Lecture 2
Vectors

GEOG 489 SPRING 2020

R data types

1) Vector: an ordered set of elements that all share the same "mode" (data type).

For instance characters, integers, or floating point numbers.

```
x <- c(5,12,13)

length(x)

mode(x)   #data type
```

R data types

2) Matrix: A matrix is, technically, a vector that has two additional attributes: number or rows and number of columns.

```
mymatrix <- matrix(data=c(1,3,5,8),nrow=2,ncol=2)
#      [,1] [,2]
# [1,]    1    5
# [2,]    3    8
mymatrix2 <- rbind(c(1,5),c(3,8))
```

R data types

3) List: A list is a *vector* in which each element can be any type of data structure, so is the most flexible type of data structure.

We'll define a list as containing a single element numeric vector, a 3-element character vector, and a matrix:

```
mylist <- list(u=2,v=c("abc","def"),w=matrix(data=c(1,2,3,4),

nrow=2,ncol=2))
```

R data types

4) Data frame: A data frame is a list, but with some restrictions, namely, each element of the list must be 1) a vector and 2) the same length of the other elements.

The vectors, however, can be different modes (unlike a matrix). In other words, a data frame is the R equivalent of a spreadsheet.

```
d <- data.frame(kids=c("Jack","Jill"),ages=c(12,10))
     kids    ages
1    Jack     12
2    Jill      10
```

Functions

- Functions: name(parameters)

```
test <- function(x)
{
print(x)
}


test(x=3)
```

Control Statement (Loops)

```
x <- 1:10

for (i in 1:length(x))

{

   # Print the current element of x

   print(x[i])

}
```

Control Statement (ifelse)

```
x <- 1:10
y <- ifelse(x %% 2 == 0,yes="even",no="odd")
```

Vector

1) add/delete vector elements

```
x <- c(88,5,12,13)
# Let's insert a 168 after the 12 and before the 13 into the vector:
x <- c(x[1:3],168,x[4])
```

2) Declare a variable

```
z <- vector(length=2)
z[1] <- 5
```

Vector

3) Recycling

If you perform an operation on two vectors that require them to be the same length (e.g. adding two vectors together), the shorter one is "recycled":

c(1,2,4)+c(6,0,9,20,22)

# is the same as (note the recycling of the first vector):

c(1,2,4,1,2)+c(6,0,9,20,22)

Vector

4) Common vector operations

Vector addition, multiplication, and division can be performed element-wise:

```
x <- c(1,2,4)

x + c(5,0,-1)

x*c(5,0,-1)

x/c(5,4,-1)
```

Vector

5) Vector indexing

```
# We can extract subvectors of a source vector (vector1) by using an "index vector" (vector2) using this format: vector1[vector2]

y <- c(1.2,3.9,0.4,0.12)

y[c(1,3)] # Returns element 1 and 3 of y.

# We can also use logical vectors

# such that the elements are returned if the element is true:

logical_vector <- c(TRUE,TRUE,FALSE,FALSE)

y[logical_vector]
```

Vector

5) Vector indexing

```
# Negative subscripts are used to *exclude* elements:
z <- c(5,12,13)
z[-1] # exclude element 1
subset(z, z > 5)
```

Vector

6) Generating vector sequences with ":"

```
# ":" is an important operator, because it produces a vector of numbers in a regular sequence.

5:8 # produces a vector ranging from 5 to 8, incremented by 1.

5:1 # prodcues a vector ranging from 5 to 1, decremented by 1.
```

7) Generating vector sequences with seq()

```
seq(from=1,to=5,by=1)

# is the same as

1:5
```

Vector

8) which()

```
# If we want the positions of the elements that satisfy the logical argument, we use which()

z <- c(5,2,-3,8)

which(z*z > 8) # This is the numerical index of z that satisfy the logical statement.
```

9) rep()

```
x <- rep(x=8,times=4) # Repeats "8" 4 times

# We can repeat larger vectors as well:

rep(c(5,12,13),3)
```

Vector

10) any() and all()

```
# any() returns TRUE if, for a logical argument, ANY of the vectors returns TRUE:
x <- 1:10
any(x > 8) # TRUE, because vector elements 9 and 10 are greater than 8.
all(x > 8) # FALSE, because not all of the vector elements are greater than 8.
```

11) NA and NULL values

```
# NA and NULL have subtle, but important differences in their meaning.
# NA means "missing data"
# NULL means "value doesn't exist"
```

Vector

12) c() to merge data

```
# When merging multiple modes, the "lowest common denominator" mode will be used.

# Part of the order (from lowest to highest) is as follows:

# list, character, numeric:

x <- c(5,2,"abc")
```