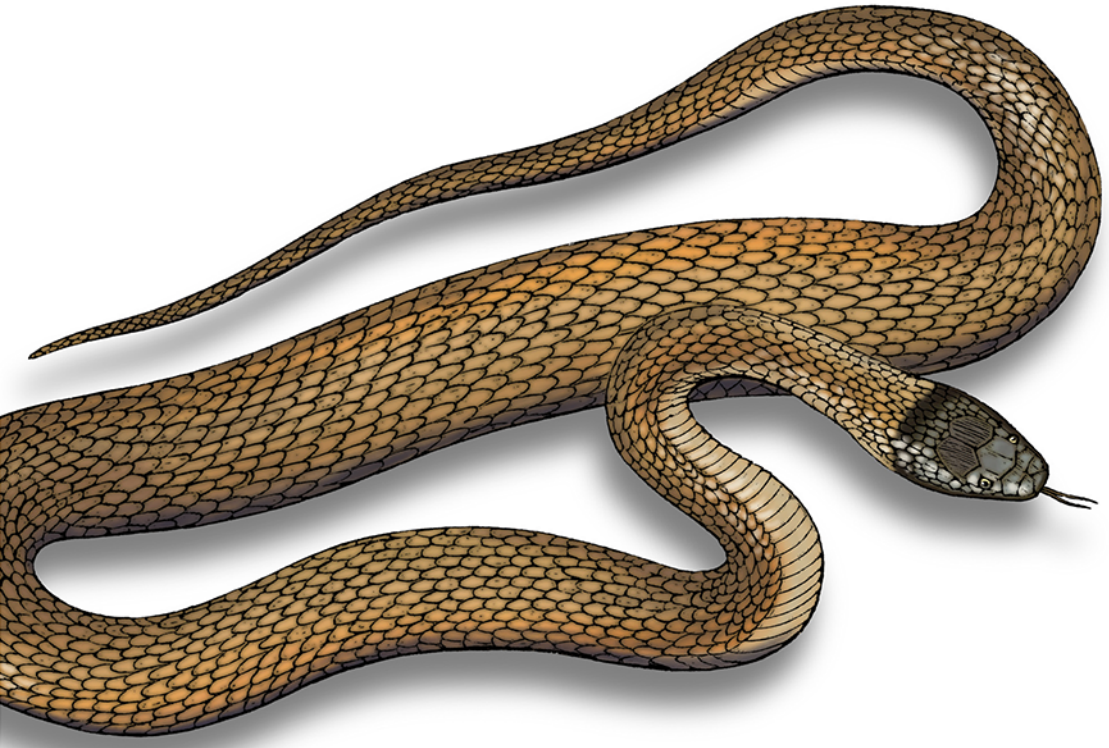
O'REILLY®

Finanse i Python

Łagodne wprowadzenie do teorii finansów



Helion

Yves Hilpisch

Tytuł oryginału: Financial Theory with Python: A Gentle Introduction

Tłumaczenie: Leszek Sagalara

ISBN: 978-83-283-8923-6

© 2022 Helion S.A.

Authorized Polish translation of the English edition of Financial Theory with Python
ISBN 9781098104351 © 2022 Yves Hilpisch.

This translation is published and sold by permission of O'Reilly Media, Inc.,
which owns or controls all rights to publish and sell the same.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from the Publisher.

Wszelkie prawa zastrzeżone. Nieautoryzowane rozpowszechnianie całości lub fragmentu niniejszej publikacji w jakiejkolwiek postaci jest zabronione. Wykonywanie kopii metodą kserograficzną, fotograficzną, a także kopiowanie książki na nośniku filmowym, magnetycznym lub innym powoduje naruszenie praw autorskich niniejszej publikacji.

Wszystkie znaki występujące w tekście są zastrzeżonymi znakami firmowymi bądź towarowymi ich właścicieli.

Autor oraz wydawca dołożyli wszelkich starań, by zawarte w tej książce informacje były kompletne i rzetelne. Nie biorą jednak żadnej odpowiedzialności ani za ich wykorzystanie, ani za związane z tym ewentualne naruszenie praw patentowych lub autorskich. Autor oraz wydawca nie ponoszą również żadnej odpowiedzialności za ewentualne szkody wynikłe z wykorzystania informacji zawartych w książce.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 231 22 19, 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/finpyt
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

Pliki z przykładami omawianymi w książce można znaleźć pod adresem:
https://ftp.helion.pl/przyklady/finpyt.zip

Printed in Poland.

• Kup książkę
• Poleć książkę
• Oceń książkę

• Księgarnia internetowa
• Lubię to! » Nasza społeczność

Spis treści

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

Kup książkę Poleć książkę

ROZDZIAŁ 2.
Gospodarka dwustanowa

> Finanse, jako dziedzina empiryczna, dążą do uzyskania konkretnych odpowiedzi, takich jak ustalenie właściwej wartości danego papieru wartościowego czy optymalnej liczby posiadanych akcji.
>
> — Darrell Duffie (1988)

> Pojęcie arbitrażu ma kluczowe znaczenie dla współczesnej teorii finansów.
>
> — Delbaen i Schachermayer (2006)

Analiza przedstawiona w tym rozdziale jest oparta na najprostszym **modelu gospodarki**, który jest jednak wystarczająco bogaty, aby wprowadzić wiele ważnych pojęć i koncepcji finansów: gospodarce z dwoma istotnymi punktami w czasie i tylko dwoma niepewnymi stanami przyszłymi. Pozwala nam to również na przedstawienie niektórych ważnych wyników w tej dziedzinie, takich jak **fundamentalne twierdzenia wyceny aktywów**, które są omawiane w tym rozdziale[1].

Wybrany prosty model jest środkiem ułatwiającym formalne wprowadzenie czasami dość abstrakcyjnych pojęć matematycznych i idei finansowych przez unikanie jak największej liczby szczegółów technicznych. Kiedy te idee zostaną rozwinięte i dobrze zrozumiane, przejście do bardziej realistycznych modeli finansowych zwykle okazuje się bezproblemowe.

Ten rozdział porusza głównie następujące kluczowe tematy z dziedziny finansów, matematyki i programowania w Pythonie:

Finanse	Matematyka	Python
Czas	Liczby naturalne $\mathbb{N}$	`int`, `type`
Pieniądze (waluta)	Liczby rzeczywiste $\mathbb{R}$	`float`
Przepływy pieniężne	Krotka	`tuple`, `list`
Zysk, odsetki	Liczby rzeczywiste $\mathbb{R}$	`abs`
Wartość bieżąca (netto)	Funkcja	`def`, `return`
Niepewność	Przestrzeń wektorowa $\mathbb{R}^2$	NumPy, `ndarray`, `np.array`

[1] Szczegółowe informacje na temat fundamentalnych twierdzeń wyceny aktywów można znaleźć w przełomowych pracach Harrisona i Krepsa (1979) oraz Harrisona i Pliski (1981).

Kup książkę Poleć książkę

Finanse	Matematyka	Python
Aktywa finansowe	Proces	`ndarray`, `tuple`
Ryzyko	Prawdopodobieństwo, przestrzeń stanów, zbiór potęgowy, odwzorowanie	`ndarray`
Oczekiwanie, oczekiwany zysk	Iloczyn skalarny	`np.dot`
Zmienność	Wariancja, odchylenie standardowe	`np.sqrt`
Roszczenia warunkowe	Zmienna losowa	`np.arange`, `np.maximum`, `plt.plot`
Replikacja, arbitraż	Równania liniowe, postać macierzy	`ndarray(2d)`, `np.linalg.solve`, `np.dot`
Zupełność, papiery wartościowe Arrowa-Debreu	Niezależność liniowa, rozpiętość	`np.linalg.solve`
Wycena martyngałowa	Martyngał, miara martyngałowa	`np.dot`
Średnia-wariancja	Oczekiwanie, wariancja, odchylenie standardowe	`np.linspace`, `.std()`, `[x for y in z]`

Gospodarka

Pierwszym elementem modelu finansowego jest idea **gospodarki**. Gospodarka jest abstrakcyjnym pojęciem, które obejmuje inne elementy modelu finansowego, takie jak aktywa (rzeczowe, finansowe), agenci (ludzie, instytucje) czy pieniądze. Podobnie jak w świecie rzeczywistym, gospodarki nie można zobaczyć ani dotknąć. Nie można jej również formalnie bezpośrednio modelować — możliwość użycia skróconego terminu upraszcza komunikację. Te poszczególne elementy modelu razem tworzą gospodarkę[2].

Aktywa rzeczowe

W gospodarce dostępnych jest wiele **aktywów rzeczowych**, które mogą być wykorzystane do różnych celów. Składnikiem aktywów rzeczowych może być kurze jajo lub skomplikowana maszyna do produkcji innych aktywów rzeczowych. W tym momencie nie jest istotne, kto np. wytwarza aktywa rzeczowe lub kto jest ich właścicielem.

Agenci

Agenci mogą być postrzegani jako indywidualne istoty ludzkie aktywne w gospodarce. Mogą oni być zaangażowani w produkcję aktywów rzeczowych, ich konsumpcję lub handel nimi. Przyjmują i wydają pieniądze przy dokonywaniu transakcji. Agentem może być również instytucja taka jak bank, pozwalająca innym agentom deponować pieniądze, od których następnie płaci odsetki.

[2] Bardziej formalne ujęcie pojęcia gospodarki jest przedstawione w rozdziale 5.

Kup książkę Poleć książkę

Czas

Działalność gospodarcza, podobnie jak obrót aktywami rzeczowymi, ma miejsce tylko w konkretnych punktach w czasie. Formalnie rzecz ujmując, odbywa się to dla punktu w czasie $t \in 0$, 1, 2, 3, … lub $t \in \mathbb{N}_0$. W dalszej części rozważań istotne będą tylko dwa punkty w czasie: $t = 0$ i $t = 1$. Najlepiej interpretować je jako *dziś* i *za rok od dziś*, choć niekoniecznie jest to jedyna interpretacja istotnego przedziału czasowego. W wielu kontekstach można również myśleć o dniu *dzisiejszym* i *jutrzejszym*. W każdym razie w teorii finansów, jeśli istotne są tylko dwa punkty w czasie, mówimy o **gospodarce statycznej**.

Typ danych Pythona reprezentujący liczby naturalne $\mathbb{N}$ to `int` (od ang. *integers*, czyli liczby całkowite[3]). Na liczbach całkowitych można wykonywać typowe operacje arytmetyczne, takie jak dodawanie, odejmowanie, mnożenie:

```
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 Dodawanie dwóch wartości całkowitych.

❷ Mnożenie dwóch wartości całkowitych.

❸ Przypisanie wartości 0 do zmiennej o nazwie `t`.

❹ Wypisanie wartości zmiennej `t`.

❺ Przypisanie zmiennej `t` nowej wartości 1.

❻ Wyszukanie i wypisanie typu danych Pythona dla zmiennej `t`.

Pieniądze

W gospodarce **pieniądz** (lub **waluta**) jest dostępny w nieograniczonym zakresie. Jest też nieskończenie podzielny. O pieniądzu i walucie należy myśleć wyłącznie w kategoriach abstrakcyjnych, a nie w kategoriach gotówki (fizycznych monet lub banknotów).

Pieniądz zasadniczo służy w gospodarce jako *numeraire*, w tym sensie, że wartość jednej jednostki pieniądza (USD, EUR, GBP itd.) jest znormalizowana do dokładnie 1. W takich

[3] Szczegółowe informacje na temat standardowych typów danych w Pythonie można znaleźć w dokumentacji *Built-in Types* (*https://oreil.ly/YTWep*).

Kup książkę Poleć książkę

jednostkach wyrażane są ceny wszystkich innych dóbr, które są ułamkami lub wielokrotnościami takich jednostek. Formalnie jednostki waluty są reprezentowane jako (nieujemne) liczby rzeczywiste $c \in \mathbb{R}_{\geq 0}$.

W Pythonie standardowym typem danych używanym do reprezentowania liczb rzeczywistych $\mathbb{R}$ jest `float` (od ang. *floating point numbers*, co oznacza liczby zmiennoprzecinkowe). Podobnie jak typ `int`, dopuszcza on m.in. wykonywanie typowych operacji arytmetycznych, takich jak dodawanie i odejmowanie:

```
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 Dodawanie dwóch liczb.

❷ Odejmowanie dwóch liczb.

❸ Przypisanie wyniku dodawania do zmiennej `c`.

❹ Wypisanie wartości zmiennej `c`.

❺ Wyszukanie i wypisanie typu Pythona dla zmiennej `c`.

Pieniądz oprócz tego, że pełni funkcję numeraire, pozwala agentom kupować i sprzedawać aktywa rzeczowe oraz przechowywać wartość w czasie. Te dwie funkcje opierają się na zaufaniu, że pieniądz rzeczywiście ma wewnętrzną wartość zarówno dziś, jak i za rok. Ogólnie rzecz biorąc, przekłada się to na ufność w to, że osoby i instytucje są skłonne akceptować pieniądze zarówno dziś, jak i w przyszłości w ramach dowolnej transakcji. Funkcja numeraire jest niezależna od tego zaufania, ponieważ jest to jedynie operacja liczbowa.

Przepływy pieniężne

Połączenie czasu z walutą prowadzi do pojęcia **przepływu pieniężnego**. Rozważmy projekt inwestycyjny, który wymaga dziś zainwestowania, powiedzmy, 9,5 jednostek walutowych, a po roku zwraca 11,75 jednostek walutowych. Inwestycja jest zwykle uważana za *wypływ* środków pieniężnych i często przedstawia się ją jako ujemną liczbę rzeczywistą, $c \in \mathbb{R}_{<0}$, lub konkretnie $c = -9{,}5$. Zwrot z inwestycji to *wpływ* gotówki i tym samym dodatnia liczba rzeczywista, $c \in \mathbb{R}_{\geq 0}$, czyli w tym przykładzie $c = +11{,}75$.

Aby wskazać punkty w czasie, w których występują przepływy pieniężne, stosuje się indeks czasowy: w tym przykładzie $c_{t=0} = -9{,}5$ i $c_{t=1} = 11{,}75$ lub w skrócie $c_0 = -9{,}5$ i $c_1 = 11{,}75$.

Kup książkę Poleć książkę

Dwa przepływy pieniężne (teraz i za rok) są przedstawiane matematycznie jako **para uporządkowana** (krotka 2-elementowa), która łączy dwa odpowiednie przepływy pieniężne w jeden obiekt: $c \in \mathbb{R}^2$ z $c = (c_0, c_1)$ i $c_0, c_1 \in \mathbb{R}$.

W Pythonie dostępnych jest wiele struktur danych pozwalających na modelowanie takiego obiektu matematycznego. Dwie najbardziej podstawowe to `tuple` (krotka) i `list` (lista). Obiekty typu `tuple` są niemutowalne, co oznacza, że nie można ich zmieniać po utworzeniu, natomiast obiekty typu `list` są mutowalne i mogą być zmieniane po ich utworzeniu. Oto przykład obiektów typu `tuple` (oznaczanych nawiasami):

```
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 Zdefiniowanie dzisiejszego wypływu środków pieniężnych.

❷ Zdefiniowanie wpływu środków pieniężnych rok później.

❸ Zdefiniowanie obiektu `c` typu `tuple` (zwróć uwagę na użycie nawiasów).

❹ Wypisanie pary przepływów pieniężnych (zwróć uwagę na nawiasy).

❺ Wyszukanie i wyświetlenie typu obiektu `c`.

❻ Uzyskanie dostępu do pierwszego elementu obiektu `c`.

❼ Uzyskanie dostępu do drugiego elementu obiektu `c`.

A teraz przykład obiektu `list` (w nawiasach kwadratowych):

```
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
```


Kup książkę Poleć książkę

```
In [24]: c[0] = 10 ❻

In [25]: c ❼
Out[25]: [10, 11.75]
```

❶ Zdefiniowanie obiektu `c` typu `list` (zwróć uwagę na użycie nawiasów kwadratowych).

❷ Wypisanie pary przepływów pieniężnych (zwróć uwagę na nawiasy kwadratowe).

❸ Wyszukanie i wyświetlenie typu obiektu `c`.

❹ Uzyskanie dostępu do pierwszego elementu obiektu `c`.

❺ Uzyskanie dostępu do drugiego elementu obiektu `c`.

❻ Nadpisanie wartości na pierwszej pozycji indeksu w obiekcie `c`.

❼ Wyświetlenie powstałych zmian.

Zysk

Rozważmy projekt inwestycyjny o przepływach pieniężnych $c = (c_0, c_1) = (-10, 12)$. **Zysk** $R \in \mathbb{R}$ z projektu to suma przepływów pieniężnych $R = c_0 + c_1 = -10 + 12 = 2$. **Stopa zwrotu**, $r \in \mathbb{R}$, to zysk, R, podzielony przez $|c_0|$, czyli przez wartość bezwzględną dzisiejszych nakładów inwestycyjnych:

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

W Pythonie sprowadza się to do prostych operacji arytmetycznych:

```
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 Zdefiniowanie pary przepływów pieniężnych jako obiektu `tuple`.

❷ Obliczenie zysku `R` przez zsumowanie wszystkich elementów `c` i…

❸ …wypisanie wyniku.

❹ Obliczenie stopy zwrotu `r` za pomocą funkcji `abs(x)`, podającej wartość bezwzględną `x`, i…

❺ …wypisanie wyniku.

Odsetki

Istnieje różnica między przepływem pieniężnym dzisiaj a przepływem pieniężnym za rok. Różnica ta wynika z **odsetek**, które zarabia się na jednostkach walutowych lub które trzeba zapłacić, aby pożyczyć jednostki walutowe. Odsetki w tym kontekście to *cena*, jaką trzeba zapłacić za posiadanie kontroli nad pieniędzmi należącymi do innego agenta.

Kup książkę Poleć książkę

Agent posiadający jednostki walutowe, których nie potrzebuje w danej chwili, może je zdeponować w banku lub pożyczyć innemu agentowi, aby *zarobić na odsetkach*. Jeśli agent potrzebuje więcej jednostek walutowych, niż ma obecnie do dyspozycji, może je pożyczyć z banku lub od innych agentów, ale będzie musiał *zapłacić odsetki*.

Załóżmy, że agent wpłaca dziś do banku kwotę $c_0 = -10$ jednostek walutowych. Zgodnie z umową depozytową po roku otrzyma od banku $c_1 = 11$ jednostek walutowych. Odsetki, $I \in \mathbb{R}$, płacone od depozytu wynoszą $I = c_0 + c_1 = -10 + 11 = 1$. Stopa procentowa, $i \in \mathbb{R}$, wynosi, odpowiednio, $i = \frac{I}{|c_0|} = 0{,}1$.

W dalszej części zakłada się, że odnośna stopa procentowa jest taka sama zarówno w przypadku udzielania, jak i zaciągania kredytów oraz że jest ona stała dla całej gospodarki.

Wartość bieżąca

Dostępność opcji kredytowych lub depozytowych prowadzi do powstania **kosztów alternatywnych** związanych z wykorzystaniem środków pieniężnych w projekcie inwestycyjnym. Przepływ pieniężny o wartości, powiedzmy, $c_1 = 12{,}1$ w ciągu jednego roku nie może być bezpośrednio porównywany pod względem wartości z przepływem pieniężnym o wartości $c_0 = 12{,}1$ dzisiaj, ponieważ na jednostkach walutowych niewykorzystanych w projekcie można zarobić odsetki.

Aby odpowiednio porównać przepływy pieniężne występujące za rok z tymi występującymi dzisiaj, należy obliczyć **wartość bieżącą**. Dokonuje się tego przez **dyskontowanie** przy użyciu stałej stopy procentowej w gospodarce. Dyskontowanie może być modelowane jako funkcja $D{:}\mathbb{R} \to \mathbb{R}$, $c_1 \longmapsto D(c_1)$, która odwzorowuje liczbę rzeczywistą (przepływ pieniężny za rok) na inną liczbę rzeczywistą (dzisiejszy przepływ pieniężny). Utrzymuje się ona na poziomie

$$\begin{aligned} c_0 &= D(c_1) \\ &= \frac{c_1}{1+i} \\ &= \frac{12{,}1}{1+0{,}1} \\ &= 11 \end{aligned}$$

dla stopy procentowej $i = 0{,}1$. Zależność ta wynika z alternatywnego „inwestowania” w depozyty złożone w banku:

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

Funkcje Pythona są dobrze przystosowane do reprezentowania funkcji matematycznych, w tym dyskontowania:

```
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
```


Kup książkę Poleć książkę

```
In [34]: D(11) ❺
Out[34]: 10.0
```

❶ Ustalenie stopy procentowej `i`.

❷ Zdefiniowanie funkcji za pomocą instrukcji `def`; `D` jest nazwą funkcji; `c1` jest nazwą parametru.

❸ Zwraca wartość bieżącą za pomocą instrukcji `return`.

❹ Obliczenie wartości bieżącej 12,1; zauważ błąd zaokrąglenia spowodowany wewnętrznymi problemami z reprezentacją liczb zmiennoprzecinkowych.

❺ Obliczenie wartości bieżącej 11 (w tym przypadku „dokładnie”).

Wartość bieżąca netto

W jaki sposób agent powinien zdecydować, czy przeprowadzić projekt inwestycyjny, czy nie? Jednym z kryteriów jest **wartość bieżąca netto** (NPV, ang. *net present value*). Wartość bieżąca netto, $NPV \in \mathbb{R}$, jest sumą dzisiejszego wypływu pieniężnego i wartości bieżącej wpływu pieniężnego za rok:

$$NPV(c) = c_0 + D(c_1)$$

Kalkulacja wartości bieżącej netto jest tu funkcją $NPV{:}\mathbb{R}^2 \to \mathbb{R}$, odwzorowującą krotkę przepływu pieniężnego na liczbę rzeczywistą. Jeśli wartość bieżąca netto jest dodatnia, projekt powinien zostać zrealizowany; jeśli jest ujemna, to nie — ponieważ bardziej atrakcyjna jest opcja alternatywna, polegająca na zdeponowaniu pieniędzy w banku.

Rozważmy projekt inwestycyjny o przepływach pieniężnych $c^A = (-10{,}5, 12{,}1)$. Wartość bieżąca netto wynosi $NPV(c^A) = -10{,}5 + D(12{,}1) = -10{,}5 + 11 = 0{,}5$. Projekt powinien być realizowany. Rozważmy alternatywny projekt inwestycyjny o $c^B = (-10{,}5, 11)$. Ma on ujemną wartość bieżącą netto i nie powinien być realizowany: $NPV(c^B) = -10{,}5 + D(11) = -10{,}5 + 10 = -0{,}5$.

Opierając się na poprzednich definicjach, można łatwo zdefiniować odpowiednią funkcję Pythona:

```
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 Projekt o dodatniej wartości bieżącej netto.

❷ Projekt o ujemnej wartości bieżącej netto.

Kup książkę Poleć książkę

Niepewność

Wpływy pieniężne z projektu inwestycyjnego w perspektywie jednego roku są z reguły *niepewne*. Może na nie wpływać wiele czynników występujących w warunkach rzeczywistych (działania konkurencji, nowe technologie, wzrost gospodarczy, pogoda, problemy przy realizacji projektu itp.) W gospodarce modelowej pojęcie *stanu* gospodarki w ciągu jednego roku obejmuje wpływ wszystkich istotnych czynników.

Przyjmijmy, że w jednym roku gospodarka może znajdować się w jednym z dwóch różnych stanów, *u* i *d* (od ang. *up* i *down*), które mogą być interpretowane jako *wzrost* („dobrze") i *spadek* („źle"). Przepływy pieniężne c_1 danego projektu w okresie jednego roku stają się wtedy **wektorem**

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

z dwiema różnymi wartościami

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

reprezentującymi odpowiednie przepływy pieniężne dla każdego stanu gospodarki. Jest to formalnie reprezentowane jako tzw. **wektor kolumnowy**:

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

Z matematycznego punktu widzenia istnieją pewne operacje wykonywane na takich wektorach, np. **mnożenie przez skalar** i **dodawanie**:

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

Inną ważną operacją na wektorach jest tworzenie **kombinacji liniowych** wektorów. Rozważmy dwa różne wektory: $c_1, d_1 \in \mathbb{R}^2$. Kombinacja liniowa jest wtedy wyrażana przez

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

Tu i poprzednio zakłada się, że $\alpha, \beta \in \mathbb{R}$.

Najpopularniejszym sposobem modelowania wektorów (i macierzy) w Pythonie jest wykorzystanie pakietu NumPy (*http://numpy.org*), który jest pakietem zewnętrznym i musi być zainstalowany oddzielnie. Dla poniższego kodu rozważmy projekt inwestycyjny z $c_0 = -10$ i $c_1 = (20, 5)^T$, gdzie indeks górny T oznacza transpozycję wektora (przekształcenie **wektora wierszowego** lub **poziomego** w **wektor kolumnowy** lub **pionowy**). Podstawową klasą używaną do modelowania wektorów jest klasa `ndarray`, co oznacza **tablicę *n*-wymiarową** (od ang. *n-dimensional array*):

```
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
```


Kup książkę Poleć książkę

```
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 Import pakietu `numpy` jako `np`.

❷ Dzisiejszy wypływ pieniężny.

❸ Niepewny wpływ pieniężny za rok; jednowymiarowe obiekty `ndarray` nie dokonują rozróżnienia między wierszem (poziomo) a kolumną (pionowo).

❹ Wyszukanie i wypisanie typu `c1`.

❺ Wypisanie wektora przepływów pieniężnych.

❻ Zestawienie przepływów pieniężnych w obiekt typu `tuple`.

❼ Obiekt `tuple`, podobnie jak `list`, może zawierać inne złożone struktury danych.

❽ Liniowe przekształcenie wektora za pomocą mnożenia przez skalar i dodawania; formalnie rzecz biorąc, mówi się również o wektoryzowanej operacji numerycznej oraz o rozgłaszaniu.

❾ Kombinacja liniowa dwóch obiektów `ndarray` (wektorów).

Aktywa finansowe

Aktywa finansowe to instrumenty finansowe („kontrakty"), które mają stałą cenę dzisiaj i niepewną cenę za rok. Rozpatrzmy np. udział w kapitale firmy, która realizuje projekt inwestycyjny. Taka akcja może być dostępna dziś w cenie $S_0 \in \mathbb{R}_{>0}$. Cena akcji za rok zależy od powodzenia projektu inwestycyjnego, tzn. od tego, czy w stanie u wystąpi wysoki wpływ gotówki czy też niski w stanie d. Formalnie mówiąc, $S_1^u, S_1^d \in \mathbb{R}_{\geq 0}$, przy $S_1^u > S_1^d$.

Mówi się również o **procesie cenowym** aktywów finansowych $S{:}\mathbb{N}_0 \cdot \{u, d\}\ \mathbb{R}_{\geq 0}$, odwzorowującym czas i stan gospodarki na cenę aktywów finansowych. Zauważmy, że dzisiejsza cena jest niezależna od stanu $S_0^u = S_0^d \equiv S_0$, natomiast w przypadku ceny po roku na ogół jest inaczej. Piszemy również $(S_t)_{t \in \{0, 1\}} = (S_0, S_1)$ lub w skrócie $S = (S_0, S_1)$. Narzędziem do modelowania jest ponownie pakiet NumPy:

```
In [49]: S0 = 10 ❶

In [50]: S1 = np.array((12.5, 7.5)) ❷

In [51]: S = (S0, S1) ❸
```


Kup książkę Poleć książkę

```
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 Dzisiejsza cena aktywów finansowych.

❷ Niepewna cena za rok jako wektor (obiekt `ndarray`).

❸ Proces cenowy jako obiekt `tuple`.

❹ Wypisanie informacji o procesie cenowym.

❺ Dostęp do ceny dzisiejszej.

❻ Dostęp do ceny za rok w stanie *u* (pierwszym).

❼ Dostęp do ceny za rok w stanie *d* (drugim).

Ryzyko

Często przyjmuje się domyślnie, że te dwa stany gospodarki są *równie prawdopodobne*. Generalnie oznacza to, że gdy eksperyment w gospodarce jest powtarzany (nieskończenie wiele razy), to obserwuje się, że w połowie przypadków urzeczywistnia się stan *u*, a w drugiej połowie stan *d*.

Jest to podejście obiektywne, zgodnie z którym prawdopodobieństwo urzeczywistnienia się jakiegoś stanu oblicza się na podstawie częstości obserwowania tego stanu podzielonej przez całkowitą liczbę eksperymentów prowadzących do obserwacji. Jeśli stan *u* jest obserwowany 30 razy na 50 eksperymentów, to prawdopodobieństwo $p \in \mathbb{R}_{\geq 0}$ przy $0 \leq p \leq 1$ wynosi, odpowiednio, $p = 30/50 = 0{,}6$, czyli 60%.

W kontekście modelowania zakłada się, że prawdopodobieństwa wystąpienia wszystkich możliwych stanów są dane *a priori*. Czasami mówi się o prawdopodobieństwach *obiektywnych* lub *fizycznych*.

Miara probabilistyczna

Prawdopodobieństwa dla zdarzeń, które są fizycznie możliwe, tworzą razem **miarę probabilistyczną**. Taką miarą probabilistyczną jest funkcja P: $\wp(\{u, d\}) \to \mathbb{R}_{\geq 0}$, odwzorowująca wszystkie elementy **zbioru potęgowego** $\{u, d\}$ — przy czym $\wp(\{u, d\}) = \{\emptyset, \{u\}, \{d\}, \{u, d\}$ — na przedział jednostkowy. Zbiór potęgowy w tym przypadku odzwierciedla wszystkie zdarzenia, które są fizycznie możliwe.

W tym kontekście zbiór $\{u, d\}$ jest również nazywany **przestrzenią stanów** i oznaczany symbolem Ω. Trójka $(\Omega, \wp(\Omega), P)$ razem jest nazywana **przestrzenią prawdopodobieństwa**.

Kup książkę Poleć książkę

Funkcja P, reprezentująca miarę probabilistyczną, musi spełniać trzy warunki:

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

Pierwszy warunek oznacza, że przynajmniej jeden ze stanów musi się zmaterializować. Drugi oznacza, że prawdopodobieństwo urzeczywistnienia się danego stanu zawiera się w przedziale między 0 a 1. Trzeci mówi, że wszystkie prawdopodobieństwa sumują się do 1.

W prostym modelu gospodarki z tylko dwoma stanami wygodnie jest zdefiniować $p \equiv P(u)$ i mieć, odpowiednio, $P(d) = 1 - p$, uwzględniając trzeci z powyższych warunków. Ustalenie p definiuje wtedy miarę probabilistyczną P.

Mając dostępną w pełni określoną miarę probabilistyczną, gospodarkę modelową nazywa się zazwyczaj **gospodarką w warunkach ryzyka**. Gospodarka modelowa bez w pełni określonej miary probabilistycznej jest często nazywana **gospodarką w warunkach niejednoznaczności**.

W zastosowaniach miara probabilistyczna jest zwykle modelowana również jako, odpowiednio, wektor i obiekt `ndarray`. Jest to możliwe przynajmniej dla dyskretnej przestrzeni stanów, o skończonej liczbie elementów:

```
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])
```

Pojęcia związane z niepewnością

Niepewność w kontekście finansowym może przybierać różne formy. *Ryzyko* na ogół odnosi się do sytuacji, w której znany jest (w założeniu) pełny rozkład prawdopodobieństwa przyszłych stanów gospodarki. *Niejednoznaczność* odnosi się do sytuacji, w których taki rozkład nie jest znany. Tradycyjnie finanse opierają się prawie wyłącznie na modelach gospodarki w warunkach ryzyka, chociaż istnieje nurt badań, który zajmuje się problemami finansów w warunkach niejednoznaczności (zob. Guidolin i Rinaldi [2012] w celu zapoznania się z literaturą badawczą).

Oczekiwana wartość

Na podstawie miary probabilistycznej można obliczyć **oczekiwaną wartość** niepewnej wielkości, takiej jak cena składnika aktywów finansowych za rok. Oczekiwana wartość może być interpretowana jako **średnia ważona**, gdzie wagi są określone przez prawdopodobieństwa. Jest to średnia, ponieważ prawdopodobieństwa sumują się do jedynki.

Kup książkę Poleć książkę

Rozważmy składnik aktywów finansowych z procesem cenowym $S = (S_0, S_1)$. Oczekiwana wartość niepewnej ceny S_1 za rok przy miarze probabilistycznej P wynosi

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

przy $p \equiv P(u)$. Jeśli $S_1 = (20, 5)^T$, a p utrzymuje wartość 0,4, to wartość oczekiwana wynosi

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

Matematycznie wartość oczekiwana może być wyrażona jako **iloczyn skalarny** (lub **wewnętrzny**) dwóch wektorów. Jeśli $x, y \in \mathbb{R}^2$, to iloczyn skalarny definiujemy jako

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

Zatem przy $P = (p, 1 - p)^T$ i $S_1 = (S_1^u, S_1^d)^T$ wartość oczekiwana wynosi

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

Podczas pracy z obiektami `ndarray` w Pythonie iloczyn skalarny jest definiowany jako funkcja udostępniana przez pakiet NumPy:

```
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 Zdefiniowana wcześniej miara probabilistyczna.

❷ Dzisiejsza cena składnika aktywów finansowych.

❸ Wektor niepewnej ceny za rok.

❹ Iloczyn skalarny dwóch wektorów obliczający wartość oczekiwaną.

Oczekiwany zysk

W warunkach niepewności należy dostosować pojęcia zysku i stopy zwrotu. W takiej sytuacji **oczekiwany zysk** z aktywów finansowych jest określany jako oczekiwana cena za rok pomniejszona o dzisiejszą cenę. Można to zobaczyć, przyjmując oczekiwaną wartość niepewnego zysku $R = (R^u, R^d)^T$ i przekształcając ją w następujący sposób:

Kup książkę Poleć książkę

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

Przyjmując wcześniejsze założenia, otrzymujemy:

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

Oczekiwana stopa zwrotu jest więc po prostu oczekiwanym zyskiem podzielonym przez dzisiejszą cenę.

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

Można ją również wyprowadzić krok po kroku za pomocą podobnych przekształceń jak w przypadku oczekiwanego zysku. W dalszej części książki oczekiwaną stopę zwrotu oznaczono dla zwięzłości symbolem $\mu \equiv \mathbf{E}^P(r)$.

Obliczanie oczekiwanego zysku i stopy zwrotu może być modelowane w Pythonie za pomocą dwóch prostych funkcji:

```
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 Definicja oczekiwanego zysku.

❷ Oczekiwany zysk dla uprzednio zdefiniowanego składnika aktywów finansowych.

❸ Definicja oczekiwanej stopy zwrotu.

❹ Oczekiwana stopa zwrotu obliczona dla tego składnika aktywów.

Zmienność

W finansach dominującą parą pojęć jest **ryzyko** i **oczekiwany zysk**. Ryzyko może być mierzone na wiele sposobów, choć prawdopodobnie najbardziej powszechną miarą jest **zmienność** określana przez odchylenie standardowe stóp zwrotu. W obecnym kontekście **wariancja** stóp zwrotu aktywów finansowych jest definiowana przez

Kup książkę Poleć książkę

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

gdzie $r^\omega \equiv (S_1^\omega - S_0) / S_0$, $\omega \in \Omega$. **Zmienność** definiuje się jako odchylenie standardowe stóp zwrotu, które jest pierwiastkiem kwadratowym z wariancji

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

Poniżej podano funkcje Pythona modelujące te dwie miary ryzyka, a także funkcję pomocniczą do obliczania wektora stóp zwrotu:

```
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 Wektoryzowane obliczenie wektora stóp zwrotu.

❷ Zastosowanie funkcji do wcześniejszych aktywów finansowych.

❸ Oczekiwana stopa zwrotu obliczona poprzez iloczyn skalarny…

❹ …i wypisanie jej wartości.

❺ Definicja wariancji stóp zwrotu.

❻ Zastosowanie funkcji do wektora stóp zwrotu.

❼ Definicja zmienności.

❽ I zastosowanie jej do wektora stóp zwrotu.

Kup książkę Poleć książkę

Wektory, macierze i NumPy

Finanse, jako stosowana dyscyplina matematyczna, w dużym stopniu opierają się na algebrze liniowej i teorii prawdopodobieństwa. W dyskretnym modelu gospodarki obie dyscypliny matematyczne można skutecznie obsługiwać w Pythonie za pomocą pakietu NumPy z jego zaawansowanym obiektem `ndarray`. Dotyczy to nie tylko modelowania, ale także obsługi, obliczeń, optymalizacji, wizualizacji i innych aspektów. W zasadzie wszystkie przykłady w tej książce będą potwierdzać te twierdzenia.

Roszczenia warunkowe

Przypuśćmy teraz, że przedmiotem obrotu handlowego w gospodarce jest **roszczenie warunkowe** (ang. *contingent claim*). Jest to składnik aktywów finansowych — sformalizowany przez pewien kontrakt — który oferuje zależną od stanu wypłatę za rok od dziś. Takie roszczenie warunkowe może przewidywać dowolną wypłatę zależną od stanu lub taką, która jest pochodną wypłaty z innych aktywów finansowych. W tym drugim przypadku mówi się zazwyczaj o **aktywach pochodnych** lub **instrumentach pochodnych**. Formalnie roszczenie warunkowe to funkcja $C_1{:}\Omega \to \mathbb{R}_{\geq 0}$, $\omega \longmapsto C_1(\omega)$, odwzorowująca zdarzenia na (nieujemne) liczby rzeczywiste.

Załóżmy, że w gospodarce przedmiotem obrotu są dwa aktywa finansowe: wolna od ryzyka obligacja z procesem cenowym $B = (B_0, B_1)$ oraz ryzykowna akcja z procesem cenowym

$$S = (S_0, (S_1^u, S_1^d)^T)$$

Opcja kupna (ang. *call option*) dla akcji daje za rok wypłatę w wysokości $C_1(S_1(\omega)) = \max(S_1(\omega) - K, 0)$ oraz $\omega \in \Omega$. $K \in \mathbb{R}_{\geq 0}$ nazywamy **ceną wykonania** (ang. *strike price*) opcji.

W teorii prawdopodobieństwa roszczenie warunkowe zwykle jest określane mianem **zmiennej losowej**, której cechą charakterystyczną jest to, że odwzorowuje elementy przestrzeni stanów na liczby rzeczywiste — potencjalnie za pośrednictwem innych zmiennych losowych, jak ma to miejsce w przypadku aktywów pochodnych. W tym sensie cena akcji za rok, $S_1{:}\Omega \to \mathbb{R}_{\geq 0}$, $\omega \longmapsto S_1(\omega)$, jest również zmienną losową[4].

Aby to zilustrować — poniższy kod Pythona wizualizuje wypłatę z opcji kupna na wycinku prostej rzeczywistej. W gospodarce istnieją oczywiście tylko dwa stany, a co za tym idzie, dwie istotne wartości. Rysunek 2.1 przedstawia graficznie funkcję wypłaty:

```
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
```

[4] Formalna definicja zmiennej losowej znajduje się w rozdziale 5.

Kup książkę Poleć książkę

```
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
         # konfiguracja wykresu
         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 Wygenerowanie obiektu `ndarray` z liczbami od 0 do 19.

❷ Wyświetlenie kilku pierwszych liczb.

❸ Ustalenie ceny wykonania dla opcji kupna.

❹ Wektoryzowane obliczenia wartości wypłaty z opcji kupna.

❺ Wyświetlenie tych wartości — wiele z nich jest równych 0.

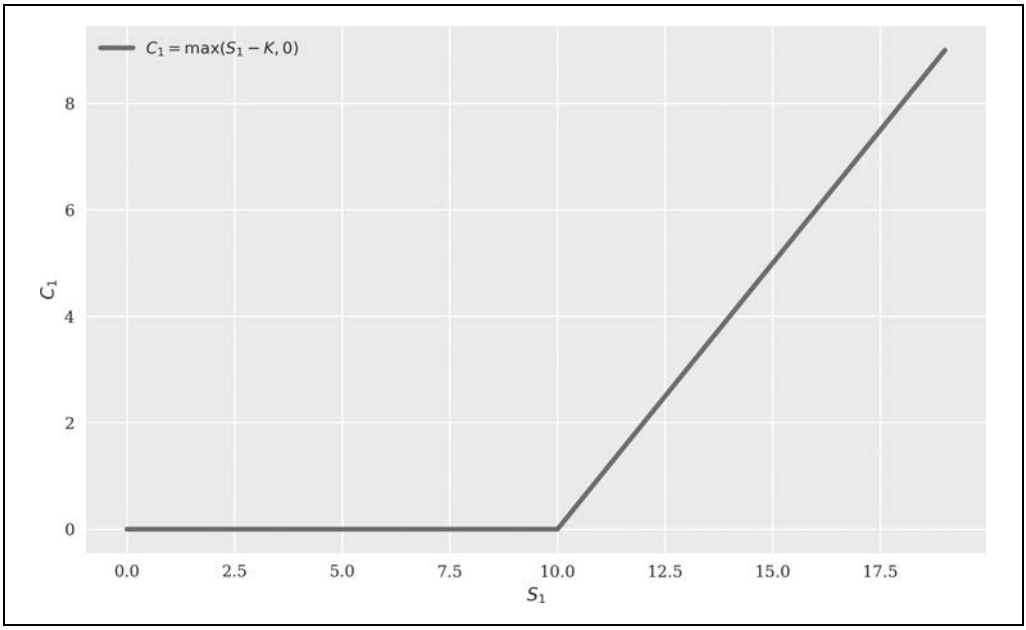
❻ Zaimportowanie głównego podpakietu do tworzenia wykresów z `matplotlib` (*http://matplotlib.org*).

❼ Wykreślenie wartości wypłaty z opcji kupna względem wartości akcji, ustawienie szerokości linii na 3 piksele i zdefiniowanie etykiety jako obiektu łańcucha z kodem `Latex`.

❽ Umieszczenie legendy w optymalnym miejscu (najmniejsze nakładanie się z elementami wykresu).

❾ Umieszczenie etykiety na osi *x*…

❿ …i na osi *y*.



Rysunek 2.1. Wypłata z opcji kupna

Kup książkę Poleć książkę

Replikacja

Gdy wprowadzamy do gospodarki roszczenie warunkowe, pojawia się ważne pytanie, czy wypłata roszczenia warunkowego jest redundantna, czy nie. Z matematycznego punktu widzenia mówi się, że wektor wypłat roszczenia warunkowego jest **liniowo zależny** lub **liniowo niezależny**.

Mówimy, że wypłata z opcji kupna jest liniowo zależna — lub redundantna — gdy istnieje rozwiązanie następującego problemu:

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

gdzie $b, s \in \mathbb{R}$.

Problem ten można przedstawić w postaci **układu równań liniowych**:

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

Przy $S_1^u \neq S_1^d$ rozwiązania otrzymujemy przez:

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

oraz

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

Przyjmijmy, jak poprzednio, że przedmiotem obrotu są dwa aktywa finansowe: wolna od ryzyka obligacja $B = (10, 11)$ i ryzykowna akcja $S = (10, (20, 5)^T)$. Przyjmijmy również, że $K = 15$, a $C_1 = (5, 0)^T$. Optymalnymi rozwiązaniami są wtedy:

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

oraz

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

Innymi słowy, kupno jednej trzeciej akcji i krótka sprzedaż $^5/_{33}$ obligacji doskonale replikuje wypłatę z opcji kupna. Dlatego też wypłata z opcji kupna jest liniowo zależna od wektorów wypłat z obligacji i akcji.

Formalnie rzecz biorąc, **krótka sprzedaż** polega na pożyczeniu dzisiaj odpowiedniej liczby jednostek danego składnika aktywów finansowych od innego agenta i natychmiastowej sprzedaży tych jednostek na rynku. Za rok pożyczający kupuje na rynku dokładnie taką samą liczbę jednostek składnika aktywów finansowych po aktualnej wówczas cenie i przekazuje je z powrotem drugiemu agentowi.

Kup książkę Poleć książkę

W przedstawionej tu analizie zakłada się, że wszystkie aktywa finansowe — podobnie jak pieniądz — są nieskończenie podzielne, co w praktyce może nie być prawdą. Zakłada się również, że możliwa jest krótka sprzedaż wszystkich aktywów finansowych będących przedmiotem obrotu, co zresztą zdarza się w praktyce.

W ramach przygotowania do implementacji w Pythonie rozważmy jeszcze inny sposób sformułowania problemu replikacji. W tym celu potrzebna jest matematyczna koncepcja macierzy. Podczas gdy wektor jest obiektem jednowymiarowym, macierz jest obiektem dwuwymiarowym. Na potrzeby tego punktu rozważmy czteroelementową macierz kwadratową $\mathcal{M}$ — co oznacza, że $\mathcal{M} \in \mathbb{R}^{2\times2}$ — o wartościach:

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

Wektory przyszłych wypłat z obligacji i akcji reprezentują wartości, odpowiednio, w pierwszej i drugiej kolumnie macierzy. Pierwszy wiersz zawiera wypłaty z obu aktywów finansowych w stanie *u*, a drugi zawiera wypłaty ze stanu *d*. Stosując te konwencje, problem replikacji można przedstawić w **postaci macierzowej** jako

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

gdzie $\phi \in \mathbb{R}$ jest wektorem zawierającym pozycje portfela obligacji i akcji dla replikacji $\phi \equiv (b, s)^T$. ϕ zazwyczaj nazywamy po prostu **portfelem** lub **strategią tradingową**. Zatem:

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

W tym kontekście **mnożenie macierzy** definiuje się jako

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

co pokazuje równoważność pomiędzy tym sposobem przedstawienia problemu replikacji a poprzednim.

Klasa `ndarray` pozwala na modelowanie macierzy w Pythonie. Pakiet NumPy dostarcza w podpakiecie `np.linalg` wiele funkcji do operacji z zakresu algebry liniowej, wśród których znajduje się również funkcja do rozwiązywania układów równań liniowych w postaci macierzowej — dokładnie tego, co jest tutaj potrzebne:

```
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
```


Kup książkę Poleć książkę

```
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 Zdefiniowanie procesu cenowego dla obligacji wolnych od ryzyka.

❷ Zdefiniowanie procesu cenowego dla ryzykownych akcji.

❸ Zdefiniowanie macierzy — tj. dwuwymiarowego obiektu `ndarray` — z wektorami przyszłych wypłat.

❹ Wyświetlenie macierzy z wartościami liczbowymi.

❺ Ustalenie ceny wykonania dla opcji kupna i…

❻ …obliczenie wartości wektora wypłat za rok.

❼ Wyświetlenie wartości liczbowych wektora wypłat.

❽ Rozwiązanie problemu replikacji w postaci macierzowej w celu uzyskania optymalnych pozycji portfela.

Arbitraż cenowy

Ile kosztuje replikacja wypłaty z opcji kupna? Po wyznaczeniu portfela replikacyjnego odpowiedź na to pytanie jest prosta. Zdefiniujmy dzisiejszą wartość portfela replikacyjnego jako $V_0(\phi)$. Jest ona określona przez iloczyn skalarny

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

lub liczbowo:

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

Niepewna wartość portfela replikacyjnego za rok $V_1(\phi)$ może być przedstawiona przez mnożenie macierzy jako

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

Sumarycznie otrzymujemy **proces wyznaczania wartości** portfela jako $V(\phi) = (V_0(\phi), V_1(\phi))$, w skrócie $V = (V_0, V_1)$, jeśli nie ma żadnej niejednoznaczności co do portfela.

Posiadanie dostępnego portfela, który doskonale replikuje przyszłą wypłatę z roszczenia warunkowego, rodzi kolejne pytanie: Co w sytuacji, gdy dzisiejsza cena roszczenia warunkowego różni się od kosztów utworzenia portfela replikującego? Odpowiedź jest prosta, ale poważna: w takim przypadku w gospodarce istnieje **arbitraż** lub **możliwość arbitrażu**. Arbitraż to strategia tradingowa ϕ, która tworzy wolny od ryzyka zysk przy zerowej inwestycji. Formalnie rzecz ujmując, ϕ jest arbitrażem, jeśli

Kup książkę Poleć książkę

$$V_0(\phi) = 0 \text{ i } \mathbf{E}^P(V_1(\phi)) > 0$$

lub

$$V_0(\phi) > 0 \text{ i } V_1(\phi) = 0$$

Załóżmy, że cena opcji kupna wynosi $C_0 = 2$, co przewyższa koszt utworzenia portfela replikującego. Strategia tradingowa, która polega na sprzedaży opcji kupna na rynku za 2 i kupnie portfela replikującego za 1,81818, przynosi natychmiastowy zysk w wysokości różnicy. Po upływie roku zysk z portfela replikującego i z opcji kupna znoszą się nawzajem:

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

zgodnie z definicją portfela replikującego. W drugim przypadku, gdy dzisiejsza cena opcji kupna jest niższa od ceny portfela replikującego, powiedzmy $C_0 = 1{,}5$, strategia tradingowa polegająca na kupnie opcji kupna i sprzedaży portfela replikującego przynosi wolny od ryzyka zysk równy różnicy między ceną rynkową opcji kupna a kosztem utworzenia portfela replikującego. Oczywiście, w obu przypadkach wolny od ryzyka zysk można zwiększyć przez zwykłe pomnożenie pozycji przez dodatni współczynnik większy niż jeden.

Model gospodarki, który dopuszcza możliwość arbitrażu, można uznać za nierealny. Dlatego jedyną ceną, która jest spójna z brakiem arbitrażu, jest $C_0 = 1{,}818181$. Nazywamy to **ceną arbitrażową** opcji kupna. Jeśli istnieje portfel ϕ replikujący wypłatę z roszczenia warunkowego $V_1(\phi) = C_1$, to cena arbitrażowa roszczenia warunkowego wynosi $C_0 = V_0(\phi)$.

Formalnie rzecz biorąc, cena arbitrażowa jest iloczynem skalarnym portfela replikacyjnego i wektora cen replikujących aktywów finansowych:

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

co powoduje powstanie procesu cenowego wolnego od arbitrażu dla roszczenia warunkowego $C = (C_0, C_1)$.

W Pythonie jest to pojedyncze działanie, z uwzględnieniem poprzednich definicji i obliczeń:

```
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
```

Zupełność rynku

Czy wycena arbitrażowa działa w przypadku każdego roszczenia warunkowego? Tak, przynajmniej dla tych, które są replikowalne przez portfele aktywów finansowych będących przedmiotem obrotu w gospodarce. **Zbiór osiągalnych roszczeń warunkowych** $\mathbb{A}$ obejmuje wszystkie te

Kup książkę Poleć książkę

roszczenia warunkowe, które są replikowalne przez obrót aktywami finansowymi. Jest on określony przez **rozpiętość**, która jest zbiorem wszystkich liniowych kombinacji wektorów przyszłych cen aktywów finansowych będących przedmiotem obrotu

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

jeżeli krótka sprzedaż jest zabroniona, oraz

$$\mathbb{A} = \{\mathcal{M} \cdot \phi, \phi \in \mathbb{R}^2\}$$

jeśli jest dozwolona bez ograniczeń.

Rozważmy wcześniejszy przypadek obligacji bez ryzyka i ryzykownej akcji z procesami cenowymi wynoszącymi, odpowiednio, $B = (B_0, B_1)$ i $S = (S_0, (S_1^u, S_1^d)^T)$, gdzie $B_1, S_1 \in \mathbb{R}^2_{\geq 0}$ i $S_1^u \neq S_1^d$. Łatwo wówczas wykazać, że problem replikacji

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

ma unikatowe rozwiązanie dla każdego $C_1 \in \mathbb{R}^2_{\geq 0}$. Rozwiązanie to jest wyznaczone przez

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

i w konsekwencji jako

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

co zostało wyprowadzone w kontekście replikacji szczególnej wypłaty z opcji kupna. Rozwiązanie przenosi się na przypadek ogólny, nie przyjęto bowiem żadnych szczególnych założeń dotyczących wypłaty poza $C_1 \in \mathbb{R}^2_{\geq 0}$.

Ponieważ *każde* roszczenie warunkowe można zreplikować za pomocą portfela, który składa się z pozycji obejmujących wolne od ryzyka obligacje i ryzykowne akcje, mówi się o **modelu rynku zupełnego**. A zatem każde roszczenie warunkowe można wycenić za pomocą replikacji i arbitrażu. Formalnie jedynym wymogiem jest to, aby wektory cen dwóch aktywów finansowych w ciągu jednego roku były liniowo niezależne. Wynika z tego, że

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

ma tylko jedno unikatowe rozwiązanie $\phi^* = (0, 0)^T$ i żadnego więcej. W rzeczywistości wszystkie problemy z replikacją dla dowolnych roszczeń warunkowych mają unikatowe rozwiązania w warunkach zupełności rynku. Wektory wypłat dwóch zbywalnych aktywów finansowych rozpinają przestrzeń $\mathbb{R}^2$, ponieważ tworzą **bazę** dla **przestrzeni wektorowej** $\mathbb{R}^2$.

Kup książkę Poleć książkę

Własność rozpinania może być zwizualizowana przy użyciu Pythona i pakietu `matplotlib` (*https://matplotlib.org*). W tym celu symuluje się 1000 losowych składów portfela. Pierwsze ograniczenie polega na tym, że pozycje portfela powinny być dodatnie i sumować się do 1. Otrzymany wynik jest pokazany na rysunku 2.2:

```
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 Ustalenie ziarna dla generatora liczb losowych.

❷ Liczba wartości, które mają być symulowane.

❸ Symulacja pozycji obligacji dla wartości pomiędzy 0 a 1 przy użyciu rozkładu jednostajnego.

❹ Wyznaczenie pozycji akcji jako różnicy pomiędzy 1 a pozycją obligacji.

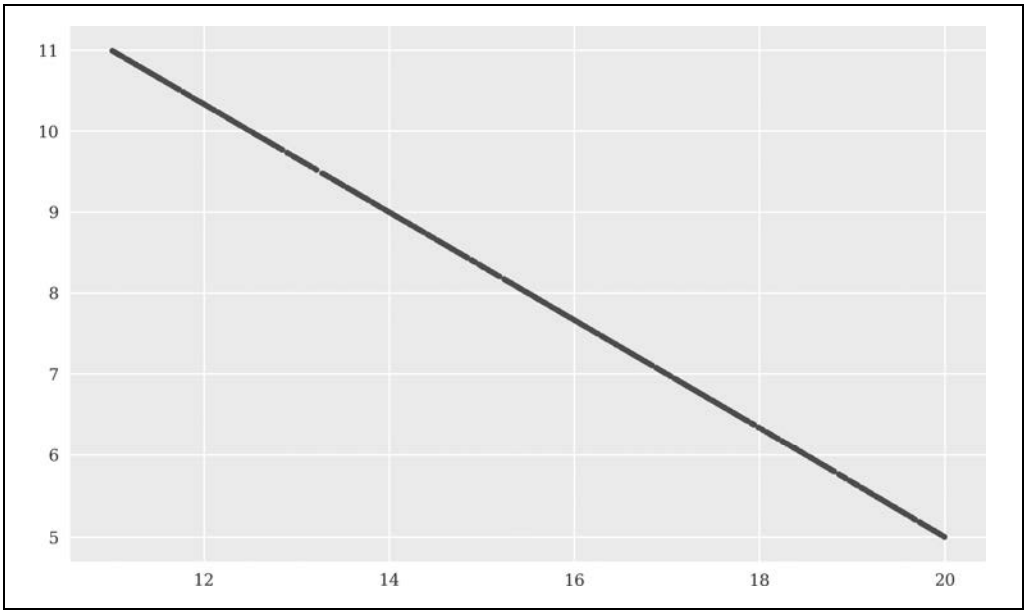
❺ Obliczenie wektorów wypłat portfela dla wszystkich losowych składów portfeli i zebranie ich w obiekcie `list`; ten idiom Pythona nosi nazwę listy składanej.

❻ Funkcja zwracająca tablicę `ndarray` z wynikami.

❼ Rozpoczęcie obliczeń.

❽ Wykreślenie wyników.

Kup książkę Poleć książkę



Rysunek 2.2. Losowe portfele rozpinające jedynie jednowymiarową linię

Rysunek 2.3 przedstawia graficznie wyniki w przypadku, gdy pozycje portfela nie muszą sumować się do 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 Pozycja akcji jest symulowana swobodnie dla wartości pomiędzy 0 a 1.

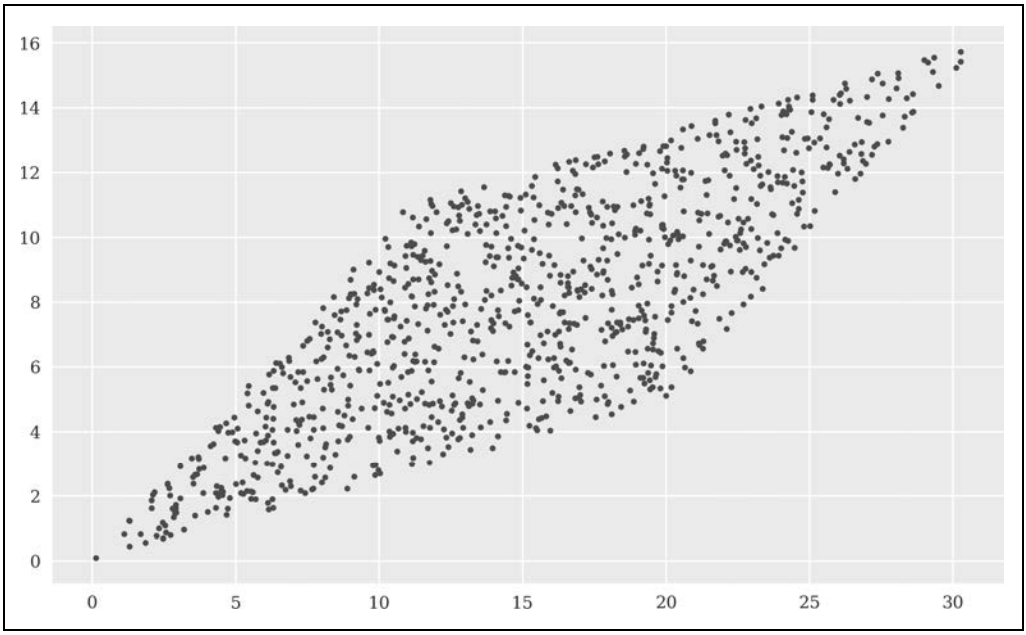
❷ Obliczanie wektorów wypłat portfela.

Na koniec, rysunek 2.4 dopuszcza zarówno dodatnie, jak i ujemne pozycje portfela zarówno dla obligacji, jak i akcji. Wynikowe wektory wypłat portfela pokrywają (eliptyczny) obszar wokół początku:

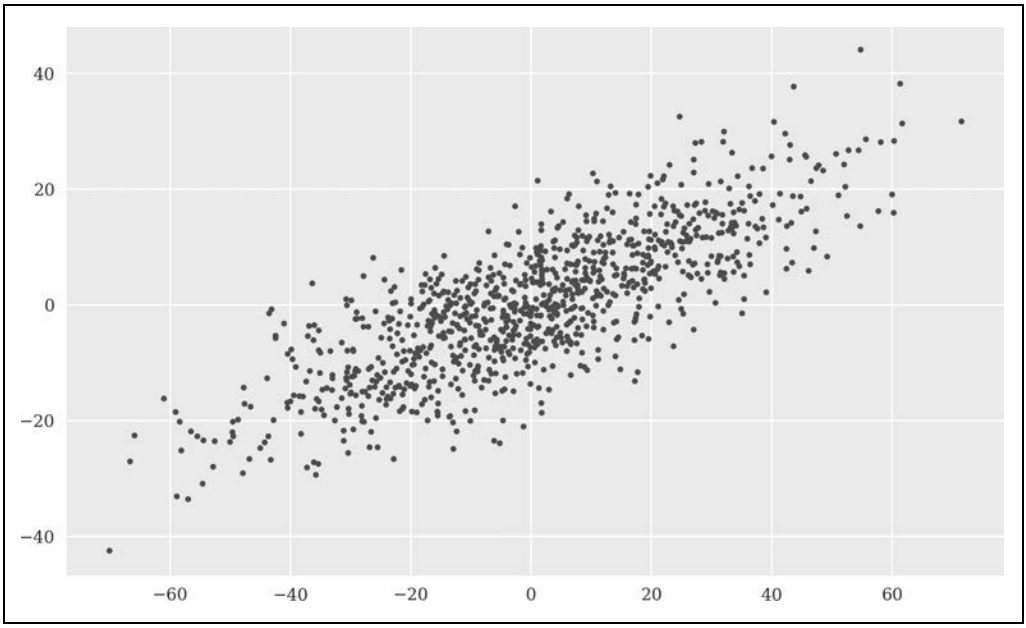
```
In [109]: b = rng.standard_normal(n) ❶

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 Pozycje dodatnie i ujemne w portfelu są symulowane przy użyciu standardowego rozkładu normalnego.

Kup książkę Poleć książkę



Rysunek 2.3. Losowe portfele rozpinające dwuwymiarowy obszar (romb)



Rysunek 2.4. Losowe portfele rozpinające dwuwymiarowy obszar (wokół początku)

Jeśli b i s mogą przyjmować dowolne wartości na prostej rzeczywistej, $b, s \in \mathbb{R}$, to portfele wynikowe całkowicie pokrywają przestrzeń wektorową $\mathbb{R}^2$. Jak wspomniano wcześniej, wektory wypłat z aktywów finansowych będących przedmiotem obrotu rozpinają w tym przypadku przestrzeń $\mathbb{R}^2$.

Kup książkę Poleć książkę

Papiery wartościowe Arrowa-Debreu

Papier wartościowy Arrowa-Debreu jest definiowany przez fakt, że zapewnia on wypłatę dokładnie jednej jednostki waluty w określonym stanie przyszłym. W modelowej gospodarce z dwoma różnymi stanami przyszłymi mogą istnieć tylko dwa różne papiery wartościowe tego typu. Papier wartościowy Arrowa-Debreu jest po prostu szczególnym przypadkiem roszczenia warunkowego, w którym ma zastosowanie przedstawiony wcześniej argument replikacji. Innymi słowy, ponieważ rynek jest zupełny, papiery wartościowe Arrowa-Debreu mogą być replikowane przez portfele obligacji i akcji. A zatem oba problemy replikacji mają (unikatowe) rozwiązania i oba papiery wartościowe mają unikatowe ceny arbitrażowe. Dwa problemy replikacji to

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

i

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

Dlaczego te papiery wartościowe są tak ważne? Z matematycznego punktu widzenia dwa wektory wypłat tworzą **bazę standardową** (zwaną też **bazą naturalną**) dla przestrzeni wektorowej $\mathbb{R}^2$. To z kolei implikuje, że każdy wektor tej przestrzeni może być jednoznacznie wyrażony (zreplikowany) jako kombinacja liniowa wektorów, które tworzą bazę standardową. Z finansowego punktu widzenia utworzenie bazy dla gospodarki modelowej przez zastąpienie pierwotnych wektorów przyszłych cen obligacji i akcji papierami wartościowymi Arrowa-Debreu znacząco upraszcza problem replikacji dla wszystkich pozostałych roszczeń warunkowych.

Proces ten polega na wyznaczeniu w pierwszej kolejności portfeli replikacyjnych dla dwóch papierów wartościowych Arrowa-Debreu i wynikających z nich cen arbitrażowych dla obu papierów. Pozostałe roszczenia warunkowe są wtedy replikowane i wyceniane na podstawie bazy standardowej i cen arbitrażowych tych dwóch papierów wartościowych.

Rozważmy dwa papiery wartościowe Arrowa-Debreu z procesami cenowymi $\gamma^u = (\gamma_0^u, (1, 0)^T)$ oraz $\gamma^d = (\gamma_0^d, (0, 1)^T)$ i zdefiniujmy:

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

Rozważmy ogólne roszczenie warunkowe z wektorem przyszłych wypłat:

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

Portfel replikacyjny ϕ^γ dla roszczenia warunkowego jest wtedy trywialnie określony przez $\phi^\gamma = (C_1^u, C_1^d)^T$, ponieważ:

$$V_1(\phi^\gamma) = \mathcal{M}^\gamma \cdot \phi^\gamma$$

Kup książkę Poleć książkę

$$= \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix}$$
$$= \begin{pmatrix} C_1^u \\ C_1^d \end{pmatrix}$$

W związku z tym cena arbitrażowa dla roszczenia warunkowego wynosi:

$$C_0 = V_0(\phi^\gamma) = C_1^u \cdot \gamma_0^u + C_1^d \cdot \gamma_0^d$$

To pokazuje, jak wprowadzenie papierów wartościowych Arrowa-Debreu upraszcza replikację roszczeń warunkowych i wycenę arbitrażową.

Wycena martyngałowa

Miara martyngałowa Q:$\wp(\Omega) \to \mathbb{R}_{\geq 0}$ jest szczególnym rodzajem miary probabilistycznej. Sprawia ona, że zdyskontowany proces cenowy składnika aktywów finansowych jest **martyngałem**. Aby akcja była martyngałem w ramach Q, musi zachodzić następująca zależność:

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

Jeżeli $i = \frac{B_1 - B_0}{B_0}$, zależność ta jest trywialnie spełniona dla wolnych od ryzyka obligacji:

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

Mówi się również o tym, że procesy cenowe dryfują (średnio) wraz z wolną od ryzyka stopą procentową w ramach miary martyngałowej:

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

Oznaczmy $q \equiv Q(u)$. Otrzymujemy:

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

lub po kilku prostych manipulacjach:

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

Biorąc pod uwagę poprzednie założenia, aby q definiowało poprawną miarę probabilistyczną, musi zachodzić $S_1^u > S_0 \cdot (1+i) > S_1^d$. Jeśli tak jest, to otrzymujemy nową przestrzeń probabilistyczną $(\Omega, \wp(\Omega), Q)$, gdzie Q zastępuje P.

A co, jeśli te zależności dla S_1 nie zachodzą? Wtedy **prosty arbitraż** polega albo na zakupie ryzykownego składnika aktywów w przypadku $S_0 \cdot (1+i) \leq S_1^d$, albo po prostu na jego sprzedaży w innym przypadku, $S_0 \cdot (1+i) \geq S_1^u$.

Kup książkę Poleć książkę

Jeśli w tych relacjach zachodzi równość, mówi się także o **słabym arbitrażu**, ponieważ wolnego od ryzyka zysku można oczekiwać tylko średnio, a nie z pewnością.

Zakładając wcześniejsze liczbowe procesy cenowe, obliczenie q w Pythonie oznacza po prostu działanie arytmetyczne na liczbach zmiennoprzecinkowych:

```
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
```

Pierwsze fundamentalne twierdzenie wyceny aktywów

Rozważania na końcu poprzedniego punktu wskazują na związek pomiędzy miarami martyngałowymi z jednej strony a arbitrażem z drugiej. W matematyce finansowej najważniejszym wnioskiem, który formalnie łączy te pozornie niepowiązane pojęcia, jest **pierwsze fundamentalne twierdzenie wyceny aktywów**. Pionierskie prace w tym zakresie opublikowali Cox i Ross (1976), Harrison i Kreps (1979) oraz Harrison i Pliska (1981).

Pierwsze fundamentalne twierdzenie wyceny aktywów (1FTWA)

Następujące stwierdzenia są równoważne:

1. Istnieje miara martyngałowa.
2. Gospodarka jest wolna od arbitrażu.

Biorąc pod uwagę wcześniejsze obliczenia i rozważania, twierdzenie jest łatwe do udowodnienia dla modelu gospodarki z wolnymi od ryzyka obligacjami i ryzykownymi akcjami.

Po pierwsze, stwierdzenie 1. implikuje stwierdzenie 2.: jeśli istnieje miara martyngałowa, procesy cenowe nie pozwalają na proste (słabe) arbitraże. Ponieważ dwa wektory przyszłych cen są liniowo niezależne, każde roszczenie warunkowe można zreplikować przez obrót dwoma aktywami finansowymi, co implikuje unikatowe ceny arbitrażowe. Zatem arbitraże nie istnieją.

Po drugie, stwierdzenie 2. implikuje stwierdzenie 1.: jeśli model gospodarki jest wolny od arbitrażu, istnieje miara martyngałowa, jak pokazano wcześniej.

Wycena na podstawie oczekiwań

Z 1FTWA wypływa wniosek, że każde osiągalne roszczenie warunkowe $C_1 \in \mathbb{A}$ można wycenić, uwzględniając oczekiwanie na podstawie miary martyngałowej jego przyszłej wypłaty i dyskontując za pomocą wolnej od ryzyka stopy procentowej. Cena arbitrażowa opcji kupna jest znana poprzez replikację. Zakładając te same liczbowe procesy cenowe dla aktywów finansowych będących przedmiotem obrotu oraz ten sam liczbowy wektor przyszłych wypłat dla opcji kupna, **cena martyngałowa** opcji kupna wynosi:

Kup książkę Poleć książkę

$$
\begin{aligned}
C_0 &= \frac{1}{1+i} \cdot \mathbf{E}^Q(C_1) \\
&= \frac{1}{1+i} \cdot (q \cdot C_1^u + (1-q) \cdot C_1^d) \\
&= \frac{1}{1+0.1} \cdot (0{,}4 \cdot 5 + (1-0{,}4) \cdot 0) \\
&= 1{,}818181
\end{aligned}
$$

Innymi słowy, zdyskontowany proces cenowy opcji kupna — i każdego innego roszczenia warunkowego — jest martyngałem w ramach miary martyngałowej, takim że:

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

W Pythonie wycena martyngałowa sprowadza się do oszacowania iloczynu skalarnego:

```
In [118]: Q = (q, 1 - q) ❶

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

❶ Zdefiniowanie miary martyngałowej jako krotki `Q`.

❷ Implementacja formuły wyceny martyngałowej.

Drugie fundamentalne twierdzenie wyceny aktywów

Istnieje jeszcze jeden ważny wniosek, często nazywany **drugim fundamentalnym twierdzeniem wyceny aktywów**, który wiąże unikatowość miary martyngałowej z zupełnością rynku.

Drugie fundamentalne twierdzenie wyceny aktywów (2FTWA)

Następujące stwierdzenia są równoważne:

1. Miara martyngałowa jest unikatowa.
2. Model rynku jest zupełny.

Wniosek ten wynika również z wcześniejszych rozważań dotyczących prostego modelu gospodarki. Bardziej szczegółowa analiza zupełności rynku jest zawarta w rozdziale 3.

Portfel średniej-wariancji

Istotnym przełomem w finansach było sformalizowanie i kwantyfikacja inwestowania portfelowego dzięki **teorii portfela średniej-wariancji** (MVP, ang. *mean-variance portfolio theory*), której pionierem był Markowitz (1952). Do pewnego stopnia podejście to można uznać za początek matematyki finansowej, zapoczątkowało bowiem trend wprowadzania do finansów coraz więcej matematyki.

Kup książkę Poleć książkę

MVP redukuje składnik aktywów finansowych do pierwszego i drugiego momentu jego zwrotów, czyli **średniej** jako oczekiwanej stopy zwrotu i **wariancji** stóp zwrotu lub **zmienności**, określonej jako odchylenie standardowe stóp zwrotu. Chociaż podejście to jest ogólnie określane jako „średnia-wariancja", często używa się zestawienia „średnia-zmienność".

Rozważmy omawiany uprzednio przypadek wolnej od ryzyka obligacji i ryzykownej akcji z procesami cenowymi $B = (B_0, B_1)$ i $S = (S_0, (S_1^u, S_1^d)^T)$ oraz macierz przyszłych cen $\mathscr{M}$, której dwie kolumny są wektorami przyszłych cen tych dwóch aktywów finansowych. Jaka jest oczekiwana stopa zwrotu i zmienność portfela ϕ, który składa się z b procent zainwestowanych w obligacje i s procent zainwestowanych w akcje? Zauważmy, że teraz zakłada się sytuację, w której $b + s = 1$, gdzie $b, s \in \mathbb{R}_{\geq 0}$. To założenie można oczywiście nieco złagodzić, ale upraszcza to prezentację w tym rozdziale.

Oczekiwana wypłata z portfela wynosi:

$$\begin{aligned}
\mathbf{E}^P(\mathscr{M} \cdot \phi) &= p \cdot (b \cdot B_1 + s \cdot S_1^u) + (1-p) \cdot (b \cdot B_1 + s \cdot S_1^d) \\
&= p \cdot (b \cdot B_1) + (1-p) \cdot (b \cdot B_1) + p \cdot (s \cdot S_1^u) + (1-p)(s \cdot S_1^d) \\
&= b \cdot \mathbf{E}^P(B_1) + s \cdot \mathbf{E}^P(S_1) \\
&= b \cdot B_1 + s \cdot \mathbf{E}^P(S_1)
\end{aligned}$$

Innymi słowy, oczekiwana wypłata z portfela to po prostu iloczyn b i wypłaty z wolnych od ryzyka obligacji plus iloczyn s i oczekiwanej wypłaty z akcji.

Przez zdefiniowanie $\mathscr{R} \in \mathbb{R}^{2\times 2}$ jako **macierzy stóp zwrotu** o wartościach

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

otrzymujemy oczekiwaną stopę zwrotu z portfela:

$$\begin{aligned}
\mathbf{E}^P(\mathcal{R} \cdot \phi) &= b \cdot \mathbf{E}^P(i) + s \cdot \mathbf{E}^P(r_1) \\
&= b \cdot i + s \cdot \mu
\end{aligned}$$

Innymi słowy, oczekiwana stopa zwrotu portfela to iloczyn b i stopy procentowej wolnej od ryzyka plus iloczyn s i oczekiwanej stopy zwrotu z akcji.

Następnym krokiem jest obliczenie **wariancji portfela**:

$$\begin{aligned}
\sigma^2(\mathcal{R} \cdot \phi) &= \mathbb{E}^P((r - \mathbf{E}^P(\mathcal{R} \cdot \phi))^2) \\
&= \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} (b \cdot i + s \cdot r_1^u - b \cdot i - s \cdot \mu)^2 \\ (b \cdot i + s \cdot r_1^d - b \cdot i - s \cdot \mu)^2) \end{pmatrix} \right) \\
&= \left(\begin{pmatrix} p \\ 1-p \end{pmatrix}, \begin{pmatrix} (s \cdot r_1^u - s \cdot \mu)^2 \\ (s \cdot r_1^d - s \cdot \mu)^2 \end{pmatrix} \right) \\
&= s^2 \cdot \sigma^2(r_1)
\end{aligned}$$

Kup książkę Poleć książkę

Innymi słowy, wariancja portfela to iloczyn s^2 i wariancji akcji, co ma intuicyjne uzasadnienie, ponieważ obligacja jest wolna od ryzyka i nie powinna się przyczyniać do wariancji portfela. Wynika to bezpośrednio z wyniku proporcjonalności dla **zmienności portfela**:

$$\sigma(\mathcal{R} \cdot \phi) = \sqrt{\sigma^2(\mathcal{R} \cdot)}$$
$$= \sqrt{s^2 \cdot \sigma^2(r_1)}$$
$$= \quad s \cdot \sigma(r_1)$$

Całą analizę można łatwo zaimplementować w Pythonie. Na początek kilka wstępnych założeń:

```
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 Macierz z przyszłymi cenami aktywów finansowych.

❷ Wektor z dzisiejszymi cenami aktywów finansowych.

❸ Wektoryzowane obliczenie macierzy stóp zwrotu.

❹ Wyświetlenie wyników obliczeń.

❺ Zdefiniowanie miary probabilistycznej.

Przy takich definicjach oczekiwany zysk z portfela i zmienność oblicza się jako iloczyny skalarne:

```
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
```


Kup książkę Poleć książkę

```
In [133]: sigma = s * R[:, 1].std() ❻

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

❶ Oczekiwane zyski z obligacji i akcji.

❷ Przykładowa alokacja dla akcji w procentach (jako wartość po przecinku).

❸ Portfel wynikowy z wagą znormalizowaną równą 1.

❹ Oczekiwany zwrot z portfela przy uwzględnieniu alokacji.

❺ Wartość mieści się pomiędzy zyskiem wolnym od ryzyka a zyskiem z akcji.

❻ Zmienność portfela; kod Pythona ma tutaj zastosowanie tylko ze względu na $p = 0{,}5$.

❼ Również w tym przypadku wartość mieści się pomiędzy zmiennością obligacji (= 0) a zmiennością akcji (= 0,75).

Zmiana wagi akcji w portfelu prowadzi do różnych kombinacji ryzyka i zwrotu. Na rysunku 2.5 przedstawiono oczekiwany zysk z portfela oraz zmienność dla różnych wartości `s` z przedziału od 0 do 1. Jak widać na wykresie, zarówno oczekiwany zysk z portfela (od 0,1 do 0,25), jak i zmienność (od 0,0 do 0,75) rosną liniowo wraz ze wzrostem alokacji akcji `s`:

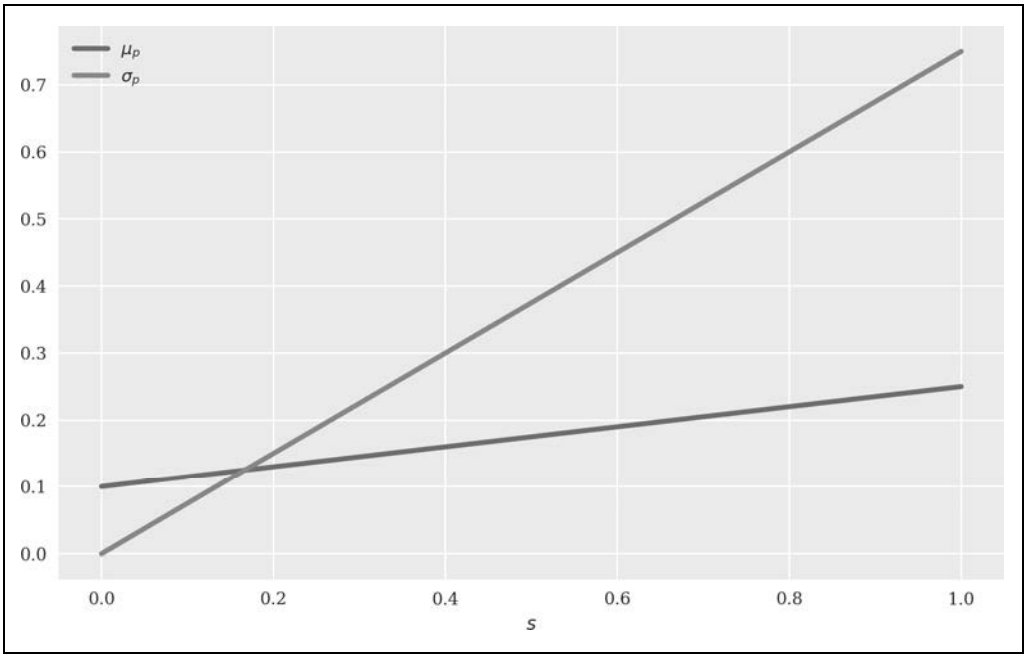
```
In [135]: values = np.linspace(0, 1, 25) ❶

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 Wygenerowanie obiektu `ndarray` zawierającego 24 równomiernie rozmieszczone przedziały pomiędzy 0 a 1.

❷ Obliczenie oczekiwanego zysku z portfela dla każdego elementu w `values` i zapisanie go w obiekcie `list`.

❸ Obliczenie zmienności portfela dla każdego elementu w `values` i zapisanie ich w kolejnym obiekcie `list`.

Kup książkę Poleć książkę



Rysunek 2.5. Oczekiwany zysk z portfela i zmienność dla różnych alokacji

Zauważmy, że lista składana `sigma = [s * R[:, 1].std() for s in values]` w poprzednim kodzie jest skrótem dla poniższego kodu[5]:

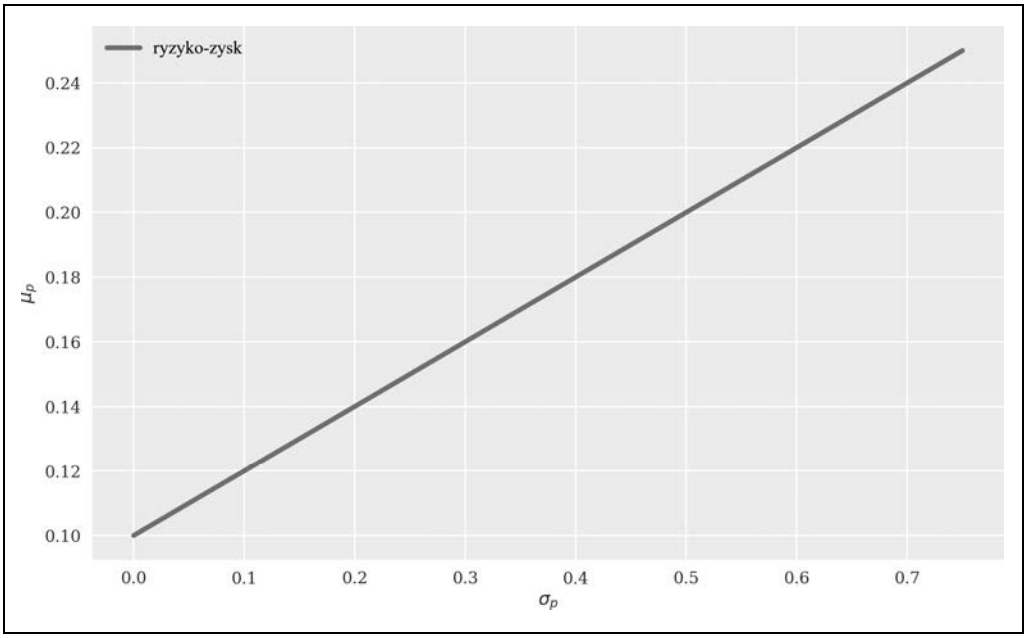
```
sigma = list()
for s in values:
    sigma.append(s * R[:, 1].std())
```

W kontekście MVP najczęściej można spotkać wykresy przedstawiające oczekiwany zysk z portfela w stosunku do zmienności portfela. Rysunek 2.6 pokazuje, że inwestor może oczekiwać tym wyższego zysku z kapitału, im większe ryzyko (zmienność) jest skłonny ponieść. W przedstawionym tutaj szczególnym przypadku zależność ta ma charakter liniowy:

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

[5] Więcej informacji na temat struktur danych i list składanych można znaleźć w rozdziale *Data Structures* dokumentacji Pythona (*https://oreil.ly/0dbCi*).

Kup książkę Poleć książkę



Rysunek 2.6. Możliwe kombinacje oczekiwanego zysku z portfela i zmienności

Wnioski

Ten rozdział stanowi wprowadzenie do finansów od podstaw, przedstawiając najważniejsze obiekty matematyczne i pojęcia finansowe za pomocą prostych przykładów w kodzie Pythona. Piękne jest to, że podstawowe idee finansów, takie jak wycena arbitrażowa czy relacja między ryzykiem a zyskiem, mogą być wprowadzone i wyjaśnione nawet na przykładzie statycznej gospodarki dwustanowej. Znajomość tych podstawowych pojęć oraz odrobina intuicji finansowej i matematycznej znacznie ułatwiają przechodzenie do coraz bardziej realistycznych modeli finansowych. W następnym rozdziale do przestrzeni stanów dodany zostanie trzeci stan przyszły, co pozwoli na omówienie kwestii pojawiających się w kontekście niezupełności rynku.

Inne źródła

Książki i artykuły cytowane w tym rozdziale:

Cox John, Ross Stephen, *The Valuation of Options for Alternative Stochastic Processes*, „Journal of Financial Economics", 1976, 3, s. 145 – 166.

Delbaen Freddy, Schachermayer Walter, *The Mathematics of Arbitrage*, Springer Verlag, Berlin 2006.

Kup książkę Poleć książkę

Guidolin Massimo, Rinaldi Francesca, *Ambiguity in Asset Pricing and Portfolio Choice: A Review of the Literature*, „Theory and Decision", 2013, 74, s. 183 – 217, *https://ssrn.com/abstract=1673494*.

Harrison Michael, Kreps David, *Martingales and Arbitrage in Multiperiod Securities Markets*, „Journal of Economic Theory", 1979, 20, s. 381 – 408.

Harrison Michael, Pliska Stanley, *Martingales and Stochastic Integrals in the Theory of Continuous Trading*, „Stochastic Processes and their Applications", 1981, 11, s. 215 – 260.

Markowitz Harry, *Portfolio Selection*, „Journal of Finance", 1952, 7, 1, s. 77 – 91.

Kup książkę Poleć książkę

Kup książkę Poleć książkę


PROGRAM
PARTNERSKI
GRUPY HELION
1. ZAREJESTRUJ SIĘ
2. PREZENTUJ KSIĄŻKI
3. ZBIERAJ PROWIZJĘ
Zmień swoją stronę WWW w działający bankomat!
Dowiedz się więcej i dołącz już dzisiaj!
http://program-partnerski.helion.pl
GRUPA
Helion

Teoria finansów? Z Pythonem to dziecinnie proste!

Rozwój technologii i dostęp do danych finansowych stały się ogromnym ułatwieniem w korzystaniu z globalnych rynków finansowych. Jeśli zechcesz, możesz szybko zacząć przygodę na przykład z handlem algorytmicznym. Wystarczy, że masz niewielkie pojęcie o matematyce, programowaniu i ekonomii. Niestety, nieliczne programy nauczania o finansach integrują ze sobą te trzy dziedziny. Tymczasem koncepcje matematyczne wspaniale ułatwiają zrozumienie pojęć z zakresu inżynierii finansowej, a wczesne włączanie ćwiczeń programistycznych pozwala na znaczne zwiększenie efektywności takiej edukacji.

Dzięki tej praktycznej, przystępnie napisanej książce szybko zrozumiesz podstawy teorii finansów, modelowania danych finansowych i zastosowania Pythona w finansach obliczeniowych. Znajdziesz tu systematyczne wprowadzenie do inżynierii finansowej, handlu algorytmicznego czy zarządzania aktywami. Zdobędziesz umiejętności tworzenia w Pythonie programów, które ułatwią Ci rozwiązywanie takich problemów jak ustalanie składu portfeli inwestycyjnych zgodnie z nowoczesną teorią portfela, a także wycena opcji i innych instrumentów pochodnych. Jeśli zajmujesz stanowisko kierownicze w branży finansowej, z pewnością przyda Ci się wiedza o zastosowaniu Pythona w finansach. Jeśli już biegle kodujesz w Pythonie, łatwiej skorzystasz ze swoich umiejętności w tworzeniu przydatnych aplikacji z zakresu inżynierii finansowej.

W książce między innymi:

- matematyczne podstawy teorii finansów i programowania w Pythonie
- modele ekonomiczne i modelowanie danych finansowych
- zastosowanie Pythona w obliczeniach związanych z finansami
- wycena, podejmowanie decyzji, równowaga i alokacja aktywów
- zastosowanie bibliotek i narzędzi Pythona w modelowaniu finansowym

Dr Yves J. Hilpisch jest przedsiębiorcą i wykładowcą. Zarządza również programami szkoleniowymi online, dzięki którym można zdobyć certyfikaty w zakresie zastosowania Pythona w handlu algorytmicznym i finansach obliczeniowych. Napisał bibliotekę do analizy finansowej — DX Analytics. Występował na wielu prestiżowych konferencjach naukowych w USA, Europie i Azji.

Helion
helion.pl
HELION SA
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl
Sprawdź nasze szkolenia!
SZKOLENIA
AKADEMIA IT & BUSINESS
HELIONSZKOLENIA.PL
KOD KORZYŚCI
Sięgnij po więcej!
ISBN 978-83-283-8923-6
9 788328 389236
INFORMATYKA W NAJLEPSZYM WYDANIU
Cena: 49,90 zł