Instructor: Nara Yoon, Ph.D. (yoonnx@jmu.edu)

Spring 2024 | Thursdays 4-7pm, Hart 3008

Data Analytics for Managerial Leadership

WK 8: Data structure I

PLAN FOR TODAY

Overview of core concepts

- Go deeper dive: Vectors
- Data types
- Data frame
- Combining row/column

CASE IN THE NEWS

Marr, B. (Oct 18, 2019). What's the difference between structured, semi-structured and unstructured data? Forbes

READINGS

Long, J.D., & Teeter, P. (2019). R Cookbook (2nd edition).

- Ch 5. Data structures

Matloff, N. (2011). The art of R programming a tour of statistical software design. No Starch Press.

- Ch 5: Data frames (5.1 through 5.4)

Vector basics

VECTOR ARITHMETIC

```
score_class1 <- c(25,30,35,40,45) # exam score in class 1
score_class2 <- c(15,20,25,30,35) # exam score in class 2
```

```
score_class1 + score_class2
```

→ sum of score in two classes

```
## [1] 40 50 60 70 80
```

```
(score_class1 + score_class2)/2
```

→ mean of score in two classes

```
## [1] 20 25 30 35 40
```

OPERATORS

Operator	When to use it
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
==	Equal to
!=	Not equal to
X & Y	X and Y
X | Y	X or Y
!X	Not X

Operator	When to use it
is.na(x)	To see if x is NA
!is.na(x)	To see if x is not NA
X %in% y	To see if x is in y

PAIRWISE COMPARISONS (WITH OPERATORS)

```
score_class1
```

```
## [1] 25 30 35 40 45
```

```
score_class2
```

```
## [1] 15 20 25 30 35
```

```
score_class1 > score_class2
```

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

LOGICAL OPERATORS

```
gender <- c("female","male","female","male")
gender == "male"
```

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

```
number <- c(10,20,30,40)
number > 25
```

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

COMPOUND LOGICAL STATEMENTS

Use

& (and),

| (or),

! (opposite) operators

to select more specific groups in the data

```
numbers <- c(10,20,30,40)
greater_than_20 <- numbers > 20
greater_than_20
```

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

```
!greater_than_20
```

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

Vector functions

VECTOR FUNCTIONS

sum()	summation of all elements
mean()	mean
median()	median
min(), max()	minimum, maximum value
sd(), var()	standard deviation, variance
length()	the number of elements
sort()	return observations in sorted order
order()	reorder the index
unique()	lists unique elements
summary()	summary statistics

VECTOR FUNCTIONS (EXAMPLE)

```
score_class1 <- c(25,30,35,40,45) # exam score in class 1
```

```
mean(score_class1)
```

← mean()

```
## [1] 35
```

```
median(score_class1)
```

← median()

```
## [1] 35
```

VECTOR FUNCTIONS (EXAMPLE)

```r
score_class1 <- c(25,30,35,40,45) # exam score in class 1
```

```r
sort(score_class1)
```

← sort()

```
## [1] 25 30 35 40 45
```

```r
max(score_class1)
```

← max()

```
## [1] 45
```

```r
min(score_class1)
```

← min()

```
## [1] 25
```

Data types

STRUCTURED DATA

Students table

Unique ID	Student ID	First name	Last name
110	110_1	...	...
111	111_1	...	...
112	112_1	...	...

School table

Unique ID	School ID	School name	Location
110	110_a		
111	111_b		
112	112_c		

Student-School table

Unique ID	Student ID	School ID
110	110_1	110_a
111	111_1	111_b
112	112_1	112_c

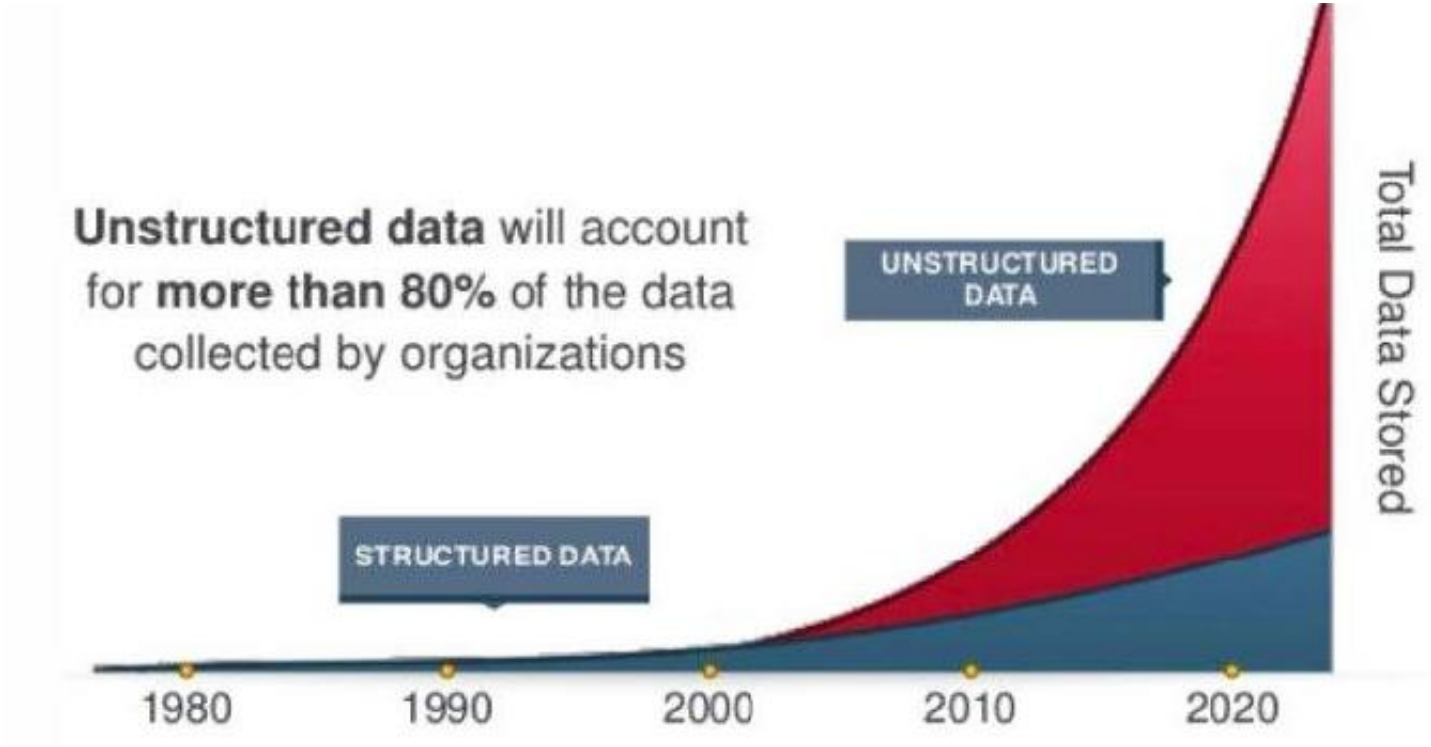
UNSTRUCTURED DATA

Do not have common identifiers across elements

Thus, difficult to identify structural relationships

For example: text data, image data ….

RISING TREND OF UNSTRUCTURED DATA



Source: Web

Data frame

DATA FRAME

Collection of vectors in rectangular form (rows & columns)

data.frame()

MAKING AS A DATA FRAME: as.data.frame()

as.data.frame() convert an object to data frame

```
example_list <- list(Name = c("Charlie","Cindy","David","Flower"), Age=c(10,15,20,24),
                     Gender= c("Male","Female","Male","Female"),
                     Bloodtype=c("AB","A","B","O"))
example_df <- as.data.frame(example_list)
```

ROWS & COLUMN IN A DATA FRAME: WHAT DO THEY REPRESENT?

Name	Age	Gender	Bloodtype
Charlie	10	Male	AB
Cindy	15	Female	A
David	20	Male	B
Flower	25	Female	O

COLUMN: CONSISTENCY IN TYPE/MODE OF DATA

Each column has the **same type of data**

Each column has the **same number of observations**

Name	Age	Gender	Bloodtype
Charlie	10	Male	AB
Cindy	15	Female	A
David	20	Male	B
Flower	25	Female	O

Using R Functions to explore data frame structure

SOME MORE BASIC R FUNCTIONS

- head()
- tail()
- str()
- nrow()
- ncol()
- name()
- class()

DATA FRAME STRUCTURE: EXAMPLE DATASET

```
this.data <- data.frame(x=c("Charlie", "Cindy", "David", "Flower"), y=c(10, 15, 20, 25),
                        z=c("Male", "Female", "Male", "Female"))
```

```
this.data
```

```
##         x  y      z
## 1 Charlie 10   Male
## 2   Cindy 15 Female
## 3   David 20   Male
## 4  Flower 25 Female
```

head() & tail()

head() – view the first few observations

tail() – view the last few observations

head()

How to show the first 2 observations?

```
head(this.data, 2)
```

```
##         x  y      z
## 1 Charlie 10   Male
## 2   Cindy 15 Female
```

tail()

How to show the last 2 observations?

```
tail(this.data, 2)
```

```
##        x  y      z
## 3  David 20   Male
## 4 Flower 25 Female
```

str()

str() – Show the structure of the data

```
str(this.data)  ## What does str() do?
```

```
## 'data.frame':    4 obs. of  3 variables:
##  $ x: chr  "Charlie" "Cindy" "David" "Flower"
##  $ y: num  10 15 20 25
##  $ z: chr  "Male" "Female" "Male" "Female"
```

nrow()

nrow() – number of rows

```
nrow(this.data)
```

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

ncol()

ncol() – number of columns

```
ncol(this.data)
```

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

names()

names() – show attribute names of the data

```
names(this.data)
```

```
## [1] "x" "y" "z"
```

class()

class() – show the class of the object

```
class(this.data$x)
```

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

```
class(this.data$y)
```

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

```
class(this.data$z)
```

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

Understanding data frames

CHECK DATASET DESCRIPTION (iris dataset)

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`.

view()

```
View(iris) # show as a spreadsheet
```

	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
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa

summary()

```
summary(iris)
```

```
##   Sepal.Length    Sepal.Width     Petal.Length    Petal.Width
##  Min.   :4.300   Min.   :2.000   Min.   :1.000   Min.   :0.100
##  1st Qu.:5.100   1st Qu.:2.800   1st Qu.:1.600   1st Qu.:0.300
##  Median :5.800   Median :3.000   Median :4.350   Median :1.300
##  Mean   :5.843   Mean   :3.057   Mean   :3.758   Mean   :1.199
##  3rd Qu.:6.400   3rd Qu.:3.300   3rd Qu.:5.100   3rd Qu.:1.800
##  Max.   :7.900   Max.   :4.400   Max.   :6.900   Max.   :2.500
##        Species
##  setosa    :50
##  versicolor:50
##  virginica :50
```

str()

```
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 ...
```

class(), head(), colnames()

```
class(iris) ## Iris dataset is data frame
```

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

```
head(iris)  ## First few observations in the dataset (default is 6 rows)
```

```
##   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
```

```
colnames(iris) ## There are 5 columns in iris dataset
```

```
## [1] "Sepal.Length" "Sepal.Width"  "Petal.Length" "Petal.Width"  "Species"
```

head()

```
head(iris)  ## First few observations in the dataset (default is 6 rows)
```

```
##   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
```

UNDERSTANDING DATA FRAMES: head()

```
head(iris[c("Sepal.Length","Petal.Length")])
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
## 4          4.6          1.5
## 5          5.0          1.4
## 6          5.4          1.7
```

UNDERSTANDING DATA FRAMES: head()

```
head(iris[,c(1,3)])
```

```
##   Sepal.Length Petal.Length
## 1          5.1          1.4
## 2          4.9          1.4
## 3          4.7          1.3
## 4          4.6          1.5
## 5          5.0          1.4
## 6          5.4          1.7
```

nrow() & ncol()

```
nrow(iris) ## iris dataset consists of 150 rows
```

```
## [1] 150
```

```
ncol(iris) ## iris dataset consists of 5 columns
```

```
## [1] 5
```

```
dim(iris)
```

```
## [1] 150   5
```

VARIABLE TYPE: **class()**

```
class(iris$Sepal.Length)
```

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

← Sepal.Length variable type is numeric

```
class(iris$Species)
```

```
## [1] "factor"
```

← Species variable is factor variable

summary(), levels()

```
summary(iris$Species)
```

```
##     setosa versicolor  virginica
##         50         50         50
```

```
levels(iris$Species)
```

```
## [1] "setosa"     "versicolor" "virginica"
```

SUBSETTING DATA

```
iris[iris$Species=="setosa" & iris$Sepal.Length>=5,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1           5.1         3.5          1.4         0.2  setosa
## 5           5.0         3.6          1.4         0.2  setosa
## 6           5.4         3.9          1.7         0.4  setosa
## 8           5.0         3.4          1.5         0.2  setosa
## 11          5.4         3.7          1.5         0.2  setosa
## 15          5.8         4.0          1.2         0.2  setosa
## 16          5.7         4.4          1.5         0.4  setosa
## 17          5.4         3.9          1.3         0.4  setosa
## 18          5.1         3.5          1.4         0.3  setosa
## 19          5.7         3.8          1.7         0.3  setosa
## 20          5.1         3.8          1.5         0.3  setosa
## 21          5.4         3.4          1.7         0.2  setosa
## 22          5.1         3.7          1.5         0.4  setosa
```

SORTING: ASCENDING

```
sorted_iris <- iris[order(iris$Sepal.Length),]
head(sorted_iris)
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 14          4.3         3.0          1.1         0.1  setosa
## 9           4.4         2.9          1.4         0.2  setosa
## 39          4.4         3.0          1.3         0.2  setosa
## 43          4.4         3.2          1.3         0.2  setosa
## 42          4.5         2.3          1.3         0.3  setosa
## 4           4.6         3.1          1.5         0.2  setosa
```

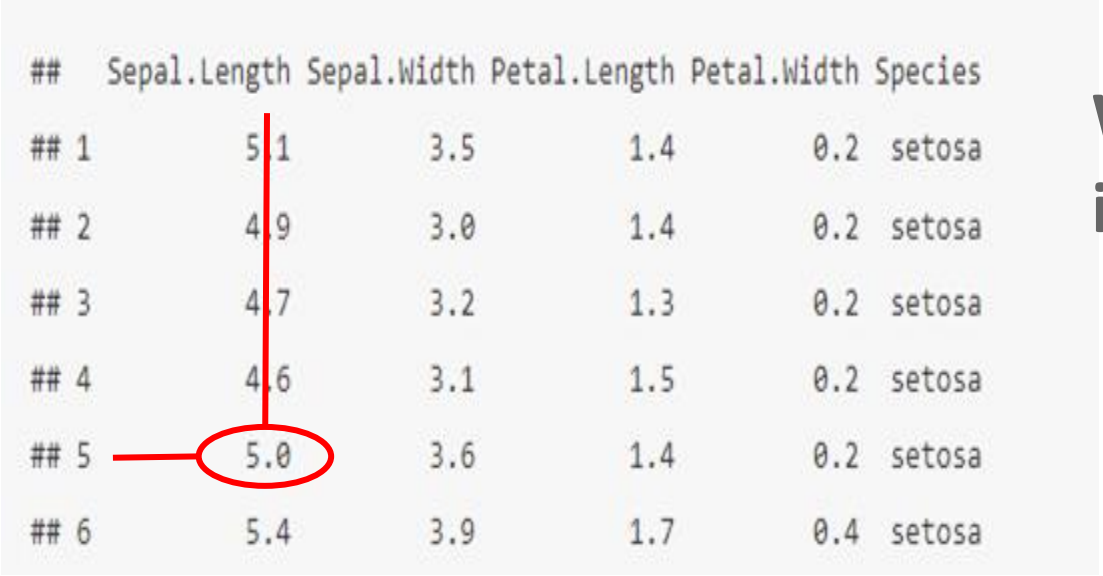
SORTING: DESCENDING

```
sorted_iris <- iris[order(-iris$Sepal.Length),]
head(sorted_iris)
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 132          7.9         3.8          6.4         2.0 virginica
## 118          7.7         3.8          6.7         2.2 virginica
## 119          7.7         2.6          6.9         2.3 virginica
## 123          7.7         2.8          6.7         2.0 virginica
## 136          7.7         3.0          6.1         2.3 virginica
## 106          7.6         3.0          6.6         2.1 virginica
```

Indexing data frame

INDEXING DATA FRAMES



```
##   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
```

What does iris[5,1] indicates?

RETURNING VALUES IN DATA FRAME: USING ROW & COLUMN INDICES

iris[5,1] # returns values in row 5 & column 1

iris[5,] # returns values in row 5

iris[, 1] # returns values in column 1

iris[5, 1:3] # returns values in row 5 & column from 1 to 3

MULTIPLE WAYS TO INDEXING:
ONE WAY TO DO IT – double bracket [[ ]]

```
iris[["Sepal.Length"]] ## "Sepal.Length" element
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
##  [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
##  [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
##  [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

MULTIPLE WAYS TO INDEXING: ANOTHER WAY TO DO IT – dollar sign $

```
iris$Sepal.Length      ## "Sepal.Length" element
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0 4.4 4.9 5.4 4.8 4.8 4.3 5.8 5.7 5.4 5.1
##  [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0 5.0 5.2 5.2 4.7 4.8 5.4 5.2 5.5 4.9 5.0
##  [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0 5.1 4.8 5.1 4.6 5.3 5.0 7.0 6.4 6.9 5.5
##  [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9 6.0 6.1 5.6 6.7 5.6 5.8 6.2 5.6 5.9 6.1
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7 5.5 5.5 5.8 6.0 5.4 6.0 6.7 6.3 5.6 5.5
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2 5.1 5.7 6.3 5.8 7.1 6.3 6.5 7.6 4.9 7.3
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4 6.5 7.7 7.7 6.0 6.9 5.6 7.7 6.3 6.7 7.2
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3 6.1 7.7 6.3 6.4 6.0 6.9 6.7 6.9 5.8 6.8
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

INDEXING A SPECIFIED ENTRY: df[rows, cols]

```
iris[3,2] ## Returns entry for row 3, column 2
```

```
## [1] 3.2
```

INDEXING A SPECIFIED ENTRY: df[rows, cols]

```
iris[1:3,]        ## Showing first three entries of all columns
```

```
##   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
```

```
iris[1:3, c(1,2)]  ## Showing first three entries of first & second column
```

```
##   Sepal.Length Sepal.Width
## 1          5.1         3.5
## 2          4.9         3.0
## 3          4.7         3.2
```

INDEXING: [ ] vs. [[ ]]

Single brackets [ ]

returns single column data frame containing “Sepal.Length” only

```
iris["Sepal.Length"]
```

```
##     Sepal.Length
## 1            5.1
## 2            4.9
## 3            4.7
## 4            4.6
## 5            5.0
## 6            5.4
## 7            4.6
## 8            5.0
## 9            4.4
## 10           4.9
```

INDEXING: [ ] vs. [[ ]]

Double brackets [[ ]]

returns "Sepal.Length" column as a vector

```
iris[["Sepal.Length"]]
```

```
##   [1] 5.1 4.9 4.7 4.6 5.0 5.4 4.6 5.0
##  [19] 5.7 5.1 5.4 5.1 4.6 5.1 4.8 5.0
##  [37] 5.5 4.9 4.4 5.1 5.0 4.5 4.4 5.0
##  [55] 6.5 5.7 6.3 4.9 6.6 5.2 5.0 5.9
##  [73] 6.3 6.1 6.4 6.6 6.8 6.7 6.0 5.7
##  [91] 5.5 6.1 5.8 5.0 5.6 5.7 5.7 6.2
## [109] 6.7 7.2 6.5 6.4 6.8 5.7 5.8 6.4
## [127] 6.2 6.1 6.4 7.2 7.4 7.9 6.4 6.3
## [145] 6.7 6.7 6.3 6.5 6.2 5.9
```

Combining row/column

CHECK DATASET DESCRIPTION (state dataset)

US State Facts and Figures

Description

Data sets related to the 50 states of the United States of America.

Usage

```
state.abb
state.area
state.center
state.division
state.name
state.region
state.x77
```

Details

R currently contains the following "state" data sets. Note that all data are arranged according to alphabetical order of the state names.

THE U.S. STATE DATASET

state.abb: 2-letter abbreviations for state names

state.x77

- Population: population estimate (year 1975)
- Income: per capita income (1974)
- Illiteracy: illiteracy (% of population in 1970)
- Murder (murder rate per 100,000 population in 1976)
- HS grad (% high school graduates in 1970)

Let’s start with rbind()

– Need to have the same number of columns

LOAD SAMPLE DATA

```
state_x77 <- as.data.frame(state.x77)    # as.data.frame() converts table-like object into a data frame
str(state_x77)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
##  $ HS Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
```

rbind()

```
binding_data <- c(1,10,1,100,10,20,21,10000)
new_state_x77 <- rbind(state_x77, binding_data)
str(new_state_x77)
```

```
## 'data.frame':    51 obs. of  8 variables:
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2
##  $ Life Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10
##  $ HS Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
```

How do we use cbind() then?

– Need to have the same number of observations

LOAD BUILT-IN DATA

```
state_x77 <- as.data.frame(state.x77)    # as.da
str(state_x77)
```

```
## 'data.frame':    50 obs. of  8 variables:
##  $ Population: num  3615 365 2212 2110 21198
##  $ Income    : num  3624 6315 4530 3378 5114
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1
##  $ Life Exp  : num  69 69.3 70.5 70.7 71.7 ..
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6
##  $ HS Grad   : num  41.3 66.7 58.1 39.9 62.6
##  $ Frost     : num  20 152 15 65 20 166 139 1
##  $ Area      : num  50708 566432 113417 51945
```

```
state_abb <- as.data.frame(state.abb)
str(state_abb)
```

```
## 'data.frame':    50 obs. of  1 variable:
##  $ state.abb: chr  "AL" "AK" "AZ" "AR" ...
```

```
head(state_abb)
```

```
##   state.abb
## 1        AL
## 2        AK
## 3        AZ
## 4        AR
## 5        CA
## 6        CO
```

cbind()

```
state_abb <- as.data.frame(state.abb)
state_combined <- cbind(state_x77,state_abb)
str(state_combined)
```

```
## 'data.frame':    50 obs. of  9 variables:
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9 ...
##  $ HS Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40.6 ...
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
##  $ state.abb : chr  "AL" "AK" "AZ" "AR" ...
```

Data binding

CHECK DATASET DESCRIPTION (state dataset)

US State Facts and Figures

Description

Data sets related to the 50 states of the United States of America.

Usage

```
state.abb
state.area
state.center
state.division
state.name
state.region
state.x77
```

THE U.S. STATE DATASET

Three built-in dataset

state.abb: 2-letter abbreviations for state names

state.x77

- Population: population estimate (year 1975)
- Income: per capita income (1974)
- Illiteracy: illiteracy (% of population in 1970)
- Murder (murder rate per 100,000 population in 1976)
- HS grad (% high school graduates in 1970)

state.region: factor indicating regions (northeast, south, north central, west)

LOAD DATA (state.abb)

```
state_abb <- as.data.frame(state.abb)
str(state_abb)
```

```
## 'data.frame':    50 obs. of  1 variable:
##  $ state.abb: chr  "AL" "AK" "AZ" "AR" ...
```

LOAD DATA (state.abb)

```
head(state_abb)
```

```
##   state.abb
## 1        AL
## 2        AK
## 3        AZ
## 4        AR
## 5        CA
## 6        CO
```

LOAD DATA (state.x77)

```
state_x77 <- as.data.frame(state.x77)
state_x77 <- state_x77 %>% rownames_to_column(var="state name")
str(state_x77)
```

```
## 'data.frame':    50 obs. of  9 variables:
##  $ state name: chr  "Alabama" "Alaska" "Arizona" "Arkansas" .
##  $ Population: num  3615 365 2212 2110 21198 ...
##  $ Income    : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy: num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life Exp  : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder    : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7
##  $ HS Grad   : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.
##  $ Frost     : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area      : num  50708 566432 113417 51945 156361 ...
```

LOAD DATA (state.x77)

```
head(state_x77)
```

```
##   state name Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
## 1    Alabama       3615   3624        2.1    69.05   15.1    41.3    20  50708
## 2     Alaska        365   6315        1.5    69.31   11.3    66.7   152 566432
## 3    Arizona       2212   4530        1.8    70.55    7.8    58.1    15 113417
## 4   Arkansas       2110   3378        1.9    70.66   10.1    39.9    65  51945
## 5 California      21198   5114        1.1    71.71   10.3    62.6    20 156361
## 6   Colorado       2541   4884        0.7    72.06    6.8    63.9   166 103766
```

LOAD DATA (state.region)

```
state_region <- as.data.frame(state.region)
str(state_region)
```

```
## 'data.frame':    50 obs. of  1 variable:
##  $ state.region: Factor w/ 4 levels "Northeast","South"
```

LOAD DATA (state.region)

```
head(state_region)
```

```
##    state.region
## 1         South
## 2          West
## 3          West
## 4         South
## 5          West
## 6          West
```

MERGE THREE DATA INTO ONE DATASET

```
state_comb <- cbind(state_abb,state_x77, state_region)
str(state_comb)
```

```
## 'data.frame':    50 obs. of  11 variables:
##  $ state.abb   : chr  "AL" "AK" "AZ" "AR" ...
##  $ state name  : chr  "Alabama" "Alaska" "Arizona" "Arkansas" ...
##  $ Population  : num  3615 365 2212 2110 21198 ...
##  $ Income      : num  3624 6315 4530 3378 5114 ...
##  $ Illiteracy  : num  2.1 1.5 1.8 1.9 1.1 0.7 1.1 0.9 1.3 2 ...
##  $ Life Exp    : num  69 69.3 70.5 70.7 71.7 ...
##  $ Murder      : num  15.1 11.3 7.8 10.1 10.3 6.8 3.1 6.2 10.7 13.9
##  $ HS Grad     : num  41.3 66.7 58.1 39.9 62.6 63.9 56 54.6 52.6 40
##  $ Frost       : num  20 152 15 65 20 166 139 103 11 60 ...
##  $ Area        : num  50708 566432 113417 51945 156361 ...
##  $ state.region: Factor w/ 4 levels "Northeast","South",..: 2 4 4 2
```