**CHƯƠNG 3**

**HÀM KHỐI XÁC SUẤT**

The code for this chapter is in probability.py. For information about downloading and working with this code, see “Using the Code” on page xi.

Mã code cho chương này trong tập tin probability.py. Để biết thông tin về cái tải và cách làm việc với đoạn mã này hãy xem “Using the Code” ở trang xi.

**Pmfs**

Another way to represent a distribution is a probability mass function (PMF), which maps from each value to its probability. A probability is a frequency expressed as a fraction of the sample size, n. To get from frequencies to probabilities, we divide through by n, which is called normalization.

Một cách khác để biểu diễn một phân bổ là hàm khối xác xuất (PMF), ánh xạ từ mỗi giá trị đến xác xuất của nó. Một xác xuất là một tần suất được biểu diễn dưới dạng một phần nhỏ của kích thước mẫu n. Để chuyển từ tần số sang xác xuất, chúng ta chia cho n, được gọi là chuẩn hóa.

Given a Hist, we can make a dictionary that maps from each value to its probability:

Đưa ra một Lịch sử, chúng ta có thể tạo một từ điển ánh xạ từ từng giá trị đến xác suất của nó:

n = hist.Total()

d = {}

for x, freq in hist.Items():

d[x] = freq / n

Or we can use the Pmf class provided by thinkstats2. Like Hist, the Pmf constructor can take a list, pandas Series, dictionary, Hist, or another Pmf object. Here’s an example with a simple list:

Pmf do thinkstats2 cung cấp.Giống như Hist, hàm tạo Pmf có thể lấy một danh sách, pandas Series, từ điển, Hist hoặc một đối tượng Pmf khác. Đây là một ví dụ với một danh sách đơn giản:

>>> import thinkstats2

>>> pmf = thinkstats2.Pmf([1, 2, 2, 3, 5])

>>> pmf Pmf({1: 0.2, 2: 0.4, 3: 0.2, 5: 0.2})

The Pmf is normalized so total probability is 1.

Pmf được chuẩn hóa nên tổng xác suất là 1.

Pmf and Hist objects are similar in many ways; in fact, they inherit many of their meth‐ ods from a common parent class. For example, the methods Values and Items work the same way for both. The biggest difference is that a Hist maps from values to integer counters; a Pmf maps from values to floating-point probabilities.

Các đối tượng Pmf và Hist giống nhau theo nhiều cách; trên thực tế, chúng kế thừa nhiều phương thức của chúng từ một lớp cha chung. Ví dụ: các phương thức Giá trị và Mục hoạt động theo cùng một cách cho cả hai. Sự khác biệt lớn nhất là Hist ánh xạ từ các giá trị sang bộ đếm số nguyên; một ánh xạ Pmf từ các giá trị sang xác suất dấu phẩy động.

To look up the probability associated with a value, use Prob:

Để tra cứu xác suất liên quan đến một giá trị, hãy sử dụng Prob:

>>> pmf.Prob(2)

0.4

The bracket operator is equivalent:

Toán tử dấu ngoặc là tương đương:

>>> pmf[2]

0.4

You can modify an existing Pmf by incrementing the probability associated with a value:

Bạn có thể sửa đổi một Pmf hiện có bằng cách tăng xác suất được liên kết với một giá trị:

>>> pmf.Incr(2, 0.2)

>>> pmf.Prob(2)

0.6

Or you can multiply a probability by a factor:

Hoặc bạn có thể nhân một xác suất với một thừa số:

>>> pmf.Mult(2, 0.5)

>>> pmf.Prob(2)

0.3

If you modify a Pmf, the result may not be normalized; that is, the probabilities may no longer add up to 1. To check, you can call Total, which returns the sum of the proba‐ bilities:

Nếu bạn sửa đổi Pmf, kết quả có thể không được chuẩn hóa; nghĩa là, các xác suất có thể không còn bằng 1 nữa. Để kiểm tra, bạn có thể gọi Total, hàm trả về tổng của các xác suất:

>>> pmf.Total()

0.9

To renormalize, call Normalize:

Để chuẩn hóa lại, hãy gọi Chuẩn hóa:

>>> pmf.Normalize()

>>> pmf.Total()

1.0

Pmf objects provide a Copy method so you can make and modify a copy without affecting the original.

My notation in this section might seem inconsistent, but there is a system: I use Pmf for the name of the class, pmf for an instance of the class, and PMF for the mathematical concept of a probability mass function.

Các đối tượng Pmf cung cấp một phương thức Sao chép để bạn có thể tạo và sửa đổi một bản sao mà không ảnh hưởng đến bản gốc.

Ký hiệu của tôi trong phần này có vẻ không nhất quán, nhưng có một hệ thống: Tôi sử dụng Pmf cho tên của lớp, pmf cho một thể hiện của lớp và PMF cho khái niệm toán học của hàm khối lượng xác suất.

Plotting PMFs thinkplot provides two ways to plot Pmfs:

Vẽ biểu đồ PMFs thinkplot cung cấp hai cách để vẽ biểu đồ Pmfs:

• To plot a Pmf as a bar graph, you can use thinkplot.Hist. Bar graphs are most useful if the number of values in the Pmf is small.

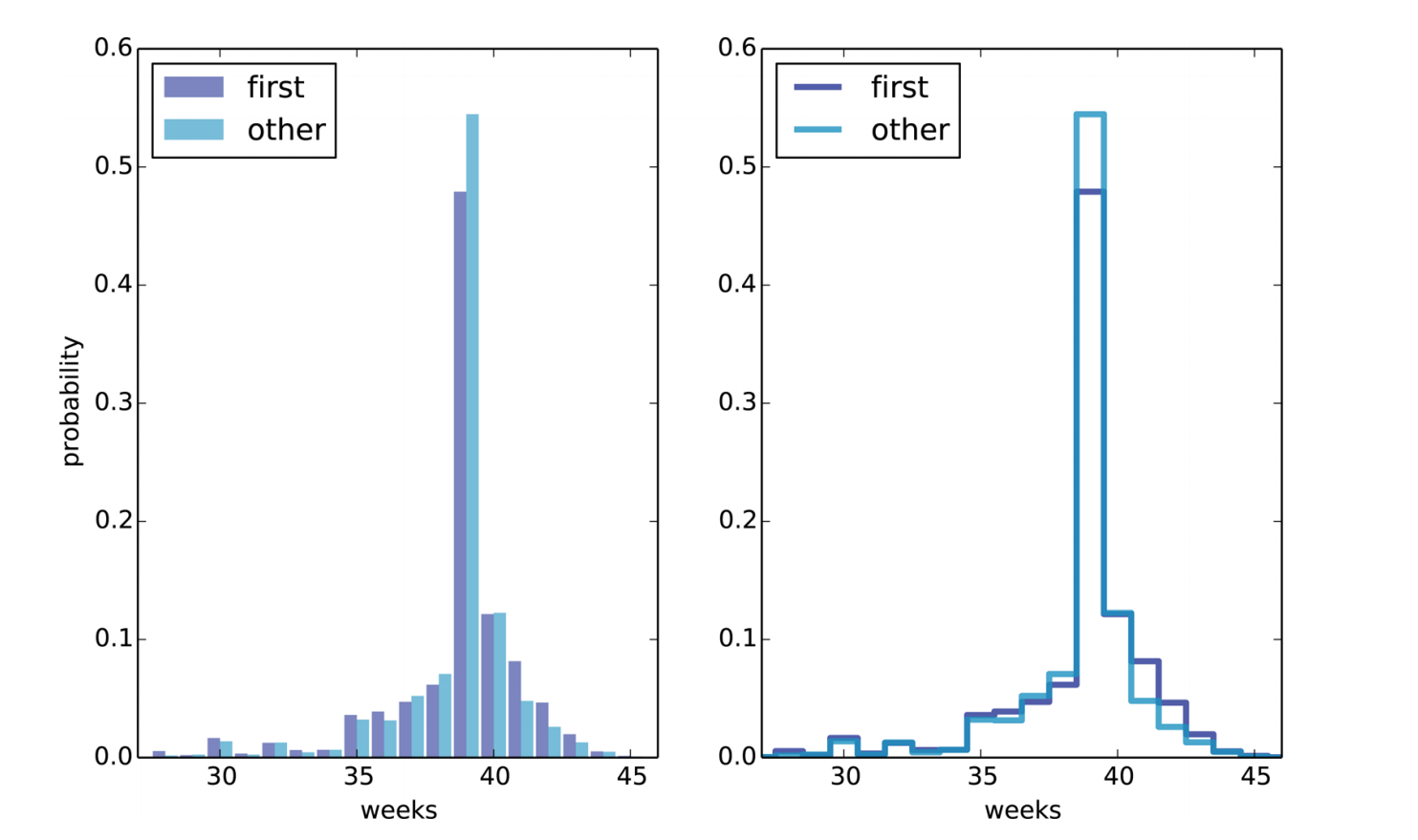
• To plot a Pmf as a step function, you can use thinkplot.Pmf. This option is most useful if there are a large number of values and the Pmf is smooth. This function also works with Hist objects.

• Để vẽ biểu đồ Pmf dưới dạng biểu đồ thanh, bạn có thể sử dụng thinkplot.Hist. Biểu đồ thanh hữu ích nhất nếu số lượng giá trị trong Pmf nhỏ.

• Để vẽ biểu đồ Pmf dưới dạng hàm bước, bạn có thể sử dụng thinkplot.Pmf. Tùy chọn này hữu ích nhất nếu có một số lượng lớn các giá trị và Pmf trơn tru. Chức năng này cũng hoạt động với các đối tượng Hist.

In addition, pyplot provides a function called hist that takes a sequence of values, computes a histogram, and plots it. Since I use Hist objects, I usually don’t use pyplot.hist. Figure 3-1 shows PMFs of pregnancy length for first babies and others using bar graphs (left) and step functions (right).

Ngoài ra, pyplot cung cấp một hàm gọi là hist, hàm này nhận một chuỗi các giá trị, tính toán một biểu đồ tần suất và vẽ đồ thị của nó. Vì tôi sử dụng các đối tượng Hist nên tôi thường không sử dụng pyplot.hist. Hình 3-1 cho thấy các PMF về thời gian mang thai của trẻ sơ sinh đầu lòng và những trẻ khác sử dụng biểu đồ thanh (trái) và hàm bước (phải).



*Figure 3-1. PMF of pregnancy lengths for first babies and others, using bar graphs and step functions.*

*Hình 3-1. PMF về thời gian mang thai của trẻ sơ sinh đầu lòng và những trẻ khác, sử dụng biểu đồ thanh và hàm bước.*

Here’s the code that generates Figure 3-1:

Đây là mã tạo Hình 3-1:

thinkplot.PrePlot(2, cols=2)

thinkplot.Hist(first\_pmf, align='right', width=width)

thinkplot.Hist(other\_pmf, align='left', width=width)

thinkplot.Config(xlabel='weeks', ylabel='probability', axis=[27, 46, 0, 0.6])

thinkplot.PrePlot(2) thinkplot.SubPlot(2)

thinkplot.Pmfs([first\_pmf, other\_pmf])

thinkplot.Show(xlabel='weeks', axis=[27, 46, 0, 0.6])

By plotting the PMF instead of the histogram, we can compare the two distributions without being misled by the difference in sample size. Based on this figure, first babies Plotting PMFs | 29 seem to be less likely than others to arrive on time (week 39) and more likely to be late (weeks 41 and 42).

Bằng cách vẽ biểu đồ PMF thay vì biểu đồ, chúng ta có thể so sánh hai bản phân phối mà không bị nhầm lẫn bởi sự khác biệt về kích thước mẫu. Dựa trên con số này, những đứa trẻ đầu tiên Vẽ sơ đồ PMF | 29 dường như ít đến đúng giờ hơn những người khác (tuần 39) và có nhiều khả năng đến muộn hơn (tuần 41 và 42).

PrePlot takes optional parameters rows and cols to make a grid of figures, in this case one row of two figures. The figure on the left displays the Pmfs using thinkplot.Hist, as we have seen before.

PrePlot lấy tham số tùy chọn các hàng và cột để tạo một lưới các hình, trong trường hợp này là một hàng gồm hai hình. Hình bên trái hiển thị các Pmf sử dụng thinkplot.Hist, như chúng ta đã thấy trước đây.

The second call to PrePlot resets the color generator. Then SubPlot switches to the second figure (on the right) and displays the Pmfs using thinkplot.Pmfs. I used the axis option to ensure that the two figures are on the same axes, which is generally a good idea if you intend to compare two figures.

Cuộc gọi thứ hai tới PrePlot đặt lại bộ tạo màu. Sau đó, SubPlot chuyển sang hình thứ hai (ở bên phải) và hiển thị Pmfs bằng cách sử dụng thinkplot.Pmfs. Tôi đã sử dụng tùy chọn trục để đảm bảo rằng hai hình nằm trên cùng một trục, đây thường là một ý tưởng hay nếu bạn định so sánh hai hình.

**Other Visualizations**

**Hình dung khác**

Histograms and PMFs are useful while you are exploring data and trying to identify patterns and relationships. Once you have an idea what is going on, a good next step is to design a visualization that makes the patterns you have identified as clear as possible.

Biểu đồ và PMF rất hữu ích khi bạn đang khám phá dữ liệu và cố gắng xác định các mẫu và mối quan hệ. Khi bạn có ý tưởng về những gì đang diễn ra, bước tiếp theo tốt là thiết kế một hình ảnh trực quan hóa làm cho các mẫu bạn đã xác định rõ ràng nhất có thể.

In the NSFG data, the biggest differences in the distributions are near the mode. So it makes sense to zoom in on that part of the graph, and transform the data to emphasize differences:

Trong dữ liệu NSFG, sự khác biệt lớn nhất trong các bản phân phối là gần chế độ. Vì vậy, thật hợp lý khi phóng to phần đó của biểu đồ và chuyển đổi dữ liệu để nhấn mạnh sự khác biệt:

weeks = range(35, 46)

diffs = [] for week in weeks:

p1 = first\_pmf.Prob(week)

p2 = other\_pmf.Prob(week)

diff = 100 \* (p1 - p2)

diffs.append(diff)

thinkplot.Bar(weeks, diffs)

In this code, weeks is the range of weeks; diffs is the difference between the two PMFs in percentage points. Figure 3-2 shows the result as a bar chart. This figure makes the pattern clearer: first babies are less likely to be born in week 39, and somewhat more likely to be born in weeks 41 and 42.

Trong mã này, tuần là phạm vi tuần; diffs là sự khác biệt giữa hai PMF tính theo điểm phần trăm. Hình 3-2 hiển thị kết quả dưới dạng biểu đồ thanh. Con số này làm cho mô hình rõ ràng hơn: những đứa trẻ đầu tiên ít có khả năng được sinh ra vào tuần 39 và có nhiều khả năng được sinh ra vào tuần 41 và 42.

For now we should hold this conclusion only tentatively. We used the same dataset to identify an apparent difference and then chose a visualization that makes the difference apparent. We can’t be sure this effect is real; it might be due to random variation. We’ll address this concern later.

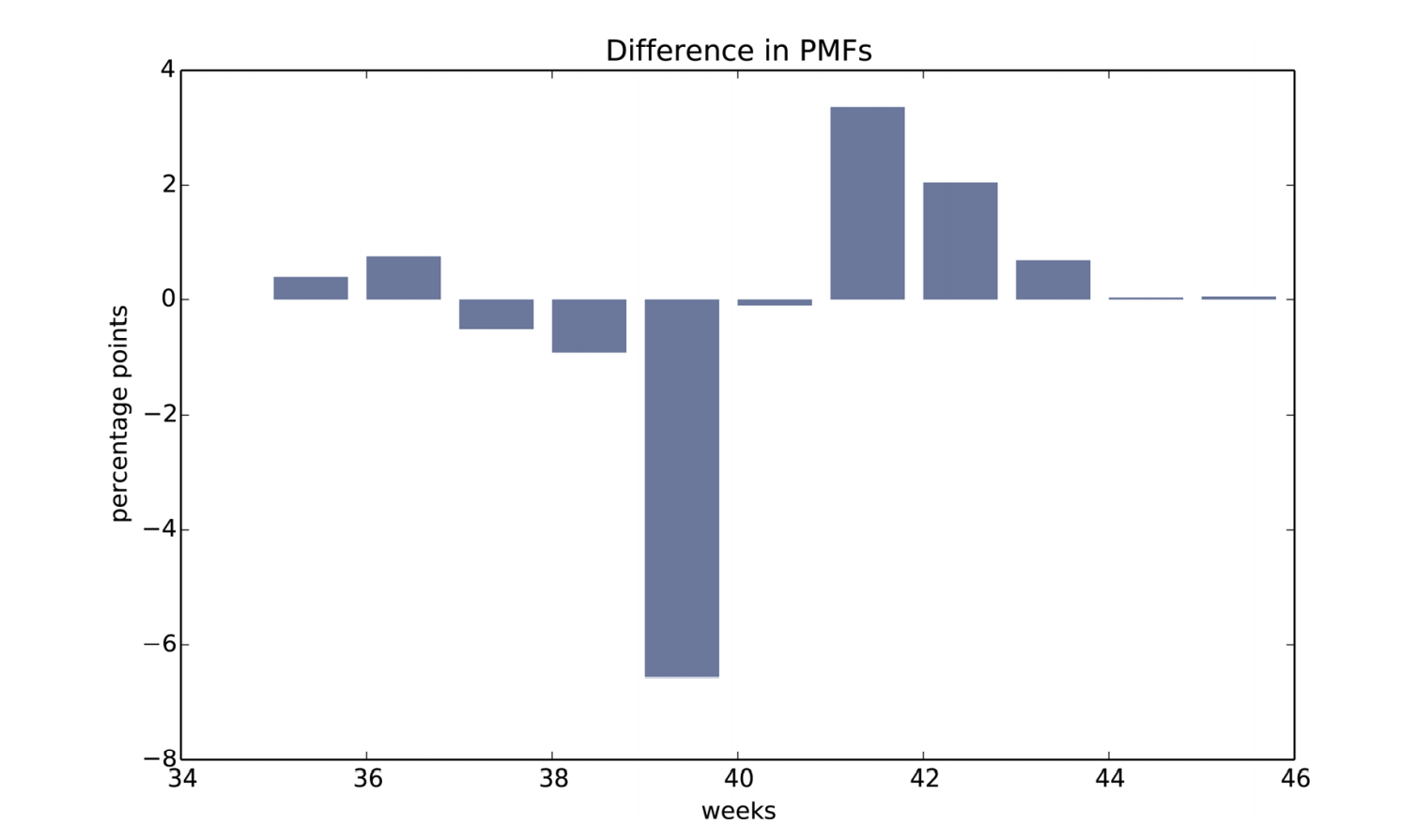
Hiện tại chúng ta chỉ nên giữ kết luận này một cách tạm thời. Chúng tôi đã sử dụng cùng một bộ dữ liệu để xác định sự khác biệt rõ ràng và sau đó chọn một hình ảnh trực quan hóa làm cho sự khác biệt trở nên rõ ràng. Chúng tôi không thể chắc chắn hiệu ứng này là có thật; nó có thể là do sự thay đổi ngẫu nhiên. Chúng tôi sẽ giải quyết mối quan tâm này sau.

**The Class Size Paradox**

**Nghịch lý quy mô lớp học**

Before we go on, I want to demonstrate one kind of computation you can do with Pmf objects; I call this example the “class size paradox.”

Trước khi chúng ta tiếp tục, tôi muốn chứng minh một kiểu tính toán mà bạn có thể thực hiện với các đối tượng Pmf; Tôi gọi ví dụ này là “nghịch lý quy mô lớp học”.



*Figure 3-2. Difference, in percentage points, by week.*

*Hình 3-2. Sự khác biệt, tính bằng điểm phần trăm, theo tuần.*

At many American colleges and universities, the student-to-faculty ratio is about 10:1. But students are often surprised to discover that their average class size is bigger than 10. There are two reasons for the discrepancy:

Tại nhiều trường cao đẳng và đại học Mỹ, tỷ lệ sinh viên-giảng viên là khoảng 10:1. Nhưng sinh viên thường ngạc nhiên khi phát hiện ra rằng quy mô lớp học trung bình của họ lớn hơn 10. Có hai lý do cho sự khác biệt:

• Students typically take 4–5 classes per semester, but professors often teach 1 or 2.

• The number of students who enjoy a small class is small, but the number of students in a large class is (ahem!) large.

• Học sinh thường học 4–5 lớp mỗi học kỳ, nhưng các giáo sư thường dạy 1 hoặc 2 lớp.

• Số học sinh thích lớp nhỏ thì ít, nhưng số học sinh lớp đông thì (ahem!) lớn.

The first effect is obvious, at least once it is pointed out; the second is more subtle. Let’s look at an example. Suppose that a college offers 65 classes in a given semester, with the following distribution of sizes:

Hiệu ứng đầu tiên là rõ ràng, ít nhất là một khi nó được chỉ ra; thứ hai là tinh tế hơn. Hãy xem một ví dụ. Giả sử rằng một trường đại học cung cấp 65 lớp học trong một học kỳ nhất định, với sự phân bổ sĩ số như sau:

size count

5- 9 8

10-14 8

15-19 14

20-24 4

25-29 6

30-34 12

35-39 8

40-44 3

45-49 2

If you ask the Dean for the average class size, he would construct a PMF, compute the mean, and report that the average class size is 23.7. Here’s the code:

Nếu bạn hỏi Hiệu trưởng về sĩ số trung bình của lớp học, ông ấy sẽ xây dựng PMF, tính toán giá trị trung bình và báo cáo rằng sĩ số trung bình của lớp học là 23,7. Đây là mã:

d = { 7: 8, 12: 8, 17: 14, 22: 4, 27: 6, 32: 12, 37: 8, 42: 3, 47: 2 }

pmf = thinkstats2.Pmf(d, label='actual')

print('mean', pmf.Mean())

But if you survey a group of students, ask them how many students are in their classes, and compute the mean, you would think the average class size was bigger. Let’s see how much bigger.

First, I compute the distribution as observed by students, where the probability asso‐ ciated with each class size is “biased” by the number of students in the class.

Nhưng nếu bạn khảo sát một nhóm sinh viên, hỏi họ có bao nhiêu sinh viên trong lớp học của họ và tính giá trị trung bình, bạn sẽ nghĩ rằng sĩ số trung bình của lớp học lớn hơn. Hãy xem lớn hơn bao nhiêu.

Đầu tiên, tôi tính toán phân phối theo quan sát của sinh viên, trong đó xác suất liên quan đến quy mô mỗi lớp học bị “sai lệch” bởi số lượng sinh viên trong lớp.

def BiasPmf(pmf, label):

new\_pmf = pmf.Copy(label=label)

for x, p in pmf.Items():

new\_pmf.Mult(x, x)

new\_pmf.Normalize()

return new\_pmf

For each class size, x, we multiply the probability by x, the number of students who observe that class size. The result is a new Pmf that represents the biased distribution.

Now we can plot the actual and observed distributions:

Đối với mỗi quy mô lớp học, x, chúng tôi nhân xác suất với x, số học sinh quan sát quy mô lớp học đó. Kết quả là một Pmf mới đại diện cho phân phối sai lệch.

Bây giờ chúng ta có thể vẽ biểu đồ phân phối thực tế và được quan sát:

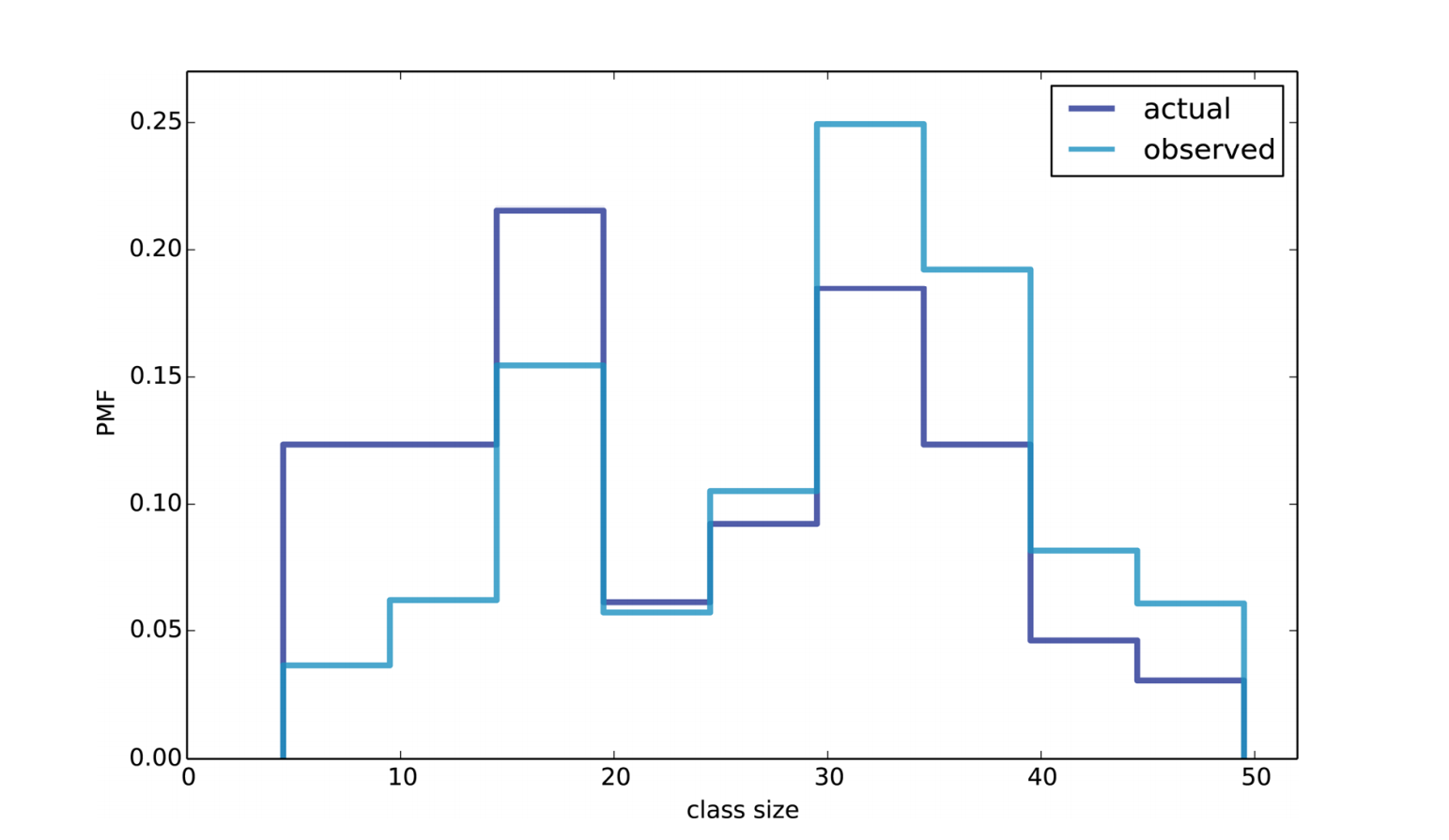
biased\_pmf = BiasPmf(pmf, label='observed')

thinkplot.PrePlot(2) thinkplot.Pmfs([pmf, biased\_pmf])

thinkplot.Show(xlabel='class size', ylabel='PMF')

Figure 3-3 shows the result. In the biased distribution there are fewer small classes and more large ones. The mean of the biased distribution is 29.1, almost 25% higher than the actual mean.

Hình 3-3 cho thấy kết quả. Trong phân phối thiên vị, có ít lớp nhỏ hơn và nhiều lớp lớn hơn. Giá trị trung bình của phân phối sai lệch là 29,1, cao hơn gần 25% so với giá trị trung bình thực tế.



*Figure 3-3. Distribution of class sizes, actual and as observed by students.*

*Hình 3-3. Phân bố sĩ số lớp học, thực tế và theo quan sát của học sinh.*

It is also possible to invert this operation. Suppose you want to find the distribution of class sizes at a college, but you can’t get reliable data from the Dean. An alternative is to choose a random sample of students and ask how many students are in their classes.

The result would be biased for the reasons we’ve just seen, but you can use it to estimate the actual distribution. Here’s the function that unbiases a Pmf:

Cũng có thể đảo ngược hoạt động này. Giả sử bạn muốn tìm sự phân bố quy mô lớp học tại một trường đại học, nhưng bạn không thể lấy dữ liệu đáng tin cậy từ Trưởng khoa. Một cách khác là chọn một mẫu học sinh ngẫu nhiên và hỏi có bao nhiêu học sinh trong lớp của họ.

Kết quả sẽ bị sai lệch vì những lý do mà chúng ta vừa thấy, nhưng bạn có thể sử dụng nó để ước tính phân phối thực tế. Đây là chức năng không thiên vị một Pmf:

def UnbiasPmf(pmf, label):

new\_pmf = pmf.Copy(label=label)

for x, p in pmf.Items():

new\_pmf.Mult(x, 1.0/x)

new\_pmf.Normalize()

return new\_pmf

It’s similar to BiasPmf; the only difference is that it divides each probability by x instead of multiplying.

**DataFrame Indexing**

**Lập chỉ mục khung dữ liệu**

In “DataFrames” on page 4 we read a pandas DataFrame and used it to select and modify data columns. Now let’s look at row selection. To start, I create a NumPy array of random numbers and use it to initialize a DataFrame:

Trong “DataFrames” ở trang 4, chúng tôi đọc một DataFrame của gấu trúc và sử dụng nó để chọn và sửa đổi các cột dữ liệu. Bây giờ hãy xem xét lựa chọn hàng. Để bắt đầu, tôi tạo một mảng NumPy gồm các số ngẫu nhiên và sử dụng nó để khởi tạo DataFrame:

>>> import numpy as np

>>> import pandas >>>

array = np.random.randn(4, 2)

>>> df = pandas.DataFrame(array)

>>> df

0 1

0 -0.143510 0.616050

1 -1.489647 0.300774

2 -0.074350 0.039621

3 -1.369968 0.545897

By default, the rows and columns are numbered starting at zero, but you can provide column names:

Theo mặc định, các hàng và cột được đánh số bắt đầu từ 0, nhưng bạn có thể cung cấp tên cột:

>>> columns = ['A', 'B']

>>> df = pandas.DataFrame(array, columns=columns)

>>> df

A B

0 -0.143510 0.616050

1 -1.489647 0.300774

2 -0.074350 0.039621

3 -1.369968 0.545897

You can also provide row names. The set of row names is called the index; the row names themselves are called labels.

Bạn cũng có thể cung cấp tên hàng. Tập hợp các tên hàng được gọi là chỉ số; bản thân các tên hàng được gọi là nhãn.

>>> index = ['a', 'b', 'c', 'd']

>>> df = pandas.DataFrame(array, columns=columns, index=index)

>>> df

A B

a -0.143510 0.616050

b -1.489647 0.300774

c -0.074350 0.039621

d -1.369968 0.545897

As we saw in the previous chapter, simple indexing selects a column, returning a Series:

Như chúng ta đã thấy trong chương trước, lập chỉ mục đơn giản sẽ chọn một cột, trả về một Chuỗi:

>>> df['A']

a -0.143510

b -1.489647

c -0.074350

d -1.369968

Name: A, dtype: float64

To select a row by label, you can use the loc attribute, which returns a Series:

Để chọn một hàng theo nhãn, bạn có thể sử dụng thuộc tính loc, thuộc tính này trả về một Sê-ri:

>>> df.loc['a']

A -0.14351

B 0.61605

Name: a, dtype: float64

If you know the integer position of a row, rather than its label, you can use the iloc attribute, which also returns a Series.

Nếu bạn biết vị trí số nguyên của một hàng, chứ không phải nhãn của nó, bạn có thể sử dụng thuộc tính iloc, thuộc tính này cũng trả về một Sê-ri.

>>> df.iloc[0]

A -0.14351

B 0.61605

Name: a, dtype: float64

loc can also take a list of labels; in that case, the result is a DataFrame.

loc cũng có thể lấy một danh sách các nhãn; trong trường hợp đó, kết quả là một DataFrame.

>>> indices = ['a', 'c']

>>> df.loc[indices]

A B

a -0.14351 0.616050

c -0.07435 0.039621

Finally, you can use a slice to select a range of rows by label:

Cuối cùng, bạn có thể sử dụng một lát cắt để chọn một loạt các hàng theo nhãn:

>>> df['a':'c']

A B

a -0.143510 0.616050

b -1.489647 0.300774

c -0.074350 0.039621

Or by integer position:

>>> df[0:2]

A B

a -0.143510 0.616050

b -1.489647 0.300774

The result in either case is a DataFrame, but notice that the first result includes the end of the slice; the second doesn’t. My advice: if your rows have labels that are not simple integers, use the labels consistently and avoid using integer positions.

Kết quả trong cả hai trường hợp là một DataFrame, nhưng lưu ý rằng kết quả đầu tiên bao gồm phần cuối của lát cắt; cái thứ hai thì không. Lời khuyên của tôi: nếu các hàng của bạn có nhãn không phải là số nguyên đơn giản, hãy sử dụng nhãn một cách nhất quán và tránh sử dụng vị trí số nguyên.

**Exercises**

Solutions to these exercises are in chap03soln.ipynb and chap03soln.py

Lời giải cho các bài tập này có trong chap03soln.ipynb và chap03soln.py

Exercise 3-1.

Something like the class size paradox appears if you survey children and ask how many children are in their family. Families with many children are more likely to appear in your sample, and families with no children have no chance to be in the sample.

Use the NSFG respondent variable NUMKDHH to construct the actual distribution for the number of children under 18 in the household.

Now compute the biased distribution we would see if we surveyed the children and asked them how many children under 18 (including themselves) are in their household.

Plot the actual and biased distributions, and compute their means. As a starting place, you can use chap03ex.ipynb.

Bài tập 3-1.

Điều gì đó giống như nghịch lý về quy mô lớp học sẽ xuất hiện nếu bạn khảo sát trẻ em và hỏi gia đình chúng có bao nhiêu trẻ em. Các gia đình có nhiều con có nhiều khả năng xuất hiện trong mẫu của bạn hơn và các gia đình không có con sẽ không có cơ hội xuất hiện trong mẫu.

Sử dụng biến số người trả lời NSFG NUMKDHH để xây dựng phân phối thực tế cho số trẻ em dưới 18 tuổi trong hộ gia đình.

Bây giờ hãy tính phân phối sai lệch mà chúng ta sẽ thấy nếu chúng ta khảo sát trẻ em và hỏi chúng có bao nhiêu trẻ em dưới 18 tuổi (bao gồm cả chúng) trong gia đình chúng.

Vẽ sơ đồ phân phối thực tế và sai lệch, và tính toán phương tiện của chúng. Để bắt đầu, bạn có thể sử dụng chap03ex.ipynb.

Exercise 3-2.

In “Summarizing Distributions” on page 22 we computed the mean of a sample by adding up the elements and dividing by n. If you are given a PMF, you can still compute the mean, but the process is slightly different:

x¯ = ∑ i pi xi

where the xi are the unique values in the PMF and pi = PMF (xi). Similarly, you can compute variance like this:

S 2 = ∑ i pi (xi - x¯ ) 2

Write functions called PmfMean and PmfVar that take a Pmf object and compute the mean and variance. To test these methods, check that they are consistent with the methods Mean and Var provided by Pmf.

Bài tập 3-2.

Trong “Tóm tắt phân phối” trên trang 22, chúng tôi đã tính giá trị trung bình của một mẫu bằng cách cộng các phần tử và chia cho n. Nếu bạn được cung cấp PMF, bạn vẫn có thể tính giá trị trung bình, nhưng quá trình này hơi khác một chút:

x¯ = ∑ i pi xi

trong đó xi là các giá trị duy nhất trong PMF và pi = PMF (xi). Tương tự, bạn có thể tính phương sai như sau:

S 2 = ∑ i pi (xi - x¯ ) 2

Viết các hàm có tên là PmfMean và PmfVar lấy một đối tượng Pmf và tính giá trị trung bình và phương sai. Để kiểm tra các phương thức này, hãy kiểm tra xem chúng có phù hợp với các phương thức Mean và Var do Pmf cung cấp hay không.

Exercise 3-3.

I started with the question, “Are first babies more likely to be late?” To address it, I computed the difference in means between groups of babies, but I ignored the possibility that there might be a difference between first babies and others for the same woman.

To address this version of the question, select respondents who have at least two babies and compute pairwise differences. Does this formulation of the question yield a different result?

Hint: use nsfg.MakePregMap.

Bài tập 3-3.

Tôi bắt đầu với câu hỏi, “Có phải những đứa trẻ đầu lòng thường sinh muộn hơn không?” Để giải quyết vấn đề này, tôi đã tính toán sự khác biệt về phương tiện giữa các nhóm trẻ sơ sinh, nhưng tôi đã bỏ qua khả năng có thể có sự khác biệt giữa những đứa trẻ đầu lòng và những đứa trẻ khác đối với cùng một phụ nữ.

Để giải quyết phiên bản câu hỏi này, hãy chọn những người trả lời có ít nhất hai con và tính toán sự khác biệt theo cặp. Công thức này của câu hỏi có mang lại một kết quả khác không?

Gợi ý: sử dụng nsfg.MakePregMap.

Exercise 3-4.

In most foot races, everyone starts at the same time. If you are a fast runner, you usually pass a lot of people at the beginning of the race, but after a few miles everyone around you is going at the same speed.

When I ran a long-distance (209 miles) relay race for the first time, I noticed an odd phenomenon: when I overtook another runner, I was usually much faster, and when another runner overtook me, he was usually much faster.

Probability Mass Functions At first I thought that the distribution of speeds might be bimodal; that is, there were many slow runners and many fast runners, but few at my speed.

Then I realized that I was the victim of a bias similar to the effect of class size. The race was unusual in two ways: it used a staggered start, so teams started at different times; also, many teams included runners at different levels of ability.

As a result, runners were spread out along the course with little relationship between speed and location. When I joined the race, the runners near me were (pretty much) a random sample of the runners in the race.

So where does the bias come from? During my time on the course, the chance of over‐ taking a runner, or being overtaken, is proportional to the difference in our speeds. I am more likely to catch a slow runner, and more likely to be caught by a fast runner. But runners at the same speed are unlikely to see each other.

Write a function called ObservedPmf that takes a Pmf representing the actual distribu‐ tion of runners’ speeds, and the speed of a running observer, and returns a new Pmf representing the distribution of runners’ speeds as seen by the observer.

To test your function, you can use relay.py, which reads the results from the James Joyce Ramble 10K in Dedham MA and converts the pace of each runner to mph.

Compute the distribution of speeds you would observe if you ran a relay race at 7.5 mph with this group of runners. A solution to this exercise is in relay\_soln.py.

Bài tập 3-4.

Trong hầu hết các cuộc đua chân, mọi người bắt đầu cùng một lúc. Nếu bạn là một người chạy nhanh, bạn thường vượt qua rất nhiều người khi bắt đầu cuộc đua, nhưng sau một vài dặm, mọi người xung quanh bạn đều chạy với tốc độ như nhau.

Khi tôi chạy một cuộc chạy tiếp sức đường dài (209 dặm) lần đầu tiên, tôi nhận thấy một hiện tượng kỳ lạ: khi tôi vượt qua một vận động viên khác, tôi thường nhanh hơn nhiều, và khi một vận động viên khác vượt qua tôi, anh ta thường nhanh hơn nhiều.

Hàm khối lượng xác suất Lúc đầu, tôi nghĩ rằng sự phân bố tốc độ có thể là hai hàm; nghĩa là có nhiều người chạy chậm và nhiều người chạy nhanh, nhưng ít người chạy bằng tốc độ của tôi.

Sau đó, tôi nhận ra rằng mình là nạn nhân của một thành kiến tương tự như tác động của quy mô lớp học. Cuộc đua không bình thường theo hai cách: nó sử dụng xuất phát so le, vì vậy các đội xuất phát vào những thời điểm khác nhau; Ngoài ra, nhiều đội bao gồm các vận động viên chạy ở các cấp độ khả năng khác nhau.

Kết quả là, các vận động viên được dàn trải dọc theo đường chạy với rất ít mối quan hệ giữa tốc độ và vị trí. Khi tôi tham gia cuộc đua, những người chạy gần tôi (khá nhiều) là mẫu ngẫu nhiên của những người chạy trong cuộc đua.

Vậy sự thiên vị đến từ đâu? Trong thời gian tôi tham gia cuộc đua, khả năng vượt qua người chạy hoặc bị vượt qua tỷ lệ thuận với sự khác biệt về tốc độ của chúng tôi. Tôi có nhiều khả năng bị bắt bởi một người chạy chậm, và nhiều khả năng bị bắt bởi một người chạy nhanh. Nhưng những người chạy cùng tốc độ khó có thể nhìn thấy nhau.

Viết một hàm có tên là ObservedPmf lấy một Pmf biểu thị sự phân bố tốc độ thực tế của người chạy và tốc độ của một người quan sát đang chạy, đồng thời trả về một Pmf mới biểu thị sự phân bố tốc độ của người chạy mà người quan sát nhìn thấy.

Để kiểm tra chức năng của mình, bạn có thể sử dụng relay.py, đọc kết quả từ James Joyce Ramble 10K ở Dedham MA và chuyển đổi tốc độ của mỗi người chạy thành mph.

Tính toán sự phân bố tốc độ mà bạn sẽ quan sát được nếu bạn chạy một cuộc chạy tiếp sức với vận tốc 17,5 km/h với nhóm vận động viên này. Giải pháp cho bài tập này có trong relay\_soln.py.

**Glossary**

Probability mass function (PMF)

a representation of a distribution as a function that maps from values to probabil‐ ities.

probability

A frequency expressed as a fraction of the sample size.

normalization

The process of dividing a frequency by a sample size to get a probability.

index

In a pandas DataFrame, the index is a special column that contains the row labels.

Bảng chú giải

Hàm khối lượng xác suất (PMF)

một biểu diễn của một phân phối dưới dạng một hàm ánh xạ từ các giá trị đến xác suất.

xác suất

Một tần suất được biểu thị dưới dạng một phần nhỏ của kích thước mẫu.

bình thường hóa

Quá trình chia tần số cho một cỡ mẫu để có xác suất.

mục lục

Trong DataFrame của gấu trúc, chỉ mục là một cột đặc biệt chứa các nhãn hàng.