Data Objects

1. Vectors

Vectors consist of data of the same type, either all numbers, or all characters. Scalars does not really exist in R. A scalar is a vector of length 1.

- Creating vector

```
a <- 1:3
b <- c(1, 2, 3)
c <- seq(1, 3, by = 1)
d <- seq(1, 3, length = 3)
```

- Recycling: when operating on two vectors, R requires them to be of the same length. If not, R would recycle the shorter one until it matches with the long one. You will see an warning if R can not do that.

```
a <- 1:3
b <- 1:5
a + b
```

```
## Warning: longer object length is not a multiple of
shorter object length
```

```
## [1] 2 4 6 5 7
```

- Filter/subset

  ```
  b[c(1, 2)]
  ```

  ```
  ## [1] 1 2
  ```

  ```
  b[b > 1]
  ```

  ```
  ## [1] 2 3 4 5
  ```

- Vectorization

```
a <- 1:5
b <- 6:10
a + b
```

```
## [1]  7  9 11 13 15
```

```
a * b
```

```
## [1]  6 14 24 36 50
```

```
exp(b)
```

```
## [1]   403.4  1096.6  2981.0  8103.1 22026.5
```

```
gamma(b)
```

```
## [1]    120    720   5040  40320 362880
```

- Loops: Ex: replace the “NA” with 0

```
a <- c(1, 2, 3, 4, NA, 47, 10, 100)
for (i in 1:length(a)) {
  if (is.na(a[i]))
      a[i] <- 0
}
a
```

```
## [1]   1   2   3   4   0  47  10 100
```

Avoid loops! This can be vectorized:

```
a <- ifelse(is.na(a), 0, a)
a
```

```
## [1]   1   2   3   4   0  47  10 100
```

Or,

```
a[is.na(a)] <- 0
a
```

```
## [1]   1   2   3   4   0  47  10 100
```

We'll talk more about vectorization after we introduced functions. `sapply`, `Vectorize`

Character strings

```
paste("Yiwen ", "Zhang", sep = "")
```

```
## [1] "Yiwen Zhang"
```

```
strsplit("Yiwen Zhang", split = " ")
```

```
## [[1]]
## [1] "Yiwen" "Zhang"
```

```
c <- c("a", "b", "c", "d")
paste(c, collapse = ".")
```

```
## [1] "a.b.c.d"
```

All elements in one vector must be of the same mode

```
mix <- c(1, 2, 3, "C")
mix
```

```
## [1] "1" "2" "3" "C"
```

2. Matrices

Technically, matrices are vectors with some attributes, i.e. the number of columns and the number of rows. The internal storage of a matrix is in column-major order, meaning that first all of column 1 is stored, and then all of column 2, and so on.

```
x <- matrix(1:10, nrow = 2, ncol = 5)
y <- matrix(nrow = 3, ncol = 3)  ## create NA matrix
```

Matrix operation, dimension match!

```
y <- matrix(1, nrow = 2, ncol = 5)
x * y
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    1    3    5    7    9
## [2,]    2    4    6    8   10
```

```
x + y
```

```
##      [,1] [,2] [,3] [,4] [,5]
## [1,]    2    4    6    8   10
## [2,]    3    5    7    9   11
```

```
z <- t(y)
x %*% z
```

```
##      [,1] [,2]
## [1,]   25   25
## [2,]   30   30
```

Vector matrix operation: R recycle the vector to match the size of the matrix

```
a <- c(10, 20)
a + z
```

```
##      [,1] [,2]
## [1,]   11   21
## [2,]   21   11
## [3,]   11   21
## [4,]   21   11
## [5,]   11   21
```

Matrix size matters!

- R 2.x: vector length limit is \$2$^{\{31\}}$ -1 \approx 2\$billion elements. The biggest matrix is $50,000$ by $50,000$.
- R 3.0 allows vector of length $2^{52} \approx 4.5 \times 10^{15}$. Note that 'save()' and load()' handles vectors up to 2^{48} elements.

3. Lists and data frame

A list is an object that has elements of lists can be of different types. They are very important to real data analysis. The concept is similar to Matlab data structure, python dictionary, perl hash, C struct.

A data frame is essentially a list, of which each element is a vector of the same length. You can still construct a data frame with a list that each element is a matrix (or, even list), of the same length. You don't want to do that on purpose because that would make a lot of weird errors in the analysis.

Example:

```
data(iris)
str(iris)
```

```
## 'data.frame':    150 obs. of  5 variables:
##  $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4
4.9 ...
##  $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9
3.1 ...
##  $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5
1.4 1.5 ...
##  $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2
0.2 0.1 ...
##  $ Species     : Factor w/ 3 levels
"setosa","versicolor",..: 1 1 1 1 1 1 1 1 1 1 ...
```

4. Getting help

- Tab in Rstudio
- help()
- example()
- ?function
- Google!