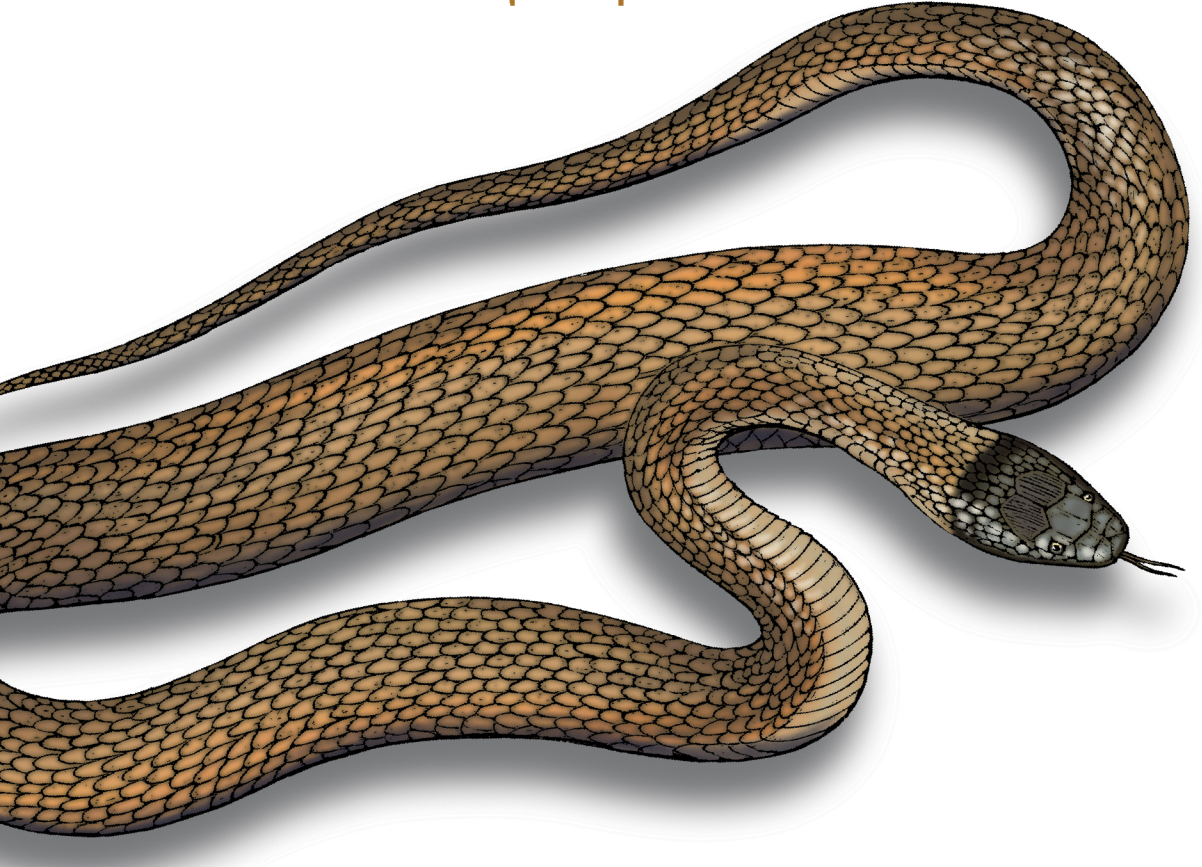
O'REILLY®

Python для финансистов

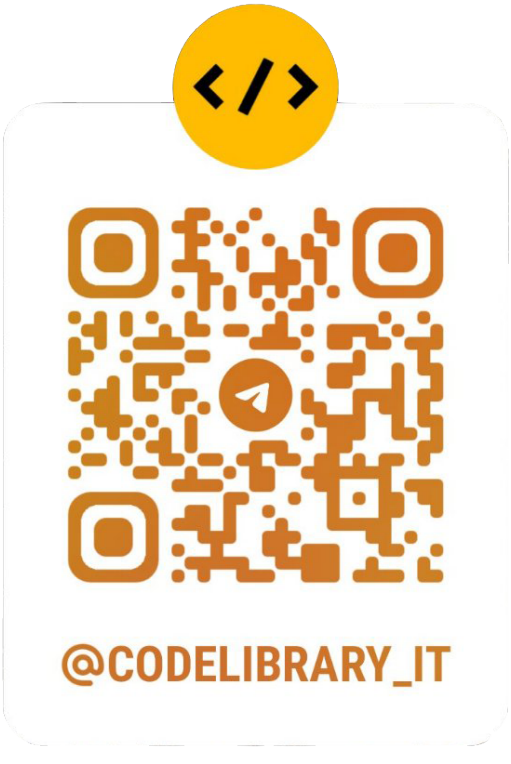
Базовые концепции



Ив Хилпиш

Financial Theory with Python

A Gentle Introduction



Yves Hilpisch

Beijing • Boston • Farnham • Sebastopol • Tokyo

O'REILLY®

Python для финансистов

Базовые концепции

Ив Хилпиш

ПИТЕР®

Санкт-Петербург • Москва • Минск

2023

ББК 65.010.65:32.973.2-018.1
УДК 336:004.43
Х45

Хилпиш Ив

Х45 Python для финансистов. — СПб.: Питер, 2023. — 208 с.: ил. — (Серия «Бестселлеры O'Reilly»).

ISBN 978-5-4461-2250-9

Программирование, математика и финансы неразрывно связаны между собой. Ив Хилпиш, автор бестселлера «Python для финансовых расчетов», объясняет базовые концепции и дает в ваши руки все необходимые инструменты для работы в мире финансовой инженерии.

В этой книге вы:

- изучите основы программирования на Python и познакомитесь с теорией финансов через математику;
- узнаете о моделировании данных и использовании Python в финансовой инженерии;
- научитесь статическому и динамическому моделированию финансовых задач: ценообразованию, принятию решений и распределению активов;
- получите общее представление о необходимых библиотеках Python: NumPy, SciPy, Matplotlib и SymPy.

16+ (В соответствии с Федеральным законом от 29 декабря 2010 г. № 436-ФЗ.)

ББК 65.010.65:32.973.2-018.1
УДК 336:004.43

Информация, содержащаяся в данной книге, получена из источников, рассматриваемых издательством как надежные. Тем не менее, имея в виду возможные человеческие или технические ошибки, издательство не может гарантировать абсолютную точность и полноту приводимых сведений и не несет ответственности за возможные ошибки, связанные с использованием книги.

В книге возможны упоминания организаций, деятельность которых запрещена на территории Российской Федерации, таких как Meta Platforms Inc., Facebook, Instagram и др.

Издательство не несет ответственности за доступность материалов, ссылки на которые вы можете найти в этой книге. На момент подготовки книги к изданию все ссылки на интернет-ресурсы были действующими.

ISBN 978-1098104351 англ.

ISBN 978-5-4461-2250-9

Краткое содержание

Оглавление

Введение

> Python быстро становится языком науки о данных, машинного обучения и обработки естественного языка. Его использование постоянно открывает источники инноваций. У Python есть многочисленное сообщество разработчиков приложений с открытым исходным кодом, что позволяет быстро внедрять и адаптировать передовые технологии[1].
>
> *Киндман и Тейлор (2021)*

Почему именно эта книга?

Эта книга обучает финансам и языку программирования Python (http://python.org/) с нуля. Сейчас финансы и программирование — тесно переплетенные дисциплины, а Python — один из наиболее часто используемых в финансовой отрасли языков программирования. Здесь комплексно изложены основы математики, финансов и программирования в понятном для обычных людей виде. Долгое время теория финансов и финансовая инженерия были отдельными дисциплинами. Однако то, что программирование (например, на Python и C++) стало неотъемлемой частью магистратуры по финансовой инженерии и подобных университетских программ, доказывает, насколько важным стал этот навык в данной области.

---

[1] *Kindman A., Taylor T.* Why We Rewrote Our USD30 Billion Asset Management Platform in Python. March 29, 2021. https://oreil.ly/GghS6.

Платформы для онлайн-торговли, программное обеспечение с открытым исходным кодом и финансовая информация, находящаяся в свободном доступе, значительно понизили, а то и полностью ликвидировали порог входа на мировые финансовые рынки. Теперь всего за несколько часов обычный человек с ограниченным бюджетом может начать заниматься алгоритмической торговлей. Студенты и преподаватели финансовых дисциплин, немного знающие программирование, могут применить последние инновации в области машинного и глубокого обучения к финансовым данным, пользуясь только ноутбуками, которые у них всегда с собой. Что касается технических средств, то облачные провайдеры с почасовой оплатой и практически неограниченной масштабируемостью за очень небольшую цену предоставляют возможность быстро и качественно вычислять и обрабатывать данные. Сегодня профессиональное финансовое образование лишь частично соответствует этим технологическим тенденциям.

Тем не менее все еще довольно часто *основы математики*, *теория финансов* и *основы программирования* преподаются независимо друг от друга и только в самом конце обучения — в комплексе с *финансовой инженерией*. В этой книге используется другой подход: финансовые концепции и методы программирования представлены во взаимосвязи с математическими понятиями (например, из линейной алгебры и теории вероятностей). Таким образом, абстрактные математические понятия объясняются с двух различных точек зрения — финансов и программирования. Вдобавок такой подход позволяет получить новый полезный опыт, поскольку и математические, и финансовые понятия могут быть переведены непосредственно в исполняемый код и исследованы в интерактивном режиме.

Несколько человек, прочитавших одну из моих предыдущих книг, «Python для финансовых расчетов»[1], справедливо отметили, что она не подходит тем, кто только начинает знакомство с теорией финансов и программированием на Python. Действительно, предполагается, что читатель той книги имеет хотя бы небольшой опыт в данных сферах. Книга «Python для финансистов» восполняет этот пробел, поскольку фокусируется на основах и тем самым естественным образом подготавливает к прочтению «Python для финансовых расчетов», что в дальнейшем позволит развиваться и совершенствовать навыки работы с Python применительно к финансовым расчетам. Более подробно об этом рассказано в последней главе.

[1] *Хилпиш И.* Python для финансовых расчетов. — 2021.

Целевая аудитория

Об использовании Python в финансовой сфере я написал несколько книг, а моя компания, The Python Quants, предлагает соответствующее онлайн-обучение. И книги, и курсы предполагают, что читатель или слушатель уже обладает определенными знаниями в области финансов и программирования на Python или аналогичном ему языке.

Эта же книга знакомит читателя с данными темами с нуля, и ему нужны лишь базовые знания в области математики, в частности математического анализа, линейной алгебры и теории вероятностей. Материал содержит практически полную информацию обо всех описанных в нем математических понятиях. Тем не менее может пригодиться вводный учебник математики, например учебник Пембертона и Рау[1, 2].

Книга предназначена для студентов, ученых и специалистов, которые хотят получить знания по теории финансов, финансовому моделированию и использованию Python в финансовой инженерии. Она также может послужить систематически выстроенной основой для создания более сложных книг и курсов по этим темам.

Даже если читатель не собирается переходить к более сложным темам финансовой инженерии, вычислительных финансов, алгоритмической торговли или управления активами, знания по Python и финансам, которые он почерпнет из этой книги, можно использовать при выполнении стандартных финансовых задач, например, при составлении инвестиционных портфелей в соответствии с современной портфельной теорией (modern portfolio theory, MPT). Книга также рассказывает об оценке опционов и других деривативов с помощью стандартных методов, таких как портфельная репликация или риск-нейтральный подход к ценообразованию.

Эта книга подойдет руководителям, которые хотят узнать о применении Python в области финансов. В то же время она будет полезна тем, кто уже владеет Python или другим языком программирования и хочет узнать, как их можно использовать в данной сфере.

---

[1] *Pemberton M., Rau N.* Mathematics for Economists: An Introductory Textbook. 4th ed. — Manchester University Press, 2016.

[2] Русскоязычным читателям рекомендуем главную книгу по финансовой математике в России: *Ширяев А. Н.* Основы стохастической финансовой математики. В двух томах. Том 1. Факты. Модели. Том 2. Теория. — М.: МЦНМО, 2016. — *Примеч. науч. ред.*

Краткое описание книги

Книга состоит из следующих глав.

- **Глава 1. Финансы и Python.** Эта глава формирует основу всей книги. В ней кратко излагается история финансов, освещается подход к Python как к инструменту для финансовых расчетов, показывается, как работать с базовой инфраструктурой Python, на примере кода, представленного в интерактивных блокнотах Jupyter.
- **Глава 2. Экономика с двумя состояниями.** Здесь рассматривается наиболее простая модель экономики, в которой возможен финансовый анализ в условиях неопределенности, когда есть только две релевантные даты и два неопределенных состояния в будущем. Иногда ее называют статической экономикой с двумя состояниями. Несмотря на свою простоту, она позволяет представить такие базовые финансовые понятия, как чистая приведенная стоимость, ожидаемая доходность, волатильность, условные требования, репликация опционов, арбитражное ценообразование, мартингальная мера, полнота рынка, риск-нейтральный подход к ценообразованию и портфели Марковица.
- **Глава 3. Экономика с тремя состояниями.** В этой главе в модель добавляется третье неопределенное состояние в будущем и анализируется статическая экономика с тремя состояниями, что позволяет рассмотреть такие понятия, как неполнота рынка, неопределенность мартингальных мер, суперрепликация условных требований и аппроксимирующая репликация условных требований. Вдобавок здесь представлена модель ценообразования капитальных активов (Capital Asset Pricing Model, CAPM) в качестве подхода к равновесному ценообразованию финансовых активов.
- **Глава 4. Оптимальность и равновесие.** Основной темой здесь являются экономические агенты с их индивидуальными проблемами принятия решений. Анализ здесь опирается на доминирующую парадигму финансов для принятия решений в условиях неопределенности — *максимизацию ожидаемой полезности*. Через так называемого репрезентативного агента вводятся понятия равновесия, демонстрируется связь между оптимальностью и равновесием, с одной стороны, и мартингальными мерами и риск-нейтральным подходом к ценообразованию — с другой. Ко всему прочему, использование концепции репрезентативного агента представляет собой

один из способов преодоления трудностей, возникающих в экономиках с неполными рынками.

- **Глава 5. Статическая экономика.** Обобщает рассмотренные понятия и сводит полученные результаты к конечному (возможно, большому) числу неопределенных будущих состояний. Для анализа такой *обобщенной статической экономики* требуется немного больше математических формул.
- **Глава 6. Динамическая экономика.** Основываясь на анализе обобщенной статической экономики, здесь в финансовое моделирование добавляется динамика, что позволяет рассмотреть два частных случая динамической экономики в дискретном времени. Основная идея заключается в том, что неопределенность относительно будущих состояний экономики в целом разрешается постепенно с течением времени. Это можно смоделировать с помощью стохастических процессов, например биномиального процесса, который можно представить в виде биномиального дерева.
- **Глава 7. Что дальше?** В заключительной главе приводится множество дополнительных материалов для изучения из области математики, теории финансов и программирования на Python, также читателю даются рекомендации о дальнейших действиях.

Условные обозначения

В книге используются следующие условные обозначения.

Курсив

Служит для выделения новых понятий.

`Интерфейс`

Применяется для выделения URL-адресов, электронных адресов.

`Моноширинный шрифт`

Служит для выделения в тексте элементов программ, таких как переменные или названия функций, баз данных, типов данных, переменных окружения, операторов и ключевых слов, имен и расширений файлов. Кроме того, используется для оформления листингов.

Этот рисунок обозначает обычное примечание.

А этот — предупреждение или предостережение.

Этим рисунком обозначена важная информация.

Примеры кода, использованные в книге

Все вспомогательные материалы — примеры кода, упражнения и т. д. — можно скачать здесь: https://finpy.pqp.io.

По техническим вопросам или проблемам, связанным с примерами кода, можно обращаться по адресу bookquestions@oreilly.com.

В общем случае все примеры кода из книги вы можете использовать в своих программах и в документации. Вам не нужно обращаться в издательство за разрешением, если вы не собираетесь воспроизводить существенные части программного кода. Если вы разрабатываете программу и используете в ней несколько фрагментов кода из книги, вам не нужно обращаться за разрешением. Но для продажи или распространения примеров из книги вам потребуется разрешение от издательства O'Reilly. Вы можете отвечать на вопросы, цитируя данную книгу или примеры из нее, но для включения существенных объемов программного кода из книги в документацию вашего продукта потребуется разрешение.

Мы рекомендуем, но не требуем добавлять ссылку на первоисточник при цитировании. Под ссылкой на первоисточник мы подразумеваем указание авторов, издательства и ISBN.

За получением разрешения на использование значительных объемов программного кода из книги обращайтесь по адресу permissions@oreilly.com.

Благодарности

В книге приведены ценные отзывы участников наших сертификационных программ по Python для финансовых расчетов.

Благодарю научных редакторов за их полезные комментарии. Они помогли внести многочисленные улучшения.

Я также признателен всей команде O'Reilly за помощь и поддержку.

Эта книга посвящена моей жене Сандре. Ты — любовь всей моей жизни.

От издательства

Ваши замечания, предложения, вопросы отправляйте по адресу comp@piter.com (издательство «Питер», компьютерная редакция).

Мы будем рады узнать ваше мнение!

На веб-сайте издательства www.piter.com вы найдете подробную информацию о наших книгах.

ГЛАВА 1

Финансы и Python

> История теории финансов — интересный пример взаимосвязи между абстрактным теоретизированием и практическим применением.
>
> *Франк Мильне (1995)*

> Благодаря современным технологиям в последние годы хедж-фонды поглотили десятки миллиардов долларов инвестиций. Эти же технологии приносят прибыль людям, принимающим финансовые решения в таких организациях.
>
> *Лоуренс Флетчер (2020)*

Цель главы 1 состоит в том, чтобы заложить основы для последующих глав, поэтому в ней дается краткий обзор освещаемых тем. Она начинается с раздела «Краткая история финансов», в котором в общих чертах рассказывается об истории развития и современном состоянии финансов. В разделе «Главные тенденции в области финансов» рассматриваются основные тенденции, касающиеся математики, технологий, данных, искусственного интеллекта и определяющие развитие финансов в долгосрочной перспективе. В связи с этим раздел «Четырехъязычная сфера» говорит о том, что финансы в наше время — это дисциплина, состоящая из четырех тесно связанных между собой типов языков: естественного (в большей мере английского), языка финансов, математического и языка программирования. Далее объясняется, как устроена книга (раздел «Структура книги»). А в последнем разделе, «Вводная информа-

ция о Python», рассказывается об установке окружения Python на компьютер и об альтернативном варианте работы с ним: весь код может использоваться и исполняться через обычный браузер на Quant Platform.

Краткая история финансов

Чтобы лучше понять текущее состояние финансов и финансовой отрасли, нужно взглянуть, как они развивались. В 2006 году Рубинштейн условно разделил историю финансов как науки на три периода.

- *Древний период (до 1950 года)*. Период, характеризующийся в основном неформальными рассуждениями, эмпирическими правилами и опытом субъектов финансового рынка.
- *Классический период (1950–1980 годы)*. В это время произошло внедрение в финансы более строгих обоснований и математики. Были разработаны финансовые модели, например модель ценообразования опционов Блэка — Шоулза (1973), а также заложены некоторые основы, например риск-нейтральный подход к ценообразованию Харрисона и Крепса (1979).
- *Современный период (1980–2000 годы)*. Это период прогресса в отдельных областях финансов, например в финансовой инженерии, и решения проблем, связанных с важными эмпирическими явлениями на финансовых рынках, такими как стохастическая модель процентных ставок (Кокс, Ингерсолл и Росс, 1985) или модель стохастической волатильности (Хестон, 1993).

Сейчас, спустя 15 лет после публикации книги Рубинштейна, можно выделить четвертый и пятый периоды, которые обеспечили появление и нынешнюю популярность Python в финансах.

- *Вычислительный период (2000–2020 годы)*. В это время произошел переход от теоретической направленности финансов к вычислительной, что обусловлено развитием программно-технических средств. Эту смену парадигмы хорошо иллюстрирует численный алгоритм для оценки американских опционов методом Монте-Карло, представленный в работе Лонгстаффа и Шварца в 2001 году. Алгоритм требует больших вычислительных ресурсов, поскольку для оценки стоимости всего одного опциона нужны сотни тысяч численных моделирований и анализ множества обычных регрессий методом наименьших квадратов (см.: Хилпиш, 2018).

- *Период использования искусственного интеллекта (после 2020 года)*. Прогресс в области искусственного интеллекта (ИИ) и связанные с ним истории успеха подстегнули интерес к использованию возможностей ИИ в финансовой сфере. Несмотря на существование эффективных приложений в данной области (см.: Хилпиш, 2020), можно предположить, что с 2020 года парадигма планомерно смещается в сторону *финансовых систем, основанных на ИИ*. Таким образом, мы становимся свидетелями перехода от простых, в основном линейных, финансовых моделей к сложным моделям и использованию алгоритмов ИИ (глубоких нейронных сетей или обучения с подкреплением) для фиксации, описания и объяснения финансовых явлений.

Главные тенденции в области финансов

Как и многие другие предметы и отрасли, наука о финансах со временем стала формализованной научной дисциплиной. Этому прогрессу посодействовали растущая популярность формальной математики, развитие технологий, доступность данных и усовершенствованные алгоритмы, например ИИ-алгоритмы. В целом в эволюции финансов можно выделить четыре основных направления.

- *Математика*. Начиная с 1950-х годов наука о финансах становится все более формализованной дисциплиной, в которой систематически используются знания из различных областей математики, например линейной алгебры и стохастического исчисления. Появление портфельной теории Марковица (1952) стало прорывом в количественном анализе и точкой перехода от древнего периода с его неформальными рассуждениями к классическому периоду.
- *Технологии*. Благодаря рабочим станциям, серверам и компьютерам, которые начали широко использоваться в конце 1980-х — начале 1990-х годов, технологии стали проникать и в финансовую отрасль. И если сначала техника имела довольно небольшую вычислительную мощность, то сейчас даже самые сложные финансовые задачи можно решить с помощью специальной программы, не прибегая к специальным моделям и методам, характерным для классического и современного периодов. Кредо стало таким: «Обновляйте оборудование и используйте современное программное обеспечение вместе с правильными вычислительными методами». При этом большинство современных компьютеров, предназначенных для

использования массовым потребителем, уже обладает мощностью, необходимой для высокопроизводительной работы (например, для параллельной обработки), что значительно упрощает изучение финансовой инженерии и взаимодействие с финансовыми системами, основанными на ИИ.

- *Данные*. Если в древний и классический периоды теоретики и практики черпали информацию, касающуюся финансов, в основном из специализированных печатных изданий (здесь можно вспомнить Wall Street Journal или Financial Times), то сегодня возросла доступность массива финансовых данных в электронном виде. Массивы высокочастотных внутридневных данных стали нормой и заменили цены на момент закрытия биржи в качестве основной базы для эмпирических исследований. Каждый торговый день одна акция может генерировать массивы внутридневных данных, содержащие более 100 000 значений, что приблизительно эквивалентно ценам на момент закрытия биржи для той же акции за 400 лет (250 торговых дней в году умножить на 400 лет). Вдобавок с недавнего времени наблюдается распространение открытых массивов данных, что также значительно снижает порог входа в финансовую инженерию, алгоритмическую торговлю или финансовую эконометрику.
- *Искусственный интеллект*. Увеличение объема финансовых данных (возникновение «больших финансовых данных») в наши дни делает применение алгоритмов искусственного интеллекта: алгоритмов машинного обучения, глубокого обучения или обучения с подкреплением (см.: Хилпиш, 2020) — не только возможным, но и во многих случаях необходимым подходом к работе с финансами. Традиционные статистические методы из финансовой эконометрики уже не подходят для решения большинства современных проблем, возникающих на финансовых рынках. В условиях нелинейной, многомерной, постоянно меняющейся финансовой среды алгоритмы на основе искусственного интеллекта нередко становятся единственным средством обнаружения релевантных взаимосвязей и закономерностей, получения важных данных и использования улучшенных возможностей прогнозирования.

В книге дается основная информация о финансовой математике и современных технологиях, используемых для реализации формальных финансовых моделей. С ее помощью можно приобрести навыки работы с массивами финансовых данных, наиболее часто встречающимися в финансовой сфере. Книга предназначена для того, чтобы подготовить читателя к более сложным темам финансовой инженерии и применения ИИ к финансовым расчетам.

Python и финансы

Все чаще основой финансовых расчетов становятся алгоритмы, требующие больших вычислительных ресурсов, постоянно растущая доступность данных и искусственный интеллект. Python оказался подходящим языком программирования и технологической платформой для удовлетворения потребностей и решения задач, обусловленных тенденциями развития финансовой сферы.

Четырехъязычная сфера

На фоне перечисленных выше тенденций финансы стали сферой применения четырех языков.

- *Естественный язык*. Сегодня основным языком публикуемых финансовых исследований, книг, статей и новостей является английский.
- *Финансовый язык*. Как и в любой другой области, в финансах есть свои технические термины, понятия и выражения, описывающие определенные финансовые явления или понятия.
- *Математический язык*. Математика — самый удобный языковой инструмент для формализации финансовых понятий и концепций.
- *Язык программирования*. Как отмечается в эпиграфе к введению, Python — лучший из существующих языков программирования для работы в финансовой индустрии.

Таким образом, чтобы отлично разбираться в науке о финансах, и теоретик, и практик должны свободно владеть всеми четырьмя языками. Конечно, нельзя утверждать, что в финансовой области нет других языков, кроме английского или, например, Python. Здесь подразумевается, что при ограниченном количестве времени, выделенном на изучение финансов, наряду с финансовой математикой лучше сосредоточить свои силы на Python, а не на другом языке программирования.

Структура книги

Как в издании совмещается использование всех этих четырех языков? С естественным языком все понятно: можно либо читать книгу в оригинале, либо пользоваться переводом на родной язык. Осталось решить вопрос с остальными.

К примеру, книга не может подробно объяснить каждый отдельный раздел математики, необходимый в финансах, или представить все концепции программирования (Python), использующиеся в финансовой инженерии. Тем не менее в ней была сделана попытка связать понятия из финансов, математики и программирования между собой.

Начиная с главы 2, информация в книге будет подаваться через введение финансового понятия (или концепции) вместе с его математическим представлением и реализацией на Python. К примеру, в таблице из главы 3 перечислены финансовые понятия, о которых будет идти речь, соответствующие им основные математические элементы и основная структура данных Python, используемая для реализации финансовой математики.

Финансы	Математика	Python
Неопределенность	Вероятностное пространство	`ndarray`
Финансовые активы	Векторы, матрицы	`ndarray`
Достижимые условные требования	Линейная оболочка векторов, базис векторного пространства	`ndarray`

Чтобы проиллюстрировать общую структуру книги, далее приведу пример описания финансового понятия, который будет рассматриваться подробнее в следующих главах.

Возьмем из таблицы концепцию *неопределенности в финансах*. Под неопределенностью понимается невозможность заранее узнать состояние модельной экономики в будущем. Будущее состояние экономики важно, например, для определения выплаты по европейскому колл-опциону (то есть по опциону на покупку). В условиях дискретности мы имеем дело с конечным числом таких состояний, например двумя, тремя или более. В самом простом случае, когда рассматриваются только два будущих состояния, выплата по европейскому колл-опциону математически представляется как *случайная величина*, которая, в свою очередь, формально может быть представлена вектором $\mathbf{v}$, являющимся элементом *векторного пространства* $\mathbb{R}^2$. Векторное пространство — это набор объектов (векторов), для которых определены сложение и скалярное умножение. Формально вектор $\mathbf{v}$ можно представить следующим образом:

$$\mathbf{v} = \begin{pmatrix} v^u \\ v^d \end{pmatrix} \in \mathbb{R}^2_{\geq 0},$$

Здесь предполагается, что оба векторных элемента являются неотрицательными вещественными (действительными) числами $v^u, v^d \in \mathbb{R}_{\geq 0}$. Другими словами, если неопределенная, зависящая от состояния экономики цена акций, на которые выписан европейский колл-опцион, определяется как:

$$S = \begin{pmatrix} 20 \\ 5 \end{pmatrix} \in \mathbb{R}^2_{\geq 0},$$

а цена исполнения опциона $K = 15$, то выплата C по европейскому колл-опциону будет равна:

$$C = \max(S - K, 0) = \begin{pmatrix} \max(20 - 15, 0) \\ \max(5 - 15, 0) \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix} \in \mathbb{R}^2_{\geq 0}.$$

Вот так понятия *неопределенной цены акции* и *зависящей от состояния экономики выплаты по европейскому опциону* могут быть математически смоделированы в виде вектора. Дисциплина, занимающаяся векторами и векторными пространствами в математике, называется *линейной алгеброй*.

Как же все это перевести на Python? Во-первых, вещественные числа в нем представлены как *числа с плавающей запятой* или объекты `float`:

```
In [1]: vu = 1.5 ❶

In [2]: vd = 3.75 ❷

In [3]: type(vu) ❸
Out[3]: float

In [4]: vu + vd ❹
Out[4]: 5.25
```

❶ Определение переменной `vu` и ее значения `1.5`.

❷ Определение переменной `vd` и ее значения `3.75`.

❸ Поиск и вывод типа объекта `vu` (`float`).

❹ Сложение значений `vu` и `vd`.

Во-вторых, наборы объектов одного типа в программировании обычно называются *массивами*. Поддержку таких структур данных в Python обеспечивает библиотека `NumPy` (http://numpy.org/). Основной структурой данных в ней

является ndarray — аббревиатура для *n*-мерного массива (*n*-dimensional array). NumPy легко моделирует векторы с вещественными значениями:

```
In [5]: import numpy as np ❶

In [6]: v = np.array((vu, vd)) ❷

In [7]: v ❸
Out[7]: array([1.5 , 3.75])

In [8]: v.dtype ❹
Out[8]: dtype('float64')

In [9]: v.shape ❺
Out[9]: (2,)

In [10]: v + v ❻
Out[10]: array([3. , 7.5])

In [11]: 3 * v ❼
Out[11]: array([ 4.5 , 11.25])
```

❶ Импорт библиотеки NumPy.

❷ Создание объекта ndarray.

❸ Вывод хранимых в объекте данных.

❹ Поиск и вывод типа данных всех элементов.

❺ Поиск и вывод формы объекта.

❻ Сложение векторов.

❼ Скалярное умножение.

Мы увидели работу Python с математическими понятиями, связанными с векторами. Осталось рассмотреть применение этих возможностей к финансам:

```
In [12]: S = np.array((20, 5)) ❶

In [13]: K = 15 ❷

In [14]: C = np.maximum(S - K, 0) ❸

In [15]: C ❹
Out[15]: array([5, 0])
```

❶ Обозначение неопределенной цены акции как объект `ndarray`.

❷ Определение цены исполнения опциона через переменную Python с целочисленным значением (объект `int`).

❸ Поэлементное вычисление функции максимума.

❹ Итоговые данные, хранящиеся в объекте `ndarray`, обозначенном `C`.

Таким образом, при написании книги использовался следующий подход.

1. Вводятся финансовые понятия и концепции.
2. Дается их математическое представление и модель.
3. Математическая модель переводится в исполняемый код Python.

В этом плане наука о финансах обосновывает использование математики, которая, в свою очередь, объясняет применение методов программирования на Python.

Вводная информация о Python

Одним из преимуществ Python является открытый исходный код, который облегчает его установку на все основные операционные системы — macOS, Windows и Linux. При этом для работы с кодом из этой книги и финансов в целом в дополнение к базовому интерпретатору Python необходимы всего несколько основных пакетов и библиотек, тоже имеющих открытый исходный код.

- `NumPy` (http://numpy.org/). Позволяет эффективно работать с большими *n*-мерными массивами числовых данных.
- `pandas` (http://pandas.pydata.org/). Предназначен в первую очередь для эффективной работы с табличными наборами данных и финансовыми временными рядами. В данной книге `pandas` не будет задействован, однако стоит отметить его особую популярность в области финансов.
- `SciPy` (http://scipy.org/). Является набором научных функций, необходимых, например, для решения типичных задач, связанных с оптимизацией.

- `SymPy` (http://sympy.org/). Позволяет использовать символьную математику, что иногда бывает полезно в работе с финансовыми моделями и алгоритмами.
- `matplotlib` (http://matplotlib.org/). Представляет собой стандартную библиотеку Python для визуализации данных. Она позволяет создавать и настраивать различные типы графиков, например линейные графики, столбчатые диаграммы и гистограммы.

Кроме того, для начала работы с интерактивным кодированием на Python требуются еще два инструмента.

- *IPython* (http://ipython.org/). Самая популярная среда для интерактивного кодирования на Python в командной строке (в терминале, оболочке `shell`).
- *JupyterLab* (http://jupyter.org/). Интерактивная среда для интерактивного кодирования и разработки на Python в браузере.

Технические требования для изучения программирования на Python минимальны. Есть два варианта запуска кода Python.

- *Quant Platform*. На бесплатной платформе Quant Platform (http://finpy.pqp.io/) находится полноценная среда для интерактивной финансовой аналитики с помощью Python. На ней можно запускать код из данной книги через браузер, что делает ненужной установку программы на компьютер. После бесплатной регистрации вы получаете автоматический доступ ко всему коду и всем блокнотам Jupyter Notebook, используемым в книге, что дает возможность сразу выполнять код в браузере.
- *Локальная среда Python*. В настоящее время несложно установить локальную среду Python, которая позволит вам погрузиться в финансовую аналитику и выполнить код из книги на своем компьютере. Далее описано, как произвести установку.

Выбор между локальной средой Python и Quant Platform

Практика показывает, что локальная установка правильной среды Python иногда может вызвать сложности у новичков в программировании. Поэтому, если на начальном этапе установка Python на компьютер вызывает какие-то проблемы, лучше не тратить слишком много времени на нее, а воспользоваться платформой Quant Platform (http://finpy.pqp.io/). К установке Python на компьютер можно вернуться позже, когда у вас будет больше опыта.

Простым и современным способом установки Python является использование менеджера пакетов и среды под названием conda (рис. 1.1).

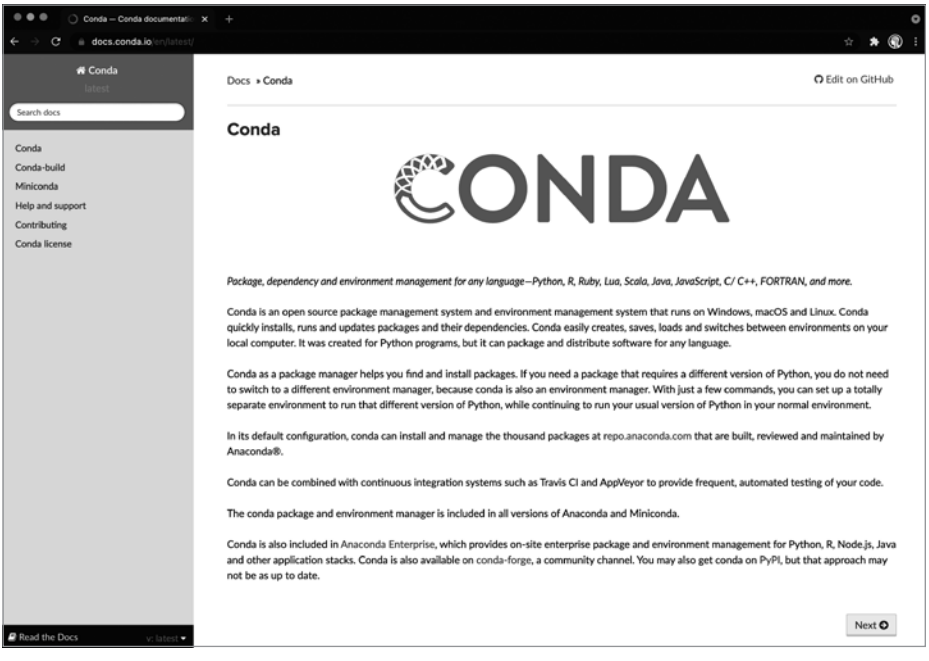


Рис. 1.1. Веб-страница conda

Обычно conda и базовый интерпретатор Python устанавливаются с помощью дистрибутива Miniconda (https://oreil.ly/NI0Wi). На странице загрузки (https://oreil.ly/gaWTP) представлены установочные пакеты последних версий Python для основных операционных систем (рис. 1.2). Дополнительные опции, например, для чипов Apple M1 (из серии Apple Silicon) предоставляются Miniforge (https://oreil.ly/gKeo3).

После установки Miniconda или Miniforge необходимо открыть оболочку или командную строку и проверить доступность conda. Вот, например, результат, полученный на базе conda, установленной через Miniforge на компьютере Apple Mac с чипом M1:

```
(base) minione:finpy yves$ conda --version
conda 4.10.3
(base) minione:finpy yves$
```

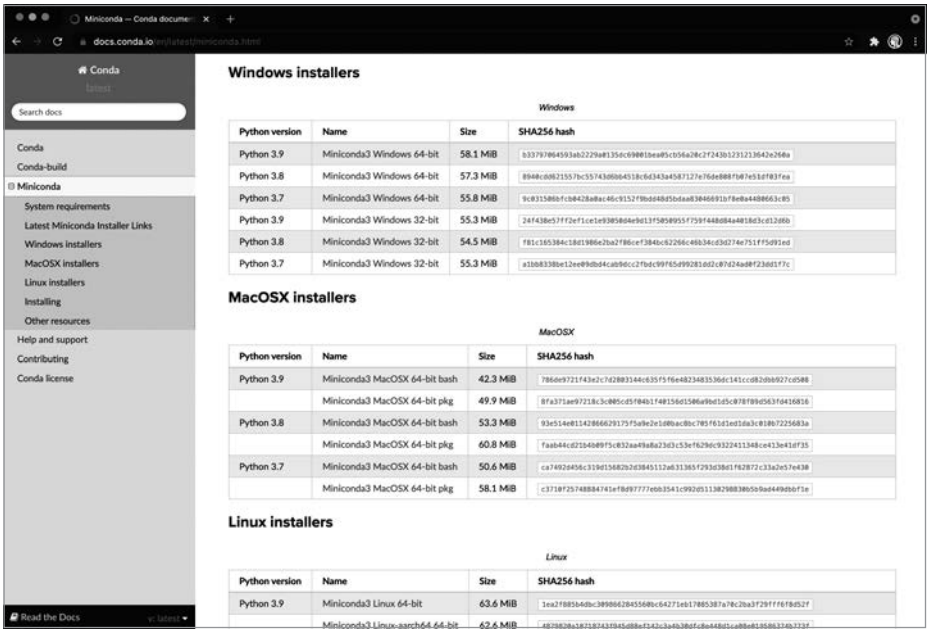


Рис. 1.2. Страница загрузки Miniconda

Обратите внимание на основную часть запроса, которая характерна для установки Python посредством conda. Следующим шагом будет создание новой *среды Python* (с подтверждением запросов с помощью значения `y`):

```
pro:finpy yves$ conda create --name finpy python=3.9
...
Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate finpy
#
# To deactivate an active environment, use
#
#     $ conda deactivate
```

Затем следует активизировать среду:

```
(base) minione:finpy yves$ conda activate finpy
(finpy) minione:finpy yves$
```

Обратите внимание на то, как изменился запрос. Далее устанавливаем инструменты IPython и JupyterLab и подтверждаем появляющиеся запросы с помощью значения y:

```
(finpy) minione:finpy yves$ conda install ipython jupyterlab
...
```

После этого следует установить основные библиотеки и пакеты Python, необходимые для работы с финансовыми данными (флаг `-y` позволяет избежать запроса на подтверждение):

```
(finpy) minione:finpy yves$ conda install -y numpy pandas matplotlib scipy sympy
...
```

Здесь представлены наиболее важные библиотеки и пакеты Python для анализа данных в целом и финансовой аналитики в частности. Проверить, все ли установлено, можно следующим образом:

```
(finpy) minione:finpy yves$ conda list
# packages in environment at /Users/yves/Python/envs/finpy:
#
# Name                    Version                   Build    Channel
anyio                     3.3.0            py39h2804cbe_0    conda-forge
appnope                   0.1.2            py39h2804cbe_1    conda-forge
argon2-cffi               20.1.0           py39h5161555_2    conda-forge
...
jupyterlab                3.1.12             pyhd8ed1ab_0    conda-forge
...
numpy                     1.21.2           py39h1f3b974_0    conda-forge
...
python                    3.9.7        h54d631c_1_cpython    conda-forge
...
zipp                      3.5.0              pyhd8ed1ab_0    conda-forge
zlib                      1.2.11           h31e879b_1009    conda-forge
zstd                      1.5.0               h861e0a7_0    conda-forge
(finpy) minione:finpy yves$
```

Затем с помощью команды `python` можно запустить интерактивную сессию Python:

```
(finpy) minione:finpy yves$ python
Python 3.9.7 | packaged by conda-forge | (default, Sep 14 2021, 01:14:24)
[Clang 11.1.0 ] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print('Hello Finance World.')
```

```
Hello Finance World.
>>> exit()
(finpy) minione:finpy yves$
```

IPython предоставляет расширенную интерактивную оболочку, которую можно запустить командой `ipython`:

```
(finpy) minione:finpy yves$ ipython
Python 3.9.7 | packaged by conda-forge | (default, Sep 14 2021, 01:14:24)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.27.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: from numpy.random import default_rng

In [2]: rng = default_rng(100)

In [3]: rng.random(10)
Out[3]:
array([0.83498163, 0.59655403, 0.28886324, 0.04295157, 0.9736544 ,
       0.5964717 , 0.79026316, 0.91033938, 0.68815445, 0.18999147])

In [4]: exit
(finpy) minione:finpy yves$
```

Однако начинающим пользователям рекомендуется работать в браузерном JupyterLab. Для этого необходимо ввести команду `jupyter lab` в оболочке операционной системы, которая должна выдать сообщение, похожее на следующее:

```
(finpy) minione:finpy yves$ jupyter lab
...
[I 2021-09-16 14:18:21.774 ServerApp] Jupyter Server 1.11.0 is running at:
[I 2021-09-16 14:18:21.774 ServerApp] http://localhost:8888/lab
[I 2021-09-16 14:18:21.774 ServerApp]  or http://127.0.0.1:8888/lab
[I 2021-09-16 14:18:21.774 ServerApp] Use Control-C to stop this server
         and shut down all kernels (twice to skip confirmation).
```

Как правило, эта команда автоматически открывает новую вкладку браузера со стартовой страницей JupyterLab (рис. 1.3).

Теперь можно открыть новый блокнот Jupyter Notebook и начать интерактивное кодирование на Python (рис. 1.4). Осталось щелкнуть на ячейке, ввести в нее код и выполнить его, нажав Shift+Enter, Ctrl+Enter или Alt+Enter (вы заметите разницу).

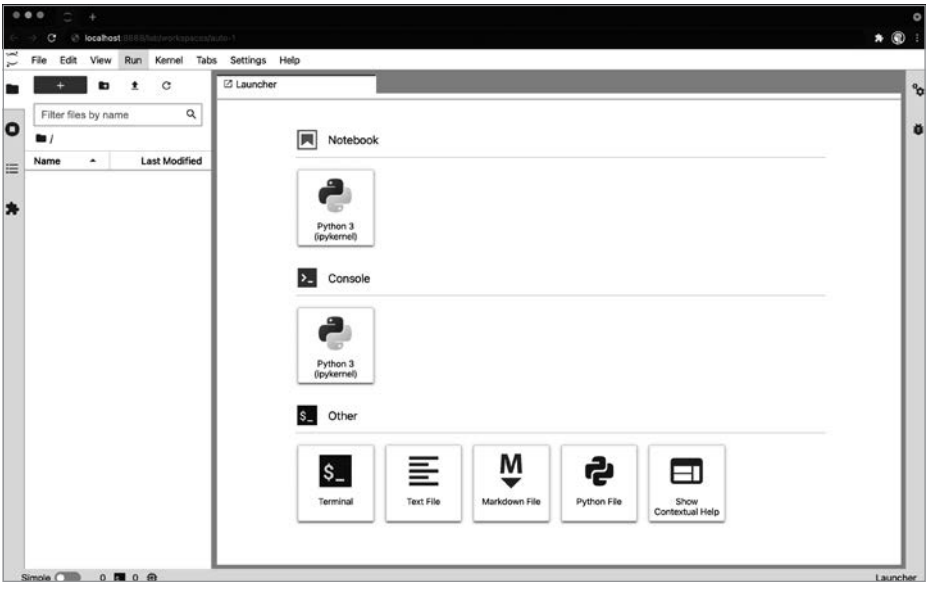


Рис. 1.3. Стартовая страница JupyterLab

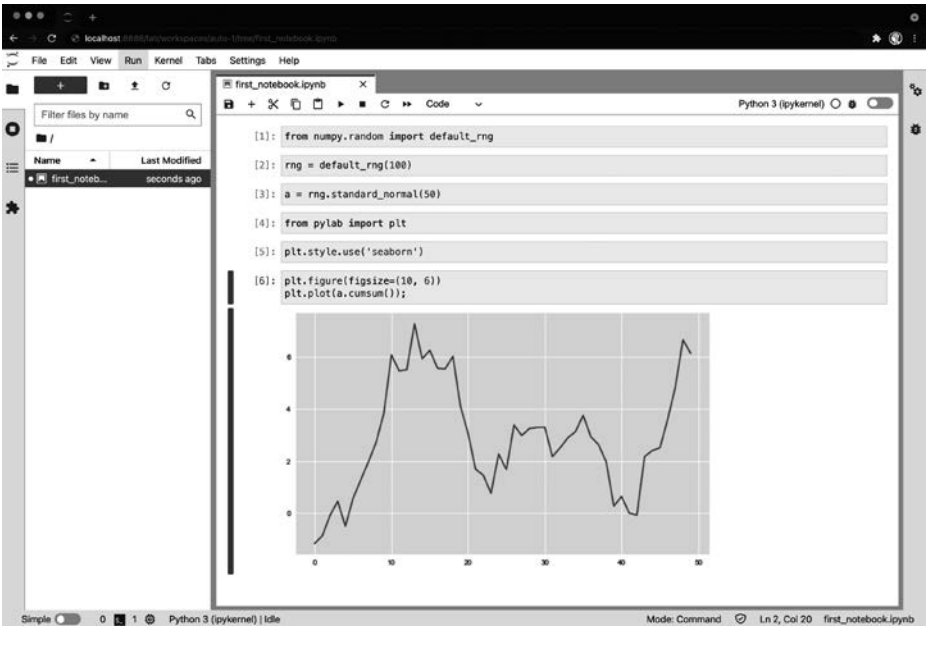


Рис. 1.4. Новый блокнот Jupyter Notebook

Кроме того, можно открыть уже существующий блокнот Jupyter Notebook, представленный в этой книге (рис. 1.5).

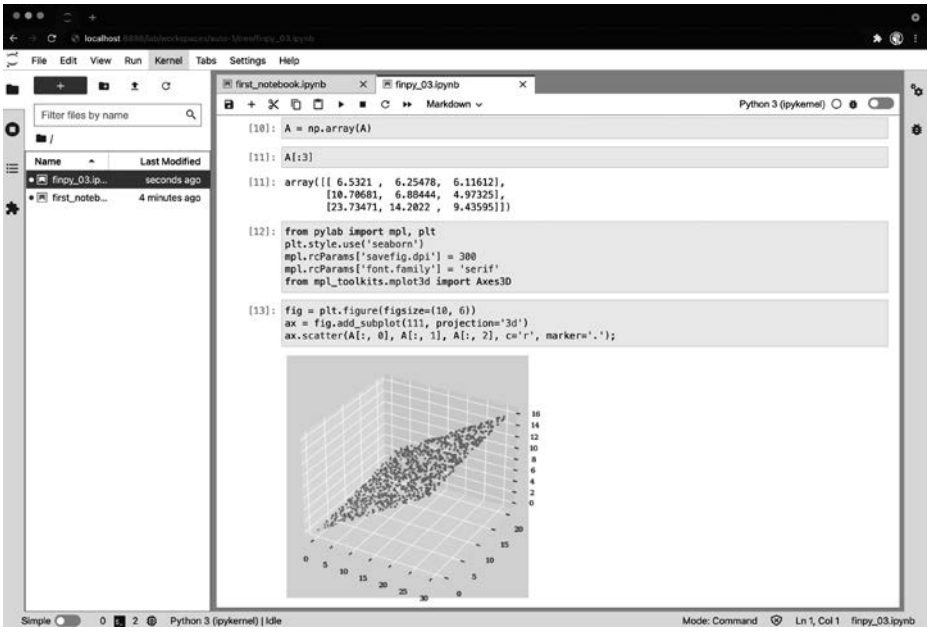


Рис. 1.5. Блокнот Jupyter Notebook из книги

В этом разделе представлены лишь самые основы начала работы с Python и связанными с ним инструментами IPython и JupyterLab. Более подробную информацию (например, о том, как работать с IPython) можно получить из книги Вандерпласа[1] (2016).

Резюме

Наука о финансах может похвастаться длинной историей. Период с 1950 по 1980 год характеризуется внедрением строгого математического анализа в эту область. Начиная с 1980-х годов, и особенно в 2000-х, роль компьютеров и финансовой инженерии значительно возросла. Тенденция дальнейшей компьютеризации будет только усиливаться в связи с распространением

[1] *Vanderplas J.* Python Data Science Handbook. — O'Reilly, 2016.

искусственного интеллекта с его алгоритмами машинного (machine learning, ML) и глубокого обучения (deep learning, DL), требующими больших вычислительных ресурсов.

В финансовой сфере используются четыре типа языка: естественный (по большей части английский), финансовый (понятия и выражения, используемые в данной сфере), математический (например, линейная алгебра или теория вероятностей) и язык программирования (например, Python).

Книга построена таким образом, чтобы познакомить читателя с параллельными понятиями из финансов, математики и программирования. Требования для использования Python минимальны, при этом основным инструментом управления средой чаще всего является conda.

Теперь можно перейти к главе 2, в которой рассматривается самая простая финансовая модель и вводится бóльшая часть базовых финансовых понятий. После работы с такой финансовой моделью, как правило, вырабатывается профессиональное чутье, которое должно облегчить переход к более сложным моделям и методам работы с финансами из главы 3.

Справочные материалы

В этой главе были упомянуты следующие статьи и книги.

- *Cox J., Ingersoll J., Ross S.* A Theory of the Term Structure of Interest Rates // Econometrica, 1985. — № 53 (2). — P. 385–407.
- *Fletcher L.* Hedge Funds Exploit Technology to Reduce Cost and Waste. Financial Times, December 15, 2020. https://oreil.ly/HE4Cc.
- *Heston S.* A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options // The Review of Financial Studies, 1993. — № 6 (2). — P. 327–343.
- *Hilpisch Y.* Python for Finance: Mastering Data-Driven Finance. 2nd ed. — O'Reilly, 2018[1].
- *Hilpisch Y.* Artificial Intelligence in Finance: A Python-Based Guide. — O'Reilly, 2020.

[1] *Хилпиш И.* Python для финансовых расчетов.

- *Longstaff F., Schwartz E.* Valuing American Options by Simulation: A Simple Least Squares Approach // Review of Financial Studies, 2001. — № 14 (1). — P. 113–147.
- *Markowitz H.* Portfolio Selection // Journal of Finance, 1952. — № 7 (1). — P. 77–91.
- *Milne F.* Finance Theory and Asset Pricing. — N. Y.: Oxford University Press, 1995.
- *Rubinstein M.* A History of the Theory of Investments. — Wiley Finance, 2006.

ГЛАВА 2

Экономика с двумя состояниями

> С эмпирической точки зрения наука о финансах направлена на получение конкретных ответов, например на определение правильной стоимости той или иной ценной бумаги или оптимального количества ее акций в портфеле.
>
> *Даррелл Дуффи (1988)*

> Понятие арбитража играет крайне важную роль в современной теории финансов.
>
> *Дельбаен и Шахермайер (2006)*

Эта глава посвящена анализу самой простой *модели экономики* с двумя релевантными моментами времени и двумя неопределенными будущими состояниями. Несмотря на свою простоту, она позволяет объяснить большое количество важных финансовых понятий и концепций, а также некоторые важные достижения, сделанные в финансовой экономике, в частности фундаментальную теорему ценообразования финансовых активов[1].

Модель экономики с двумя состояниями позволяет формально представить иногда довольно абстрактные математические и финансовые понятия с мини-

[1] Подробно о фундаментальной теореме ценообразования финансовых активов можно узнать из оригинальных статей Харрисона и Крепса (1979) и Харрисона и Плиски (1981).

мальным использованием технических моментов. После объяснения таких концепций обычно несложно перейти к более реалистичным финансовым моделям.

Здесь будут рассмотрены следующие темы из области финансов, математики и программирования на Python.

Финансы	Математика	Python
Время	Натуральные числа N	`int`, `type`
Деньги (валюта)	Вещественные числа R	`float`
Денежный поток	Кортеж	`tuple`, `list`
Доход, проценты	Вещественные числа R	`abs`
(Чистая) приведенная стоимость	Функция	`def`, `return`
Неопределенность	Векторное пространство R^2	`NumPy`, `ndarray`, `np.array`
Финансовый актив	Процесс	`ndarray`, `tuple`
Риск	Вероятность, пространство состояний, булеан (показательное множество), отображение	`ndarray`
Ожидание, ожидаемая доходность	Скалярное произведение	`np.dot`
Волатильность	Дисперсия, стандартное отклонение	`np.sqrt`
Условные требования	Случайная величина	`np.arange`, `np.maximum`, `plt.plot`
Репликация, арбитраж	Линейные уравнения, матричная форма	`ndarray(2d)`, `np.linalg.solve`, `np.dot`
Полнота рынка, ценные бумаги Эрроу — Дебре	Линейная независимость, линейная оболочка	`np.linalg.solve`
Ценообразование по мартингалу	Мартингал, мартингальная мера	`np.dot`
Среднее отклонение	Математическое ожидание, дисперсия, стандартное отклонение	`np.linspace`, `.std()`, `[x for y in z]`

Экономика

Основой финансовой модели является идея *экономики*. Экономика — это абстрактное понятие, включающее в себя отдельные элементы финансовой модели: активы (реальные и финансовые), агентов (люди и учреждения) или деньги. Как и в реальном мире, экономику нельзя увидеть или потрогать. Ее невозможно представить в какой-то одной определенной форме — обобщающий термин служит скорее для упрощения коммуникации[1].

Реальные активы

В экономике существует множество *реальных активов*, которые могут быть использованы для различных целей. Реальным активом является как куриное яйцо, так и сложная машина для производства других реальных активов. На данном этапе не рассматривается, кто производит реальные активы или кто ими владеет.

Агенты

Агентами могут быть активно участвующие в экономике люди, которые производят реальные активы, потребляют их или торгуют ими, а также принимают и тратят деньги во время сделок. Агентом может выступать и целое учреждение. Например, банк как отдельный агент позволяет другим агентам вносить депозиты, а затем выплачивает им проценты по этим вкладам.

Время

Экономическая деятельность, как и торговля реальными активами, возможна только в дискретные моменты времени. Формально это можно представить как $t \in 0, 1, 2, 3...$ или $t \in \mathbb{N}_0$. Однако в дальнейшем мы рассмотрим лишь два момента времени — $t = 0$ и $t = 1$, которые обозначают *«сегодня»* и *«ровно через год»* соответственно. Это не единственная возможная интерпретация: например, во многих ситуациях под $t = 0$ и $t = 1$ подразумеваются *«сегодня»* и *«завтра»*. В любом случае если в экономической модели рассматриваются только два *релевантных* момента времени, то согласно финансовой теории она является *статической*.

[1] Более подробное объяснение концепции экономики изложено в главе 5.

В Python натуральные числа $\mathbb{N}$ представлены типом данных `int` (от integers — «целые числа»). С целыми числами можно проводить основные арифметические действия — сложение, вычитание, умножение и другие операции:

```
In [1]: 1 + 3 ❶
Out[1]: 4

In [2]: 3 * 4 ❷
Out[2]: 12

In [3]: t = 0 ❸

In [4]: t ❹
Out[4]: 0

In [5]: t = 1 ❺

In [6]: type(t) ❻
Out[6]: int
```

❶ Сложение двух целых чисел.

❷ Умножение двух целых чисел.

❸ Присваивание переменной `t` значения `0`.

❹ Вывод значения переменной `t`.

❺ Присваивание переменной `t` нового значения `1`.

❻ Поиск и вывод типа данных `t`.

Деньги

В экономике *деньги* (или *валюта*) характеризуются неограниченностью и бесконечной делимостью. Необходимо понимать, что деньги и валюта рассматриваются здесь как абстрактное явление, а не с точки зрения наличных денег — физических монет или купюр.

Деньги в целом служат *мерой стоимости*, поскольку стоимость одной денежной единицы, например доллара США, евро, фунта стерлингов и т. д., приравнена к единице. Цены на все товары выражены в соответствии с этими единицами. Формально денежные единицы представлены в виде неотрицательных вещественных чисел $c \in \mathbb{R}_{\geq 0}$.

В Python вещественные числа $\mathbb{R}$ представлены стандартным типом данных `float` в форме *чисел с плавающей запятой*. Как и `int`, `float`, помимо прочего, позволяет выполнять обычные арифметические действия, такие как сложение и вычитание:

```
In [7]: 1 + 0.5 ❶
Out[7]: 1.5

In [8]: 10.5 - 2 ❷
Out[8]: 8.5

In [9]: c = 2 + 0.75 ❸

In [10]: c ❹
Out[10]: 2.75

In [11]: type(c) ❺
Out[11]: float
```

❶ Сложение двух чисел.

❷ Вычитание двух чисел.

❸ Обозначение результата сложения через переменную `c`.

❹ Вывод значения переменной `c`.

❺ Поиск и печать типа данных переменной `c`.

Помимо определения стоимости чего-либо, деньги могут использоваться агентами для покупки и продажи реальных активов и как средство накопления. Эти две функции основаны на вере в сохранность внутренней ценности денег и в целом на готовности людей и учреждений принимать деньги в любое время и при любой сделке. Функция же определения стоимости не связана с верой во внутреннюю ценность денег, поскольку представляет собой лишь операцию с числами.

Денежный поток

Время и деньги в совокупности образуют *денежный поток*. Рассмотрим инвестиционный проект, который требует вложения, скажем, 9,50 денежной единицы сегодня и возвращает 11,75 денежной единицы через год. Инвестиции обычно считаются *оттоком* денежных средств и часто являются отрицательными вещественными числами, $c \in \mathbb{R}_{<0}$, или в нашем случае $c = -9{,}50$. Возврат

инвестированных средств — это *приток* и, следовательно, положительное вещественное число, $c \in \mathbb{R}_{\geq 0}$, или $c = +11{,}75$.

Для обозначения моментов времени, когда происходят отток и приток денежных средств, используются отметки времени. В нашем случае это $c_{t=0} = -9{,}50$ и $c_{t=1} = 11{,}75$, или сокращенно $c_0 = -9{,}50$ и $c_1 = 11{,}75$.

Денежный поток сейчас и денежный поток через год математически моделируются как *упорядоченная пара*, или *кортеж*, объединяющий эту пару в один объект: $c \in \mathbb{R}^2$, где $c = (c_0, c_1)$ и $c_0, c_1 \in \mathbb{R}$.

В Python есть несколько структур данных для такого математического объекта, основными из которых являются `tuple` и `list`. Они различаются тем, что объекты типа `tuple` нельзя изменить после создания, а объекты типа `list` — можно. Сначала посмотрим на объекты типа `tuple` (обозначены круглыми скобками):

```
In [12]: c0 = -9.5 ❶

In [13]: c1 = 11.75 ❷

In [14]: c = (c0, c1) ❸

In [15]: c ❹
Out[15]: (-9.5, 11.75)

In [16]: type(c) ❺
Out[16]: tuple

In [17]: c[0] ❻
Out[17]: -9.5

In [18]: c[1] ❼
Out[18]: 11.75
```

❶ Определение оттока денежных средств сегодня.

❷ Определение оттока денежных средств ровно через год.

❸ Определение объекта `tuple` через `c` (обратите внимание на круглые скобки).

❹ Вывод пары денежных потоков (обратите внимание на круглые скобки).

❺ Поиск и вывод типа объекта `c`.

❻ Вывод первого элемента объекта `c`.

❼ Вывод второго элемента объекта `c`.

Перейдем к объекту `list` (обозначенному квадратными скобками):

```
In [19]: c = [c0, c1] ❶

In [20]: c ❷
Out[20]: [-9.5, 11.75]

In [21]: type(c) ❸
Out[21]: list

In [22]: c[0] ❹
Out[22]: -9.5

In [23]: c[1] ❺
Out[23]: 11.75

In [24]: c[0] = 10 ❻

In [25]: c ❼
Out[25]: [10, 11.75]
```

❶ Определение объекта `list` через `c` (обратите внимание на квадратные скобки).

❷ Вывод пары денежных потоков (обратите внимание на квадратные скобки).

❸ Поиск и вывод типа объекта `c`.

❹ Вывод первого элемента объекта `c`.

❺ Вывод второго элемента объекта `c`.

❻ Перезапись значения в первой позиции объекта `c`.

❼ Вывод результатов изменения.

Доходность

Доходность, $R \in \mathbb{R}$, инвестиционного проекта с денежными потоками $c = (c_0, c_1) = (-10, 12)$ равна сумме этих денежных потоков, или $R = c_0 + c_1 = -10 + 12 = 2$. *Ставка доходности*, $r \in \mathbb{R}$, — это доход R, деленный на инвестиционные затраты сегодня по модулю $|c_0|$:

$$r = \frac{R}{|c_0|} = \frac{-10+12}{10} = \frac{2}{10} = 0,2.$$

В Python данное вычисление сводится к следующим арифметическим операциям:

```
In [26]: c = (-10, 12) ❶

In [27]: R = sum(c) ❷

In [28]: R ❸
Out[28]: 2

In [29]: r = R / abs(c[0]) ❹

In [30]: r ❺
Out[30]: 0.2
```

❶ Определение пары денежных потоков через объект `tuple`.

❷ Вычисление дохода R путем сложения всех элементов c и...

❸ ...Вывод результата.

❹ Вычисление ставки доходности r через функцию `abs(x)`, возвращающую абсолютное значение `x`...

❺ ...Вывод результата.

Проценты

Денежный поток сегодня отличается от денежного потока через год, и разница между ними обусловлена *процентами*, которые либо начисляются на сумму, предоставляемую в долг, либо которые необходимо заплатить за пользование чужими деньгами. В данном контексте проценты — это *стоимость* управления деньгами других агентов.

Агент, имеющий денежные средства, которые ему сегодня не нужны, может поместить их на депозит в банк или одолжить другому агенту и через определенное время *получить* за это *проценты*. Если агенту нужно больше денег, чем у него есть, он может занять их у банка или других агентов, однако ему придется *заплатить проценты*, соответствующей сумме долга.

Предположим, сегодня агент размещает в банке депозит $c_0 = -10$ денежных единиц. Согласно депозитному договору через год он получит от банка $c_1 = 11$

денежных единиц. Процент по вкладу, $I \in \mathbb{R}$, равен $I = c_0 + c_1 = -10 + 11 = 1$. Следовательно, *процентная ставка*, $i \in \mathbb{R}$, составляет $i = \frac{I}{|c_0|} = 0{,}1$.

В дальнейшем вычислениях мы будем исходить из того, что процентная ставка как по кредиту, так и по депозиту одинакова и постоянна.

Приведенная стоимость

Доступность кредитования и депонирования влекут за собой *альтернативные издержки* при вложении денег в инвестиционный проект. К примеру, денежный поток $c_1 = 12{,}1$ через год нельзя напрямую сравнивать по стоимости с денежным потоком $c_0 = 12{,}1$ сегодня, поскольку на деньгах, не вложенных в проект, можно заработать проценты.

Для правильного сравнения этих денежных потоков необходимо рассчитать *приведенную стоимость* через *дисконтирование* с использованием фиксированной процентной ставки в экономике. Дисконтирование можно смоделировать как функцию $D\colon \mathbb{R} \to \mathbb{R}$, $c_1 \mapsto D(c_1)$, которая сопоставляет одно вещественное число (денежный поток через год) с другим вещественным числом (денежным потоком сегодня). С учетом процентной ставки $i = 0{,}1$ приведенная стоимость составит

$$c_0 = D(c_1) = \frac{c_1}{1+i} = \frac{12{,}1}{1+0{,}1} = 11.$$

Данное соотношение является результатом выбора, предусматривающего открытие депозита в банке:

$$c_1 = (1+i)c_0 \Leftrightarrow c_0 = \frac{c_1}{1+i}.$$

Представление дисконтирования и подобных ему математических функций реализовано в Python довольно просто:

```
In [31]: i = 0.1 ❶

In [32]: def D(c1): ❷
             return c1 / (1 + i) ❸

In [33]: D(12.1) ❹
Out[33]: 10.999999999999998

In [34]: D(11) ❺
Out[34]: 10.0
```

❶ Фиксирование процентной ставки `i`.

❷ Определение функции через оператор `def`, где `D` — название функции, `c1` — имя параметра.

❸ Вывод приведенной стоимости через оператор `return`.

❹ Расчет приведенной стоимости `12.1` (обратите внимание на ошибку округления из-за внутренних проблем представления чисел с плавающей запятой).

❺ Расчет приведенной стоимости `11` (здесь получилось целое число).

Чистая приведенная стоимость

Как агенту решить, стоит ли реализовывать инвестиционный проект? Одним из критериев является *чистая приведенная стоимость*. Чистая приведенная стоимость, $\text{NPV} \in \mathbb{R}$, представляет собой сумму оттока денежных средств сегодня и приведенной стоимости притока денежных средств через год:

$$\text{NPV}(c) = c_0 + D(c_1).$$

Расчет чистой приведенной стоимости представляет собой функцию NPV: $\mathbb{R}^2 \to \mathbb{R}$, сопоставляющую кортеж денежного потока с вещественным числом. Если чистая приведенная стоимость проекта имеет положительное значение, то проект следует реализовать, если отрицательное — нет, поскольку в таком случае альтернатива в виде банковского депозита более выгодна.

Рассмотрим инвестиционный проект с денежными потоками $c_A = (-10{,}5, 12{,}1)$. Чистая приведенная стоимость $\text{NPV}(c_A) = -10{,}5 + D(12{,}1) = -10{,}5 + 11 = 0{,}5$, то есть данный проект выгоден. Другой инвестиционный проект, $c_B = (-10{,}5, 11)$, имеет отрицательную чистую приведенную стоимость $\text{NPV}(c^B) = -10{,}5 + D(11) = -10{,}5 + 10 = -0{,}5$, поэтому агенту стоит отказаться от него.

Эти примеры легко реализуются в Python через соответствующую функцию:

```
In [35]: def NPV(c):
             return c[0] + D(c[1])

In [36]: cA = (-10.5, 12.1) ❶
```

```
In [37]: cB = (-10.5, 11) ❷

In [38]: NPV(cA) ❶
Out[38]: 0.4999999999999982

In [39]: NPV(cB) ❷
Out[39]: -0.5
```

❶ Проект с положительной чистой приведенной стоимостью.

❷ Проект с отрицательной чистой приведенной стоимостью.

Неопределенность

Приток денежных средств от инвестиционного проекта через год в целом имеет *неопределенный* характер, так как в реальных условиях на него могут повлиять множество факторов — конкуренция, появление новых технологий, рост экономики, погода, проблемы в ходе реализации проекта и т. д. В модельной экономике понятие *состояний* экономики через год включает в себя влияние всех соответствующих факторов.

Предположим, что через год экономика будет находиться в одном из двух состояний, u и d[1], или, другими словами, в состоянии роста (хорошем) или спада (плохом). Тогда денежный поток проекта через год c_1 можно выразить *вектором*:

$$c_1 \in \mathbb{R}^2$$

с двумя различными значениями:

$$c_1^u,\ c_1^d \in \mathbb{R}^2,$$

представляющими соответствующие денежные потоки для каждого состояния экономики. Формально данный денежный поток представлен так называемым *вектором-столбцом*:

$$c_1 = \begin{pmatrix} c_1^u \\ c_1^d \end{pmatrix}.$$

[1] u — от up («вверх»), d — от down («вниз»). — *Примеч. пер.*

Математическими операциями над такими векторами являются *скалярное умножение* и *сложение*, например:

$$\alpha \cdot c_1 + \beta = \alpha \cdot \begin{pmatrix} c_1^u \\ c_1^d \end{pmatrix} + \beta = \begin{pmatrix} \alpha \cdot c_1^u + \beta \\ \alpha \cdot c_1^d + \beta \end{pmatrix}.$$

Еще одной важной операцией над векторами является создание *линейных комбинаций* векторов. Рассмотрим два вектора, $c_1, d_1 \in \mathbb{R}^2$. Их линейная комбинация составляется следующим образом:

$$\alpha \cdot c_1 + \beta \cdot d_1 = \begin{pmatrix} \alpha \cdot c_1^u + \beta \cdot d_1^u \\ \alpha \cdot c_1^d + \beta \cdot d_1^d \end{pmatrix}.$$

Следует отметить, что здесь и ранее предполагается $\alpha, \beta \in \mathbb{R}$.

Наиболее распространенным способом моделирования векторов и матриц в Python является библиотека NumPy. Для следующего примера кода рассмотрим инвестиционный проект с денежными потоками $c_0 = -10$ и $c_1 = (20,5)^T$, где T обозначает транспонирование вектора — преобразование *горизонтального вектора (вектора-строки)* в *вертикальный вектор (вектор-столбец)*. Основной класс, используемый для моделирования векторов, — `ndarray` (*n*-мерный массив):

```
In [40]: import numpy as np ❶

In [41]: c0 = -10 ❷

In [42]: c1 = np.array((20, 5)) ❸

In [43]: type(c1) ❹
Out[43]: numpy.ndarray

In [44]: c1 ❺
Out[44]: array([20, 5])

In [45]: c = (c0, c1) ❻

In [46]: c ❼
Out[46]: (-10, array([20, 5]))

In [47]: 1.5 * c1 + 2 ❽
Out[47]: array([32. , 9.5])

In [48]: c1 + 1.5 * np.array((10, 4)) ❾
Out[48]: array([35., 11.])
```

❶ Импорт библиотеки `numpy`.

❷ Отток денежных средств сегодня.

❸ Приток денежных средств неопределенного размера через год: одномерные объекты `ndarray` не различают строки (горизонталь) и столбцы (вертикаль).

❹ Поиск и вывод типа `c1`.

❺ Вывод вектора денежного потока.

❻ Объединение денежных потоков в объект `tuple`.

❼ Объект `tuple`, как и объект `list`, может содержать другие сложные структуры данных.

❽ Линейное преобразование вектора путем скалярного умножения и сложения (векторная числовая операция и трансляция).

❾ Линейная комбинация двух объектов `ndarray` (векторов).

Финансовые активы

Финансовыми активами называются финансовые инструменты (контракты), которые имеют фиксированную цену сегодня и неопределенную цену через год. К примеру, доля в собственном капитале фирмы, реализующей инвестиционный проект, может быть доступна по цене сегодня $S_0 \in \mathbb{R}_{>0}$. Ее стоимость через год зависит от успеха инвестиционного проекта, то есть от того, каков будет приток денежных средств: высокий в состоянии *u* или низкий в состоянии *d*. Формально это выражается как $S_1^u,\ S_1^d \in \mathbb{R}_{\geq 0}$ при $S_1^u > S_1^d$.

Процесс ценообразования финансового актива S: $\mathbb{N}_0 \times \{u, d\} \to \mathbb{R}_{\geq 0}$ заключается в установлении цены финансового актива на основании времени и состояния экономики. Следует отметить, что, в отличие от цены сегодня $S_0^u = S_0^d \equiv S_0$, цена через год полностью зависит от состояния экономики, что можно выразить формулой $(S_t)_{t\in\{0,1\}} = (S_0, S_1)$ или ее укороченной версией $S = (S_0, S_1)$. Для моделирования процесса ценообразования используется также библиотека NumPy:

```
In [49]: S0 = 10 ❶

In [50]: S1 = np.array((12.5, 7.5)) ❷
```

```
In [51]: S = (S0, S1) ❸

In [52]: S ❹
Out[52]: (10, array([12.5, 7.5]))

In [53]: S[0] ❺
Out[53]: 10

In [54]: S[1][0] ❻
Out[54]: 12.5

In [55]: S[1][1] ❼
Out[55]: 7.5
```

❶ Цена финансового актива сегодня.

❷ Неопределенная цена финансового актива через год в виде вектора (объекта `ndarray`).

❸ Процесс ценообразования в виде объекта `tuple`.

❹ Вывод информации касательно процесса ценообразования.

❺ Вывод цены финансового актива сегодня.

❻ Вывод цены финансового актива через год при u (первом) состоянии.

❼ Вывод цены финансового актива через год при d (втором) состоянии.

Риск

Часто неявно предполагается *равновероятность* двух состояний экономики, то есть при бесконечном многократном повторении рассматриваемого эксперимента в экономике наблюдается, что в одной половине случаев реализуется состояние u, а в другой — d.

При определении вероятности на основе частоты вероятность реализации состояния рассчитывается как соотношение частоты наблюдения состояния к общему числу испытаний. Если состояние u наблюдается в 30 экспериментах из 50, то вероятность $p \in \mathbb{R}_{\geq 0}$ при $0 \leq p \leq 1$ составляет $p = 30 / 50 = 0{,}6$, или 60 %.

В моделировании предполагается, что вероятность реализации всех возможных состояний задана *априори*. Ее называют также объективной вероятностью или физической вероятностью.

Вероятностная мера

Совокупность вероятностей физически возможных событий образует *вероятностную меру* с функцией $P: \wp(\{u, d\}) \to \mathbb{R}_{\geq 0}$, отображающей на единичном интервале все элементы *булеана (показательного множества)* $\{u, d\}$ при $\wp(\{u, d\}) = \{\varnothing, \{u\}, \{d\}, \{u, d\}\}$. В данном случае булеан представляет собой все физически возможные события (множество всех подмножеств).

В нашем случае множество $\{u, d\}$ является *пространством состояний* и обозначается символом Ω. Тройка $(\Omega, \wp(\Omega), P)$ называется *вероятностным пространством*.

Функция P, представляющая вероятностную меру, должна удовлетворять трем условиям.

1. $P(\varnothing) = 0$.
2. $0 \leq P(\omega),\ \omega \in \Omega \leq 1$.
3. $P(\Omega) = P(u) + P(d) = 1$.

Первое условие предполагает обязательную реализацию по крайней мере одного из состояний. Второе означает, что вероятность реализации того или иного состояния должна находиться в диапазоне от 0 до 1. Под третьим условием подразумевается, что все вероятности составляют в сумме 1.

В простой модельной экономике с двумя состояниями вероятность можно выразить как $p \equiv P(u)$ и на основе третьего условия посчитать $P(d) = 1 - p$. Таким образом, вероятностная мера P определяется через значение вероятности p.

При наличии полностью определенной вероятностной меры модельная экономика обычно называется *экономикой в условиях риска*, а при ее отсутствии — *экономикой в условиях неопределенности*.

Как правило, вероятностная мера моделируется через вектор и объект `ndarray`. По крайней мере это возможно в условиях дискретного пространства состояний с конечным числом элементов:

```
In [56]: p = 0.4

In [57]: 1 - p
Out[57]: 0.6
```

```
In [58]: P = np.array((p, 1-p))

In [59]: P
Out[59]: array([0.4, 0.6])
```

Понятие неопределенности

Неопределенность в финансовой экономике может принимать различные формы. Риск в целом относится к ситуации, когда известно (предполагается, что известно) полное распределение вероятностей по будущим состояниям экономики. Неопределенность относится к ситуациям, в которых такое распределение неизвестно. Практически всегда финансы опираются на модельную экономику в условиях риска, однако существует ряд исследований, посвященных финансовым проблемам именно в условиях неопределенности. См. соответствующую научную литературу в разделе «Справочные материалы» в конце главы.

Математическое ожидание

На основе вероятностной меры можно рассчитать *математическое ожидание* случайной величины — например, стоимости финансового актива через год. Математическое ожидание можно интерпретировать как *средневзвешенное значение*, где вес задается вероятностями, а усредненность обусловлена тем, что все вероятности в сумме должны составлять 1.

Рассмотрим финансовый актив с процессом ценообразования $S = (S_0, S_1)$. Математическое ожидание цены S_1 через год по вероятностной мере P составляет:

$$\mathbf{E}^P\left(S_1\right) \equiv \sum_{\omega \in \Omega} P\left(\omega\right) \cdot S_1^{\omega} = p \cdot S_1^u + \left(1-p\right) \cdot S_1^d,$$

где $p \equiv P(u)$. Если $S_1 = (20,5)^T$ и $p = 0,4$, то значение ожидания:

$$\mathbf{E}^P\left(S_1\right) = 0,4 \cdot 20 + \left(1-0,4\right) \cdot 5 = 11.$$

Математическое ожидание можно выразить *скалярным произведением* (или *внутренним произведением*) двух векторов. При $x, y \in \mathbb{R}^2$ скалярное произведение определяется как:

$$\left(x, y\right) = \sum_{i=1}^{2} x_i \cdot y_i = x_1 \cdot y_1 + x_2 \cdot y_2.$$

Таким образом, при $P = (p, 1 - p)^{\mathrm{T}}$ и $S_1 = \left(S_1^u, S_1^d\right)^{\mathbf{T}}$ математическое ожидание можно рассчитать как:

$$\mathbf{E}^P\left(S_1\right) = \left(P, S_1\right) = \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} S_1^u \\ S_1^d \end{pmatrix}\right) = p \cdot S_1^u + \left(1-p\right) \cdot S_1^d.$$

В Python скалярное произведение определяется в виде функции через объект `ndarray`:

```
In [60]: P ❶
Out[60]: array([0.4, 0.6])

In [61]: S0 = 10 ❷

In [62]: S1 = np.array((20, 5)) ❸

In [63]: np.dot(P, S1) ❹
Out[63]: 11.0
```

❶ Ранее определенная вероятностная мера.

❷ Цена финансового актива сегодня.

❸ Вектор неопределенной цены финансового актива через год.

❹ Скалярное произведение двух векторов, рассчитывающее ожидание.

Ожидаемая доходность

В условиях неопределенности необходимо подкорректировать понятия дохода и ставки доходности. *Ожидаемая доходность* финансового актива равна разности ожидаемой цены через год и фактической цены сегодня. В этом можно убедиться, если взять математическое ожидание дохода $R = (R^u, R^d)^{\mathrm{T}}$ и преобразовать его следующим образом:

$$\mathbf{E}^P\left(R\right) = \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} R^u \\ R^d \end{pmatrix}\right) = \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} S_1^u - S_0 \\ S_1^d - S_0 \end{pmatrix}\right) =$$

$$= p \cdot \left(S_1^u - S_0\right) + \left(1-p\right) \cdot \left(S_1^d - S_0\right) = p \cdot S_1^u + \left(1-p\right) \cdot S_1^d - S_0 = \mathbf{E}^P\left(S_1\right) - S_0.$$

Подставляем полученные ранее значения и получаем:

$$\mathbf{E}^P\left(R\right) = 0{,}4 \cdot \left(20-10\right) + \left(1-0{,}4\right)\left(5-10\right) = 11-10 = 1.$$

Таким образом, *ожидаемая ставка доходности* — это обычное соотношение ожидаемой доходности к цене сегодня:

$$\mathbf{E}^P(r) = \frac{\mathbf{E}^P(R)}{S_0},$$

которое можно получить, выполнив аналогичные преобразования для ожидаемой доходности. В дальнейшем ожидаемая ставка доходности для краткости будет обозначаться выражением $\mu \equiv \mathbf{E}^P(r)$.

В Python смоделировать расчет ожидаемой доходности и ожидаемой ставки доходности можно с помощью двух простых функций:

```
In [64]: def ER(x0, x1):
             return np.dot(P, x1) - x0 ❶

In [65]: ER(S0, S1) ❷
Out[65]: 1.0

In [66]: def mu(x0, x1):
             return (np.dot(P, x1) - x0) / x0 ❸

In [67]: mu(S0, S1) ❹
Out[67]: 0.1
```

❶ Определение ожидаемой доходности.

❷ Ожидаемая доходность определенного ранее финансового актива.

❸ Определение ожидаемой ставки доходности.

❹ Ожидаемая ставка доходности для определенного актива.

Волатильность

В финансах главная пара понятий — это *риск* и *ожидаемая доходность*. Риск можно измерить множеством способов, а *волатильность*, измеряемая стандартным отклонением ставок доходности, вероятно, является наиболее распространенным финансовым показателем. В нашем случае *дисперсия* ставок доходности финансового актива определяется как:

$$\sigma^2(r) = \mathbf{E}^P\left((r-\mu)^2\right) = \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} (r^u-\mu)^2 \\ (r^d-\mu)^2 \end{pmatrix}\right),$$

где $r^{\omega} \equiv \left(S_1^{\omega} - S_0\right) \Big/ S_0$, $\omega \in \Omega$. Поскольку *волатильность* определяется как стандартное (среднеквадратическое) отклонение ставок доходности, она представляет собой квадратный корень из дисперсии:

$$\sigma(r) = \sqrt{\sigma^2(r)}.$$

Далее приведены функции Python, моделирующие эти два показателя риска, а также вспомогательная функция для расчета вектора ставок доходности:

```
In [68]: def r(x0, x1):
             return (x1 - x0) / x0 ❶

In [69]: r(S0, S1) ❷
Out[69]: array([ 1. , -0.5])

In [70]: mu = np.dot(P, r(S0, S1)) ❸

In [71]: mu ❹
Out[71]: 0.10000000000000003

In [72]: def sigma2(P, r, mu):
             return np.dot(P, (r - mu) ** 2) ❺

In [73]: sigma2(P, r(S0, S1), mu) ❻
Out[73]: 0.54

In [74]: def sigma(P, r, mu):
             return np.sqrt(np.dot(P, (r - mu) ** 2)) ❼

In [75]: sigma(P, r(S0, S1), mu) ❽
Out[75]: 0.7348469228349535
```

❶ Векторное вычисление ставки доходности.

❷ Применение функции к ранее определенному финансовому активу.

❸ Ожидаемая ставка доходности через скалярное произведение и...

❹ ...Вывод результата.

❺ Определение дисперсии ставок доходности.

❻ Применение функции к вектору ставок доходности.

❼ Определение волатильности.

❽ И ее применение к вектору ставок доходности.

Векторы, матрицы и библиотека NumPy

Финансы как прикладная математическая дисциплина в значительной степени опираются на линейную алгебру и теорию вероятностей. Для дискретной модели экономики оба эти раздела прекрасно обрабатываются с помощью библиотеки NumPy (в частности, посредством объекта ndarray). Помимо моделирования, NumPy подходит для обработки информации, расчетов, оптимизации, визуализации и др. Практически все примеры в книге являются подтверждением этого.

Условные требования

Предположим, в экономике торгуется *условное требование*, то есть финансовый актив, закрепленный некоторым контрактом, выплата по которому зависит от состояния мира через год. Размер выплаты по такому условному требованию произвольный и зависит от состояния или от выплат по другим финансовым активам. В последнем случае, как правило, речь идет о *производных финансовых активах* или *производных финансовых инструментах (деривативах)*. Формально условное требование выражается функцией $C_1\colon \Omega \to \mathbb{R}_{\geq 0},\ \omega \mapsto C_1(\omega)$, сопоставляющей события с (положительными) вещественными числами.

Допустим, в экономике торгуются два финансовых актива: безрисковая облигация с процессом ценообразования $B = (B_0, B_1)$ и рисковая акция с процессом ценообразования:

$$S = \left(S_0, \left(S_1^u, S_1^d\right)^{\mathrm{T}}\right).$$

Выплата *по колл-опциону* (то есть *опциону на покупку*) на акцию через год составляет $C_1(S_1(\omega)) = \max(S_1(\omega) - K, 0)$ и $\omega \in \Omega$. $K \in \mathbb{R}_{\geq 0}$ называется *ценой исполнения опциона*.

В теории вероятностей условным требованием обычно называют *случайную величину*, определяющей характеристикой которой является то, что она сопоставляет элементы пространства состояний (событий) с вещественными числами, как правило, через другие случайные величины, как это происходит с производными активами. Следовательно, цена акции через год $S_1\colon \Omega \to \mathbb{R}_{\geq 0},\ \omega \mapsto S_1(\omega)$ также является случайной величиной[1].

[1] Формальное определение случайной величины приводится в главе 5.

Следующий код Python визуализирует на отрезке вещественной оси выплату по *колл-опциону*. Предполагается, что существует только два состояния экономики и соответственно два соответствующих им значения. Для наглядности на рис. 2.1 графически представлена функция выплаты:

```
In [76]: S1 = np.arange(20) ❶

In [77]: S1[:7] ❷
Out[77]: array([0, 1, 2, 3, 4, 5, 6])

In [78]: K = 10 ❸

In [79]: C1 = np.maximum(S1 - K, 0) ❹

In [80]: C1 ❺
Out[80]: array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

In [81]: from pylab import mpl, plt ❻
         # plotting configuration
         plt.style.use('seaborn')
         mpl.rcParams['savefig.dpi'] = 300
         mpl.rcParams['font.family'] = 'serif'

In [82]: plt.figure(figsize=(10, 6))
         plt.plot(S1, C1, lw = 3.0, label='$C_1 = \max(S_1 - K, 0)$') ❼
         plt.legend(loc=0) ❽
         plt.xlabel('$S_1$') ❾
         plt.ylabel('$C_1$'); ❿
```

❶ Генерация объекта `ndarray` с числами от 0 до 19.

❷ Вывод первых нескольких чисел.

❸ Установление цены исполнения колл-опциона.

❹ Векторный способ расчета выплат по колл-опциону.

❺ Вывод полученных значений: многие из них равны 0.

❻ Импорт основного графического модуля из `matplotlib`.

❼ Построение графика выплат по колл-опциону на основании стоимости акции, установка ширины линии 3 пиксела и подписи к ней в виде объекта-строки с записью формулы в формате LaTeX.

❽ Размещение условных обозначений, используемых в графике, в оптимальном месте (там, где они меньше всего закрывают элементы графика).

❾ Размещение подписи на оси *X*...

❿ ...На оси *Y*.

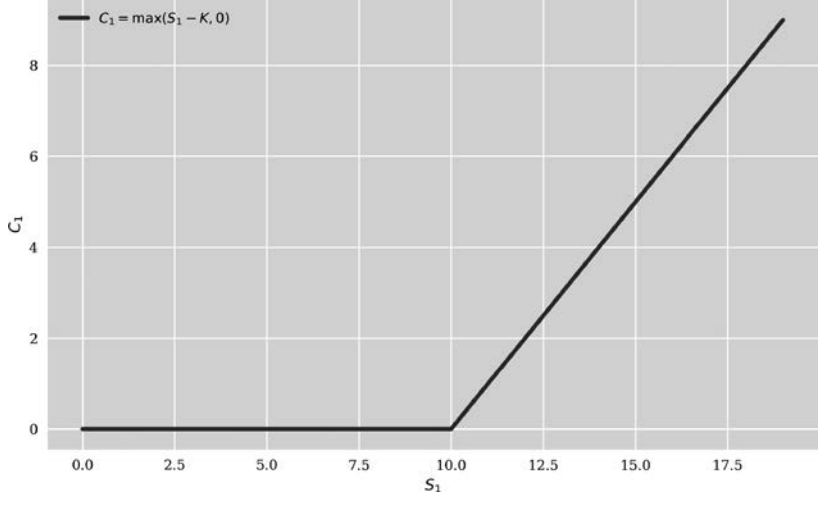


Рис. 2.1. Выплата по колл-опциону

Репликация

При введении в экономику условного требования возникает важный вопрос: является ли выплата по условному требованию избыточной? С математической точки зрения вектор выплат по условному требованию может быть *линейно зависимым* или *линейно независимым*.

Выплата по колл-опциону считается линейно зависимой (или избыточной), если существует решение задачи:

$$b \cdot \begin{pmatrix} B_1 \\ B_1 \end{pmatrix} + s \cdot \begin{pmatrix} S_1^u \\ S_1^d \end{pmatrix} = \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix}$$

при $b, s \in \mathbb{R}$.

Ее можно представить в виде *системы линейных уравнений*:

$$\begin{cases} b \cdot B_1 + s \cdot S_1^u = C_1^u; \\ b \cdot B_1 + s \cdot S_1^d = C_1^d. \end{cases}$$

При условии $S_1^u \neq S_1^d$ эту систему можно развернуть так:

$$s^* = \frac{C_1^u - C_1^d}{S_1^u - S_1^d}$$

и

$$b^* = \frac{1}{B_1} \frac{C_1^d \cdot S_1^u - C_1^u \cdot S_1^d}{S_1^u - S_1^d}.$$

Предположим, как и прежде, что торгуются два финансовых актива: безрисковая облигация $B = (10, 11)$ и рисковая акция $S = (10, (20, 5)^{\mathrm{T}})$. Предположим также, что $K = 15$, в результате чего $C_1 = (5, 0)^{\mathrm{T}}$. Подставив в приведенные ранее формулы эти числа, мы получим:

$$s^* = \frac{5-0}{20-5} = \frac{1}{3}$$

и

$$b^* = \frac{1}{11} \cdot \frac{0 \cdot 20 - 5 \cdot 5}{20-5} = -\frac{5}{33}.$$

Другими словами, покупка одной трети акции и продажа 5/33 облигации без покрытия полностью реплицирует (имитирует) выплату по колл-опциону. Следовательно, выплата по колл-опциону линейно зависит от векторов выплат по облигации и по акции.

В финансах *продажа без покрытия (short sale)* подразумевает следующее: один агент занимает у другого агента определенное количество единиц финансового актива и сразу же продает эту часть на рынке по сегодняшней цене. Через год агент-заемщик выкупает на рынке это же количество единиц финансового актива по актуальной на данный момент цене и возвращает их владельцу.

Здесь допускается бесконечная делимость всех финансовых активов (например, денег), что в действительности может оказаться не так. Вдобавок

предполагается возможность продажи без покрытия всех торгуемых финансовых активов, что, учитывая рыночную практику, может оказаться вполне реалистичным.

В качестве подготовки к реализации репликации на Python рассмотрим еще один способ формулировки данной задачи. Для этого необходимо использовать математическое понятие матрицы. Если вектор — это одномерный объект, то матрица — двумерный.

Рассмотрим квадратную матрицу $\mathcal{M}$ с четырьмя элементами при условии $\mathcal{M} \in \mathbb{R}^{2\times 2}$:

$$\mathcal{M} = \begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^d \end{pmatrix}.$$

Векторы будущих выплат по облигации и по акции представляют собой значения в первом и втором столбцах матрицы соответственно. Первая строка содержит выплаты по обоим финансовым активам в состоянии *u*, а вторая — выплаты при реализации состояния *d*. С учетом этих условностей задача репликации может быть представлена в матричной форме как:

$$\mathcal{M} \cdot \varphi = C_1,$$

где $\phi \in \mathbb{R}^2$ — вектор, содержащий позиции портфеля облигации и акции для репликации $\phi \equiv (b, s)^T$. Как правило, ϕ называют просто портфелем или торговой стратегией. Таким образом:

$$\begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^d \end{pmatrix} \cdot \begin{pmatrix} b \\ s \end{pmatrix} = \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix},$$

в связи с чем *умножение матриц* определяется как:

$$\begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^d \end{pmatrix} \cdot \begin{pmatrix} b \\ s \end{pmatrix} \equiv \begin{pmatrix} B_1 \cdot b + S_1^u \cdot s \\ B_1 \cdot b + S_1^d \cdot s \end{pmatrix},$$

что показывает равенство между этим способом представления задачи репликации и предыдущим.

Матрицы в Python моделируются посредством класса `ndarray`. Модуль `np.linalg` в библиотеке NumPy предоставляет множество функций для

выполнения операций линейной алгебры, среди которых есть и функция для решения систем линейных уравнений в матричной форме — именно то, что нужно в нашем случае:

```
In [83]: B = (10, np.array((11, 11))) ❶

In [84]: S = (10, np.array((20, 5))) ❷

In [85]: M = np.array((B[1], S[1])).T ❸

In [86]: M ❹
Out[86]: array([[11, 20],
                [11,  5]])

In [87]: K = 15 ❺

In [88]: C1 = np.maximum(S[1] - K, 0) ❻

In [89]: C1 ❼
Out[89]: array([5, 0])

In [90]: phi = np.linalg.solve(M, C1) ❽

In [91]: phi ❽
Out[91]: array([-0.15151515, 0.33333333])
```

❶ Определение процесса ценообразования безрисковой облигации.

❷ Определение процесса ценообразования рисковой акции.

❸ Определение матрицы (двумерного объекта `ndarray`) с векторами будущих выплат.

❹ Отображение матрицы с числовыми значениями.

❺ Установление цены исполнения колл-опциона и...

❻ ...Вычисление значения вектора выплат через год.

❼ Отображение числовых значений вектора выплат.

❽ Решение задачи репликации в матричной форме для получения оптимальной позиции портфеля.

Арбитражное ценообразование

Сколько стоит реплицировать (имитировать) выплату по колл-опциону? На этот вопрос можно легко ответить после выведения реплицирующего портфеля. Определим стоимость реплицирующего портфеля сегодня $V_0(\phi)$ через скалярное произведение:

$$V_0(\varphi) \equiv \left(\begin{pmatrix} b \\ s \end{pmatrix}, \begin{pmatrix} B_0 \\ S_0 \end{pmatrix} \right) = b \cdot B_0 + s \cdot S_0,$$

или, если перевести в числовую форму:

$$V_0(\varphi) = b \cdot B_0 + s \cdot S_0 = \frac{10}{3} - \frac{50}{33} = 1{,}818181.$$

Случайная стоимость реплицирующего портфеля через год $V_1(\phi)$ может быть представлена умножением матриц:

$$V_1(\varphi) = \begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^d \end{pmatrix} \cdot \begin{pmatrix} b \\ s \end{pmatrix} = \begin{pmatrix} 5 \\ 0 \end{pmatrix}.$$

В совокупности *процесс формирования стоимости* портфеля выглядит как $V(\phi) = (V_0(\phi), V_1(\phi))$, или $V = (V_0, V_1)$ для краткости, если нет неопределенности относительно портфеля.

Наличие портфеля, идеально реплицирующего будущие выплаты по условному требованию, поднимает следующий вопрос: что, если цена условного требования сегодня отличается от затрат на создание реплицирующего портфеля? Ответ одновременно прост и сложен: значит, в экономике существует *арбитраж* или *арбитражная возможность*. Арбитраж — это торговая стратегия ϕ, которая создает безрисковую прибыль из инвестиций, равных нулю. Формально ϕ является арбитражем, если:

$$V_0(\varphi) = 0 \text{ и } \mathbf{E}^P\left(V_1(\varphi)\right) > 0$$

или

$$V_0(\varphi) > 0 \text{ и } V_1(\varphi) = 0.$$

Предположим, что цена колл-опциона составляет $C_0 = 2$, что выше затрат на создание реплицирующего портфеля. Торговая стратегия продажи колл-опциона на рынке за 2 и покупки реплицирующего его портфеля за 1,818181 приносит немедленную прибыль в размере разницы этих двух чисел. По определению реплицирующего портфеля, через год выплаты по нему и колл-опциону компенсируют друг друга:

$$-\begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix} + b^* \begin{pmatrix} B_1 \\ B_1 \end{pmatrix} + s^* \begin{pmatrix} S_1^u \\ S_1^d \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}.$$

В случае, когда цена колл-опциона сегодня ниже цены реплицирующего портфеля, то есть $C_0 = 1{,}5$, торговая стратегия покупки колл-опциона и продажи реплицирующего портфеля приносит безрисковую прибыль в размере разницы между рыночной ценой колл-опциона и затратами на создание реплицирующего портфеля. Разумеется, безрисковая прибыль в обоих случаях может быть увеличена простым умножением позиции на положительный коэффициент больше единицы.

Модель экономики, допускающую возможность арбитража, можно считать нежизнеспособной. Поэтому $C_0 = 1{,}818181$ — единственная цена, которая согласуется с отсутствием арбитража, то есть является *арбитражной ценой* колл-опциона. Если существует портфель ϕ, реплицирующий выплаты по условному требованию $V_1(\phi) = C_1$, то арбитражная цена условного требования $C_0 = V_0(\phi)$.

Формально арбитражная цена — это скалярное произведение реплицирующего портфеля и вектора цен реплицируемых финансовых активов, которое формирует безарбитражный процесс ценообразования для условного требования $C = (C_0, C_1)$:

$$C_0 \equiv V_0(\varphi) = \left(\varphi^*, \begin{pmatrix} B_0 \\ S_0 \end{pmatrix}\right) = b^* \cdot B_0 + s^* \cdot S_0.$$

В Python арбитражная цена представлена одним расчетом, основанным на предыдущих определениях и вычислениях:

```
In [92]: C0 = np.dot(phi, (B[0], S[0]))

In [93]: C0
Out[93]: 1.8181818181818183

In [94]: 10/3 - 50/33
Out[94]: 1.8181818181818183
```

Полнота рынка

Работает ли арбитражное ценообразование для каждого условного требования? Да, по крайней мере для тех, которые реплицируются портфелями финансовых активов, торгуемых в экономике. *Набор достижимых условных требований* $\mathbb{A}$ включает в себя все те условные требования, которые реплицируются посредством торговли финансовыми активами. Он задается *линейной оболочкой*, представляющей собой множество всех линейных комбинаций векторов будущих цен торгуемых финансовых активов:

$$\mathbb{A} = \left\{ \mathcal{M} \cdot \varphi, \varphi \in \mathbb{R}^2_{\geq 0} \right\},$$

если продажа без покрытия запрещена, и:

$$\mathbb{A} = \left\{ \mathcal{M} \cdot \varphi, \varphi \in \mathbb{R}^2 \right\},$$

если разрешена в неограниченном объеме.

Вернемся к ранее представленным безрисковой облигации и рисковой акции с процессами ценообразования $B = (B_0, B_1)$ и $S = \left(S_0, \left(S_1^u, S_1^d \right)^{\mathrm{T}} \right)$ соответственно, где $B_1, S_1 \in \mathbb{R}^2_{\geq 0}$ и $S_1^u \neq S_1^d$. Явно видно, что задача репликации:

$$\mathcal{M} \cdot \varphi = C_1$$

имеет для любого $C_1 \in \mathbb{R}^2_{\geq 0}$ одно решение:

$$\varphi^* = \begin{pmatrix} b^* \\ s^* \end{pmatrix}$$

или

$$\varphi^* = \begin{pmatrix} \frac{1}{B_1} \frac{C_1^d \cdot S_1^u - C_1^u \cdot S_1^d}{S_1^u - S_1^d} \\ \frac{C_1^u - C_1^d}{S_1^u - S_1^d} \end{pmatrix}.$$

Это решение было выведено при осуществлении репликации выплат по определенному колл-опциону и может считаться универсальным, поскольку не было задано никаких специальных условий относительно выплат, кроме $C_1 \in \mathbb{R}^2_{\geq 0}$.

Поскольку *каждое* условное требование может быть реплицировано портфелем, содержащим позиции по безрисковой облигации и рисковой акции, можно говорить о *модели полного рынка*. Следовательно, каждое условное требование может быть оценено через репликацию и арбитраж. Формально единственное требование заключается в том, чтобы векторы цен двух финансовых активов через год были линейно независимы. Отсюда следует, что:

$$\begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^d \end{pmatrix} \cdot \begin{pmatrix} b \\ s \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

имеет единственное решение $\varphi^* = (0, 0)^T$. Фактически все задачи репликации для произвольных условных требований имеют единственные решения при условии полноты рынка. Векторы выплат двух торгуемых финансовых активов порождают пространство $\mathbb{R}^2$, поскольку они образуют *базис векторного пространства* $\mathbb{R}^2$.

Свойство линейной оболочки может быть визуализировано в Python посредством библиотеки matplotlib. Для этого моделируется 1000 случайных структур портфеля. Первое ограничение заключается в том, что позиции портфеля должны быть положительными и суммарно равными единице. На рис. 2.2 показан результат:

```
In [95]: from numpy.random import default_rng
         rng = default_rng(100) ❶

In [96]: n = 1000 ❷

In [97]: b = rng.random(n) ❸

In [98]: b[:5] ❸
Out[98]: array([0.83498163, 0.59655403, 0.28886324, 0.04295157, 0.9736544 ])

In [99]: s = (1 - b) ❹

In [100]: s[:5] ❹
Out[100]: array([0.16501837, 0.40344597, 0.71113676, 0.95704843, 0.0263456 ])

In [101]: def portfolio(b, s):
              A = [b[i] * B[1] + s[i] * S[1] for i in range(n)] ❺
              return np.array(A) ❻

In [102]: A = portfolio(b, s) ❼
```

```
In [103]: A[:3] ❼
Out[103]: array([[12.48516533, 10.00988978],
                 [14.63101376,  8.57932416],
                 [17.40023082,  6.73317945]])

In [104]: plt.figure(figsize=(10, 6))
          plt.plot(A[:, 0], A[:, 1], 'r.'); ❽
```

❶ Установление начального числа для генератора случайных чисел.

❷ Количество значений для моделирования.

❸ Моделирование позиции облигации для значений от 0 до 1 с помощью равномерного распределения.

❹ Определение позиции акции через разницу между 1 и позицией облигации.

❺ Вычисление векторов выплат по всем случайным портфелям с последующим сбором результатов в объект `list` (это выражение языка Python называется генерацией списков).

❻ Функция возвращает `ndarray`-версию результатов.

❼ Начало вычисления.

❽ Вывод результатов вычисления на экран.

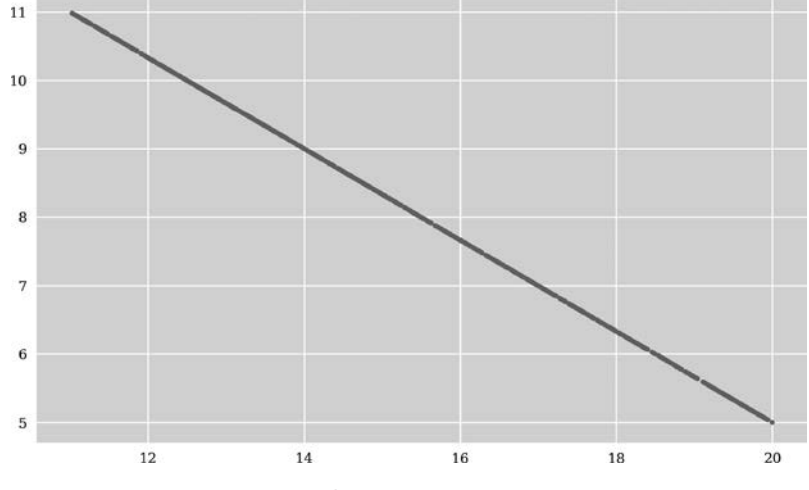


Рис. 2.2. Случайные портфели, расположенные на одномерной линии

На рис. 2.3 показаны результаты, когда позиции портфеля не должны быть равны 1:

```
In [105]: s = rng.random(n) ❶

In [106]: b[:5] + s[:5]
Out[106]: array([1.36885777, 1.5863474 , 0.71245805, 0.32077672, 1.5401562 ])

In [107]: A = portfolio(b, s) ❷

In [108]: plt.figure(figsize=(10, 6))
          plt.plot(A[:, 0], A[:, 1], 'r.');
```

❶ Свободное моделирование акций для значений от 0 до 1.

❷ Вычисление векторов выплат портфеля.

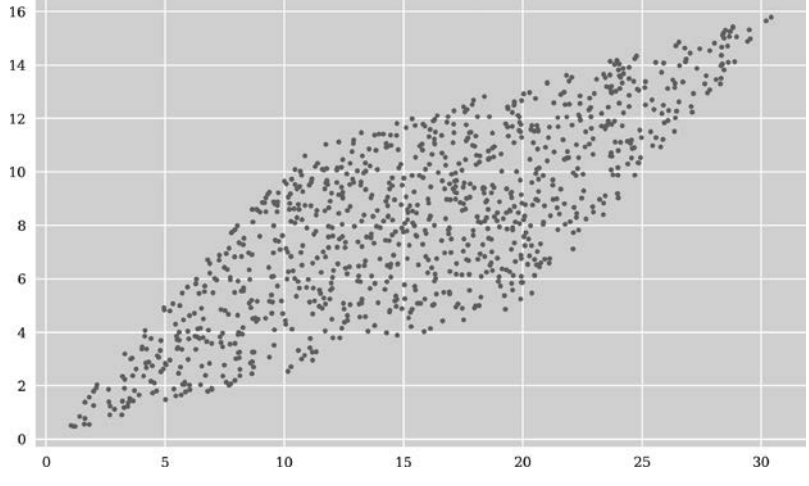


Рис. 2.3. Случайные портфели, расположенные на двумерной области (ромб)

На рис. 2.4 допускаются как положительные, так и отрицательные позиции портфеля по облигации и по акции. Полученные векторы выплат по портфелю лежат в области эллиптической формы вокруг начала координат:

```
In [109]: b = rng.standard_normal(n) ❶

In [110]: s = rng.standard_normal(n) ❶

In [111]: b[:5] + s[:5] ❶
```

```
Out[111]: array([-0.23046605, -3.45760465, 1.10260637, -2.44445777,
                 1.05866637])

In [112]: A = portfolio(b, s)

In [113]: plt.figure(figsize=(10, 6))
          plt.plot(A[:, 0], A[:, 1], 'r.');
```

❶ Положительные и отрицательные позиции портфеля моделируются средствами стандартного нормального распределения.

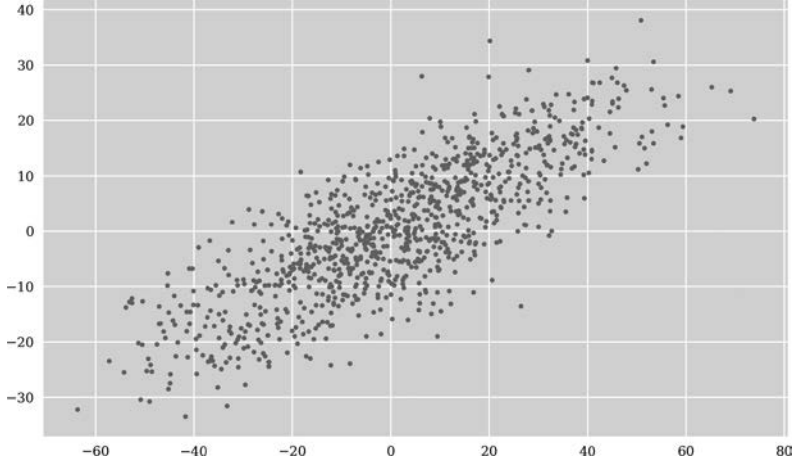


Рис. 2.4. Случайные портфели, расположенные в двумерной области вокруг начала координат

Если b и s могут принимать произвольные значения на вещественной прямой, $b, s \in \mathbb{R}$, то получаемые портфели полностью покрывают векторное пространство $\mathbb{R}^2$. Как говорилось ранее, векторы выплат торгуемых финансовых активов в таком случае порождают пространство $\mathbb{R}^2$.

Ценные бумаги Эрроу — Дебре

Ценная бумага Эрроу — Дебре определяется тем, что выплачивает ровно одну единицу валюты в определенном будущем состоянии. В модельной экономике с двумя различными будущими состояниями может быть только две такие ценные бумаги. Ценная бумага Эрроу — Дебре — это просто частный

случай условного требования, к которому применима репликация. Другими словами, поскольку рынок является полным, ценные бумаги Эрроу — Дебре могут быть реплицированы портфелями по облигациям и акциям. Поэтому обе задачи репликации:

$$\mathcal{M} \cdot \varphi = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

и

$$\mathcal{M} \cdot \varphi = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

имеют (уникальные) решения, и обе ценные бумаги имеют уникальные арбитражные цены.

В чем заключается важность этих ценных бумаг? С математической точки зрения два вектора выплат образуют *стандартный базис*, или *естественный базис*, для векторного пространства $\mathbb{R}^2$. Любой вектор этого пространства может быть однозначно выражен (реплицирован) как линейная комбинация векторов, образующих стандартный базис. С финансовой точки зрения замена исходных векторов будущих цен облигации и акции на ценные бумаги Эрроу — Дебре в качестве базиса для модельной экономики значительно упрощает задачу репликации для всех остальных условных требований.

Процесс репликации в таком случае начинается с составления реплицирующих портфелей и расчета арбитражных цен для двух ценных бумаг Эрроу — Дебре. После чего на основе стандартного базиса и арбитражных цен этих двух ценных бумаг можно реплицировать и оценить другие условные требования.

Рассмотрим две ценные бумаги Эрроу — Дебре с процессами ценообразования $\gamma^u = \left(\gamma_0^u, (1,0)^{\mathrm{T}}\right)$ и $\gamma^d = \left(\gamma_0^d, (0,1)^{\mathrm{T}}\right)$ и определим:

$$M^{\gamma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Общее условное требование с вектором будущих выплат будет иметь вид:

$$C_1 = \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix}.$$

Следовательно, реплицирующий портфель ϕ^{γ} для условного требования тривиально задается через формулу $\varphi^{\gamma} = \left(C_1^u, C_1^d\right)^{\mathbf{T}}$, так как:

$$V_1\left(\varphi^{\gamma}\right) = \mathcal{M}^{\gamma} \cdot \varphi^{\gamma} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix} = \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix}.$$

Получается, что арбитражную цену условного требования можно рассчитать через:

$$C_0 = V_0\left(\varphi^{\gamma}\right) = C_1^u \cdot \gamma_0^u + C_1^d \cdot \gamma_0^d.$$

Таким образом мы продемонстрировали, как ценные бумаги Эрроу — Дебре упрощают репликацию условных требований и процесс арбитражного ценообразования.

Ценообразование по мартингалу

Особым вариантом вероятностной меры является *мартингальная мера* $Q : \wp(\Omega) \to \mathbb{R}_{\geq 0}$, которая преобразует процесс дисконтирования цены финансового актива в *мартингал*. Акция становится мартингалом Q, если:

$$S_0 = \frac{1}{1+i} \cdot \mathbf{E}^Q\left(S_1\right).$$

При $i = \dfrac{B_1 - B_0}{B_0}$ данное выражение тривиально выполнено для безрисковой облигации:

$$B_0 = \frac{1}{1+i} \cdot \mathbf{E}^Q\left(B_1\right) = \frac{1}{1+i} \cdot B_1.$$

Также можно говорить о том, что в случае мартингала процессы ценообразования дрейфуют (в среднем) с безрисковой ставкой:

$$\begin{cases} B_0 \cdot (1+i) = B_1; \\ S_0 \cdot (1+i) = \mathbf{E}^Q\left(S_1\right). \end{cases}$$

Если обозначим $q \equiv Q(u)$, то получим следующее равенство:

$$q \cdot S_1^u + (1-q) \cdot S_1^d = S_0 \cdot (1+i)$$

и после простых преобразований:

$$q = \frac{S_0 \cdot (1+i) - S_1^d}{S_1^u - S_1^d}.$$

Учитывая изложенные ранее условия, для того чтобы q определяло вероятностную меру, необходимо, чтобы выполнялось условие $S_1^u > S_0 \cdot (1+i) > S_1^d$, в результате которого появляется новое вероятностное пространство $(\Omega, \wp(\Omega), Q)$, где Q заменяет P.

Но что, если эти соотношения не выполняются для S_1? Тогда *простой арбитраж* заключается либо в покупке рискового актива при условии $S_1^u > S_0 \cdot (1+i) > S_1^d$, либо в его продаже при другом условии — $S_0 \cdot (1+i) \geq S_1^u$.

В случае выполнения равенства при обоих условиях можно говорить о *слабом арбитраже*, поскольку величину безрисковой прибыли нельзя назвать точно — лишь приблизительно.

Если брать за основу процессы ценообразования из примеров выше, расчет q в Python подразумевает простую арифметическую операцию над числами с плавающей запятой:

```
In [114]: i = (B[1][0] - B[0]) / B[0]

In [115]: i
Out[115]: 0.1

In [116]: q = (S[0] * (1 + i) - S[1][1]) / (S[1][0] - S[1][1])

In [117]: q
Out[117]: 0.4
```

Первая фундаментальная теорема ценообразования финансовых активов

Рассуждения в конце предыдущего раздела наводят на мысль о связи между мартингальными мерами и арбитражем. В финансовой математике эти, казалось бы, несвязанные понятия формально объединены *первой фундаментальной*

теоремой ценообразования финансовых активов. Первые работы по этой теме были опубликованы Коксом и Россом (1976), Харрисоном и Крепсом (1979) и Харрисоном и Плиской (1981).

Первая фундаментальная теорема ценообразования финансовых активов. Эквивалентны следующие утверждения.

1. Мартингальная мера существует.
2. Экономика является безарбитражной.

В отношении модельной экономики с безрисковой облигацией и рисковой акцией данную теорему легко доказать с помощью теории и приведенных ранее расчетов.

Из утверждения 1 следует утверждение 2: если мартингальная мера существует, то процессы ценообразования не допускают простых (слабых) арбитражей. Поскольку два вектора будущих цен линейно независимы, каждое условное требование может быть реплицировано через торговлю двумя финансовыми активами, что подразумевает уникальность арбитражных цен. Следовательно, арбитража не существует.

Из утверждения 2 следует утверждение 1: как было показано ранее, если модельная экономика свободна от арбитража, то мартингальная мера существует.

Ценообразование через математическое ожидание

Следствием первой фундаментальной теоремы является то, что любое достижимое условное требование $C_1 \in \mathbb{A}$ может быть оценено через расчет математического ожидания с мартингальной мерой будущих выплат по нему и дисконтирования с безрисковой процентной ставкой. Арбитражная цена колл-опциона выводится путем репликации. В предположении, что числа процессов ценообразования для торгуемых финансовых активов одинаковые и числовой вектор будущих выплат для колл-опциона одинаковый, *мартингальная цена* колл-опциона составляет:

$$C_0 = \frac{1}{1+i} \cdot \mathbf{E}^Q\left(C_1\right) = \frac{1}{1+i} \cdot \left(q \cdot C_1^u + \left(1-q\right) \cdot C_1^d\right) =$$
$$= \frac{1}{1+0{,}1} \cdot \left(0{,}4 \cdot 5 + \left(1-0{,}4\right) \cdot 0\right) = 1{,}818181.$$

Другими словами, процесс дисконтирования ценообразования колл-опциона (и любого другого условного требования) является мартингалом с мартингальной мерой:

$$\begin{cases} B_0 \cdot (1+i) = B_1; \\ S_0 \cdot (1+i) = \mathbf{E}^Q(S_1); \\ C_0 \cdot (1+i) = \mathbf{E}^Q(C_1). \end{cases}$$

В Python ценообразование по мартингалу сводится к вычислению скалярного произведения:

```
In [118]: Q = (q, 1 - q) ❶

In [119]: np.dot(Q, C1) / (1 + i) ❷
Out[119]: 1.8181818181818181
```

❶ Определение мартингальной меры Q в виде объекта `tuple`.

❷ Реализация формулы ценообразования по мартингалу.

Вторая фундаментальная теорема ценообразования финансовых активов

Существует и *вторая фундаментальная теорема ценообразования финансовых активов*, которая связывает уникальность мартингальной меры с полнотой рынка.

Вторая фундаментальная теорема ценообразования финансовых активов. Эквивалентны следующие утверждения.

1. Мартингальная мера уникальна.
2. Модель рынка является полной.

Данная теорема также подходит для простой модельной экономики из предыдущих обсуждений. Более подробно полнота рынка анализируется в главе 3.

Портфель Марковица

Крупным прорывом в области финансов стала формализация и квантификация портфельных инвестиций с помощью портфельной теории (mean-variance portfolio theory, MVP), впервые предложенной Марковицем в 1952 году. В какой-то степени данную теорию можно считать началом зарождения количественных финансов и, следовательно, расширения использования математики в финансовой сфере.

Портфельная теория сводит финансовый актив к первому и второму моменту его случайной доходности, а именно к *среднему значению* ожидаемой ставки доходности и *дисперсии* ставок доходности (или *волатильности*), определяемой как стандартное отклонение ставок доходности. Хотя этот подход обычно называют анализом среднего отклонения, часто используется также словосочетание «средняя волатильность».

Вернемся к безрисковой облигации и рисковой акции с процессами ценообразования $B = \left(B_0, B_1\right)$ и $S = \left(S_0, \left(S_1^u, S_1^d\right)^{\mathrm{T}}\right)$ и матрице будущих цен $\mathcal{M}$, для которой два столбца заданы векторами будущих цен двух финансовых активов. Каковы ожидаемая ставка доходности и волатильность портфеля ϕ, состоящего из b процентов, вложенных в облигацию, и s процентов, вложенных в акцию? Обратите внимание на то, что сейчас предполагается ситуация, для которой $b + s = 1$, причем $b, s \in \mathbb{R}_{\geq 0}$. Конечно, это условие можно смягчить, но оно упрощает объяснение, приводимое в данном разделе.

Ожидаемая доходность портфеля имеет следующий вид:

$$\begin{aligned}\mathbf{E}^P\left(\mathcal{M}\cdot\varphi\right) &= p\cdot\left(b\cdot B_1 + s\cdot S_1^u\right) + \left(1-p\right)\cdot\left(b\cdot B_1 + s\cdot S_1^d\right) = \\ &= p\cdot\left(b\cdot B_1\right) + \left(1-p\right)\cdot\left(b\cdot B_1\right) + p\cdot\left(s\cdot S_1^u\right) + \left(1-p\right)\cdot\left(s\cdot S_1^d\right) = \\ &= b\cdot\mathbf{E}^P\left(B_1\right) + s\cdot\mathbf{E}^P\left(S_1\right) = b\cdot B_1 + s\cdot\mathbf{E}^P\left(S_1\right).\end{aligned}$$

Другими словами, ожидаемая доходность портфеля — это просто сумма b-кратной выплаты по безрисковой облигации и s-кратной ожидаемой выплаты по акции.

Определив $\mathcal{R} \in \mathbb{R}^{2\times 2}$ как матрицу ставок доходности при:

$$\mathcal{R} = \begin{pmatrix} i & r_1^u \\ i & r_1^d \end{pmatrix},$$

мы получаем ставку ожидаемой доходности портфеля:

$$\mathbf{E}^P\left(\mathcal{R}\cdot\varphi\right) = b\cdot\mathbf{E}^P\left(i\right) + s\cdot\mathbf{E}^P\left(r_1\right) = b\cdot i + s\cdot\mu.$$

Другими словами, ожидаемая ставка доходности портфеля — это сумма b-кратной безрисковой процентной ставки и s-кратной ожидаемой ставки доходности акции.

Далее вычислим *дисперсию портфеля*:

$$\sigma^2\left(\mathcal{R}\cdot\varphi\right) = \mathbb{E}^P\left(\left(r - \mathbf{E}^P\left(\mathcal{R}\cdot\varphi\right)\right)^2\right) =$$

$$= \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} \left(b\cdot i + s\cdot r_1^u - b\cdot i - s\cdot\mu\right)^2 \\ \left(b\cdot i + s\cdot r_1^d - b\cdot i - s\cdot\mu\right)^2 \end{pmatrix}\right) =$$

$$= \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} \left(s\cdot r_1^u - s\cdot\mu\right)^2 \\ \left(s\cdot r_1^d - s\cdot\mu\right)^2 \end{pmatrix}\right) = s^2\cdot\sigma^2\left(r_1\right).$$

Иными словами, дисперсия портфеля в s^2 раз больше дисперсии акции, что интуитивно понятно, поскольку облигация не подвержена риску и никак не влияет на отклонение портфеля. Следовательно, *волатильность портфеля* определяется как:

$$\sigma\left(\mathcal{R}\cdot\varphi\right) = \sqrt{\sigma^2\left(\mathcal{R}\cdot\varphi\right)} = \sqrt{s^2\cdot\sigma^2\left(r_1\right)} = s\cdot\sigma\left(r_1\right).$$

Весь этот анализ легко реализуется на языке Python. Сначала необходимо выполнить небольшую подготовку:

```
In [120]: B = (10, np.array((11, 11)))

In [121]: S = (10, np.array((20, 5)))

In [122]: M = np.array((B[1], S[1])).T ❶
```

```
In [123]: M ❶
Out[123]: array([[11, 20],
                 [11,  5]])

In [124]: M0 = np.array((B[0], S[0])) ❷

In [125]: R = M / M0 - 1 ❸

In [126]: R ❹
Out[126]: array([[ 0.1, 1.  ],
                 [ 0.1, -0.5]])

In [127]: P = np.array((0.5, 0.5)) ❺
```

❶ Матрица с будущими ценами финансовых активов.

❷ Вектор с ценами финансовых активов сегодня.

❸ Вычисление матрицы доходности в векторном виде.

❹ Результаты расчета.

❺ Определение вероятностной меры.

На этой основе можно рассчитать ожидаемую доходность и волатильность портфеля через скалярное произведение:

```
In [128]: np.dot(P, R) ❶
Out[128]: array([0.1 , 0.25])

In [129]: s = 0.55 ❷

In [130]: phi = (1-s, s) ❸

In [131]: mu = np.dot(phi, np.dot(P, R)) ❹

In [132]: mu ❺
Out[132]: 0.18250000000000005

In [133]: sigma = s * R[:, 1].std() ❻

In [134]: sigma ❼
Out[134]: 0.41250000000000003
```

❶ Ожидаемая доходность облигации и акции.

❷ Пример веса акции в портфеле (в десятичных долях).

❸ Полученный в результате портфель с нормализованным весом 1.

❹ Ожидаемая доходность портфеля с учетом распределения.

❺ Значение находится между безрисковой доходностью и доходностью акции.

❻ Волатильность портфеля: код Python здесь работает только при $p = 0{,}5$.

❼ Значение находится между волатильностью облигации, равной 0, и волатильностью акции, равной 0,75.

Изменение веса акции в портфеле приводит к различным комбинациям «риск — доходность». На рис. 2.5 показаны ожидаемые доходность и волатильность портфеля для значений s от 0 до 1. Как видно из графика, ожидаемая доходность портфеля (от 0,1 до 0,25) и волатильность (от 0 до 0,75) линейно возрастают с увеличением веса акции в портфеле s:

```
In [135]: values = np.linspace(0, 1, 25) ❶

In [136]: mu = [np.dot(((1-s), s), np.dot(P, R))
                for s in values] ❷

In [137]: sigma = [s * R[:, 1].std() for s in values] ❸

In [138]: plt.figure(figsize=(10, 6))
          plt.plot(values, mu, lw = 3.0, label='$\mu_p$')
          plt.plot(values, sigma, '--', lw = 3.0, label='$\sigma_p$')
          plt.legend(loc=0)
          plt.xlabel('$s$');
```

❶ Создание объекта `ndarray` с 24 равномерно распределенными интервалами между 0 и 1.

❷ Вычисление ожидаемой доходности портфеля для каждого элемента `values` с последующим сохранением результатов в объекте `list`.

❸ Вычисление волатильности портфеля для каждого элемента `values` с последующим сохранением результатов в объекте `list`.

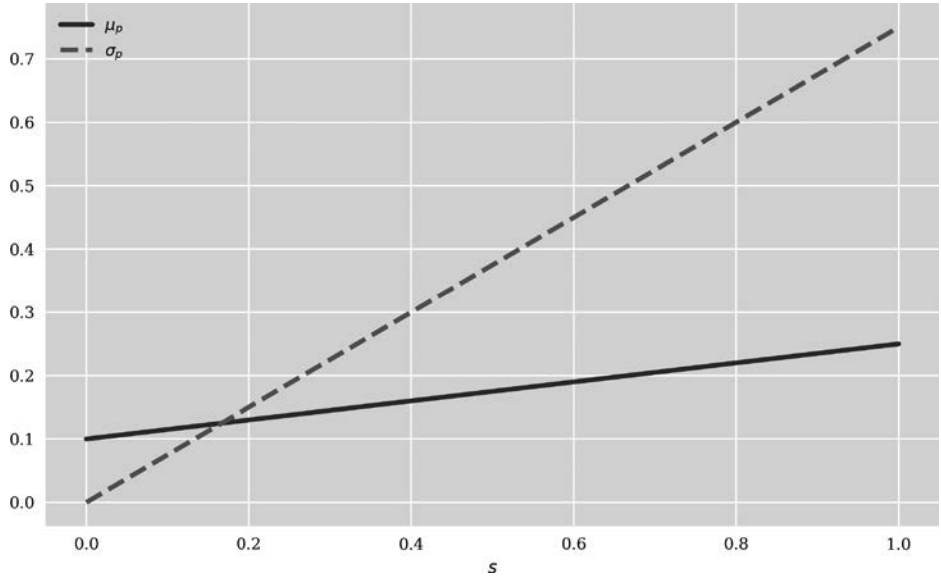


Рис. 2.5. Ожидаемые доходность и волатильность портфеля с различным распределением

Обратите внимание на то, что генерация списка `sigma = [s * R[:, 1].std() for s in values]` является сокращенным вариантом следующего кода[1]:

```
sigma = list()
for s in values:
    sigma.append(s * R[:, 1].std())
```

Типичный график, который можно увидеть в контексте портфельной теории, — это график соотношения ожидаемой доходности и волатильности портфеля. Рисунок 2.6 показывает, что доход инвестора зависит от риска (волатильности), который он готов принять. В нашем случае зависимость линейная:

```
In [139]: plt.figure(figsize=(10, 6))
          plt.plot(sigma, mu, lw = 3.0, label='risk-return')
          plt.legend(loc=0)
          plt.xlabel('$\sigma_p$')
          plt.ylabel('$\mu_p$');
```

[1] Подробнее о структурах данных и выражениях генерации — в документации Python, в разделе «Структуры данных» (https://oreil.ly/0dbCi).

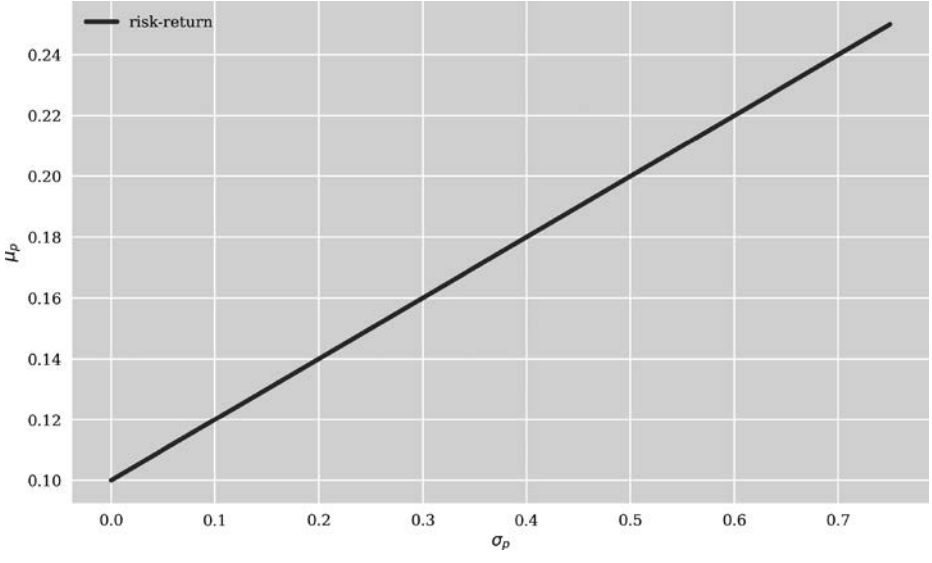


Рис. 2.6. Реальные комбинации ожидаемой доходности и волатильности портфеля

Резюме

В этой главе представлены основы финансов и на простых примерах кода Python показаны важнейшие математические объекты и финансовые понятия. Такие фундаментальные понятия из сферы финансов, как арбитражное ценообразование или взаимосвязь риска и доходности, можно представить и объяснить даже в статичной экономике с двумя состояниями. Базовое понимание и уже сформировавшееся в некоторой степени чутье в финансовых и математических вопросах значительно упрощают переход к более реалистичным финансовым моделям. В следующей главе в пространство состояний добавлено третье, будущее состояние, что позволит рассмотреть проблемы, возникающие в условиях неполноты рынка.

Справочные материалы

В этой главе были упомянуты следующие книги и статьи.

- *Cox J., Ross S.* The Valuation of Options for Alternative Stochastic Processes // Journal of Financial Economics, 1976. — № 3. — P. 145–166.

- *Delbaen F., Schachermayer W.* The Mathematics of Arbitrage. — Berlin: Springer Verlag, 2006.
- *Guidolin M., Rinaldi F.* Ambiguity in Asset Pricing and Portfolio Choice: A Review of the Literature // Theory and Decision, 2013. — № 74. — P. 183–217. https://ssrn.com/abstract=1673494.
- *Harrison M., Kreps D.*. Martingales and Arbitrage in Multiperiod Securities Markets // Journal of Economic Theory, 1979. — № 20. — P. 381–408.
- *Harrison M., Pliska S.* Martingales and Stochastic Integrals in the Theory of Continuous Trading // Stochastic Processes and their Applications, 1981. — № 11. — P. 215–260.
- *Markowitz H.* Portfolio Selection // Journal of Finance, 1952. — № 7 (1). — P. 77–91.

ГЛАВА 3

Экономика с тремя состояниями

> Модель считается полной, если каждое условное требование можно сформировать торговой стратегией. В противном случае модель называется неполной.
>
> *Стэнли Плиска (1997)*

> Предположим, человек рассматривает результат любой инвестиции с точки зрения вероятности, то есть размышляет о возможных результатах через призму некоторого распределения вероятностей. Тем не менее при оценке привлекательности конкретной инвестиции он готов действовать на основании только двух параметров этого распределения — его математического ожидания и стандартного отклонения.
>
> *Уильям Шарп (1964)*

В предыдущей главе мы рассмотрели простейшую модельную экономику, на которой, помимо всего прочего, смогли изучить понятие финансовой неопределенности. В этой главе в экономику с двумя торгуемыми финансовыми

активами добавляется еще одно состояние. На основе *статической экономики* уже с *тремя состояниями* будут рассмотрены неполнота рынка и неопределенность мартингальной меры, а также представлены подходы к решению задачи, связанной с неполнотой рынка и ее последствиями для ценообразования условных требований: суперрепликация и аппроксимирующая репликация. В конце будет рассмотрена модель ценообразования капитальных активов (Capital Asset Pricing Model, CAPM), которая основывается на анализе портфеля среднего отклонения и добавляет параметры равновесия для выведения стоимости реплицируемых и нереплицируемых финансовых активов в пространстве средней волатильности.

Далее представлены основные темы третьей главы.

Финансы	Математика	Python
Неопределенность	Вероятностное пространство	`ndarray`
Финансовые активы	Векторы, матрицы	`ndarray`
Достижимые условные требования	Линейная оболочка векторов, базис векторного пространства	`ndarray`
Ценообразование по мартингалу, арбитраж	Группы вероятностных мер, математическое ожидание	`ndarray`, `np.dot`
Суперрепликация	Минимизация, ограничения	`scipy.optimize.minimize`, `dict`, `lambda`
Аппроксимирующая репликация	Среднеквадратическая ошибка, метод наименьших квадратов (МНК)	`np.linalg.lstsq`
Линия рынка капитала	Математическое ожидание, стандартное отклонение	`NumPy`
Модель ценообразования капитальных активов	Корреляция, ковариация	`NumPy`

Предположения и расчеты, относящиеся к экономике с двумя состояниями (см. предыдущую главу), переносятся на экономику с тремя состояниями, рассматриваемую далее, если не указано иное.

Неопределенность

В экономике с тремя состояниями рассматриваются два *момента времени*: «сегодня», $t = 0$, и «через один год», $t = 1$. *Пространство состояний* имеет вид $\Omega = \{u, m, d\}$, где $\{u, m, d\}$ — это три различных возможных состояния экономики через год. *Булеан* пространства состояний задан как:

$$\wp(\Omega) = \{\varnothing, \{u\}, \{m\}, \{d\}, \{u, m\}, \{u, d\}, \{m, d\}, \Omega\}.$$

На булеане определена вероятностная мера P с условиями $P(\omega) = 1/3$, $\omega \in \Omega$, а вероятностное пространство $(\Omega, \wp(\Omega), P)$ представляет собой *неопределенность* в модельной экономике.

Финансовые активы

В модельной экономике торгуются два финансовых актива. Первый — это *безрисковая облигация* $B = (B_0, B_1)$, где $B_0 = 10$ и $B_1 = (11, 11, 11)^{\mathrm{T}}$. Безрисковая процентная ставка $i = 0{,}1$.

Второй актив — это *рисковая акция* $S = \left(S_0, \left(S_1^u, S_1^m, S_1^d\right)^{\mathrm{T}}\right)$, где $S_0 = 10$ и:

$$S_1 = \begin{pmatrix} 20 \\ 10 \\ 5 \end{pmatrix}.$$

Матрицу рыночных выплат $\mathcal{M} \in \mathbb{R}^{3\times 2}$ обозначим как:

$$\mathcal{M} \equiv \begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^m \\ B_1 & S_1^d \end{pmatrix} = \begin{pmatrix} 11 & 20 \\ 11 & 10 \\ 11 & 5 \end{pmatrix}.$$

Достижимые условные требования

Линейная оболочка торгуемых финансовых активов называется также *множеством достижимых условных требований* $\mathbb{A}$. Условное требование $C_1\colon \Omega \to \mathbb{R}_{\geq 0}$ считается достижимым, если выплата по нему может быть выражена линейной комбинацией векторов выплат по торгуемым активам, или, другими словами,

существует портфель ϕ, который удовлетворяет условию $V_1(\varphi) = \mathcal{M} \cdot \varphi = C_1$. Следовательно:

$$A = \left\{ \mathcal{M} \cdot \varphi, \varphi \in \mathbb{R}^2 \right\},$$

если нет ограничений касательно позиций портфеля, либо:

$$A = \left\{ \mathcal{M} \cdot \varphi, \varphi \in \mathbb{R}^2_{\geq 0} \right\},$$

если продажа без покрытия запрещена.

Доказать линейную независимость векторов выплат по двум финансовым активам не составит труда, однако таких векторов всего два, а состояний — три. Из линейной алгебры мы знаем, что базис векторного пространства $\mathbb{R}^3$ состоит из трех линейно независимых векторов, то есть получается, что в нашем случае не каждое условное требование реплицируется портфелем торгуемых финансовых активов. Если мы попытаемся восполнить базис векторного пространства $\mathbb{R}^3$ *одной из ценных бумаг Эрроу — Дебре*, система линейных уравнений для репликации примет следующий вид:

$$\begin{cases} b \cdot 11 + s \cdot 20 = 1; \\ b \cdot 11 + s \cdot 10 = 0; \\ b \cdot 11 + s \cdot 5 = 0. \end{cases}$$

Вычитание второго уравнения из первого дает $s = 1/10$. Вычитание третьего уравнения из первого дает $s = 1/15$, что, очевидно, противоречит первому решению. Таким образом, эта задача репликации не имеет решения.

В Python множество достижимых условных требований может быть визуализировано в трех измерениях. Данный подход основан на составлении портфеля через моделирование Монте-Карло, в котором для простоты допускаются только положительные позиции портфеля в диапазоне от 0 до 1. На рис. 3.1 изображен график, где два вектора охватывают только двумерную область трехмерного пространства. Если бы рынок был полным, смоделированные векторы выплат заполнили бы куб (финансовые активы покрыли бы векторное пространство $\mathbb{R}^3$), а не образовали в нем прямоугольную область ($\mathbb{R}^2$). Моделирование неопределенности происходит по аналогии с кодом, представленным в предыдущей главе, но с некоторыми изменениями, необходимыми для работы с тремя возможными будущими состояниями экономики:

```
In [1]: import numpy as np
        from numpy.random import default_rng
        np.set_printoptions(precision=5, suppress=True)

In [2]: rng = default_rng(100)

In [3]: B = (10, np.array((11, 11, 11)))

In [4]: S = (10, np.array((20, 10, 5)))

In [5]: n = 1000 ❶

In [6]: b = rng.random(n) ❷

In [7]: b[:5] ❷
Out[7]: array([0.83498, 0.59655, 0.28886, 0.04295, 0.97365])

In [8]: s = rng.random(n) ❸

In [9]: A = [b[i] * B[1] + s[i] * S[1] for i in range(n)] ❹

In [10]: A = np.array(A) ❹

In [11]: A[:3] ❹
Out[11]: array([[19.86232, 14.52356, 11.85418],
                [26.35796, 16.46003, 11.51106],
                [11.64939, 7.41344, 5.29547]])

In [12]: from pylab import mpl, plt ❺
         plt.style.use('seaborn')
         mpl.rcParams['savefig.dpi'] = 300
         mpl.rcParams['font.family'] = 'serif'
         from mpl_toolkits.mplot3d import Axes3D ❻

In [13]: fig = plt.figure(figsize=(10, 6)) ❼
         ax = fig.add_subplot(111, projection='3d') ❽
         ax.scatter(A[:, 0], A[:, 1], A[:, 2], c='r', marker='.'); ❾
```

❶ Количество портфелей, которые необходимо смоделировать.

❷ Случайная позиция в облигации с несколькими примерами (все значения находятся в диапазоне от 0 до 1).

❸ Случайная позиция в акции.

❹ Генерация списка с расчетом векторов выплат по случайным портфелям.

❺ Импорт графического модуля matplotlib и задание основных параметров для него.

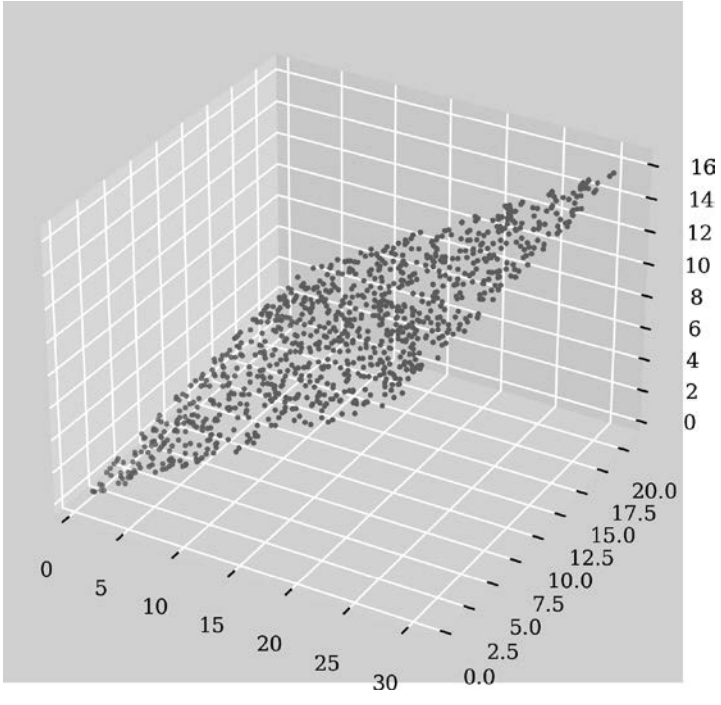


Рис. 3.1. Трехмерная визуализация векторов выплат по случайным портфелям

❻ Импорт инструментов формирования трехмерного графика.

❼ Создание пустого холста.

❽ Добавление области для трехмерного объекта.

❾ Визуализация векторов выплат в виде множества отдельных красных точек.

Неполнота рынка

Если в полной модельной экономике все условные требования достижимы, то в условиях неполного рынка обычно достижимо лишь их небольшое подмножество. В этом плане полные и неполные модельные экономики довольно сильно отличаются друг от друга. Представленное в главе 2 ценообразование через репликацию основывается на достижимости условного требования. Что же делать с ценообразованием, когда репликация не работает? На эти и другие вопросы вы получите ответы далее.

Ценообразование по мартингалу

О важном значении мартингальных мер в финансовом анализе явно свидетельствуют первая и вторая фундаментальные теоремы ценообразования финансовых активов.

Мартингальные меры

Любая вероятностная мера преобразует процесс дисконтированного ценообразования облигации в мартингал. А как насчет процесса ценообразования акции? Уравнением, определяющим мартингальную меру Q: $\wp(\Omega) \to \mathbb{R}_{\geq 0}$, является:

$$S_0 \cdot (1+i) = \mathbf{E}^Q(S_1)$$

или

$$S_0 \cdot (1+i) = q^u \cdot S_1^u + q^m \cdot S_1^m + q^d \cdot S_1^d,$$

где $q^\omega \equiv Q(\omega)$, $\omega \in \Omega$. В численном виде и при условии $q^d = 1 - q^u - q^m$ это уравнение можно представить следующим образом:

$$11 = q^u \cdot 20 + q^m \cdot 10 + (1 - q^u - q^m) \cdot 5 \Leftrightarrow q^m = \frac{6 - 15q^u}{5}.$$

Согласно свойствам вероятностной меры необходимо, чтобы выполнялись следующие условия — обязательное:

$$6 - 15q^u \geq 0 \Leftrightarrow q^u \leq \frac{2}{5}$$

и необязательное:

$$\frac{6 - 15q^u}{5} \leq 1 \Leftrightarrow q^u \geq \frac{1}{15},$$

а также обязательное:

$$q^d = 1 - q^u - q^m \geq 0 \Leftrightarrow q^u \geq \frac{1}{10}$$

и необязательное:

$$q^d = 1 - q^u - q^m \leq 1 \Leftrightarrow q^u \leq \frac{3}{5}.$$

Таким образом, можно заключить, что существует неограниченное количество вероятностных мер, которые преобразуют процесс дисконтированного ценообразования акции в мартингал. При $q \equiv q^u$ множество всех мартингальных мер $\mathbb{Q}$, согласующихся с рыночной моделью, имеет следующий вид:

$$\mathbb{Q} = \left\{ \begin{pmatrix} q \\ \frac{6-15q}{5} \\ 1-q-\frac{6-15q}{5} \end{pmatrix}, \frac{1}{10} \le q \le \frac{2}{5} \right\}.$$

Для примера подставим сюда $q = 3/10$, тогда:

$$Q\left(q = \frac{3}{10}\right) = \begin{pmatrix} \frac{3}{10} \\ \frac{6-15\cdot\frac{3}{10}}{5} \\ 1-\frac{3}{10}-\frac{3}{10} \end{pmatrix} = \begin{pmatrix} \frac{3}{10} \\ \frac{3}{10} \\ \frac{4}{10} \end{pmatrix}$$

и

$$\frac{3}{10}\cdot 20 + \frac{3}{10}\cdot 10 + \frac{4}{10}\cdot 5 = 11,$$

как и требуется для процесса ценообразования акции.

Следовательно, к предыдущему коду мы можем добавить следующее вычисление:

```
In [14]: Q = np.array((0.3, 0.3, 0.4))

In [15]: np.dot(Q, S[1])
Out[15]: 11.0
```

Если исходить из второй фундаментальной теоремы ценообразования финансовых активов, то рыночная модель является *неполной*, поскольку с ней согласуются больше одной мартингальной меры.

Мартингальные меры в условиях неполноты рынка

Модели полного рынка характеризуются наличием лишь одной мартингальной меры. Модели неполного рынка, наоборот, обычно допускают неограниченное их количество. Следует четко понимать, что данный факт значительно влияет на ценообразование условных требований, так как при отсутствии арбитража различные мартингальные меры влекут за собой различные значения выплат по условным требованиям.

Риск-нейтральное ценообразование

К каким последствиям приводит неограниченное количество согласованных с рынком мартингальных мер при арбитражном ценообразовании условных требований? Во-первых, арбитражное ценообразование достижимых условных требований $\mathbb{A}$ выполняется так же, как в условиях полного рынка: стоимость реплицирующего портфеля равна цене реплицируемого условного требования, в противном случае возникают арбитражные возможности. Формально это можно выразить как $C_0 = V_0(\phi)$ при $V_1(\phi) = C_1$.

С недостижимыми условными требованиями $C_1 \in \overline{\mathbb{A}} = \mathbb{R}^3 \setminus \mathbb{A}$ все не так просто. Предположим, что у нас есть одна из ценных бумаг Эрроу — Дебре — γ^u. Как было показано ранее, она нереплицируема и поэтому принадлежит множеству $\overline{\mathbb{A}}$. Ее мартингальная цена составляет:

$$\gamma_0^u(q) = \frac{1}{1+i} \cdot \mathbb{E}^Q\left((1,0,0)^{\mathrm{T}}\right) = \frac{1}{1+i} \cdot q.$$

Величину γ_0^ω часто называют *зависящей от состояния экономики ценой* одной денежной единицы в состоянии $\omega \in \Omega$, и она является простой дисконтированной мартингальной вероятностью для данного состояния.

В модельной экономике должно выполняться условие $1/10 \le q \le 2/5$, поэтому мартингальная цена — цена, при которой невозможен арбитраж, — лежит в интервале:

$$\frac{10}{11} \cdot \frac{1}{11} = \frac{1}{11} \le \gamma_0^u \le \frac{10}{11} \cdot \frac{2}{5} = \frac{4}{11}.$$

Другими словами, в рассматриваемых нами условиях модельной экономики любая цена между $1/11$ и $4/11$ для ценной бумаги Эрроу — Дебре согласуется с отсутствием арбитража. Расчеты для других недостижимых условных требований будут иметь аналогичные результаты.

Суперрепликация

Репликация играет важную роль не только в *ценообразовании*. Она также *хеджирует риск*, возникающий в результате неопределенности выплаты по условному требованию. Рассмотрим произвольное достижимое условное требование. Продажа без покрытия, или короткая продажа, реплицирующего портфеля вместе с удержанием условного требования устраняет любой риск, возникающий из-за неопределенности в отношении будущих выплат. Причиной является то, что выплаты по условному требованию и реплицирующему портфелю полностью компенсируют друг друга. Формально, если портфель φ^* реплицирует условное требование C_1, то:

$$C_1 - V_1\left(\varphi^*\right) = C_1 - \mathcal{M} \cdot \varphi^* = 0.$$

Для недостижимого условного требования такое идеальное хеджирование недоступно, однако всегда можно составить портфель, *суперреплицирующий* выплаты по нему. Портфель ϕ суперреплицирует условное требование C_1, если выплата по нему в каждом будущем состоянии экономики больше выплаты по условному требованию или равна ей — $V_1(\phi) \geq C_1$.

Вернемся к ценной бумаге Эрроу — Дебре γ^u, которая является недостижимым условным требованием. Выплата по ней может быть суперреплицирована, например, портфелем, содержащим лишь безрисковую облигацию:

$$\varphi = \left(\frac{1}{B_1}, 0\right)^{\mathrm{T}}.$$

В результате мы получаем выплату:

$$V_1(\varphi) = \frac{1}{B_1} \cdot \begin{pmatrix} B_1 \\ B_1 \\ B_1 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \geq \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix} = C_1,$$

где знак $\geq$ следует понимать поэлементно. Данное математическое выражение удовлетворяет определению суперрепликации, однако она может оказаться не лучшим выбором с точки зрения затрат, требующихся для создания суперреплицирующего портфеля. Поэтому в большинстве случаев вводится аргумент *минимизации затрат*.

Следовательно, *задача суперрепликации* для условного требования C_1 при *минимальных затратах* имеет вид:

$$\min_{\varphi} V_0(\varphi),$$

таким образом:

$$V_1(\varphi) \geq C_1$$

или

$$\min_{b,s} b \cdot B_0 + s \cdot S_0,$$

таким образом:

$$\begin{cases} b \cdot B_1 + s \cdot S_1^u \geq C_1^u; \\ b \cdot B_1 + s \cdot S_1^m \geq C_1^m; \\ b \cdot B_1 + s \cdot S_1^d \geq C_1^d. \end{cases}$$

Задачи по минимизации затрат могут быть смоделированы и решены в Python посредством библиотеки SciPy. Код, представленный далее, начинается с расчета затрат на неэффективный суперреплицирующий портфель, содержащий только облигацию, а далее задается функция стоимости портфеля. Помимо этого, показаны и другие варианты портфелей, которые могут быть более выгодными:

```
In [16]: C1 = np.array((1, 0, 0)) ❶

In [17]: 1 / B[1][0] * B[1] >= C1❷
Out[17]: array([ True, True, True])

In [18]: 1 / B[1][0] * B[0] ❸
Out[18]: 0.9090909090909092

In [19]: def V(phi, t): ❹
             return phi[0] * B[t] + phi[1] * S[t]

In [20]: phi = np.array((0.04, 0.03)) ❺

In [21]: V(phi, 0) ❻
Out[21]: 0.7

In [22]: V(phi, 1) ❼
Out[22]: array([1.04, 0.74, 0.59])
```

❶ Выплата по условному требованию (ценная бумага Эрроу — Дебре).

❷ Портфель, содержащий только облигацию, проверенный на наличие свойства суперрепликации.

❸ Затраты на создание данного портфеля.

❹ Функция для вычисления стоимости портфеля `phi` сегодня, $t = 0$, или через один год, $t = 1$.

❺ Другой вариант суперреплицирующего портфеля.

❻ Затраты на его создание (которые ниже, чем на портфель, содержащий только облигацию).

❼ И стоимость портфеля, суперреплицирующего ценную бумагу Эрроу — Дебре (то есть выплата по нему), через год.

Вторая часть кода реализует *программу минимизации затрат* на основе предыдущих ограничений в виде векторного неравенства. Оптимальный по стоимости суперреплицирующий портфель намного дешевле, чем портфель, содержащий только облигацию, или более эффективный портфель, включающий облигацию и акцию:

```
In [23]: from scipy.optimize import minimize ❶

In [24]: cons = ({'type': 'ineq', 'fun': lambda phi: V(phi, 1) - C1}) ❷

In [25]: res = minimize(lambda phi: V(phi, 0), ❸
                        (0.01, 0.01), ❹
                        method='SLSQP', ❺
                        constraints=cons) ❻

In [26]: res ❼
Out[26]:      fun: 0.3636363636310989
              jac: array([10., 10.])
          message: 'Optimization terminated successfully'
             nfev: 6
              nit: 2
             njev: 2
           status: 0
          success: True
                x: array([-0.0303 ,  0.06667])

In [27]: V(res['x'], 0) ❽
Out[27]: 0.3636363636310989
```

```
In [28]: V(res['x'], 1) ❾
Out[28]: array([ 1.     ,  0.33333, -0.     ])
```

❶ Импорт функции `minimize` из модуля `scipy.optimize`.

❷ Определение ограничения в виде векторного неравенства на основе функции `lambda`: функция λ моделируется здесь как $\lambda(\phi) = V_1(\phi) - C_1$, для которой должно выполняться ограничение в виде неравенства $\lambda(\phi) \geq 0$.

❸ Оптимизируемая функция `lambda`.

❹ Исходное предположение оптимального решения (здесь играет не слишком важную роль).

❺ Метод минимизации, здесь — последовательный метод наименьших квадратов (Sequential Least Squares Programming, SLSQP).

❻ Определенные ранее ограничения для задачи по минимизации.

❼ Полный словарь с результатами минимизации с оптимальными параметрами x и минимальным значением функции `fun`.

❽ Значение оптимального суперреплицирующего портфеля сегодня.

❾ Будущая неопределенная стоимость оптимального суперреплицирующего портфеля. Оптимальный портфель, продающий облигацию без покрытия и приобретающий акцию, в двух состояниях с точностью реплицирует соответствующую выплату, а в промежуточном состоянии — суперреплицирует ее.

Аппроксимирующая репликация

Суперрепликация предполагает несколько экстремальную ситуацию: выплата по суперреплицируемому условному требованию должна быть одинакова или превышена в любом заданном состоянии при любых обстоятельствах. Например, компания по страхованию жизни инвестирует таким образом, чтобы иметь возможность при любых обстоятельствах (что в реальном мире часто обозначает «с вероятностью 99,9 %») выполнить свои будущие финансовые и договорные обязательства (условные требования). Однако во многих случаях суперрепликация может оказаться экономически нецелесообразным или даже нежизнеспособным вариантом.

Именно здесь в игру вступает *аппроксимация*, смысл которой заключается в том, чтобы реплицировать выплаты по условному требованию *«настолько*

хорошо, насколько это возможно» на фоне целевой функции, а затем минимизировать *погрешность репликации* с учетом торгуемых финансовых активов.

Возможным вариантом целевой функции или функции погрешности является средняя квадратичная погрешность (mean squared error, MSE). Пусть $V_1(\phi)$ — вектор стоимости портфеля репликации ϕ, тогда MSE для условного требования C_1 с учетом портфеля ϕ составляет:

$$\mathrm{MSE}\left(V_1(\varphi)-C_1\right)=\frac{1}{|\Omega|}\sum_{\omega\in\Omega}\left(V_1^{\omega}(\varphi)-C_1^{\omega}\right)^2.$$

Данное уравнение определяет величину, которую необходимо минимизировать. Для достижимого условного требования MSE равна нулю. Сама задача в матричной форме имеет вид:

$$\min_{\varphi}\mathrm{MSE}\left(\mathcal{M}\cdot\varphi-C_1\right).$$

В линейной алгебре эта задача относится к обычной регрессии МНК. Предыдущее уравнение является частным случаем задачи *линейной* регрессии МНК.

Стандартным и эффективным подходом к решению задач такого рода в Python является использование функции `np.linalg.lstsq` из библиотеки NumPy:

```
In [29]: M = np.array((B[1], S[1])).T ❶

In [30]: M ❶
Out[30]: array([[11, 20],
                [11, 10],
                [11,  5]])

In [31]: reg = np.linalg.lstsq(M, C1, rcond=-1) ❷

In [32]: reg
         # (array, ❸
         # array, ❹
         # int, ❺
         # array) ❻
Out[32]: (array([-0.04545,  0.07143]), array([0.07143]), 2, array([28.93836,
          7.11136]))

In [33]: V(reg[0], 0) ❼
Out[33]: 0.2597402597402598

In [34]: V(reg[0], 1) ❽
Out[34]: array([ 0.92857,  0.21429, -0.14286])
```

```
In [35]: V(reg[0], 1) - C1 ❾
Out[35]: array([-0.07143,  0.21429, -0.14286])

In [36]: np.mean((V(reg[0], 1) - C1) ** 2) ❿
Out[36]: 0.02380952380952381
```

❶ Матрица будущих цен двух торгуемых финансовых активов.

❷ Процесс решения задачи линейной регрессии МНК путем минимизации MSE.

❸ Оптимальные позиции портфеля, то есть решение задачи.

❹ Значение MSE, полученное в результате процедуры оптимизации (минимальная средняя квадратичная погрешность репликации).

❺ Ранг матрицы *M*...

❻ ...И ее сингулярные значения.

❼ Стоимость аппроксимирующего портфеля, которая меньше стоимости портфеля, созданного с учетом минимальных затрат.

❽ Доходность аппроксимирующего портфеля.

❾ Вектор погрешностей репликации.

❿ MSE аппроксимирующей репликации.

Аппроксимирующая репликация применима не во всех обстоятельствах, но инвестор или финансовый менеджер, обладающий свободой действий в процессе принятия решений, может выбрать именно ее, например, при слишком высокой стоимости суперрепликации.

Линия рынка капитала

Предположим, что речь идет о средней дисперсии или, скорее, средней волатильности. В дальнейшем под рисковой акцией мы будем иметь в виду *рыночный портфель*, который можно представить в виде широкого фондового индекса наподобие S&P 500.

Как и раньше, агенты могут составлять портфели из облигации и рыночного портфеля. Ставка доходности облигации — безрисковая процентная ставка — составляет $i = 0{,}1$, а волатильность равна 0. Ожидаемая ставка доходности рыночного портфеля:

$$\mu_S = \frac{\mathbf{E}^P(S_1)}{S_0} - 1 = \frac{7}{6} - 1 = \frac{1}{6}.$$

Его волатильность:

$$\sigma_S = \sqrt{\mathbf{E}^P\left(\left(\frac{S_1 - S_0}{S_0} - \mu_S\right)^2\right)}.$$

Быстрые вычисления в Python дают соответствующие числовые значения:

```
In [37]: mu_S = 7 / 6 - 1 ❶

In [38]: mu_S ❶
Out[38]: 0.16666666666666674

In [39]: sigma_S = (S[1] / S[0]).std() ❷

In [40]: sigma_S ❷
Out[40]: 0.6236095644623235
```

❶ Ожидаемая доходность рыночного портфеля.

❷ Волатильность доходности рыночного портфеля[1].

Возможные средние значения для нормализованного портфеля, состоящего из облигации и рыночного портфеля без коротких продаж с общим весом 1, или 100 %, находятся в диапазоне от 0 до ~0,166. Что касается волатильности, возможны значения от 0 до ~0,623.

С учетом доступности продаж без покрытия на рис. 3.2 показана *линия рынка капитала* (capital market line, CML), возникающая при различных составах портфеля. Поскольку разрешена продажа облигации без покрытия, возможны комбинации «риск — доходность» на верхней линии (с положительным наклоном), что в целом является *той самой* линией рынка капитала. Нижняя линия (с отрицательным наклоном) здесь неважна, поскольку портфели, являющиеся результатом коротких позиций в рыночном портфеле, имеют более низкие ожидаемые ставки доходности при том же риске, что и портфели с соответствующей длинной позицией в рыночном портфеле:

```
In [41]: s = np.linspace(-2, 2, 25) ❶

In [42]: b = (1 - s) ❷
```

[1] Обратите внимание на то, что данный расчет волатильности действителен только в том случае, если предполагается равная вероятность материализации трех состояний.

```
In [43]: i = 0.1 ❸

In [44]: mu = b * i + s * mu_S ❹

In [45]: sigma = np.abs(s * sigma_S) ❺

In [46]: plt.figure(figsize=(10, 6))
         plt.plot(sigma, mu) ❻
         plt.xlabel('$\sigma$')
         plt.ylabel('$\mu$');
```

❶ Позиции рыночного портфеля принимают значения от –200 до 200 %.

❷ Позиция портфеля облигации заполняет до 100 % общего веса портфеля.

❸ Безрисковая процентная ставка.

❹ Ожидаемые ставки доходности портфеля.

❺ Волатильность портфеля.

❻ Построение графика CML для коротких и длинных позиций в рыночном портфеле.

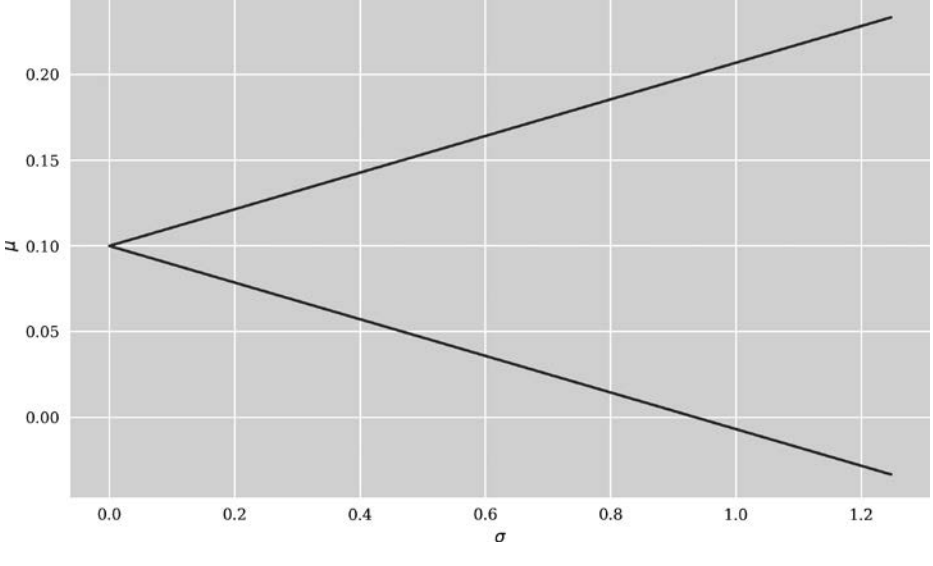


Рис. 3.2. Линия рынка капитала (верхняя, возрастающая часть)

Описать верхнюю, возрастающую часть линии рынка капитала можно с помощью уравнения:

$$\mu = i + \frac{\mu_S - i}{\sigma_S} \cdot \sigma.$$

Модель ценообразования капитальных активов

В 1964 году У. Шарп разработал *модель ценообразования капитальных активов* (Capital Asset Pricing Model, CAPM) — модель равновесного ценообразования, которая соотносит ожидаемую ставку доходности произвольного финансового актива или портфеля и его волатильность с ожидаемой нормой доходности и волатильностью рыночного портфеля.

Большое значение здесь имеет *корреляция* между ставками доходности финансового актива и рыночного портфеля. В дополнение к рыночному портфелю рассмотрим другой рисковый финансовый актив с процессом ценообразования $T = (T_0, T_1)$. Корреляция ρ определяется как:

$$\rho_{ST} = \frac{\mathbf{E}^P\left(\left(r_1^S - \mu_S\right) \cdot \left(r_1^T - \mu_T\right)\right)}{\sigma_S \cdot \sigma_T},$$

где $-1 \le \rho_{ST} \le 1$. При положительной корреляции два финансовых актива имеют тенденцию к движению в одном направлении, при отрицательной — в противоположных. В особом случае, когда положительная корреляция идеальна, то есть $\rho_{ST} = 1$, два финансовых актива, например портфели, лежащие на CML, ставки доходности которых представляют собой ставки доходности рыночного портфеля, увеличенные на вес рыночного портфеля, всегда движутся в одном направлении. Неопределенность здесь возникает только из-за рыночного портфеля, поэтому от изменения его веса будет зависеть степень риска.

Рассмотрим финансовый актив T, представляющий собой часть рыночного портфеля, для которого корреляция с рыночным портфелем неидеальна. Его ожидаемая ставка доходности определяется следующим уравнением:

$$\mu_T = i + \frac{\left(\mu_S - i\right) \cdot \rho_{ST}}{\sigma_S} \cdot \sigma_T = i + \frac{\rho_{ST} \cdot \sigma_S \cdot \sigma_T}{\sigma_S^2} \cdot \left(\mu_S - i\right).$$

Легко проверить, что корреляция между рыночным портфелем и облигацией равна нулю. Значит, если финансовый актив T — это безрисковая облигация, то ее ожидаемой доходностью будет безрисковый процент.

Ковариация между S_1 и T_1 определяется по формуле $\sigma_{ST} = \rho_{ST} \cdot \sigma_S \cdot \sigma_T$, вследствие чего:

$$\mu_T = i + \frac{\sigma_{ST}}{\sigma_S^2} \cdot (\mu_S - i) = i + \beta_T \cdot (\mu_S - i)$$

при

$$\beta_T \equiv \frac{\sigma_{ST}}{\sigma_S^2}.$$

В результате получается знаменитая линейная зависимость CAPM между ожидаемой ставкой доходности рыночного портфеля и ожидаемой доходностью финансового актива T или любого другого. В CAPM ставка доходности любого финансового актива определяется только через ожидаемую избыточную доходность рыночного портфеля, превышающую безрисковую процентную ставку и β-коэффициент, что представляет собой ковариацию между этими двумя показателями, увеличенную на квадрат волатильности рыночного портфеля (дисперсии доходности).

В CAPM линия рынка капитала заменяется *линией рынка ценных бумаг* (security market line, SML), которая строится в пространстве β-доходности с помощью следующего кода (график представлен на рис. 3.3):

```
In [47]: beta = np.linspace(0, 2, 25) ❶

In [48]: mu = i + beta * (mu_S - i) ❷

In [49]: plt.figure(figsize=(10, 6))
         plt.plot(beta, mu, label='security market line') ❸
         plt.xlabel('$\\beta$')
         plt.ylabel('$\mu$')
         plt.ylim(0, 0.25) ❹
         plt.plot(1, mu_S, 'ro', label='market portfolio') ❺
         plt.legend(loc=0);
```

❶ Генерация объекта `ndarray` со значениями `beta`.

❷ Расчет ожидаемой доходности `mu` в соответствии с CAPM.

❸ Построение графика комбинаций `beta-mu`.

❹ Настройка границ для оси *Y*.

❺ Построение графика со значениями β и ожидаемой доходности рыночного портфеля.

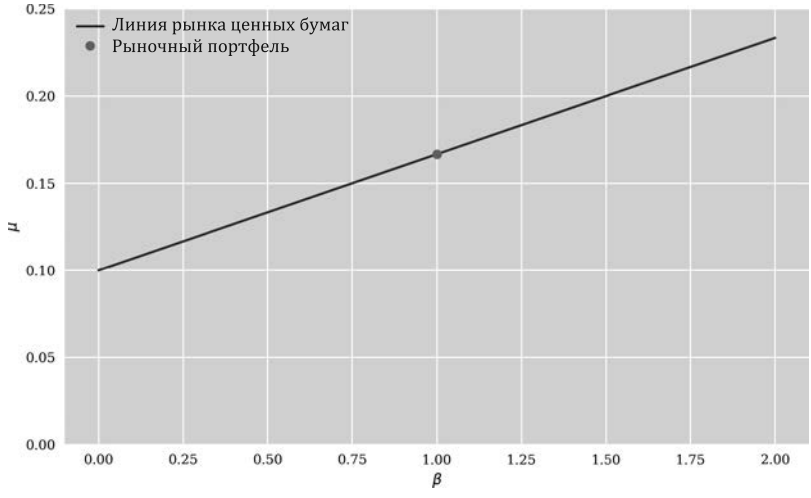


Рис. 3.3. Линия рынка ценных бумаг

Тем не менее остается вопрос: почему предыдущие уравнения, в частности формула CAPM, вообще верны? Чтобы ответить на него, рассмотрим портфель с весом 1, или 100 %, доля которого *a* инвестирована в финансовый актив *T*, а оставшаяся часть 1 – *a* — в рыночный портфель. Ожидаемая ставка доходности этого портфеля составляет:

$$\mu(a) = a \cdot \mu_T + (1-a) \cdot \mu_S.$$

Его волатильность, согласно Шарпу:

$$\sigma(a) = \left(a^2 \cdot \sigma_T^2 + (1-a)^2 \cdot \sigma_S^2 + 2a(1-a) \cdot \sigma_{ST}\right)^{1/2}.$$

Незначительное изменение ожидаемой ставки доходности портфеля при незначительном изменении распределения финансового актива *T* определяется следующей частной производной:

$$\frac{\partial \mu}{\partial a} = \mu_T - \mu_S.$$

Незначительное изменение волатильности портфеля при незначительном изменении распределения финансового актива T определяется следующей частной производной:

$$\frac{\partial\sigma}{\partial a}=\frac{a\cdot\sigma_T^2-\sigma_S^2+a\cdot\sigma_S^2+\sigma_{ST}-2a\cdot\sigma_{ST}}{\sqrt{a^2\cdot\sigma_T^2+(1-a)^2\cdot\sigma_S^2+2a\cdot(1-a)\cdot\sigma_{ST}}}.$$

Основная идея CAPM как *равновесной модели* заключается в том, что финансовый актив T уже является частью рыночного портфеля. Поэтому долю a можно считать здесь только как *избыточный спрос* на финансовый актив, а в условиях равновесия, когда избыточный спрос на все финансовые активы равен *нулю*, доля a также равна нулю. Следовательно, в условиях равновесия частная производная для ожидаемой ставки доходности портфеля остается неизменной, в то время как для волатильности портфеля она значительно упрощается при вычислении в точке равновесия, где $a = 0$:

$$\left.\frac{\partial\sigma}{\partial a}\right|_{a=0}=\frac{1}{2}\sigma_S\cdot\left(-2\sigma_S^2+2\sigma_{ST}\right)=\sigma_S\cdot\sigma_{ST}-\sigma_S\cdot\sigma_S^2=\frac{\sigma_{ST}-\sigma_S^2}{\sigma_S}.$$

Таким образом, при рыночном равновесии *соотношение между риском и доходностью* имеет следующий вид:

$$\left.\frac{\frac{\partial\mu}{\partial a}}{\frac{\partial\sigma}{\partial a}}\right|_{a=0}=\frac{\mu_T-\mu_S}{\frac{\sigma_{ST}-\sigma_S^2}{\sigma_S}}.$$

Последняя мысль, необходимая для выведения *формулы CAPM*, заключается в том, что в условиях равновесия предыдущее выражение должно равняться наклону линии рынка капитала:

$$\frac{\mu_T-\mu_S}{\frac{\sigma_{ST}-\sigma_S^2}{\sigma_S}}=\frac{\mu_S-i}{\sigma_S}\Leftrightarrow\mu_T=i+\frac{\sigma_{ST}}{\sigma_S^2}\cdot(\mu_S-i)=i+\beta_T\cdot(\mu_S-i).$$

Как CAPM помогает в ценообразовании нереплицируемого финансового актива? С учетом вектора неопределенных цен через год финансового актива T_1 неопределенные ставки доходности составляют:

$$r^T=\frac{T_1-T_0}{T_0},$$

где T_0 — равновесная цена сегодня, которую необходимо определить. Помимо этого, выполняется следующее соотношение:

$$\mu_T = \frac{\mathbf{E}^P(T_1) - T_0}{T_0}.$$

На основе CAPM теперь можно определить *цену риска* как избыточную доходность рыночного портфеля на единицу дисперсии через:

$$\lambda = \frac{\mu_S - i}{\sigma_S^2},$$

результатом чего является:

$$\mu_T = i + \lambda \cdot \sigma_{ST}.$$

Таким образом, в условиях равновесия ожидаемая ставка доходности финансового актива определяется только его ковариацией с рыночным портфелем. Если приравнять ее к предыдущему выражению, вычисляющему ожидаемую доходность финансового актива T, то получается:

$$\frac{\mathbf{E}^P(T_1) - T_0}{T_0} = i + \lambda \cdot \sigma_{ST} \Leftrightarrow T_0 = \frac{\mathbf{E}^P(T_1)}{1 + i + \lambda \cdot \sigma_{ST}}.$$

Знаменатель здесь является *коэффициентом дисконтирования с поправкой на риск*.

MVP, CAPM и полнота рынка

И MVP, и CAPM опираются только на статистику высокого уровня: математическое ожидание, волатильность и ковариацию. При репликации условных требований, например, важную роль играет каждая отдельная выплата в каждом возможном будущем состоянии (с представленными эффектами, возникающими вследствие неполноты рынка). В MVP и CAPM (неявно) предполагается, что инвесторов волнует только совокупная статистика, а не каждое отдельное состояние.

Но рассмотрим, например, два финансовых актива с одинаковой начальной стоимостью при равной вероятностной мере. Выплата 20 денежных единиц по первому активу происходит только в одном определенном состоянии, по второму — только в другом определенном состоянии. Оба финансовых актива имеют одинаковые характеристики «риск — доходность», однако агент может предпочесть один из них другому, несмотря на то что их статистические характеристики одинаковы. При выборе актива ориентация на его риск и доходность часто является удобным, но не всегда уместным упрощением.

Резюме

Основной темой главы стала неполнота финансовых рынков, которая сразу же образуется, если в экономику с двумя состояниями добавляется третье состояние, а количество торгуемых финансовых активов остается прежним (два). В этой ситуации, как правило, невозможно идеально реплицировать условные требования портфелями, состоящими из безрисковой облигации и рисковой акции.

Однако портфель может суперреплицировать условное требование, если при любых обстоятельствах выплата по нему больше выплаты по данному условному требованию или равна ей. Как правило, существует неограниченное количество суперреплицирующих портфелей, однако наилучшим вариантом является тот, который можно составить с наименьшими затратами. При отсутствии жестких рамок можно также аппроксимировать выплаты по недостижимым условным требованиям. В таком случае задача репликации заменяется оптимизационной задачей, в которой минимизируется средняя квадратичная погрешность реплицирующего портфеля. Математически аппроксимация сводится к задаче линейной регрессии МНК.

Модель ценообразования капитальных активов основана на аргументах равновесного ценообразования для выведения ожидаемых ставок доходности, а также цен на торгуемые финансовые активы с учетом их характеристик «риск — доходность» в пространстве средней волатильности. Центральную роль в ней играют корреляция и ковариация финансового актива с рыночным портфелем, который, как предполагается, содержит сам финансовый актив.

Справочные материалы

В этой главе были упомянуты следующие книги и статьи.

- *Pliska S.* Introduction to Mathematical Finance. — Malden and Oxford: Blackwell Publishers, 1997.
- *Sharpe W.* Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk // The Journal of Finance, 1964. — № 19 (3). — P. 425–442.

ГЛАВА 4

Оптимальность и равновесие

> Большая часть экономической теории основана на том, что при наличии двух альтернатив агент может выбрать и, если будет возможно, выберет лучшую для себя.
>
> *Даррелл Дюффе (1988)*

> Анализ портфеля начинается с изучения информации об отдельных ценных бумагах и заканчивается выводами относительно портфелей в целом. Задачей такого анализа является нахождение портфелей, которые наилучшим образом отвечают целям инвестора.
>
> *Гарри Марковиц (1959)*

Данная глава посвящена моделированию поведения и предпочтений агента и оптимизационных задач, с которыми он сталкивается. Здесь представлены некоторые фундаментальные элементы микроэкономической теории (см.: Вариан, 1992) и финансовой экономики (см.: Айхбергер и Харпер, 1997). В основе ее содержания лежит парадигма *максимизации ожидаемой полезности*, которая чаще всего используется при моделировании предпочтений агента в финансовой экономике. На базе этой парадигмы обсуждаются две центральные темы.

Сначала поговорим о том, как агент выбирает оптимальный инвестиционный портфель с учетом своих предпочтений и первоначального капитала. При решении этой задачи, которая обычно называется *выбором оптимального портфеля*, мы не будем опираться на какое-либо упрощение, как при портфельной теории Марковица и модели ценообразования капитальных активов (CAPM), которые

сводят проблему выбора инвестиционных портфелей к первому и второму моментам распределения доходности финансовых активов, а также их ковариации.

Если в двух предыдущих главах цены на финансовые активы были заданы априори, то здесь они будут выводиться из фундаментальных принципов, поскольку анализ *ценообразования финансовых активов* станет выполняться на основе задачи на оптимизацию так называемого *репрезентативного агента* и *аргументов равновесия*. Репрезентативный агент выступает здесь как совокупность неограниченно большого количества агентов, действующих на финансовых рынках независимо друг от друга. Условия существования такого репрезентативного агента хорошо известны (см. главу 6), однако здесь мы не будем их рассматривать, поскольку в целом финансовая теория принимает факт существования агента как данность.

Со стороны Python вводится всего пару новых элементов. Основные инструменты для работы с финансами в дискретных моделях уже были представлены в предыдущих двух главах.

Финансы	Математика	Python
Предпочтения и полезность	Функция полезности	NumPy
Максимизация полезности	Целевая функция, бюджетное ограничение, теорема Лагранжа	`scipy.optimize.minimize`
Кривые безразличия, прямая бюджетного ограничения	Функция	NumPy, `matplotlib`
Логарифмическая полезность	Натуральный логарифм	NumPy, `matplotlib`
Аддитивная полезность относительно времени	Функция полезности	NumPy, `scipy.optimize.minimize`
Аддитивная ожидаемая полезность относительно времени	Вероятностная мера, теорема Лагранжа	NumPy, `scipy.optimize.minimize`
Оптимальный инвестиционный портфель	Теорема Лагранжа, условия первого порядка	NumPy, `scipy.optimize.minimize`
Ценообразование в условиях равновесия, репрезентативный агент	Теорема Лагранжа, условия первого порядка	NumPy, `scipy.optimize.minimize`, SumPy
Мартингальные меры в условиях неполного рынка	Множество вероятностных мер	SymPy, `sy.Symbol`, `sy.solve`
Восполнение рынка условными требованиями	Теорема Лагранжа, условия первого порядка	NumPy, `scipy.optimize.minimize`

Максимизация полезности

Формально агентное моделирование осуществляется с помощью *функции полезности*, упорядочивающей варианты выбора, с которыми сталкивается агент, и являющейся представлением его *предпочтений* (см. главу 7). Рассмотрим статическую экономику без неопределенности из главы 2, а также предположим, что агент наделен некоторым первоначальным капиталом $w \in \mathbb{R}_{>0}$. Агент может решить, какую часть своего капитала потратить сегодня ($t = 0$), а какую — сберечь для будущего потребления, открыв банковский депозит. Представим человека, перед которым встал вопрос: сколько откладывать на пенсию?

Выяснить, какую пользу приносят агенту его деньги сегодня (c_0) и через год (c_1), можно с помощью соответствующей функции:

$$U: \mathbb{R}^2_{\geq 0} \to \mathbb{R}_{\geq 0},\ (c_0, c_1) \mapsto u(c_0, c_1).$$

В качестве примера предположим, что $u(c_0, c_1) = c_0 \cdot c_1$ говорит о том, что деньги сегодня и через год являются субститутами, хотя и не совершенными (если стоимость денег в любой момент времени равна нулю, то и полезность равна нулю). Каков оптимальный план накопления для агента? Формально задача на условный экстремум имеет вид:

$$\max_{c_0, c_1} c_0 \cdot c_1,$$

с ограничением:

$$c_0 + c_1 = w.$$

Согласно теореме Лагранжа (см. главу 5), задача на условный экстремум может быть преобразована в задачу безусловного экстремума:

$$\max_{c_0, c_1, \lambda} f(c_0, c_1, \lambda) = c_0 \cdot c_1 - \lambda \cdot (c_0 + c_1 - w).$$

Необходимыми условиями оптимальности первого порядка являются:

$$\begin{cases} \dfrac{\partial f}{\partial c_0} = c_1 - \lambda = 0; \\ \dfrac{\partial f}{\partial c_1} = c_0 - \lambda = 0; \\ \dfrac{\partial f}{\partial \lambda} = c_0 + c_1 - w = 0. \end{cases}$$

Из данных условий легко вывести оптимальный план экономии — $c_0 = c_1 = w/2$.

В Python такую задачу на оптимизацию, где, например, $w = 10$, можно смоделировать и решить в числовой форме следующим образом:

```
In [1]: def u(c):
            return -c[0] * c[1] ❶

In [2]: w = 10 ❷

In [3]: from scipy.optimize import minimize

In [4]: cons = ({'type': 'eq', 'fun': lambda c: c[0] + c[1] - w}) ❸

In [5]: opt = minimize(u, (1, 1), constraints=cons) ❹

In [6]: opt
Out[6]:      fun: -24.999999999999996
             jac: array([-5., -5.])
         message: 'Optimization terminated successfully'
            nfev: 6
             nit: 2
            njev: 2
          status: 0
         success: True
               x: array([5., 5.])

In [7]: opt['x'] ❺
Out[7]: array([5., 5.])

In [8]: -opt['fun'] ❻
Out[8]: 24.999999999999996
```

❶ Функция полезности с отрицательным знаком для достижения максимизации через минимизацию.

❷ Первоначальный капитал агента, который необходимо распределить между двумя моментами времени — «сегодня» и «через год».

❸ Бюджетное ограничение в виде ограничения типа равенства для выполнения функции `minimize`.

❹ Экстремум с исходным предположением и бюджетным ограничением.

❺ Оптимальный план экономии.

❻ Максимальная полезность, получаемая по оптимальному плану.

Кривые безразличия

Решение задачи на оптимальность можно визуализировать с помощью *кривых безразличия*. Кривую безразличия образуют все комбинации $c = (c_0, c_1)$, которые дают одинаковую полезность $\overline{u}$. В пространстве (c_0, c_1) ее можно описать с помощью уравнения:

$$\overline{u} = c_0 \cdot c_1 \Leftrightarrow c_1 = \frac{\overline{u}}{c_0},$$

а прямую *бюджетного ограничения* —

$$w = c_0 + c_1 \Leftrightarrow c_1 = w - c_0.$$

Графически задача на оптимизацию представлена на рис. 4.1. На нем оптимальный план обозначен точкой, в которой кривая безразличия для $\overline{u} = 25$ пересекается с прямой бюджетного ограничения.

В Python данный график строится следующим образом:

```
In [9]: def iu(u, c0):
            return u / c0 ❶

In [10]: def c1(c0):
             return w - c0 ❷

In [11]: import numpy as np
         np.set_printoptions(precision=5)

In [12]: from pylab import mpl, plt
         plt.style.use('seaborn')
         mpl.rcParams['savefig.dpi'] = 300
         mpl.rcParams['font.family'] = 'serif'

In [13]: c0 = np.linspace(1, w) ❸

In [14]: plt.figure(figsize=(10, 6))
         plt.plot(c0, c1(c0), label='budget constraint', lw=3.0)
         plt.plot(c0, iu(15, c0), '--', label='$u=15$')
         plt.plot(c0, iu(25, c0), label='$u=25$')
         plt.plot(c0, iu(35, c0), '-.', label='$u=35$')
         plt.plot(opt['x'][0], opt['x'][1], 'ro', label='$c=(5, 5)$')
         plt.legend(loc=0);
```

❶ Функция кривой безразличия.

❷ Функция линии бюджетного ограничения.

❸ Область, на которой строятся оба графика.

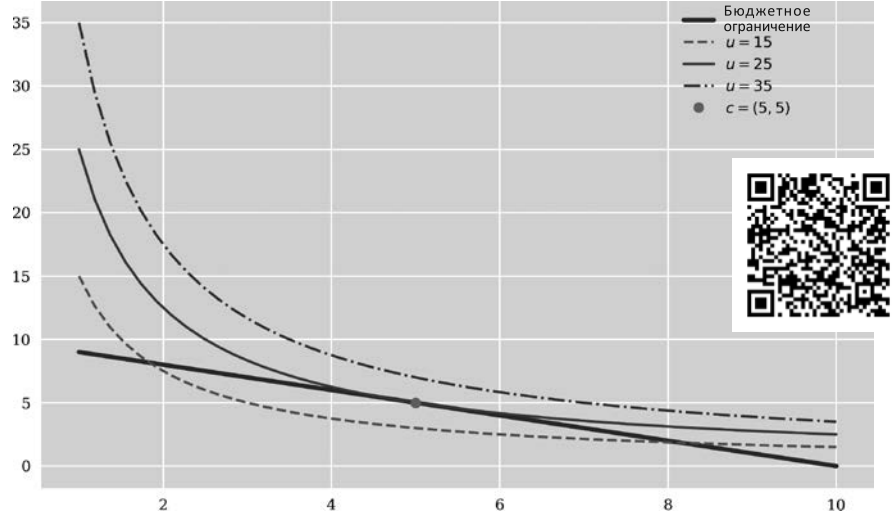


Рис. 4.1. Задача на максимизацию полезности

Условия функции полезности

В финансах польза, которую агент получает от денег, имеющихся у него в определенный момент времени (например, в качестве субститута реального актива, который можно купить на эти деньги), обычно выражается в виде функции u: $\mathbb{R}_{\geq 0} \to \mathbb{R}$, удовлетворяющей, как предполагается, трем условиям.

1. $u(x)$ — это дважды дифференцируемая функция.
2. $\dfrac{du}{dx} > 0.$
3. $\dfrac{d^2u}{dx^2} \leq 0.$

Первое условие — это необходимое требование для двух других. Второе выражает мысль о том, что при прочих равных условиях больше денег лучше, чем меньше денег. Здесь предполагается бесконечность спроса агентов. Согласно третьему условию предельная полезность каждой следующей денежной единицы меньше (или такая же в максимальной точке) предельной полезности предыдущей денежной единицы. При этом предполагается, что функция является возрастающей и квазивогнутой.

Логарифмическая полезность

В этом разделе представлена функция, которая хорошо подходит для финансового анализа, основанного на максимизирующем полезность агенте, — *натуральный логарифм* $u(x) = \ln x$. Она удовлетворяет трем условиям, приведенным в предыдущем подразделе, и регулярно используется в финансах для моделирования пользы, которую агент получает от денег (или потребления). При условии, что $x \in \mathbb{R}_{>0}$, получается следующее.

1. $\frac{du}{dx} = \frac{1}{x} > 0$ для $x \in \mathbb{R}_{>0}$.
2. $\frac{d^2u}{dx^2} = -\frac{1}{x^2} < 0$ для $x \in \mathbb{R}_{>0}$.

Python позволяет графически изобразить три рассмотренные нами функции посредством библиотеки NumPy в сочетании с векторными вычислениями. На рис. 4.2 показан график, сгенерированный следующим кодом:

```
In [15]: x = np.linspace(0.5, 10, 50) ❶

In [16]: x[:5] ❷
Out[16]: array([0.5    , 0.69388, 0.88776, 1.08163, 1.27551])

In [17]: u = np.log(x) ❸

In [18]: u1 = 1 / x ❹

In [19]: u2 = -1 / x ** 2 ❺

In [20]: plt.figure(figsize=(10, 6)) ❻
         plt.plot(x, u, label='$u$') ❼
         plt.plot(x, u1, '--', label='$du/dx$') ❽
         plt.plot(x, u2, '-.', label='$d^2u/dx^2$') ❾
         plt.legend(loc=0); ❿
```

❶ Создание объекта `ndarray` с числами с плавающей запятой от 0,5 до 10 и равномерным интервалом для получения 50 значений.

❷ Вывод выборки из полученных чисел.

❸ Вычисление значений для функции полезности.

❹ И для ее первой производной, а также...

❺ ...Для второй производной.

❻ Создание нового холста для построения графика и задание параметров размера.

❼ Нанесение на график функции полезности.

❽ Нанесение на график первой производной.

❾ Нанесение на график второй производной.

❿ Размещение условных обозначений в оптимальном месте (`loc=0`).

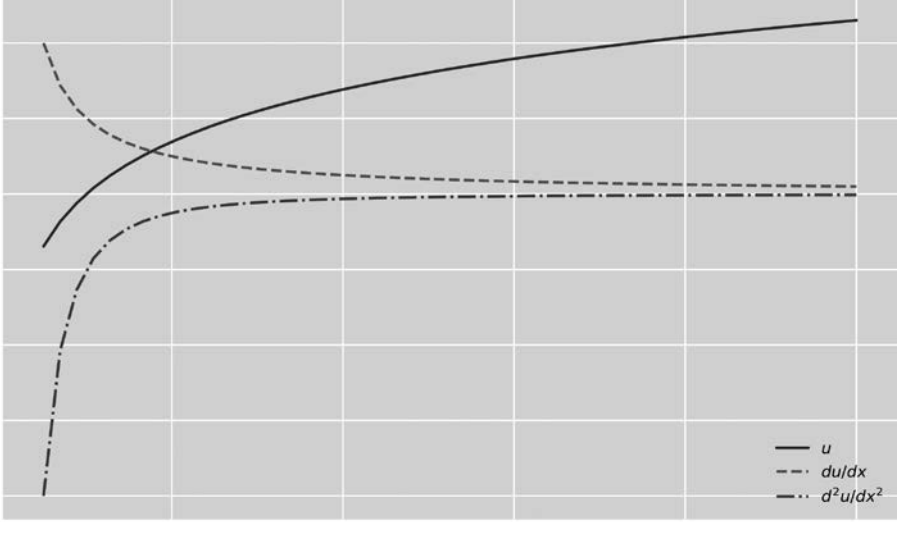


Рис. 4.2. Функция натурального логарифма и две его производные

Аддитивная полезность относительно времени

С учетом натурального логарифма, использованного в качестве функции для моделирования полезности денег для агента, предпочтения агента относительно планов экономии $c = (c_0, c_1)$ могут быть описаны как аддитивная функция полезности относительно времени следующего вида:

$$U: \mathbb{R}^2_{\geq 0} \to \mathbb{R},\ (c_0, c_1) \mapsto \ln c_0 + \kappa \cdot \ln c_1.$$

Предполагается, что $\kappa \in \mathbb{R}_{\geq 0}$ принимает значения $0 < \kappa \leq 1$ и представляет собой *временные предпочтения* агента. Суть данной функции в том, что деньги и потребление сегодня ценятся выше, чем через год: например, 100 долларов сейчас предпочтительнее 100 долларов через год независимо от того, какая точная функция описывает полезность (при условии постоянства предпочтений с течением времени). Ее также можно рассматривать как неденежный коэффициент дисконтирования. Вдобавок на основе частных производных по отношению к c_0 и c_1 легко убедиться, что эта функция удовлетворяет трем условиям, описанным ранее: она является дважды дифференцируемой, вогнутой и возрастающей.

При наличии у агента первоначального капитала w задача на условный экстремум имеет следующий вид:

$$\max_{c_0, c_1} \ln c_0 + \kappa \cdot \ln c_1,$$

таким образом:

$$c_0 + c_1 = w,$$

или:

$$\max_{c_0, c_1, \lambda} f(c_0, c_1, \lambda) = \ln c_0 + \kappa \cdot \ln c_1 - \lambda \cdot (c_0 + c_1 - w).$$

Необходимыми условиями оптимальности первого порядка в таком случае являются:

$$\begin{cases} \dfrac{\partial f}{\partial c_0} = \dfrac{1}{c_0} - \lambda = 0; \\ \dfrac{\partial f}{\partial c_1} = \kappa \cdot \dfrac{1}{c_1} - \lambda = 0; \\ \dfrac{\partial f}{\partial \lambda} = c_0 + c_1 - w = 0, \end{cases}$$

результатом которых становится:

$$\frac{1}{c_0} = \kappa \cdot \frac{1}{c_1} \Leftrightarrow c_1 = \kappa \cdot c_0.$$

Оптимальный план экономии теперь отражает временные предпочтения в том, что потребление через год c_1 — это $\kappa \cdot c_0$. Также верны равенства:

$$c_0 + \kappa \cdot c_0 = w \Leftrightarrow c_0 = \frac{w}{1+\kappa}$$

и

$$\frac{w}{1+\kappa} + c_1 = w \Leftrightarrow c_1 = \frac{\kappa \cdot w}{1+\kappa}.$$

Обязательным условием является бюджетное ограничение:

$$\frac{w}{1+\kappa} + \frac{\kappa \cdot w}{1+\kappa} = \frac{w + \kappa \cdot w}{1+\kappa} = w.$$

Следующий код решает задачу на оптимизацию в числовом виде при $w = 10$. Полученный оптимальный план отражает временные предпочтения агента:

```
In [21]: import math

In [22]: from scipy.optimize import minimize

In [23]: kappa = 10 / 11

In [24]: def U(c):
             return -(math.log(c[0]) +  kappa * math.log(c[1])) ❶

In [25]: w = 10

In [26]: cons = ({'type': 'eq', 'fun': lambda c: c[0] + c[1] - w})❷

In [27]: opt = minimize(U, (1, 1), constraints=cons)

In [28]: opt
Out[28]:      fun: -3.0747286083026886
              jac: array([-0.19091, -0.19091])
          message: 'Optimization terminated successfully'
             nfev: 18
              nit: 6
             njev: 6
           status: 0
          success: True
                x: array([5.23811, 4.76189])

In [29]: opt['x'] ❸
Out[29]: array([5.23811, 4.76189])

In [30]: -opt['fun'] ❹
Out[30]: 3.0747286083026886
```

❶ Функция полезности со знаком минус для достижения максимизации через минимизацию.

❷ Бюджетное ограничение в виде ограничения типа равенства для выполнения функции `minimize`.

❸ Оптимальный план экономии, отражающий временные предпочтения, при котором c_0 больше c_1 ровно на 10 %.

❹ Максимальная полезность, получаемая по оптимальному плану[1].

Ожидаемая полезность

Перейдем к статической экономике с двумя состояниями и *неопределенностью*. Предположим, что у агента есть некоторый первоначальный капитал $w \in \mathbb{R}_{>0}$, пользу от которого он получит только за счет денег, доступных через год. Полезность этих денег разнится в зависимости от того, какое из двух возможных состояний материализуется. Данная ситуация представляет собой задачу на чистую инвестицию, где весь имеющийся первоначальный капитал должен быть вложен в оптимальные торгуемые финансовые активы.

Допустим, что торгуются *два финансовых актива*: безрисковая облигация с процессом ценообразования:

$$B = \left(B_0, \left(B_1, B_1\right)^{\mathrm{T}}\right)$$

и рисковая акция с процессом ценообразования:

$$S = \left(S_0, \left(S_1^u, S_1^d\right)^{\mathrm{T}}\right).$$

Финансовые активы являются средством переноса первоначального капитала с сегодняшнего дня на более поздний момент времени. Основная проблема агента при принятии решения заключается в том, чтобы определиться, какая часть денежных средств должна быть расходована в любом из будущих состояний.

Модель инвестиционной задачи, с которой сталкивается агент в условиях неопределенности, задается *ожидаемой полезностью* для агента, которая должна

[1] Полезность необходимо понимать только через призму ординалов, то есть через приведение различных планов в определенный порядок. Сравнение этой числовой величины с оптимальной, выведенной ранее, не имеет никакого смысла, поскольку сами функции полезности различны.

быть максимизирована с учетом значения w. *Функция ожидаемой полезности* имеет вид:

$$U: \mathbb{R}^2_{\geq 0} \to \mathbb{R},\ c_1 \mapsto \mathbf{E}^P\left(u\left(c_1\right)\right).$$

С *вектором цен* $\mathcal{M} = \left(B_0, S_0\right)^{\mathbf{T}}$ агент может распределить свой первоначальный капитал, исходя из следующего условия:

$$\mathcal{M}_0 \cdot \varphi = w \Leftrightarrow B_0 \cdot b + S_0 \cdot s = w,$$

где $\varphi = \left(b, s\right)^{\mathbf{T}} \in \mathbb{R}^2_{\geq 0}$ — составленный агентом *портфель*, содержащий безрисковую облигацию и рисковую акцию. Данное *бюджетное ограничение* всегда будет обязательным условием из-за бесконечности спроса агента. Помимо этого, запрещены продажи без покрытия («короткие» продажи).

Матрица рыночных выплат представлена как:

$$\mathcal{M} = \begin{pmatrix} B_1 & S_1^u \\ B_1 & S_1^d \end{pmatrix}.$$

Сколько денег будет у агента в любом из состояний через год? Сумма определяется составленным им портфелем:

$$c_1 = \mathcal{M} \cdot \varphi \begin{pmatrix} B_1 \\ B_1 \end{pmatrix} \cdot b + \begin{pmatrix} S_1^u \\ S_1^d \end{pmatrix} \cdot s,$$

который можно преобразовать в:

$$c_1 = \begin{pmatrix} b \cdot B_1 + s \cdot S_1^u \\ b \cdot B_1 + s \cdot S_1^d \end{pmatrix}$$

или

$$\begin{cases} c_1^u = b \cdot B_1 + s \cdot S_1^u; \\ c_1^d = b \cdot B_1 + s \cdot S_1^d. \end{cases}$$

Полная задача принятия агентом решений касательно выбора оптимального портфеля может быть представлена в виде *задачи условного экстремума*:

$$\begin{gathered} \max_{c_1} \mathbf{E}^P\left(u\left(c_1\right)\right); \\ w = \mathcal{M}_0 \cdot \varphi; \\ c_1 = \mathcal{M} \cdot \varphi, \end{gathered}$$

которую можно упростить до:

$$\max_{\varphi} \mathbf{E}^{P}\left(u\left(\mathcal{M}\cdot\varphi\right)\right);$$
$$w = \mathcal{M}_0 \cdot \varphi.$$

Согласно теореме Лагранжа эту задачу можно преобразовать в *задачу безусловного экстремума*:

$$\max_{b,\,s,\,\lambda} f\left(b, s, \lambda\right) = \mathbf{E}^{P}\left(u\left(b\cdot B_1 + s\cdot S_1\right)\right) - \lambda\cdot\left(b\cdot B_0 + s\cdot S_0 - w\right),$$

где агент выбирает b и s для максимизации ожидаемой полезности с учетом бюджетного ограничения.

Теория ожидаемой полезности

Спустя десятилетия после разработки и внедрения теория ожидаемой полезности (Expected Utility Theory, EUT) все еще остается доминирующей парадигмой принятия финансовых решений, несмотря на то что одно из ее основных допущений — агенты имеют полное представление о возможных будущих состояниях и вероятности их реализации — практически никогда не выполняется в реальности. Тем не менее для многих EUT является интеллектуально привлекательной теоремой с приятными результатами, которые часто легко понять и интерпретировать. Подробнее о проблемах данной парадигмы в финансах можно узнать у Хилпиша (2020, главы 3 и 4).

Оптимальный инвестиционный портфель

Как выглядит оптимальное решение для агента, стремящегося максимизировать ожидаемую полезность? Ответ в общих чертах можно дать на основе условий первого порядка, необходимых и достаточных для оптимального решения:

$$\begin{cases} \dfrac{\partial f}{\partial b} = 0; \\ \dfrac{\partial f}{\partial s} = 0; \\ \dfrac{\partial f}{\partial \lambda} = 0 \end{cases}$$

или

$$\frac{\partial f}{\partial b} = p \cdot B_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1^u\right) + \left(1-p\right) \cdot B_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1^d\right) - \lambda \cdot B_0 = 0$$

и

$$\frac{\partial f}{\partial s} = p \cdot S_1^u \cdot u'\left(b \cdot B_1 + s \cdot S_1^u\right) + \left(1-p\right) \cdot S_1^d \cdot u'\left(b \cdot B_1 + s \cdot S_1^d\right) - \lambda \cdot S_0 = 0,$$

где $u'\left(x\right) \equiv \frac{du}{dx}$, а также:

$$b \cdot B_0 + s \cdot S_0 = w.$$

При логарифмической полезности двух торгуемых финансовых активов с процессами ценообразования $B = (10, 11)$ и $S = (10, (20, 5)^{\mathrm{T}})$ и $w = 10$ получается, что $b + s = 1$, вследствие чего позиции портфеля представляют собой процентные значения.

В Python функция `minimize` в модуле `scipy.optimize` подходит и для решения инвестиционной задачи агента:

```
In [31]: B = (10, (11, 11)) ❶

In [32]: S = (10, (20, 5)) ❷

In [33]: M0 = np.array((B[0], S[0])) ❸

In [34]: M = np.array((B[1], S[1])).T ❹

In [35]: p = 0.5 ❺

In [36]: P = np.array((p, 1-p)) ❺

In [37]: def U(phi):
             c1 = np.dot(M, phi) ❻
             return -np.dot(P, np.log(c1)) ❻

In [38]: -U((1, 0)) ❼
Out[38]: 2.3978952727983707

In [39]: -U((0, 1)) ❼
Out[39]: 2.3025850929940455

In [40]: -U((0.5, 0.5)) ❼
Out[40]: 2.410140782802518
```

```
In [41]: w = 10

In [42]: cons = ({'type': 'eq',
                  'fun': lambda phi: np.dot(M0, phi) - w}) ❽

In [43]: opt = minimize(U, (1, 1), constraints=cons) ❾

In [44]: opt
Out[44]:      fun: -2.4183062699261972
              jac: array([-1. , -0.99999])
          message: 'Optimization terminated successfully'
             nfev: 15
              nit: 5
             njev: 5
           status: 0
          success: True
                x: array([0.69442, 0.30558])

In [45]: opt['x'] ❿
Out[45]: array([0.69442, 0.30558])

In [46]: -opt['fun'] ⓫
Out[46]: 2.4183062699261972

In [47]: -U(opt['x']) ⓫
Out[47]: 2.4183062699261972

In [48]: np.dot(M, opt['x']) ⓬
Out[48]: array([13.75022,  9.16652])
```

❶ Процесс ценообразования облигации...

❷ ...И акции.

❸ Вектор цен двух торгуемых финансовых активов.

❹ Матрица рыночных выплат по двум торгуемым финансовым активам.

❺ Физическая мера вероятности относительно экономики.

❻ Функция ожидаемой полезности вместе с логарифмической полезностью.

❼ Некоторые примерные значения для общего веса портфеля, равного единице, — диверсификация окупается.

❽ Бюджетное ограничение, основанное на скалярном произведении векторов цены и портфеля.

❾ Задача на максимизацию ожидаемой полезности в виде минимизации.

❿ Оптимальное распределение между облигацией и акцией.

⓫ Оптимальное значение ожидаемой полезности.

⓬ Зависящий от состояния экономики доход по оптимальному портфелю.

Аддитивная ожидаемая полезность относительно времени

Задачу принятия решений агентом можно сформулировать так, чтобы включить в нее полезность денег сегодня:

$$U: R_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R},\ \left(c_0, c_1\right) \mapsto u\left(c_0\right) + \kappa \cdot \mathbf{E}^P\left(u\left(c_1\right)\right).$$

При первоначальном капитале w задача на оптимизацию без ограничений приобретает вид:

$$\max_{c_0, b, s, \lambda} f\left(c_0, b, s, \lambda\right) = u\left(c_0\right) + \kappa \cdot \mathbf{E}^P\left(u\left(b \cdot B_1 + s \cdot S_1\right)\right) - \lambda \cdot \left(c_0 + b \cdot B_0 + s \cdot S_0 - w\right).$$

С учетом сделанных ранее предположений оптимальное решение выводится посредством следующего кода:

```
In [49]: M0 = np.array((1, B[0], S[0])) ❶

In [50]: kappa = 10 / 11 ❷

In [51]: def U(phi):
             c0 = phi[0] ❸
             c1 = np.dot(M, phi[1:]) ❸
             return -(np.log(c0) + kappa * np.dot(P, np.log(c1))) ❸

In [52]: opt = minimize(U, (1, 1, 1), constraints=cons)

In [53]: opt
Out[53]:      fun: -3.1799295980286093
              jac: array([-0.19088, -1.90932, -1.90974])
          message: 'Optimization terminated successfully'
             nfev: 32
              nit: 8
             njev: 8
           status: 0
          success: True
                x: array([5.23899, 0.33087, 0.14523])
```

```
In [54]: -opt['fun']
Out[54]: 3.1799295980286093

In [55]: opt['x'][0] ❹
Out[55]: 5.23898714830318

In [56]: np.dot(M, opt['x'][1:]) ❺
Out[56]: array([6.54422, 4.36571])
```

❶ Вектор цен, включающий единицу, на потребление сегодня.

❷ Фактор временных предпочтений.

❸ Функция ожидаемой полезности с учетом потребления сегодня и временных предпочтений.

❹ Это часть w, которую агент потребляет сегодня.

❺ Зависящий от состояния экономики доход по позиции облигации и акции.

Ценообразование в условиях полного рынка

Перейдем от анализа к назначению цен, основанному на принципах оптимизации. Предположим, что две ценные бумаги Эрроу — Дебре торгуются в экономике с двумя будущими состояниями и *чистое предложение* обеих ценных бумаг составляет единицу. Два вектора выплат образуют *стандартный базис* $\mathbb{R}^2$, а матрица рыночных выплат имеет вид:

$$\mathcal{M} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Допустим также, что в экономике только один *репрезентативный агент* торгует этими двумя ценными бумагами. В *условиях равновесия* он должен удерживать чистое предложение обеих ценных бумаг из-за отсутствия конкуренции. Механизмом, обеспечивающим равновесие, являются цены двух ценных бумаг сегодня, то есть вектор цен:

$$\mathcal{M}_0 = \left(\gamma^u, \gamma^d\right)^{\mathbf{T}} \in \mathbb{R}^2_{\geq 0}.$$

Согласно принципу равновесного ценообразования этот вектор цен должен быть скорректирован таким образом, чтобы репрезентативный агент удерживал чистое предложение всех доступных финансовых активов — в противном случае равновесия не будет.

При инвестиционном портфеле $\phi = (\phi^u, \phi^d)^T$ задача по максимизации ожидаемой полезности имеет вид:

$$\max_{\varphi} \mathbf{E}^P\left(u\left(\mathcal{M}\cdot\varphi\right)\right),$$

таким образом:

$$\mathcal{M}_0\cdot\varphi = w$$

или

$$\max_{\varphi,\,\lambda} \mathbf{E}^P\left(u\left(\mathcal{M}\cdot\varphi\right)\right) - \lambda\cdot\left(\mathcal{M}_0\cdot\varphi - w\right).$$

Из-за особой матрицы рыночных выплат это выражение можно преобразовать следующим образом:

$$\max_{\varphi^u,\,\varphi^d} p\cdot u\left(\varphi^u\right) + \left(1-p\right)\cdot u\left(\varphi^d\right),$$

таким образом:

$$\gamma^u\cdot\varphi^u + \gamma^d\cdot\varphi^d = w$$

и

$$\max_{\varphi^u,\,\varphi^d,\,\lambda} f\left(\varphi^u, \varphi^d, \lambda\right) = p\cdot u\left(\varphi^u\right) + \left(1-p\right)\cdot u\left(\varphi^d\right) - \lambda\cdot\left(\gamma^u\cdot\varphi^u + \gamma^d\cdot\varphi^d - w\right).$$

Задача безусловного экстремума имеет три условия первого порядка:

$$\begin{cases} \dfrac{\partial f}{\partial\varphi^u} = p\cdot u'\left(\varphi^u\right) - \lambda\cdot\gamma^u = 0; \\ \dfrac{\partial f}{\partial\varphi^d} = \left(1-p\right)\cdot u'\left(\varphi^d\right) - \lambda\cdot\gamma^d = 0; \\ \dfrac{\partial f}{\partial\lambda} = \gamma^u\cdot\varphi^u + \gamma^d\cdot\varphi^d - w = 0. \end{cases}$$

Как данные условия оптимальности влияют на цены $\mathcal{M}_0$? Первое условие касается относительной цены двух ценных бумаг Эрроу — Дебре:

$$\frac{\gamma^u}{\gamma^d} = \frac{p\cdot u'\left(\varphi^u\right)}{\left(1-p\right)\cdot u'\left(\varphi^d\right)}.$$

Относительная цена полностью определяется вероятностями реализации каждого из двух состояний и предельной полезностью, получаемой от потребления. В *условиях равновесия* $\phi^u = \phi^d = 1$ относительная цена определяется только вероятностной мерой:

$$\frac{\gamma^u}{\gamma^d} = \frac{p}{(1-p)}.$$

В результате добавления этого условия получается, что:

$$\gamma^u + \gamma^d = w.$$

В то время как относительная цена в данном случае определяется мерой вероятности, абсолютные цены определяются имеющимся первоначальным капиталом, так как интуитивно понятно, что при фиксированном характере чистого предложения двух ценных бумаг цены зависят от начального капитала.

Например, приведение первоначального капитала к единице, $w = 1$, устанавливает такие цены:

$$\gamma^u = 1 - \gamma^d,$$

чтобы в итоге прийти к равновесным ценам:

$$\begin{cases} \gamma^u = p; \\ \gamma^d = 1 - p - \end{cases}$$

или вектору равновесных цен $\mathcal{M}_0^* = (p, 1-p)^{\mathbf{T}}$.

Арбитражное ценообразование

А что насчет арбитражных цен условных требований с учетом вектора равновесных цен $\mathcal{M}_0^*$? На полных рынках каждое условное требование достижимо и цена любого их них $C_1 \in \mathbb{A} = \mathbb{R}^2_{\geq 0}$ задается как:

$$C_0 = \mathcal{M}_0^* \cdot C_1 = \gamma^u \cdot C_1^u + \gamma^d \cdot C_1^d.$$

Это связано с тем, что портфель репликации — не что иное, как доход, зависящий от состояния экономики, $\phi = C_1$, в отдельном случае двух ценных бумаг Эрроу — Дебре. Поэтому цены таких ценных бумаг также считаются *ценами, зависящими от состояния экономики*, поскольку они представляют цену за одну единицу валюты (потребления) в определенном состоянии.

Ценообразование по мартингалу

Как выглядит уникальная мартингальная мера для текущей экономики? Мартингальная мера Q преобразует все процессы дисконтирования цен торгуемых финансовых активов в мартингалы. В матричной форме это свойство выглядит следующим образом:

$$\mathcal{M}_0^* = \frac{1}{1+i} \cdot \mathbf{E}^Q(\mathcal{M}).$$

А в более развернутом виде:

$$\begin{cases} p \cdot (1+i) = q; \\ (1-p) \cdot (1+i) = 1-q. \end{cases}$$

Отсюда следует, что $i = 0$ и $q = p$. Физическая вероятностная мера сразу преобразует все вероятности в мартингалы, а цены двух ценных бумаг Эрроу — Дебре уравновешиваются таким образом, чтобы процессы дисконтирования цен являлись мартингалами.

Цена каждого достижимого условного требования $C_1 \in \mathbb{A}$ может быть установлена через обычное математическое ожидание, которое выражается физической вероятностной мерой в этом особом экономическом случае, связанном с репрезентативным агентом. Формально это означает:

$$C_0 = \mathbf{E}^P(C_1).$$

Безрисковая процентная ставка

Почему безрисковая процентная ставка в условиях равновесия равна нулю? Ответ довольно прост: потому что не существует безрискового финансового актива, который бы фиксировал другую процентную ставку. Рассмотрим *безрисковый финансовый актив*, приносящий единицу валюты в каждом состоянии, $B_1 = (1, 1)^{\mathrm{T}}$. *Арбитражная цена* данного финансового актива составляет:

$$B_0 = \mathcal{M}_0^* \cdot B_1 = p + (1-p) = 1$$

и дает основания полагать, что безрисковая процентная ставка $i = 0$. Любая другая цена $0 < B_0 < 1$, подразумевающая положительную безрисковую процентную ставку, также будет предполагать возможность арбитража.

Пример в числовом виде (часть 1)

Анализ ценообразования в условиях равновесия до сих пор опирался на ряд упрощающих предположений, которые позволяли получить изящные решения и простые расчеты. Рассмотрим теперь случай с более реалистичными предположениями в числовом виде, которые неоднократно использовались ранее.

В частности, предположим максимизацию ожидаемой полезности на основе логарифмической полезности для репрезентативного агента. Процесс ценообразования для безрисковой облигации составляет:

$$B = \left(B_0, (11, 11)^{\mathrm{T}}\right),$$

а для рисковой акции:

$$S = \left(S_0, (20, 5)^{\mathrm{T}}\right).$$

Матрица рыночных выплат имеет вид:

$$\mathcal{M} = \begin{pmatrix} 11 & 20 \\ 11 & 5 \end{pmatrix}.$$

Ко всему прочему, дана физическая мера вероятности $P = (p, (1 - p))^{\mathrm{T}}$, где $p = 1/3$. Чистое предложение безрисковой облигации и рисковой акции составляет $b = 1$ и $s = 1$ соответственно. Агент располагает первоначальным капиталом $w = 15$.

Задачей репрезентативного агента является:

$$\max_{\varphi,\,\lambda} f(\varphi, \lambda) = \mathbf{E}^P\left(u\left(\mathcal{M}\cdot\varphi\right)\right) - \lambda\cdot\left(\mathcal{M}_0\cdot\varphi - w\right)$$

или

$$\max_{b,\,s,\,\lambda} f(b, s, \lambda) = \mathbf{E}^P\left(u\left(b\cdot B_1 + s\cdot S_1\right)\right) - \lambda\cdot\left(b\cdot B_0 + s\cdot S_0 - w\right).$$

Три условия первого порядка:

$$\begin{cases} \dfrac{\partial f}{\partial b} = \mathbf{E}^P\left(B_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1\right)\right) - \lambda \cdot B_0 = 0; \\ \dfrac{\partial f}{\partial s} = \mathbf{E}^P\left(S_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1\right)\right) - \lambda \cdot S_0 = 0; \\ \dfrac{\partial f}{\partial \lambda} = b \cdot B_0 + s \cdot S_0 - w = 0. \end{cases}$$

В соответствии с условиями оптимальности и с учетом чистого предложения единицы для обоих финансовых активов, а также логарифмической полезности *относительная цена* двух финансовых активов составляет:

$$\frac{S_0}{B_0} = \frac{\mathbf{E}^P\left(S_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1\right)\right)}{\mathbf{E}^P\left(B_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1\right)\right)} = \frac{\mathbf{E}^P\left(\dfrac{S_1}{B_1 + S_1}\right)}{\mathbf{E}^P\left(\dfrac{B_1}{B_1 + S_1}\right)} \equiv \zeta.$$

Добавление бюджетного ограничения фиксирует не только *относительную цену* ζ, но и уровни абсолютных цен:

$$B_0 + \zeta \cdot B_0 = w \Leftrightarrow B_0 = \frac{w}{1 + \zeta}.$$

В Python эти расчеты преобразуются в простые векторные операции, выполняемые с помощью библиотеки NumPy:

```
In [57]: p = 1 / 3 ❶

In [58]: P = np.array((p, (1-p))) ❶

In [59]: B1 = np.array((11, 11))

In [60]: S1 = np.array((20, 5))

In [61]: zeta = np.dot(S1 / (B1 + S1), P) / np.dot(B1 / (B1 + S1), P) ❷

In [62]: zeta ❷
Out[62]: 0.7342657342657343

In [63]: w = 15 ❸

In [64]: B0 = w / (1 + zeta) ❹
```

```
In [65]: B0 ❹
Out[65]: 8.649193548387098

In [66]: S0 = zeta * B0 ❺

In [67]: S0 ❺
Out[67]: 6.350806451612904

In [68]: B0 + S0 ❻
Out[68]: 15.000000000000002

In [69]: i = B1.mean() / B0 - 1❼

In [70]: i ❼
Out[70]: 0.2717948717948717

In [71]: mu = np.dot(S1, P) / S0 - 1 ❽

In [72]: mu ❽
Out[72]: 0.5746031746031743
```

❶ Вероятностная мера.

❷ Ценовой коэффициент ζ (дзета) с учетом условий оптимальности.

❸ Первоначальный капитал.

❹ Уровень равновесной цены безрисковой облигации с учетом коэффициента цен ζ и первоначального капитала w.

❺ Уровень равновесной цены рисковой акции.

❻ Бюджетное ограничение как обязательное условие.

❼ Равновесная процентная ставка с учетом уровня цены безрисковой облигации.

❽ Равновесная ожидаемая ставка доходности рисковой акции.

В этом примере равновесное ценообразование не приводит к «мартингализации» процессов дисконтирования цен при физической вероятностной мере. Однако вывести здесь *мартингальную меру* не составит труда — для этого понадобится библиотека SymPy для символьных вычислений в Python:

```
In [73]: import sympy as sy ❶

In [74]: q = sy.Symbol('q') ❷
```

```
In [75]: eq = (q * 20 + (1 - q) * 5) / (1 + i) - S0 ❸

In [76]: eq ❹
Out[76]: 11.7943548387097*q - 2.41935483870968

In [77]: q = sy.solve(eq)[0] ❺

In [78]: q ❻
Out[78]: 0.205128205128205

In [79]: Q = np.array((q, 1 - q)) ❻

In [80]: np.dot(B1, Q) / (1 + i) ❼
Out[80]: 8.64919354838710

In [81]: np.dot(S1, Q) / (1 + i) ❼
Out[81]: 6.35080645161290
```

❶ Импорт библиотеки символьных вычислений SymPy.

❷ Определение символа q.

❸ Формулировка уравнения для q с учетом условия мартингала.

❹ Упрощение уравнения.

❺ Решение уравнения в числовой форме.

❻ Полученная в результате решения мартингальная мера.

❼ Оба процесса дисконтирования цен являются мартингалами Q.

Ценообразование в условиях неполного рынка

Как работает ценообразование относительно репрезентативного агента на *неполных рынках*? К счастью, точно так же, как и на полных рынках.

Рассмотрим экономику с двумя датами и тремя состояниями, в которой торгуются безрисковая облигация с процессом ценообразования:

$$B = \left(B_0, (11, 11, 11)^{\mathrm{T}}\right)$$

и рисковая акция с процессом ценообразования:

$$S = \left(S_0, (20, 10, 5)^{\mathrm{T}}\right).$$

Физической вероятностной мерой является $P = (p, p, p)^{\mathrm{T}}$, где $p = 1/3$. Все остальные условия не меняются.

Формально задача репрезентативного агента на оптимизацию не меняется:

$$\max_{b,s,\lambda} f(b,s,\lambda) = \mathbf{E}^P\left(u\left(b \cdot B_1 + s \cdot S_1\right)\right) - \lambda \cdot \left(b \cdot B_0 + s \cdot S_0 - w\right).$$

Расчет относительной цены тоже остается прежним:

$$\frac{S_0}{B_0} = \frac{\mathbf{E}^P\left(\frac{S_1}{B_1 + S_1}\right)}{\mathbf{E}^P\left(\frac{B_1}{B_1 + S_1}\right)} \equiv \zeta.$$

В Python необходимо скорректировать лишь векторы будущих цен финансовых активов и вектор, представляющий вероятностную меру:

```
In [82]: p = 1 / 3 ❶

In [83]: P = np.array((p, p, p)) ❶

In [84]: B1 = np.array((11, 11, 11))

In [85]: S1 = np.array((20, 10, 5))

In [86]: zeta = np.dot(S1 / (B1 + S1), P) / np.dot(B1 / (B1 + S1), P) ❷

In [87]: zeta ❷
Out[87]: 0.9155274934101636

In [88]: w = 15 ❸

In [89]: B0 = w / (1 + zeta) ❹

In [90]: B0 ❹
Out[90]: 7.8307411674347165

In [91]: S0 = zeta * B0 ❺

In [92]: S0 ❺
Out[92]: 7.169258832565284

In [93]: B0 + S0 ❻
Out[93]: 15.0

In [94]: i = B1.mean() / B0 - 1 ❼
```

```
In [95]: i ❼
Out[95]: 0.40472016183411985

In [96]: mu = np.dot(S1, P) / S0 - 1 ❽

In [97]: mu ❽
Out[97]: 0.6273183796451287
```

❶ Вероятностная мера.

❷ Ценовой коэффициент ζ с учетом условий оптимальности.

❸ Первоначальный капитал.

❹ Уровень равновесной цены безрисковой облигации с учетом коэффициента цен ζ и первоначального капитала w.

❺ Уровень равновесной цены рисковой акции.

❻ Бюджетное ограничение как обязательное условие.

❼ Равновесная процентная ставка с учетом уровня цены безрисковой облигации.

❽ Равновесная ожидаемая ставка доходности рисковой акции.

Ценообразование относительно репрезентативного агента

Ценообразование ценных бумаг, основанное на расчетах оптимизации относительно репрезентативного агента, является одним из подходов, который применяется как к полным, так и к неполным рынкам. Вместо корректировки рынков (например, посредством восполнения рынка добавочными ценными бумагами) вводятся дополнительные предположения в отношении репрезентативного агента. Так, для получения конкретных абсолютных цен на ценные бумаги вместо относительных цен необходимо сделать предположения о размере первоначального капитала агента.

Мартингальные меры

Несмотря на то что ценообразование через репрезентативного агента работает на неполных рынках так же, как и на полных, оно, к сожалению, не решает непосредственно задачу ценообразования недостижимых условных требований.

Существует бесконечно большое количество мартингальных мер, согласующихся с рынком.

Код, представленный далее, демонстрирует бесконечно большое количество мартингальных мер, которые согласуются с выведенными в предыдущем разделе процессами ценообразования в условиях равновесия:

```
In [98]: qu = sy.Symbol('qu') ❶
         qm = sy.Symbol('qm') ❶

In [99]: eq = (qu * 20 + qm * 10 + (1 - qu - qm) * 5) / (1 + i) - S0 ❷

In [100]: eq ❸
Out[100]: 3.55942780337942*qm + 10.6782834101383*qu - 3.60983102918587

In [101]: Q = sy.solve(eq, set=True) ❹

In [102]: Q ❺
Out[102]: ([qm], {(1.01416048550236 - 3.00000000000001*qu,)})
```

❶ Определение символов `qu` и `qm`.

❷ Формулировка уравнения для `qu` и `qm` с учетом условия мартингала.

❸ Упрощение уравнения.

❹ Решение уравнения в числовой форме с выводом множества результатов (без учета условия $0 \leq q^u, q^d \leq 1$).

❺ Соотношение между `qu` и `qm` как результат решения, указывающий на бесконечно большое количество решений.

Мартингальная мера в условиях неполного рынка

Ценообразование по мартингалу — это удобный и простой подход к оценке условных требований на полных рынках. На неполных рынках, как правило, существует бесконечно большое количество мартингальных мер, согласованных с рынком. На практике данную задачу часто решают, опираясь на публично наблюдаемую рыночную стоимость ликвидных условных требований, например обычных европейских пут- или колл-опционов. Эти цены используются для калибровки параметров модели или непосредственно мартингальной меры, чтобы достичь согласованности с рынком. Более подробная информация о калибровке модели представлена в работе Хилпиша 2015 года.

Ценообразование в условиях равновесия

Что насчет ценообразования условных требований? Если условное требование *достижимо* через реплицируемый портфель, состоящий из торгуемых финансовых активов, его цена может быть установлена обычным *арбитражным аргументом*. Что, если условное требование *недостижимо*? В простых условиях рассматриваемого нами неполного рынка недостижимость условного требования обозначает линейную независимость вектора выплат по нему от двух векторов будущих цен торгуемых финансовых активов. Введение условного требования с таким вектором выплат *восполняет рынок*, поскольку три линейно независимых вектора так или иначе образуют базис $\mathbb{R}^3$.

Рассмотрим матрицу рыночных выплат по первым двум ценным бумагам Эрроу — Дебре, каждая из которых доступна при чистом предложении, равном единице:

$$\mathcal{M} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}.$$

Этот рынок явно неполный, поскольку две ценные бумаги не покрывают пространство $\mathbb{R}^3$. Введение условного требования с чистым предложением, равным единице, которое приносит одну единицу валюты в состоянии d (то есть условное требование, которое приносит ровно столько, сколько приносила бы третья ценная бумага Эрроу — Дебре), восполняет рынок, что демонстрирует следующая матрица выплат:

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Сейчас три вектора выплат формируют стандартный базис $\mathbb{R}^3$.

Формально задача репрезентативного агента на оптимизацию не отличается от представленной ранее:

$$\max_{\varphi} \mathbf{E}^P \left(u\left(\mathcal{M} \cdot \varphi \right) \right),$$

таким образом:

$$\mathcal{M}_0 \cdot \varphi = w.$$

Здесь $\mathcal{M}_0 = \left(\gamma^u, \gamma^m, \gamma^d\right)^{\mathbf{T}}$ — вектор цен, зависящих от состояния экономики, а $\phi = (1, 1, 1)^{\mathrm{T}}$ — рыночный портфель.

Согласно теореме Лагранжа задача безусловного экстремума в развернутом виде такова:

$$\max_{\varphi^u, \varphi^m, \varphi^d, \lambda} f\left(\varphi^u, \varphi^m, \varphi^d, \lambda\right) = p^u \cdot u\left(\varphi^u\right) + p^m \cdot u\left(\varphi^m\right) + p^d \cdot u\left(\varphi^d\right) -$$
$$-\lambda \cdot \left(\gamma^u \cdot \varphi^u + \gamma^m \cdot \varphi^m + \gamma^d \cdot \varphi^d - w\right).$$

Она имеет четыре условия первого порядка:

$$\begin{cases} \dfrac{\partial f}{\partial \varphi^u} = p^u \cdot u'\left(\varphi^u\right) - \lambda \cdot \gamma^u = 0; \\ \dfrac{\partial f}{\partial \varphi^m} = p^m \cdot u'\left(\varphi^m\right) - \lambda \cdot \gamma^m = 0; \\ \dfrac{\partial f}{\partial \varphi^d} = p^d \cdot u'\left(\varphi^d\right) - \lambda \cdot \gamma^d = 0; \\ \dfrac{\partial f}{\partial \lambda} = \gamma^u \cdot \varphi^u + \gamma^m \cdot \varphi^m + \gamma^d \cdot \varphi^d - w = 0. \end{cases}$$

В результате получаем следующие относительные цены:

$$\begin{cases} \dfrac{\gamma^u}{\gamma^m} = \dfrac{p^u \cdot u'\left(\varphi^u\right)}{p^m \cdot u'\left(\varphi^m\right)}; \\ \dfrac{\gamma^u}{\gamma^d} = \dfrac{p^u \cdot u'\left(\varphi^u\right)}{p^d \cdot u'\left(\varphi^d\right)}, \end{cases}$$

через которые устанавливается и третья относительная цена. При логарифмической полезности и первоначальном капитале $w = 1$ мы приходим к такому уравнению:

$$\begin{cases} \gamma^u = p^u; \\ \gamma^m = p^m; \\ \gamma^d = p^d. \end{cases}$$

Вектором равновесных цен является $\mathcal{M}_0^* = \left(p^u, p^m, p^d\right)^{\mathrm{T}}$, что соответствует вектору вероятностной меры, а это значит, что все процессы дисконтирования цен являются мартингалами при физической вероятностной мере.

Пример в числовом виде (часть 2)

Вернемся к предыдущему числовому примеру: предположим максимизацию ожидаемой полезности на основе *логарифмической полезности* для репрезентативного агента. Процесс ценообразования для безрисковой облигации составляет:

$$B = \left(B_0, (11, 11, 11)^{\mathrm{T}}\right),$$

а для *рисковой акции*:

$$S = \left(S_0, (20, 10, 5)^{\mathrm{T}}\right).$$

Матрица рыночных выплат имеет вид:

$$\mathcal{M} = \begin{pmatrix} 11 & 20 \\ 11 & 10 \\ 11 & 5 \end{pmatrix}.$$

Ко всему прочему, дана физическая мера вероятности $P = (p, p, p)^{\mathrm{T}}$, где $p = 1/3$. Чистое предложение безрисковой облигации и рисковой акции составляет $b = 1$ и $s = 1$ соответственно. Агент располагает первоначальным капиталом $w = 15$.

Введем условное требование с выплатой $C_1 = (5, 0, 0)^{\mathrm{T}}$ и чистым предложением $c = 1$. В результате задача репрезентативного агента имеет вид:

$$\max_{\varphi, \lambda} f(\varphi, \lambda) = \mathbf{E}^P\left(u\left(\mathcal{M} \cdot \varphi\right)\right) - \lambda \cdot \left(\mathcal{M}_0 \cdot \varphi - w\right)$$

или

$$\max_{b, s, c, \lambda} f(b, s, \lambda) = \mathbf{E}^P\left(u\left(b \cdot B_1 + s \cdot S_1 + c \cdot C_1\right)\right) - \lambda \cdot \left(b \cdot B_0 + s \cdot S_0 + c \cdot C_0 - w\right).$$

Четырьмя условиями первого порядка для этой задачи являются:

$$\begin{cases} \dfrac{\partial f}{\partial b} = \mathbf{E}^P\left(B_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1 + c \cdot C_1\right)\right) - \lambda \cdot B_0 = 0; \\ \dfrac{\partial f}{\partial S} = \mathbf{E}^P\left(S_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1 + c \cdot C_1\right)\right) - \lambda \cdot S_0 = 0; \\ \dfrac{\partial f}{\partial c} = \mathbf{E}^P\left(C_1 \cdot u'\left(b \cdot B_1 + s \cdot S_1 + c \cdot C_1\right)\right) - \lambda \cdot C_0 = 0; \\ \dfrac{\partial f}{\partial \lambda} = b \cdot B_0 + s \cdot S_0 + c \cdot C_0 - w = 0. \end{cases}$$

Устанавливаем также *относительные цены* для трех финансовых активов:

$$\begin{cases} \dfrac{S_0}{B_0} = \dfrac{\mathbf{E}^P\left(\dfrac{S_1}{B_1 + S_1 + C_1}\right)}{\mathbf{E}^P\left(\dfrac{B_1}{B_1 + S_1 + C_1}\right)} \equiv \zeta_1; \\ \dfrac{C_0}{B_0} = \dfrac{\mathbf{E}^P\left(\dfrac{C_1}{B_1 + S_1 + C_1}\right)}{\mathbf{E}^P\left(\dfrac{B_1}{B_1 + S_1 + C_1}\right)} \equiv \zeta_2. \end{cases}$$

Добавление *бюджетного ограничения* фиксирует не только относительные цены ζ_1 и ζ_2, но и уровни абсолютных цен:

$$B_0 + \zeta_1 \cdot B_0 + \zeta_2 \cdot B_0 = w \Leftrightarrow B_0 = \frac{w}{1 + \zeta_1 + \zeta_2}.$$

Переходим к Python и немного изменяем код для полного рынка:

```
In [103]: p = 1 / 3 ❶

In [104]: P = np.array((p, p, p)) ❶

In [105]: B1 = np.array((11, 11, 11)) ❷

In [106]: S1 = np.array((20, 10, 5)) ❷

In [107]: C1 = np.array((5, 0, 0)) ❷
```

```
In [108]: zeta_1 = (np.dot(S1 / (B1 + S1 + C1), P) /
                    np.dot(B1 / (B1 + S1 + C1), P)) ❸

In [109]: zeta_1 ❸
Out[109]: 0.8862001308044474

In [110]: zeta_2 = (np.dot(C1 / (B1 + S1 + C1), P) /
                    np.dot(B1 / (B1 + S1 + C1), P)) ❹

In [111]: zeta_2 ❹
Out[111]: 0.09156311314584695

In [112]: w = 15 ❺

In [113]: B0 = w / (1 + zeta_1 + zeta_2) ❻

In [114]: B0 ❻
Out[114]: 7.584325396825396

In [115]: S0 = zeta_1 * B0 ❼

In [116]: S0 ❼
Out[116]: 6.721230158730158

In [117]: C0 = zeta_2 * B0 ❽

In [118]: C0 ❽
Out[118]: 0.6944444444444443

In [119]: B0 + S0 + C0 ❾
Out[119]: 14.999999999999998

In [120]: i = B1.mean() / B0 - 1 ❿

In [121]: I ❿
Out[121]: 0.45035971223021587

In [122]: muS = np.dot(S1, P) / S0 - 1 ⓫

In [123]: muS ⓫
Out[123]: 0.7357933579335794

In [124]: muC = np.dot(C1, P) / C0 - 1 ⓬

In [125]: muC ⓬
Out[125]: 1.4000000000000004
```

❶ Вероятностная мера.

❷ Векторы выплат.

❸ Первая относительная цена.

❹ Вторая относительная цена.

❺ Первоначальный капитал...

❻ ...И получаемая цена безрисковой облигации.

❼ Равновесная цена рисковой акции.

❽ Равновесная цена условного требования.

❾ Бюджетное ограничение как обязательное условие.

❿ Безрисковая процентная ставка.

⓫ Равновесная ожидаемая ставка доходности рисковой акции.

⓬ Равновесная ожидаемая ставка доходности условного требования.

О том, что введение условного требования в качестве третьего торгуемого финансового актива восполняет рынок, можно судить по тому, что теперь существует уникальная мартингальная мера:

```
In [126]: M = np.array((B1, S1, C1)).T ❶

In [127]: M ❶
Out[127]: array([[11, 20, 5],
                 [11, 10, 0],
                 [11,  5, 0]])

In [128]: M0 = np.array((B0, S0, C0)) ❷

In [129]: Q = np.linalg.solve(M.T / (1 + i), M0) ❸

In [130]: Q ❹
Out[130]: array([0.20144, 0.34532, 0.45324])

In [131]: sum(Q) ❹
Out[131]: 1.0

In [132]: np.allclose(np.dot(M.T, Q), M0 * (1 + i)) ❺
Out[132]: True
```

❶ Новая матрица рыночных выплат, включающая в себя условное требование.

❷ Вектор цен трех финансовых активов, включая условное требование.

❸ Решение относительно вектора `Q`, представляющего мартингальную меру Q (обратите внимание на использование оператора транспонирования `.T`).

❹ Вектор результата решения, сумма компонентов которого равна 1.

❺ Итоговая проверка того, что все процессы дисконтирования цен действительно являются мартингалами.

Резюме

Глава 4 посвящена моделированию поведения и предпочтений агента и оптимизационных задач, с которыми он сталкивается, с применением подхода максимизации ожидаемой полезности. Были рассмотрены две центральные темы: *выбор оптимального портфеля* и *равновесное ценообразование финансовых активов и условных требований* в условиях полного и неполного рынков. Если в первом случае цены заданы предварительно, а содержание инвестиционного портфеля можно выбрать, то во втором случае, наоборот, содержание портфеля зафиксировано, а цены корректируются и таким образом репрезентативный агент может принять самое оптимальное решение. В очередной раз было подтверждено, что Python — это мощная экосистема с полезными для моделирования и решения оптимизационных задач инструментами.

Модели экономики в этой и предыдущих двух главах упрощены, однако представленные приемы и методы можно перенести на *общие статические модели экономики*, то есть на те, которые имеют гораздо больше, можно сказать, бесчисленное множество различных будущих состояний, а не два или три. Их можно перенести даже на *динамические экономики* с потенциально бесчисленными релевантными моментами времени вместо двух, для чего необходимо ввести всего несколько дополнительных формул.

Справочные материалы

В этой главе были упомянуты следующие книги.

- *Duffie D.* Security Markets — Stochastic Models. — San Diego: Academic Press, 1988.
- *Eichberger J., Harper I.* Financial Economics. — N. Y.: Oxford University Press, 1997.

- *Hilpisch Y.* Artificial Intelligence in Finance. — Sebastopol: O'Reilly, 2020.
- *Hilpisch Y.* Derivatives Analytics with Python. — Wiley Finance, 2015.
- *Markowitz H.* Portfolio Selection — Efficient Diversification of Investments. — New York: John Wiley & Sons, 1959.
- *Milne F.* Finance Theory and Asset Pricing. — N. Y.: Oxford University Press, 1995.
- *Sundaram R.* A First Course in Optimization Theory. — Cambridge: Cambridge University Press, 1996.
- *Varian H.* Microeconomic Analysis. 3rd ed. — New York and London: W. W. Norton & Company, 1992.

ГЛАВА 5

Статическая экономика

> Модель рынка ценных бумаг жизнеспособна тогда и только тогда, когда для нее существует хотя бы одна эквивалентная мартингальная мера.
>
> *Харрисон и Крепс (1979)*

> Аргументы безарбитражности связаны с мартингальной теорией так называемой фундаментальной теоремой ценообразования финансовых активов.
>
> *Дельбаен и Шахермайер (2006)*

В данной главе представлены новые формулы для моделирования обобщенной статической экономики, которая характеризуется довольно большим, но все же конечным пространством состояний. Как и прежде, в анализе будут рассмотрены два релевантных момента времени — «сегодня» и «через год», поэтому здесь вводится одно крупное обобщение относительно пространства состояний. В следующей главе модельная экономика будет иметь еще более обобщенный вид за счет увеличения количества моментов времени, что также позволит моделировать динамику экономики.

В этой главе используются понятия линейной алгебры и теории вероятностей. Данные темы подробно освещены в книгах Алескерова и др., Якода и Проттера соответственно. Плавное введение в общую статичную экономику и ее анализ можно найти в книге Мильне. Учебник Плиски (1997) — это хорошее, понятное и полное введение в данную тему. Учебник Дюффе

(1988) — более сложный и подробный, он во всей полноте представляет все необходимые элементы линейной алгебры и теории вероятностей для работы с финансами.

Здесь будут рассмотрены общие дискретные вероятностные пространства, финансовые активы и условные требования, полнота рынка, две фундаментальные теоремы ценообразования финансовых активов, репликация и арбитражное ценообразование, модели ценообразования опционов Блэка — Шоулза — Мертона (1973) и Мертона (1976), а также ценообразование через репрезентативного агента с добавлением ценных бумаг Эрроу — Дебре.

Финансы	Математика	Python
Неопределенность	Пространство состояний, алгебра, вероятностная мера, вероятностное пространство	NumPy, `ndarray`, `rng.normal`
Финансовые активы, условные требования	Случайная величина, математическое ожидание	`rng`, `mean()`, `np.dot`
Матрица рыночных выплат	Матрица	`ndarray`, `mean()`, `std()`
Репликация, арбитражное ценообразование	Решение системы линейных уравнений, скалярное произведение	`np.maximum`, `np.linalg.solve`, `np.dot`
Полнота рынка	Ранг, линейная оболочка, векторное пространство	`ndarray`, `np.dot`, `np.linalg.matrix_rank`
Мартингальная мера	Вероятностная мера	`ndarray`, `scipy.optimize.minimize`
Модель Блэка — Шоулза — Мертона	Геометрическое броуновское движение, нормальное распределение, моделирование Монте-Карло, репликация	`rng.standard_normal`, `np.linalg.lstsq`
Модель Мертона, логарифмически нормальный скачок	Скачкообразная диффузия, распределение Пуассона	`rng.poisson`, `np.linalg.lstsq`

Основная цель этой главы — *обобщение*. Несмотря на то что почти все представленные здесь концепции и расчеты были введены в предыдущих главах,

расширение пространства состояний требует использования дополнительных формул. Тем не менее код Python останется таким же лаконичным, как и раньше. Преимущества такого обобщения очевидны: нереально моделировать возможную будущую цену акции, скажем, Apple на основании только двух или трех возможных состояний. Гораздо объективнее предположить, что цена акции может принимать какое-то значение из возможных 100, 500 или более значений, — это важный шаг при переходе к более реалистичной финансовой модели.

Неопределенность

Рассмотрим экономику с *общим дискретным пространством состояний* Ω с конечным числом элементов $|\Omega| < \infty$. *Алгебра* $\mathcal{F}$ в Ω — это семейство множеств, для которых справедливы следующие утверждения.

1. $\Omega \in \mathcal{F}$.
2. $\mathbb{E} \in \mathcal{F} \Rightarrow \mathbb{E}^c \in \mathcal{F}$.
3. $\mathbb{E}_1, \mathbb{E}_2 \ldots \mathbb{E}_I \in \mathcal{F} \Rightarrow \bigcup_{i=1}^{I} \mathbb{E}_i \in \mathcal{F}$.

$\mathbb{E}^c$ обозначает дополнение множества E. Алгеброй называется модель *наблюдения за событиями*, происходящими в экономике. Булеан $\wp(\Omega)$ является самой большой алгеброй, а множество $\mathcal{F} = \{\varnothing, \Omega\}$ — самой маленькой алгеброй в пространстве Ω. В данном контексте одно состояние экономики $\omega \in \Omega$ можно интерпретировать как *атомарное событие*.

Вероятность соотносит действительное число $0 \le p_\omega \equiv P(\{\omega\}) \le 1$ с состоянием $\omega \in \Omega$ или действительное число $0 \le P(\mathbb{E}) \le 1$ с событием $\mathbb{E} \in \mathcal{F}$. Если известны вероятности реализации всех состояний, то $P(\mathbb{E}) = \sum_{\omega \in \mathbb{E}} p_\omega$.

Вероятностная мера P: $\mathcal{F} \to [0, 1]$ имеет следующие свойства.

1. $\forall \mathbb{E} \in \mathcal{F} : P(\mathbb{E}) \ge 0$.
2. $P\left(\bigcup_{i=1}^{I} \mathbb{E}_i\right) = \sum_{i=1}^{I} \mathbb{E}_i$ для непересекающихся множеств $\mathbb{E}_i \in \mathcal{F}$.
3. $P(\Omega) = 1$.

Вместе три элемента $\{\Omega, \mathcal{F}, P\}$ образуют *вероятностное пространство*, которое в модельной экономике формально представляет *неопределенность*.

Случайные величины

В вероятностном пространстве $\{\Omega, \mathcal{F}, P\}$ *случайная величина* является $\mathcal{F}$-измеримой функцией:

$$S: \Omega \to \mathbb{R}_{\geq 0},\ \omega \mapsto S(\omega).$$

$\mathcal{F}$-измеримость подразумевает, что для каждого множества $\mathbb{E} \in \{[a, b]: a, b \in \mathbb{R}, a < b\}$, существует:

$$S^{-1}(\mathbb{E}) \equiv \{\omega \in \Omega : S(\omega) \in \mathbb{E}\} \in \mathcal{F}.$$

Если $\mathcal{F} \equiv \wp(\Omega)$, то *математическое ожидание* случайной величины определяется через:

$$\mathbf{E}^P(S) = \sum_{\omega \in \Omega} P(\omega) \cdot S(\omega),$$

если нет, то через:

$$\mathbf{E}^P(S) = \sum_{\mathbb{E} \in \mathcal{F}} P(\mathbb{E}) \cdot S(\mathbb{E}).$$

Примеры в числовом виде

Приведу конкретный пример. Предположим, что существует пространство состояний (элементарных событий) $\Omega = \{\omega_1, \omega_2, \omega_3, \omega_4\}$, а также задана алгебра с множеством $\mathcal{F} = \{\varnothing, \{\omega_1, \omega_2\}, \{\omega_3, \omega_4\}, \Omega\}$, которое удовлетворяет трем свойствам алгебры в пространстве Ω. Вероятностная мера определяется как $P(\{\omega_1, \omega_2\}) = P(\{\omega_3, \omega_4\}) = 1/2$, где P является вероятностной мерой на множестве $\mathcal{F}$ в приведенных ранее условиях.

Рассмотрим теперь функцию T, в которой $T(\omega_1) = 1$, $T(\omega_2) = 2$, $T(\omega_3) = 3$ и $T(\omega_4) = 4$. Данная функция *не является* определенной на вероятностном пространстве случайной величиной, поскольку алгебра $\mathcal{F}$ не различает, например, ω_1 и ω_2 из-за их принадлежности к множеству $\{\omega_1, \omega_2\}$. Здесь можно сказать, что алгебра недостаточно «гранулирована».

Взглянем на другую функцию, S, в которой $S(\omega_1) = 20$, $S(\omega_2) = 20$, $S(\omega_3) = 5$ и $S(\omega_4) = 5$. Она является случайной величиной, определенной на вероятностном пространстве, с математическим ожиданием:

$$\mathbf{E}^P(S) = \sum_{\mathbb{E} \in \mathcal{F}} P(\mathbb{E}) \cdot S(\mathbb{E}) = \frac{1}{2} \cdot 20 + \frac{1}{2} \cdot 5 = 12{,}5.$$

Тем не менее, как правило, предполагается, что $\mathcal{F} \equiv \wp(\Omega)$, при этом P определена так, что функция (случайная величина), например T, также является $\mathcal{F}$-измеримой с правильно определенным математическим ожиданием.

С помощью Python можно без труда проиллюстрировать работу с очень большими размерами пространства Ω. Следующий код предполагает равные вероятности реализации для всех возможных состояний:

```
In [1]: import numpy as np
        from numpy.random import default_rng
        np.set_printoptions(precision=5, suppress=True)

In [2]: rng = default_rng(100) ❶

In [3]: I = 1000 ❷

In [4]: S = rng.normal(loc=100, scale=20, size=I) ❸

In [5]: S[:14] ❸
Out[5]: array([ 76.84901, 105.79512, 115.61708, 110.87947,  80.77235,
121.42017,
               114.02911, 114.09947, 114.90125, 122.08694, 144.85945,
87.77014,
               100.94422, 135.08469])

In [6]: S.mean() ❹
Out[6]: 100.88376804485935
```

❶ Установление начального значения для генератора случайных чисел NumPy для сходимости результатов.

❷ Установление количества состояний в пространстве состояний (для последующего моделирования).

❸ Вывод нормально распределенных псевдослучайных величин I со средним значением `loc` и стандартным отклонением `scale`.

❹ Вычисление математического ожидания (средней величины) при условии равной вероятности каждого моделируемого значения (состояния).

Разумеется, можно выбрать любую другую вероятностную меру:

```
In [7]: P = rng.random(I) ❶

In [8]: P[:10] ❷
Out[8]: array([0.34914, 0.33408, 0.41319, 0.06102, 0.6339 , 0.51285, 0.51177,
               0.92149, 0.72853, 0.58985])
```

```
In [9]: P /= P.sum() ❷

In [10]: P.sum() ❷
Out[10]: 1.0

In [11]: P[:10] ❸
Out[11]: array([0.00072, 0.00069, 0.00085, 0.00013, 0.00131, 0.00106,
                0.00106, 0.0019 , 0.0015 , 0.00122])

In [12]: np.dot(P, S) ❹
Out[12]: 100.71981640185018
```

❶ Вывод равномерно распределенных случайных чисел от 0 до 1.

❷ Приведение суммы значений в объекте `ndarray` к единице.

❸ Полученный вес в соответствии со случайной вероятностной мерой.

❹ Математическое ожидание как скалярное произведение вектора вероятностей и вектора случайной величины.

Математические методы

В дискретных финансах, помимо линейной алгебры, особо важную роль играет традиционная теория вероятностей. Она позволяет систематически и просто фиксировать и анализировать неопределенность, или, если быть точнее, риск в экономике. В большинстве случаев стандартная линейная алгебра и теория вероятностей оказываются достаточно мощными инструментами для работы с дискретными финансовыми моделями. При переходе к непрерывным финансовым моделям требуются уже более сложные подходы, например стохастические методы. Более подробную информацию о дискретных финансах можно найти в работе Плиски (1997).

Финансовые активы

Рассмотрим экономику с двумя датами $t \in 0, 1$, «сегодня» и «через год» (или любой другой период времени в будущем). Предположим, что задано вероятностное пространство $\{\Omega, \mathcal{F} \equiv \wp(\Omega), P\}$, которое представляет собой *неопределенность* относительно будущего в модельной экономике с возможными будущими состояниями $|\Omega| \equiv I$. В данном случае $\Omega = \{\omega_1, \omega_2 ... \omega_I\}$.

Торгуемый *финансовый актив* представлен процессом ценообразования $S = (S_0, S_1)$, где цена сегодня составляет $S_0 \in \mathbb{R}_{>0}$, а цена через год S_1: $\Omega \to \mathbb{R}_{\geq 0}$

является случайной $\mathcal{F}$-измеримой величиной. Формально вектор будущих цен торгуемого финансового актива — это вектор с элементами I:

$$S_1 = \begin{pmatrix} S_1(\omega_1) \\ S_1(\omega_2) \\ \dots \\ S_1(\omega_I) \end{pmatrix}.$$

Если торгуется несколько финансовых активов, скажем, $K > 1$, то они представлены несколькими процессами ценообразования, $S^k = \left(S_0^k, S_1^k\right)$, $k = 1, 2 \dots K$. *Матрица рыночных выплат* состоит из векторов будущих цен торгуемых финансовых активов:

$$\mathcal{M} = \begin{pmatrix} S_1^1(\omega_1) & S_1^2(\omega_1) & \dots & S_1^K(\omega_1) \\ S_1^1(\omega_2) & S_1^2(\omega_2) & \dots & S_1^K(\omega_2) \\ \dots & \dots & \dots & \dots \\ S_1^1(\omega_3) & S_1^2(\omega_3) & \dots & S_1^K(\omega_3) \end{pmatrix}.$$

Обозначим *множество торгуемых финансовых активов* через $\mathcal{L} \equiv \left(S^1, S^2 \dots S^K\right)$. Таким образом, *статичную модельную экономику* можно представить в таком общем виде:

$$\mathcal{E} = \left(\{\Omega, \mathcal{F}, P\}, \mathcal{L}\right).$$

Как правило, здесь подразумевается, что $\mathcal{F} \equiv \wp(\Omega)$.

Предположим, что в экономике существует пять возможных будущих состояний $\Omega = \{\omega_1 \dots \omega_5\}$ с равной вероятностью материализации $\forall\omega \in \Omega$: $P(\omega) = 1/5$ и торгуются пять финансовых активов. На языке Python данную ситуацию можно представить следующим кодом:

```
In [13]: M = np.array((
             (11, 25,  0,  0, 25),
             (11, 20, 30, 15, 25),
             (11, 10,  0, 20, 10),
             (11,  5, 30, 15,  0),
             (11,  0,  0,  0,  0)
         )) ❶
```

```
In [14]: M0 = np.array(5 * [10.]) ❷

In [15]: M0 ❷
Out[15]: array([10., 10., 10., 10., 10.])

In [16]: M.mean(axis=0) ❸
Out[16]: array([11., 12., 12., 10., 12.])

In [17]: mu = M.mean(axis=0) / M0 – 1 ❹

In [18]: mu ❹
Out[18]: array([0.1, 0.2, 0.2, 0. , 0.2])

In [19]: (M / M0 - 1) ❺
Out[19]: array([[ 0.1,  1.5, -1. , -1. ,  1.5],
                [ 0.1,  1. ,  2. ,  0.5,  1.5],
                [ 0.1,  0. , -1. ,  1. ,  0. ],
                [ 0.1, -0.5,  2. ,  0.5, -1. ],
                [ 0.1, -1. , -1. , -1. , -1. ]])

In [20]: sigma = (M / M0 - 1).std(axis=0) ❻

In [21]: sigma ❻
Out[21]: array([0.     , 0.92736, 1.46969, 0.83666, 1.1225 ])
```

❶ Предполагаемая матрица рыночных выплат, столбцы которой представляют собой векторы будущих неопределенных цен на торгуемые финансовые активы.

❷ Вектор текущих цен пяти активов, для каждого из которых установлена цена 10.

❸ Расчет ожидаемой (или средней) будущей цены для каждого торгуемого финансового актива.

❹ Расчет ожидаемой (или средней) ставки доходности.

❺ Расчет и вывод матрицы ставок доходности.

❻ Стандартное отклонение ставок доходности или волатильность, рассчитанная для каждого торгуемого финансового актива. Первый актив является безрисковым, поэтому можно считать его облигацией.

Условные требования

В модельной экономике $\mathcal{E}$ *условное требование* характеризуется процессом ценообразования $C = (C_0, C_1)$, где C_1 — это $\mathcal{F}$-измеримая случайная величина.

Европейские колл- и пут-опционы являются каноническими примерами условных требований. Например, выплата по европейскому колл-опциону может определяться относительно второго торгуемого финансового актива в соответствии с выплатой $C_1 = \max\left(S_1^2 - K, 0\right)$, где $K \in \mathbb{R}_{\geq 0}$ — цена исполнения опциона. Поскольку выплата по опциону производна от другого актива, мы говорим о *производных финансовых инструментах*, или, сокращенно, *деривативах*.

Если условное требование может быть *реплицировано* портфелем $\phi \in \mathbb{R}^K$ торгуемых финансовых активов $\mathcal{L}$:

$$\mathcal{M} \cdot \varphi = C_1,$$

то его *арбитражная цена* равна:

$$\mathcal{M}_0 \cdot \varphi = C_0,$$

где $\mathcal{M}_0 = \left(S_0^1, S_0^2 \ldots S_0^K\right)^{\mathbf{T}} \in \mathbb{R}_{\geq 0}^I$ является *вектором текущих цен* торгуемых финансовых активов.

Продолжим код из предыдущего раздела. Репликация условных требований на основе линейной алгебры имеет следующий вид:

```
In [22]: K = 15 ❶

In [23]: M[:, 1] ❶
Out[23]: array([25, 20, 10,  5,  0])

In [24]: C1 = np.maximum(M[:, 1] - K, 0) ❷

In [25]: C1 ❷
Out[25]: array([10,  5,  0,  0,  0])

In [26]: phi = np.linalg.solve(M, C1) ❸

In [27]: phi ❸
Out[27]: array([ 0.,  0.5,  0.01667, -0.2, -0.1])
```

```
In [28]: np.allclose(C1, np.dot(M, phi)) ❹
Out[28]: True

In [29]: C0 = np.dot(M0, phi) ❺

In [30]: C0 ❺
Out[30]: 2.1666666666666665
```

❶ Цена исполнения европейского колл-опциона и вектор выплат по соответствующему финансовому активу.

❷ Выписка колл-опциона на второй торгуемый финансовый актив с будущей выплатой $S_1^2 = (25, 20, 10, 5, 0)$.

❸ Решение задачи репликации с учетом матрицы рыночных выплат.

❹ Проверка того, что реплицирующий портфель действительно воспроизводит будущие выплаты по европейскому колл-опциону.

❺ Вывод из реплицирующего портфеля арбитражной цены вместе с вектором текущих цен торгуемых финансовых активов.

Полнота рынка

Полнота рынка в статичной модельной экономике может быть проанализирована на основе *ранга* матрицы рыночных выплат $\mathcal{M}$, определяемой торгуемыми финансовыми активами $\mathcal{L}$. Ранг матрицы равен числу линейно независимых векторов-столбцов (см.: Алескеров и др., 2011, раздел 2.7). Рассмотрим векторы-столбцы, представляющие векторы будущих цен торгуемых финансовых активов. Они *линейно независимы*, если:

$$\mathcal{M} \cdot \varphi = 0$$

имеет одно решение, а именно изотропный вектор $\phi = (0, 0... 0)^T \in \mathbb{R}^K$.

В то же время *линейная оболочка* матрицы рыночных выплат $\mathcal{M}$ задается всеми линейными комбинациями векторов-столбцов:

$$\text{span}(\mathcal{M}) = \{\varphi \in \mathbb{R}^K : \mathcal{M} \cdot \varphi\}.$$

Модельная экономика $\mathcal{E}$ является полной, если множество достижимых условных требований удовлетворяет условию $\mathbb{A} = \mathbb{R}^I$. К тому же по определению

это множество равно линейной оболочке торгуемых финансовых активов $\mathbb{A} \equiv \text{span}(\mathcal{M})$. Таким образом, модельная экономика $\mathcal{E}$ является *полной* при

$$\text{span}(\mathcal{M}) = \mathbb{R}^I.$$

При каких обстоятельствах полнота рынка возможна? Она возможна, если ранг матрицы равен числу возможных будущих состояний или больше него:

$$\text{rank}(\mathcal{M}) \geq I.$$

Другими словами, векторы-столбцы $\mathcal{M}$ образуют *базис* векторного пространства $\mathbb{R}^I$ (с потенциально большим количеством базисных векторов, чем требуется). *Векторное пространство* $\mathbb{V}$ представляет собой множество элементов *(векторов)*, которому свойственно следующее.

1. Функция сложения, сопоставляющая два вектора, $v_1, v_2 \in \mathbb{V}$, с другим элементом векторного пространства, $(v_1 + v_2) \in \mathbb{V}$.
2. Функция скалярного умножения, сопоставляющая скаляр $\alpha \in \mathbb{R}$ и вектор $v \in \mathbb{V}$ с другим элементом векторного пространства $(\alpha \cdot v) \in \mathbb{V}$.
3. Специальный элемент — нулевой, или нейтральный, — $0 \in \mathbb{V}$ с условием $v + 0 = v$.

Таким образом, например, не остается сомнений, что множества $\mathbb{R}$, $\mathbb{R}^5$, или $\mathbb{R}^I$, $I \in \mathbb{N}_{>0}$, являются векторными пространствами.

В соответствии с изложенным ранее модельная экономика считается *неполной*, если:

$$\text{rank}(\mathcal{M}) < I.$$

Для большей конкретизации рассмотрим пространство состояний с тремя возможными будущими состояниями $\Omega = \{\omega_1, \omega_2, \omega_3\}$. Все случайные переменные, таким образом, являются векторами в пространстве $\mathbb{R}^3$. Следующая матрица рыночных выплат, состоящая из трех торгуемых финансовых активов, имеет ранг 2 только из-за линейной зависимости двухэлементных векторов-столбцов, что и приводит к неполноте рынка:

$$\mathcal{M} = \begin{pmatrix} 11 & 20 & 10 \\ 11 & 10 & 5 \\ 11 & 5 & 2{,}5 \end{pmatrix} \Rightarrow \text{rank}(\mathcal{M}) = 2.$$

Подтвердить линейную зависимость финансовых активов 2 и 3 не составляет труда:

$$S_1^2 = \begin{pmatrix} 20 \\ 10 \\ 5 \end{pmatrix} = 2 \cdot \begin{pmatrix} 10 \\ 5 \\ 2{,}5 \end{pmatrix} = 2S_1^3.$$

Для сравнения посмотрим на другую матрицу рыночных выплат с другими тремя торгуемыми финансовыми активами. Она имеет ранг 3, то есть отображает полноту рынка. В таком случае также говорят о *полном ранге* матрицы:

$$\mathcal{M} = \begin{pmatrix} 11 & 20 & 10 \\ 11 & 10 & 25 \\ 11 & 5 & 10 \end{pmatrix} \Rightarrow \operatorname{rank}(\mathcal{M}) = 3.$$

Далее предположим вероятностное пространство, в котором пространство состояний состоит из пяти элементов, $\Omega = \{\omega_1 \ldots \omega_5\}$. Векторы будущих неопределенных цен и выплат по пяти торгуемым финансовым активам $\mathcal{L}$ и, соответственно, всех условных требований теперь являются элементами векторного пространства $\mathbb{R}^5$. Следующий код анализирует репликацию условных требований на основе такой модельной экономики $\mathcal{E}$. Он начинается с допущения о том, что все пять ценных бумаг Эрроу — Дебре участвуют в торговле:

```
In [31]: M = np.eye(5) ❶

In [32]: M ❶
Out[32]: array([[1., 0., 0., 0., 0.],
                [0., 1., 0., 0., 0.],
                [0., 0., 1., 0., 0.],
                [0., 0., 0., 1., 0.],
                [0., 0., 0., 0., 1.]])

In [33]: np.linalg.linalg.matrix_rank(M) ❷
Out[33]: 5

In [34]: C1 = np.arange(10, 0, -2) ❸

In [35]: C1 ❸
Out[35]: array([10,  8,  6,  4,  2])

In [36]: np.linalg.solve(M, C1) ❹
Out[36]: array([10.,  8.,  6.,  4.,  2.])
```

❶ Создание двумерного объекта `ndarray` — матрицы тождества, которую можно интерпретировать как матрицу рыночных выплат, полученную в результате торговли пятью ценными бумагами Эрроу — Дебре. Она образует так называемый *канонический* базис векторного пространства $\mathbb{R}^5$.

❷ Вычисление ранга матрицы.

❸ Выплата по условному требованию, которая должна быть реплицирована торгуемыми финансовыми активами.

❹ Решение задачи репликации (реализации).

Далее идет генерация рандомизированной матрицы рыночных выплат с полным рангом:

```
In [37]: rng = default_rng(100) ❶

In [38]: M = rng.integers(1, 10, (5, 5)) ❷

In [39]: M ❷
Out[39]: array([[7, 8, 2, 6, 1],
                [3, 4, 1, 6, 9],
                [9, 6, 4, 8, 9],
                [9, 1, 7, 7, 2],
                [5, 9, 7, 3, 3]])

In [40]: np.linalg.matrix_rank(M) ❸
Out[40]: 5

In [41]: np.linalg.matrix_rank(M.T) ❸
Out[41]: 5

In [42]: phi = np.linalg.solve(M, C1) ❹

In [43]: phi ❹
Out[43]: array([-1.16988, 0.52471, -0.3861 , 2.56409, -0.62085])

In [44]: np.dot(M, phi) ❺
Out[44]: array([10., 8., 6., 4., 2.])
```

❶ Установление начального значения для генератора случайных чисел NumPy для сходимости результатов.

❷ Создание рандомизированной матрицы рыночных выплат (объекта `ndarray` с формой (5, 5), заполненного случайными целыми числами от 1 до 10).

❸ Матрица имеет полный ранг: векторы-столбцы и векторы-строки линейно независимы.

❹ Ненулевое решение задачи репликации с рандомизированным базисом векторного пространства $\mathbb{R}^5$.

❺ Проверка решения на достижение абсолютной репликации.

Фундаментальные теоремы ценообразования финансовых активов

Рассмотрим общую статичную модельную экономику $\mathcal{E} = (\{\Omega, \mathcal{F}, P\}, \mathcal{L})$ с возможными состояниями I и торгуемыми финансовыми активами K. Предположим, что безрисковая краткосрочная процентная ставка предоставления и получения займов в экономике $r \in \mathbb{R}_{\geq 0}$[1].

Арбитражная возможность возникает тогда, когда цена портфеля $\phi \in \mathbb{R}^K$ торгуемых финансовых активов $\mathcal{L}$ равна нулю:

$$S_0 \cdot \varphi = \sum_{k=1}^{K} S_0^k \cdot \varphi^k = 0,$$

а ожидаемая выплата по нему — больше нуля:

$$\mathbf{E}^P\left(\mathcal{M} \cdot \varphi\right) > 0.$$

Обозначим *множество всех арбитражных возможностей* через:

$$\mathbb{O} \equiv \left\{\varphi \in \mathbb{R}^K : S_0 \cdot \varphi = 0, \mathbf{E}^P\left(\mathcal{M} \cdot \varphi\right) > 0\right\}.$$

Мартингальная мера Q для модельной экономики преобразует процессы дисконтирования цен в мартингалы и поэтому удовлетворяет следующему условию:

$$\frac{1}{1+r} \cdot \mathbf{E}^Q\left(\mathcal{M}\right) = S_0.$$

---

[1] Мы заменили обозначение процентной ставки с i на r, чтобы подчеркнуть ее краткосрочность.

Таким образом, можно сформулировать *первую фундаментальную теорему ценообразования финансовых активов* (см. главу 2), согласно которой мартингальная мера существует только при отсутствии арбитражных возможностей. Данная теорема была рассмотрена и доказана Стэнли Плиской (1997, раздел 1.3).

Первая фундаментальная теорема ценообразования финансовых активов. Эквивалентны следующие утверждения.

1. Мартингальная мера Q существует.
2. Экономика является безарбитражной, $\mathbb{O} = \varnothing$.

Формула выведения мартингальной меры совпадает с решением задачи репликации для условного требования $C = (C_0, C_1)$:

$$\mathcal{M} \cdot \varphi = C_1.$$

В ней реплицирующий портфель ϕ является неизвестным элементом, найти который можно с помощью решения системы линейных уравнений, как было показано в главе 2. Формулу нахождения мартингальной меры можно записать в виде:

$$\frac{1}{1+r} \cdot \mathbf{E}^Q(\mathcal{M}) = \frac{1}{1+r} \mathcal{M}^{\mathbf{T}} \cdot Q = \tilde{\mathcal{M}} \cdot Q = S_0,$$

где $\tilde{\mathcal{M}} \equiv \frac{1}{1+r} \mathcal{M}^{\mathbf{T}}$ и

$$Q = \begin{pmatrix} Q(\omega_1) \\ Q(\omega_2) \\ \cdots \\ Q(\omega_I) \end{pmatrix}, \ \forall \omega \in \Omega: 0 \le Q(\omega) \le 1, \ \sum_{\omega \in \Omega} Q(\omega) = 1.$$

Эта задача выступает *двойственной* для задачи репликации, хотя и с некоторыми ограничениями. Эти ограничения, вытекающие из требования, согласно которому решение должно быть вероятностной мерой, меняют подход к кодированию в Python. Вместо обычного решения системы линейных уравнений задачу на нахождение мартингальной меры можно смоделировать через условную минимизацию. В примере далее предполагается пространство состояний с пятью элементами и структура рыночных выплат, описанная ранее:

```
In [45]: import scipy.optimize as sco ❶

In [46]: M = np.array((
             (11, 25,  0,  0, 25),
             (11, 20, 30, 15, 25),
             (11, 10,  0, 20, 10),
             (11,  5, 30, 15,  0),
             (11,  0,  0,  0,  0)
          )) ❷

In [47]: np.linalg.matrix_rank(M) ❸
Out[47]: 5

In [48]: M0 = np.ones(5) * 10 ❹

In [49]: M0 ❺
Out[49]: array([10., 10., 10., 10., 10.])

In [50]: r = 0.1 ❻

In [51]: def E(Q):
             return np.sum((np.dot(M.T, Q) - M0 * (1 + r)) ** 2) ❼

In [52]: E(np.array(5 * [0.2]))
Out[52]: 4.0

In [53]: cons = ({'type': 'eq', 'fun': lambda Q: Q.sum() - 1}) ❽

In [54]: bnds = (5 * [(0, 1)]) ❾

In [55]: bnds ❾
Out[55]: [(0, 1), (0, 1), (0, 1), (0, 1), (0, 1)]

In [56]: res = sco.minimize(E, 5 * [1], ❿
                            method='SLSQP', ⓫
                            constraints=cons, ⓬
                            bounds=bnds) ⓭

In [57]: Q = res['x'] ⓮

In [58]: Q ⓮
Out[58]: array([0.14667, 0.18333, 0.275 , 0.18333, 0.21167])

In [59]: np.dot(M.T, Q) / (1 + r) ⓯
Out[59]: array([10.     , 9.99998,  9.99999, 10.00001,  9.99998])

In [60]: np.allclose(M0, np.dot(M.T, Q) / (1 + r))
Out[60]: True
```

❶ Импорт модуля `optimize` из `scipy`.

❷ Определение матрицы рыночных выплат.

❸ Проверка полноты ранга матрицы.

❹ Определение вектора цен для торгуемых финансовых активов...

❺ ...И вывод значений — все они равны 10.

❻ Установление постоянной краткосрочной ставки.

❼ Определение целевой функции, которая должна быть минимизирована. Данный шаг обусловлен тем, что система линейных уравнений должна быть решена с учетом ограничений и пределов для всех параметров.

❽ Ограничение, согласно которому сумма единичных вероятностей равна единице.

❾ Определение границ для каждой единичной вероятности.

❿ Процедура оптимизации, которая минимизирует функцию `E`...

⓫ ...Определяет используемый метод...

⓬ ...Устанавливает ограничения, которые должны соблюдаться...

⓭ ...И устанавливает пределы для параметров.

⓮ Вектор с результатами — мартингальная мера.

⓯ Согласно определению мартингальной меры, процессы дисконтирования цен являются мартингалами.

Вторая фундаментальная теорема о ценообразовании финансовых активов также действует в отношении обобщенной статичной модельной экономики $\mathcal{E}$. Она также была рассмотрена и доказана Стэнли Плиской (1997, раздел 1.5).

Вторая фундаментальная теорема ценообразования финансовых активов. Эквивалентны следующие утверждения.

1. Мартингальная мера Q уникальна.
2. Модель рынка является полной, $\mathbb{A} = \mathbb{R}^I_+$.

Фундаментальные теоремы

Поиски приемлемых способов представления ценообразования опционов привели к созданию прорывных моделей Блэка и Шоулза (1973) и Мертона (1973), образовавших в совокупности модель Блэка — Шоулза — Мертона. Эти модели довольно специфичны, поскольку предполагают геометрическое броуновское движение в качестве модели процесса ценообразования одного рискового актива. Исследования конца 1970-х — начала 1980-х годов, проведенные Харрисоном совместно с Крепсом (1979) и Плиской (1981), послужили основой для представления ценообразования условных требований. Центральную роль в них играют мартингальные меры и семимартингальные процессы. Класс семимартингальных процессов довольно велик и включает в себя как ранние модели (например, геометрическое броуновское движение), так и многие более сложные финансовые модели, появившиеся и исследованные гораздо позже (диффузия со скачками или процессы стохастической волатильности). Это одна из причин, почему представленные теоремы называют фундаментальными, — они применимы к большому количеству интересных и важных финансовых моделей.

Модель ценообразования опционов Блэка — Шоулза — Мертона

Основой для модели ценообразования опционов Блэка — Шоулза — Мертона послужила непрерывная модель экономики, обычно представляемая стохастическими дифференциальными уравнениями (СДУ) с подходящими граничными условиями. СДУ, которые описывают развитие *единственного рискового актива* (например, акции или фондового индекса), используются также для описания *геометрического броуновского движения*. Помимо рискового актива, в модельной экономике торгуется еще один безрисковый актив, по которому уплачивается непрерывная безрисковая краткосрочная ставка.

В статичной модели с двумя релевантными моментами времени, скажем $t = 0$ и $t = T > 0$, будущую, неопределенную стоимость рискового актива ST можно вычислить через:

$$S_T = S_0 \cdot e^{\left(r-\left(\sigma^2/2\right)\right)T+\sigma\sqrt{T}z},$$

где $S_0 \in \mathbb{R}_{>0}$ — это цена рискового актива сегодня, $r \in \mathbb{R}_{\geq 0}$ — постоянная безрисковая краткосрочная ставка, $\sigma \in \mathbb{R}_{>0}$ — постоянный коэффициент

волатильности, а z — стандартная нормально распределенная случайная величина (см.: Якод и Проттер, 2004, глава 16).

В дискретных условиях, чтобы получить в предыдущем уравнении числовые значения I для S_T, можно взять, например, стандартные нормально распределенные псевдослучайные величины z_i, $i = 1, 2... I$:

$$S_T(z_i) = S_0 \cdot e^{\left(r-\left(\sigma^2/2\right)\right)T+\sigma\sqrt{T}z_i}, \; i = 1, 2... I.$$

Такая процедура обычно называется *моделированием методом Монте-Карло*. Для упрощения обозначений в дальнейшем S_T будет означать вектор смоделированных будущих цен акции:

$$S_T \equiv \begin{pmatrix} S_T(z_1) \\ S_T(z_2) \\ \cdots \\ S_T(z_I) \end{pmatrix}.$$

Таким образом, модельная экономика выглядит следующим образом. Существует общее вероятностное пространство $\{\Omega, \mathcal{F} \equiv \wp(\Omega), P\}$ с возможными будущими состояниями экономики I. Предполагается, что каждое состояние одинаково вероятно, то есть:

$$\forall \omega \in \Omega : P(\omega) = \frac{1}{I}.$$

Множество торгуемых финансовых активов $\mathcal{L}$ состоит из безрискового актива — облигации с процессом ценообразования $B = (B_0, B_0 \cdot e^{rT})$, рискового актива — акции (по которой выплачиваются дивиденды) с процессом ценообразования $S = (S_0, S_T)$ и определенного ранее S_T. Все вместе они образуют экономическую модель Блэка — Шоулза — Мертона:

$$\mathcal{E}_{\text{BSM}} = \left(\{\Omega, \mathcal{F}, P\}, \mathcal{L} = \{B, S\}\right).$$

Предположим, что в качестве условного требования на акцию выписан европейский колл-опцион. Выплата по нему составляет:

$$C_T \equiv (S_T - K, 0)$$

с ценой исполнения $K \in \mathbb{R}_{\geq 0}$. Стоимость (используется *оценка методом Монте-Карло*) колл-опциона здесь дается в виде ожидаемой (средней) дисконтированной выплаты:

$$C_0 = e^{-rT} \frac{1}{I} \sum_{i=1}^{I} \max\left(S_T\left(z_i\right) - K, 0\right).$$

Представленная модельная экономика и подход к ценообразованию по методу Монте-Карло легко реализуются на языке Python. На рис. 5.1 показано частотное распределение смоделированных цен акции со средним значением и стандартным отклонением вокруг него:

```
In [61]: import math

In [62]: S0 = 100 ❶
         r = 0.05 ❷
         sigma = 0.2 ❸
         T = 1.0 ❹
         I = 10000 ❺

In [63]: rng = default_rng(100) ❻

In [64]: ST = S0 * np.exp((r - sigma ** 2 / 2) * T +
              sigma * math.sqrt(T) * rng.standard_normal(I)) ❼

In [65]: ST[:8].round(1) ❼
Out[65]: array([ 81.7, 109.2, 120.5, 114.9,  85. , 127.7, 118.6, 118.6])

In [66]: ST.mean() ❽
Out[66]: 105.6675325917807

In [67]: S0 * math.exp(r * T) ❾
Out[67]: 105.12710963760242

In [68]: from pylab import mpl, plt
         plt.style.use('seaborn')
         mpl.rcParams['savefig.dpi'] = 300
         mpl.rcParams['font.family'] = 'serif'

In [69]: plt.figure(figsize=(10, 6))
         plt.hist(ST, bins=35, label='frequency');
         plt.axvline(ST.mean(), color='r', label='mean')
         plt.axvline(ST.mean() + ST.std(), color='y', label='sd up')
         plt.axvline(ST.mean() - ST.std(), color='y', label='sd down')
         plt.legend(loc=0); ❿
```

❶ Начальная цена акции.

❷ Постоянная краткосрочная ставка.

❸ Коэффициент волатильности.

❹ Период времени в долях года.

❺ Количество состояний и количество моделирований.

❻ Установление начального значения для сходимости результатов.

❼ Основная строка кода — реализация моделирования методом Монте-Карло с помощью NumPy в векторном режиме через пошаговое моделирование значений *I*.

❽ Среднее значение, полученное из смоделированного множества цен акции.

❾ Теоретически ожидаемая цена акции.

❿ Вывод результатов моделирования в виде гистограммы и добавление некоторых основных статистических данных.

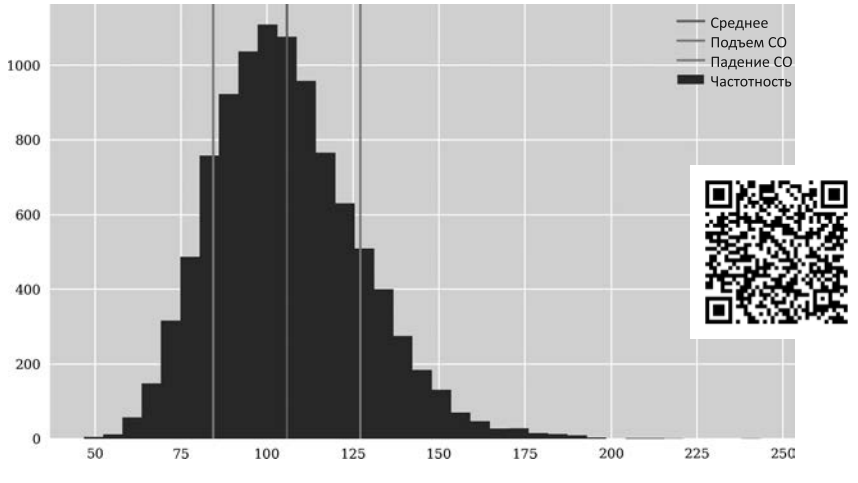


Рис. 5.1. Частотное распределение смоделированных цен акции по Блэку — Шоулзу — Мертону

Смоделированные значения цены акции упрощают ценообразование европейских опционов до двух векторных операций:

```
In [70]: K = 105 ❶

In [71]: CT = np.maximum(ST - K, 0) ❷

In [72]: CT[:8].round(1)
Out[72]: array([ 0. ,  4.2, 15.5,  9.9,  0. , 22.7, 13.6, 13.6])

In [73]: C0 = math.exp(-r * T) * CT.mean() ❸

In [74]: C0 ❸
Out[74]: 8.288763195530931
```

❶ Цена исполнения опциона.

❷ Вектор выплат по опциону.

❸ Оценка стоимости опциона методом Монте-Карло.

Полнота модельной экономики в системе Блэка — Шоулза — Мертона

Рассмотрим модельную экономику Блэка — Шоулза — Мертона $\mathcal{E}_{\text{BSM}}$ с точки зрения ее полноты. В предыдущем разделе методом Монте-Карло мы установили арбитражную цену европейского колл-опциона, несмотря на то что состояний экономики ($I \gg 2$) больше, чем торгуемых финансовых активов ($K = 2$). Таким образом, можно сделать следующие выводы.

- *Неполнота рынка в общем.* В более широком смысле экономика является неполной, поскольку не каждое условное требование может быть реплицировано портфелем торгуемых активов и не существует уникальной мартингальной меры, то есть не выполняются условия второй фундаментальной теоремы ценообразования финансовых активов.
- *Полнота рынка в частности.* В узком смысле модель является полной, поскольку каждое условное требование, которое может быть представлено функцией вектора цены на акцию $C_1 = f\left(S_1^2\right)$, можно реплицировать позициями в облигации и акции.

В приведенном ранее расчете арбитражной цены через моделирование методом Монте-Карло используется тот факт, что модельная экономика $\mathcal{E}_{\text{BSM}}$ является полной в конкретном, узком смысле. Выплата по европейскому

колл-опциону зависит только от вектора будущей цены акции. Пока что не хватает реплицирующего портфеля и расчета арбитражной цены, чтобы убедиться в обоснованности подхода моделирования методом Монте-Карло.

Функция `np.linalg.solve` из библиотеки NumPy, использовавшаяся до сих пор для решения репликационных задач, требует квадратной матрицы (рыночных выплат). В экономике Блэка — Шоулза — Мертона, где есть только два торгуемых финансовых актива, а возможных будущих состояний гораздо больше, данный элемент отсутствует. Однако задачу репликации в таком случае можно решить способом наименьших квадратов — `np.linalg.lstsq`:

```
In [75]: B0 = 100 ❶

In [76]: M0 = np.array((B0, S0)) ❷

In [77]: BT = B0 * np.ones(len(ST)) * math.exp(r * T) ❸

In [78]: BT[:4] ❸
Out[78]: array([105.12711, 105.12711, 105.12711, 105.12711])

In [79]: M = np.array((BT, ST)).T ❹

In [80]: M ❹
Out[80]: array([[105.12711,  81.74955],
                [105.12711, 109.19348],
                [105.12711, 120.4628 ],
                ...,
                [105.12711,  71.10624],
                [105.12711, 105.32038],
                [105.12711, 134.77647]])

In [81]: phi = np.linalg.lstsq(M, CT, rcond=None)[0] ❺

In [82]: phi ❺
Out[82]: array([-0.51089,  0.59075])

In [83]: np.mean((np.dot(M, phi) - CT)) ❻
Out[83]: 1.1798206855928583e-14

In [84]: np.dot(M0, phi) ❼
Out[84]: 7.9850808951857335
```

❶ Произвольно установленная стоимость облигации.

❷ Вектор цен сегодня для двух торгуемых финансовых активов.

❸ Вектор будущей цены облигации с учетом начальной цены и краткосрочной ставки.

❹ Полученная в результате матрица рыночных выплат с рангом 2 (сопоставленная с 10 000 будущих состояний).

❺ Решение задачи репликации способом наименьших квадратов. Для репликации колл-опциона облигация должна быть продана, а акция куплена.

❻ Средняя погрешность репликации, возникающая, например, из-за неточности плавающей запятой и используемых численных методов, не равна нулю, но очень близка к нему.

❼ Расчет арбитражной цены с учетом численно оптимального реплицирующего портфеля. Она близка к стоимости, ранее установленной методом Монте-Карло.

Модель ценообразования опционов со скачкообразной диффузией Мертона

В этом разделе представлена еще одна важная экономическая модель, которая основывается на идеях Мертона и отличается от модели ценообразования акции Блэка — Шоулза — Мертона наличием скачков. Случайные скачки являются причиной неполноты модели экономики $\mathcal{M}_{\text{M76}}$ в целом. Однако в дискретных условиях можно применить те же численные подходы к ценообразованию опционов, которые были введены для $\mathcal{E}_{\text{BSM}}$ в предыдущих двух разделах. Несмотря на то что диффузия определяется только в динамике, $\mathcal{M}_{\text{M76}}$ представляет собой модель со *скачкообразной диффузией*.

В реальных финансовых временных рядах скачки имеют определенные закономерности. Они могут быть вызваны обвалом фондового рынка, или другими редкими событиями, или чрезвычайными происшествиями. Модель Мертона подробно моделирует такие редкие события и их влияние на стоимость финансовых активов. Как правило, модели без скачков плохо подходят для объяснения определенных показателей и явлений, регулярно наблюдающихся в финансовых временных рядах. К тому же модель способна формировать как положительные, так и отрицательные скачки. На практике отрицательный скачок (большое падение) можно наблюдать в отношении фондовых индексов, а положительные скачки (пики) — в отношении индексов волатильности.

Также *экономика со скачкообразной диффузией Мертона* $\mathcal{E}_{\text{M76}}$ отличается от экономики Блэка — Шоулза — Мертона $\mathcal{E}_{\text{BSM}}$ будущей ценой акций в момент времени T, которую можно смоделировать посредством следующей формулы:

$$S_T(z_i) = S_0 \cdot \left(e^{\left(r - r_j - \left(\sigma^2/2\right)\right)T + \sigma\sqrt{T} z_i^1} + \left(e^{\mu + \delta z_i^2} - 1 \right) y_i \right), \; i = 1, 2 \ldots I,$$

где z_i^1, z_i^2 — это стандартные нормально распределенные величины, а y^i — распределенная по закону Пуассона случайная величина с интенсивностью λ (см.: Якод и Проттер, 2004, глава 4). Скачки, в свою очередь, имеют логарифмически нормальное распределение с математическим ожиданием μ и стандартным отклонением δ (см.: Якод и Проттер, 2004, глава 7). Ожидаемый размер скачка составляет:

$$r_j = \lambda \cdot \left(e^{\mu + \left(\delta^2/2\right)} - 1 \right).$$

Реализация этой модели в Python требует определения дополнительных параметров и моделирования трех случайных величин. На рис. 5.2 показано частотное распределение смоделированных цен, которые с учетом принятых параметров и используемого кода могут стать отрицательными:

```
In [85]: M0 = np.array((100, 100)) ❶

In [86]: r = 0.05
         sigma = 0.2
         lmbda = 0.3
         mu = -0.3
         delta = 0.1
         rj = lmbda * (math.exp(mu + delta ** 2 / 2) - 1)
         T = 1.0
         I = 10000

In [87]: BT = M0[0] * np.ones(I) * math.exp(r * T)

In [88]: z = rng.standard_normal((2, I)) ❷
         z -= z.mean() ❸
         z /= z.std() ❸
         y = rng.poisson(lmbda, I) ❹

In [89]: ST = S0 * (
            np.exp((r - rj - sigma ** 2 / 2) * T +
                   sigma * math.sqrt(T) * z[0]) +
            (np.exp(mu + delta * z[1]) - 1) * y
        ) ❺
```

```
In [90]: ST.mean() * math.exp(-r * T) ❻
Out[90]: 100.53765025420363

In [91]: plt.figure(figsize=(10, 6))
         plt.hist(ST, bins=35, label='frequency');
         plt.axvline(ST.mean(), color='r', label='mean')
         plt.axvline(ST.mean() + ST.std(), color='y', label='sd up')
         plt.axvline(ST.mean() - ST.std(), color='y', label='sd down')
         plt.legend(loc=0);
```

❶ Вектор начальных цен для двух торгуемых финансовых активов — облигации и акции.

❷ Первое множество стандартных нормально распределенных случайных величин.

❸ Второе множество стандартных нормально распределенных случайных величин.

❹ Множество распределенных по закону Пуассона случайных величин с интенсивностью `lambda`.

❺ Моделирование цен акции в момент времени T с учетом трех множеств случайных величин.

❻ Вычисление средней дисконтированной стоимости акции на основе смоделированных цен.

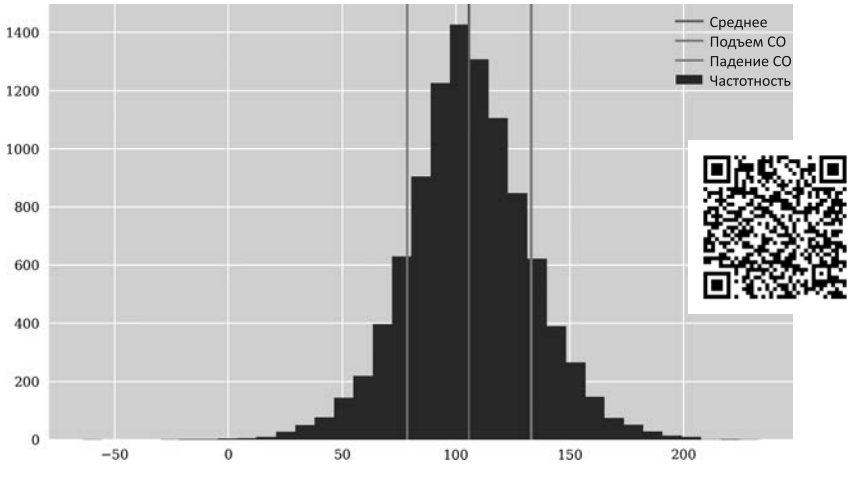


Рис. 5.2. Частотное распределение смоделированных цен акции по Мертону

Исключить отрицательные значения в моделировании методом Монте-Карло можно с помощью функции максимума (рис. 5.3):

```
In [92]: ST = np.maximum(S0 * (
             np.exp((r - rj - sigma ** 2 / 2) * T +
                    sigma * math.sqrt(T) * z[0]) +
             (np.exp(mu + delta * z[1]) - 1) * y
         ), 0) ❶

In [93]: plt.figure(figsize=(10, 6))
         plt.hist(ST, bins=35, label='frequency') ❷
         plt.axvline(ST.mean(), color='r', label='mean')
         plt.axvline(ST.mean() + ST.std(), color='y', label='sd up')
         plt.axvline(ST.mean() - ST.std(), color='y', label='sd down')
         plt.legend(loc=0);
```

❶ Функция максимума...

❷ ...Исключает отрицательные значения цены акции.

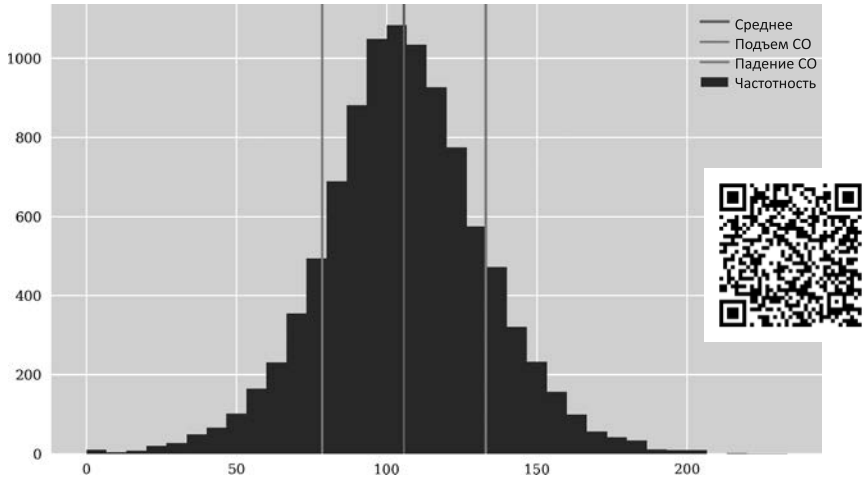


Рис. 5.3. Смоделированные (округленные) цены акции по Мертону

Заключительным этапом является установление цены европейского колл-опциона посредством оценки методом Монте-Карло и аппроксимирующей репликации:

```
In [94]: K = 105

In [95]: CT = np.maximum(ST - K, 0)
```

```
In [96]: C0 = math.exp(-r * T)  * np.mean(CT) ❶

In [97]: C0 ❶
Out[97]: 10.306374338651601

In [98]: M = np.array((BT, ST)).T

In [99]: phi = np.linalg.lstsq(M, CT, rcond=-1)[0] ❷

In [100]: phi ❷
Out[100]: array([-0.41827,  0.51847])

In [101]: np.mean(np.dot(M, phi) - CT) ❸
Out[101]: 1.1823431123048067e-15

In [102]: np.dot(M0, phi) ❹
Out[102]: 10.020157308565008
```

❶ Оценка европейского колл-опциона методом Монте-Карло.

❷ Аппроксимирующий портфель.

❸ Погрешность репликации в оптимальном портфеле.

❹ Арбитражная цена по оптимальному портфелю.

Неполнота модели из-за наличия скачков

Если модель Блэка — Шоулза — Мертона является полной в узком смысле, то наличие скачков из модели скачкообразной диффузии Мертона делает ее неполной в широком смысле. Даже введение дополнительных финансовых активов не сможет ее восполнить, так как скачки могут принимать бесконечное число значений и, следовательно, для восполнения модели потребуется бесконечное количество финансовых активов.

Ценообразование через репрезентативного агента

Предположим, что в обобщенной статичной экономике $\mathcal{E}$ действует *репрезентативный агент, максимизирующий ожидаемую полезность*. Его первоначальный капитал составляет $w \in \mathbb{R}_{>0}$, а предпочтения представлены функцией полезности $u : c \to \mathbb{R}$, $u(c) \mapsto \ln c$. Формально задача агента такая же, как и в главе 4:

$$\max_{\varphi} \mathbf{E}^{P}\left(u\left(\mathcal{M}\cdot\varphi\right)\right);\ w = \mathcal{M}_0 \cdot \varphi.$$

Однако, в отличие от предыдущей главы, сейчас мы рассматриваем экономику с гораздо большим количеством возможных будущих состояний и торгуемых финансовых активов.

Более того, если предположить, что полный набор ценных бумаг Эрроу — Дебре с чистым предложением единицы за каждую — это $K = I$, то *матрица рыночных выплат* имеет вид:

$$\mathcal{M} = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \dots & \dots & \dots & \dots \\ 0 & 0 & \dots & 1 \end{pmatrix}.$$

Согласно теореме Лагранжа задача безусловного экстремума выражается формулой:

$$\max_{\varphi,\,\lambda} f(\varphi, \lambda) = \mathbf{E}^P\left(u\left(\mathcal{M}\cdot\varphi\right)\right) - \lambda\cdot\left(\mathcal{M}_0\cdot\varphi - w\right).$$

Отсюда следует, что условиями первого порядка для всех позиций портфеля ϕ_i, $i = 1, 2... I$, где i — окупающаяся в состоянии ω_i ценная бумага Эрроу — Дебре, являются:

$$\frac{\partial f}{\partial \varphi_i} = P(\omega_i) - \lambda\cdot S_0^i = 0,\ i = 1, 2\dots I,$$

где S_0^i — это стоимость ценной бумаги Эрроу — Дебре, погашающейся в состоянии ω_i. Соотношение цен всех ценных бумаг Эрроу — Дебре определяется вероятностями реализации соответствующих состояний выплат:

$$\frac{S_0^i}{S_0^j} = \frac{P(\omega_i)}{P(\omega_j)},\ \omega_i, \omega_j \in \Omega.$$

Таким образом, при $w = 1$ можно вывести абсолютные цены:

$$S_0^i = P(\omega_i).$$

Другими словами, стоимость ценной бумаги Эрроу — Дебре, погашающейся в состоянии ω_i, равна вероятности $P(\omega_i)$ материализации этого состояния.

Следовательно, можно сделать вывод, что в обобщенной статичной экономике формулы решения задачи по ценообразованию через репрезентативного агента почти не отличаются от формул, использующихся для решения такой же задачи в условиях простой экономики из главы 4.

Резюме

В этой главе мы рассмотрели *общие статические экономики* с большим числом возможных будущих состояний: например, в модели Блэка — Шоулза — Мертона предполагается 10 000 различных состояний. Введенные дополнительные формулы вполне оправдывают себя, поскольку позволяют создать гораздо более реалистичные модели, которые можно применять на практике, например, для оценки европейских колл- и пут-опционов на фондовый индекс или отдельную акцию.

Библиотека NumPy — мощный инструмент для моделирования статических экономик с очень большим количеством матриц рыночных выплат. Векторизация позволяет быстро и эффективно применять моделирование методом Монте-Карло. Метод наименьшей квадратичной регрессии значительно упрощает создание аппроксимирующих портфелей.

Тем не менее в отношении моделирования в финансовом пространстве статичные экономики ограничены. Так, в них не учитывается досрочное исполнение опционов, как бывает, например, в работе с американскими опционами. Данное упущение будет восполнено увеличением релевантных моментов времени, что станет следующим шагом к динамическим экономикам, представленным в следующей главе.

Справочные материалы

В этой главе были упомянуты следующие статьи.

- *Black F., Scholes M.* The Pricing of Options and Corporate Liabilities // Journal of Political Economy, 1973. — № 81 (3). — P. 638–659.
- *Harrison M., Kreps D.* Martingales and Arbitrage in Multiperiod Securities Markets // Journal of Economic Theory, 1979. — № (20). — P. 381–408.
- *Harrison M., Pliska S.* Martingales and Stochastic Integrals in the Theory of Continuous Trading // Stochastic Processes and their Applications, 1981. — № (11). — P. 215–260.
- *Merton R.* Option Pricing when the Underlying Stock Returns are Discontinuous // Journal of Financial Economics, 1976. — № 3 (3). — P. 125–144.
- *Merton R.* Theory of Rational Option Pricing // Bell Journal of Economics and Management Science, 1973. — № (4). — P. 141–183.

В этой главе были упомянуты следующие книги.

- *Aleskerov F, Ersel Hasan, Piontkovski D.* Linear Algebra for Economists. — Heidelberg: Springer, 2011.
- *Delbaen F., Schachermayer W.* The Mathematics of Arbitrage. — Berlin: Springer Verlag, 2006.
- *Duffie D.* Security Markets — Stochastic Models. — San Diego: Academic Press, 1988.
- *Jacod J., Protter P.* Probability Essentials. — Berlin and Heidelberg: Springer, 2004.
- *Milne F.* Finance Theory and Asset Pricing. — N. Y.: Oxford University Press, 1995.
- *Pliska S.* Introduction to Mathematical Finance. — Malden and Oxford: Blackwell Publishers, 1997.

ГЛАВА 6

Динамическая экономика

> Многопериодные модели рынков ценных бумаг гораздо более реалистичны, чем однопериодные, и широко используются в финансовой практике.
>
> *Стэнли Плиска (1997)*

> Рынки не являются полными в каждый момент времени. Однако они являются динамично полными в том смысле, что любой процесс потребления может быть профинансирован путем торговли данной группой финансовых ценных бумаг с временной корректировкой портфелей по мере постепенного устранения неопределенности.
>
> *Даррел Дюффе (1986)*

В реальности количественная информация, например изменение цен акции или процентных ставок, раскрывается постепенно с течением времени. С помощью статичных модельных экономик можно без труда ввести основные финансовые понятия, однако реалистичная финансовая модель требует динамического представления финансового мира.

Формулы, необходимые для правильного моделирования динамических экономик, сложнее тех, с которыми вы познакомились ранее. В этой главе не будет их подробного разбора, зато мы рассмотрим две наиболее важные динамические модельные экономики, основанные на динамике дискретного

времени: модель биномиального ценообразования опционов Кокса — Росса — Рубинштейна (1979) и модель ценообразования опционов Блэка — Шоулза — Мертона в ее дискретной версии моделирования методом Монте-Карло. Под *дискретным временем* понимается увеличение количества релевантных дат от двух до большего, но все еще конечного числа, скажем до 5 или 50.

Используемые в этой главе инструменты почти те же, что и раньше: линейная алгебра, теория вероятностей, а также стохастические элементы для реализации моделирования методом Монте-Карло. Отличными источниками по теме создания динамической финансовой модели с дискретным временем являются работы Дюффе и Плиски, а Глассерман дает всеобъемлющую справочную информацию по финансовому моделированию методом Монте-Карло.

В этой главе будут затронуты такие темы, как стохастические процессы, ценообразование опционов на динамично полных рынках, биномиальное ценообразование опционов, динамическое моделирование Блэка — Шоулза — Мертона, досрочное исполнение и ценообразование американских опционов, а также ценообразование опционов методом наименьших квадратов Монте-Карло.

В таблице приведен краткий перечень основных тем главы.

Финансы	Математика	Python
Неопределенность, древовидное решение	Стохастический процесс, биномиальное дерево	NumPy, `ndarray`
Неопределенность, решение на основе моделирования	Стохастический процесс, моделирование методом Монте-Карло	NumPy, `ndarray`, `rng.standard_normal`
Ценообразование европейского опциона	Внутренняя стоимость, обратная индукция, риск-нейтральное математическое ожидание	NumPy, `ndarray`, `np.maximum`
Ценообразование американского опциона	Внутренняя стоимость, продленная стоимость, регрессия МНК, обратная индукция, риск-нейтральное математическое ожидание	NumPy, `ndarray`, `np.polyval`, `np.polyfit`, `np.where`

Основной целью данной главы также является *обобщение*. Однако если в главе 5 обобщается пространство состояний, то здесь — *дискретное множество релевантных моментов времени*, в которые появляется новая информация и происходят события, связанные с экономикой. Несмотря на то что такое

обобщение требует введения дополнительных формул, здесь они даны в минимальном количестве, поскольку глава не пытается представить общую структуру динамичных экономик с дискретным временем, а фокусируется только на двух конкретных моделях.

Биномиальное ценообразование опционов

Биномиальная модель ценообразования опционов стала популярной сразу после своей публикации в 1979 году как в качестве численно эффективного метода ценообразования европейских и американских опционов, так и в качестве учебного пособия по данной теме. Если модель Блэка — Шоулза — Мертона основывается на непрерывности времени и стохастическом исчислении, то биномиальная модель в некотором смысле является ее дискретной версией, хорошо разобраться в которой можно и с помощью элементарной математики.

В модели Кокса — Росса — Рубинштейна существует два торгуемых финансовых актива: рисковый (акция) и безрисковый (облигация). Модельная экономика рассматривается на *конечном множестве дат* $\mathfrak{I} \equiv \{t_0 = 0, t_1, t_2 ... t_M = T\}$ с элементами $M + 1$, $M > 1$.

При условии, что в начале стоимость акции равна S_{t_i}, в следующем моменте времени ее цена $S_{t_{i+1}}$ может принять только два значения:

$$S_{t_{i+1}} = \begin{cases} S_{t_i} \cdot u; \\ S_{t_i} \cdot d, \end{cases}$$

где u означает *повышение цены*, а d — ее *понижение*[1].

Для упрощения предположим равномерное распределение временной сетки с M временных интервалов, длина каждого из которых составляет $\Delta t = T / M$. Тогда конечное множество дат можно записать в виде $\mathfrak{I} \equiv \{t_0 = 0, t_1 = \Delta t, t_2 = 2\Delta t ... T\}$. Кроме того, определим, что:

$$\begin{cases} u \equiv e^{\sigma\sqrt{\Delta t}}; \\ d \equiv e^{-\sigma\sqrt{\Delta t}} = u^{-1}. \end{cases}$$

[1] u — upward movement, d — downward movement. — *Примеч. пер.*

Таким образом мы получаем свойство $u \cdot d = 1$, которое создает так называемое *рекомбинирующее* биномиальное дерево. $\sigma \in \mathbb{R}_{>0}$ представляет собой постоянный *коэффициент волатильности*.

Предположим, что постоянная краткосрочная безрисковая процентная ставка задана через $r \in \mathbb{R}_{\geq 0}$. При условии, что цена облигации равна B_{t_i}, в следующем временном периоде она составит:

$$B_{t_{i+1}} = B_{t_i} \cdot e^{r(t_{i+1}-t_i)}$$

или для некоторых значений $t \in \Im \backslash T$

$$B_{t+\Delta t} = B_t e^{r\Delta t}.$$

Основное числовое значение параметра, которое необходимо получить, исходя из предыдущих предположений, является мартингальной вероятностью повышения цен на любом заданном узле дерева. Так как для каждого узла существует только две ветви, вероятность понижения цены также известна. Обозначим *мартингальную вероятность* повышения цены через $q \in \mathbb{R}_{>0}$, $0 < q < 1$. Из свойства мартингала для цены акции следует:

$$S_t = e^{-r\Delta t}\mathbf{E}^Q\left(S_{t+\Delta t}\right) = e^{-r\Delta t}\left(qS_t u + (1-q)S_t d\right) \Leftrightarrow$$
$$\Leftrightarrow 1 = e^{-r\Delta t}\left(qu + (1-q)d\right) \Leftrightarrow q = \frac{e^{r\Delta t} - d}{u - d}.$$

Здесь видно, что мартингальная мера фиксирована в каждом узле и, следовательно, на всем дереве.

Основы биномиальной модели ценообразования опционов легко реализуются кодом Python:[1]

```
In [1]: import math
        import numpy as np

In [2]: S0 = 36. ❶
        K = 40. ❷
        r = 0.06 ❸
        T = 1.0 ❹
        sigma = 0.2 ❺
```

[1] Используемые в этой главе параметры взяты из работы Лонгстаффа и Шварца (2001, табл. 1).

```
In [3]: m = 4 ❻
        dt = T / m ❼
        df = math.exp(-r * dt) ❽
        up = math.exp(sigma * math.sqrt(dt)) ❾
        down = 1 / up ❾

In [4]: q = (1 / df - down) / (up - down) ❿
```

❶ Начальная стоимость акции.

❷ Цена исполнения оцениваемого опциона.

❸ Постоянная безрисковая краткосрочная процентная ставка.

❹ Горизонт расчета и срок погашения опциона.

❺ Постоянный коэффициент волатильности.

❻ Количество временных интервалов.

❼ Длина каждого временного интервала.

❽ Коэффициент дисконтирования в установленном временном интервале.

❾ Повышающий и понижающий коэффициенты.

❿ Мартингальная вероятность повышения цены.

Формирование процесса ценообразования акции и оценка опционов в этой модели требует чуть больших умений. Далее представлены два способа реализации моделирования и оценки: один основан на простых для понимания *циклах Python*, а другой — на более лаконичном и эффективном, но более сложном *векторизованном коде* NumPy.

Моделирование и оценка посредством циклов Python

Несмотря на то что в этом разделе идет речь о циклах Python, основной задействованной в реализации структурой данных является объект NumPy `ndarray`:

```
In [5]: S = np.zeros((m + 1, m + 1)) ❶
        S ❶
Out[5]: array([[0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.],
               [0., 0., 0., 0., 0.]])
```

```
In [6]: S[0, 0] = S0 ❷
        S ❷
Out[6]: array([[36.,  0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.,  0.],
               [ 0.,  0.,  0.,  0.,  0.]])

In [7]: z = 1 ❸
        for t in range(1, m + 1): ❹
            for i in range(0, z): ❺
                S[i, t] = S[i, t - 1] * up ❻
                S[i + 1 ,t] = S[i, t - 1] * down ❻
            z += 1 ❼

In [8]: np.set_printoptions(formatter=
                {'float_kind': lambda x: '%7.3f' % x})

In [9]: S ❽
Out[9]: array([[ 36.000,  39.786,  43.970,  48.595,  53.706],
               [  0.000,  32.574,  36.000,  39.786,  43.970],
               [  0.000,   0.000,  29.474,  32.574,  36.000],
               [  0.000,   0.000,   0.000,  26.669,  29.474],
               [  0.000,   0.000,   0.000,   0.000,  24.132]])
```

❶ Начало работы с объектом `ndarray`.

❷ Установление начальной цены акции в левом верхнем углу матрицы.

❸ Установление счетчика `z` на значение `1`.

❹ Переход от `1` к `m+1`, то есть проход по всем временным шагам после 0.

❺ Проход по узлам, релевантным для заданного временного шага.

❻ Расчет верхнего и нижнего значений с последующим их помещением в объект `ndarray`.

❼ Увеличение счетчика на единицу для включения большего количества релевантных узлов на следующем шаге.

❽ Полученное в результате предыдущих действий рекомбинирующее биномиальное дерево.

Ценообразование европейского опциона

Оценка европейского опциона на основе доступного процесса ценообразования акции происходит путем расчета внутренней стоимости опциона при наступлении срока его погашения и применения *обратной индукции*. То есть необходимо начинать с конца, двигаться шаг за шагом от будущего к настоящему времени и в каждом узле раз за разом применять парадигму риск-нейтрального ценообразования, как было представлено на примере простой статичной экономики с двумя состояниями (см. главу 2).

Следующий код представляет выплату по европейскому пут-опциону:

```
In [10]: h = np.zeros_like(S) ❶

In [11]: z = 1
         for t in range(0, m + 1):
             for i in range(0, z):
                 h[i, t] = max(K - S[i, t], 0) ❷
             z += 1

In [12]: h ❷
Out[12]: array([[  4.000,  0.214,  0.000,  0.000,  0.000],
                [  0.000,  7.426,  4.000,  0.214,  0.000],
                [  0.000,  0.000, 10.526,  7.426,  4.000],
                [  0.000,  0.000,  0.000, 13.331, 10.526],
                [  0.000,  0.000,  0.000,  0.000, 15.868]])

In [13]: V = np.zeros_like(S)
         V[:, -1] = h[:, -1]
         V
Out[13]: array([[  0.000,  0.000,  0.000,  0.000,  0.000],
                [  0.000,  0.000,  0.000,  0.000,  0.000],
                [  0.000,  0.000,  0.000,  0.000,  4.000],
                [  0.000,  0.000,  0.000,  0.000, 10.526],
                [  0.000,  0.000,  0.000,  0.000, 15.868]])

In [14]: m
Out[14]: 4

In [15]: # Ценообразование европейского опциона
         z = 0
         for t in range(m - 1, -1, -1):
             for i in range(0, m - z):
```

```
                V[i, t] = df * (q * V[i, t + 1] +
                               (1-q) * V[i + 1, t + 1]) ❸
            z += 1

In [16]: V ❹
Out[16]: array([[  3.977,  2.190,  0.784,  0.000,  0.000],
                [  0.000,  6.299,  3.985,  1.771,  0.000],
                [  0.000,  0.000,  9.344,  6.830,  4.000],
                [  0.000,  0.000,  0.000, 12.735, 10.526],
                [  0.000,  0.000,  0.000,  0.000, 15.868]])

In [17]: V[0, 0] ❺
Out[17]: 3.9771456941187893
```

❶ Объект `ndarray` для внутренней стоимости.

❷ Расчет внутренней стоимости для релевантных узлов.

❸ Оценка по узлам с использованием риск-нейтрального ценообразования.

❹ Полученное в результате предыдущих действий биномиальное дерево приведенной стоимости.

❺ Приведенная стоимость европейского пут-опциона сегодня.

Ценообразование американского опциона

Одна из главных особенностей биномиальной модели ценообразования опционов заключается в том, что американские опционы оцениваются так же легко, как и европейские. Исполнение *американского опциона* может быть произведено в любое время *до наступления срока погашения и при его наступлении*. Корректировка, которую необходимо внести в алгоритм обратной оценки, проста: требуется проверить, является ли внутренняя стоимость американского опциона в любой заданный момент времени выше ее продленной стоимости, то есть приведенной стоимости неисполненного опциона. Если проверка имеет положительный результат, то опцион считается исполняемым и стоимость американского опциона приравнивается к его внутренней стоимости, что можно выразить следующей формулой:

$$V_t = \max\left[h_t, e^{-r\Delta t}\mathbf{E}^Q\left(V_{t+\Delta t}\right)\right],$$

где h_t — внутренняя стоимость опциона в момент времени t, а $e^{-r\Delta t}\mathbf{E}^Q\left(V_{t+\Delta t}\right)$ — его продленная стоимость.

Для того чтобы предыдущий код успешно реализовал ценообразование американского опциона, нужно добавить к нему всего одну строку:

```
In [18]: # Ценообразование американского опциона
         z = 0
         for t in range(m - 1, -1, -1):
             for i in range(0, m-z):
                 V[i, t] = df * (q * V[i, t + 1] +
                           (1 - q) * V[i + 1, t + 1])
                 V[i, t] = max(h[i, t], V[i, t]) ❶
             z += 1
In [19]: V ❷
Out[19]: array([[  4.540,  2.307,  0.784,  0.000,  0.000],
                [  0.000,  7.426,  4.249,  1.771,  0.000],
                [  0.000,  0.000, 10.526,  7.426,  4.000],
                [  0.000,  0.000,  0.000, 13.331, 10.526],
                [  0.000,  0.000,  0.000,  0.000, 15.868]])
In [20]: V[0, 0] ❸
Out[20]: 4.539560595224299
```

❶ Данная строка проверяет решение о досрочном исполнении и определяет внутреннюю стоимость в качестве стоимости американского опциона, если она выше продленной стоимости.

❷ Полученное в результате этих действий биномиальное дерево для приведенной стоимости американского пут-опциона.

❸ Приведенная стоимость американского пут-опциона сегодня, которая значительно выше, чем его стоимость при наступлении срока погашения.

Моделирование и оценка посредством векторизованного кода

Следующая реализация алгоритма планомерно использует возможности векторизации библиотеки NumPy. Некоторые строки кода выполняют только поясняющую функцию и для самого алгоритма не нужны:

```
In [21]: u = np.arange(m + 1) ❶
         u ❶
Out[21]: array([0, 1, 2, 3, 4])

In [22]: u ** 2 ❷
Out[22]: array([ 0, 1, 4, 9, 16])
```

```
In [23]: 2 ** u ❸
Out[23]: array([ 1, 2, 4, 8, 16])

In [24]: u = np.resize(u, (m + 1, m + 1)) ❹
         u
Out[24]: array([[0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4],
                [0, 1, 2, 3, 4]])

In [25]: d = u.T ❺
         d ❺
Out[25]: array([[0, 0, 0, 0, 0],
                [1, 1, 1, 1, 1],
                [2, 2, 2, 2, 2],
                [3, 3, 3, 3, 3],
                [4, 4, 4, 4, 4]])

In [26]: (u - 2 * d) ❻
Out[26]: array([[ 0, 1, 2, 3, 4],
                [-2, -1, 0, 1, 2],
                [-4, -3, -2, -1, 0],
                [-6, -5, -4, -3, -2],
                [-8, -7, -6, -5, -4]])
```

❶ Создание объекта `ndarray` для количества повышения цены от `0` до `m`.

❷ Расчет квадратов через векторную операцию.

❸ Расчет чисел, кратных 2, с использованием объекта `u` в качестве векторной экспоненты.

❹ Изменение размера объекта `u` с одного измерения на два. Количество повышения цены теперь хранится в каждом ряду.

❺ Транспозиция объекта `u` для получения двумерного объекта `ndarray d`. В каждом столбце находится количество повышений цен.

❻ Комбинация объектов `u` и `d` для получения чистого количества повышений и понижений цен. Например, `+2` означает «повышение больше на 2 пункта, чем понижение», а `–1` — «понижение больше на один пункт, чем повышение»[1].

[1] Здесь стоит обратить внимание на то, что имеют значение только те числа, которые расположены на диагонали матрицы и выше ее. Числа, находящиеся ниже диагонали, можно игнорировать. Они являются результатом специфических векторных операций, реализованных на объекте ndarray.

При наличии матрицы, содержащей чистое количество изменений цен в биномиальном дереве, моделирование процесса ценообразования сводится к одной строке кода:

```
In [27]: S = S0 * np.exp(sigma * math.sqrt(dt) * (u - 2 * d)) ❶
         S ❷
Out[27]: array([[ 36.000,  39.786,  43.970,  48.595,  53.706],
                [ 29.474,  32.574,  36.000,  39.786,  43.970],
                [ 24.132,  26.669,  29.474,  32.574,  36.000],
                [ 19.757,  21.835,  24.132,  26.669,  29.474],
                [ 16.176,  17.877,  19.757,  21.835,  24.132]])
```

❶ Векторное моделирование процесса ценообразования акции (биномиальное дерево).

❷ Рассматриваются только те числа, которые расположены на диагонали матрицы и выше ее.

Следует отметить, что оценка как европейских, так и американских пут-опционов является в некоторой степени векторизованной. Остается лишь один перебор временных шагов:

```
In [28]: h = np.maximum(K - S, 0) ❶
         h ❷
Out[28]: array([[  4.000,  0.214,  0.000,  0.000,  0.000],
                [ 10.526,  7.426,  4.000,  0.214,  0.000],
                [ 15.868, 13.331, 10.526,  7.426,  4.000],
                [ 20.243, 18.165, 15.868, 13.331, 10.526],
                [ 23.824, 22.123, 20.243, 18.165, 15.868]])

In [29]: V = h.copy() ❸

In [30]: # Ценообразование европейского опциона
         for t in range(m - 1, -1, -1): ❹
             V[0:-1, t] = df * (q * V[:-1, t + 1] +
                                (1-q) * V[1:, t + 1]) ❹

In [31]: V[0, 0] ❺
Out[31]: 3.977145694118792

In [32]: # Ценообразование американского опциона
         for t in range(m - 1, -1, -1): ❻
             V[0:-1, t] = df * (q * V[:-1, t + 1] +
                                (1-q) * V[1:, t + 1]) ❻
             V[:, t] = np.maximum(h[:, t], V[:, t]) ❻

In [33]: V
```

```
Out[33]: array([[  4.540,  2.307,  0.784,  0.000,  0.000],
                [ 10.526,  7.426,  4.249,  1.771,  0.000],
                [ 15.868, 13.331, 10.526,  7.426,  4.000],
                [ 20.243, 18.165, 15.868, 13.331, 10.526],
                [ 23.824, 22.123, 20.243, 18.165, 15.868]])

In [34]: V[0, 0] ❼
Out[34]: 4.5395605952243
```

❶ Полностью векторизованный расчет внутренней стоимости пут-опциона.

❷ Рассматриваются только те числа, которые расположены на диагонали матрицы и выше ее.

❸ Создание копии объекта `h`.

❹ Частично векторизованный алгоритм оценки стоимости европейского пут-опциона.

❺ Приведенная стоимость европейского пут-опциона.

❻ Частично векторизованный алгоритм оценки стоимости американского пут-опциона.

❼ Приведенная стоимость американского пут-опциона.

Европейские и американские опционы

Прелесть рекомбинирующей биномиальной модели ценообразования опционов, разработанной Коксом, Россом и Рубинштейном, заключается не только в ее простоте, но и в высокоточных результатах, которые она дает при оценке как европейских, так и американских опционов. В пределе, бесконечно уменьшая временные интервалы, она сходится к модели Блэка — Шоулза — Мертона, что является еще одним ее преимуществом.

Сравнение скорости работы

Векторизация не только делает код Python более лаконичным, но и в целом позволяет значительно повысить скорость работы с ним. Докажем это утверждение с помощью сравнения следующих фрагментов кода, реализующих предыдущие алгоритмы на большем, более реалистичном числе временных шагов. Для начала настроим основные числовые параметры:

```
In [35]: m = 500 ❶
         dt = T / m
```

```
        df = math.exp(-r * dt)
        up = math.exp(sigma * math.sqrt(dt))
        down = 1 / up
        q = (1 / df - down) / (up - down)
        q
Out[35]: 0.5044724639230862
```

❶ Увеличение количества временных интервалов до реалистичного уровня, в результате чего получаются довольно точные числовые значения стоимости опционов.

Функция `binomial_looping()` объединяет все элементы *циклической реализации* алгоритма моделирования и оценки американского пут-опциона:

```
In [36]: def binomial_looping():
             # Моделирование цен акции в виде биномиального дерева
             S = np.zeros((m + 1, m + 1))
             S[0, 0] = S0
             z = 1
             for t in range(1, m + 1):
                 for i in range(0, z):
                     S[i, t] = S[i, t - 1] * up
                     S[i + 1 ,t] = S[i, t - 1] * down
                 z += 1
             # Расчет внутренней стоимости
             h = np.zeros_like(S)
             z = 1
             for t in range(0, m + 1):
                 for i in range(0, z):
                     h[i, t] = max(K - S[i, t], 0)
                 z += 1
             # Ценообразование американского опциона
             V = np.zeros_like(S)
             V[:, -1] = h[:, -1]
             z = 0
             for t in range(m - 1, -1, -1):
                 for i in range(0, m - z):
                     V[i, t] = df * (q * V[i, t + 1] +
                               (1 - q) * V[i + 1, t + 1])
                     V[i, t] = max(h[i, t], V[i, t])
                 z += 1
             return V[0, 0]
```

Как правило, выполнение кода занимает не более 200 мс:

```
In [37]: %time binomial_looping()
         CPU times: user 190 ms, sys: 4.69 ms, total: 194 ms
         Wall time: 190 ms
```

```
Out[37]: 4.486374777505983

In [38]: %timeit binomial_looping()
         173 ms ± 2.48 ms per loop (mean ± std. dev. of 7 runs, 10 loops each)
```

Функция `binomial_vectorization()` объединяет все элементы *векторной реализации* алгоритма моделирования и оценки американского пут-опциона:

```
In [39]: def binomial_vectorization():
             u = np.arange(m + 1)
             u = np.resize(u, (m + 1, m + 1))
             d = u.T
             # Моделирование цен акции
             S = S0 * np.exp(sigma * math.sqrt(dt) * (u - 2 * d))
             # Расчет внутренней стоимости
             h = np.maximum(K - S, 0)
             # Ценообразование американского опциона
             V = h.copy()
             for t in range(m-1, -1, -1):
                 V[0:-1, t] = df * (q * V[:-1, t + 1] +
                                    (1-q) * V[1:, t + 1])
                 V[:, t] = np.maximum(h[:, t], V[:, t])
             return V[0, 0]
```

Векторная реализация примерно в 40 раз быстрее циклической, что явно говорит в пользу первой с точки зрения производительности:

```
In [40]: %time binomial_vectorization()
         CPU times: user 4.67 ms, sys: 2.39 ms, total: 7.07 ms
         Wall time: 8.73 ms

Out[40]: 4.486374777506075

In [41]: %timeit binomial_vectorization()
         4.7 ms ± 252 µs per loop (mean ± std. dev. of 7 runs, 100 loops each)
```

Инфраструктура и производительность

Представленная здесь абсолютная скорость зависит как от используемого оборудования, так и от конфигурации программного обеспечения. Например, можно применять NumPy в сочетании с библиотекой Math Kernel Library (MKL) от Intel. Такая сборка значительно ускоряет выполнение числовых операций на системах, работающих на базе процессоров Intel. В свою очередь, относительная скорость и коэффициент ускорения должны быть примерно одинаковыми вне зависимости от используемой инфраструктуры.

Модель ценообразования опционов Блэка — Шоулза — Мертона

В данном разделе представлена *динамическая* модель ценообразования опционов Блэка — Шоулза — Мертона, статическую версию которой мы рассматривали в предыдущей главе.

Динамика в экономику Блэка — Шоулза — Мертона вводится через стохастическое дифференциальное уравнение:

$$dS_t = rS_t dt + \sigma S_t dZ_t,$$

где $S_t \in \mathbb{R}_{>0}$ — цена акции в момент времени t, $r \in \mathbb{R}_{\geq 0}$ — постоянная краткосрочная ставка, $\sigma \in \mathbb{R}_{>0}$ — постоянный коэффициент волатильности, а Z_t — арифметическое броуновское движение[1].

Моделирование ценовой траектории акций методом Монте-Карло

Предположим, что имеется конечное множество релевантных моментов времени $I \equiv \{t_0 = 0, t_1, t_2 \ldots t_M = T\}$ с элементами $M + 1$, $M > 1$ и зафиксированной длиной интервала Δt. Стоимость акции S_t, $0 < t \leq T$, с учетом предыдущей цены на нее $S_{t-\Delta t}$ может быть смоделирована через дифференциальное уравнение:

$$S_t = S_{t-\Delta t} \cdot \exp\left(\left(r - \frac{\sigma^2}{2}\right)\Delta t + \sigma\sqrt{\Delta t}\, z\right),$$

где z — стандартная нормально распределенная случайная величина. Такое уравнение называется *дискретизацией Эйлера*. Она известна своей точностью, поскольку обеспечивает сходимость дискретного процесса к непрерывному процессу для значения Δt, которое, в свою очередь, сходится к 0.

Реализация динамического моделирования методом Монте-Карло на языке Python не составит труда при наличии базовых знаний о статическом

[1] См.: Glasserman, Monte Carlo Methods in Financial Engineering (2004, глава 3) и «Python для финансовых расчетов» (2018, глава 12).

моделировании из главы 5. На рис. 6.1 показаны десять смоделированных ценовых траекторий акции:

```
In [43]: S0 = 36. ❶
         K = 40. ❶
         r = 0.06 ❶
         T = 1.0 ❶
         sigma = 0.2 ❶

In [44]: M = 100 ❷
         I = 50000 ❷

In [45]: dt = T / M ❷
         dt ❷
Out[45]: 0.01

In [46]: df = math.exp(-r * dt) ❷
         df ❷
Out[46]: 0.9994001799640054

In [47]: from numpy.random import default_rng
         rng = default_rng(100)

In [48]: rn = rng.standard_normal((M + 1, I)) ❸
         rn ❸
Out[48]: array([[ -1.160,  0.290,  0.780, ...,  1.890,  0.050, -0.760],
                [  0.460, -1.400,  0.140, ..., -1.350,  0.150, -0.530],
                [  0.200, -0.040, -0.730, ...,  2.140,  0.170, -0.340],
                ...,
                [ -0.220, -1.310,  0.730, ..., -0.820, -0.600, -0.400],
                [ -2.130, -1.240,  0.580, ...,  0.960,  0.890,  0.780],
                [  2.130, -0.410,  0.710, ...,  1.190,  0.100, -0.520]])
In [49]: S = np.zeros_like(rn) ❹
         S[0] = S0 ❹
         S ❹
Out[49]: array([[ 36.000, 36.000, 36.000, ..., 36.000, 36.000, 36.000],
                [  0.000,  0.000,  0.000, ...,  0.000,  0.000,  0.000],
                [  0.000,  0.000,  0.000, ...,  0.000,  0.000,  0.000],
                ...,
                [  0.000,  0.000,  0.000, ...,  0.000,  0.000,  0.000],
                [  0.000,  0.000,  0.000, ...,  0.000,  0.000,  0.000],
                [  0.000,  0.000,  0.000, ...,  0.000,  0.000,  0.000]])

In [50]: for t in range(1, M + 1):
         S[t] = S[t - 1] * np.exp((r - sigma ** 2 / 2) * dt +
         sigma * math.sqrt(dt) * rn[t]) ❺
```

```
In [51]: S ❺
Out[51]: array([[ 36.000,  36.000,  36.000, ...,  36.000,  36.000,  36.000],
                [ 36.349,  35.023,  36.114, ...,  35.056,  36.119,  35.633],
                [ 36.508,  35.009,  35.602, ...,  36.601,  36.259,  35.402],
                ...,
                [ 42.689,  39.760,  40.681, ...,  37.516,  47.893,  42.846],
                [ 40.921,  38.804,  41.175, ...,  38.260,  48.769,  43.534],
                [ 42.716,  38.499,  41.782, ...,  39.200,  48.884,  43.103]])

In [52]: from pylab import mpl, plt
         plt.style.use('seaborn')
         mpl.rcParams['font.family'] = 'serif'
         mpl.rcParams['savefig.dpi'] = 300

In [53]: plt.figure(figsize=(10, 6))
         plt.plot(S[:, :10]); ❻
```

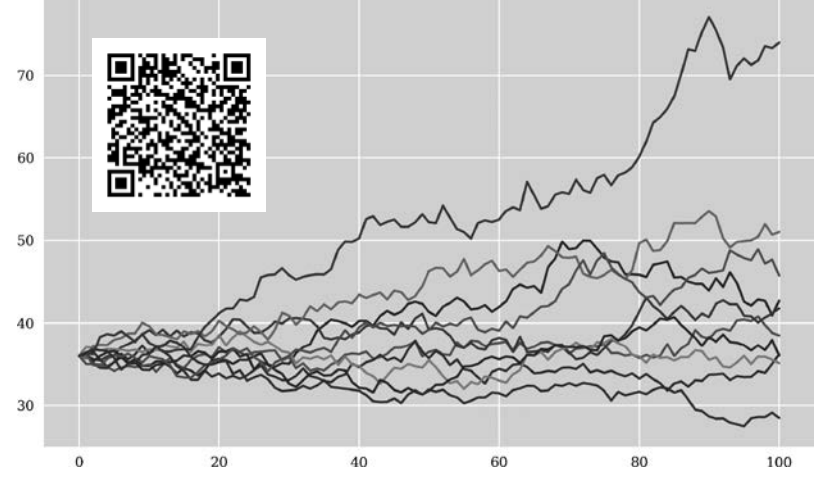


Рис. 6.1. Смоделированные ценовые траектории акции по Блэку — Шоулзу — Мертону

❶ Финансовые параметры остаются прежними.

❷ Параметры моделирования методом Монте-Карло (ценовые траектории, временные шаги, длина временного интервала, коэффициент дисконтирования в одном временном интервале).

❸ Генерация двумерного объекта `ndarray` со стандартными нормально распределенными случайными величинами соответствующего размера.

❹ Создание еще одного двумерного объекта `ndarray` той же формы и установление начальных значений для ценовой траектории отдельной акции.

❺ Моделирование ценовых траекторий отдельной акции на основе ее начальной стоимости, матрицы случайных чисел и дифференциального уравнения для геометрического броуновского движения.

❻ Построение графика первых десяти смоделированных траекторий

Как и в случае со статической моделью, стоимость акции на конец периода можно представить в виде гистограммы (рис. 6.2):

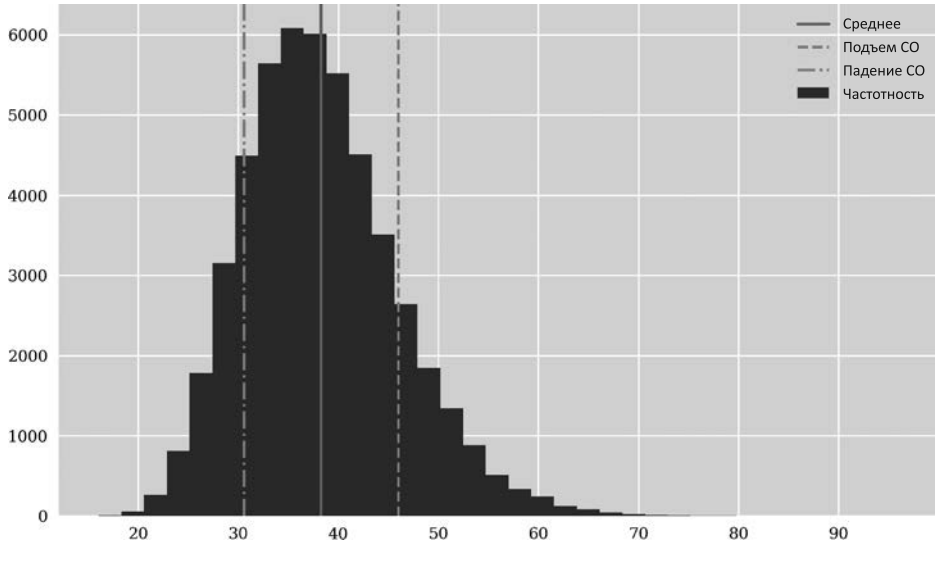


Рис. 6.2. Частотное распределение смоделированных цен акции в конце периода по Блэку — Шоулзу — Мертону

```
In [54]: ST = S[-1]
         plt.figure(figsize=(10, 6))
         plt.hist(ST, bins=35, color='b', label='frequency');
         plt.axvline(ST.mean(), color='r', label='mean')
         plt.axvline(ST.mean() + ST.std(), ls='--', color='y', label='sd up')
         plt.axvline(ST.mean() - ST.std(), ls='-.', color='y', label='sd down')
         plt.legend(loc=0);

In [55]: S0 * math.exp(r * T) ❶
Out[55]: 38.22611567563295
```

```
In [56]: ST.mean() ❷
Out[56]: 38.25248936738523
```

❶ Математическое ожидание для S_t.

❷ Усреднение по всем смоделированным значениям ST.

Оценка европейского пут-опциона методом Монте-Карло

Стоимость европейского пут-опциона методом Монте-Карло можно оценить через:

$$P_0 = e^{-rT} \frac{1}{I} \sum_{i=1}^{I} \max\left(K - S_T(i), 0\right),$$

где I — количество смоделированных ценовых траекторий. Таким образом, ценообразование европейского пут-опциона сводится к нескольким строкам кода, реализующим моделирование ценовых траекторий акции. На рис. 6.3 показана гистограмма смоделированной внутренней стоимости опциона при наступлении срока погашения:

```
In [57]: h = np.maximum(K - ST, 0) ❶
         h ❶
Out[57]: array([ 0.000, 1.501, 0.000, ..., 0.800, 0.000, 0.000])

In [58]: plt.figure(figsize=(10, 6))
         plt.hist(h, color='b', bins=35); ❷

In [59]: math.exp(-r * T) * h.mean() ❸
Out[59]: 3.818117261795047
```

❶ Векторный расчет внутренней стоимости.

❷ Построение графика частотного распределения внутренней стоимости опциона при наступлении срока погашения, на котором изображены характерные для него крайне асимметричные выплаты[1].

❸ Расчет среднего значения по всей внутренней стоимости и дисконтирование среднего значения к настоящему моменту.

[1] Здесь, как это часто бывает на практике, существует много случаев, когда опцион истекает, не принося прибыли, то есть с выплатой, равной нулю.

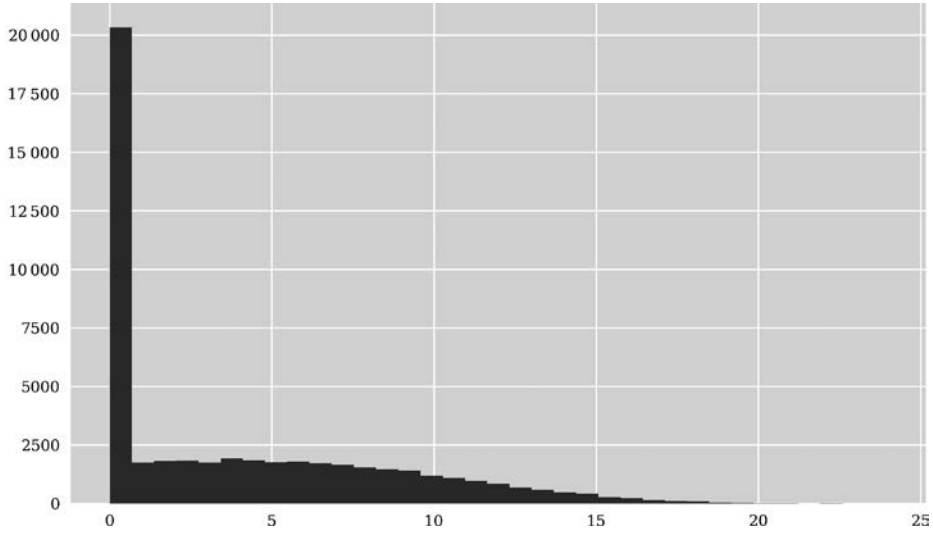


Рис. 6.3. Частотное распределение смоделированной внутренней стоимости акции при наступлении срока погашения европейского пут-опциона

Оценка американского пут-опциона методом Монте-Карло

Немного сложнее происходит оценка американских пут-опционов на основе моделирования методом Монте-Карло. Наиболее популярным алгоритмом в этом отношении является *ценообразование опционов методом наименьших квадратов Монте-Карло*, разработанное Лонгстаффом и Шварцем (2001) из-за своей относительной простоты и эффективности в применении с численной и вычислительной точки зрения. Мы не будем подробно описывать данный подход, а только представим ее краткую, сильно векторизованную реализацию на языке Python.

Вот как реализуется метод наименьших квадратов Монте-Карло для оценки американских опционов на языке Python:

```
In [60]: h = np.maximum(K - S, 0) ❶

In [61]: # Оценка методом наименьших квадратов Монте-Карло
         V = h[-1] ❷
         for t in range(M - 1, 0, -1): ❸
             reg = np.polyfit(S[t], df * V, deg=5) ❹
```

```
            C = np.polyval(reg, S[t]) ❹
            V = np.where(h[t] > C, h[t], df * V) ❺

In [62]: df * V.mean() ❻
Out[62]: 4.454837750511421
```

❶ Расчет внутренней стоимости опциона на всей его ценовой траектории, выраженной объектом `ndarray`.

❷ Приравнивание начальных смоделированных цен американского опциона к его внутренней стоимости при наступлении срока погашения.

❸ Алгоритм работает также на основе обратной индукции, начиная с $T - \Delta t$ и заканчивая Δt.

❹ Основной этап алгоритма, в котором устанавливается (приблизительно определяется) продленная стоимость на основе регрессии МНК текущей смоделированной стоимости опциона относительно уровней цен на акцию.

❺ Если внутренняя стоимость выше предполагаемой (приблизительной) продленной стоимости, опцион исполняем, в противном случае — нет.

❻ Приведенная стоимость рассчитывается как среднее значение вектора цены американского опциона в момент времени $t = \Delta t$, полученное на основе алгоритма наименьших квадратов Монте-Карло и дисконтированное для последнего временного интервала к настоящему времени $t = 0$.

Досрочное исполнение опциона и моделирование методом Монте-Карло

Проблема эффективной оценки стоимости опционов и деривативов с функцией досрочного исполнения на основе моделирования Монте-Карло оставалась нерешенной до конца 1990-х годов. Только в начале 2000-х исследователи смогли предложить оптимальные алгоритмы расчета для работы с досрочным исполнением в финансовом моделировании. Как часто бывает в науке и финансах, как только вводится какой-нибудь новый алгоритм (например, метод наименьших квадратов Монте-Карло), его реализация и применение сразу начинают казаться вполне естественными. В сущности, для точной оценки стоимости американского пут-опциона посредством моделирования, представленной в данном разделе, требуется всего несколько строк кода Python. Тем не менее метод наименьших квадратов Монте-Карло следует считать крупным прорывом, поспособствовавшим началу вычислительного периода в финансах (см. главу 1).

Резюме

В этой главе с минимальным использованием формул были представлены две популярные динамически полные финансовые модели. Первая — так называемая модель рекомбинирующего биномиального дерева Кокса — Росса — Рубинштейна. Ее прелесть заключается в простоте и эффективности для реализации ценообразования европейских и американских опционов на основе только математики, изучаемой в старших классах. Данная модель — хороший обучающий инструмент по сравнению с непрерывными финансовыми моделями, которые требуют углубленного стохастического исчисления.

Вторая модель представляет собой *динамическую версию* непрерывной модели ценообразования опционов Блэка — Шоулза — Мертона, *разработанной по методу Монте-Карло*. Динамическое моделирование методом Монте-Карло может быть реализовано численно эффективным образом посредством использования библиотеки NumPy. Даже требовательный к вычислительным ресурсам и включающий в себя трудоемкую регрессию метод наименьших квадратов Монте-Карло, разработанный Лонгстаффом и Шварцем (2001), работает довольно быстро, если реализован на основе методов векторизации.

В целом, мы увидели, что имеющая мощные возможности векторизации библиотека NumPy позволяет не только создавать лаконичный алгоритмический код, но и быстро выполнять его даже в ходе работы с более реалистичными и сложными динамичным модельными экономиками.

Справочные материалы

В этой главе были упомянуты следующие статьи.

- *Black F., Scholes M.* The Pricing of Options and Corporate Liabilities // Journal of Political Economy, 1973. — № 81 (3). — P. 638–659.
- *Cox J., Ross S., Rubinstein M.* Option Pricing: A Simplified Approach // Journal of Financial Economics, 1979. — № 7 (3). — P. 229–263.
- *Duffie D.* Stochastic Equilibria: Existence, Spanning Number and the No Expected Gains from Financial Trade Hypothesis // Econometrica, 1986. — № 54 (5). — P. 1161–1183.

- *Longstaff F., Schwartz E.* Valuing American Options by Simulation: A Simple Least Squares Approach // Review of Financial Studies, 2001. — № 14 (1). — P. 113–147.
- *Merton R.* Theory of Rational Option Pricing // Bell Journal of Economics and Management Science, 1973. — № 4. — P. 141–183.

В этой главе были упомянуты следующие книги.

- *Duffie D.* Security Markets —Stochastic Models. — San Diego: Academic Press, 1988.
- *Glasserman P.* Monte Carlo Methods in Financial Engineering. — N. Y.: Springer Verlag, 2004.
- *Hilpisch Y.* Derivatives Analytics with Python. — Wiley Finance, 2015.
- *Hilpisch Y.* Python for Finance. 2nd ed. — Sebastopol: O'Reilly, 2018.
- *Pliska S.* Introduction to Mathematical Finance. — Malden and Oxford: Blackwell Publishers, 1997.

ГЛАВА 7

Что дальше?

> Инвестиции в знания приносят наибольшую прибыль.
>
> *Бенджамин Франклин*

> Политика — для настоящего, уравнения — для вечности.
>
> *Альберт Эйнштейн*

Поздравляю! Вы добрались до последней главы, а значит, познакомились со многими важными идеями и концепциями теории финансов и программирования на языке Python. Это замечательно! Представленные здесь темы дают хорошую базу для понимания захватывающего и быстро меняющегося мира финансовой инженерии, однако многое еще предстоит узнать. В заключительной главе содержатся предложения по развитию и углублению знаний в одном или нескольких направлениях использования Python в финансовой деятельности.

Математика

В книге для решения финансовых задач применялись различные математические инструменты и методы, в основном из линейной алгебры, теории вероятностей и теории оптимизации. Они довольно широко распространены и для интеграции с Python не требуют продвинутых навыков. Тем не менее

современные финансы можно рассматривать как *прикладную математическую дисциплину*, а некоторые финансовые области, например ценообразование опционов или управление финансовыми рисками, в значительной степени опираются на высшую математику.

В следующем списке представлены несколько учебников, которые можно использовать для совершенствования математических навыков, применяемых в финансах.

- *Aleskerov F., Ersel H., Piontkovski D.* Linear Algebra for Economists. — Heidelberg: Springer Verlag, 2011.
- *Bhattacharya R., Waymire E.* A Basic Course in Probability Theory. — N. Y.: Springer Verlag, 2007.
- *Jacod J, Protter P.* Probability Essentials. — Berlin and Heidelberg: Springer Verlag, 2004.
- *Pemberton M., Rau N.* Mathematics for Economists — An Introductory Textbook. 4th ed. — Manchester and New York: Manchester University Press, 2016.
- *Protter P.* Stochastic Integration and Differential Equations. 2nd ed. — Berlin and Heidelberg: Springer Verlag, 2005.
- *Rudin W.* Real and Complex Analysis. 3rd ed. — London: McGraw-Hill, 1987.
- *Sundaram R.* A First Course in Optimization Theory. — Cambridge: Cambridge University Press, 1996.
- *Williams D.* Probability with Martingales. Reprint 2001. — Cambridge: Cambridge University Press, 1991.

Теория финансов

Финансы — обширная сфера деятельности с множеством различных специализаций. В книге были рассмотрены некоторые из наиболее важных и популярных финансовых моделей: теория портфеля среднего отклонения, модель ценообразования капитальных активов и модель ценообразования опционов Блэка — Шоулза — Мертона. В целом, книга охватывает простые и более реалистичные статичные модельные экономики, которые ограничиваются двумя моментами времени, а также динамичные модельные экономики

с постепенным исчезновением неопределенности. Однако здесь не были затронуты целые области математических финансов, например непрерывные модели ценообразования опционов, которые требуют более продвинутых математических инструментов. В книге также была опущена такая важная тема, как гипотеза эффективного рынка (efficient market hypothesis, EMH).

В следующем списке представлены несколько основных книг по финансам, которые дают более широкое представление о финансовой теории.

- *Copeland T, Weston F., Shastri K.* Financial Theory and Corporate Policy. 4th ed. — Boston: Addison Wesley, 2005.
- *Eichberger J., Harper I.* Financial Economics. — N. Y.: Oxford University Press, 1997.
- *Markowitz H.* Portfolio Selection — Efficient Diversification of Investments. — N. Y.: John Wiley & Sons, 1959.
- *Milne F.* Finance Theory and Asset Pricing. — N. Y.: Oxford University Press, 1995.
- *Pliska S.* Introduction to Mathematical Finance. — Malden and Oxford: Blackwell Publishers, 1997.
- *Rubinstein M.* A History of the Theory of Investments. — Hoboken: Wiley Finance, 2006.
- *Varian H.* Microeconomic Analysis. 3rd ed. — New York and London: W.W. Norton & Company, 1992.

Для тех, кто хочет глубже изучить математическое моделирование в финансах, далее приведен список учебников по финансовой математике.

- *Baxter M, Rennie A.* Financial Calculus — An Introduction to Derivative Pricing. — Cambridge: Cambridge University Press, 1996.
- *Bjork T.* Arbitrage Theory in Continuous Time. 2nd ed. — Oxford: Oxford University Press, 2004.
- *Delbaen F., Schachermayer W.* The Mathematics of Arbitrage. — Berlin: Springer Verlag, 2006.
- *Duffie D.* Security Markets —Stochastic Models. — San Diego: Academic Press, 1988.
- *Duffie D.* Dynamic Asset Pricing Theory. 3rd ed. — Princeton: Princeton University Press, 2001.

- *Elliot R., Kopp E.* Mathematics of Financial Markets. 2nd ed. — N. Y.: Springer Verlag, 2005.
- *Glasserman P.* Monte Carlo Methods in Financial Engineering. — N. Y.: Springer Verlag, 2004.

Несомненно, будет полезным и познавательным познакомиться со статьями, в которых впервые были изложены финансовые модели и теории. На удивление, многие из них находятся в открытом доступе. Далее приведена подборка таких статей, сделанная на основе рассмотренных в данной книге тем и методов.

- *Black F., Scholes M.* The Pricing of Options and Corporate Liabilities // Journal of Political Economy, 1973. — № 81 (3). — P. 638–659.
- *Boyle P.* Options: A Monte Carlo Approach // Journal of Financial Economics, 1977. — № 4 (4). — P. 322–338.
- *Cox J., Ingersoll J., Ross S.* A Theory of the Term Structure of Interest Rates // Econometrica, 1985. — № 53 (2). — P. 385–407.
- *Cox J., Ross S.* The Valuation of Options for Alternative Stochastic Processes // Journal of Financial Economics, 1976. — № 3. — P. 145–166.
- *Cox J., Ross S., Rubinstein M.* Option Pricing: A Simplified Approach // Journal of Financial Economics, 1979. — № 7 (3). — P. 229–263.
- *Duffie D.* Stochastic Equilibria: Existence, Spanning Number and the No Expected Gains from Financial Trade Hypothesis // Econometrica, 1986. — № 54 (5). — P. 1161–1183.
- *Harrison M., Kreps D.* Martingales and Arbitrage in Multiperiod Securities Markets // Journal of Economic Theory, 1979. — № 20. — P. 381–408.
- *Harrison M., Pliska S.* Martingales and Stochastic Integrals in the Theory of Continuous Trading // Stochastic Processes and their Applications, 1981. — № 11. — P. 215–260.
- *Heston S.* A Closed-Form Solution for Options with Stochastic Volatility with Applications to Bond and Currency Options // The Review of Financial Studies, 1993. — № 6 (2). — P. 327–343.
- *Longstaff F., Schwartz E.* Valuing American Options by Simulation: A Simple Least Squares Approach // Review of Financial Studies, 2001. — № 14 (1). — P. 113–147.
- *Markowitz H.* Portfolio Selection // Journal of Finance, 1952. — № 7 (1). — P. 77–91.

- *Merton R.* Option Pricing when the Underlying Stock Returns are Discontinuous // Journal of Financial Economics, 1976. — № 3 (3). — P. 125–144.
- *Perold A.* The Capital Asset Pricing Model // Journal of Economic Perspectives, 2004. — № 18 (3). — P. 3–24.
- *Protter P.* A Partial Introduction to Financial Asset Pricing Theory // Stochastic Processes and their Applications, 2001. — № 91. — P. 169–203.
- *Sharpe W.* Capital Asset Prices: A Theory of Market Equilibrium under Conditions of Risk // The Journal of Finance, 1964. — № 19 (3). — P. 425–442.

Тем, кто ищет единый, всеобъемлющий справочник, стоит присмотреться к серии книг по *анализу рыночных рисков*.

- *Alexander C.* Market Risk Analysis I — Quantitative Methods in Finance. — Chicester: John Wiley & Sons, 2008.
- *Alexander C.* Market Risk Analysis II — Practical Financial Econometrics. — Chicester: John Wiley & Sons, 2008.
- *Alexander C.* Market Risk Analysis III — Pricing, Hedging and Trading Financial Instruments. — Chicester: John Wiley & Sons, 2008.
- *Alexander C.* Market Risk Analysis IV — Value-at-Risk Models. — Chicester: John Wiley & Sons, 2008.

Программирование на языке Python

В настоящее время существует большое количество ресурсов для изучения программирования на языке Python. Перечисленные далее книги мне были полезны, однако, если бы нужно было выбрать из этого списка только одну, советую остановиться на книге Рамальо (2021), которая глубоко погружает в сам язык Python.

- *Harrison M.* Illustrated Guide to Python 3: A Complete Walkthrough of Beginning Python with Unique Illustrations Showing how Python Really Works, 2017. http://hairysun.com.
- *McKinney W.* Python for Data Analysis. 2nd ed. — O'Reilly, 2017.
- *Ramalho L.* Fluent Python. 2nd ed. — O'Reilly, 2021.
- *Ravenscroft A., Holden S., Martelli A.* Python in a Nutshell. 3rd ed. — O'Reilly, 2017.
- *VanderPlas J.* Python Data Science Handbook. — O'Reilly, 2016.

Python для финансовых расчетов

«Python для финансистов» — моя шестая книга об использовании языка Python в финансовой сфере. Вы можете удивиться: «Почему книга с базовой, вводной информацией выходит после пяти других, более сложных учебников?» На этот вопрос нет короткого и простого ответа, но можно сказать, что одной из причин ее создания стали просьбы читателей других книг и участников нашей программы онлайн-обучения.

Многие искали плавное введение как в финансы, *так и* в программирование на языке Python[1], которое дополняло бы другие мои книги. Поэтому здесь обе эти темы рассматриваются с нуля и тем самым дается необходимый старт для того, чтобы лучше понять «Python для финансовых расчетов», где ожидается, что читатель имеет некоторый опыт в финансах и в программировании.

Вот остальные мои книги.

- *Hilpisch Y.* Artificial Intelligence in Finance: A Python-Based Guide. — O'Reilly, 2020.
- *Hilpisch Y.* Python for Algorithmic Trading: From Idea to Cloud Deployment. — O'Reilly, 2020.
- *Hilpisch Y.* Python for Finance: Mastering Data-Driven Finance. 2nd ed. — O'Reilly, 2018.
- *Hilpisch Y.* Listed Volatility and Variance Derivatives: A Python-Based Guide. — Wiley Finance, 2017.
- *Hilpisch Y.* Derivatives Analytics with Python: Data Analysis, Models, Simulation, Calibration and Hedging. — Wiley Finance, 2015.

Наука о финансовых данных

Наука о данных стала важной дисциплиной и направлением практически в каждой отрасли человеческой деятельности. То же самое можно сказать и о финансовой сфере. Постоянно растущие объемы данных делают необходимым применение более сложных подходов к логистике и управлению данными. Таблиц Excel, конечно, уже недостаточно. Моя книга «Python для финансовых расчетов» в первую очередь посвящена использованию Python в науке о финансовых

[1] Мне нравится думать об этой книге, как о «Хоббите» Дж. Р. Р. Толкиена по отношению к трилогии «Властелин колец».

данных. Во второй и третьей частях книги рассматриваются типы и структуры данных, численные вычисления с помощью библиотеки NumPy, анализ данных посредством pandas, объектно-ориентированное программирование, визуализация данных, финансовые временные ряды, операции ввода-вывода, производительность Python, математические инструменты, стохастика и статистика, включая основы машинного обучения. Именно эта книга является следующим шагом в развитии навыков работы с Python в финансовой сфере.

Алгоритмическая торговля

Систематическая или алгоритмическая торговля стала обычным делом не только для хедж-фондов, но и для многих розничных трейдеров. Доступность относительно недорогих, но мощных API привела к распространению алгоритмических торговых стратегий практически во всех классах активов. В то время как крупные финансовые институты в целом имеют специально выделенные команды для каждого этапа торгового процесса — от анализа данных, исследований и бэктестинга до развертывания, мониторинга и управления рисками, — розничные трейдеры обычно занимаются всем этим самостоятельно.

То, что еще несколько лет назад казалось практически невыполнимой задачей для одного человека, сегодня можно относительно легко осуществить благодаря мощной экосистеме Python. В основном у розничных трейдеров, владеющих навыками программирования на данном языке, создание алгоритмической торговой операции может занять всего нескольких недель или даже дней. Именно *использование Python в алгоритмической торговле* стало главной темой моей одноименной книги *(Python for Algorithmic Trading)*. Она ведет читателя от управления данными и генерации идей до бэктестинга торговых стратегий и их автоматизированного развертывания в облаке.

О ключевых навыках работы с языком Python в алгоритмической торговле также рассказывается в части IV книги «Python для финансовых расчетов». Там данная тема освещается не столь подробно, как в *Python for Algorithmic Trading*, однако на основе представленной информации читатели смогут с помощью Python сгенерировать и развернуть торговую стратегию с автоматическим размещением сделок.

Чтобы лучше понять часть IV «Python для финансовых расчетов» и книгу *Python for Algorithmic Trading*, будет полезно, но не обязательно прочитать «Python для финансистов» и первые три части «Python для финансовых расчетов».

Финансовая инженерия

В количественных и вычислительных финансах долго доминировали компилируемые языки программирования C или C++. Это объясняется важностью скорости выполнения зачастую сложных численных расчетов и моделирования, в частности, когда масштабируемость требуется крупным финансовым учреждениям. Чистый Python может быть слишком медленным для реализации, к примеру, алгоритмов моделирования, требующих больших вычислительных ресурсов. Значительно ускорить его работу в данном направлении призваны библиотеки NumPy и pandas, которые добавляют высокоуровневые программные API с функциональностью кода C. В результате мы получаем Python со скоростью работы в 10–30 раз больше по сравнению с его чистой версией, что делает его достойной альтернативой для выполнения задач финансовой инженерии.

Моя книга *Derivatives Analytics with Python* («Использование Python в аналитике деривативов») знакомит с основными математическими и финансовыми концепциями, необходимыми для рыночного ценообразования и хеджирования деривативов на базе зависящих от рынка моделей ценообразования. В ней представлена законченная кодовая база Python, которая с нуля реализует все алгоритмы и методики с активным использованием возможностей библиотеки NumPy. Качественному усвоению этой информации и углублению навыков в области финансовой математики и финансовой инженерии могут поспособствовать также «Python для финансистов» и «Python для финансовых расчетов» (части I–III).

Темой части V книги «Python для финансовых расчетов» является упрощенная версия разработанной мной библиотеки DX Analytics, которую можно использовать для работы с ценообразованием деривативов. В ней показано, как концепции, подходы и численные методы, изложенные в книге *Listed Volatility and Variance Derivatives* («Деривативы на бирже в условиях волатильности и дисперсии»), могут быть использованы для создания гибкой и мощной аналитической библиотеки, основанной на моделировании по методу Монте-Карло. Сама по себе библиотека DX Analytics с открытым исходным кодом — это хороший источник моделей и еще больших возможностей в анализе деривативов.

В последнее время волатильность как класс активов приобрела большое значение. Будь то управление риском или получение дополнительного

коэффициента «альфа», *торговля деривативами на бирже в условиях волатильности и дисперсии* систематически проводится по всему миру. Познакомиться с основными концепциями торговли и ценообразования этих финансовых инструментов и с законченной кодовой базой на языке Python, которая в простой форме, помимо всего прочего, представляет безмодельную репликацию дисперсии или расчет индексов волатильности, можно в книге *Listed Volatility and Variance Derivatives.*

Искусственный интеллект

Можно с уверенностью говорить, что искусственный интеллект (ИИ) в будущем станет играть доминирующую роль в финансовой сфере, как уже происходит во многих других отраслях человеческой деятельности. Практически каждая финансовая организация инициировала проекты по изучению потенциала ИИ для улучшения работы с финансовыми операциями, экономии расходов, получения коэффициента «альфа» и т. д. Алгоритмы машинного обучения, глубокого обучения и обучения с подкреплением в основном тестируются и применяются во всех финансовых областях. Также все чаще публикуются научно-исследовательские работы по темам, находящимся на стыке ИИ и финансов.

Применению искусственного интеллекта в финансах посвящена еще одна моя книга, *Artificial Intelligence in Finance* («Искусственный интеллект в финансах»). Ее часть I дает справочную и историческую информацию об искусственном интеллекте, а также рассказывает истории его успешного применения. В части II рассматриваются традиционная финансовая теория и последние достижения в этой области: управление данными и использование искусственного интеллекта, а также машинное обучение как процесс. Часть III посвящена основным моделям и алгоритмам глубокого обучения: плотным и рекуррентным нейронным сетям и обучению с подкреплением (Q-обучению). В следующей части книги показано, как можно использовать статистическую неэффективность на финансовых рынках с помощью алгоритмической торговли, то есть с помощью бота, интерактивно обучающегося торговать на основе алгоритма Q-обучения. Последняя часть раскрывает последствия внедрения ИИ для конкурентной среды в финансовой отрасли. В ней также разбирается возможность финансовой сингулярности: появления в будущем искусственного финансового интеллекта (ИФИ), который,

например, сможет генерировать почти совершенные прогнозы относительно будущих рыночных цен.

Artificial Intelligence in Finance можно считать дополнением к *Python for Algorithmic Trading*, поскольку в ней подробно рассматриваются вопросы разработки, бэктестинга и управления рисками алгоритмических торговых стратегий на базе ИИ. Прежде чем погрузиться в увлекательный мир ИИ в финансах, не обязательно читать книги по ведению алгоритмической торговли посредством Python или использованию Python в финансовых расчетах, однако глубокое понимание этих тем будет весьма полезным.

Другие ресурсы

Вы могли заметить, что в этом разделе упоминались только мои книги, посвященные применению Python в финансовой деятельности. Его целью было познакомить читателя с прочими моими работами. Конечно, существует множество других книг по данной теме, которые полны полезной теоретической и практической информации. Тем не менее, скорее всего, тем людям, которым понравится эта книга, понравятся и другие мои работы, поскольку они схожи по стилю и подходу к изложению.

В то время как некоторые читатели наиболее эффективно обучаются, используя только книги и включенный в них код, другим нравится более интерактивное, управляемое обучение. Моя компания The Python Quants GmbH уже много лет предлагает комплексные программы онлайн-обучения, в которых вся информация представлена в систематизированном и структурированном виде. На момент написания этой книги доступны три программы онлайн-обучения, которые можно объединить в одну и тем самым получить полный разбор всех основных тем[1]:

- «Использование Python в алгоритмической торговле»[2];
- «Использование Python в финансовой инженерии»[3];
- «Использование Python в управлении активами»[4].

---

[1] https://oreil.ly/xc4qe.

[2] https://oreil.ly/r4l7y.

[3] https://oreil.ly/0h4Ej.

[4] https://oreil.ly/ubbLm.

Послесловие

Еще раз примите мои поздравления! С помощью «Python для финансистов» вы сделали первые шаги в увлекательный мир финансов с языком Python. В этой главе было дано множество ресурсов для изучения. Регулярная, усердная и систематическая практика быстро сделает вас экспертом в применении языка Python в финансовых расчетах. Такое достижение не только принесет личное удовлетворение, но и гарантирует успешное будущее, поскольку умение работать с Python, несомненно, стало ключевым навыком в финансовой индустрии. Да пребудет с вами сила Python!

Об авторе

Ив Хилпиш — основатель и руководитель проекта The Python Quants, ориентированного на применение технологий с открытым исходным кодом в финансовом анализе, искусственном интеллекте, алгоритмической торговле, финансовой инженерии и управлении активами. Он также возглавляет компанию The AI Machine, которая занимается разработкой систем алгоритмической торговли на базе искусственного интеллекта.

Ив является внештатным преподавателем финансовой инженерии и читает лекции по алгоритмической торговле в рамках программы CQF. Он также руководит первыми в своем роде программами онлайн-обучения по таким предметам, как «Использование Python в алгоритмической торговле», «Использование Python в финансовой инженерии» и «Использование Python в управлении активами», с получением сертификата университетского образца.

Помимо этого, Ив Хилпиш разработал библиотеку DX Analytics для финансового анализа. Он регулярно проводит в крупных городах мира конференции и семинары, посвященные использованию Python в финансовой инженерии и алгоритмической торговле, а также выступал с программной речью на технологических конференциях в США, Европе и Азии.

Иллюстрация на обложке

На обложке изображена диадемовая фурина (лат. *Furina ornata*), более известная как оранжевоголовая змея. Эта небольшая ядовитая змея обитает на севере и северо-западе Австралии. Ее можно распознать по рыжему пятну на задней части головы.

Изображение змеи на обложке было выполнено на основе черно-белой гравюры из работы Ричарда Лиидеккера *The royal natural history*.

Ив Хилпиш

Python для финансистов

Перевел с английского С. Черников

Руководитель дивизиона	*Ю. Сергиенко*
Руководитель проекта	*А. Киреева*
Ведущий редактор	*Н. Гринчик*
Литературные редакторы	*В. Беляева, Н. Рощина*
Художественный редактор	*В. Мостипан*
Корректор	*Е. Павлович*
Верстка	*Г. Блинов*

Изготовлено в России. Изготовитель: ООО «Прогресс книга».
Место нахождения и фактический адрес: 194044, Россия, г. Санкт-Петербург,
Б. Сампсониевский пр., д. 29А, пом. 52. Тел.: +78127037373.

Дата изготовления: 05.2023. Наименование: книжная продукция. Срок годности: не ограничен.

Налоговая льгота — общероссийский классификатор продукции ОК 034-2014, 58.11.12.000 — Книги печатные профессиональные, технические и научные.

Импортер в Беларусь: ООО «ПИТЕР М», 220020, РБ, г. Минск, ул. Тимирязева, д. 121/3, к. 214,
тел./факс: 208 80 01.

Подписано в печать 15.03.23. Формат 70×100/16. Бумага офсетная. Усл. п. л. 16,770. Тираж 700. Заказ 0000.

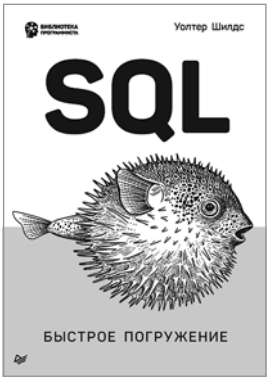
Борис Пасхавер

PANDAS В ДЕЙСТВИИ

Язык Python помогает упростить анализ данных. Если вы научились пользоваться электронными таблицами, то сможете освоить и pandas! Несмотря на сходство с табличной компоновкой Excel, pandas обладает большей гибкостью и более широкими возможностями. Эта библиотека для Python быстро выполняет операции с миллионами строк и способна взаимодействовать с другими инструментами. Она дает идеальную возможность выйти на новый уровень анализа данных.

Уолтер Шилдс

SQL: БЫСТРОЕ ПОГРУЖЕНИЕ



Что общего между самыми востребованными профессиями и стремительным увеличением количества информации в мире? Ответ: язык структурированных запросов (SQL). SQL — рабочая лошадка среди языков программирования, основа основ для современного анализа и управления данными.

Книга «SQL: быстрое погружение» идеальна для всех, кто ищет новые перспективы карьерного роста; для разработчиков, которые хотят расширить свои навыки и знания в программировании; для любого человека, даже без опыта, кто хочет воспользоваться возможностями будущего, в котором будут править данные.

Грег Хорин

УПРАВЛЕНИЕ ПРОЕКТАМИ С НУЛЯ

Управлять проектами не так сложно, как может показаться! Эта книга — кратчайший путь для освоения всех необходимых навыков: от бюджетирования и планирования до секретов управления командой и работы над ошибками, лучшее на сегодняшний день руководство по современному проектному менеджменту для начинающих. Здесь вы найдете простые инструкции и чек-листы для успешного выполнения всех задач, которые могут возникнуть в ходе работы! Бонус — глава с полезными советами для подготовки к сдаче сертификационного экзамена PMP.