R for Statistics 571
Contingency Tables through Two Samples

Bret Larget

October 22, 2010

1 Creating Tables

The function `matrix()` will make a table of numbers, here filled with counts in a contingency table, using `rownames()` and `colnames()` to add the names for each category. This code creates a table for the data from Example 9.3-1.

```
> fish = matrix(c(1, 49, 10, 35, 37, 9), 2, 3)
> rownames(fish) = c("Eaten", "Not Eaten")
> colnames(fish) = c("Uninfected", "Lightly Infected", "Highly Infected")
> fish

          Uninfected Lightly Infected Highly Infected
Eaten              1               10              37
Not Eaten         49               35               9
```

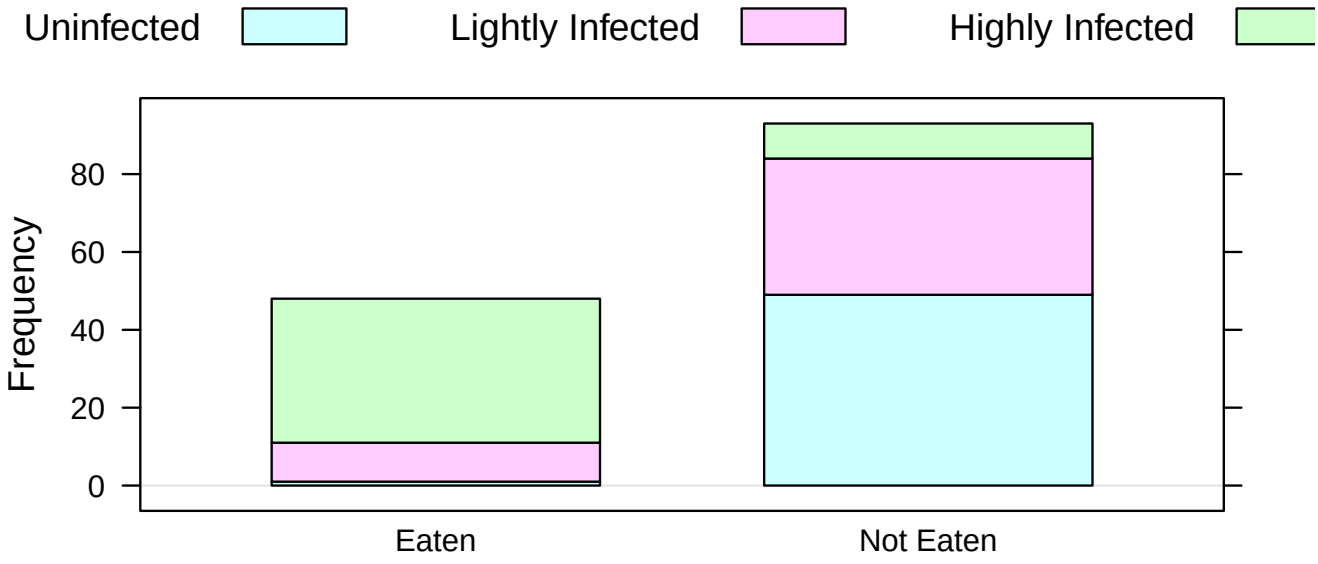
The first argument to `matrix()` is an array of the data in the matrix by columns. (If the data is entered by row, add the argument `byrow=true` to the `matrix()` command.) The second and third arguments are the number of rows and number of columns of data; the command could have been called `matrix(c(1,49,10,35,37,9),nrow=2,ncol=3)`. If only one of these dimension commands is specified, R will compute the other using the size of the data. Like all R functions, if the arguments are not named, R interprets based on their position in the list of arguments. Named arguments can appear anywhere in the list.

2 Bar Graphs

We use `barchart()` from the `lattice` package to make stacked bar graphs to display contingency tables. By default, `barchart()` draws horizontal bars; set `horizontal=F` makes the bars vertical. The function `t()` will transpose a matrix; transposing the input matrix will cause the other variable to be the one used for the main division of the data. The argument `ylab` can be set to change the label on the y-axis (and `xlab` has the same obvious behavior). The argument `auto.key` works for may `lattice` packages to add a key to a graph. Here, we specify the key to be written in columns across the top of the graph. One command creates an object which is then plotted with `plot()`. Compare the use of `fish` and `t(fish)` in these next two plots.

```
> library(lattice)
> fish.plot = barchart(t(fish), horizontal = F, auto.key = list(columns = nrow(fish)),
+     ylab = "Frequency")
> plot(fish.plot)
```


```
> fish.2.plot = barchart(fish, horizontal = F, auto.key = list(columns = ncol(fish)),
+     ylab = "Frequency")
> plot(fish.2.plot)
```



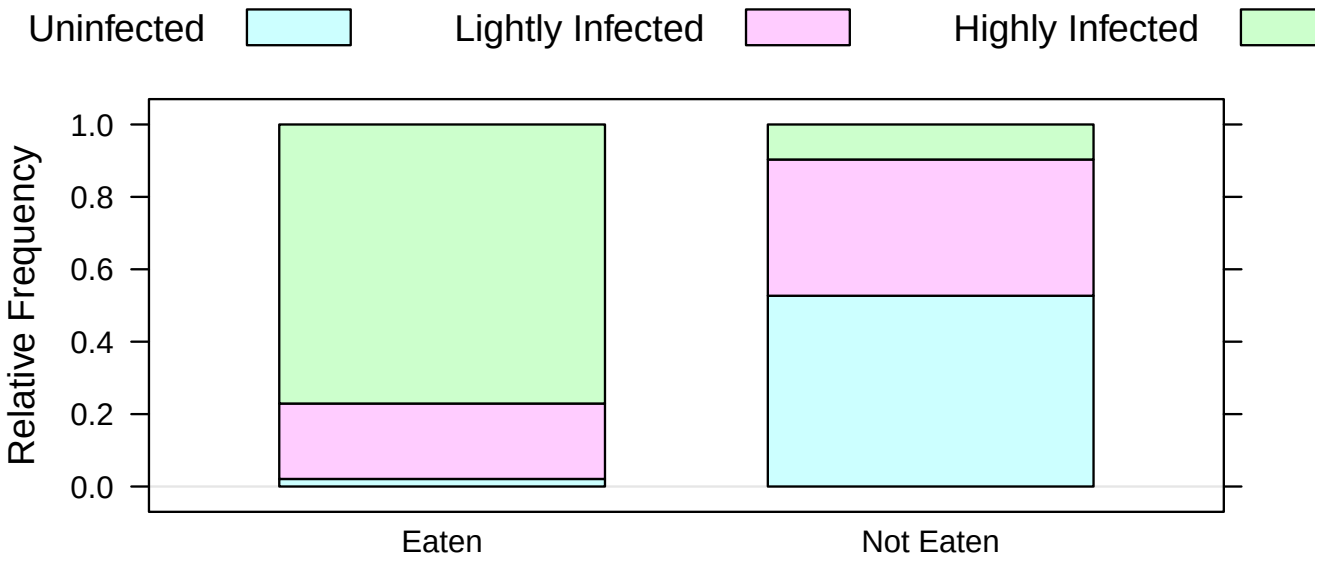
2.1 Mosaic Plots

The text introduces the idea of a mosaic plot which shows nearly the same information as the stacked bar graphs, but each bar is rescaled to the same size so that the graph highlights comparisons between relative frequencies and not absolute frequencies. The function `mosaic()` below handles the conversion from frequencies to relative frequencies, so you can just call `mosaic()` on a matrix (or its transpose).

```
> mosaic = function(x, ...) {
+     col.sums = apply(x, 2, sum)
+     for (j in 1:ncol(x)) x[, j] = x[, j]/col.sums[j]
+     my.plot = barchart(t(x), horizontal = F, ylab = "Relative Frequency",
+         auto.key = list(columns = nrow(x)), ...)
+     plot(my.plot)
+ }
```

Here is an example of its use.

```
> mosaic(t(fish))
```



3 Chisquare Test and G-Test

The function `chisq.test()` will carry out a χ^2 test. The only argument you need to include is the matrix.

```
> chisq.test(fish)

        Pearson's Chi-squared test

data:  fish
X-squared = 69.7557, df = 2, p-value = 7.124e-16
```

For the G-test, there is no built-in function in base R. An internet search will undoubtedly find someone who has made the function available. However, coding a few line of R code to find the expected counts in each table and then the values of the test statistics (for both G- and χ^2 tests) is informative. The following code uses the function `apply()` to apply the `sum()` function to both rows and columns to find the marginal counts and the `%o%` operator to form the *outer sum* of the arrays of row and column counts. In the example with a 2-array for rows and a 3-array for columns, the outer sum will be a 2×3 matrix where each element is the product of the corresponding values for the row and column. Dividing this table by the sum of all elements of the table gives the expected counts. The three arguments for `apply()` are a matrix, a number to indicate the dimension (1 for rows, 2 for columns), and a function to apply to each row or column. The observed counts and expected counts now be combined to find both test statistics. The following function takes a matrix as an argument, computes both the χ^2 and $G - test$ test statistics, and returns a list with the expected counts, the values of the test statistics, and the p-values. In R, a `list()` is an object that can contain any number of potentially different types of items. Components of a list can be accessed with $ and the name of the component. For example, to print only the matrix of expected values, one could append `$expected` to the name of the returned value from this function.

```
> chisq.g.test = function(x) {
+     row.sum = apply(x, 1, sum)
+     col.sum = apply(x, 2, sum)
+     n = sum(x)
+     x.expected = row.sum %o% col.sum/n
+     x2 = sum((x - x.expected)^2/x.expected)
+     g = 2 * sum(x * log(x/x.expected))
+     degf = (length(row.sum) - 1) * (length(col.sum) - 1)
+     p.x2 = 1 - pchisq(x2, degf)
+     p.g = 1 - pchisq(g, degf)
+     return(list(expected = x.expected, x2 = x2, p.x2 = p.x2, g = g,
+         p.g = p.g))
+ }
> fish.out = chisq.g.test(fish)
> fish.out$expected

          Uninfected Lightly Infected Highly Infected
Eaten       17.02128         15.31915        15.65957
Not Eaten   32.97872         29.68085        30.34043

> fish.out

$expected
          Uninfected Lightly Infected Highly Infected
Eaten       17.02128         15.31915        15.65957
Not Eaten   32.97872         29.68085        30.34043

$x2
[1] 69.7557

$p.x2
```

```
[1] 6.661338e-16

$g
[1] 77.89698

$p.g
[1] 0
```

4 Fisher's Exact Test

The R function `fisher.test()` conducts Fisher's exact test ona 2×2 matrix. Note that the alternative hypothesis is in reference to the top left cell of the matrix. In this example, we ask what is the chance if the interior of the table were resampled leavin the margins constant, what is the probability that the upper left corner would be 15 or more. Here is code for an example from lecture.

```
> x = matrix(c(15, 7, 6, 322), nrow = 2, ncol = 2)
> fisher.test(x, alternative = "greater")

        Fisher's Exact Test for Count Data

data:  x
p-value < 2.2e-16
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 35.49817      Inf
sample estimates:
odds ratio
  108.3894
```

The p-value of Fisher's exact test is found by summing hypergeometric probabilities (the probability distribution for sampling without replacement). The arguments to `dhyper()` are: (1) the value or values for which to compute the probability; (2) the number of *good* balls in the bucket; (3) the number of *bad* balls in the bucket; and (4) the sample size (without replacement). Either the row or column marginal totals can be used for the number of balls; the other becomes the sample size. So, in this example, the p-value is the probability of choosing 15 or more balls from either: (1) a bucket with 21 good balls and 329 bad balls with a sample of 22; or (2) a bucket with 22 good balls and 328 bad balls with a sample of 21.

```
> sum(dhyper(15:21, 21, 329, 22))

[1] 1.004713e-16

> sum(dhyper(15:21, 22, 328, 21))

[1] 1.004713e-16
```

5 Normal Distribution

The functions `pnorm()` and `qnorm()` find probabilities (areas to the left) or quantiles from standard or other normal distributions. The first argument is the quantity of interest. The second and third (optional) specify the mean and standard deviation if these are not 0 or 1. Here are several examples:

1. The area to the left of -1.57 under a standard normal curve;

```
> pnorm(-1.57)

[1] 0.05820756
```

2. The area to the right of 2.12 under a standard normal curve;

```
> 1 - pnorm(2.12)

[1] 0.01700302
```

3. The 0.975 quantile of the standard normal distribution;

```
> qnorm(0.975)

[1] 1.959964
```

4. The cutoffs for the middle 99% of the standard normal distribution;

```
> qnorm(c(0.005, 0.995))

[1] -2.575829  2.575829
```

5. The area between 90 and 105 for the $\mathrm{N}(100, 4^2)$ distribution;

```
> pnorm(105, 100, 4) - pnorm(90, 100, 4)

[1] 0.8881406
```

6. The lower and upper quartiles of the same distribution;

```
> qnorm(c(0.25, 0.75), 100, 4)

[1]  97.30204 102.69796
```

6 Graphs for Quantitative Data

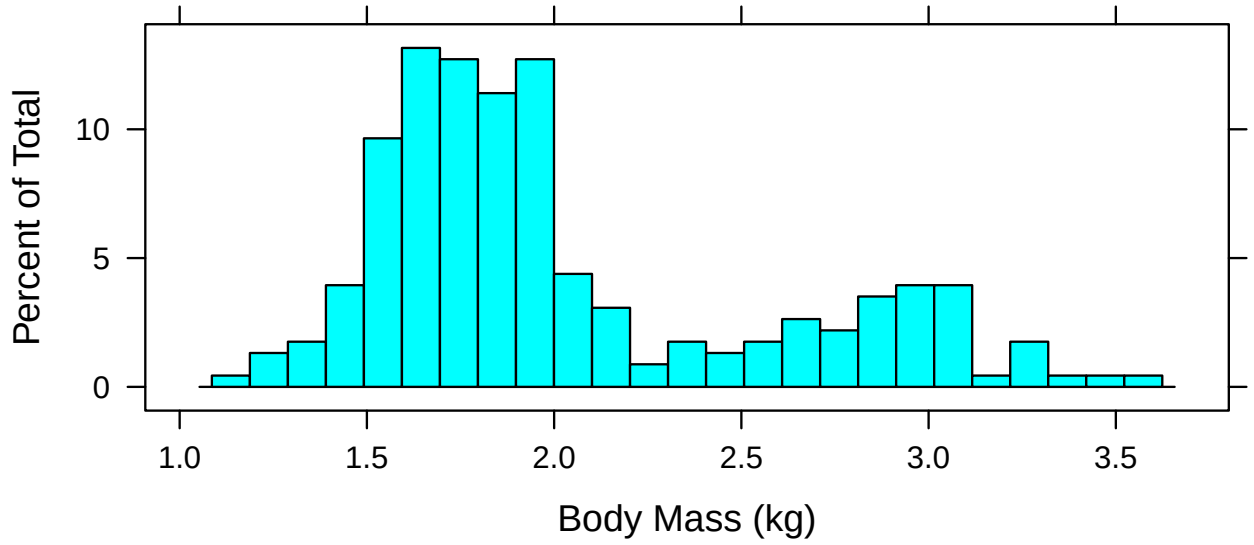
The following examples show examples of histograms, density plots, box-and-whisker plots, and dot plots for the female sockeye salmon mass data set. First, read in the data. There is a single variable named mass.

```
> salmon = read.table("sockeye.txt", header = T)
> str(salmon)

'data.frame':        228 obs. of  1 variable:
 $ mass: num  3.09 2.91 3.06 2.69 2.88 2.98 1.61 2.16 1.56 1.76 ...
```

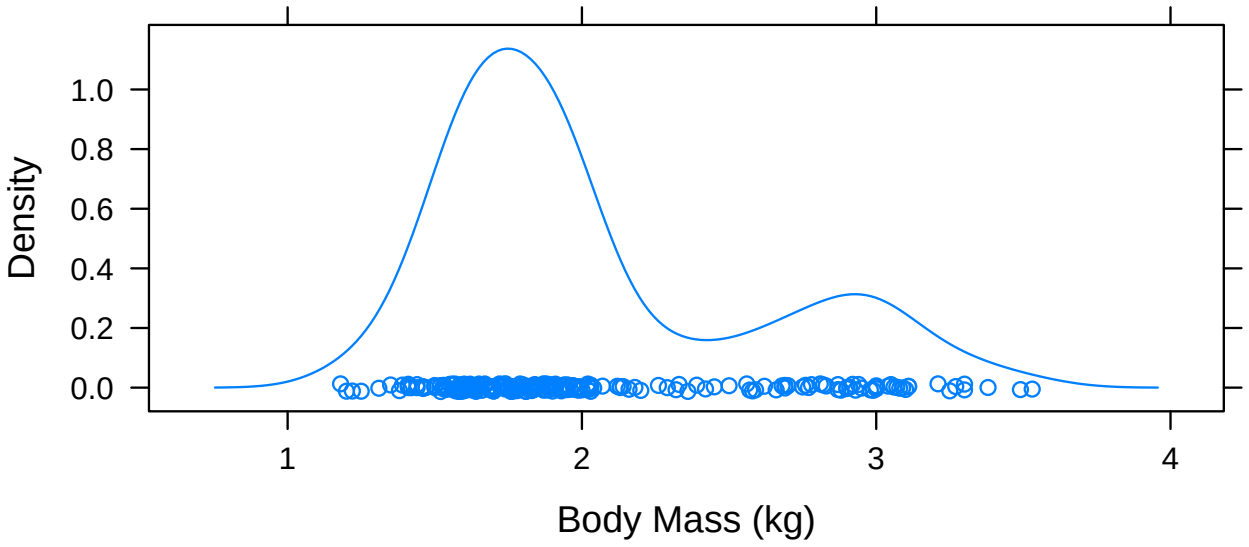
Histogram. The first argument is a formula specifying the variable to graph. The second argument is the name of the data frame where the variable can be found. Note the arguments `nint` to suggest using 25 intervals and `xlab` to set the label on the x-axis. The function `plot()` is need for this to execute when read in from a file, but is optional when typing at the command line.

```
> library(lattice)
> plot(histogram(~mass, salmon, nint = 25, xlab = "Body Mass (kg)"))
```



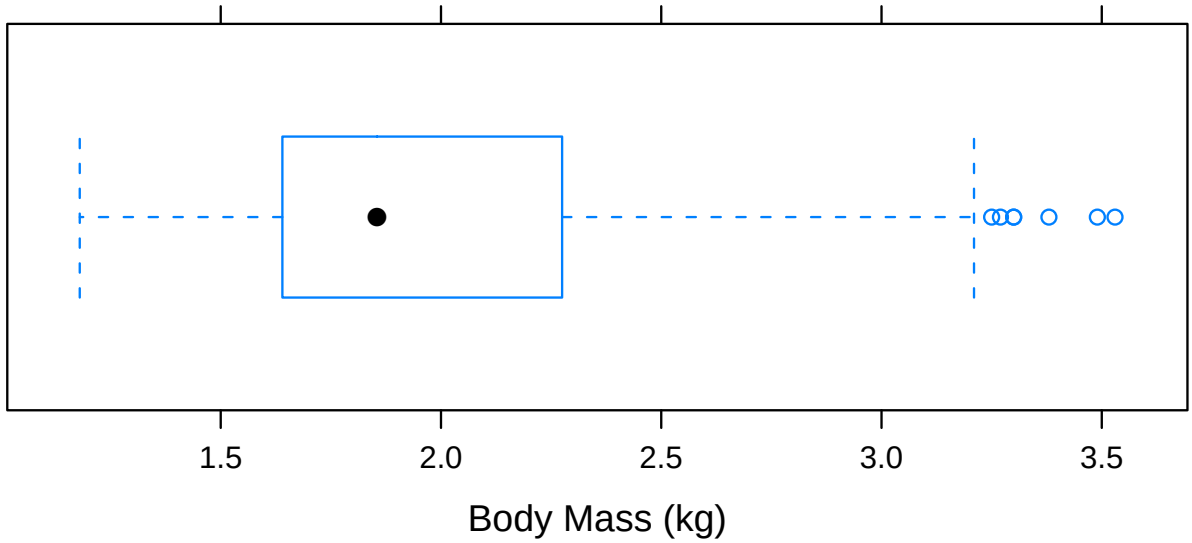
Density plot. The density plot draws a curve which is estimated by averaging many (50 by default) histograms, with the breakpoints shifted slightly for each. I usually bump this number up using the argument `n=201` so that the resulting curve appears smoother. To suppress the plotting of points (useful when the sample size is enormous), add the argument `plot.points=F`.

```
> plot(densityplot(~mass, data = salmon, xlab = "Body Mass (kg)",
+     ylab = "Density", n = 201))
```



Box-and-Whisker Plots. A box and whisker plot represents the middle half of the data, between the 0.25 and 0.75 quantiles, with a box. The center of the box includes a point at the median. Whiskers are drawn left and right to extreme points; here, the largest and smallest individual values *between the fences*, where there are invisible fences to the left and right of the box a distance 1.5 IQR (interquartile range units) from the ends of the box. In other words, if the lower and upper quartiles are $q_{0.25}$ and $q_{0.75}$, the lower fence is at $q_{0.25} - 1.5(q_{0.75} - q_{0.25})$, the upper fence is at $q_{0.75} + 1.5(q_{0.75} - q_{0.25})$, the left whisker extends to the smallest value to the right of the lower fence, the right whisker extends to the largest value to the left of the upper fence, and any observations beyond the fences are marked individually.

```
> plot(bwplot(~mass, data = salmon, xlab = "Body Mass (kg)"))
```



Dot Plots. A dot plot is simply a plot of dots on a number line for each value. To reduce plotting points on top of one another, the argument `jitter.y` adds a small bit of random noise to each point in the y direction. `jitter.x` is also available.

```
> plot(dotplot(~mass, data = salmon, xlab = "Body Mass (kg)", jitter.y = T))
```

7 T-tests and Confidence Intervals

We demonstrate using `t.test()` for one-sample confidence intervals and hypothesis tests using a sample of 25 body temperatures.

```
> temp = read.table("temperature.txt", header = T)
> str(temp)

'data.frame':        25 obs. of  1 variable:
 $ temperature: num  98.4 98.6 97.8 98.8 97.9 99 98.2 98.8 98.8 99 ...
```

The variable can be specified from the data frame with the $ operator. The mean of the null hypothesis is set with `mu=98.6` and the confidence level is set with `conf.level=0.99` (this can be shortened to `conf`.

```
> t.test(temp$temperature, mu = 98.6, conf = 0.99)

        One Sample t-test

data:  temp$temperature
t = -0.5606, df = 24, p-value = 0.5802
alternative hypothesis: true mean is not equal to 98.6
99 percent confidence interval:
```

```
 98.14485 98.90315
sample estimates:
mean of x
   98.524
```

8 The Bootstrap

We used the bootstrap to find a 95% confidence interval for the salmon data in lecture. Here is the R code for this. The basic idea is to create a large matrix with B rows and n columns where n is the sample size of the original data and B is the number of bootstrap data sets we wish to replicate. We use `matrix()` to create the matrix and `sample()` with `replace=T` to sample data with replacement. The function `apply()` with second argument `1` (the number one) and third argument `mean` applies the function `mean()` to each row of the matrix. Finally, we use `quantile()` to find the corresponding quantiles of the sample. Here is an application of the bootstrap using $B = 10,000$.

```
> B = 10000
> n = length(salmon$mass)
> mass.boot = apply(matrix(sample(salmon$mass, size = n * B, replace = T),
+     nrow = B, ncol = n), 1, mean)
> quantile(mass.boot, c(0.025, 0.975))

    2.5%    97.5%
1.959121 2.098378
```

A different bootstrap sample may differ a bit, but not a lot if B is large.

```
> mass.boot.2 = apply(matrix(sample(salmon$mass, size = n * B, replace = T),
+     nrow = B, ncol = n), 1, mean)
> quantile(mass.boot.2, c(0.025, 0.975))

    2.5%    97.5%
1.959602 2.098689
```

Compare to the t-test.

```
> t.test(salmon$mass)

        One Sample t-test

data:  salmon$mass
t = 56.8042, df = 227, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 1.957804 2.098512
sample estimates:
mean of x
 2.028158
```

The 95% confidence intervals are identical when rounded to two decimal places.

9 Randomization Test

In lecture on October 21, we explored the randomization test to test if two population means were equal. Here is R code to read in the data set using `read.csv()` which expects a text file where the first line is a header line with variable names and subsequent lines have data. Values on each line are separated by commas.

```
> pscorp = read.csv("pseudoscorpions.csv")
> str(pscorp)

'data.frame':        36 obs. of  2 variables:
 $ mating: Factor w/ 2 levels "Different","Same": 2 2 2 2 2 2 2 2 2 2 ...
 $ broods: int  4 0 3 1 2 3 4 2 4 2 ...
```

To carry out the randomization test, we will write a special function that will compute the mean of the first 20 observations, the mean of the next 16 observations, and return the difference.

```
> f = function(x) {
+     return(mean(x[1:20]) - mean(x[21:36]))
+ }
```

Next, we create an array to store the difference in means for each randomized data set. We will do this $R = 100,000$ times. We will the array with missing values (NA) which we will replace.

```
> R = 100000
> out = rep(NA, R)
```

The function `sample()` with only one argument consisting of an array returns a random permutation of the elements of the array. We think of this as the first 20 elements being the one randomly sampled group and the next 16 as the second group. This command rerandomizes one time and calls `f()` to find the difference in means.

```
> f(sample(pscorp$broods))

[1] 0.9375
```

Now, we ask R to do this $R = 100,000$ times with the `for()` command. The variable `i` is set to each value from 1 to R, and the difference in means for that particular rerandomization is stored in `out[i]`. The test statistic is found by applying `f()` to the original data (which works because the data is ordered with the *Same* group having the first 20 observations and the *Different* group having the final 16 observations). The p-value is the proportion of randomized differences in sample means that are less than this test statistic.

```
> test.stat = f(pscorp$broods)
> for (i in 1:R) {
+     out[i] = f(sample(pscorp$broods))
+ }
> p.value = sum((out <= test.stat))/R
> p.value

[1] 0.01376
```

10 Two-sample T-test

Compare the previous result to the two-sample independent t-test.

```
> sample1 = with(pscorp, broods[mating == "Same"])
> sample2 = with(pscorp, broods[mating == "Different"])
> t.test(sample1, sample2, alternative = "less")

        Welch Two Sample t-test

data:  sample1 and sample2
t = -2.3424, df = 28.883, p-value = 0.01313
alternative hypothesis: true difference in means is less than 0
95 percent confidence interval:
       -Inf -0.3911856
sample estimates:
mean of x mean of y
    2.200     3.625
```

11 Graphs to Compare Distributions

The following graph compares the distribution of the number of broods for each group with histograms. The vertical bar in the formula specifies the variable on the left to be split according to the variable on the right and displayed in different panels. The `layout` argument specifies one column and two rows (so the histograms are aligned vertically, making them easier to compare).

```
> plot(histogram(~broods | mating, data = pscorp, layout = c(1, 2),
+     breaks = seq(-0.5, 7.5)))
```