УДК 332.01 JEL R10, R19, Y10 DOI 10.26425/1816-4277-2024-7-44-53

Обработка больших массивов данных в Python для целей прикладных социально-экономических исследований: преимущества и актуальные вопросы

Конищев Евгений Сергеевич

Аспирант, мл. науч. сотр.
ORCID: 0000-0001-9733-4821, e-mail: eskonishchev@fa.ru

Финансовый университет при Правительстве Российской Федерации, г. Москва, Россия

Аннотация

Статья посвящена исследованию особенностей процессов обработки больших массивов данных с помощью языка программирования Python. В отличие от табличных процессоров или готовых программных продуктов, языки программирования предлагают пользователю гибкий инструментарий для реализации поставленных задач. Вместе с тем это создает определенные риски, связанные с эффективностью использования соответствующих инструментов и оптимизации работы программы. Целью статьи является изучение особенностей обработки больших массивов данных в Python на примерах непосредственных исследовательских задач. Актуальность темы и цели статьи обусловлена существующим научным пробелом, связанным с комплексным рассмотрением технических аспектов использования языков программирования и сопряженного инструментария для социально-экономических исследований. Так, многие авторы, которые применяют языки программирования в своих работах, крайне редко предоставляют информацию, касающуюся преимуществ определенных алгоритмов или подходов. В рамках статьи автор на примере конкретных исследовательских задач рассматривает процессы и алгоритм обработки большого массива данных. В заключении сделаны выводы об особенностях и преимуществах Python при работе с большими массивами данных, а также о перспективности развития соответствующей научной тематики.

Ключевые слова

Язык программирования, Python, анализ данных, обработка данных, большие массивы данных, алгоритм обработки, прикладные исследования, социально-экономические исследования, технические аспекты, научная работа

Благодарности. Статья подготовлена по результатам исследований, выполненных за счет бюджетных средств по государственному заданию Финансового университета при Правительстве Российской Федерации.

Для цитирования: Конищев Е.С. Обработка больших массивов данных в Python для целей прикладных социально-экономических исследований: преимущества и актуальные вопросы//Вестник университета. 2024. № 7. С. 44–53.

Processing large amounts of data in Python for the purposes of applied socio-economic research: advantages and current issues

Evgeniy S. Konishchev
Postgraduate Student, Junior Researcher
ORCID: 0000-0001-9733-4821, e-mail: eskonishchev@fa.ru

Financial University under the Government of the Russian Federation, Moscow, Russia

Abstract

The article is devoted to the study of the features of processing large amounts of data using the Python programming language. Unlike tabular processors or finished software products, programming languages offer the user a flexible toolkit for the implementation of tasks. At the same time, this creates certain risks associated with the effectiveness of using appropriate tools and optimising the operation of the programme. The purpose of the article is to study the features of processing large amounts of data in Python on the examples of immediate research tasks. The relevance of the topic and purpose of the article is due to the existing scientific gap related to a comprehensive consideration of the technical aspects of the use of programming languages and associated tools for socio-economic research. Thus, many authors who use programming languages in their works rarely provide information regarding the advantages of certain algorithms or approaches. Within the framework of the article, the author examines the procedures and algorithm of processing a large array of data on the example of specific research tasks. The conclusions are drawn about the features and advantages of Python when working with large amounts of data as well as about the prospects for the development of the relevant scientific topics.

Keywords

Programming language, Python, data analysis, data processing, large data sets, processing algorithm, applied research, socio-economic research, technical aspects, scientific work

Acknowledgements. The article has been prepared based on the results of research conducted at the expense of budgetary funds according to the state assignment of the Financial University under the Government of the Russian Federation.

For citation: Konishchev E.S. (2024) Processing large amounts of data in Python for the purposes of applied socio-economic research: advantages and current issues. *Vestnik universiteta*, no. 7, pp. 44–53.

ВВЕДЕНИЕ

С развитием информационных технологий вопросы, связанные с обработкой и анализом данных, приобретают все большую актуальность, что обусловлено постоянным увеличением доступных массивов данных, а также количества инструментов для работы с ними. Проведение прикладных экономических исследований, призванных решать задачи практического характера, неизбежно оказывается сопряжено с решением соответствующих вопросов, а именно касающихся поиска необходимых данных, инструментов их обработки и комплексного анализа, а также корректного представления результатов [1]. Так, цель статьи состоит в рассмотрении особенностей обработки больших массивов данных с применением инструментария языка программирования Python для социально-экономических работ на примерах непосредственных исследовательских задач. Настоящая статья будет полезна исследователям, которые не являются специалистами в области анализа данных и языках программирования и ищут инструменты, позволяющие эффективно работать с большими объемами данных.

ОБЗОР ВЫПОЛНЕНИЯ ИССЛЕДОВАНИЯ

Процесс обработки данных в Python будет рассмотрен на примере массива статистических данных по межрегиональной торговле (название показателя – Ввоз (покупка) – Вывоз (продажа) продукции без учета поставки на собственную территорию) в Российской Федерации (далее – РФ, Россия) (в натуральном выражении) за период с 2010 г. по 2021 г. (источник данных – Федеральная служба государственной статистики). Исходные данные представлены совокупностью xls- и xlsx-файлов с общим размером 56,1 Мб. Их структура (данные по межрегиональной торговле за год) не является унифицированной: группы товаров по Общероссийскому классификатору продукции по видам экономической деятельности расположены либо на разных листах, либо последовательно на одном листе. Каждый файл содержит приблизительно 140–190 групп товаров. Кроме того, структура самих таблиц также не является унифицированной (количество пустых строк в верху таблицы, наличие/отсутствие пустых ячеек, размерность таблицы).

Данные представлены в виде матриц размерностью приблизительно 95 на 106 (размерности таблиц различаются от года к году): строки – субъекты РФ (включая сумму по РФ, федеральные округа и другие дополнительные категории), куда осуществляется ввоз соответствующей группы товаров, столбцы – субъекты РФ (включая сумму по РФ и федеральные округа), откуда осуществляется вывоз соответствующей группы товаров (рис. 1). Необходимо отметить, что для последующего анализа и визуализации данных подобная структура представляется не полностью подходящей. Иными словами, и табличный процессор, и язык программирования будут видеть таблицу как совокупность записей (строк), столбцы при этом выполняют роль своего рода разметки каждой записи на составляющие. В этой связи данные были реструктурированы в таблицу однотипных построчных записей (рис. 1). По сути, речь идет о модели, которая широко применяется в базах данных.

Иными словами, возможности последующей работы (комплексный анализ, расчет вспомогательных параметров и т.д.) с матричными таблицами ограничены в силу того, что табличный процессор или язык программирования будут читать таблицу построчно, при этом столбцы будут выделяться в качестве отдельных параметров.

Регион оттока → Регион притока ↓	Субъект РФ_1	Субъект РФ_2	Субъект РФ_n
Субъект РФ_1		[значение 2-1]	[значение n-1]
Субъект РФ_2	[значение 1-2]		[значение n-2]
Субъект РФ_n	[значение 1-n]	[значение 2-n]	

Регион оттока	Регион притока	Значение
Субъект РФ_1	Субъект РФ_1	
Субъект РФ_1	Субъект РФ_2	[значение 1-2]
Субъект РФ_1	Субъект РФ_n	[значение 1-n]
Субъект РФ_2	Субъект РФ_1	[значение 2-1]
Субъект РФ_2	Субъект РФ_2	
Субъект РФ_2	Субъект РФ_n	[значение 2-n]
Субъект РФ_n	Субъект РФ_1	[значение n-1]
Субъект РФ_n	Субъект РФ_2	[значение n-2]
Субъект РФ_n	Субъект РФ_n	

Составлено автором по материалам исследования

Рис. 1. Сравнительное представление матричной и построчной таблиц

Например, чтобы рассчитать нормированные значения для матричной таблицы, необходимо создать дополнительную таблицу такой же размерности, как и первоначальная, в которой станет производиться расчет нормированных значений, в то время как для таблицы с построчными записями нужно лишь сделать дополнительный столбец, который будет выполнять для каждой строки роль параметра, содержащего соответствующие значения.

Кроме того, принимая во внимание представленное выше описание данных, можно подсчитать, что общий объем значений всех файлов составляет примерно 14 млн:

1) 12 файлов – около 14 млн значений (по всему массиву данных);
2) 140–190 категорий товаров – около 1,4–1,9 млн значений (в каждом файле);
3) матрица 95 на 106 – около 10 тыс. значений (в каждой таблице).

Несмотря на большую популярность и достаточно широкие функциональные возможности, Microsoft Excel также характеризуется ограничениями в части работы с данными. Самое значимое из них: xlsx-файлы не могут содержать более 1 млн строк, и, соответственно, Microsoft Excel не может работать с файлами, которые выходят за пределы этого ограничения. В Google Таблицах присутствует схожий лимит (не более 2,5 млн строк). Таким образом, табличные процессоры существенно ограничены в объеме обрабатываемых данных, а следовательно, их полноценное применение было невозможно для целей анализа межрегиональной торговли. Необходимо отметить, что соответствующий лимит можно обойти созданием дополнительных листов в книге (файле), однако эта опция не представляется лучшим решением (создание дополнительных листов не позволяет проводить обработку их всех как единого массива; преобразования будет необходимо делать для каждого листа). Вместе с тем, даже если рассматривать массив данных объемом приблизительно в 1 млн строк, можно утверждать, что неавтоматизированная обработка данных в табличном процессоре крайне затруднительна и малоэффективна, что связано с объективными возможностями человеческого восприятия и техническими возможностями интерфейса табличного процессора [2].

Говоря о более автоматизированной обработке данных с помощью макросов или встроенного в табличные процессоры интерпретатора Visual Basic, можно отметить несколько проблемных аспектов. Во-первых, макросы позволяют автоматизировать преимущественно шаблонные операции/преобразования, которые возможны для уже обработанных данных как минимум в части их структуризации. В иных случаях обработка только с помощью макросов может требовать значительных затрат времени (на запись макросов для каждой возможной вариации необходимого преобразования).

Резюмируя вводную и обзорную части статьи, нужно подчеркнуть, что подходы и инструменты, которые будут рассмотрены далее, в большей степени актуальны для работы именно с большими массивами данных.

ОБОСНОВАНИЕ ВЫБОРА ИНСТРУМЕНТАРИЯ PYTHON

Среди возможных инструментов автоматизированной обработки больших массивов данных для целей исследования выбор в пользу языка программирования Python был сделан по нескольким основным причинам:

1) простой и понятный синтаксис в сравнении с другими языками программирования;
2) большое количество доступных библиотек, расширяющих и дополняющих стандартные возможности Python;
3) отсутствие ограничений на объем массива обрабатываемых данных;
4) гибкие возможности экспорта данных и интеграции с другими средами выполнения и программными продуктами.

Нельзя отрицать, что у Python есть свои слабые места, если сравнивать его с другими языками программирования. Однако они преимущественно сосредоточены в технических аспектах работы языка, а следовательно, будут значимы при реализации более специфичных задач и при проведении более специализированных исследований (например, по результатам которых нужно получить не только массив обработанных данных или их аналитику, а какой-то программный продукт).

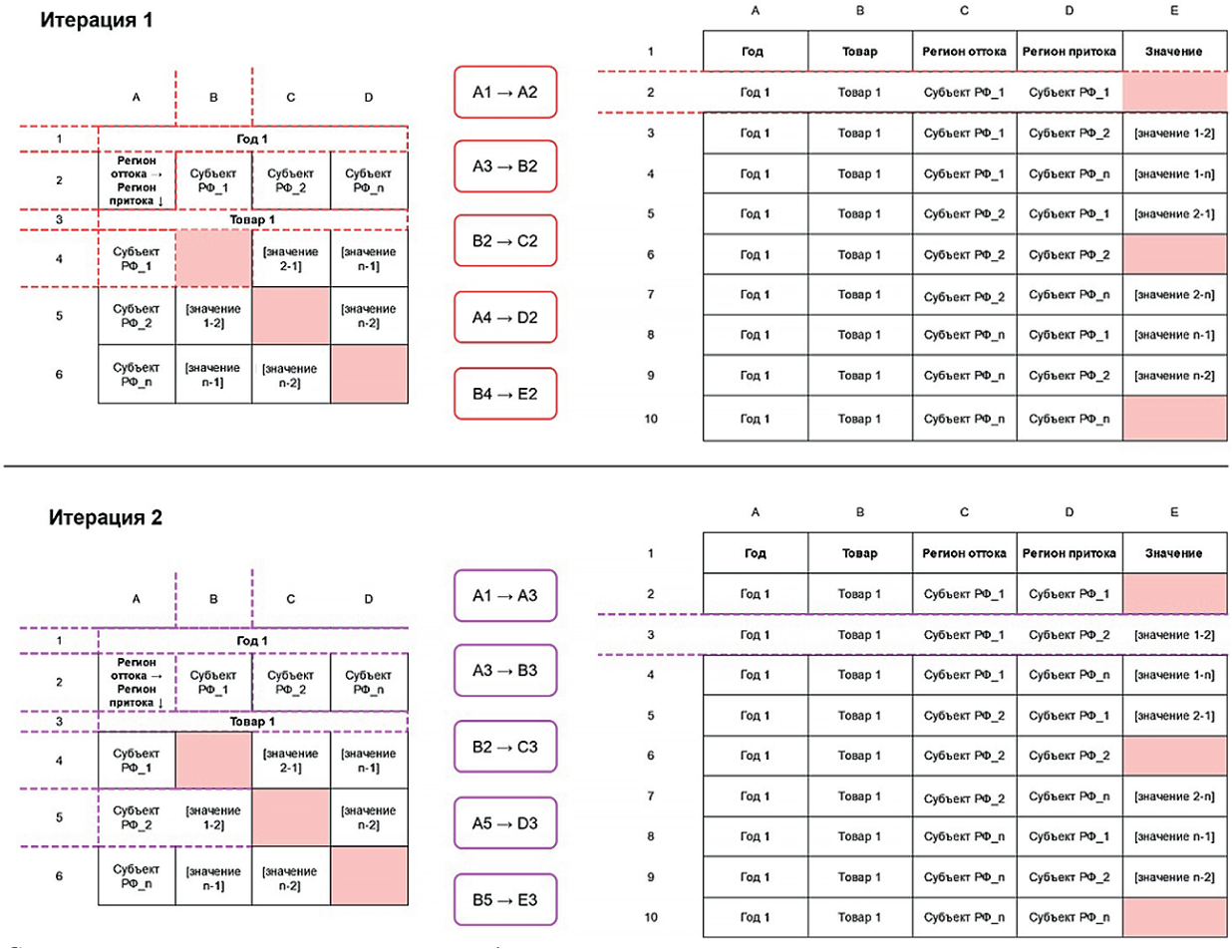
В качестве среды выполнения исследовательских задач был выбран Jupyter Lab[1]. Существенным преимуществом Jupyter Lab или его предшествующей версии Jupyter Notebook в сравнении с другими интерпретаторами или интегрированными средами разработки (англ. integrated development environment)

[1] Jupyter. Официальный сайт. Режим доступа: https://docs.jupyter.org/en/latest/ (дата обращения: 28.03.2024).

Python можно считать возможность реализации подхода к грамотному программированию, когда части программного кода дополняются комментариями на естественном языке, что упрощает понимание кода и программных алгоритмов. В рамках процесса обработки данных использовались библиотеки Pandas[2] и NumPy[3]. NumPy предоставляет возможности работы с многомерными массивами и матрицами, а также комплекс соответствующих вычислительных алгоритмов. В свою очередь работа Pandas во многом основана на возможностях NumPy: библиотека дает инструментарий для обработки и анализа данных за счет манипулирования таблицами и временными рядами.

ОСОБЕННОСТИ ПРОЦЕССОВ И АЛГОРИТМОВ ОБРАБОТКИ ДАННЫХ

Принимая во внимание значительную разницу структуры исходных и итоговых данных, отметим, что их реструктуризация фактически представляла собой парсинг (процедура автоматизированного сбора и систематизации информации). Другими словами, реструктурировать матричную таблицу в построчную с помощью простых манипуляций (под простыми манипуляциями автор понимает набор действий с массивом, которые подразумевают его разделение на отдельные ряды данных и их последующую компиляцию в соответствии с заданными условиями) не представляется эффективным (рис. 2). Неэффективность заключается в том, что реструктурируемые ячейки в матричной таблице не являются серией (ячейки находятся в разных столбцах и строках). Таким образом для их корректной обработки необходимо ввести промежуточный этап, в рамках которого из нужных ячеек будет создан интерируемый объект (объект, способный возвращать элементы по одному и хранящий их в определенной последовательности). Его элементы станут впоследствии присваиваться соответствующей строке новой таблицы. При этом практическая целесообразность такого подхода остается под вопросом, так как в алгоритм обработки добавляется дополнительная операция, которая не дает никаких очевидных преимуществ.



Составлено автором по материалам исследования

Рис. 2. Процесс реструктуризации исходных данных на примере первых двух итераций

[2] Pandas. Официальный сайт. Режим доступа: https://pandas.pydata.org/ (дата обращения: 28.03.2024).

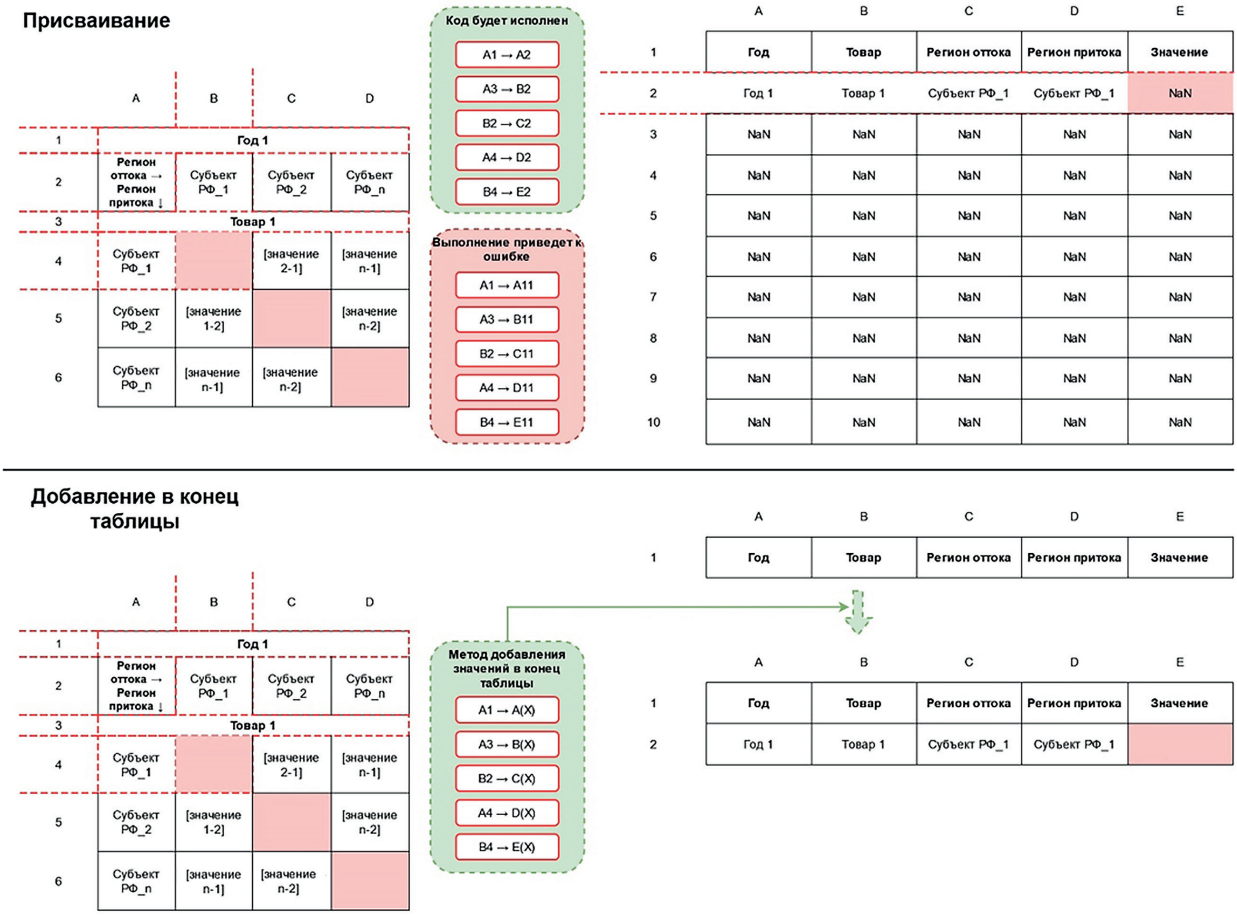
[3] NumPy. Официальный сайт. Режим доступа: https://numpy.org/ (дата обращения: 28.03.2024).

Поэтому автором исследования было принято решение полного переписывания таблиц под новую структуру с последующим их объединением. Вместо описанного выше подхода значения ячеек исходной таблицы по отдельности расставляются в нужные ячейки новой. Иными словами, таблицы реструктурируются поэлементно. Подобный алгоритм позволяет избежать применения дополнительных операций с созданием переменных, а также присваиванием и хранением в них значений исходной таблицы, что сокращает время исполнения кода. Однако необходимо отметить, что поэлементная обработка массива данных все равно остается крайне времязатратной операцией, срок выполнения которой будет увеличиваться относительно объема исходного и конечного массивов.

Это объясняется рядом особенностей работы с массивами в Python. Так, добавление значений в новую таблицу может быть реализовано двумя основными способами (рис. 3):

1) присваивание значений конкретным ячейкам. Позволяет адресно проставлять значения в соответствии с заданными условиями (например, номер строки (индекс) и название столбца). Однако для этого необходимо, чтобы размерность таблицы была задана изначально, то есть таблица перед заполнением должна содержать нужные ячейки, которым будут присваиваться значения. В рассматриваемом примере мы знаем, что у нашей таблицы 5 столбцов, при этом точное число строк остается неизвестным. Если ее размерность окажется меньше, чем количество данных, которые мы хотим ей присвоить, то при исполнении кода будет выдана ошибка (программа не сможет присвоить значения несуществующим ячейкам);

2) добавление каждого значения в конец таблицы. Данный способ не требует изначального указания размерности, так как по сути таблица дополняется новыми ячейками и значениями. При этом для добавления каждого следующего значения алгоритм будет просматривать сначала всю таблицу, чтобы найти ее конец, и только затем дополнит новым. Таким образом, по мере увеличения количества добавленных значений будет расти время, затрачиваемое на просмотр таблицы и поиск ее окончания, чтобы дополнить очередным значением.



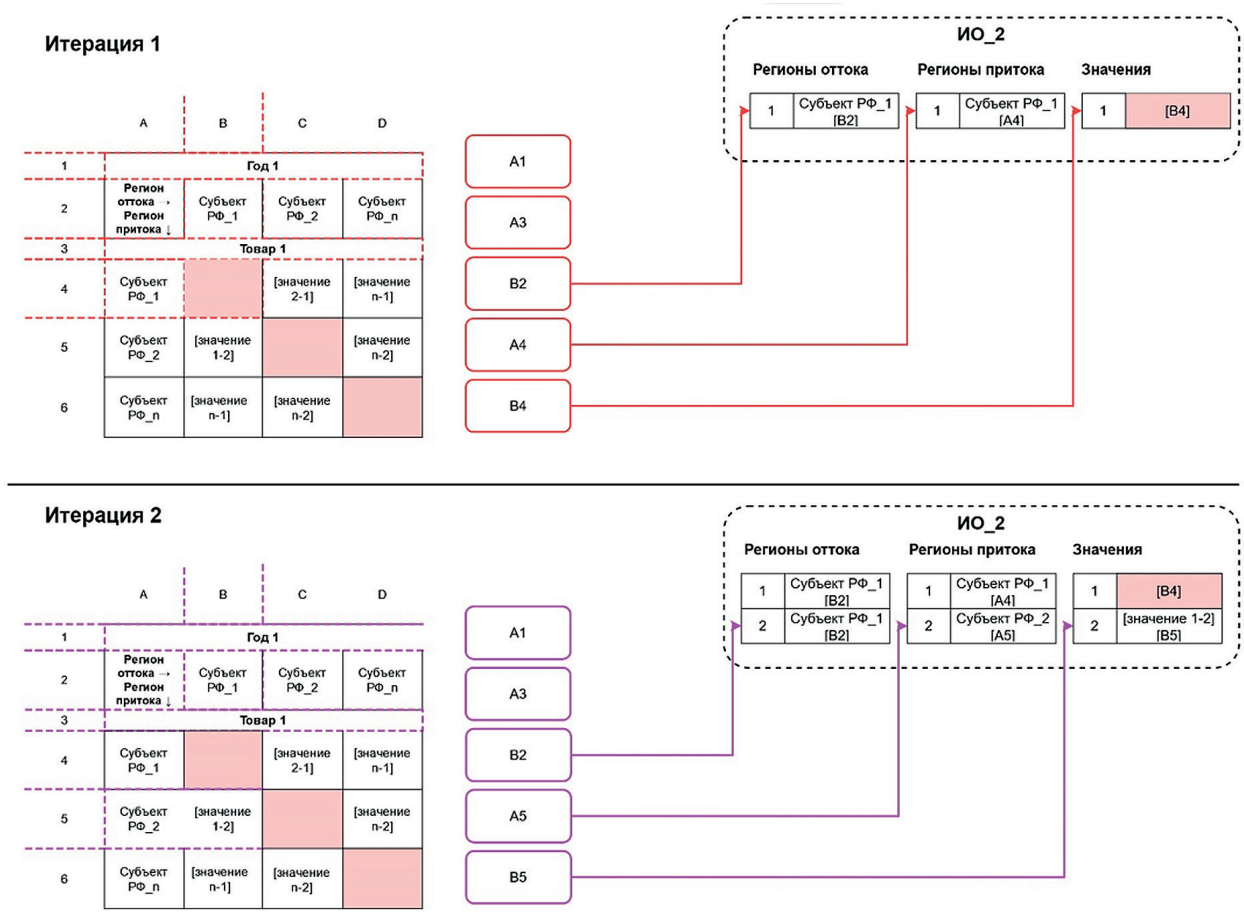
Примечание: NaN – not a number (англ. не число)

Составлено автором по материалам исследования

Рис. 3. Концептуальная визуализация процесса добавления значений в новую таблицу

В процессе тестирования алгоритмов реструктуризации данных было выявлено, что время, затрачиваемое на обработку данных по каждому файлу, увеличивалось более чем в 5 раз (с 8 до 40 мин). С учетом общего количества товарных групп автор признал соответствующие затраты времени нецелесообразными и произвел оптимизацию алгоритма.

Суть оптимизации заключалась во включении в алгоритм промежуточных итерируемых объектов (далее – ИО), в которые будут добавляться значения исходной таблицы в процессе обработки одной группы товаров. Несмотря на то, что ранее автором высказывалась позиция против промежуточных этапов обработки, их включение в итоговый вариант алгоритма обусловлено необходимостью уменьшить количество операций по добавлению значений в конец таблицы, чтобы сократить время исполнения кода. Так, были выделены два основных цикла обработки: цикл по году (по файлу) и цикл по категории товаров. Для каждого из них введен свой набор ИО, которые будут использоваться для временного хранения исходных данных. Для простоты восприятия далее они упоминаются как ИО первого (ИО_1, цикл по году) и второго (ИО_2, цикл по категории товаров) уровней. В рамках цикла по категории товара исходные данные собираются в ИО второго уровня (рис. 4). При этом на настоящем этапе объединяются в ИО данные «Региона оттока», «Региона притока» и «Значений».

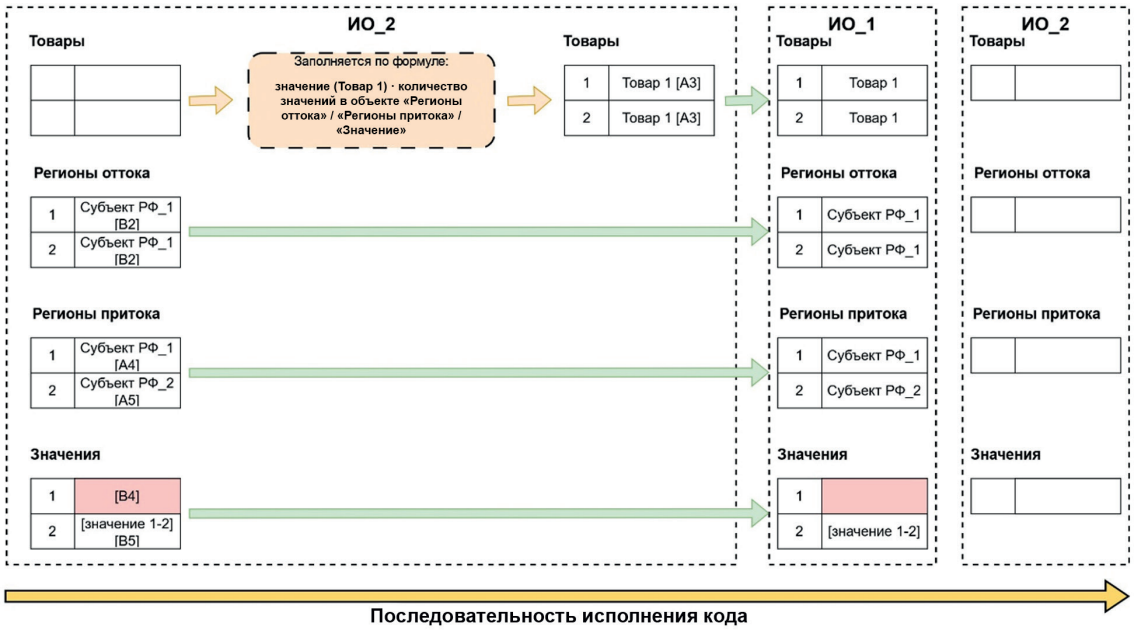


Составлено автором по материалам исследования

Рис. 4. Сбор данных в ИО второго уровня

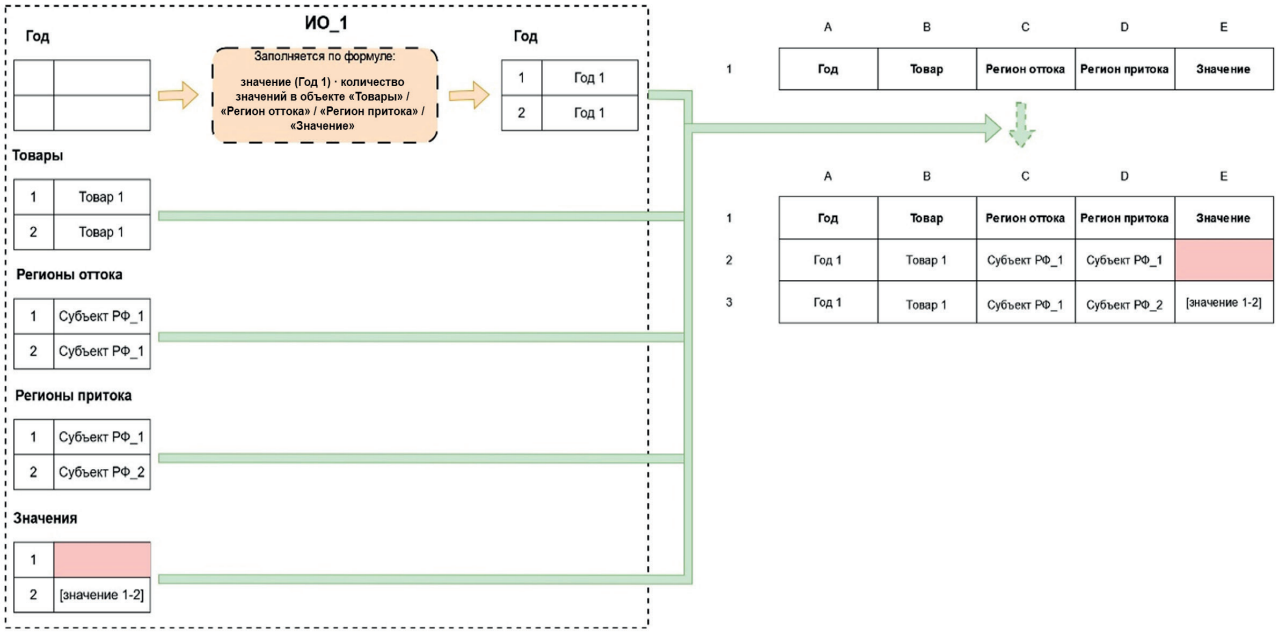
Когда все данные по категории товаров собраны, производится заполнение ИО «Товар» (значение товара заполняется в конце данного цикла, так как оно будет одинаковым для всех субъектов), а значения из ИО второго уровня переносятся (добавляются в конец) в ИО первого уровня, после чего данные из ИО второго уровня удаляются и начинается сбор данных по следующей категории товаров (рис. 5).

После завершения сбора данных по всем категориям товаров в файле производятся заполнение ИО «Год» (значение года заполняется после обработки файла, так как оно будет одинаковым для всех субъектов по всем категориям товаров) и транзит (добавляются в конец) собранных значений из ИО первого уровня в новую таблицу, после чего данные из ИО первого уровня удаляются и начинается сбор данных из следующего файла (за следующий год) (рис. 6).



Составлено автором по материалам исследования

Рис. 5. Процесс транзита значений между ИО разного уровня (между циклами)



Составлено автором по материалам исследования

Рис. 6. Процесс транзита значений между ИО первого уровня и новой таблицей

Иными словами, в рамках оптимизации в алгоритм были включены контейнеры, которые служат для временного накопления и хранения данных, чтобы сократить время, затрачиваемое на добавление исходных данных в новую таблицу. Суть соответствующего решения состоит в том, что промежуточные объекты для хранения данных позволяют нивелировать ключевую слабую сторону операции по их добавлению в конец массива – рост времени на просмотр массива при увеличении в нем количества данных. Так, по результатам оптимизации время, затрачиваемое на обработку данных по одной товарной группе, сократилось с 8 до 4 мин, а также удалось избежать накопительного роста временных затрат на каждый цикл. Если ранее при обработке данных по каждой следующей группе товаров наблюдалось увеличение сроков, то после оптимизации алгоритма эта проблема была полностью устранена.

По завершении обработки всех исходных файлов новые таблицы объединяются в единый массив и сохраняются в формате .csv (англ. comma-separated values – значения, разделенные запятыми, далее – CSV). Его выбор обусловлен тем, что CSV – это текстовый формат для хранения табличных данных, где каждая строка файла соответствует строке таблицы (что подходит для сконструированной модели данных). Также среди ключевых преимуществ CSV можно выделить быстрый доступ к данным и эффективное использование памяти в сравнении с форматами Excel. Кроме того, у SCV-файлов нет ограничений на количество данных (в отличие от .xlsx и подобных).

КОММЕНТАРИИ И ОБСУЖДЕНИЕ

Поскольку в рамках исследования не стояло задачи развертывания постоянной системы или базы данных, вопрос выбора формата их хранения не был актуальным. Несмотря на это, важно уточнить, что по мере роста объема обрабатываемых данных, в условиях необходимости создания системы работы с данными и обращения к ним может быть актуальным выбор более современных форматов или систем сериализации данных (например, Apache Avro, Apache Parquet и др.). В отличие от CSV-файлов, Avro или Parquet можно в меньшей степени назвать человекочитаемыми. Однако они предоставляют большие функциональные возможности, а также обладают расширенной поддержкой сжатия (CSV не поддерживает сжатие), что будет важным фактором скорости работы системы. Также подобные форматы позволяют хранить метаданные блоков данных. Блоковое разделение и наличие метаданных помогают существенно оптимизировать задачи, связанные с параллельной обработкой данных.

Как было отмечено во введении, общий вес исходных файлов составил 56,1 Мб (около 5 Мб на один файл). При этом общий вес реструктурированных данных составляет 2,5 Гб (около 190 Мб на один файл). Такая значительная разница в весе объясняется тем, что при матричной структуре отсутствует многократное дублирование одних и тех же значений (в данном случае – года, названий товаров и регионов). Увеличение объема файлов представляется неизбежным следствием оптимизации структуры данных. Необходимо отметить, что хотя рассмотренный массив данных очень условно можно отнести к большим данным, его обработка проводилась в одном узле (другими словами, на одном устройстве), поскольку временные рамки выполнения кода удалось уменьшить благодаря оптимизации алгоритмов. Однако, работая с массивами большого размера или при необходимости выполнения большего количества операций, целесообразно применять параллельную обработку данных: операционные контейнеры, фреймворк Apache Spark[4] и т.д.

ЗАКЛЮЧЕНИЕ

Таким образом, Python предоставляет гибкие возможности для автоматизации работы с большими массивами данных, что может стать крайне актуальным в решении отдельных исследовательских задач. Однако основной задачей автора было не рассмотреть весь спектр возможностей, а продемонстрировать особенности и проблемы их практической реализации. Существующий пробел российского научного поля по соответствующей тематике заключается в недостаточном внимании к изучению более практических особенностей работы с языками программирования при решении научно-исследовательских задач, в частности по социально-экономической тематике.

Несмотря на растущую популярность инструментов программирования, комплексный анализ их применения в российских работах по социально-экономической тематике остается на сравнительно низком уровне. Иными словами, крайне редко можно встретить исследования, в которых не только упоминается, что автор использовал инструментарий языка программирования [3], но и приводятся ответы на следующие вопросы: «Зачем использовать язык программирования для этой исследовательской задачи?», «В чем преимущества именно такого алгоритма/подхода к решению соответствующей задачи?» и т.п.

Необходимо уточнить, что языки программирования (Python и др.), предоставляя исследователю гибкие возможности работы с данными, вместе с тем несут определенные риски, касающиеся как эффективности их использования, так и корректности авторских подходов в целом. Поэтому накопление исследовательского опыта, включая представление не только результатов, но и самих алгоритмов работы [4–7], может внести значительный вклад в дальнейшую популяризацию и качественное развитие подобных инструментов. Кроме того, в работе с языками программирования одной из ключевых и наиболее

[4] Apache Spark. Официальный сайт. Режим доступа: https://spark.apache.org/ (дата обращения: 04.04.2024).

трудозатратных задач остается во многом не написание самой программы, а ее оптимизация. Как было рассмотрено выше, эффективность неоптимизированной и оптимизированной программ отличается значительно, а следовательно, вопросы эффективности программного кода и возможностей его оптимизации должны освещаться учеными, которые непосредственно решают эти задачи.

Список литературы

1. *Цыпин А.П., Сорокин А.С.* Статистические пакеты программ в социально-экономических исследованиях. Азимут научных исследований: экономика и управление. 2016;4(17(5):379–384.
2. *Карпович А.О.* Способы обработки экономической информации посредством языка программирования Python. В кн.: XVI Машеровские чтения: материалы международной научно-практической конференции студентов, аспирантов и молодых ученых, том 1, Витебск, 21 октября 2022 г. Витебск: Витебский государственный университет имени П.М. Машерова; 2022. С. 31–32.
3. *Рабинович А.Е., Седова Е.А.* Постановка первоначальных аналитических гипотез с использованием визуализации данных на примере изучения влияния социально-экономических факторов на состояние психического здоровья населения. Российский экономический интернет-журнал. 2020;4:44–52.
4. *Русинова М.Р.* Стратегирование развития территориальных социально-экономических систем на основе использования инструментария «умного» бенчмаркинга. Дис. ... канд. экон. наук: 08.00.05. Пермь: Пермский национальный исследовательский политехнический университет; 2021. 152 с.
5. *Семенюта О.Г., Дубинина И.В., Дегтярев А.С.* Совершенствование инструментария идентификации ключевых социально-экономических факторов влияния на рынок ипотечного жилищного кредитования России. Финансовые исследования. 2021;4(73):106–113.
6. *Rungskunroch P., Shen Z.-J., Kaewunruen S*. Benchmarking socio-economic impacts of high-speed rail networks using k-nearest neighbour and Pearson's correlation coefficient techniques through computational model-based analysis. Applied Sciences. 2022,3(12). http://dx.doi.org/10.3390/app12031520
7. *Chen M.-Y., Chen T.-H.* Modeling public mood and emotion: blog and news sentiment and socio-economic phenomena. Future Generation Computer Systems. 2019;96:692–699. http://dx.doi.org/10.1016/j.future.2017.10.028

References

1. *Tsypin A.P., Sorokin A.S*. Statistical software packages in social and economic researches. Azimut of Scientific Research: Economics and Administration. 2016;4(17(5):379–384. (In Russian).
2. *Karpovich A.O.* Methods of processing economic information through the Python, programming language. In: XVI Masherov readings: Proceedings of the International Scientific and Practical Conference of Students, Postgraduate Students and Young Scientists, volume 1, Vitebsk, October 21, 2022. Vitebsk: Vitebsk State University named after P.M. Masherov; 2022. Pp. 31–32. (In Russian).
3. *Rabinovich A.E., Sedova E.A.* Statement of initial analytical hypotheses using data visualization on the example of studying the influence of socio-economic factors on the state of mental health of the population. Russian Economics online journal. 2020;4:44–52. (In Russian).
4. *Rusinova M.R.* Strategizing the development of territorial socio-economic systems based on the use of "smart" benchmarking tools. Diss. ... Cand. Sci. (Econ.): 08.00.05. Perm: Perm National Research Polytechnic University; 2021. 152 p. (In Russian).
5. *Semenyuta O.G., Dubinina I.V., Degtyarev A.S*. Improvement of tools to identify key socio-economic factors of influence on the market of housing mortgage lending in Russia. Financial Research. 2021;4(73):106–113. (In Russian).
6. *Rungskunroch P., Shen Z.-J., Kaewunruen S*. Benchmarking socio-economic impacts of high-speed rail networks using k-nearest neighbour and Pearson's correlation coefficient techniques through computational model-based analysis. Applied Sciences. 2022,3(12). http://dx.doi.org/10.3390/app12031520
7. *Chen M.-Y., Chen T.-H.* Modeling public mood and emotion: blog and news sentiment and socio-economic phenomena. Future Generation Computer Systems. 2019;96:692–699. http://dx.doi.org/10.1016/j.future.2017.10.028