SESSION -3 DATA STRUCTURES IN R

Some key points

In R,

1. No static typing, means
    - No upfront variable type declarations necessary
    - Variable type can change without explicit recasting
2. Variable names can contain any combination of alphanumeric characters along with periods (.) and underscores (_). However, they cannot start with a number or an underscore
3. Index values start from 1. This means, there's no 0-th element.

1. Creation

A. Vector aka *atomic vector*

c(values)

```
log_vec <- c(TRUE,FALSE,T,F) ## logical

char_vec <- c("hello","world!") ## character

num_vec <- c(1L,3L,5L,7L) ## integer

dbl_vec <- c(1.5, 3.1416) ## double or floating point
```

B. Matrix

matrix(values, nrow = m, ncol = n)

```
mat1 <- matrix(1:12)
mat1

mat2 <- matrix(1:12, nrow = 4)
mat2

mat3 <- matrix(1:12, ncol = 4)
```

```
mat3

mat4 <- matrix(1:12, ncol = 4, byrow = TRUE)
mat4

mat5 <- matrix(1:12, ncol = 4, nrow = 5)
mat5
```

C. Dataframe

data.frame(colname1 = vector, colname2 = vector)

```
df_league <- data.frame(
    city=c("green bay", "new england", "seattle", "chicago"),
    teams=c("packers", "pats", "seahawks", "bears"))
df_league
```

D. Lists aka *recursive vectors*

```
qb_stats <- list(name=c("Brett Favre", "Peyton Manning",
                        "Dan Marino", "Drew Brees", "Tom Brady"),
                 yards=c(71838, 69866, 61361, 56388, 53546))
qb_stats
```

---

2. Inspection

A. Metadata: `attributes`

```
attributes(log_vec) # vector

attributes(mat5) # matrix

attributes(df_league) # dataframe

attributes(qb_stats) # list
```

B. Structure and summary: `str` and `summary`

```
str(num_vec)

str(mat2)

str(df_league)

str(qb_stats)
```

C. Names: `names`, `rownames`, `colnames`

```
## vector
names(num_vec)
rownames(num_vec)
```

```
colnames(num_vec)

## list

names(qb_stats)
rownames(qb_stats)
colnames(qb_stats)

## matrix

names(mat3)
rownames(mat3)
colnames(mat3)

## dataframe

names(df_league)
rownames(df_league)
colnames(df_league)
```

D. Size and Dimensionality: `length` and `dim`

```
length(num_vec)

length(mat4)

length(qb_stats)

length(df_league)

mat1
dim(mat1)

mat3
dim(mat3)

df_league
dim(df_league)
```

E. Type of data: `class` and `typeof`

```
## class: enclosing structure information

class(df_league)

class(qb_stats)

class(num_vec)
class(dbl_vec)
class(log_vec)
class(char_vec)

class(mat5)

## typeof: data inside

typeof(num_vec)
```

```
typeof(dbl_vec)

typeof(mat5)

typeof(qb_stats)
typeof(df_league)
```

F. Quick view: head

```
head(log_vec)

head(df_league)

head(qb_stats)

mat <- cbind(norm=rnorm(4000),unif=runif(4000))
dim(mat)
head(mat)
head(mat, n = 10)
```

---

3. Adding values, columns and rows

A. Vector

```
v1 <- 1:10
v1

v1 <- c(v1, 100) ## using c to create a new vector
v1
```

B. Matrix

```
m1 <- matrix(1:20, nrow = 5)
m1

## adding a column
cbind(m1, c(1:5)) # cbind

## adding a row
rbind(m1, c(1:4)) #rbind
```

C. Dataframes

```
df1 <- data.frame(norm=rnorm(10),
                  unif=runif(10),
                  alpha=letters[1:10])

df1
```

```
## adding a column

## method 1: creating and merging data frames
df2 <- data.frame(df1,
           newcol=sample(c(T,F), size = 10, replace = T))
class(df2)

## using cbind
df3 <- cbind(
    df1,
            newcol=sample(c(T,F), size = 10, replace = T)
)
class(df3)

## adding a row

df4 <- df3[4,1:3]
df4

rbind(df1,df4)
```

D. Lists

```
c(list(1:10),list(sample(c(T,F), size = 100, replace = T)))
```

---

4. Indexing / Subsetting

A. Vector

```
v1 <- 100:150
v1

v1[5]

v1[3:5]

v1[c(20,37,45)]

v1[-c(1:25)]
```

B. Matrix

```
m1 <- matrix(1:20, nrow = 5)
m1

m1[5,4] ## individual element

m1[5,] ## include entire row

m1[,4] ## include entire column
```

```
m1[-3,] ## exclude row

m1[-c(1,5),] ## exlude groups of rows

m1[,-1] ## exclude column

m1[,-c(1,2)] ## exclude groups of columns
```

C. Dataframe

Dataframes have same indexing methods as matrices. In addition,

```
df1

df1$norm

df1$unif[5]

df1$alpha[-c(1:5)]
```

D. Lists

Tricky ones.

Pay careful attention to `[ ]` vs `[[ ]]`

```
l <- list(
            data.frame(norm=rnorm(10),unif=runif(10)),
            c(1:5),
            matrix(1:20, nrow = 10)
        )
l

l[1]

#l[1][1,2]

l[[1]][1,2]
```

5. Filtering

A. Vector

```
v1

v1[v1 > 110]

v1[v1 > 110 & v1 < 130] # compound condition

v1[v1 < 110 | v1 > 130]
```

B. Matrix

```
m1

m1[m1 > 5]

m1[m1[,1] < 5, ]
```

C. Dataframe

```
df1[df1[,3]=='f',]

df1[df1$alpha=='f',]
```

6. Type casting and coercion

A. Vector

```
v1 <- c(1L, 3L, 5L)
v1
typeof(v1)

v1 <- c(v1, 6.5) ## coercion
v1
typeof(v1)

v1 <- as.integer(v1) ## typecasting
v1
typeof(v1)
```

B. Matrix

```
m1
class(m1)

d <- as.data.frame(m1[m1[,1] < 5, ]) ## coercion
d
class(d)
```

C. Dataframes

```
d
class(d)

m <- as.matrix(d)
class(m)
```