Лабораторная работа №8

Компьютерный практикум по статистическому анализу данных

Николаев Дмитрий Иванович

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

Под оптимизацией в математике и информатике понимается решение задачи нахождения экстремума (минимума или максимума) целевой функции в некоторой области конечномерного векторного пространства, ограниченной набором линейных и/или нелинейных равенств и/или неравенств.

Оптимизационной задачей называется задача определения наилучших с точки зрения структуры или значений параметров объектов.

## 2.1 Повторение примеров

Повторим примеры, представленные в лабораторной работе ([1]).

### 2.1.1 Линейное программирование

Линейное программирование рассматривает решения экстремальных задач на множествах –мерного векторного пространства, задаваемых системами линейных уравнений и неравенств.

Общей (стандартной) задачей линейного программирования называется задача нахождения минимума линейной целевой функции вида:

где — некоторые коэффициенты, .

Основной задачей линейного программирования называется задача, в которой есть ограничения в форме неравенств:

Задачи линейного программирования со смешанными ограничениями, такими как равенства и неравенства, с наличием переменных, свободных от ограничений, могут быть сведены к эквивалентным с тем же множеством решений путём замены переменных и замены равенств на пару неравенств.

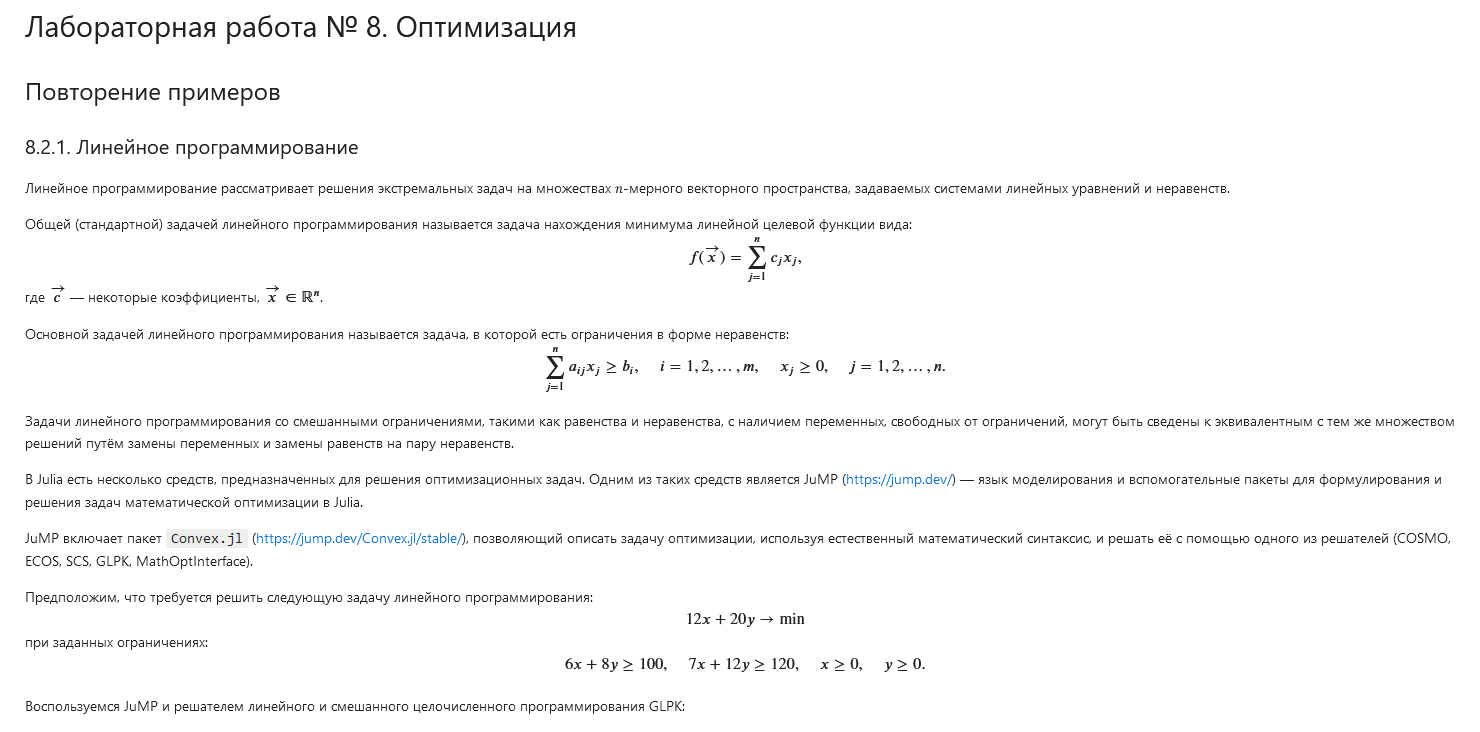
В Julia есть несколько средств, предназначенных для решения оптимизационных задач. Одним из таких средств является JuMP — язык моделирования и вспомогательные пакеты для формулирования и решения задач математической оптимизации в Julia.

JuMP включает пакет Convex.jl, позволяющий описать задачу оптимизации, используя естественный математический синтаксис, и решать её с помощью одного из решателей (COSMO, ECOS, SCS, GLPK, MathOptInterface).

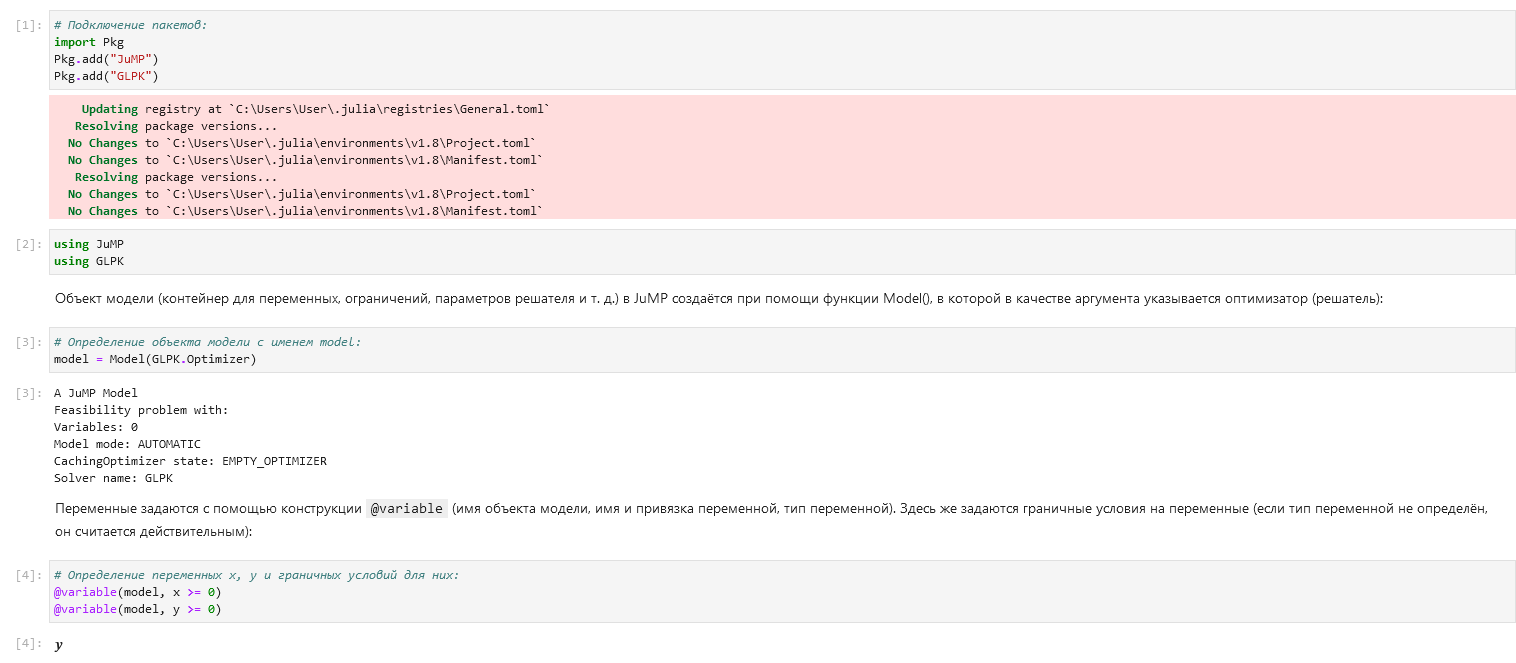
Предположим, что требуется решить следующую задачу линейного программирования:

при заданных ограничениях:

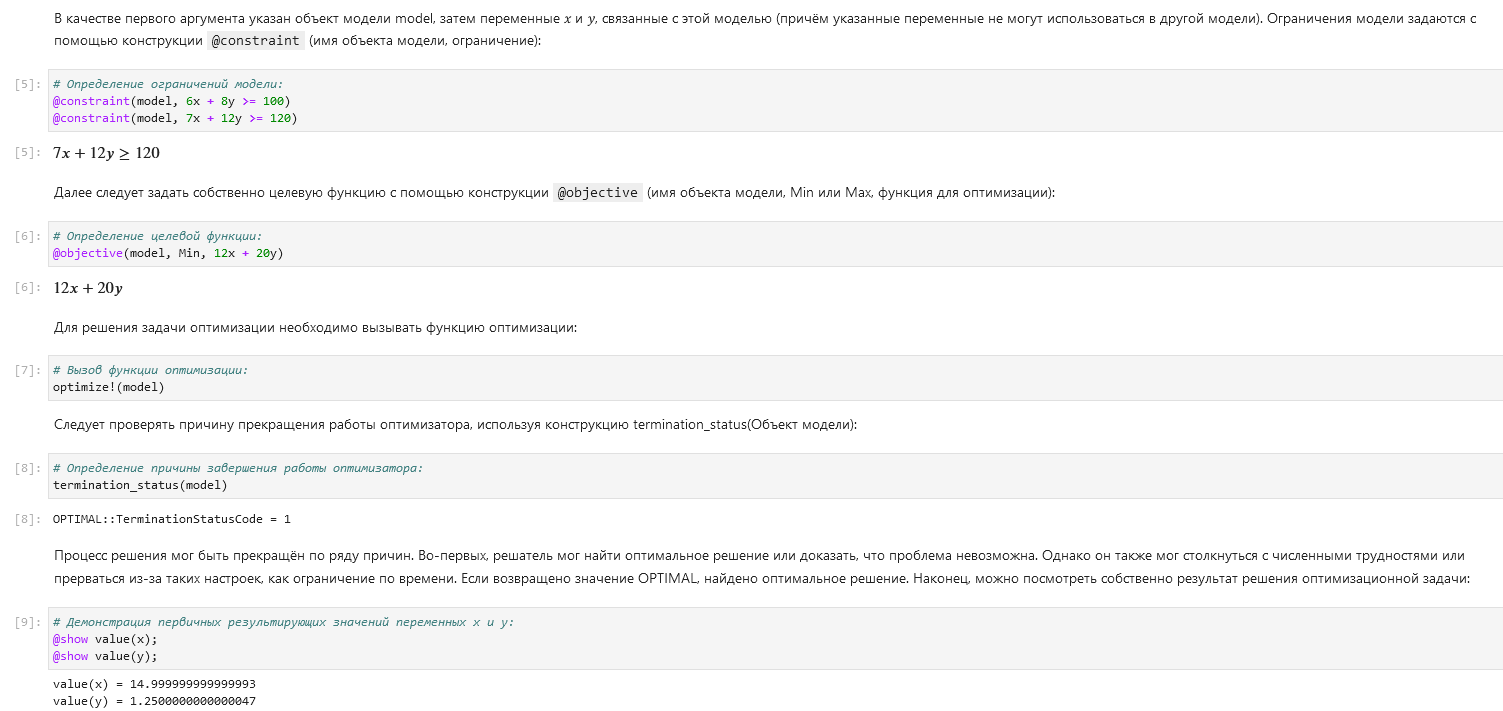
Решим задачу линейного программирования ([??-??]).



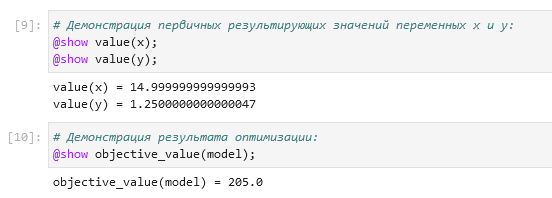
Линейное программирование (1)



Линейное программирование (2)



Линейное программирование (3)



Линейное программирование (4)

### 2.1.2 Векторизованные ограничения и целевая функция оптимизации

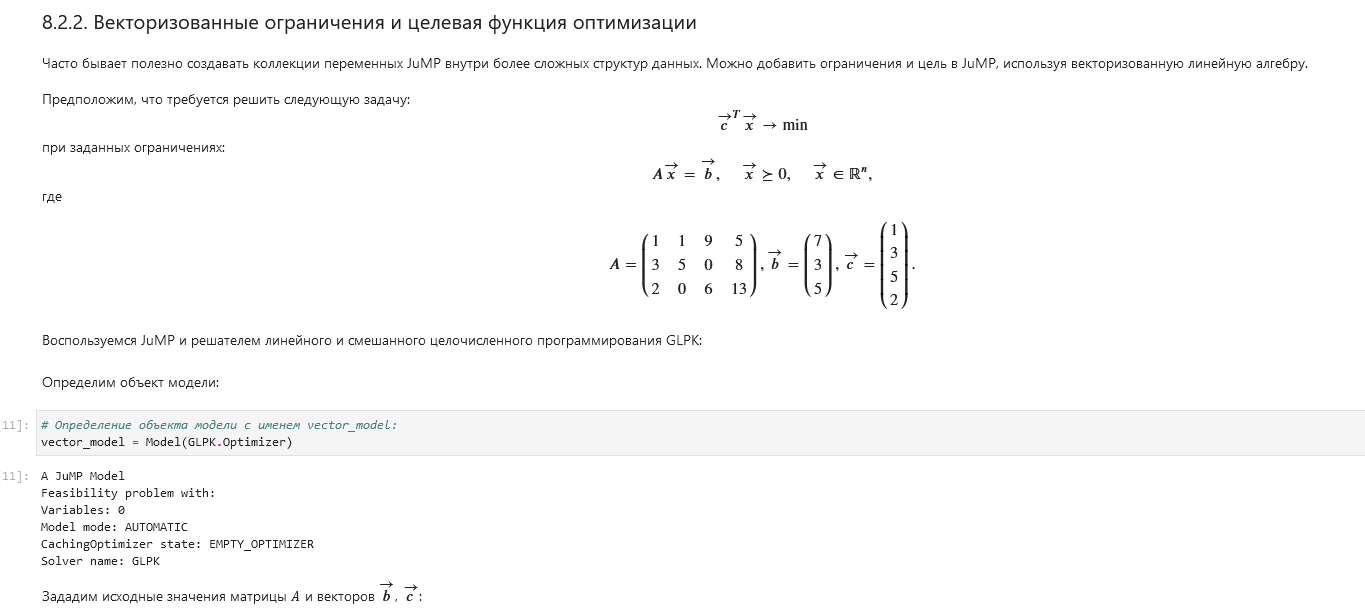
Часто бывает полезно создавать коллекции переменных JuMP внутри более сложных структур данных. Можно добавить ограничения и цель в JuMP, используя векторизованную линейную алгебру.

Предположим, что требуется решить следующую задачу:

при заданных ограничениях:

где

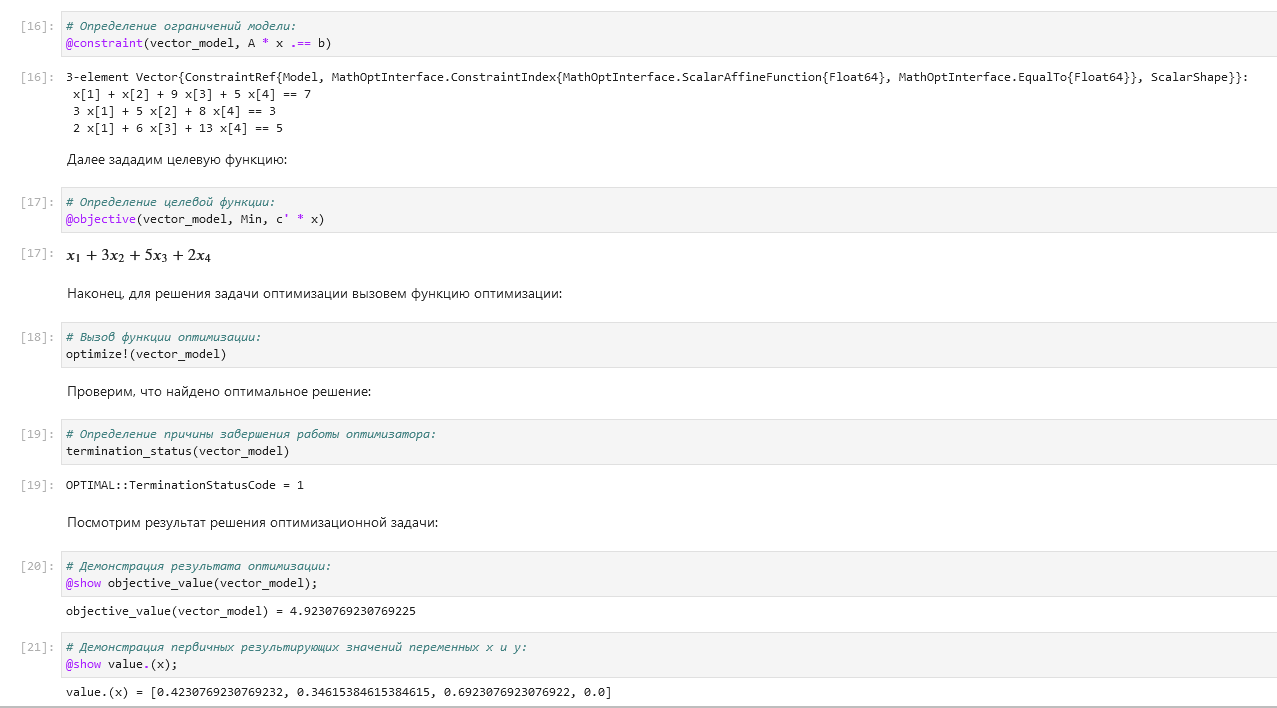
Решим задачу линейного программирования, используя векторизированную линейную алгебру ([??-??]).



Векторизованные ограничения и целевая функция оптимизации (1)



Векторизованные ограничения и целевая функция оптимизации (2)



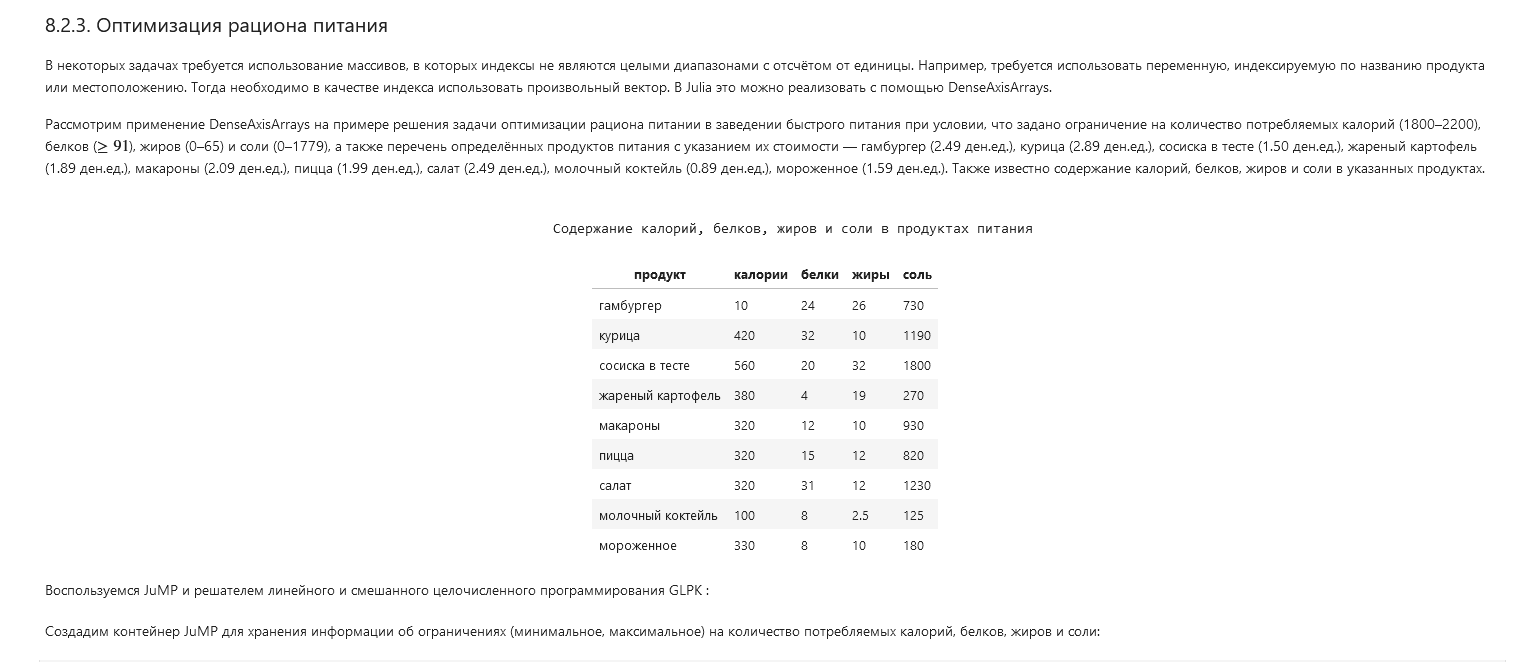
Векторизованные ограничения и целевая функция оптимизации (3)

### 2.1.3 Оптимизация рациона питания

В некоторых задачах требуется использование массивов, в которых индексы не являются целыми диапазонами с отсчётом от единицы. Например, требуется использовать переменную, индексируемую по названию продукта или местоположению. Тогда необходимо в качестве индекса использовать произвольный вектор. В Julia это можно реализовать с помощью DenseAxisArrays.

Рассмотрим применение DenseAxisArrays на примере решения задачи оптимизации рациона питании в заведении быстрого питания при условии, что задано ограничение на количество потребляемых калорий (1800–2200), белков (), жиров (0–65) и соли (0–1779), а также перечень определённых продуктов питания с указанием их стоимости — гамбургер (2.49 ден.ед.), курица (2.89 ден.ед.), сосиска в тесте (1.50 ден.ед.), жареный картофель (1.89 ден.ед.), макароны (2.09 ден.ед.), пицца (1.99 ден.ед.), салат (2.49 ден.ед.), молочный коктейль (0.89 ден.ед.), мороженное (1.59 ден.ед.). Также известно содержание калорий, белков, жиров и соли в указанных продуктах.

Решим задачу оптимизации описанного выше рациона питания ([??-??]).



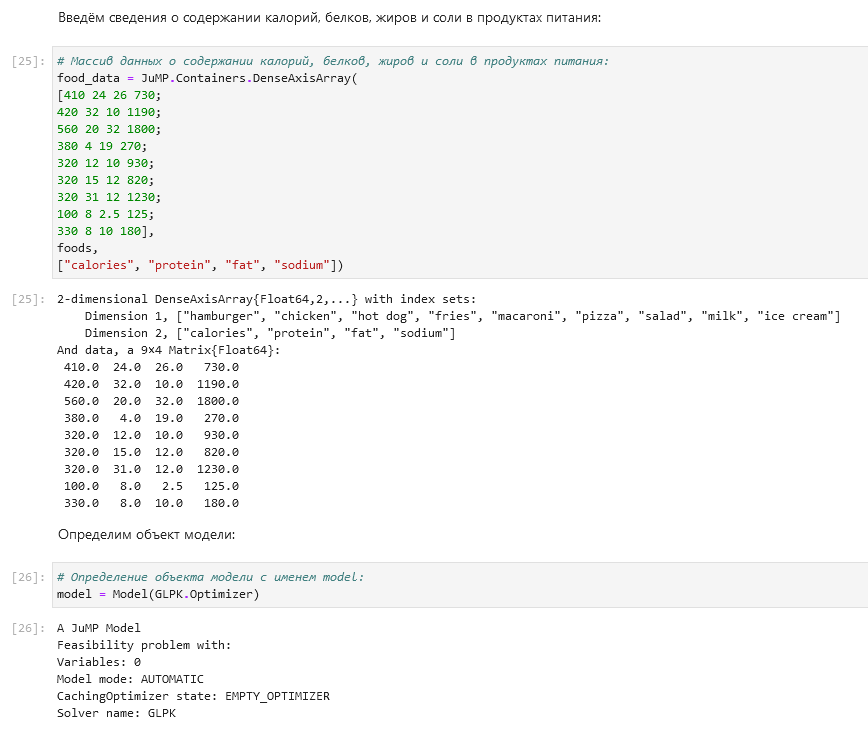
Оптимизация рациона питания (1)



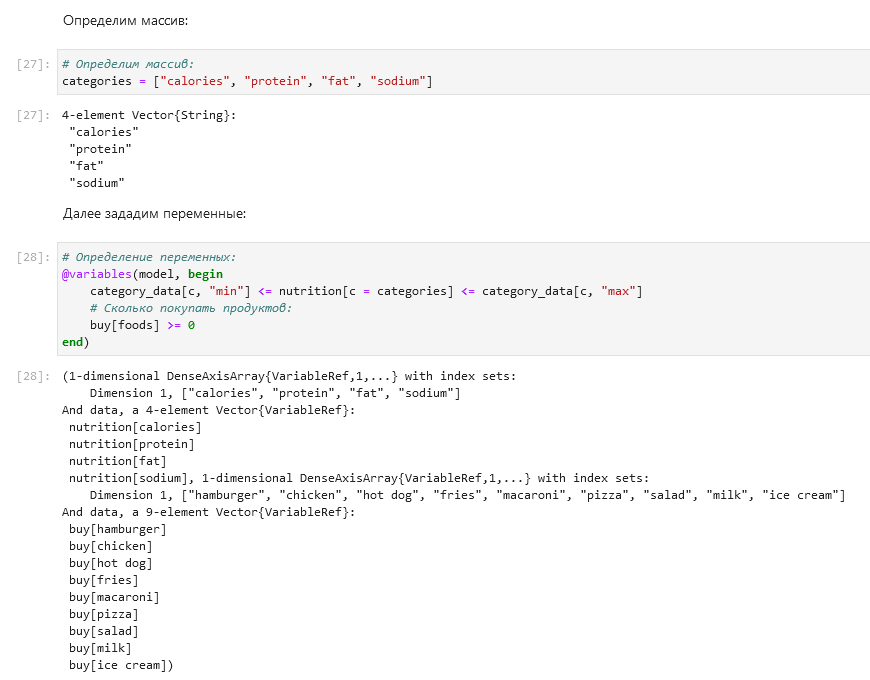
Оптимизация рациона питания (2)



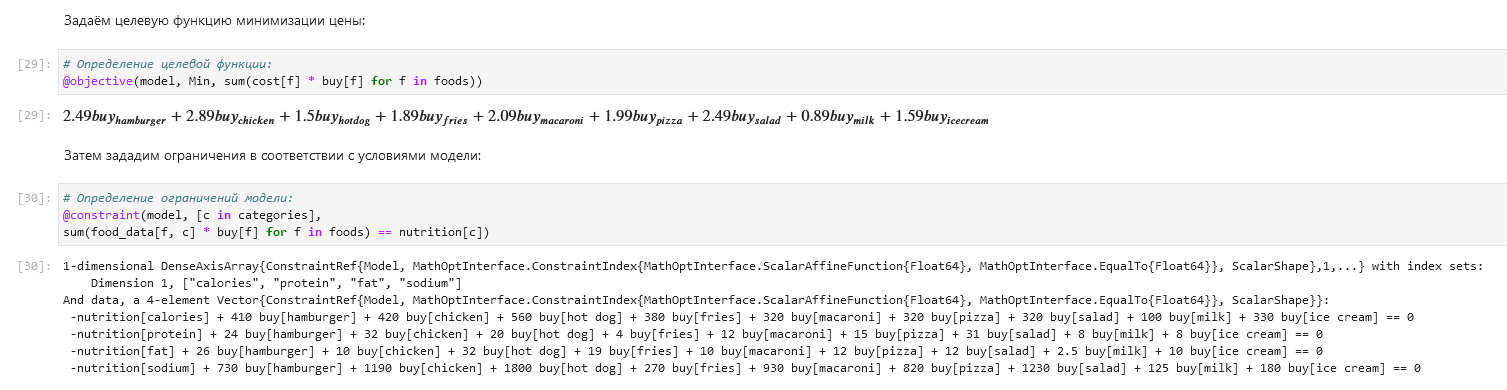
Оптимизация рациона питания (3)



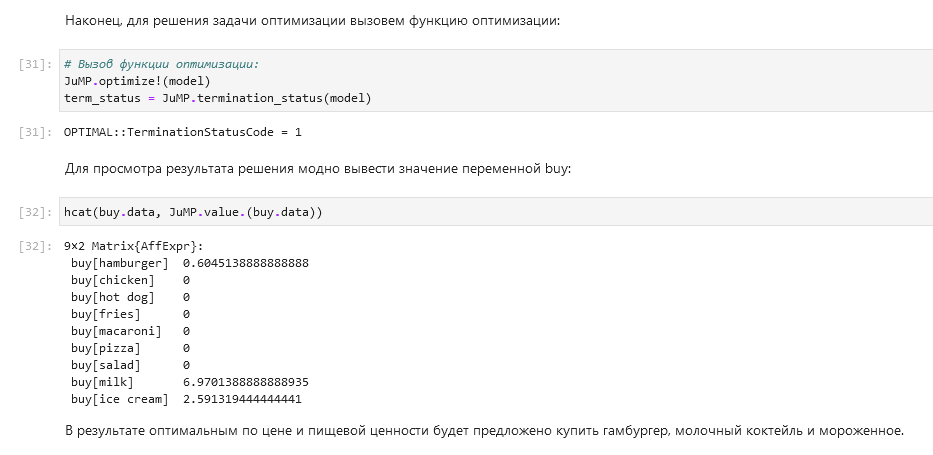
Оптимизация рациона питания (4)



Оптимизация рациона питания (5)



Оптимизация рациона питания (6)

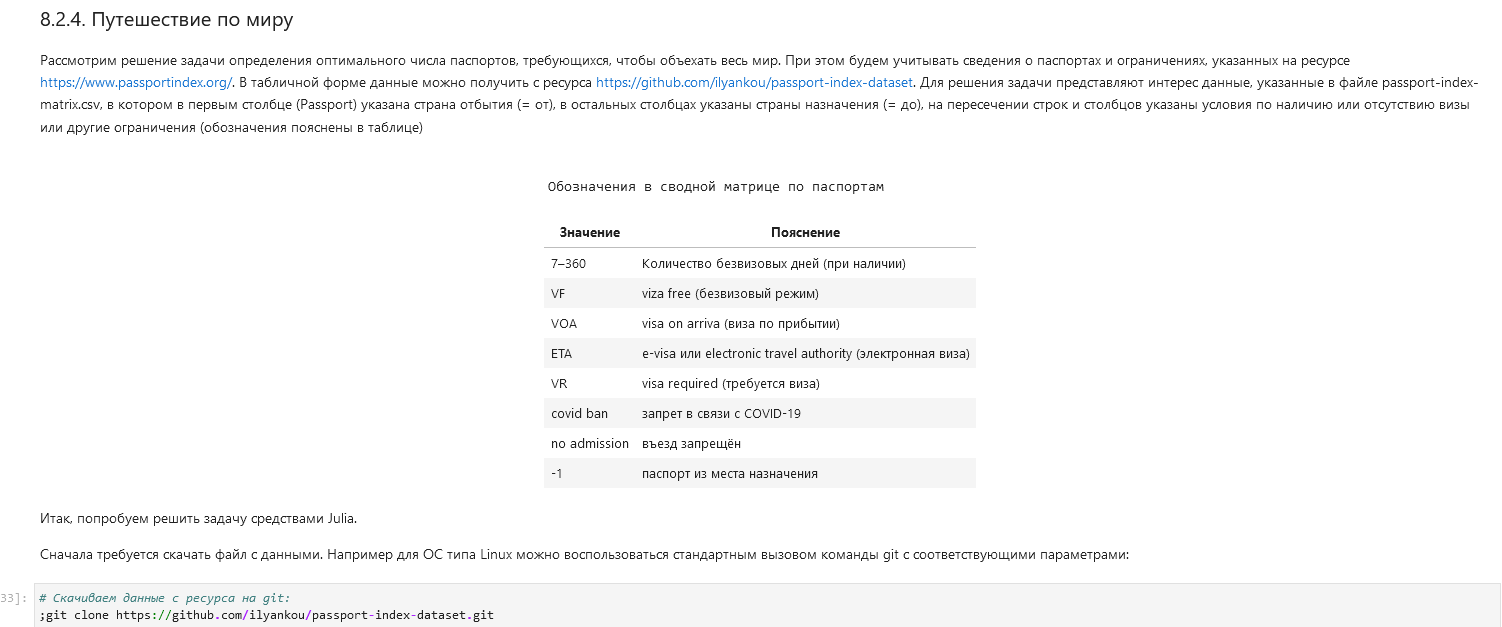


Оптимизация рациона питания (7)

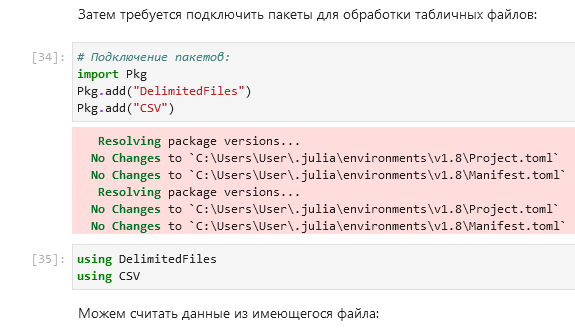
### 2.1.4 Путешествие по миру

Рассмотрим решение задачи определения оптимального числа паспортов, требующихся, чтобы объехать весь мир. При этом будем учитывать сведения о паспортах и ограничениях, указанных на ресурсе <https://www.passportindex.org/>. В табличной форме данные можно получить с ресурса <https://github.com/ilyankou/passport-index-dataset>. Для решения задачи представляют интерес данные, указанные в файле passport-index-matrix.csv, в котором в первым столбце (Passport) указана страна отбытия (= от), в остальных столбцах указаны страны назначения (= до), на пересечении строк и столбцов указаны условия по наличию или отсутствию визы или другие ограничения (обозначения пояснены в таблице)

Найдем наименьшее число паспортов, необходимых для путешествия по любой стране мира ([??-??]).



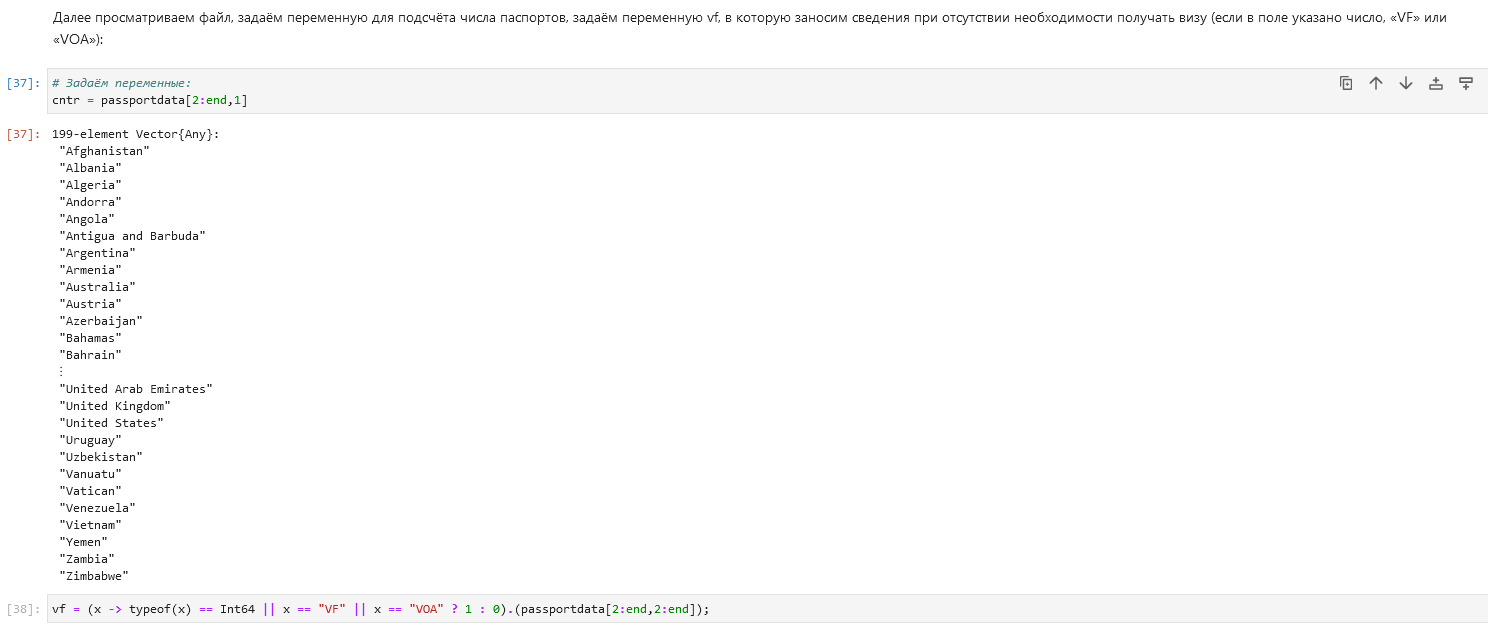
Путешествие по миру (1)



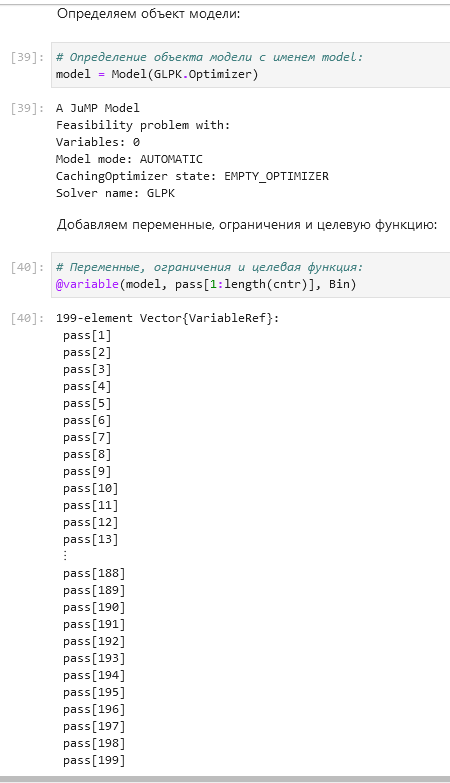
Путешествие по миру (2)



Путешествие по миру (3)



Путешествие по миру (4)



Путешествие по миру (5)



Путешествие по миру (6)



Путешествие по миру (7)

### 2.1.5 Портфельные инвестиции

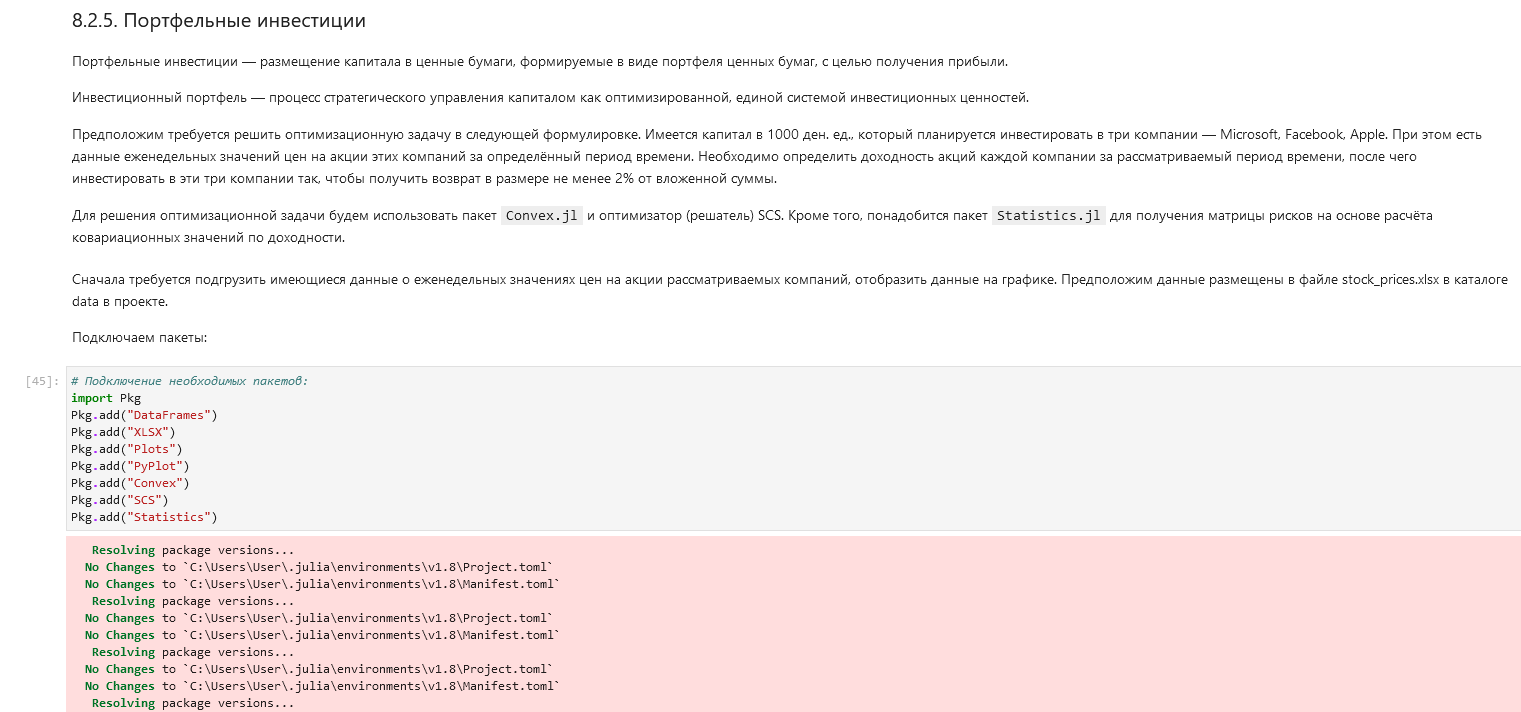
Портфельные инвестиции — размещение капитала в ценные бумаги, формируемые в виде портфеля ценных бумаг, с целью получения прибыли.

Инвестиционный портфель — процесс стратегического управления капиталом как оптимизированной, единой системой инвестиционных ценностей.

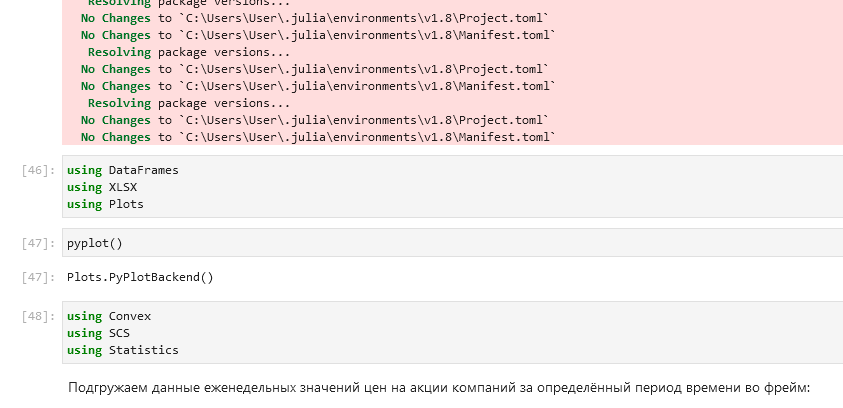
Предположим требуется решить оптимизационную задачу в следующей формулировке. Имеется капитал в 1000 ден. ед., который планируется инвестировать в три компании — Microsoft, Facebook, Apple. При этом есть данные еженедельных значений цен на акции этих компаний за определённый период времени. Необходимо определить доходность акций каждой компании за рассматриваемый период времени, после чего инвестировать в эти три компании так, чтобы получить возврат в размере не менее 2% от вложенной суммы.

Для решения оптимизационной задачи будем использовать пакет Convex.jl и оптимизатор (решатель) SCS. Кроме того, понадобится пакет Statistics.jl для получения матрицы рисков на основе расчёта ковариационных значений по доходности.

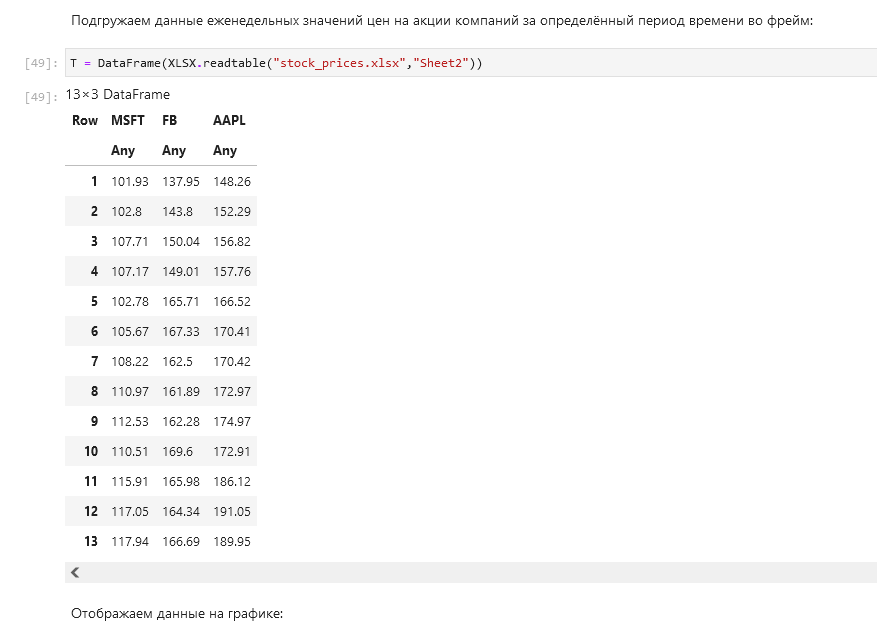
Найдем оптимальное число инвестиций в ценные бумаги, при получении хотя бы 2% возврата ([??-??]).



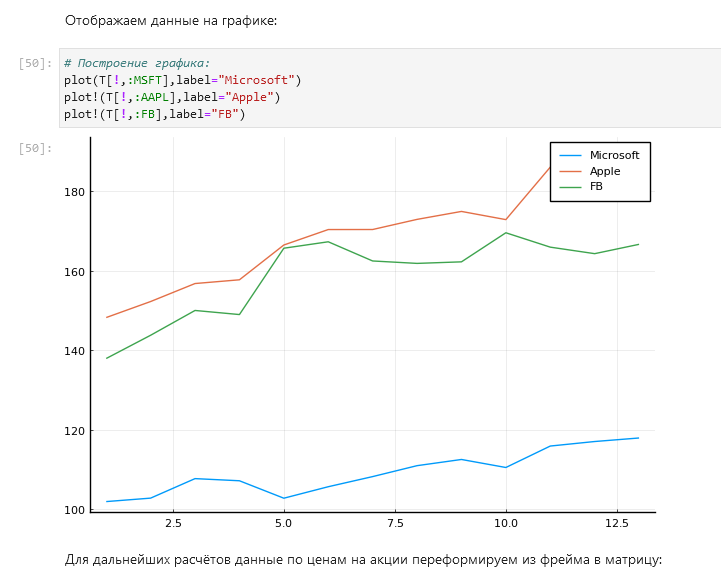
Портфельные инвестиции (1)



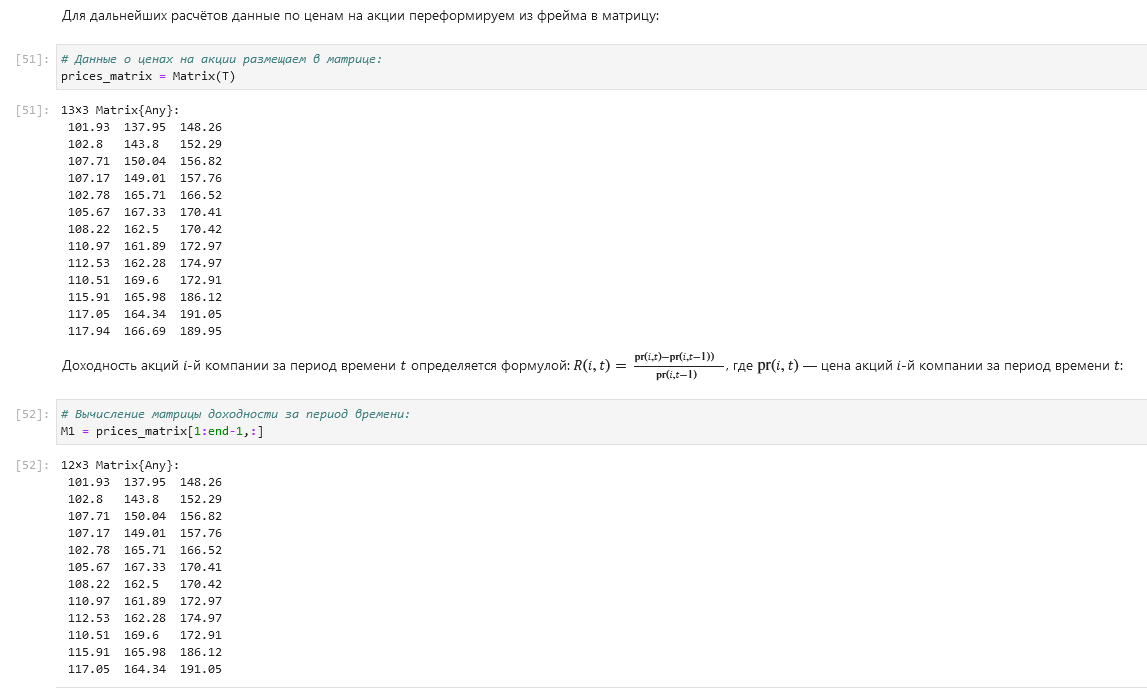
Портфельные инвестиции (2)



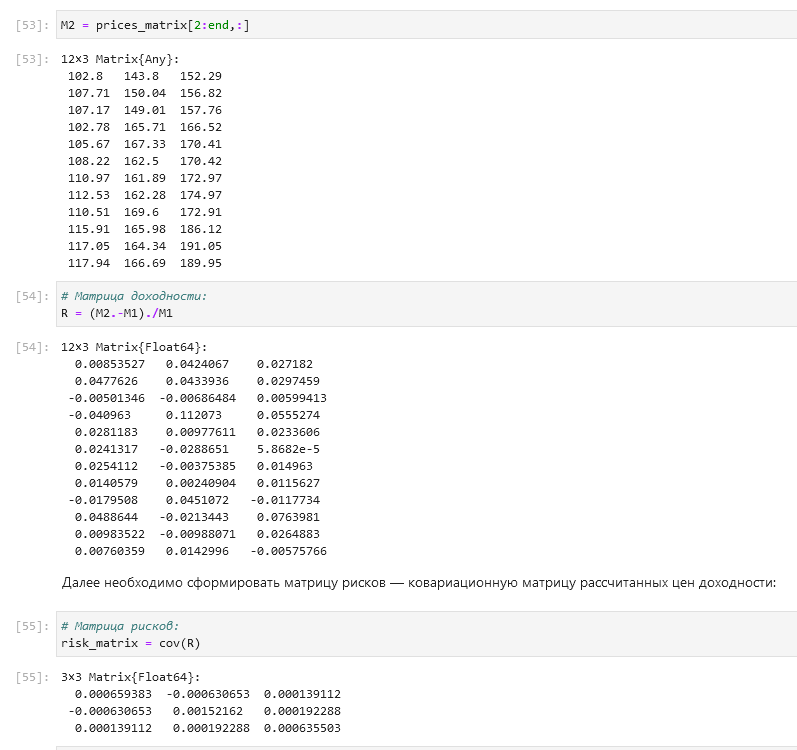
Портфельные инвестиции (3)



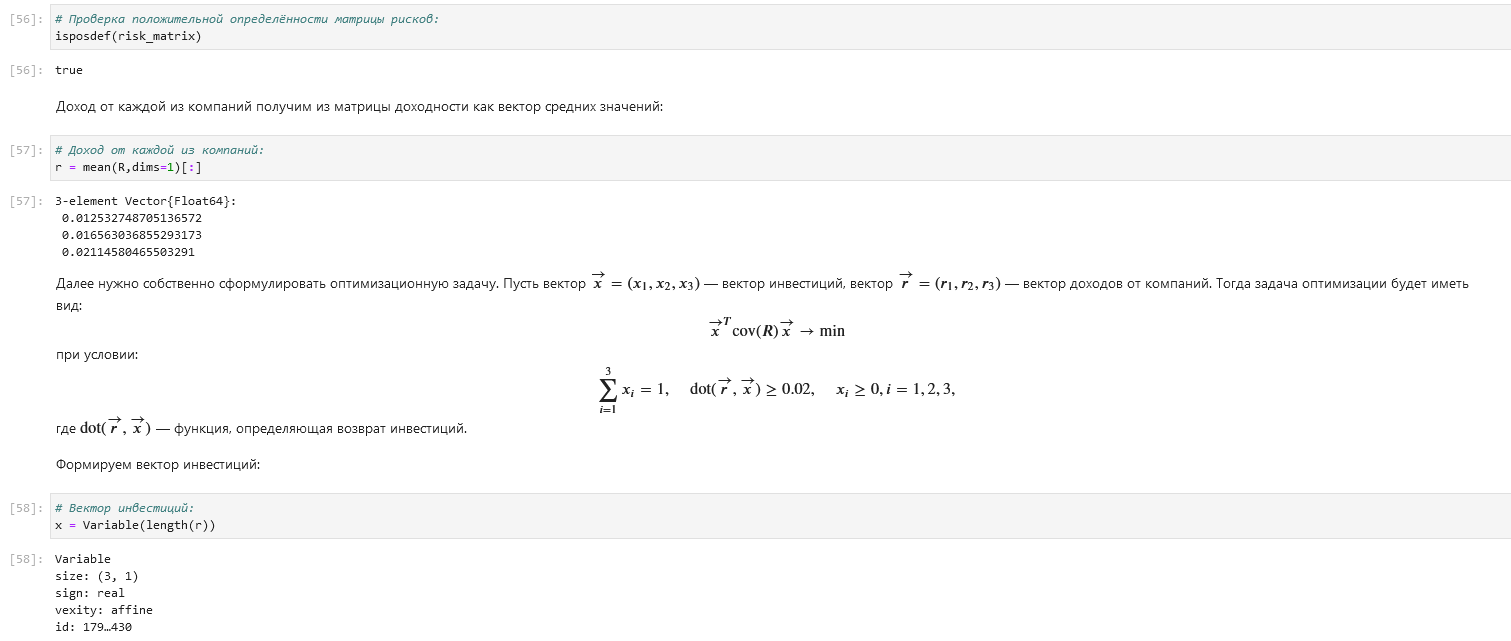
Портфельные инвестиции (4)



Портфельные инвестиции (5)



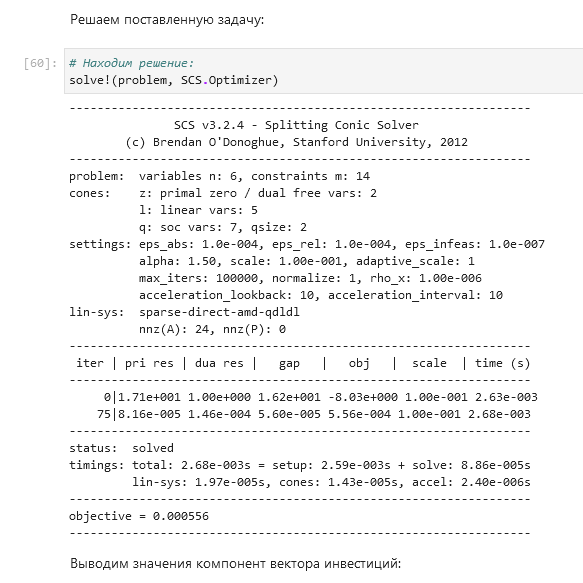
Портфельные инвестиции (6)



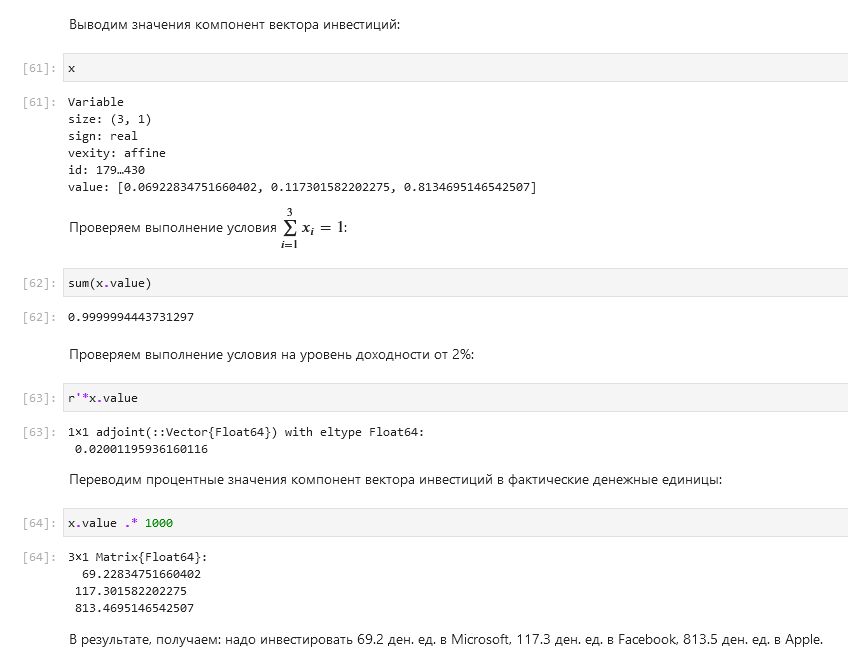
Портфельные инвестиции (7)



Портфельные инвестиции (8)



Портфельные инвестиции (9)

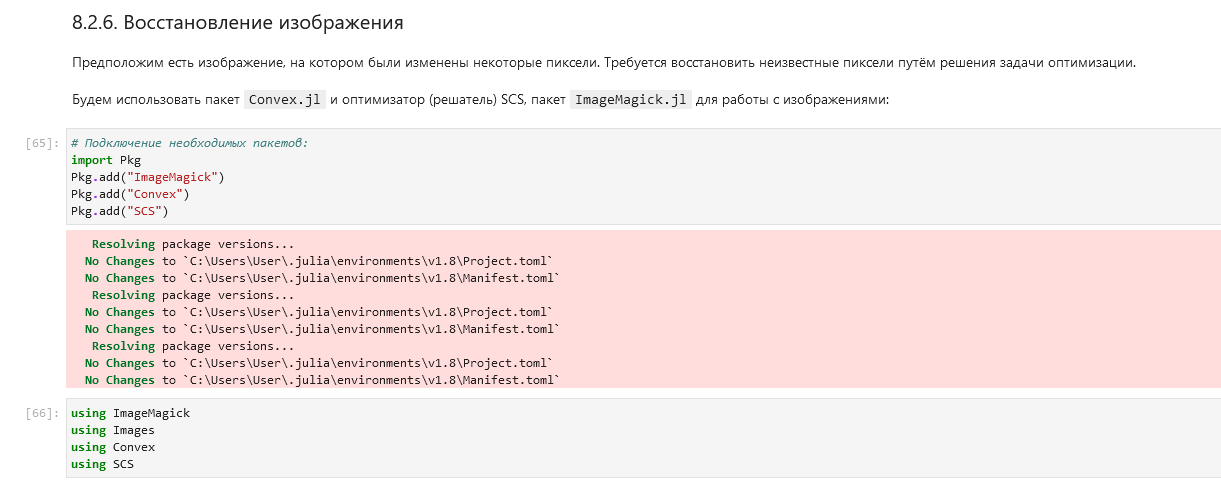


Портфельные инвестиции (10)

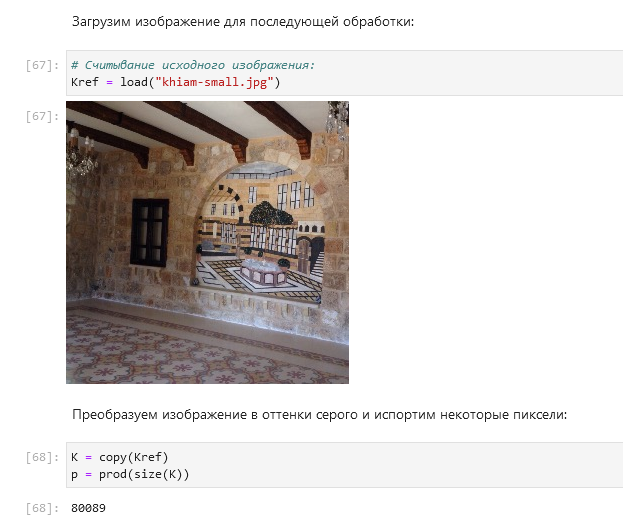
### 2.1.6 Восстановление изображения

Предположим есть изображение, на котором были изменены некоторые пиксели. Требуется восстановить неизвестные пиксели путём решения задачи оптимизации.

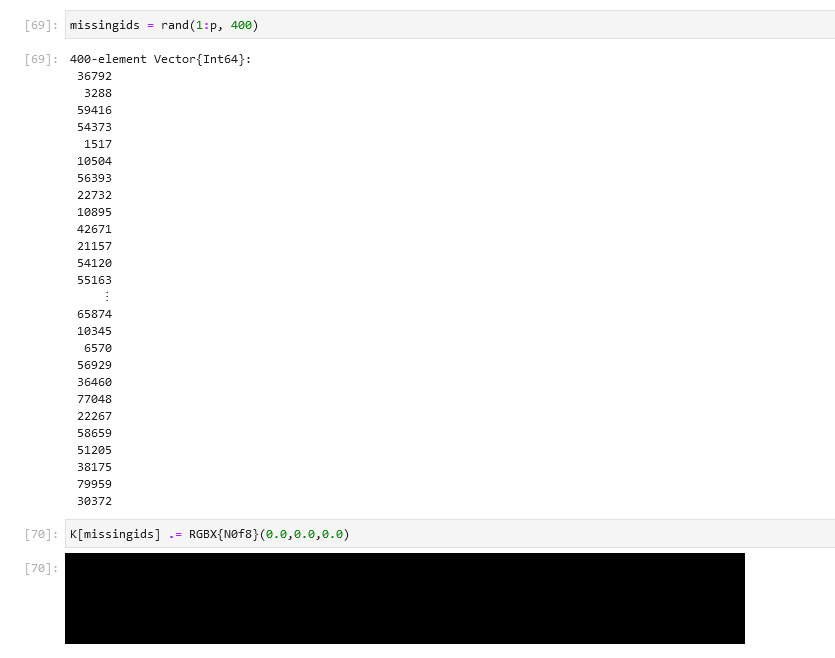
Восстановим изображения, решая задачу оптимизации ([??-??]).



Восстановление изображения (1)



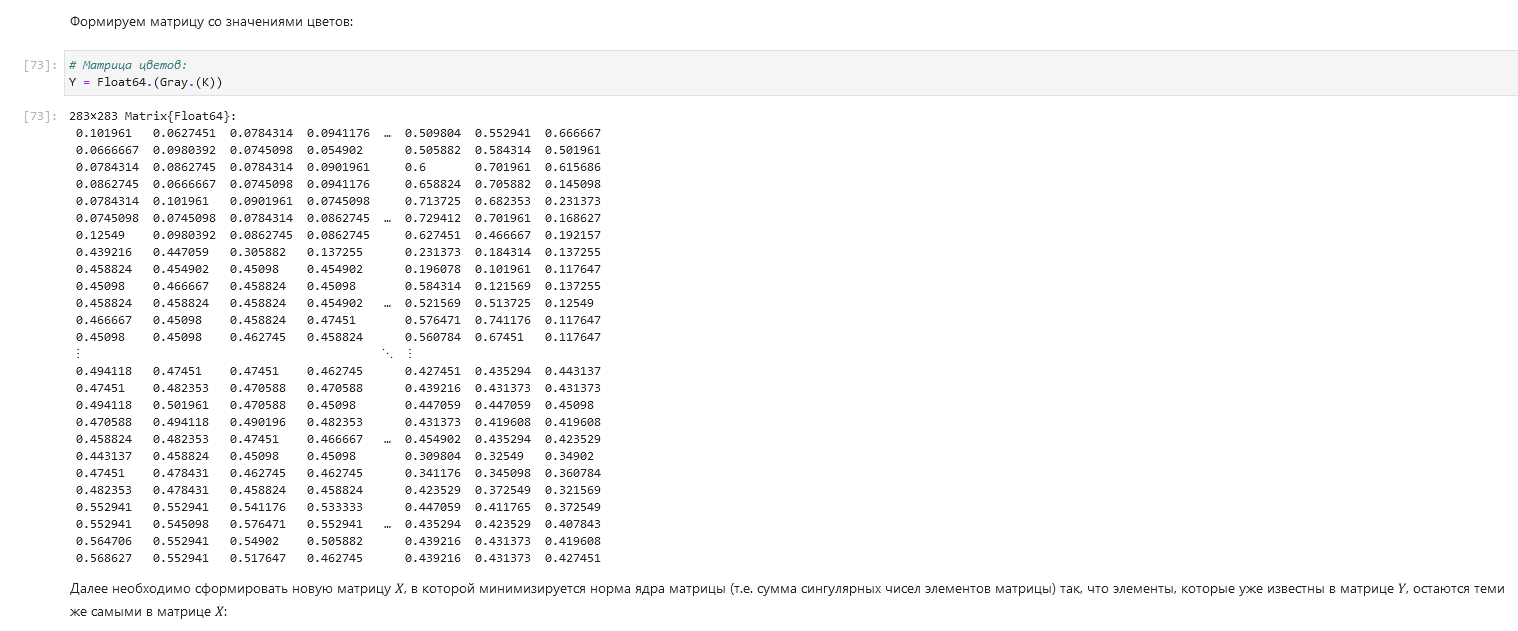
Восстановление изображения (2)



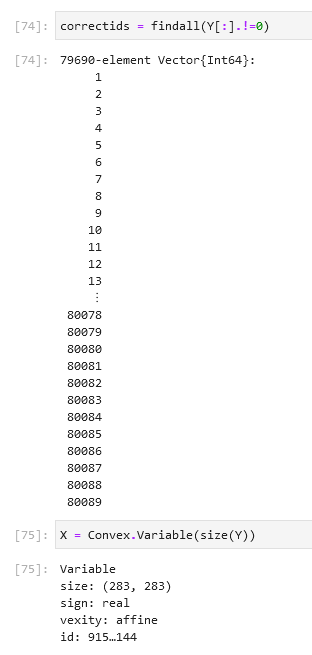
Восстановление изображения (3)



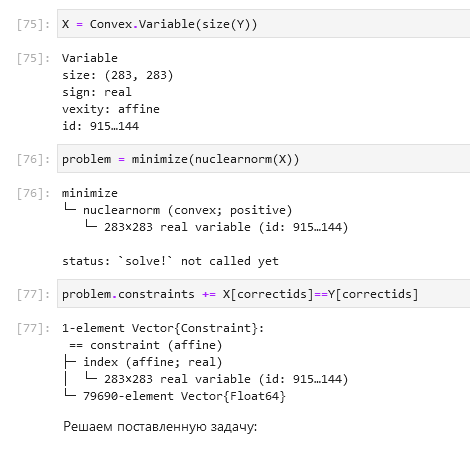
Восстановление изображения (4)



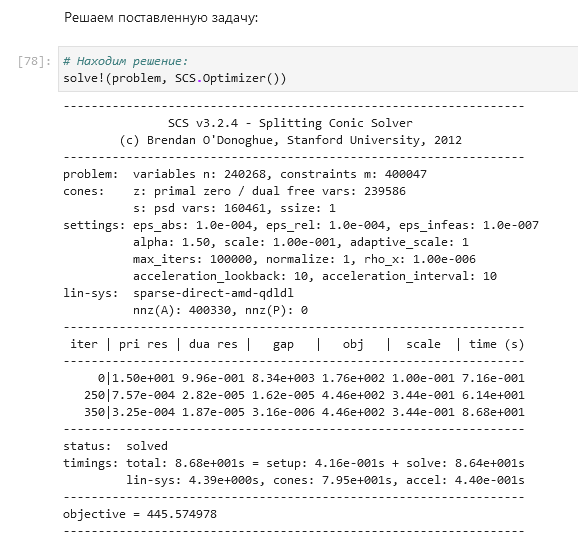
Восстановление изображения (5)



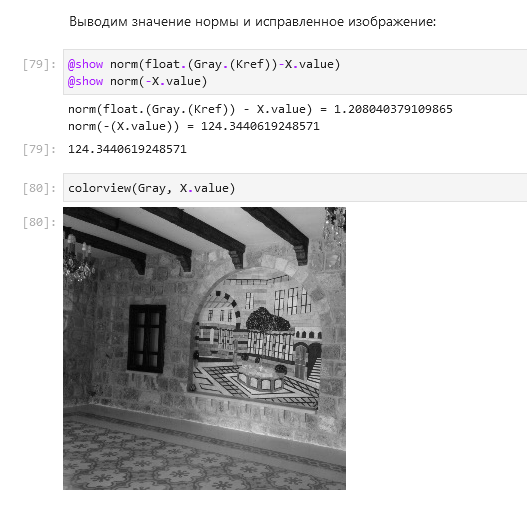
Восстановление изображения (6)



Восстановление изображения (7)



Восстановление изображения (8)



Восстановление изображения (9)

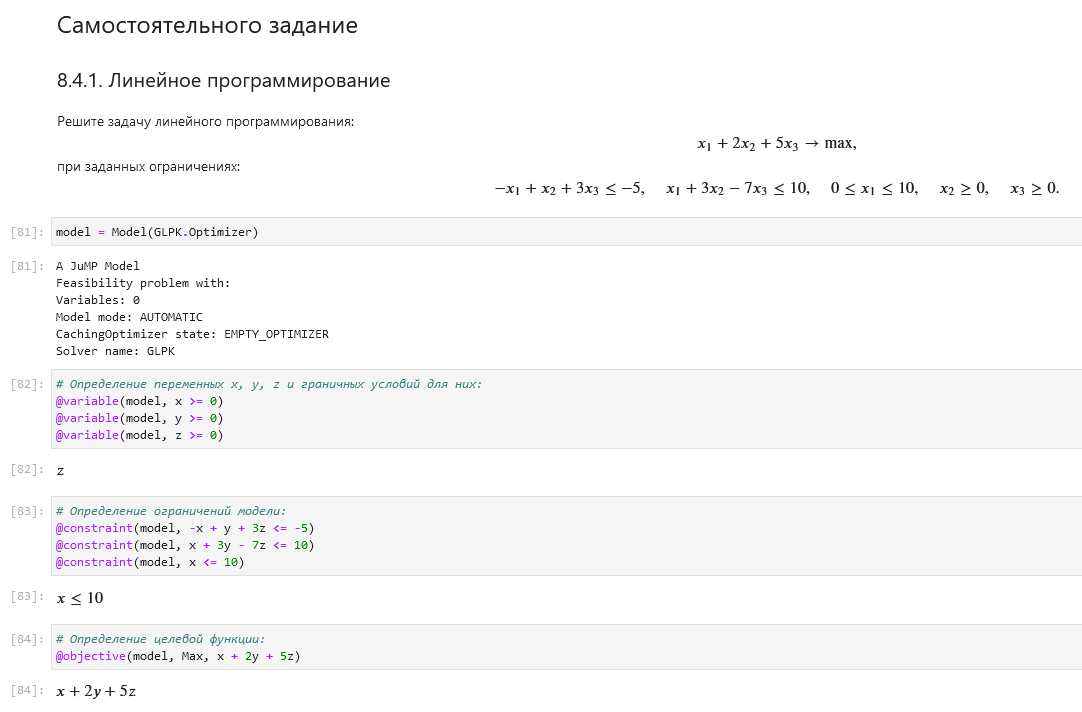
## 2.2 Самостоятельная работа

### 2.2.1 Линейное программирование

Решим задачу линейного программирования:

при заданных ограничениях:

Найдем значения переменных в данной оптимизационной задаче ([??,??])



Задание 8.4.1. Линейное программирование (1)



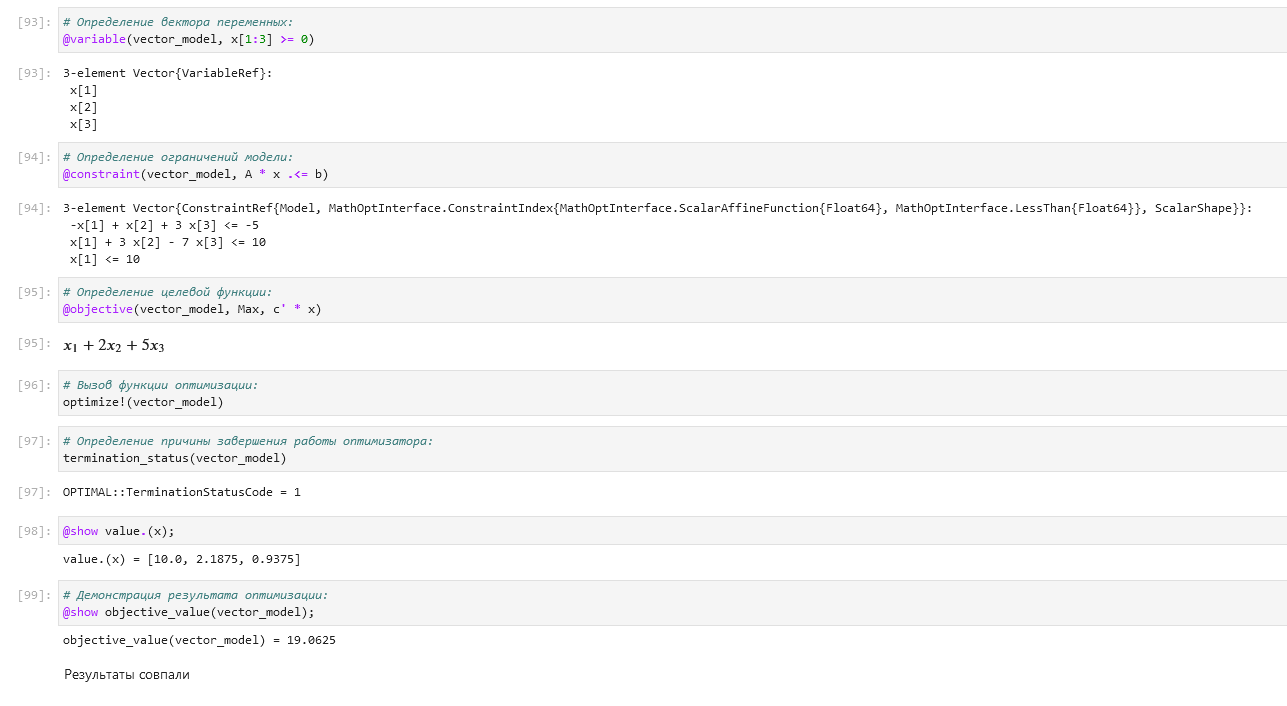
Задание 8.4.1. Линейное программирование (2)

### 2.2.2 Линейное программирование. Использование массивов

Решим предыдущее задание, используя массивы вместо скалярных переменных ([??,??]).



Задание 8.4.2. Линейное программирование. Использование массивов (1)



Задание 8.4.2. Линейное программирование. Использование массивов (2)

### 2.2.3 Выпуклое программирование

Решим задачу оптимизации:

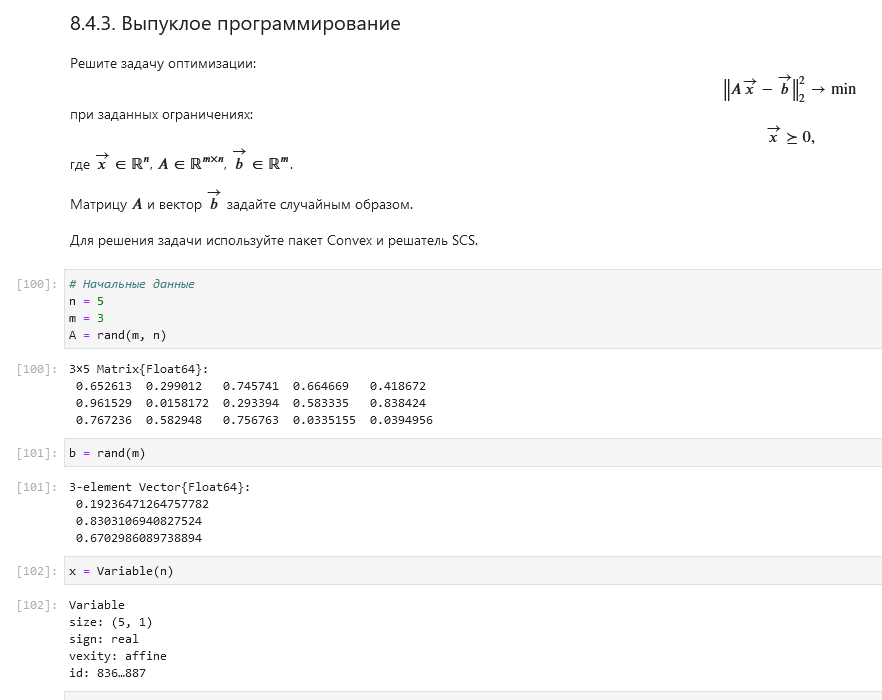
при заданных ограничениях:

где .

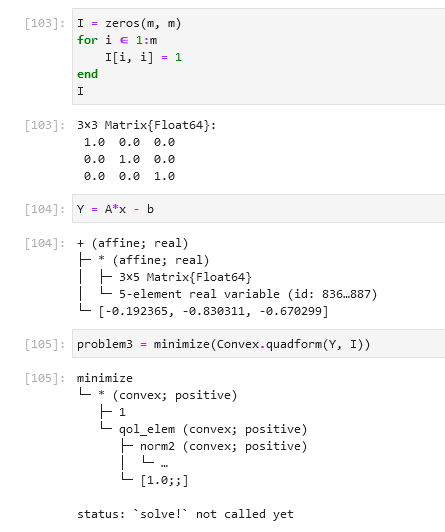
Матрицу и вектор зададим случайным образом.

Для решения задачи используем пакет Convex и решатель SCS.

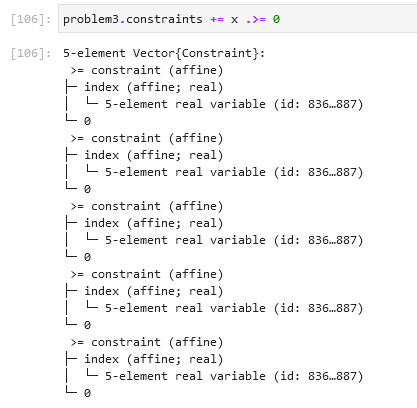
Найдем значения вектора в данной оптимизационной задаче ([??-??])



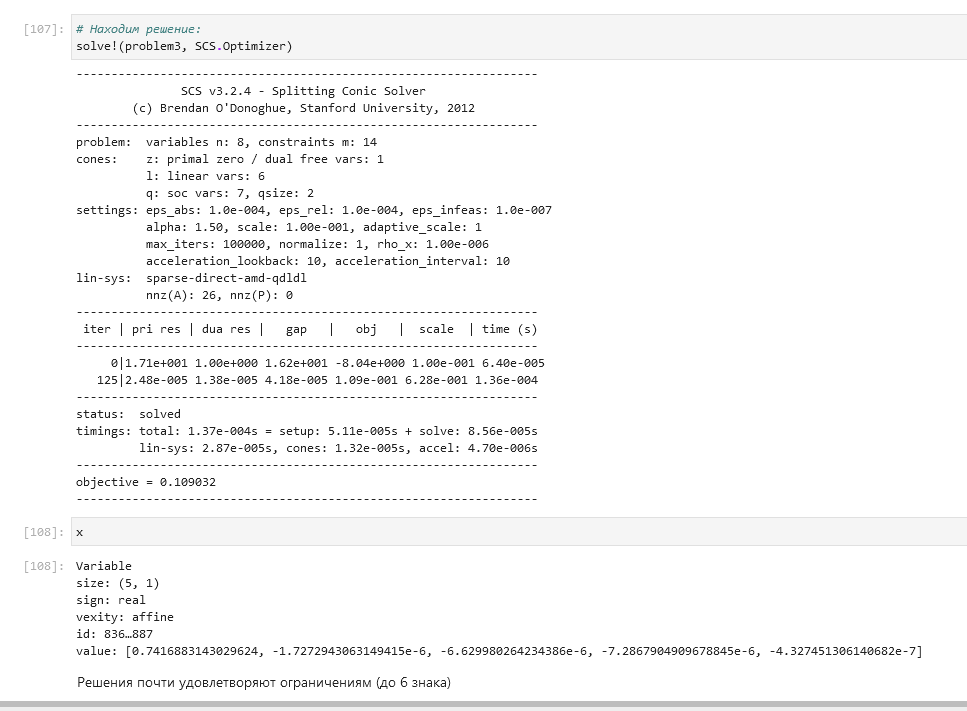
Задание 8.4.3. Выпуклое программирование (1)



Задание 8.4.3. Выпуклое программирование (2)



Задание 8.4.3. Выпуклое программирование (3)



Задание 8.4.3. Выпуклое программирование (4)

### 2.2.4 Оптимальная рассадка по залам

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.

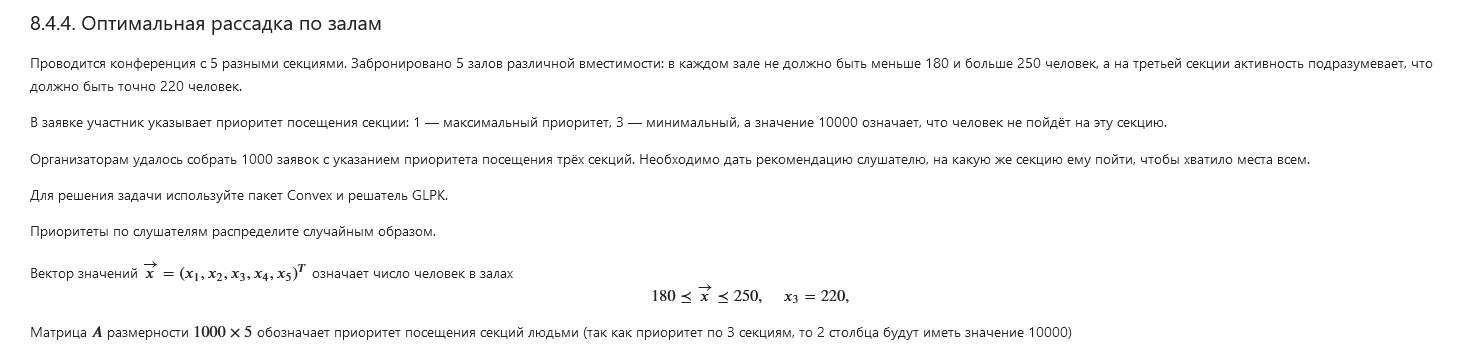
В заявке участник указывает приоритет посещения секции: 1 — максимальный приоритет, 3 — минимальный, а значение 10000 означает, что человек не пойдёт на эту секцию.

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.

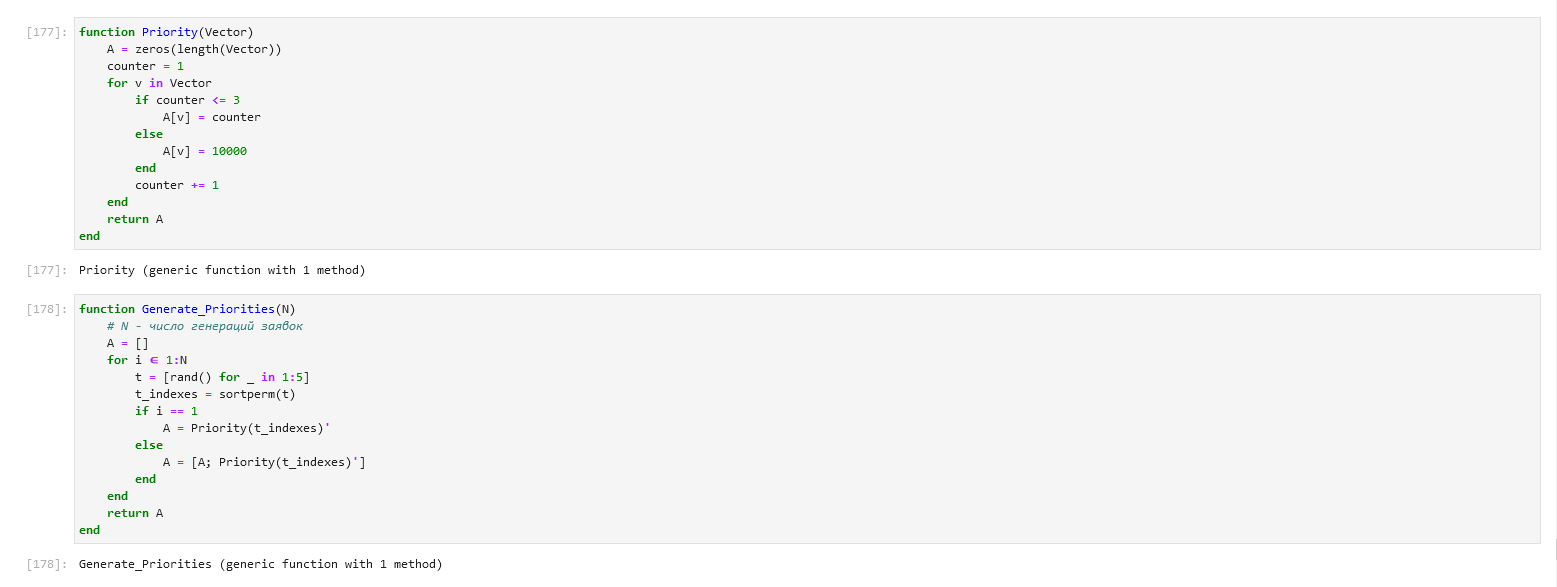
Для решения задачи используем пакет Convex и решатель GLPK.

Приоритеты по слушателям распределим случайным образом.

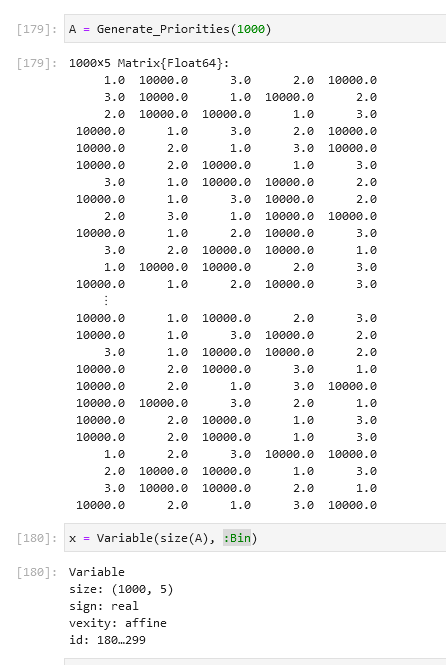
Согласно заданию ([??]), будем считать, что приоритеты по секциям у участников, кроме отказа от секции, не могут повторяться ([??,??]). После чего зададим и решим оптимизационную задачу ([??-??]), найдя оптимальную рассадку по секциям ([??]).



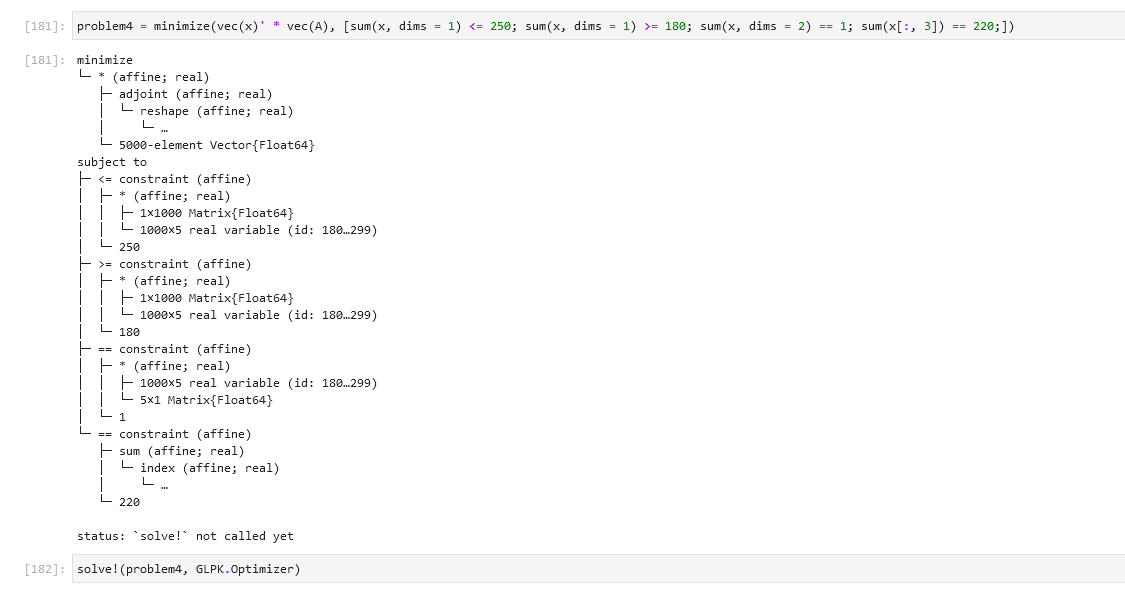
Задание 8.4.4. Оптимальная рассадка по залам (1)



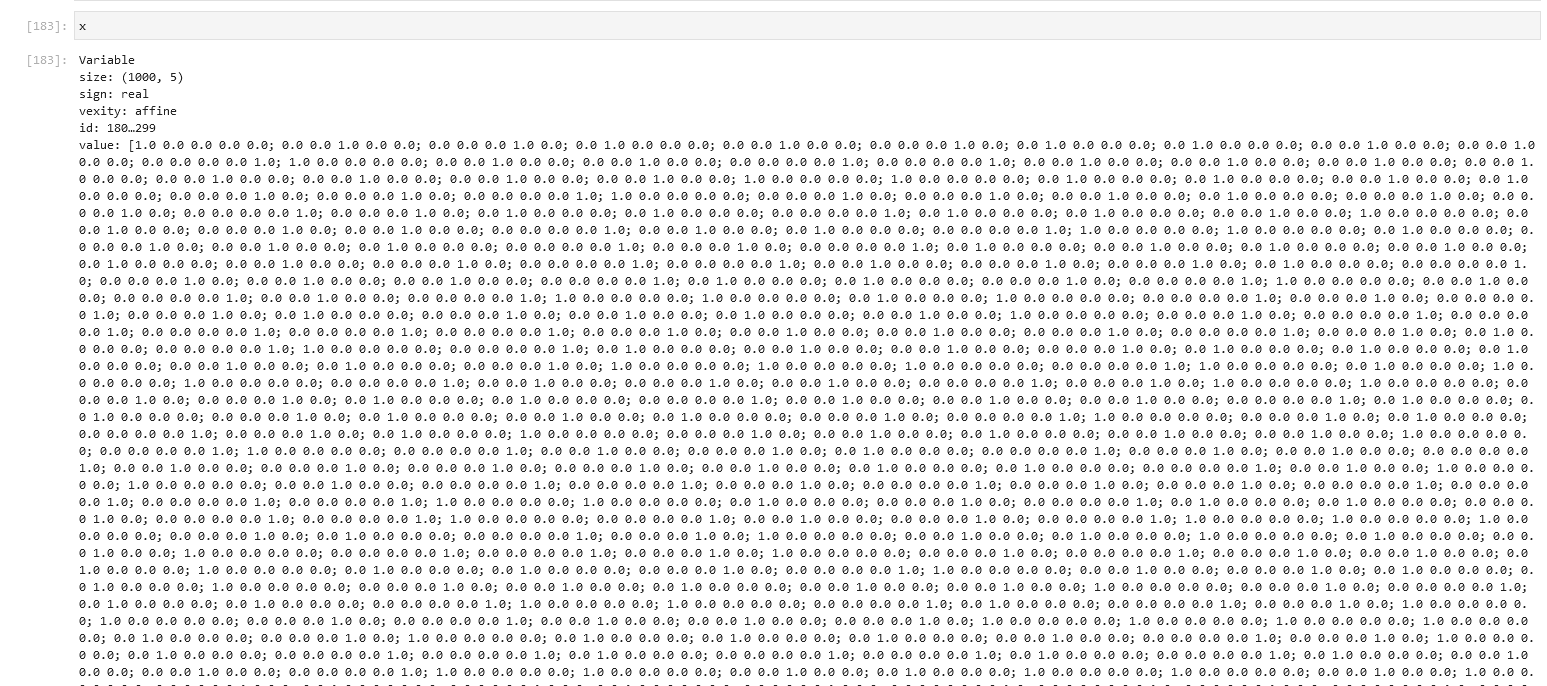
Задание 8.4.4. Оптимальная рассадка по залам (2)



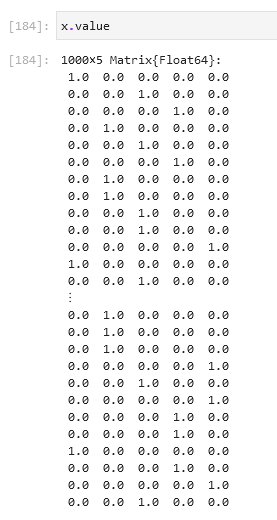
Задание 8.4.4. Оптимальная рассадка по залам (3)



Задание 8.4.4. Оптимальная рассадка по залам (4)



Задание 8.4.4. Оптимальная рассадка по залам (5)



Задание 8.4.4. Оптимальная рассадка по залам (6)



Задание 8.4.4. Оптимальная рассадка по залам (7)

### 2.2.5 План приготовления кофе

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.

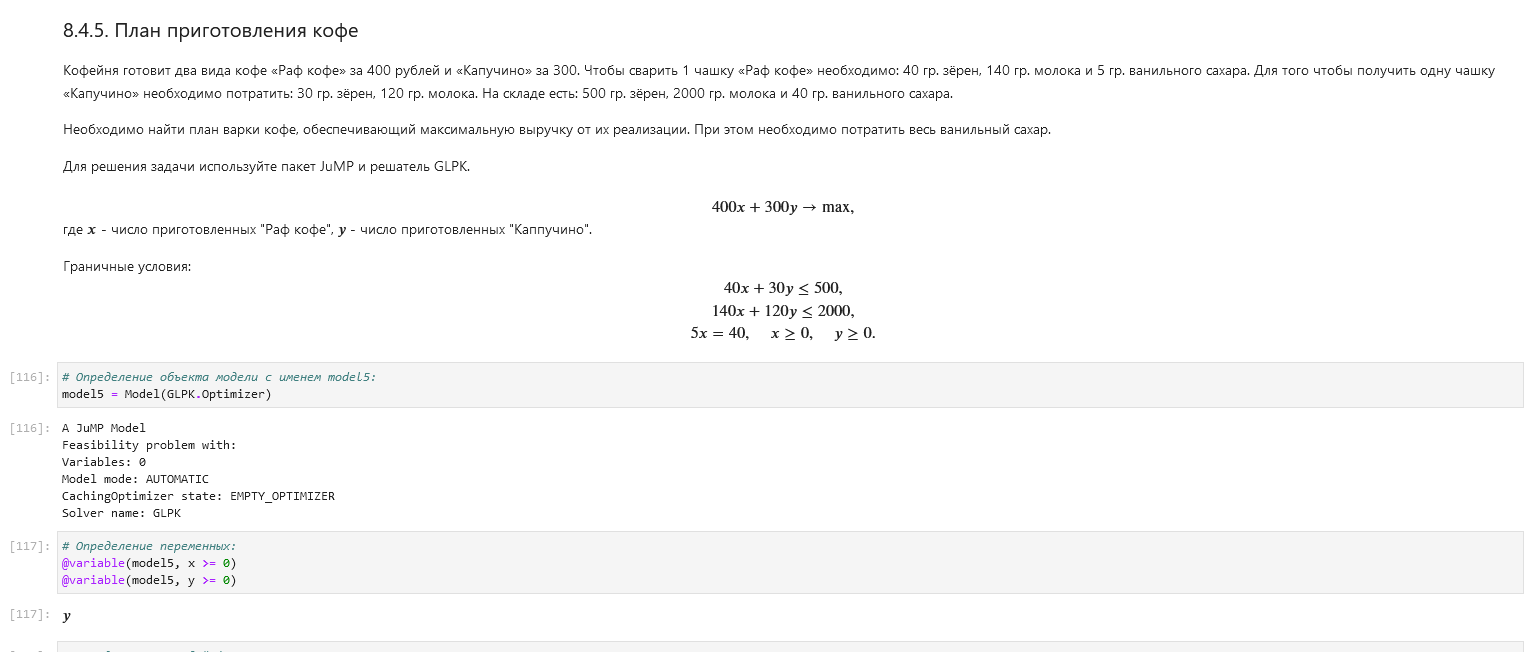
Для решения задачи используем пакет JuMP и решатель GLPK.

Переформулируем задачу. Введем обозначения: — число приготовленных “Раф кофе”, — число приготовленных “Каппучино”. Тогда необходимо решить следующую оптимизационную задачу:

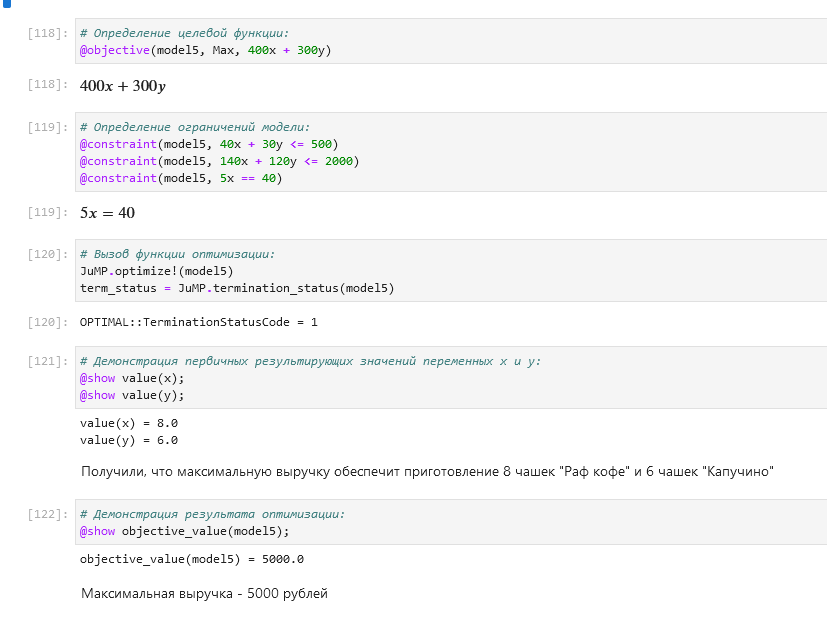
при заданных ограничениях

где первое ограничение обозначает массу затраченных зёрен, второе — массу затраченного молока, третье — ванильного сахара.

Теперь решим полученную задачу, найдя и ([??,??]).



Задание 8.4.5. План приготовления кофе (1)



Задание 8.4.5. План приготовления кофе (2)

# 3 Выводы

В ходе выполнения лабораторной работы я освоил пакеты Julia для решения задач оптимизации.

# Список литературы

1. Королькова А. В., Кулябов Д. С. Лабораторная работа № 8. Оптимизация [Электронный ресурс]. RUDN, 2023. URL: <https://esystem.rudn.ru/pluginfile.php/2231367/mod_resource/content/2/008-lab_optimization.pdf>.