Introduction to R

R is not a typical statistical analysis system. It is a flexible computer language that facilitates statistical computations. The distinct advantage of R is that the user is free to create new data-analysis schemes out of the built in components to R. Furthermore, R programs can be developed in an interactive environment. Another valuable aspect of R is its ability to create excellent graphical displays.

The R command line prompt is:

```
>
```

To exit R:

```
> q()
```

To get help in R:

```
> help(q)
```

The comment symbol is:

```
> # comment
```

Values can be stored in R as single values, in vectors, and in matrices. For example,

```
> x = 5
```

gives the value 5 to x, or

```
> x = c(1,5,7,8,9,10)
```

gives x a vector of 6 values, or

```
> x = matrix(x, 2, 3)
```

makes a matrix. Elements of a matrix can be accessed by

```
> x[1,2]
```

A nice way to save commands you might want to save for later execution is in a script file (*.r). You can type your commands into a script file and the cut-and-paste them into the command window or run them. Type the following commands into a script file and run them.

```
z = c(5,4,8,12,10,20,25,27,14,80)
z
z = z + 5
z
z.new = z[z>10]
```

```
z.new
n = length(z)
n
p = length(z.new)/length(z)
p
```

Some useful statistics commands are:

```
z.mean = mean(z)
z.mean
z.var = var(z)
z.var
z.sd = sqrt(z.var)
z.sd
z.median = median(z)
z.median
z.q.50 = quantile(z,0.50)
z.q.50
```

R functions for many common statistical distributions are available. For each distribution there is a `p` function that calculates c.d.f. values, a `q` function that calculates quantiles, a `d` function that calculates heights of the density curve, and a `r` function that calculates random values from the given distribution.

```
#########################################################################################
# Plotting density functions.

# Plot the density function of the beta distribution for different values of
# beta1 and beta2.

x = (1:99)/100

beta1 = 1
beta2 = 1
p.beta = dbeta(x,beta1,beta2)
plot(x,p.beta,main="beta densities",type="l")

par(new=T)

beta1 = 5
beta2 = 1
p.beta = dbeta(x,beta1,beta2)
plot(x,p.beta,main="beta densities",pch="+")

par(new=T)

beta1 = 1
beta2 = 5
p.beta = dbeta(x,beta1,beta2)
plot(x,p.beta,main="beta densities",pch="*")

par(new=T)
```

```
beta1 = 10
beta2 = 10
p.beta = dbeta(x,beta1,beta2)
plot(x,p.beta,main="beta densities",pch="-")

# Take a random sample from a beta.

beta1 = 5
beta2 = 1

n = 1000

y = rbeta(n,beta1,beta2)

hist(y,main="histogram of a random sample from a beta distribution",xlim=c(0,1))

###########################################################################################
```

All commands, variables, functions, and all responses to R commands are recorded by R. To see what has been created

```
> ls()
```

To remove an object such as z, type

```
> rm(z)
```

Simulation is a way to approximate real-world processes. The key to simulation techniques is a sequence of random numbers. Some examples of simulation follow.

```
###########################################################################################
### Simulate flipping a fair coin.

# There are two possible outcomes.  Let x be the number of heads in n flips of a fair coin.

n = 500
x = sample(0:1,n,replace=T)
p.hat = cumsum(x)
index = 1:n
p.hat = p.hat/index

plot(p.hat,main="relative frequency of heads",type="l",ylim=c(0,1))

p.obs = p.hat[n]
p.exp = 1/2

cbind(p.obs,p.exp)

### Simulate rolling a fair die.

# There are six possible outcomes.  Let x be the number of pips you see on each roll of a
# fair die.
```

```
n = 10000
x = sample(1:6,n,replace=T)

br = (1:7)-0.5

par(mfcol=c(4,1))
hist(x[1:100],nclass=6,breaks=br,probability=T,main="relative frequency plot - roll one die")
hist(x[1:500],nclass=6,breaks=br,probability=T,main="relative frequency plot - roll one die")
hist(x[1:1000],nclass=6,breaks=br,probability=T,main="relative frequency plot - roll one die")
hist(x[1:10000],nclass=6,breaks=br,probability=T,main="relative frequency plot - roll one die")

p.obs = table(x)/n
p.exp = c(1,1,1,1,1,1)/6

cbind(p.obs,p.exp)

### Simulate rolling a pair of fair dice.

# There are 12 possible outcomes.  Let x be the sum of pips you see on each roll of the
# pair of fair dice.

n = 10000
x = matrix(sample(1:6,2*n,replace=T),2,n)
s = apply(x,2,sum)

br = (1:13)-0.5

par(mfcol=c(4,1))
hist(s[1:100],nclass=12,breaks=br,probability=T,main="relative frequency plot - roll a pair of dice")
hist(s[1:500],nclass=12,breaks=br,probability=T,main="relative frequency plot - roll a pair of dice")
hist(s[1:1000],nclass=12,breaks=br,probability=T,main="relative frequency plot - roll a pair of dice")
hist(s[1:10000],nclass=12,breaks=br,probability=T,main="relative frequency plot - roll a pair of dice")

p.obs = table(s)/n
p.exp = c(1,2,3,4,5,6,5,4,3,2,1)/36

cbind(p.obs,p.exp)

##########################################################################################
```

Just to exhibit a variance stabilizing transformation, consider a sample from the Poisson distribution.

```
###########################################################################################
### Variance Stabilizing Transformation

# Suppose X is a Poisson random variable.
# Consider a sample of n Poisson random variables.

n = 10000

lambda = 0.4  #lambda = 1 #lambda = 5 #lambda = 10

x = rpois(n,lambda)

x.max = max(x)

br = (0:(x.max+1))-0.5

hist(x,nclass=(x.max+1),breaks=br,probability=T,main="relative frequency plot")

x.samplemean = mean(x)
x.samplevar = var(x)

x.ev = lambda
x.var = lambda

cbind(x.samplemean,x.ev,x.samplevar,x.var)

y = sqrt(x)

y.samplemean = mean(y)
y.samplevar = var(y)

cbind(y.samplemean,y.samplevar)

###########################################################################################
```

To see the CLT in action consider the following examples:

```
###########################################################################################
### Central Limit Theorem

# Take samples from a normal population.  Plot a histogram of the sampling distribution
# of x.bar.

m = 50
sd = 10

par(mfrow=c(2,2))

x = (0:100)

p.norm = dnorm(x,m,sd)
```

```
plot(x,p.norm,main="normal density",type="l",xlim=c(0,100))

x = matrix(rnorm(3000,m,sd),nrow=30)
x.bar = apply(x,2,mean)

hist(x.bar,main="sampling distribution of x.bar",xlim=c(40,60))

plot(density(x.bar,width=5),main="sampling distribution of x.bar",type="l",xlim=c(0,100))

# Take samples from a beta population.  Plot a histogram of the sampling distribution
# of x.bar.

beta1 = 5
beta2 = 1

par(mfrow=c(2,2))

x = (1:99)/100

p.beta = dbeta(x,beta1,beta2)
plot(x,p.beta,main="beta density",type="l")

x = matrix(rbeta(3000,50,10),nrow=30)
x.bar = apply(x,2,mean)

hist(x.bar,main="sampling distribution of x.bar",xlim=c(0.6,1))

plot(density(x.bar,width=0.2),main="sampling distribution of x.bar",type="l",xlim=c(0,1))

##########################################################################################
```