Lab 2 **Statistics for Laboratory Scientists**
Due: 22 Feb 2006

Introduction

In this lab, we will consider a couple of probability-related tasks in R: computer simulations and calculations of probability distributions. The simulation-related tasks involve quite complex R code. Some students may want to learn enough R to completely understand this code and write their own code of similar complexity. Others may wish to take the code for granted and focus entirely on its use and the results it produces. The direction you take is entirely up to you. We hope that the complex code does not get in the way of your understanding of (or your pleasure in) this lab.

Regarding the questions whose responses should be handed in: in this lab, they will be typed in bold (and marked with the symbol $\Longrightarrow$) within the lab, and then collected again on the last page.

The code for this lab will again be available at the following webpage.

```
http://www.biostat.jhsph.edu/~kbroman/teaching/labstat/third/labs.html
```

For this lab, we will separate out the code for the major *functions* that will be used from the other code that you will be typing in.

The file `func2.R` contains the major functions. Download this file to `c:\func2.R`, or some other convenient location. I'd recommend going to the above page, right-clicking on the file, and selecting "Save Target As…"

When you start R, type the following command, replacing `c:/` with whatever is appropriate, considering the directory in which you placed the file.

```
source("c:/func2.R")
```

Alternatively, you may "`source`" the file directly from the web as follows:

```
source("http://www.biostat.jhsph.edu/~kbroman/teaching/labstat/third/func2.R")
```

If you type `ls()` you should now see a number of new objects—the functions that we will be using.

The file `lab2.R` contains all of the other code. As with the first lab, you may wish to download this file (saving it as `c:\lab2.R`) and open it within R using, from the menu bar, File → Display file, so that you may copy and paste commands from the file into the R console window.

Functions in R

In R, we use *functions* to perform all sorts of tasks, including calculations, the manipulation of data, and graphics. One of the advantages of R is that you can create your own functions, or write modified versions of built-in functions, in order to more efficiently perform complex tasks.

A new function can be defined using the function `function`, as in the following example—a function to calculate the coefficient of variation of a vector (CV = SD/mean).

```
cv <-
function(x, na.rm=FALSE)
{
  m <- mean(x, na.rm)
  s <- sd(x, na.rm)
  s/m
}
```

The stuff within the parentheses after `function` are the functions *arguments*. The use of `na.rm=FALSE` specifies the *default value* of that argument—if, when the function is used, the value of this argument is not specified, the default

value will be assumed.

The value of the last command inside the function is what gets "returned." Note the use of brackets `{}` in order to define the set of commands that comprise the function.

To define this function within R, one can either type the above commands in the R console window, or type the commands into a text file and use the function `source` to load it into R. The latter is better, because the function can be more easily modified, plus it will be saved for future use.

To use this function, type it into R or copy and paste it from the file `lab2.R`, and then type, for example:

```
x <- runif(100)
cv(x)
```

The first line generates 100 random numbers between 0 and 1; the second line calculates the coefficient of variation of those numbers.

A population genetics simulation

In this section, we will use R to perform computer simulations of the genetic drift in the allele frequency at a locus, in a small population, under quite unrealistic assumptions. Our primary aim is to get an appreciation for the value of R for computer simulation.

Consider a population of $2m$ diploid individuals, composed of m males and m females. Suppose that generations are non-overlapping, and that at each generation, the individuals form, at random, m couples, and that each couple produces exactly one male and one female offspring. (These are indeed completely unrealistic assumptions, but hopefully this will still be interesting to you.)

We are interested in following the allele frequency at a diallelic, autosomal locus. Suppose the alleles are *A* and *a*, and let us keep track of the proportion of the *a* alleles in the population.

Suppose further that the genotypes of individuals in the initial generation are assigned at random, according to the probabilities for Hardy-Weinberg equilibrium: $[(1-p)^2,\ 2p(1-p),\ p^2]$ for genotypes *AA*, *Aa* and *aa*, respectively, where p is the frequency of the *a* allele.

Simulating the initial generation

The function `init` can be used to simulate the genotypes for the individuals in the initial generation. In order to save space, I won't include the code here; look at the file `func2.R` or type the name of the function within R.

The function takes as input two arguments: `m` is the number of pairs of individuals and `p` is the allele frequency with which to simulate the initial generation. The output is a matrix with two columns—the genotypes of the males and the females, respectively. The numbers in the matrix are 0, 1 or 2, according to whether the individual has genotype *AA*, *Aa*, or *aa*, respectively. Note that the function `sample` is used to simulate the actual genotypes; `repl=TRUE` is used to specify "sample with replacement" and `prob=genop` is used to specify the sampling probabilities for the elements of the vector `0:2`.

Type `dat <- init(25,0.2)` to simulate an initial population with 15 pairs of individuals, using an allele frequency of 20%. Type `dat` on its own to see the result. Type `table(dat)` to get the observed genotype frequencies. `mean(dat)/2` returns the observed allele frequency. Repeat this a few times.

Let's look at how much the observed allele frequency in the initial population varies. The following may be used to get the observed allele frequency in 1000 replicates of an initial population of 25 pairs of individuals, generated using an allele frequency of 20%.

```
freq <- 1:1000
for(i in 1:1000)
  freq[i] <- mean(init(25,0.2))/2
```

```
hist(freq, breaks=seq(0, 1, 0.01))
abline(v=0.2, lwd=2)
```

The first command is used to create a vector in which to place the simulation results. We then use a `for` loop to repeatedly call the function `init`. `hist` plots a histogram of the results; the argument `breaks` is used to specify the endpoints of the bins in the histogram. `abline` is used to plot a vertical line at the allele frequency used in generating these individuals.

Repeat the above for $m = 250$ pairs of individuals, and for an allele frequency of $p = 50\%$.

$\Longrightarrow$ **How are the results different for different sample sizes and different allele frequencies?**

Simulating further generations

We now consider two functions which may be used to simulate future generations. The first function, `simkids`, is used to simulate the genotypes for a pair of children, given the genotypes of the parents. The input is a vector of length 2, each element of which is expected to be 0, 1, or 2, corresponding to the genotypes of the parents. The output is also a vector of length two—the genotypes for the two children.

The second function, `simnewgen`, is used to simulate the genotypes for the next generation, given the genotypes of the individuals in the current generation. In the function, the genotypes of the males are shuffled, corresponding to the random choices of mates. Then the `apply` function is used to pass *one row* of the data at a time to the function `simkids`, to simulate the generation of two offspring for each couple. The output of `simnewgen` is of the same form as the input: a matrix with two columns, whose elements are 0, 1, or 2, according to the genotypes of the individuals.

Let us simulate 30 generations using the above functions, and save the observed allele frequency in each generation.

```
freq <- 1:30
dat <- init(25,0.2)
freq[1] <- mean(dat)/2
for(i in 2:30) {
  dat <- simnewgen(dat)
  freq[i] <- mean(dat)/2
}
plot(freq, type="l", ylim=c(0,1), lwd=2)
```

Note the use of brackets `{}` in the `for` loop. These must be used in order to have multiple commands within each iteration of the loop.

In order to more easily repeat these simulations, it'd be better to create a function to do all of this. The function `simfreq` has input arguments `n`, the number of generations; `m`, the number of couples in each generation; and `p`, the allele frequency used to generate the initial population. It returns a vector containing the allele frequencey at each generation.

Let's add 25 additional "realizations" of this simulation (of 30 generations) to our plot. Note how these different realizations are more spread out at 30 generations than they were at the initial generation.

```
plot(freq, type="l", ylim=c(0,1), lwd=2)
for(i in 1:25)
  lines(simfreq(30, 25, 0.2), col="gray")
lines(freq, lwd=2)
```

Plot the results for 100 generations, using both 25 couples and 100 couples. Repeat this several times, so that you can get a sense of what is going on. Here is some code that you might use:

```
freq.sm <- simfreq(100, 25, 0.2)
freq.lg <- simfreq(100, 100, 0.2)
plot(freq.sm, type="l", ylim=c(0,1))
```

```
lines(freq.lg, col="red")
```

$\Longrightarrow$ **How is the case of 100 couples different than the case of 25 couples, in terms of the fluctuation in the observed allele frequency across time/generation?**

Allele fixation

Plot the results for several simulation replicates, following a population of 25 couples per generation out to 500 generations, again using 20% allele frequency to generate the initial population.

```
plot(simfreq(500, 25, 0.2), type="l", ylim=c(0,1))
```

Two questions that may come to mind when looking at these results: What is the distribution of the generation at which the allele frequency becomes fixed at 0 or 100% (all individuals homozygous with the same genotype)? What is the chance that the frequency is fixed at 100% (versus at 0)?

In order to get approximate answers to these questions, we may wish to create another function, `fixtime`, which follows the population until the allele frequency is fixed. To ensure that the function doesn't run forever, we stop at 2,000 generations, if the frequency still hasn't been fixed.

We can use the following code to run 100 simulations of the population up to the generation at which the allele frequency becomes fixed. This may take a while (depending on the speed of your computer). That is why we included the line with `cat` (below), which allows us to follow the progress of the simulations. In R for Windows, this only works if you turn off "buffered output," which otherwise would suspend any printing to the console until the command has been completely carried out. To turn off buffered output, click (on the menu bar, while the console window is "in focus") "Misc" and then de-select "Buffered output." Alternatively, you can turn buffered output on and off by typing Ctrl-W.

```
res <- matrix(nrow=100,ncol=2)
colnames(res) <- c("allele", "generation")
for(i in 1:100) {
  res[i,] <- fixtime(25, 0.2)
  cat(i, "\n")
}
```

We may now see what proportion of the time the allele frequency was fixed at 100%, and plot a histogram of the generations at which fixation occured.

```
mean(res[,1]==1)
hist(res[,2], breaks=20)
```

$\Longrightarrow$ **What proportion of the time was the allele frequency fixed at 100%?**

$\Longrightarrow$ **What was the average and SD of the generations at which fixation occurred?**

Binomial, Poisson and normal probabilities

One benefit of statistical software is that one no longer needs statistical tables for various distributions, such as the normal, binomial and Poisson.

Binomial random variables

Consider the number of heads in n independent tosses of a biased coin, where the probability of getting a head in each toss is p. In this case, the number of heads (a *random variable*) follows a binomial distribution. For the binomial distribution, you'll want to know about the functions `rbinom`, `dbinom`, `pbinom` and `qbinom`. Type `?rbinom` to view the help file for all of these functions.

The function `rbinom` is used to simulate data from (i.e., independent realizations of) a binomial distribution. The following will simulate 1000 realizations of a binomial random variable with parameters $n = 120$ and $p = 0.3$. The

second line draws a histogram of the results.

```
x <- rbinom(1000, 120, 0.3)
hist(x, breaks=30)
```

The function `dbinom` is used to calculate the probability distribution for a binomial random variable. For example, suppose the chance that a newborn human is male is 0.512. Imagine sampling 6 newborns at random. What is the chance that exactly 3 of them are male? This is calculated by

```
dbinom(3, 6, 0.512)
```

You can get multiple probabilities by giving it a vector. For example, you could get the entire distribution by typing

```
dbinom(0:6, 6, 0.512)
```

You might want to round these off; round them to the hundredths place by typing

```
round(dbinom(0:6, 6, 0.512), 2)
```

The function `pbinom` is used to calculate the *cumulative distribution function* of a binomial random variable—the probability that the random variable is *less than or equal to* a given value. For example, the chance of observing three or fewer males among 6 randomly selected newborns is calculated by

```
pbinom(3, 6, 0.512)
```

The chance of observing fewer than three males is obtained by

```
pbinom(2, 6, 0.512)
```

The chance of observing 40 or fewer males in a sample of 120 newborns is obtained by

```
pbinom(40, 120, 0.512)
```

Note that these calculations could also be performed using `dbinom`, though it would be somewhat less efficient. For example, the last calculation could also be perfomed by typing

```
sum(dbinom(0:40, 120, 0.512))
```

If you wanted to know the chance of observing between 55 and 65 males (*inclusive*) in a random sample of 120 newborns, you could type either of the following.

```
pbinom(65, 120, 0.512) - pbinom(54, 120, 0.512)
sum(dbinom(55:65, 120, 0.512))
```

The function `qbinom` is the inverse of `pbinom`. It returns *quantiles* for the binomial distribution. What is the 75th percentile of the number of males to expect in a random sample of 120 newborns? Type

```
qbinom(0.75, 120, 0.512)
```

Note that there is a issue here related to the granularity (or discreteness) of the binomial distribution. Type

```
pbinom( qbinom(0.75,120,0.512), 120, 0.512).
```

$\Longrightarrow$ **Explain why this last command doesn't return exactly 0.75.**

Screening rats

Suppose I wanted to perform some sort of analysis on rats that are known to be infected with a particular virus. I want ten such rats. If I were to obtain a random sample of rats from Baltimore City, and 10% of Baltimore rats are infected with this virus, how many rats should I sample in order to obtain at least 10 infected ones?

One thought would be to sample 100 rats.

$\Longrightarrow$ **If I sample 100 rats, what is the chance that at least 10 of them are infected?**

How many rats should I obtain in order for there to be a greater than 90% chance that I will obtain at least 10 infected ones? This can be answered by the following code. (The answer is 140. 139 is still too few.) Try to figure out why.

```
x <- qbinom(0.1, 100:200, 0.1)
max( (100:200)[x < 10]) + 1
```

Poisson random variables

The functions `rpois`, `dpois`, `ppois` and `qpois` give the same sort of information for Poisson random variables. Type `?rpois` to view the help file. Instead of the arguments n and p (`size` and `prob` in the four `binom` functions), there's a single argument λ or `lambda`, which is the mean of the Poisson random variable.

$\Longrightarrow$ **Find the probability that a Poisson random variable with mean 0.5 will exceed (i.e., is strictly greater than) 1. Give both the answer and the code.**

Consider tossing a *large* number (say $n = 5000$) of *extremely* biased coins, so that the probability of heads on each toss is very small (say $p = 0.001$). The number of heads will follow a binomial distribution with parameters n and p, but if n is large and p is small, this will be well approximated by a Poisson distribution with parameter $\lambda = np$. Let's plot the distributions binomial($n = 5000$, $p = 0.001$) and Poisson($\lambda = 5000 \times 0.001$).

```
plot(0:10, dbinom(0:10, 5000, 0.001))
lines(0:10, dpois(0:10, 5000*0.001))
```

The points are the probabilities for the binomial distribution. The line goes through the probabilities for the Poisson distribution. It is hard to see any difference at all. Let's look at the ratio, instead.

```
x <- dbinom(0:10, 5000, 0.001)
y <- dpois(0:10, 5000*0.001)
plot(0:10, x/y, ylab="binomial/Poisson")
```

Normal random variables

It should come as no surprise that there are functions `rnorm`, `dnorm`, `pnorm` and `qnorm` for getting these same sorts of things for normally distributed random variables.

Suppose that the heights of adult men in the United States follow an approximately normal distribution with mean 69 inches (that is, 5 feet 9 inches, or 175 cm) and SD 3 inches (8 cm). Let's draw this normal distribution.

```
x <- seq(55, 80, by=0.5)
plot(x, dnorm(x, 69, 3), type="l")
```

$\Longrightarrow$ **If I pick (at random) an adult male from this population, what's the chance that he will be over 67 inches? over 72 inches? Give both the answers and the R code.**

A tiny bit of sampling

Suppose we sample 4 adult males from the U.S. What's the chance that their average height is great than 70 in? What's the chance that all 5 are above 70 in? We could do a simulation to figure this out. First, we do it with `for` loops. Let's do 1000 replicates.

```
me <- 1:1000
mi <- 1:1000
for(i in 1:1000) {
  x <- rnorm(4, 69, 3)
  me[i] <- mean(x)
  mi[i] <- min(x)
}
```

Note that we perform the above task without the `for` loop.

```
x <- matrix( rnorm(4000, 69, 3), ncol=4)
me <- apply(x, 1, mean)
mi <- apply(x, 1, min)
```

Now let's calculate the proportion of times that the mean and the minimum exceeded 70.

```
mean(me > 70)
mean(mi > 70)
```

These last two lines calculate the *proportion* of replicates where the mean height and the minimum height was above 70. The code `me > 70` results in a vector of `TRUE`s and `FALSE`s. The function `mean` treats `TRUE` as 1 and `FALSE` as 0, and the *mean* of these 1's and 0's is the same as the *proportion* of 1's.

Draw histograms of these sample means and minimums. Repeat all of this for sample sizes of 20 and 100. You might try to write a function to do all of this.

Built-in R commands used in this lab

`source`	`function`	`runif`
`table`	`for`	`hist`
`abline`	`mean`	`plot`
`lines`	`colnames`	`rbinom`
`dbinom`	`pbinom`	`qbinom`
`sum`	`rpois`	`dpois`
`ppois`	`qpois`	`rnorm`
`dnorm`	`pnorm`	`qnorm`
`max`	`min`	`apply`

Lab 2 — Statistics for Laboratory Scientists

Due: 22 Feb 2006

Questions to be answered

Population genetics simulation

1. Concerning the distribution of the observed allele frequencies in the initial population, how do they vary for different sample sizes and different generating allele frequencies?
2. Concerning the trace of the observed allele frequencies across generation, how is the case of 100 couples different than the case of 25 couples, in terms of the fluctuation in the observed allele frequency across time/generation?
3. Concerning the investigation of allele fixation:
   (a) What proportion of the time was the allele frequency fixed at 100%?
   (b) What was the average and SD of the generation time at which fixation occurred?

Binomial, Poisson and normal probabilities

4. Explain why the command `pbinom( qbinom(0.75,120,0.512), 120, 0.512)` doesn't return exactly 0.75.
5. If 10% of rats in Baltimore are infected with a particular virus, and I obtain a random sample of 100 rats, what is the chance that at least 10 of them are infected?
6. Use the function `ppois` to find the probability that a Poisson random variable with mean 0.5 will exceed (i.e., is strictly greater than) 1. Give both the answer and the R code.
7. Suppose the heights of adult men in the US follow a normal distribution with mean 69 in and SD 3 in. If I pick a random adult male, what is the chance that he will be over 67 inches? over 72 inches? Give both the answers and the R code.