The for loop

If you have a statement or statements that you want to repeat a number of times, you can use the "for" statement to do so.

The "for" statement loops over all elements in a list or vector:

```
for (variable in sequence) statement.1
```

```
x = numeric(100)

for( i in 1:100)
{
x[i] = i + exp(i)
}
```

Note the colon (:) operator generates a sequence of integers (from before).

You can use this in places other than the for loop or array indexing, too. Also note that statement 1 can be a group of statements inside braces.

The if statement

This is used when you want to do something if a condition is true and something else otherwise. The statement looks like this:

```
if ( condition ) statement 1 else statement 2
```

This will execute statement 1 if the condition is true and statement 2 if the condition is false.

```
if (x < 3) print("x less than 3") else print ("x not less than
3")
```

If you want statement 1 and / or statement 2 to consist of more than one statement, then the if construct looks like this:

The group of statements between a { and a } are treated as one statement by the if and else.

```
if (condition) {
statement 1a
statement 1b
…
} else {
statement 2a
```

```
statement 2b
…
}
```

Other flow control statements

R has two statements break and next: these discontinue normal execution in the middle of "for" loop. The "next" statement causes the next pass through the loop to begin at once, while "break" causes a jump to the statement immediately after the end of the loop.

Dice Roll simulation

Recall how to construct a vector of consecutive integers. Lets a construct a vector "die" with a sequence of values from 1 to 6.

```
>die=1:6
```

Lets sample N values from this vector (using sampling with replacement). (The R function is **`sample()`**.)

This is equivalent to rolling a fair dice N times. This time let N =100.
Also, let us use the table function to analyse the outcome.

```
## Initialize variables
die = 1:6
N=100

## Calculations
x=sample(die,N, replace=TRUE)

table(x)
```

We should get approximately equal numbers for each outcome.

Write down the mean, standard deviation and variance of your vector.

```
mean(x)
```

```
sd(x)

var(x)
```

Using control loops, we can repeat this experiment "M" times. Let us specifically study the mean of the vector, for each of the M iterations.

We will save these values in a vector "y".

```
y=numeric()

M=1000;N=100
for(i in 1:M)

{
X =sample(die,N, replace =TRUE)
Xbar=mean(X)

y=c(y,Xbar)
}
```

What is the mean value of y? what is its standard deviation and variance?

How many values of y are less than 3.1? how many are greater than 3.9?

Histograms

A histogram is a commonly used graphical technique consists of parallel vertical bars that graphically shows the frequency distribution of a numeric variable. The area of each bar is equal to the frequency of items found in each class.

The command is simply **`hist()`**.

We will revert to graphical methods in a later class.

```
hist(y)
```

Comment on the shape of this resultant histogram. (Recall the Central Limit Theorem)

Introduction to Inference

Suppose we perform this experiment with an unknown die. The resultant sum of 100 throws is 301, (i.e. the resultant mean is 3.01).

It is an unusual, but not impossible outcome from a fair die.
Consider another possibility: that the die is crooked, favouring low values. If this was the case, a sum of 301 would not be unusual.

Before we began the experiment we had no reason to suspect the die wasn't fair, and we expected a value of around 350. This is our null hypothesis.

H_0: The die is fair.
H_A: The die is crooked.

Understanding Standard Error

Let us repeat the same experiment as before, varying N the number of throws in each experiment.

Write down the standard deviation and variance for y for each of the following cases; N = 50,100,200,500

Recall from your previous statistics modules the concept of "Standard Error". This is the standard deviation of the sampling distribution. A key component in standard error is the sample size (analogous to the N value).

p values

The probability of getting a values as extreme or more as some statistic, such as sum or mean, is known as a *p-value*. When performing statistical calculations using computer software they are the most commonly used item for making statistical decisions.

In this last instance, we would usually fail to reject the null hypothesis. Many *R* outputs will give a group of asterisks beside the data to help the user in interpreting the data, depending on how significant the result is.

p-value < 0.0001 ***
p-value < 0.001 **
p-value < 0.01 *
p-value < 0.1

For this module, as a rule of thumb, we will use the threshold of 0.01 for deciding whether to accept or reject the null hypothesis. If the p-value is less than 0.01 we reject the null hypothesis. If not, we fail to reject the null hypothesis.