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 данных анализ

Студент: Доре Стевенсон Эдгар

Группа: НКН-бд-01-19

МОСКВА

2023 г.

Лабораторная работа №8. Оптимизация

Цель работы:

Основная цель работа — освоить пакеты Julia для решения задач оптимизации

Ход работы:

8.2.1. Линейное программирование

8.2.1. Линейное программирование

In [1]:
```
# Подключение пакетов:
import Pkg
Pkg.add("JuMP")
Pkg.add("GLPK")
using JuMP
using GLPK
```

```
  Updating registry at `C:\Users\Admin\.julia\registries\General`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [2]:
```
# Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

Out[2]:
feasibility
Subject to

In [3]:
```
# Определение переменных x, y и граничных условий для них:
@variable(model, x >= 0)
@variable(model, y >= 0)
```

Out[3]: y

In [4]:
```
# Определение ограничений модели:
@constraint(model, 6x + 8y >= 100)
@constraint(model, 7x + 12y >= 120)
```

Out[4]: $7x + 12y \geq 120.0$

In [5]:
```
# Определение целевой функции:
@objective(model, Min, 12x + 20y)
```

Out[5]: $12x + 20y$

In [6]:
```
# Вызов функции оптимизации:
optimize!(model)
```

In [7]:
```
# Определение причины завершения работы оптимизатора:
termination_status(model)
```

Out[7]: OPTIMAL::TerminationStatusCode = 1

In [8]:
```
# Демонстрация первичных результирующих значений переменных x и y:
@show value(x);
@show value(y);
```

```
value(x) = 14.999999999999993
value(y) = 1.2500000000000047
```

In [9]:
```
# Демонстрация результата оптимизации:
@show objective_value(model);
```

```
objective_value(model) = 205.0
```

8.2.2. Векторизованные ограничения и целевая функция оптимизации

8.2.2. Векторизованные ограничения и целевая функция оптимизации

In [10]:
```
# Подключение пакетов:
import Pkg
Pkg.add("JuMP")
Pkg.add("GLPK")
using JuMP
using GLPK
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [11]:
```
# Определение объекта модели с именем vector_model:
vector_model = Model(GLPK.Optimizer)
```

Out[11]:
feasibility
Subject to

In [12]:
```
# 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]
```

Out[12]:
```
4-element Array{Int64,1}:
 1
 3
 5
 2
```

In [13]:
```
# Определение вектора переменных:
@variable(vector_model, x[1:4] >= 0)
```

Out[13]:
```
4-element Array{VariableRef,1}:
 x[1]
 x[2]
 x[3]
 x[4]
```

In [14]:
```
# Определение ограничений модели:
@constraint(vector_model, A * x .== b)
```

Out[14]:
```
3-element Array{ConstraintRef{Model,MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},MathOptInte
rface.EqualTo{Float64}},ScalarShape},1}:
 x[1] + x[2] + 9 x[3] + 5 x[4] == 7.0
 3 x[1] + 5 x[2] + 8 x[4] == 3.0
 2 x[1] + 6 x[3] + 13 x[4] == 5.0
```

In [15]:
```
# Определение целевой функции:
@objective(vector_model, Min, c' * x)
```

Out[15]: $x_1 + 3x_2 + 5x_3 + 2x_4$

In [16]:
```
# Вызов функции оптимизации:
optimize!(vector_model)
```

In [17]:
```
# Определение причины завершения работы оптимизатора:
termination_status(vector_model)
```

Out[17]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [18]:
```
# Демонстрация результата оптимизации:
@show objective_value(vector_model);
```

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

8.2.3. Оптимизация рациона питания

8.2.3. Оптимизация рациона питания

In [19]:
```
# Подключение пакетов:
import Pkg
Pkg.add("JuMP")
Pkg.add("GLPK")
using JuMP
using GLPK
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [20]:
```
# 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"])
```

Out[20]:
```
2-dimensional DenseAxisArray{Float64,2,...} with index sets:
    Dimension 1, ["calories", "protein", "fat", "sodium"]
    Dimension 2, ["min", "max"]
And data, a 4×2 Array{Float64,2}:
 1800.0  2200.0
   91.0    Inf
    0.0    65.0
    0.0  1779.0
```

In [21]:
```
# массив данных с наименованиями продуктов:
foods = ["hamburger", "chicken", "hot dog", "fries", "macaroni",
 "pizza","salad", "milk", "ice cream"]
```

Out[21]:
```
9-element Array{String,1}:
 "hamburger"
 "chicken"
 "hot dog"
 "fries"
 "macaroni"
 "pizza"
 "salad"
 "milk"
 "ice cream"
```

In [22]:
```
# Массив стоимости продуктов:
cost = JuMP.Containers.DenseAxisArray(
[2.49, 2.89, 1.50, 1.89, 2.09, 1.99, 2.49, 0.89, 1.59],
foods)
```

Out[22]:
```
1-dimensional DenseAxisArray{Float64,1,...} with index sets:
    Dimension 1, ["hamburger", "chicken", "hot dog", "fries", "macaroni", "pizza", "salad", "milk", "ice cream"]
And data, a 9-element Array{Float64,1}:
 2.49
 2.89
 1.5
 1.89
 2.09
 1.99
 2.49
 0.89
 1.59
```

In [23]:
```
# Массив данных о содержании калорий, белков, жиров и соли в продуктах питания:
food_data = JuMP.Containers.DenseAxisArray(
[410 24 26 730;
420 32 10 1190;
560 20 32 1800;
380 4 19 270;
320 12 10 930;
320 15 12 820;
320 31 12 1230;
100 8 2.5 125;
330 8 10 180],
foods,
["calories", "protein", "fat", "sodium"])
```

Out[23]:
```
2-dimensional DenseAxisArray{Float64,2,...} with index sets:
    Dimension 1, ["hamburger", "chicken", "hot dog", "fries", "macaroni", "pizza", "salad", "milk", "ice cream"]
    Dimension 2, ["calories", "protein", "fat", "sodium"]
And data, a 9×4 Array{Float64,2}:
 410.0  24.0  26.0   730.0
 420.0  32.0  10.0  1190.0
 560.0  20.0  32.0  1800.0
 380.0   4.0  19.0   270.0
 320.0  12.0  10.0   930.0
 320.0  15.0  12.0   820.0
 320.0  31.0  12.0  1230.0
 100.0   8.0   2.5   125.0
 330.0   8.0  10.0   180.0
```

In [24]:
```
# Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

Out[24]:
feasibility
Subject to

In [25]:
```
# Определим массив:
categories = ["calories", "protein", "fat", "sodium"]
```

Out[25]:
```
4-element Array{String,1}:
 "calories"
 "protein"
 "fat"
 "sodium"
```

In [26]:
```
# Определение переменных:
@variables(model, begin
category_data[c, "min"] <= nutrition[c = categories] <= category_data[c, "max"]
# Сколько покупать продуктов:
buy[foods] >= 0
end)
```

In [27]:
```
# Определение целевой функции:
@objective(model, Min, sum(cost[f] * buy[f] for f in foods))
```

Out[27]:
$$2.49buy_{hamburger} + 2.89buy_{chicken} + 1.5buy_{hotdog} + 1.89buy_{fries} + 2.09buy_{macaroni} + 1.99buy_{pizza} + 2.49buy_{salad} + 0.89buy_{milk} + 1.59buy_{icecream}$$

In [28]:
```
# Определение ограничений модели:
@constraint(model, [c in categories],
sum(food_data[f, c] * buy[f] for f in foods) == nutrition[c])
```

Out[28]:
```
1-dimensional DenseAxisArray{ConstraintRef{Model,MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float6
4},MathOptInterface.EqualTo{Float64}},ScalarShape},1,...} with index sets:
    Dimension 1, ["calories", "protein", "fat", "sodium"]
And data, a 4-element Array{ConstraintRef{Model,MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float6
4},MathOptInterface.EqualTo{Float64}},ScalarShape},1}:
 -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] == 0.0
 -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] == 0.0
 -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] == 0.0
```

In [29]:
```
# Вызов функции оптимизации:
JuMP.optimize!(model)
term_status = JuMP.termination_status(model)
```

Out[29]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [30]:
```
#Для просмотра результата решения модно вывести значение переменной buy:
hcat(buy.data,JuMP.value.(buy.data))
```

Out[30]:
```
9×2 Array{GenericAffExpr{Float64,VariableRef},2}:
 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
```

8.2.4. Путешествие по миру

8.2.4. Путешествие по миру

In [31]:
```
# Скачиваем данные с ресурса на git:
```

In [32]:
```
# Подключение пакетов:
import Pkg
Pkg.add("DelimitedFiles")
Pkg.add("CSV")
using DelimitedFiles
using CSV
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [33]:
```
# Считывание данных:
passportdata = readdlm("passport-index-matrix.csv",',')
```

Out[33]:
```
200×200 Array{Any,2}:
 "Passport"               "Albania"        …    "Afghanistan"
 "Afghanistan"            "visa required"       -1
 "Albania"               -1                      "visa required"
 "Algeria"                "visa required"        "visa required"
 "Andorra"               90                      "visa required"
 "Angola"                 "visa required"   …    "visa required"
 "Antigua and Barbuda"   90                      "visa required"
 "Argentina"             90                      "visa required"
 "Armenia"               90                      "visa required"
 "Australia"             90                      "visa required"
 "Austria"               90                   …  "visa required"
 "Azerbaijan"            90                      "visa required"
```

In [34]:
```
# Подключение пакетов:
Pkg.add("JuMP")
Pkg.add("GLPK")
using JuMP
using GLPK
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
```

In [35]:
```
# Задаём переменные:
cntr = passportdata[2:end,1]
vf = (x -> typeof(x)==Int64 || x == "VF" || x == "VOA" ? 1 : 0).(passportdata[2:end,2:end]);
```

In [36]:
```
# Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

Out[36]:
feasibility
Subject to

In [37]:
```
# 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))
```

Out[37]:
$$pass_1 + pass_2 + pass_3 + pass_4 + pass_5 + pass_6 + pass_7 + pass_8 + pass_9 + pass_{10} + pass_{11} + pass_{12} + pass_{13} + pass_{14} + pass_{15} + pass_{16} + pass_{17} + pass_{18} + pass_{19} + pass_{20} + pass_{21} + pass_{22} + pass_{23} + pass_{24} + pass_{25} + pass_{26} + pass_{27} + pass_{28} + pass_{29} + pass_{30} + pass_{31} + pass_{32} + pass_{33} + pass_{34} + pass_{35} + pass_{36} + pass_{37} + pass_{38} + pass_{39} + pass_{40} + pass_{41} + pass_{42} + pass_{43} + pass_{44} + pass_{45} + pass_{46} + pass_{47} + pass_{48} + pass_{49} + pass_{50} + pass_{51} + pass_{52} + pass_{53} + pass_{54} + pass_{55} + pass_{56} + pass_{57} + pass_{58} + pass_{59} + pass_{60} + pass_{61} + pass_{62} + pass_{63} + pass_{64} + pass_{65} + pass_{66} + pass_{67} + pass_{68} + pass_{69} + pass_{70} + pass_{71} + pass_{72} + pass_{73} + pass_{74} + pass_{75} + pass_{76} + pass_{77} + pass_{78} + pass_{79} + pass_{80} + pass_{81} + pass_{82} + pass_{83} + pass_{84} + pass_{85} + pass_{86} + pass_{87} + pass_{88} + pass_{89} + pass_{90} + pass_{91} + pass_{92} + pass_{93} + pass_{94} + pass_{95} + pass_{96} + pass_{97} + pass_{98} + pass_{99} + pass_{100} + pass_{101} + pass_{102} + pass_{103} + pass_{104} + pass_{105}$$

In [38]:
```
# Вызов функции оптимизации:
JuMP.optimize!(model)
termination_status(model)
```

Out[38]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [39]:
```
# Просмотр результата:
print(JuMP.objective_value(model)," passports:",join(cntr[findall(JuMP.value.(pass) .== 1)],", "))
```

```
63.0 passports:Afghanistan, Andorra, Argentina, Australia, Azerbaijan, Bahrain, Brunei, Cambodia, Cameroon, Canada, Chile, Colo
mbia, Comoros, DR Congo, Djibouti, Equatorial Guinea, Eritrea, Fiji, Gabon, Georgia, Guinea, Guinea-Bissau, Hong Kong, Hungary,
Indonesia, Iraq, Ireland, Israel, Jamaica, Japan, Kuwait, Laos, Liberia, Libya, Macao, Madagascar, Malaysia, Maldives, Marshall
Islands, Mauritania, Mauritius, Mongolia, Mozambique, Nauru, Nepal, New Zealand, North Korea, Palestine, Papua New Guinea, Qata
r, Saudi Arabia, Solomon Islands, Somalia, South Sudan, Sri Lanka, Syria, Taiwan, Timor-Leste, Togo, Turkmenistan, United State
s, Uruguay, Vietnam
```

8.2.5. Портфельные инвестиции

8.2.5. Портфельные инвестиции

In [40]:
```
# 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
```

```
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Manifest.toml`
  Resolving package versions...
No Changes to `C:\Users\Admin\.julia\environments\v1.5\Project.toml`
```

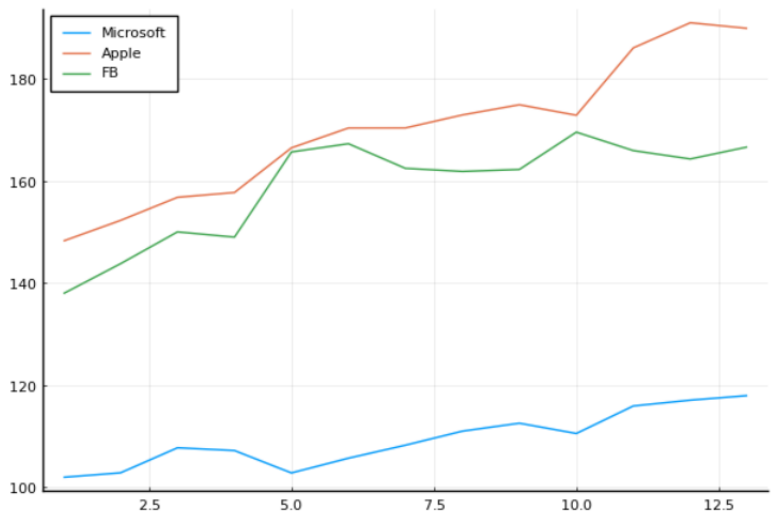
In [41]:
```
# Считываем данные и размещаем их во фрейм:
T = DataFrame(XLSX.readtable("stock_prices.xlsx","Sheet2")...)
```

Out[41]: 13 rows × 3 columns

	MSFT	FB	AAPL
	Any	Any	Any
1	101.93	137.95	148.26
2	102.8	143.8	152.29
3	107.71	150.04	156.82
4	107.17	149.01	157.76
5	102.78	165.71	166.52
6	105.67	167.33	170.41
7	108.22	162.5	170.42
8	110.97	161.89	172.97
9	112.53	162.28	174.97
10	110.51	169.6	172.91
11	115.91	165.98	186.12
12	117.05	164.34	191.05
13	117.94	166.69	189.95

In [42]:
```
# Построение графика:
plot(T[!,:MSFT],label="Microsoft")
plot!(T[!,:AAPL],label="Apple")
plot!(T[!,:FB],label="FB")
```

Out[42]:



In [43]:
```
# Данные о ценах на акции размещаем в матрице:
prices_matrix = Matrix(T)
```

Out[43]:
```
13×3 Array{Any,2}:
 101.93  137.95  148.26
 102.8   143.8   152.29
 107.71  150.04  156.82
 107.17  149.01  157.76
 102.78  165.71  166.52
 105.67  167.33  170.41
 108.22  162.5   170.42
 110.97  161.89  172.97
```

In [44]:
```
# Вычисление матрицы доходности за период времени:
M1 = prices_matrix[1:end-1,:]
M2 = prices_matrix[2:end,:]
```

Out[44]:
```
12×3 Array{Any,2}:
 102.8   143.8   152.29
 107.71  150.04  156.82
 107.17  149.01  157.76
 102.78  165.71  166.52
 105.67  167.33  170.41
 108.22  162.5   170.42
 110.97  161.89  172.97
 112.53  162.28  174.97
 110.51  169.6   172.91
 115.91  165.98  186.12
 117.05  164.34  191.05
 117.94  166.69  189.95
```

In [45]:
```
# Матрица доходности:
R = (M2.-M1)./M1
```

Out[45]:
```
12×3 Array{Float64,2}:
  0.00853527   0.0424067    0.027182
  0.0477626    0.0433936    0.0297459
 -0.00501346  -0.00686484   0.00599413
 -0.040963     0.112073     0.0555274
  0.0281183    0.00977611   0.0233606
  0.0241317   -0.0288651    5.8682e-5
  0.0254112   -0.00375385   0.014963
  0.0140579    0.00240904   0.0115627
 -0.0179508    0.0451072   -0.0117734
  0.0488644   -0.0213443    0.0763981
  0.00983522  -0.00988071   0.0264883
  0.00760359   0.0142996   -0.00575766
```

In [46]:
```
# Матрица рисков:
risk_matrix = cov(R)
```

Out[46]:
```
3×3 Array{Float64,2}:
  0.000659383  -0.000630653  0.000139112
 -0.000630653   0.00152162   0.000192288
  0.000139112   0.000192288  0.000635503
```

In [47]:
```
# Проверка положительной определённости матрицы рисков:
isposdef(risk_matrix)
```

Out[47]:
```
true
```

In [48]:
```
# Доход от каждой из компаний:
r = mean(R,dims=1)[:]
```

Out[48]:
```
3-element Array{Float64,1}:
 0.012532748705136572
 0.016563036855293173
 0.02114580465503291
```

In [49]:
```
# Вектор инвестиций:
x = Variable(length(r))
```

Out[49]:
```
Variable
size: (3, 1)
sign: real
vexity: affine
id: 111…765
```

In [50]:
```
# Объект модели:
problem = minimize(Convex.quadform(x,risk_matrix),[sum(x)==1;r'*x>=0.02;x.>=0])
```

Out[50]:
```
minimize
└─ * (convex; positive)
   ├─ 1
   └─ qol_elem (convex; positive)
      ├─ norm2 (convex; positive)
      └─ norm2 (convex; positive)
         └─ …
      └─ [1.0]
subject to
├─ == constraint (affine)
│  ├─ sum (affine; real)
│  │  └─ 3-element real variable (id: 111…765)
│  └─ 1
├─ >= constraint (affine)
│  ├─ * (affine; real)
│  │  ├─ [0.0125327 0.016563 0.0211458]
│  │  └─ 3-element real variable (id: 111…765)
│  └─ 0.02
├─ >= constraint (affine)
│  ├─ index (affine; real)
│  │  └─ 3-element real variable (id: 111…765)
│  └─ 0
├─ >= constraint (affine)
│  ├─ index (affine; real)
│  │  └─ 3-element real variable (id: 111…765)
│  └─ 0
└─ >= constraint (affine)
   ├─ index (affine; real)
   │  └─ 3-element real variable (id: 111…765)
   └─ 0

status: `solve!` not called yet
```

In [51]:
```
# Находим решение:
solve!(problem, SCS.Optimizer)
```

```
------------------------------------------------------------------------------
        SCS v2.1.2 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------------------
Lin-sys: sparse-indirect, nnz in A = 24, CG tol ~ 1/iter^(2.00)
eps = 1.00e-005, alpha = 1.50, max_iters = 5000, normalize = 1, scale = 1.00
acceleration_lookback = 10, rho_x = 1.00e-003
```

In [52]:
```
#Проверяем выполнение условия
sum(x.value)
```

Out[52]:
```
1.0000000000510323
```

In [53]:
```
#Проверяем выполнение условия на уровень доходности от 2%:
r'*x.value
```

Out[53]:
```
1×1 LinearAlgebra.Adjoint{Float64,Array{Float64,1}}:
 0.020000000000662013
```

In [54]:
```
#Переводим процентные значения компонент вектора инвестиций в фактические денежные единицы:
x.value .* 1000
```

Out[54]:
```
3×1 Array{Float64,2}:
  67.95414742252918
 122.30857118794069
 809.7372814405625
```

8.2.6. Восстановление изображения

8.4. Задания для самостоятельного выполнения

8.4.1. Линейное программирование

In [1]:
```
# Подключение пакетов:
using JuMP
using GLPK
```

In [2]:
```
# Определение объекта модели с именем model:
task_1 = Model(GLPK.Optimizer)
```

Out[2]:
feasibility
Subject to

In [3]:
```
# Определение переменных x1, x2, x3 и граничных условий для них:
@variable(task_1, 0 <= x1 <= 10)
@variable(task_1, x2 >= 0)
@variable(task_1, x3 >= 0)
# Определение ограничений модели:
@constraint(task_1, -x1 + x2 +3x3 <= -5)
@constraint(task_1, x1 + 3x2 - 7x3 <= 10)
# Определение целевой функции:
@objective(task_1, Max, x1 + 2x2 +5x3)
```

Out[3]:
$$x1 + 2x2 + 5x3$$

In [4]:
```
# Вызов функции оптимизации:
optimize!(task_1)
# Определение причины завершения работы оптимизатора:
termination_status(task_1)
```

Out[4]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [5]:
```
# Демонстрация первичных результирующих значений переменных x1, x2 и x3:
@show value(x1);
@show value(x2);
@show value(x3);
```

```
value(x1) = 10.0
```

In [5]:
```
# Демонстрация первичных результирующих значений переменных x1, x2 и x3:
@show value(x1);
@show value(x2);
@show value(x3);
```

```
value(x1) = 10.0
value(x2) = 2.1875
value(x3) = 0.9375
```

In [6]:
```
# Демонстрация результата оптимизации:
@show objective_value(task_1);
```

```
objective_value(task_1) = 19.0625
```

8.4.2. Линейное программирование. Использование массивов

In [7]:
```
# Определение объекта модели с именем vector_model:
task_2 = Model(GLPK.Optimizer)
```

Out[7]:

feasibility

Subject to

In [8]:
```
# Определение начальных данных:
A= [ -1 1 3;
     1 3 -7 ]
b = [-5; 10]
c = [1; 2; 5];
```

In [9]:
```
# Определение вектора переменных и граничных условий для них:
@variable(task_2, x[1:3] >= 0)
set_upper_bound(x[1], 10) # 0 ⩽ x1 ⩽ 10
```

In [10]:
```
# Определение ограничений модели:
@constraint(task_2, A * x .== b)
```

Out[10]:
```
2-element Array{ConstraintRef{Model,MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float64},MathOptInte
rface.EqualTo{Float64}},ScalarShape},1}:
 -x[1] + x[2] + 3 x[3] = -5.0
 x[1] + 3 x[2] - 7 x[3] = 10.0
```

In [11]:
```
# Определение целевой функции:
@objective(task_2, Max, c' * x)
```

Out[11]: $x_1 + 2x_2 + 5x_3$

In [12]:
```
# Вызов функции оптимизации:
optimize!(task_2)
```

In [13]:
```
# Определение причины завершения работы оптимизатора:
termination_status(task_2)
```

Out[13]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [14]:
```
# Демонстрация результата оптимизации:
@show objective_value(task_2);
```

```
objective_value(task_2) = 19.0625
```

8.4.3. Выпуклое программирование

In [15]:
```
using Convex, SCS
```

In [16]:
```
# Задаем переменные для определения размерности матрицы A
m = 5
n = 3
# Задаем матрицу A и вектор b
A = randn(m, n)
b = randn(m, 1)
```

Out[16]:
```
5×1 Array{Float64,2}:
  0.37974429867172604
 -1.5525315174702539
  0.4883412588264075
 -0.4069536239842097
  0.3731970925049839
```

In [17]:
```
# Создаем вектор-столбец размера n x 1
x = Variable(n)
```

Out[17]:
```
Variable
size: (3, 1)
sign: real
vexity: affine
id: 318…347
```

In [18]:
```
# Объект модели:
task_3 = minimize(sumsquares(A * x - b), [x >= 0])
```

Out[18]:
```
minimize
└─ qol_elem (convex; positive)
   ├─ norm2 (convex; positive)
   │  └─ + (affine; real)
   │     ├─ …
   │     └─ …
   └─ [1.0]
subject to
└─ >= constraint (affine)
   ├─ 3-element real variable (id: 318…347)
   └─ 0

status: `solve!` not called yet
```

In [19]:
```
# Находим решение:
solve!(task_3, SCS.Optimizer)
```

```
------------------------------------------------------------------
        SCS v2.1.2 - Splitting Conic Solver
        (c) Brendan O'Donoghue, Stanford University, 2012
------------------------------------------------------------------
Lin-sys: sparse-indirect, nnz in A = 24, CG tol ~ 1/iter^(2.00)
eps = 1.00e-05, alpha = 1.50, max_iters = 5000, normalize = 1, scale = 1.00
acceleration_lookback = 10, rho_x = 1.00e-03
Variables n = 6, constraints m = 14
Cones:  primal zero / dual free vars: 1
        linear vars: 4
        soc vars: 9, soc blks: 2
Setup time: 1.08e-02s
------------------------------------------------------------------
 Iter | pri res | dua res | rel gap | pri obj | dua obj | kap/tau | time (s)
------------------------------------------------------------------
     0| 1.60e+19  1.34e+19  1.00e+00 -3.92e+19  2.50e+19  5.11e+19  2.10e-05
    37| 5.15e-11  1.37e-10  1.98e-11  2.41e+00  2.41e+00  1.11e-16  8.74e-03
------------------------------------------------------------------
Status: Solved
Timing: Solve time: 8.75e-03s
        Lin-sys: avg # CG iterations: 2.00, avg solve time: 5.74e-07s
        Cones: avg projection time: 9.62e-08s
        Acceleration: avg step time: 2.28e-04s
------------------------------------------------------------------
Error metrics:
dist(s, K) = 2.2204e-16, dist(y, K*) = 0.0000e+00, s'y/|s||y| = 0.0000e+00
primal res: |Ax + s - b|_2 / (1 + |b|_2) = 5.1514e-11
dual res:   |A'y + c|_2 / (1 + |c|_2) = 1.3742e-10
rel gap:    |c'x + b'y| / (1 + |c'x| + |b'y|) = 1.9793e-11
```

```
------------------------------------------------------------------------------
Error metrics:
dist(s, K) = 2.2204e-16, dist(y, K*) = 0.0000e+00, s'y/|s||y| = 0.0000e+00
primal res: |Ax + s - b|_2 / (1 + |b|_2) = 5.1514e-11
dual res:   |A'y + c|_2 / (1 + |c|_2) = 1.3742e-10
rel gap:    |c'x + b'y| / (1 + |c'x| + |b'y|) = 1.9793e-11
------------------------------------------------------------------------------
c'x = 2.4056, -b'y = 2.4056
==============================================================================
```

In [20]:
```
# Проверяем статус (найдено ли оптимальное решение)
task_3.status
```

Out[20]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [21]:
```
# Получаем оптимальное значение
task_3.optval
```

Out[21]:
```
2.405571018460159
```

8.4.4. Оптимальная рассадка по залам

8.4.5. План приготовления кофе

In [39]:
```
# Контейнер для хранения данных о запасах на складе:
coffee_data = JuMP.Containers.DenseAxisArray(
    [0 500;
    0 2000;
    0 40;],
    ["coffee bean", "milk", "vanilla sugar"],
    ["min", "max"])
```

Out[39]:
```
2-dimensional DenseAxisArray{Int64,2,...} with index sets:
    Dimension 1, ["coffee bean", "milk", "vanilla sugar"]
    Dimension 2, ["min", "max"]
And data, a 3×2 Array{Int64,2}:
 0   500
 0  2000
 0    40
```

In [41]:
```
# массив данных с наименованиями кофе:
coffee = ["raf coffee", "cappuccino"]
```

Out[41]:
```
2-element Array{String,1}:
 "raf coffee"
 "cappuccino"
```

In [42]:
```
# Массив стоимости кофе:
cost = JuMP.Containers.DenseAxisArray(
    [400, 300],
    coffee)
```

Out[42]:
```
1-dimensional DenseAxisArray{Int64,1,...} with index sets:
    Dimension 1, ["raf coffee", "cappuccino"]
And data, a 2-element Array{Int64,1}:
 400
 300
```

In [43]:
```
# Расход продуктов для приготовления кофе:
coffee_data = JuMP.Containers.DenseAxisArray(
    [40 140 5;
     30 120 0],
    coffee,
    ["coffee bean", "milk", "vanilla sugar"])
```

Out[43]:
```
2-dimensional DenseAxisArray{Int64,2,...} with index sets:
    Dimension 1, ["raf coffee", "cappuccino"]
    Dimension 2, ["coffee bean", "milk", "vanilla sugar"]
And data, a 2×3 Array{Int64,2}:
 40  140  5
 30  120  0
```

In [44]:
```
# Определение объекта модели с именем model:
model = Model(GLPK.Optimizer)
```

Out[44]:
feasibility
Subject to

In [45]:
```
# Определим массив:
products = ["coffee bean", "milk", "vanilla sugar"]
```

Out[45]:
```
3-element Array{String,1}:
 "coffee bean"
 "milk"
 "vanilla sugar"
```

In [46]:
```
# Определение переменных:
@variables(model, begin
coffee_data[c, "min"] <= nutrition[c = products] <= coffee_data[c, "max"]
# Сколько покупать продуктов:
buy[coffee] >= 0
end)
```

In [47]:
```
# Определение целевой функции:
@objective(model, Max, sum(cost[f] * buy[f] for f in coffee))
```

Out[47]: $400buy_{rafcoffee} + 300buy_{cappuccino}$

In [48]:
```
# Определение ограничений модели:
@constraint(model, [c in products],
sum(coffee_data[f, c] * buy[f] for f in coffee) == nutrition[c])
```

Out[48]:
```
1-dimensional DenseAxisArray{ConstraintRef{Model,MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float6
4},MathOptInterface.EqualTo{Float64}},ScalarShape},1,...} with index sets:
    Dimension 1, ["coffee bean", "milk", "vanilla sugar"]
And data, a 3-element Array{ConstraintRef{Model,MathOptInterface.ConstraintIndex{MathOptInterface.ScalarAffineFunction{Float6
4},MathOptInterface.EqualTo{Float64}},ScalarShape},1}:
 -nutrition[coffee bean] + 40 buy[raf coffee] + 30 buy[cappuccino] = 0.0
 -nutrition[milk] + 140 buy[raf coffee] + 120 buy[cappuccino] = 0.0
 -nutrition[vanilla sugar] + 5 buy[raf coffee] = 0.0
```

In [49]:
```
# Вызов функции оптимизации:
JuMP.optimize!(model)
term_status = JuMP.termination_status(model)
```

Out[49]:
```
OPTIMAL::TerminationStatusCode = 1
```

In [50]:
```
#Для просмотра результата решения модно вывести значение переменной buy:
hcat(buy.data,JuMP.value.(buy.data))
```

Out[50]:
```
2×2 Array{GenericAffExpr{Float64,VariableRef},2}:
 buy[raf coffee]  8
 buy[cappuccino]  6
```

Вывод:

Освоил пакеты Julia для решения задач оптимизации.