1. Overview

February 16, 2014

Part I

Overview of the Julia Language

1 References:

- http://learnxinyminutes.com/docs/julia/
- http://dmbates.blogspot.com/2012/04/r-programmer-looks-at-julia.html
- http://www.r-bloggers.com/a-beginners-look-at-julia/
- https://github.com/JuliaLang/julia
- http://nbviewer.ipython.org/urls/raw.github.com/johnmyleswhite/DCStats.jl/master/Setup.ipynb
- http://julialang.org/blog/2012/03/shelling-out-sucks/
- http://docs.julialang.org/en/latest/manual/running-external-programs/
- http://julialang.org/blog/2013/04/put-this-in-your-pipe/
- http://julialang.org/blog/2013/09/fast-numeric/
- http://www.admin-magazine.com/HPC/Articles/Parallel-Julia-Jumping-Right-In
- http://web.mit.edu/julia_v1.24.13/www/index.html
- http://nbviewer.ipython.org/url/jdj.mit.edu/~stevenj/IJulia%2520Preview.ipynb
- http://strata.oreilly.com/2012/10/matlab-r-julia-languages-for-data-analysis.html
- http://asbidyarthy.blogspot.com/2012/06/julia-programming-language-downloads.html
- http://jurjenbokma.com/ApprenticesNotes/getting_statlinked_binaries_on_debian.html

2 Preliminaries

- Quitting Julia: [ctrl]-D or quit()
- Abort or clear current command: [ctrl]-C
- Parentheses ( ) are used for functions and multiple outputs
- Brackets [ ] are used for indexing
- Braces { } are used for arrays
- Running a file: include("file.jl") or require("file.jl")
- Printing values: print() or println()

- Deployment: main.jl #!/fullpath/julia

  arg=ARGS[1]; println("Hello $arg");

in shell: ./main.jl World => Hello World

External Shell Calls:

```
In [1]: run(`cal`)
```

```
   February 2014
Su Mo Tu We Th Fr Sa
                   1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
23 24 25 26 27 28
```

C-Function Call:

```
In [2]: ccall(:clock,Int32,())
```

```
6290918
```

```
Out [2]:
```

```
In [3]: bytestring(ccall(:ctime, Ptr{Uint8}, ()))
```

```
"Fri Nov 26 18:10:08 2106\n"
```

```
Out [3]:
```

Adding Package:

```
In [5]: Pkg.add("Calendar")
```

```
INFO: Nothing to be done.
```

```
In [6]: using Calendar
        Calendar.now()
```

```
16 Feb 2014 12:12:22 GMT
```

```
Out [6]:
```

```
In [7]: Calendar.dayofyear(Calendar.now())
```

```
47
```

```
Out [7]:
```

3 Variables, Vectors, and Matrices

```
In [8]: x=3
```

```
3
```

```
Out [8]:
```

```
In [9]: x=[1,2,3] # column vector
```

```
3-element Array{Int64,1}:
 1
 2
 3
```

```
Out [9]:
```

```
In [10]: x[2]
```

```
2
```

```
Out [10]:
```

```
In [11]: x=[1 2 3 4] # row vector
```

```
1x4 Array{Int64,2}:
 1  2  3  4
```

Out [11]:

```
x=(1,2,3,4)
```

In [12]:

```
(1,2,3,4)
```

Out [12]:

```
x1,x2=1,2
println(x1,",",x2)
```

In [13]:

```
1,2
```

```
x1,x2=x2,x1 #swap x1,x2 values
```

In [14]:

```
(2,1)
```

Out [14]:

```
x1
```

In [15]:

```
2
```

Out [15]:

```
M=[1 2 3;4 5 6;7 8 9;10 11 12]  # matrix
```

In [16]:

```
4x3 Array{Int64,2}:
  1   2   3
  4   5   6
  7   8   9
 10  11  12
```

Out [16]:

```
M=reshape(M,3,4)
```

In [17]:

```
3x4 Array{Int64,2}:
 1  10   8   6
 4   2  11   9
 7   5   3  12
```

Out [17]:

```
M[1,:] # first row
```

In [18]:

```
1x4 Array{Int64,2}:
 1  10  8  6
```

Out [18]:

```
M[:,1] # first column
```

In [19]:

```
3-element Array{Int64,1}:
 1
 4
 7
```

Out [19]:

```
M[3,1:3] # 3rd row, first to third column
```

In [20]:

```
1x3 Array{Int64,2}:
 7  5  3
```

Out [20]:

```
rand(5) # column vector of random numbers
```

In [21]:

```
5-element Array{Float64,1}:
 0.689187
 0.481103
 0.422456
 0.332155
 0.44626
```

Out [21]:

```
rand(5,5) # matrix of random numbers
```

In [22]:

```
5x5 Array{Float64,2}:
 0.275435  0.174196  0.791267   0.0491136  0.76978
 0.460359  0.443341  0.0415981  0.512503   0.143485
 0.641527  0.477086  0.783718   0.944992   0.418183
 0.231538  0.884084  0.587011   0.988167   0.89029
 0.577792  0.76283   0.440785   0.436356   0.154204
```

Out [22]:

```
In [23]: eye(5)  # identity matrix
```
```
Out [23]: 5x5 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0  0.0
 0.0  0.0  1.0  0.0  0.0
 0.0  0.0  0.0  1.0  0.0
 0.0  0.0  0.0  0.0  1.0
```
```
In [24]: a=zeros(5,5)
```
```
Out [24]: 5x5 Array{Float64,2}:
 0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0
 0.0  0.0  0.0  0.0  0.0
```
```
In [25]: b=ones(5,5)
```
```
Out [25]: 5x5 Array{Float64,2}:
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
 1.0  1.0  1.0  1.0  1.0
```
```
In [26]: linspace(1,2,10)
```
```
Out [26]: 10-element Array{Float64,1}:
 1.0
 1.11111
 1.22222
 1.33333
 1.44444
 1.55556
 1.66667
 1.77778
 1.88889
 2.0
```
```
In [27]: A=diagm(ones(5))
```
```
Out [27]: 5x5 Array{Float64,2}:
 1.0  0.0  0.0  0.0  0.0
 0.0  1.0  0.0  0.0  0.0
 0.0  0.0  1.0  0.0  0.0
 0.0  0.0  0.0  1.0  0.0
 0.0  0.0  0.0  0.0  1.0
```
```
In [28]: diag(A)
```
```
Out [28]: 5-element Array{Float64,1}:
 1.0
 1.0
 1.0
 1.0
 1.0
```
```
In [29]: [sin(i) for i in 1:5] #comprehension
```
```
Out [29]: 5-element Array{Float64,1}:
  0.841471
  0.909297
  0.14112
```

```
 -0.756802
 -0.958924
```

```
In [30]: V=[(i,j) for i=1:3, j=4:10]
```

```
Out [30]: 3x7 Array{(Int64,Int64),2}:
 (1,4)  (1,5)  (1,6)  (1,7)  (1,8)  (1,9)  (1,10)
 (2,4)  (2,5)  (2,6)  (2,7)  (2,8)  (2,9)  (2,10)
 (3,4)  (3,5)  (3,6)  (3,7)  (3,8)  (3,9)  (3,10)
```

```
In [31]: M=[sin(i+j) for i=1:5,j=4:7]
```

```
Out [31]: 5x4 Array{Float64,2}:
 -0.958924  -0.279415   0.656987   0.989358
 -0.279415   0.656987   0.989358   0.412118
  0.656987   0.989358   0.412118  -0.544021
  0.989358   0.412118  -0.544021  -0.99999
  0.412118  -0.544021  -0.99999   -0.536573
```

```
In [32]: M = @parallel [sin(i+j) for i=1:5,j=4:7]
```

```
Out [32]: 5x4 DArray{Float64,2,Array{Float64,2}}:
 -0.958924  -0.279415   0.656987   0.989358
 -0.279415   0.656987   0.989358   0.412118
  0.656987   0.989358   0.412118  -0.544021
  0.989358   0.412118  -0.544021  -0.99999
  0.412118  -0.544021  -0.99999   -0.536573
```

4 Function and Operations

```
In [33]: a=3*8;
         b=8/4;
         c=4//8;
         d=2^8;
         println("$a,$b,$c,$d")
```

```
24,2.0,1//2,256
```

```
In [34]: 2^(2+3im)
```

```
Out [34]: -1.947977671863125 + 3.493620327099486im
```

```
In [35]: sin(pi/3)
```

```
Out [35]: 0.8660254037844386
```

```
In [36]: besselj(2,5)
```

```
Out [36]: 0.0465651162777522
```

```
In [37]: f(x)=x*x
```

```
Out [37]: f (generic function with 1 method)
```

```
In [38]: function f(x)
             x*x
         end
```

```
Out [38]: f (generic function with 1 method)
```

```
In [39]: f(3)
```

```
Out [39]: 9
```

```
In [40]: ff=function(x)
           x*x
         end
```

```
(anonymous function)
```

Out [40]:

In [41]:
```
p=println
p(34)
```
```
34
```

In [42]:
```
p=ff
p(4)
```
```
16
```

Out [42]:
```
fff(x,y,z)=x*y*z
```

In [43]:
```
fff (generic function with 1 method)
```

Out [43]:
```
fff(3,4,5)
```

In [44]:
```
60
```

Out [44]:
```
map(x->x*x,[1,4,2])
```

In [45]:
```
3-element Array{Int64,1}:
```

Out [45]:
```
  1
 16
  4
```

In [46]:
```
map(f,1:10)
```
```
10-element Array{Int64,1}:
```

Out [46]:
```
   1
   4
   9
  16
  25
  36
  49
  64
  81
 100
```

In [47]:
```
map(x->x*x,1:10)
```
```
10-element Array{Int64,1}:
```

Out [47]:
```
   1
   4
   9
  16
  25
  36
  49
  64
  81
 100
```

In [48]:
```
map(1:10) do x
y=x*x
end
```
```
10-element Array{Int64,1}:
```

Out [48]:
```
   1
   4
   9
  16
  25
  36
  49
```

```
 64
 81
100
```

In [49]:
```
x=[1,2,3]
```

Out [49]:
```
3-element Array{Int64,1}:
 1
 2
 3
```

In [50]:
```
3x
```

Out [50]:
```
3-element Array{Int64,1}:
 3
 6
 9
```

In [51]:
```
x+3
```

Out [51]:
```
3-element Array{Int64,1}:
 4
 5
 6
```

In [52]:
```
y=[2,4,6]
```

Out [52]:
```
3-element Array{Int64,1}:
 2
 4
 6
```

In [53]:
```
x' * y   # dot product
```

Out [53]:
```
1-element Array{Int64,1}:
 28
```

In [54]:
```
dot(x,y) # dot product
```

Out [54]:
```
28
```

In [55]:
```
sum(conj(x).* y)  # dot product
```

Out [55]:
```
28
```

In [56]:
```
x .* y  # elementwise multiplication
```

Out [56]:
```
3-element Array{Int64,1}:
  2
  8
 18
```

In [57]:
```
cos(x)
```

Out [57]:
```
3-element Array{Float64,1}:
  0.540302
 -0.416147
 -0.989992
```

In [58]:
```
A=rand(5,5)
```

Out [58]:
```
5x5 Array{Float64,2}:
 0.696506    0.0310848  0.8425    0.360767  0.164839
 0.00884683  0.864284   0.647085  0.170653  0.0546205
 0.127655    0.56418    0.541102  0.661327  0.462087
 0.877254    0.849883   0.75203   0.127464  0.610566
 0.635287    0.239945   0.122218  0.497725  0.015649
```

```
In [59]: b=rand(5)
```

```
Out [59]: 5-element Array{Float64,1}:
 0.625647
 0.796183
 0.437029
 0.690231
 0.237018
```

```
In [62]: x=A\b # solving for unknown x such that Ax=b
```

```
Out [62]: 5-element Array{Float64,1}:
  0.0807766
  0.390174
  0.741278
  0.0169814
 -0.44526
```

```
In [63]: x=pinv(A)*b # solving for unknown x such that Ax=b
```

```
Out [63]: 5-element Array{Float64,1}:
  0.0807766
  0.390174
  0.741278
  0.0169814
 -0.44526
```

```
In [64]: A*x # Ax=b
```

```
Out [64]: 5-element Array{Float64,1}:
 0.625647
 0.796183
 0.437029
 0.690231
 0.237018
```

```
In [67]: l,v=eig(A)
```

```
Out [67]: ([2.196362464168248 + 0.0im,0.8199456549845309 +
0.0im,-0.10905071799417637 + 0.4821037975732591im,-0.10905071799417637
- 0.4821037975732591im,-0.553202273707936 + 0.0im],
5x5 Array{Complex{Float64},2}:
 0.453294+0.0im    0.591942+0.0im  0.237517+0.362899im  ...
-0.00207193+0.0im
 0.323786+0.0im   -0.686189+0.0im  0.232124+0.188288im
-0.0264009+0.0im
 0.475423+0.0im  -0.0431197+0.0im      -0.556921+0.0im
0.203077+0.0im
 0.595485+0.0im    0.191197+0.0im  0.192784-0.268164im
-0.751397+0.0im
 0.330239+0.0im    0.374611+0.0im   0.15865-0.527397im
0.627265+0.0im)
```

```
In [68]: methods(eig)
```

```
Out [68]: # 3 methods for generic function "eig":
eig(m::SymTridiagonal{T<:Union(Float32,Complex{Float64},Float64,Comple
x{Float32})}) at linalg/tridiag.jl:67
eig(A::AbstractArray{T,2},B::AbstractArray{T,2}) at
linalg/factorization.jl:579
eig(A::Union(Number,AbstractArray{T,2})) at
linalg/factorization.jl:502
```

```
In [69]: v
```
```
Out [69]: 5x5 Array{Complex{Float64},2}:
 0.453294+0.0im    0.591942+0.0im  0.237517+0.362899im  ...
-0.00207193+0.0im
 0.323786+0.0im   -0.686189+0.0im  0.232124+0.188288im
-0.0264009+0.0im
 0.475423+0.0im  -0.0431197+0.0im      -0.556921+0.0im
0.203077+0.0im
 0.595485+0.0im    0.191197+0.0im  0.192784-0.268164im
-0.751397+0.0im
 0.330239+0.0im    0.374611+0.0im   0.15865-0.527397im
0.627265+0.0im
```
```
In [70]: function f(x,y)
           return(x+y,x-y)
         end
```
```
Out [70]: f (generic function with 2 methods)
```
```
In [71]: (res1,res2)=f(1,2)
```
```
Out [71]: (3,-1)
```
```
In [71]: K=rand(2,2);
         (u,d,v)=svd(K);
```
```
In [72]: K
```
```
Out [72]: 2x2 Array{Float64,2}:
 0.146306  0.350496
 0.475416  0.94116
```
```
In [73]: v
```
```
Out [73]: 2x2 Array{Float64,2}:
 -0.443469   0.89629
 -0.89629   -0.443469
```
```
In [78]: u*diagm(d)*v'
```
```
Out [78]: 2x2 Array{Float64,2}:
 0.146306  0.350496
 0.475416  0.94116
```
```
In [79]: ndims(K)
```
```
Out [79]: 2
```
```
In [80]: size(K)
```
```
Out [80]: (2,2)
```
```
In [81]: typeof(K)
```
```
Out [81]: Array{Float64,2}
```
```
In [81]: x=randn(10);
```
```
In [82]: [(i>0)?"+":"-" for i in x]
```
```
Out [82]: 10-element Array{ASCIIString,1}:
 "-"
 "+"
 "+"
 "+"
 "-"
 "+"
 "+"
 "+"
 "+"
```

"+"