Probabilities, Gaussians, and Bayes' Theorem

Introduction

最后一章讨论了离散贝叶斯滤波器的一些缺点。对于许多跟踪和过滤问题，我们希望有一个单峰和连续的过滤器。也就是说，我们想用浮点数学(连续的)来为系统建模，并且只表示一个信念(单峰的)。例如，我们想说一架飞机位于纬度、经度和高度(12.34，-95.54,2389.5)。我们不希望过滤器告诉我们“它可能是(1.65， -78.01,2100.45)或它可能是(34.36，-98.23,2543.79)”。这与我们对世界如何运作的物理直觉不相符，正如我们所讨论的，计算多模态的情况可能非常昂贵。当然，多位置估计使导航变得不可能。

我们希望用一种单峰、连续的方式来表示概率，以模拟现实世界的工作方式，这在计算上是有效的。高斯分布提供了所有这些特征。

Mean, Variance, and Standard Deviations

你们大多数人都接触过统计学，但不管怎样，请允许我介绍一下这些材料。我要求你阅读材料，即使你确定你很了解它。我这样问有两个原因。首先，我想确定我们使用的术语是相同的。其次，我努力形成对统计数据的直观理解，这将在后面的章节中很好地为您服务。我们很容易通过一门统计课程，只记住公式和计算，可能对所学内容的含义感到模糊。

Random Variables

每次掷骰子的结果都在 1 到 6 之间。如果我们掷一个公平的骰子一百万次，我们期望得到 1/6 的概率。因此我们说结果 1 的概率是 1/6。同样地，如果我问你下一次掷出 1 的概率你会回答 1/6。

这种值和相关概率的组合被称为随机变量。这里的随机并不意味着过程是不确定的，只是我们缺乏关于结果的信息。掷骰子的结果是确定的，但我们缺乏足够的信息来计算结果。除了概率之外，我们不知道会发生什么。

当我们定义术语时，值的范围称为样本空间。对于一个骰子，样本空间是{1,2,3,4,5,6}。对于一枚硬币，样本空间为{H, T}。空间是一个数学术语，它指的是具有结构的集合。骰子的样本空间是 1 到 6 范围内的自然数的子集。

另一个随机变量的例子是大学学生的身高。这里的样本空间是由生物学定义的两个界限之间的一个实数范围。

掷硬币和掷骰子等随机变量是离散的随机变量。这意味着它们的样本空间由有限数量的值或可数的无限数量的值(如自然数)表示。人类的身高被称为连续随机变量，因为它们可以在两个极限之间取任何实值。

不要混淆随机变量的测量值与实际值。如果我们只能测量 0.1 米的高度，我们就只能记录 0.1、0.2、0.3……2.7 的数值，从而产生 27 个离散的选择。尽管如此，一个人的身高可以在这些范围内的任意实值之间变化，所以身高是一个连续的随机变量。

在统计学中，大写字母用于表示随机变量，通常来自字母表的后半部分。因此，我们可以说 X 是代表掷骰子的随机变量，或者 Y 是新生诗歌班学生的身高。后面的章节使用线性代数来解决这些问题，所以我们将遵循惯例，用小写表示向量，用大写表示矩阵。不幸的是，这些约定相互冲突，您必须根据上下文确定作者使用的是哪一个。我总是用粗体符号来表示

向量和矩阵，这有助于区分两者。

Probability Distribution

概率分布给出了随机变量在样本空间中取任意值的概率。例如，对于一个六面骰子，我们可以说:

Value	Probability
1	1/6
2	1/6
3	1/6
4	1/6
5	1/6
6	1/6

我们用小写 p 表示这个分布:p(x)。使用普通函数表示法，我们可以这样写:

$$P(X=4) = p(4) = \frac{1}{6}$$

这表示骰子落在 4 上的概率是 1/6。$P(X=x_k)$ 是 X 是 x_k 的概率表示法。注意符号上的细微差别。大写 P 事件发生的概率，小写 p 表示概率分布函数。如果你不留心观察，这可能会使你误入歧途。一些文本使用 Pr 而不是用 P 来改善这一点。

另一个例子是均匀硬币。它有样本空间{H, T}。硬币是公平的，所以正面(H)的概率是 50%反面(T)的概率是 50%。我们把它写成：

$$P(X=H) = 0.5$$
$$P(X=T) = 0.5$$

样本空间不是唯一的。一个模的样本空间是{1,2,3,4,5,6}。另一个有效的样本空间是{偶，奇}。另一个可能是{所有角落的点，而不是所有角落的点}。只要样本空间包含所有的可能性，并且任何单一事件仅由一个元素描述，那么它就是有效的。{偶数，1,3,4,5}不是骰子的有效样本空间，因为值为 4 的“偶数”和“4”都匹配。

一个离散随机值的所有值的概率称为离散概率分布，一个连续随机值的所有值的概率称为连续概率分布。

要成为一个概率分布，每个值 x_i>=0，因为没有概率可以小于 0。其次，所有值的概率之和必须等于 1。

这对于抛硬币来说应该是直观清楚的:如果得到正面的几率是 70%，那么得到反面的几率一定是 30%。我们把这一要求规定为：

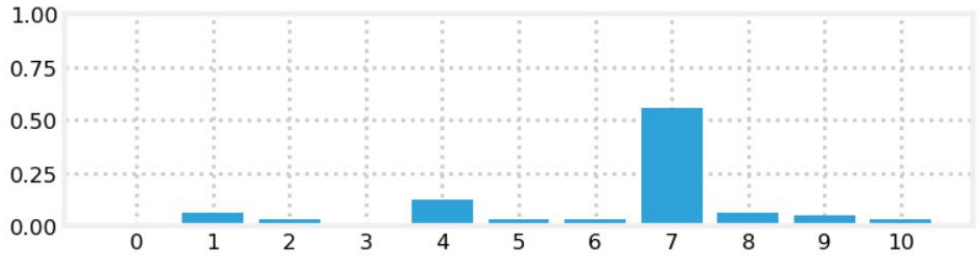
$$\sum_u P(X=u) = 1$$

对于离散分布，和连续分布。

$$\int_u P(X=u)\,du = 1$$

在前一章中，我们使用概率分布来估计一条狗在走廊中的位置。例如:

```
In [3]: import numpy as np
        import kf_book.book_plots as book_plots

        belief = np.array([1, 4, 2, 0, 8, 2, 2, 35, 4, 3, 2])
        belief = belief / np.sum(belief)
        with book_plots.figsize(y=2):
            book_plots.bar_plot(belief)
        print('sum = ', np.sum(belief))

sum =  1.0
```



每个位置的概率在 0 到 1 之间，所有的和等于 1，所以这是一个概率分布。每个概率都是离散的，所以我们可以更准确地称之为离散概率分布。在实践中，除非我们有特别的理由加以区分，否则我们会忽略离散和连续这两个词。

The Mean, Median, and Mode of a Random Variable

对于一组数据，我们通常想知道这组数据的代表值或平均值。有很多测量方法，这个概念被称为集中趋势测量。例如，我们可能想知道一个班级学生的平均身高。我们都知道如何找到一组数据的平均值，但是让我来详细说明这一点，以便引入更正式的符号和术语。平均的另一个词是平均值。我们通过将这些值相加并除以值的个数来计算平均值。如果学生的高度以米为单位：

$$X = \{1.8, 2.0, 1.7, 1.9, 1.6\}$$

我们计算的均值是：

$$\mu = \frac{1.8 + 2.0 + 1.7 + 1.9 + 1.6}{5} = 1.8$$

传统上用 μ (mu) 来表示平均值。

我们可以用这个方程来形式化计算

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i$$

NumPy 提供 NumPy .mean()来计算平均值。

```
In [4]: x = [1.8, 2.0, 1.7, 1.9, 1.6]
        np.mean(x)

Out[4]: 1.8
```

为了方便起见，NumPy 数组提供了方法 mean()。

```
In  [5]: x = np.array([1.8, 2.0, 1.7, 1.9, 1.6])
         x.mean()
Out[5]: 1.8
```

一组数字的众数是最常出现的数。如果只有一个数字最常出现，我们说它是单峰集，如果有两个或两个以上的数字出现的频率最多，比这个集合是多峰集。例如集合{1,2,2,2,3,4,4,4}有 2、4 阶模态，为多模态;集合{5,7,7,13}有 7 阶模态，为单模态。在本书中，我们将不以这种方式计算模态，但我们确实在更一般的意义上使用单模态和多模态的概念。例如，在离散贝叶斯一章中，我们谈到我们相信狗的位置是一个多模态分布，因为我们对不同的位置分配了不同的概率。

最后，一组数字的中位数是该组数字的中点，因此，一半的值在中位数以下，一半在中位数以上。在这里，上面和下面与被排序的集合有关。如果集合包含偶数个值，则将中间两个数取平均值。

Numpy 提供 Numpy .median()来计算中值。可以看到{1.8,2.0,1.7,1.9,1.6}的中位数是 1.8，因为 1.8 是这个集合排序后的第三个元素。在这种情况下，中位数等于平均值，但这通常不是正确的。

```
In  [6]: np.median(x)
Out[6]: 1.8
```

Expected Value of a Random Variable

一个随机变量的期望值是它的平均值如果我们取它的无限个样本然后取这些样本的平均值。假设我们有 x=[1,3,5]，每个值都是等可能的。平均而言，我们期望 x 具有什么价值?

它应该是 1 3 5 的平均值，当然，是 3。这应该说得通;我们期望相同数量的 1 3 和 5 出现，所以(1+3+5)/3=3 显然是这个无限样本序列的平均值。也就是说，这里的期望值就是样本空间的均值。

现在假设每个值都有不同的发生概率。假设 1 有 80%的机会发生，3 有 15%的机会，5 只有 5%的机会。在本例中，我们通过将 x 的每个值乘以事件发生的概率百分比，并将结果相加来计算期望值。对于这种情况，我们可以计算

$$\mathbb{E}[X] = (1)(0.8) + (3)(0.15) + (5)(0.05) = 1.5$$

在这里，我为 x 的期望值引入了符号 E[X]，一些其他的文本是 E(x)。x 的值 1.5 更直观，因为 x 更可能是 1，而不是 3 或 5,3 也更可能是 5。

我们可以将其形式化（x_i 是 x 的第 i 个值，p_i 是 x_i 的概率）

$$\mathbb{E}[X] = \sum_{i=1}^{n} p_i x_i$$

如果 x 是连续的，我们用求和来代替积分，像这样

$$\mathbb{E}[X] = \int_a^b x f(x)\, dx$$

F(x)为 x 的概率分布函数。我们还不会用到这个方程，但我们会在下一章用到它。

我们可以编写一些 Python 来模拟这一点。这里我取 100 万个样本，计算刚刚解析计算的分布的期望值。

```
In [7]: total = 0
        N = 1000000
        for r in np.random.rand(N):
            if r <= .80: total += 1
            elif r < .95: total += 3
            else: total += 5

        total / N
Out[7]: 1.502562
```

您可以看到，计算值接近于解析导出的值。它不是精确的，因为要得到一个精确的值需要无限的样本容量。

Variance of a Random Variable

上面的计算告诉了我们学生的平均身高，但它并没有告诉我们想要知道的一切。例如，假设我们有三个班级的学生，我们将其标记为 X、Y、Z，高度如下:

```
In [8]: X = [1.8, 2.0, 1.7, 1.9, 1.6]
        Y = [2.2, 1.5, 2.3, 1.7, 1.3]
        Z = [1.8, 1.8, 1.8, 1.8, 1.8]
```

使用 NumPy，我们看到每个类的平均高度是相同的。

```
In [9]: print(np.mean(X), np.mean(Y), np.mean(Z))
1.8 1.8 1.8
```

每一类的平均值是 1.8 米，但请注意，第二类的高度变化要比第一类的高度变化大得多，而第三类的高度则完全没有变化。

平均值告诉我们数据的一些信息，但不是全部。我们想要弄清楚，学生的身高有多大的变化。你可以想象很多原因。也许一个学区需要订购 5000 张课桌，他们希望确保购买的尺寸能适应学生的身高范围。

统计学已经将测量变异的概念形式化为标准偏差和方差的概念。计算方差的公式为:

$$VAR(X) = \mathbb{E}[(X - \mu)^2]$$

暂时忽略平方，您可以看到方差是样本空间 X 与平均值 u 的期望值。根据期望值的公式所以我们可以把它代入上面的方程得到：

$$VAR(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$$

让我们计算这三个类的方差，看看我们会得到什么值，并熟悉这个概念。

X 的平均值是 1.8，所以我们计算：

$$\begin{aligned} VAR(X) &= \frac{(1.8-1.8)^2 + (2-1.8)^2 + (1.7-1.8)^2 + (1.9-1.8)^2 + (1.6-1.8)^2}{5} \\ &= \frac{0 + 0.04 + 0.01 + 0.01 + 0.04}{5} \\ VAR(X) &= 0.02\,m^2 \end{aligned}$$

NumPy 提供函数 var()来计算方差:

```
In [10]: print(f"{np.var(X):.2f} meters squared")
```

```
0.02 meters squared
```

这可能有点难以解释。高度单位是米，方差是米的平方。因此，我们有了一个更常用的度量标准偏差，它被定义为方差的平方根:

$$\sigma = \sqrt{VAR(X)} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$$

通常使用 σ 作为标准差， σ 的平方作为方差，在本书的大部分内容中，我将使用方差而不是标准差。

对于第一类，我们计算标准差

$$\begin{aligned}\sigma_x &= \sqrt{\frac{(1.8-1.8)^2 + (2-1.8)^2 + (1.7-1.8)^2 + (1.9-1.8)^2 + (1.6-1.8)^2}{5}} \\ &= \sqrt{\frac{0 + 0.04 + 0.01 + 0.01 + 0.04}{5}} \\ \sigma_x &= 0.1414\end{aligned}$$

我们可以使用 NumPy 方法 NumPy .std()来验证这个计算，该方法会计算标准差。'std'是标准偏差(standard deviation)的常见缩写。

```
In [11]: print(f"std {np.std(X):.4f}")
         print(f"var {np.std(X)**2:.4f}")
```

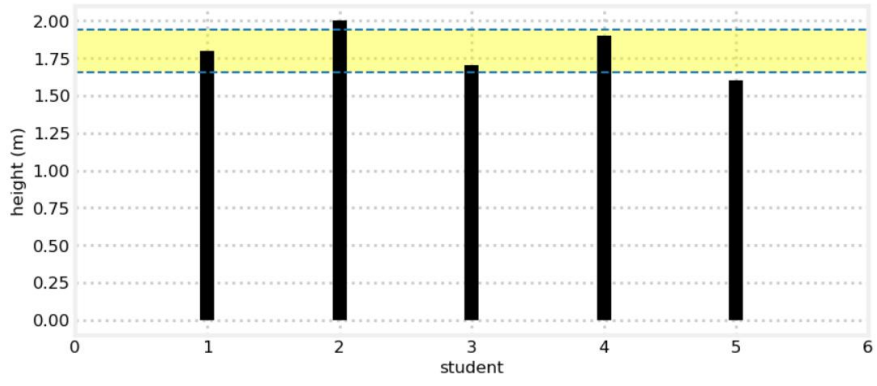
```
std 0.1414
var 0.0200
```

当然， $0.1414^2 = 0.02$, 这和之前方差的计算是一致的。

标准差代表什么?它告诉我们高度之间有多大的变化。“多少”不是一个数学术语。一旦我们在下一节引入高斯函数的概念，我们将能够更精确地定义它。很多情况下 68%的值都在一个标准差范围内。换句话说，我们可以得出这样的结论:随机选取一个班级，68%的学生身高在 1.66(1.8-0.1414)米到 1.94(1.8+0.1414)米之间。

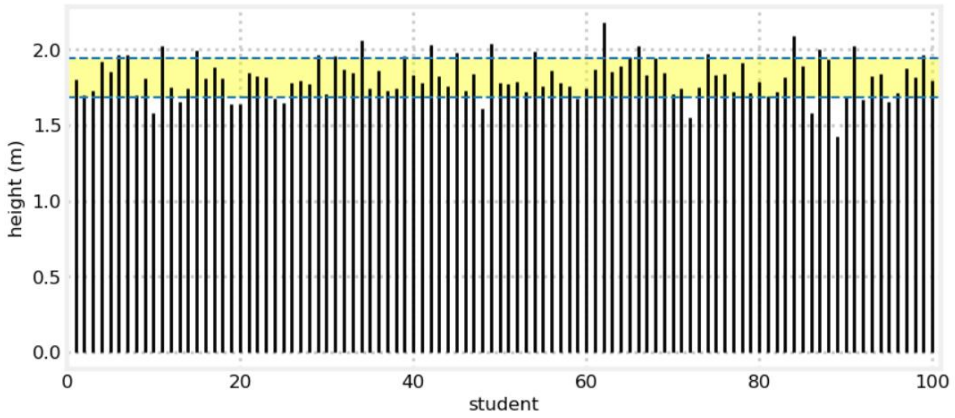
我们可以用图表来说明:

```
In [13]: from kf_book.gaussian_internal import plot_height_std
         import matplotlib.pyplot as plt

         plot_height_std(X)
```



对于 5 个学生，我们显然不能在一个标准差内得到 68%我们确实看到，5 个学生中有 3 个的得分在±1 以内，即 60%，这是 5 个样本中最接近 68%的结果。让我们看看有 100 名学生的班级的结果。

```
In [14]: from numpy.random import randn
         data = 1.8 + randn(100)*.1414
         mean, std = data.mean(), data.std()

         plot_height_std(data, lw=2)
         print(f'mean = {mean:.3f}')
         print(f'std  = {std:.3f}')
```




```
mean = 1.816
std  = 0.129
```

肉眼看，大约 68%的高度位于平均值 1.8 的±1σ 范围内，但我们可以用代码验证这一点。

```
In [15]: np.sum((data > mean-std) & (data < mean+std)) / len(data) * 100.
Out[15]: 67.0
```

我们将很快对此进行更深入的讨论。现在我们来计算标准差

$$Y = [2.2, 1.5, 2.3, 1.7, 1.3]$$

Y 的平均值是 1.8，所以

$$\sigma_y = \sqrt{\frac{(2.2-1.8)^2 + (1.5-1.8)^2 + (2.3-1.8)^2 + (1.7-1.8)^2 + (1.3-1.8)^2}{5}}$$
$$= \sqrt{0.152} = 0.39\ m$$

我们将用 NumPy 来验证

```
In [16]: print(f'std of Y is {np.std(Y):.2f} m')
std of Y is 0.39 m
```

这符合我们的预期。Y 的高度变化较大，标准差也较大。

最后，让我们计算 Z 的标准差。这些值没有变化，所以我们期望标准差为零。

$$\sigma_z = \sqrt{\frac{(1.8-1.8)^2 + (1.8-1.8)^2 + (1.8-1.8)^2 + (1.8-1.8)^2 + (1.8-1.8)^2}{5}}$$
$$= \sqrt{\frac{0+0+0+0+0}{5}}$$
$$\sigma_z = 0.0\ m$$

```
In [17]: print(np.std(Z))
```

```
0.0
```

在我们继续之前，我需要指出，我忽略了男性平均身高高于女性这一事实。一般来说，一个只包含男性或女性的班级的身高方差要小于一个包含两种性别的班级。其他因素也是如此。营养良好的儿童比营养不良的儿童高。斯堪的纳维亚人比意大利人高。在设计实验时，统计学家需要考虑这些因素。

我建议我们可以执行这个分析来为一个学区订购书桌。对于每个年龄段来说，可能会有两个不同的平均值——一个集中在女性的平均身高周围，另一个集中在男性的平均身高周围。整个班级的平均值将介于两者之间。如果我们为所有学生购买书桌，我们很可能最终会得到既不适合学校里的男生也不适合女生的书桌!

我们将不在本书中讨论这些问题。如果您需要学习处理这些问题的技术，可以参考任何标准概率文本。

Why the Square of the Differences

为什么方差要取差值的平方呢?我可以做很多数学运算，但让我们用一种简单的方式来看。下图是 X 的值与 X=[3,−3,3,−3]的平均值的对比图。

```
In [18]: X = [3, -3, 3, -3]
mean = np.average(X)
for i in range(len(X)):
    plt.plot([i ,i], [mean, X[i]], color='k')
plt.axhline(mean)
plt.xlim(-1, len(X))
plt.tick_params(axis='x', labelbottom=False)
```

如果我们不取差值的平方这些符号就会抵消掉：

$$\frac{(3-0)+(-3-0)+(3-0)+(-3-0)}{4}=0$$

这显然是不正确的，因为数据中有大于 0 的方差。

也许我们可以用绝对值?通过检查，我们可以看到结果是 12/4=3，这当然是正确的-每个值与平均值相差 3。但是如果我们有 Y=[6,−2,−3,1]?在这种情况下，我们得到 12/4=3。Y 显然比 X 要分散。但计算得到相同的方差。如果我们使用使用平方的公式，我们得到 Y 的方差为 3.5，这反映了其更大的变化。

这并不是正确性的证明。事实上，这项技术的发明者卡尔·弗里德里希·高斯(Carl Friedrich Gauss)认识到，它有些武断。如果存在异常值，那么将差异平方给这一项不成比例的权重。例如，让我们看看会发生什么，如果有:

```
In [19]: X = [1, -1, 1, -2, -1, 2, 1, 2, -1, 1, -1, 2, 1, -2, 100]
         print(f'Variance of X with outlier    = {np.var(X):6.2f}')
         print(f'Variance of X without outlier = {np.var(X[:-1]):6.2f}')

Variance of X with outlier    = 621.45
Variance of X without outlier =   2.03
```

这是“正确”吗?你告诉我。没有 100 的异常值，我们得到 $\sigma^2 = 2.03$，这准确地反映了 X 在没有异常值的情况下是如何变化的。一个异常值淹没了方差计算。我们是想要搅乱计算以便我们知道有一个异常值，还是稳健地合并异常值并仍然提供一个接近没有异常值的值的估计?再说一遍，你来告诉我。显然这取决于你的问题。

我不会继续沿着这条路走下去;如果你有兴趣，你可能想看看 James Berger 在这个问题上做的工作，在一个叫做贝叶斯鲁棒性的领域，或者 Peter J. Huber 关于鲁棒统计的优秀出版物。在本书中，我们总是使用高斯定义的方差和标准差。

从这里收集的要点是，这些汇总的统计数据总是告诉我们一个关于数据的不完整的故事。在这个例子中，高斯定义的方差并没有告诉我们有一个大的异常值。然而，它是一个强大的工具，因为我们可以用几个数字简明地描述一个大的数据集。如果我们有 10 亿个数据点，我们就不想用肉眼检查地块或看数字列表;摘要统计为我们提供了一种有用地描述数据形状的方法。

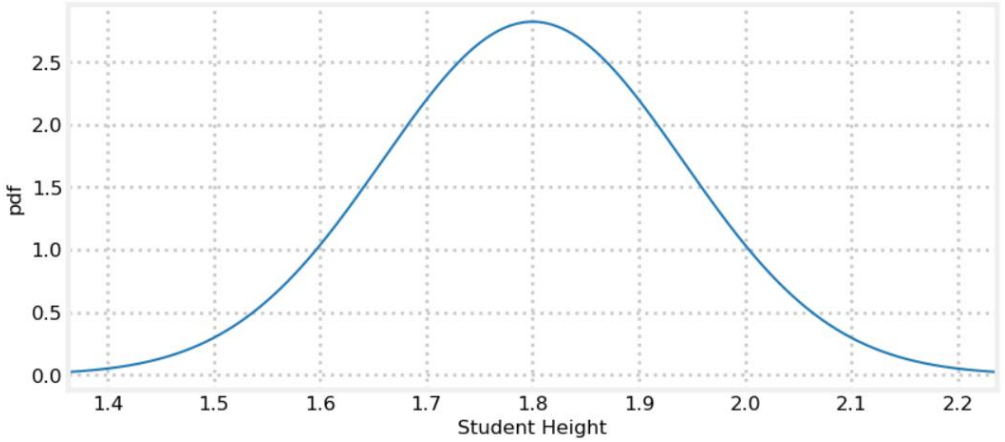
Gaussians

我们现在准备学习高斯函数。让我们提醒自己这一章的动机。

我们希望用一种单峰、连续的方式来表示概率，以模拟现实世界的工作方式，这在计算上是有效的。

让我们看一个高斯分布的图来对我们正在讨论的东西有一个感觉。

```
In [20]: from filterpy.stats import plot_gaussian_pdf
         plot_gaussian_pdf(mean=1.8, variance=0.1414**2,
                           xlabel='Student Height', ylabel='pdf');
```



这条曲线是概率密度函数，简称 pdf。它显示了随机变量取值的相对可能性。从图表中我们可以看出，身高接近 1.8 米的学生比接近 1.7 米的学生更有可能，身高接近 1.9 米的学生比身高接近 1.4 米的学生更有可能。换句话说，很多学生的身高接近 1.8 米，很少有学生的身高达到 1.4 米或 2.2 米。最后，请注意曲线以 1.8 米的平均值为中心。

我在 supporting_notebook 文件夹中的 Notebook computing_and_plotting_pdf 中解释了如何绘制高斯曲线，以及更多内容。你可以在线阅读[1]。

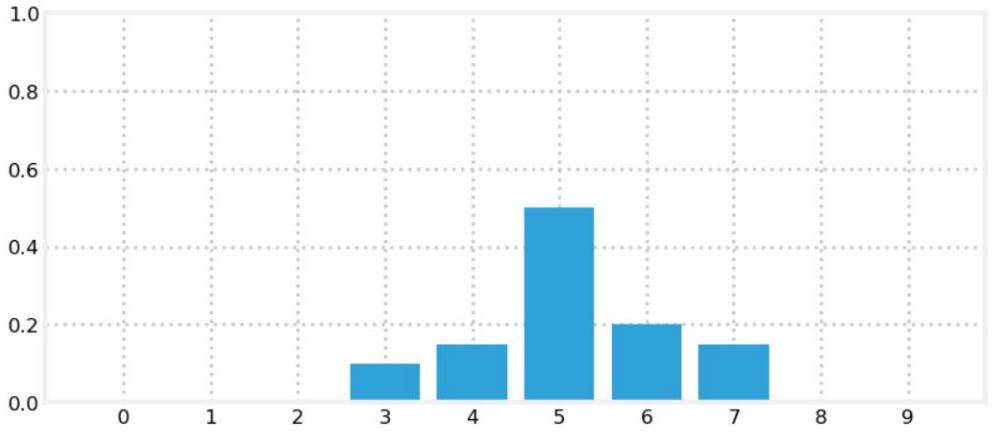
这对你来说可能就是“钟形曲线”。这条曲线是普遍存在的，因为在真实世界的条件下，

许多观测都是以这种方式分布的。我不会用“钟形曲线”来指代高斯分布，因为许多概率分布都有类似的钟形曲线形状。非数学来源可能不那么精确，所以当你看到这个术语在没有定义的情况下使用时，要明智地得出结论。

这条曲线并不是高度所特有的——大量的自然现象都表现出这种分布，包括我们用来过滤问题的传感器。正如我们将看到的，它也具有我们正在寻找的所有属性——它以概率的形式表示一个单峰信念或值，它是连续的，而且它是计算高效的。我们很快就会发现，它还有其他我们可能没有意识到自己渴望的令人满意的品质。

为了进一步激励你，回想一下离散贝叶斯一章中概率分布的形状:

```
In [21]: import kf_book.book_plots as book_plots
belief = [0., 0., 0., 0.1, 0.15, 0.5, 0.2, .15, 0, 0]
book_plots.bar_plot(belief)
```

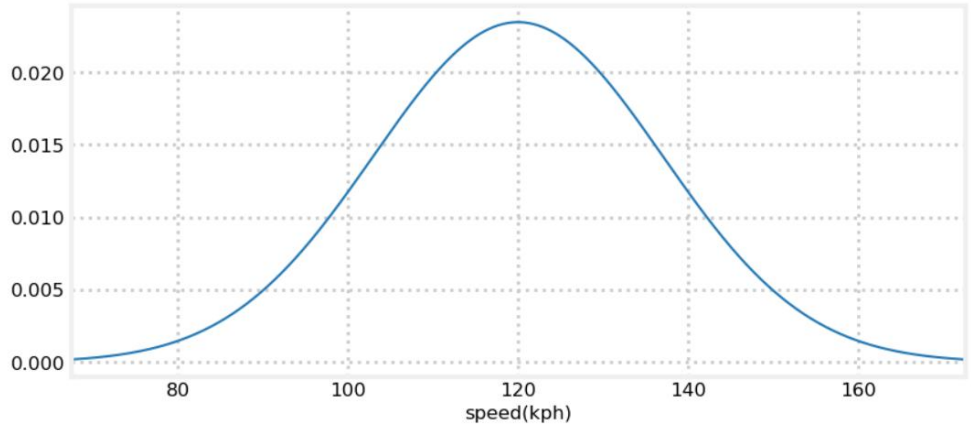


它们不是完美的高斯曲线，但它们很相似。我们将使用高斯函数来代替那一章中使用的离散概率!

Nomenclature

在我们继续之前，先来介绍一下术语——这个图表描述了一个随机变量的概率密度，它的值在(−∞..∞)之间。这是什么意思?想象一下，我们在高速公路上对汽车的速度进行无限精确的测量。然后，我们可以通过显示以任何给定速度通过的汽车的相对数量来绘制结果。如果平均速度是 120 公里每小时，它可能是这样的:

```
In [22]: plot_gaussian_pdf(mean=120, variance=17**2, xlabel='speed(kph)');
```



y 轴表示概率密度，即以相应的 x 轴速度行驶的汽车的相对数量。我将在下一节进一步解释这一点。

高斯模型并不完美。虽然这些图表没有显示出来，但分布的尾部一直延伸到无穷远。反面是曲线的远端，那里的值是最低的。当然，人的高度或汽车的速度不能小于零，更不用说

−∞或∞。地图不是领土”是一个常见的表达，对于贝叶斯过滤和统计也是如此。上述高斯分布模型反映了实测车速的分布，但作为一种模型必然存在缺陷。模型和现实之间的差异将在这些过滤器中一次又一次地出现。高斯函数被用于数学的许多分支，不是因为它们完美地模拟了现实，而是因为它们比其他任何相对准确的选择更容易使用。然而，即使在这本书中，高斯也无法模拟现实，迫使我们使用计算成本很高的替代方法。

这些分布被称为高斯分布或正态分布。高斯和法线在这种情况下都意味着同一件事，并且可以互换使用。我在整本书中都会用到这两个词，因为不同的资料会用到这两个词，我希望你们能习惯这两个词。最后，就像这一段一样，通常会将名称缩短，然后讨论高斯分布或正态分布——这两个都是高斯分布的典型快捷名称。

Gaussian Distributions

让我们来探索高斯函数是如何工作的。高斯分布是一个连续的概率分布，它完全由两个参数描述，即均值(u)和方差(σ^2)，它被定义为：

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

Exp[x]是e^x。

如果你以前没见过这个方程，不要被它吓倒;你不需要记忆或操作它。这个函数的计算存储在' stats.py '与函数' gaussian(x, mean, var, normed=True) '。

除去常数，你可以看到它是一个简单的指数:

$$f(x) \propto e^{-x^2}$$

哪个具有我们熟悉的钟形曲线形状

```
In [23]: x = np.arange(-3, 3, .01)
         plt.plot(x, np.exp(-x**2));
```

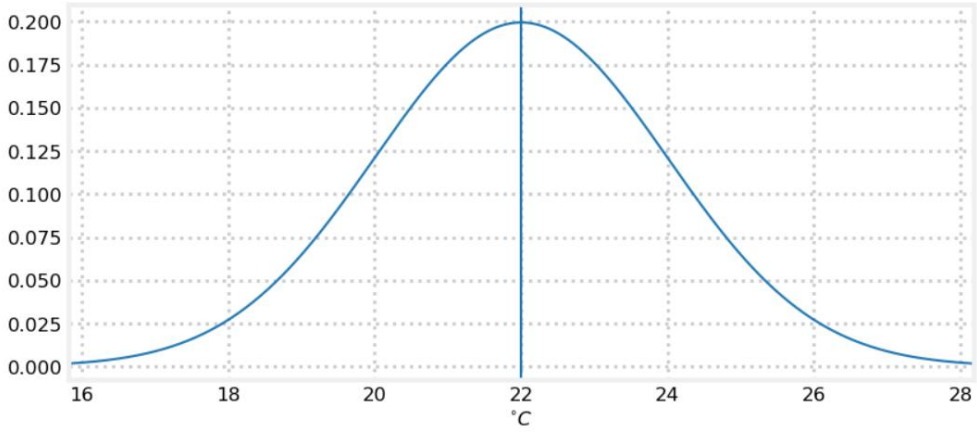
让我们回顾一下如何查看函数的代码。在单元格中，键入函数名称，后跟两个问号，然后按 CTRL+ENTER。这将打开一个显示源代码的弹出窗口。取消下一个单元格的注释，现在尝试一下。

```
In [24]: from filterpy.stats import gaussian
         #gaussian??
```

Let's plot a Gaussian with a mean of 22 ($\mu = 22$), with a variance of 4 ($\sigma^2 = 4$).

让我们绘制一个均值为 22 的高斯曲线，方差为 4。

```
In [25]: plot_gaussian_pdf(22, 4, mean_line=True, xlabel='$^{\circ}C$');
```



这条曲线意味着什么?假设我们有一个 22°C 的温度计。没有温度计是完全准确的，所以我们预计每次读数会与实际值略有偏差。然而，一个叫做中心极限定理的定理指出，如果我们进行多次测量，那么这些测量值将是正态分布的。当我们看这张图表时，我们可以看到它与给定 22° C 的实际温度下，温度计读取特定值的概率成正比。

回想一下高斯分布是连续的。想象一条无限长的直线你随机选取的一个点在 2 处的概率是多少。显然是 0%，因为有无数种选择可供选择。正态分布也是如此;在上面的图表中，恰好是 2°C 的概率是 0%，因为读数可以取无限多个值。

这条曲线是什么?我们称之为概率密度函数。曲线下任何区域的面积给出了这些值的概率。例如，如果你计算曲线下 20 到 22 之间的面积结果面积将是温度读数在这两个温度之间的概率。

这是另一种理解它的方式。岩石或海绵的密度是多少?它是衡量有多少质量被压缩到一个给定的空间。岩石密度大，海绵密度小。所以，如果你想知道一块石头的重量，但没有秤，你可以用它的体积乘以它的密度。这就得到了它的质量。在实践中，密度在大多数物体中是不同的，所以你需要将局部密度通过岩石的体积进行积分。

$$M = \iiint_R p(x, y, z)\, dV$$

我们用概率密度做同样的事情。如果你想知道 20°C 到 21°C 之间的温度你需要对上面的曲线从 20 到 21 积分。众所周知，曲线的积分就是曲线下的面积。因为这是概率密度的曲线，密度的积分就是概率。

温度正好是 22°C 的概率是多少？直观地说，0。这些都是实数，22°C 和 22.00000000000017°C 的概率是无穷小。数学上，从 22 到 22 积分会得到什么?零。

回到岩石，岩石上一个点的重量是多少?一个无限小的点必须没有权值。问单个点的权重是没有意义的，问连续分布有单个值的概率也是没有意义的。两者的答案都是 0。

实际上，我们的传感器没有无限的精度，所以 22°C 的读数意味着一个范围，比如 22±0.1°C，我们可以通过从 21.9 到 22.1 的积分来计算这个范围的概率。

我们可以用贝叶斯术语或者频率论术语来考虑。作为一个贝叶斯，如果温度计的读数正好是 22°C，那么我们的信念就用曲线来描述——我们认为实际(系统)温度接近 22°C 是非常高的，而我们认为实际温度接近 18°C 是非常低的。作为一个频率学家，我们会说，如果我们在 22°C 对一个系统进行 10 亿次温度测量，那么测量结果的直方图就会像这条曲线。

如何计算概率，或者曲线下的面积?对高斯函数方程积分

$$\int_{x_0}^{x1} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(x-\mu)^2/\sigma^2} dx$$

这被称为累积概率分布，通常缩写为 cdf。

我写了 filterpy.stats。Norm_cdf，它为你计算积分。例如，我们可以计算

```
In [26]: from filterpy.stats import norm_cdf
         print('Cumulative probability of range 21.5 to 22.5 is {:.2f}%'.format(
               norm_cdf((21.5, 22.5), 22,4)*100))
         print('Cumulative probability of range 23.5 to 24.5 is {:.2f}%'.format(
               norm_cdf((23.5, 24.5), 22,4)*100))

         Cumulative probability of range 21.5 to 22.5 is 19.74%
         Cumulative probability of range 23.5 to 24.5 is 12.10%
```

平均值 u 就像它听起来的那样——所有可能的概率的平均值。由于曲线的对称形状，它也是曲线最高的部分。温度计的读数是 22°C，所以我们用它来表示平均值。

随机变量 X 的正太分布符号 $X \sim \mathcal{N}(\mu, \sigma^2)$ 其中~意味着分布根据。这意味着我可以把温度计的温度读数表示为

$$\text{temp} \sim \mathcal{N}(22, 4)$$

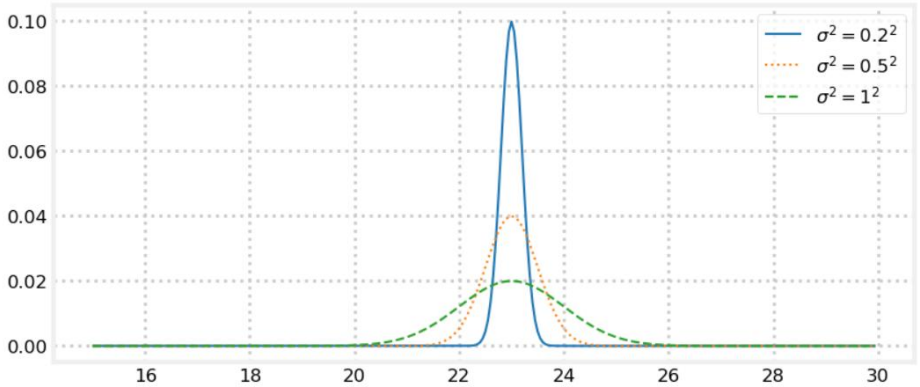
这是一个非常重要的结果。高斯函数允许我仅用两个数字就能捕获无限多个可能的值!通过均值 22 和方差 4，我可以计算任意范围内的测量值分布。

有的地方的高斯分布使用的标准差而不是方差，要注意区别。

```
In [27]: print(gaussian(x=[3.0, 2.0], mean=2.0, var=1, normed=False))

         [0.242 0.399]
```

如果高斯函数没有被归一化，它被称为高斯函数而不是高斯分布。

```
In [28]: xs = np.arange(15, 30, 0.05)
         plt.plot(xs, gaussian(xs, 23, 0.2**2), label='$\sigma^2=0.2^2$')
         plt.plot(xs, gaussian(xs, 23, .5**2), label='$\sigma^2=0.5^2$', ls=':')
         plt.plot(xs, gaussian(xs, 23, 1**2), label='$\sigma^2=1^2$', ls='--')
         plt.legend();
```



这告诉我们什么?$\sigma^2 = 0.2^2$的高斯分布非常窄。也就是说，我们相信 x=23，并且我们非常确定:在±0.2 std 范围内。相比之下，$\sigma^2 = 1^2$ 的高斯函数也认为 x=23，但我们对此不太确定。我们认为 x=23 更低，因此我们认为 x 可能的可能值是分散的，例如，我们认为 x=20 或 x=26 很有可能。$\sigma^2 = 0.2^2$几乎完全排除了 22 或 24 作为可能的值，而$\sigma^2 = 1^2$ 认为它们几乎和 23 一样可能。

如果我们回想一下温度计，我们可以认为这三条曲线代表了来自三个不同温度计的读数。$\sigma^2 = 0.2^2$ 的曲线表示一个非常精确的温度计，$\sigma^2 = 1^2$的曲线表示一个相当不准确的温度计。请注意高斯分布给我们提供了一个非常强大的特性——我们可以用两个数字——平均值和方差——来完全表示温度计的读数和误差。

高斯分布的等效形式是$\mathcal{N}(\mu, 1/\tau)$，其中μ是平均值，τ是精度。$1/\tau = \sigma^2$;它是方差的倒数。虽然我们在这本书中没有使用这个公式，但它强调了方差是衡量我们的数据是多么精确的一个指标。小的方差产生大的精度，我们的测量是非常精确的。相反，一个大的方差产生低精度我们的信念是散布在一个大的区域。你们应该会习惯用这些等价形式来思考高斯函数。用贝叶斯的术语来说，高斯函数反映了我们对测量的信念，它们表达了测量的精度，它们表达了测量中有多少方差。这些都是陈述同一事实的不同方式。

我讲得有点超前了，但在接下来的章节中，我们将使用高斯函数来表达我们的信念，比如我们正在跟踪的物体的估计位置，或者我们正在使用的传感器的准确性。

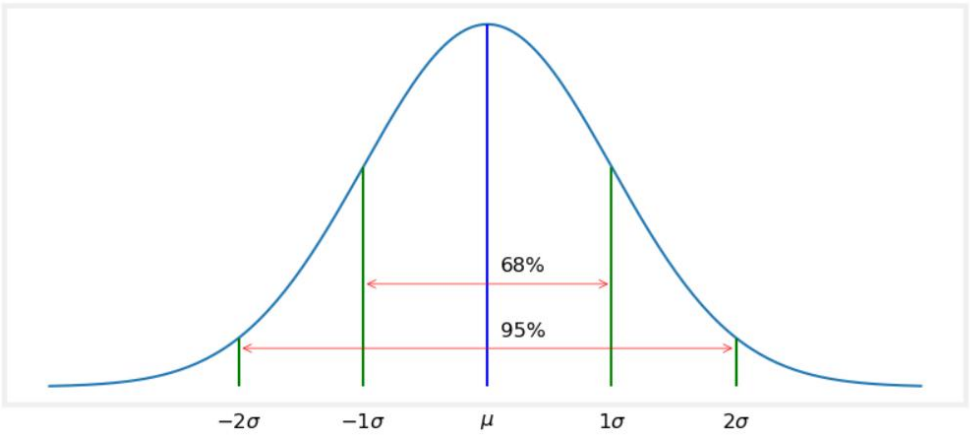
The 68-95-99.7 Rule

现在有必要花点时间讨论一下标准偏差。标准差是数据偏离均值的程度。对于高斯分布，68%的数据落在一个标准差范围内$(\pm 1\sigma)$，95%落在两个标准差范围内$(\pm 2\sigma)$，99.7%落在三个标准差范围内 $\pm 3\sigma$。这通常被称为 68-95-99.7 规则。如果你被告知一个班级的平均测试分数是 71，标准差是 9.4，你可以得出结论，如果分布是正态分布(以 71(2 9.4)计算)，95%的学生得到的分数在 52.2 到 89.8 之间。

最后，这些数字不是任意的。如果我们位置的高斯分布是$\mu = 22$，那么标准差也有米的单位。因此，$\sigma = 0.2$ 意味着 68%的测量值范围在 21.8 米到 22.2 米之间。方差是标准差的平方，因此$\sigma^2 = .04$。正如您在上一节中看到的，编写$\sigma^2 = 0.2^2$ 可能会使这个结果更有意义，因为 0.2 与数据的单位相同。

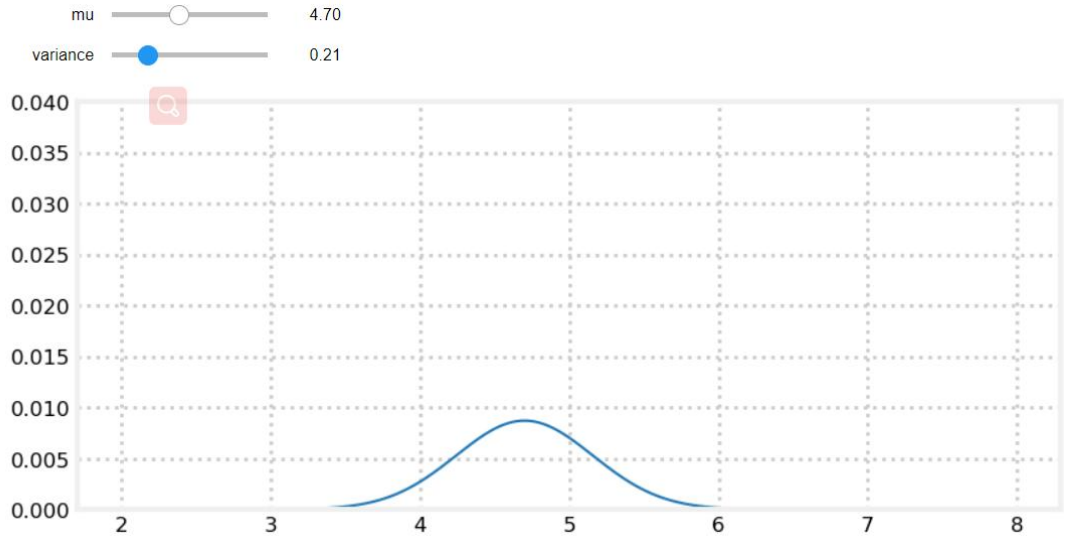
下图描述了标准差和正态分布之间的关系。

```
In [30]: from kf_book.gaussian_internal import display_stddev_plot
         display_stddev_plot()
```



Interactive Gaussians

对于那些在 jupyter notebook 上阅读这篇文章的人来说，这里有一个交互式的高斯曲线版本。使用滑块修改 μ and σ^2。调整 μ 将使图形左右移动，因为您正在调整平均值，而调整 σ^2 将使钟形曲线变厚变薄。



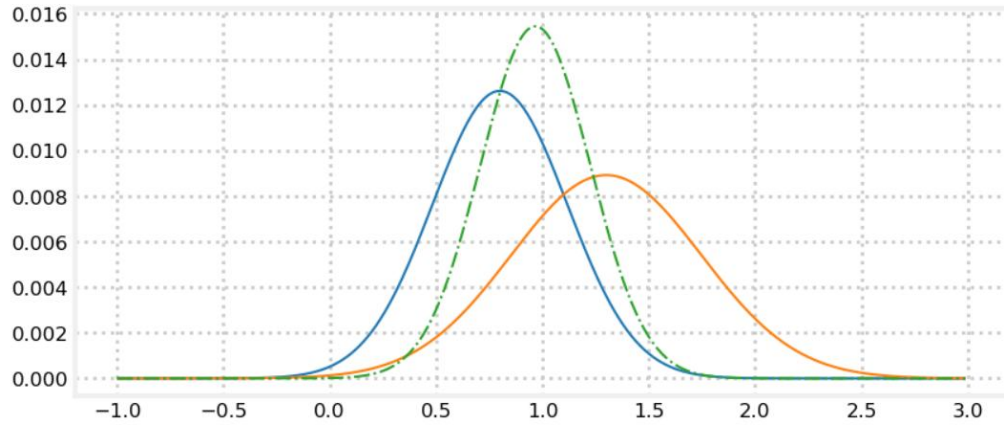
Computational Properties of Normally Distributed Random Variables

离散贝叶斯滤波器的工作原理是将任意概率的随机变量相乘并相加。卡尔曼滤波使用高斯函数而不是任意随机变量，但算法的其余部分保持不变。这意味着我们需要将高斯随机变量相乘并相加(高斯随机变量是正态分布随机变量的另一种说法)。

高斯随机变量的一个显著性质是两个独立的高斯随机变量的和也是正态分布的!这个乘积不是高斯分布，而是与高斯分布成比例。我们可以说，两个高斯分布相乘的结果是一个高

斯函数(在这种情况下，召回函数意味着值和为 1 的属性不保证)。

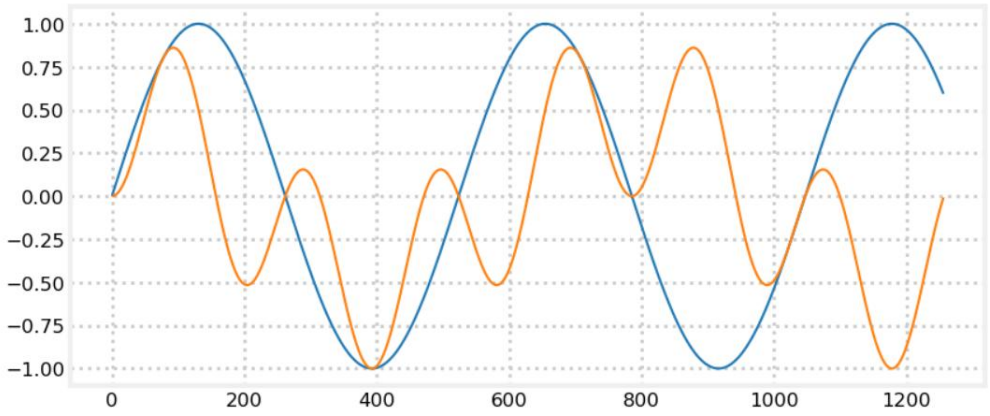
```
In [32]: x = np.arange(-1, 3, 0.01)
         g1 = gaussian(x, mean=0.8, var=.1)
         g2 = gaussian(x, mean=1.3, var=.2)
         plt.plot(x, g1, x, g2)

         g = g1 * g2  # element-wise multiplication
         g = g / sum(g)  # normalize
         plt.plot(x, g, ls='-.');
```



在这里我创建了两个高斯函数，g1=(0.8,0.1)和 g2=(1.3,0.2)并绘制了它们。然后我将它们相乘并将结果标准化。正如你所看到的，结果看起来像一个高斯分布。

高斯函数是非线性函数。通常，如果你把一个非线性方程相乘你会得到一个不同类型的函数。例如，两个 sin 相乘的形状与 sin(x)非常不同。

```
In [33]: x = np.arange(0, 4*np.pi, 0.01)
         plt.plot(np.sin(1.2*x))
         plt.plot(np.sin(1.2*x) * np.sin(2*x));
```



但是两个高斯分布相乘的结果是高斯函数。这是卡尔曼滤波器在计算上可行的一个关键原因。换句话说，卡尔曼滤波器使用高斯因为它们在计算上很好。

两个独立高斯函数的乘积为:

$$\mu = \frac{\sigma_1^2\mu_2 + \sigma_2^2\mu_1}{\sigma_1^2 + \sigma_2^2}$$

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

两个高斯随机变量的和由

$$\mu = \mu_1 + \mu_2$$
$$\sigma^2 = \sigma_1^2 + \sigma_2^2$$

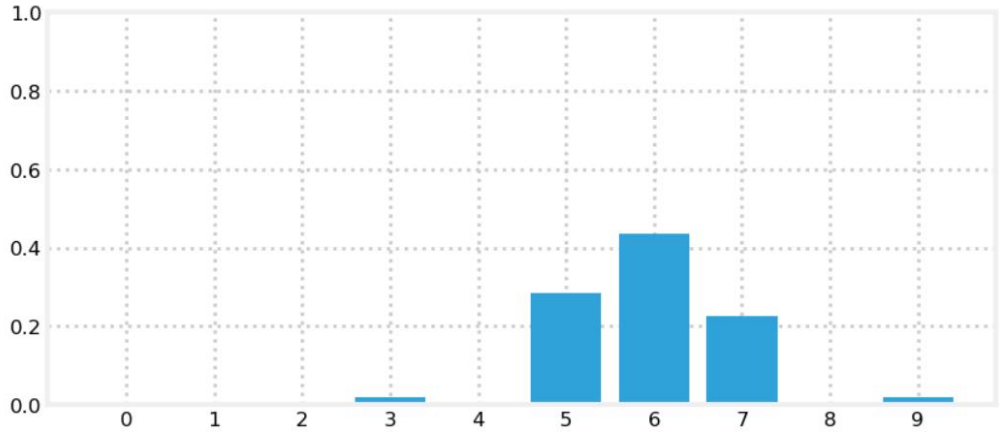
在本章的最后，我推导出了这些方程。然而，理解推导过程并不是很重要。

Putting it all Together

现在我们来讨论高斯函数在滤波中的应用。在下一章中，我们将使用高斯函数实现一个滤波器。这里我将解释为什么我们要使用高斯函数。

在前一章中，我们用数组表示概率分布。我们通过计算该分布与另一个代表每个点测量可能性的分布的元素级乘积来执行更新计算，如下所示:

In [34]:
```
def normalize(p):
    return p / sum(p)

def update(likelihood, prior):
    return normalize(likelihood * prior)

prior =      normalize(np.array([4, 2, 0, 7, 2, 12, 35, 20, 3, 2]))
likelihood = normalize(np.array([3, 4, 1, 4, 2, 38, 20, 18, 1, 16]))
posterior = update(likelihood, prior)
book_plots.bar_plot(posterior)
```

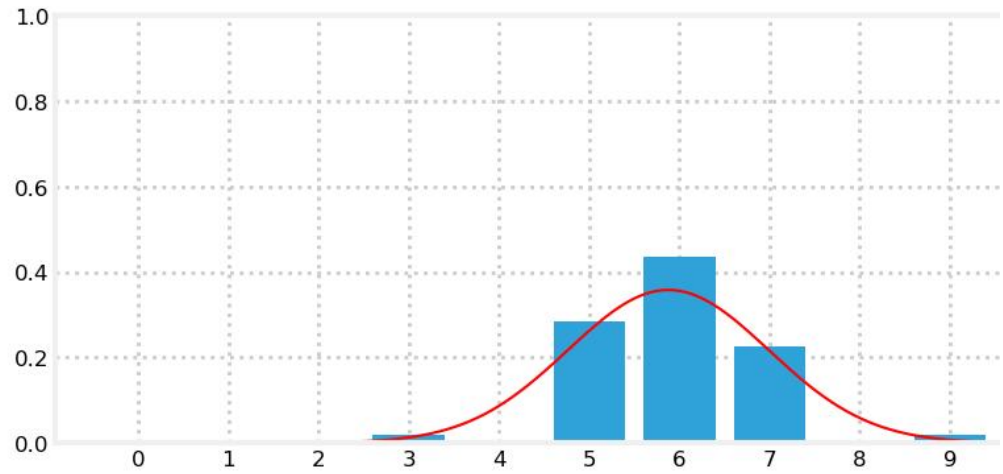


换句话说，我们需要计算 10 次乘法才能得到这个结果。对于一个具有多维数组的真正的过滤器，我们将需要数十亿次的乘法运算，以及大量的内存。

但这个分布看起来像高斯分布。如果我们用高斯函数代替数组呢?我将计算后验的均值和方差并将其绘制在柱状图上。

In [35]:
```
xs = np.arange(0, 10, .01)

def mean_var(p):
    x = np.arange(len(p))
    mean = np.sum(p * x,dtype=float)
    var = np.sum((x - mean)**2 * p)
    return mean, var

mean, var = mean_var(posterior)
book_plots.bar_plot(posterior)
plt.plot(xs, gaussian(xs, mean, var, normed=False), c='r');
print('mean: %.2f' % mean, 'var: %.2f' % var)
```

```
mean: 5.88 var: 1.24
```



这是令人印象深刻。我们可以用两个数来描述数字的整个分布。也许这个例子没有说服力，因为在这个分布中只有 10 个数字。但一个真正的问题可能有数百万个数字，但仍然只需要两个数字来描述它。

接下来，回忆一下我们的过滤器实现的更新函数

```
def update(likelihood, prior):
    return normalize(likelihood * prior)
```

如果数组包含一百万个元素，那就是一百万个乘法。但是，如果我们用高斯函数替换数组我们就可以用

$$\mu = \frac{\sigma_1^2\mu_2 + \sigma_2^2\mu_1}{\sigma_1^2 + \sigma_2^2}$$

$$\sigma^2 = \frac{\sigma_1^2\sigma_2^2}{\sigma_1^2 + \sigma_2^2}$$

也就是三乘二除。

Bayes Theorem

在上一章中，我们通过对每个时刻的信息进行推理，开发了一个算法，我们将这些信息表示为离散概率分布。在这个过程中，我们发现了贝叶斯定理。贝叶斯定理告诉我们如何在给定先验信息的情况下计算事件发生的概率。

我们使用这样的概率计算来实现 update()函数:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{normalization}}$$

这就是贝叶斯定理。我马上就会发展数学，但在很多方面，它掩盖了这个方程中表达的简单思想。我们将其解读为:

$$updated\ knowledge = \|likelihood\ of\ new\ knowledge \times prior\ knowledge\|$$

其中，$\|\cdot\|$表示术语的规范化。

我们得出这个结论的原因很简单:一只狗在走廊上走。然而，正如我们将看到的，同样的方程适用于所有的过滤问题。我们将在以后的每一章中使用这个方程。

回顾一下，先验是在我们纳入测量的概率(可能性)之前发生的事情的概率，后验是我们纳入测量的信息后计算的概率。

贝叶斯定理

$$P(A \mid B) = \frac{P(B \mid A)\, P(A)}{P(B)}$$

$P(A \mid B)$ 称为条件概率。也就是说，如果 B 发生，它代表 A 发生的概率。例如，如果昨天也下过雨，今天更有可能下雨，因为降雨系统通常会持续一天以上。我们可以把昨天下雨的情况写成푃(rain today | rain yesterday)。

我忽略了一个要点。在上面的代码中，我们处理的不是单一的概率，而是一系列的概率——一个概率分布。我刚才给贝叶斯的方程用的是概率，而不是概率分布。然而，它同样适用于概率分布。我们用小写 p 表示概率分布

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{p(B)}$$

在上面的公式中 B 是证据，p(A)是先验，p(B|A)是可能性，p(A|A)是后验。通过用相应的词替换数学术语，你可以看到贝叶斯定理与我们的更新方程相匹配。让我们根据问题重写一下这个方程。我们将使用 x_i 作为 i 的位置，并使用 z 作为测量。因此，我们想知道 $P(x_i \mid z)$，也就是说，狗在 x_i 的概率给出了测量 z。

我们把它代入方程，解出来。

$$p(x_i \mid z) = \frac{p(z \mid x_i)p(x_i)}{p(z)}$$

这看起来很难看，但实际上很简单。我们来看看右边的每一项是什么意思。首先是 $p(z \mid x_i)$。这是可能性，或者说是在每个细胞上测量的概率푆|푋。푋(푆|푋)是我们在纳入测量之前的信念。我们把它们相乘。这只是 update()函数中的非规范化乘法:

```
def update(likelihood, prior):
    posterior = prior * likelihood    # p(z|x) * p(x)
    return normalize(posterior)
```

要考虑的最后一项是分母 $p(z)$。这是得到测量的概率 Z 没有考虑到位置。它通常被称为证据。我们通过计算 X 的和，或者代码中的 sum(belief)来计算。这就是我们如何计算归一化的!因此，update()函数所做的只是计算贝叶斯定理。

文献中经常以积分的形式给出这些方程。毕竟，积分只是一个连续函数的和。你可能会看到贝叶斯定理写成

$$p(A \mid B) = \frac{p(B \mid A)\, p(A)}{\int p(B \mid A_j) p(A_j)\, \mathrm{d}A_j}.$$

这个分母通常不可能用解析法解出来;当它能被解决时，数学就变得极其困难。英国皇家统计学会(Royal Statistical Society)最近发表的一篇评论文章称其为“狗的早餐”[8]。采用贝叶斯方法的过滤教科书充满了没有解析解的满载积分的方程。不要被这些方程所吓倒，因为我们通过规范后验来处理这个积分。

我们将在粒子滤波器一章中学习更多处理这个问题的技术。在此之前，请认识到在实践中它只是一个标准化的项，我们可以对其求和。我想说的是，当你面对一页的积分时，就把它们看成求和，然后把它们与本章联系起来，通常困难就会消失。

问问自己“为什么我们要加这些值”，“为什么我要除以这一项”。令人惊讶的是，答案往往显而易见。令人惊讶的是，作者经常忽略提及这种解释。

很可能贝叶斯定理的力量还没有完全显现出来。我们想计算 $p(x_i \mid Z)$。也就是，在第 i 步，给出一个测量值，我们的可能状态是什么。这是一个非常难的问题。贝叶斯定理具有普遍性。根据癌症测试结果，我们可能想知道患癌症的概率，或者根据各种传感器读数，想知道下雨的概率。这样说，这些问题似乎无法解决。

但贝叶斯定理让我们可以通过使用逆 $p(Z \mid x_i)$,来计算这个值，这通常很容易计算:

$$p(x_i \mid Z) \propto p(Z \mid x_i)\, p(x_i)$$

也就是说，在给定特定传感器读数的情况下，要计算下雨的可能性，我们只需要在下雨的情况下计算传感器读数的可能性!这是一个简单得多的问题!嗯，天气预报仍然是一个困难的问题，但贝叶斯使它变得容易处理。

同样地，正如你在离散贝叶斯章节中看到的，我们通过计算传感器读数显示西蒙在 x 位置的可能性来计算西蒙在走廊任何给定部分的可能性。

Total Probability Theorem

现在我们知道了 update()函数背后的数学形式;那么 predict()函数呢?Predict()实现了全概率定理。让我们回顾一下 predict()计算了什么。根据所有可能的运动事件的概率计算出在任何给定位置的概率。我们用方程来表示。在 t 时间出现在任何位置 i 的概率可以写成 $P(X_i^t)$.。

我们将其计算为时间$t-1\ P(X_j^{t-1})$乘以从格子x_j to x_i的概率的总和。这是：

$$P(X_i^t) = \sum_j P(X_j^{t-1})P(x_i|x_j)$$

这个方程被称为全概率定理。引用维基百科[6]“它表达了一个结果的总概率，这个结果可以通过几个不同的事件实现”。我本可以给您这个方程并实现 predict()，但是您理解这个方程为什么有效的可能性很小。提醒一下，下面是计算这个方程的代码

```
for i in range(N):
    for k in range (kN):
        index = (i + (width-k) - offset) % N
        result[i] += prob_dist[index] * kernel[k]
```

Computing Probabilities with scipy.stats

在本章中，我使用了 FilterPy 中的代码来计算和绘制高斯函数。我这样做是为了让您有机会查看代码并了解这些函数是如何实现的。然而，正如俗话所说，Python 自带“电池”，它在 scipy.stats 模块中自带广泛的统计函数。所以让我们来看看如何使用 scipy。Stats 用来计算统计数据和概率。

Scipy.Stats 模块包含许多对象，您可以使用它们来计算各种概率分布的属性。此模块的完整文档在这里:http://docs.scipy.org/doc/scipy/reference/stats.html。我们将重点关注 norm 变量，它实现了正态分布。让我们来看一些使用 scipy.stats.norm 计算高斯函数的代码，并将其值与 FilterPy 中的高斯()函数返回的值进行比较。

```
In [36]: from scipy.stats import norm
         import filterpy.stats
         print(norm(2, 3).pdf(1.5))
         print(filterpy.stats.gaussian(x=1.5, mean=2, var=3*3))
```

```
0.13114657203397997
0.13114657203397995
```

调用 norm(2,3)创建了 scipy 所称的“冻结”分布—它创建并返回一个平均值为 2、标准差为 3 的对象。然后你可以多次使用这个对象来获得各种值的概率密度，就像这样:

```
In [37]: n23 = norm(2, 3)
         print('pdf of 1.5 is       %.4f' % n23.pdf(1.5))
         print('pdf of 2.5 is also  %.4f' % n23.pdf(2.5))
         print('pdf of 2 is         %.4f' % n23.pdf(2))
```

```
pdf of 1.5 is       0.1311
pdf of 2.5 is also  0.1311
pdf of 2 is         0.1330
```

scipy.stats.norm[2]的文档列出了许多其他函数。例如，我们可以使用 rvs()函数从分发版生成

```
In [38]: np.set_printoptions(precision=3, linewidth=50)
         print(n23.rvs(size=15))

[ 0.101  4.852  3.311  7.011  2.773  1.301  1.812
 -0.207  4.264 -3.22  -1.67   1.184  4.28   2.365
 -1.545]
```

我们可以得到累积分布函数(CDF)，这是从分布中随机抽取的值小于或等于 x 的概率。

```
In [40]: # probability that a random value is less than the mean 2
         print(n23.cdf(2))

0.5
```

我们可以得到分布的各种性质:

```
In [41]: print('variance is', n23.var())
         print('standard deviation is', n23.std())
         print('mean is', n23.mean())

variance is 9.0
standard deviation is 3.0
mean is 2.0
```

Limitations of Using Gaussians to Model the World

前面我提到了中心极限定理，它指出，在一定条件下，任何独立随机变量的算术和都是正态分布的，无论随机变量如何分布。这对我们很重要，因为自然界充满了非正态分布，但当我们将中心极限定理应用到大总体上时我们最终得到正态分布。

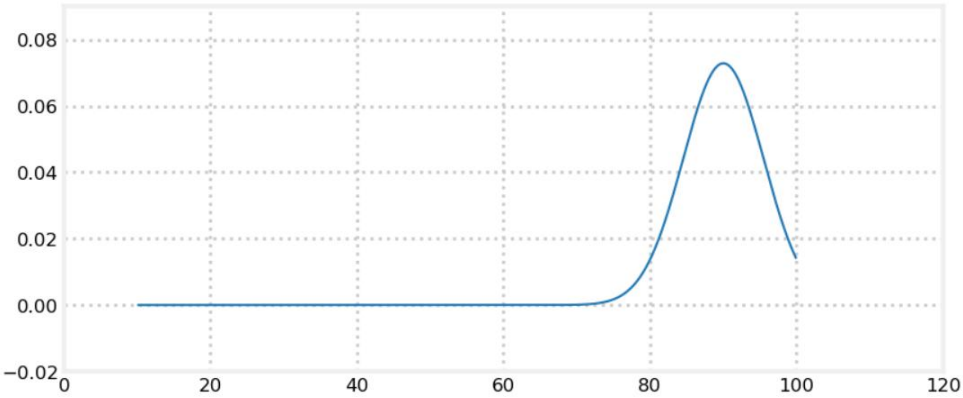
然而，证明的一个关键部分是“在一定条件下”。这些条件通常不适用于物质世界。例如，厨房秤的读数不能低于 0，但是如果我们用高斯函数表示测量误差，曲线左侧会延伸到负无穷远，这意味着给出负读数的可能性很小。

这是一个广泛的话题，我不会详尽地讨论。

让我们考虑一个简单的例子。我们认为考试分数是正态分布的。如果你曾经遇到过教授用曲线评分的情况，那么你就会受到这种假设的影响。当然，考试成绩不可能服从正态分布。这是因为，无论距离均值有多远，该分布都会为任何值分配非零概率分布。例如，均值是 90，标准差是 13。正态分布假设某个人得到 90 分的概率很大，某个人得到 40 分的概率很小。然而，它也意味着，有人得到-10 分或 150 分的几率很小。它分配了一个极小的机会得到-10^300 或 10^32986 分。高斯分布的尾部是无限长的。

但对于一个测试来说，我们知道这不是真的。不考虑额外学分，你的分数不能低于 0 或超过 100。让我们用正态分布来画出这个值的范围，看看它代表真实测试分数的分布有多差。

```
In [42]: xs = np.arange(10, 100, 0.05)
         ys = [gaussian(x, 90, 30) for x in xs]
         plt.plot(xs, ys, label='var=0.2')
         plt.xlim(0, 120)
         plt.ylim(-0.02, 0.09);
```



曲线下的面积不等于 1，所以它不是一个概率分布。实际发生的情况是，比正态分布预测的更多的学生的分数更接近范围的上端(例如)，而那个尾巴变成了“胖”。此外，测试可能无法完全区分学生们在技能上的细微差异，所以均值左边的分布可能也有点集中。

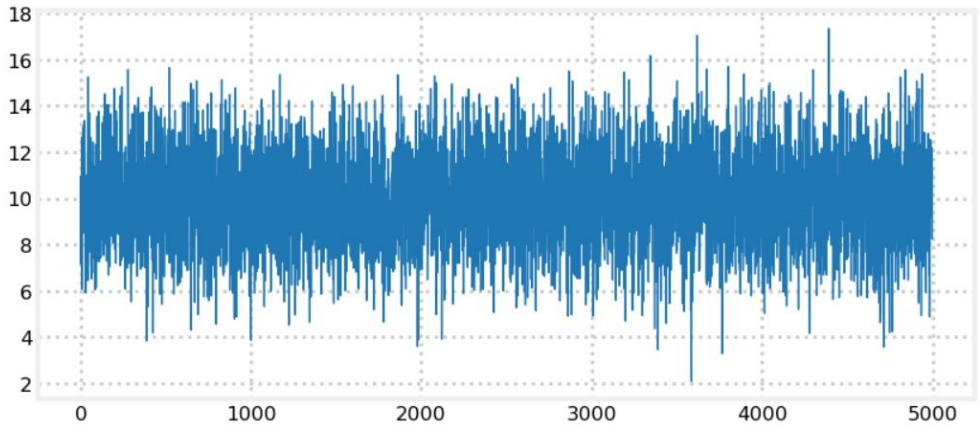
传感器测量世界。传感器测量的误差很少是高斯分布的。现在谈论这给卡尔曼滤波器设计者带来的困难还为时过早。值得记住的是，卡尔曼滤波数学是基于一个理想化的世界模型。现在，我将提供一些代码，我将在本书后面使用这些代码来形成模拟各种进程和传感器的分发版。这个分布被称为 *Student's t-distribution*.

假设我想建模一个输出有白噪声的传感器。为了简单起见，假设信号是常数 10，噪声的标准差是 2。我们可以使用 numpy.random.randn()函数来获得一个均值为 0、标准差为 1 的随机数。我可以用以下语句来模拟:

```
In [42]: from numpy.random import randn
         def sense():
             return 10 + randn()*2
```

让我们画出这个信号，看看它是什么样子的。

```
In [43]: zs = [sense() for i in range(5000)]
         plt.plot(zs, lw=1);
```

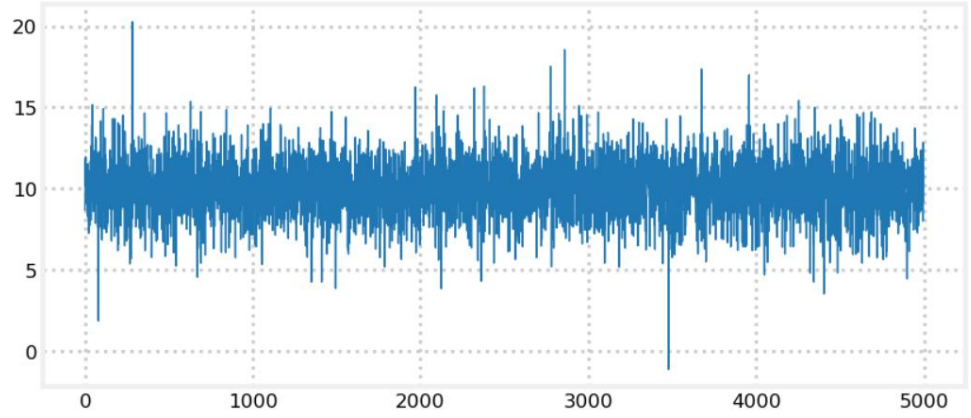


这跟我想的一样。信号集中在 10 左右。标准差为 2 意味着 68%的测量值在 2 到 10 之间，99%的测量值在 6 到 10 之间，这就是结果。

现在让我们看看用 Student 的 t-distribution 生成的分布。我就不细讲数学了，只给你它的源代码，然后用它绘制一个分布。

In [44]:
```
import random
import math

def rand_student_t(df, mu=0, std=1):
    """return random number distributed by Student's t
    distribution with `df` degrees of freedom with the
    specified mean and standard deviation.
    """
    x = random.gauss(0, std)
    y = 2.0*random.gammavariate(0.5*df, 2.0)
    return x / (math.sqrt(y / df)) + mu
```

In [45]:
```
def sense_t():
    return 10 + rand_student_t(7)*2

zs = [sense_t() for i in range(5000)]
plt.plot(zs, lw=1);
```



从图中我们可以看到，虽然输出类似于正态分布，但离均值(7 到 13)远超过 3 个标准差的异常值。

学生的 t-distribution 不太可能是你的传感器(比如 GPS 或多普勒)表现的准确模型，而且这不是一本关于如何建模物理系统的书。然而，当出现真实世界的噪声时，它会产生合理的数据来测试滤波器的性能。我们将使用分布像这些在我们的模拟和测试的书的其余部分。

这不是杞人忧天。卡尔曼滤波方程假设噪声是正态分布的，并执行次优化，如果这不是真的。任务关键滤波器的设计者，如航天器上的滤波器，需要掌握大量关于航天器上传感器性能的理论和经验知识。例如，我在 NASA 的一个任务中看到的一个演示中说，虽然理论上说他们应该用 3 个标准差来区分噪音和有效的测量，但实际上他们必须用 5 到 6 个标准差。这是他们决定的

rand_student_t 的代码包含在 filterpy.stats 中。你可以用它

```
from filterpy.stats import rand_student_t
```

虽然我不会在这里介绍它，但统计学已经定义了描述概率分布形状的方法，即它与指数分布的区别。正态分布对称分布在平均值周围，就像钟形曲线。然而，概率分布在平均值周围是不对称的。这种情况的测量方法叫做倾斜。尾巴可以缩短，更胖，更薄，或以其他方式形成不同于指数分布。这种度量叫做峰度。scipy。Stats 模块包含了计算这些统计信息的函数描述

```
In [46]: import scipy
         scipy.stats.describe(zs)

Out[46]: DescribeResult(nobs=5000, minmax=(-1.113328249202814, 20.265251933765818), mean=9.999719916590507, variance=2.671845621223685, skewness=
         0.005817070412044161, kurtosis=1.5862956949751155)
```

Let's examine two normal populations, one small, one large:

```
In [48]: print(scipy.stats.describe(np.random.randn(10)))
         print()
         print(scipy.stats.describe(np.random.randn(300000)))

         DescribeResult(nobs=10, minmax=(-1.6067574452018323, 0.9450912932696207), mean=0.061069946645338914, variance=0.5378488482745439, skewne
         ss=-1.0245852382217775, kurtosis=0.7347742232945969)

         DescribeResult(nobs=300000, minmax=(-4.574191271496192, 4.829896954300753), mean=-0.0034476193682445757, variance=0.9986850806211145, sk
         ewness=-0.0015183684651360612, kurtosis=0.000260816792021501)
```

小样本的偏度和峰度不为零，因为小样本的数量没有很好地分布在 0 的均值附近。通过将计算的平均值和方差与 0 和方差 1 的理论平均值进行比较，您也可以看到这一点。相比之下，大样本的均值和方差都非常接近理论值，偏度和峰度都接近于零。

Summary and Key Points

这一章对一般的统计是一个很差的介绍。在本书的其余部分中，我只介绍了使用高斯函数所需的概念。如果您打算阅读卡尔曼滤波文献，那么我所介绍的内容不会让您了解很多。如果这对你来说是一个新话题，我建议你读一本统计学教科书。我一直很喜欢 Schaum 的自学系列，Alan Downey 的《Think Stats[5]》也很不错，可以在网上免费获得。

在我们继续之前，您必须了解以下几点:

- Normals express a continuous probability distribution
- They are completely described by two parameters: the mean (μ) and variance (σ^2)
- μ is the average of all possible values
- The variance σ^2 represents how much our measurements vary from the mean
- The standard deviation (σ) is the square root of the variance (σ^2)
- Many things in nature approximate a normal distribution, but the math is not perfect.
- In filtering problems computing $p(x \mid z)$ is nearly impossible, but computing $p(z \mid x)$ is straightforward. Bayes' lets us compute the former from the latter.