STA 517 3.0 Programming and Statistical Computing with R

🛠️ Data Structures in R

Dr Thiyanga Talagala

Today's menu

- Vector
- Matrix
- Array
- Data Frame
- List

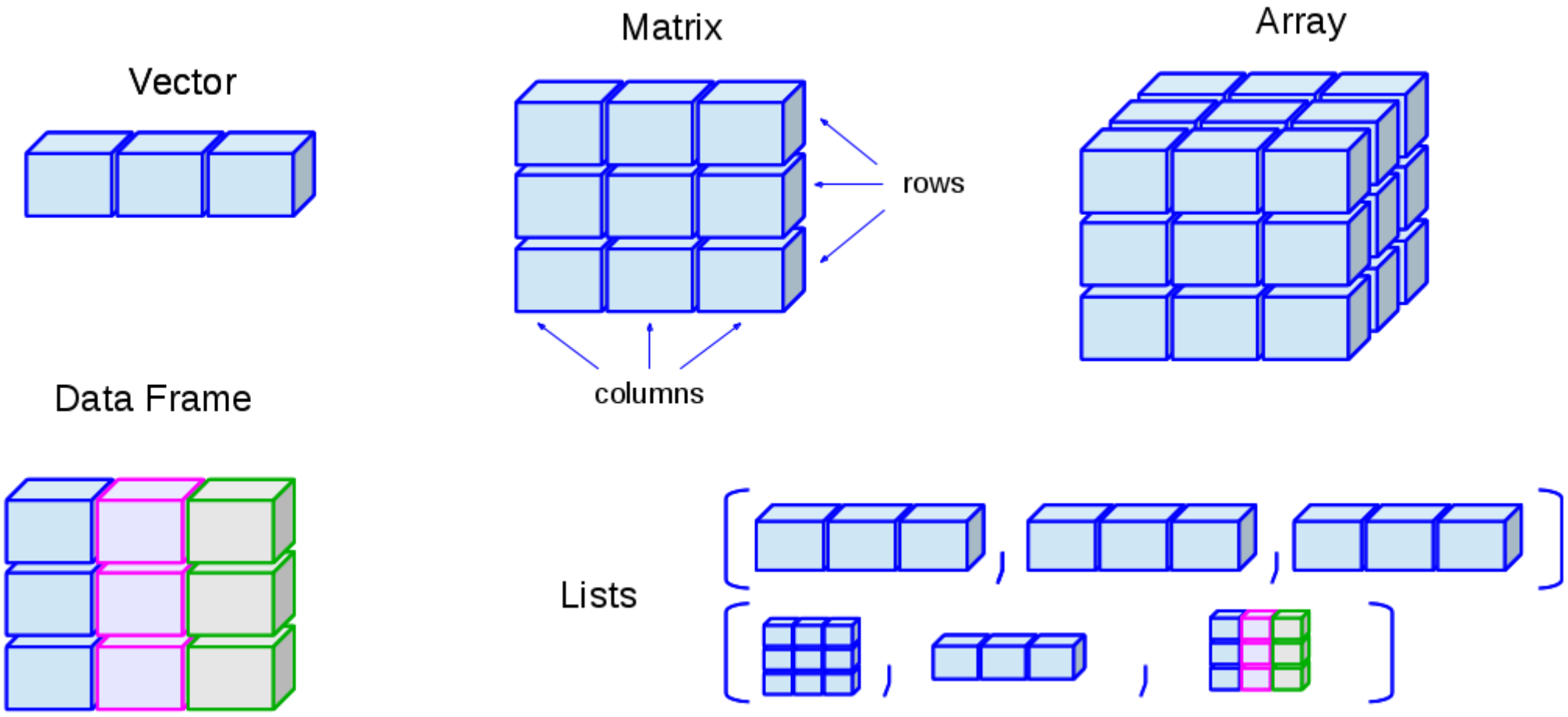


Image Credit: venus.ifca.unican.es

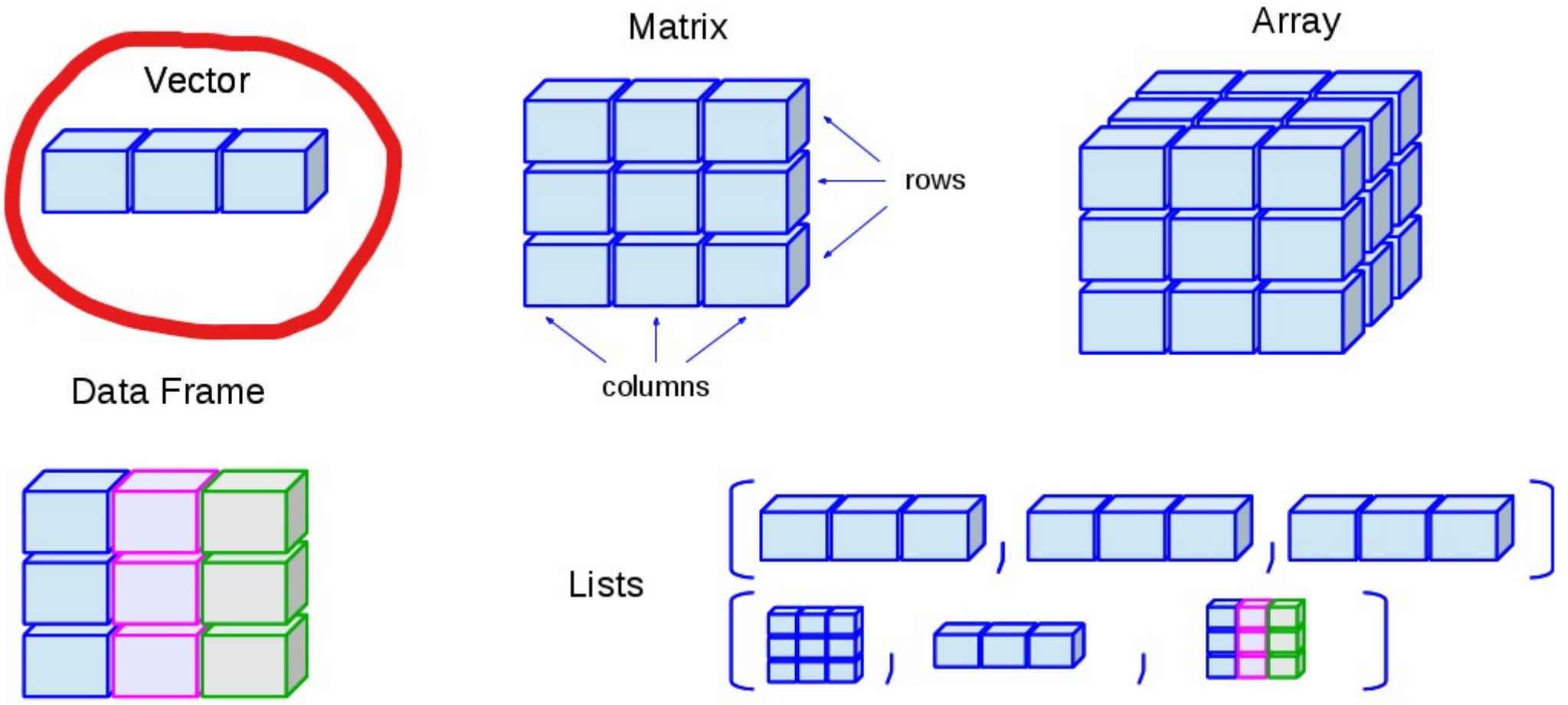


Image Credit: venus.ifca.unican.es

Recap

Write an R code to create the following vector?

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

- Method 1

```
c(1, 2, 3, 4, 5, 5, 4, 3, 2, 1)
```

- Method 2

```
c(1:5, 5:1)
```

02:00

Recap

- Name elements

```
a <- c(1:5, 5:1)
a
```

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

```
names(a) <- c("a1", "a2", "a3", "a4", "a5", "b1", "b2", "b3", "b4"
a
```

```
a1 a2 a3 a4 a5 b1 b2 b3 b4 b5
 1  2  3  4  5  5  4  3  2  1
```

Operations between vectors

```
a
```

```
a1 a2 a3 a4 a5 b1 b2 b3 b4 b5
 1  2  3  4  5  5  4  3  2  1
```

```
a * c(10, 100)
```

```
 a1  a2  a3  a4  a5  b1  b2  b3  b4  b5
 10 200  30 400  50 500  40 300  20 100
```

Vectors: subsetting

Select some particular elements (i.e., a subset) from a vector.

Vectors: Subsetting

```
myvec <- 1:20; myvec
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
myvec[1]
```

```
[1] 1
```

```
myvec[5:10]
```

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

Vectors: Subsetting (cont.)

```
myvec[-1]
```

```
 [1]  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
myvec[myvec > 3]
```

```
 [1]  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

vector subsetting - special cases

```
myvec[0]
```

```
integer(0)
```

```
myvec[]
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
myvec
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

- Extract elements present in vector `a` from `myvec` (cont.).

```
a; myvec
```

```
a1 a2 a3 a4 a5 b1 b2 b3 b4 b5
 1  2  3  4  5  5  4  3  2  1

 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
myvec %in% a
```

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

```
myvec[myvec %in% a]
```

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

Vectors: Subsetting (cont.)

```
 myvec
```

```
 [1]  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20
```

```
b <- 100:105
b
```

```
[1] 100 101 102 103 104 105
```

```
myvec[myvec %in% b]
```

```
integer(0)
```

Your turn

1. Generate a sequence using the code `seq(from=1, to=10, by=1)`.

2. What other ways can you generate the same sequence?

3. Using the function `rep` , create the below sequence 1, 2, 3, 4, 1, 2, 3, 4, 1, 2, 3, 4

03:00

Changing values of a vector

```
covid <- c(100, 30, 40, 50, -1, 100)
covid
```

```
[1] 100  30  40  50  -1 100
```

```
covid[1] <- 50000
covid
```

```
[1] 50000    30    40    50    -1   100
```

Changing values of a vector (cont.)

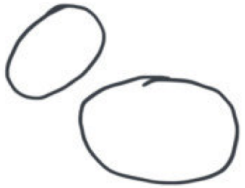
```
covid[covid < 0] <- 0
covid
```

```
[1] 50000    30    40    50     0   100
```

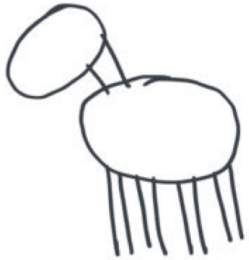
```
covid[c(1, 2)] <- c(1000, 10000)
covid
```

```
[1]  1000 10000    40    50     0   100
```

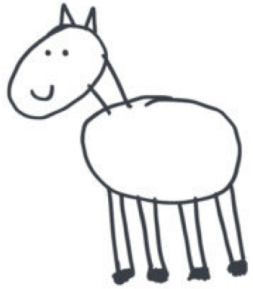
HOW TO:
DRAW A HORSE



① DRAW 2 CIRCLES

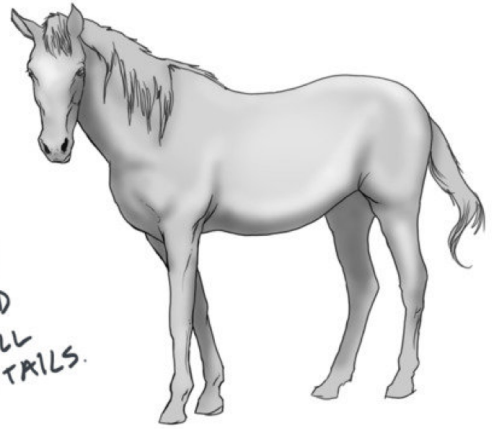


② DRAW THE LEGS



③ DRAW THE FACE

④ DRAW THE HAIR

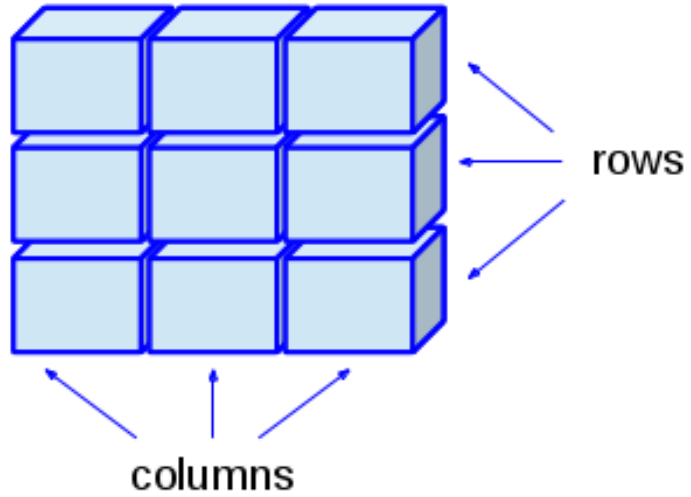


⑤ ADD SMALL DETAILS.

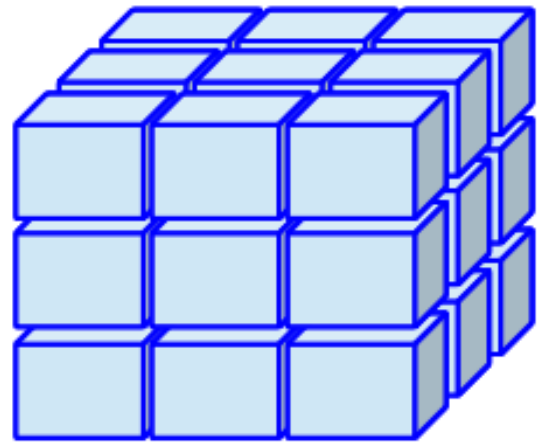
2. Matrices

Vector

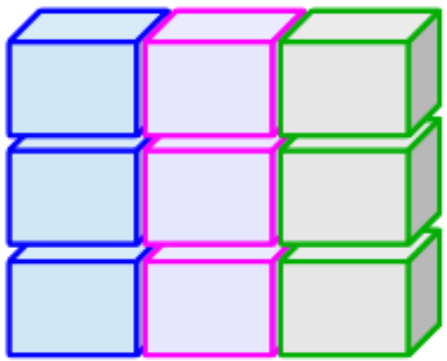
Matrix



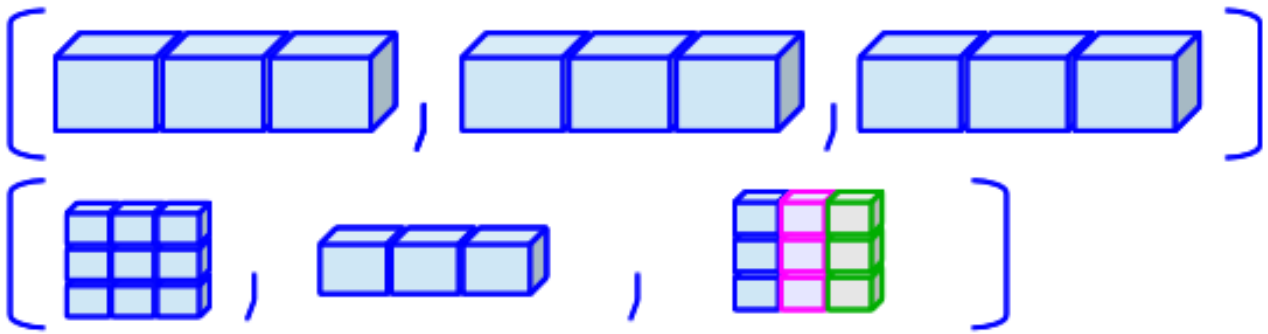
Array



Data Frame



Lists



Matrix is a 2-dimentional and a homogeneous data structure

Syntax to create a matrix

```
matrix_name <- matrix(vector_of_elements,
                      nrow=number_of_rows,
                      ncol=number_of_columns,
                      byrow=logical_value, # If byrow=TRUE, then t
                      dimnames=list(rnames, cnames)) # To assign r
```

Example

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

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

cont.

```
matrix(1:6, nrow=2)
```

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

```
matrix(1:6, ncol=3)
```

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

Matrix fill by rows/ columns

```
values <- c(10, 20, 30, 40)
matrix1 <- matrix(values, nrow=2) # Matrix filled by columns (defa
matrix1
```

```
     [,1] [,2]
[1,]   10   30
[2,]   20   40
```

```
matrix2 <- matrix(values, nrow=2, byrow=TRUE) # Matrix filled by r
matrix2
```

```
     [,1] [,2]
[1,]   10   20
[2,]   30   40
```

Matrix fill by rows/ columns (cont.)

- byrow=TRUE: matrix is filled in by row
- byrow=FALSE: matrix is filled in by column
- Default is by column

Naming matrix rows and columns

```
rnames <- c("R1", "R2")
cnames <- c("C1", "C2")
matrix_with_names <- matrix(values, nrow=2, dimnames=list(rnames,
matrix_with_names
```

```
   C1 C2
R1 10 30
R2 20 40
```

`class` function on vectors vs matrices

with vectors

```
vec <- 1:10
class(vec)
```

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

with matrices

```
mat <- matrix(1:10, ncol=5)
class(mat)
```

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

Matrix: subsetting

Matrix: subsetting

```
matrix1
```

```
     [,1] [,2]
[1,]   10   30
[2,]   20   40
```

`matraix_name[i, ]` gives the ith row of a matrix

```
matrix1[1, ]
```

```
[1] 10 30
```

`matraix_name[, j]` gives the jth column of a matrix

```
matrix1[, 2]
```

```
[1] 30 40
```

Matrix: subsetting (cont.)

`matraix_name[i, j]` gives the ith row and jth column element

```
matrix1
```

```
     [,1] [,2]
[1,]   10   30
[2,]   20   40
```

```
matrix1[1, 2]
```

```
[1] 30
```

```
matrix1[1, c(1, 2)]
```

```
[1] 10 30
```

Beware!

```
amat <- matrix(10:90, 3, 3); am
```

```
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
[3,]   12   15   18
```

```
bmat <- amat[1:2,]; bmat
```

```
     [,1] [,2] [,3]
[1,]   10   13   16
[2,]   11   14   17
```

```
class(bmat)
```

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

```
cmat <- amat[1,]; cmat
```

```
[1] 10 13 16
```

```
class(cmat)
```

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

```
dmat <- amat[, 1]; dmat
```

```
[1] 10 11 12
```

```
class(dmat)
```

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

drop = FALSE

```
cmat <- amat[1,]; cmat
```

```
[1] 10 13 16
```

```
class(cmat)
```

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

```
cmat <- amat[1, , drop = FALSE]
```

```
     [,1] [,2] [,3]
[1,]   10   13   16
```

```
class(cmat)
```

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

Diagonal elements

```
a <- matrix(1:9, ncol=3)
a
```

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

```
diag(a)
```

```
[1] 1 5 9
```

Replacing elements in a matrix

```
a <- matrix(1:9, ncol=3); a
```

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

```
a[1, 1] <- 100; a
```

```
     [,1] [,2] [,3]
[1,]  100    4    7
[2,]    2    5    8
[3,]    3    6    9
```

```
diag(a) <- 0
a
```

```
     [,1] [,2] [,3]
[1,]    0    4    7
[2,]    2    0    8
[3,]    3    6    0
```

cbind and rbind

Matrices can be created by column-binding and row-binding with `cbind()` and `rbind()`

```
x <- 1:3
y <- c(10, 100, 1000)
cbind(x, y) # binds matrices horizontally
```

```
     x    y
[1,] 1   10
[2,] 2  100
[3,] 3 1000
```

```
rbind(x, y) #binds matrices vertically
```

```r
rbind(x, y) #binds matrices vertically
```

```
  [,1] [,2] [,3]
x    1    2    3
y   10  100 1000
```

cbind and rbind (cont.)

```
a <- matrix(1:3, ncol=3); a
```

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

```
b <- matrix(c(10, 100, 1000),
            ncol=3)
b
```

```
     [,1] [,2] [,3]
[1,]   10  100 1000
```

```
cbind(a, b) # binds matrices ho
```

```
     [,1] [,2] [,3] [,4] [,5] [,6]
[1,]    1    2    3   10  100 1000
```

```
rbind(a, b) #binds matrices ver
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]   10  100 1000
```

Matrix operations

Matrix operations

Transpose

```
t(x)
```

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

Matrix multiplication

```
y <- matrix(seq(10, 60, by=10), nrow=3)
z <- x %*% y
z
```

```
     [,1] [,2]
[1,]  140  320
```

Matrix operations (cont.)

- matrix inverse is solve()

Find x in: m*x=n

```
solve(m, n)
```

Your turn

Mr. Perera who lives in Soratha Mawatha - Wijerama wants to sell his house. He wants to decide a price for his house to list it in the market. He believes that the size of the house is one likely determinant of price. He asked from 10 homes in the neighbourhood, "what price should you ask for your home?" and the house size (in square feet). The collected data are shown below:

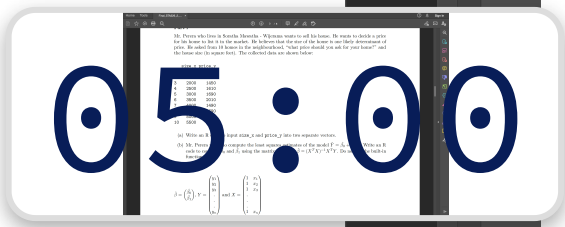
	size_x	price_y
1	1000	810
2	1500	1210
3	2000	1450
4	2500	1610
5	3000	1690
6	3500	2010
7	4000	1490
8	4500	1690
9	5000	1890
10	5500	2410

(a) Write an R code to input `size_x` and `price_y` into two separate vectors.

(b) Mr. Perera wants to compute the least squares estimates of the model $\hat{Y} = \hat{\beta}_0 + \hat{\beta}_1 X$. Write an R code to compute $\hat{\beta}_0$ and $\hat{\beta}_1$ using the matrix operation $\hat{\beta} = (X^T X)^{-1} X^T Y$. Do not use the built-in function `lm`.

Where,

$$\hat{\beta} = \begin{pmatrix} \hat{\beta}_0 \\ \hat{\beta}_1 \end{pmatrix}, Y = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ . \\ . \\ . \\ y_n \end{pmatrix} \text{ and } X = \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ 1 & x_3 \\ . & \\ . & \\ . & \\ 1 & x_n \end{pmatrix}$$



$$\hat{\beta} = (X^T X)^{-1} X^T Y$$

The SLR Model in Scalar Form

$$Y_i = \beta_0 + \beta_1 X_i + \epsilon_i \quad \text{where} \quad \epsilon_i \sim^{iid} N(0, \sigma^2)$$

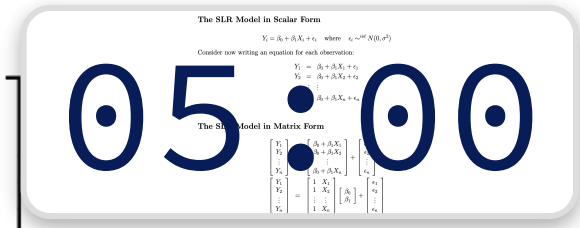
Consider now writing an equation for each observation:

$$\begin{aligned} Y_1 &= \beta_0 + \beta_1 X_1 + \epsilon_1 \\ Y_2 &= \beta_0 + \beta_1 X_2 + \epsilon_2 \\ &\vdots \quad \vdots \quad \vdots \\ Y_n &= \beta_0 + \beta_1 X_n + \epsilon_n \end{aligned}$$

The SLR Model in Matrix Form

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} \beta_0 + \beta_1 X_1 \\ \beta_0 + \beta_1 X_2 \\ \vdots \\ \beta_0 + \beta_1 X_n \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

$$\begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} = \begin{bmatrix} 1 & X_1 \\ 1 & X_2 \\ \vdots & \vdots \\ 1 & X_n \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix} + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{bmatrix}$$



Logical operators with matrices

```
a <- matrix(c(1:12), nrow = 3,
a > 10
```

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

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

```
a > b # Error
```

%in% operator

```
b %in% a
```

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

```
a %in% b
```

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

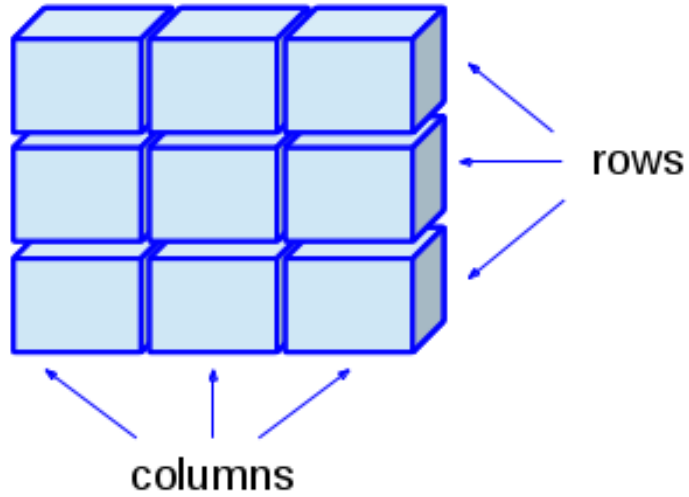
```
class(a %in% b)
```

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

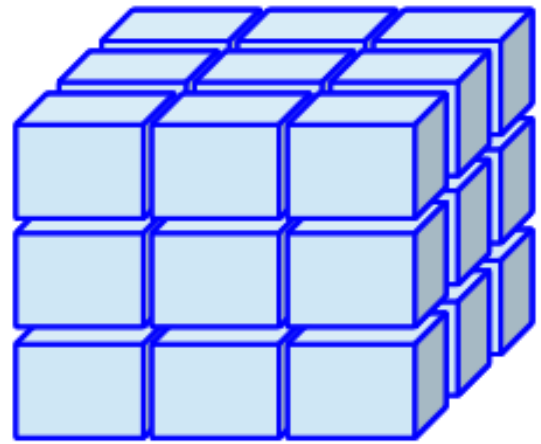
3. Arrays

Vector

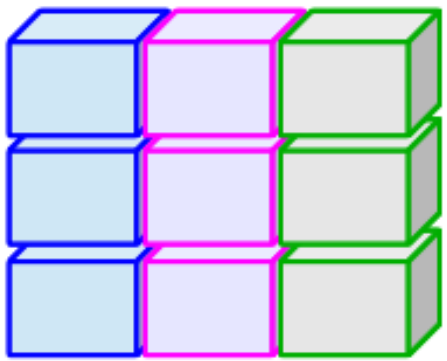
Matrix



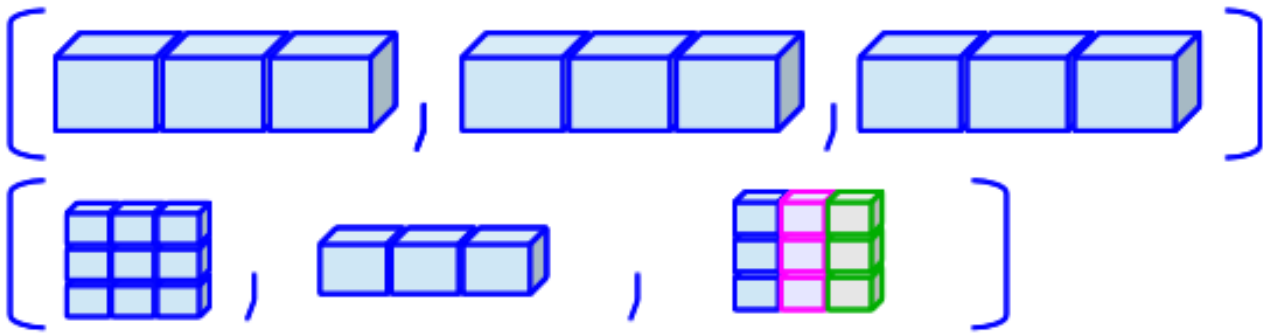
Array



Data Frame



Lists



Array

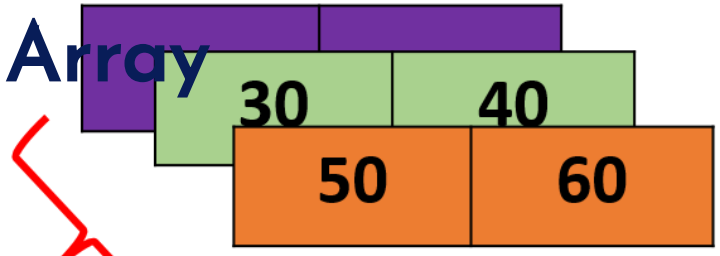
- data structures for storing **higher** dimensional data.
- a **homogeneous** data structure.
- a special case of the array is the matrix.

```
array(vector, dimensions, dimnames) #dimnames-as a list
```

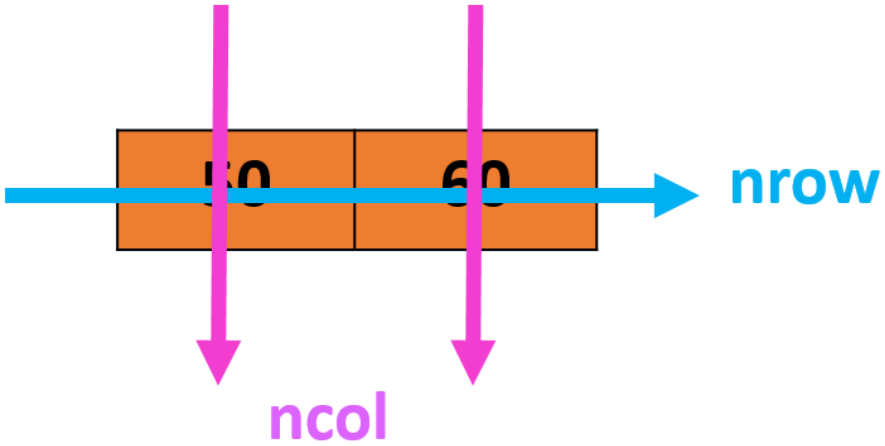
```
a <-  array(c(10, 20, 30, 40, 50, 60), c(1, 2, 3))
```

```r
a <-  array(c(10, 20, 30, 40, 50, 60), c(1, 2, 3))
a
```

```
, , 1

     [,1] [,2]
[1,]   10   20

, , 2

     [,1] [,2]
[1,]   30   40

, , 3

     [,1] [,2]
[1,]   50   60
```



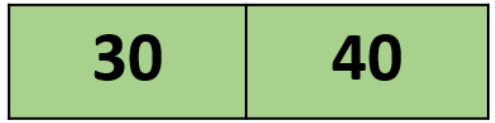
ndim = 3

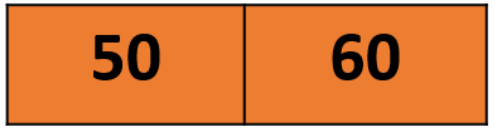



```
a <-  array(c(10, 20, 30, 40, 50, 60), c(1, 2, 3))
a
```

nrow ncol ndim

```
, , 1

     [,1] [,2]
[1,]   10   20

, , 2

     [,1] [,2]
[1,]   30   40

, , 3

     [,1] [,2]
[1,]   50   60
```





Subsetting arrays

```
a
```

```
, , 1

     [,1] [,2]
[1,]   10   20

, , 2

     [,1] [,2]
[1,]   30   40

, , 3

     [,1] [,2]
[1,]   50   60
```

```
a[, , 1] # Extract first entry
```

```
[1] 10 20
```

Subsetting arrays (cont.)

```
a
```

```
, , 1

     [,1] [,2]
[1,]   10   20

, , 2

     [,1] [,2]
[1,]   30   40

, , 3

     [,1] [,2]
[1,]   50   60
```

```
a[1, ,] # All rows in each entr
```

```
     [,1] [,2] [,3]
[1,]   10   30   50
[2,]   20   40   60
```

Your turn

1. Create the following matrix using the `array` function

```
matrix(1:20, ncol=5)
```

```
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
```

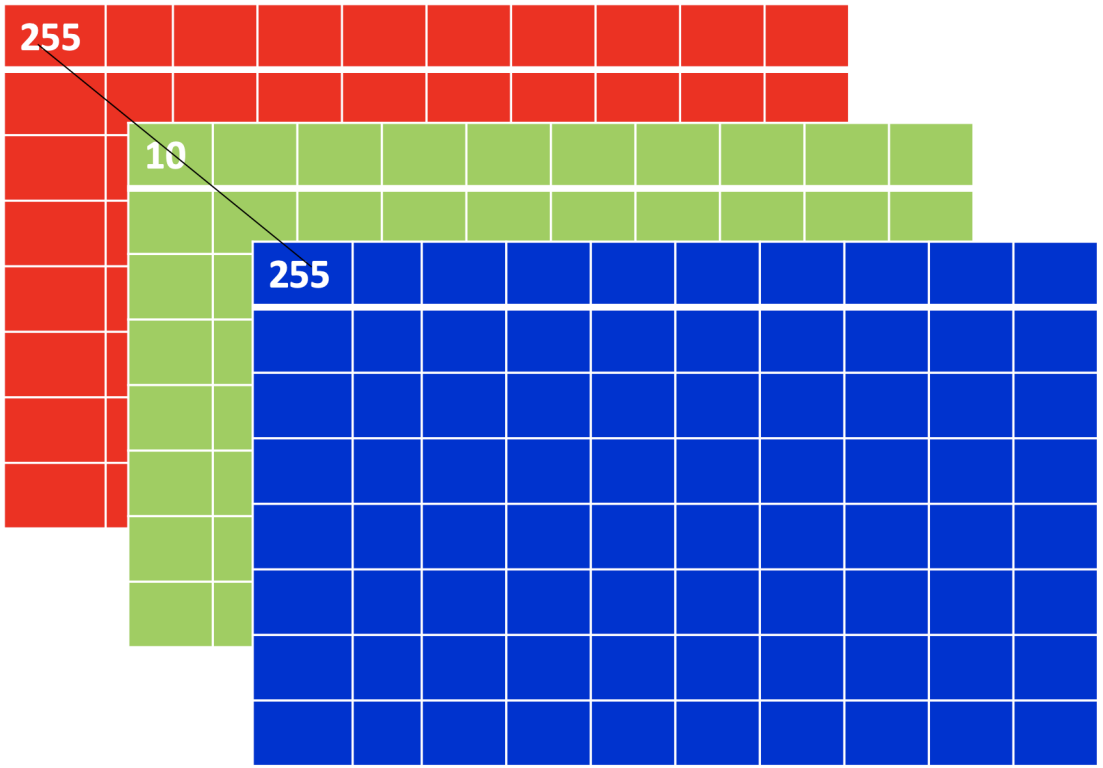
02:00

Array with dimnames

```
dim1 <- c("A1", "A2"); dim2 <- c("B1", "B2", "B3"); dim3 <- c("c1"
z <- array(1:24, c(2, 3, 4), dimnames = list(dim1, dim2, dim3))
z
```

```
, , c1

   B1 B2 B3
A1  1  3  5
A2  2  4  6

, , c2

   B1 B2 B3
A1  7  9 11
A2  8 10 12
```

```
, , c1

   B1 B2 B3
A1  1  3  5
A2  2  4  6

, , c2

   B1 B2 B3
A1  7  9 11
A2  8 10 12

, , c3

   B1 B2 B3
A1 13 15 17
A2 14 16 18

, , c4
```

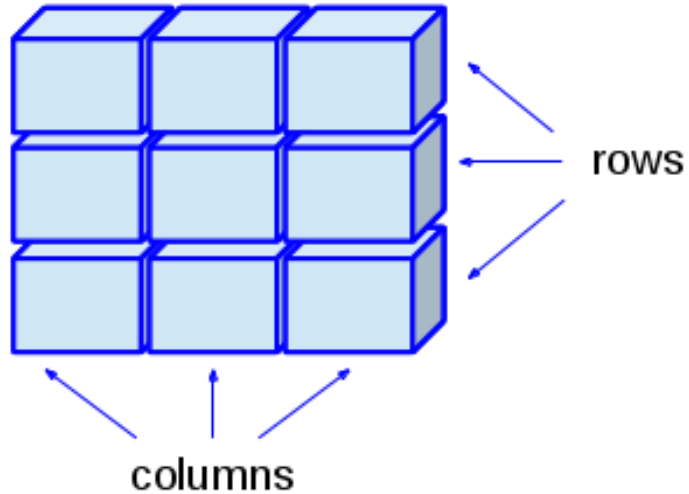


Pixel of an RGB image are formed from the corresponding pixel of the Red, Green and Blue components.

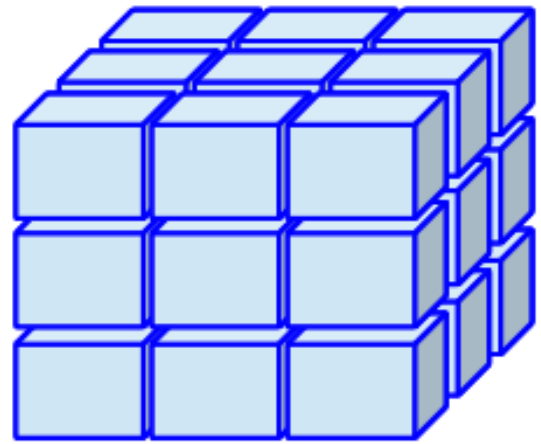
4. Data Frames

Vector

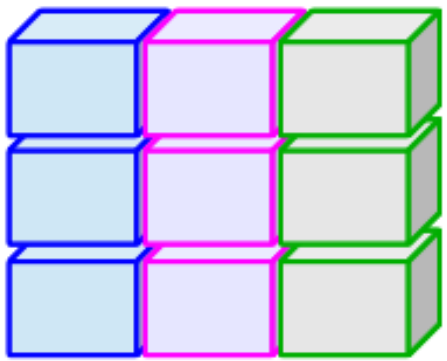
Matrix



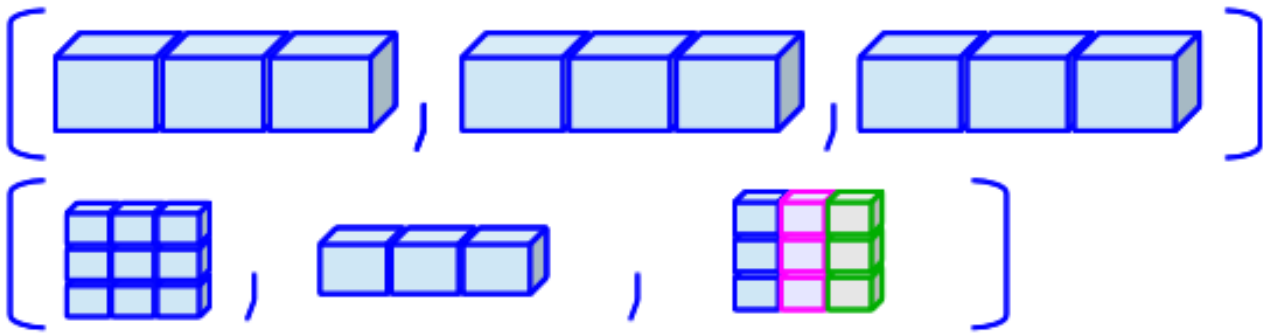
Array



Data Frame



Lists



Data frames

- Rectangular arrangement of data with rows corresponding to observational units and columns corresponding to variables.
- More general than a matrix in that different columns can contain different modes of data.
- It’s similar to the datasets you’d typically see in SPSS and MINITAB.
- Data frames are the most common data structure you’ll deal with in R.

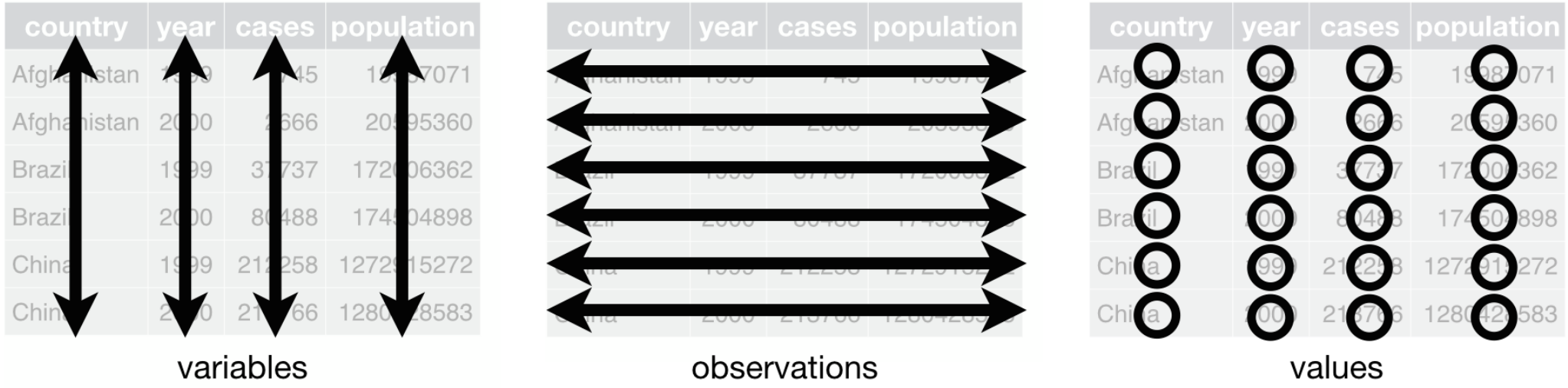


Image Credit: Hadley Wickham

Create a data frame

Syntax

```
name_of_the_dataframe <- data.frame(
                          var1_name=vector of values of the first
                          var2_names=vector of values of the secon
```

Example

```
corona <- data.frame(ID=c("C001", "C002", "C003", "C004"),
                     Location=c("Beijing", "Wuhan", "Shanghai", "B
                     Test_Results=c(FALSE, TRUE, FALSE, FALSE))
```

```
corona
```

```
    ID Location Test_Results
1 C001  Beijing         FALSE
2 C002    Wuhan          TRUE
3 C003 Shanghai         FALSE
4 C004  Beijing         FALSE
```

To check if it is a dataframe

```
is.data.frame(corona)
```

```
[1] TRUE
```

Some useful functions with dataframes

```
colnames(corona)
```

```
[1] "ID"           "Location"     "Test_Results"
```

```
length(corona)
```

```
[1] 3
```

```
dim(corona)
```

```
[1] 4 3
```

```
nrow(corona)
```

```
[1] 4
```

Some useful functions with dataframes (cont.)

```
summary(corona)
```

```
      ID                Location          Test_Results   
 Length:4           Length:4           Mode :logical  
 Class :character   Class :character   FALSE:3  
 Mode  :character   Mode  :character   TRUE :1
```

```
str(corona)
```

```
'data.frame':    4 obs. of  3 variables:
 $ ID          : chr  "C001" "C002" "C003" "C004"
 $ Location    : chr  "Beijing" "Wuhan" "Shanghai" "Beijing"
 $ Test_Results: logi  FALSE TRUE FALSE FALSE
```

Convert a matrix to a dataframe

```
mat <- matrix(1:16, ncol=4)
mat
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    5    9   13
[2,]    2    6   10   14
[3,]    3    7   11   15
[4,]    4    8   12   16
```

```
mat_df <- as.data.frame(mat)
mat_df
```

```
  V1 V2 V3 V4
1  1  5  9 13
2  2  6 10 14
```

Subsetting data frames

Select rows

```
head(mat_df) # default it shows
```

```
  V1 V2 V3 V4
1  1  5  9 13
2  2  6 10 14
3  3  7 11 15
4  4  8 12 16
```

```
head(mat_df, 3) # To extract or
```

```
  V1 V2 V3 V4
1  1  5  9 13
```

```
tail(mat_df, 2)
```

```
  V1 V2 V3 V4
3  3  7 11 15
4  4  8 12 16
```

Subsetting data frames

```
mat_df
```

```
  V1 V2 V3 V4
1  1  5  9 13
2  2  6 10 14
3  3  7 11 15
4  4  8 12 16
```

To select some specific rows

```
mat_df[4, ]
```

```
  V1 V2 V3 V4
4  4  8 12 16
```

```
index <- c(1, 3)
mat_df[index, ]
```

```
  V1 V2 V3 V4
1  1  5  9 13
3  3  7 11 15
```

Subsetting data frames: select columns

```
mat_df
```

```
  V1 V2 V3 V4
1  1  5  9 13
2  2  6 10 14
3  3  7 11 15
4  4  8 12 16
```

Select column(s) by variable names

```
mat_df$V1 # Method 1
```

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

```
mat_df[, "V1"] # Method 2
```

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

Subsetting data frames: select columns

```
mat_df
```

```
  V1 V2 V3 V4
1  1  5  9 13
2  2  6 10 14
3  3  7 11 15
4  4  8 12 16
```

Select column(s) by index

```
mat_df[, 2]
```

```
[1] 5 6 7 8
```

Your turn

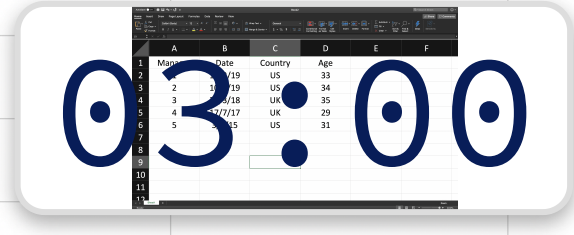
AutoSave OFF | Book2 | Search Sheet

Home | Insert | Draw | Page Layout | Formulas | Data | Review | View | Share | Comments

Paste | Cut | Copy | Format | Calibri (Body) | 12 | Wrap Text | Merge & Center | General | Conditional Formatting | Format as Table | Cell Styles | Insert | Delete | Format | AutoSum | Fill | Clear | Sort & Filter | Find & Select | Ideas | Sensitivity

C9

	A	B	C	D	E	F
1	Manager	Date	Country	Age		
2	1	10/2/19	US	33		
3	2	10/5/19	US	34		
4	3	21/3/18	UK	35		
5	4	17/7/17	UK	29		
6	5	3/3/15	US	31		
7						
8						
9						
10						
11						
12						



Sheet1 | Zoom

Ready | 400%

Built-in dataframes

Built-in dataframes

```
data(iris)
```

Use `help` function to find more about the iris dataset.

Go to file/function | Addins | Rprogramming

Console

Files | Plots | Packages | Help | Viewer

R: Edgar Anderson's Iris Data | Find in Topic

iris {datasets} R Documentation

Edgar Anderson's Iris Data

Description

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

Usage

```
iris
iris3
```

Format

`iris` is a data frame with 150 cases (rows) and 5 variables (columns) named `Sepal.Length`, `Sepal.Width`, `Petal.Length`, `Petal.Width`, and `Species`.

`iris3` gives the same data arranged as a 3-dimensional array of size 50 by 4 by 3, as represented by S-PLUS. The first dimension gives the case number within the species subsample, the second the measurements with names `Sepal L.`, `Sepal W.`, `Petal L.`, and `Petal W.`, and the third the species.

Source

Fisher, R. A. (1936) The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, **7**, Part II, 179–188.

The data were collected by Anderson, Edgar (1935). The irises of the Gaspe Peninsula, *Bulletin of the American Iris Society*, **59**, 2–5.

References

Becker, R. A., Chambers, J. M. and Wilks, A. R. (1988) *The New S Language*. Wadsworth & Brooks/Cole. (has `iris3` as `iris`.)

See Also

`matplot` some examples of which use `iris`.

Examples

```
dni3 <- dimnames(iris3)
ii <- data.frame(matrix(aperm(iris3, c(1,3,2)), ncol = 4,
                        dimnames = list(NULL, sub(" L.",".Length",
                                                  sub(" W.",".Width", dni3[[2]])))),
    Species = gl(3, 50, labels = sub("S", "s", sub("V", "v", dni3[[3]]))))
all.equal(ii, iris) # TRUE
```

[Package *datasets* version 3.6.0 Index]

iris dataset

```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

```
head(iris)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

- Introduced by Sir Ronald A. Fisher, 1936

Use the R dataset “iris” to answer the following questions:

1. How many rows and columns does iris have?
2. Select the first 4 rows.
3. Select the last 6 rows.
4. Select rows 10 to 20, with all columns in the iris dataset.
5. Select rows 10 to 20 with only the Species, Petal.Width and Petal.Length.

05:00

cont.

1. Create a single vector (a new object) called ‘width’ that is the Sepal.Width column of iris.

2. What are the column names and data types of the different columns in iris?

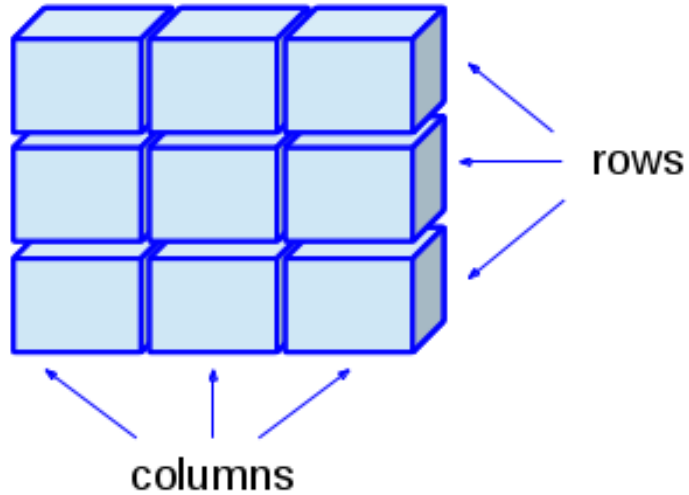
3. How many rows in the iris dataset have `Petal.Length` larger than 5 and `Sepal.Width` smaller than 3?

05:00

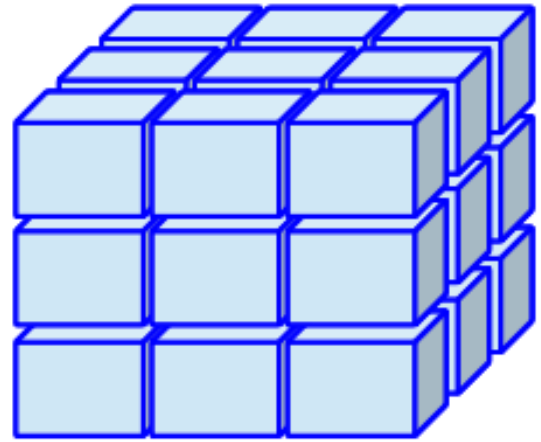
5. List

Vector

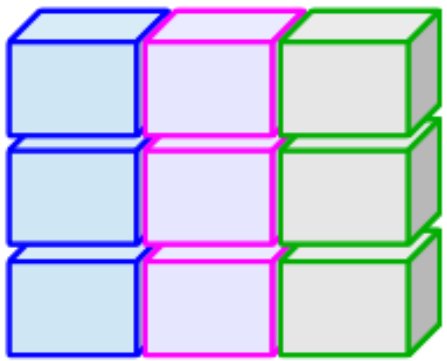
Matrix



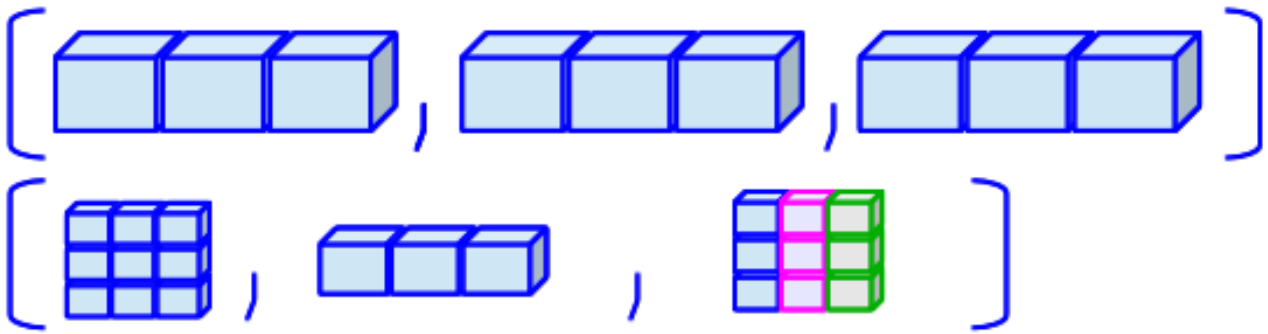
Array



Data Frame



Lists



List

- Lists are heterogeneous

Syntax

```
list_name <- list(entry1, entry2, entry3, ...)
```

Example

```
first_list <-list(1:3, matrix(1:6, nrow=2), iris)
```

List (str)

```
first_list <-list(1:3, matrix(1:6, nrow=2), iris)
first_list
```

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

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

[[3]]
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
3            4.7         3.2          1.3         0.2     setosa
```

Structure of a list

```
str(first_list)
```

```
List of 3
 $ : int [1:3] 1 2 3
 $ : int [1:2, 1:3] 1 2 3 4 5 6
 $ :'data.frame':    150 obs. of  5 variables:
  ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
  ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 .
  ..$ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
  ..$ Petal.Width : num [1:150] 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
```

Extract elements

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

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

```
first_list[[3]]$Species
```

```
 [1] setosa     setosa     setosa     setosa     setosa     setosa
 [7] setosa     setosa     setosa     setosa     setosa     setosa
[13] setosa     setosa     setosa     setosa     setosa     setosa
[19] setosa     setosa     setosa     setosa     setosa     setosa
[25] setosa     setosa     setosa     setosa     setosa     setosa
[31] setosa     setosa     setosa     setosa     setosa     setosa
[37] setosa     setosa     setosa     setosa     setosa     setosa
[43] setosa     setosa     setosa     setosa     setosa     setosa
[49] setosa     setosa     versicolor versicolor versicolor versicolor
```

Name entries in a list

```
first_list_with_names <-list(a=1:3, b=matrix(1:6, nrow=2), c=iris)
first_list_with_names
```

```
$a
[1] 1 2 3

$b
     [,1] [,2] [,3]
[1,]    1    3    5
[2,]    2    4    6

$c
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
1            5.1         3.5          1.4         0.2     setosa
2            4.9         3.0          1.4         0.2     setosa
```

Extract elements using names

```
str(first_list_with_names)
```

```
List of 3
 $ a: int [1:3] 1 2 3
 $ b: int [1:2, 1:3] 1 2 3 4 5 6
 $ c:'data.frame':    150 obs. of  5 variables:
  ..$ Sepal.Length: num [1:150] 5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
  ..$ Sepal.Width : num [1:150] 3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 .
  ..$ Petal.Length: num [1:150] 1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5
  ..$ Petal.Width : num [1:150] 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
```

```
first_list_with_names$a
```

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

```
first_list_with_names$c$Species
```

```
 [1] setosa     setosa     setosa     setosa     setosa     setosa
 [7] setosa     setosa     setosa     setosa     setosa     setosa
[13] setosa     setosa     setosa     setosa     setosa     setosa
[19] setosa     setosa     setosa     setosa     setosa     setosa
[25] setosa     setosa     setosa     setosa     setosa     setosa
[31] setosa     setosa     setosa     setosa     setosa     setosa
[37] setosa     setosa     setosa     setosa     setosa     setosa
[43] setosa     setosa     setosa     setosa     setosa     setosa
[49] setosa     setosa     versicolor versicolor versicolor versicolor
[55] versicolor versicolor versicolor versicolor versicolor versicolor
[61] versicolor versicolor versicolor versicolor versicolor versicolor
[67] versicolor versicolor versicolor versicolor versicolor versicolor
[73] versicolor versicolor versicolor versicolor versicolor versicolor
[79] versicolor versicolor versicolor versicolor versicolor versicolor
[85] versicolor versicolor versicolor versicolor versicolor versicolor
[91] versicolor versicolor versicolor versicolor versicolor versicolor
```

Your turn

```
c("Jan","Feb","Mar"); matrix(c(3,9,5,1,-2,8), nrow = 2); list("gre
```

```
[1] "Jan" "Feb" "Mar"

     [,1] [,2] [,3]
[1,]    3    5   -2
[2,]    9    1    8

[[1]]
[1] "green"

[[2]]
[1] 12.3
```

1. Create a list containing the above vector, matrix and the list.

2. Name the elements as `first`, `second` and `third`.

Thank you!

Slides available at: hellor.netlify.app