Introduction to Probability

This chapter approaches the topic of probability by considering outcomes that are uncertain but nevertheless reflect a well defined underlying structure. This structure is a probability distribution. A probability *experiment* produces an outcome that is governed by a probability distribution. The outcome of a probability experiment is one of a fixed set of outcomes, which have stable probabilities of occurrence. A probability experiment produces a *draw* from a probability distribution.

A simple example of a probability experiment is flipping a fair coin and recording the outcome as heads or tails. Another example is rolling a fair standard die and recording the outcome as the number of face up pips.

Random Variates

Given knowledge of the generating process, point predictions of fully deterministic outcomes are error free, but point predictions of random outcomes may prove to be erroneous. This chapter illustrates this distinction by introducing basic concepts in probability and statistics, along with a few related computational tools.

Pseudorandom Outcomes

Scientific programming languages typically include facilities for the generation of numbers that appear to be truly random even though they are generated algorithmically. Since the algorithm producing the numbers is completely deterministic, the generated values are *pseudo-random*. For concision, this book nevertheless typically refers to them as random numbers, produced by random-number generators (RNGs).

Seeding the RNG

From a given initial state, a random-number generator (RNG) produces a determinate sequence of values. This initial state may be determined by a *seed* for the RNG. Truly replicable computational results require explicitly seeding the RNG. The **SeedRandom** command takes one argument (the seed) and initializes the random-number generator. All WL functions that return pseudo-random values are affected by **SeedRandom** and by any subsequent generation of pseudo-random values. The following example illustrates the use of **RandomChoice** to select one choice from a list of alternatives; the particular seed is arbitrary.

Out[ ]=

Expression	Result
`alternatives=Range[1000];`	
`SeedRandom[314159];RandomChoice[alternatives]`	727
`RandomChoice[alternatives]`	696
`SeedRandom[314159];RandomChoice[alternatives]`	727

A caveat is in order. Any expression whose value depends on the program state requires care in use. It involve a loss of referential transparency, correspondingly making it harder to test program code for correctness. Nevertheless, such expressions are often used in scientific programming. Specifically, it is common to rely on random number generators whose outputs depend on a hidden global state (such as the random seed and the number of calls to the RNG). With this caveat in mind, this chapter follows the common practice.

Coin Flipping

To model a coin toss with the **RandomChoice** command, let **h** denote an outcome of heads, and let **t** denote an outcome of tails. From the list {h,t}, the **RandomChoice** command picks heads or tails with equal likelihood. Provide a positive integer as a second argument to produce a list of such outcomes.

Out[ ]=

Expression	Result	Comment
`RandomChoice[{h,t}]`	t	one coin toss
`RandomChoice[{h,t},3]`	{h, t, t}	three coin tosses

Random numbers can help highlight the difference between **Set** and **SetDelayed**, covered in Chapter 1. Consider repeatedly tossing a coin. To represent an experiment, the value of **toss** should be an independent coin toss. Using **Set** to bind a variable to a **RandomChoice** outcome immediately evaluates the right-hand side and assigns the value produced to **toss**, which then evaluates to that single value every time. However, using **SetDelayed**, instead ensures that **toss** generates a new random choice each time it is evaluated.

Out[ ]=

Expression	Result	Comment
`toss=RandomChoice[{h,t}]`	t	use of Set
`Table[toss,5]`	{t, t, t, t, t}	identical items
`toss:=RandomChoice[{h,t}]`		use of SetDelayed
`Table[toss,5]`	{h, t, t, h, t}	varied items

Subexperiments and Trials

A probability experiment produces a random outcome from a probability distribution. Sometimes an experiment can be decomposed into independent subexperiments. For example, consider a coin flip along with an unrelated roll of a three-sided die. This single experiment may be decomposed into two independent subexperiments. Use the **Tuples** command to represent the sample space of the experiment as the direct product of the sample spaces of the two subexperiments: in this case, **Tuples[{{h,t},{1,2,3}}]**.

Sometimes, subexperiments share a common description. For example, decompose an experiment involving *n* identical coin flips into *n* independent subexperiments of one coin flip, or decompose a roll of *n* identical dice into *n* independent subexperiments of one roll each. In these two example experiments, the subexperiments are called *trials*, which implies that each has an identical description. The sample space of the overall experiment is the direct product of the sample spaces of the trials. For example, a coin-flipping experiment with *n* trials has 2^n possible sequences as outcomes, each of which may be represented by an ordered *n*-tuple. The **Tuples** command can again produce the entire sample space, and since each trial is identical, its two-argument form is particularly convenient: **Tuples[{h,t},n]** produces the desired result for any *n*. (But as always, remember that 2^n grows much more quickly than *n*.)

Random Variables and Random Variates

A *random variable* maps random outcomes to real numbers. This standard meaning may cause confusion: a random variable is defined to be a function. There is nothing random about the function itself: it is a fixed rule for turning sample-space inputs into numerical outputs. Any randomness in the function values is therefore entirely attributable to randomness in the underlying outcome. (For example, the outcome of a fair coin flip.) To add to the potential terminological confusion, the realized function values are called *random variates*.

These concepts are best illustrated by example. In the coin-flipping example, it is traditional to map the outcome **h** to 1 and the outcome **t** to 0. Flipping a coin produces a random outcome (**h** or **t**), which is then mapped to a real number (**1** or **0**). The resulting real number is called a *random variate*, or a *draw* from the probability distribution, or a *realization* of the random variable. The following example defines this function with an association and then maps it across a sequence of outcomes to produce a sequence of random variates. (Recall that **/@** is the shorthand operator notation for **Map**.)

Out[•]=

Expression	Result
`outcomes = RandomChoice[{h,t},5]`	`{t, h, t, t, h}`
`variates = <|h→1,t→0|> /@ outcomes`	`{0, 1, 0, 0, 1}`

Coin flip simulations do not typically bother with the intermediate representation of the outcomes. Instead, the variates are directly produced. Consider the following model of flipping an unfair coin. Using **RandomChoice**, the first argument can be a rule that provides weights that determine the odds of each outcome. For example, suppose a head should only occur about 40 percent of the time. This can be implemented with any weights that imply this relative frequency. (The exact number of heads in an experiment will of course be random; we will come back to this.) In the following example, an experiment with 100 trials produces data, and applying the **Counts** command to the data helps to determine whether the results of the experiment approaches our expectations.

Out[∘]=

Expression	Result
`unfairToss:=RandomChoice[{40,60}→{1,0}]`	
`variates=Table[unfairToss,100];`	
`Counts[variates]`	⟨|0 → 62, 1 → 38|⟩

This is a fine approach to generating these random variates. However, it is somewhat more natural in WL to specify a probability distribution and then use the **RandomVariate** command to generate the needed variates. One simple way to define a distribution with finite support is with the **EmpiricalDistribution** command. The result is a data-distribution object, as illustrated in the following example.

Out[∘]=

Expression	Result
`edist01=EmpiricalDistribution[{60,40}→{0,1}];`	
`RandomVariate[edist01,5]`	{1, 0, 0, 1, 0}

Probability

A random variable has a range of possible values. The underlying probability distribution determines the probability that the function will produce values in this range. These probabilities are summarized by a *cumulative distribution function* (CDF), which for each real number *x* returns the probability that a random variable will take on a value no bigger than *x*. Letting *X* be represent a random variable, characterize the associated cumulative distribution function F_X as follows.

$$F_X = x \longmapsto P[\{\omega \mid X[\omega] \leq x\}] \tag{1}$$

That is, for an real number *x*, a CDF returns the probability of an outcome *ω* that the function *X* produces a value no bigger than *x*, given the underlying probability distribution of outcomes. Later sections provide concrete examples of CDFs and some uses for them.

Probability and Counting

A *sample space* is the set of all possible distinguishable outcomes. A random outcome cannot be known with certainty before it is observed, but it must occur in the sample space. For example, characterize the sample space for a coin flip as {*h*, *t*} for heads or tails. This means that a coin flip is always resolved as heads or tails and never anything else (such as standing on edge or being snatched in midair by a passing bird). Similarly, characterize the sample space for a roll of a standard six-sided die as the set of possible pip counts, {1, 2, 3, 4, 5, 6}.

An *event* is any subset of the sample space. For example, consider the event of an even roll of a six-sided die, characterized as {2, 4, 6}. An event may be a singleton, containing a single outcome, in which case it is an *elementary event*. Events are *mutually exclusive* if they are pairwise disjoint. For example, for the roll of a die, the even roll event and the odd roll event are mutually exclusive.

Events have associated probabilities. A mapping from events to probabilities is a *probability distribution function*. A probability function *P* is nonnegative, normalized, and additive. The following statement of this adds a little detail, given a sample space Ω and a collection of events E_i.

- nonnegative definite: $0 \le P[E_i]$
- normalized: $P[\Omega] == 1$
- countably additive: $P[\bigcup_{i=1}^{\infty} E_i] == \sum_{i=1}^{\infty} P[E_i]$ whenever the events E_i are mutually exclusive

The last property is often called σ-additivity. Since the null set is disjoint from any other set (including itself), σ-additivity implies that $P[\Omega] = P[\Omega] + \sum_{i=2}^{\infty} P[\emptyset]$, implying that $P[\emptyset] == 0$. Similar reasoning implies finite additivity: the union of any finite number of mutually exclusive events has a probability equal to the sum of the individual event probabilities. In particular, for any event E, $1 == P[\Omega] == P[E \cup E^c] = P[E] + P[E^c]$, so we know that $P[E^c] == 1 - P[E]$. Furthermore, similar reasoning produces a monotonicity property: $E_1 \subseteq E_2 \Rightarrow P[E_1] \le P[E_2]$. Simply put, an event the includes additional outcomes cannot have lower probability.

Recall that a partition of a set is a set of non-empty, pairwise disjoint sets whose union is the original set. Let $\{E_i\}_{i=1}^{n}$ be a partition of the sample space Ω, and consider an arbitrary event E. Then $\bigcup_{i=1}^{n} (E \cap E_i)$ is a union of pairwise disjoint sets that is equal to E. Applying the additivity of the probability distribution function produces the *law of total probability*:

$$P[E] = \sum_{i=1}^{n} P[E \cap E_i] \qquad (2)$$

Consider a finite sample space (Ω) with cardinality N, and consider the special case where each outcome is equally likely. This means that the probability of any individual outcome ω_i is $1/N$, since distinct elementary events are mutually exclusive. The following equation illustrates this, following the standard convention that $P[\omega_i]$ means $P[\{\omega_i\}]$, the probability of a singleton.

$$1 == P[\Omega] = P\left[\bigcup_{n=1}^{N} \{\omega_n\}\right] = \sum_{n=1}^{N} P[\omega_n] = \sum_{n=1}^{N} P[\omega_1] == N * P[\omega_1] \qquad (3)$$

On a finite sample space with equiprobable outcomes, the probability of an event E is the proportion of the possible outcomes in the event: $P[E] = |E| / |\Omega|$. This links probability to counting. As a concrete example, suppose two students are selected randomly from a class of three men and three women. Assuming each student is equally likely to be selected, compute the probability that both are male. For this example, the sample space is the set of unordered pairs of students. Recall that there are $15 == C[6, 2]$ different ways to pick two students from a class of six; this is the size of the sample space. Similarly, there are $3 == C[3, 2]$ ways to pick two students from the group of men. So the probability is $1/5 == 3/15$ that both students are male. (By symmetry, this is also the probability that both are female.)

Out[ ]=

Expression	Result	Comment
npairs=Binomial[6,2]	15	count all pairs
mpairs=Binomial[3,2]	3	count male pairs
mpairs/npairs	1/5	P[two males]

Conditional Probability and Independence

An event is a subset of the sample space, which is the set of possible outcomes. If event E_1 provides no information about the likelihood of event E_2, the two events are *independent*. In this case, knowing that E_1 occurred does not help to determine whether E_2 occurred. This independence is sometimes written as $E_2 \perp E_1$. If E_1 is uninformative about E_2 then it must also be uninformative about E_2^c, so E_1 and E_2^c must also be independent.

By definition, E_1 and E_2 are independent if the probability that both occur is the product of their probabilities: $P[E_1 \cap E_2] == P[E_1] \times P[E_2]$. To illustrate the independence of events, consider an experiment comprising two independent coin flips. Let E_1 be the event that the first flip comes up heads. Let E_2 be the event that the second flip comes up heads. Let p be the probability of a flip coming up heads. Since the flips are independent, the probability that both come up heads is p^2.

In contrast, sometimes knowing that one event (E_1) has occurred is informative about whether another event (E_2) has occurred. Write the probability of E_2 conditional on knowing E_1 as $P[E_2 \mid E_1]$, defined as follows: scale up the probability of the joint event (i.e., that both occurred) by taking into account that E_1 occurred.

$$P[E_2 \mid E_1] = P[E_2 \cap E_1] \,/\, P[E_1] \tag{4}$$

Note that we can partition the event E_2 into two mutually exclusive blocks:
$P[E_2] = P[(E_2 \cap E_1) \cup (E_2 \cap E_1^c)] = P[E_2 \cap E_1] + P[E_2 \cap E_1^c]$. Applying the definition of conditional probability then produces $P[E_2] = P[E_2 \mid E_1] \times P[E_1] + P[E_2 \mid E_1^c] \times P[E_1^c]$.

If the two events are mutually exclusive, then observing E_1 is very informative: E_2 cannot have happened. For example, if the roll of a die is even, it cannot be odd. However, if two events are independent, then knowing one occurred is uninformative about whether the other occurs. In the case of independence, it follows from the definition of independence and of conditional probability that $P[E_1 \mid E_2] = P[E_1]$ and $P[E_2 \mid E_1] = P[E_2]$.

The following figure illustrates these considerations by assuming that outcomes fall uniformly in an interval Ω. Events are displayed visually by offsetting them from the sample space. This figure illustrates four events: $P[A] = 1/2$, $P[B] = 1/3$, and $P[C] = P[D] = 1/4$. Because of the way they overlap, event A is very informative about event C. Knowing that A occurred doubles the likelihood that C occurred: $P[C \mid A] = P[C \cap A]/P[A] = (1/4)/(1/2) = 1/2$. And, because A and D are mutually exclusive, observing A is very informative about D: it tells us that D must not have occurred. However, even though A and B overlap, knowing that A occurred is not informative about the likelihood that B occurred. That is, $P[B \mid A] = P[B \cap A]/P[A] = (1/6)/(1/2) = 1/3$, which equals $P[B]$.

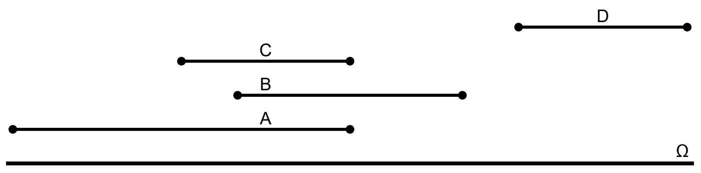
Out[∘]=


Figure 14: Intersection and Conditional Probability

Conditioning on E_1 treats the occurrence of E_1 as given. As implied by the definition, $P[E_1 \mid E_1] == 1$. Nonnegative definiteness and σ-additivity of conditional probability are implied by the unconditional probability distribution, since $P[E_2 \cap E_1] \geq 0$ and since intersection distributes over union. In short, conditional probability is effectively a probability measure on a restricted sample space.

As a simple example of independence, consider a single roll of a fair die. Let $E_1 = \{1, 3, 5\}$ and $E_2 = \{5, 6\}$, so that $P[E_1] = 1/2$ and $P[E_2] = 1/3$. Using the definition of independence, confirm that these two events are independent: the probability of both occurring is $1/6 = (1/2)\,(1/3)$, which is the probability of rolling a 5. A two-way relative frequency table exposes the independence of the events E_1 and E_2. For example, knowing whether a roll was in E_1 does not change the likelihood that it was in E_2. Specifically, a third of the time that E_1 occurs then E_2 also occurs, but it is also true that a third the time that E_1^c occurs then E_2 also occurs. Conditioning on whether or not E_1 occurred does not reduce the uncertainty about whether or not E_2 occurred.

One way of visualizing this is with a two-way relative frequency table, which displays the relative frequencies of the four joint events. An outcome is either in E_1 or E_1^c (odd or even), and it is either in E_2 or E_2^c ({5, 6} or {1, 2, 3, 4}). By the product rule of counting, there are 4 different possible joint events. A two-way relative frequency table exposes the independence of the events E_1 and E_2. For example, knowing whether a roll was odd does not change the likelihood that it was either 5 or 6.

Out[∘]=

Table 2: Fair-Die Outcomes and Joint Probabilities (E2={5,6})

	E_2	E_2^c
odd	{5}	{1, 3}
even	{6}	{2, 4}

⇒

	E_2	E_2^c
odd	1/6	1/3
even	1/6	1/3

In contrast, knowing that a roll of a six-sided odd is even provides information about whether the outcome is prime. These two events are not independent. There is only one even prime, which has an unconditional probability of 1/6. The unconditional probability of rolling a prime is $P[\{2, 3, 5\}] = 3/6 = 1/2$, but the probability that a roll is prime given that it is odd is the conditional probability $P[\{2, 3, 5\} \mid \{1, 3, 5\}] = P[\{3, 5\}]/P[\{1, 3, 5\}] = (2/6)/(3/6) = 2/3$. Again, approach this by means of a two-way relative frequency table. An outcome is either in E_1 or E_1^c (odd or even), and it is either in E_2 or E_2^c (prime or not prime). As usual, applying the product rule of counting, that gives us 4 different possible joint events. Conditioning on whether or not the rolls was odd in informative (but not determinative) about whether it was prime.

Out[∘]=

Table 21: Outcome Lists and Joint Probabilities for a Fair Die

	prime	not prime
odd	{3, 5}	{1}
even	{2}	{4, 6}

⇒

	prime	not prime
odd	1/3	1/6
even	1/6	1/3

When more than two events involved, the concept of independence becomes more complicated. Independence is not a transitive relation: does not get passed along a chain of events. That is, independence of events A and B along with independence of events B and C does not imply that A and C are independent. The following figure illustrates this.

Out[∘]=

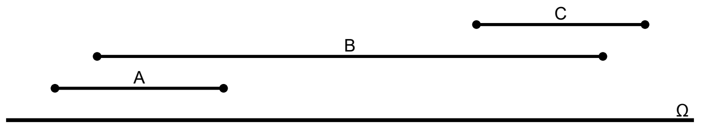


Figure 58: Independence Is Not Transitive

In this figure, events A and B are independent, and events B and C are also independent. Yet A is not independent of C. Additionally, A is not even independent of $B \cap C$. This suggests that pairwise independence is not adequately informative about larger collections of events. The extended concept for any collection of events $\{E_1, \ldots, E_N\}$ is mutual independence: they are *mutually independent* if for every subset of this collection the probability of the joint event (i.e., that all occurred) equals the product of the unconditional probabilities of the events.

Bayes’s Rule

The probability of event E_2 conditional on event E_1 is $P[E_2 \mid E_1] = P[E_2 \cap E_1]/P[E_1]$. Symmetrically, the probability of event E_1 conditional on event E_2 is $P[E_1 \mid E_2] = P[E_2 \cap E_1]/P[E_2]$. Together these imply Bayes’s Rule:

$$P[E_2 \mid E_1] = P[E_1 \mid E_2] \times P[E_2] \,/\, P[E_1] \tag{5}$$

The law of total probability and the definition of conditional probability imply that $P[E_1] = P[E_1 \mid E_2] \times P[E_2] + P[E_1 \mid E_2^c] \times P[E_2^c]$. Correspondingly, Bayes’s Rule is often written in the following second form.

$$P[E_2 \mid E_1] = P[E_1 \mid E_2] \times P[E_2] \,/\, (P[E_1 \mid E_2] \times P[E_2] + P[E_1 \mid E_2^c] \times P[E_2^c]) \tag{6}$$

As an application, consider automated auditing of a companies for possible fraud. If a company’s accounts are fraudulent, then 90% of the time the audit software generates an alert: $P[\text{alert} \mid \text{fraud}] = 0.90$. Unfortunately, the audit software also generates a fraud alert 10% of the time when there is no fraud: $P[\text{alert} \mid \neg\,\text{fraud}] = 0.10$. Fortunately, fraud is rather rare, occurring in only 1% of all firms: $P[\text{fraud}] = 0.01$. Compute the probability of fraud given a fraud alert using the second form of Bayes’s Rule. Note that $P[\text{alert} \mid \text{fraud}] \times P[\text{fraud}] = 0.90 \times 0.01 = 0.09$. Also, $P[\text{alert} \mid \neg\,\text{fraud}] \times P[\neg\,\text{fraud}] = 0.10 \times 0.99 = 0.099$. Applying the second form of Bayes’s Rule produces $P[\text{fraud} \mid \text{alert}] = 0.09/(0.09 + 0.099) \approx 0.48$. So despite the rather good performance of the fraud detection software, an alert indicates actual fraud less than half of the time.

Empirical Distributions and Statistics

Empirical work often scrutinizes the distribution of observed outcomes. When these outcomes manifest randomness with an underlying stable structure, researchers often characterize the outcomes as being drawn from a probability distribution. An empirical distribution summarizes the data in a way that sheds light on the underlying probability distribution.

A Coin-Flipping Example

When applied to a list of observations, the **EmpricialDistribution** command produces a data-distribution object, which provides access to the empirical distribution. As an example, produce 100 outcomes with the **unfairToss** function, defined earlier, and then produce an associated data-distribution object.

Out[∘]=

Expression	Result

data = Table[unfairToss, 100];
edist = EmpiricalDistribution[data]
	DataDistribution[Type: Empirical Data points: 100]

When there are only a countable number of possible outcomes, the distribution is *discrete*. For example, a coin-flipping distribution has only two possible outcomes, so it is a discrete distribution. This distribution may be parameterized by the probability of drawing a head.

Draw random variates from a probability distribution produces information about the distribution. The empirical probability density function (PDF) of a discrete distribution is simply a summary of the observed relative frequencies of each of the outcomes. (This is often called a probability mass function.) The empirical cumulative distribution function (CDF) of a discrete distribution simply accumulates these relative frequencies after sorting on the values of the outcomes. The empirical PDF and empirical CDF can provide useful nonparametric characterizations of the underlying distribution. An empirical distribution gives us access to both of these. Recalling that **Boole** converts **False** and **True** to 0 and 1, let us take a look at the PDF and CDF implied by our data.

Out[∘]=

Expression	Result
PDF[edist,x]	(16*Boole[0 == x])/25 + (9*Boole[1 == x])/25
CDF[edist,x]	(16*Boole[0 <= x])/25 + (9*Boole[1 <= x])/25

Charts can help us visualize these functions. The following charts utilize the **DiscretePlot** command to create charts from the empirical distribution. This produces a nice stick-pin plot of the PDF. For our plot of the CDF, we can achieve an easy and clear designation of the function value at the points of discontinuity by setting the **ExtentSize** and **ExtentMarkers** options as follows. (For convenience, we again assign a sequence of options that are shared across charts to an **options** variable.)

Out[∘]=

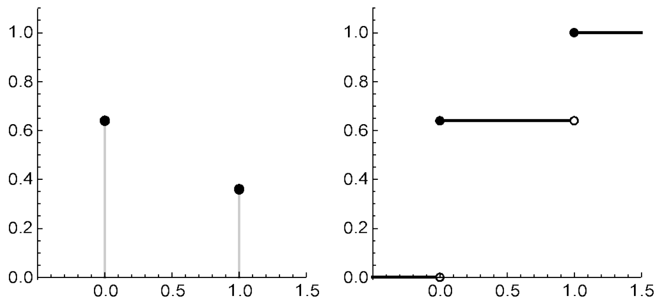


Figure 2: PDF and CDF of Empirical Distribution

The adequacy of the empirical distribution in representing the underlying distribution depends on the quality and quantity of the underlying data. When the number of possible outcomes is small compared to the number of draws, the data are likely to include every possible outcome. However, it is possible that that data will include no outcomes for some values that have a small but positive probability of occurring, especially with a small number of observations. The missing values will receive zero weight in the empirical distribution.

Statistics

A statistic is a number that summarizes some feature of a dataset. To sample a population or a distribution is to collect observations from it. A sample statistic is therefore a number that summarizes some feature of a sample. Two important sample statistics are the sample mean and the sample variance.

After sampling a probability distribution, produce a sample statistic by applying a function to the resulting collection of random variates. The function itself is also often termed a statistic. Producing a sample statistic creates a random variate, with properties that depend on the underlying probability distribution. This means that sample statistics may shed light on the underlying probability distribution. For example, the sample mean and sample variance might be used to draw inferences about the mean and variance of the underlying probability distribution.

The data-distribution object produced by the **EmpiricalDistribution** command provides access to many sample statistics, including the sample mean and variance. To see this, first apply **Mean** to the empirical distribution, and then recall that the sample mean deflates the sum of the observations by the number of observations: $\overline{X} = \sum_{n=1}^{N} X_n / N$. Applying the **Mean** command directly to the data produces the same result.

```
Out[∘]=    Expression                                  Result
           mean = Mean[edist]                          9/25
           mean == Total[data] / Length[data]          True
```

Next, apply the **Variance** command to the empirical distribution, and then recall that the (unadjusted) variance of a sample deflates the sum of the squared deviations of observations from their mean by the number of observations: $\text{var}[X] = \sum_{n=1}^{n} \left(X_n - \overline{X}\right)^2 / N$.

```
Out[∘]=    Expression                                       Result
           var = Variance[edist]                            144/625
           var == Total[(data-mean)^2] / Length[data]       True
```

However, when applied directly to the data, the **Variance** command includes a Bessel correction. When used as an estimate of the population variance, this correction can reduce the bias while increasing the mean squared error. Although it is fairly common to refer to the result as the sample variance, it is less ambiguous to refer to it as the corrected sample variance.

Out[∘]=

Expression	Result
nobs = Length[data]	100
Variance[data] == var*nobs/(nobs-1)	True

The sample mean and sample variance are useful descriptive statistics: they help describe salient properties of the data. In addition, they often provide more than description. They provide information about the underlying probability distribution—information that can guide inferences about the process generating the data.

Test Statistics

A test statistic is a sample statistic used to provide information about a probability distribution. A test statistic may be used to test hypotheses about the process that generated a dataset. For example, consider the hypothesis that a coin is fair. Under this hypothesis, the expectation is that half of the flips will come up head and half tails. More generally, if K different categories have associated probabilities p_k, then the expectation is that in N draws there should be $N\,p_k$ outcomes in category k. Actual data naturally differ from this expectation due to randomness, but big differences should be rare.

When we encounter data that our beliefs imply should occur very rarely, the data provides evidence against those beliefs. A statistical test attempts to quantify this idea. For example, **data** holds the outcomes of repeatedly flipping a simulated coin, and **Counts** produces an association from each outcome to its frequency. (The use of **KeySort** just ensures that the outcomes are in a natural order.) Consider the hypothesis that this is a fair coin; call this the null hypothesis. An association to summarize this null hypothesis by mapping each outcome to its expected frequency. The beliefs embodied in the null hypothesis deviate from the observed outcomes. Produce these deviations. (Subtraction of associations produces the differences of the values if the keys match.)

Out[∘]=

Expression	Result
observedFrequency=KeySort@Counts[data]	〈|0 → 64, 1 → 36|〉
beliefs=<|0→1/2,1→1/2|>;	
expectedFrequency=Length[data]*beliefs	〈|0 → 50, 1 → 50|〉
deviations=observedFrequency-expectedFrequency	〈|0 → 14, 1 → -14|〉

Of course, such deviations can happen just by chance. How unlikely are deviations of this size if the null hypothesis is correct? The following test statistic is attributed to Karl Pearson. (Again, the total is computed across the values, not across the keys.)

Out[∘]=

Expression	Result
pearson = Total[deviations^2 / expectedFrequency]	196/25

In order to use this statistic for testing, we need information about its distribution. Pearson showed that if the null hypothesis is true, then the distribution is approximately Chi-square with degrees of freedom equal to the number of categories reduced by 1. (That is two categories, in the present case, reduced by 1 because—as we can see by examining the value of our **deviations** variable—the deviations are constrained to sum to 0.) A later section discusses the Chi-square distribution, which underpins many statistical tests. For now, just apply the **PearsonChiSquareTest** command to two arguments: the data under consideration, and a distribution representing our null hypothesis. (Use the **EmpiricalDistribution** command to provide the distribution for the null hypothesis.) This command reports the probability (under the null hypothesis) of deviations as big or bigger than those observed. Under the null hypothesis of a fair coin, we should encounter such data in only one such experiment out of every two hundred. These data provide evidence against the null hypothesis.

Out[∘]=

Expression	Result
`PearsonChiSquareTest[data, EmpiricalDistribution[{0.5,0.5}→{0,1}]]`	0.00511026

Distributions Related to the Binomial

This section introduces probability distributions that have finite sample spaces. Coin flipping and die rolling are classic examples: the sample space for a coin flip has only two elements, and the sample space for a six-sided die roll has six elements. Finite sample spaces often represent conceptual categories. Social science often categorizes outcomes and therefore works with distributions that are fundamentally categorical. Letting **ncats** denote the number of categories and **ntrials** denote the number of trials, the following table names a few distributions of particular interest.

Out[∘]=

Multinomial Distributions

	ntrials=1	ntrials>1
ncats=2	Bernoulli	Binomial
ncats>2	Categorical	Multinomial

Bernoulli and Binomial Distributions

A probability function with a finite sample space may be naturally represented by a finite map from possible outcomes to their probabilities. Recall from Chapter 2 that WL provides the **Association** data type for the representation of finite maps. For example, **<|t→0.6,h→0.4|>** is one way to characterize the probability distribution for a flip of an unfair coin, where a coin flip is expected to turn up tails 60% of the time.

Out[∘]=

Expression	Result
`pf=<|t→0.6,h→0.4|>;`	
`pf[t]`	0.6
`pf[h]`	0.4

The Bernoulli and binomial distributions are based on random outcomes that fall in one of two categories, traditionally illustrated by a tails-or-heads outcome of a coin flip. These categories are traditionally called failure and success, and these distributions assign them corresponding values of 0 or 1. The probability that an outcome will be success, traditionally designated by *p*, parameterizes the distributions. The Bernoulli distribution characterizes a single such outcome (e.g., a single flip of a possibly unfair coin). The binomial distribution characterizes a collection of such outcomes (e.g., 10 flips of a coin). Evidently, the Bernoulli distribution is a special case of the binomial distribution.

Bernoulli Distribution

The Bernoulli distribution is the simplest example of a discrete probability distribution. There are only two possible outcomes: 1 (success) or 0 (failure). The distribution is governed by a single parameter: the probability of drawing a 1 instead of a 0. Here are three examples of creating a Bernoulli distribution with the **BernoulliDistribution** command. Perhaps surprisingly, a symbolic value is completely acceptable.

Use the RandomVariate command to draw random variates from a distribution.

Out[ ]=

Expression	Result
`RandomVariate[bern50,10]`	`{0, 0, 0, 0, 0, 1, 0, 1, 0, 0}`
`RandomVariate[bern60,10]`	`{1, 1, 1, 1, 0, 0, 1, 1, 0, 1}`

Naturally, this is not possible if the parameter is symbolic. Nevertheless, symbolic manipulations of probability distributions are remarkably powerful. For example, we can produce a symbolic representation of the PDF and CDF for a symbolic probability parameter. (Interpret the domain values of **True** to be all those not otherwise specified.)

Out[ ]=

Expression	Result
`PDF[bernp, x]`	$\begin{cases} 1-p & x == 0 \\ p & x == 1 \\ 0 & \text{True} \end{cases}$
`CDF[bernp, x]`	$\begin{cases} 0 & x < 0 \\ 1-p & 0 \le x < 1 \\ 1 & \text{True} \end{cases}$

The PDF tells us that, for a single draw from the Bernoulli distribution, the probability is (1 – *p*) of drawing a 0 and *p* of drawing a 1. All other values have zero probability of occurring. Correspondingly, the CDF says that drawing a value less than 0 is impossible, while drawing a value less than or equal to 1 is certain.

The plots for a concretely parameterized PDF and CDF are correspondingly simple. The following chart plots the PDF and CDF of the Bernoulli distribution with a probability of success of 0.6. The plot of the PDF shows the probability of each of the two possible outcomes, 0 or 1. The plot of the CDF has a jump at each value that is given a positive probability of occurring by the PDF.

Out[•]=

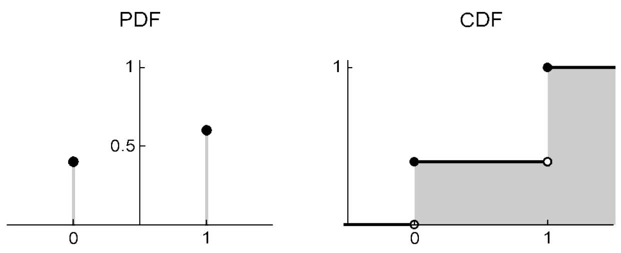


Figure 4: Bernouilli Distribution (p=0.6)

Use the **Mean** and **Variance** commands to compute the mean and variance of a distribution. It is even possible to express the mean and variance symbolically.

Out[•]=

Expression	Result
{Mean[bern50],Variance[bern50]}	{0.5, 0.25}
{Mean[bern60],Variance[bern60]}	{0.6, 0.24}
{Mean[bernp],Variance[bernp]}	{p, (1 - p) * p}

Moments of a Distribution

The mean is the expected value of the level of a random variable. Interpret this as follows: many draws from the distribution will have an arithmetic average that tends to equal the mean of the distribution. (Later sections make this more precise.) Calculate the mean of a distribution as the probability-weighted average of the possible values. For example, calculate the mean of a Bernoulli distribution as follows.

$$(1-p)*0 + p*1 == p \tag{7}$$

The variance is the expected squared deviation from the mean. A low variance indicates that draws from the distribution tend to lie close to the mean. Calculate the variance of a distribution as the probability-weighted average of the possible values of the squared deviation from the mean. For example, calculate the mean of a Bernoulli distribution as follows.

$$(1-p)*(0 - p)^2 + p*(1 - p)^2 == p*(1-p) \tag{8}$$

The mean and variance of a distribution are examples of moments of the distribution. Define the r-th moment of the distribution about a value k to be the expected value of $(X - k)^r$. With this definition, the mean is the first moment about 0 of the distribution. A moment about 0 is sometimes called a *raw moment*. WL provides a variety of facilities for moment computation, including the **Moment** command for computing raw moments and the **Expectation** command for computing the expected value of arbitrary expressions. The arguments of the **Moment** command may be a distribution and a specification of the order of the moment. The arguments for the **Expectation** command may be an expression and a specification of the distribution of one of the variables in the expression. (The distribution symbol, produced as ESC**dist**ESC, is shorthand for the **Distributed** command.) The commands can be used with symbolic distributions.

Out[∘]=

Expression	Result
`Moment[bernp,1]`	p
`Expectation[x,x≈bernp]`	p

A moment around the mean is usually called a *central moment*. The **CentralMoment** command computes central moments. Naturally, the **Expectation** command can do so as well.

Out[∘]=

Expression	Result
`CentralMoment[bernp,2]`	$(1-p)\,p$
`Expectation[(x-p)^2,x≈bernp]`	$p-p^2$

Binomial Distribution

The Bernoulli distribution is the simplest case of a discrete distribution, where 0 and 1 are the only two possible outcomes. The previous subsection modeled a coin flipping experiment with a Bernoulli distribution. Suppose that instead of of a single coin flip, an experiment involves two successive flips, again mapping tails to 0 and heads to 1. By the product rule, there are four possible outcomes. The **Tuples** command can enumerate them.

Out[∘]=

Expression	Result
`possibleOutcomes = Tuples[{0,1},2]`	`{{0, 0}, {0, 1}, {1, 0}, {1, 1}}`

Each outcome has an associated total number of successes (heads), which is just the summation over the subexperiments. Partition the sample space of possible outcomes into three different events, based on the total number of successes.

Out[∘]=

Table 6: Categorization of Outcomes by Total

event	total
`{{0, 0}}`	0
`{{0, 1}, {1, 0}}`	1
`{{1, 1}}`	2

These events are one useful way to categorize the outcomes. Here is a way to display the information that is easily generalized: use the **GroupBy** command to group the possible outcomes by their total number of heads. The result is an association that maps each possible number of heads to the list of outcomes that produce that number.

Out[∘]=

Expression	Result
`g = GroupBy[Total] @ possibleOutcomes`	`<|0 → {{0, 0}}, 1 → {{0, 1}, {1, 0}}, 2 → {{1, 1}}|>`

If the two flips are independent with the same probability of success, the result of each flip can be considered a Bernoulli *trial* of the larger 2-flip experiment. This implies a probability for each possible two-flip outcome that derives from the single probability of success. Naturally the probabilities over the four possible outcomes must sum to 1.

Out[∘]=

Outcome Probabilities (2 Trials)

		Second Trial	
		0	1
First Trial	0	$(1-p)^2$	$(1-p)\,p$
	1	$(1-p)\,p$	p^2

Unsurprisingly, an experiment comprising two sequential draws from a categorical distribution produces another categorical distribution, where the new categories are ordered pairs of the old categories. A new random variable, based on this new categorical distribution, is the total number of heads realized by an outcome. For the case of two flips, shown above, this random variable has three possible outcomes: 0, 1, or 2 successes. The table of head counts together with the table of outcome probabilities implies a probability of any particular head count *k*, which is the sum of the probabilities of all the outcomes that produce exactly *k* heads. Note that every way of producing exactly *k* heads has the same probability, which when multiplied by the number of ways of producing *k* heads produces the probability of *k* heads. A simple association can once again characterize the probability density function for this new discrete distribution.

This association characterizes the PDF of a 2-trial binomial distribution. Applying the **PDF** command to WL's predefined **BinomialDistribution** expresses the same result. As expected from the development to this point, a binomial distribution is determined by two arguments: the number of Bernoulli trials, and the probability of success.

Out[∘]=

Expression	Result
`PDF[BinomialDistribution[2,p],k] // Simplify[#,Assumptions→{0≤k≤2}]&`	$(1-p)^{2-k}\,p^k$ Binomial[2, k]

The PDF for the binomial distribution involves an old friend: the binomial coefficients. Generalizing, consider the case of *n* trials, and produce a random variable that is the count of the resulting number of successes. This count can take on any integer value in [0 .. *n*], so this is also a draw from a finite discrete distribution. It is called the *n*-trial binomial distribution. The binomial distribution has two parameters: *n* is the number of Bernoulli trials per draw, and *p* is the probability that a trial outcome is 1 (instead of 0). This gives us a very simple PDF.

Out[∘]=

Expression	Result
`PDF[BinomialDistribution[n,p],k] // Simplify[#,Assumptions→{0≤k≤n}]&`	$(1-p)^{-k+n}\,p^k$ Binomial[n, k]

WL can also express the CDF, but the expression involves a special function that is not as simple or intuitive. Still, as always for a discrete distribution, the CDF is a cumulative sum of the PDF values. The next chart illustrates this for **BinomialDistribution[5,0.5]**.

Out[∘]=

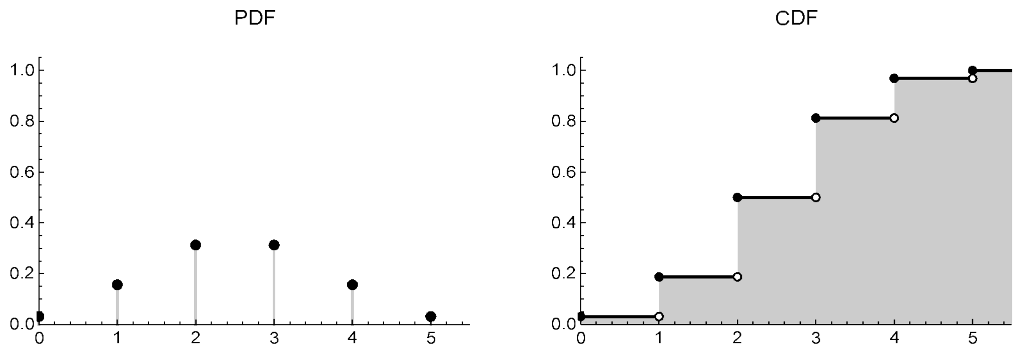


Figure 8: Binomial Distribution (n=5, p=0.5)

Unsurprisingly, increasing the number of trials raises the mean and variance of the distribution. The symbolic expression of these values confirms that both are strictly increasing in *n*.

Out[∘]=

Expression	Result

`bnp=BinomialDistribution[n,p];`
`{Mean @ bnp, Variance @ bnp}`
	{n p, n (1 - p) p}

Suppose 20 students show up for class and each has an independent 1% probability of carrying an infectious virus. The number of infected students is a random variable that is distributed *B*[20, 0.01]. Since the mean of the binomial distribution is *n p* == 0.2, less than a single student is infected in such a class, on average. The probability that at least one student is infectious is nevertheless nearly 20%: $1 - P[k == 0] == 1 - 0.99^{20} == 0.182$.