

Spring 2008 - Stat C141/ Bioeng C141 - Statistics for Bioinformatics

Course Website: <http://www.stat.berkeley.edu/users/hhuang/141C-2008.html>

Section Website: <http://www.stat.berkeley.edu/users/mgoldman>

GSI Contact Info:

Megan Goldman

mgoldman@stat.berkeley.edu

Office Hours: 342 Evans M 10-11, Th 3-4, and by appointment

1 Markov Chains - Stationary Distributions

The stationary distribution of a Markov Chain with transition matrix P is some vector, ψ , such that $\psi P = \psi$. In other words, over the long run, no matter what the starting state was, the proportion of time the chain spends in state j is approximately ψ_j for all j .

Let's try to find the stationary distribution of a Markov Chain with the following transition matrix:

$$P = \begin{bmatrix} 0.6 & 0.1 & 0.3 \\ 0.1 & 0.7 & 0.2 \\ 0.2 & 0.2 & 0.6 \end{bmatrix}$$

There are two ways to go about this. First, since we know that $\psi P = \psi$, we can write this as a series of equations:

$$\begin{aligned} .6\psi_1 + .1\psi_2 + .2\psi_3 &= \psi_1 \\ .1\psi_1 + .7\psi_2 + .2\psi_3 &= \psi_2 \\ \psi_1 + \psi_2 + \psi_3 &= 1 \end{aligned}$$

We can use R to solve this series of equations:

```
solve(matrix(c(-.4, .1, .2, .1, -.3, .2, 1, 1, 1),nrow=3, byrow = T),c(0,0,1))
```

and we get that $\psi \approx (0.2759, .3448, .3793)$. Last week, we used `solve` to invert a matrix. Here, we use it to solve a series of equations. The first argument is the left hand side of the equations, in matrix form. The second argument is the right hand side. Since we don't want any ψ values on the right hand side, I did a bit of algebra to get all the ψ bits on the left hand side.

It's also possible to arrive at the stationary distribution by just multiplying the transition matrix by itself over and over again until it finally converges:

```
x <- matrix(c(.6, .1, .3, .1, .7, .2, .2, .2, .6), nrow = 3, byrow = T)
y <- x
```

```
for(i in 1:25)
y <- y %*% x
```

You'll note that you have a 3 by 3 matrix with all rows equal. This tells you that the n-step transition matrix has converged.

2 Hidden Markov Models - Muscling one out by hand

Consider a Markov chain with 2 states, A and B. The initial distribution is $\pi = (.5 \ .5)$. The transition matrix is

$$P = \begin{bmatrix} .9 & .1 \\ .8 & .2 \end{bmatrix}$$

The alphabet has only the numbers 1 and 2. The emission probabilities are

$$\begin{aligned} e_A(1) &= .5 & e_A(2) &= .5 \\ e_B(1) &= .25 & e_B(2) &= .75 \end{aligned}$$

Now suppose we observe the sequence $\mathcal{O} = 2, 1, 2$. What sequence of states gives the highest probability to this observed sequence? Let's consider all possibilities:

Sequence	Probability
AAA	$.5 \times .5 \times .9 \times .5 \times .9 \times .5 = 0.050625$
AAB	$.5 \times .5 \times .9 \times .5 \times .1 \times .75 = 0.0084375$
ABA	$.5 \times .5 \times .1 \times .25 \times .8 \times .5 = 0.0025$
ABB	$.5 \times .5 \times .1 \times .25 \times .2 \times .75 = 0.0009375$
BAA	$.5 \times .75 \times .8 \times .5 \times .9 \times .5 = 0.0675$
BAB	$.5 \times .75 \times .8 \times .5 \times .1 \times .75 = 0.01125$
BBA	$.5 \times .75 \times .2 \times .25 \times .8 \times .5 = 0.0075$
BBB	$.5 \times .75 \times .2 \times .25 \times .2 \times .75 = 0.0028125$

From the calculations above, we can see that the hidden state sequence BAA has the highest probability of giving an observed sequence of 2,1,2.

3 Writing functions in R

Say you want a function that you can't find in R, or that you'd spend more time hunting down than it would take to just write it yourself. Here's a function, "borrowed" from the official R Tutorial, that does a two-sample t test statistic:

```
twosam <- function(y1, y2) {
  n1 <- length(y1); n2 <- length(y2)
  yb1 <- mean(y1); yb2 <- mean(y2)
  s1 <- var(y1); s2 <- var(y2)
  s <- ((n1-1)*s1 + (n2-1)*s2)/(n1+n2-2)
```

```

    tst <- (yb1 - yb2)/sqrt(s*(1/n1 + 1/n2))
    tst
}

```

Let's parse what they've done:

First, they've declared "twosam" to be a function with two arguments, which they've called y1 and y2. Then, inside the curly brackets, they did a number of operations on those two arguments. n1 and n2 are calculating the length of each argument, i.e., the number of items in each vector. Then they calculate means and variances, and a pooled variance, s. Penultimately, they declare a value, tst, which contains the test statistic. Lastly, there's just a line `tst`, which will display the value of the test statistic just calculated.

Many functions can be written in just the same way: give your function a name, and say what arguments it needs. Then, inside curly brackets, write all the manipulations that should be done with those arguments. There are many more examples of functions, including some that are much more complex, in the R Tutorial, available online at <http://cran.r-project.org/doc/manuals/R-intro.html>

Now let's use our function to calculate a two sample t statistic. First, let's build two groups of observations:

```

group1 <- rnorm(100)
group2 <- rnorm(100, mean = .1)

```

Both groups have 100 values. The first are standard normal, the second group has mean of .1 and sd 1. Recall that the defaults for normal are mean 0 and sd 1, so if those are the values you want, you don't have to declare them. Now try plugging them in to our function:

```

twosam(group1, group2)

```

Now, let's try the built in t-test function to see if we get the same test statistic:

```

t.test(group1, group2)

```

There's a whole lot of output here, but the key part is the bit that starts "t = " This number should be within rounding error of what we got using our homemade function. This test is asking if there is a difference in means between the two groups, Here, since we made the two groups ourselves, we know there's a difference in means of .1. However, 100 observations from each group just weren't enough to see the difference from a statistical standpoint, as examination of the t-test output shows a non-significant p-value.