Univariate and Multivariate Distributions

Kim Seefeld
University of New Hampshire, Durham, NH
Department of Mathematics & Statistics

May 2007

1 Univariate versus Multivariate Statistics

The models introduced in the next are all univariate models, meaning that there is only one random variable on which data is measured and one outcome probability measure associated with that random variable in the model. Often many random variables are measured at the same time. To model how multiple random variables affect an outcome of an experiment a more complicated branch of statistics, multivariate statistics, is used. To compare these, suppose you are examining the annealing temperature of a DNA PCR primer. Univariate statistics would look how the percentage of one nucleotide affected annealing temperature whereas multivariate statistics would take into consideration how the composition of all four nucleotides and perhaps environmental conditions such as salt concentration affected annealing temperature and other factors.
In the following sections we will introduce some univariate and other multivariate models according to Kim Seefeld's book *Statistics Using R with Biological Examples* (2007) of New Hampshire, Durham, NH. We will reproduce chapters 7 and 8.

2 Discrete versus Continuous Random Variables

Random variables can be discrete or continuous. Discrete random variables are used when the set of possible outcomes (sample space) for an experiment is countable. Although many discrete random variables define sample spaces with finite numbers of outcomes, countable does not mean finite. The outcomes can be countably infinite (the integers are countably infinite because they are discrete and go on forever). Examples of experimental outcomes that are modeled with discrete random variables include numbers of people standing in a line, number of A's in a nucleotide sequence, and the number of mutations, which occur during a certain time interval.
A random variable that is not discrete but can assume any value, usually within a defined interval, is defined as a continuous random variable. Measurable quantities are generally modeled with continuous random variables. Examples of experimental outcomes that can be modeled with continuous random variables are: height, weight, intensity of a signal, and time required for a radioactive substance to decay.
Because much of the information bioinformatics deals with is discrete data (sequence information is usually analyzed using discrete random variables) the emphasis of this document is on

this type of data. However continuous random variables are not ignored and play an important role in some areas of bioinformatics, especially in Bayesian statistics and in microarray analysis.

3 Univariate Discrete Distributions

Univariate discrete distributions are standard probability models that utilize a discrete random variable to define the outcomes of an experiment. Presented here are two models frequently used in analyzing biological data: the binomial model and the related Poisson model.

3.1 The Binomial Distribution

The foundation for the binomial distribution is the Bernoulli random variable. A Bernoulli random variable arises in an experiment where there are only two outcomes, generally referred to as “success” and “failure”. For the success outcome the value of the random variable is assigned the value 1, and for the failure outcome the value of the random variable is assigned the value 0. The probability of success is a value p, a proportion between 0 and 1. The probability of failure (using the law that probability adds to 1 and that the complement probability is $1-$probablity of all other events) is $1-p$.
For a one trial experiment the probability distribution function for a Bernoulli experiment is trivial and the distribution of a Bernoulli random variable can be written as follows:

$$f(x) = p^x(1-p)^{1-x}$$

where x can take either of the value 0 or 1.
Let’s consider the case of having a child and use a Bernoulli random variable to represent whether the child has blue eyes. Let’s assume the probability of the child having blue eyes is 0.16 (not empirically verified!) and this is the “success” outcome. For this experiment the distribution of the random Bernoulli variable X is given in Table 1.

Table 1: Distribution of outcomes of a Bernoulli Trial

Outcome	**Random Variable** $X = x$	$P(X = x)$	**Probability of outcome**
Blue eyes	1	p	0.16
Not blue eyes	0	$1-p$	0.84

What about if you really want a blue-eyed child, so you have 10 children and you want to know the probability that k out of the 10 have blue eyes? This is a more complicated question. Each outcome of having a child is independent of other children — meaning whether the first child had blue eyes has no statistical influence on the second child having blue eyes. Independence will be discussed in more detail in the next chapter.
In order to answer this you want to create a model for how many of 10 children will have blue eyes based on the probability that a given child has blue eyes is 0.16. Such a model is called a binomial model. Using the Bernoulli probability distribution function equation above we can extend it to work for more than one trial by changing the exponents to $n =$ the number of trials and $k =$ the number of successes as follows:

$$p^k(1-p)^{n-k}$$

Note this is not a probability distribution function anymore, as it will only model the probability of a particular sequence of $n = 10$ children k of which have blue eyes. For example if $k = 3$, $n = 10$, the above represents the probability of a sequence such as $\{1,0,0,0,1,1,1,0,0,0,0\}$, where as indicated 1 denotes "blue eyes" and 0 denotes "not blue eyes". But when there are 10 children, the blue-eyed children can be any one of the 10 children. Using the counting method of combinations discussed in the previous chapter (formula below) we note that the number of such sequences with k ones and $n-k$ zeros is

$$C_{n,k} = \frac{n!}{k!(n-k)!}$$

Remember that for a series of 10 children, one blue-eyed child could be positioned in 10 different ways (the blue eyed child could be first of 10, second of 10, etc.), corresponding to a combination of

$$C_{10,1} = \frac{10!}{1!9!} = 10$$

In order to model the correct probability of observing 1 child of 10 children with blue eyes, the probability distribution function needs to account for the 10 different arrangements of children, so the proper way to write the probability distribution is

$$P(X = k) = P(k \text{ successes in } n \text{ trials}) = \binom{n}{k} p^k (1-p)^{n-k}$$

Note $\binom{n}{k} = C_{n,k} = \frac{n!}{k!(n-k)!}$ is another popular notation used for the number of combinations of k out of n distinct objects. This symbol is commonly described as *n choose k* and is also called the *binomial coefficient* in some contexts.

Of course to make this a distribution function we want to calculate not only the probability of having 1 in 10 children with blue eyes, but the whole distribution of how many kids $(0,1,2...10)$ in 10 will have blue eyes. This is tedious to do by hand and hence using R comes in handy.

In R the binomial distribution is the function `dbinom()`. In R, all probability distributions (or densities in the case of continuous random variables) use the letter `d` as the first letter in the function and then part or the entire name of the distribution for the rest of the function name. `dbinom()` takes as parameter arguments `(x, size, prob)` where `x` = the vector of k values to be used, `size` is the total number of trials n, and `prob` is the probability of success on each trial.

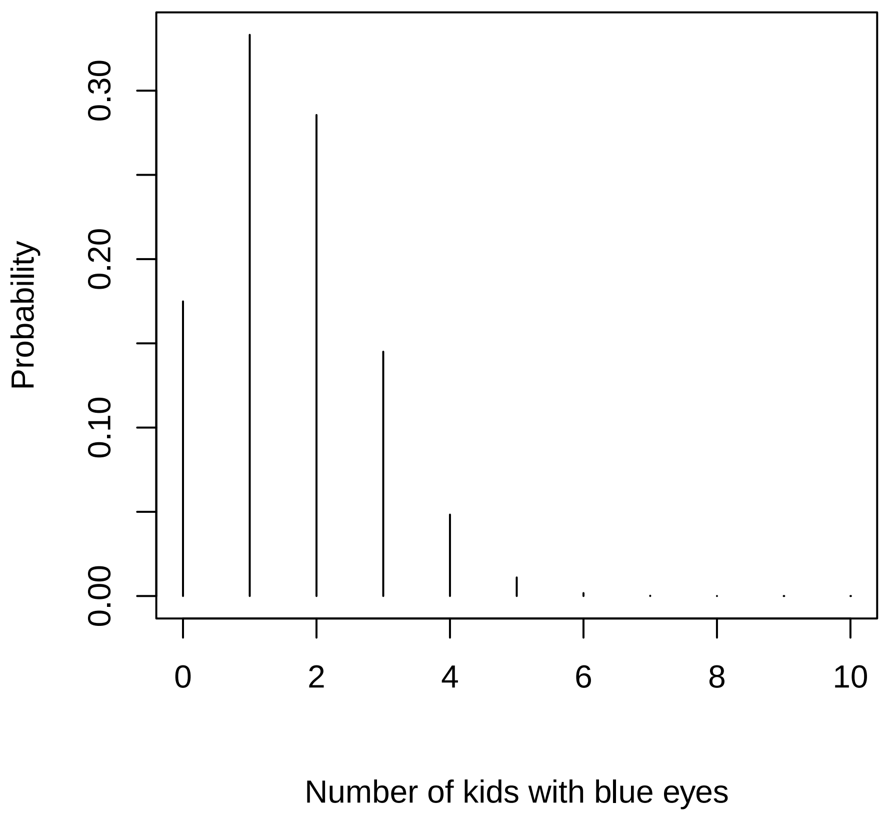
Using some simple commands in R to generate the probability values for the binomial distribution for the number of children in 10 with blue eyes using $p = 0.16$

```
> x <- 0:10
> y <- dbinom(0:10, 10, 0.16)
> data.frame(Prob = y, row.names = x)
```

we obtain the following:

```
           Prob
0  1.749012e-01
1  3.331452e-01
2  2.855530e-01
3  1.450428e-01
```

Figure 1: Example of a Binomial Distribution, $p = 0.16$



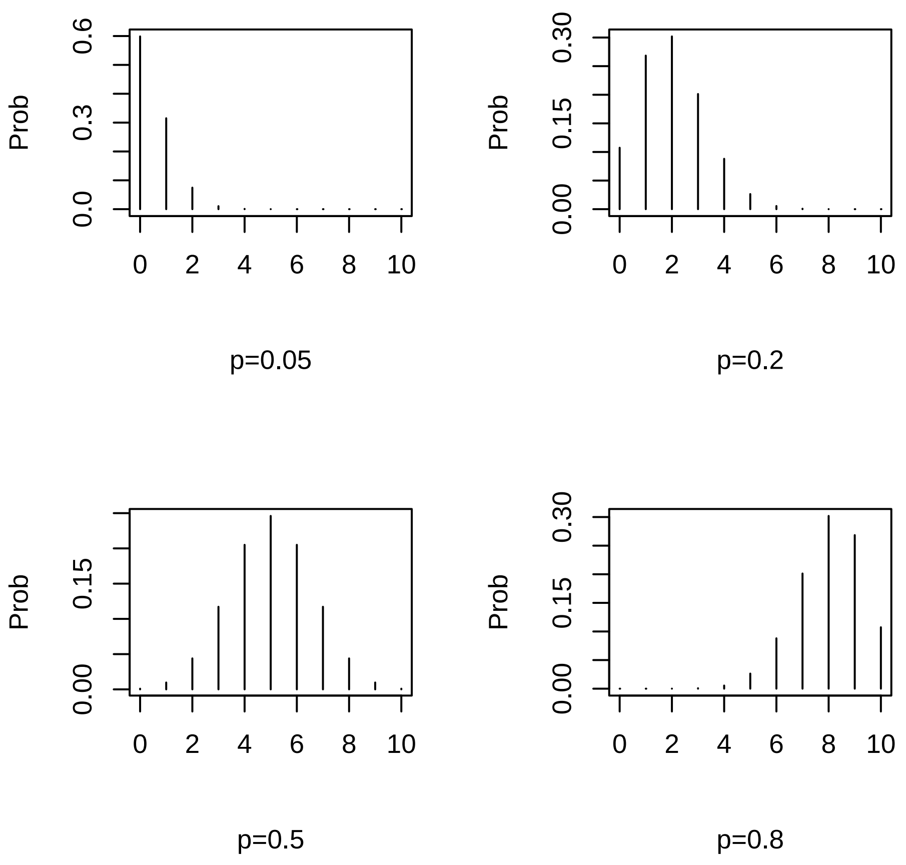
```
4  4.834760e-02
5  1.105088e-02
6  1.754108e-03
7  1.909233e-04
8  1.363738e-05
9  5.772436e-07
10 1.099512e-08
```

Thus given $p = 0.16$, the probability of 0 in 10 children with blue eyes is 0.175; the probability of one with blue eyes child is 0.333 and so forth.
Of course, writing out tables of probability as above is only practical for simple scenarios and in most cases a graphical model for the probability distribution will be used. To get a graphical model in R for the same distribution above, simply use the plot command and put the binomial function call right in the plot function call as follows:

```
> plot(0:10, dbinom(0:10, 10, 0.16), type = "h", xlab = "", ylab = "Probability",
+     sub = "Number of kids with blue eyes")
```

Figure 1 illustrates the graphic model of the probability distribution function for this example. Note that this distribution is pretty skewed toward lower values of the random variable ($X =$

Figure 2: Illustrating the Effect of Changing the Value of p in the Binomial Distribution

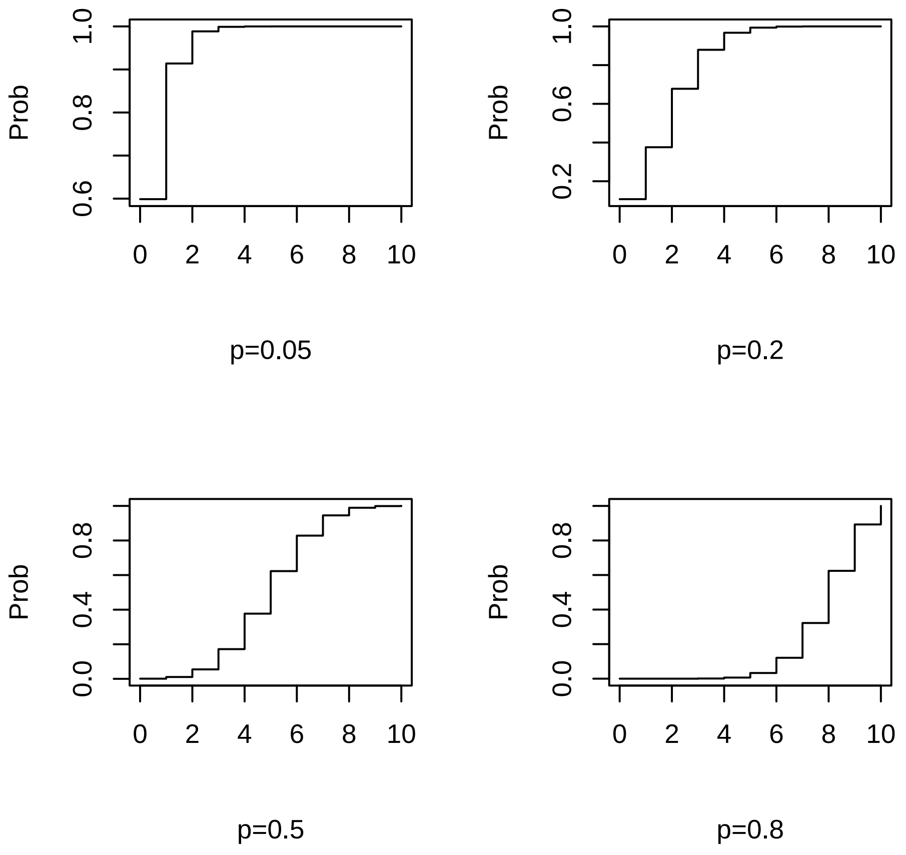


number of kids with blue eyes) because the value of p is 0.16. The graph should seem reasonable given the value of p. What if the value of p is changed?
Let's re-run this example with probabilities that a child has blue eyes is 0.05, 0.2, 0.5, and 0.8 and see how this changes the distribution.

```
> par(mfrow = c(2, 2))
> plot(0:10, dbinom(0:10, 10, 0.05), type = "h", xlab = "", ylab = "Prob",
+     sub = "p=0.05")
> plot(0:10, dbinom(0:10, 10, 0.2), type = "h", xlab = "", ylab = "Prob",
+     sub = "p=0.2")
> plot(0:10, dbinom(0:10, 10, 0.5), type = "h", xlab = "", ylab = "Prob",
+     sub = "p=0.5")
> plot(0:10, dbinom(0:10, 10, 0.8), type = "h", xlab = "", ylab = "Prob",
+     sub = "p=0.8")
```

Notice in Figure 2 how the larger p shifts the distribution more toward higher values of the random variable. This should make sense because a higher p makes it more likely a child will have blue eyes and therefore more children in 10 will have blue eyes, as represented by the shift of the graphical models with higher p. Note also for $p = 0.5$ that the distribution is symmetric. This is always the case for a binomial distribution with $p = 0.5$ since it equally likely that success or failure occurs.

Figure 3: Binomial CDFs Using Different Values of p



So far we have only considered the probability distribution function for the binomial, but what about the cumulative distribution function? Recall that this is the function which models the total probability up to and including a certain value of the random variable $X = x$.
This is easy to do in R using the `pbinom()` distribution function, which takes the same parameters as the `dbinom()`. In fact we can use the same code as above to get plots of the CDF of the binomial for the example above changing only the type of the plot to `s` for step and change the function used from `dbinom()` to `pbinom()`:

```
> par(mfrow = c(2, 2))
> plot(0:10, pbinom(0:10, 10, 0.05), type = "s", xlab = "", ylab = "Prob",
+     sub = "p=0.05")
> plot(0:10, pbinom(0:10, 10, 0.2), type = "s", xlab = "", ylab = "Prob",
+     sub = "p=0.2")
> plot(0:10, pbinom(0:10, 10, 0.5), type = "s", xlab = "", ylab = "Prob",
+     sub = "p=0.5")
> plot(0:10, pbinom(0:10, 10, 0.8), type = "s", xlab = "", ylab = "Prob",
+     sub = "p=0.8")
```

The pattern of cumulative probability for binomials produced using different values of p is illustrated in Figure 3. When p is small (as in 0.05) the cumulative probability reaches 1

quickly whereas a large value of p results in the cumulative probability not reaching 1 until the higher range of values for the random variable.
If all you need is a simple calculation for one value in R all you need to do is enter the appropriate function and relevant parameter values. For example, suppose you want to know the probability that (exactly) 4 kids in 10 will have blue eyes when $p = 0.5$. Simply use the `dbinom()` function in R as follows and it calculates this value for you:

```
> dbinom(4, 10, 0.5)

[1] 0.2050781
```

Thus, the chance of 4 in 10 kids with blue eyes is 0.205 or 20.5% with $p = 0.5$, which should make sense based on earlier graphical models.
The binomial distribution is an important model and one of the simplest to understand. Several other distributions are related to the binomial and also based on Bernoulli random variables (success or failure experiments). The geometric distribution (`dgeom()`, `pgeom()` in R) considers the random variable X as the number of failures before the first success. The negative binomial distribution (`dnbinom()`, `pnbinom()` in R) considers X as a measure of the number of failures before the r^{th} success. A multivariate version of the binomial, the multinomial distribution, will be introduced in the next chapter. The example used with the binomial could easily have been modeled using one of the related distributions. For example, the geometric could be used to model the number of children born before the first child with blue eyes. In many cases when you have data to model you have some choices how to model it based on choice of random variable measurement outcome and choice of distribution used.

3.2 The Poisson Distribution

The next discrete univariate distribution to be introduced is called the Poisson distribution, named after Simeon D. Poisson. The Poisson is one of the most utilized discrete probability distributions. Mathematically the Poisson is actually a limiting case of the binomial, the details of which will not be dealt with here but can be found in most mathematical probability books.
The Poisson has many applications, including numerous applications in biology. In general, the Poisson is used to model the counts of events occurring randomly in space or time. Simple real world examples which may use a Poisson model include the number of typing errors on a page, the number of accidents occurring at an intersection in a given time period and the number of discarded products made by a manufacturing process. In bioinformatics some examples where the Poisson could be used include: to model the instances of mutation or recombination in a genetic sequence, the distribution of errors produced in a sequencing process, the probability of random sequence matches, or in counting occurrences of rare DNA patterns.
The mathematical formula for the Poisson distribution is:

$$P(X = x) = e^{-\lambda}\frac{\lambda^x}{x!}$$

Here x represents counts and thus can be any integer value ≥ 0. Note also that in this equation, there is one parameter, the Greek letter lambda λ that defines the distribution. In a random process (such as mutation) there will be lambda events per unit time interval. Thus, lambda represents a rate. Because of the relation of the Poisson to the binomial, lambda can be obtained by the relationship $\lambda = np$ where p is the probability of the event occurring and n the number of trials. The relation holds when p is small (rare events) and the number of trials n are large.

Suppose we are using a new sequencing technique and the error rate is one mistake per 10000 base pairs. Suppose we are sequencing 2000 base pair regions at a time. What is the probability of 0 mistakes using this technique? Of 1 mistake, 4 mistakes? The Poisson model can be used to model these probabilities. To use the Poisson, you must first calculate lambda. In this case the *trial* can be considered an individual base pair, so $n = 2000$ trials for a 2000 base pair sequence. The probability of *success* here is the probability of getting an error, where $p = 1/10000$. To calculate lambda, multiply $n \times p$, or $2000/10000$ which results in a rate of 0.2 mistakes per 2000 base pairs so lambda is 0.2. Note it is common to adjust lambda based on n. If we were using 5000 base pair regions to sequence at a time we would use a lambda of 0.5.
To calculate the probability of one mistake in the 2000 base pair sequence, we could do this by hand with the following equation:

$$P(X = 1) = e^{-0.2}\frac{0.2^1}{1!} = 0.1637$$

However, we are interested in the whole distribution of probability values for the random variable $X =$ number of mistakes in the sequence and it is much easier to computer these in R than doing individual hand calculations. In R the `dpois()` function is used to compute Poisson distributions, and has parameters `(x, lambda)` where `x` is the value or vector of values of the random variable to be calculated and lambda is the parameter.
As we did with the binomial, first let's generate a simple table of probabilities that $X = x$ for the values of this distribution. We have a little bit of a problem in that in this case, theoretically X can be anywhere from 0 (no sequence errors) to 2000 (every bp an error). However, knowing lambda is 0.2 (also the mean or expected number of errors) the number of sequence errors is not likely to exceed 10, so the following code is used to generate the table:

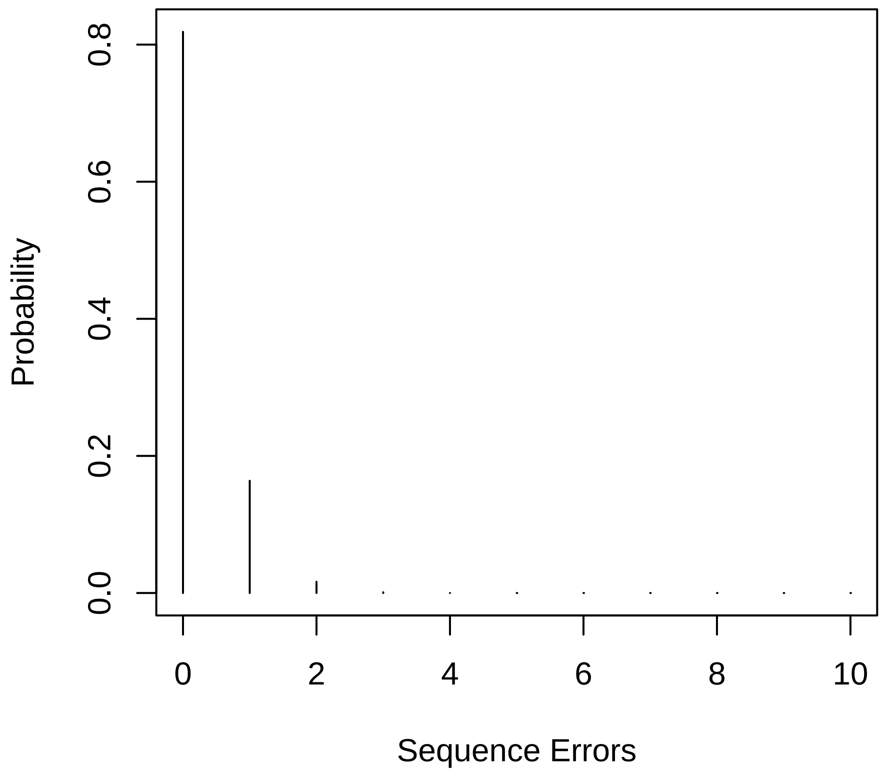
```
> x <- 0:10
> y <- dpois(0:10, 0.2)
> data.frame(Prob = y, row.names = x)
```

which produces the following results:

```
           Prob
0  8.187308e-01
1  1.637462e-01
2  1.637462e-02
3  1.091641e-03
4  5.458205e-05
5  2.183282e-06
6  7.277607e-08
7  2.079316e-09
8  5.198290e-11
9  1.155176e-12
10 2.310351e-14
```

The chance of no errors $X = 0$ is 0.818, the chance of 1 error $X = 1$ is 0.163, etc. in a 2000 base pair sequence. As expected, even at a value as low as 10 there is virtually no probability left. This can be viewed graphically as illustrated in Figure 4.

Figure 4: Poisson Distribution, $\lambda = 0.2$



```
> plot(0:10, dpois(0:10, 0.2), type = "h", xlab = "Sequence Errors",
+     ylab = "Probability")
```

The cumulative distribution in this case may be a bit more interesting. The CDF for the Poisson uses the `ppois()` function call with the same parameters as `dpois()` discussed above.

```
> plot(0:10, ppois(0:10, 0.2), xlab = "# Seq Errors", ylab = "Cum Prob",
+     type = "s")
```

As is clear from the CDF graph in Figure 5, the number of sequence errors using this method in a 2000 base pair sequence is highly unlikely to be more than 3. This should leave you pretty confident the process is not going to produce a lot of errors (unless you are looking for more stringent reliability). Below R is used to find the probability of 1 or fewer, 2 or fewer and 3 or fewer errors in this example.

```
> ppois(1, 0.2)

[1] 0.9824769

> ppois(2, 0.2)
```

Figure 5: CDF for Poisson Model of Sequencing Errors with $\lambda = 0.2$

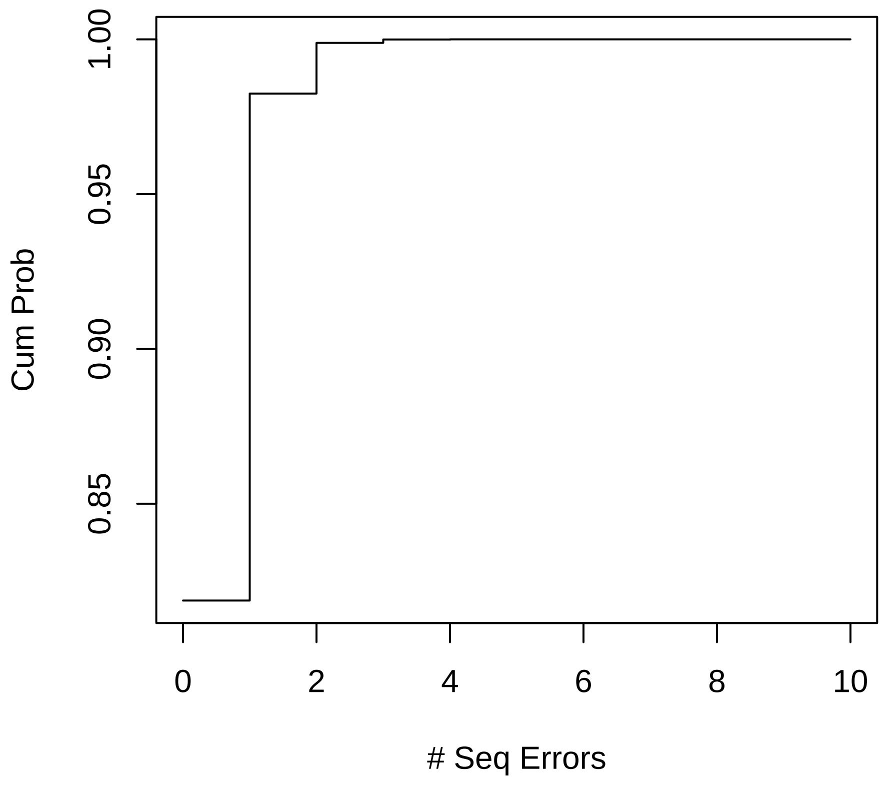
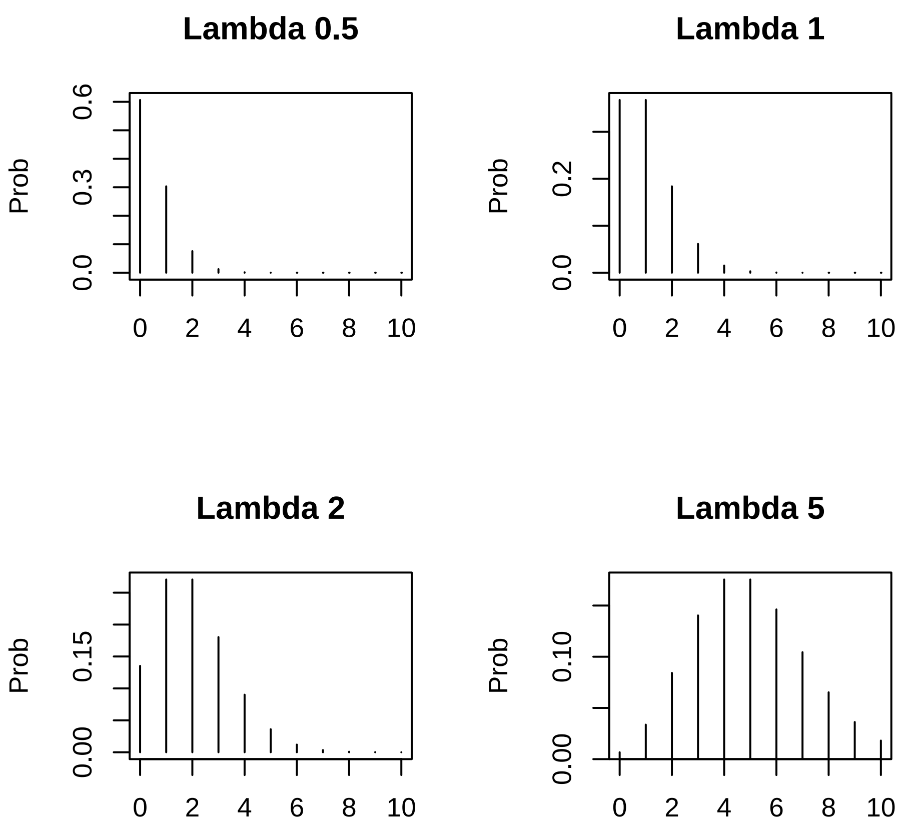


Figure 6: Poisson Distributions with Different Lambda Values



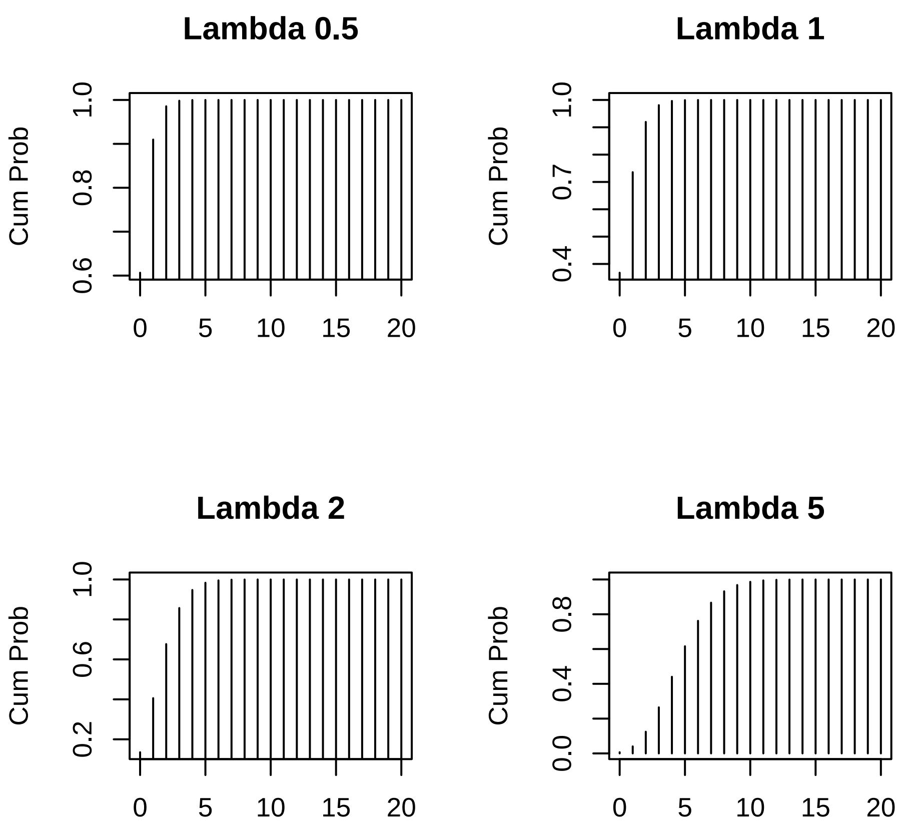
```
[1] 0.9988515

> ppois(3, 0.2)

[1] 0.9999432
```

What happens to the Poisson probability distribution when the parameter is changed? To examine this, let's look at a few examples with different values for lambda:

```
> par(mfrow = c(2, 2))
> plot(0:10, dpois(0:10, 0.5), xlab = "", ylab = "Prob", type = "h",
+     main = "Lambda 0.5")
> plot(0:10, dpois(0:10, 1), xlab = "", ylab = "Prob", type = "h",
+     main = "Lambda 1")
> plot(0:10, dpois(0:10, 2), xlab = "", ylab = "Prob", type = "h",
+     main = "Lambda 2")
> plot(0:10, dpois(0:10, 5), xlab = "", ylab = "Prob", type = "h",
+     main = "Lambda 5")
```

Not surprisingly the way the Poisson distribution changes (Figure 6) when lambda changes looks a lot like the way the binomial changes when p changes. Considering the relationship

Figure 7: Poisson CDFs with Different Lambda Values



$\lambda = np$ this should come as no surprise. Remember that the plots above only consider $X = x$ for the range of $[0, 10]$ so in the case of $\lambda = 5$ there is more of the distribution shifted to the right. To see where the probability levels off to 1 similar analysis can be done looking at the cumulative distributions `ppois()` for the Poisson as was done with the binomial. Let's do this with the range $X = x$ from 0 to 20 for all the values of lambda used in the previous plot.

```
> par(mfrow = c(2, 2))
> plot(0:20, ppois(0:20, 0.5), xlab = "", ylab = "Cum Prob", type = "h",
+     main = "Lambda 0.5")
> plot(0:20, ppois(0:20, 1), xlab = "", ylab = "Cum Prob", type = "h",
+     main = "Lambda 1")
> plot(0:20, ppois(0:20, 2), xlab = "", ylab = "Cum Prob", type = "h",
+     main = "Lambda 2")
> plot(0:20, ppois(0:20, 5), xlab = "", ylab = "Cum Prob", type = "h",
+     main = "Lambda 5")
```

For the lambda values 2 or less it is pretty clear from the CDF plots (Figure 7) that it is unlikely more than 10 of 2000 base pairs would contain errors. Be careful though although the graph for $\lambda = 5$ appear to level off at 1 around $x = 10$ there is still some significant probability of obtaining a value of X higher than 10, which can be analyzed by doing some additional calculations in R as is done below:

```
> ppois(10, 5)

[1] 0.9863047

> ppois(12, 5)

[1] 0.9979811

> ppois(15, 5)

[1] 0.999931

> ppois(20, 5)

[1] 0.9999999
```

This concludes discussion, for now, of discrete univariate probability distributions. You should have a feel for these distributions and how to work with them in R.

4 Univariate Continuous Distributions

4.1 Univariate Normal Distribution

The normal distribution is the typical bell curve distribution used to characterize many types of measurable data such as height, weight, test scores, etc. The normal is also the distribution that is used to model the distribution of data that is sampled, as will be discussed later in this book under the topic of inferential statistics. Sometimes the normal distribution is called the Gaussian distribution, in honor of Karl Gauss. It is a ritual that all introductory statistics students are saturated with details about the normal distribution, far more than will be covered here. The probability density equation for the normal distribution, presented below, should ring a bell of familiarity to graduates of statistics courses:

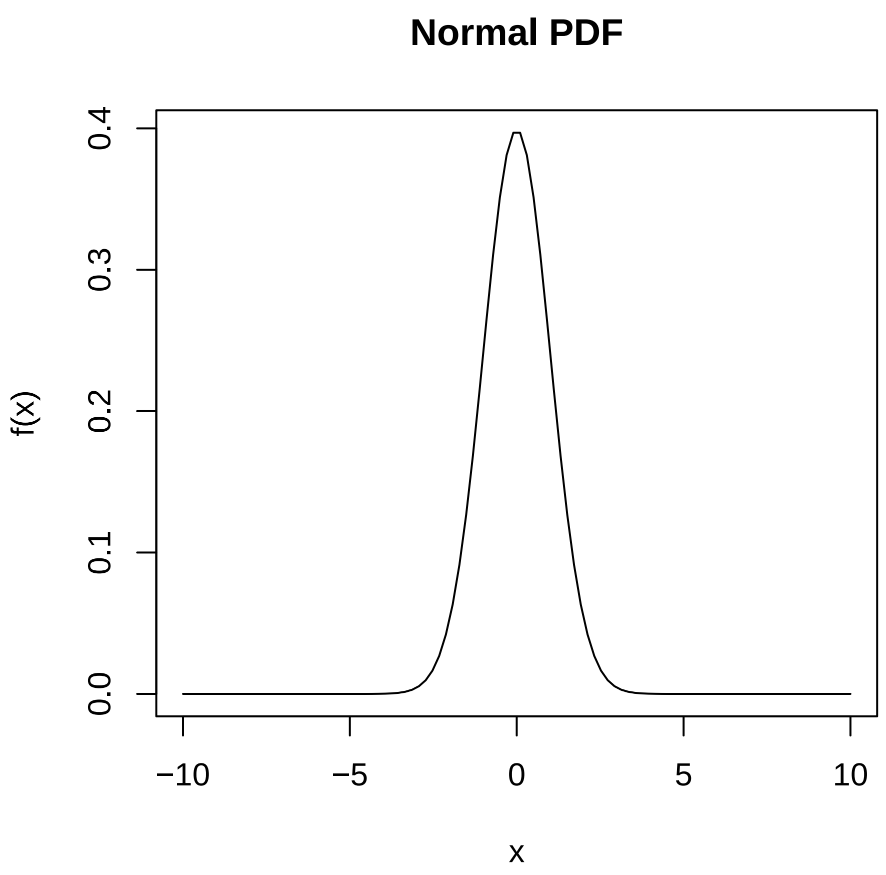
$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In the equation above, the Greek letter mu (μ) represents the mean of the distribution (aka: average value, expected value) and the Greek letter sigma (σ) represents the standard deviation of the distribution (sigma squared is the variance). Mu and sigma serve as the parameters for the distribution. For the normal distribution, the location and scale parameters correspond to the mean and standard deviation, respectively. However, this is not necessarily true for other distributions. In fact, it is not true for most distributions.

One of the tricks with the normal distribution is that it is easily standardized to a standard scale. If X is a continuous random variable with mean mu and standard deviation sigma it can be standardized by transforming X to Z where Z is a normally distributed variable with mean 0 and standard deviation 1 (which also equals the variance since $1^2 = 1$). This is useful if you have a bunch of different X's and want to put them all on the same Z system so you can compare them, with a scoring system called Z-scores (see your favorite introductory statistics book for further discussion). The transformation of X to Z is simply:

$$Z = \frac{X - \mu}{\sigma}$$

Figure 8: Univariate Standard Normal Distribution



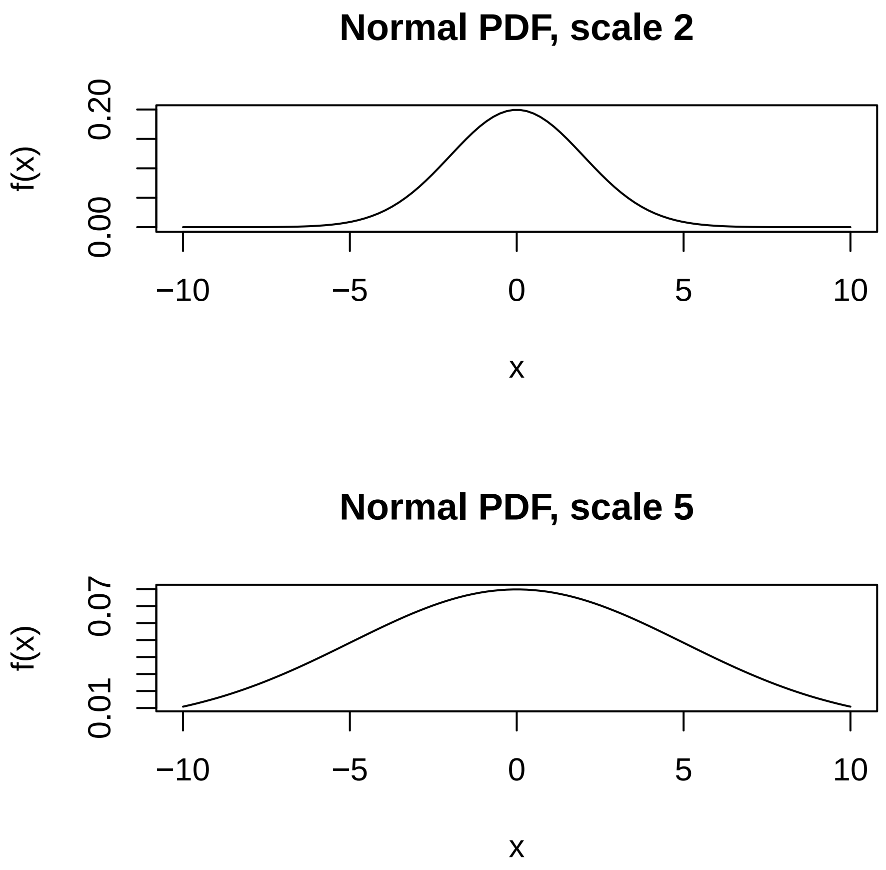
R contains functionality for both the probability density function and cumulative distribution function for the normal model in the base package. Like previous distributions discussed, the command for the probability density function starts with "d" and is named `dnorm()`. The parameters of `dnorm()` are the data vector of x's, the mean and standard deviation. As an example, let's plot a normal density for a range of x from ˘10 to 10 with mean 0 and standard deviation 1:

```
> x <- seq(-10, 10, length = 100)
> plot(x, dnorm(x, 0, 1), xlab = "x", ylab = "f(x)", type = "l",
+     main = "Normal PDF")
```

In discussing location parameters in the previous chapter, we looked at an example of changing the location parameter for the normal. But what happens if we change the scale parameter? Remember that increasing the scale parameter, which is the standard deviation in the case of the normal, increases how spread out the data are. To look at this in R, simply change the standard deviation parameter of `dnorm()`:

```
> par(mfrow = c(2, 1))
> plot(x, dnorm(x, 0, 2), xlab = "x", ylab = "f(x)", type = "l",
+     main = "Normal PDF, scale 2")
```

Figure 9: Changing the scale parameter (standard deviation) in the normal distribution



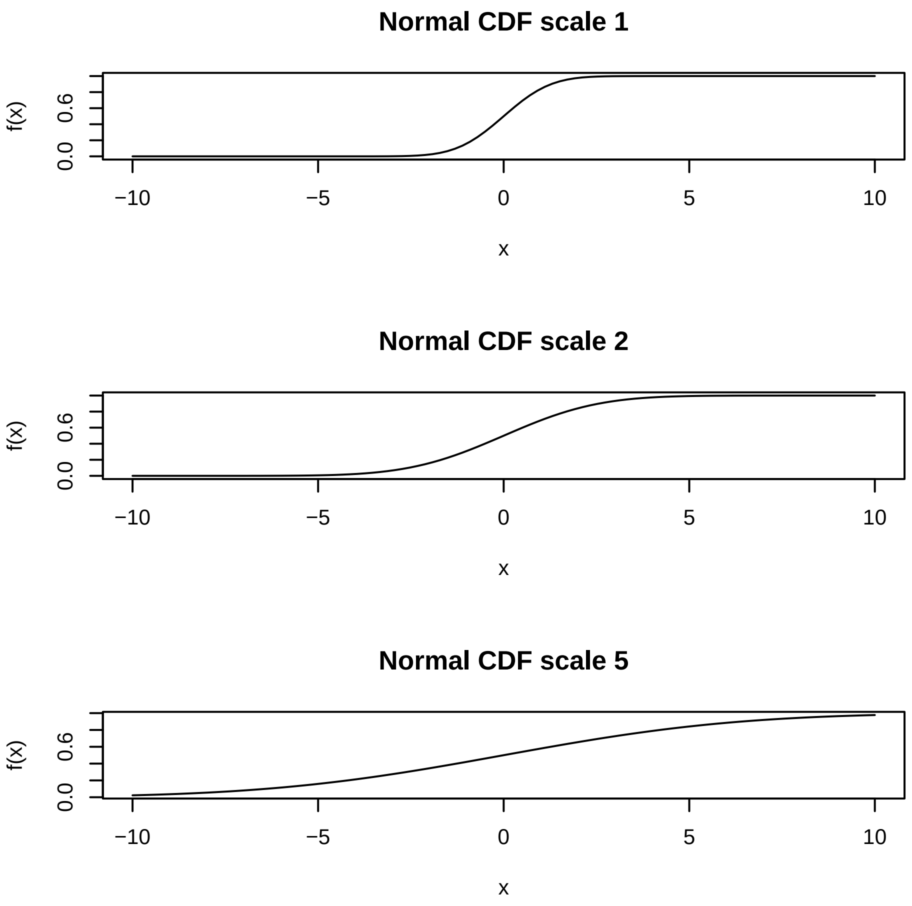
```
> plot(x, dnorm(x, 0, 5), xlab = "x", ylab = "f(x)", type = "l",
+     main = "Normal PDF, scale 5")
```

The change in the scale parameter from 2 to 5 is illustrated in Figure 9.
The normal also has a cumulative density function, which in R utilizes the `pnorm()` function with the same parameters as `dnorm()`. The code below computers some normal CDFs, using a few different scale paramters with the same mean of 0.

```
> par(mfrow = c(3, 1))
> plot(x, pnorm(x, 0, 1), xlab = "x", ylab = "f(x)", type = "l",
+     main = "Normal CDF scale 1")
> plot(x, pnorm(x, 0, 2), xlab = "x", ylab = "f(x)", type = "l",
+     main = "Normal CDF scale 2")
> plot(x, pnorm(x, 0, 5), xlab = "x", ylab = "f(x)", type = "l",
+     main = "Normal CDF scale 5")
```

Notice in Figure 10 how increasing the scale parameter (standard deviation) causes the CDF to increase much less precipitously, reflecting the less concentrated (more spread out) data distribution from the probability density.
In inferential statistics, the normal is frequently used to answer questions with regard to test scores and other such measures. For example suppose you take a test and score 85, which

Figure 10: Normal Distribution CDFs with Different Scale Parameters



sounds great until the professor announces the mean score is 92 with a standard deviation of 8. Then you begin to wonder, how badly did you do? To answer this use the `pnorm()` function in R as follows:

```
> pnorm(85, mean = 92, sd = 8)

[1] 0.190787
```

This means you scored better than 19.1% of the class. You could have answered this question looking at the other end of the distribution as well using 1−`pnorm()`.

```
> 1 - pnorm(85, mean = 92, sd = 8)

[1] 0.809213
```

This means that 80.9% of the class scored better than you (better luck next time). Knowing the mean and standard deviation of a normal distribution makes such calculations easy. And if all you have is a data set that you are assuming is normally distributed, you can enter your data, and then use R to find the mean and standard deviation. For example, suppose you chop up a piece of DNA with an enzyme and get 20 fragments with sizes you measure on a gel (to the nearest base pair) and you guess from the gel pattern that the size of the fragments is normally distributed. You can record (in data vector `x`) and analyze your data in R as follows:

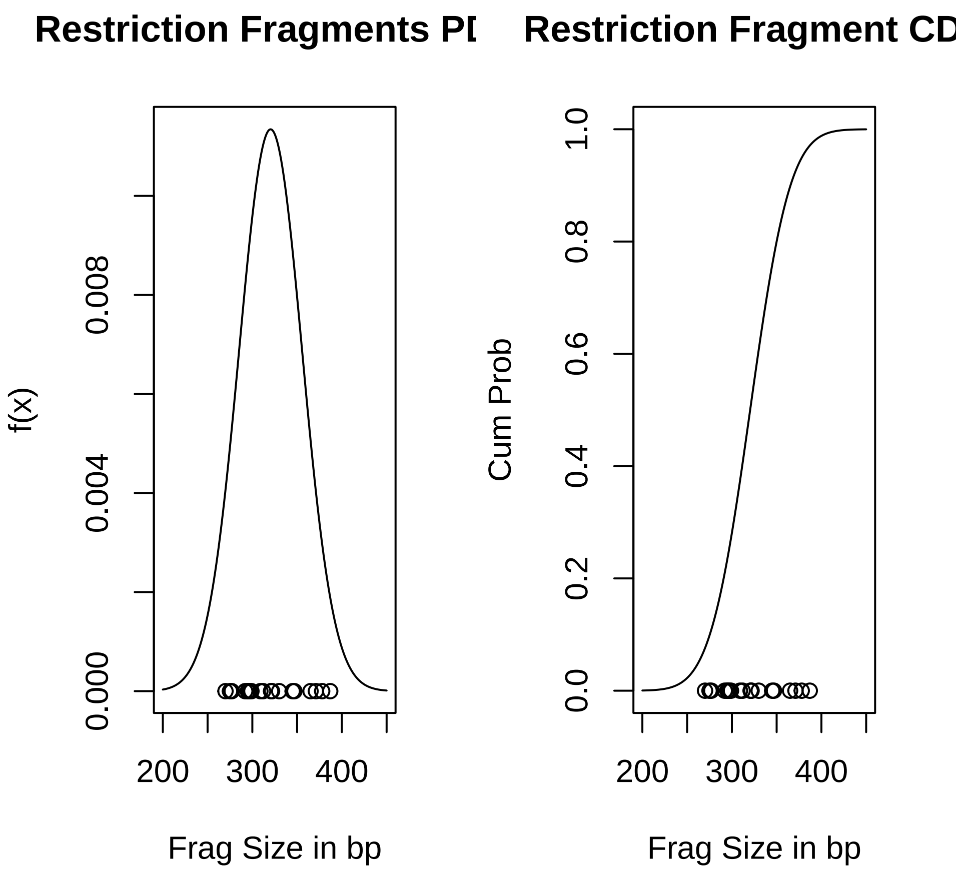
```
> x <- c(321, 275, 345, 347, 297, 309, 312, 371, 330, 295, 299,
+     365, 378, 387, 295, 322, 292, 270, 321, 277)
> mean(x)

[1] 320.4

> sd(x)

[1] 35.16787
```

And now, knowing the mean and standard deviation, you have a probability density function normal model for this data that you can use to perform further statistical tests on. It is very simple to graph this model and its CDF function using the code below, with results depicted in Figure 11.

```
> par(mfrow = c(1, 2))
> xx <- seq(200, 450, by = 0.25)
> plot(xx, dnorm(xx, mean(x), sd(x)), type = "l", xlab = "Frag Size in bp",
+     ylab = "f(x)", main = "Restriction Fragments PDF")
> n <- length(x)
> points(x, rep(0, n))
> plot(xx, pnorm(xx, mean(x), sd(x)), type = "l", xlab = "Frag Size in bp",
+     ylab = "Cum Prob", main = "Restriction Fragment CDF")
> points(x, rep(0, n))
```

In addition, you now have a model you can make inferences on. For example, suppose you want to know the probability of getting a fragment bigger than 400 bp. A simply query R computes this.

Figure 11: PDF and CDF for restriction fragment data



```
> 1 - pnorm(400, mean(x), sd(x))

[1] 0.0118046
```

Based on the above result, there is a 1.18% chance of getting a fragment this big from the given distribution.

Another useful feature in R is that all distributions have an associated quantile function built in, designated by a q before the distribution name code (qnorm(), qbinom(), etc). This automatically calculates what percentage of the distribution corresponds to the given cumulative probability regions. For example, suppose for the distribution above you want to calculate the 10th and 90th percentiles of the distribution. Simply use the qnorm() function in R to do this.

```
> qnorm(0.1, mean(x), sd(x))

[1] 275.3306

> qnorm(0.9, mean(x), sd(x))

[1] 365.4694
```

This means that 10% of the data is 275 or fewer base pairs in size, and 90% of the data is 365 or fewer base pairs in size, given this distribution. Quantiles are called quartiles for the 25%, 50% and 75% regions and also called percentiles on standardized tests. The quantile function can be helpful for analyzing data in a distribution.

Even though the normal is widely taught in statistics courses, and widely used in many areas of statistics, it is not the only continuous probability distribution to be familiar with for effective data analysis. Many times, especially when dealing with physical phenomena (as opposed to humanly generated measurable data such as that from standardized school tests) the data will not be normally distributed. For non-normally distributed continuous data modeling we now turn to two important families of distributions, the gamma family and the beta family.

4.2 The Gamma Family

The gamma family consists of a few related distributions including the gamma distribution, the exponential distribution and the Chi-Square distribution. The base distribution of the family is the gamma distribution, which provides a versatile model for working with continuous data that may not be normally distributed. Popular applications of the gamma distribution are to measurements of time until failure, concentrations of pollutants, etc. The gamma distribution is only defined for positive real numbers and it takes different forms depending on the parameter values. The probability density of the gamma distribution has the following general form:

$$f(x) = \frac{1}{\beta^\alpha \Gamma(\alpha)} x^{\alpha-1} e^{-x/\beta}$$

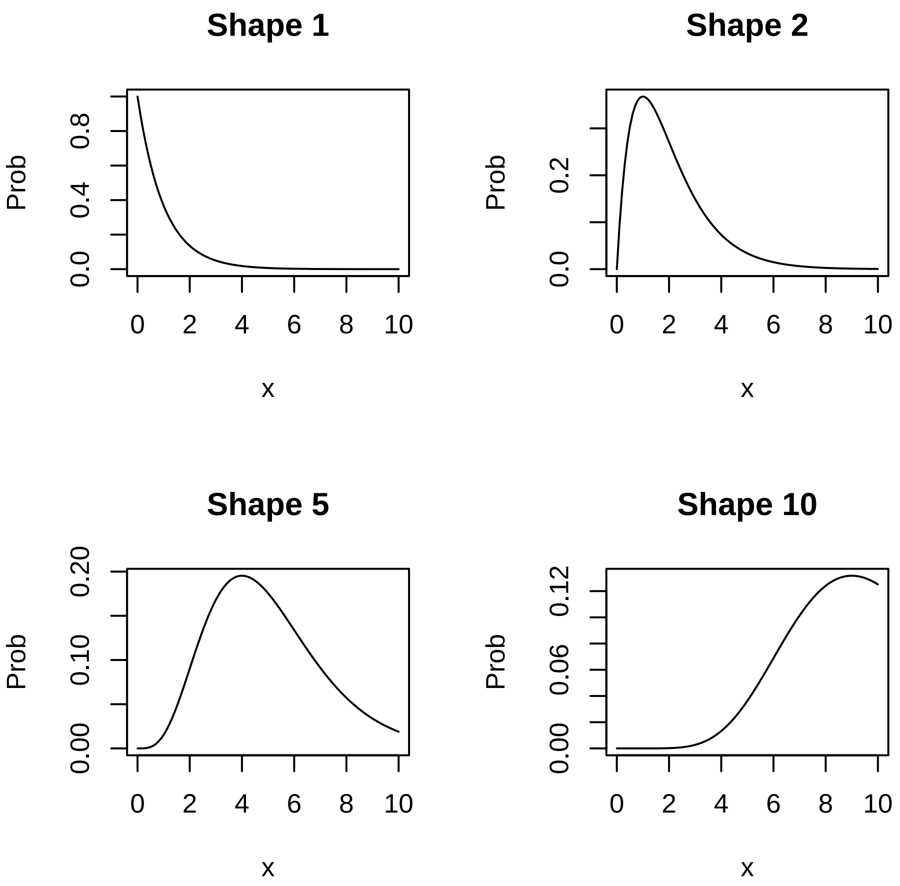
Before you scream and shriek and give up, thinking you cannot possibly understand this crazy equation, relax and realize it's only a mathematical model for a probability density function. Long gone are the days when anyone would hand calculate $f(x)$ values for this equation because computer packages such as R are very happy to do the calculations for us. The only things in the equation besides the familiar mathematical terms x and e are two parameters – alpha

(designated by the Greek letter α) and beta (designated by the Greek letter β) and the gamma function (introduced in the pervious chapter) where the gamma function of the parameter alpha is part of the equation. For the gamma distribution alpha is the shape parameter and beta is the scale parameter.
Like all the other distributions, the probability distribution for the gamma is simple to use in R. The `dgamma()` function call takes as parameters the data vector and then the shape and scale parameters.
First, let's make a few graphs of changing the shape parameter, alpha, while keeping the scale parameter, beta, is constant at 1:

```
> x <- seq(0, 10, length = 100)
> par(mfrow = c(2, 2))
> plot(x, dgamma(x, shape = 1, scale = 1), type = "l", xlab = "x",
+     ylab = "Prob", main = "Shape 1")
> plot(x, dgamma(x, shape = 2, scale = 1), type = "l", xlab = "x",
+     ylab = "Prob", main = "Shape 2")
> plot(x, dgamma(x, shape = 5, scale = 1), type = "l", xlab = "x",
+     ylab = "Prob", main = "Shape 5")
> plot(x, dgamma(x, shape = 10, scale = 1), type = "l", xlab = "x",
+     ylab = "Prob", main = "Shape 10")
```

Note in Figure 12 that for shape = 10 the distribution shifts toward the higher end (and the graph doesn't depict the entire distribution only the same x range as the other graphs for comparison).

Figure 12: Gamma distributions with different shape parameters



Next, let's look at how the gamma changes when the shape parameter, alpha, is held constant (at 2) and the scale parameter, beta, is changed.

```
> x <- seq(0,30,length=100)
> plot(x,dgamma(x,shape=2,scale=1), type='l',xlab="x", ylab="f(x)",
+ main="Gamma pdf's")
> lines(x,dgamma(x,shape=2,scale=2),lty=2)
> lines(x,dgamma(x,shape=2,scale=4),lty=3)
> lines(x,dgamma(x,shape=2,scale=8),lty=4)
> legend(x=10,y=.3,paste("Scale=",c(1,2,4,8)),lty=1:4)
```

Note that, although they might not look it, all of the graphs in Figure 7-13 are the same shape. The higher scale parameter values just spread the distribution out. For the higher values the distribution extends beyond the x range shown in the graph (but for comparison all the graphs are on the same x scale range).

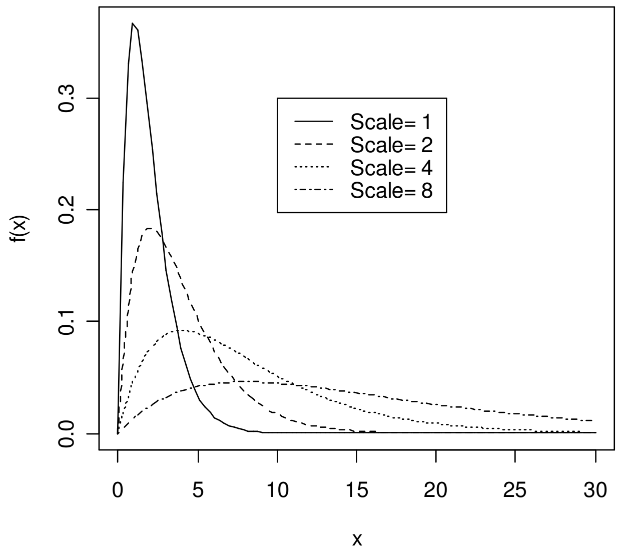


Figure 7-13: Gamma distributions with different scale parameters

You may be wondering how the gamma distribution is used to model the distribution of a set of data.

Suppose you are measuring survival times of an enzyme in a solution (as measured by some kind of assay for enzyme activity) and you get the following data in hours: 4.75, 3.4, 1.8, 2.9, 2.2, 2.4, 5.8, 2.6, 2.4, and 5.25. How could you decide on a probability model to model the probability of the enzyme surviving in solution?

Because you know you cannot always assume data is normally distributed (although you often hope so) the first thing to do is to look at a plot of the data. There is actually a statistician's secret tool to check whether data are normally distributed. It is a plot called a Q-Q plot and what it does is line quantiles of the data against normal quantiles. If the line is a straight line, the data can be considered normally distributed and you can use the normal probability model.

All you have to do to run a Q-Q plot in R is enter the data and use the qqnorm and qqline functions.

```
> x<-c(4.75, 3.4, 1.8, 2.9, 2.2, 2.4, 5.8, 2.6, 2.4, 5.25)
> qqnorm(x)
> qqline(x)
```

Running this code produces the Q-Q plot in Figure 7-14.

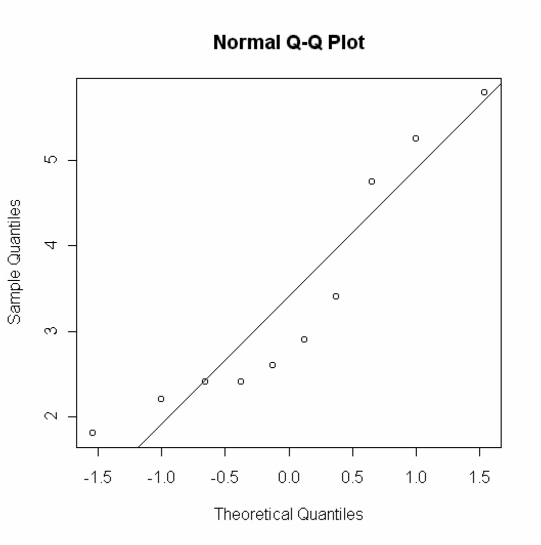


Figure 7-14: Q-Q plot of enzyme data

And you know, by looking at the squiggle pattern of the data that it does not align nicely with the expected normal line. When you see such a nonlinear data pattern on a Q-Q plot you should train yourself to think that you'd probably better find a model other than the normal distribution to model the distribution of this data.

Knowing about the flexibility of the gamma distribution, you suspect there may be a gamma distribution model for this data. But how do you determine the alpha (shape) and beta (scale) parameters you need in order to use the gamma model?

Actually it's not too hard. Although the scale and location parameters for the gamma model are not equal to the standard deviation and mean like in the normal case, there is a mathematical relationship between the mean and standard deviation and the scale (beta) and shape (alpha) parameters for the gamma distribution.

The mean of the gamma distribution is equal to alpha * beta and the variance (=standard deviation squared) is related to alpha and beta by being equal to alpha*beta^2. In R it is always easy to get the mean and variance (or sd) of a data vector:

```
> mean(x)
[1] 3.35
> var(x)
[1] 1.985556
```

Let's be lazy and just call the variance =2. But based on this we know that:

$$\text{Mean}=3.35=\alpha\beta$$

and

$$\text{Var} = 2 =\alpha\beta^2$$

Doing a little algebra:

$$3.35/\beta=\alpha$$

and then substituting this into the variance equation for alpha allows you to solve for beta:

$$3.35*\beta=2, \text{ so } \beta=0.6 \text{ (roughly)}$$

and subsequently, you can solve for alpha

$$3.35=\alpha\ (0.6) \text{ so alpha} = 5.6$$

So the distribution of the data can be modeled using a gamma probability density function with shape (alpha) = 5.6 and scale (beta)=0.6. Note that because we don't have many data points (only 10) this might not be the best possible fit, but since we only have such a limited amount of data it's impossible to assess the goodness of fit (collecting more data values would be better of course). Also note that alpha and beta need not be integer values, allowing even greater flexibility in how the gamma model fits data. However, there is a requirement that both alpha and beta be positive values (so if you do the algebra and get negative values for alpha and beta, you did something wrong).

Let's look at a graphical model of the data and a dgamma plot using the parameters determined above:

```
data <- c(4.75, 3.4, 1.8, 2.9, 2.2, 2.4, 5.8, 2.6, 2.4, 5.25)
n <- length(data)
x <-seq(0,8,length=200)
plot(x,dgamma(x,shape=5.6,scale=0.6),type='l',ylab="f(x)")
points(data,rep(0,n))
```

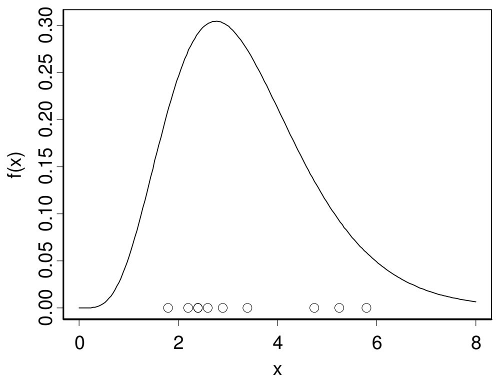


Figure 7-15: Comparing the data distribution to the gamma fit

As with other distributions the cumulative distribution function for the gamma is designated starting with a p, in this case pgamma. The CDF for the model used in this example can simply be graphed in R as:

```
plot(x,pgamma(x,shape=5.6,scale=0.6),type='l',ylab="P(X<=x)",
+ main="Gamma CDF Fit")
points(data,rep(0,n))
```

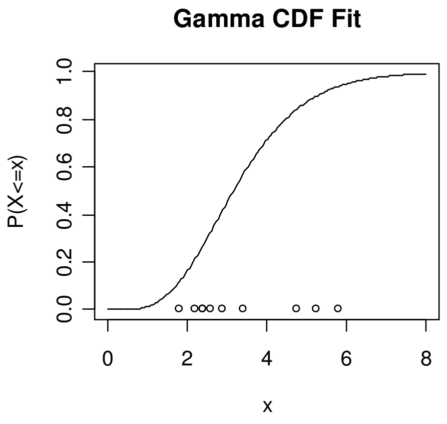


Figure 7-16:Gamma CDF

It looks from the CDF plot in Figure 7-16 that there may still be some probability density at x values higher than 5. We can perform a simple calculation to check this out.

```
> 1-pgamma(5,shape=5.6,scale=0.6)
[1] 0.1263511
```

Thus, there 12.6% of the density has values greater than 5 and you may want to re do the plots to reflect this (an exercise left to the reader).

The Exponential Distribution

The exponential distribution, famous for modeling survival times (as in the case with radioactive decay), is just a special case of the gamma distribution where the shape parameter, alpha = 1. This reduces the mathematical formula for the gamma to:

$$f(x) = \frac{1}{\beta} e^{-x/\beta}, x > 0$$

The exponential is often written in terms of a rate parameter lambda where $\lambda = 1/\beta$, or

$$f(x) = \lambda e^{-\lambda x}, x > 0$$

Most students of science have seen this form for radioactive decay rates, or survival rates of bacteria or something of that sort. Although details will not be discussed here, the R functions dexp (x, rate =) is used with the rate parameter lambda value to model the probability density, and the function pexp is used to model the CDF.

The Chi Square Distribution

The Chi-Square distribution is another gamma distribution variant, and the term "Chi-Square" should be familiar to genetics students for its role in analyzing count data and other applications. The Chi-Square distribution always uses a value of beta=2 for the scale parameter and a value of alpha=k/2 for the shape parameter where k is the number of "degrees of freedom". The Chi-Square distribution and concept of "degrees of freedom" will be returned to in later chapters, but is noted here to show its relationship to the gamma distribution. In R the probability density for this distribution is denoted as dchisq, and the cumulative density is pchisq. Both take as parameters the data vector and the degrees of freedom.

The Beta Family

Like the gamma family, the beta family is group of distributions that use alpha and beta parameters. The beta family has the following mathematically formula:

$$f(x) = \frac{1}{\beta(\alpha,\beta)} x^{\alpha-1}(1-x)^{\beta-1}, 0<x<1$$

And you ask, what is that $\beta(\alpha, \beta)$ thing in the denominator? Well, that thing is called the beta function, and the beta function is actually a ratio of gamma functions, as follows:

$$\beta(\alpha,\beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}$$

One very important thing to note – the range of x in the equation for the beta probability density is clearly denoted as being between 0 and 1. This is key. The beta function is used to model data measured as proportions. For example, if you have data on the proportion of an amino acid in a protein motif (say a leucine zipper) and it is not likely to be normally distributed (check with a Q-Q plot), then you should model the data with a beta density function.

Unfortunately the interpretation of the parameters with the beta are not as clear as with the gamma, and with the beta distribution, the alpha and beta parameters are sometimes referred to as the "shape 1" and "shape 2" parameters. Both parameters play a role in how the data fits the distribution. As with the gamma, the alpha and beta parameters have a relationship to the mean and variance of the distribution, with the following mathematical formulas:

$$\text{mean} = \frac{\alpha}{\alpha+\beta}$$
$$\text{variance} = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)}$$

The formula for the mean is not too bad, but remember to solve for alpha and beta you need to solve for both – and algebraically it's not quite so pretty to do with the above relationships. Because the beta distribution is a sine qua non in Bayesian statistics (and literally, used all over the place, along with its multivariate counterpart the Dirichlet) it is worth the time to write a small program to calculate these parameter values. Meanwhile, let's learn how to work with the dbeta density function and the pbeta cumulative distribution functions in R.

Let's use as an example the proportion of acidic amino acids found in a particular motif of proteins (a generic example with no particular type of motif). Assume we have already determined the parameters of the beta density that models our data. We have alpha ("shape 1") =2 and beta ("shape 2") = 10. A graphical model of this density is made in R with the following command:

```
> x
 [1] 0.11 0.10 0.10 0.16 0.20 0.32 0.01 0.02 0.07 0.05 0.25 0.14 0.11 0.12
0.08
[16] 0.13 0.08 0.14 0.09 0.08
> data<-x
> n <- length(data)
> x <- seq(0,1,length=200)

> plot(x,dbeta(x,2,10),xlab="prop. of acidic residues", ylab="f(x)",
+ main="Beta PDF for residue data")
> points(data,rep(0,n))
```

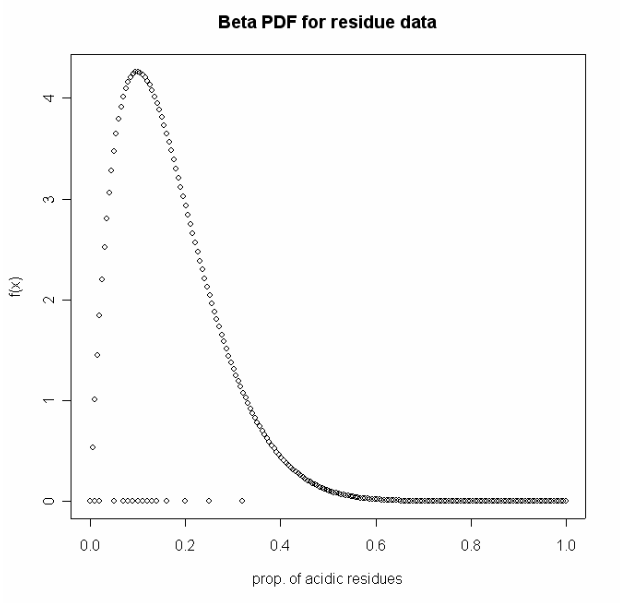


Figure 7-17: Beta PDF

Similarly a CDF plot can be generated using the pbeta function:

```
> plot(x,pbeta(x,2,10),xlab="prop. of acidic residues", ylab="Cum. prob",
+ main="Beta CDF for residue data")
> points(data,rep(0,n)
```

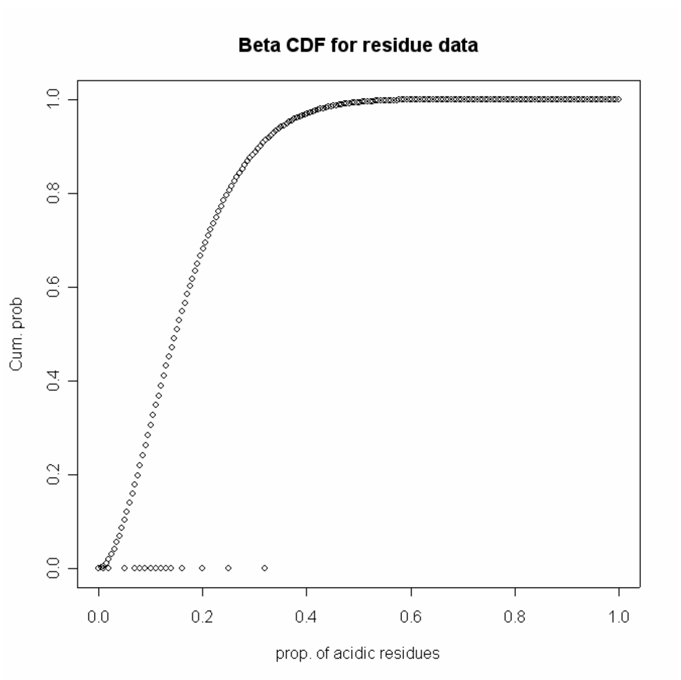


Figure 7-18: CDF for data distrbuted with a beta model

Other Continuous Distributions

Many other continuous distributions exist, but will not be discussed here. The interested reader is advised to consult a probability and/or mathematical statistics book for further information. R contains functionality to analyze many of these distributions and some distributions can be computed using a transformation of an existing distribution. For example, there is such a thing as an inverse gamma distribution, which can be computed using gamma distribution functionality. Two other continuous distributions which the reader may be familiar with will be introduced later – the t distribution and the F distribution. The t distribution will be introduced in the inferential statistics chapter and is the distribution which introductory statistics students are taught to use in the context of hypothesis testing. The F distribution will be introduced in the chapter on experimental design and plays an important role in microarray data analysis.

Simulations

One of the greatest powers of using a computer lies in the ability to simulate things. What if you don't have any data, but you know the probability model that you want to work with? Through the power of simulation, you can use the computer to generate sample values for you. It's like doing an experiment, only in the virtual world instead of the real world. Simulation is an essential technique in computational statistics.

In R simulations are very easy to do. Every distribution mentioned in this chapter has a corresponding function in R that starts with “r” for random. All you have to do to obtain simulated values is specify the number of simulated values you want and the parameters of the function you are simulating from.

For example, let’s simulate 20 values generated from a standard normal distribution. We just run the rnorm command with the appropriate parameters, storing the values in a data vector:

```
> y<-rnorm(10,mean=0,sd=1)
> y
 [1]  1.37100652  0.46028398 -0.83283766 -1.56743758  1.24318977 -0.43508915
 [7] -1.64050749  0.08383233 -1.56016713 -0.45454076
```

Note that every run of the above simulation should produce different values (do not expect the same results as above).

Once you have simulated data you often want to look at them graphically. One way to look at the simulated values is to plot a histogram, which is very simply coded below:

```
hist(y)
```

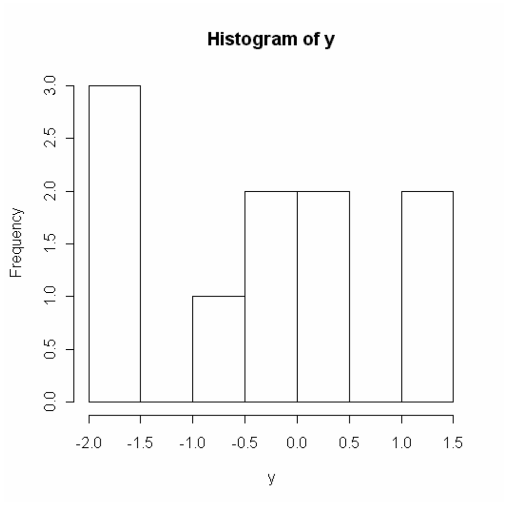


Figure 7-19

A histogram from a simulation of only 10 values is admiringly dull and usually hundreds or thousands of values are simulated. Let’s try the above simulation a few more times, this time with 50, 100, 500, and 1000 values:

```
> y1<-rnorm(50,mean=0,sd=1)
> y2<-rnorm(100,mean=0,sd=1)
> y3<-rnorm(500,mean=0,sd=1)
> y4<-rnorm(1000,mean=0,sd=1)
> par(mfrow=c(2,2))
> hist(y1,nclass=10,main="N=50")
> hist(y2,nclass=10,main="N=100")
> hist(y3,nclass=10,main="N=500")
> hist(y4,nclass=10,main="N=1000")
```

As you can see from the plots in Figure 7-20, the more values you simulate the closer the histogram will appear to the distribution you are simulating from. The distribution with N=1000 appears to approximate a continuous standard normal distribution quite nicely.

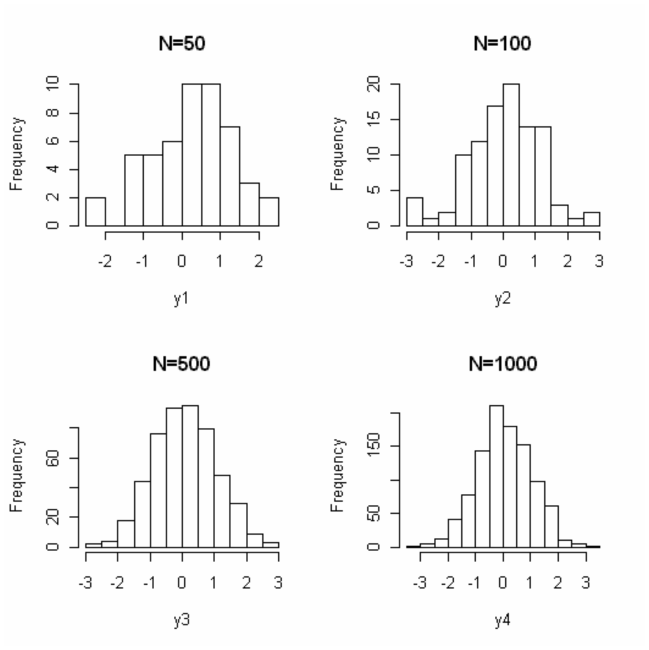


Figure 7-20: Increasing the number of simulations from a normal distribution

The reader is encouraged to try more simulation experiments and plots with the distributions discussed in this chapter. There will be much more discussion of simulations in coming chapters.

8

Probability and Distributions Involving Multiple Variables

The previous two chapters have looked at the basic principles of probability and probability distributions of one random variable. In this chapter we introduce some more concepts of probability and then extend looking at probability distributions to include distributions modeling two or more random variables.

Expanded Probability Concepts

Conditional Probability

Conditional probability is a powerful concept that allows us to calculate the probability of an event given that some prior event, which we have probability information about, has occurred. Using the concept of conditional probability allows us to solve problems where "things happen sequentially" with rather simple probability models, instead of complicated mathematical models that would be the alternative if it were not for conditional probability. Understanding conditional probability, as we will see in the next chapter, is an essential foundation for Bayesian statistics. But understanding the concept is also of importance on its own.

Let's illustrate the use of conditional probability with an example from classical genetics by considering the case of pea color as a trait encoded by one gene that has two alleles. The dominant allele, which we will denote by Y, codes for yellow pea color. The recessive allele we will denote by y, which codes for green pea color.

Assuming they are diploid, peas can have a genotype that is homozygote (yy or YY) or heterozygote (Yy or yY). Assuming equal frequencies of both alleles, the probability of a heterozygote is ½, and the probability of the homozygote is ½ as well. Peas of genotype yy are green, with a probability of ¼, and peas of genotypes Yy, yY, and YY are yellow, and therefore the probability of a yellow pea is 3/4.

Next, suppose we have a yellow pea, and want the probability that the pea is also a heterozygote. In terms of probability theory, we are restricting the sample space for the event pea color to the event that the pea is yellow, and creating a new sample space consisting only of yellow pea color. Within this new restricted space, we are asking what is the probability of the event heterozygous pea. In conditional probability jargon, we are conditioning the event of heterozygous pea on the event of yellow pea. The event yellow pea is our prior event that we already have information about.

Mathematically we can look at this example using language and notation from probability theory. We know from our basic genetic information that the probability of a pea being yellow is ¾ and we know that the probability of a pea being heterozygous and yellow is 2/4, based on the calculation that peas can be of Yy, yy, yY, and YY and 2 of four of these events (Yy and yY) are both yellow and heterozygous (we look more at joint probabilities later in this chapter). We can then use the joint probability of the event "heterozygous and yellow" and divide this by the event of "yellow" to calculate the probability of being heterozygous and yellow as follows:

$$\text{P(yellow and heterozygous)} = 2/4$$

$$\text{P(yellow)}=3/4$$

$$\text{P(heterozygous|yellow)} = \frac{\text{P(yellow and heterozygous)}}{\text{P(yellow)}} = 2/3$$

The notation P(heterozygous|yellow) is standard notation for conditional probability where the event being conditioned on comes after the "|" notation. P(A|B) is read as "the conditional probability of the event A given that the event B has occurred".

Using Trees to Represent Conditional Probability

Often looking at a graphical illustration helps to understand a concept. Trees are a visual way to represent conditional events and probabilities. Initial branches of the tree depend on the stem and finer branches depend on the previous branch. Let's use a simple tree diagram to illustrate our example (Figure 8-1).

The first branch of the tree represents the event of pea color and the second branch of the tree represents genotype (homozygous versus heterozygous) conditioned on the pea color.

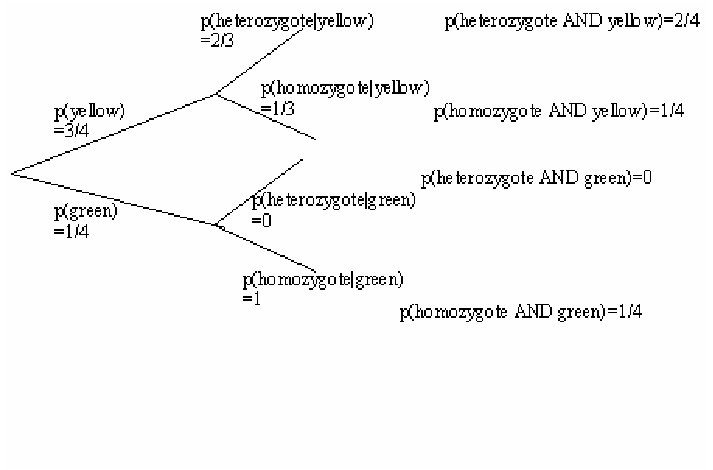


Figure 8-1: Conditional Probability Tree Diagram

Note that the second branch tree could have been written with branches for each of the four genotypes but is drawn here using the more simplified two branch version of heterozygous versus homozygous. The combined result at the end of the second branch is the joint probability of both events. Later in this chapter joint probability will be explained in more detail.

It is also important to observe in Figure 8-1 that probabilities within each set of branches add to 1. In the second branch, this stems from the fact that when we condition on an event, we define a new conditional sample space and the conditional probabilities obey the axioms and rules of probability within this new (conditional) sample space. In our example, the conditional sample space is based on pea colors.

Independence

It often happens that knowledge that a certain event E has occurred has no effect on the probability that some other event F occurs. In other words, the conditional probability of event F given event E is just the probability of event F. This can be written mathematically as $P(F \mid E) = P(F)$. One would expect that in this case, the equation $P(E \mid F) = P(E)$ would also be true. In fact each equation implies the other. If these equations are both true, then F is independent of E and this is formalized in the definition of independent events, which states that two events E and F are independent if $P(E|F)=P(E)$ and $P(F|E)=P(E)$.

Here is an alternative way to define independence. Two events E and F are independent if both E and F have positive probability and if $P(E \cap F) = P(E)P(F)$.

You may look at this and wonder, why? The logic for this alternative definition of independence comes from the definition of conditional probability:

$$P(E|F)=\frac{P(E\cap F)}{P(F)}$$

This can be algebraically rewritten as

$$P(E\cap F)=P(E|F)P(F)$$

But since we just defined the independence of E and F as $P(E|F)=P(E)$ this simplifies to

$$P(E\cap F) =P(E)P(F)$$

This form of the definition of independence comes in very handy for calculating joint probabilities of independent events.

It is important to note here that determining that events are independent is not equivalent to determining that events are disjoint or mutually exclusive, which was previously defined by two events having no common intersection ($P(E\cap F)= \varnothing$). Disjoint and mutually exclusive mean the same thing, but independence is a very different concept!

Independence can easily be extended to include more than two events. Three events A, B, and C are independent if $P(A\cap B\cap C)=P(A)P(B)P(C)$. In this case we can say that A, B, and C are mutually independent and independence of pairs of these events can be concluded (A is independent of B, B is independent of C, etc.). However, it is not always the case that the reverse is true, and it is possible to have three events, A, B, and C where A and B are independent, B and C are independent but A and C are not independent and therefore A, B, and C are not mutually independent.

In practice, determining whether events are independent can be tricky. Some times it's based on common logic. For example, most people would agree that the outcomes for each toss of a fair coin are independent, meaning the outcome of one toss of a coin (heads or tails) has no impact on the next toss of a coin. But in general, you should not assume independence without good reason to do so.

Independence is often utilized in bioinformatics in analyzing sequence information. Although this issue is often debatable, assuming independence of sequence elements is key in many data analysis algorithms commonly used. Independence makes calculations easy and the assumption of independence can greatly simplify a complicated algorithm.

For example, suppose nucleotides in a DNA sequence are mutually independent with equal probabilities (that is, $P(A)=P(T)=P(C)=P(G)=1/4$). The probability of observing a sequence ATCGA is simply P(A)P(T)P(C)P(G)P(A) or $(1/4)^5=1/1024$.

In the case above the nucleotides are assumed equally likely. However the concept of independence can easily be applied to the case of nucleotides of different frequencies. Suppose that P(C)=P(G)=1/3 and P(A)=P(T)=1/6. Then, assuming independence, the probability of sequence ATCGA is $(1/6)^3(1/3)^2$ which calculates to 1/1944. The importance here is that the event of a particular nucleotide in a sequence is independent of other nucleotides in the sequence, not that the probabilities of each nucleotide be the same.

In many instances, in sequence analysis or in analyzing other events, it is clear events are not independent. Two events that are not independent are said to be dependent. For example, in analyzing the nucleotide sequence for a start codon (ATG) or other sequence motif independence does not hold and the subsequent nucleotides are dependent on the prior nucleotide within that sequence motif.

Joint and Marginal Probabilities

Joint probability is a pretty self-descriptive concept – it's the probability of two (or more) events at once. Here we will look at the concept of joint probabilities, which will serve as preparation for coverage of joint distributions later in this chapter, but is also a subject of use on its own. Joint probability is officially defined as the probability of the intersection of two events. Joint probability was diagrammed with a Venn diagram in chapter 6 when we discussed the set theory concept of intersection. Recall that the intersection of two events uses the symbol "$\cap$". Using this notation, $P(A\cap B)$ symbolizes the joint probability of events A and B.

Tables are often used to display joint probabilities. For example, given a particular DNA sequence we obtain the following (hypothetical) joint probabilities for two adjacent nucleotide sites (Table 8-1):

Table 8-1: Joint Probabilities of Nucleotides at Adjacent Sites

▨	▨	**Nucleotide at position 1**			
Nucleotide at position 2		**A**	**T**	**C**	**G**
	A	0.2	0.1	0	0.1
	T	0	0.1	0.1	0.1
	C	0.1	0	0.1	0
	G	0	0.1	0	0

Although not immediately obvious to the untrained eye, a skilled probability practitioner can harvest from Table 8-1 a lot of useful information. Each table cell contains the probability of the intersection (joint probability) of events. For example, in the first cell the entry is the joint probabilitiy of nucleotide A in position 1 and nucleotide A in position 2. Therefore, the joint probability is 0.2. Fundamental to a joint probability table is the fact that all probability entries add up to 1, which is simply an expression of the axiom of probability that the probabilities of all events have to sum to 1.

What if we want to know the probability of nucleotide A being at position 1 regardless of the nucleotide at position 2? This calculation is the column total of the nucleotide A in position 1 column, or 0.3. This probability is called the marginal probability. Table 8-2 expands this idea calculating all the marginal probabilities for all columns (marginal probabilities for nucleotide at position 1) and all rows (marginal probabilities for nucleotide at position 2).

Table 8-2: Computing Marginal Probabilities

▨	▨	**Nucleotide at position 1**				Marginal probabilities for rows
Nucleotide at position 2		**A**	**T**	**C**	**G**	
	A	0.2	0.1	0	0.1	0.4
	T	0	0.1	0.1	0.1	0.3
	C	0.1	0	0.1	0	0.2
	G	0	0.1	0	0	0.1
Marginal probabilities for columns		0.3	0.3	0.2	0.2	1

Calculating conditional probabilities from the information in the table is also a breeze. For example, to calculate the conditional probability of a nucleotide at position 2 being T given the nucleotide at position 1 is a G we use the following formula from the definition of conditional probability…

$$P(E|F)=\frac{P(E\cap F)}{P(F)}$$

…and simply apply the formula to the desired conditions.

$$P\ (T\text{ at }P2\mid G\text{ at }P1)=\frac{P(T\text{ at }P2\cap G\text{ at }P1)}{P(G\text{ at }P1)}=\frac{0.1}{0.2}=1/2$$

Here, the joint probability, P(T at P2 $\cap$ G at P1), is obtained from the cell in Table 8-2 containing the joint probability for nucleotide G at position 1 and T at position 2, and P (G at P1) is the marginal probability of nucleotide G at position 1 obtained from the column total for that column in Table 8-2.

The Law of Total Probability

The law of total probability provides a method of calculating the probability of an event, which we'll denote by A, by conditioning on a set of mutually exclusive and exhaustive events, which we'll denote by $B_1, B_2, \ldots, B_n$. Note that the B_k are usually the different outcomes of a sample experiment (all possible events in a given sample space). Remember that mutually exclusive means that two events have no common intersection and that their joint probability is 0, that is $(Bi \cap Bj) = \varnothing$ for any i, j. Exhaustive means the entire sample space or union of all events, $B_1 \cup B_2 \cup \ldots \cup B_n$ = the sample space.

The law of total probability is best illustrated using the "Pizza Venn Diagram" in Figure 8-2, and can be summarized mathematically as follows:

$$P(A) = \sum_{i=1}^{n} P(A \mid B_i)P(B_i)$$

In the above formula and in Figure 8-2, A is the union of disjoint (mutually exclusive) sets, A∩Bi, for all i. P(A) can also be written as:

$$P(A) = \sum_{i=1}^{n} P(A \cap Bi)$$

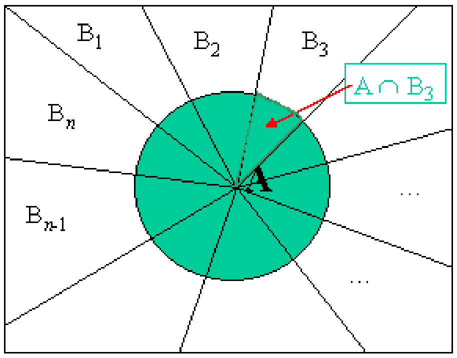


Figure 8-2: Illustrating the Law of Total Probability

Although the law of total probability may seem confusing, we just applied the law of total probability earlier in this chapter when calculating marginal probabilities in the adjacent nucleotide example. In this case we used the formula $P(A) = \sum_{i=1}^{n} P(A \cap Bi)$ where A is the nucleotide we are calculating the marginal probability of and the Bi's are the four nucleotides we are calculating the marginal probability over.

For example, to calculate the marginal probability of A in the first nucleotide position: P (A in position 1) = P(A in P1∩ A in P2) + P(A in P1∩ T in P2) + P(A in P1∩ C in P2) + P(A in P1∩ G in P2). Doing the math, P (A in P1) is 0.3.

Probability Distributions Involving More than One Random Variable

So far we have only considered the study of distributions where one random variable is modeled. However, modeling probability for most real life phenomena, and certainly most phenomena in bioinformatics, usually requires modeling the distribution of more than one random variable. Being able to use multiple variable distribution models provides us with a very powerful data analysis tool.

Examples of phenomena being modeled with more than one variable are abound in scientific applications. An environmental study may include measures of temperature, green house gases, moisture, and other conditions measured at different locations and each measure modeled by a random variable. In biochemistry it may be possible to develop a statistical model predicting tertiary protein structure using random variables to model factors such as percentages of certain amino acid residues or motifs. Understanding how to use probability distributions involving two or more random variables is key to being able to model data involving more than one measurable dimension.

Joint Distributions of Discrete Random Variables

Let's revisit our example of joint probability of nucleotides in two positions discussed earlier in this chapter. Previously we considered joint probabilities of events of a particular nucleotide being in position 1 and a particular nucleotide being in position 2. Let's step this up and model the scenario using random variables. Let X be the random variable to model the nucleotide at position 1 and Y be the random variable to model the nucleotide at position 2.

We can re-write our familiar table as a joint probability mass function (pmf) of two random variables (Table 8-3).

Table 8-3: Joint Probability Mass Function for Nucleotides at Two Adjacent Sites

▨	▨	**X= Nucleotide at position 1**			
Y=Nucleotide at position 2		**A**	**T**	**C**	**G**
	A	0.2	0.1	0	0.1
	T	0	0.1	0.1	0.1
	C	0.1	0	0.1	0
	G	0	0.1	0	0

This is a small but important change to the table. The table now models probability distributions for the two random variables. We can view each cell as representing P (X=xi, Y=yi)=P (X=xi ∩Y=yi), an individual value of the joint probability mass function denoted by $p(x_i, y_i)$.

Although we will not do so here, extending joint distributions to include more than 2 variables is pretty straightforward. We may extend the nucleotide example to include the distribution of the nucleotide in position 3 of a sequence. We could use the random variable Z to model third nucleotide. The probability of any given 3-nucleotide sequence such as ATG would be given by the joint distribution of random variables X, Y, and Z representing the respective probabilities of each nucleotide at each position. We write this as P ((X=A) ∩(Y=T)∩(Z=G)).

Marginal Distributions

Using the same logic used in calculating marginal probabilities described earlier, we can take the joint distribution and sum over all values of the other variable to create the marginal distribution of one variable. The only novel idea here is that we are assigning a random variable to the marginal probability, creating a marginal probability mass function for a discrete random variable.

For example, summing over all values for the second nucleotide position produces the marginal distribution or marginal probability mass function (pmf) of the first nucleotide, X, as depicted in Table 8-4.

Table 8-4: Marginal Probability Mass Function for X

X= Nucleotide at position 1			
A	**T**	**C**	**G**
0.3	0.3	0.2	0.2

Similarly, we could calculate the marginal probability mass function (pmf) of random variable Y by summing over all X values, as in Table 8-5.

Table 8-5: Marginal Probability Mass Function for Y

Y=Nucleotide at position 2	**A**	0.4
	T	0.3
	C	0.2
	G	0.1

The marginal distribution can be denoted using mathematical shorthand. For example, to denote the marginal probability of X we would write the following:

$$p_x(x_i) = \sum_y p(x_i, y)$$

This formula denotes the probability of X summed over all Y values in the joint distribution. Note that the sum of probabilities for each marginal distribution adds to 1 (obeying the law sum of all probabilities in a sample space sums to 1), which should always be the case (and serves as a good check for determining if you did the correct calculations).

Conditional Distributions

Sometimes we may be interested in the distribution of one variable conditional on a specified value of the second variable.

For example, we may be interested in the distribution of second nucleotides (Y) given that the first nucleotide is an A. For each nucleotide modeled by the distribution of Y, the conditional probability is calculated by using the conditional probability formula:

$$\text{P (Y=yi|X=A)=}\frac{\text{P(Y = yi} \cap \text{X = A)}}{\text{P(X = A)}}$$

In this formula, the numerator is the joint probability of the first nucleotide being A and second nucleotide being nucleotide yi. The denominator is the marginal probability of the first nucleotide being A.

For example, using previous data from Table 8-1, to calculate the probability that the second nucleotide is an A (Y=A) conditioned on the probability that the first nucleotide is an A we perform the following probability calculation:

$$P\,(Y{=}A|X{=}A) = \frac{P(Y = A \cap X = A)}{P(X = A)} = \frac{0.2}{0.3} = 0.66$$

Continuing this calculation for the other nucleotides, we obtain the conditional distribution of Y given X=A in Table 8-6.

Table 8-6: Conditional Distribution of Y given X=A

Y=Nucleotide at position 2 GIVEN X=A	**A**	0.66
	T	0
	C	0.33
	G	0

Again, note that the sum of conditional probabilities adds to 1 and fulfills the law of probability that the sum of probabilities of events in a sample space adds to 1. When calculating a conditional distribution we are redefining the sample space to a specific condition and redefining probability calculations to be valid within the new, redefined sample space.

Joint, Marginal and Conditional Distributions for Continuous Variables

Because of the importance of discrete data in bioinfomatics and the relative simplicity of working with discrete data, only discrete joint, marginal and conditional distributions have been discussed in detail. However, although most sequence analysis will concern itself with discrete data and distributions, other models often utilize continuous variables. Conceptually the joint, marginal, and conditional distributions are the same as for discrete variables, but the mathematics is more complicated.

The joint distribution of two continuous random variables can be modeled using a joint probability density function (pdf). The joint pdf of two continuous random variables X and Y is a two dimensional area A and can be evaluated by integrating over this area with respect to each variable for given values of X and Y

$$P((X,Y)\in A)=\iint_A f(x,y)dydx$$

Since evaluating this requires integration techniques from multivariable calculus, we will not evaluate such integrals here. But conceptually the probability that (X, Y) lies in area A is equal to the volume underneath the function f(x,y) over the area A.

Calculating a marginal distribution of a continuous variable is similar to calculating a marginal discrete random variable distribution. In the case of the discrete random variable, this is done by summing over the other variable(s) whereas in the case of a continuous random variable, this is done by integrating over the other variable(s).

For example to determine the marginal pdf of X given the joint distribution of continuous random variables X and Y, integrate over the distribution of Y (which if X and Y were discrete would be summing over all distribution of Y) as follows:

$$f_x(x)=\int_y f(x,y)dy$$

Again the calculus of computing these distributions is beyond our coverage here, but a conceptual understanding of how to compute a marginal distribution for a continuous variable is important and seeing how discrete and random variable distribution concepts are very similar is important as well.

The conditional probability distribution for two continuous random variables can also be calculated using some simple calculus. If X and Y have joint probability density function f(x,y), then the conditional probability density function of X, given that Y=y, is defined for any values as the joint probability of X and Y divided by the marginal probability that Y=y. This can be written mathematically where the conditional distribution is denoted by $f_{x|y}(x \mid y)$.

$$f_{x|y}(x \mid y)=\frac{f(x,y)}{f_y(y)}$$

Working with more than two continuous random variables is a simple extension of the concepts and techniques presented here for two random variables. The analytical methods and calculus for performing such calculations can become quite tedious. However, using a computer program can greatly simplify these types of calculations as well as perform simulations from complex distributions and their derived distributions. Many examples presented in this book will perform the task of working with high-dimensional distributions using the computational power of R.

Graphical models beyond two random variables are not very practical, so among other applications, marginal and conditional distributions are often used to look at graphics of higher dimensional distributions, as we shall in some examples in the next section of this chapter.

Common Multivariable Distributions

Distributions of more than one random variable are extensions of univariate distributions. The distributions presented here are not should not seem entirely novel, because they build on univariate distributions presented previously by including more than one random variable in the model. A solid understanding of the univariate binomial, normal, and beta distributions (which can be reviewed in the previous chapter) is the foundation for understanding the three distributions we will look at here: the multinomial, the multivariate normal, and the Dirichlet. These three distributions are selected because they are the key multivariable distributions used in modeling data in bioinformatics.

The Multinomial Distribution

The multinomial distribution is the most commonly used discrete, high-dimensional probability distribution. The multinomial is an extension of the binomial distribution. Instead of just two possible outcomes (as in the case of the binomial), the multinomial models the case of multiple possible outcomes.

Consider an experiment that models the outcome of *n* independent trials. Each trial can result in any of *r* different types of outcomes (compared to just r=2 in the binomial case). The probability of any of the outcomes is constant, just as in the binomial model the probability of success and probability of failure were held constant for a particular model. These probabilities are denoted by $p_1,p_2,\ldots,p_r$ and the sum of the probabilities for all outcomes sums to one, that is $p_1+ p_2 +\ldots+ p_r = 1$.

If we count how many outcomes of each type occur, we have a set of *r* random variables $X_1,X_2,\ldots,X_r$. Each X_j = the number of outcomes of the j^{th} type (where j=1 to r) and the actual counts are values of each random variable, denoted by $X_j = x_j$, etc. Note the sum of the values of the random variables is n, the total number of trials, that is $x_1+x_2+\ldots+x_r = n$.

Because we are dealing with a series of independent events, any particular sequence of outcomes consists of x_1 of the first kind, $x_2$of the second kind, etc and has probability

$$p_1^{x_1} p_2^{x_2} \cdot \ldots \cdot p_r^{x_r}$$

Using combinatorics, we can calculate the number of possible divisions of n sequences into r groups of size x1, x2…xr with what is called the multinomial coefficient. This can be written as:

$$\binom{n}{x_1, x_2, \ldots x_r} = \frac{n!}{x_1! x_2! \ldots x_r!}$$

Combining these results produces the joint distribution of observed events (a formula that directly parallels the binomial case of two possible outcomes described in the previous chapter) under the multinomial model.

$$p(x_1, x_2, \ldots, x_r) = \binom{n}{x_1 x_2 \ldots x_r} p_1^{x_1} p_2^{x_2} \cdot \ldots \cdot p_r^{x_r}$$

Among its many applications in bioinformatics, the multinomial model is frequently used in modeling the joint distribution of the number of observed genotypes. Any number of loci and any number of alleles can be modeled this way, but the simplest example is the case of looking at a genetic locus which has two alleles, A and a. If we sample n diploid individual in the population and record their genotype at that locus, a number of individuals will be of genotype AA, which we can represent as just as n_{AA}. Likewise, a number of individuals will have Aa genotype and can be represented by n_{Aa}, and the number of individual of aa genotype can be represented by n_{aa}. To formalize this into a probability model, we can use the random variable X to represent n_{AA}, the random variable Y to represent n_{Aa}, and the random variable Z to represent n_{aa}. We can label these proportions (probabilities) as P_{AA}, P_{Aa}, and P_{aa} for each of the three respective possible genotypes.

The multinomial distribution formula represents the joint distribution of the three genotypes is given below.

$$P\ (X=n_{AA}, Y=n_{Aa}, Z=n_{aa}) = \frac{n!}{n_{AA}!\, n_{Aa}!\, n_{aa}!} (P_{AA})^{n_{AA}} (P_{Aa})^{n_{Aa}} (P_{aa})^{n_{aa}}$$

Since you probably wouldn't want to perform paper and pencil calculations using this formula, the question now is how would you work with such a model in R? Clearly models with 3 random variables are not as simple to work with as univariate models, but R can handle analyzing and performing simulations on these more complicated distributions quite easily.

As an example, suppose we have 20 individuals and genotype them and find that $n_{AA}=4$, $n_{Aa}=14$, and $n_{aa}=2$. Given this information, we can easily estimate our parameters for the multinomial distribution by simply using the sample proportions $P_{AA}=0.2$, $P_{Aa}=0.7$ and $P_{aa}=0.1$. Since we do not have a lot of data it is difficult to examine the properties of this model. However, using our empirical parameters we can extend our data set by doing simulations of more values to look at graphs and other details of the distribution. Later in chapter 13 we will do similar types of simulations using a technique called bootstrapping.

To perform simulations, we can write a simple function in R that generates values of the three random variables (genotype counts) from a multinomial distribution given the parameter values of the proportions of each genotype:

```
#function for drawing random values
#from a multinomial distribution

#takes as parameters
## parameter N number of simulations
## parameter n is number of trials simulated (sample size)
## parameter p is a vector of proportions

rmnomial_function(N,n,p){
    l<-length(p)
    x<-rbinom(N,n,p[1])
    if(l==2)
            {cbind(x,-x+n)}
    else
            {cbind(x,rmnomial(N,-x+n,p[2:1]/sum(p[2:1])))}

}
```

To illustrate the use of this function, let’s perform 10 simulations of 20 individuals using our empirical parameters for the proportion vector.

```
> ## Define N to be 10 trials
> N<-10
> ## Define n to 20
> n<-10
> ## Define our p vector containing empirical values
> # pAA=0.2, pAa=0.7, paa=0.1
> p<-c(0.2,0.7,0.1)
> ## Call function with these parameters, store in results
> results<-rmnomial(N,n,p)
```

This produces the following matrix of simulated values of the three random variables we are modeling:

```
> results

 [1,] 4 11 5
 [2,] 4 13 3
 [3,] 2 12 6
 [4,] 3 13 4
 [5,] 4 13 3
 [6,] 6 11 3
 [7,] 3 13 4
 [8,] 4 15 1
 [9,] 1 13 6
[10,] 7  7 6
```

We could easily write some code to calculate proportions for the values for the simulated random variable values:

```
> results2<-results/(results[,1]+results[,2]+results[,3])
> results2

 [1,] 0.20 0.55 0.25
 [2,] 0.20 0.65 0.15
 [3,] 0.10 0.60 0.30
 [4,] 0.15 0.65 0.20
 [5,] 0.20 0.65 0.15
 [6,] 0.30 0.55 0.15
 [7,] 0.15 0.65 0.20
 [8,] 0.20 0.75 0.05
 [9,] 0.05 0.65 0.30
[10,] 0.35 0.35 0.30
```

Looking at the proportions makes it clearer that the simulated values are indeed based on the empirical proportion parameters (0,2,0.7,0.1) supplied.

You could write your own functions like the above to sample from multinomial distributions, but there is a package called combinat that contains some pre-written functions to sample from multinomial distributions. This package also contains a number of other functions useful in combinatorial calculations.

Note that if you were interested in doing some statistical tests, you could simulate values from distributions with alternative parameters, and then perform tests to determine whether the empirical values differ from this theoretical distribution. For example, you could test the empirical values against a theoretical population with parameters P_{AA}=0.25, P_{Aa}=0.5, and P_{aa}=0.25. This will not be done here because it requires techniques of inferential statistics not yet discussed, but is presented here to illustrate some of the powerful applications you can perform using simulations of distributions.

The marginal distributions for each of the random variables X, Y and Z can easily be obtained from the multinomial. Suppose we are interested only in the marginal probability mass function of the random variable X? We could go about finding the marginal probability mass function using lots of messy algebra or instead we consider the following argument.

If we are only interested in the number of outcomes that result in the first type, *X*, then we simply lump all the other types (Y and Z) into one category called "other". Now we have reduced this to a situation we have two outcomes. This should ring a bell of familiarity, as it has now become a case of Bernoulli trials. The number of times genotype X occurs resulting from *n* independent trials follows a Binomial probability distribution with parameters *n* and p_1. Note that the probability of "failure" = prob("other") = 1 - p_1 = p_2 +…+ p_r (sum of all the others). Thus, the one-dimensional marginal distribution for a multinomial distribution is simply a binomial:

$$p_{X_j}(x_j) = \binom{n}{x_j} p_j^{x_j} (1-p_j)^{n-x_j}$$

To examine this more concretely, let's continue our example and do some illustrative simulations. Only now let's just be concerned with the first variable, X. Instead of considering both Y and Z variables, let's consolidate them into a variable W, representing "everything else" which is not genotype AA, and make W=X+Y=(n_{Aa}+n_{aa}) and pW=(1-pAa-paa)=0.8. We now have a binomial model that models X (counting the trials of AA as "successes" and all other outcomes as "failures") alone. Using R, let's simulate this distribution in two ways and compare the results.

First let's simulate 10,000 values using the multinomial model and all 3 random variables. To do this, repeat the simulation described above but changing parameter N to 10,000.

```
> results<-rmnomial(10000,20,c(0.2,0.7,0.1))
> ## Change results to proportions
> results2<-results/(results[,1]+results[,2]+results[,3])
> ## Store column 1 of results in X
> X<-results2[,1]
> ## Store column 2 and 3 results in W
> W<-(results2[,2]+results2[,3])
```

Summarize values by getting the means for X and W:

```
> mean(X)
[1] 0.19913
> mean(W)
[1] 0.80087
```

The mean for X is roughly 0.2 and the mean (the proportion of "successes" in the binomial) and the mean for W is 0.8 (the proportion of "failures" in the binomial). These values should seem logical.

To look at the result visually, plotting the proportions with a histogram is simple, as coded below and shown in Figure 8-3.

```
>  hist(X,nclass=10,main="Simulated prop. AA using Multinomial")
```

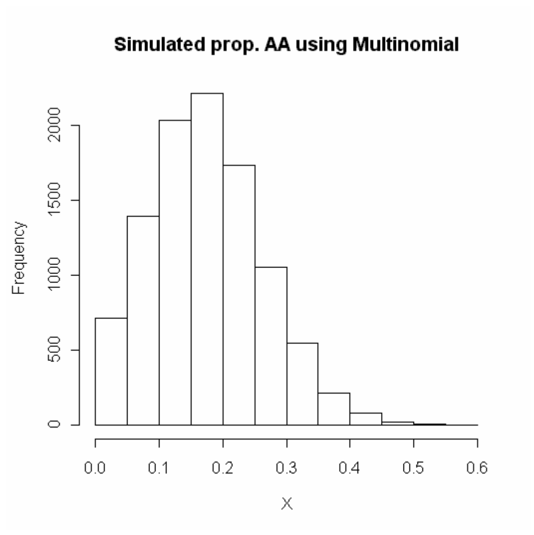


Figure 8-3

For comparison, a quick simulation from the binomial of 10,000 values using p=0.2 is simple to code using the rbinom function:

```
> ##Simulate 10000 values from binomial
> ##Simulate rv for n=20 and p=0.2
> B<-rbinom(10000,20,0.2)
> ##Convert to proportions by dividing by 20
> hist(B/20,nclass=10,main="Simulated prop. AA using Binomial")
```

The histogram using the binomial simulation is shown in Figure 8-4. It should be apparent from the graphs that the distributions in Figure 8-3 and Figure 8-4 are virtually identical.

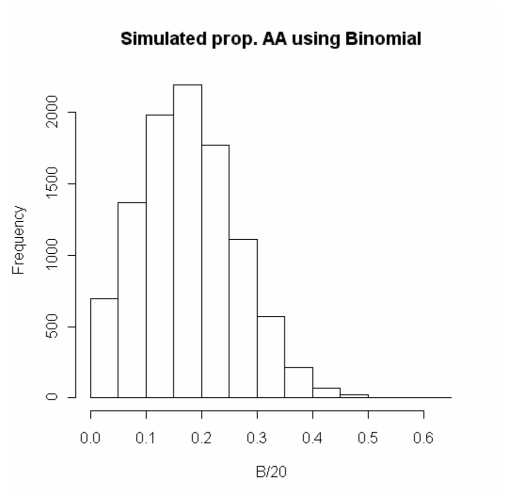


Figure8-4

Multivariate Normal Distribution

The multivariate normal is a favorite distribution in multivariate statistics. Often data are mathematically transformed to fit a normal model. Such transformation of data can be somewhat controversial, because sometimes they are performed for convenience and simplicity and the results may be misleading or difficult to translate back to the original variables. Nonetheless, most of inferential multivariate statistics utilize the multivariate normal. Understanding how the normal distribution can be extended to include more than one random variable is important.

Because most of the mathematical aspects of dealing with the multivariate normal involve advanced techniques of calculus and matrix algebra, we will consider the details of only one limited example of the multivariate normal. We will consider the bivariate normal model, which models the joint distribution of two independent normally distributed random variables.

Recall, from the previous chapter, the mathematical formula for the normal distribution:

$$\mathrm{f}(\mathrm{x}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

We can model their joint distribution of the two random variables X and Y by simply looking at the product of the marginal distributions of their two marginal distributions since they are independent variables. Thus the model for the joint distribution of two independent normally distributed random variables X and Y (aka: the bivariate normal) is:

$$f(x, y) = f_x(x) f_y(x)$$

We can write this by using the normal distribution equation for both random variables and multiplying them:

$$\mathrm{f}(\mathrm{x},\mathrm{y}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-\mu)^2}{2\sigma^2}}$$

Let’s now take a look at what this looks like graphically using R. Let’s reduce the case even further by considering only a standard bivariate normal distribution. Recall that the standard normal distribution has mean of 0 and standard deviation of 1. Thus, we can further re-write the equation above considering X and Y as standard normal random variables, each having mean 0 and standard deviation 1:

$$f(x,y)=\frac{1}{\sqrt{2\pi}}e^{-\frac{(x)^2}{2}}\frac{1}{\sqrt{2\pi}}e^{-\frac{(y)^2}{2}}$$

By performing some algebra, this can be simplified to:

$$f(x,y)=\frac{1}{2\pi}e^{-\frac{(x^2+y^2)}{2}}$$

This is a pretty workable equation to plot. To do this, we can write a function in R to simulate draws from a bivariate standard normal. In the code below we create two vectors of length 20, x and y and then write a function to calculate values of f(x,y). The results of the function call are stored in a variable matrix z. Then the data can be viewed using a simple 3-d perspective plot, presented in Figure 8-5.

```
> x<-seq(-2,2,length=20)
> y<-x
> bvn_function(x,y){
+ (1/2*pi)*exp(-0.5*(x^2+y^2))
+ }
> z<-x%*%t(y)
> for(i in 1:20){
+ for(j in 1:20){z[i,j]<-bvn(x[i],y[j])}}
> persp(x,y,z)
```

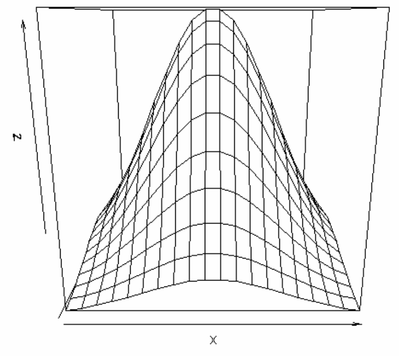


Figure8-5: Perspective Plot of Bivariate Standard Normal Distribution

Figure 8-5 is shaped like a symmetric upside down cone with its base on the XY plane. Along the X-axis (front view of the plot) the perspective lines show a pattern of the normal distribution reflecting that any "slice" of the cone would have the shape of a normal distribution.

Once again, we didn't have to write a function, because random generations of multivariate normal distributions of any dimension can also be done using the package mvtnorm. Mvtnorm contains functionality for generating random values from all kinds of multivariate normal distributions, and from related multivariate t distributions for any model, and with any parameter values, not just with standard parameter values. Let's use the package's function rmvnorm to generate 1000 values from a bivariate standard normal distribution:

```
> data<-rmvnorm(1000,mean=c(0,0))
> data
                 [,1]          [,2]
   [1,] -0.9152555414  0.5950708803
   [2,] -1.2240565493  0.3079036163
   [3,] -1.2205942482 -0.9042616927
…
  [999,]  2.0735458497 -1.7003787054
 [1000,] -0.0962237236  0.0056516042
```

In our result we have a matrix of data where the first column has values of random variable X values from standard normal simulation, and the second column is random variable Y values from a standard normal simulation. A scatter plot in Figure 8-6 demonstrates that the joint density is shaped as we would expect data from a bivariate normal simulation to be shaped.

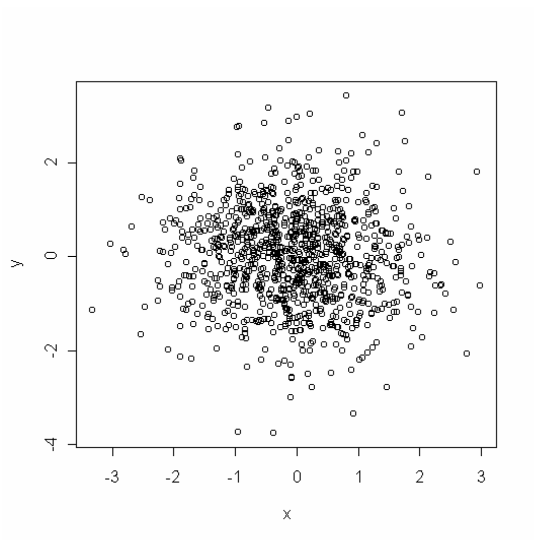


Figure8-6: Scatter plot of X and Y values from bivariate standard normal simulation

If we want to look at marginal distributions of X and Y, we can break down the data matrix into an X and Y matrix and look at the marginal distributions of X and Y very easily. All we have to do is store the X and Y values in new variables (although we technically don't even have to do this and could have just plotted the columns from the matrix) and do a histogram plot of the marginal distribution of interest.

```
> x<-data[,1]
> y<-data[,2]
> hist(x,nclass=20,main
+ ="Marginal distribution of x")
```

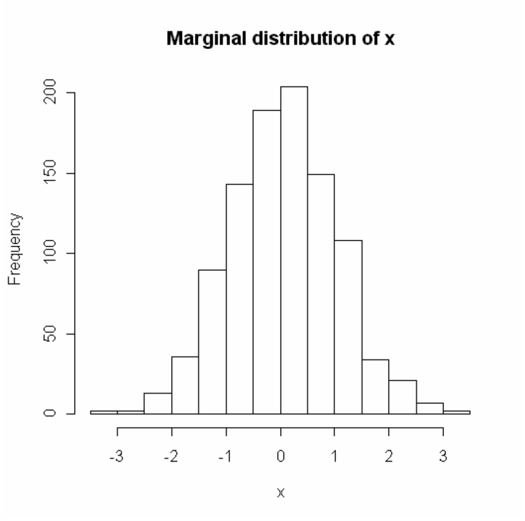


Figure 8-7: Marginal Distribution of X

From the histogram in Figure 8-7, the marginal distribution of X is quite normally distributed. If we did a simulation of 1000 values from a standard univariate normal distribution we would obtain a virtually identical result.

Dirichlet Distribution

The Dirichlet is the multivariable version of the beta distribution. Recall that the beta distribution is the distribution often used to model data in the form of proportions, i.e. values between 0 and 1. It should make sense that if you are modeling something such as the proportions of nucleotides in a DNA sequence, each proportion (A, T, C, G) can be modeled with an individual random variable, and the joint distribution for the proportions of all four nucleotides can be modeled using a Dirichlet.

We now denote by $X_1 \ldots X_k$ a set of proportions noting that $X_1 + \ldots + X_k = 1$ (and each $X_i > 0$). Mathematically the formula for the Dirichlet distribution, for k random proportions is:

$$f(X_1, X_2, ..X_K) = \frac{\Gamma(\sum_{i=1}^{k} \alpha_i)}{\prod_{i=1}^{k} \Gamma(\alpha_i)} \prod_{i=1}^{k} X_i^{\alpha_i - 1}$$

Although this formula may look intimidating, it's not all that complicated! The Γ is just the symbol for the gamma function presented in chapter 6 and used in both the gamma and beta distributions in chapter 7. Capitol sigma $\sum$ is the symbol for addition (sum of), and the capitol pi, Π is the symbol for multiplication. The alpha's are the parameters for the random variable X's.

The constraint on the distribution is that the sum of proportions adds to one, that is, $\sum_{i=1}^{k} X_i = 1$ which should seem logical since when modeling the joint proportions of nucleotides in a sequence, for example, the total of all proportions is always 1. This follows the usual law that the total probability of all events in a sample space is 1.

As a simple example where we might use the Dirichlet, let's model the joint proportions of purines (A or G) and pyrimidines (C and T) in a given sequence. Let's use X1 to model the proportion of purines, and X2 to model the proportion of pyrimidines. Let's use the arbitrary choice of alpha=1 as a parameter for X1 and alpha=2 for X2. We model the proportion of purines as p1 and the proportion of pyrimidines as p2. Mathematically, with k=2 since there are two random variables being modeled, the joint distribution Dirichlet model is:

$$f(X_1, X_2) = \frac{\Gamma\left(\sum_{i=1}^{2} \alpha_i\right)}{\prod_{i=1}^{2} \Gamma(\alpha_i)} \prod_{i=1}^{2} X_i^{\alpha_i - 1}$$

Simplifying the above expression by substituting in the alpha values and p_i's and then performing addition and multiplication produces a formula that is not very intimidating and just involves algebra and calculation of a few gamma functions:

$$f(X_1, X_2) = \frac{\Gamma(3)}{\Gamma(1)\Gamma(2)} (p1)^{1-1} (p2)^{2-1}$$

Note that since $X_2 = 1 - X_1$ we could simply view this as a marginal distribution of X_1 because for any given X_1, X_2 would be completely determined.

$$f_{X_1}(x_1) = \frac{\Gamma(3)}{\Gamma(1)\Gamma(2)} (x_1)^{1-1} (1 - x_1)^{2-1}$$

We note that this is simply the Beta distribution for X_1.with parameters α=α1=1, and β=α2=2. Therefore the joint Dirichlet distribution for two random proportions X_1,X_2 ($X_1 + X_2 = 1$) is equivalent to the univariate Beta distribution for X_1 alone. Although in this example only two random variables are modeled and the model is pretty straightforward, when more random variables are modeled the Dirichlet can become quite complex. Because calculating the gamma functions in this equation can be computationally intense (even for quite high powered computers), sometimes the distribution is evaluated by taking logarithms (which make it computationally more efficient). It actually doesn't matter which base you use for your log calculations as long as you are consistent. The right side of the Dirichlet model can be calculated using logarithms like this:

$$=\log\left(\Gamma\left(\sum_{i=1}^{k}\alpha_i\right)\right)-\log\left(\prod_{i=1}^{k}\Gamma(\alpha_i)\right)+\log\left(\prod_{i=1}^{k}X_i^{\alpha_i-1}\right)$$

Another computation quirk of the Dirichlet distribution is that it is simpler to sample from the Dirichlet indirectly by using a method that draws k independent gamma samples and then computing random proportions (X_i) as the value of each sample divided by the sum of the k samples. It can be shown that the proportions X1,..,Xk have a Dirichlet distribution. We will not discuss mathematical details of this here, but in computer programs and in literature you will see Dirichlet being simulated using draws from the gamma distribution so it of interest to note this here and this trick is used in the code example below.

To simulate from the Dirichlet in R you could write your own function. The code below gives an example of a function that simulates n draws with a parameter vector of p:

```
rDir_function(n,a){
    l<-length(a)
    m<-matrix(nrow=n,ncol=p)
    for(i in 1:p){m[,i]<-rgamma(n,a[i])}
    sum<-m%*%rep(1,p)
    m/as.vector(sum)
}
```

Using this function to simulate values for n=20 with a vector of parameters (alpha values) for 3 random variables where alpha=1 for all three variables produces matrix of results where each column represents simulations for each random variable:

```
x<-rDir(20,c(1,1,1))
> x
             [,1]         [,2]       [,3]
 [1,] 0.0018936588 8.005894e-01 0.19751690
 [2,] 0.3344705893 1.894494e-03 0.66363492
 [3,] 0.0372989768 2.437322e-02 0.93832781
 [4,] 0.5159592455 6.830641e-03 0.47721011
…
```

Again there is no need to write a function to perform simulations, because the package gregmisc contains pre-written functions for computing density of or generating random values from the Dirichlet distribution. Performing the same computation as above using the function rdirichlet from this package produces the following:

```
>  x <- rdirichlet(20, c(1,1,1) )
> x
            [,1]        [,2]       [,3]
 [1,] 0.74226607 0.034630906 0.22310302
 [2,] 0.61271253 0.359267638 0.02801983
 [3,] 0.20446723 0.180993424 0.61453934
 [4,] 0.77386208 0.004850972 0.22128695
 …
```

Let's return to our example of modeling the joint distribution of the proportion of purines and pyrimidines and simulate 1000 values using the rdirichlet function:

```
> x<-rdirichlet(1000,c(1,2))
```

If we look at the mean values simulated for p1 and p2, these are the simulated proportions of x1 purines and x2 pyrimidines given that our parameters alpha=1 and alpha=2.

```
> mean(x[,1])
[1] 0.3354725
> mean(x[,2])
[1] 0.6645275
```

Each proportion has a marginal distribution. If we plot the marginal of x[,1] we get the following as depicted in Figure 8-8:

```
> hist(x[,1],nclass=20,main="Marginal of x[,1]")
```

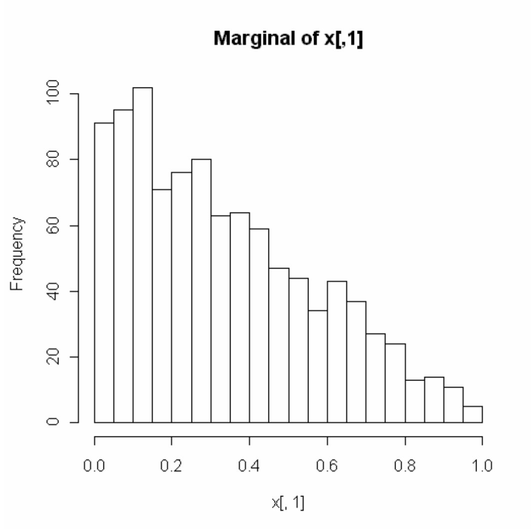


Figure 8-8

Likewise if we plot the marginal of x[,2] we get the plot in Figure 8-9

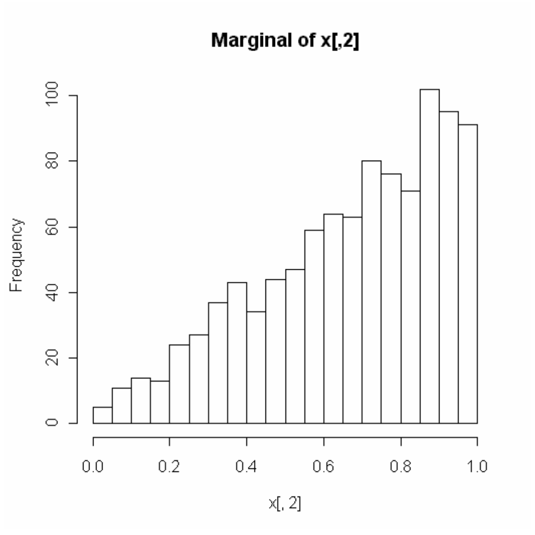


Figure 8-9

The Dirichlet is sometimes used on its own, but is used extensively in Bayesian statistics in conjunction with the multinomial distribution to form powerful models for working with counts and proportion data. It is important here to grasp the basics of the Dirichlet and its mathematical model and how to draw simulations from the Dirichlet using R.

Based on the discussions in chapters 6, 7, and 8 you should have an understanding of the basics of probability theory and a feel for working with univariate distributions and some select distributions of more than one variable. If you can understand the different models, what they are used for, how to perform simulations and how to graphically analyze results, you should be prepared to utilize these capabilities in some more advanced applications. The next few chapters contain an introduction to Bayesian statistics and an overview of Markov Chain methods, and coming discussions will build upon the concepts and methods presented thus far.