Topics

- R code
- R Variables
- R Naming Conventions
- Assignment Operators
- DataTypes
  - numeric
  - character
  - date
  - logical
  - complex

More Data Structures

- Vector
- Factor
- List
- Matrix
- Array
- Data Frame

Vector

Vector is a collection of elements, all of the same type

Eg: x<-c(1,2,3,1,5) is a vector of numbers

- y<-c(“R”, ”Excel”, ”SAS”, ”Excel”) is a vector of characters
- A vector cannot be of mixed type
- Vectors are important in R because, R is a vectorised Language

ie. Operations are performed on each element of the vector automatically, without the need to loop through the vector.

```
>x<-c(1,2,3,4,5,6,7)
>x
[1] 1  2  3  4  5  6  7
```

Vectors

- A vector is a sequence of data elements of the same basic type. Members in a vector are officially called **components**.
- Numeric : c(1,2,3)
- Logical : c(TRUE,FALSE,TRUE)
- Character : c(“xyz”,”zz”,”EWQ”)
- Length :The number of elements or members in a vector is given by function *length*
- > length(c("aa", "bb", "cc", "dd", "ee"))
  [1] 5

Vector Operations

```
>x<-c(1,2,3,4,5)
>x
[1] 1  2  3  4  5
>x * 3
[1] 3 6 9 12 15
>x+2
[1] 3 4 5 6 7
>x -3
[1] -2 -1  0  1  2
>x/4
[1] 0.25  0.50 0.75  1.00 1.25
>x^2
[1] 1 4 9 16 25
>sqrt(x)
```

Vector Operations

A short cut for x<-c(1,2,3,4,5) is x<- c(1:5)

This sequence can be created in any direction

>5:1

[1] 5 4 3 2 1

Vector Addition

>x<-1:20

>y<-c(rep(2,10),rep(3,10))

>x+y

[1] 3 4 5 6 7 8 9 10 11 12 14 15 16 17 18 19 20 21 22 23

Vectors

- Vector Creation
  - Using colon operator
  - Using seq function
  - Using c() function
- Vector Operations
  - Combining Vectors
  - Vector Arithmetic
  - Recycling Rule
  - Sorting
- Vector Access

Combining Vectors

- Vectors can be combined via the function c.
- > n = c(2, 3, 5)
  > s = c("aa", "bb", "cc", "dd", "ee")
  > c(n, s)
  [1] "2" "3" "5" "aa" "bb" "cc" "dd" "ee“
- the numeric values are being changed into character strings when the two vectors are combined. This is necessary so as to maintain the same primitive data type for members in the same vector.

Vector Arithmetic

- Arithmetic operations of vectors are performed member-by-member
- For example, suppose we have two vectors a and b.

```
> a = c(1, 3, 5, 7)
> b = c(1, 2, 4, 8)
```

✓ Multiply

```
> 5 * a
[1]  5 15 25 35
```

✓ Addition

```
> a + b
[1]  2  5  9 15
```

✓ Subtraction

```
> a - b
[1]  0  1  1 -1
```

✓ Multplication

```
> a * b
[1]  1  6 20 56
```

✓ Division

```
> a / b
[1] 1.000 1.500 1.250 0.875
```

Recycling rule

- If two vectors are of unequal length, the shorter one will be recycled in order to match the longer vector.
- > u = c(10, 20, 30)
  > v = c(1, 2, 3, 4, 5, 6, 7, 8, 9)
  > u + v
  [1] 11 22 33 14 25 36 17 28 39

which, any, all

which() : produces the indices of vector the condition is satisfied

x <-c(10,2,4,5,0)

which(x>2)

Output: 1, 3, 4

all() : produces a logical vector if a condition is satisfied by all values in a vector

all(x>2): False

any(): produces a logical vector if a condition is satisfied in *any values in a vector*

any(x>2) :TRUE

Vector Access

- Elements of a Vector are accessed using indexing.
- The **[ ] brackets** are used for indexing. Indexing starts with position 1.
- Extracting part of vectors to make new vectors is called subsetting.
- Vectors can be accessed by
  - Index
  - Numeric Index Vector
    - Out of order Indexing
    - Duplicate Indexing
  - Logical Index Vector
  - Named Vector Members
  - Range Index
  - Negative Index
  - Out of Range Index

Vector index

- We retrieve values in a vector by declaring an index inside a *single square bracket* "[]"operator.
- For example, the following shows how to retrieve a vector member. Since the vector index is 1-based, we use the index position 3 for retrieving the third member.
- > s = c("aa", "bb", "cc", "dd", "ee")
  > s[3]
  [1] "cc"
- Unlike other programming languages, the square bracket operator returns more than just individual members. In fact, the result of the square bracket operator is another vector, and s[3] is a vector **slice** containing a single member "cc".

Numeric Index Vector

- A new vector can be sliced from a given vector with a **numeric index vector**, which consists of member positions of the original vector to be retrieved.
- Here it shows how to retrieve a vector slice containing the second and third members of a given vector s.
- > s = c("aa", "bb", "cc", "dd", "ee")
  > s[c(2, 3)]
  [1] "bb" "cc"

Duplicate Indexes

- The index vector allows duplicate values. Hence the following retrieves a member twice in one operation.
- > s[c(2, 3, 3)]
  [1] "bb" "cc" "cc"
- **Outer-of-order-indexes**
- The index vector can even be out-of-order. Here is a vector slice with the order of first and second members reversed.
- > s[c(2, 1, 3)]
  [1] "bb" "aa" "cc"

Logical Index Vector

- A new vector can be sliced from a given vector with a **logical index vector**, which has the same length as the original vector. Its members are TRUE if the corresponding members in the original vector are to be included in the slice, and FALSE if otherwise.
- For example, consider the following vector s of length 5.
- > s = c("aa", "bb", "cc", "dd", "ee")
- To retrieve the the second and fourth members of s, we define a logical vector L of the same length, and have its second and fourth members set as TRUE.
- > L = c(FALSE, TRUE, FALSE, TRUE, FALSE)
  > s[L]
  [1] "bb" "dd"
- The code can be abbreviated into a single line.
- > s[c(FALSE, TRUE, FALSE, TRUE, FALSE)]
  [1] "bb" "dd"

Named Vector Members

We can assign names to vector members.

For example, the following variable v is a character string vector with two members.

```
> v = c("Mary", "Sue")
> v
[1] "Mary" "Sue"
```

We now name the first member as First, and the second as Last.

```
> names(v) = c("First", "Last")
> v
 First   Last
"Mary"  "Sue"
```

Then we can retrieve the first member by its name.

```
> v["First"]
[1] "Mary"
```

Furthermore, we can reverse the order with a character string index vector.

```
> v[c("Last", "First")]
  Last  First
 "Sue" "Mary"
```

Range index

To produce a vector slice between two indexes, we can use the colon operator ":". This can be convenient for situations involving large vectors.

```
> s[2:4]
[1] "bb" "cc" "dd"
```

Negative index

- If the index is negative, it would strip the member whose position has the same absolute value as the negative index. For example, the following creates a vector slice with the third member removed.
- > s[-3]
  [1] "aa" "bb" "dd" "ee"

Out-of-Range Index

- If an index is out-of-range, a missing value will be reported via the symbol NA.
- > s[10]
  [1] NA

Factors

Tell **R** that a variable is **nominal** by making it a factor.

The factor stores the nominal values as a vector of integers in the range [1... k] (where k is the number of unique values in the nominal variable), and an internal vector of character strings (the original values) mapped to these integers.

Variable gender with 20 "male" entries and 30 "female" entries

```
gender <- c(rep("male",20), rep("female", 30))
gender <- factor(gender)
```

Stores gender as 20 1s and 30 2s and associates

1=female, 2=male internally (alphabetically)

```
summary(gender)  # to get the summary information
```