Содержание

Предисловие

Что нового во втором издании?

Первое издание этой книги вышло в 2012 году, когда Python-библиотеки для анализа данных с открытым исходным кодом (в частности, pandas) были еще внове и быстро развивались. В этом исправленном и дополненном издании я переработал главы, стремясь отразить как несовместимые изменения и устаревшие возможности, так и новые средства, появившиеся за прошедшие пять лет. Я также добавил новый материал с описанием инструментов, которые либо еще не существовали в 2012 году, либо были недостаточно зрелыми. Наконец, я старался не писать о новых или активно разрабатываемых проектах с открытым кодом, которым, возможно, не суждено дожить до зрелости. Хотелось бы представить читателям этого издания средства, которые не утратят актуальности ни в 2020, ни в 2021 году.

Ниже перечислены основные отличия второго издания.

- Весь код, включая пособие по Python, обновлен и доведен до уровня версии 3.6 (в первом издании использовалась версия Python 2.7).
- Обновлены инструкции по установке Python из дистрибутива Anaconda Python Distribution, а также всех дополнительных Python-пакетов.
- Внесены исправления, соответствующие последним версиям библиотеки pandas, существовавшим в 2017 году.
- Добавлена глава о дополнительных средствах pandas, дающая также ряд других советов по работе с библиотекой.
- Размещено краткое введение в библиотеки statsmodels и scikit-learn.

Кроме того, материал, вошедший в первое издание, подвергнут реорганизации, чтобы книгу было проще читать начинающим пользователям.

Графические выделения

В книге применяются следующие графические выделения.

Курсив
Новые термины, имена и расширения имен файлов.

`Моноширинный`
Листинги программ, а также элементы кода в основном тексте: имена переменных и функций, базы данных, типы данных, переменные окружения, предложения и ключевые слова языка.

`Моноширинный полужирный`
Команды или иной текст, который должен быть введен пользователем буквально.

`Моноширинный курсив`
Текст, вместо которого следует подставить значения, заданные пользователем или определяемые контекстом.

Так обозначается совет или рекомендация.

Так обозначается замечание общего характера.

Так обозначается предостережение или предупреждение.

О примерах кода

Файлы данных и прочие материалы, организованные по главам, можно найти в репозитории книги на GitHub: http://github.com/wesm/pydata-book.

Эта книга призвана помогать вам в работе, поэтому вы можете использовать приведенный в ней код в собственных программах и в документации. Спрашивать у нас разрешения необязательно, если только вы не собираетесь воспроизводить значительную часть кода. Например, не возбраняется включить в свою программу несколько фрагментов кода из книги, однако для продажи или распространения примеров из книг издательства O'Reilly на компакт-диске разрешение требуется. Цитировать книгу и примеры в ответах на вопросы можно без ограничений, но для включения значительных объемов кода в документацию по собственному продукту нужно получить позволение.

Мы высоко ценим (хотя и не требуем их размещения) ссылки на наши издания. В ссылке обычно указывается название книги, имя автора, издатель-

ство и ISBN, например: «Python for Data Analysis by Wes McKinney (O'Reilly). Copyright 2017 Wes McKinney, 978-1-491-95766-0».

Если вы полагаете, что планируемое использование кода выходит за рамки изложенной выше лицензии, пожалуйста, обратитесь к нам по адресу permissions@oreilly.com.

Отзывы и пожелания

Мы всегда рады отзывам наших читателей. Расскажите нам, что вы думаете об этой книге – что понравилось или, может быть, не понравилось. Отзывы важны для нас, чтобы выпускать книги, которые будут для вас максимально полезны.

Вы можете написать отзыв на нашем сайте www.dmkpress.com, зайдя на страницу книги и оставив комментарий в разделе «Отзывы и рецензии». Также можно послать письмо главному редактору по адресу dmkpress@gmail.com; при этом укажите название книги в теме письма.

Если вы являетесь экспертом в какой-либо области и заинтересованы в написании новой книги, заполните форму на нашем сайте по адресу http://dmkpress.com/authors/publish_book/ или напишите в издательство по адресу dmkpress@gmail.com.

Скачивание исходного кода примеров

Скачать файлы с дополнительной информацией для книг издательства «ДМК Пресс» можно на сайте www.dmkpress.com или www.дмк.рф на странице с описанием соответствующей книги.

Список опечаток

Хотя мы приняли все возможные меры для того, чтобы обеспечить высокое качество наших текстов, ошибки все равно случаются. Если вы найдете ошибку в одной из наших книг – возможно, ошибку в основном тексте или программном коде, – мы будем очень благодарны, если вы сообщите нам о ней. Сделав это, вы избавите других читателей от недопонимания и поможете нам улучшить последующие издания этой книги.

Если вы найдете какие-либо ошибки в коде, пожалуйста, сообщите о них главному редактору по адресу dmkpress@gmail.com, и мы исправим это в следующих тиражах.

Нарушение авторских прав

Пиратство в интернете по-прежнему остается насущной проблемой. Издательства «ДМК Пресс» и Packt очень серьезно относятся к вопросам защиты

авторских прав и лицензирования. Если вы столкнетесь в интернете с незаконной публикацией какой-либо из наших книг, пожалуйста, пришлите нам ссылку на интернет-ресурс, чтобы мы могли применить санкции.

Ссылку на подозрительные материалы можно прислать по адресу электронной почты dmkpress@gmail.com.

Мы высоко ценим любую помощь по защите наших авторов, благодаря которой мы можем предоставлять вам качественные материалы.

Благодарности

Эта работа – плод многолетних плодотворных обсуждений и совместной работы с многочисленными людьми со всего света. Хочу поблагодарить некоторых из них.

Памяти Джона Д. Хантера (1968–2012)

В августе 2012 года после многолетней борьбы с раком толстой кишки ушел из жизни наш дорогой друг и коллега Джон Д. Хантер. Это произошло почти сразу после того, как я закончил рукопись первого издания книги.

Роль и влияние Джона на сообщества, специализирующиеся на применении Python в научных приложениях и обработке данных, трудно переоценить. Помимо разработки библиотеки matplotlib в начале 2000-х годов (время, когда Python был далеко не так популярен, как сейчас) он помогал формировать культуру целого поколения разработчиков открытого кода, ставших впоследствии столпами экосистемы Python, которую мы часто считаем само собой разумеющейся.

Мне повезло познакомиться с Джоном в начале своей работы над открытым кодом в январе 2010 года, сразу после выхода версии pandas 0.1. Его вдохновляющее руководство помогало мне даже в самые тяжелые моменты не отказываться от своего видения pandas и Python как полноправного языка для анализа данных.

Джон был очень близок с Фернандо Пересом (Fernando Perez) и Брайаном Грейнджером (Brian Granger), которые заложили основы IPython и Jupyter и были авторами многих других инициатив в сообществе Python. Мы надеялись работать над книгой вчетвером, но в итоге только у меня оказалось достаточно свободного времени. Я уверен, что он гордился бы тем, чего мы достигли, порознь и совместно, за прошедшие пять лет.

Благодарности ко второму изданию (2017)

Прошло почти пять лет с того времени, как я закончил рукопись первого издания книги. Случилось это в июле 2012 года. С тех пор многое изменилось. Сообщество Python неизмеримо выросло, а сложившаяся вокруг него экосистема программных продуктов с открытым исходным кодом процветает.

Новое издание не появилось бы на свет без неустанных усилий разработчиков ядра pandas, благодаря которым этот проект и сложившееся вокруг него сообщество превратились в один из краеугольных камней экосистемы Python в области науки о данных. Назову лишь некоторых: Том Аугспургер (Tom Augspurger), Йорис ван ден Боше (Joris van den Bossche), Крис Бартак (Chris Bartak), Филлип Клауд (Phillip Cloud), gfyoung, Энди Хэйден (Andy Hayden), Масааки Хорикоши (Masaaki Horikoshi), Стивен Хойер (Stephan Hoyer), Адам Клейн (Adam Klein), Воутер Овермейр (Wouter Overmeire), Джэфф Ребэк (Jeff Reback), Чань Ши (Chang She), Скиппер Сиболд (Skipper Seabold), Джефф Трэтнер (Jeff Tratner) и др.

Что касается собственно подготовки издания, то я благодарю сотрудников издательства O'Reilly, которые терпеливо помогали мне на протяжении всего процесса работы над книгой, а именно Мари Божуро (Marie Beaugureau), Бена Лорика (Ben Lorica) и Коллин Топорек (Colleen Toporek). В очередной раз у меня были замечательные технические редакторы: Том Аугспургер, Пол Бэрри (Paul Barry), Хью Браун (Hugh Brown), Джонатан Коу (Jonathan Coe) и Андреас Маллер (Andreas Muller). Спасибо вам.

Первое издание книги переведено на ряд иностранных языков, в том числе на китайский, французский, немецкий, японский, корейский и русский. Перевод этого текста с целью сделать его доступным более широкой аудитории – трудное и зачастую неблагодарное занятие. Благодарю вас за то, что помогаете людям во всем мире учиться программировать и использовать средства анализа данных.

Мне также повезло пользоваться на протяжении нескольких последних лет поддержкой своих трудов по разработке ПО с открытым исходным кодом со стороны сайта Cloudera и фонда Two Sigma Investments. В то время как открытые проекты получают все меньший объем ресурсов, несопоставимый с количеством пользователей, очень важно, чтобы коммерческие компании поддерживали разработку ключевых программных проектов. Это было бы правильно.

Благодарности к первому изданию (2012)

Мне было бы трудно написать эту книгу без поддержки многих людей.

Из сотрудников издательства O'Reilly я крайне признателен редакторам – Меган Бланшетт (Meghan Blanchette) и Джулии Стил (Julie Steele), которые направляли меня на протяжении всего процесса. Майк Лоукидес (Mike Loukides) также работал со мной на стадии подачи предложения и помогал с выпуском книги в свет.

В техническом рецензировании книги многие принимали участие. Мартин Лас (Martin Blais) и Хью Браун (Hugh Brown) оказали неоценимую помощь в повышении качества примеров, ясности изложения и улучшении организации книги в целом. Джеймс Лонг (James Long), Дрю Конвей (Drew Conway), Фернандо Перес, Брайан Грейнджер, Томас Клюйвер (Thomas Kluyver), Адам

Клейн, Джон Клейн, Чань Ши и Стефан ван дер Вальт (Stefan van der Walt) отрецензировали по одной или по нескольку глав и сделали ценные замечания с разных точек зрения.

Я почерпнул немало отличных идей для примеров и наборов данных в беседах с друзьями и коллегами, в том числе с Майком Дьюаром (Mike Dewar), Джеффом Хаммербахером (Jeff Hammerbacher), Джеймсом Джондроу (James Johndrow), Кристианом Ламом (Kristian Lum), Адамом Клейном, Хилари Мейсон (Hilary Mason), Чань Ши и Эшли Вильямсом (Ashley Williams).

Конечно, я в долгу у многих лидеров сообщества, применяющего открытое ПО на Python в научных приложениях, поскольку именно они заложили фундамент моей работы и воодушевляли меня, пока я писал книгу. Это те, кто разрабатывал ядро IPython (Фернандо Перес, Брайан Грейнджер, Мин Рэган-Келли, Томас Клюйвер и др.), Джон Хантер, Скиппер Сиболд, Трэвис Олифант (Travis Oliphant), Питер Вонг (Peter Wang), Эрик Джонс (Eric Jones), Робер Керн (Robert Kern), Джозеф Перктольд (Josef Perktold), Франческ Альтед (Francesc Alted), Крис Фоннесбек (Chris Fonnesbeck) и многие, многие другие. Еще несколько человек оказывали мне значительную поддержку, делились идеями и подбадривали на протяжении всего пути: Дрю Конвей, Шон Тэйлор (Sean Taylor), Джузеппе Палеолого (Giuseppe Paleologo), Джаред Дандер (Jared Lander), Дэвид Эпштейн (David Epstein), Джон Кроуос (John Krowas), Джошуа Блум (Joshua Bloom), Дэн Пилсуорт (Den Pilsworth), Джон Майлз-Уайт (John Myles-White) и многие другие, о которых я забыл.

Я также благодарен всем, кто повлиял на становление меня как ученого. В первую очередь это мои бывшие коллеги по компании AQR, которые поддерживали мою работу над pandas в течение многих лет: Алекс Рейфман (Alex Reyfman), Майкл Вонг (Michael Wong), Тим Сарджен (Tim Sargen), Октай Курбанов (Oktay Kurbanov), Мэтью Щанц (Matthew Tschantz), Рони Израэлов (Roni Israelov), Майкл Кац (Michael Katz), Крис Уга (Chris Uga), Прасад Раманан (Prasad Ramanan), Тэд Сквэр (Ted Square) и Хун Ким (Hoon Kim). И наконец, благодарю моих университетских наставников Хэйнса Миллера (МТИ) и Майка Уэста (университет Дьюк).

Если говорить о личной жизни, то я благодарен Кэйси Динкин (Casey Dinkin), чью каждодневную поддержку невозможно переоценить. Спасибо той, кто терпел перепады моего настроения, когда я пытался собрать окончательный вариант рукописи, несмотря на свой и так уже перегруженный график. Благодарю и моих родителей, Билла и Ким, которые учили меня никогда не отступать от мечты и не соглашаться на меньшее.

Об авторе

Уэс Маккини – разработчик программного обеспечения и предприниматель из Нью-Йорка. После получения степени бакалавра математики в МТИ в 2007 году его приняли на работу в компанию AQR Capital Management в Гринвиче, где занимался финансовой математикой. Неудовлетворенный малопригодными средствами анализа данных, Уэс изучил язык Python и приступил к созданию того, что в будущем стало проектом pandas. Сейчас он активный член сообщества обработки данных на Python и агитирует за использование Python в анализе данных, финансовых задачах и математической статистике.

Впоследствии Уэс стал сооснователем и генеральным директором компании DataPad, технологические активы и коллектив которой в 2014 году приобрела компания Cloudera. С тех пор он занимается технологиями больших данных и является членом комитетов по управлению проектами Apache Arrow и Apache Parquet, курируемыми фондом Apache Software Foundation. В 2016 году автор перешел в компанию Two Sigma Investments из Нью-Йорка, где продолжает трудиться над тем, чтобы средствами ПО с открытым исходным кодом сделать анализ данных быстрее и проще.

Об иллюстрации на обложке

На обложке книги изображена перохвостая тупайя (*Ptilocercus lowii*). Это единственный представитель своего вида в семействе *Ptilocercidae* рода *Ptilocercus*; остальные тупайи принадлежат семейству *Tupaiidae*. Тупайи отличаются длинным хвостом и мягким буро-желтым мехом. У перохвостой тупайи хвост напоминает птичье перо, за что она и получила свое название. Тупайи всеядны, питаются преимущественно насекомыми, фруктами, семенами и небольшими позвоночными животными.

Эти дикие млекопитающие, обитающие в основном в Индонезии, Малайзии и Таиланде, известны хроническим потреблением алкоголя. Как выяснилось, малайские тупайи несколько часов в сутки пьют естественно ферментированный нектар пальмы *Eugeissona tristis*, что эквивалентно употреблению 10–12 стаканов вина, содержащего 3,8 % алкоголя. Но это не приводит к интоксикации их организма благодаря развитой способности расщеплять этиловый спирт, включая его в обмен веществ способами, недоступными человеку. Кроме того, поражает отношение массы мозга к массе тела – оно больше, чем у всех прочих млекопитающих, в том числе у человека.

Несмотря на название, перохвостая тупайя не является настоящей тупайей, а ближе к приматам. Вследствие такого родства перохвостые тупайи стали альтернативой приматам в медицинских экспериментах по изучению миопии, психосоциального стресса и гепатита.

Глава 1. Предварительные сведения

1.1. О чем эта книга?

Книга посвящена вопросам преобразования, обработки, очистки данных и вычислениям на языке Python. Моя цель – предложить руководство по тем частям языка программирования Python и экосистемы его библиотек и инструментов, относящихся к обработке данных, которые помогут вам стать хорошим аналитиком данных. Хотя в названии книги фигурируют слова «анализ данных», основной упор сделан на программировании на Python, на библиотеках и инструментах, а не на методологии анализа данных как таковой. Речь идет о программировании на Python, необходимом для анализа данных.

Какого рода данные?

Говоря «данные», я имею в виду прежде всего *структурированные данные*. Это намеренно расплывчатый термин, охватывающий различные часто встречающиеся виды данных, как то:

- табличные данные – в разных столбцах они могут иметь разный тип (строки, числа, даты или еще что-то). Сюда относятся данные, которые обычно хранятся в реляционных базах или в файлах с запятой в качестве разделителя;
- многомерные списки (матрицы);
- данные, представленные в виде нескольких таблиц, связанных между собой по ключевым столбцам (то, что в SQL называется первичными и внешними ключами);
- равноотстоящие и неравноотстоящие временные ряды.

Это далеко не полный список. Значительную часть наборов данных можно привести к структурированному виду, более подходящему для анализа и моделирования, хотя сразу не всегда очевидно, как это сделать. В тех случаях, когда это не удается, есть возможность извлечь из набора данных структурированное множество признаков. Например, подборку новостных статей можно преобразовать в таблицу частот слов, к которой затем применить анализ эмоциональной окраски.

Большинству пользователей электронных таблиц типа Microsoft Excel, пожалуй, самого широко распространенного средства анализа данных, такие виды данных хорошо знакомы.

1.2. Почему именно Python?

Для многих людей (и для меня в том числе) Python – язык, в который нельзя не влюбиться. С момента появления в 1991 году Python стал одним из самых популярных динамических языков программирования наряду с Perl, Ruby и др. Относительно недавно Python и Ruby приобрели особую популярность как средства создания веб-сайтов в многочисленных каркасах, например Rails (Ruby) и Django (Python). Такие языки часто называют *скриптовыми*, потому что они используются для быстрого написания небольших программ – *скриптов*. Лично мне термин «скриптовый язык» не нравится, потому что он наводит на мысль, будто для создания ответственного программного обеспечения язык не годится. Из всех интерпретируемых языков Python выделяется большим и активным сообществом научных расчетов и анализа данных. За последние десять лет Python превратился из ультрасовременного языка научных расчетов, которым пользуются на свой страх и риск, в один из самых важных языков, применяемых в науке о данных, в машинном обучении и разработке ПО общего назначения в академических учреждениях и промышленности.

При анализе данных и интерактивных научно-исследовательских расчетов с визуализацией результатов Python неизбежно приходится сравнивать со многими предметно-ориентированными языками программирования и инструментами – с открытым исходным кодом и коммерческими, такими как R, MATLAB, SAS, Stata и др. Сравнительно недавно появились улучшенные библиотеки для Python (прежде всего pandas), и он стал серьезным конкурентом в решении задач манипулирования данными. А так как Python еще – и универсальный язык программирования, то это отличный выбор для создания приложений обработки данных.

Python как клей

Своим успехом в области научных расчетов Python отчасти обязан простоте интеграции с кодом на C, C++ и FORTRAN. Во многих современных вычислительных средах применяется общий набор унаследованных библиотек,

написанных на FORTRAN и C, содержащих реализации алгоритмов линейной алгебры, оптимизации, интегрирования, быстрого преобразования Фурье и др. Поэтому многочисленные компании и национальные лаборатории используют Python как клей для объединения написанных за много лет программ.

В значительном количестве программ содержатся небольшие участки кода, на выполнение которых уходит много времени, и внушительные куски склеивающего кода, который выполняют нечасто. В большинстве случаев время выполнения склеивающего кода несущественно, реальную отдачу дает оптимизация узких мест, которые иногда имеет смысл переписать на низкоуровневом языке типа C.

Решение проблемы «двух языков»

Во многих организациях принято для научных исследований, создания опытных образцов и проверки новых идей использовать предметно-ориентированные языки типа MATLAB или R, а затем переносить удачные разработки в производственную систему, написанную на Java, C# или C++. Но все чаще люди приходят к выводу, что Python подходит не только для исследования и создания прототипа, но и для построения самих производственных систем. Полагаю, что компании большей частью будут выбирать этот путь, потому что использование учеными и технологами одного и того же набора программных средств, несомненно, выгодно для организации.

Недостатки Python

Python – великолепная среда для создания приложений для научных расчетов и большинства систем общего назначения, но тем не менее существуют задачи, которым Python не очень подходит.

Поскольку Python – интерпретируемый язык программирования, в общем случае написанный на нем код работает значительно медленнее, чем эквивалентный код на компилируемом языке типа Java или C++. Но поскольку *время программиста* обычно стоит гораздо дороже *времени процессора*, многих такой компромисс устраивает. Однако в приложениях, где задержка должна быть очень мала (например, в торговых системах с большим количеством транзакций), время, потраченное на программирование на низкоуровневом и не обеспечивающем максимальную продуктивность языке типа C++ во имя достижения максимальной производительности, будет потрачено не зря.

Python не идеальный язык для программирования многопоточных приложений с высокой степенью параллелизма, особенно при наличии многих потоков, активно использующих процессор. Проблема связана с наличием *глобальной блокировки интерпретатора* (GIL) – механизма, который не дает интерпретатору исполнять более одной команды байт-кода Python в каждый

момент времени. Объяснение технических причин существования GIL выходит за рамки этой книги, но на данный момент представляется, что GIL вряд ли скоро исчезнет. И хотя во многих приложениях обработки больших объектов данных, для того чтобы сократить срок работы, приходится организовывать кластер машин, встречаются все же ситуации, когда более желательна однопроцессная многопоточная система.

Я не хочу сказать, что Python вообще непригоден для исполнения многопоточного параллельного кода. Написанные на C или C++ расширения Python, пользующиеся платформенной многопоточностью, могут исполнять код параллельно и не ограничены механизмом GIL, при условии что им не нужно регулярно взаимодействовать с Python-объектами.

1.3. Необходимые библиотеки для Python

Для читателей, плохо знакомых с экосистемой Python и используемыми в книге библиотеками, я сделаю краткий обзор библиотек.

NumPy

NumPy, сокращение от «Numerical Python», – основной пакет для выполнения научных расчетов на Python. Большая часть этой книги базируется на NumPy и построенных поверх него библиотеках. В числе прочего он предоставляет:

- быстрый и эффективный объект многомерного массива *ndarray*;
- функции для выполнения вычислений над элементами одного массива или математических операций с несколькими массивами;
- средства для чтения и записи на диски наборов данных, представленных в виде массивов;
- операции линейной алгебры, преобразование Фурье и генератор случайных чисел;
- зрелый C API, позволяющий обращаться к структурам данных и вычислительным средствам NumPy из расширений Python и кода на C или C++.

Помимо ускорения работы с массивами, одной из основных целей NumPy в части анализа данных является организация контейнера для передачи данных между алгоритмами. Как средство хранения и манипуляции данными массивы NumPy куда эффективнее встроенных в Python структур данных. Кроме того, библиотеки, написанные на низкоуровневом языке типа C или Fortran, могут работать с данными, хранящимися в массиве NumPy, вообще без копирования в другое представление. Таким образом, многие средства вычислений, ориентированные на Python, либо используют массивы NumPy в качестве основной структуры данных, либо каким-то иным способом организуют интеграцию с NumPy.

pandas

Библиотека pandas предоставляет структуры данных и функции, призванные сделать работу со структурированными данными простой, быстрой и выразительной. С момента появления в 2010 году она способствовала превращению Python в мощную и продуктивную среду анализа данных. Основные объекты pandas, используемые в книге, – это `DataFrame` – двумерная таблица, в которой строки и столбцы имеют метки, и `Series` – объект одномерного массива с метками.

В библиотеке pandas сочетаются высокая производительность средств работы с массивами, присущая NumPy, и гибкие возможности манипулирования данными, свойственные электронным таблицам и реляционным базам данных (например, на основе SQL). Она предоставляет развитые средства индексирования, позволяющие без труда изменять форму наборов данных, формировать продольные и поперечные срезы, выполнять агрегирование и выбирать подмножества. Поскольку манипулирование данными, их подготовка и очистка играют огромную роль в анализе данных, в этой книге библиотека pandas будет одним из основных инструментов.

Если кому интересно, я начал разрабатывать pandas в начале 2008 года, когда работал в компании AQR Capital Management, занимающейся управлением инвестициями. Тогда я сформулировал специфический набор требований, которым не удовлетворял ни один отдельно взятый инструмент, имевшийся в моем распоряжении:

- структуры данных с помеченными осями, которые поддерживали бы автоматическое или явное выравнивание данных, – это предотвратило бы появление типичных ошибок при работе с невыровненными данными и данными из разных источников, которые по-разному индексированы;
- встроенная функциональность временных рядов;
- одни и те же структуры данных должны поддерживать как временные ряды, так и данные других видов;
- арифметические операции и упрощения должны сохранять метаданные;
- гибкая обработка отсутствующих данных;
- поддержка соединения и других реляционных операций, имеющихся в популярных базах данных (например, на основе SQL).

Я хотел, чтобы вся эта функциональность находилась в одном месте и предпочтительно была реализована на языке, хорошо приспособленном для разработки ПО общего назначения. Python выглядел хорошим кандидатом на эту роль, но в то время в нем не было подходящих встроенных структур данных и средств. Поскольку изначально библиотека pandas создавалась для решения финансовых задач и задач бизнес-аналитики, в ней особенно глубоко проработаны средства работы с временными рядами, ориентированные на обработку данных с временными метками, которые порождаются бизнес-процессами.

Пользователям языка статистических расчетов R название DataFrame покажется знакомым, потому что оно выбрано по аналогии с объектом `data.frame` в R. В отличие от Python, фреймы данных уже встроены в язык R и его стандартную библиотеку, поэтому многие средства, присутствующие в pandas, либо являются частью ядра R, либо предоставляются дополнительными пакетами.

Само название pandas образовано как от *panel data* (панельные данные), применяемого в эконометрике термина для обозначения многомерных структурированных наборов данных, так и от фразы *Python data analysis*.

matplotlib

Библиотека matplotlib – самый популярный в Python инструмент для создания графиков и других способов визуализации двумерных данных. Первоначально она была написана Джоном Д. Хантером (John D. Hunter), а теперь сопровождается большой группой разработчиков. Она отлично подходит для создания графиков, пригодных для публикации. Хотя программистам на Python доступны и другие библиотеки визуализации, matplotlib используется чаще всего и потому хорошо интегрирована с другими частями экосистемы. На мой взгляд, если вам нужно средство визуализации, то это самый безопасный выбор.

IPython и Jupyter

Проект IPython (http://ipython.org/) начал реализовывать в 2001 году Фернандо Перес (Fernando Perez) в качестве побочного, имеющего целью создать более удобный интерактивный интерпретатор Python. За прошедшие с тех пор 16 лет он стал одним из самых важных элементов современного инструментария Python. Хотя IPython сам по себе не содержит никаких средств вычислений или анализа данных, он изначально спроектирован, чтобы обеспечивать максимальную продуктивность интерактивных вычислений и разработки ПО. Он поощряет цикл *выполнить – исследовать* вместо привычного цикла *редактировать – компилировать – выполнить*, свойственного многим другим языкам программирования. Кроме того, он позволяет легко обращаться к оболочке и файловой системе операционной системы. Поскольку написание кода анализа данных часто подразумевает исследование методом проб и ошибок и опробование разных подходов, то благодаря IPython работу удается выполнить быстрее.

В 2014 году Фернандо и команда разработки IPython анонсировали проект Jupyter (https://jupyter.org/) – широкую инициативу проектирования языково-независимых средств интерактивных вычислений. Веб-блокнот IPython превратился в Jupyter-блокнот, который ныне поддерживает более 40 языков программирования. Систему IPython теперь можно использовать как *ядро* (языковой режим) для совместной работы Python и Jupyter.

Сам IPython стал компонентом более широкого проекта Jupyter с открытым исходным кодом, предоставляющего продуктивную среду для интерактивных исследовательских вычислений. В своем самом старом и простом режиме это улучшенная оболочка Python, имеющая целью ускорить написание, тестирование и отладку кода на Python. Систему IPython можно использовать также через Jupyter-блокнот, интерактивный веб-блокнот, поддерживающий десятки языков программирования. Оболочка IPython и Jupyter-блокноты особенно полезны для исследования и визуализации данных.

Кроме того, Jupyter-блокноты позволяют создавать контент на языках разметки Markdown и HTML, т. е. готовить комбинированные документы, содержащие код и текст. На других языках программирования также реализованы ядра для Jupyter, благодаря чему их можно использовать вместо Python.

Лично я при работе с Python использую в основном IPython – для выполнения, отладки и тестирования кода.

В сопроводительных материалах к книге (https://github.com/wesm/pydata-book) вы найдете Jupyter-блокноты, содержащие примеры кода к каждой главе.

SciPy

SciPy – собрание пакетов, предназначенных для решения различных стандартных вычислительных задач. Вот некоторые из них:

- `scipy.integrate` – подпрограммы численного интегрирования и решения дифференциальных уравнений;
- `scipy.linalg` – подпрограммы линейной алгебры и разложения матриц, дополняющие те, что включены в `numpy.linalg`;
- `scipy.optimize` – алгоритмы оптимизации функций (нахождения минимумов) и поиска корней;
- `scipy.signal` – средства обработки сигналов;
- `scipy.sparse` – алгоритмы работы с разреженными матрицами и решения разреженных систем линейных уравнений;
- `scipy.special` – обертка вокруг SPECFUN, написанной на Fortran-библиотеке, содержащей реализации многих стандартных математических функций, в том числе гамма-функции;
- `scipy.stats` – стандартные непрерывные и дискретные распределения вероятностей (функции плотности вероятности, формирования выборки, функции непрерывного распределения вероятности), различные статистические критерии и дополнительные описательные статистики.

Совместно NumPy и SciPy достаточно полно заменяют значительную часть системы MATLAB и многочисленные дополнения к ней.

scikit-learn

Проект scikit-learn (https://scikit-learn.org/stable/), запущенный в 2010 году, с самого начала стал основным инструментарием машинного обучения про-

граммистов на Python. Всего за семь лет к нему присоединилось 1500 разработчиков со всего мира. В нем имеются подмодули для следующих моделей:

- классификация: метод опорных векторов, метод ближайших соседей, случайные леса, логистическая регрессия и т. д.;
- регрессия: Lasso, гребневая регрессия и т. д.;
- кластеризация: метод *k* средних, спектральная кластеризация и т. д.;
- понижение размерности: метод главных компонент, отбор признаков, матричная факторизация и т. д.;
- выбор модели: поиск на сетке, перекрестный контроль, метрики;
- предобработка: выделение признаков, нормировка.

Наряду с pandas, statsmodels и IPython библиотека scikit-learn сыграла важнейшую роль для превращения Python в продуктивный язык программирования для науки о данных. Я не смогу включить в эту книгу полное руководство по scikit-learn, но все же предложу краткое введение в некоторые используемые в ней модели и объясню, как их использовать совместно с другими средствами.

statsmodels

Пакет статистического анализа statsmodels (http://www.statsmodels.org/stable/index.html) начал разрабатываться по инициативе профессора статистики из Стэнфордского университета Джонатана Тэйлора (Jonathan Taylor), который реализовал ряд моделей регрессионного анализа, популярных в языке программирования R. Скиппер Сиболд (Skipper Seabold) и Джозеф Перктольд (Josef Perktold) формально создали новый проект statsmodels в 2010 году, и с тех пор он набрал критическую массу заинтересованных пользователей и соразработчиков. Натаниэль Смит (Nathaniel Smith) разработал проект Patsy, который предоставляет средства для задания формул и моделей для statsmodels по образцу системы формул в R.

По сравнению со scikit-learn, пакет statsmodels содержит алгоритмы классической (прежде всего частотной) статистики и эконометрики. В него входят следующие подмодули:

- регрессионные модели: линейная регрессия, обобщенные линейные модели, робастные линейные модели, линейные модели со смешанными эффектами и т. д.;
- дисперсионный анализ (ANOVA);
- анализ временных рядов: AR, ARMA, ARIMA, VAR и другие модели;
- непараметрические методы: ядерная оценка плотности, ядерная регрессия;
- визуализация результатов статистического моделирования.

Пакет statsmodels ориентирован в большей степени на статистический вывод, он дает оценки неопределенности и *p*-значения параметров. Напротив, scikit-learn ориентирован главным образом на предсказание.

Как и для scikit-learn, я создам краткое введение в statsmodels и объясню, как им пользоваться в сочетании с NumPy и pandas.

1.4. Установка и настройка

Поскольку Python используется в самых разных приложениях, не существует единственно верной процедуры установки Python и необходимых дополнительных пакетов. У многих читателей, скорее всего, нет среды, подходящей для научных применений Python и проработки этой книги, поэтому я дам подробные инструкции для разных операционных систем. Рекомендую использовать бесплатный дистрибутив Anaconda. На момент написания книги Anaconda предлагается для версий Python 2.7 и 3.6, хотя в будущем это может измениться. В книге используется Python 3.6, и я настоятельно рекомендую работать именно с этой или более старшей версией.

Windows

В Windows начните со скачивания установщика Anaconda (https://www.anaconda.com/distribution/). Рекомендую следовать инструкциям, опубликованным на странице скачивания Anaconda, при этом надо понимать, что после выхода книги из печати они могли измениться.

По завершении установки проверьте, все ли сконфигурировано правильно. Откройте окно командной строки (это приложение называется *cmd.exe*), для чего щелкните правой кнопкой мыши по меню **Пуск** и выберите пункт **Командная строка**[1]. Попробуйте запустить интерпретатор Python, набрав команду **python**. Должно появиться сообщение, соответствующее установленной версии Anaconda:

```
C:\Users\wesm>python
Python 3.5.2 |Anaconda 4.1.1 (64-bit)| (default, Jul 5 2016, 11:41:13)
[MSC v.1900 64 bit (AMD64)] on win32
>>>
```

Чтобы выйти из оболочки, нажмите **Ctrl**+**Z** или введите команду **`exit()`** и щелкните по **Enter**.

Apple OS X

Скачайте установщик Anaconda для OS X Anaconda. Он должен называться как-то вроде *Anaconda3-4.1.0-MacOSX-x86_64.pkg*. Дважды щелкните по *pkg*-файлу для запуска установщика. Установщик автоматически добавит путь к исполняемому файлу Anaconda в ваш файл *.bash_profile*, полный путь к которому имеет вид */Users/$USER/.bash_profile*.

---

[1] В Windows 10 этого пункта в меню **Пуск** по умолчанию нет. Чтобы открыть окно командной строки, найдите программу cmd.exe с помощью средства поиска. – *Прим. перев.*

Для проверки работоспособности попробуйте запустить IPython из оболочки системы (для получения командной строки откройте приложение Terminal):

```
$ ipython
```

Чтобы выйти из оболочки, нажмите **Ctrl**+**D** или введите команду **`exit()`** и щелкните по **Enter**.

GNU/Linux

Детали установки в Linux варьируются в зависимости от дистрибутива. Я опишу процедуру, работающую в дистрибутивах Debian, Ubuntu, CentOS и Fedora. Установка в основных чертах производится так же, как для OS X, отличается только порядок установки Anaconda. Установщик представляет собой скрипт оболочки, запускаемый из терминала. В зависимости от разрядности системы нужно выбрать установщик типа x86 (32-разрядный) или x86_64 (64-разрядный). Имя соответствующего файла имеет вид *Anaconda3-4.1.0-Linux-x86_64.sh*. Для установки нужно выполнить такую команду:

```
$ bash Anaconda3-4.1.0-Linux-x86_64.sh
```

В некоторых дистрибутивах Linux менеджеры пакетов, например apt, располагают всеми необходимыми Python-пакетами. Ниже описывается установка с помощью Anaconda, поскольку она одинакова во всех дистрибутивах и упрощает обновление пакетов до последней версии.

После подтверждения согласия с лицензией вам будет предложено указать место установки файлов Anaconda. Я рекомендую устанавливать их в свой домашний каталог, например */home/$USER/anaconda* (вместо `$USER` подставьте свое имя пользователя).

Установщик Anaconda может спросить, хотите ли вы добавить каталог *bin/* в начало переменной `$PATH`. Если после установки возникнут проблемы, это можно сделать вручную, модифицировав файл *.bashrc* (или *.zshrc*, если вы пользуетесь оболочкой zsh) примерно таким образом:

```
export PATH=/home/$USER/anaconda/bin:$PATH
```

Затем запустите новый процесс терминала или повторно выполните файл *.bashrc* командой `source ~/.bashrc`.

Установка или обновление Python-пакетов

Рано или поздно вам могут понадобиться дополнительные Python-пакеты, не включенные в дистрибутив Anaconda. В общем случае они устанавливаются такой командой:

```
conda install package_name
```

Если это не работает, то, возможно, удастся установить пакет с помощью менеджера пакетов pip:

```
pip install package_name
```

Для обновления пакетов служит команда `conda update`:

```
conda update package_name
```

pip также поддерживает обновление, нужно только задать флаг `--upgrade`:

```
pip install --upgrade package_name
```

В этой книге вам не раз представится возможность попробовать эти команды в деле.

Если для установки пакетов вы используете и conda, и pip, то не следует пытаться обновлять пакеты conda с помощью pip, поскольку из-за этого может повредиться среда. При работе с Anaconda или Miniconda всегда рекомендуется сначала попробовать обновить пакет командой conda.

Python 2 и Python 3

Первая версия в линейке интерпретаторов Python 3.x была выпущена в конце 2008 года. В нее включены изменения, несовместимые с ранее написанным кодом для версий Python 2.x. Поскольку с момента выхода первой версии Python в 1991 году прошло уже 17 лет, создание несовместимой версии Python 3 рассматривалось как благо – принимая во внимание все уроки прошлого.

В 2012 году разработчики и пользователи приложений Python для научных расчетов и анализа данных работали в основном с версиями 2.x, поскольку многие пакеты еще не были переведены на Python 3. Поэтому в первом издании книги использовалась версия Python 2.7. Но теперь пользователи могут выбирать между Python 2.x и 3.x, так как большинство библиотек поддерживают обе ветви.

Однако поддержка Python 2.x прекратится в 2020 году (это относится и к критическим исправлениям безопасности), так что начинать новые проекты на Python 2.7 не стоит. Поэтому в книге используется Python 3.6 – стабильная, хорошо поддерживаемая и широко распространенная версия. Все уже начали называть Python 2.x унаследованным Python, а Python 3.x – просто Python. Призываю и вас последовать этому примеру.

Я взял за основу версию Python 3.6. У вас может быть более свежая версия, но все примеры кода должны быть совместимы с ней. В Python 2.7 некоторые примеры могут работать иначе или не работать вовсе.

Интегрированные среды разработки (IDE)

Когда меня спрашивают о том, какой средой разработки я пользуюсь, я почти всегда отвечаю: «IPython плюс текстовый редактор». Обычно я пишу программу и итеративно тестирую и отлаживаю ее по частям в IPython или Jupyter-блокнотах. Полезно также иметь возможность интерактивно экс-

периментировать с данными и визуально проверять, получается ли в результате определенных манипуляций ожидаемый результат. Библиотеки pandas и NumPy спроектированы с учетом простоты использования в оболочке.

Однако некоторые пользователи предпочитают разрабатывать программы в полноценной IDE, а не в сравнительно примитивном текстовом редакторе типа Emacs или Vim. Вот некоторые доступные варианты:

- PyDev (бесплатная) – IDE, построенная на платформе Eclipse;
- PyCharm от компании JetBrains (на основе подписки для коммерческих компаний, бесплатна для разработчиков ПО с открытым исходным кодом);
- Python Tools для Visual Studio (для работающих в Windows);
- Spyder (бесплатная) – IDE, которая в настоящий момент поставляется в составе Anaconda;
- Komodo IDE (коммерческая).

1.5. Сообщество и конференции

Помимо поиска в интернете можно пользоваться полезными рассылками, посвященными применению Python в научных расчетах и для обработки данных. Их участники быстро отвечают на вопросы. Вот некоторые из таких ресурсов:

- pydata: группа Google по вопросам, относящимся к использованию Python для анализа данных и pandas;
- pystatsmodels: вопросы, касающиеся statsmodels и pandas;
- numpy-discussion: вопросы, касающиеся NumPy;
- рассылка по scikit-learn (scikit-learn@python.org) и машинному обучению на Python вообще;
- scipy-user: общие вопросы использования SciPy и Python для научных расчетов.

Я сознательно не публикую URL-адреса, потому что они часто меняются. Поиск в интернете вам в помощь.

Ежегодно в разных странах проходят конференции для программистов на Python. Если вы захотите пообщаться с другими программистами, которые разделяют ваши интересы, то имеет смысл посетить какое-нибудь мероприятие (если есть такая возможность). Многие конференции оказывают финансовую поддержку тем, кто не может позволить себе вступительный взнос или транспортные расходы. Приведу неполный перечень конференций:

- PyCon and EuroPython: две самые крупные, проходящие соответственно в Северной Америке и в Европе;
- SciPy и EuroSciPy: конференции, ориентированные на научные применения Python, проходящие соответственно в Северной Америке и в Европе;

- PyData: мировая серия региональных конференций, посвященных науке о данных и анализу данных;
- международные и региональные конференции PyCon (полный список см. на сайте http://pycon.org).

1.6. Структура книги

Если вы раньше никогда не программировали на Python, то имеет смысл потратить время на знакомство с главами 2 и 3, где я поместил очень краткое руководство по языковым средствам Python, а также по оболочке IPython и Jupyter-блокнотам. Эти знания необходимы для чтения книги. Если у вас уже есть опыт работы с Python, то можете вообще пропустить эти главы или пролистать их по диагонали.

Далее дается краткое введение в основные возможности NumPy, а более подробное изложение имеется в приложении A. Затем мы познакомимся с pandas и посвятим оставшуюся часть книги анализу данных с применением pandas, NumPy и matplotlib (для визуализации). Я старался построить изложение по возможности поступательно, хотя иногда главы немного пересекаются, и есть несколько случаев, когда используются еще не описанные концепции.

У разных читателей могут быть разные цели, но, вообще говоря, можно предложить следующую классификацию задач.

- Взаимодействие с внешним миром – чтение и запись в файлы и хранилища данных различных форматов.
- Подготовка – очистка, переформатирование, комбинирование, нормализация, изменение формы, получение продольных и поперечных срезов, трансформация данных для анализа.
- Преобразование – применение математических и статистических операций к группам наборов данных для получения новых наборов (например, агрегирование большой таблицы по некоторым переменным).
- Моделирование и вычисления – связывание данных со статистическими моделями, алгоритмами машинного обучения и иными вычислительными средствами.
- Презентация – создание интерактивных или статических графических визуализаций или текстовых сводных отчетов.

Примеры кода

Примеры кода в большинстве случаев показаны так, как выглядят в оболочке IPython или Jupyter-блокнотах: ввод и вывод.

```
In [5]: КОД
Out[5]: РЕЗУЛЬТАТ
```

Это означает, что вы должны ввести код в блоке `In` в своей рабочей среде и выполнить его, нажав клавишу **Enter** (или **Shift-Enter** в Jupyter). Результат должен быть таким, как показано в блоке `Out`.

Данные для примеров

Наборы данных для примеров из каждой главы находятся в репозитории на сайте GitHub: https://github.com/wesm/pydata-book. Вы можете получить их либо с помощью командной утилиты системы управления версиями git, либо скачав zip-файл репозитория с сайта. Если возникнут проблемы, заходите на мой сайт (https://wesmckinney.com/), где выложены актуальные инструкции по получению материалов к книге.

Я стремился сделать так, чтобы в репозиторий попало все необходимое для воспроизведения примеров, но мог где-то ошибиться или что-то пропустить. В таком случае пишите мне на адрес book@wesmckinney.com. Самый лучший способ сообщить об ошибках, найденных в книге, – описать их на странице опечаток на сайте издательства O'Reilly (https://www.oreilly.com/catalog/errata.csp?isbn=0636920050896).

Соглашения об импорте

В сообществе Python принят ряд соглашений об именовании наиболее употребительных модулей:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
import statsmodels as sm
```

Это означает, что `np.arange` – ссылка на функцию `arange` в пакете NumPy. Так делается, потому что импорт всех имен из большого пакета, каким является NumPy (`from numpy import *`), считается среди разработчиков на Python дурным тоном.

Жаргон

Я употребляю некоторые термины, встречающиеся как в программировании, так и в науке о данных, с которыми вы, возможно, незнакомы. Поэтому приведу краткие определения.

- *Переформатирование* (Munge/Munging/Wrangling) – процесс приведения неструктурированных и (или) замусоренных данных к структурированной или чистой форме. Слово вошло в лексикон многих современных специалистов по анализу данных.
- *Псевдокод* – описание алгоритма или процесса в форме, напоминающей код, хотя фактически это не есть корректный исходный код на каком-то языке программирования.
- *Синтаксический сахар* – синтаксическая конструкция, которая не добавляет новую функциональность, а лишь вносит дополнительное удобство или позволяет сделать код короче.