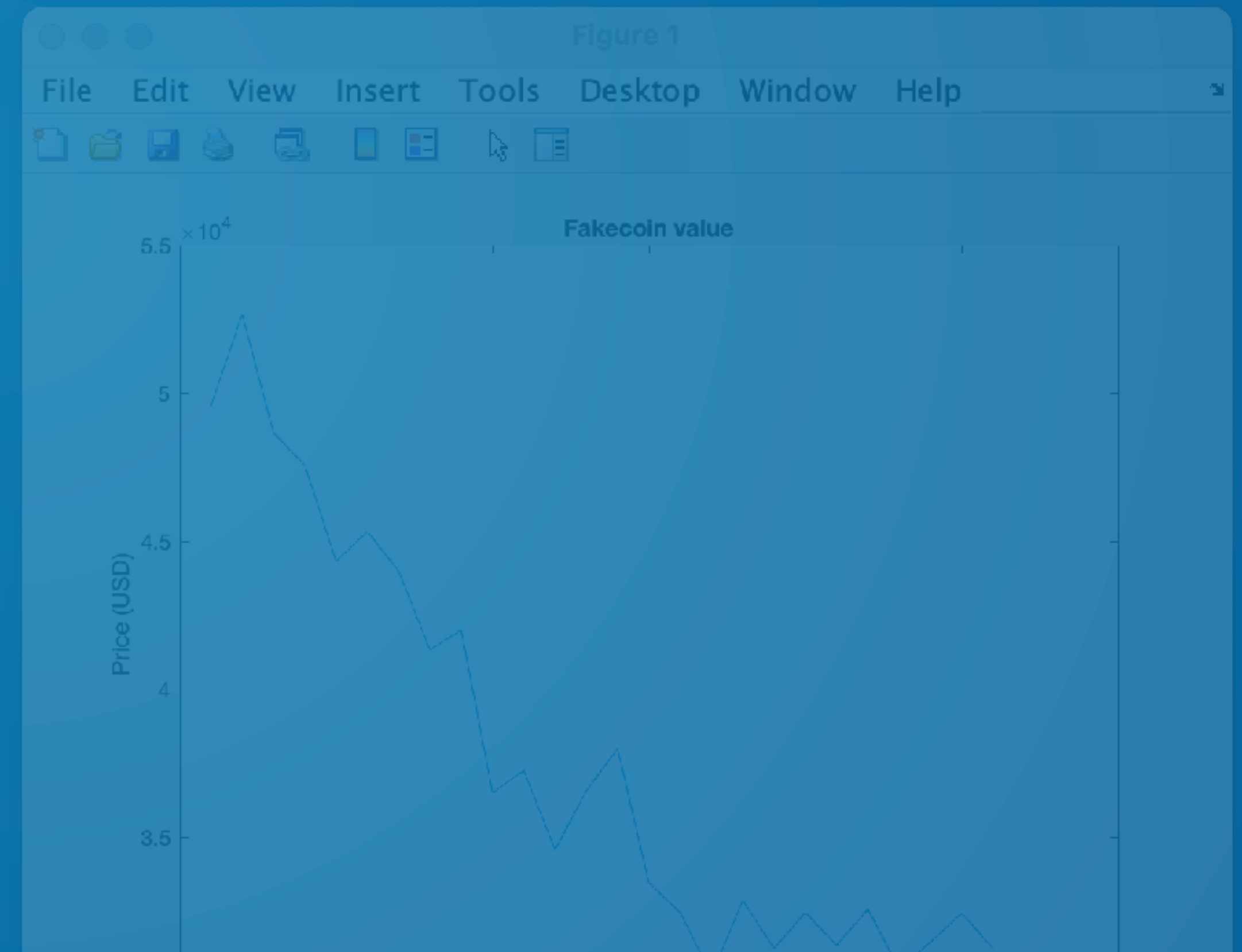




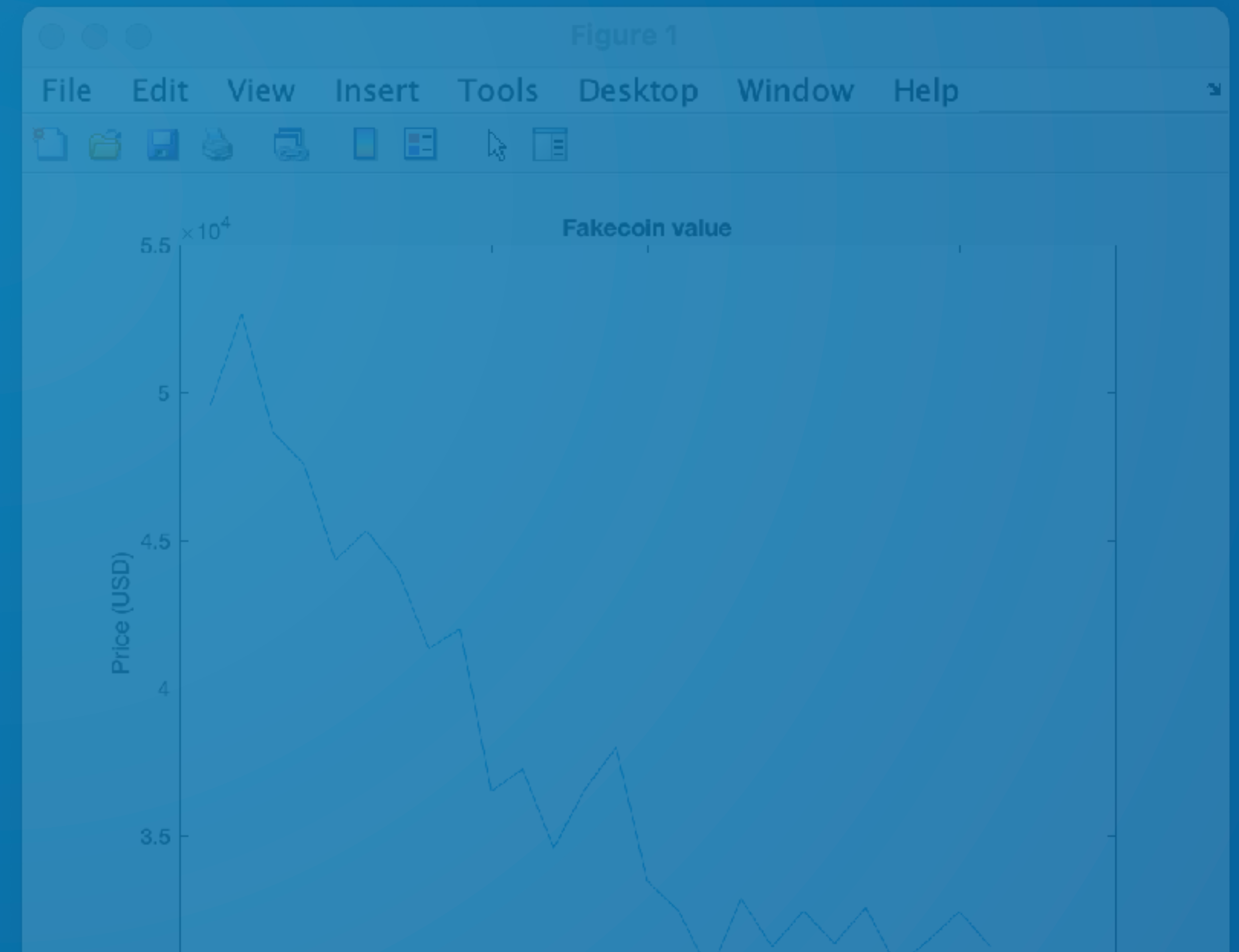
title([string])



`title([string])` `xlabel([string])`



title([string]) **xlabel([string])** **ylabel([string])**

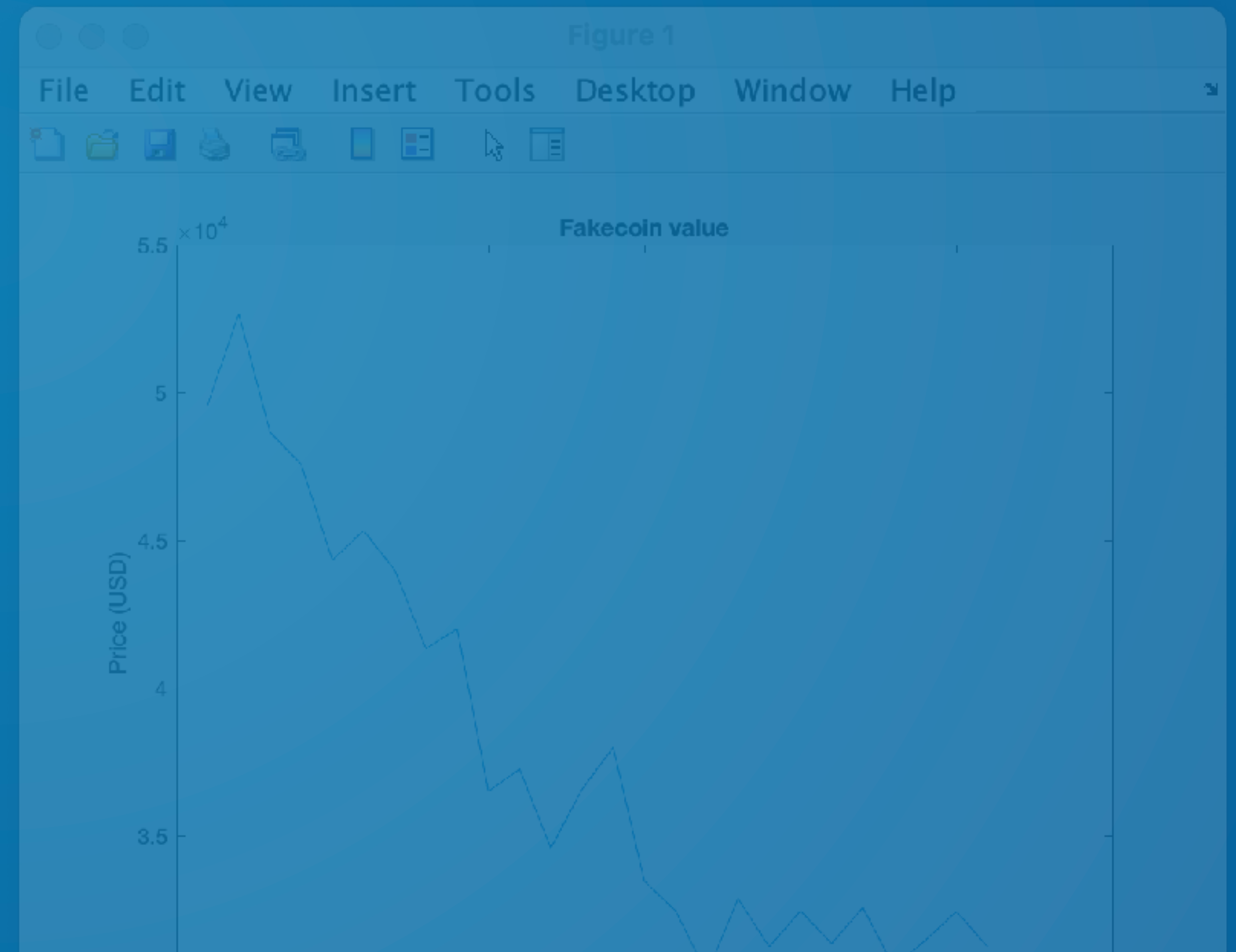


title([string]) **xlabel([string])** **ylabel([string])**

EJEMPLO DE USO:

```
bitcoin = [49567, 52657, ...
```

```
plot(bitcoin);
```



`title([string])` `xlabel([string])` `ylabel([string])`

EJEMPLO DE USO:

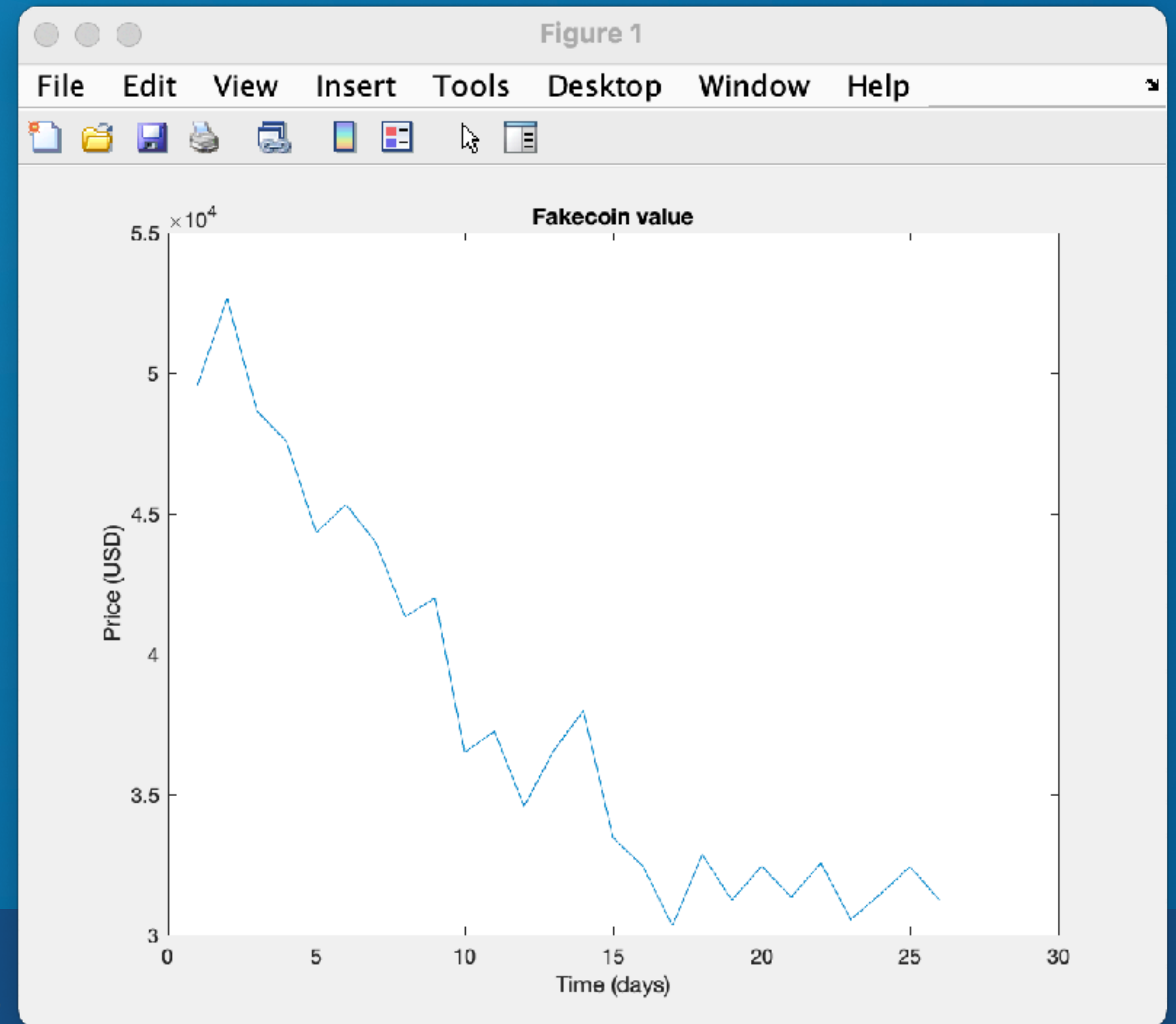
```
bitcoin = [49567, 52657, ...
```

```
plot(bitcoin);
```

```
title('Bitcoin value');
```

```
xlabel('Time (days)');
```

```
ylabel('Price (USD)');
```





subplot(f,c,p) →

subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

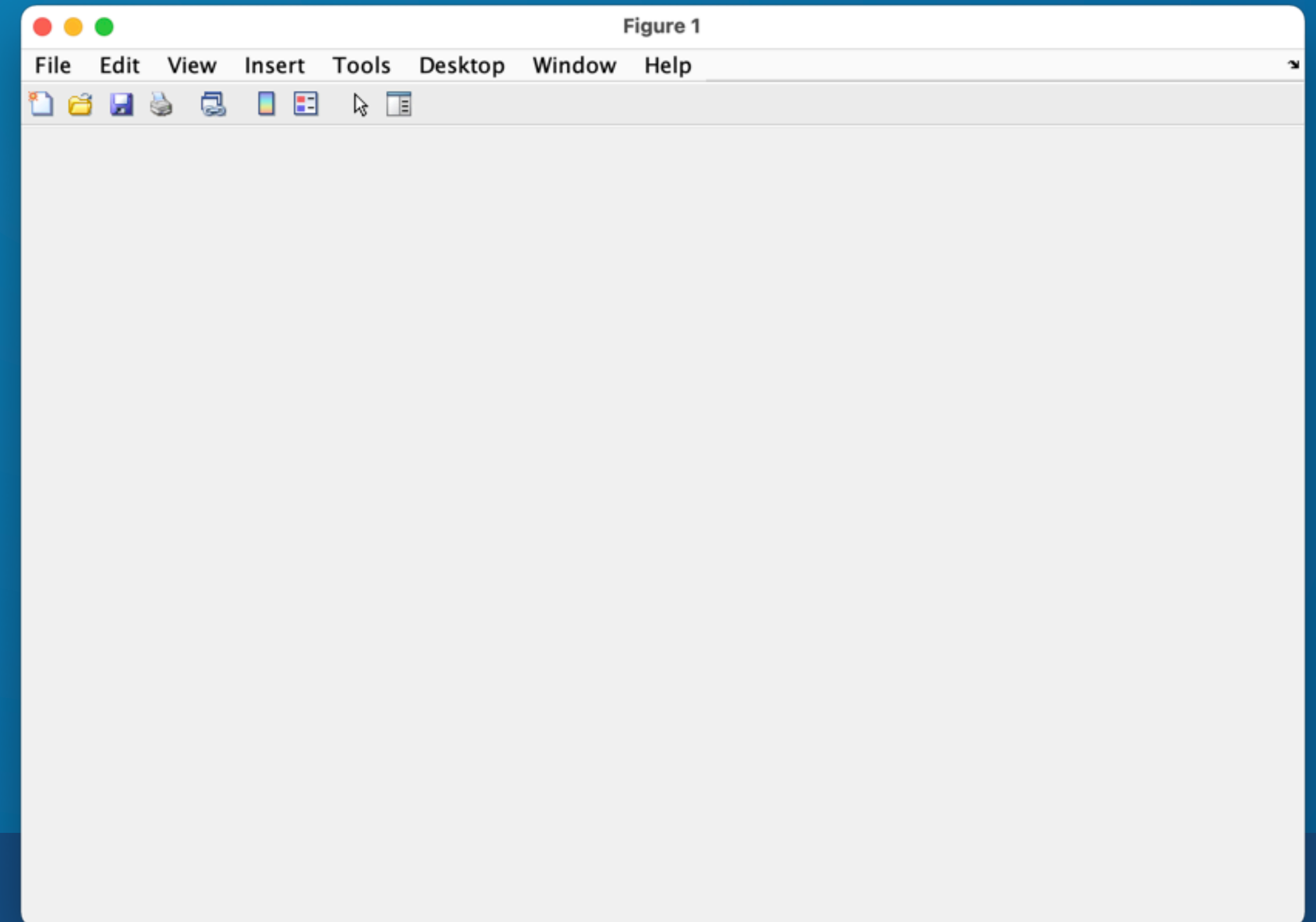
subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

EJEMPLO DE USO:

subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

EJEMPLO DE USO:

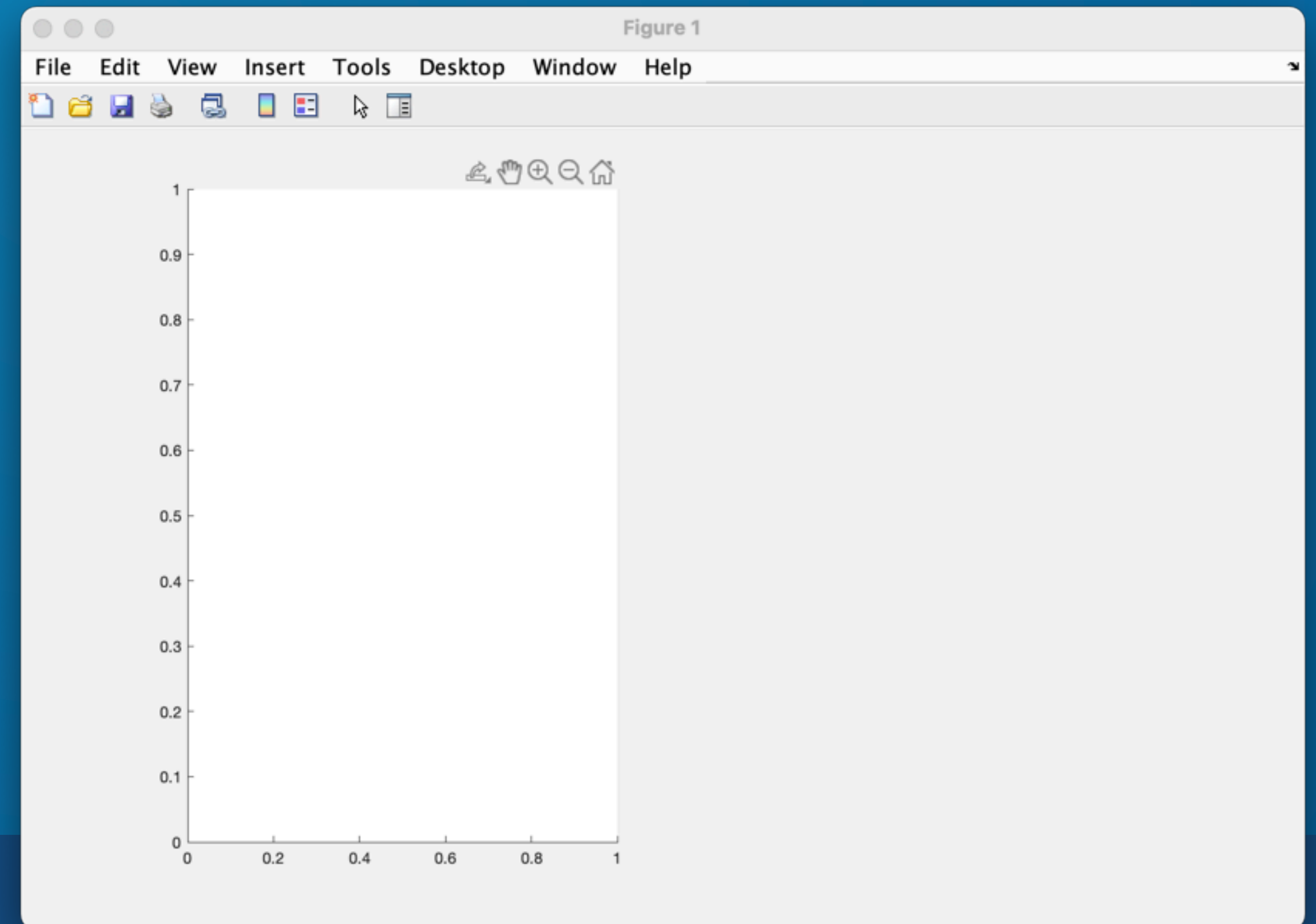
figure(1);



subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

EJEMPLO DE USO:

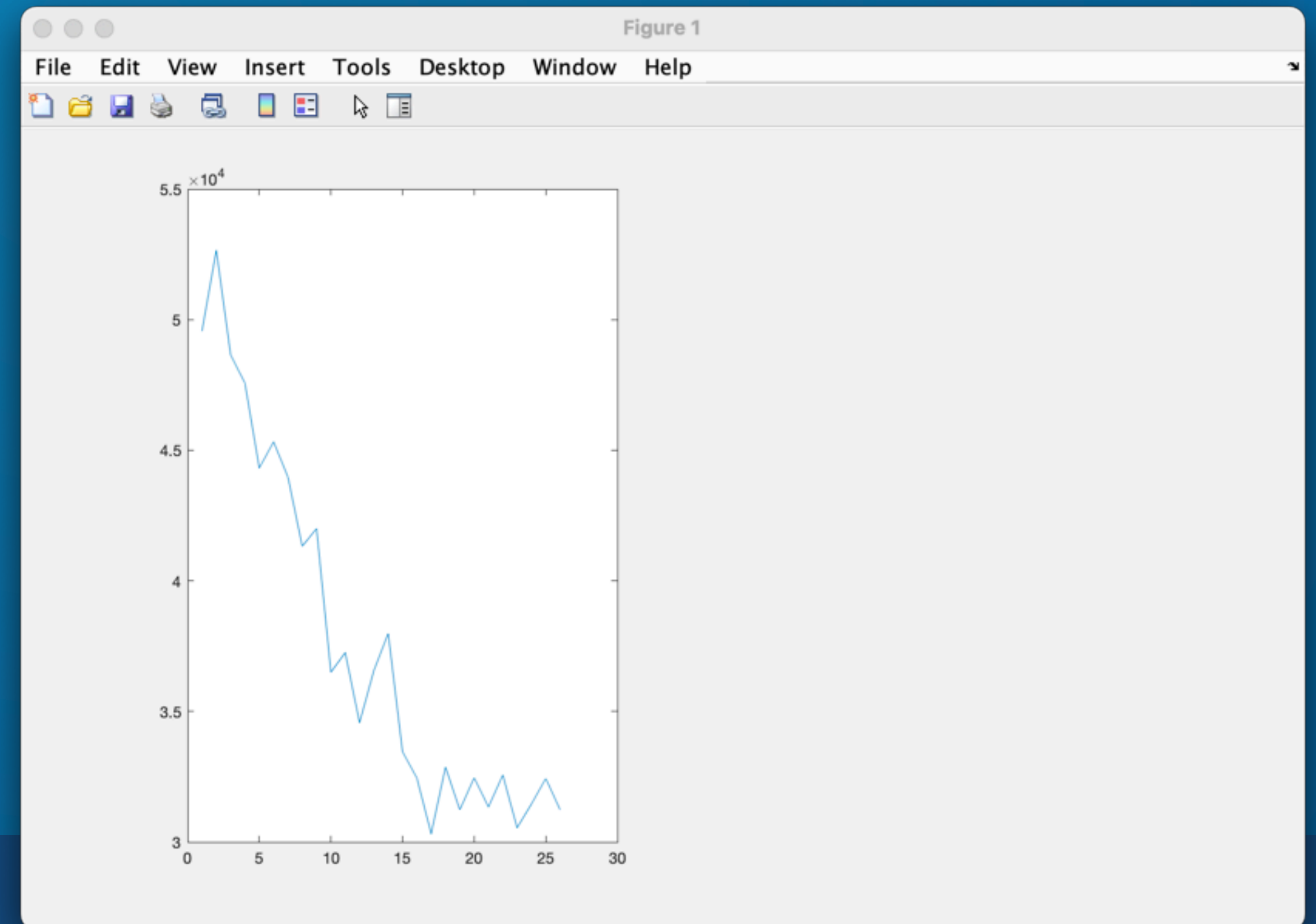
```
figure(1);  
subplot(1,2,1);
```



subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

EJEMPLO DE USO:

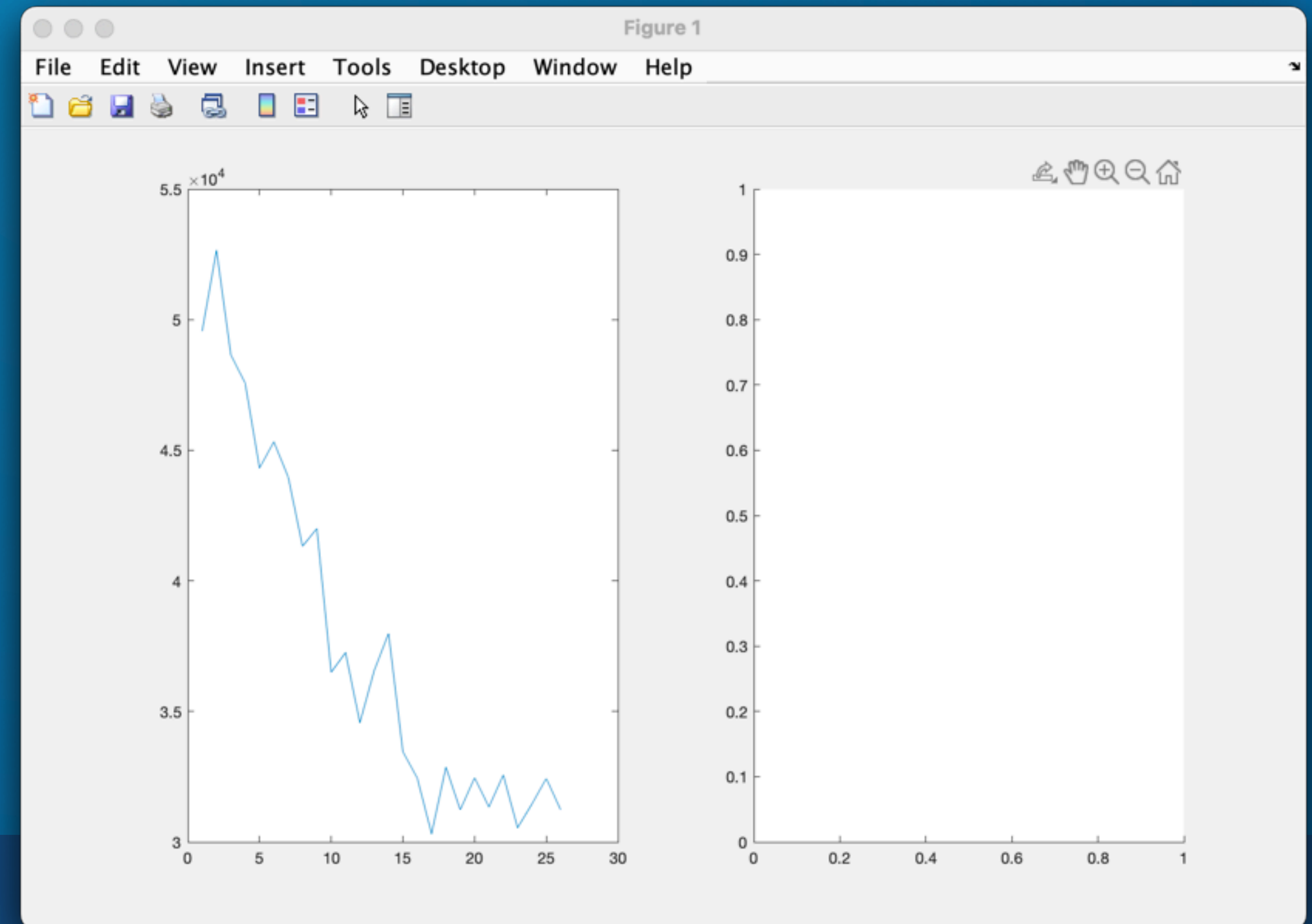
```
figure(1);  
subplot(1,2,1);  
plot(bitcoin);
```



subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

EJEMPLO DE USO:

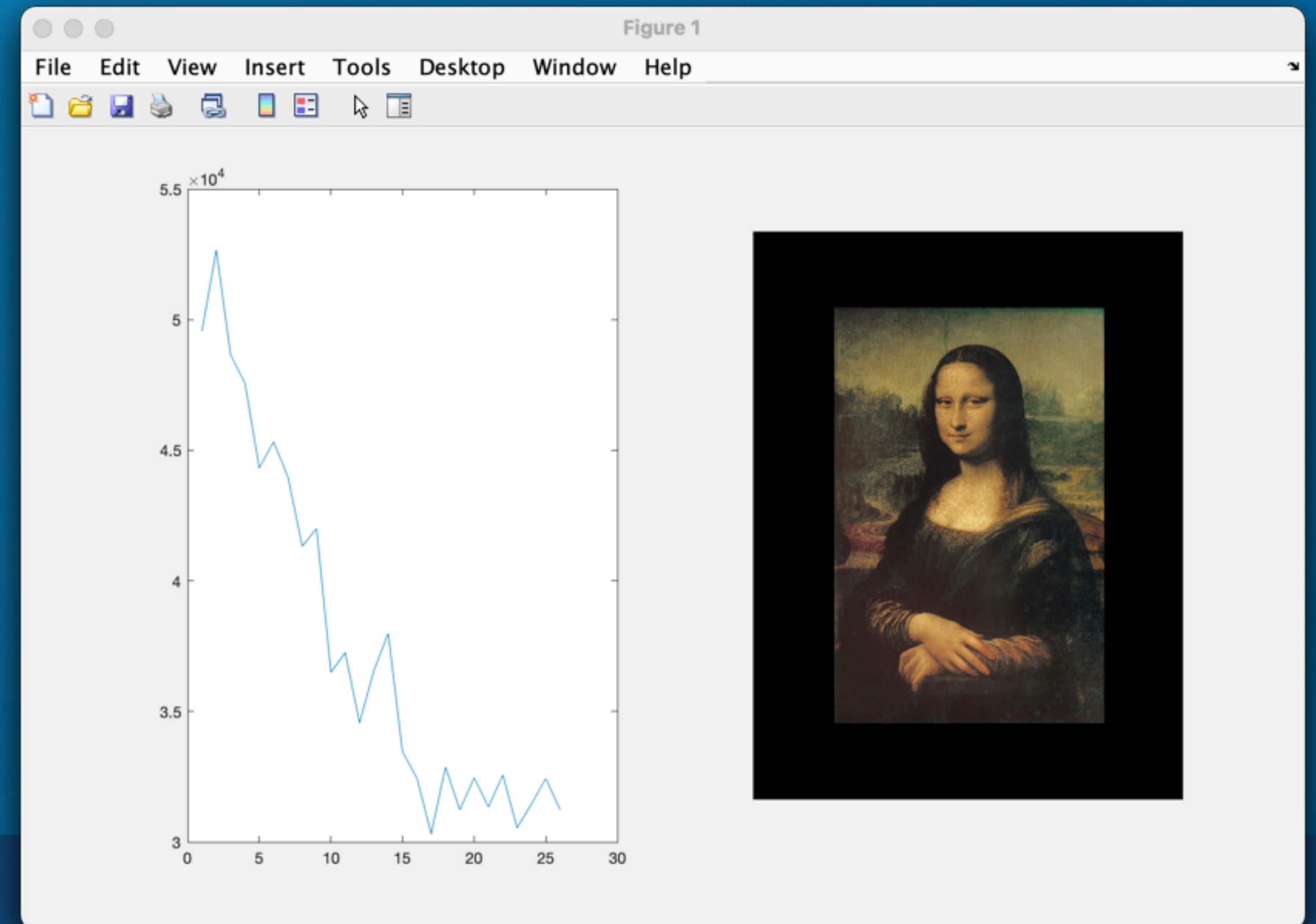
```
figure(1);  
subplot(1,2,1);  
plot(bitcoin);  
subplot(1,2,2);
```



subplot(f,c,p) → Nos permite representar distintos gráficos dentro de una figura.

EJEMPLO DE USO:

```
figure(1);  
subplot(1,2,1);  
plot(bitcoin);  
subplot(1,2,2);  
imshow(gioconda);
```





run filename.m ➡

run filename.m → Ejecuta el código contenido en el archivo .m indicado.

run filename.m → Ejecuta el código contenido en el archivo .m indicado.

EJEMPLOS DE USO:

run practiceBlock.m;

run filename.m → Ejecuta el código contenido en el archivo .m indicado.

EJEMPLOS DE USO:

```
run practiceBlock.m;
```

```
run script/blocks/practiceBlock.m;
```

run filename.m → Ejecuta el código contenido en el archivo .m indicado.

EJEMPLOS DE USO:



```
run practiceBlock.m;
```

```
run script/blocks/practiceBlock.m;
```



save filename.mat ➡



save filename.mat ➡

Guarda los datos del workspace completo en un archivo .mat con el nombre especificado.



save filename.mat ➡

Guarda los datos del workspace completo en un archivo .mat con el nombre especificado.

EJEMPLOS DE USO:



save ctbMatrix.mat;

save filename.mat ➡

Guarda los datos del workspace completo en un archivo .mat con el nombre especificado.

EJEMPLOS DE USO:



save ctbMatrix.mat;

save filename.mat ➡

Guarda los datos del workspace completo en un archivo .mat con el nombre especificado.

EJEMPLOS DE USO:



```
save ctbMatrix.mat;
```

```
save counterbalance/ctbMatrix.mat;
```




load filename.mat ➡

load filename.mat ➡ Carga en el workspace el archivo .mat indicado.

load filename.mat → Carga en el workspace el archivo .mat indicado.

EJEMPLOS DE USO:

```
load ctbMatrix.mat;
```

load filename.mat → Carga en el workspace el archivo .mat indicado.

EJEMPLOS DE USO:

```
load ctbMatrix.mat;
```

```
load counterbalance/ctbMatrix.mat;
```




Command Window

```
>> help save
save Save workspace variables to file.
  save(FILENAME) stores all variables from the current workspace in a
  MATLAB formatted binary file (MAT-file) called FILENAME. Specify
  FILENAME as a character vector or a string scalar. For example, specify
  FILENAME as 'myFile.mat' or "myFile.mat".

  save(FILENAME,VARIABLES) stores only the specified variables. Specify
  FILENAME and VARIABLES as character vectors or string scalars.

  save(FILENAME,'-struct',STRUCTNAME,FIELDNAMES) stores the fields of the
  specified scalar structure as individual variables in the file. If you
  include the optional FIELDNAMES, the save function stores only the
  specified fields of the structure. You cannot specify VARIABLES and
  the '-struct' keyword in the same call to save.

  save(FILENAME, ..., '-append') adds new variables to an existing file.
  You can specify '-append' with additional inputs such as VARIABLES,
  '-struct', FORMAT, or VERSION.

  save(FILENAME, ..., FORMAT) saves in the specified format: '-mat' or
  '-ascii'.
  You can specify FORMAT with additional inputs such as VARIABLES,
  '-struct', '-append', or VERSION.

  save(FILENAME, ..., VERSION) saves to MAT-files in the specified
  version: '-v4', '-v6', '-v7', or '-v7.3'.
  You can specify VERSION with additional inputs such as VARIABLES,
  '-struct', '-append', '-nocompression', or FORMAT.
```

help funcname ➡

Command Window

```
>> help save
save Save workspace variables to file.
  save(FILENAME) stores all variables from the current workspace in a
  MATLAB formatted binary file (MAT-file) called FILENAME. Specify
  FILENAME as a character vector or a string scalar. For example, specify
  FILENAME as 'myFile.mat' or "myFile.mat".

  save(FILENAME,VARIABLES) stores only the specified variables. Specify
  FILENAME and VARIABLES as character vectors or string scalars.

  save(FILENAME,'-struct',STRUCTNAME,FIELDNAMES) stores the fields of the
  specified scalar structure as individual variables in the file. If you
  include the optional FIELDNAMES, the save function stores only the
  specified fields of the structure. You cannot specify VARIABLES and
  the '-struct' keyword in the same call to save.

  save(FILENAME, ..., '-append') adds new variables to an existing file.
  You can specify '-append' with additional inputs such as VARIABLES,
  '-struct', FORMAT, or VERSION.

  save(FILENAME, ..., FORMAT) saves in the specified format: '-mat' or
  '-ascii'.
  You can specify FORMAT with additional inputs such as VARIABLES,
  '-struct', '-append', or VERSION.

  save(FILENAME, ..., VERSION) saves to MAT-files in the specified
  version: '-v4', '-v6', '-v7', or '-v7.3'.
  You can specify VERSION with additional inputs such as VARIABLES,
  '-struct', '-append', '-nocompression', or FORMAT.
```

help funcname



Muestra en la línea de comandos ayuda sobre la función especificada.

```
Command Window
>> help save
save Save workspace variables to file.
  save(FILENAME) stores all variables from the current workspace in a
  MATLAB formatted binary file (MAT-file) called FILENAME. Specify
  FILENAME as a character vector or a string scalar. For example, specify
  FILENAME as 'myFile.mat' or "myFile.mat".

  save(FILENAME,VARIABLES) stores only the specified variables. Specify
  FILENAME and VARIABLES as character vectors or string scalars.

  save(FILENAME,'-struct',STRUCTNAME,FIELDNAMES) stores the fields of the
  specified scalar structure as individual variables in the file. If you
  include the optional FIELDNAMES, the save function stores only the
  specified fields of the structure. You cannot specify VARIABLES and
  the '-struct' keyword in the same call to save.

  save(FILENAME, ..., '-append') adds new variables to an existing file.
  You can specify '-append' with additional inputs such as VARIABLES,
  '-struct', FORMAT, or VERSION.

  save(FILENAME, ..., FORMAT) saves in the specified format: '-mat' or
  '-ascii'.
  You can specify FORMAT with additional inputs such as VARIABLES,
  '-struct', '-append', or VERSION.

  save(FILENAME, ..., VERSION) saves to MAT-files in the specified
  version: '-v4', '-v6', '-v7', or '-v7.3'.
  You can specify VERSION with additional inputs such as VARIABLES,
  '-struct', '-append', '-nocompression', or FORMAT.
```


help funcname

➔ Muestra en la línea de comandos ayuda sobre la función especificada.

EJEMPLOS DE USO:

help save

```
Command Window
>> help save
save Save workspace variables to file.
  save(FILENAME) stores all variables from the current workspace in a
  MATLAB formatted binary file (MAT-file) called FILENAME. Specify
  FILENAME as a character vector or a string scalar. For example, specify
  FILENAME as 'myFile.mat' or "myFile.mat".

  save(FILENAME,VARIABLES) stores only the specified variables. Specify
  FILENAME and VARIABLES as character vectors or string scalars.

  save(FILENAME,'-struct',STRUCTNAME,FIELDNAMES) stores the fields of the
  specified scalar structure as individual variables in the file. If you
  include the optional FIELDNAMES, the save function stores only the
  specified fields of the structure. You cannot specify VARIABLES and
  the '-struct' keyword in the same call to save.

  save(FILENAME, ..., '-append') adds new variables to an existing file.
  You can specify '-append' with additional inputs such as VARIABLES,
  '-struct', FORMAT, or VERSION.

  save(FILENAME, ..., FORMAT) saves in the specified format: '-mat' or
  '-ascii'.
  You can specify FORMAT with additional inputs such as VARIABLES,
  '-struct', '-append', or VERSION.

  save(FILENAME, ..., VERSION) saves to MAT-files in the specified
  version: '-v4', '-v6', '-v7', or '-v7.3'.
  You can specify VERSION with additional inputs such as VARIABLES,
  '-struct', '-append', '-nocompression', or FORMAT.
```

help funcname

➔ Muestra en la línea de comandos ayuda sobre la función especificada.

EJEMPLOS DE USO:

help save

```
Command Window
>> help save
save Save workspace variables to file.
  save(FILENAME) stores all variables from the current workspace in a
  MATLAB formatted binary file (MAT-file) called FILENAME. Specify
  FILENAME as a character vector or a string scalar. For example, specify
  FILENAME as 'myFile.mat' or "myFile.mat".

  save(FILENAME,VARIABLES) stores only the specified variables. Specify
  FILENAME and VARIABLES as character vectors or string scalars.

  save(FILENAME,'-struct',STRUCTNAME,FIELDNAMES) stores the fields of the
  specified scalar structure as individual variables in the file. If you
  include the optional FIELDNAMES, the save function stores only the
  specified fields of the structure. You cannot specify VARIABLES and
  the '-struct' keyword in the same call to save.

  save(FILENAME, ..., '-append') adds new variables to an existing file.
  You can specify '-append' with additional inputs such as VARIABLES,
  '-struct', FORMAT, or VERSION.

  save(FILENAME, ..., FORMAT) saves in the specified format: '-mat' or
  '-ascii'.
  You can specify FORMAT with additional inputs such as VARIABLES,
  '-struct', '-append', or VERSION.

  save(FILENAME, ..., VERSION) saves to MAT-files in the specified
  version: '-v4', '-v6', '-v7', or '-v7.3'.
  You can specify VERSION with additional inputs such as VARIABLES,
  '-struct', '-append', '-nocompression', or FORMAT.
```




Help

history – MathWorks Seleccionar País

Documentation

Search R2020a Documentation

CONTENIDO

« Documentation Home

« MATLAB

« Graphics

« 2-D and 3-D Plots

« Data Distribution Plots

histogram

ON THIS PAGE

Description

Creation

Properties

Object Functions

Examples

Tips

Extended Capabilities

See Also

Other uses of histogram

All Examples Functions

Trials Actualizaciones de productos

histogram

Histogram plot

R2020a

expand all in page

Description

Histograms are a type of bar plot for numeric data that group the data into bins. After you create a Histogram object, you can modify aspects of the histogram by changing its property values. This is particularly useful for quickly modifying the properties of the bins or changing the display.

Creation

Syntax

histogram(X)

histogram(X,nbins)

histogram(X,edges)

histogram('BinEdges',edges,'BinCounts',counts)

histogram(C)

histogram(C,Categories)

histogram('Categories',Categories,'BinCounts',counts)

histogram(__,Name,Value)

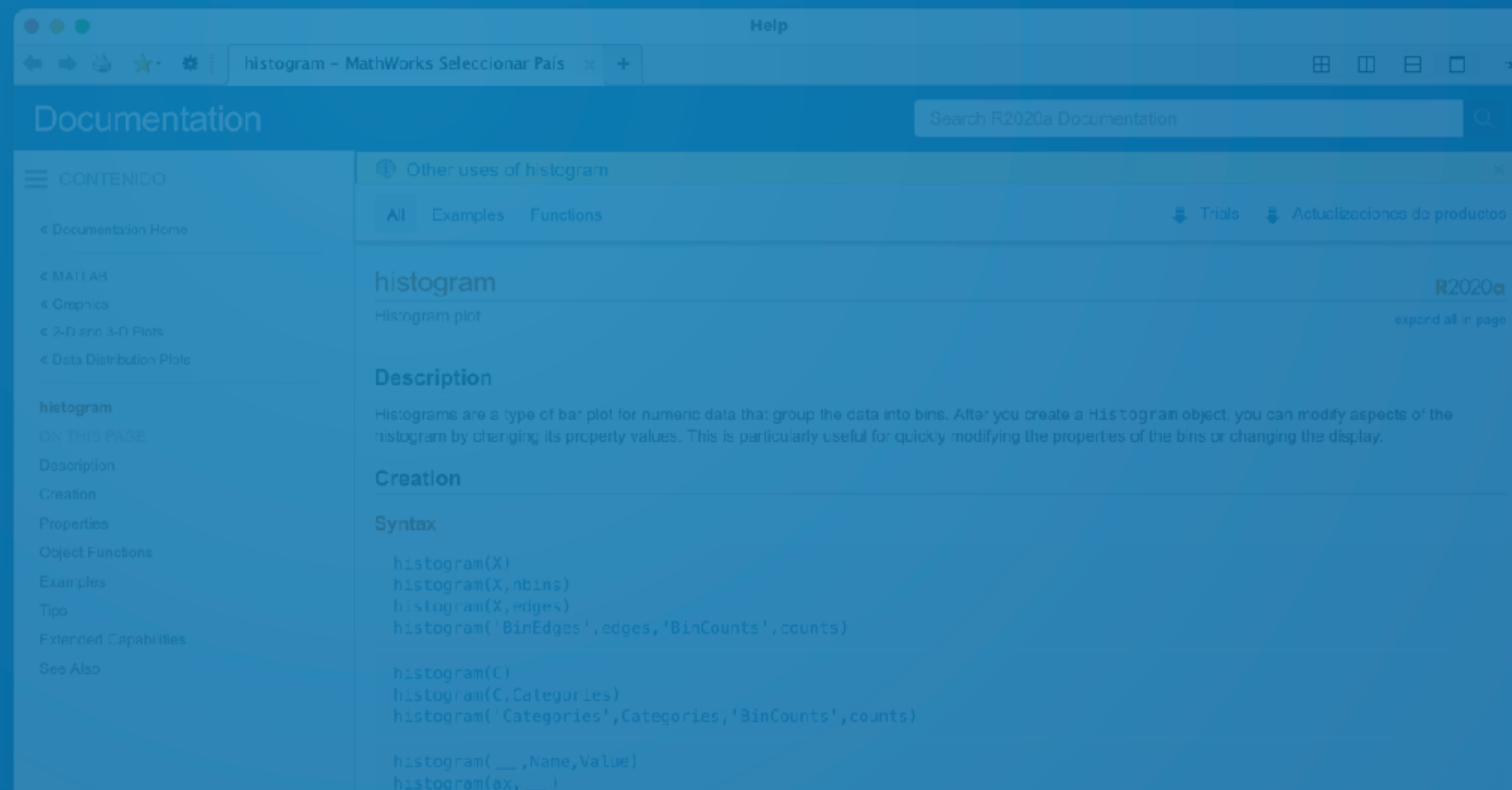
histogram(ax, __)

David López García
Centro de Investigación de Mente Cerebro y Comportamiento
Universidad de Granada

cimcyc

38

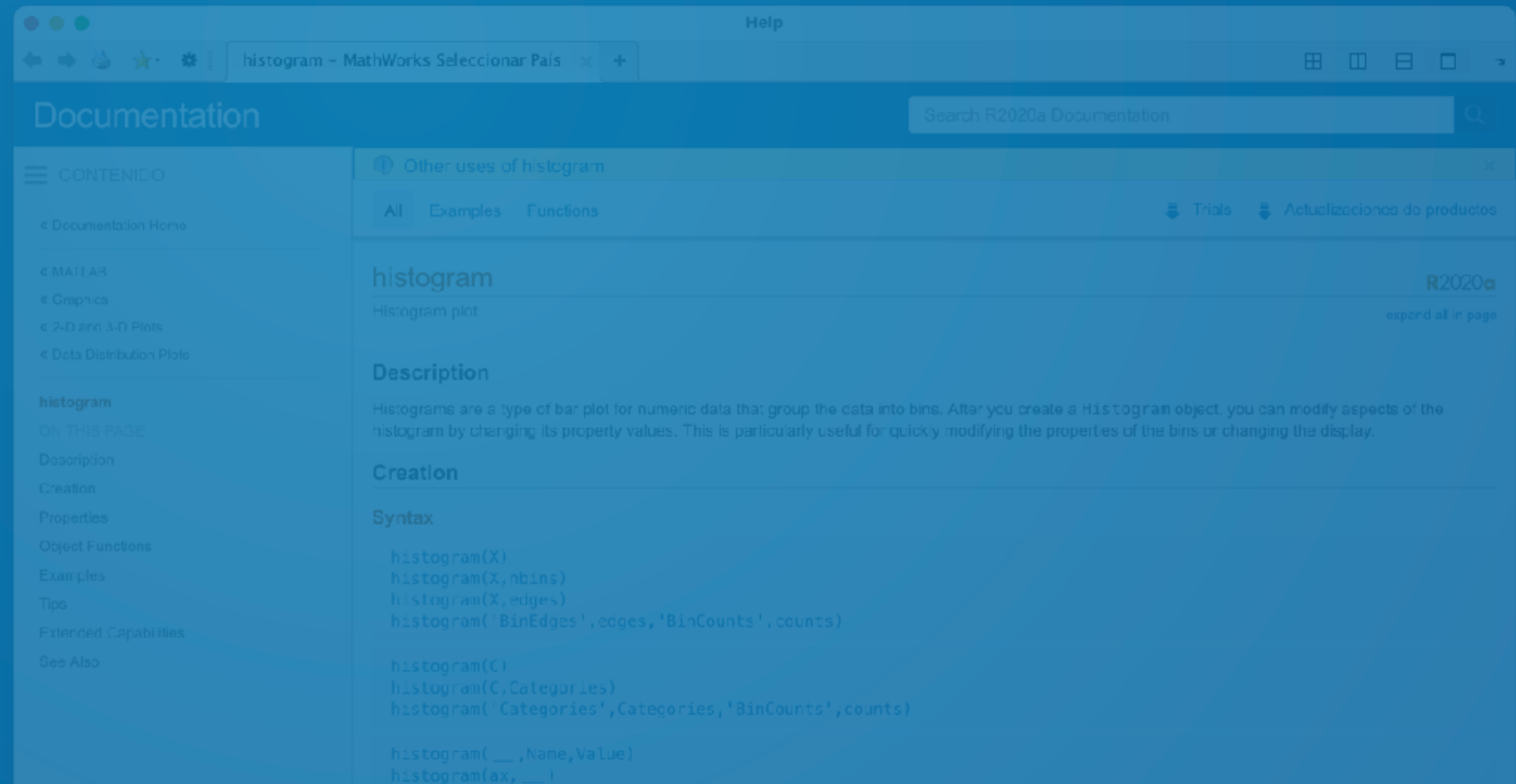
doc funcname ➡



doc funcname



Muestra en una ventana externa toda la documentación de la función pasada como argumento.



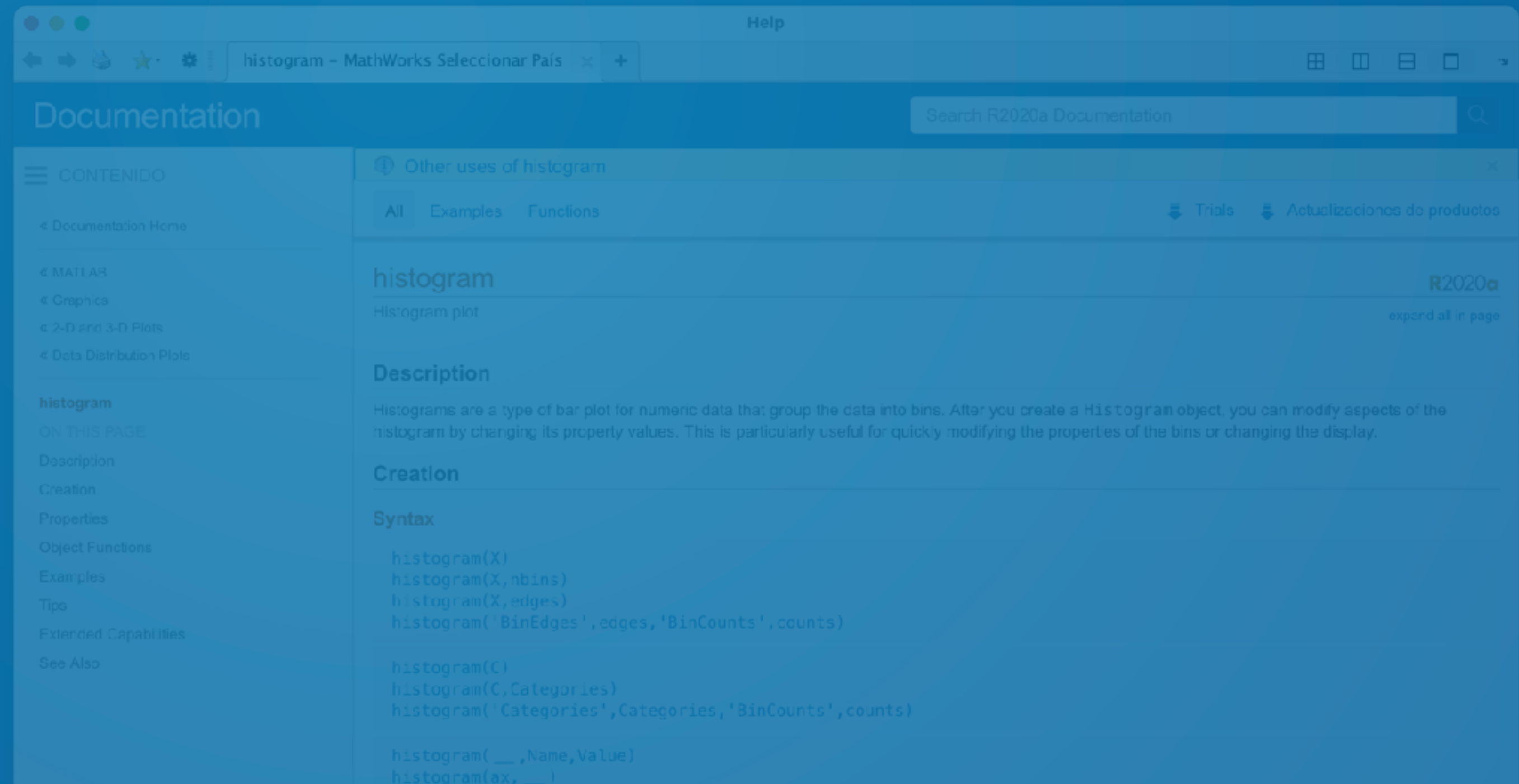
doc funcname



Muestra en una ventana externa toda la documentación de la función pasada como argumento.

EJEMPLOS DE USO:

doc histogram



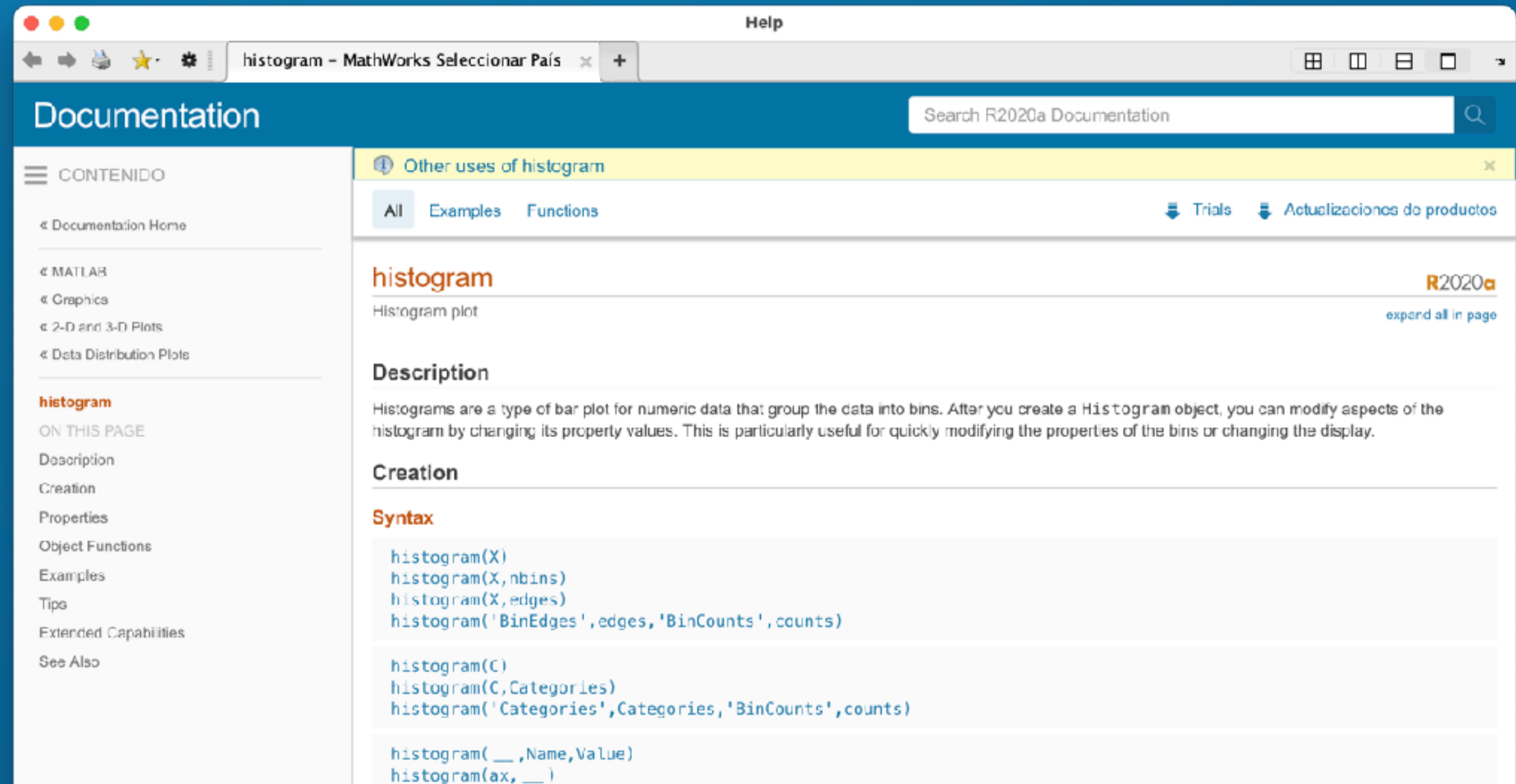
doc funcname

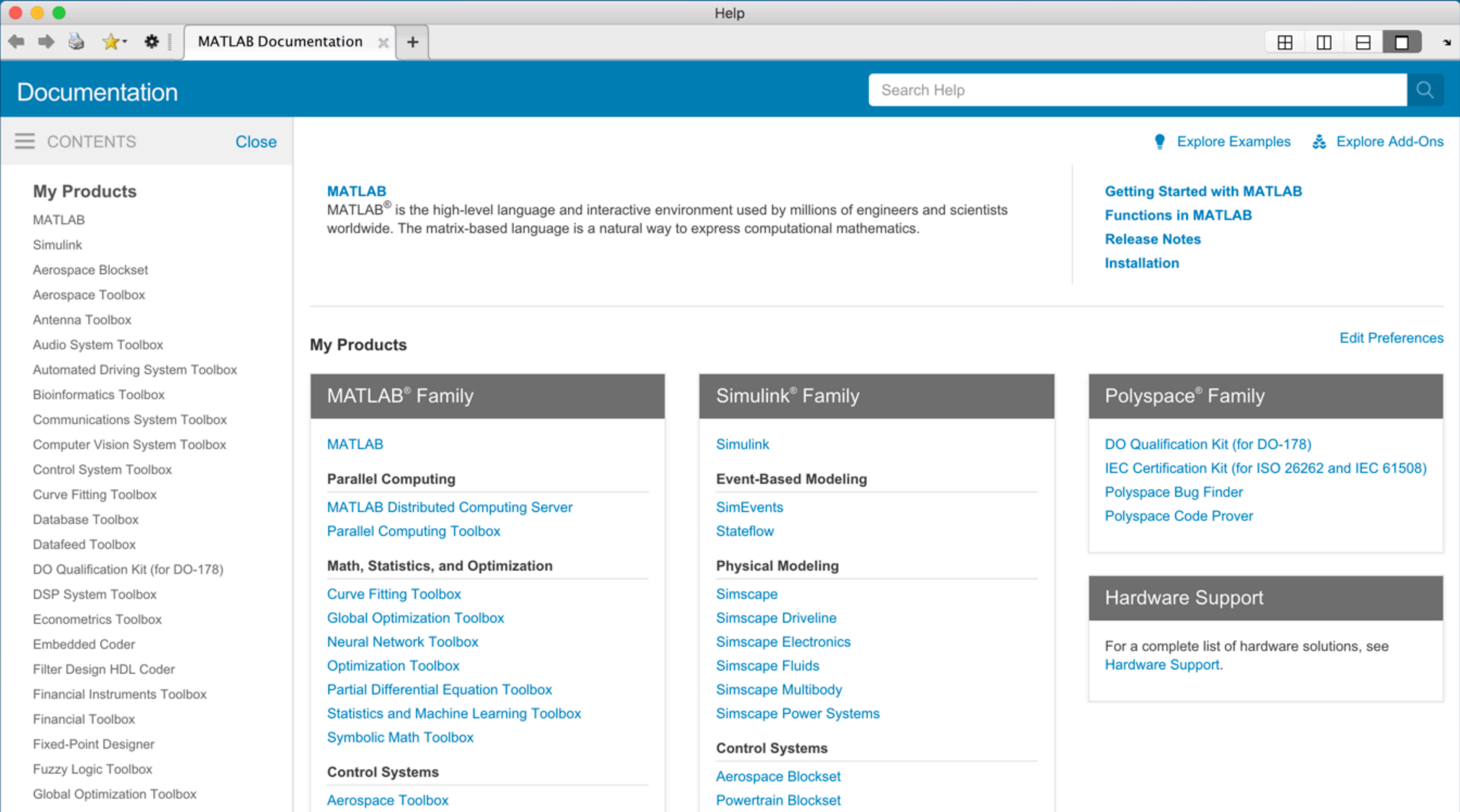


Muestra en una ventana externa toda la documentación de la función pasada como argumento.

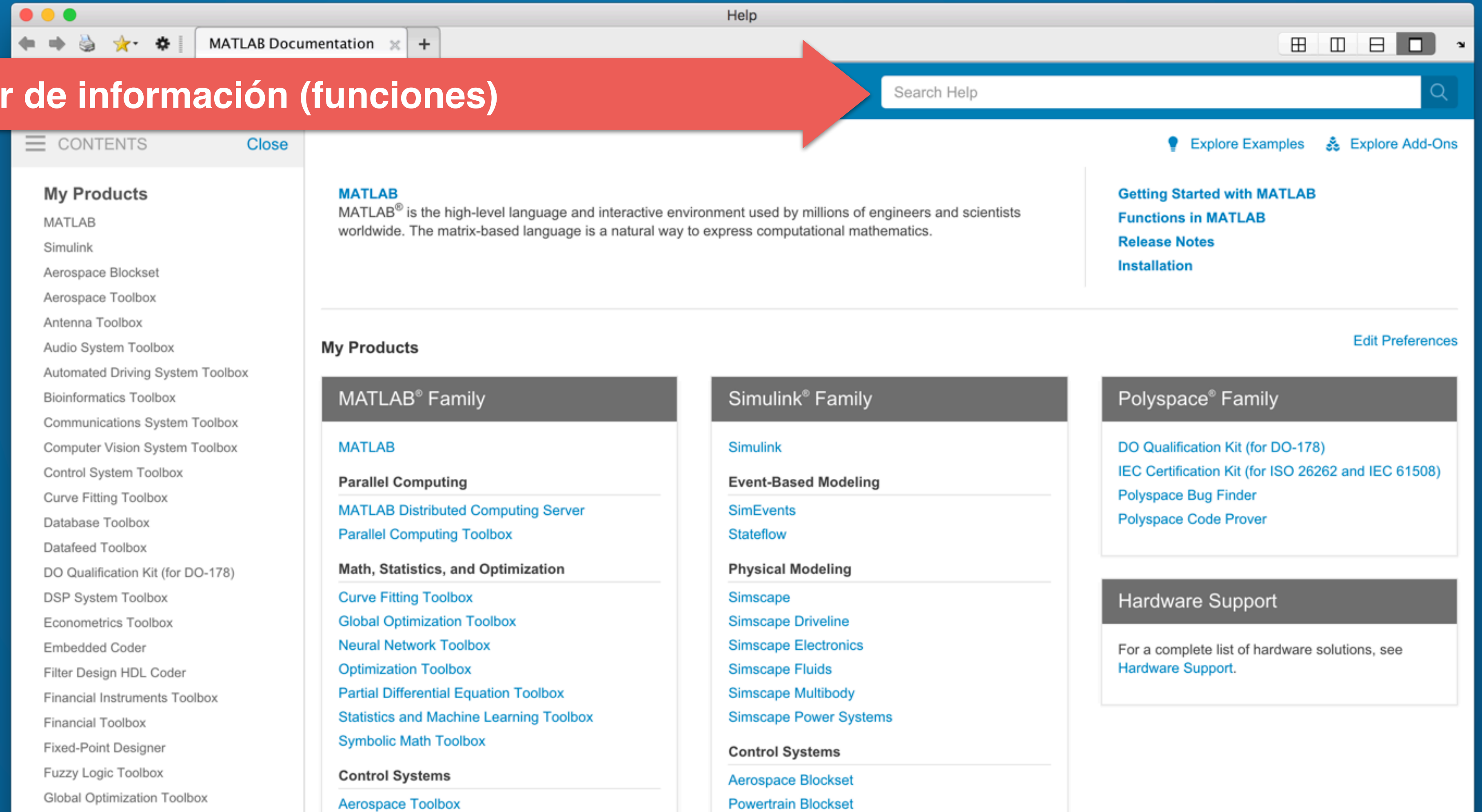
EJEMPLOS DE USO:

doc histogram





Buscador de información (funciones)



The screenshot shows the MATLAB Documentation Help window. A red arrow points from the text 'Buscador de información (funciones)' to the 'Search Help' input field in the top right corner of the window. The window has a sidebar on the left with a 'CONTENTS' menu and a list of products. The main area displays the 'MATLAB' product page, which includes a description of MATLAB and a 'My Products' section. This section is divided into three columns: 'MATLAB® Family', 'Simulink® Family', and 'Polyspace® Family'. Each column lists various toolboxes and software components. The 'MATLAB® Family' column includes links to MATLAB, Parallel Computing, Math, Statistics, and Optimization, and Control Systems. The 'Simulink® Family' column includes links to Simulink, Event-Based Modeling, Physical Modeling, and Control Systems. The 'Polyspace® Family' column includes links to DO Qualification Kit, IEC Certification Kit, Polyspace Bug Finder, and Polyspace Code Prover. There is also a 'Hardware Support' section at the bottom right.

Search Help

CONTENTS Close

My Products

- MATLAB
- Simulink
- Aerospace Blockset
- Aerospace Toolbox
- Antenna Toolbox
- Audio System Toolbox
- Automated Driving System Toolbox
- Bioinformatics Toolbox
- Communications System Toolbox
- Computer Vision System Toolbox
- Control System Toolbox
- Curve Fitting Toolbox
- Database Toolbox
- Datafeed Toolbox
- DO Qualification Kit (for DO-178)
- DSP System Toolbox
- Econometrics Toolbox
- Embedded Coder
- Filter Design HDL Coder
- Financial Instruments Toolbox
- Financial Toolbox
- Fixed-Point Designer
- Fuzzy Logic Toolbox
- Global Optimization Toolbox

MATLAB

MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists worldwide. The matrix-based language is a natural way to express computational mathematics.

Getting Started with MATLAB

- [Functions in MATLAB](#)
- [Release Notes](#)
- [Installation](#)

[Edit Preferences](#)

My Products

MATLAB® Family

- [MATLAB](#)
- Parallel Computing**
 - [MATLAB Distributed Computing Server](#)
 - [Parallel Computing Toolbox](#)
- Math, Statistics, and Optimization**
 - [Curve Fitting Toolbox](#)
 - [Global Optimization Toolbox](#)
 - [Neural Network Toolbox](#)
 - [Optimization Toolbox](#)
 - [Partial Differential Equation Toolbox](#)
 - [Statistics and Machine Learning Toolbox](#)
 - [Symbolic Math Toolbox](#)
- Control Systems**
 - [Aerospace Toolbox](#)

Simulink® Family

- [Simulink](#)
- Event-Based Modeling**
 - [SimEvents](#)
 - [Stateflow](#)
- Physical Modeling**
 - [Simscape](#)
 - [Simscape Driveline](#)
 - [Simscape Electronics](#)
 - [Simscape Fluids](#)
 - [Simscape Multibody](#)
 - [Simscape Power Systems](#)
- Control Systems**
 - [Aerospace Blockset](#)
 - [Powertrain Blockset](#)

Polyspace® Family

- [DO Qualification Kit \(for DO-178\)](#)
- [IEC Certification Kit \(for ISO 26262 and IEC 61508\)](#)
- [Polyspace Bug Finder](#)
- [Polyspace Code Prover](#)

Hardware Support

For a complete list of hardware solutions, see [Hardware Support](#).

Buscador de información (funciones)

Help

MATLAB Documentation

mean

CONTENTS

Close

My Products

MATLAB

Simulink

Aerospace Blockset

Aerospace Toolbox

Antenna Toolbox

Audio System Toolbox

Automated Driving System Toolbox

Bioinformatics Toolbox

Communications System Toolbox

Computer Vision System Toolbox

Control System Toolbox

Curve Fitting Toolbox

Database Toolbox

Datafeed Toolbox

DO Qualification Kit (for DO-178)

DSP System Toolbox

Econometrics Toolbox

Embedded Coder

Filter Design HDL Coder

Financial Instruments Toolbox

Financial Toolbox

Fixed-Point Designer

Fuzzy Logic Toolbox

Global Optimization Toolbox

MATLAB

MATLAB® is the high-level language and interactive environment used by millions of engineers and scientists worldwide. The matrix-based language is a natural way to express computational mathematics.

My Products

MATLAB® Family

MATLAB

Parallel Computing

MATLAB Distributed Computing Server

Parallel Computing Toolbox

Math, Statistics, and Optimization

Curve Fitting Toolbox

Global Optimization Toolbox

Neural Network Toolbox

Optimization Toolbox

Partial Differential Equation Toolbox

Statistics and Machine Learning Toolbox

Symbolic Math Toolbox

Control Systems

Aerospace Blockset

Aerospace Toolbox

Simulink® Family

Simulink

Event-Based Modeling

SimEvents

Stateflow

Physical Modeling

Simscape

Simscape Driveline

Simscape Electronics

Simscape Fluids

Simscape Multibody

Simscape Power Systems

Control Systems

Aerospace Blockset

Powertrain Blockset

mean

Functions

fx mean - Average or mean value of array MATLAB

fx mean - Mean value of timeseries data MATLAB

fx mean - Mean of probability distribution Statistics and Machine Learning Toolbox

fx mean - Average or mean value of fixed-point array Fixed-Point Designer

fx mean - Mean of probability distribution object Statistics and Machine Learning Toolbox

» 105 more

Blocks

Mean - Compute mean value of signal Simscape Power Systems

Mean - Find mean value of input or sequence of inputs DSP System Toolbox

Mean (Phasor) - Compute mean value of input phasor over a running window of one... Simscape Power Systems

Mean (Variable Frequency) - Compute mean value of signal Simscape Power Systems

RMS - Compute true root mean square (RMS) value of signal Simscape Power Systems

» 10 more

System Objects

dsp.RMS System object - Root mean square of vector elements DSP System Toolbox

dsp.Mean System object - Find mean value of input or sequence of inputs DSP System Toolbox

vision.Mean System object - Find mean value of input or sequence of inputs Computer Vision System Toolbox

dsp.MovingRMS System object - Moving Root Mean Square DSP System Toolbox

dsp.PeakToRMS System object - Peak-to-root-mean-square value of vector DSP System Toolbox

Buscador de información (funciones)

mean

Search Help

CONTENTS

Close

< Documentation Home

< MATLAB

< Data Import and Analysis

< Descriptive Statistics

< MATLAB

< Functions

mean

ON THIS PAGE

Syntax

Description

Examples

Input Arguments

More About

Extended Capabilities

See Also

mean

Average or mean value of array

Syntax

M = mean(A)

M = mean(A,dim)

M = mean(__ ,outtype)

M = mean(__ ,nanflag)

Description

M = mean(A) returns the mean of the elements of A along the first array dimension whose size does not equal 1.

If A is a vector, then mean(A) returns the mean of the elements.

If A is a matrix, then mean(A) returns a row vector containing the mean of each column.

If A is a multidimensional array, then mean(A) operates along the first array dimension whose size does not equal 1, treating the elements as vectors. This dimension becomes 1 while the sizes of all other dimensions remain the same.

M = mean(A,dim) returns the mean along dimension dim. For example, if A is a matrix, then mean(A,2) is a column vector containing the mean of each row.

M = mean(__ ,outtype) returns the mean with a specified data type, using any of the input arguments in the previous syntaxes. outtype can be 'default', 'double', or 'native'.

M = mean(__ ,nanflag) specifies whether to include or omit NaN values from the calculation for any of the previous syntaxes. mean(A,'includenan') includes all NaN values in the calculation while mean(A,'omitnan') ignores them.

Examples

Mean of Matrix Columns

Create a matrix and compute the mean of each column.

Buscador de información (funciones)

mean

+

Help

Search Help

CONTENTS

Close

< Documentation Home

< MATLAB

< Data Import and Analysis

< Descriptive Statistics

< MATLAB

< Functions

mean

ON THIS PAGE

Syntax

Description

Examples

Input Arguments

More About

Extended Capabilities

See Also

mean

Average or mean value of array

collapse all in page

Syntax

M = mean(A)

M = mean(A,dim)

M = mean(__ ,outtype)

M = mean(__ ,nanflag)

Description

M = mean(A)

returns the mean of the elements of A along the first array dimension whose size does not equal 1.

example

• If A is a vector, then mean(A) returns the mean of the elements.

• If A is a matrix, then mean(A) returns a row vector containing the mean of each column.

• If A is a multidimensional array, then mean(A) operates along the first array dimension whose size does not equal 1, treating the elements as vectors. This dimension becomes 1 while the sizes of all other dimensions remain the same.

M = mean(A,dim)

returns the mean along dimension dim. For example, if A is a matrix, then mean(A,2) is a column vector containing the mean of each row.

example

M = mean(__ ,outtype)

returns the mean with a specified data type, using any of the input arguments in the previous syntaxes. outtype can be 'default', 'double', or 'native'.

example

M = mean(__ ,nanflag)

specifies whether to include or omit NaN values from the calculation for any of the previous syntaxes. mean(A,'includenan') includes all NaN values in the calculation while mean(A,'omitnan') ignores them.

example

Examples

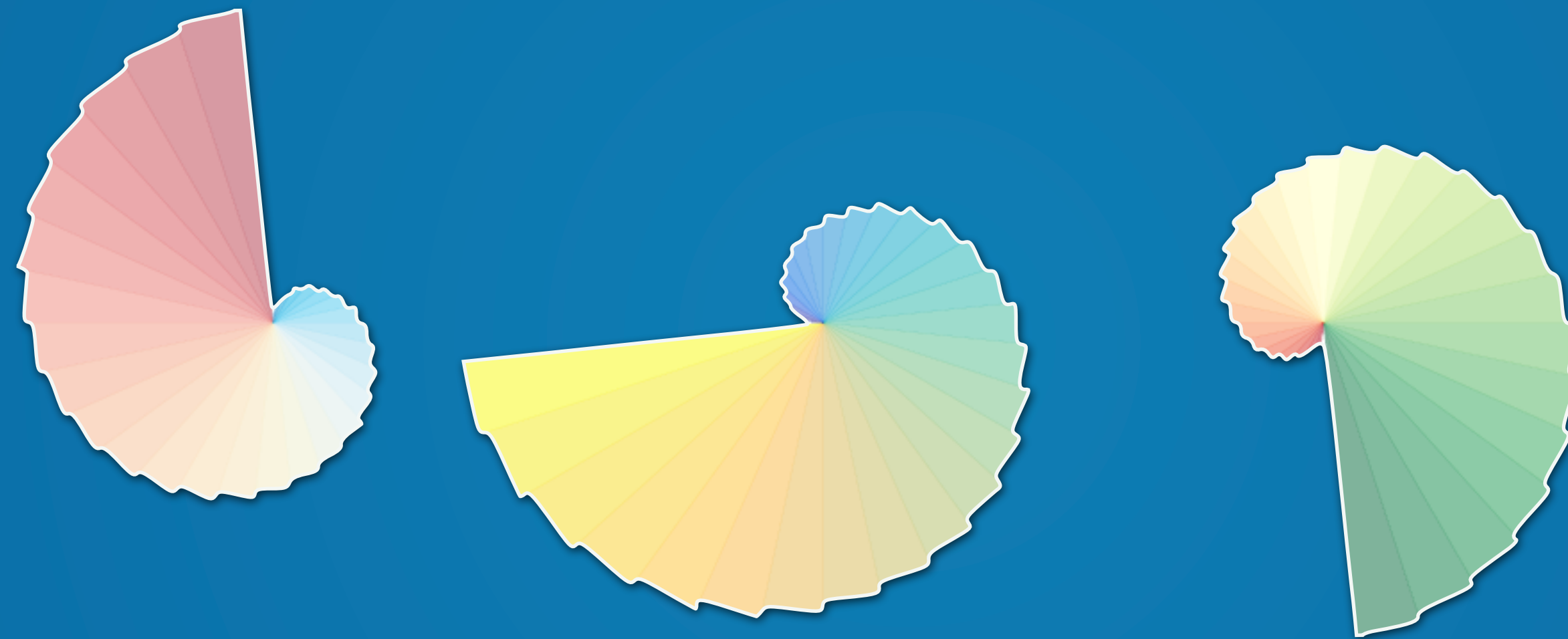
collapse all

Mean of Matrix Columns

David López García
Centro de Investigación de Mente Cerebro y Comportamiento
Universidad de Granada

cimcyc

39



Buenas prácticas

MATLAB Programming Style Guidelines

Richard Johnson

Version 1.5 October 2002
Copyright © 2002 Datatool

“Language is like a cracked kettle on which we beat tunes to dance to, while all the time we long to move the stars to pity.” Gustave Flaubert, in *Madame Bovary*

communications psychology

A Nature Portfolio journal

Primer



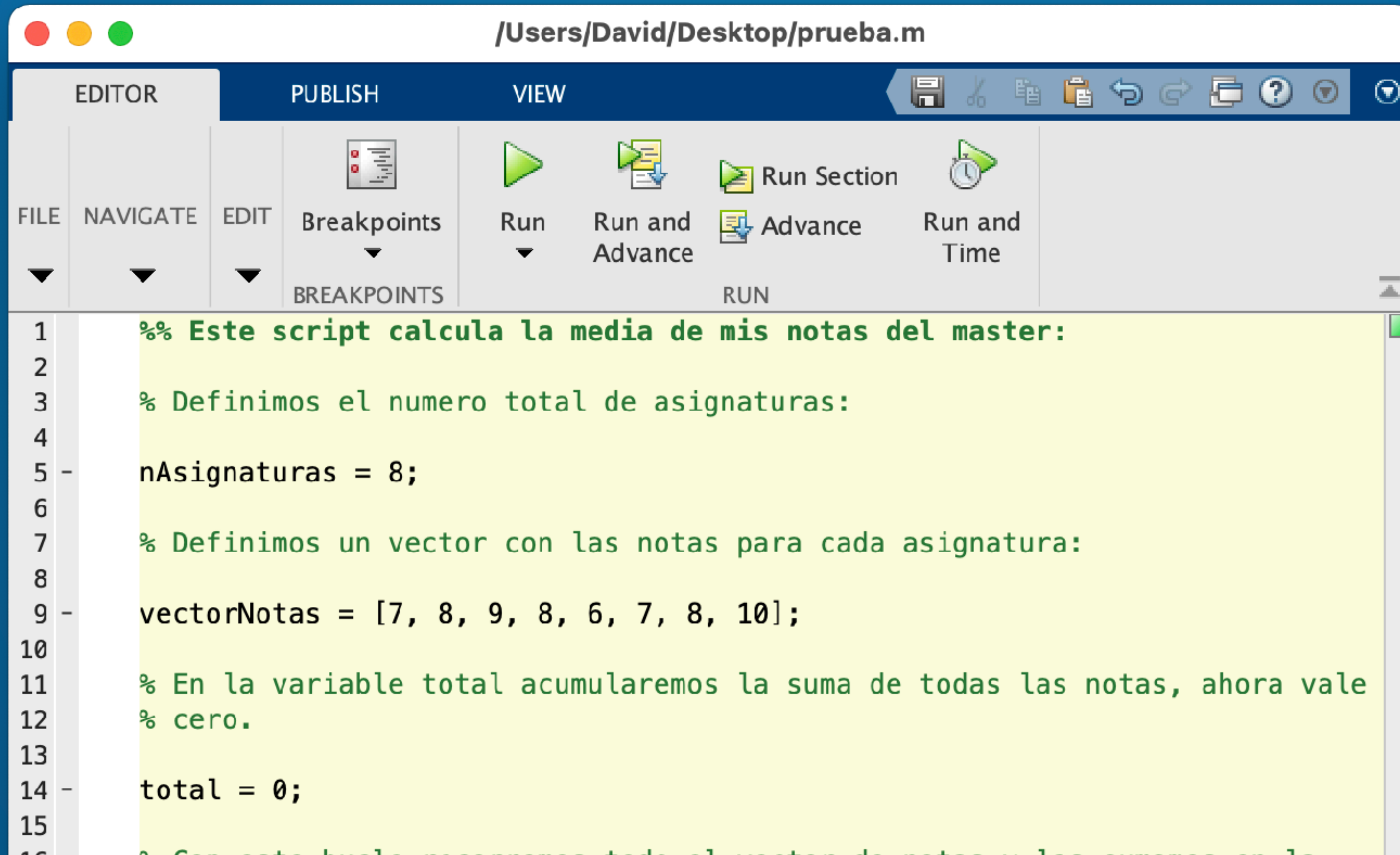
<https://doi.org/10.1038/s44271-025-00236-3>

Ten principles for reliable, efficient, and adaptable coding in psychology and cognitive neuroscience



Check for updates

Johannes Roth ^{1,2,6} , Yunyan Duan ^{3,6}, Florian P. Mahner^{1,4}, Philipp Kaniuth^{1,2},
Thomas S. A. Wallis ^{3,5,7} & Martin N. Hebart^{1,2,5,7}



```
1 %% Este script calcula la media de mis notas del master:
2
3 % Definimos el numero total de asignaturas:
4
5 nAsignaturas = 8;
6
7 % Definimos un vector con las notas para cada asignatura:
8
9 vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10];
10
11 % En la variable total acumularemos la suma de todas las notas, ahora vale
12 % cero.
13
14 total = 0;
15
16 % Con esta bucle recorremos todo el vector de notas y las sumamos en la
```

```
1 %% Este script calcula la media de mis notas del master:
2
3 % Definimos el numero total de asignaturas:
4
5 - nAsignaturas = 8;
6
7 % Definimos un vector con las notas para cada asignatura:
8
9 - vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10];
10
11 % En la variable total acumularemos la suma de todas las notas, ahora vale
12 % cero.
13
14 - total = 0;
15
16 % Con esta bucle recorremos todo el vector de notas y las sumamos en la
```


► Consejos ► Usar un correcto indentado

```
2
3 % Definimos el numero total de asignaturas:
4
5 - nAsignaturas = 8;
6
7 % Definimos un vector con las notas para cada asignatura:
8
9 - vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10];
10
11 % En la variable total acumularemos la suma de todas las notas, ahora vale
12 % cero.
13
14 - total = 0;
15
16 % Con este bucle recorreremos todo el vector de notas y las sumamos en la
17 % variable total:
18
19 - for i = 1 : nAsignaturas
20 -     notaAsignatura = vectorNotas(i);
21 -     total = total + notaAsignatura;
22 - end
23
24 % Calculamos la media dividiendo entre el total:
25 - media = total / nAsignaturas;
26
27
```

► Consejos ► Usar un correcto indentado

```
2
3 % Definimos el numero total de asignaturas:
4
5 - nAsignaturas = 8;
6
7 % Definimos un vector con las notas para cada
8
9 - vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10];
10
11 % En la variable total acumularemos la suma
12 % cero.
13
14 - total = 0;
15
16 % Con este bucle recorreremos todo el vector y
17 % variable total:
18
19 - for i = 1 : nAsignaturas
20 -     notaAsignatura = vectorNotas(i);
21 -     total = total + notaAsignatura;
22 - end
23
24 % Calculamos la media dividiendo entre el total:
25 - media = total / nAsignaturas;
26
27
```



► Consejos ► Usar un correcto indentado

```
2
3 % Definimos el numero total de asignaturas:
4
5 - nAsignaturas = 8;
6
7 % Definimos un vector con las notas para
8
9 - vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10];
10
11 % En la variable total acumularemos la su
12 % cero.
13
14 - total = 0;
15
16 % Con este bucle recorremos todo el vecto
17 % variable total:
18
19 - for i = 1 : nAsignaturas
20 -     notaAsignatura = vectorNotas(i);
21 -     total = total + notaAsignatura;
22 - end
23
24 % Calculamos la media dividiendo entre el total:
25 - media = total / nAsignaturas;
26
27
```

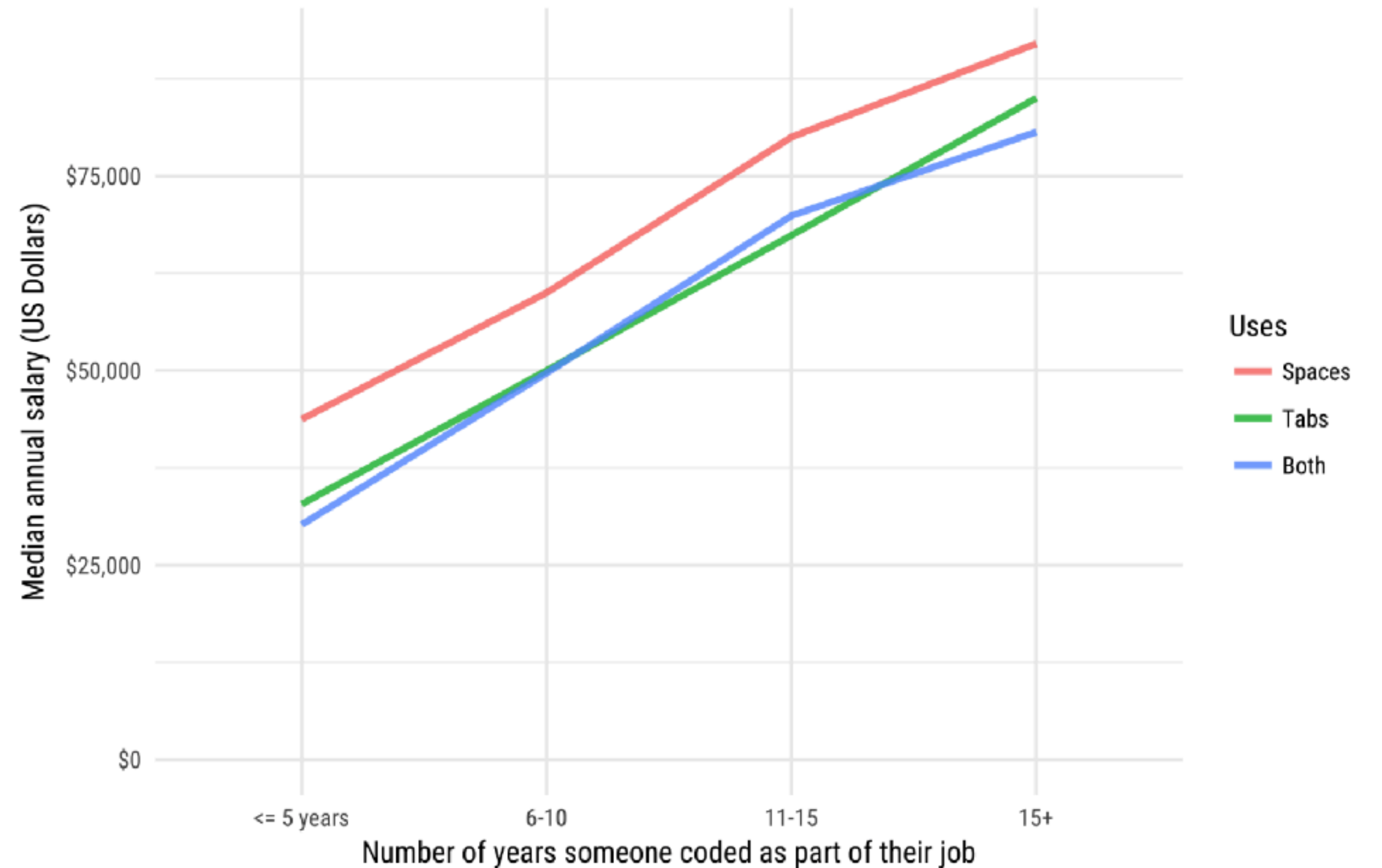


► Consejos ► Usar un correcto indentado

```
2
3 % Definimos el numero total de asignaturas
4
5 nAsignaturas = 8;
6
7 % Definimos un vector con las notas par
8
9 vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10]
10
11 % En la variable total acumularemos la
12 % cero.
13
14 total = 0;
15
16 % Con este bucle recorreremos todo el vec
17 % variable total:
18
19 for i = 1 : nAsignaturas
20     notaAsignatura = vectorNotas(i);
21     total = total + notaAsignatura;
22 end
23
24 % Calculamos la media dividiendo entre
25 media = total / nAsignaturas;
26
```

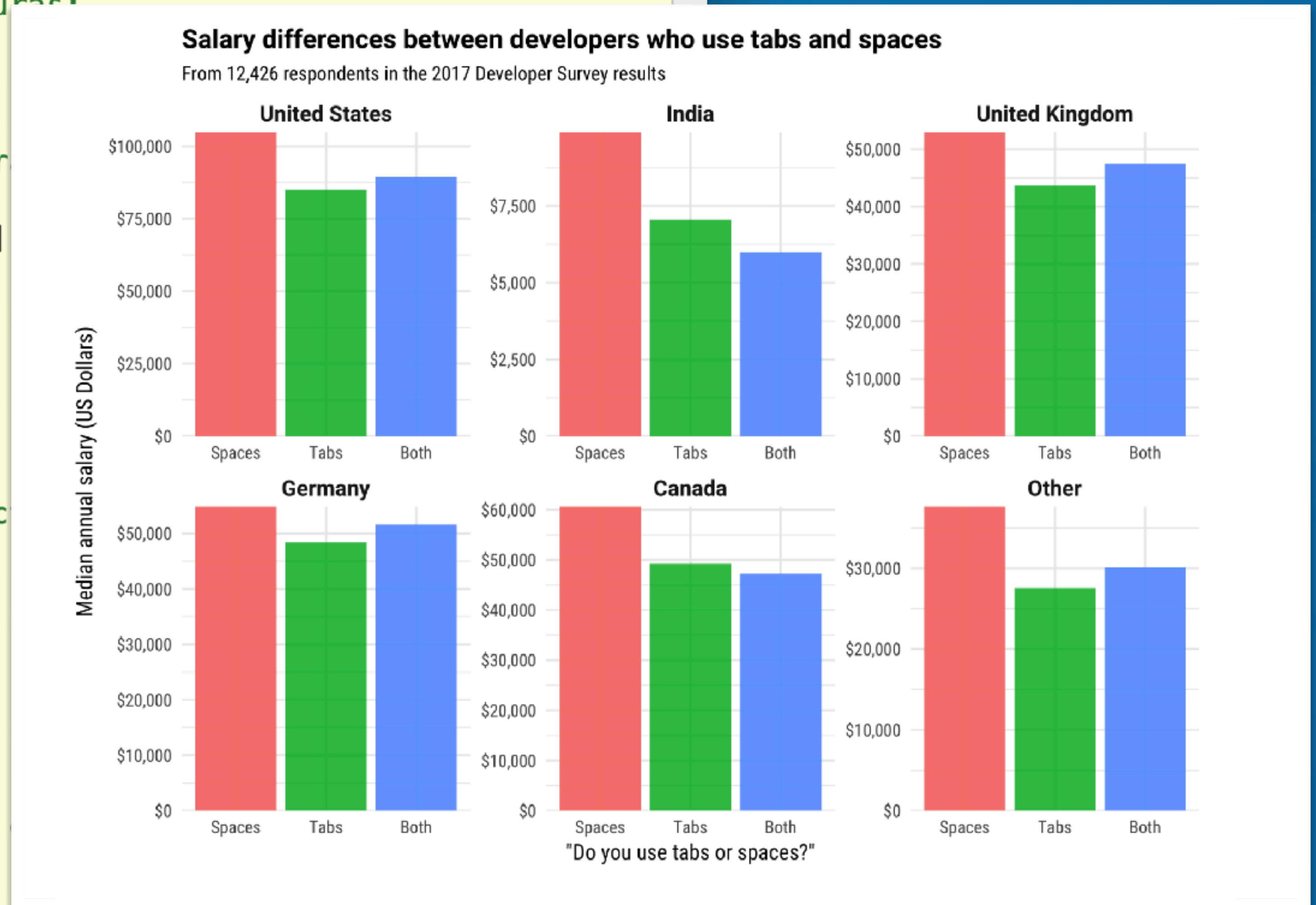
Salary differences between developers who use tabs and spaces

From 12,426 professional developers in the 2017 Developer Survey results, who provided tabs/spaces and salary



► Consejos ► Usar un correcto indentado

```
2
3 % Definimos el numero total de asignaturas
4
5 nAsignaturas = 8;
6
7 % Definimos un vector con las notas par
8
9 vectorNotas = [7, 8, 9, 8, 6, 7, 8, 10]
10
11 % En la variable total acumularemos la
12 % cero.
13
14 total = 0;
15
16 % Con este bucle recorremos todo el vec
17 % variable total:
18
19 for i = 1 : nAsignaturas
20     notaAsignatura = vectorNotas(i);
21     total = total + notaAsignatura;
22 end
23
24 % Calculamos la media dividiendo entre
25 media = total / nAsignaturas;
26
```





Depuración de código (*Debugging*)

6 STAGES OF DEBUGGING

1. That can't happen.
2. That doesn't happen
on my machine.
3. That shouldn't happen.
4. Why does that happen?
5. Oh, I see.
6. How did that ever work?



NUESTRO CÓDIGO NO FUNCIONA BIEN

(Y no sabemos por qué)



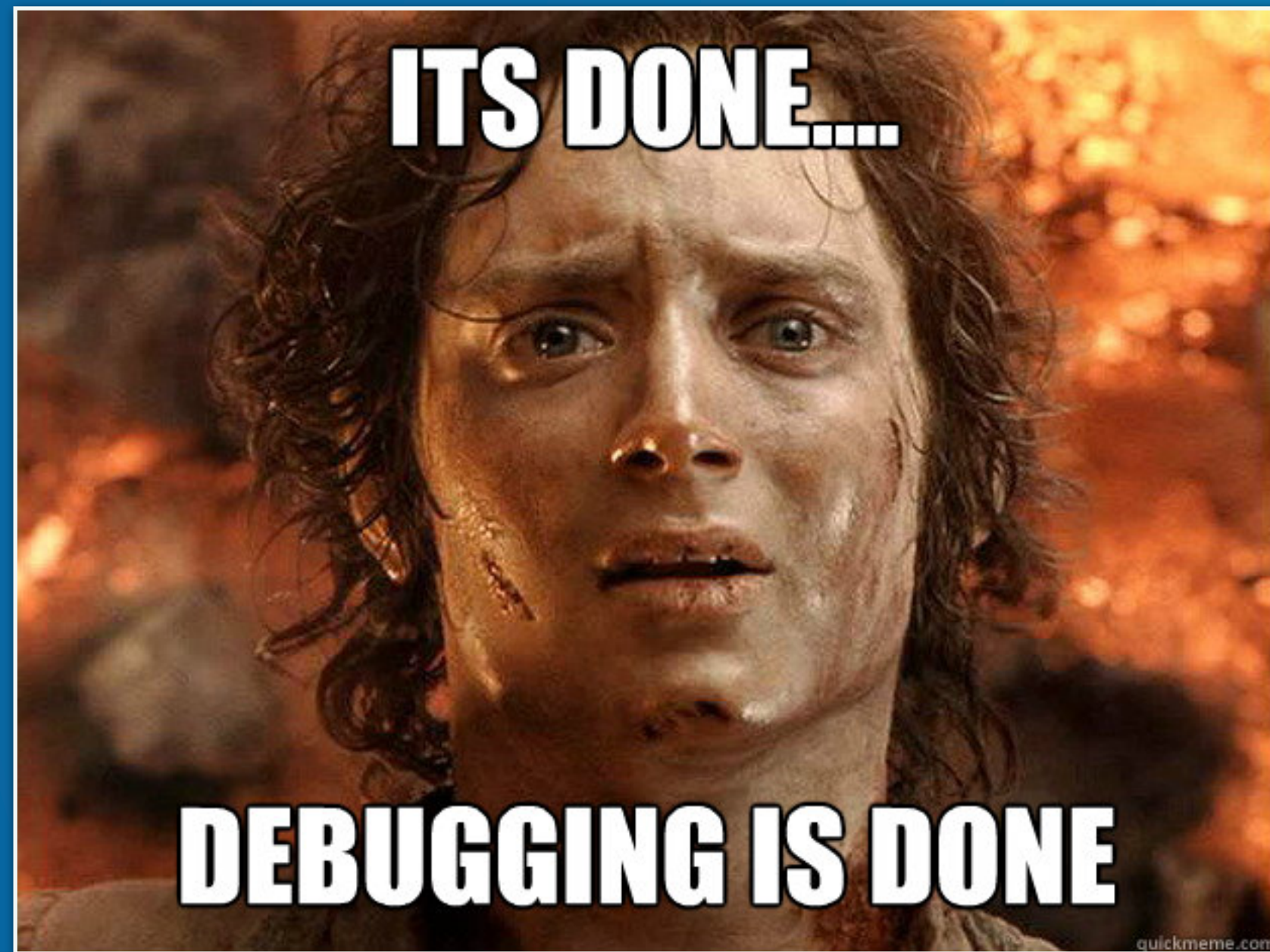
ERRORES QUE IMPIDEN SU EJECUCIÓN

(Y no sabemos por qué)



RESULTADOS NO ESPERADOS

(Y no sabemos por qué)



COMPROBAR LA LÍNEA DE EJECUCIÓN

Estructuras condicionales, bucles, etc.



COMPROBAR VALORES DE LAS VARIABLES

Que tengan los valores esperados



ENCONTRAR LA FUENTE DEL ERROR

Y resolverlo...

Name ▲	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentalInst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
practiceInst.m	●
ptb.m	●
repeatPracticeBlockInst.m	●
saveworkspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name ▲	Value
celIdePrueba	1x1 cell
matrizDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	'¡Hola mundo!'

conf.m	+
1	%% EVENT-RELATED POTENTIALS TEST (conf.m)
2	% -----
3	% David López García
4	% dlopez@ugr.es
5	% CIMCYC - Universidad de Granada
6	% -----
7	
8	%% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9	% -----
10 -	EPROP.nExpBlocks = 4;
11 -	EPROP.nPraBlocks = 4;
12	
13 -	EPROP.nExpTrials = 240;
14 -	EPROP.nPraTrials = 120;
15	
16	%% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17	% -----
18 -	EPROP.cueDuration = 1;
19 -	EPROP.stimDuration = 0.19;
20 -	EPROP.feedbackDuration = 1;
21 -	EPROP.respTime = 1.11;
22	
23	%% 3. EXPERIMENT'S PARAMETERS : COLORS
24	% -----
25 -	EPROP.topCueColor{1} = [231 76 60]/255;
26 -	EPROP.topCueColor{2} = [244 208 63]/255;
27 -	EPROP.bottomCueColor{1} = [46 204 113]/255;
28 -	EPROP.bottomCueColor{2} = [51 204 204]/255;
29 -	EPROP.feedback = [22 47 61]/255;

Command Window

```
>> variableDePrueba = '¡Hola mundo!';
>> matrizDePrueba = [1,2,3;4,5,6;7,8,9];
>> structDePrueba.data = [1 2 3 4 5 6 7 8 9];
```

Name ▲	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentalInst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
practiceInst.m	●
ptb.m	●
repeatPracticeBlockInst.m	●
saveworkspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name ▲	Value
celIdePrueba	1x1 cell
matrizDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	'¡Hola mundo!'

conf.m	+
1	%% EVENT-RELATED POTENTIALS TEST (conf.m)
2	% -----
3	% David López García
4	% dlopez@ugr.es
5	% CIMCYC - Universidad de Granada
6	% -----
7	
8	%% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9	% -----
10 -	EPROP.nExpBlocks = 4;
11 -	EPROP.nPraBlocks = 4;
12	
13	EPROP.nExpTrials = 240;
14 -	EPROP.nPraTrials = 120;
15	
16	%% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17	% -----
18 -	EPROP.cueDuration = 1;
19 -	EPROP.stimDuration = 0.19;
20 -	EPROP.feedbackDuration = 1;
21 -	EPROP.respTime = 1.11;
22	
23	%% 3. EXPERIMENT'S PARAMETERS : COLORS
24	% -----
25 -	EPROP.topCueColor{1} = [231 76 60]/255;
26 -	EPROP.topCueColor{2} = [244 208 63]/255;
27 -	EPROP.bottomCueColor{1} = [46 204 113]/255;
28 -	EPROP.bottomCueColor{2} = [51 204 204]/255;
29 -	EPROP.feedback = [22 47 61]/255;

Command Window

```
>> variableDePrueba = '¡Hola mundo!';
>> matrizDePrueba = [1,2,3;4,5,6;7,8,9];
>> structDePrueba.data = [1 2 3 4 5 6 7 8 9];
```




Name	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentalInst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
PracticeBlockInst.m	●
workspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name	Value
celIdePrueba	1x1 cell
matrizDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	'¡Hola mundo!'

```
conf.m
1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 - EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 - EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 - EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
27 - EPROP.bottomCueColor{1} = [46 204 113]/255;
28 - EPROP.bottomCueColor{2} = [51 204 204]/255;
29 - EPROP.feedback = [22 47 61]/255;
```

Command Window

```
>> variableDePrueba = '¡Hola mundo!';
>> matrizDePrueba = [1,2,3;4,5,6;7,8,9];
>> structDePrueba.data = [1 2 3 4 5 6 7 8 9];
```



Name	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentalInst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
PracticeBlockInst.m	●
workspace.m	●
session.m	●
trial.m	■

Details	
Workspace	
Name	Value
celIdePrueba	1x1 cell
matrizDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	'¡Hola mundo!'

```
conf.m
1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 - EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
27 - EPROP.bottomCueColor{1} = [46 204 113]/255;
28 - EPROP.bottomCueColor{2} = [51 204 204]/255;
29 - EPROP.feedback = [22 47 61]/255;
```

```
Command Window
>> variableDePrueba = '¡Hola mundo!';
>> matrizDePrueba = [1,2,3;4,5,6;7,8,9];
>> structDePrueba.data = [1 2 3 4 5 6 7 8 9];
```




Name	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentalInst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
PracticeBlockInst.m	●
workspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name	Value
celIdePrueba	1x1 cell
matrizDePrueba	[1 2 3;4 5 6;7 8 9]
structDePrueba	1x1 struct
variableDePrueba	'¡Hola mundo!'

```
conf.m
1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
27 - EPROP.bottomCueColor{1} = [46 204 113]/255;
28 - EPROP.bottomCueColor{2} = [51 204 204]/255;
29 - EPROP.feedbackColor = [22 47 61]/255;
```

Command Window

```
>> variableDePrueba = '¡Hola mundo!';
>> matrizDePrueba = [1,2,3;4,5,6;7,8,9];
>> structDePrueba.data = [1 2 3 4 5 6 7 8 9];
```

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Continue Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder

- Name
- Git
- func
- mexf
- blockCounter.m
- conf.m
- ctb.m
- ctbMatrix.mat
- experimental.m
- experimentallnst.m
- goodbye.m
- init.m
- instructions.m
- practice.m
- practiceInst.m
- ptb.m
- repeatPracticeBlockInst.m
- saveworkspace.m
- session.m
- trial.m

Details

Workspace

Name	Value
celldPrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3;4 5 6;7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```

1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 EPROP.nExpBlocks = 4;
11 EPROP.nPraBlocks = 4;
12
13 EPROP.nExpTrials = 240;
14 EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 EPROP.cueDuration = 1;
19 EPROP.stimDuration = 0.19;
20 EPROP.feedbackDuration = 1;
21 EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 EPROP.topCueColor{1} = [231 76 60]/255;
26 EPROP.topCueColor{2} = [244 208 63]/255;

```


MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Continue Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder

- Name
- Git
- func
- mexf
- blockCounter.m
- conf.m
- ctb.m
- ctbMatrix.mat
- experimental.m
- experimentallnst.m
- goodbye.m
- init.m
- instructions.m
- practice.m
- practiceInst.m
- ptb.m
- repeatPracticeBlockInst.m
- saveworkspace.m
- session.m
- trial.m

Details

Workspace

Name	Value
celldePrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3; 4 5 6; 7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```

1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 - EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 - EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 - EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;

```

Punto de ejecución

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Continuar hasta el siguiente breakpoint

Function Call Stack: conf Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder

- Name
- Git
- func
- mexf
- blockCounter.m
- conf.m
- ctb.m
- ctbMatrix.mat
- experimental.m
- experimentallnst.m
- goodbye.m
- init.m
- instructions.m
- practice.m
- practiceInst.m
- ptb.m
- repeatPracticeBlockInst.m
- saveworkspace.m
- session.m
- trial.m

Details

Workspace

Name	Value
celldPrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3;4 5 6;7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```
1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 - EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 - EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 - EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
```

Punto de ejecución

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

+ New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Continue Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

/ Users davidlopezgarcia Documents MATLAB PSYCHTOOLBOX ERP_PRUEBA src

Current Folder

Name	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentallnst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
practiceInst.m	●
ptb.m	●
repeatPracticeBlockInst.m	●
saveworkspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name	Value
celldPrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3;4 5 6;7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```

1  %% EVENT-RELATED POTENTIALS TEST (conf.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9  % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
  
```

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Continue Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder: /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src

Workspace:

Name	Value
celldPrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3; 4 5 6; 7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```

1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;

```

Punto de ejecución

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Continuar hasta la siguiente linea

Print Find Indent Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder

- Name
- Git
- func
- mexf
- blockCounter.m
- conf.m
- ctb.m
- ctbMatrix.mat
- experimental.m
- experimentallnst.m
- goodbye.m
- init.m
- instructions.m
- practice.m
- practiceInst.m
- ptb.m
- repeatPracticeBlockInst.m
- saveworkspace.m
- session.m
- trial.m

Details

Workspace

Name	Value
celldePrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3;4 5 6;7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```
1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
```

Punto de ejecución

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

+ New Open Save Find Files Compare Print Go To Find Insert Comment Indent Breakpoints Continue Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

/ Users / davidlopezgarcia / Documents / MATLAB / PSYCHTOOLBOX / ERP_PRUEBA / src

Current Folder

Name	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentallnst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
practiceInst.m	●
ptb.m	●
repeatPracticeBlockInst.m	●
saveworkspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name	Value
celldPrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3; 4 5 6; 7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```

1  %% EVENT-RELATED POTENTIALS TEST (conf.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9  % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - ➔ EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
  
```


MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Find Files Compare Go To Find Insert Comment Indent Breakpoints Continue Step Step In Step Out Run to Cursor Function Call Stack: conf Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder

Name	Git
func	.
mexf	.
blockCounter.m	●
conf.m	●
ctb.m	●
ctbMatrix.mat	●
experimental.m	■
experimentallnst.m	●
goodbye.m	●
init.m	●
instructions.m	●
practice.m	■
practiceInst.m	●
ptb.m	●
repeatPracticeBlockInst.m	●
saveworkspace.m	●
session.m	●
trial.m	■

Details

Workspace

Name	Value
celldPrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3; 4 5 6; 7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```

1  %% EVENT-RELATED POTENTIALS TEST (conf.m)
2  % -----
3  % David López García
4  % dlopez@ugr.es
5  % CIMCYC - Universidad de Granada
6  % -----
7
8  %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9  % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - ➔ EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;

```

Punto de ejecución

MATLAB R2017a

HOME PLOTS APPS EDITOR PUBLISH VIEW

Salir del modo de depuración

Print Find Indent Run to Cursor Quit Debugging

FILE NAVIGATE EDIT BREAKPOINTS DEBUG

Current Folder

- Name
- Git
- func
- mexf
- blockCounter.m
- conf.m
- ctb.m
- ctbMatrix.mat
- experimental.m
- experimentallnst.m
- goodbye.m
- init.m
- instructions.m
- practice.m
- practiceInst.m
- ptb.m
- repeatPracticeBlockInst.m
- saveworkspace.m
- session.m
- trial.m

Details

Workspace

Name	Value
celldePrueba	1x1 cell
EPROP	1x1 struct
matrizDePrueba	[1 2 3;4 5 6;7 8 9]

Editor - /Users/davidlopezgarcia/Documents/MATLAB/PSYCHTOOLBOX/ERP_PRUEBA/src/conf.m

```
1 %% EVENT-RELATED POTENTIALS TEST (conf.m)
2 % -----
3 % David López García
4 % dlopez@ugr.es
5 % CIMCYC - Universidad de Granada
6 % -----
7
8 %% 1. EXPERIMENT'S PARAMETERS : BLOCKS AND TRIALS PER BLOCK
9 % -----
10 - EPROP.nExpBlocks = 4;
11 - EPROP.nPraBlocks = 4;
12
13 ● EPROP.nExpTrials = 240;
14 - EPROP.nPraTrials = 120;
15
16 %% 2. EXPERIMENT'S PARAMETERS : DURATIONS
17 % -----
18 - EPROP.cueDuration = 1;
19 - EPROP.stimDuration = 0.19;
20 ● EPROP.feedbackDuration = 1;
21 - ➔ EPROP.respTime = 1.11;
22
23 %% 3. EXPERIMENT'S PARAMETERS : COLORS
24 % -----
25 ● EPROP.topCueColor{1} = [231 76 60]/255;
26 - EPROP.topCueColor{2} = [244 208 63]/255;
```

Punto de ejecución

MATLAB® & SIMULINK®

Presentación por: **David López García**