VECTORS

Non Atomic Data objects

- Non Atomic data objects are the objects of atomic classes and can store more than one value.
- An Vector is one dimensional array.
- A matrix is two dimensional array .
- A array is three dimensional array.
- Elements of an array, matrix or vector are of same mode.
- A factor is a categorical variable
- A data frame is a table composed with one or several vectors/factors of same length but possibly of different mode.
- Ts is a time series data set and so contains additional attributes such as frequency and dates.
- List can contain any type of object, included lists!

Vectors

Vectors are one dimensional set of data.
It is very much similar to a set of number in a column or row of an excel.
We can have a vector of number, character, complex and boolean.

Vectors

- The c() function can be used to create vectors of objects.
- X <- c(0.5, 0.6) ## numeric
- X <-c(TRUE, FALSE) ## logical
- X <- c(T, F) ## logical
- X <- c(“a”, “b”, “c”) ## character
- X <- 9:29 ## integer
- X <- c(1+0i, 2+4i) ## complex
- X <- vector(“numeric”, length = 10)
- > x
- [1] 0 0 0 0 0 0 0 0 0 0

Four methods of making vectors

- The syntax of creating vectors is c. We can also use rep function to repeat a number many times. Another function is seq which needs a start val, end val and difference to be given between two numbers. Last we can directly write 1:10.
- 1. v<-c(1,2,3,4,5)
- [1] 1 2 3 4 5
- 2. v<-rep(2,10)
- [1] 2 2 2 2 2
- 3 v<-seq(1:10,2)
- [1] 1,3,5,7,9
- 4 v<-1:10
- [1] 1,2,3,4,5,6,7,8,9,10

Attributes of Vector

- For an vector its mode and length are the sufficient information
- > x <- 1:5
- > mode(x)
- [1] numeric
- > length(x)
- [1] 5

Accessing vectors

- Any element of vector can be accessed by using square brackets.
- v<-c(“Sun”,”Mon”,”Tue”,”Wed”,”Thr”,”Fri”,”Sat”)
- v[1]
- [1] “Sun”
- V[5]
- [1] “Thr”
- v[c(T,F,F,F,F,F,F)]
- [1] "Sun"
- v[c(2,4,6)]
- [1] "Mon" "Wed" "Fri"
- v[c(-2,-6)]
- [1] "Sun" "Tue" "Thru" "Sat"

Vector Arithmatics

```
# Create two vectors.
v1 <- c(3,8,4,5,0,11)
v2 <- c(4,11,0,8,1,2)

# Vector addition.
add.result <- v1+v2

# Vector substraction.
sub.result <- v1-v2

# Vector multiplication.
multi.result <- v1*v2

# Vector division.
divi.result <- v1/v2
```

Recycling of Vector

- If we apply arithmatics operation on two vectors of unequal length, then the elements of the shorter vector are recycled to complete the operations.
- V1<-c(1,2,3,4,5,6)
- v2<-c(1,2)
- v3<-v1+v2
- V3
- [1] 2 4 4 6 6 8

Sort function

- Elements of a vector can be sorted using the sort() function.
- V<-(3,7,1,-6, 16,2)
- > sort(v)
- [1] 1 2 3 4
- > sort(v,decreasing=TRUE)
- [1] 4 3 2 1
- v<-c("F","S","E","A")
- sort(v,decreasing=TRUE)
- [1] "S" "F" "E" "A"
- > sort(v)
- [1] "A" "E" "F" "S"

More on Vectors

- X<-c(“a”,”b”,”c”,”d”,”a”,”b”)
- >x[1]
- [1] “a”
- >x[2]
- [1] “b”
- >x[1:4]
- [1] “a” “b” “c” “d”
- >x[x>”a”]
- “b” ”c” ”d” “b”
- >u<-x>”a”
- >u
- [1] FALSE TRUE TRUE TRUE TRUE FALSE
- >x[u]
- [1] “b” “c” “d” “b”

Vector Name

R objects can have names. This is true for all r objects. This is very useful for writing readable code and self describing objects.

```
> x<-1:3
>names(x)
NULL
>names(x) <-c("height","width","length")
> x
Height width length
1        2        3
```

What will happen if name vector is smaller or larger than data vector?

Mixing Objects

- Y<-c(1.7, “a”) ## character
- Y<-c(TRUE, 2) ## numeric
- Y<-c(“a”, TRUE) ## character
- when different objects are mixed in a vector, coercion occurs so that every element in the vector is of the same class.

Implicit Coercion

- When a vector has two different type of object, It will not give you error but will create a lease common denominator vector.
- In first case one is number and another character. Now a character can not be number so it will convert number to character and Y will become a character vector.
- In second case TRUE will be converted to a number and we will get numeric vector
- In third case TRUE will become string “TRUE” and we get a character string.
- Be aware of this coercion as mixing objects will not error out but will change the mode of vector and as well values.

Explicit Coercion

- Now we can also explicitly coerce objects from one class to another using functions that start with as.
- X <- 0:6
- > Class (x)
- [1] “integer”
- >as.numeric(x)
- [1] 0 1 2 3 4 5 6
- > as.logical (x)
- [1] FALSE TRUE TRUE TRUE TRUE TRUE TRUE
- > as.character (x)
- [1] “0”, “1”, “2”, “3”, “4”, “5”, “6”

- Questions?