An Introduction to R

Preliminaries

- Throughout these slides, red text indicates text that is typed at the R prompt or text that is to be cut from a text file and pasted to the R prompt.
- Blue text indicates R output.
- Black text indicated notes about the code and output.
- To read the help file for any R command type ?command at the R prompt. For example, to read about the mean function

```
> ?mean
```



```
> 7
[1] 7
```

If you give R 7, it will give you back 7. The 1 in square brackets marks the element number in the output. For example, the next command generates 20 standard normal random numbers. The [6] indicates that -0.6693018 is the 6th number of the 20.

```
> rnorm(20,0,1)
 [1]  0.7471442 -0.1930792 -1.4760422 -0.3172517 -0.3997149
 [6] -0.6693018  0.4165647  0.8298968  1.0748766  1.3025841
[11]  0.4675156 -0.5274312  0.2418231  0.9012613  0.6674558
[16] -0.2037051  1.0914826 -1.5400979 -1.3983107 -1.0940496
```

```
> 6+4/2
[1] 8
```

R is a calculator that uses standard order of operations.

```
> (6+4)/2
[1] 5
```

```
> (6+4)/
+ 2
[1] 5
```

If you press the enter key before completing a command, the R prompt becomes a +, which indicates that anything typed after the + will be appended to the line above for evaluation.


```
> x=c(4,-1,5,9,4.5)
> x
[1]  4.0 -1.0  5.0  9.0  4.5
```

This command combines the numbers into a vector and assigns the vector to an object called x. The vector is printed to the screen when x is typed at the R prompt.

```
> x+1
[1]  5.0  0.0  6.0 10.0  5.5
> x
[1]  4.0 -1.0  5.0  9.0  4.5
```

This command creates (but does not store) a new vector whose elements are the elements x with 1 added to each. Note that the command does not change x.

```
> x[2]
[1] -1
```

The number 2 enclosed in brackets immediately after x asks R to produce the 2nd element of the vector x.

```
> x[-2]
[1] 4.0 5.0 9.0 4.5
```

The number -2 enclosed in brackets immediately after x asks R to produce all the elements of the vector x except the 2nd element.

```
> x[c(1,5)]
[1] 4.0 4.5
```

This command requests the 1st and 5th elements of the vector x.


```
> x
[1]  4.0 -1.0  5.0  9.0  4.5
```

The vector x was not changed by the previous commands.

```
> 1:3
[1] 1 2 3
```

This command creates a vector of consecutive integers ranging from the integer before the colon through the integer after the colon.

```
> x[1:3]
[1]  4 -1  5
```

This command requests the 1st, 2nd, and 3rd elements of the vector x.

```
> x>0
[1]  TRUE FALSE  TRUE  TRUE  TRUE
```

This command creates a vector of logicals (TRUE or FALSE) by evaluating whether each element of x is greater than 0 or not.

```
> x[x>0]
[1] 4.0 5.0 9.0 4.5
```

This command requests the positive elements of x.

```
> !(x>0)
[1] FALSE  TRUE FALSE FALSE FALSE
> x<=0
[1] FALSE  TRUE FALSE FALSE FALSE
```

The exclamation point (!) means "not" or "the opposite of".

"<=" means "less than or equal". To check for equality, use "==".


```
> x
[1]  4.0 -1.0  5.0  9.0  4.5
```

The vector x was not changed by the previous commands.

```
> y=sample(1:5)
> y
[1] 2 1 4 3 5
```

The sample function can be used to shuffle a vector into a random order.

```
> y[x<0]=0
> y
[1] 2 0 4 3 5
```

This command finds the elements of y that correspond to elements of x that are less than 0, and then replaces all those elements of y with 0. In this case, only the 2nd element of x is less than 0, so only the 2nd element of y gets replaced with 0.

```
> y[y==0]=99
> y
[1]  2 99  4  3  5
```

This command finds the elements of y that are exactly equal to 0 and then replaces all those elements of y with 99.

```
> y[y<99]=-1
> y
[1] -1 99 -1 -1 -1
```

This command finds the elements of y that are less than 99 and replaces them all with -1.

```
> y[y<0]=1:4
> y
[1]  1 99  2  3  4
```

This command finds the elements of y that are less than 0 and replaces the first such element with 1, the second with 2, the third with 3, and the fourth with 4.

```
> x
[1]  4.0 -1.0  5.0  9.0  4.5
```

The vector x was not changed by the previous commands.

```
> sum(x)
[1] 21.5
```

sum is a built-in R function that returns the sum of all the elements in a vector.

```
> prod(x)
[1] -810
```

prod is a built-in R function that returns the product of all the elements in a vector.

```
> x<=0
[1] FALSE  TRUE FALSE FALSE FALSE
> sum(x<=0)
[1] 1
```

When R is asked to perform mathematical operations on logicals, TRUE gets converted to 1 and FALSE to 0. Thus, this command counts the number of elements of x that are less than or equal to 0.

```
> x*(1:5)
[1]  4.0 -2.0 15.0 36.0 22.5
```

Vectors are multiplied component by component. Thus, the elements of the resulting vector are 4*1, -1*2, 5*3, 9*4, and 4.5*5.

```
> x*(x<=0)
[1]  0 -1  0  0  0
```

Another example of componentwise multiplication. Here, the TRUEs and FALSEs become 1s and 0s, respectively.


```
dan=function(x1,x2)
{
 out=sum(x1>x2)
 out
}
```

This is an example of a function. Functions can be written in a text editor and pasted into R. This function has two arguments called x1 and x2. The function counts how many of the elements in x1 are greater than their corresponding elements in x2. The result is stored in out and provided when the function is called at the R prompt.

```
> dan(1:4,c(2,1,8,7))
[1] 1
```

In this case, x1 gets assigned 1,2,3,4 and x2 gets assigned 2,1,8,7 when the function is called.

```
dan=function(x1=1:4,x2)
{
 out=sum(x1>x2)
 out
}
```

You can specify a default value for any argument. In this case, x1 is given the default value of 1,2,3,4. If x1 is not assigned any value when the function is called, the default value will be used.

```
> dan(x2=c(2,1,8,7))
[1] 1
```

R assumes the arguments are supplied in order when the function is called, but the arguments can be given in any order provided that the argument name is specified in the call. For example, here x2 gets assigned 2,1,8,7 and x1 will have the default value 1,2,3,4.

```
dan(9:12,c(2,1,8,7))
[1] 4
```

If something is supplied for x1, it overrides the default.


```
> x
[1]  4.0 -1.0  5.0  9.0  4.5
```

```
> sort(x)
[1] -1.0  4.0  4.5  5.0  9.0
```

sort creates a new vector whose values are those of x sorted from smallest to largest.

```
> rev(sort(x))
[1]  9.0  5.0  4.5  4.0 -1.0
```

rev creates a new vector whose values are the same as those of its argument except that the order is reversed.

```
> order(x)
[1] 2 1 5 3 4
```

The order function lists the element numbers (indices) for the smallest, second smallest,..., largest value in x. For example, 2 appears first because the smallest value in x (-1.0) is the second element (see x at the top of the slide). A 4 appears last because the largest value (9.0) is the fourth element.

```
> rank(x)
[1] 2 1 4 5 3
```

The rank function creates a new vector whose values are the ranks of the values of x among all elements in x.

```
> x
[1]  4.0 -1.0  5.0  9.0  4.5
> x=sort(x)
> x
[1] -1.0  4.0  4.5  5.0  9.0
```

None of the previous commands changed x. This command replaces x with the sorted vector so that the elements of x are now in increasing order.


```
> m=matrix(1:9,nrow=3)
> m
     [,1] [,2] [,3]
[1,]    1    4    7
[2,]    2    5    8
[3,]    3    6    9
```

This command creates a matrix with three rows. The values in the matrix are determined by the first argument. In this case, the values will be 1,2,...,9. By default, the matrix is filled down the columns.

```
> t(m)
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

t is the transpose function. It creates a new matrix by exchanging the rows and columns of a given matrix. For example, row 1 of t(m) is column 1 of m.

```
> m=matrix(1:9,nrow=3,byrow=T)
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

If the byrow argument is set to T (short for TRUE), the matrix will be filled across rows rather than down columns. There is also an ncol argument that can be used to set the number of columns. In the example to the right, the number of columns are determined by the number of rows and the 9 elements in the vector 1:9.


```
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

```
> m[2,2]
[1] 5
```

Find the value in the second row and second column of the matrix m.

```
> m[2,]
[1] 4 5 6
```

Find the second row of the matrix m.

```
> m[,2]
[1] 2 5 8
```

Find the second column of the matrix m.

```
> m[1:2,c(1,3)]
     [,1] [,2]
[1,]    1    3
[2,]    4    6
```

Find the submatrix consisting of the first two rows and and first and third columns of m.


```
> m
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
```

The matrix m was not changed by the previous commands.

```
> m=m-5
> m
     [,1] [,2] [,3]
[1,]   -4   -3   -2
[2,]   -1    0    1
[3,]    2    3    4
```

Subtracting 5 from a matrix subtracts 5 from each element. There is a similar behavior for other mathematical operations.

```
> abs(m)
     [,1] [,2] [,3]
[1,]    4    3    2
[2,]    1    0    1
[3,]    2    3    4
```

Using the absolute value function on the matrix m creates a new matrix whose values are the absolute values of those in m.

```
> m
     [,1] [,2] [,3]
[1,]   -4   -3   -2
[2,]   -1    0    1
[3,]    2    3    4
```

The matrix m was not changed by the previous command.

```
> sum(m)
[1] 0
```

Add up all the elements of the matrix m.

```
> apply(m,1,sum)
[1] -9  0  9
```

Separately for each row of m, sum the values.

(1 means rows and 2 means columns.)

```
> apply(m,2,median)
[1] -1  0  1
```

Separately for each column of m, find the median of the values.

```
> sweep(m,1,c(2,9,4),"*")
     [,1] [,2] [,3]
[1,]   -8   -6   -4
[2,]   -9    0    9
[3,]    8   12   16
```

This command creates a new matrix whose 1st row is the 1st row of m multiplied by 2, whose 2nd row is the 2nd row of m multiplied by 9, and whose 3rd row is the 3rd row of m multiplied by 4.


```
> m
     [,1] [,2] [,3]
[1,]   -4   -3   -2
[2,]   -1    0    1
[3,]    2    3    4
```

The matrix m was not changed by the previous commands.

```
> sweep(m,2,c(2,9,4),"-")
     [,1] [,2] [,3]
[1,]   -6  -12   -6
[2,]   -3   -9   -3
[3,]    0   -6    0
```

This command creates a new matrix whose 1st column is the 1st column of m minus 2, whose 2nd column is the 2nd column of m minus 9, and whose 3rd column is the 3rd column of m minus 4.

```
> sweep(m,2,c(2,9,4))
     [,1] [,2] [,3]
[1,]   -6  -12   -6
[2,]   -3   -9   -3
[3,]    0   -6    0
```

Subtraction is the default operation, so this command produces the same result as the command above. Several other operators can be specified between quotation marks.


```
> m=matrix(sample(1:9),nrow=3)
> m
     [,1] [,2] [,3]
[1,]    8    4    1
[2,]    5    6    3
[3,]    9    2    7
```

This command creates a new matrix whose elements are a random shuffling of the integers 1,2,...,9.

```
> sweep(m,2,apply(m,2,median))
     [,1] [,2] [,3]
[1,]    0    0   -2
[2,]   -3    2    0
[3,]    1   -2    4
```

This command combines sweep and apply to subtract the column median from the values in each column.

```
> m[order(m[,1]),]
     [,1] [,2] [,3]
[1,]    5    6    3
[2,]    8    4    1
[3,]    9    2    7
```

This command creates a new matrix whose rows are the rows of m ordered according to the order of the first column of m.


```
> x=c(3,3,9,6,12,15,18,12)

> c(9,4,3,8)%in%x
[1]  TRUE FALSE  TRUE FALSE
```

A vector is created that is TRUE for each element in the first vector that is contained in the second vector, and FALSE for each element in the first vector that is not in the second vector.

```
> table(x)
x
 3  6  9 12 15 18
 2  1  1  2  1  1
```

The number of times that each value appears in x is reported in a table. For example, 3 appears 2 times in x, 6 appears 1 time, 9 appears 1 time, 12 appears 2 times, etc.

```
> unique(x)
[1]  3  9  6 12 15 18
```

A new vector is created that has only one value for each different value in x. For example, although x has two 3's, unique(x) has only one 3.

```
> max(x)
[1] 18
```

The largest value of x is returned.

```
> which.max(x)
[1] 7
```

The element number (index) of the largest value of x is returned.

```
> which(x<7)
[1] 1 2 4
```

The indices of the values of x that are less than 7 are returned.