3. Basic Operations and Numerical Descriptions

Contents

We look at some of the basic operations that you can perform on lists of numbers. It is assumed that you know how to enter data or read data files which is covered in the first chapter, and you know about the basic data types.

3.1. Basic Operations

Once you have a vector (or a list of numbers) in memory most basic operations are available. Most of the basic operations will act on a whole vector and can be used to quickly perform a large number of calculations with a single command. There is one thing to note, if you perform an operation on more than one vector it is often necessary that the vectors all contain the same number of entries.

Here we first define a vector which we will call “a” and will look at how to add and subtract constant numbers from all of the numbers in the vector. First, the vector will contain the numbers 1, 2, 3, and 4. We then see how to add 5 to each of the numbers, subtract 10 from each of the numbers, multiply each number by 4, and divide each number by 5.

```
> a <- c(1,2,3,4)
> a
[1] 1 2 3 4
> a + 5
[1] 6 7 8 9
> a - 10
[1] -9 -8 -7 -6
> a*4
[1]  4  8 12 16
> a/5
[1] 0.2 0.4 0.6 0.8
```

We can save the results in another vector called *b*:

```
> b <- a - 10
> b
[1] -9 -8 -7 -6
```

If you want to take the square root, find e raised to each number, the logarithm, etc., then the usual commands can be used:

```
> sqrt(a)
[1] 1.000000 1.414214 1.732051 2.000000
> exp(a)
[1]  2.718282  7.389056 20.085537 54.598150
> log(a)
[1] 0.0000000 0.6931472 1.0986123 1.3862944
> exp(log(a))
[1] 1 2 3 4
```

By combining operations and using parentheses you can make more complicated expressions:

```
> c <- (a + sqrt(a))/(exp(2)+1)
> c
[1] 0.2384058 0.4069842 0.5640743 0.7152175
```

Note that you can do the same operations with vector arguments. For example to add the elements in vector a to the elements in vector b use the following command:

```
> a + b
[1] -8 -6 -4 -2
```

The operation is performed on an element by element basis. Note this is true for almost all of the basic functions. So you can bring together all kinds of complicated expressions:

```
> a*b
[1]  -9 -16 -21 -24
> a/b
[1] -0.1111111 -0.2500000 -0.4285714 -0.6666667
> (a+3)/(sqrt(1-b)*2-1)
[1] 0.7512364 1.0000000 1.2884234 1.6311303
```

You need to be careful of one thing. When you do operations on vectors they are performed on an element by element basis. One ramification of this is that all of the vectors in an expression must be the same length. If the lengths of the vectors differ then you may get an error message, or worse, a warning message and unpredictable results:

```
> a <- c(1,2,3)
> b <- c(10,11,12,13)
> a+b
[1] 11 13 15 14
Warning message:
longer object length
        is not a multiple of shorter object length in:
a + b
```

As you work in R and create new vectors it can be easy to lose track of what variables you have defined. To get a list of all of the variables that have been defined use the ls() command:

```
> ls()
[1] "a"            "b"            "bubba"        "c"
"last.warning"
[6] "tree"         "trees"
```

Finally, you should keep in mind that the basic operations almost always work on an element by element basis. There are rare exceptions to this general rule. For example, if you look at the minimum of two vectors using the min command you will get the minimum of all of the numbers. There is a special command, called pmin, that may be the command you want in some circumstances:

```
> a <- c(1,-2,3,-4)
> b <- c(-1,2,-3,4)
> min(a,b)
[1] -4
> pmin(a,b)
[1] -1 -2 -3 -4
```

3.2. Basic Numerical Descriptions

Given a vector of numbers there are some basic commands to make it easier to get some of the basic numerical descriptions of a set of numbers. Here we assume that you can read in the tree data that was discussed in a previous chapter. It is assumed that it is stored in a variable called *tree*:

```
> tree <-
read.csv(file="trees91.csv",header=TRUE,sep=",");
> names(tree)
 [1] "C"      "N"      "CHBR"   "REP"    "LFBM"
"STBM"   "RTBM"   "LFNCC"
 [9] "STNCC"  "RTNCC"  "LFBCC"  "STBCC"  "RTBCC"
"LFCACC" "STCACC" "RTCACC"
[17] "LFKCC"  "STKCC"  "RTKCC"  "LFMGCC" "STMGCC"
"RTMGCC" "LFPCC"  "STPCC"
[25] "RTPCC"  "LFSCC"  "STSCC"  "RTSCC"
```

Each column in the data frame can be accessed as a vector. For example the numbers associated with the leaf biomass (LFBM) can be found using *tree$LFBM*:

```
> tree$LFBM
 [1] 0.430 0.400 0.450 0.820 0.520 1.320 0.900 1.180
0.480 0.210 0.270 0.310
[13] 0.650 0.180 0.520 0.300 0.580 0.480 0.580 0.580
0.410 0.480 1.760 1.210
[25] 1.180 0.830 1.220 0.770 1.020 0.130 0.680 0.610
0.700 0.820 0.760 0.770
[37] 1.690 1.480 0.740 1.240 1.120 0.750 0.390 0.870
0.410 0.560 0.550 0.670
[49] 1.260 0.965 0.840 0.970 1.070 1.220
```

The following commands can be used to get the mean, median, quantiles, minimum, maximum, variance, and standard deviation of a set of numbers:

```
> mean(tree$LFBM)
[1] 0.7649074
> median(tree$LFBM)
[1] 0.72
> quantile(tree$LFBM)
    0%    25%    50%    75%   100%
0.1300 0.4800 0.7200 1.0075 1.7600
> min(tree$LFBM)
[1] 0.13
> max(tree$LFBM)
[1] 1.76
> var(tree$LFBM)
[1] 0.1429382
> sd(tree$LFBM)
[1] 0.3780717
```

Finally, the *summary* command will print out the min, max, mean, median, and quantiles:

```
> summary(tree$LFBM)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
 0.1300  0.4800  0.7200  0.7649  1.0080  1.7600
```

The summary command is especially nice because if you give it a data frame it will print out the summary for every vector in the data frame:

```
> summary(tree)
       C               N              CHBR         REP
LFBM
 Min.   :1.000   Min.   :1.000   A1     : 3   Min.   :
1.00   Min.   :0.1300
 1st Qu.:2.000   1st Qu.:1.000   A4     : 3   1st Qu.:
9.00   1st Qu.:0.4800
 Median :2.000   Median :2.000   A6     : 3   Median
:14.00   Median :0.7200
 Mean   :2.519   Mean   :1.926   B2     : 3   Mean
:13.05   Mean   :0.7649
 3rd Qu.:3.000   3rd Qu.:3.000   B3     : 3   3rd
Qu.:20.00   3rd Qu.:1.0075
 Max.   :4.000   Max.   :3.000   B4     : 3   Max.
:20.00   Max.   :1.7600
                                 (Other):36   NA's
:11.00
      STBM             RTBM            LFNCC
STNCC
 Min.   :0.0300   Min.   :0.1200   Min.   :0.880
Min.   :0.3700
 1st Qu.:0.1900   1st Qu.:0.2825   1st Qu.:1.312   1st
Qu.:0.6400
 Median :0.2450   Median :0.4450   Median :1.550
Median :0.7850
 Mean   :0.2883   Mean   :0.4662   Mean   :1.560
Mean   :0.7872
 3rd Qu.:0.3800   3rd Qu.:0.5500   3rd Qu.:1.788   3rd
Qu.:0.9350
 Max.   :0.7200   Max.   :1.5100   Max.   :2.760
Max.   :1.2900

     RTNCC            LFBCC           STBCC
RTBCC
 Min.   :0.4700   Min.   :25.00   Min.   :14.00   Min.
:15.00
 1st Qu.:0.6000   1st Qu.:34.00   1st Qu.:17.00   1st
Qu.:19.00
 Median :0.7500   Median :37.00   Median :18.00
Median :20.00
 Mean   :0.7394   Mean   :36.96   Mean   :18.80   Mean
:21.43
 3rd Qu.:0.8100   3rd Qu.:41.00   3rd Qu.:20.00   3rd
Qu.:23.00
 Max.   :1.5500   Max.   :48.00   Max.   :27.00   Max.
:41.00

     LFCACC           STCACC           RTCACC
LFKCC
 Min.   :0.2100   Min.   :0.1300   Min.   :0.1100
Min.   :0.6500
 1st Qu.:0.2600   1st Qu.:0.1600   1st Qu.:0.1600
1st Qu.:0.8100
 Median :0.2900   Median :0.1700   Median :0.1650
Median :0.9000
 Mean   :0.2869   Mean   :0.1774   Mean   :0.1654
Mean   :0.9053
 3rd Qu.:0.3100   3rd Qu.:0.1875   3rd Qu.:0.1700
3rd Qu.:0.9900
```

```
 Max.   :0.3600   Max.   :0.2400   Max.   :0.2400
Max.   :1.1800

NA's   :1.0000
     STKCC           RTKCC           LFMGCC
STMGCC
 Min.   :0.870   Min.   :0.330   Min.   :0.0700   Min.
:0.100
 1st Qu.:0.940   1st Qu.:0.400   1st Qu.:0.1000   1st
Qu.:0.110
 Median :1.055   Median :0.475   Median :0.1200
Median :0.130
 Mean   :1.105   Mean   :0.473   Mean   :0.1109   Mean
:0.135
 3rd Qu.:1.210   3rd Qu.:0.520   3rd Qu.:0.1300   3rd
Qu.:0.150
 Max.   :1.520   Max.   :0.640   Max.   :0.1400   Max.
:0.190

     RTMGCC            LFPCC             STPCC
RTPCC
 Min.   :0.04000   Min.   :0.1500   Min.   :0.1500
Min.   :0.1000
 1st Qu.:0.06000   1st Qu.:0.2000   1st Qu.:0.2200
1st Qu.:0.1300
 Median :0.07000   Median :0.2400   Median :0.2800
Median :0.1450
 Mean   :0.06648   Mean   :0.2381   Mean   :0.2707
Mean   :0.1465
 3rd Qu.:0.07000   3rd Qu.:0.2700   3rd Qu.:0.3175
3rd Qu.:0.1600
 Max.   :0.09000   Max.   :0.3100   Max.   :0.4100
Max.   :0.2100

     LFSCC            STSCC            RTSCC
 Min.   :0.0900   Min.   :0.1400   Min.   :0.0900
 1st Qu.:0.1325   1st Qu.:0.1600   1st Qu.:0.1200
 Median :0.1600   Median :0.1800   Median :0.1300
 Mean   :0.1661   Mean   :0.1817   Mean   :0.1298
 3rd Qu.:0.1875   3rd Qu.:0.2000   3rd Qu.:0.1475
 Max.   :0.2600   Max.   :0.2800   Max.   :0.1700
```

3.3. Operations on Vectors

Here we look at some commonly used commands that perform operations on lists. The commands include the *sort*, *min*, *max*, and *sum* commands. First, the *sort* command can sort the given vector in either ascending or descending order:

```
> a = c(2,4,6,3,1,5)
> b = sort(a)
> c = sort(a,decreasing = TRUE)
> a
[1] 2 4 6 3 1 5
> b
[1] 1 2 3 4 5 6
> c
[1] 6 5 4 3 2 1
```

The *min* and the *max* commands find the minimum and the maximum numbers in the vector:

```
> min(a)
[1] 1
> max(a)
[1] 6
```

Finally, the *sum* command adds up the numbers in the vector:

```
> sum(a)
[1] 21
```