Data Engineering COMP2031/8031

- Topic Coordinator: Dr Mehwish Nasim
- Office: 3.13 Tonsley

Exploratory Data Analysis

- analyze and investigate data sets and summarize their main characteristics

Process Flow

Tweepy
Twarc
Twarc2
Website
R’s datasets

MongoDB
Mongo Cloud
JSON
JSON -> CSV

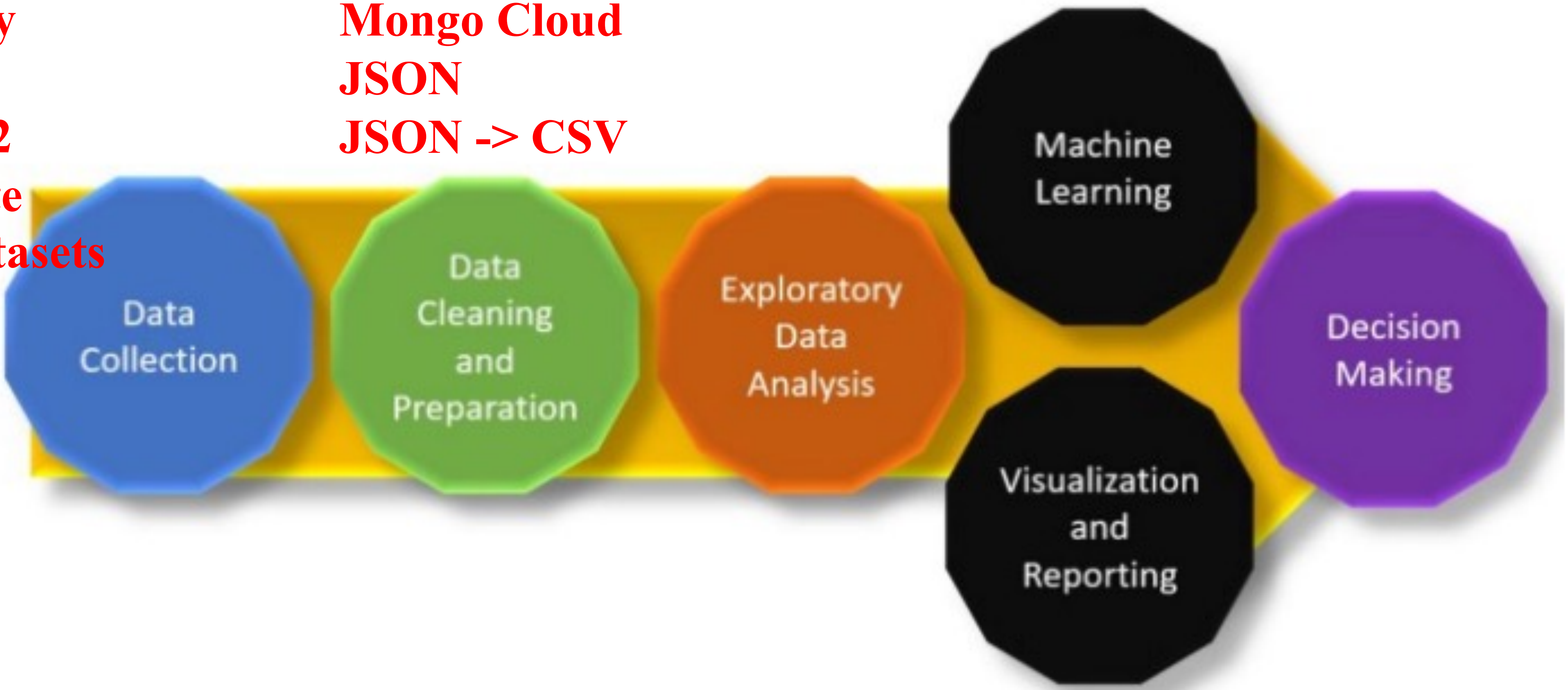

Data Collection
Data Cleaning and Preparation
Exploratory Data Analysis
Machine Learning
Visualization and Reporting
Decision Making

Figure showing the process flow from data collection to decision making.

Formulate your questions

- Be knowledgeable about your data
- Read metadata the data is not yours
- Come at an analysis with leading questions
- Hypothesis-driven studies are great, if possible
- Multiple questions are often better

Flinders
UNIVERSITY

Read in your data

- This goes without saying; you can't explore your data if it is not read in

Check the packaging

- This won’t tell you too much, but is a simple step and can help you get an understanding of the data
- Functions like ncol(), nrow(),dim(),str(),summary() are all helpful here

Use str()

- str() function in R Language is **used for compactly displaying the internal structure of a R object**.

Look at the top and the bottom of your data

- Run head() to show the top 5 rows of your data, and run tail() to show the bottom 5 rows
-

Check your numbers

- Functions like table(), dplyr::filter(), and dplyr::group_by() are excellent places to start splitting and counting your samples

Validate with at least one external data source

- Checking your data against an external data source is a great idea, and you should do it if you can
- If you cannot do this, ask yourself *how can I demonstrate that my measurements align with what others have reported?*
- Although not necessarily included in this step, here is a good place for a reminder to undertake some QA/QC by checking each variable for magnitude, direction, units, etc.

Flinders
UNIVERSITY

Try the easy solution first

- The simplest approach will also generally be the simplest code (this is good!)

Challenge your solution

- Assuming you saw something promising in your initial approach, consider variability of space and time
- Consider unbalanced data
- Consider other ways of reporting

Flinders UNIVERSITY

Follow up

- Do you have the right data?
- Do you need other data?
- Do you have the right question?

How to Interpret Summary Statistics in R

Descriptive Statistics Report

- measures of central tendency
- variability of data.

Descriptive Statistics Report

- Central tendency: refers to the tendency or the behavior of values around the mean of the dataset.

- Variability: refers to the scatter or the spread of values in the set.

Parameters

- **Mean**– It is obtained by averaging all the numerical observations in a dataset.
- **Median** – It is the midpoint that separates the data evenly into two halves. The median, unlike the mean, is not sensitive to extreme values and outliers.
- **Mode** – It tells which observation occurs most frequently in the dataset. Notice that the last two measures only apply to numerical data whereas mode can be taken for nominal data as well.

Flinders UNIVERSITY

Parameters

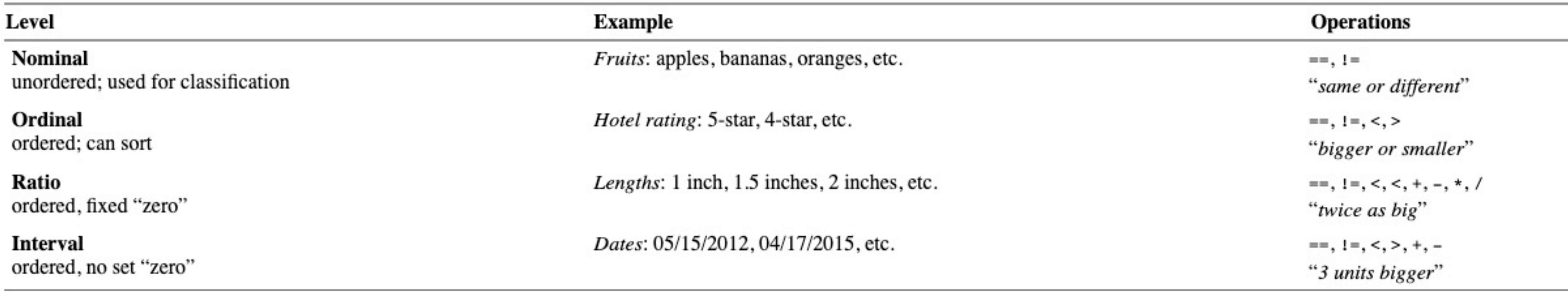
- **Range**– It’s the difference between the extremes of your data.
- **Standard Deviation**– It estimates the variation in numerical observations.
- **Variance**– It measures how *spread out* or *scattered* values are from the mean. Standard deviation squared is essentially the variance.

Flinders UNIVERSITY

Parameters

- **Skewness**– It speaks about how symmetric your data is around the average. Depending on where the extreme values lie, your data may have a positive or negative skew.
- **Kurtosis**– It is a visual estimate of the variance of a data. Your normal distribution curve may be peaked or flat, kurtosis estimates this property of your data.
- **Interquartile Range**– It divides the data into percentiles. Interquartile range is often a more interesting statistic, it is the central 50% of the data.

Types of data



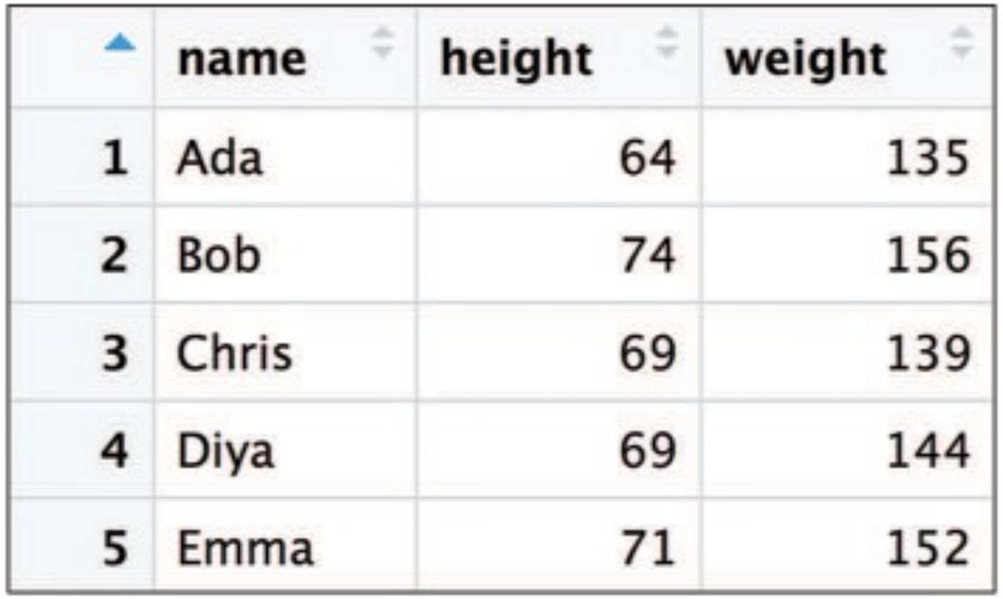
Level	Example	Operations
Nominal unordered; used for classification	*Fruits*: apples, bananas, oranges, etc.	`==`, `!=` "*same or different*"
Ordinal ordered; can sort	*Hotel rating*: 5-star, 4-star, etc.	`==`, `!=`, `<`, `>` "*bigger or smaller*"
Ratio ordered, fixed "zero"	*Lengths*: 1 inch, 1.5 inches, 2 inches, etc.	`==`, `!=`, `<`, `<`, `+`, `-`, `*`, `/` "*twice as big*"
Interval ordered, no set "zero"	*Dates*: 05/15/2012, 04/17/2015, etc.	`==`, `!=`, `<`, `>`, `+`, `-` "*3 units bigger*"

Data frames

Flinders UNIVERSITY

Data frames

- data frames act like tables, where data is organized into rows and columns.

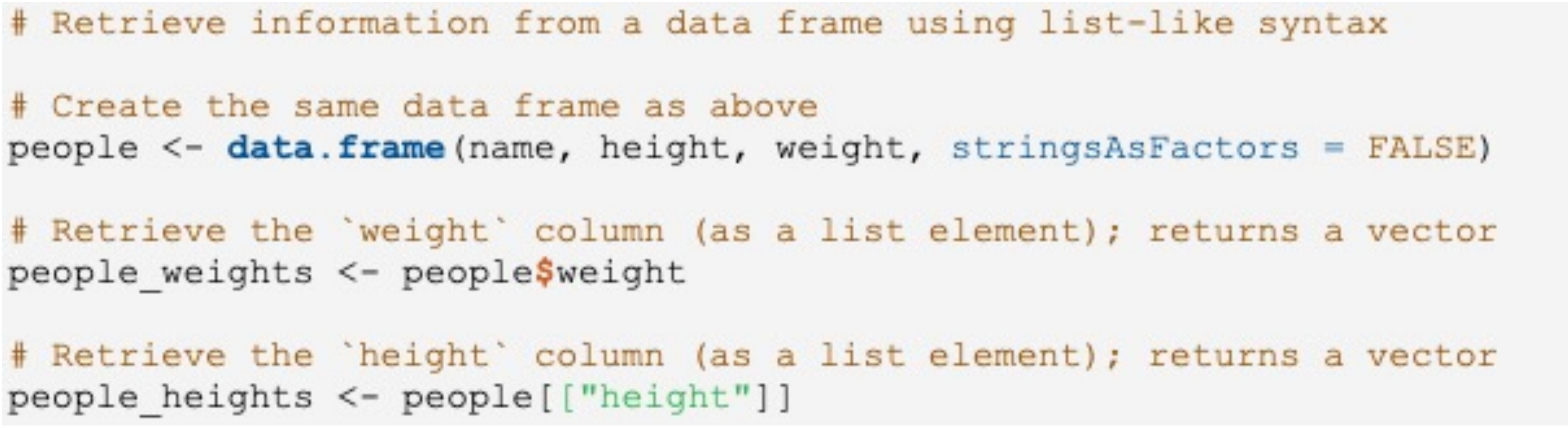


	name	height	weight
1	Ada	64	135
2	Bob	74	156
3	Chris	69	139
4	Diya	69	144
5	Emma	71	152

Figure 10.1 A table of data (of people's weights and heights) when viewed as a data frame in RStudio.

```r
# Create a data frame by passing vectors to the `data.frame()` function

# A vector of names
name <- c("Ada", "Bob", "Chris", "Diya", "Emma")

# A vector of heights
height <- c(64, 74, 69, 69, 71)

# A vector of weights
weight <- c(135, 156, 139, 144, 152)

# Combine the vectors into a data frame
# Note the names of the variables become the names of the columns!
people <- data.frame(name, height, weight, stringsAsFactors = FALSE)
```

Flinders
UNIVERSITY



```r
# Retrieve information from a data frame using list-like syntax

# Create the same data frame as above
people <- data.frame(name, height, weight, stringsAsFactors = FALSE)

# Retrieve the `weight` column (as a list element); returns a vector
people_weights <- people$weight

# Retrieve the `height` column (as a list element); returns a vector
people_heights <- people[["height"]]
```

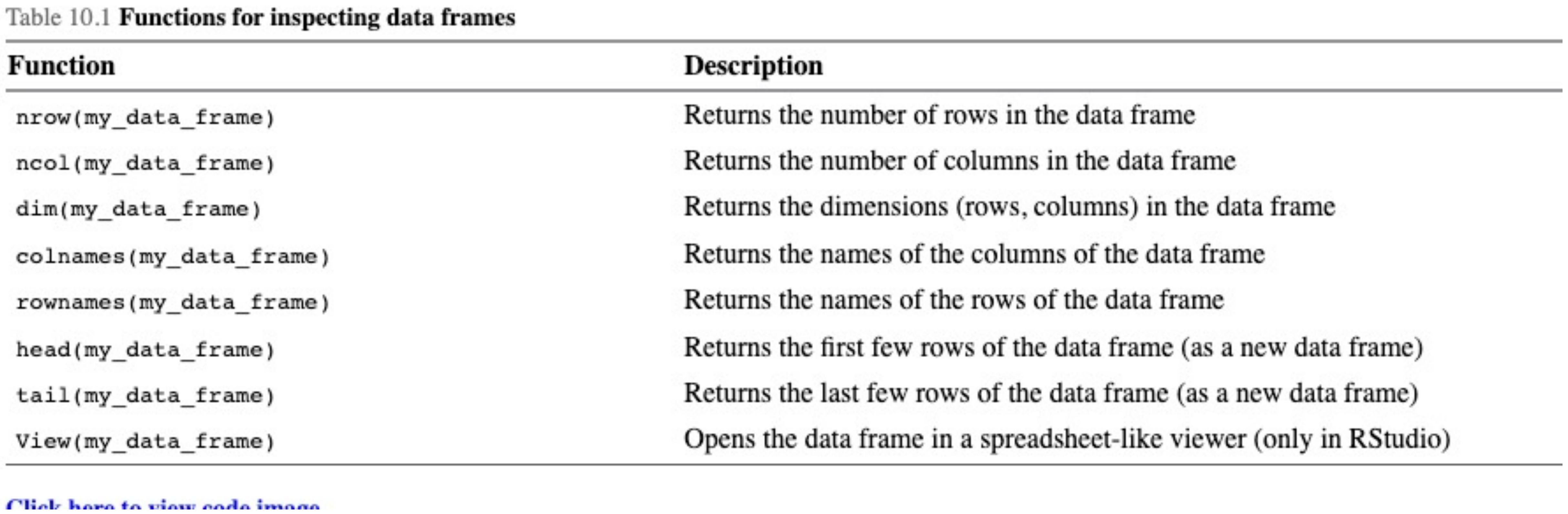


Table 10.1 **Functions for inspecting data frames**

Function	**Description**
`nrow(my_data_frame)`	Returns the number of rows in the data frame
`ncol(my_data_frame)`	Returns the number of columns in the data frame
`dim(my_data_frame)`	Returns the dimensions (rows, columns) in the data frame
`colnames(my_data_frame)`	Returns the names of the columns of the data frame
`rownames(my_data_frame)`	Returns the names of the rows of the data frame
`head(my_data_frame)`	Returns the first few rows of the data frame (as a new data frame)
`tail(my_data_frame)`	Returns the last few rows of the data frame (as a new data frame)
`View(my_data_frame)`	Opens the data frame in a spreadsheet-like viewer (only in RStudio)

Click here to view code image

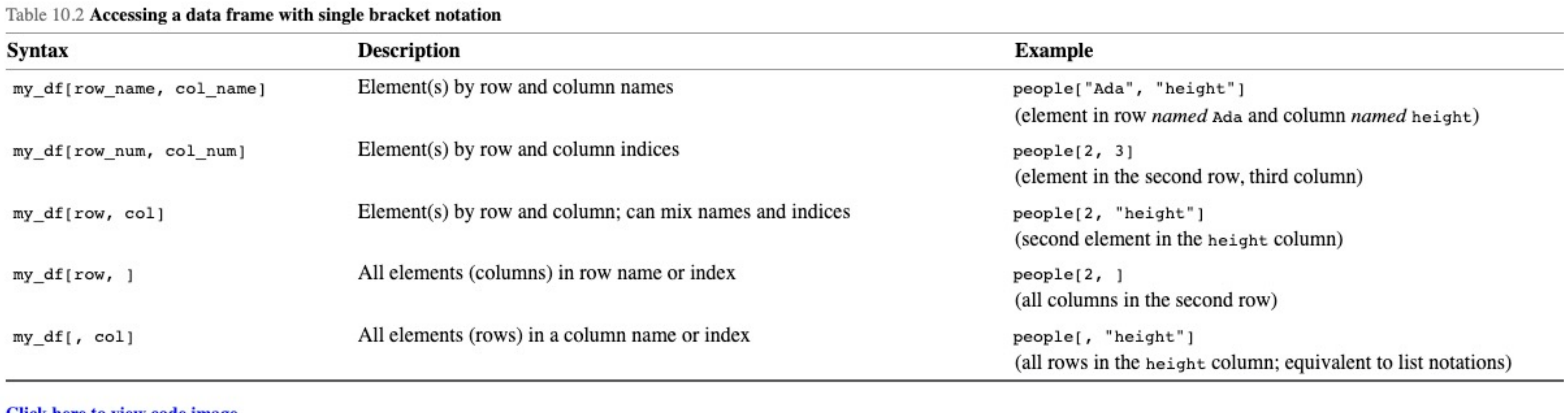


Table 10.2 **Accessing a data frame with single bracket notation**

Syntax	Description	Example
`my_df[row_name, col_name]`	Element(s) by row and column names	`people["Ada", "height"]` (element in row *named* `Ada` and column *named* `height`)
`my_df[row_num, col_num]`	Element(s) by row and column indices	`people[2, 3]` (element in the second row, third column)
`my_df[row, col]`	Element(s) by row and column; can mix names and indices	`people[2, "height"]` (second element in the `height` column)
`my_df[row, ]`	All elements (columns) in row name or index	`people[2, ]` (all columns in the second row)
`my_df[, col]`	All elements (rows) in a column name or index	`people[, "height"]` (all rows in the `height` column; equivalent to list notations)

Click here to view code image

- # Assign a set of row names for the vector
- # (using the values in the `name` column)
- rownames(people) <- people$name

- # Extract the row with the name "Ada" (and all columns)
- people["Ada",] # note the comma, indicating all columns

- # Extract the second column as a vector
- people[, "height"] # note the comma, indicating all rows

- # Extract the second column as a data frame (filtering)
- people["height"] # without a comma, it returns a data frame

Data frames

- # Get the second through fourth rows
- people[2:4,] # note the comma, indicating all columns