Notes for Stat 430 – Design of Experiments – Lab #3 Sept. 24, 2012.

Agenda:
Vector operations
Building a matrix
Matrix operations
Distributions - Density
Distributions – Cumulative
Distributions - Random Generation
Central Limit Theorem

Useful files: http://www.sfu.ca/~jackd/Stat430/Wk03_Code.txt
All lab material found at : http://www.sfu.ca/~jackd/Stat430/
A short list of the most useful commands. http://www.personality-project.org/r/r.commands.html

Vector Operations

Simple math operations are performed on every element in a vector
Starting with the numbers 1,2,3,4,5.

```
1:5
```

We can double it

```
1:5 * 2
```

Or add 3

```
1:5 + 3
```

Vectors can also be added element by element

```
c(5,10,10) + c(1,2,3)
```

```
1:5 + 5:1
```

Matrix Construction

The general form is

```
matrix(  values, nrow =    , ncol = 2)
```

values is the values of the matrix
nrow and ncol are the number of rows and columns

```
mat.1 = matrix( 1:4, nrow = 2, ncol = 2)
mat.1
```

If you don't put enough elements in the values part then R will repeat whatever is there until the matrix is full.

```
matrix(3, nrow = 2, ncol = 2)
```

You can also take the transpose of a matrix with t(name of matrix)

```
t(mat.1)
```

Or get the determinant

```
det(mat.1)
```

Matrix operations

By default matrix operations are done element by element

```
A = matrix( 1:4, nrow = 2, ncol = 2)
B = matrix( 5:8, nrow=2, ncol=2)
```

This is correct for addition,

```
A + B
```

But it isn't how matrix multiplication is done.

```
A * B
```

To do a proper matrix multiplication, you need to use `%*%`

```
A %*% B
```

We can test which one is the correct way to multiply matrices with the determinant.
Recall that det(AB) = det(A)det(B)

```
det(A)
det(B)
det(A)*det(B) ## What we should be getting

det(A * B) ## No
det(A %*% B) ## Yes
```

Distributions - Density

We can get their density of most of the classic distributions with functions like `dnorm` for the normal, `dpois` for the Poisson, and `dgeom` for the Geometric.

We can get the density of a standard normal (also called Gaussian) at 0,

```
dnorm(0)
```

and compare it to the density from the formula.

```
1 / sqrt(2*pi) * exp( - 0/2) #exp stands for "natural exponent"
```

We can modify it to suit a particular mean and standard deviation

```
dnorm(3, mean=5, sd=10)
```

and compare it to the formula again.

```
1 / sqrt(2*pi*10^2) * exp( -(3-5)^2 / (2*10^2))
```

Other distributions have different things to set. Use the ? and ?? command if you're unsure.
This is the density of a binomial, 3 successes out of 5, with chance of success 0.5.

```
dbinom(x=3, size=5, p=0.5)
```

Density of a Poisson, at x=4, with lambda=6

```
dpois(x=4,lambda=6)
```

Look up others with the ? command

```
?dbeta          ?dgamma          ?dt          ?dchisq          ?dexp
?dunif          ?dhyper     #That last one is the hypergeometric
```

Or get the list of the ones that come with the basic R package with

```
?distribution
```

Cumulative Distributions

Getting the cumulative distribution is especially useful for finding the p-value of something when that variable is assumed.

For example, if we were doing a t-test, and got a t-score of 2.2, with 17 degrees of freedom, we could get the p-value.

```
pt(2.2, df=17)
```

The function above shows **Pr(t < 2.2).**
If we wanted **Pr(t > 2.2)**, we could use the compliment property

```
1 - pt(2.2, df=17)
```

Since this is only the mass above a certain point, we can multiply it by 2 to do a two-tailed test

```
2*(1 - pt(2.2, df=17))
```

Some other tests you'll be likely to do are the Chi-squared test and ANOVA F-test

```
1 - pchisq(9, df=5)
1 - pf(4, df1=2, df2=15)
```

For the F-distribution, df1 is the numerator degrees of freedom, and df2 is the denominator degrees of freedom.

Random number generation

Sometimes, especially when doing simulations, we'll want to randomly generate values from a distribution. For that, it's r<dist>

```
rexp(mean=4)
rexp(mean=4)
```

If generating 1 value is good, generating 10 values is, like, twice as good.

```
rpois(n=10, lambda=4)
```

If you generate a huge number of values, it will print them ALL to the screen unless you assign a name

```
x = runif(n=1000000, min=2, max=4)

x[10:20]
hist(x)
mean(x)
var(x)

(4 + 2) / 2
(4 - 2)^2 / 12
```

Central Limit Theorem

We can use random generation to see what a distribution looks like

```
x = rnorm(n=100000)
hist(x)
```

There are much more efficient ways to draw a distribution. The curve function lets you draw the values of a function for a range of values of x.

```
curve( dnorm(x), from=-3, to=3)
```

Here it is again, but with less variance.

```
curve( dnorm(x, sd=0.5), from=-3, to=3)
```

With any graph, you can put it on top of another graph (instead of making a new picture with `add=TRUE`)

```
curve( dnorm(x), from=-3, to=3)
curve( dt(x,df=3), from=-3, to=3, add=TRUE)
curve( dt(x,df=10), from=-3, to=3, add=TRUE)
curve( dt(x,df=25), from=-3, to=3, add=TRUE)
```

Now that we've seen what a normal distribution looks like, we can try out the central limit theorem by simulation.

Central Limit Theorem: As n --> infinity, the distribution of the sum (and the mean) of any distribution tends towards a normal. (If the variance is finite)

We can do this by

1. Generating many sets of data from a distribution.
2. Getting the mean of each set.
3. Plotting a histogram of the means.
4. Comparing it to the normal distribution.

First, set up a vector for the means (rep means repeat, so this is 0 repeated 40000 times)

```
means1 = rep(0,40000)
means3 = rep(0,40000)
means5 = rep(0,40000)
```

Then generate and get the means

```
for(k in 1:40000)
{
      means1[k] = mean(runif(n=1)) ## From a uniform, get 1 value, then get the mean
      means3[k] = mean(runif(n=3))   ##  " " 3 values
      means5[k] = mean(runif(n=5)) ## " " 5 values
}
```

Build a histogram of the values

I've included probability = TRUE to scale to density and not frequency

```
hist(means1, probability=TRUE, ylim=c(0,1.5), n=30)
```

Draw the curve of the normal distribution

mean = 0.5, and sd=sqrt(1/12) because uniform(0,1) variance = 1/12

```
curve(dnorm(x,mean=0.5, sd=sqrt(1/12)),add=TRUE)
```

The same thing for the mean of 3 values

```
hist(means3, probability=TRUE, n=30)
curve(dnorm(x,mean=0.5, sd=sqrt(1/(12*3))),add=TRUE)
```

..and of 5 values.

```
hist(means5, probability=TRUE, n=30)
curve(dnorm(x,mean=0.5, sd=sqrt(1/(12*5))),add=TRUE)
```