Eric Pitman Summer Workshop in Computational Science

2. Data Structures: Vectors and Data Frames

CENTER FOR **COMPUTATIONAL RESEARCH**

University at Buffalo
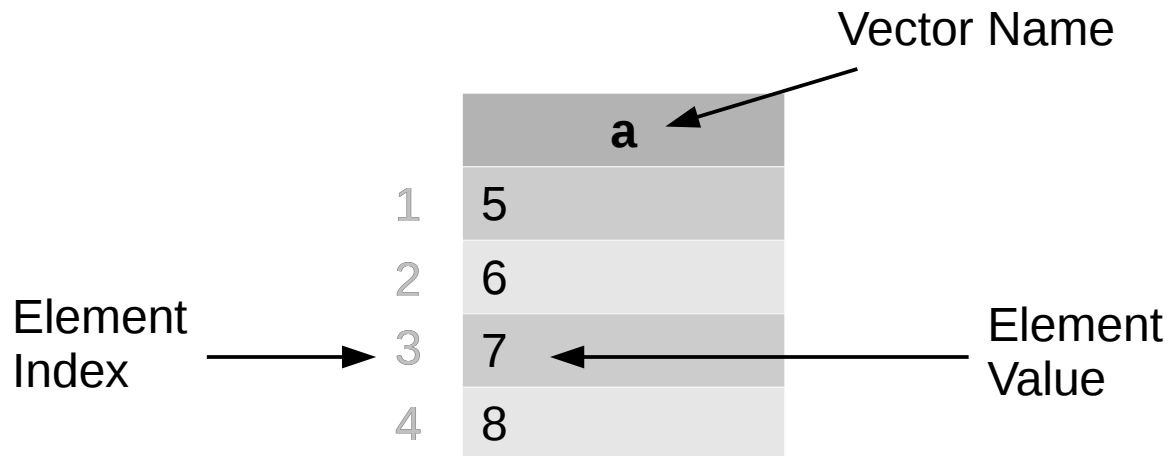*The State University of New York*

Data Objects in R

These objects, composed of multiple atomic data elements, are the bread and butter of R:

- Vectors
- Data Frames

Vector Data Object

A vector is a list of elements having the *same type*.

Vector Name

a

1   5

2   6

Element Index → 3   7 ← Element Value

4   8

Construct a Vector Data Object

	a
1	5
2	6
3	7
4	8

Use the c() function:

> a <- c(5,6,7,8)   # vector with 4 numeric values

> d <- c("red", "orange", "green") # character vector

Accessing Vector Data

d
1   "red"
2   "orange"
3   "green"

a
1   5
2   6
3   7
4   8

Access by index or range:

> d[1]   # retrieves "red"

> a[3]   # retrieves 7

> d[1:2] # retrieves "red", "orange"

Element numbering starts at 1 in R

Information about a Vector

	y
1	3
2	5
3	7
4	9

> y <- c(3,5,7,9) # vector with 4 numeric values

> length(y) # how many elements?

> class(y) # class of a vector object is the class
# of its elements

Information about a Vector

	y
1	3
2	5
3	7
4	9

> str(y)   # structure of the vector: number of
             # elements, type, and contents

num [1:4] 3 5 7 9

Type of elements

Number (and positions) of elements

contents

Some Operations on Vectors

- sum()   # Sum of all element values
- length() # Number of elements
- unique() # Generate vector of distinct values
- diff()   # Generate vector of first differences
- sort()   # Sort elements, omitting NAs
- order()   # Sort indices, with NAs last
- rev()   # Reverse the element order
- summary() # Information about object contents

Repercussions of NA

Any arithmetic operation on a structure containing an NA generates NA!

NA means "no value known"

> y = c(1, NA, 3, 2, NA)

> sum(y)

[1] NA

We must *remove NAs* to make calculations. How?

Finding NAs in a Data Structure

\> y = c(1, NA, 3, 2, NA)

\> summary(y)

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	NA's
1.0	1.5	2.0	2.0	2.5	3.0	2

Handling Missing Data

Remove NAs prior to calculation:

```
> y = c(1, NA, 3, 2, NA)    # [1, ?, 3, 2, ?]
sum(y, na.rm=TRUE)          # removes NAs, then sums
[1] 6                       # sum of 1 + 3 + 2
```

rm = "remove"

Data Frames

- A data frame is a structure consisting of columns of *various modes* (numeric, character, etc).

- Its rows and columns can be named.

- Data frames are handy containers for experimental data.

Data Frame Example

Data frames are handy containers for data that describe experimental subjects.

Student population data:

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Constructing a Data Frame

1. Construct the vectors that hold column data:

height   = c(68, 75, 60)   # inches

age   = c(16, 17, 16)   # years

handed = c("L", "R", "R")   # dominant hand: R=right, L=left

2. Construct the data frame by associating the columns:

data = data.frame(Height=height,

                          Age=age,

                          Hand=handed)

Name of the column!

Data Frame

Organized in rows and columns:

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Rows

Columns (formed from vectors)

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

First index is row, second index is column:

> data[1,1] # retrieves subject A's Height

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

```
> data[1, ]   # retrieves all subject A data
    Height  Weight  Age  Hand
A       68     120   16     L
```

Comma is a placeholder in the [row, column] notation

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Comma is a placeholder in the [row, column] notation

> data[,1]   # retrieves all Height data

[1] 68 75 60

Accessing by Index

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

> data[1,]   # retrieves all subject A data

```
    Height Weight  Age Hand
A       68    120   16    L
```

> data[,1]   # retrieves all Height data

[1] 68 75 60

Comma is a placeholder in the [row, column] notation

Try it:
Accessing by Index

> source("data-frame-simple-example.R")

> data[2,3]   # retrieves subject B's Age

> data[2,]   # retrieves all subject B data

> data[,3]   # retrieves all Age data

Accessing by Name

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

First is row, second is column:

> data["A","Height"] # retrieves subject A's Height

　　　　　　　　　　　　　# Notice the quotes!

Accessing by Name

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

> data["A",]   # retrieves all subject A data.
　　　　　　　　# Notice the comma!

Accessing by Name

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

To fetch Height column:

> data[,"Height"]   # Notice the comma

Accessing by Name

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Another way to fetch Height column:

> data$Height

Try it:
Accessing by Name

> source("data-frame-simple-example.R")

> data["B","Age"]   # retrieves B's Age

> data["B",]   # retrieves all B data

> data$Age   # retrieves all Age data

Conditional Access

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Subjects who are taller than 65 inches:

> data[data$Height > 65,] # subset of the data frame
(notice the comma!)

Conditional Access

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Heights over 65 inches:

> data$Height[data$Height > 65]   # subset of a *column*
                                  # of the data frame

Conditional Access

	Height	Weight	Age	Hand
A	68	120	16	L
B	(75)	160	17	R
C	60	118	16	R

Heights equal to 75 inches:

```
> data$Height[data$Height == 75]   # subset of a column
                                   # of the data frame
```

Try it:
Conditional Access

> source("data-frame-simple-example.R")

subset of the data frame having age<17 years:

> data[data$Age < 17,]

subset of a *column* of data frame, age<17 years:

> data$Age[data$Age < 17]

Conditional Access 2

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

Heights over 65 inches:

> data[which(data$Height > 65), "Height"]

> subset(data, Height>65, select="Height")

handy if you have NAs in the dataframe.

Conditional Access 2: which()

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

> which(data$Height > 65)

1 2

which() returns the indices for which the conditional is true!

Conditional Access 2: subset()

	Height	Weight	Age	Hand
A	68	120	16	L
B	75	160	17	R
C	60	118	16	R

> subset(data, Height>65, select="Height")

subset() arguments are:
dataset to subset,
subsetting condition to apply,
columns to return

Data Frame Information

```
str(data)        # structure
dim(data)        # dimensions
is.data.frame(data)     # returns a logical value

View(data)       # open View window of data
head(data)       # beginning of the data frame
tail(data)       # end of the data frame

names(data)        # names of the columns
rownames(data)     # names of the rows
colnames(data)     # names of the columns
```

```
> class(data)
[1] "data.frame"
```

Student Dataset Example

Let's create our own dataset and put it in an R data frame:

- `FirstInitial`

- `LastInitial`

- `School`

- `Height`

- `HtUnit`

- `Age`

- `Handed`

- `Gender`

Student Dataset Example

Now we can write some R to select subsets of our data. Examples:

- How many students younger than 17?

- List heights of students at Williamsville North

- Genders of left-handers?

Interlude

Complete vector/data frame exercises.

Open in the RStudio source editor:

<workshop>/exercises/exercises-vectors-matrices-dataframes.R