РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

дисциплина: Компьютерный практикум по статистическому анализу данных

Студент: Евдокимов Максим Михайлович (1032203019)

Группа: НФИбд-01-20

МОСКВА

2023 г.

Постановка задачи

Основная цель работа — освоить пакеты Julia для решения задач оптимизации.

Выполнение работы

1. Решите задачу линейного программирования:

$x_1 + 2x_2 + 5x_3 \to \max$

при заданных ограничениях:

$-x_1 + x_2 + 3x_3 \leqslant -5, \quad x_1 + 3x_2 - 7x_3 \leqslant 10, \quad 0 \leqslant x_1 \leqslant 10, \quad x_2 \geqslant 0, \quad x_3 \geqslant 0$

```
# 1
model = Model(GLPK.Optimizer) # основа-модель
@variable(model, x[1:3] >= 0) # начинка-переменные
@objective(model, Max, x[1] + 2x[2] + 5x[3]) # осн. условие-задача
# условия-ограничения
@constraint(model, -x[1] + x[2] + 3x[3] <= -5)
@constraint(model, x[1] + 3x[2] - 7*x[3] <= 10)
@constraint(model, 0 <= x[1] <= 10)
optimize!(model) # запуск вычислений
println("Оптимальное значение целевой функции: ", objective_value(model))
println("Оптимальное значение переменных: ", value.(x))
```

```
Оптимальное значение целевой функции: 19.0625
Оптимальное значение переменных: [10.0, 2.1875, 0.9375]
```

2. Решите предыдущее задание, используя массивы вместо скалярных переменных. Рекомендация. Запишите систему ограничений в виде A*x = b, а целевую функцию как $c^T x$

```
# 2
model = Model(GLPK.Optimizer)
c = [1, 2, 5]
A = [-1 1 3; 1 3 -7]
b = [-5, 10]

@variable(model, x[1:3] >= 0)
@constraint(model, 0 <= x[1] <= 10)
@objective(model, Max, transpose(c)*x)
@constraint(model, A * x .<= b)
optimize!(model)

println("Оптимальное значение целевой функции: ", objective_value(model))
println("Оптимальное значение переменных: ", value.(x))
```

```
Оптимальное значение целевой функции: 19.0625
Оптимальное значение переменных: [10.0, 2.1875, 0.9375]
```

3. Решите задачу оптимизации:

$$\|A\vec{x} - \vec{b}\|_2^2 \to \min$$

при заданных ограничениях:

$$\vec{x} \succeq 0$$

где x∈R^n, A∈R^{mn}, b∈R^m. Матрицу A и вектор $\vec{b}$ задайте случайным образом. Для решения задачи используйте пакет Convex и решатель SCS.

```
# 3
n = rand(3:5)
m = n

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

display(A); display(b); display(x)

objective = minimize(square(norm(A * x - b, 2)), x >= 0)
solve!(objective, SCS.Optimizer)

println("Optimal value: ", objective.optval)
println("Optimal solution: ", evaluate(x))
```

```
4×4 Matrix{Float64}:
 0.847375  0.313474  0.607809  0.53913
 0.436363  0.258419  0.26081   0.519515
 0.466991  0.619651  0.688406  0.541643
 0.250513  0.763841  0.565798  0.305177
4-element Vector{Float64}:
 0.22318782752603272
 0.6401742230098822
 0.3576409210612268
 0.26400715774758354
Variable
size: (4, 1)
sign: real
vexity: affine
id: 445…192
```

```
------------------------------------------------------------------
           SCS v3.2.4 - Splitting Conic Solver
       (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
problem:  variables n: 7, constraints m: 14
cones:    z: primal zero / dual free vars: 1
          l: linear vars: 5
          q: soc vars: 8, 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): 26, nnz(P): 0
------------------------------------------------------------------
 iter | pri res | dua res |   gap   |   obj   |  scale  | time (s)
------------------------------------------------------------------
     0|1.71e+001 1.00e+000 1.62e+001 -8.05e+000 1.00e-001 1.37e-004
   125|7.32e-006 3.40e-006 1.12e-005 9.89e-002 5.37e-001 2.16e-004
------------------------------------------------------------------
status:  solved
timings: total: 2.17e-004s = setup: 1.01e-004s + solve: 1.17e-004s
         lin-sys: 3.49e-005s, cones: 1.97e-005s, accel: 5.00e-006s
------------------------------------------------------------------
objective = 0.098878
------------------------------------------------------------------
Optimal value: 0.09887255869075212
Optimal solution: [-8.10312975053561e-7, 3.976974383540526e-7, -9.435644882564067e-7, 0.7678336809741355]
```

4. Задача оптимальная рассадка по залам линейного программирования:

Проводится конференция с 5 разными секциями. Забронировано 5 залов различной вместимости: в каждом зале не должно быть меньше 180 и больше 250 человек, а на третьей секции активность подразумевает, что должно быть точно 220 человек. В заявке участник указывает приоритет посещения секции: 1 — максимальный приоритет, 3 — минимальный, а значение 10000 означает, что человек не пойдёт на эту секцию. Организаторам удалось собрать 1000 заявок с указанием приоритета посещения трёх секций.

Необходимо дать рекомендацию слушателю, на какую же секцию ему пойти, чтобы хватило места всем. Для решения задачи используйте пакет Convex и решатель GLPK. Приоритеты по слушателям распределите случайным образом.

```
# 4
num_rooms = 5
num_sections = 5
num_participants = 1000
min_capacity = 180
max_capacity = 250
exact_capacity = 220

model = Model(GLPK.Optimizer)
priorities = rand(0:3, (num_participants, num_sections))
# я заменил 10000 на 0: проще записать, легче воспринимать, логичнее

@variables(model, begin x[1:num_sections] >= 0, Int end)

for i in 1:num_sections
    @constraint(model, min_capacity <= x[i] <= max_capacity)
end

@constraint(model, x[3] == exact_capacity)
@constraint(model, sum(x) >= num_participants)
@objective(model, Min, sum(priorities .* x'))

optimize!(model)
recommendation = argmin(value.(x))

println("Насколько заняты каждый из секций: $(value.(x))")
println("Рекомендация для участника 1001: секция $recommendation")
```

```
Насколько заняты каждый из секций: [180.0, 240.0, 220.0, 180.0, 180.0]
Рекомендация для участника 1001: секция 1
```

5. Задача плана приготовления кофе линейного программирования:

Кофейня готовит два вида кофе «Раф кофе» за 400 рублей и «Капучино» за 300. Чтобы сварить 1 чашку «Раф кофе» необходимо: 40 гр. зёрен, 140 гр. молока и 5 гр. ванильного сахара. Для того чтобы получить одну чашку «Капучино» необходимо потратить: 30 гр. зёрен, 120 гр. молока. На складе есть: 500 гр. зёрен, 2000 гр. молока и 40 гр. Ванильного сахара. Необходимо найти план варки кофе, обеспечивающий максимальную выручку от их

реализации. При этом необходимо потратить весь ванильный сахар. Для решения задачи используйте пакет JuMP и решатель GLPK.

```
# 5
model = Model(GLPK.Optimizer)
@variable(model, raf >= 0)
@variable(model, cappuccino >= 0)

const grain_limit = 500
@constraint(model, raf * 40 + cappuccino * 30 <= grain_limit)

const milk_limit = 2000
@constraint(model, raf * 140 + cappuccino * 120 <= milk_limit)

const sugar_limit = 40
@constraint(model, raf * 5 <= sugar_limit)

objective = 400 * raf + 300 * cappuccino
@objective(model, Max, objective)

optimize!(model)

println("Раф кофе: ", round(value(raf)))
println("Капучино: ", round(value(cappuccino)))
println("Прибыль: ", value(objective))
```

```
Раф кофе: 8.0
Капучино: 6.0
Прибыль: 5000.0
```

Выводы

В ходе выполнения лабораторной работы были изучены принципы работы линейного программирования и приобретены навыки по работе с библиотеками и интсрументами с его задачами на языке Julia.