# Basic R Notebook: Lecture 2

R stores objects in a variety of classes - numeric, integer, character, logical, list, matrix, dataframe and has logical overriding operations when you convert from one class to another.

```r
z <- 0:9
z
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```r
class(z) # integer
```

```
## [1] "integer"
```

```r
typeof(z)
```

```
## [1] "integer"
```

```r
str(z)
```

```
##  int [1:10] 0 1 2 3 4 5 6 7 8 9
```

```r
z1 <- c("a", "b")
z1
```

```
## [1] "a" "b"
```

```r
class(z1) # character
```

```
## [1] "character"
```

```r
typeof(z1)
```

```
## [1] "character"
```

```r
str(z1)
```

```
##  chr [1:2] "a" "b"
```

```r
w <- as.character(z)
w
```

```
##  [1] "0" "1" "2" "3" "4" "5" "6" "7" "8" "9"
```

```r
class(w) # character
```

```
## [1] "character"
```

```r
as.integer(w)
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```r
as.logical(c(5, 0))
```

```
## [1]  TRUE FALSE
```

```r
# TRUE FALSE
# converts all non-zero values to TRUE, 0 to FALSE
```

```r
z
```

```
##  [1] 0 1 2 3 4 5 6 7 8 9
```

```r
z > 1
```

```
##  [1] FALSE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
```

```r
as.numeric(z > 1)
```

```
##  [1] 0 0 1 1 1 1 1 1 1 1
```

```r
# false is zero, true is 1
```

```r
w1 <- c(1, "a")
w1
```

```
## [1] "1" "a"
```

```r
class(w1) # character
```

```
## [1] "character"
```

```r
# shortform for TRUE and FALSE
T
```

```
## [1] TRUE
```

```r
F
```

```
## [1] FALSE
```

Factors:

```r
ct <- c("jap", "kor", "sin", "kor", "jap", "sin", "sin")
class(ct) # character
```

```
## [1] "character"
```

```r
# factor function is used to identify identical values to belong to one category
# such that we can know how many levels (ie. different cities) in the ct vector
# ie. returns all the different categories
fct <- as.factor(ct)

levels(fct)
```

```
## [1] "jap" "kor" "sin"
```

```r
# returns "jap" "kor" "sin"

summary(fct)
```

```
## jap kor sin
##   2   2   3
```

```r
# jap kor sin
#   2   2   3

table(fct) # alternative command for summary()
```

```
## fct
## jap kor sin
##   2   2   3
```

```r
# table and summary give the same results returned
```

The function tapply:

- to categorise dataset into groups (factors)
- within each group, apply a function

```r
# tapply(vector, index, function)
income <- c(500, 1000, 4000, 1244, 3400, 2000, 5000)
mean(income)
```

```
## [1] 2449.143
```

```r
# returns mean over all values in the vector

# take the object income to
tapply(income, fct, mean)
```

```
##      jap      kor      sin
## 1950.000 1122.000 3666.667
```

```
# index: according to what parameter/category to use the function
# returns
#      jap      kor      sin
# 1950.000 1122.000 3666.667
```

```
# rnorm(n = no_of_obs, mean = 0, sd = 1)
# generates random values based on the specified normal distribution

# gl(n = no_of_levels, k = no_of_replications_per_level)
# generates factor levels
med <- data.frame(patient = 1:100, age = rnorm(100, mean = 60, sd = 12), treatment = gl(2, 50, labels =
head(med)
```

```
##   patient      age treatment
## 1       1 55.25237 treatment
## 2       2 16.81212 treatment
## 3       3 61.64824 treatment
## 4       4 65.27401 treatment
## 5       5 51.18895 treatment
## 6       6 46.59681 treatment
```

```
# mean age across treatment groups
tapply(med$age, med$treatment, mean)
```

```
## treatment   control
##  57.83733  59.16882
```

Matrix and arrays

Matrix operations:

```
r <- matrix(c(3:8), nrow = 3, ncol = 2, byrow = F)
# filled up column-wise
r
```

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

```
# get matrix dimensions: returns row and column
dim(r) # 3 2
```

```
## [1] 3 2
```

```
r[2,2] # 7 [row, column]
```

```
## [1] 7
```

```
r[5] # 7 - indexing still reads column wise, so the 5th element of input into the matrix
```

```
## [1] 7
```

```
r[1,] # 3 6 [row, column]
```

```
## [1] 3 6
```

```
class(r) # "matrix" "array"
```

```
## [1] "matrix" "array"
```

```
rownames(r)<- c("A", "B", "C") # specify/add row names
colnames(r) <- c("a", "a")

rownames(r) # returns all row names
```

```
## [1] "A" "B" "C"
```

```
r
```

```
##   a a
## A 3 6
## B 4 7
## C 5 8
```

Array operations:

```
a <- array(c(3:8), c(3, 2)) # values to input, c(no_of_rows, no_of_cols)
a
```

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

```
class(a) # "matrix" "array"
```

```
## [1] "matrix" "array"
```

```
z <- 1:50
dim(z) <- c(5, 2, 5)
z
```

```
## , , 1
##
##      [,1] [,2]
## [1,]    1    6
## [2,]    2    7
## [3,]    3    8
## [4,]    4    9
## [5,]    5   10
##
## , , 2
##
##      [,1] [,2]
## [1,]   11   16
## [2,]   12   17
## [3,]   13   18
## [4,]   14   19
## [5,]   15   20
##
## , , 3
##
##      [,1] [,2]
## [1,]   21   26
## [2,]   22   27
## [3,]   23   28
## [4,]   24   29
## [5,]   25   30
##
## , , 4
##
##      [,1] [,2]
## [1,]   31   36
## [2,]   32   37
## [3,]   33   38
## [4,]   34   39
## [5,]   35   40
##
## , , 5
##
##      [,1] [,2]
## [1,]   41   46
## [2,]   42   47
## [3,]   43   48
## [4,]   44   49
## [5,]   45   50
```

```
z[5,2,5]
```

```
## [1] 50
```

```
z[5,2,1:5]
```

```
## [1] 10 20 30 40 50
```

```
class(z)
```

```
## [1] "array"
```

Other operations with matrices:

```
# to get diagonal matrix where the diagonals are filled with 1 and remaining filled with 0
diag(10)
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
##  [1,]    1    0    0    0    0    0    0    0    0     0
##  [2,]    0    1    0    0    0    0    0    0    0     0
##  [3,]    0    0    1    0    0    0    0    0    0     0
##  [4,]    0    0    0    1    0    0    0    0    0     0
##  [5,]    0    0    0    0    1    0    0    0    0     0
##  [6,]    0    0    0    0    0    1    0    0    0     0
##  [7,]    0    0    0    0    0    0    1    0    0     0
##  [8,]    0    0    0    0    0    0    0    1    0     0
##  [9,]    0    0    0    0    0    0    0    0    1     0
## [10,]    0    0    0    0    0    0    0    0    0     1
```

```
# state the specific values to fill in the diagonals, and the remaining filled with 0
diag(c(5, 3, 2))
```

```
##      [,1] [,2] [,3]
## [1,]    5    0    0
## [2,]    0    3    0
## [3,]    0    0    2
```

```
# alternative way to create a matrix
# row fill
# gives 2 rows, 3 columns
rbind(c(1, 2, 3), c(4, 5, 6))
```

```
##      [,1] [,2] [,3]
## [1,]    1    2    3
## [2,]    4    5    6
```

```
#      [,1] [,2] [,3]
# [1,]    1    2    3
# [2,]    4    5    6

# column fill
# gives 3 rows, 2 columns
cbind(c(1, 2, 3), c(4, 5, 6))
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
#      [,1] [,2]
# [1,]    1    4
# [2,]    2    5
# [3,]    3    6
```

Matrix multiplication:

```
# matrix element-wise multiplication
# the two matrices must be of the same dimensions
# else, error: non-conformable arrays
x <- matrix(5:10, nrow = 3, ncol = 2)
x
```

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

```
y <- matrix(1:6, nrow = 3, ncol = 2)
y
```

```
##      [,1] [,2]
## [1,]    1    4
## [2,]    2    5
## [3,]    3    6
```

```
x*y # element wise multiplication
```

```
##      [,1] [,2]
## [1,]    5   32
## [2,]   12   45
## [3,]   21   60
```

```
# if a matrix is multiplied by a scalar value, every element in the matrix will be multiplied by the sc
```

```
# Multiplying matrix with vector
# the vector will then be promoted into a row or column matrix to make the two arguments conformable
m <- matrix(1:8, nrow=2)
vec <- 1:2

# matrix m
#      [,1] [,2] [,3] [,4]
# [1,]    1    3    5    7
# [2,]    2    4    6    8

print(vec*m)
```

```
##      [,1] [,2] [,3] [,4]
## [1,]    1    3    5    7
## [2,]    4    8   12   16
```

```
# result
# 1*1=1   1*3=3   1*5=5   1*7=7
# 2*2=4   2*4=8   2*6=12  2*8=16

# first row multiplied by 1
# second row multiplied by 2

#      [,1] [,2] [,3] [,4]
# [1,]    1    3    5    7
# [2,]    4    8   12   16




# multiplies two matrices if they are conformable (matrix multiplication)
help("%*%")

# number of columns in first matrix must equal to the number of rows in the second matrix
# resulting matrix will be number of rows from the first matrix, number of columns from the second matr
# (m x n) %*% (n x k) = (m x k)
# else, error: non-conformable arrays

# t(x) transpose matrix function
x %*% t(y)
```

```
##      [,1] [,2] [,3]
## [1,]   37   50   63
## [2,]   42   57   72
## [3,]   47   64   81
```

Solutions of linear equations:

```
# ax = b
# a: coefficients of the equation
# b: vector or matrix of the equation (ie. RHS values)
a <- array(c(2, 1, -1, 2), c(2, 2))
a
```

```
##      [,1] [,2]
## [1,]    2   -1
## [2,]    1    2
```

```
b <- c(4, 4)
b
```

```
## [1] 4 4
```

```
solve(a, b)
```

```
## [1] 2.4 0.8
```

```
# if b is not specified, takes b as an identity matrix and solve it
# ie. resulting solution is the inverse matrix of a
solve(a)
```

```
##      [,1] [,2]
## [1,]  0.4  0.2
## [2,] -0.2  0.4
```

Eigen decomposition of a matrix:

```
a <- matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2)
a
```

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

```
eigen(a) # a: matrix
```

```
## eigen() decomposition
## $values
## [1]  5.3722813 -0.3722813
##
## $vectors
##            [,1]       [,2]
## [1,] -0.5657675 -0.9093767
## [2,] -0.8245648  0.4159736
```

```
# eigenvalue is the factor by which a eigenvector is scaled
E <- eigen(a)
E$values
```

```
## [1]  5.3722813 -0.3722813
```

```
E$vectors
```

```
##            [,1]       [,2]
## [1,] -0.5657675 -0.9093767
## [2,] -0.8245648  0.4159736
```

List consists of an ordered collection of objects that can be of different or the same type.

```
barack <- list(age = 59, sex = "M", child.ages = c(22, 19))
barack
```

```
## $age
## [1] 59
##
## $sex
## [1] "M"
##
## $child.ages
## [1] 22 19
```

```r
class(barack) # list
```

```
## [1] "list"
```

```r
# accessing elements in the list

barack$age
```

```
## [1] 59
```

```r
barack[1] # age value: 59 - first list element
```

```
## $age
## [1] 59
```

```r
barack$child.ages
```

```
## [1] 22 19
```

```r
barack$child.ages[1]
```

```
## [1] 22
```

```r
barack[[1]] # 59 first component of the first list element
```

```
## [1] 59
```

```r
class(barack[1])
```

```
## [1] "list"
```

```r
class(barack[[1]])
```

```
## [1] "numeric"
```

```r
# numeric

# [ ] and [[ ]] are different in a way that [[ ]] will only return a single element via indexing using
# while [ ] allows for indexing by vectors
# both still returns a single element
```

```r
serena <- list(age = 39, sex = "F", child.ages = 3)
serena
```

```
## $age
## [1] 39
##
## $sex
## [1] "F"
##
## $child.ages
## [1] 3
```

```
class(serena) # list
```

```
## [1] "list"
```

```
celg <- c(barack, serena) # joins both lists together, elements with the same name would not combine to
celg
```

```
## $age
## [1] 59
##
## $sex
## [1] "M"
##
## $child.ages
## [1] 22 19
##
## $age
## [1] 39
##
## $sex
## [1] "F"
##
## $child.ages
## [1] 3
```

```
celg$sex # only the first list element will be returned if there are two elements of the same name
```

```
## [1] "M"
```

```
celg[1] # first element in the celg list
```

```
## $age
## [1] 59
```

```
celg[3]
```

```
## $child.ages
## [1] 22 19
```

```
celg[4]
```

```
## $age
## [1] 39
```

Dataframes are a tightly coupled collection of variables that share many of the properties of matrices and lists and is the fundamental data structure that will be used in most of this course.

```
A <- data.frame(names = c("barack", "serena"), ages = c(58, 39), children = c(2, 1))
A
```

```
##    names ages children
## 1 barack   58        2
## 2 serena   39        1
```

```
A$names
```

```
## [1] "barack" "serena"
```

```
class(A) # dataframe
```

```
## [1] "data.frame"
```

```
class(A$ages) # numeric
```

```
## [1] "numeric"
```

```
# add new column
A$spouse = c("michel", "alexis")
A
```

```
##    names ages children spouse
## 1 barack   58        2 michel
## 2 serena   39        1 alexis
```

Dyplr and tibble ... not covered now but good to know

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------------------------------------------------

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.3     v dplyr   1.0.2
## v tidyr   1.1.2     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------------------------------------------
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```r
starwars
```

```
## # A tibble: 87 x 14
##    name  height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Luke~    172    77 blond      fair       blue            19   male  mascu~
##  2 C-3PO    167    75 <NA>       gold       yellow         112   none  mascu~
##  3 R2-D2     96    32 <NA>       white, bl~ red             33   none  mascu~
##  4 Dart~    202   136 none       white      yellow          41.9 male  mascu~
##  5 Leia~    150    49 brown      light      brown           19   fema~ femin~
##  6 Owen~    178   120 brown, gr~ light      blue            52   male  mascu~
##  7 Beru~    165    75 brown      light      blue            47   fema~ femin~
##  8 R5-D4     97    32 <NA>       white, red red             NA   none  mascu~
##  9 Bigg~    183    84 black      light      brown           24   male  mascu~
## 10 Obi-~    182    77 auburn, w~ fair       blue-gray       57   male  mascu~
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

```r
class(starwars)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

```r
# $>$ is pipe
starwars %>% dim()
```

```
## [1] 87 14
```

```r
# equivalent to dim(starwars)
# starwars %>% summary()

starwars %>% select(mass) %>% summary()
```

```
##       mass
##  Min.   :  15.00
##  1st Qu.:  55.60
##  Median :  79.00
##  Mean   :  97.31
##  3rd Qu.:  84.50
##  Max.   :1358.00
##  NA's   :28
```

```r
starwars %>% filter(sex=="female")
```

```
## # A tibble: 16 x 14
##    name  height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Leia~    150 49    brown      light      brown             19 fema~ femin~
##  2 Beru~    165 75    brown      light      blue              47 fema~ femin~
##  3 Mon ~    150 NA    auburn     fair       blue              48 fema~ femin~
##  4 Shmi~    163 NA    black      fair       brown             72 fema~ femin~
##  5 Ayla~    178  55   none       blue       hazel             48 fema~ femin~
##  6 Adi ~    184  50   none       dark       blue              NA fema~ femin~
##  7 Cordé    157  NA   brown      light      brown             NA fema~ femin~
##  8 Lumi~    170  56.2 black      yellow     blue              58 fema~ femin~
##  9 Barr~    166  50   black      yellow     blue              40 fema~ femin~
## 10 Dormé    165  NA   brown      light      brown             NA fema~ femin~
## 11 Zam ~    168  55   blonde     fair, gre~ yellow            NA fema~ femin~
## 12 Taun~    213  NA   none       grey       black             NA fema~ femin~
## 13 Joca~    167  NA   white      fair       blue              NA fema~ femin~
## 14 Shaa~    178  57   none       red, blue~ black             NA fema~ femin~
## 15 Rey       NA  NA   brown      light      hazel             NA fema~ femin~
## 16 Padm~    165  45   brown      light      brown             46 fema~ femin~
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```

```
starwars %>% arrange(height,mass)
```

```
## # A tibble: 87 x 14
##    name  height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Yoda      66    17 white      green      brown            896 male  mascu~
##  2 Ratt~     79    15 none       grey, blue unknown           NA male  mascu~
##  3 Wick~     88    20 brown      brown      brown              8 male  mascu~
##  4 Dud ~     94    45 none       blue, grey yellow            NA male  mascu~
##  5 R2-D2     96    32 <NA>       white, bl~ red               33 none  mascu~
##  6 R4-P~     96    NA none       silver, r~ red, blue         NA none  femin~
##  7 R5-D4     97    32 <NA>       white, red red               NA none  mascu~
##  8 Sebu~    112    40 none       grey, red  orange            NA male  mascu~
##  9 Gasg~    122    NA none       white, bl~ black             NA male  mascu~
## 10 Watto    137    NA black      blue, grey yellow            NA male  mascu~
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

```
starwars %>% arrange(desc(height))
```

```
## # A tibble: 87 x 14
##    name  height  mass hair_color skin_color eye_color birth_year sex   gender
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
##  1 Yara~    264    NA none       white      yellow          NA   male  mascu~
##  2 Tarf~    234   136 brown      brown      blue            NA   male  mascu~
##  3 Lama~    229    88 none       grey       black           NA   male  mascu~
##  4 Chew~    228   112 brown      unknown    blue           200   male  mascu~
##  5 Roos~    224    82 none       grey       orange          NA   male  mascu~
##  6 Grie~    216   159 none       brown, wh~ green, y~       NA   male  mascu~
##  7 Taun~    213    NA none       grey       black           NA   fema~ femin~
##  8 Rugo~    206    NA none       green      orange          NA   male  mascu~
##  9 Tion~    206    80 none       grey       black           NA   male  mascu~
## 10 Dart~    202   136 none       white      yellow          41.9 male  mascu~
## # ... with 77 more rows, and 5 more variables: homeworld <chr>, species <chr>,
## #   films <list>, vehicles <list>, starships <list>
```

```r
starwars %>% slice_head(n=3)
```

```
## # A tibble: 3 x 14
##   name  height  mass hair_color skin_color eye_color birth_year sex   gender
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>          <dbl> <chr> <chr>
## 1 Luke~    172    77 blond      fair       blue              19 male  mascu~
## 2 C-3PO    167    75 <NA>       gold       yellow           112 none  mascu~
## 3 R2-D2     96    32 <NA>       white, bl~ red               33 none  mascu~
## # ... with 5 more variables: homeworld <chr>, species <chr>, films <list>,
## #   vehicles <list>, starships <list>
```