07-exemple-julia

August 21, 2020

1 Example Julia

1.1 Documentation de fonction

[1]:
```
"""
    bspline(p, j, x)

Return the value at x in [0,1[ of the B-spline with integer nodes of degree p␣
↪with support starting at j.
Implemented recursively using the [De Boor's Algorithm](https://en.wikipedia.
↪org/wiki/De_Boor%27s_algorithm)

```math
B_{i,0}(x) := \\left\\{
\\begin{matrix}
1 & \\mathrm{if}  \\quad t_i   x < t_{i+1} \\\\
0 & \\mathrm{otherwise}
\\end{matrix}
\\right.
```

```math
B_{i,p}(x) := \\frac{x - t_i}{t_{i+p} - t_i} B_{i,p-1}(x)
+ \\frac{t_{i+p+1} - x}{t_{i+p+1} - t_{i+1}} B_{i+1,p-1}(x).
```
"""
function bspline(p::Int, j::Int, x::Float64)
   if p == 0
       if j == 0
           return 1.0
       else
           return 0.0
       end
   else
       w = (x - j) / p
       w1 = (x - j - 1) / p
   end
   return (w * bspline(p - 1, j, x) + (1 - w1) * bspline(p - 1, j + 1, x))
```

```
end
```

[1]:
```
bspline
```

Switch the next cell from `Markdown` to `Code` format

?bspline

[1]:
```
using Pkg
Pkg.add("Plots")
Pkg.add("DataFrames")
Pkg.add("StatsPlots");
```

```
   Updating registry at `~/.julia/registries/General`


   Updating git-repo
`https://github.com/JuliaRegistries/General.git`


  Resolving package versions…
No Changes to `~/.julia/environments/v1.5/Project.toml`
No Changes to `~/.julia/environments/v1.5/Manifest.toml`
  Resolving package versions…
No Changes to `~/.julia/environments/v1.5/Project.toml`
No Changes to `~/.julia/environments/v1.5/Manifest.toml`
  Resolving package versions…
No Changes to `~/.julia/environments/v1.5/Project.toml`
No Changes to `~/.julia/environments/v1.5/Manifest.toml`
```

[ ]:
```
using Random, Plots

function generate_data( n = 2000, seed = 1234 )
    seuil = 0.25
    rng = MersenneTwister(seed)
    X1 = rand( rng, n)
    X2 = rand( rng, n)
    U  = rand( rng, n)
    Y  = zeros(Int,n)
    Y[(X1 .<= 0.25) .& (U  .<= seuil)] .= 1
    Y[(X1 .>  0.25) .& (X2 .>= 0.75) .& (U .<= seuil)] .= 1
    Y[(X1 .>  0.25) .& (X2 .<  0.75) .& (U .>  seuil)] .= 1
    return X1, X2, Y
end

X1, X2, Y = generate_data()
scatter(X1,X2, marker_z = Y)
```

[ ]:
```
using DataFrames, StatsPlots
```

[ ]:
```
data = DataFrame( X1=X1, X2=X2, Y=Y)
head(data)
```

[ ]:
```
@df data scatter(:X1,:X2, zcolor= :Y, xaxis = "X1", yaxis="X2", lab="Y")
```

[ ]:
```
"""
[x1,x2,x1^2,x1x2,x2^2……x2^6]
"""
function map_features(X1,X2)
    degree = 6
    out = ones(size(X1[:,1]))
    for i=1:6
        for j=0:i
            out = hcat(out,(X1.^(i-j)).*(X2.^j))
        end
    end
    return out
end
```

[ ]:
```
X = map_features(X1,X2)
```

[ ]:
```

```