Российский университет дружбы народов

Факультет физико-математических и естественных наук

Отчёт по лабораторной работе №8

1032203967
Быстров Глеб

Москва 2023

Цель работы (задание)

- Освоить пакеты Julia для решения задач оптимизации

Задачи (метод выполнения)

- Линейное программирование

```
# Подключение пакетов:
import Pkg
Pkg.add("JuMP")
Pkg.add("GLPK")
using JuMP
using GLPK
```

```
    Updating registry at `C:\Users\GlebB\.julia\registries\G
   Resolving package versions...
   Installed CodecBzip2 ──────── v0.8.1
   Installed SnoopPrecompile ─── v1.0.3
```

```
# Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

```
# Определение переменных x, y и граничных условий для них:
@variable(model, x >= 0)
@variable(model, y >= 0)
```

y

```
# Определение ограничений модели:
@constraint(model, 6x + 8y >= 100)
@constraint(model, 7x + 12y >= 120)
```

$7x + 12y \geq 120$

```
# Определение целевой функции:
@objective(model, Min, 12x + 20y)
```

$12x + 20y$

```
# Вызов функции оптимизации:
optimize!(model)
```

```
# Определение причины завершения работы оптимизатора:
termination_status(model)
```

```
OPTIMAL::TerminationStatusCode = 1
```

```
# Демонстрация первичных результирующих значений переменных x и y:
@show value(x);
@show value(y);
# Демонстрация результата оптимизации:
@show objective_value(model);
```

```
value(x) = 14.999999999999993
value(y) = 1.2500000000000047
objective_value(model) = 205.0
```

Задачи (метод выполнения)

- Векторизованные ограничения и целевая функция оптимизации

```
# Определение объекта модели с именем vector_model:
vector_model = Model(GLPK.Optimizer)
```

```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

```
# Определение начальных данных:
A= [ 1 1 9 5;
3 5 0 8;
2 0 6 13]
b = [7; 3; 5]
c = [1; 3; 5; 2]
```

```
4-element Vector{Int64}:
 1
 3
 5
 2
```

```
# Определение вектора переменных:
@variable(vector_model, x[1:4] >= 0)
```

```
4-element Vector{VariableRef}:
 x[1]
 x[2]
 x[3]
 x[4]
```

```
# Определение ограничений модели:
@constraint(vector_model, A * x .== b)
```

```
3-element Vector{ConstraintRef{Model, MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64}, MathOptI
nterface.EqualTo{Float64}}, ScalarShape}}:
 x[1] + x[2] + 9 x[3] + 5 x[4] == 7
 3 x[1] + 5 x[2] + 8 x[4] == 3
 2 x[1] + 6 x[3] + 13 x[4] == 5
```

```
# Определение целевой функции:
@objective(vector_model, Min, c' * x)
```

$$x_1 + 3x_2 + 5x_3 + 2x_4$$

```
# Вызов функции оптимизации:
optimize!(vector_model)
```

```
# Определение причины завершения работы оптимизатора:
termination_status(vector_model)
```

```
OPTIMAL::TerminationStatusCode = 1
```

```
# Демонстрация результата оптимизации:
@show objective_value(vector_model);
```

```
objective_value(vector_model) = 4.9230769230769225
```

Задачи (метод выполнения)

- Оптимизация рациона питания

```
# Контейнер для хранения данных об ограничениях на количество потребляемых калорий, белков, жиров и соли:
category_data = JuMP.Containers.DenseAxisArray(
[1800 2200;
91 Inf;
0 65;
0 1779],
["calories", "protein", "fat", "sodium"],
["min", "max"])
```

```
2-dimensional DenseAxisArray{Float64,2,...} with index sets:
    Dimension 1, ["calories", "protein", "fat", "sodium"]
    Dimension 2, ["min", "max"]
And data, a 4×2 Matrix{Float64}:
 1800.0  2200.0
   91.0     Inf
    0.0    65.0
    0.0  1779.0
```

```
# массив данных с наименованиями продуктов:
foods = ["hamburger", "chicken", "hot dog", "fries", "macaroni", "pizza","salad", "milk", "ice cream"]
```

```
9-element Vector{String}:
 "hamburger"
 "chicken"
 "hot dog"
 "fries"
 "macaroni"
 "pizza"
```

```
4}, MathOptInterface.EqualTo{Float64}}, ScalarShape}}:
 -nutrition[calories] + 410 buy[hamburger] + 420 buy[chicken] + 560 buy[hot dog] + 380 buy[fries] + 320 buy[macaroni] + 320 buy
[pizza] + 320 buy[salad] + 100 buy[milk] + 330 buy[ice cream] == 0
 -nutrition[protein] + 24 buy[hamburger] + 32 buy[chicken] + 20 buy[hot dog] + 4 buy[fries] + 12 buy[macaroni] + 15 buy[pizza]
+ 31 buy[salad] + 8 buy[milk] + 8 buy[ice cream] == 0
 -nutrition[fat] + 26 buy[hamburger] + 10 buy[chicken] + 32 buy[hot dog] + 19 buy[fries] + 10 buy[macaroni] + 12 buy[pizza] + 1
2 buy[salad] + 2.5 buy[milk] + 10 buy[ice cream] == 0
 -nutrition[sodium] + 730 buy[hamburger] + 1190 buy[chicken] + 1800 buy[hot dog] + 270 buy[fries] + 930 buy[macaroni] + 820 buy
[pizza] + 1230 buy[salad] + 125 buy[milk] + 180 buy[ice cream] == 0
```

```
# Вызов функции оптимизации:
JuMP.optimize!(model)
term_status = JuMP.termination_status(model)
```

```
OPTIMAL::TerminationStatusCode = 1
```

```
hcat(buy.data,JuMP.value.(buy.data))
```

```
9×2 Matrix{AffExpr}:
 buy[hamburger]  0.6045138888888888
 buy[chicken]    0
 buy[hot dog]    0
 buy[fries]      0
 buy[macaroni]   0
 buy[pizza]      0
 buy[salad]      0
 buy[milk]       6.9701388888888935
 buy[ice cream]  2.591319444444441
```

Задачи (метод выполнения)

- Путешествие по миру

```
# Подключение пакетов:
import Pkg
Pkg.add("DelimitedFiles")
Pkg.add("CSV")
using DelimitedFiles
using CSV
```

```
  Resolving package versions...
   Updating `C:\Users\GlebB\.julia\environments\v1.9\Project.toml`
 [8bb1440f] + DelimitedFiles v1.9.1
 No Changes to `C:\Users\GlebB\.julia\environments\v1.9\Manifest.toml`
  Resolving package versions...
 No Changes to `C:\Users\GlebB\.julia\environments\v1.9\Project.toml`
 No Changes to `C:\Users\GlebB\.julia\environments\v1.9\Manifest.toml`
```

```
# Считывание данных:
passportdata = readdlm(joinpath("passport-index-matrix.csv"),',')
```

```
200×200 Matrix{Any}:
 "Passport"              "Albania"  …  "Afghanistan"
 "Afghanistan"           "e-visa"      -1
 "Albania"               -1            "visa required"
 "Algeria"               "e-visa"      "visa required"
 "Andorra"               90            "visa required"
 "Angola"                "e-visa"   …  "visa required"
 "Antigua and Barbuda"   90            "visa required"
 "Argentina"             90            "visa required"
 "Armenia"               90            "visa required"
 "Australia"             90            "visa required"
```

```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

```
# Переменные, ограничения и целевая функция:
@variable(model, pass[1:length(cntr)], Bin)
@constraint(model, [j=1:length(cntr)], sum( vf[i,j]*pass[i] for i in 1:length(cntr)) >= 1)
@objective(model, Min, sum(pass))
```

$$pass_1 + pass_2 + pass_3 + pass_4 + pass_5 + pass_6 + pass_7 + pass_8 + pass_9 + pass_{10} + pass_{11} + pass_{12} + pass_{13} + pass_{14} + pass_{15} + pass_{16} + pass_{17} + pass_{18} + pass_{19} + pass_{20} + pass_{21} + pass_{22} + pass_{23} + pass_{24} + pass_{25} + pass_{26} + pass_{27} + pass_{28} + pass_{29} + pass_{30} + [[\ldots\ 139\ \text{terms omitted}\ \ldots]] + pass_{170} + pass_{171} + pass_{172} + pass_{173} + pass_{174} + pass_{175} + pass_{176} + pass_{177} + pass_{178} + pass_{179} + pass_{180} + pass_{181} + pass_{182} + pass_{183} + pass_{184} + pass_{185} + pass_{186} + pass_{187} + pass_{188} + pass_{189} + pass_{190} + pass_{191} + pass_{192} + pass_{193} + pass_{194} + pass_{195} + pass_{196} + pass_{197} + pass_{198} + pass_{199}$$

```
# Вызов функции оптимизации:
JuMP.optimize!(model)
termination_status(model)
```

```
OPTIMAL::TerminationStatusCode = 1
```

```
# Просмотр результата:
print(JuMP.objective_value(model)," passports: ",join(cntr[findall(JuMP.value.(pass) .== 1)],", "))
```

```
34.0 passports: Afghanistan, Australia, Bahrain, Cameroon, Canada, Comoros, Congo, Denmark, Djibouti, Eritrea, Guinea-Bissau, Hong Kong, Iran, Kenya, Kuwait, Liberia, Libya, Madagascar, Maldives, Mauritania, Morocco, Nauru, Nepal, New Zealand, North Korea, Palestine, Papua New Guinea, Qatar, Saudi Arabia, Singapore, Somalia, Sri Lanka, Syria, Turkmenistan
```

Задачи (метод выполнения)

- Портфельные инвестиции

```
# Подключение необходимых пакетов:
import Pkg
Pkg.add("DataFrames")
Pkg.add("XLSX")
Pkg.add("Plots")
Pkg.add("PyPlot")
Pkg.add("Convex")
Pkg.add("SCS")
Pkg.add("Statistics")
using DataFrames
using XLSX
using Plots
pyplot()
using Convex
using SCS
using Statistics
```

```
in expression starting at stdin:3
ERROR: LoadError: Failed to precompile PyCall [438e738f-606a-5dbb-bf0a-cddfbfd45ab0] to "C:\\Users\\GlebB\\.julia\\compiled
\\v1.9\\PyCall\\jl_A8F2.tmp".
Stacktrace:
  [1] error(s::String)
    @ Base .\error.jl:35
  [2] compilecache(pkg::Base.PkgId, path::String, internal_stderr::IO, internal_stdout::IO, keep_loaded_modules::Bool)
    @ Base .\loading.jl:2294
  [3] compilecache
    @ .\loading.jl:2167 [inlined]
  [4] require(pkg::Base.PkgId, env::String)
```

Задачи (метод выполнения)

- Восстановление изображения

```
K = copy(Kref)
p = prod(size(K))
missingids = rand(1:p,400)
K[missingids] .= RGBX{N0f8}(0.0,0.0,0.0)
K
Gray.(K)
```

```
# Находим решение:
solve!(problem, SCS.Optimizer(eps=1e-3, alpha=1.5))
```

```
MethodError: no method matching SCS.Optimizer(; eps::Float64, alpha::Float64)

Closest candidates are:
  SCS.Optimizer() got unsupported keyword arguments "eps", "alpha"
   @ SCS C:\Users\GlebB\.julia\packages\SCS\mqg7w\src\MOI_wrapper\MOI_wrapper.jl:131


Stacktrace:
 [1] top-level scope
   @ In[67]:3
```

Задачи (метод выполнения)

- Задания для самостоятельного выполнения

```
#Задание №1
model = Model(GLPK.Optimizer)
```

```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

```
@variable(model, 0 <= x1 <= 10)
@variable(model, x2 >= 0)
@variable(model, x3 >= 0)
```

$x3$

```
@constraint(model, -x1 + x2 + 3x3 <= -5)
@constraint(model, x1 + 3x2 - 7x3 <= 10)
```

$x1 + 3x2 - 7x3 \leq 10$

```
@objective(model, Max, x1 + 2x2 + 5x3)
```

$x1 + 2x2 + 5x3$

```
optimize!(model)
```

```
termination_status(model)
```

```
OPTIMAL::TerminationStatusCode = 1
```

```
@show value(x1);
@show value(x2);
@show value(x3);

@show objective_value(model);
```

```
value(x1) = 10.0
value(x2) = 2.1875
value(x3) = 0.9375
objective_value(model) = 19.0625
```

Задачи (метод выполнения)

- Задания для самостоятельного выполнения

```
#Задание №2
vector_model_2 = Model(GLPK.Optimizer)
```

```
A JuMP Model
Feasibility problem with:
Variables: 0
Model mode: AUTOMATIC
CachingOptimizer state: EMPTY_OPTIMIZER
Solver name: GLPK
```

```
A= [-1 1 3;
    1 3 -7]
b = [-5; 10]
c = [1; 2; 5]
```

```
3-element Vector{Int64}:
 1
 2
 5
```

```
@variable(vector_model_2, x[1:3] >= 0)
set_upper_bound(x[1], 10)
```

```
@constraint(vector_model_2, A * x .== b)
```

```
2-element Vector{ConstraintRef{Model, MathOptInter
nterface.EqualTo{Float64}}, ScalarShape}}:
 -x[1] + x[2] + 3 x[3] == -5
 x[1] + 3 x[2] - 7 x[3] == 10
```

```
@objective(vector_model_2, Max, c' * x)
```

$$x_1 + 2x_2 + 5x_3$$

```
optimize!(vector_model_2)
```

```
termination_status(vector_model_2)
```

```
OPTIMAL::TerminationStatusCode = 1
```

```
@show value(x[1]);
@show value(x[2]);
@show value(x[3]);
@show objective_value(vector_model_2);
```

```
value(x[1]) = 10.0
value(x[2]) = 2.1875
value(x[3]) = 0.9375
objective_value(vector_model_2) = 19.0625
```

Задачи (метод выполнения)

- Задания для самостоятельного выполнения

```
#Задание №3
using Convex
using SCS

m = 5
n = 4

A = rand(m, n)
b = rand(m)

display(A)
println()
display(b)

x = Variable(n)

display(x)

model_new = minimize(Convex.sumsquares(A*x - b),[x >= 0])

solve!(model_new, SCS.Optimizer)

model_new.status

model_new.optval
```

```
5×4 Matrix{Float64}:
 0.773851   0.731559  0.357961   0.927942
 0.314247   0.176215  0.0499934  0.0422559
 0.466721   0.716408  0.0630813  0.348065
 0.0164345  0.729675  0.735458   0.333719
```

```
sign: real
vexity: affine
id: 144…234

------------------------------------------------------------------
               SCS v3.2.4 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
problem:  variables n: 7, constraints m: 15
cones:    z: primal zero / dual free vars: 1
          l: linear vars: 5
          q: soc vars: 9, qsize: 2
settings: eps_abs: 1.0e-004, eps_rel: 1.0e-004, eps_infeas: 1.0e-007
          alpha: 1.50, scale: 1.00e-001, adaptive_scale: 1
          max_iters: 100000, normalize: 1, rho_x: 1.00e-006
          acceleration_lookback: 10, acceleration_interval: 10
lin-sys:  sparse-direct-amd-qdldl
          nnz(A): 30, nnz(P): 0
------------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
------------------------------------------------------------------
     0|1.71e+001 1.00e+000 1.62e+001 -8.04e+000 1.00e-001 4.58e-003
   125|1.00e-005 5.56e-006 1.04e-005 4.84e-002 5.56e-001 4.71e-003
------------------------------------------------------------------
status:  solved
timings: total: 4.71e-003s = setup: 4.52e-003s + solve: 1.91e-004s
         lin-sys: 5.93e-005s, cones: 3.31e-005s, accel: 7.20e-006s
------------------------------------------------------------------
objective = 0.048427
------------------------------------------------------------------

0.048422158379626865
```

Результаты и их анализ

- Успешно удалось освоить пакеты Julia для решения задач оптимизации

Благодарю за внимание