BỘ GIÁO DỤC VÀ ĐÀO TẠO

**TRƯỜNG ĐẠI HỌC SƯ PHẠM KỸ THUẬT HƯNG YÊN**



**TIỂU LUẬN**

**CƠ SỞ TOÁN CHO HỌC MÁY**

**Tên tiểu luận: Chương 6 – Hàm mật độ xác xuất**

.....

Giảng viên HD: **TS. Nguyễn Văn Hậu**

Học viên thực hiện: **Đào Văn Hùng**

Lớp: **H01222**

*Hưng Yên, 6/2023*

Chương 6

**Hàm mật độ xác xuất**

Mã cho chương này nằm trong density.py Để biết thông tin về cách tải xuống và làm việc với mã này, hãy xem “Using the Code” trên trang xi.

**PDF (hàm mật độ xác suất)**

Đạo hàm của một CDF được gọi là **hàm mật độ xác suất**, hay PDF (probability density function). Chẳng hạn, PDF của một phân bố lũy thừa là

|  |
| --- |
| *PDFexpo*(*x*) = λ *e*−λ*x* |

PDF của một phân bố chuẩn là

Việc lượng giá PDF cho một giá trị cụ thể của *x* thường không mấy hữu ích. Kết quả không phải là một xác suất mà là một *mật độ* xác suất.

Trong vật lý, mật độ là khối lượng có trong một đơn vị thể tích. Để có được khối lượng, bạn bải đem nhân mật độ với thể tích, hay nếu mật độ không phải hằng số, thì cần lấy tích phân trên thể tích.

Tương tự, mật độ xác suất là số đo xác suất trên mỗi đơn vị của *x*. Để có được khối xác suất, bạn phải lấy tích phân theo *x*.

thinkstats2 cung cấp một lớp có tên Pdf để biểu diễn cho hàm mật độ xác suất. Mỗi đối tượng Pdf cung cấp những phương thức sau đây:

* Density, nhận vào một giá trị, x, rồi trả lại mật độ của phân bố tại x.
* Render, tính ra mật độ tại một tập hợp rời rạc gồm các giá trị, và trả lại một cặp gồm hai dãy: các giá trị đã sắp xếp, xs, cùng mật độ xác suất của chúng, ds.
* MakePmf, tính ra Density tại một tập hợp rời rạc gồm các giá trị, và trả lại một Pmf đã chuẩn hóa để xấp xỉ Pdf đó.
* GetLinspace, nhận vào tập hợp điểm mặc định được Render và MakePmf dùng đến.

Pdf là một lớp mẹ trừu tượng, nghĩa là bạn không nên khởi tạo nó; nói cách khác bạn không thể tạo một đối tượng Pdf. Thay vì vậy, bạn cần định nghĩa một lớp con thừa hưởng từ Pdf và cung cấp các định nghĩa về Density và GetLinspace. Pdf thì cung cấp Render và MakePmf.

Chẳng hạn, thinkstats2 cung cấp một lớp có tên NormalPdf để ước lượng hàm mật độ chuẩn.

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  1 | class NormalPdf(Pdf):  def \_\_init\_\_(self, mu=0, sigma=1, label=''):  self.mu = mu  self.sigma = sigma  self.label = label  def Density(self, xs):  return scipy.stats.norm.pdf(xs, self.mu, self.sigma)  def GetLinspace(self):  low, high = self.mu-3\*self.sigma, self.mu+3\*self.sigma  return np.linspace(low, high, 101) |

Đối tượng NormalPdf có chứa các tham số mu và sigma. Density sử dụng scipy.stats.norm, vốn là một đối tượng biểu diễn một phân bố chuẩn và cung cấp cdf, pdf, cùng những phương thức khác (xem Mục [5.2](https://greenteapress.com/thinkstats2/html/thinkstats2006.html#normal)).

Ví dụ sau đây tạo một NormalPdf với trị trung bình và độ lệch chuẩn của chiều cao người trưởng thành, tính bằng cm, từ số liệu BRFSS (xem Mục [5.4](https://greenteapress.com/thinkstats2/html/thinkstats2006.html#brfss)). Sau đó, nó tính mật độ phân bố tại vị trí cách trị trung bình một khoảng bằng một độ lệch chuẩn.

>>> mean, var = 163, 52.8

>>> std = math.sqrt(var)

>>> pdf = thinkstats2.NormalPdf(mean, std)

>>> pdf.Density(mean + std)

0.0333001

Kết quả bằng khoảng 0.03, đơn vị khối xác suất trên cm. Một lần nữa, mật độ xác suất bản thân nó thì không mang nhiều ý nghĩa. Nhưng nếu ta vẽ hàm Pdf, ta có thể thấy hình dạng của phân bố này:

>>> thinkplot.Pdf(pdf, label='normal')

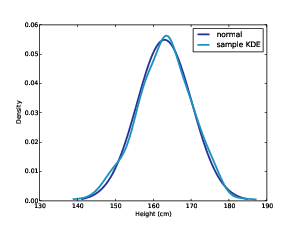
>>> thinkplot.Show()

thinkplot.Pdf vẽ nên Pdf dưới dạng hàm trơn, khác với thinkplot.Pmf, vốn vẽ ra Pmf dạng hàm bậc thang. Hình [6.1](https://greenteapress.com/thinkstats2/html/thinkstats2007.html#pdf_example) cho thấy kết quả, cũng như PDF ước tính từ một mẫu, mà ta sẽ tính đến trong mục tiếp sau.

Bạn có thể dùng MakePmf để xấp xỉ Pdf:

>>> pmf = pdf.MakePmf()

Theo mặc định, kết quả Pmf có chứa 101 điểm cách đều nhau từ mu - 3\*sigma tới mu + 3\*sigma. Tuỳ chọn cho phép MakePmf và Render nhận các đối số từ khoá low, high, và n.



|  |
| --- |
| Hình 6.1: Một PDF chuẩn để mô hình hoá chiều cao nữ giới Mỹ trưởng thành, và ước tính mật độ nhân cho một mẫu với *n*=500. |

**6.2  Ước tính mật độ nhân**

**Ước tính mật độ nhân** (kernel density estimation, KDE) là một thuật toán nhận vào một mẫu và tìm hàm PDF trơn phù hợp để khớp với số liệu. Bạn có thể đọc thêm chi tiết tại <http://en.wikipedia.org/wiki/Kernel_density_estimation>.

scipy cung cấp mã lệnh thực hiện KDE và thinkstats2 cung cấp một lớp có tên EstimatedPdf để sử dụng nó:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | class EstimatedPdf(Pdf):    def \_\_init\_\_(self, sample):   self.kde = scipy.stats.gaussian\_kde(sample)  def Density(self, xs):  return self.kde.evaluate(xs) |

\_\_init\_\_ nhận vào một mẫu rồi tính ước lượng mật độ nhân. Kết quả là một đối tượng gaussian\_kde cung cấp phương thức evaluate.

Density nhận vào một giá trị hoặc một dãy, gọi gaussian\_kde.evaluate, và rồi trả về mật độ tính được. Chữ “Gaussian” xuất hiện trong cái tên trên vì nó dùng một bộ lọc dựa theo phân bố chuẩn (phân bố Gauss) để làm trơn KDE này.

Sau đây là một ví dụ phát sinh ra một mẫu từ một phân bố chuẩn rồi tạo một EstimatedPdf để khớp với phân bố đó:

>>> sample = [random.gauss(mean, std) for i in range(500)]

>>> sample\_pdf = thinkstats2.EstimatedPdf(sample)

>>> thinkplot.Pdf(sample\_pdf, label='sample KDE')

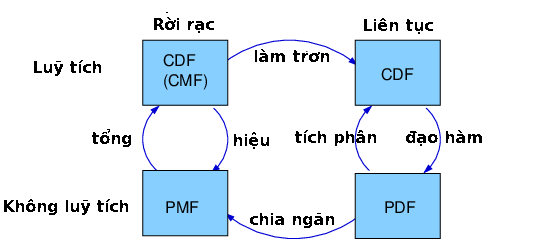
sample là một danh sách gồm 500 chiều cao ngẫu nhiên. sample\_pdf là một đối tượng Pdf có chứa KDE ước tính từ mẫu.

Hình [6.1](https://greenteapress.com/thinkstats2/html/thinkstats2007.html#pdf_example) cho thấy hàm mật độ chuẩn và một KDE dựa trên mẫu gồm 500 chiều cao ngẫu nhiên. Ước tính khớp tốt với phân bố gốc.

Việc ước tính một hàm mật độ bằng KDE hữu dụng với một số mục đích sau:

* *Trực quan hóa:* Trong giai đoạn khám phá của một dự án, các CDF thường là dạng hiển thị trực quan tốt nhất cho một phân bố. Sau khi nhìn vào một CDF, bạn có thể quyết định liệu một PDF ước tính được có là một mô hình phù hợp cho phân bố đó không. Nếu được, nó có thể là lựa chọn hợp lý hơn để biểu diễn phân bố với những khán giả không quen với các CDF.
* *Nội suy:* Một PDF được ước tính là một cách để đi từ một mẫu tới một mô hình cho tổng thể. Nếu bạn có lý do để tin rằng phân bố của tổng thể là một hàm trơn, thì bạn có thể dùng KDE để nội suy mật độ cho những giá trị không xuất hiện trong mẫu.
* *Mô phỏng:* Các mô phỏng thường được dựa trên phân bố của một mẫu. Nếu cỡ mẫu nhỏ, thì có thể nên làm trơn phân bố mẫu bằng KDE; cách này cho phép mô phỏng khám phá được thêm những kết quả khả dĩ, thay vì chỉ lặp lại số liệu đã quan sát được.

**6.3  Sơ đồ phân bố**

[](https://quangchien.files.wordpress.com/2012/01/distribution_functions_vn.png)

|  |
| --- |
| Hình 6.2: Một sơ đồ liên hệ giữa các dạng biểu diễn của hàm phân bố. |

Đến lúc này ta đã gặp các PMF, CDF và PDF; ta hãy cùng dành một phút để ôn lại. Hình  [6.2](https://greenteapress.com/thinkstats2/html/thinkstats2007.html#dist_framework) cho thấy cách liên hệ giữa các hàm này. Chúng ta bắt đầu từ PMF, vốn biểu diễn cho tần suất trong một tập hợp các giá trị rời rạc. Để thu được CDF từ PMF, ta đi tính tổng lũy tích. Để tính được PMF từ CDF, bạn phải tính các hiệu số giữa các xác suất lũy tích. Ta sẽ thấy cách thực hiện các thao tác này trong một vài mục tiếp theo.

Một PDF là đạo hàm của CDF liên tục; hoặc theo cách nói tương đương, CDF là tích phân của PDF. Nhưng cần nhớ rằng một PDF cho ánh xạ từ giá trị đến mật độ xác suất; để thu được xác suất, bạn phải tính tính phân.

Để đi từ phân bố rời rạc đến liên tục, bạn có thể thực hiện một số phương pháp làm trơn. Một cách làm trơn là giả sử rằng số liệu thu được từ một phân bố liên tục có công thức giải tích (như phân bố lũy thừa hoặc chuẩn) và ước tính các tham số của phân bố đó. Một lựa chọn khác là ước lượng mật độ nhân.

Đối nghịch với việc làm trơn là **rời rạc hóa**, hoặc lượng tử hóa. Nếu bạn ước tính PDF tại các điểm rời rạc, bạn có thể phát sinh ra một PMF để xấp xỉ được PDF đó. Bạn có thể xấp xỉ tốt hơn bằng cách dùng tích phân số trị.

Để phân biệt giữa các CDF và rời rạc, có lễ tốt hơn là gọi CDF rời rạc là một “hàm khối lũy tích”, dù rằng theo tôi biết thì chưa có ai dùng thuật ngữ đó cả.

**6.4  Lập trình Hist**

Đến đây bạn đã biết cách dùng những kiểu dữ liệu cơ bản được cung cấp bởi thinkstats2, đó là Hist, Pmf, Cdf, và Pdf. Trong vài mục tới sẽ trình bày chi tiết cách lập trình chúng. Phần nội dung này có thể giúp bạn sử dụng các lớp này một cách hiệu quả hơn, song cũng không phải thiết yếu.

Hist và Pmf kế thừa từ một lớp mẹ có tên \_DictWrapper. Dấu gạch phía trước cho thấy lớp này là “nội bộ”; nghĩa là các mã lệnh trong mô đun khác không nên dùng đến nó. Cái tên này cho biết công dụng của lớp này là gì: một lớp bọc từ điển. Thuộc tính chính của nó là d, một từ điển để ánh xạ từ giá trị đến các tần suất tương ứng của chúng.

Các giá trị này có thể là bất kì kiểu dữ liệu nào hash được [hiểu nôm na đó là những kiểu dữ liệu đơn giản trong Python, bao gồm int, float, str, tuple, NoneType; mà ta có thể dễ dàng so sánh các biến]. Các tần suất nên chọn là số nguyên, song cũng có thể là bất kì kiểu số nào.

\_DictWrapper có chứa các phương thức thích hợp cho cả Hist và Pmf, bao gồm \_\_init\_\_, Values, Items và Render. Nó cũng cung cấp các phương thức chỉnh sửa là Set, Incr, Mult, là Remove. Những phương thức này tất cả đều được lập trình với các phép toán từ điển. Chẳng hạn:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10 | # class \_DictWrapper      def Incr(self, x, term=1):          self.d[x] = self.d.get(x, 0) + term  def Mult(self, x, factor):  self.d[x] = self.d.get(x, 0) \* factor    def Remove(self, x):  del self.d[x] |

Hist còn cung cấp Freq, để tra tìm tần suất của một giá trị cho trước.

Vì các toán tử và phương thức Hist đều dựa trên từ điển nên những phương thức này đều là phép toán có thời gian hằng số; nghĩa là thời gian thực thi chúng thì không tăng lên khi Hist trở nên lớn hơn.

**6.5  Lập trình Pmf**

Pmf và Hist hầu như là một, ngoại trừ Pmf ánh xạ các giá trị tới những xác suất có dạng số phẩy động, thay vì các tần suất số nguyên. Nếu tổng của những xác suất này bằng 1, ta nói rằng Pmf được chuẩn hóa.

Pmf cung cấp Normalize, vốn tính tổng các xác suất và đem chia cho một hệ số:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  1  1 | # class Pmf  def Normalize(self, fraction=1.0):  total = self.Total()  if total == 0.0:  raise ValueError('Total probability is zero.')  factor = float(fraction) / total  for x in self.d:  self.d[x] \*= factor  return total |

fraction quyết định tổng các xác suất sau khi chuẩn hóa; giá trị mặc định bằng 1. Nếu tổng xác suất bằng 0 thì Pmf không thể chuẩn hóa được, khi đó Normalize báo lỗi ValueError.

Hist và Pmf có cùng phương thức khởi tạo (constructor). Nó có thể nhận một tham số có dạng một từ điển (dict), một Hist, một Pmf hay một Cdf, một Series của pandas, một danh sách các cặp (trị số, tần suất), hay một dãy các giá trị.

Nếu bạn khởi tạo một Pmf, kết quả được chuẩn hóa. Nếu bạn khởi tạo một Hist, nó sẽ không được chuẩn hóa. Để tạo nên một Pmf không chuẩn hóa, bạn cần tạo một Pmf rỗng rồi chỉnh sửa nó. Các hàm điều chỉnh tham biến trong Pmf thì đều không chuẩn hóa lại Pmf.

**6.6  Lập trình Cdf**

Một hàm CDF ánh xạ từ giá trị tới các xác suất lũy tích, vì vậy tôi đã có thể lập trình Cdf như một \_DictWrapper. Song các giá trị trong CDF được xếp thứ tự còn các giá trị trong \_DictWrapper thì không. Ngoài ra, cũng nên tính nghịch đảo CDF; đó là ánh xạ từ xác suất lũy tích tới giá trị. Do vậy cách lập trình của tôi có hai danh sách sắp xếp. Bằng cách này tôi có thể dùng phép tìm kiếm nhị phân để tra tìm xuôi hoặc ngược trong thời gian cấp độ logarit.

Phương thức khởi tạo Cdf có thể nhận vào tham biến là một dãy các giá trị hoặc một Series kiểu pandas, một từ điển ánh xạ từ giá trị tới xác suất, một dãy các cặp (trị số, tần suất), một Hist, một Pmf hay một Cdf. Hoặc nếu nó được cung cấp hai tham số thì sẽ coi đó là một dãy các giá trị đã sắp xếp và dãy các xác suất lũy tích tương ứng.

Cho trước một dãy, một Series pandas, hoặc một từ điển, phương thức khởi tạo sẽ lập nên một Hist. Sau đó, nó dùng Hist này để khởi tạo các thuộc tính:

|  |  |
| --- | --- |
| 1  2  3 | self.xs, freqs = zip(\*sorted(dw.Items()))  self.ps = np.cumsum(freqs, dtype=np.float)  self.ps /= self.ps[-1] |

xs là danh sách các giá trị được sắp xếp; freqs là danh sách các tần suất tương ứng. np.cumsum tính ra tổng lũy tích các tần suất. Bằng cách chia cho tổng tần suất, ta được xác suất lũy tích. Với n giá trị, thời gian để lập nên Cdf này sẽ tỉ lệ thuận với *n* log*n*.

Sau đây là cách thực hiện Prob, vốn nhận vào một giá trị rồi trả lại xác suất lũy tích tương ứng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # class Cdf   def Prob(self, x):      if x < self.xs[0]:    return 0.0  index = bisect.bisect(self.xs, x)  p = self.ps[index - 1]  return p |

Module bisect cung cấp một cách thực hiện tìm kiếm nhị phân. Và sau đây là cách thực hiện Value, vốn nhận một xác suất lũy tích rồi trả lại giá trị tương ứng:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | # class Cdf      def Value(self, p):      if p < 0 or p > 1:  raise ValueError('p must be in range [0, 1]')   index = bisect.bisect\_left(self.ps, p)  return self.xs[index] |

Cho trước một Cdf, ta có thể tính được Pmf bằng cách tính hiệu số giữa các xác suất lũy tích kế tiếp. Nếu bạn gọi phương thức tạo lập (constructor) của Cdf rồi truyền vào một Pmf, nó sẽ tính các hiệu số bằng cách gọi Cdf.Items:

|  |  |
| --- | --- |
| 1  2  3  4  5  6 | # class Cdf    def Items(self):        a = self.ps      b = np.roll(a, 1)      b[0] = 0      return zip(self.xs, a-b) |

np.roll đẩy các phần tử của a về phía phải, và “cuộn” phần tử cuối trở lại đầu. Ta thay thế phần tử đầu tiên của b bằng 0 rồi tính hiệu số a-b. Kết quả là một ma trận NumPy chứa các xác suất.

Cdf cung cấp Shift và Scale, vốn sửa đổi các giá trị trong Cdf, nhưng các xác suất cần được coi là đối tượng không biến đổi được (immutable).

**6.7  Mô men**

Bất kì khi nào bạn lấy một mẫu và giản hóa nó thành một con số, thì con số đó được gọi là một đặc trưng thống kê. Các đặc trưng thống kê mà ta đã gặp gồm có trị trung bình, phuong sai, trung vị, và khoảng tứ phân vị.

**Mô men gốc** là một loại đặc trưng thống kê. Nếu bạn có một mẫu các giá trị, *xi*, thì mô men gốc bậc *k* là:

m'_k = \frac{1}{n} \sum{i} x_i^k 

Hay nếu bạn muốn viết bằng Python:

|  |  |
| --- | --- |
| 1  2 | def RawMoment(xs, k):     return sum(x\*\*k for x in xs) / len(xs) |

Khi *k*= 1 kết quả sẽ là trị trung bình, \bar{x} . Các mô men gốc khác thì bản thân chúng không có nghĩa lắm, nhưng vẫn được dùng trong một số khâu tính toán.

Các **mô men trung tâm** thì có ích hơn. Mô men trung tâm bậc *k* là:

m_k = \frac{1}{n} \sum{i} \left(x_i - \bar{x} \right) ^k 

Hay viết bằng Python:

|  |  |
| --- | --- |
| 1  2  3 | def CentralMoment(xs, k):   mean = RawMoment(xs, 1)   return sum((x - mean)\*\*k for x in xs) / len(xs) |

Khi *k* = 2 kết quả là mô men trung tâm bậc 2, và bạn có thể nhận ra đó là phương sai. Định nghĩa của phương sai cho ta một lời gợi ý giải thích tại sao những đặc trưng thống kê này được gọi là mô men. Nếu như ta gắn các vật nặng lên một thanh thước ở từng vị trí *xi*, rồi cho thanh quanh xung quanh vị trí trung bình, thì mô men quán tính của các vật nặng sẽ bằng phương sai của các trị số trọng lượng. Nếu bạn không quen với khái niệm mô men quán tính, hãy xem <http://en.wikipedia.org/wiki/Moment_of_inertia>.

Khi bạn báo cáo các đặc trưng thống kê kiểu mô men, điều quan trọng là nghĩ về đơn vị đo. Chẳng hạn, nếu các trị số *xi* đo bằng cm, thì mô men gốc thứ nhất cũng đo bằng cm. Nhưng mô men bậc 2 có đơn vị cm2, và mô men bậc 3 có đơn vị cm3, và cứ như vậy.

Do mang theo đơn vị nên các mô men này rất khó diễn giải. Đó là lí do tại sao với mô men bậc 2, ta thường báo cáo độ lệch chuẩn, vốn là căn bậc hai của phương sai; vì vậy độ lệch chuẩn cùng đơn vị với *xi*.

**6.8  Độ bất đối xứng**

**Độ bất đối xứng** là một thuộc tính để mô tả hình dạng của một phân bố. Nếu phân bố này đối xứng quanh trung tâm của nó, thì phân bố có tính đối xứng. Nếu như các giá trị kéo dài về phía bên phải thì nó được gọi là “lệch phải” còn nếu các giá trị kéo dài về phía trái – “lệch trái.”

Việc dùng chữ “lệch (skewed)” ở đây không đồng nghĩa với “chệch (biased).” Lệch chỉ đơn thuần mô tả hình dạng của phân bố chứ không nói gì về quá trình lấy mẫu có bị chệch hay không.

Một vài đặc trưng thống kê thường được dùng để lượng hóa độ bất đối xứng của một phân bố. Với một chuỗi giá trị cho trước, *x*i, **hệ số bất đối xứng của mẫu**, *g*1, có thể được tính bởi:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | def StandardizedMoment(xs, k):  var = CentralMoment(xs, 2)  std = math.sqrt(var)    return CentralMoment(xs, k) / std\*\*k  def Skewness(xs):      return StandardizedMoment(xs, 3) |

*g*1 là **momen chuẩn hóa** bậc ba, nghĩa là nó đã được chuẩn hóa để không mang đơn vị.

Độ bất đối xứng âm cho thấy một phân bố lệch qua trái; độ bất đối xứng dương cho thấy một phân bố lệch qua phải. Độ lớn của *g*1 chỉ thị mức độ bất đối xứng, song bản thân nó thì không dễ lý giải.

Trên thực tế, việc tính toán độ bất đối xứng của một mẫu thường không phải là cách làm hay. Một điểm biệt lập, nếu có mặt, sẽ làm ảnh hưởng đến *g*1 không theo tỉ lệ thông thường.

Một cách khác để đánh giá độ bất đối xứng của một phân bố là nhìn vào tương quan giữa trị trung bình và số trung vị. Các giá trị cực hạn sẽ có ảnh hưởng nhiều đến trị trung bình hơn là đến số trung vị, vì vậy nếu một phân bố bị lệch trái thì trị trung bình sẽ nhỏ hơn số trung vị.

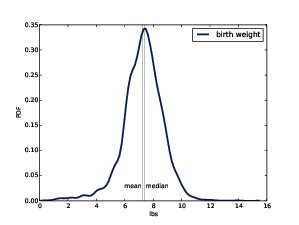
**Hệ số bất đối xứng trung vị của Pearson** là một cách đo khác đối với độ bất đối xứng, dựa trên hiệu số giữa trị trung bình số trung vị:

|  |
| --- |
| *gp* = 3 (x − *m*) / *S* |

Trong đó *x* là trị trung bình mẫu, *m* là trung vị, còn *S* là độ lệch chuẩn. Hay viết bằng Python:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  1  1 | def Median(xs):  cdf = thinkstats2.Cdf(xs)   return cdf.Value(0.5)  def PearsonMedianSkewness(xs):     median = Median(xs)    mean = RawMoment(xs, 1)  var = CentralMoment(xs, 2)    std = math.sqrt(var)  gp = 3 \* (mean - median) / std  return gp |

Đặc trưng thống kê này **vững**, theo nghĩa nó ít bị ảnh hưởng bởi các điểm biệt lập.

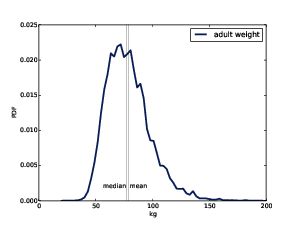


|  |
| --- |
| Hình 6.3: Hàm PDF ước tính cho số liệu cân nặng trẻ sơ sinh, số liệu theo nguồn NSFG. |

Lấy ví dụ, hãy cùng xem độ bất đối xứng trong cân nặng trẻ sơ sinh từ số liệu sinh sản NSFG. Sau đây là mã lệnh ước tính và vẽ hàm PDF:

|  |  |
| --- | --- |
| 1  2  3  4 | live, firsts, others = first.MakeFrames()  data = live.totalwgt\_lb.dropna()  pdf = thinkstats2.EstimatedPdf(data)  thinkplot.Pdf(pdf, label='birth weight') |

Hình [6.3](https://greenteapress.com/thinkstats2/html/thinkstats2007.html#density_totalwgt_kde) cho thấy kết quả. Phần đuôi phía trái dường như dài hơn bên phải, vì vậy ta ngờ rằng phân bố này bị lệch trái. Trị trung bình 7.27 lbs, hơi nhỏ hơn trung vị, 7.38 lbs, điều này thống nhất với lệch trái. Và cả hai hệ số bất đối xứng đều âm: độ bất đối xứng mẫu là -0.59; độ bất đối xứng trung vị Pearson là -0.23.



|  |
| --- |
| Hình 6.4: Hàm PDF ước tính cho số liệu cân nặng người trưởng thành, số liệu theo nguồn BRFSS. |

Bây giờ ta hãy so sánh phân bố này với phân bố của cân nặng người trưởng thành theo BRFSS. Một lần nữa, xét mã lệnh sau:

|  |  |
| --- | --- |
| 1  2  3  4 | df = brfss.ReadBrfss(nrows=None)  data = df.wtkg2.dropna()  pdf = thinkstats2.EstimatedPdf(data)  thinkplot.Pdf(pdf, label='adult weight') |

Hình [6.4](https://greenteapress.com/thinkstats2/html/thinkstats2007.html#density_wtkg2_kde) cho thấy kết quả. Phân bố này dường như lệch về bên phải. Dĩ nhiên, trị trung bình, 79.0, lớn hơn so với trung vị, 77.3. Độ bất đối xứng mẫu là 1.1 và độ bất đối xứng trung vị Pearson là 0.26.

Dấu của hệ số bất đối xứng cho thấy rằng phân bố bị lệch về trái hay phải, nhưng ngoài điều đó ra thì hệ số này rất khó diễn giải. Hệ số bất đối xứng mẫu thì kém vững; nghĩa là nó dễ bị ảnh hưởng bởi các điểm biệt lập. Hệ quả là, khi áp dụng hệ số này cho các phân bố lệch – lúc cần đến nó nhất – thì kết quả kém tin cậy.

Hệ số bất đối xứng trung vị Pearson được dựa trên trị trung bình và phương sai tính được, nên nó cũng bị ảnh hưởng bởi điểm biệt lập. Song vì nó không phụ thuộc vào một mô men bậc 3, nên phần nào nó vững hơn.

**6.9  Bài tập**

Lời giải bài tập này có ở file chap06soln.py.

**Bài 1**

*Nhiều người biết rằng phân bố khoản thu nhập có dạng lệch sang phải. Trong bài tập này, ta sẽ tính xem mức độ lệch đó mạnh cỡ nào.*

*Chương trình khảo sát dân số Current Population Survey (CPS) là dự án liên kết giữa Phòng dữ liệu lao động Bureau of Labor Statistics và Cục điều tra Census Bureau để nghiên cứu thu nhập cũng những biến số liên quan. Dữ liệu được thu thập năm 2013 có ở*[*http://www.census.gov/hhes/www/cpstables/032013/hhinc/toc.htm*](http://www.census.gov/hhes/www/cpstables/032013/hhinc/toc.htm)*. Tôi đã tải về hinc06.xls, một bảng tính Excel chứa thông tin về thu nhập hộ gia đình; và đã chuyển đổi thành hinc06.csv, một file CSV mà bạn có thẻ tìm thấy trong kho mã lệnh kèm theo cuốn sách này. Bạn cũng sẽ tìm thấy hinc2.py, mã lệnh này để đọ file dữ liệu trên và biến đổi dữ liệu.*

*Bộ số liệu tồn tại ở dạng một chuỗi các khoảng thu nhập và số người được điều tra rơi vào trong từng khoảng. Khoảng thấp nhất có chứa những người kê khai mức thu nhập hộ gia đình hằng năm “dưới $5000.” Khoảng cao nhất bao gồm những người kê khai mức “$250,000 hoặc hơn.”*

*Để ước tính trị trung bình cùng các đặc trưng thống kê khác từ dữ liệu này, ta cần phải đặt giả thiết về các mức giới hạn dưới và giới hạn trên, cũng như các giá trị sẽ phân bố thế nào trong mỗi khoảng. hinc2.py cung cấp InterpolateSample, trong đó thể hiện một cách mô hình hóa dữ liệu này. Nó nhận vào một DataFrame có một cột, income, cột này chứa giới hạn trên từng khoảng, và cột freq, chứa số người được điều tra của mỗi khoảng.*

*Nó cũng chứa log\_upper, một giới hạn trên được giả định cho mức cao nhất, được biểu diễn theo log10 đô-la. Trị số mặc định, log\_upper=6.0 biểu thị giả thiết rằng thu nhập cao nhất trong những người được khảo sát là*106*, hay 1 triệu đô-la.*

*InterpolateSample phát sinh ra một mẫu giả; nghĩa là một mẫu các mức thu nhập hộ gia đình sao cho số người khỏa sát được phân vào trong mỗi khoảng đúng như con số trên thực tế. Nó giả thiết rằng mức thu nhập trong mỗi khoảng thì cách đều trên thang đo log10.*

*Hãy tính số trung vị, trị trung bình, độ bất đối xứng và hệ số bất đối xứng Pearson của mẫu thu được. Có bao nhiêu phần dân số đã báo cáo thu nhập chịu thuế dưới mức trung bình? Kết quả phụ thuộc thế nào vào giới hạn trên đã chọn?*

**6.10  Thuật ngữ**

* **hàm mật độ xác suất (Probability density function, PDF)**: Đạo hàm của một hàm CDF liên tục; một hàm ánh xạ từ giá trị tới mật độ xác suất của nó.
* **mật độ xác suất (Probability density)**: Một đại lượng có thể được tích phân trên một khoảng giá trị để cho ra xác suất. Nếu các giá trị có đơn vị là cm chẳng hạn, thì mật độ xác suất được tính bằng đơn vị xác suất trên cm.
* **ước tính mật độ nhân (Kernel density estimation, KDE)**: Một thuật toán để ước tính hàm PDF dựa trên một mẫu.
* **rời rạc hóa (discretize)**: Việc xấp xỉ một hàm hoặc một phân bố liên tục bởi một hàm rời rạc. Trái nghĩa với làm trơn.
* **mô men gốc (raw moment)**: Một đặc trưng thống kê dựa trên tổng các số liệu, được lấy lũy thừa.
* **mô men trung tâm (central moment)**: Một đặc trưng thống kê dựa trên mức độ phân tán khỏi trị trung bình, được lấy lũy thừa.
* **mô men chuẩn hóa (standardized moment)**: Tỉ số giữa các mô men; tỉ số này không có đơn vị.
* **độ bất đối xứng (skewness)**: Một độ đo sự mất cân đối của phân bố.
* **hệ số bất đối xứng mẫu (sample skewness)**: Một đặc trưng thống kê loại mô men, nhằm định lượng mức độ bất đối xứng của một phân bố.
* **hệ số bất đối xứng trung vị Pearson (Pearson’s median skewness coefficient)**: Một đặc trưng thống kê nhằm định lượng mức độ bất đối xứng của một phân bố; dựa trên trung vị, trị trung bình, và độ lệch chuẩn.
* **vững (robust)**: Một đặc trưng thống kê được gọi là vững khi nó ít chịu ảnh hưởng bởi điểm biệt lập.