



Scilab

INITIATION – NIVEAU 1

FANNY CLEMENT

COURBON BENOIT

EL RHAYTI HOUARIYA

GAUDIN NICOLAS

I. Présentation de l'outil

- Logiciel libre
- Version open-source de Matlab
- Langage de programmation interprété
- Langage non typé

Les avantages :

- Gratuit
- Intuitif & interactif (même logique que la programmation sur R)
- Environnement de calcul pour des applications scientifiques

Les inconvénients :

- Difficulté de mise en œuvre des méthodes statistiques complexes
- Temps d'exécution (langage interprété)
- Des packages ne fonctionnent pas sur la version 6.0.0

II. Installation

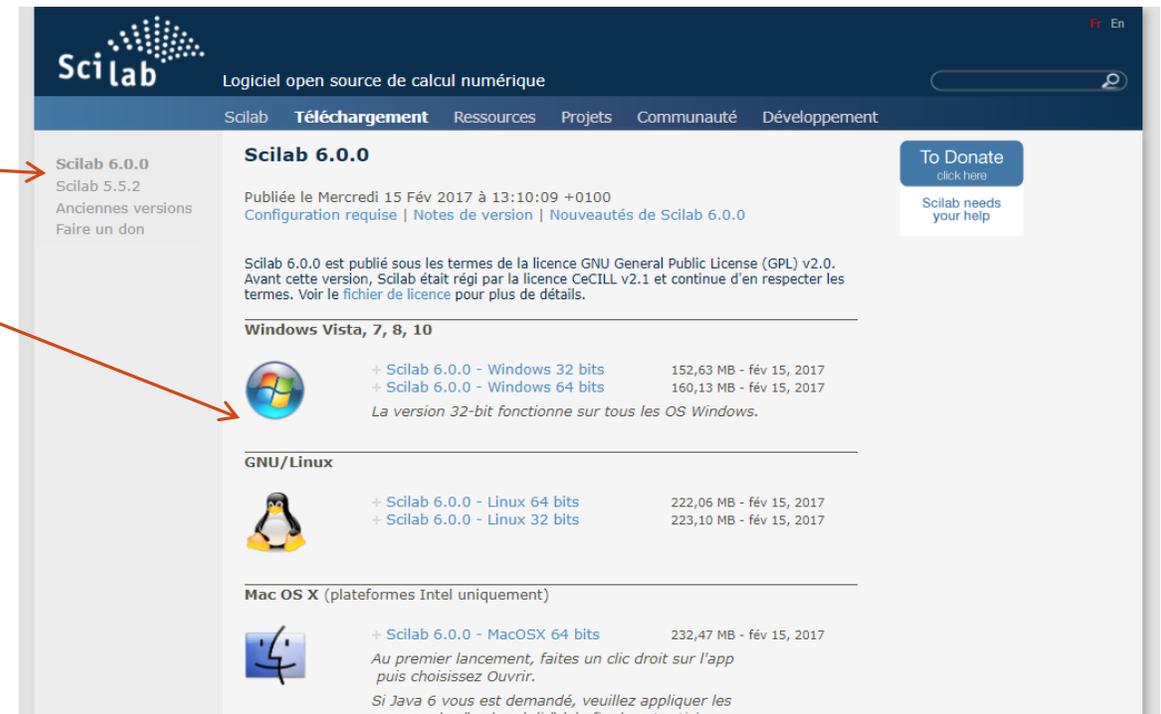
Sur le site de Scilab (<https://www.scilab.org/fr>) , cliquer sur l'onglet Téléchargements.

Plusieurs versions sont possibles :

- V 6.0.0
- V 5.5.2

Choisissez ensuite sur quel environnement vous travaillez

Une inscription est parfois demandée, remplissez les champs et le téléchargement peut commencer.



The screenshot shows the Scilab website's download page for version 6.0.0. The page is in French and features a dark blue header with the Scilab logo and navigation links: Scilab, Téléchargement, Ressources, Projets, Communauté, and Développement. A search bar is visible in the top right corner. The main content area is divided into sections for different operating systems: Windows Vista, 7, 8, 10; GNU/Linux; and Mac OS X (Intel only). Each section lists available download packages with their file sizes and release dates. A sidebar on the left contains links for Scilab 6.0.0, Scilab 5.5.2, and a 'Faire un don' button. A 'To Donate' button is also present in the top right. Two red arrows point from the text on the left to the 'Téléchargement' menu item and the Windows download section.

OS	Package	Size	Release Date
Windows Vista, 7, 8, 10	+ Scilab 6.0.0 - Windows 32 bits	152,63 MB	fév 15, 2017
	+ Scilab 6.0.0 - Windows 64 bits	160,13 MB	fév 15, 2017
GNU/Linux	+ Scilab 6.0.0 - Linux 64 bits	222,06 MB	fév 15, 2017
	+ Scilab 6.0.0 - Linux 32 bits	223,10 MB	fév 15, 2017
Mac OS X (plateformes Intel uniquement)	+ Scilab 6.0.0 - MacOSX 64 bits	232,47 MB	fév 15, 2017

III. Organisation du logiciel

Console Scilab 6.0.0

```
--> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\Exemple.sce', -1)
```

somme

13.

-->

Nom	Valeur	Type	Visibilité
FDR_theo	1x100	Double	local
X	[-0.745, -0.58...	Double	local
a	1	Double	local
ans	[1, 5; 4, 3]	Double	local
b	12	Double	local
densite_theo	1x200	Double	local
ech	1x100	Double	local
ech_tri	1x100	Double	local
histo	1x10	Double	local
m	[-2.58, 0; 0, 6...	Double	local

Historique des commandes

```
-- 23/10/2017 08:56:03 --
```

News feed
News feed unavailable.

Console : traitements et affichage des résultats

Exécution du script

Exemple.sce (C:\Users\NicolasGaudin\Desktop\M2 SISE...

Fichier Édition Format Options Fenêtre Exécuter ?

```
1 //Exemple de script
2 a = 1
3 b = 12
4 //Calcul de la somme
5 somme = a + b
6 //Affichage
7 disp(somme, 'somme')
```

On peut aussi utiliser la commande *exec*

IV. Les bases du langage

Les commandes de la console :

- `quit()` : quitter le logiciel
- `help` : ouvrir navigateur d'aide
- `editor()` : ouvrir éditeur
- `disp(objet)` : afficher objet
- `variable = input(« Saisir la variable »)` : saisie d'une variable
- `//` et `/* ... */` : ajouter un commentaire
- `=` : réaliser une affectation
- `clear objet` : supprimer un objet
- `exec('chemin_fichier.sce',-1)` : exécuter un script
“-1” permet d'afficher seulement les “disp”
- `clc` : effacer la console
- `typeof(variable)` : connaître le type d'une variable

Les opérateurs de comparaison : `==` , `>` , `>=` , `<` , `<=` , `<>`

Les opérateurs algébriques : `+` , `-` , `*` , `/` , `^`

Les opérateurs logiques :

- `%t` : variable booléenne « true »
- `%f` : variable booléenne « false »
- `~objet` (négation), `&` (et) , `|` (ou)

V. Les branchements conditionnels

	Syntaxe	Exemple	Résultat
IF	<pre> if (<i>condition</i>) then (<i>traitement</i>) [elseif (<i>condition</i>) then (<i>traitement</i>)] else (<i>traitement</i>) end </pre>	<pre> n = input(« Saisir un nombre : ») if (n == 0) then disp(« Le nombre saisi est nul. ») elseif (n > 0) then disp(« Le nombre saisi est positif. ») else disp(« Le nombre saisi est négatif. ») end </pre>	<pre> --> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\exemple2.sce', -1) Saisir un nombre : 10 Le nombre saisi est positif. </pre>
CASE	<pre> select (<i>variable</i>), case (<i>expr1</i>) then (<i>instructions1</i>), case (<i>expr2</i>) then (<i>instructions2</i>), ... case (<i>exprn</i>) then (<i>instructions</i>), [else (<i>instructions</i>)], end </pre>	<pre> n = input(« Saisir un entier entre 1 et 3 : ») select n case 1 then disp(« Vous avez choisi le 1. ») case 2 then disp(« Vous avez choisi le 2. ») case 3 then disp(« Vous avez choisi le 3. ») else disp(« Vous n'avez rien compris ! ») end </pre>	<pre> --> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\exemple2.sce', -1) Saisir un entier entre 1 et 3 : 5 Vous n'avez rien compris ! </pre>

VI. Les boucles

	FOR	WHILE
Syntaxe	<code>for indice = séquence instructions end</code>	<code>Initialisation_indice while condition instruction(s) [break] end</code>
Exemple	<code>for i = 1:5 disp(i) end</code>	<code>i = 1 while i < 6 disp(i) i = i + 1 //Incrémentation end</code>

```
--> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\exemple2.sce', -1)
1.
2.
3.
4.
5.
```

Séquences

- `1:7` → 1. 2. 3. 4. 5. 6. 7.
- `[1, 3, 8, 10]` → 1. 3. 8. 10.
- `n1:n2:n3` → Renvoie les valeurs de `n1` à `n3` par pas de `n2`
 - `1:2:10` → 1. 3. 5. 7. 9. (pas ascendant)
 - `10:-2:1` → 10. 8. 6. 4. 2. (pas descendant)

VII. Les fonctions

Syntaxe	<pre>function [<i>sortie(s)</i>] = <i>nomfunction</i>(<i>paramètre(s)</i>) Traitement(s) endfunction</pre> <p>//Appel de la fonction <i>[sortie(s)]</i> = <i>nomfunction</i>(<i>paramètres(s)</i>)</p>
Exemple	<pre>// Fonction de calcul de la somme et de la différence function [<i>s</i>, <i>d</i>] = <i>calcul</i> (<i>a</i>, <i>b</i>) <i>s</i> = <i>a</i> + <i>b</i> <i>d</i> = <i>a</i> - <i>b</i> endfunction</pre> <p>// Appel de la fonction <i>a</i> = 5; <i>b</i> = 10 <i>[somme, diff]</i> = <i>calcul</i>(<i>a</i> , <i>b</i>) disp(somme, « somme ») disp(diff, « différence »)</p> <div data-bbox="1286 921 2440 1245" style="border: 1px solid black; padding: 5px;"><pre>--> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\exemple2.sce', -1) somme 15. différence -5.</pre></div>

VIII. Les vecteurs

Création d'un vecteur

$v = [1\ 2\ 3\ 4\ 5\ 6]$ ou $v = [1,2,3,4,5,6]$
 $v = 1:6$ ou $v = \text{linspace}(1, 6, 6)$

Affectations, ajout, modification

$v(2) = 10$
 $v = [v, 10]$

Opérations sur le vecteur

$\text{length}(v)$ //Nombre d'éléments
 $\text{sum}(v)$, $\text{mean}(v)$, $\text{min}(v)$... //somme, moyenne...

Sélections

$v(2)$ //2^{ème} élément
 $v(2:4)$ //du 2^{ème} au 4^{ème} élément
 $v(\text{condition}(s))$ //selon une condition

Calculs entre 2 vecteurs

$\text{somme} = v1 + v2$
 $\text{mult_termes} = v1 .* v2$
 $\text{prod_scal} = v1 * v2'$

```
1 //Création
2 v = 1:8 //1-2-3-4-5-6-7-8
3 u = [5-6-7-8-9-10-11-12]
4
5 //Nombre d'éléments
6 long = length(v) //8
7
8 //Modification
9 v(8) = 10 //v = 1-2-3-4-5-6-7-10
10
11 //Somme des éléments
12 s = sum(v) //s = 38
13
14 //Accès par condition
15 v2 = v(v>5) //v2 = 5-6-7-10
16
17 //Calculs entre 2 vecteurs
18 somme = u + v //somme = 6-8-10-12-14-16-18-22
19 mult_terme = u .* v //mult_terme = 5-12-21-32-45-60-77-120
20 prod_scal = u * v' //prod_scal = 372
```

IX. Les matrices

Création d'une matrice

```
m = [1 2 3; 4 5 6]
m = matrix( [1 2 3 4 5 6], 2, 3)
```

1.	2.	3.
4.	5.	6.

Affectations, ajout, modification

```
m(2, 1) = 10
m = matrix(m, 3, 2) //Changement des dimensions de m
```

Opérations sur la matrice

```
length(m) //Nombre d'éléments
size(m) //Dimensions
size(m, « r ») //Nombre de lignes (« c » pour les colonnes)
sum(m, « c »), mean(m)... //somme par colonne, moyenne des termes
```

Sélections

```
v(2,1) //élément de la 2ème ligne et 1ère colonne
v(:, 2) //de la 2ème colonne
v(:, $) //de la dernière colonne
v(condition1, :) //selon une condition sur les lignes
```

Transposé, déterminant et diagonale

```
transpose = m'
det = det(m)
d = diag(m)
```

Matrice nulle, Matrice Identité

```
mat_nulle = zeros(nbl, nbc)
mat_id = eye(nbl, nbc)
```

Inversion et diagonalisation

```
inverse = inv(m)
[valp, vectp] = bdiag(m)
```

Calculs entre 2 matrices

```
somme = m1 + m2
mult_termes = m1 .* m2
prod_mat = m1 * m2'
```

VIII. Statistique

Indicateurs statistiques

moyenne = **mean**(objet [, « r »/ « c »])

mediane = **median**(objet [, « r »/ « c »])

ecart_type = **stdev**(objet [, « r »/ « c »])

variance = **variance** (objet [, « r »/ « c »])

//Calcul de la variance sans biais (1/N-1 ...)

somme = **sum**(objet [, « r »/ « c »])

X = [2 8 0 3 7 6 8 7 9 1 6 7 7 2 5 2 2 2 9 7]

Fonctions tabul et cumsum

Tabul compte les effectifs de chaque modalité dans un vecteur/ une matrice

Cumsum calcule les effectifs cumulés dans un vecteur

//Effectifs

--> m1 = tabul(X)

m1 =

9. 2.

8. 2.

7. 5.

6. 2.

5. 1.

3. 1.

2. 5.

1. 1.

0. 1.

//Modalités triées

--> m2 = tabul(X, "i")

m2 =

0. 1.

1. 1.

2. 5.

3. 1.

5. 1.

6. 2.

7. 5.

8. 2.

9. 2.

//Somme cumulée sur m2

--> effc=cumsum(m2(:,2))

effc =

1.

2.

7.

8.

9.

11.

16.

18.

20.

//Calcul de fréquences cumulées

--> frec=effc/sum(m2(:,2))

frec =

0.05

0.1

0.35

0.4

0.45

0.55

0.8

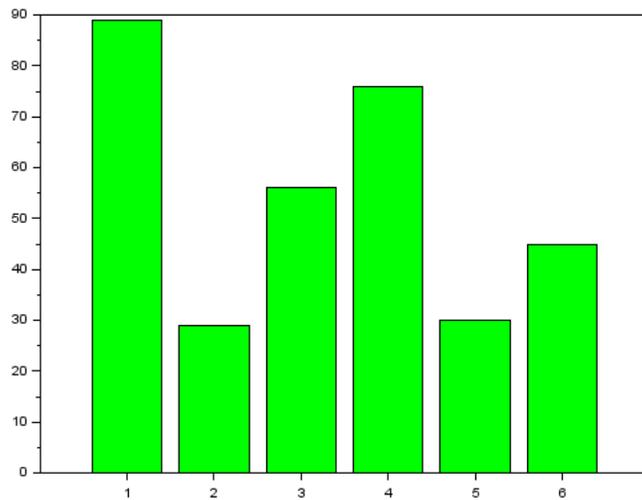
0.9

1.

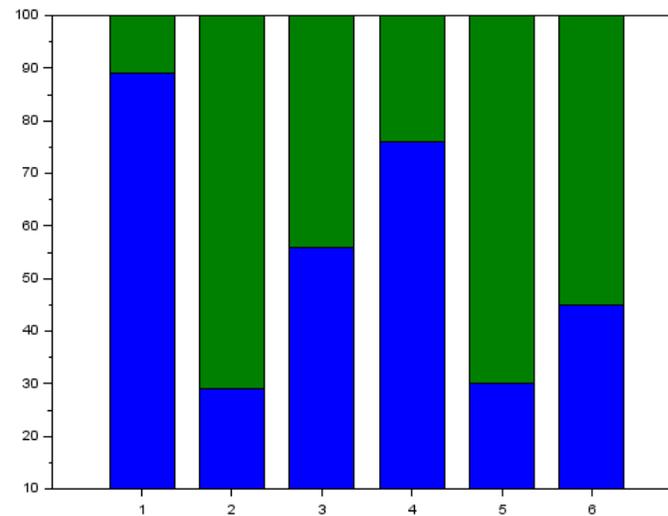
X. Graphiques

$$a = \begin{bmatrix} 1 & 89 & 11 \\ 2 & 29 & 71 \\ 3 & 56 & 44 \\ 4 & 76 & 24 \\ 5 & 30 & 70 \\ 6 & 45 & 55 \end{bmatrix}$$

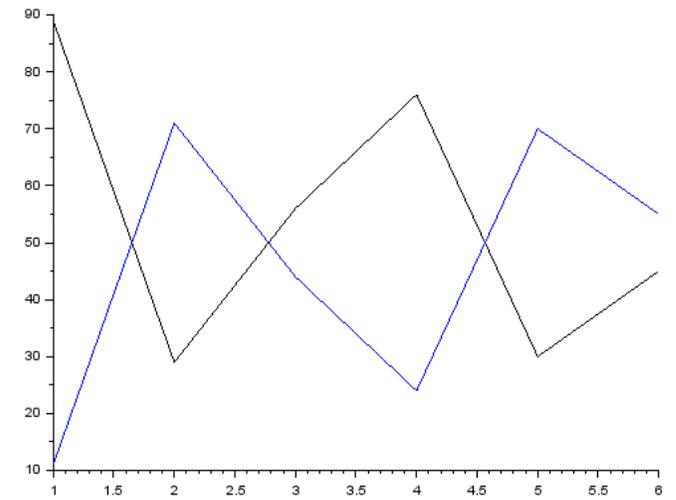
➤ `bar(a(:,1),a(:,2),'green')`



➤ `bar(a(:,1),a(:,2:3),'stacked')`

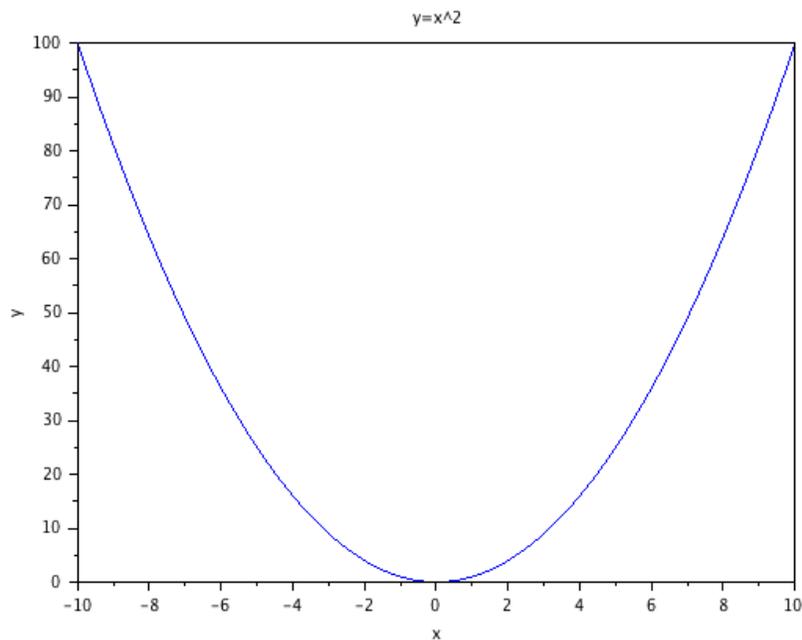


➤ `plot2d(a(:,1),a(:,2:3))`



X. Graphiques

- `x=linspace(-10,10,100)`
- `y=f(x)`
- `plot(x,y)`



- `x=linspace(-10,10,100)`
- `y=linspace(-10,10,100)`
- `z=feval(x,y,f)`
- `surf(x,y,z)`

