Notebook1

R – Data Structure

- Vectors
- Lists
- Matrices
- Arrays
- Factors
- Data Frames

Vectors

- The most basic type of R object is a vector.

The c() function can be used to create vectors.

```
x <- c("asas", "sds")
x
```

```
[1] "asas" "sds"
```

```
class(x)
```

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

```
x <- c(TRUE, FALSE)
x
```

```
[1]  TRUE FALSE
```

```
  class(x)
```

```
[1] "logical"
```

```
  x <- 3:10
  x
```

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

```
  class(x)
```

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

```
  x <- c(1,5)
  x
```

```
[1] 1 5
```

```
  class(x)
```

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

- You can also use the vector() function to initialize vectors.

```
  x <- vector("numeric", length = 10)
  x
```

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

- A vector can only contain objects of the same class.

```
  x <- c(7, 'd')
  x
```

```
[1] "7" "d"
```

```r
x <- c(TRUE, 2)
x
```

```
[1] 1 2
```

```r
x <- c('a', T)
x
```

```
[1] "a"    "TRUE"
```

- When different objects are mixed in a vector, coercion occurs so that every element in the vector is of the same class.

Explicit Coercion

```r
x <- 5
class(x)
```

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

```r
as.logical(x)
```

```
[1] TRUE
```

```r
as.character(x)
```

```
[1] "5"
```

```r
x <- c('a', 'b')
as.numeric(x)
```

```
Warning: NAs introduced by coercion
```

```
[1] NA NA
```

Matrics

```r
m <- matrix(nrow=2,ncol=3)
m
```

```
     [,1] [,2] [,3]
[1,]   NA   NA   NA
[2,]   NA   NA   NA
```

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

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

```r
dim(m) #dimension of
```

```
[1] 2 3
```

Matrices can also be created directly from vectors by adding a dimension attribute.

```r
v <- 1:10
v
```

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

```r
dim(v) <-c(2,5)
v
```

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

Matrices can be created by column-binding or row-binding with the cbind() and rbind() functions.

```r
x <- 1:3
y <- 10:12
cbind(x, y)
```

```
     x  y
[1,] 1 10
[2,] 2 11
[3,] 3 12
```

```r
rbind(x, y)
```

```
  [,1] [,2] [,3]
x    1    2    3
y   10   11   12
```

Lists

Lists are a special type of vector that can contain elements of different classes.

```r
x <- list(1, "a", TRUE, 1 + 4i)
x
```

```
[[1]]
[1] 1

[[2]]
[1] "a"

[[3]]
[1] TRUE

[[4]]
[1] 1+4i
```

We can also create an empty list of a prespecified length with the vector() function.

```r
x <- vector("list", length = 5)
x
```

```
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

[[4]]
NULL

[[5]]
NULL
```

Factors

Factors are used to represent categorical data and can be unordered or ordered. One can think of a factor as an integer vector where each integer has a label.

```
x <- factor(c("yes", "yes", "no", "yes", "no"))
x
```

```
[1] yes yes no  yes no
Levels: no yes
```

```
## See the underlying representation of factor
unclass(x)
```

```
[1] 2 2 1 2 1
attr(,"levels")
[1] "no"  "yes"
```

Often factors will be automatically created for you when you read a dataset in using a function like read.table(). Those functions often default to creating factors when they encounter data that look like characters or strings.

Data Frames

- Data frames are used to store tabular data in R. They are an important type of object in R and are used in a variety of statistical modeling applications.
- Unlike matrices, data frames can store different classes of objects in each column.
- Data frames are usually created by reading in a dataset using the read.table() or read.csv().
- data frames can also be created explicitly with the data.frame()

```
x <- data.frame(X = 1:4, Y = c(T, T, F, F))
x
```

```
  X     Y
1 1  TRUE
2 2  TRUE
3 3 FALSE
4 4 FALSE
```

```
x <- data.frame(X = 1:4, Y = c(T, T, F, F))
x
```

```
  X     Y
1 1  TRUE
2 2  TRUE
3 3 FALSE
4 4 FALSE
```

```
nrow(x)
```

```
[1] 4
```

```
ncol(x)
```

```
[1] 2
```

Data frames can be converted to a matrix by calling data.matrix()

```
m <- data.matrix(x)
m
```

```
     X Y
[1,] 1 1
[2,] 2 1
[3,] 3 0
[4,] 4 0
```

Get the structure of Data Frame

```
str(x)
```

```
'data.frame':   4 obs. of  2 variables:
 $ X: int  1 2 3 4
 $ Y: logi  TRUE TRUE FALSE FALSE
```

```
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.2     v readr     2.1.4
v forcats   1.0.0     v stringr   1.5.0
v ggplot2   3.4.2     v tibble    3.2.1
v lubridate 1.9.2     v tidyr     1.3.0
v purrr     1.0.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to beco
```

```
glimpse(x)
```

```
Rows: 4
Columns: 2
$ X <int> 1, 2, 3, 4
$ Y <lgl> TRUE, TRUE, FALSE, FALSE
```

```
summary(x)
```

```
       X            Y
 Min.   :1.00   Mode :logical
 1st Qu.:1.75   FALSE:2
 Median :2.50   TRUE :2
 Mean   :2.50
 3rd Qu.:3.25
 Max.   :4.00
```

Add column

```
x$z <- c("a","b","c","d")
x
```

```
  X     Y z
1 1  TRUE a
2 2  TRUE b
3 3 FALSE c
4 4 FALSE d
```

Add row

```
x1 <- rbind(x,c(5,T,"e"))
x1
```

```
  X     Y z
1 1  TRUE a
2 2  TRUE b
3 3 FALSE c
4 4 FALSE d
5 5  TRUE e
```

Reading Data Files with read.table()

The read.table() function has a few important arguments:

- ***file***, the name of a file, or a connection
- ***header***, logical indicating if the file has a header line

- ***sep***, a string indicating how the columns are separated
- ***colClasses***, a character vector indicating the class of each column in the dataset
- ***nrows***, the number of rows in the dataset. By default read.table() reads an entire file.
- ***comment.char***, a character string indicating the comment character. This defalts to "#".
- ***skip***, the number of lines to skip from the beginning
- ***stringsAsFactors***, should character variables be coded as factors? This defaults to TRUE.

```
x <- read.csv('iris.csv', header = T, nrows = 50, skip = 10)
head(x)
```

```
  X10 X4.9 X3.1 X1.5 X0.1 Iris.setosa
1  11  5.4  3.7  1.5  0.2 Iris-setosa
2  12  4.8  3.4  1.6  0.2 Iris-setosa
3  13  4.8  3.0  1.4  0.1 Iris-setosa
4  14  4.3  3.0  1.1  0.1 Iris-setosa
5  15  5.8  4.0  1.2  0.2 Iris-setosa
6  16  5.7  4.4  1.5  0.4 Iris-setosa
```

https://www.kaggle.com/datasets/uciml/iris?resource=download

Read and Write xlsx files in R

```
# Install tidyverse
#install.packages("tidyverse")
# or just readxl
#install.packages("readxl")
```

```
require(readxl)
```

```
Loading required package: readxl
```

https://web.stanford.edu/~ashishg/msande111/excel/iris.xls

```
df <- read_excel("iris (1).xls")
```

```
New names:
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
* `` -> `...6`
* `` -> `...7`
* `` -> `...8`
* `` -> `...9`
* `` -> `...10`
* `` -> `...11`
```

```
head(df)
```

```
# A tibble: 6 x 11
  `Des:`             ...2  ...3  ...4  ...5  ...6   ...7 ...8  ...9  ...10 ...11
  <chr>              <chr> <chr> <chr> <chr> <lgl> <dbl> <chr> <chr> <dbl> <dbl>
1 Sepal Length (cm)  Sepa~ Peta~ Peta~ Class NA       NA alpha obj      NA    NA
2 7                  3.20~ 4.70~ 1.39~ Iris~ NA        0 0     0         0     1
3 6.4000000000000004 3.20~ 4.5   1.5   Iris~ NA        0 <NA>  <NA>      0     1
4 6.9000000000000004 3.10~ 4.90~ 1.5   Iris~ NA        0 <NA>  <NA>      0     1
5 5.5                2.29~ 4     1.3   Iris~ NA        0 <NA>  <NA>      0     1
6 6.5                2.79~ 4.59~ 1.5   Iris~ NA       NA <NA>  <NA>      0     1
```

```
df <- read_excel("iris (1).xls",range=cell_cols("B:F"))
```

```
New names:
* `` -> `...2`
* `` -> `...3`
* `` -> `...4`
* `` -> `...5`
```

```
head(df)
```

```
# A tibble: 6 x 5
  `Des:`             ...2              ...3              ...4           ...5
  <chr>              <chr>             <chr>             <chr>          <chr>
1 Sepal Length (cm)  Sepal Width (cm)  Petal Length (cm) Petal Width (c~ Class
2 7                  3.2000000000000002 4.7000000000000002 1.399999999999~ Iris~
```

```
3 6.4000000000000004 3.2000000000000002 4.5                1.5             Iris~
4 6.9000000000000004 3.1000000000000001 4.9000000000000004 1.5             Iris~
5 5.5                2.2999999999999998 4                  1.3             Iris~
6 6.5                2.7999999999999998 4.5999999999999996 1.5             Iris~
```

```
library("writexl")
# saves the dataframe at the specified path
write_xlsx(df,"df.xlsx",col_names = TRUE,format_headers = TRUE)
```

```
write.csv(df,"df.csv")
```

Subsetting R Objects

There are three operators that can be used to extract subsets of R objects.

> The [[operator is used to extract elements of a list or a data frame.
>
> The $ operator is used to extract elements of a list or data frame by literal name.

Subsetting a Vector

```
x <- c("a", "b", "c", "c", "d", "a")

x[1]
```

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

```
x[1:4]
```

```
[1] "a" "b" "c" "c"
```

```
x[c(1,3,4)]
```

```
[1] "a" "c" "c"
```

Arbitrary subsetting

```r
x[c(1,3,4)]
```

```
[1] "a" "c" "c"
```

We can also pass a logical sequence

```r
x <- c(5,6,10,12,3,4,5)

x[x > 7]
```

```
[1] 10 12
```

```r
x[x>3&x<7]
```

```
[1] 5 6 4 5
```

Subsetting a Matrix

```r
x <- matrix(1:6, 2, 3)
x
```

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

```r
x[1,2]
```

```
[1] 3
```

```r
x[,1]
```

```
[1] 1 2
```

```r
x[2,]
```

```
[1] 2 4 6
```

Subsetting Lists

```
x <- list(1, "a", TRUE, 1 + 4i)

x[[1]]
```

```
[1] 1
```

```
x <- list(foo = 1:4, bar = 0.6)
x
```

```
$foo
[1] 1 2 3 4

$bar
[1] 0.6
```

```
x$foo
```

```
[1] 1 2 3 4
```

```
x[['foo']]
```

```
[1] 1 2 3 4
```

```
x <- list(1,2,3,4,5,66,777)
x[1:4]
```

```
[[1]]
[1] 1

[[2]]
[1] 2

[[3]]
[1] 3

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

```r
x <- list(a = list(10, 12, 14), b = c(3.14, 2.81))
x[[1]][[1]]
```

```
[1] 10
```

Subsetting Nested Elements of a List

```r
x <- list(a = list(10, 12, 14), b = c(3.14, 2.81))

## Get the 3rd element of the 1st element
x[[1]][[1]]
```

```
[1] 10
```

```r
x[[c(1,3)]]
```

```
[1] 14
```

```r
## 1st element of the 2nd element
```

Extracting Multiple Elements of a List

```r
x <- list(1, 4, 6, 8, 9, 'a', 'b', 'c')

#x[1:4]

x[c(1,2,6)]
```

```
[[1]]
[1] 1

[[2]]
[1] 4

[[3]]
[1] "a"
```

Vectorized Operations

- addition, subtraction, multiplication and division

```
x <- 1:4
y <- 6:9

x+y
```

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

- Logical comparision

```
x > 3
```

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

Vectorized Matrix Operations

we can do element-by-element operations on matrices.

```
x <- matrix(1:4, 2, 2)
y <- matrix(3:6, 2, 2)
x
```

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

```
y
```

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

```
x+y
```

```
     [,1] [,2]
[1,]    4    8
[2,]    6   10
```

Names

R objects can have names, which is very useful for writing readable code and self-describing objects.

```
x <- 1:5
x
```

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

```
names(x)
```

```
NULL
```

```
names(x) <- c("V1","V2","V3","V4","V5")
x
```

```
V1 V2 V3 V4 V5
 1  2  3  4  5
```

- Lists can also have names, which is often very useful.

```
x <- list("A"=1:4, "B"=2, "C"=3)
x
```

```
$A
[1] 1 2 3 4

$B
[1] 2

$C
[1] 3
```

- Matrices can have both column and row names.

```
m <- matrix(1:4, nrow = 2, ncol = 2)
m
```

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

```
dimnames(m) <- list(c("a", "b"), c("c", "d"))
m
```

```
  c d
a 1 3
b 2 4
```

```
colnames(m) <- c("h", "f")
rownames(m) <- c("x", "z")
m
```

```
  h f
x 1 3
z 2 4
```

```
df <- data.frame(11:15,c("a","b","c","d","e"))
df
```

```
  X11.15 c..a....b....c....d....e..
1     11                         a
2     12                         b
3     13                         c
4     14                         d
5     15                         e
```

```
names(df) <- c("A","B")
df
```

```
   A B
1 11 a
2 12 b
3 13 c
4 14 d
5 15 e
```

```r
row.names(df) <- 1:5
df
```

```
  A B
1 11 a
2 12 b
3 13 c
4 14 d
5 15 e
```

Missing Values

- is.na() is used to test objects if they are NA
- is.nan() is used to test for NaN
- NA values have a class also, so there are integer NA, character NA, etc.
- A NaN value is also NA but the converse is not true.

```r
x <- c(1, 2, NA, 10, 3)
is.na(x)
```

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

```r
x[!(is.na(x))]
```

```
[1]  1  2 10  3
```

```r
is.nan(x)
```

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

```r
x <- c(1, 2, NaN, NA, 4)
is.na(x)
```

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

```r
is.nan(x)
```

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

```r
any(is.nan(x))
```

```
[1] TRUE
```

```r
NA > 5
```

```
[1] NA
```

```r
10 == NA
```

```
[1] NA
```

```r
NA | T
```

```
[1] TRUE
```

```r
NA | F
```

```
[1] NA
```

```r
x <- NA
y <- NA
x==y
```

```
[1] NA
```

```r
NA*0
```

```
[1] NA
```

Removing NA Values

```r
x <- c(1, 2, NA, 4, NA, 5)
#remove missing values (NAs).
```

Complete.cases()

Complete cases in R, To eliminate missing values from a vector, matrix, or data frame.

```r
x <- c(1, 2, NA, 4, NA, 5)

x[complete.cases(x)]
```

```
[1] 1 2 4 5
```

```r
dim(airquality)
```

```
[1] 153   6
```

```r
head(airquality)
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

```r
complete.cases(airquality)
```

```
  [1]  TRUE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE
 [13]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
 [25] FALSE FALSE FALSE  TRUE  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE
 [37] FALSE  TRUE FALSE  TRUE  TRUE FALSE FALSE  TRUE FALSE FALSE  TRUE  TRUE
 [49]  TRUE  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
 [61] FALSE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
 [73]  TRUE  TRUE FALSE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE FALSE
 [85]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE
```

```
 [97] FALSE FALSE  TRUE  TRUE  TRUE FALSE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
[109]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE FALSE  TRUE
[121]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[133]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
[145]  TRUE  TRUE  TRUE  TRUE  TRUE FALSE  TRUE  TRUE  TRUE
```

```
head(airquality[complete.cases(airquality),])
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
7    23     299  8.6   65     5   7
8    19      99 13.8   59     5   8
```

```
df <- data.frame(A=c(10, 2, NA, 16, NA, 23),
                 B=c(NA, 45, 45, 12, NA, 18),
C=c(NA, NA, 12, 5, 18, 22))
df
```

```
   A  B  C
1 10 NA NA
2  2 45 NA
3 NA 45 12
4 16 12  5
5 NA NA 18
6 23 18 22
```

Rows containing NA in specific columns of a data frame should be removed.

```
df[ , c('A', 'B')]
```

```
   A  B
1 10 NA
2  2 45
3 NA 45
4 16 12
5 NA NA
6 23 18
```

```
complete.cases(df)
```

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

```
complete.cases(df[ , c('A', 'B')])
```

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

```
df[complete.cases(df), ]
```

```
   A  B  C
4 16 12  5
6 23 18 22
```

```
df[complete.cases(df[ , c('A', 'B')]), ]
```

```
   A  B  C
2  2 45 NA
4 16 12  5
6 23 18 22
```

The function **na.omit()** returns the object with listwise deletion of missing values.

```
df1 <- df
na.omit(df1)
```

```
   A  B  C
4 16 12  5
6 23 18 22
```

```
df
```

```
   A  B  C
1 10 NA NA
2  2 45 NA
3 NA 45 12
4 16 12  5
5 NA NA 18
6 23 18 22
```

```
sum(is.na(df))
```

```
[1] 6
```

```
colSums(is.na(df))
```

```
A B C
2 2 2
```

```
colSums(is.na(df))
```

```
A B C
2 2 2
```

```
sum(rowSums(is.na(airquality)) > 0)
```

```
[1] 42
```

```
head(airquality)
```

```
  Ozone Solar.R Wind Temp Month Day
1    41     190  7.4   67     5   1
2    36     118  8.0   72     5   2
3    12     149 12.6   74     5   3
4    18     313 11.5   62     5   4
5    NA      NA 14.3   56     5   5
6    28      NA 14.9   66     5   6
```

Q. Find number of rows with missing value **rowSums()**

```r
#rowSums(!is.na(airquality))
rS <- rowSums(is.na(airquality))
length(rS[rS>0])
```

```
[1] 42
```

Q. Find rows with missing value **rowSums()**

```r
names(rS)<-1:nrow(airquality)
names(rS[rS>0])
```

```
 [1] "5"   "6"   "10"  "11"  "25"  "26"  "27"  "32"  "33"  "34"  "35"  "36"
[13] "37"  "39"  "42"  "43"  "45"  "46"  "52"  "53"  "54"  "55"  "56"  "57"
[25] "58"  "59"  "60"  "61"  "65"  "72"  "75"  "83"  "84"  "96"  "97"  "98"
[37] "102" "103" "107" "115" "119" "150"
```

Q. Find columns with missing value **(colSums)**

```r
cS <- colSums(is.na(airquality))
names(cS[cS>0])
```

```
[1] "Ozone"   "Solar.R"
```