IntermediateProblemAnswers

May 16, 2018

1 Intermediate Problem Answers

1.1 MyRange and MyLinSpace Problem

1.1.1 Part 1

```
In [8]: struct MyRange
            start
            step
            stop
        end

        function _MyRange(a::MyRange,i::Int)
            tmp = a.start + a.step*(i-1)
            if tmp > a.stop
                error("Index is out of bounds!")
            else
                return tmp
            end
        end
```

```
Out[8]: _MyRange (generic function with 1 method)
```

```
In [9]: a = MyRange(1,2,20)
        _MyRange(a,5) == (1:2:20)[5]
```

```
Out[9]: true
```

```
In [12]: Base.getindex(a::MyRange,i::Int) = _MyRange(a,i)
         a[5]
```

```
Out[12]: 9
```

1.1.2 Part 2

```
In [13]: ?linspace
```

```
search: linspace LinSpace
```

```
Out[13]:

linspace(start, stop, n=50)
```

Construct a range of `n` linearly spaced elements from `start` to `stop`.

```
julia> linspace(1.3,2.9,9)
1.3:0.2:2.9
```

```
In [25]: struct MyLinSpace
             start
             stop
             n
         end

         function Base.getindex(a::MyLinSpace,i::Int)
             dx = (a.stop-a.start)/a.n
             a.start + dx*(i-1)
         end

In [26]: l = MyLinSpace(1,2,50)
         l[6]

Out[26]: 1.1

In [27]: linspace(1,2,50)[6]

Out[27]: 1.1020408163265305
```

1.1.3 Part 3

```
In [29]: (a::MyRange)(x) = a.start + a.step*(x-1)
         a = MyRange(1,2,20)
         a(1.1)

Out[29]: 1.2000000000000002
```

1.1.4 Part 4

```
In [32]: using Unitful
         a = MyRange(1u"kg",2u"kg",20u"kg")
         a[5]

Out[32]: 9 kg

search: end endof endswith ENDIAN_BOM send append! QuoteNode RangeIndex getindex
```

1.2 Operator Problem

```
In [56]: struct StrangMatrix end
         A = StrangMatrix()

         function Base.A_mul_B!(C,A::StrangMatrix,B::AbstractVector)
             for i in 2:length(B)-1
                 C[i] = B[i-1] - 2B[i] + B[i+1]
             end
             C[1] = -2B[1] + B[2]
             C[end] = B[end-1] - 2B[end]
             C
         end

         Base.:*(A::StrangMatrix,B::AbstractVector) = (C = similar(B); A_mul_B!(C,A
```

```
In [38]: A*ones(10)
```

```
Out[38]: 10-element Array{Float64,1}:
          -1.0
           0.0
           0.0
           0.0
           0.0
           0.0
           0.0
           0.0
           0.0
          -1.0
```

```
In [64]: struct SizedStrangMatrix
             size
         end

         Base.eltype(A::SizedStrangMatrix) = Float64
         Base.size(A::SizedStrangMatrix) = A.size
         Base.size(A::SizedStrangMatrix,i::Int) = A.size[i]
```

```
In [65]: b = sin.(0:0.1:2π)
```

```
Out[65]: 63-element Array{Float64,1}:
           0.0
           0.0998334
           0.198669
           0.29552
           0.389418
           0.479426
           0.564642
           0.644218
```

```
 0.717356
 0.783327
 0.841471
 0.891207
 0.932039
 ⋮
-0.925815
-0.883455
-0.832267
-0.772764
-0.70554
-0.631267
-0.550686
-0.464602
-0.373877
-0.279415
-0.182163
-0.0830894
```

```
In [66]: A = SizedStrangMatrix((length(b),length(b)))

         function Base.A_mul_B!(C,A::SizedStrangMatrix,B)
             for i in 2:length(B)-1
                 C[i] = B[i-1] - 2B[i] + B[i+1]
             end
             C[1] = -2B[1] + B[2]
             C[end] = B[end-1] - 2B[end]
             C
         end
         Base.:*(A::SizedStrangMatrix,B::AbstractVector) = (C = similar(B); A_mul_B
```

```
In [73]: using IterativeSolvers
         x = gmres(A,b,tol=1e-14)
```

```
Out[73]: 63-element Array{Float64,1}:
            -9.75574
           -19.5118
           -29.1685
           -38.6273
           -47.7917
           -56.5679
           -64.8662
           -72.6018
           -79.6955
           -86.0744
           -91.6729
           -96.4332
          -100.306
```

```
              ⋮
             92.7311
             88.6305
             83.6451
             77.8263
             71.2334
             63.934
             56.0017
             47.5193
             38.5723
             29.2516
             19.6511
              9.86787
```

```
In [75]: A*x - b
```

```
Out[75]: 63-element Array{Float64,1}:
          -0.000289122
          -0.000548773
          -0.000747631
          -0.00109653
          -0.00121106
          -0.00156251
          -0.001876
          -0.00234776
          -0.00257171
          -0.0029435
          -0.0031805
          -0.00349519
          -0.00378336
           ⋮
          -0.00139662
          -0.00132261
          -0.00122719
          -0.00123587
          -0.00103713
          -0.00164304
           0.000594124
           0.00010594
           9.1898e-5
          -0.000444569
          -0.000461939
          -0.00158795
```

1.3 Regression Problem

```
In [ ]: #### Prepare Data
```

```
X = rand(1000, 3)                # feature matrix
a0 = rand(3)                     # ground truths
y = X * a0 + 0.1 * randn(1000);  # generate response

X2 = hcat(X,ones(1000))
println(X2\y)

using MultivariateStats
println(llsq(X,y))

using DataFrames, GLM
data = DataFrame(X1=X[:,1], X2=X[:,2], X3=X[:,3],Y=y)
OLS = lm(@formula(Y ~ X1 + X2 + X3), data)


X = rand(100);
y = 2X  + 0.1 * randn(100);

using Plots
b = X\y
println(b)
gr()
scatter(X,y)
Plots.abline!(b[1],0.0, lw=3) # Slope,Intercept
```

1.4 Type Hierarchy Problem

In [80]:
```
abstract type AbstractPerson end
abstract type AbstractStudent <: AbstractPerson end

struct Person <: AbstractPerson
    name
end

struct Student <: AbstractStudent
    name
    grade
end

struct GraduateStudent <: AbstractStudent
    name
    grade
end

person_info(p::AbstractPerson) = println(p.name)
person_info(s::AbstractStudent) = (println(s.name); println(s.grade))
```

Out[80]:
```
person_info (generic function with 2 methods)
```

```
In [78]: person_info(Person("Bob"))

Bob


In [81]: person_info(Student("Bob",2))

Bob
2


In [82]: person_info(GraduateStudent("Bob",2))

Bob
2
```

1.5 Distribution Dispatch Problem

This is from Josh Day's talk: https://www.youtube.com/watch?v=EwcTNzpQ6Sc
Solution is from: https://github.com/joshday/Talks/blob/master/SLG2016_IntroToJulia/Slides.ipynb

```
In [ ]: function myquantile(d::UnivariateDistribution, q::Number)
            θ = mean(d)
            tol = Inf
            while tol > 1e-5
                θold = θ
                θ = θ - (cdf(d, θ) - q) / pdf(d, θ)
                tol = abs(θold - θ)
            end
            θ
        end

        for dist in [Gamma(5, 1), Normal(0, 1), Beta(2, 4)]
            @show myquantile(dist, .75)
            @show quantile(dist, .75)
            println()
        end
```