Ranges

Entrée [2]:

```
collect(1:10)
```

Out[2]:

```
10-element Vector{Int32}:
  1
  2
  3
  4
  5
  6
  7
  8
  9
 10
```

Entrée [3]:

```
collect(1.5:5.5)
```

Out[3]:

```
5-element Vector{Float64}:
 1.5
 2.5
 3.5
 4.5
 5.5
```

Entrée [11]:

```
collect(100:-20:0)
```

Out[11]:

```
6-element Vector{Int32}:
 100
  80
  60
  40
  20
   0
```

Entrée [12]:

```
c1=collect(100:-20:0)
```

Out[12]:

```
6-element Vector{Int32}:
 100
  80
  60
  40
  20
   0
```

Entrée [13]:

```
c1
```

Out[13]:

```
6-element Vector{Int32}:
 100
  80
  60
  40
  20
   0
```

Entrée [14]:

```
c1[4]
```

Out[14]:

```
40
```

Entrée [15]:

```
c1[end]
```

Out[15]:

```
0
```

Entrée [17]:

```
c1[end-4]
```

Out[17]:

```
80
```

Tuples

Entrée [7]:

```
a1 = [1,3,5]
t1 = (1,3,5)
```

Out[7]:

```
(1, 3, 5)
```

Entrée [8]:

```
a1
```

Out[8]:

```
3-element Vector{Int32}:
 1
 3
 5
```

Entrée [9]:

```
t1
```

Out[9]:

```
(1, 3, 5)
```

Entrée [22]:

```
t2 = ((1,3),(4,5))
```

Out[22]:

```
((1, 3), (4, 5))
```

Entrée [23]:

```
t2[1][2]
```

Out[23]:

```
3
```

Entrée [24]:

```
marks = (SVT = (90,100), Mathematique = (95,100), Français = (75,100))
```

Out[24]:

```
(SVT = (90, 100), Mathematique = (95, 100), Français = (75, 100))
```

Entrée [25]:

```
marks.SVT
```

Out[25]:

```
(90, 100)
```

Entrée [26]:

```
marks2 = (Informatique =(70,100), Chimie=(75,100))
```

Out[26]:

```
(Informatique = (70, 100), Chimie = (75, 100))
```

Entrée [27]:

```
merge(marks,marks2)
```

Out[27]:

```
(SVT = (90, 100), Mathematique = (95, 100), Français = (75, 100), Informatiq
ue = (70, 100), Chimie = (75, 100))
```

Dictonaire

Entrée [28]:

```
Voiture = Dict("Voit1"=> 1000000,"Voit2"=>2000000,"Voit3"=>3000000)
```

Out[28]:

```
Dict{String, Int32} with 3 entries:
  "Voit2" => 2000000
  "Voit3" => 3000000
  "Voit1" => 1000000
```

Entrée [31]:

```
Voiture2 = Dict(:Voit1=> 1000000,:Voit2=>2000000,:Voit3=>3000000)
```

Out[31]:

```
Dict{Symbol, Int32} with 3 entries:
  :Voit2 => 2000000
  :Voit3 => 3000000
  :Voit1 => 1000000
```

Entrée [33]:

```
Voiture2[:Voit1]
```

Out[33]:

```
1000000
```

Entrée [35]:

```
haskey(Voiture, :Voit4)
```

Out[35]:

```
false
```

Entrée [36]:

```
delete!(Voiture2,:Voit1)
```

Out[36]:

```
Dict{Symbol, Int32} with 2 entries:
  :Voit2 => 2000000
  :Voit3 => 3000000
```

Entrée [37]:

```
keys(Voiture)
```

Out[37]:

```
KeySet for a Dict{String, Int32} with 3 entries. Keys:
  "Voit2"
  "Voit3"
  "Voit1"
```

Entrée [38]:

```
values(Voiture)
```

Out[38]:

```
ValueIterator for a Dict{String, Int32} with 3 entries. Values:
  2000000
  3000000
  1000000
```

Entrée [39]:

```
dict3= merge(Voiture, Voiture2)
```

Out[39]:

```
Dict{Any, Int32} with 5 entries:
  :Voit2  => 2000000
  :Voit3  => 3000000
  "Voit2" => 2000000
  "Voit3" => 3000000
  "Voit1" => 1000000
```

Ensembles

Entrée [40]:

```
sports_marques = Set(["Adidas","Nike","Puma"])
```

Out[40]:

```
Set{String} with 3 elements:
  "Nike"
  "Adidas"
  "Puma"
```

Entrée [41]:

```
in("HRX", sports_marques)
```

Out[41]:

```
false
```

Entrée [42]:

```
in("Nike",sports_marques)
```

Out[42]:

```
true
```

Entrée [43]:

```
sports_marques_inde = Set(["Adidas","Nike","HRX"])
```

Out[43]:

```
Set{String} with 3 elements:
  "Nike"
  "HRX"
  "Adidas"
```

Entrée [44]:

```
union(sports_marques,sports_marques_inde)
```

Out[44]:

```
Set{String} with 4 elements:
  "Nike"
  "HRX"
  "Adidas"
  "Puma"
```

Entrée [45]:

```
intersect(sports_marques,sports_marques_inde)
```

Out[45]:

```
Set{String} with 2 elements:
  "Nike"
  "Adidas"
```

Entrée [46]:

```
setdiff(sports_marques,sports_marques_inde)
```

Out[46]:

```
Set{String} with 1 element:
  "Puma"
```

Entrée [47]:

```
setdiff(sports_marques_inde,sports_marques)
```

Out[47]:

```
Set{String} with 1 element:
  "HRX"
```

Entrée [48]:

```
push!(sports_marques,"HRX")
```

Out[48]:

```
Set{String} with 4 elements:
  "Nike"
  "HRX"
  "Adidas"
  "Puma"
```

Date et Temps

Entrée [50]:

```
Dates.Time
Dates.Date
Dates.Datetime
```

```
UndefVarError: Dates not defined

Stacktrace:
 [1] top-level scope
   @ In[50]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::Stri
ng, filename::String)
   @ Base .\loading.jl:1094
```

Entrée [51]:

```
using Dates
```

Entrée [52]:

```
now()
```

Out[52]:

```
2021-06-24T17:07:17.078
```

Entrée [53]:

```
today()
```

Out[53]:

```
2021-06-24
```

Entrée [54]:

```
anniversaire = Date(1999,12,17) #YYYY,MM,DD
```

Out[54]:

```
1999-12-17
```

Entrée [56]:

```
DateTime(1999,12,17, 5,5,00)
```

Out[56]:

```
1999-12-17T05:05:00
```

Entrée [57]:

```
now(UTC)
```

Out[57]:

```
2021-06-24T15:11:37.232
```

Entrée [58]:

```
month(anniversaire)
```

Out[58]:

```
12
```

Entrée [59]:

```
dayofweek(anniversaire)
```

Out[59]:

```
5
```

Entrée [60]:

```
dayname(anniversaire)
```

Out[60]:

```
"Friday"
```

Condition

Entrée [61]:

```
a= 15
```

Out[61]:

```
15
```

Entrée [64]:

```
a > 15 ? "Yes" : "No" #attention il ne faut pas coller les :
```

Out[64]:

```
"No"
```

Entrée [65]:

```
a <= 15 ? "Yes" : "No"
```

Out[65]:

```
"Yes"
```

Entrée [66]:

```
b = 20
```

Out[66]:

```
20
```

Entrée [67]:

```
a >= 10 || b<20
```

Out[67]:

```
true
```

Entrée [68]:

```
a >= 10 && b >=20
```

Out[68]:

```
true
```

Entrée [70]:

```
if a > 12
    print("a est plus grand que 12")
elseif a<12
    print("a est moins grand que 12")
else
    print("a est egal a 12")
end
```

```
a est plus grand que 12
```

Entrée [75]:

```
pays = "Inde"
```

Out[75]:

```
"Inde"
```

Entrée [76]:

```
if pays == "France"
    print("Tu parle de la France ?")
else
    print("Tu parle d'un autre pays que la France")
end
```

```
Tu parle d'un autre pays que la France
```

Boucle

Entrée [77]:

```
for i in ["Adidas","Nike","Puma"]
    println(i)
end
```

```
Adidas
Nike
Puma
```

Entrée [91]:

```
for i in ("Nike")
    print(i)
end
```

```
Nike
```

Entrée [78]:

```
for i in ("Nike")
    println(i)
end
```

```
N
i
k
e
```

Entrée [92]:

```
for c in Dict(:Voit1 => 1000000, :Voit2 => 2000000)
    print(c, " ") # mettre un " " pour faire l'espace et le mettre a la ligne
end
```

```
:Voit2 => 2000000 :Voit1 => 1000000
```

Entrée [84]:

```
c1 = Dict(:Voit1 => 1000000, :Voit2 => 2000000)
```

Out[84]:

```
Dict{Symbol, Int32} with 2 entries:
  :Voit2 => 2000000
  :Voit1 => 1000000
```

Entrée [86]:

```
for c in c1
    print(c, " ")
end
```

```
:Voit2 => 2000000 :Voit1 => 1000000
```

Entrée [87]:

```
for range in 1:5
    @show range
end
```

```
range = 1
range = 2
range = 3
range = 4
range = 5
```

Entrée [90]:

```
for x in 1:20
    if x % 2 == 0
        print(x , " ")
    end
end
```

```
2 4 6 8 10 12 14 16 18 20
```

Entrée [3]:

```
for x in 1:20
    i = x*20
    println("$(i) est un multiple de $(x) et de 20")
    # mettre $(i) pour afficher les valeur de i dans un commentaire
end
```

```
20 est un multiple de 1 et de 20
40 est un multiple de 2 et de 20
60 est un multiple de 3 et de 20
80 est un multiple de 4 et de 20
100 est un multiple de 5 et de 20
120 est un multiple de 6 et de 20
140 est un multiple de 7 et de 20
160 est un multiple de 8 et de 20
180 est un multiple de 9 et de 20
200 est un multiple de 10 et de 20
220 est un multiple de 11 et de 20
240 est un multiple de 12 et de 20
260 est un multiple de 13 et de 20
280 est un multiple de 14 et de 20
300 est un multiple de 15 et de 20
320 est un multiple de 16 et de 20
340 est un multiple de 17 et de 20
360 est un multiple de 18 et de 20
380 est un multiple de 19 et de 20
400 est un multiple de 20 et de 20
```

Import un packge

Entrée [29]:

```
using Pkg
```

Entrée [33]:

```
Pkg.add("Plots")
```

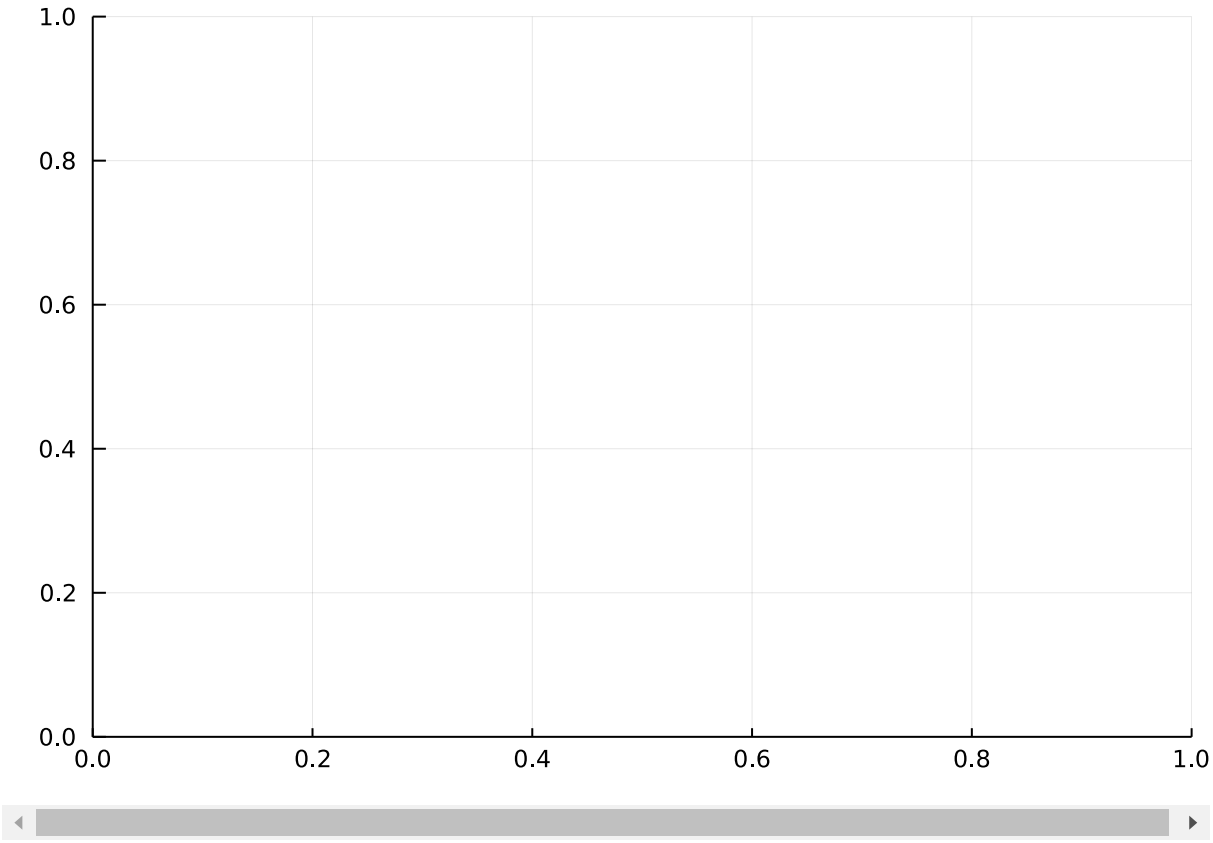
Entrée [34]:

```
using Plots
```

Entrée [35]:

```
plot()
```

Out[35]:



Entrée [46]:

```
x = 1:20; y = rand(20)
```

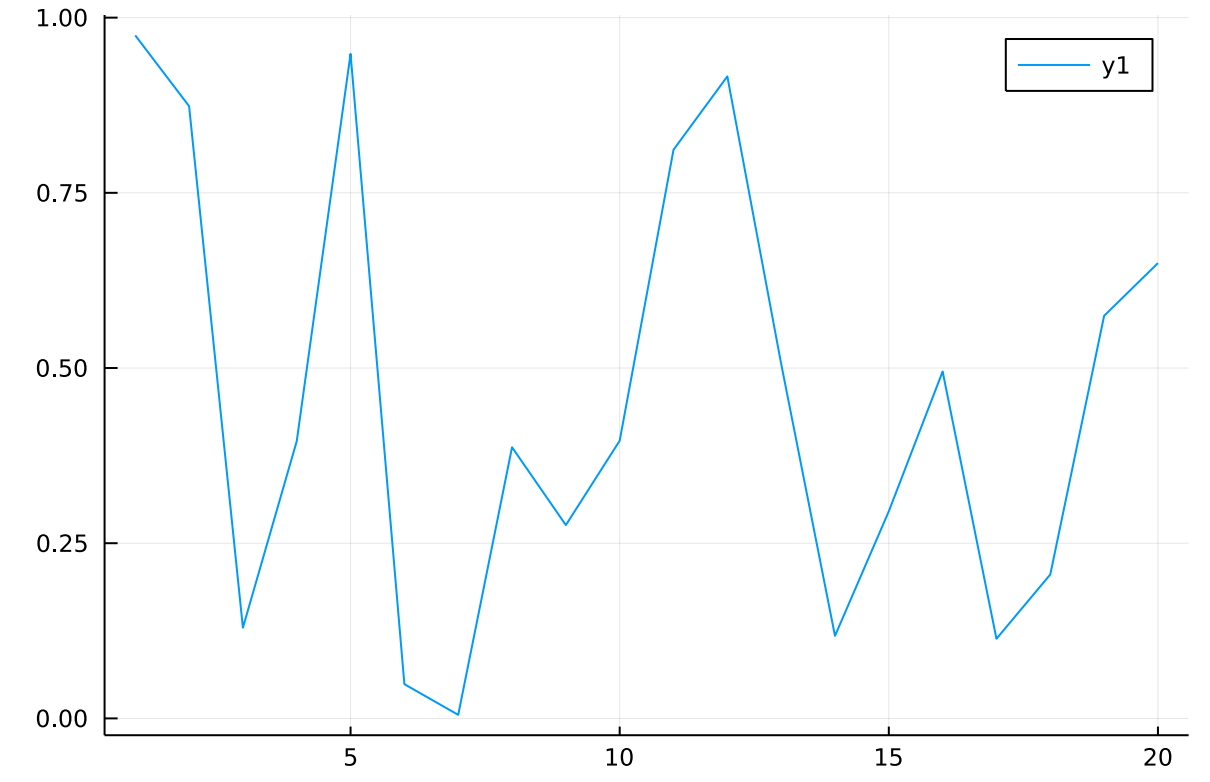
Out[46]:

```
20-element Vector{Float64}:
 0.9746725715602189
 0.8737110398672745
 0.1294348693001075
 0.3955429742657348
 0.948339902130152
 0.048864455516638206
 0.005078326691905888
 0.3869411104612537
 0.27585363665192686
 0.3962578608752232
 0.8114636698156166
 0.9161408642885653
 0.5060533318294691
 0.1177908874465945
 0.2955649509794065
 0.4949880031522318
 0.1136119320757214
 0.20537285509254466
 0.574484405162081
 0.6497113906880987
```

Entrée [47]:

```
plot(x,y) # mettre x = 1:y donc ici 20
```

Out[47]:



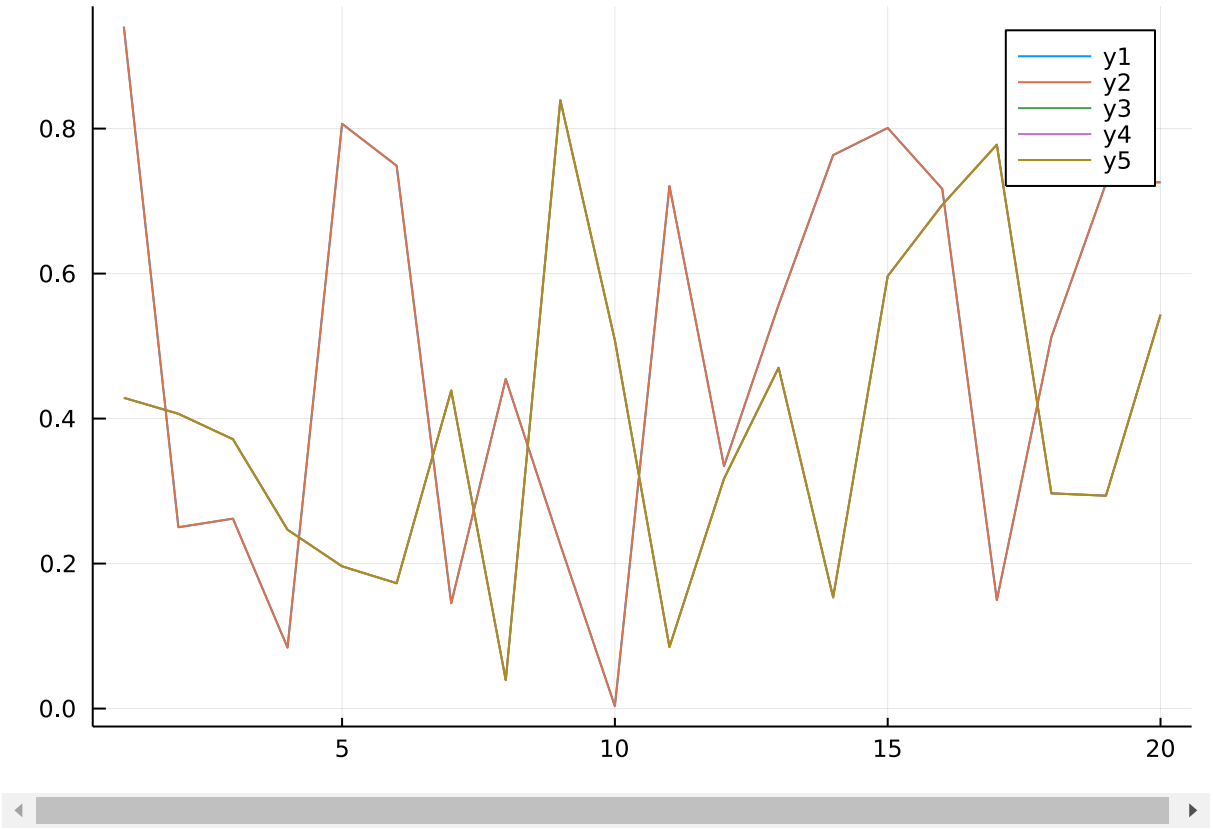
Entrée [56]:

```
z = rand(20);
```

Entrée [59]:

```
plot!(x,z) # y1=y2 ey y4=y3=y5
```

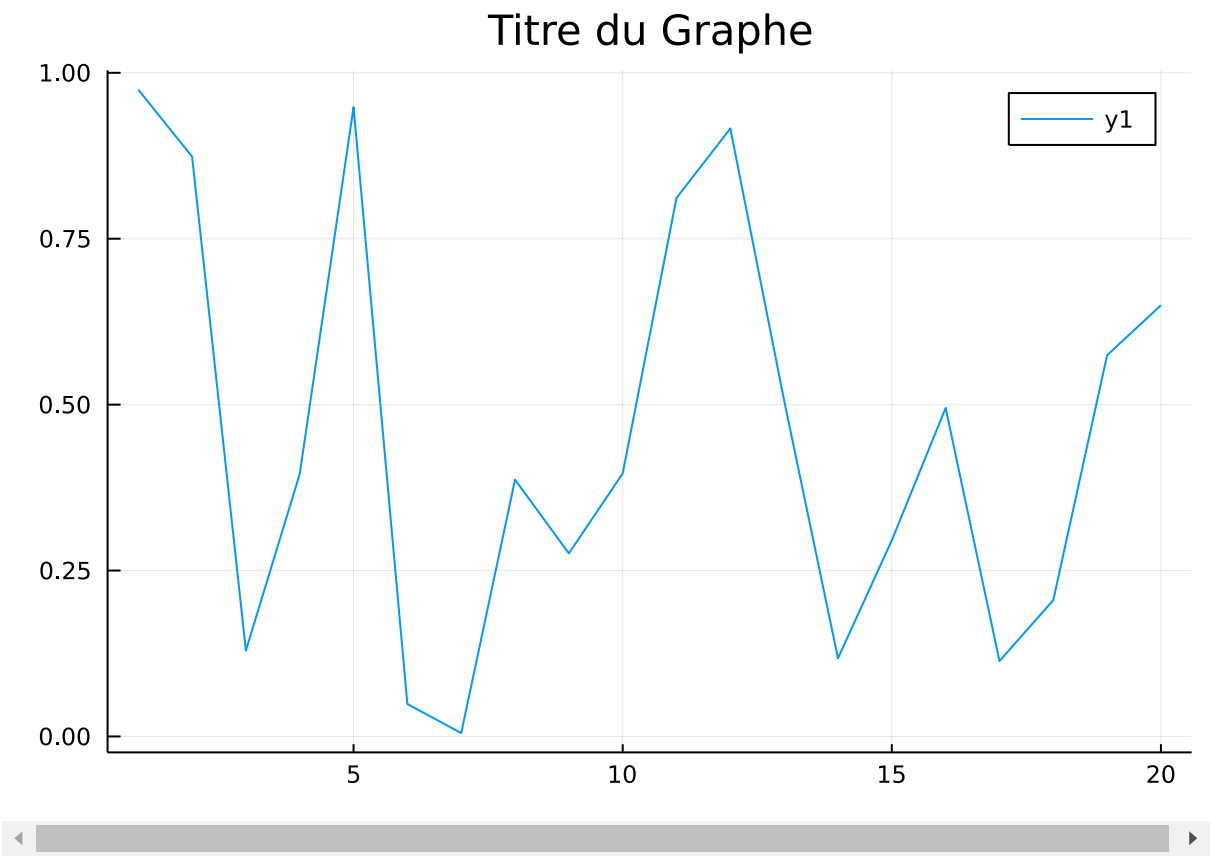
Out[59]:



Entrée [60]:

```
plot(x,y,title="Titre du Graphe")
```

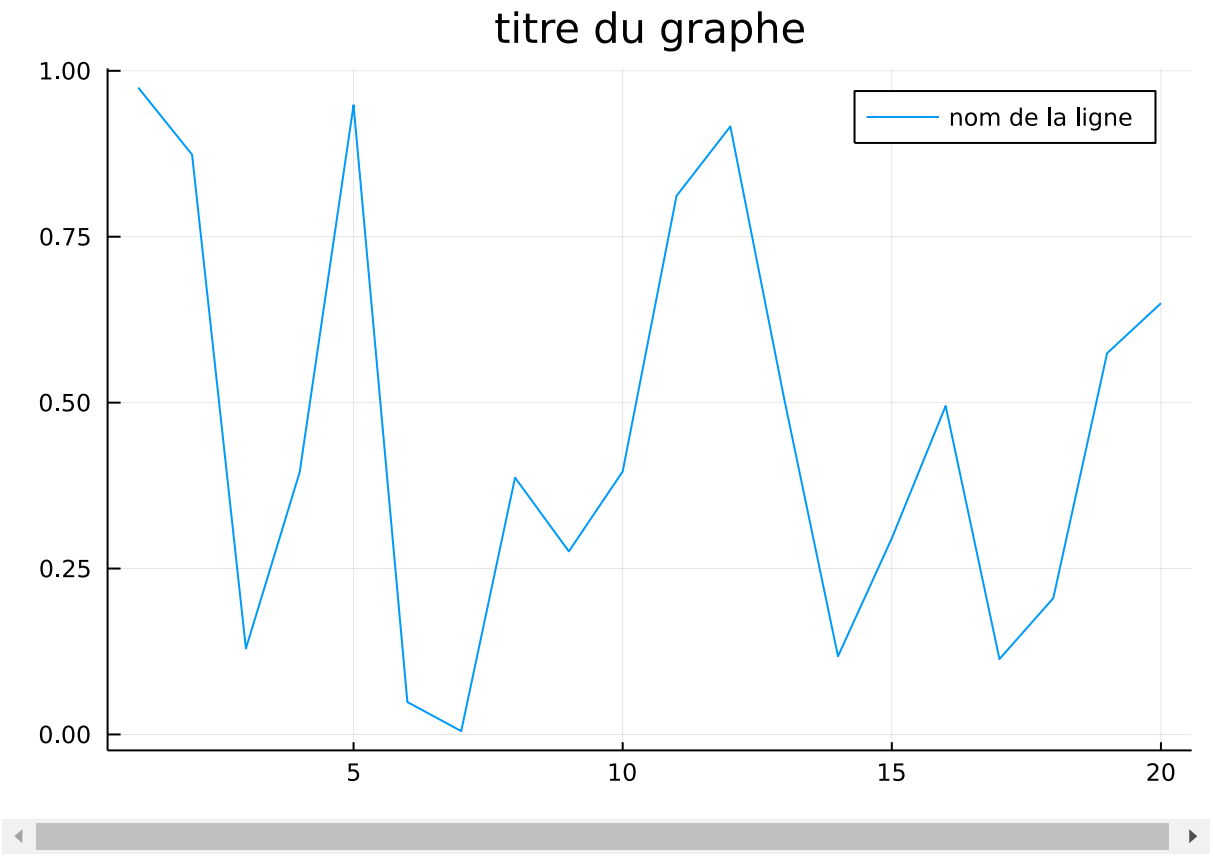
Out[60]:



Entrée [62]:

```
plot(x,y,title ="titre du graphe",label = "nom de la ligne")
```

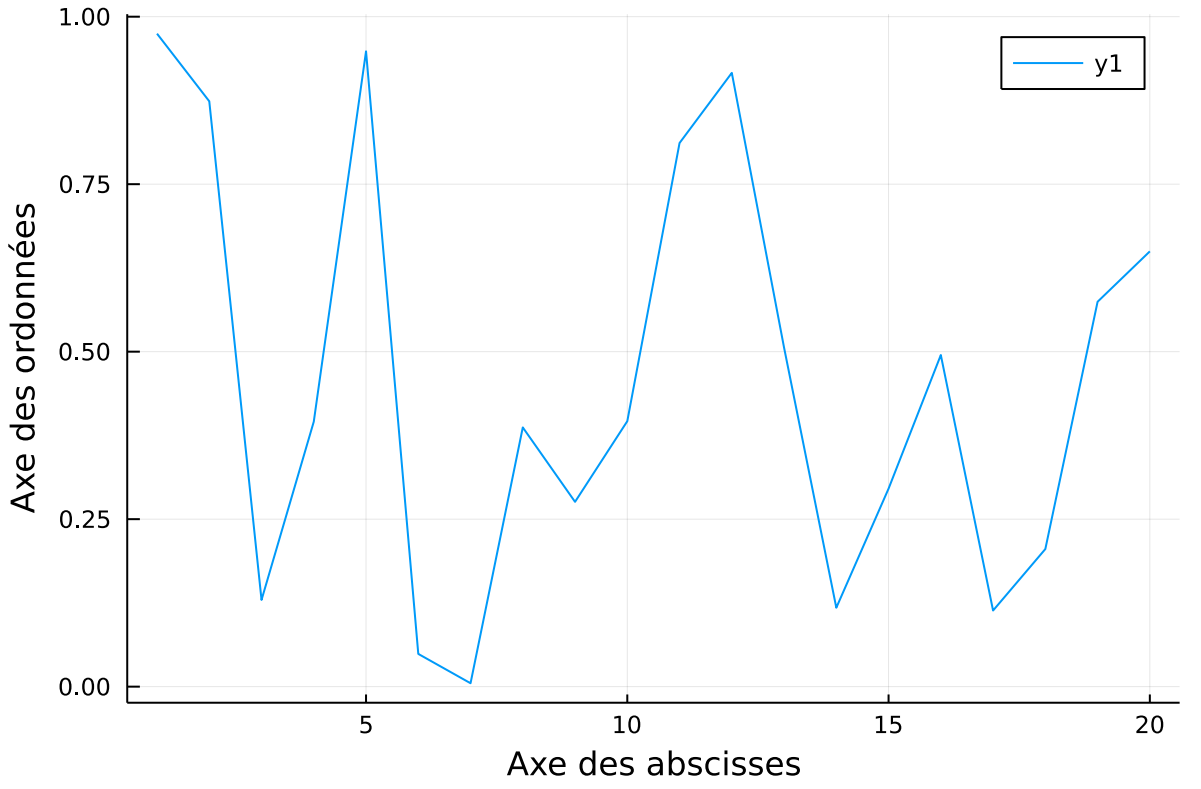
Out[62]:



Entrée [63]:

```
plot(x,y,xlabel = "Axe des abscisses",ylabel = "Axe des ordonnées")
```

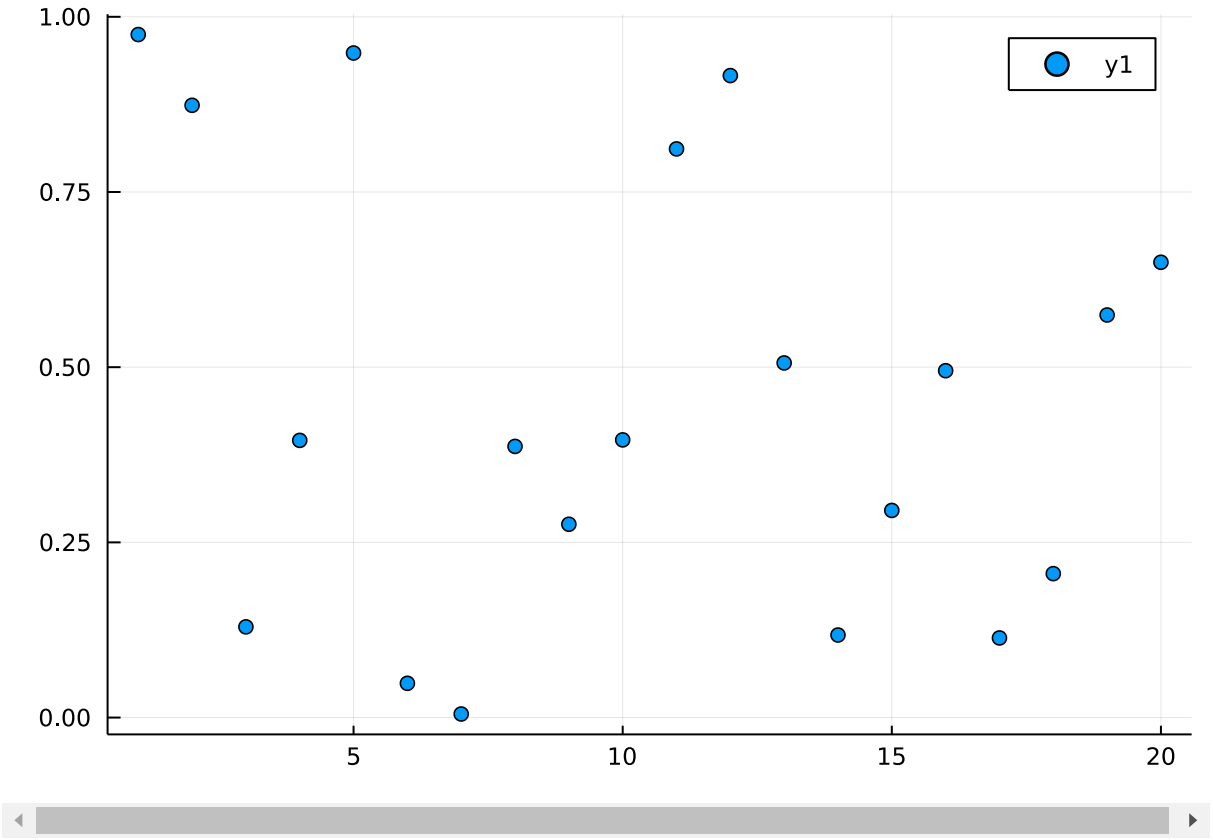
Out[63]:



Entrée [64]:

```
scatter(x,y) #graphe en point
```

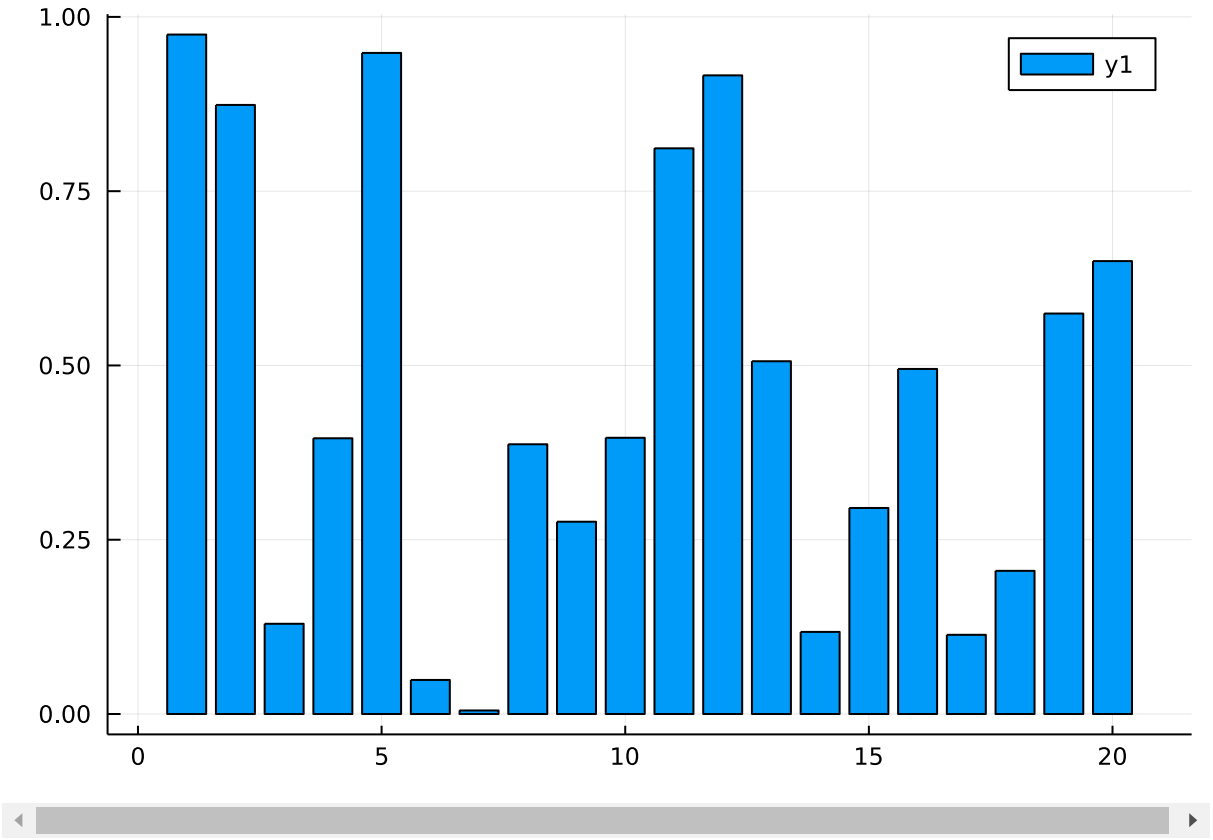
Out[64]:



Entrée [65]:

```
bar(x,y) #graphe en barre
```

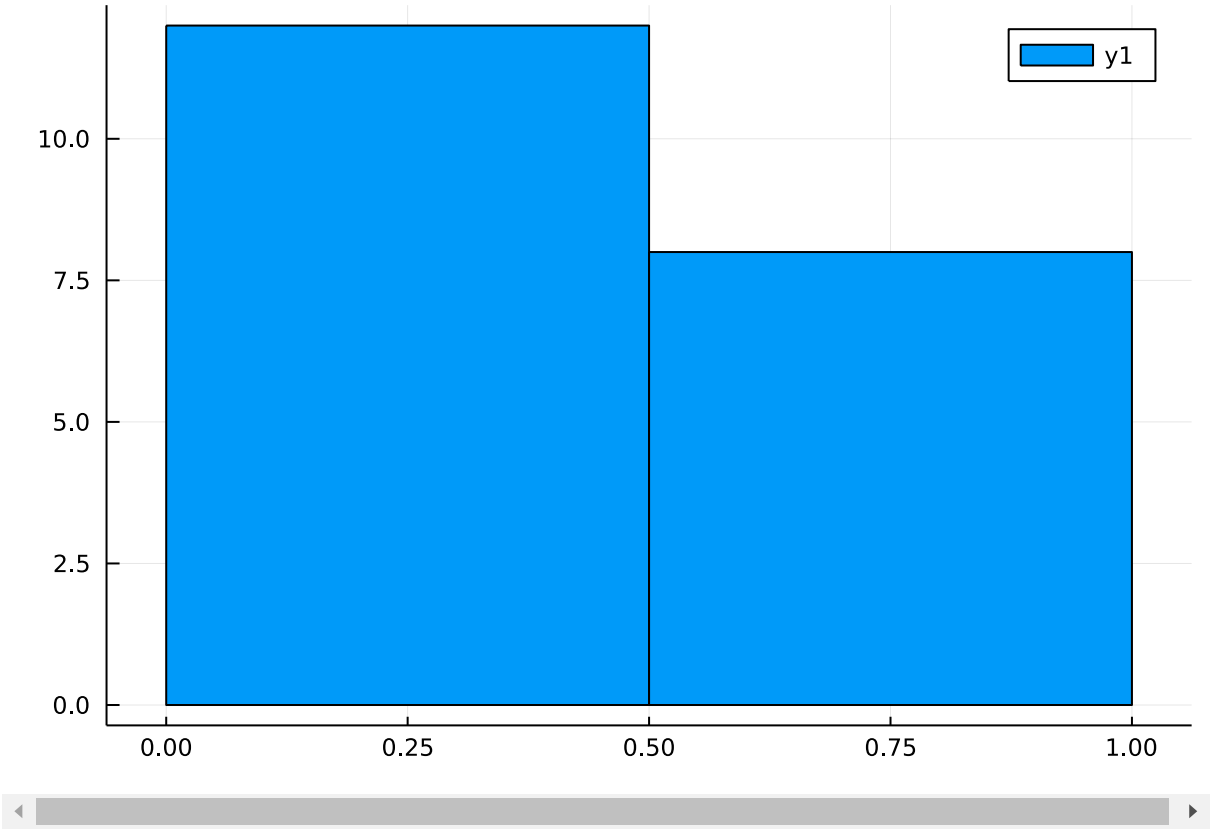
Out[65]:



Entrée [67]:

```
histogram(y)
```

Out[67]:

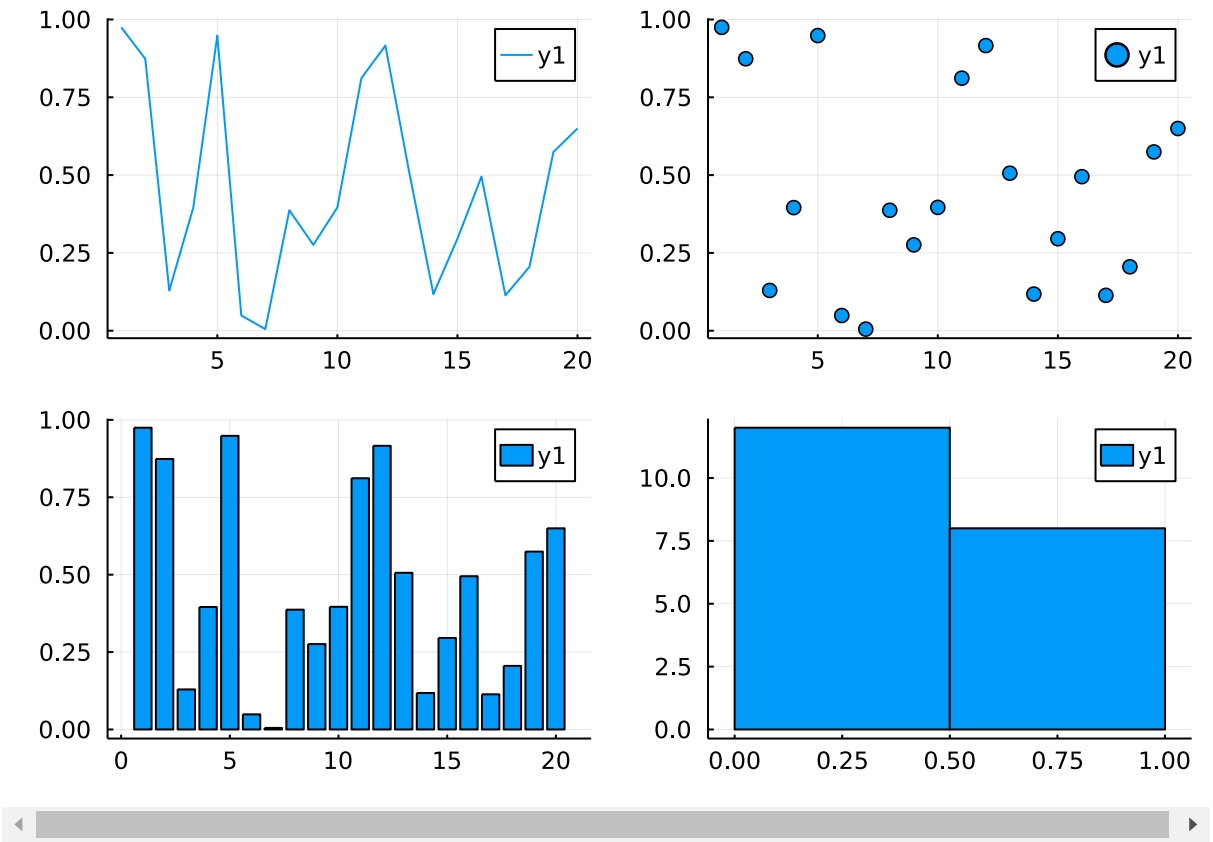


Entrée [72]:

```
p1 = plot(x,y)
p2 = scatter(x,y)
p3 = bar(x,y)
p4 = histogram(y)

plot(p1,p2,p3,p4,layout=(2,2),legend = true)
```

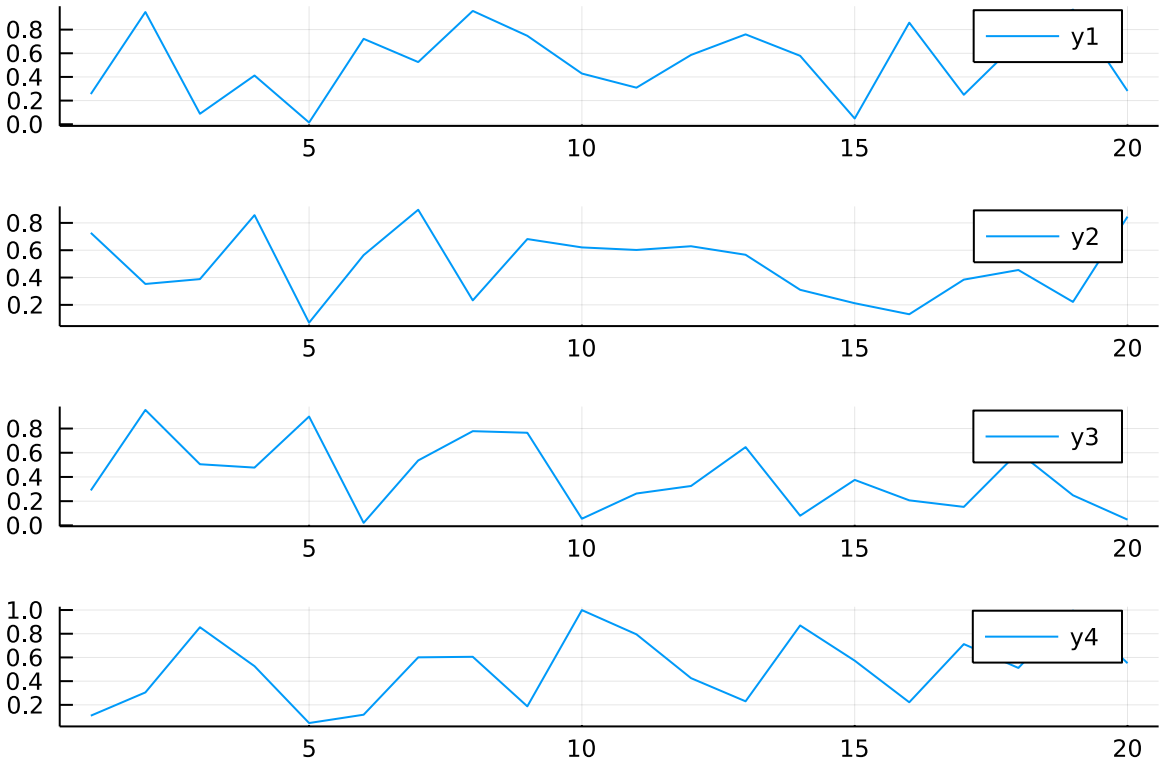
Out[72]:



Entrée [74]:

```
y = rand(20,4)
plot(x,y,layout=(4,1))
```

Out[74]:



Python Packages

Entrée [10]:

```
using Pkg # ATTENTION il faut refaire chaque étape pour les packages car
          # il ne garde pas l'opération que le Pkg a été installer
```

Entrée [6]:

```
Pkg.add("PyCall")
```

Entrée [7]:

```
using PyCall
```

Entrée [8]:

```
using SQLite
```

Entrée [9]:

```
np = pyimport("numpy")
```

Out[9]:

```
PyObject <module 'numpy' from 'C:\\Users\\user\\.julia\\conda\\3\\lib\\site-
packages\\numpy\\__init__.py'>
```

Entrée [12]:

```
a1 = np.array([2,3,4,5,6])
```

Out[12]:

```
5-element Vector{Int32}:
 2
 3
 4
 5
 6
```

Entrée [13]:

```
np.mean(a1)
```

Out[13]:

```
4.0
```

Entrée [14]:

```
np.median(a1)
```

Out[14]:

```
4.0
```

Entrée [16]:

```
math = pyimport("math")
```

Out[16]:

```
PyObject <module 'math' (built-in)>
```

Entrée [17]:

```
cos(4)
```

Out[17]:

```
-0.6536436208636119
```

Basse de données

Entrée [22]:

```
using Pkg
Pkg.add("SQLite")
```

Entrée [23]:

```
using SQLite
```

Entrée [24]:

```
db = SQLite.DB("Movies")
```

Out[24]:

```
SQLite.DB("Movies")
```

Entrée [42]:

```
DDL - Create, Alter and Drop - SQLite.DB
DML - Insert Update and Delete -SQLite.execute
DQL - Select - SQLite.Query
```

```
syntax: extra token "and" after end of expression

Stacktrace:
 [1] top-level scope
   @ In[42]:1
 [2] eval
   @ .\boot.jl:360 [inlined]
 [3] include_string(mapexpr::typeof(REPL.softscope), mod::Module, code::Stri
ng, filename::String)
   @ Base .\loading.jl:1094
```

Entrée [51]:

```
SQLite.execute(db,"CREATE TABLE IF NOT EXISTS movies(movie_id Real,movie_name TEXT,loction
```

Out[51]:

```
101
```

Entrée [49]:

```
SQLite.tables(db)
```

Out[49]:

```
(name = ["movies"],)
```

Entrée [52]:

```
SQLite.execute(db,"INSERT INTO movies(movie_id,movie_name,loction) Values(1,'Spiderman','US
```

Out[52]:

```
101
```

Entrée [64]:

```
SQLite.columns(db,"movies")
```

Out[64]:

```
(cid = Int64[0, 1, 2], name = ["movie_id", "movie_name", "loction"], type =
["Real", "TEXT", "TEXT"], notnull = Int64[0, 0, 0], dflt_value = [missing, m
issing, missing], pk = Int64[0, 0, 0])
```

Entrée [ ]:

```

```