

ÖGOR Summer-Workshop for PhD-candidates and Post-Docs

An introduction to Julia and JuMP for Operations Research

Prof. Dr. Xavier Gandibleux

Nantes Université – France
Département Informatique – Faculté des Sciences et Technique

Session 10

Selected topics

Random numbers



Random numbers (1/4)

Float number in $[0;1]$:

```
rand()
```

```
julia> rand()
```

Vector of n floats in $[0;1]$:

```
rand( $n$ )
```

```
julia> rand(3)
```

Matrix of $n_1 \times n_2$ floats in $[0;1]$:

```
rand( $n_1, n_2$ )
```

```
julia> rand(3,2)
```

Random numbers (1/4)

Float number in $[0;1]$:

```
rand()
```

```
julia> rand()
```

Vector of n floats in $[0;1]$:

```
rand( $n$ )
```

```
julia> rand(3)
```

Matrix of $n_1 \times n_2$ floats in $[0;1]$:

```
rand( $n_1, n_2$ )
```

```
julia> rand(3,2)
```

Random numbers (1/4)

Float number in $[0;1]$:

```
rand()
```

```
julia> rand()
```

Vector of n floats in $[0;1]$:

```
rand( $n$ )
```

```
julia> rand(3)
```

Matrix of $n_1 \times n_2$ floats in $[0;1]$:

```
rand( $n_1, n_2$ )
```

```
julia> rand(3,2)
```

Random numbers (2/4)

Integer number in $[a;b]$:

```
rand(a:b)
```

```
julia> rand(1:6)
```

Vector of n integers in $[a;b]$:

```
rand(a:b,n)
```

```
julia> rand(1:6,2)
```

Matrix of $n_1 \times n_2$ integers in $[a;b]$:

```
rand(a:b,n1,n2)
```

```
julia> rand(1:6,3,2)
```

Random numbers (2/4)

Integer number in $[a;b]$:

```
rand(a:b)
```

```
julia> rand(1:6)
```

Vector of n integers in $[a;b]$:

```
rand(a:b,n)
```

```
julia> rand(1:6,2)
```

Matrix of $n_1 \times n_2$ integers in $[a;b]$:

```
rand(a:b,n1,n2)
```

```
julia> rand(1:6,3,2)
```

Random numbers (3/4)

```
julia> using Random
```

A random permutation of length n :

```
randperm( $n$ )
```

```
julia> randperm(5)
```

Return a randomly permuted copy of vector v :

```
shuffle( $v$ )
```

```
julia> shuffle(Vector(1:10))
```


Random numbers (3/4)

```
julia> using Random
```

A random permutation of length n :

```
randperm( $n$ )
```

```
julia> randperm(5)
```

Return a randomly permuted copy of vector v :

```
shuffle( $v$ )
```

```
julia> shuffle(Vector(1:10))
```

Random numbers (4/4)

Reseed the random number generator:

```
Random.seed! (init)
```

```
julia> Random.seed! (1234)
julia> rand()
julia> rand()
julia> Random.seed! (1234)
julia> rand()
julia> rand()
```

→ a reproducible sequence of numbers when a seed is provided.

Selected topics

Search and Sort



Search functions on an array (1/3)

Compute the minimum (maximum) value of an array (a vector V or a matrix M) over the given dimensions

```
minimum(array, dims)
```

```
maximum(array, dims)
```

```
julia> minimum(V)
```

```
julia> minimum(M)
```

```
julia> minimum(M,dims=1)
```

```
julia> minimum(M,dims=2)
```

```
julia> maximum(V)
```

```
julia> maximum(M)
```

```
julia> maximum(M,dims=1)
```

```
julia> maximum(M,dims=2)
```

Search functions on an array (2/3)

For an array (a vector V or a matrix M), return the indices of the minimum (maximum) elements over the given dimensions

```
argmin(array, dims)
```

```
argmax(array, dims)
```

```
julia> argmin(V)
```

```
julia> argmin(M)
```

```
julia> argmin(M,dims=1)
```

```
julia> argmin(M,dims=2)
```

```
julia> argmax(V)
```

```
julia> argmax(M)
```

```
julia> argmax(M,dims=1)
```

```
julia> argmax(M,dims=2)
```

Search functions on an array (3/3)

For an array (a vector V or a matrix M), return the index or key of the first element of A for which predicate returns true. Return nothing if there is no such element.

```
findfirst(predicate, array)
```

```
julia> findfirst(isequal(3), M)
```

index i, j of the first element of M for which $M[i, j]=3$

```
julia> findfirst(iseven, M)
```

```
julia> findfirst("Linz", "Hello Linz")
```

See also `findlast`, `findprev`, `findnext`

Sort functions on a vector (1/2)

Returns a sorted copy of a vector V

```
sort(vector)
```

```
julia> sort(V)
```

Returns the vector V sorted

```
sort!(vector)
```

```
julia> sort!(V)
```

Option to reverse the sorting order

```
sort(vector, rev=true)
```

```
julia> sort(V, rev=true)
```

Sort functions on a vector (2/2)

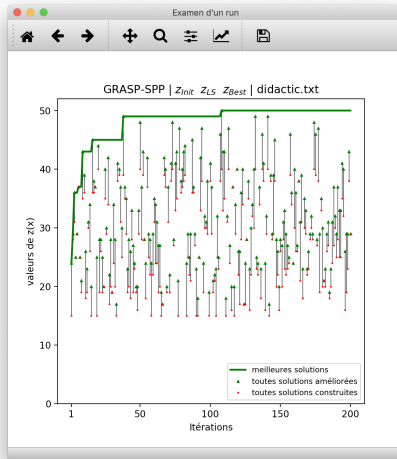
Return a permutation vector I that puts $V[I]$ in sorted order

```
sortperm(vector)  
sortperm(vector, rev=true)
```

```
julia> V=[2,7,4]  
julia> p=sortperm(V)  
julia> V[p]  
julia> p=sortperm(V, rev=true)  
julia> V[p]
```


Selected topics

Plotting



Overview

- ▶ **UnicodePlots**: a quick and easy way to draw plots in the REPL

<https://github.com/Envizero/UnicodePlots.jl>

- ▶ **PyPlot**: a collection of command style functions that make matplotlib work like MATLAB

<https://github.com/JuliaPy/PyPlot.jl>

<https://matplotlib.org/tutorials/introductory/pyplot.html>

- ▶ **Plots**: a high-level plotting package that interfaces with other plotting packages

<https://github.com/JuliaPlots/Plots.jl>

<http://docs.juliaplots.org>

To avoid any interactions (and thus error messages) between plotting packages, use only one plotting package simultaneously.

Overview

- ▶ **UnicodePlots**: a quick and easy way to draw plots in the REPL

<https://github.com/Envizero/UnicodePlots.jl>

- ▶ **PyPlot**: a collection of command style functions that make matplotlib work like MATLAB

<https://github.com/JuliaPy/PyPlot.jl>

<https://matplotlib.org/tutorials/introductory/pyplot.html>

- ▶ **Plots**: a high-level plotting package that interfaces with other plotting packages

<https://github.com/JuliaPlots/Plots.jl>

<http://docs.juliaplots.org>

To avoid any interactions (and thus error messages) between plotting packages, use only one plotting package simultaneously.

Overview

- ▶ **UnicodePlots**: a quick and easy way to draw plots in the REPL

<https://github.com/Evizero/UnicodePlots.jl>

- ▶ **PyPlot**: a collection of command style functions that make matplotlib work like MATLAB

<https://github.com/JuliaPy/PyPlot.jl>

<https://matplotlib.org/tutorials/introductory/pyplot.html>

- ▶ **Plots**: a high-level plotting package that interfaces with other plotting packages

<https://github.com/JuliaPlots/Plots.jl>

<http://docs.juliaplots.org>

To avoid any interactions (and thus error messages) between plotting packages, use only one plotting package simultaneously.

UnicodePlots

Install:

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

Setup:

```
using UnicodePlots
```

Example:

```
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]
```

```
julia> lineplot(X,Y)
```

```
julia> lineplot(X,Y,title="y=f(x)",name="y=4x+10")
```

UnicodePlots

Install:

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

Setup:

```
using UnicodePlots
```

Example:

```
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]
```

```
julia> lineplot(X,Y)
```

```
julia> lineplot(X,Y,title="y=f(x)",name="y=4x+10")
```

UnicodePlots

Install:

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

Setup:

```
using UnicodePlots
```

Example:

```
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]
```

```
julia> lineplot(X,Y)
```

```
julia> lineplot(X,Y,title="y=f(x)",name="y=4x+10")
```

Install:

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

Setup:

```
using PyPlot
```

Example:

```
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]
```

```
julia> plot(X,Y)
```


Install:

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

Setup:

```
using PyPlot
```

Example:

```
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]
```

```
julia> plot(X,Y)
```

Example (Cont'd):

```
julia> title("y=f(x)")
julia> xlabel("x values"); ylabel("y values")
julia> grid()
julia> plot(X,Y,label="y=4x+10")
julia> legend(loc=4, fontsize="small")
julia> xlim(-10,10); ylim(-20,40)
```

```
julia> plot(X,Y,label="y=4x+10",
            color="red",linewidth=2.0,linestyle="--")
```

```
julia> figure("Example with PyPlot",figsize=(6,6))
```

Example (Cont'd):

```
julia> title("y=f(x)")
julia> xlabel("x values"); ylabel("y values")
julia> grid()
julia> plot(X,Y,label="y=4x+10")
julia> legend(loc=4, fontsize="small")
julia> xlim(-10,10); ylim(-20,40)
```

```
julia> plot(X,Y,label="y=4x+10",
            color="red",linewidth=2.0,linestyle="--")
```

```
julia> figure("Example with PyPlot",figsize=(6,6))
```

Example (Cont'd):

```
julia> title("y=f(x)")
julia> xlabel("x values"); ylabel("y values")
julia> grid()
julia> plot(X,Y,label="y=4x+10")
julia> legend(loc=4, fontsize="small")
julia> xlim(-10,10); ylim(-20,40)
```

```
julia> plot(X,Y,label="y=4x+10",
           color="red",linewidth=2.0,linestyle="--")
```

```
julia> figure("Example with PyPlot",figsize=(6,6))
```

PyPlot: example of a 2x2 subplot

```
julia> fig = figure("pyplot subplot",figsize=(5,5))
julia> suptitle("Example of a 2x2 Subplot")
```

```
julia> subplot(221) #-----
julia> PyPlot.title("221 - NW"); grid("on")
julia> plot(X,Y,color="red")
```

```
julia> subplot(222) #-----
julia> PyPlot.title("222 - NE")
julia> scatter(X,Y)
julia> ax = subplot(223) #-----
```

```
julia> PyPlot.title("223 - SW")
julia> x = randn(100); nb = 25
julia> hist(x,nb)
julia> subplot(224) #-----
julia> PyPlot.title("224 - SE"); xlabel("X axis")
julia> Xc=[i for i=0:0.1:2* $\pi$ ]; Yc = sin.(Xc)
julia> plot(Xc,Yc) #-----
```

```
julia> fig.canvas.draw()
```

PyPlot: example of a 2x2 subplot

```
julia> fig = figure("pyplot subplot",figsize=(5,5))
```

```
julia> suptitle("Example of a 2x2 Subplot")
```

```
julia> subplot(221) #-----
```

```
julia> PyPlot.title("221 - NW"); grid("on")
```

```
julia> plot(X,Y,color="red")
```

```
julia> subplot(222) #-----
```

```
julia> PyPlot.title("222 - NE")
```

```
julia> scatter(X,Y)
```

```
julia> ax = subplot(223) #-----
```

```
julia> PyPlot.title("223 - SW")
```

```
julia> x = randn(100); nb = 25
```

```
julia> hist(x,nb)
```

```
julia> subplot(224) #-----
```

```
julia> PyPlot.title("224 - SE"); xlabel("X axis")
```

```
julia> Xc=[i for i=0:0.1:2* $\pi$ ]; Yc = sin.(Xc)
```

```
julia> plot(Xc,Yc) #-----
```

```
julia> fig.canvas.draw()
```

PyPlot: example of a 2x2 subplot

```
julia> fig = figure("pyplot subplot",figsize=(5,5))
julia> suptitle("Example of a 2x2 Subplot")

julia> subplot(221)                                #-----
julia> PyPlot.title("221 - NW"); grid("on")
julia> plot(X,Y,color="red")

julia> subplot(222)                                #-----
julia> PyPlot.title("222 - NE")
julia> scatter(X,Y)
julia> ax = subplot(223)                            #-----
julia> PyPlot.title("223 - SW")
julia> x = randn(100); nb = 25
julia> hist(x,nb)
julia> subplot(224)                                #-----
julia> PyPlot.title("224 - SE"); xlabel("X axis")
julia> Xc=[i for i=0:0.1:2* $\pi$ ]; Yc = sin.(Xc)
julia> plot(Xc,Yc)                                  #-----
julia> fig.canvas.draw()
```

Plots (1/2)



Plots supports multiple backends:

- ▶ UnicodePlots
- ▶ PyPlot
- ▶ GR
- ▶ Plotly(JS)
- ▶ etc.

within the REPL
the universe of Matplotlib
simple and fast
interactive

Without changing syntax, we can create plots with different backends!

Install:

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

Setup:

```
using Plots
```


Plots (1/2)



Plots supports multiple backends:

- ▶ UnicodePlots
- ▶ PyPlot
- ▶ GR
- ▶ Plotly(JS)
- ▶ etc.

within the REPL
the universe of Matplotlib
simple and fast
interactive

Without changing syntax, we can create plots with different backends!

Install:

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

Setup:

```
using Plots
```

Plots (2/2)



Example with GR:

```
julia> gr()  
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]  
julia> plot(X,Y, label="line")
```

```
julia> scatter!(X,Y, label="points")
```

```
julia> title!("y=f(x)")  
julia> xlabel!("x values"); ylabel!("y values")
```

Example with UnicodePlots:

```
julia> unicodeplots()  
julia> plot(X,Y, label="line")
```



Example with GR:

```
julia> gr()  
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]  
julia> plot(X,Y, label="line")
```

```
julia> scatter!(X,Y, label="points")
```

```
julia> title!("y=f(x)")  
julia> xlabel!("x values"); ylabel!("y values")
```

Example with UnicodePlots:

```
julia> unicodeplots()  
julia> plot(X,Y, label="line")
```



Example with GR:

```
julia> gr()  
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]  
julia> plot(X,Y, label="line")
```

```
julia> scatter!(X,Y, label="points")
```

```
julia> title!("y=f(x)")  
julia> xlabel!("x values"); ylabel!("y values")
```

Example with UnicodePlots:

```
julia> unicodeplots()  
julia> plot(X,Y, label="line")
```



Example with GR:

```
julia> gr()  
julia> f(x)=4*x+10  
julia> X=[x for x=-5:5]; Y=[f(x) for x in X]  
julia> plot(X,Y, label="line")
```

```
julia> scatter!(X,Y, label="points")
```

```
julia> title!("y=f(x)")  
julia> xlabel!("x values"); ylabel!("y values")
```

Example with UnicodePlots:

```
julia> unicodeplots()  
julia> plot(X,Y, label="line")
```

Review and exercises

(notebook)

