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

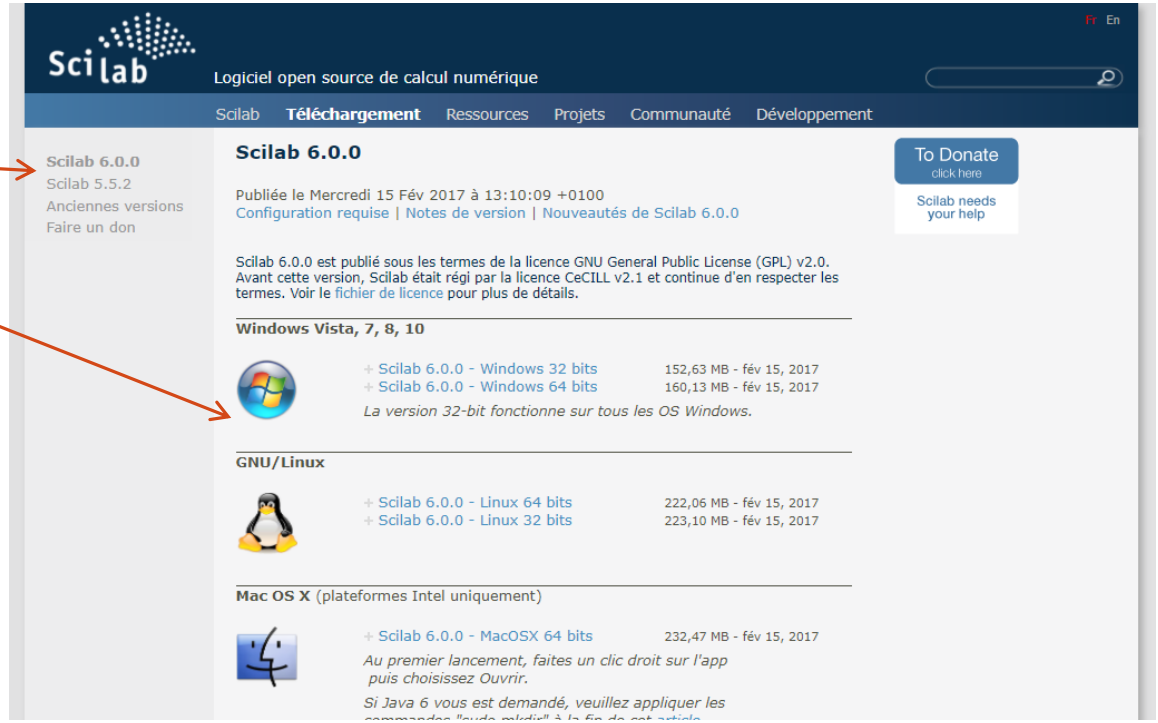
II. Installation

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

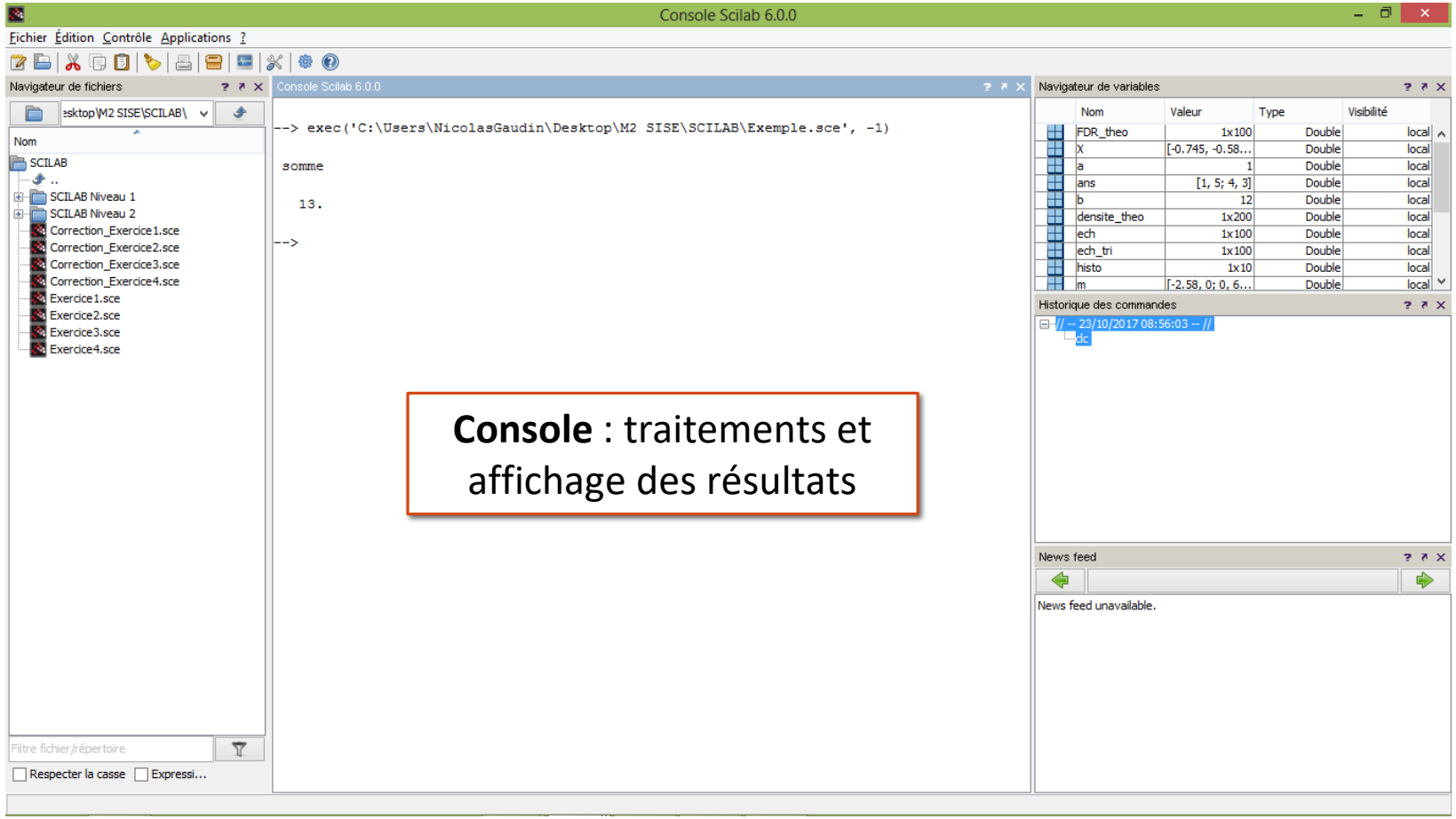
Plusieurs versions sont possibles :
- V 6.0.0
- V 5.5.2

Choisissez ensuite sur quel environnement vous travaillez

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

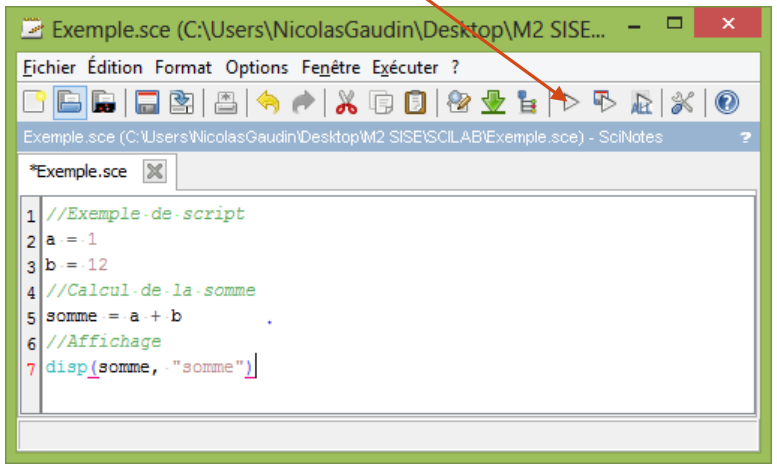


III. Organisation du logiciel



Console : traitements et affichage des résultats

Exécution du **script**



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	**if** (*condition*) **then**
	(*traitement*)
	[**elseif** (*condition*) **then**
	(*traitement*)]
	else
	(*traitement*)
	end
		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
			--> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\exemple2.sce', -1)
			Saisir un nombre :
			10
			
			Le nombre saisi est positif.

CASE	**select** (*variable*), **case** (*expr1*) **then** (*instructions1*), **case** (*expr2*) **then** (*instructions2*), ... **case** (*exprn*) **then** (*instructions*), [**else** (*instructions*)], **end**	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**	--> exec('C:\Users\NicolasGaudin\Desktop\M2 SISE\SCILAB\exemple2.sce', -1) Saisir un entier entre 1 et 3 : 5 Vous n'avez rien compris !

VI. Les boucles

	FOR	WHILE
Syntaxe	**for** *indice* = *séquence* *instructions* **end**	*Initialisation_indice* **while** *condition* *instruction(s)* [break] **end**

Exemple	**for** i = 1:5
	disp(i)
	end
		i = 1
		while i<6
		disp(i)
		i = i + 1 //Incrémentation
		end

```
--> 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

```
function [sortie(s)] = nomfunction(paramètre(s))
    Traitement(s)
endfunction

//Appel de la fonction
[sortie(s)] = nomfunction(paramètres(s))
```

Exemple

```
// Fonction de calcul de la somme et de la différence
function [s, d] = calcul (a, b)
    s = a + b
    d = a - b
endfunction

// Appel de la fonction
a = 5; b = 10
[somme, diff] = calcul(a , b)
disp(somme, « somme »)
disp(diff, « différence »)
```

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

somme

  15.

différence

 -5.
```

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 = **linspace**(1, 6, 6)

Affectations, ajout, modification
v(2) = 10
v = [v, 10]

Opérations sur le vecteur
length(v) //Nombre d'éléments
sum(v), mean(v), min(v)… //somme, moyenne…

Sélections
v(2) //2ème élément
v(2:4) //du 2ème au 4ème élément
v(condition(s)) //selon une condition

Calculs entre 2 vecteurs
somme = v1 **+** v2
mult_termes = v1 **.*** v2
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.
```

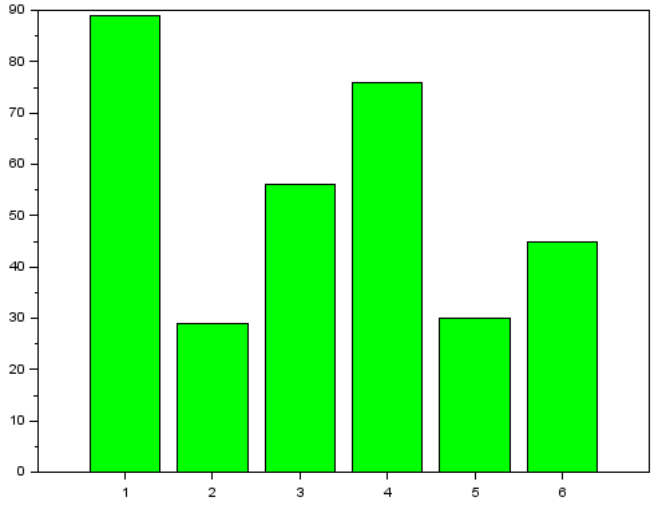
```
//Calcul de fréquences cumulées
--> frec=effc/sum(m2(:,2))
 frec  =

   0.05
   0.1
   0.35
   0.4
   0.45
   0.55
   0.8
   0.9
   1.
```

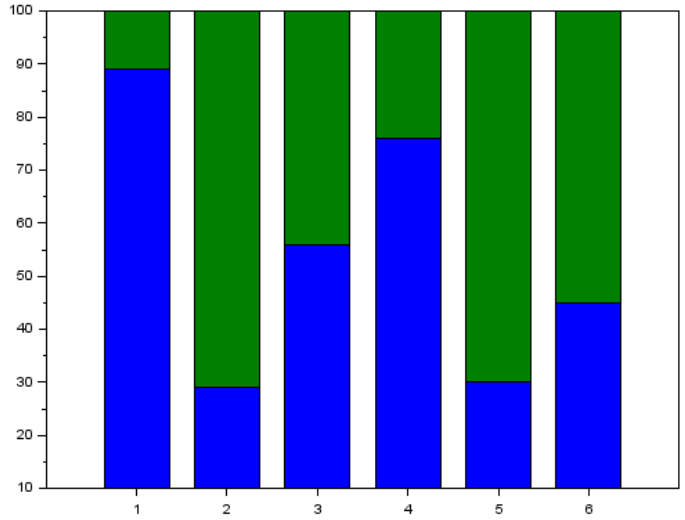
X. Graphiques

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

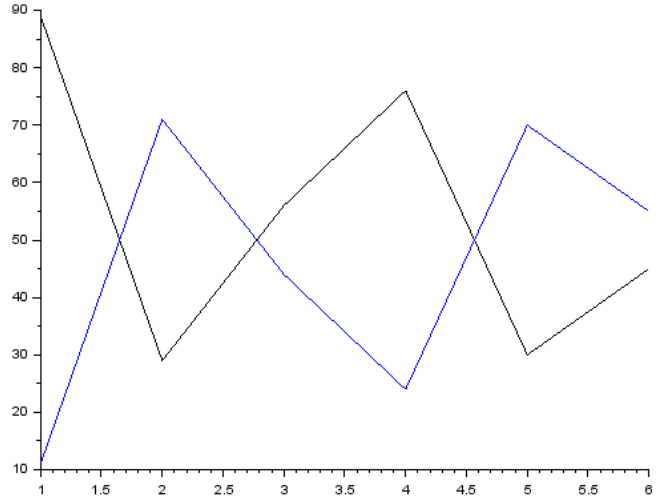
➢ bar(a(:,1),a(:,2),'green')



➢bar(a(:,1),a(:,2:3),'stacked')

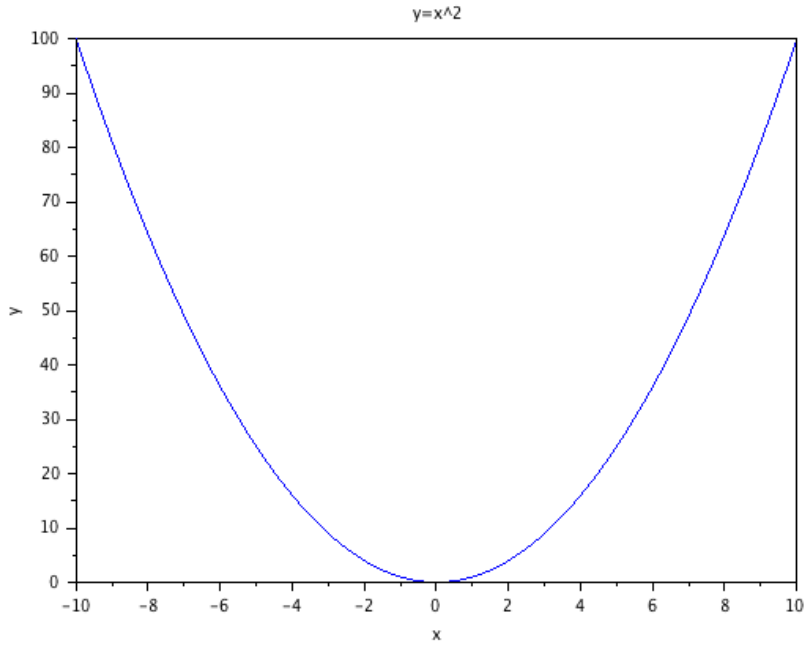


➢plot2d(a(:,1),a(:,2:3))



X. Graphiques

- x=linespace(-10,10,100)
- y=f(x)
- plot(x,y)



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