

Data Frame and NAs

COMM 205 - Lecture 18 - R3

Hasan Cavusoglu

2020

Agenda

- Libraries
- tidyverse
- tibble data frame
- Accessing elements/vectors from data frame
- Operations with data frames
- NA
- NA in operations

Libraries

- R comes with many *base* functions that you can readily use.
 - ▶ Examples include the functions such `max()`, `min()`, or `sum()` that we covered last time.
- However, there are many more useful R functions which come in *packages*, free libraries of code written by R's active user community. To **install** an R package, you execute `install.packages()` with the name(s) of library(ies) you want to install.

```
install.packages("name_of_the_package")
```

- R will download the package from **CRAN**, so you must be connected to the Internet.
- When installing a library, R Console will display what R is performing.
 - ▶ If you see the blue greater than sign (i.e., `>`) without an error message, it means that your installation was completed without any problem.
- Once you have a package installed, you can make its contents available to use in your R session. However, you need to **load** the package to use it by running:

```
library(name_of_the_package)
```

- If you need a package in a session, you need to load it before being able to use its functions.

tidyverse library

The tidyverse is an opinionated collection of R packages designed for data science. The tidyverse includes the packages that are designed to be used in every day data analyses. The list of packages included in tidyverse can be found <https://www.tidyverse.org/packages/>. In COMM 205, we will use a couple of packages shipped with tidyverse. Today, I will cover **a function** that comes with one of such packages, `tibble`.

How to install tidyverse

If you execute the following code, R will install all the packages included in tidyverse with all the dependencies to your machine.

```
install.packages("tidyverse", dependencies = TRUE)
```

Loading tidyverse

- Note that, like any package, before using anything from tidyverse, you need to load the package at your current R session.

```
library(tidyverse)
```

- Here is what you see at your R Console

```
> library(tidyverse)
— Attaching packages — tidyverse 1.3.0 —
✓ ggplot2 3.3.2    ✓ purrr  0.3.4
✓ tibble  3.0.4    ✓ dplyr  1.0.2
✓ tidyr   1.1.2    ✓ stringr 1.4.0
✓ readr   1.4.0    ✓ forcats 0.5.0
— Conflicts — tidyverse_conflicts() —
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
>
```

- The messages that you see above tell you what packages are loaded and warns you that function names appear in more than one package (*for the purpose of this course, you do not need to worry about it*).

tibble Data Frame

A **data.frame** is an R object used to store a collection of observations for one or more variables. You can think of R data frame as a table with columns representing variables and rows representing observations.

A **tibble** Tidyverse's interpretation of `data.frame`.

How to construct a tibble

- To construct a tibble data frame from scratch, we will use `tibble()` function from tibble package. The basic usage of `tibble()` requires you to specify the named or unnamed vectors as the data frame's column as follows.

Syntax

```
tibble(...[column_namei =] vectori, ...)
```

- More detailed description of the function can be obtained [here](#).
- While `tibble()` allows you to include *unnamed* vector as a column for the data frame, I would encourage you to give a name to each column.
- We will **not** use `data.frame()` from **Base R** to constructs data frame objects as it has some limitations.

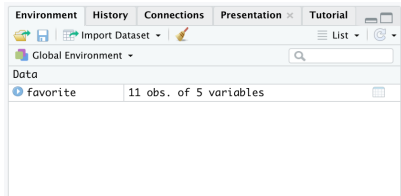
Movie Example

- Let's now create our first tibble data frame about movies. favorite data frame will have title, year, duration, budget, rating of 11 movies with large budgets, which were released before 2005:

```
favorite <- tibble(title = c("Spider-Man 2",  
  "Titanic",  
  "Troy",  
  "Terminator 3: Rise of the Machines",  
  "Waterworld",  
  "Wild Wild West",  
  "Van Helsing",  
  "Alexander",  
  "Master and Commander: The Far Side of the World",  
  "Polar Express, The",  
  "Tarzan"),  
  
  year = c(2004, 1997, 2004, 2003, 1995, 1999, 2004, 2004, 2003, 2004, 1999),  
  
  duration = c(127, 194, 162, 109, 176, 107, 132, 175, 138, 99, 88),  
  
  budget = c(200000000, 200000000, 185000000, 175000000, 175000000, 170000000,  
    160000000, 150000000, 150000000, 150000000, 150000000),  
  
  rating = c(7.9, 6.9, 7.1, 6.9, 5.4, 4.0, 5.4, 5.5, 7.5, 6.8, 7.1))
```

- Once the above function is properly executed, the object called favorite is created.

Data frame information



Each data frame created is added to the list of data in the environment pane as follows.

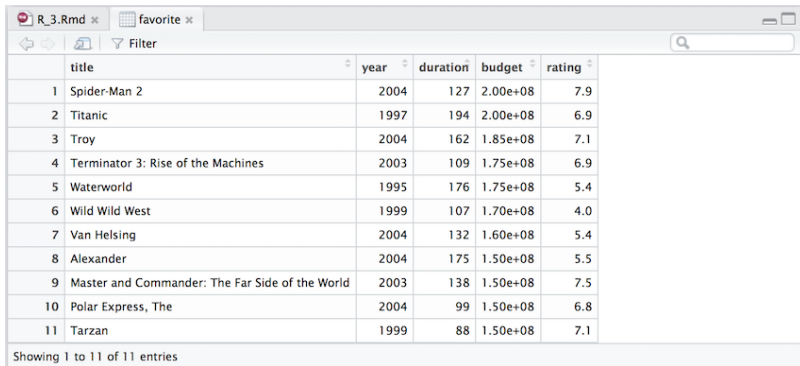
The screenshot shows the RStudio Environment pane with the 'favorite' data frame selected. The data is displayed in a table format with the following columns and values:

title	chr [1:11]	"Spider-Man 2"	"Titanic"	"Troy"	"Ter..."
year	num [1:11]	2004	1997	2004	2003 1995 ...
duration	num [1:11]	127	194	162	109 176 107 132 175 13...
budget	num [1:11]	2.00e+08	2.00e+08	1.85e+08	1.75e+08...
rating	num [1:11]	7.9	6.9	7.1	6.9 5.4 4 5.4 5.5 7.5 6...

On the environment pane, if you click on the arrow before the name of the data frame, the column names and their data types are summarized with some examples of the actual observations

Viewing a data frame in a tabular format

- If you click on the the name of the data frame listed in the environment pane, the data frame will be displayed in a new tab in the code/editor pane as seen in the screenshot below.
- Alternatively, you could execute `View(favorite)`.



The screenshot shows an RStudio window with two tabs: 'R_3.Rmd' and 'favorite'. The 'favorite' tab is active, displaying a data frame in a tabular format. The table has columns for 'title', 'year', 'duration', 'budget', and 'rating'. The data is sorted by 'rating' in descending order. The table shows 11 entries, with the first 10 visible in the main pane and the 11th at the bottom. The status bar at the bottom indicates 'Showing 1 to 11 of 11 entries'.

	title	year	duration	budget	rating
1	Spider-Man 2	2004	127	2.00e+08	7.9
2	Titanic	1997	194	2.00e+08	6.9
3	Troy	2004	162	1.85e+08	7.1
4	Terminator 3: Rise of the Machines	2003	109	1.75e+08	6.9
5	Waterworld	1995	176	1.75e+08	5.4
6	Wild Wild West	1999	107	1.70e+08	4.0
7	Van Helsing	2004	132	1.60e+08	5.4
8	Alexander	2004	175	1.50e+08	5.5
9	Master and Commander: The Far Side of the World	2003	138	1.50e+08	7.5
10	Polar Express, The	2004	99	1.50e+08	6.8
11	Tarzan	1999	88	1.50e+08	7.1

Showing 1 to 11 of 11 entries

Accessing a value from a data frame

You can extract any data entry by using its row and column indexes.

Suppose you are interested in the value in the 2nd row (correspond to Titanic) and the 3rd column (corresponding to duration.)

As an atomic value by using `[[]]`

```
> favorite[[2,3]]  
[1] 194
```

Note that the **output** is an **atomic** value

As a data frame by using `[]`

```
> favorite[2,3] # duration of Titanic?  
# A tibble: 1 x 1  
  duration  
    <dbl>  
1      194
```

Note that the **output** is a **data frame** with one row and one column

Accessing to a column

- `[[.]]` or `$` operators can be used to access a column of a data frame.
- In COMM 205, We will cover **only** `$` to access to a column of a data frame.
- **`$` operator** extracts a column of a data frame as a **vector**. That is, the outcome of the column extraction is a vector.

```
> favorite$title
[1] "Spider-Man 2"
[2] "Titanic"
[3] "Troy"
[4] "Terminator 3: Rise of the Machines"
[5] "Waterworld"
[6] "Wild Wild West"
[7] "Van Helsing"
[8] "Alexander"
[9] "Master and Commander: The Far Side of the World"
[10] "Polar Express, The"
[11] "Tarzan"

> favorite$rating
[1] 7.9 6.9 7.1 6.9 5.4 4.0 5.4 5.5 7.5 6.8 7.1
```

- Note that the result of the `$` operation is a vector. You can confirm this by `is.vector(favorite$title)` where `is.vector()` is a function that returns TRUE if the object passed on is a vector.

Subsetting a column of data frame

Question

Suppose you want to find the title(s) of the movies which were released after 2000 in our **favorite** data frame.

- First, we obtain the indexes (or indices) of `favorite$year` which are larger than 2000 by using `which()` function.
- Then, we subset `favorite$title` column with corresponding indexes found above. That means that we should nest `which()` in `[]` as follows.

```
> favorite$title[which(favorite$year > 2000)]  
[1] "Spider-Man 2"  
[2] "Troy"  
[3] "Terminator 3: Rise of the Machines"  
[4] "Van Helsing"  
[5] "Alexander"  
[6] "Master and Commander: The Far Side of the World"  
[7] "Polar Express, The"
```

Pictorially

	title	year	duration	budget	rating
1	Spider-Man 2	2004	127	2.00e+08	7.9
2	Titanic	1997	194	2.00e+08	6.9
3	Troy	2004	162	1.85e+08	7.1
4	Terminator 3: Rise of the Machines	2003	109	1.75e+08	6.9
5	Waterworld	1995	176	1.75e+08	5.4
6	Wild Wild West	1999	107	1.70e+08	4.0
7	Van Helsing	2004	132	1.60e+08	5.4
8	Alexander	2004	175	1.50e+08	5.5
9	Master and Commander: The Far Side of the World	2003	138	1.50e+08	7.5
10	Polar Express, The	2004	99	1.50e+08	6.8
11	Tarzan	1999	88	1.50e+08	7.1

```
favorite$title[which(favorite$year > 2000)]
```

- Remember `which(favorite$year > 2000)` gives you a vector of these indexes: 1, 3, 4, 7, 8, 9, 10. These indexes correspond to the row numbers of the data frame that satisfied the condition. By using these indexes, we can subset `favorite$title` is a vector.

How to modify a data frame

Suppose I just realized that the rating of Troy has increased to 7.2 on www.imdb.com. How can I update my data frame with this information?

```
favorite$rating[which(favorite$title == "Troy")] <- 7.2
```

- `which(favorite$title == "Troy")` will give you the index of the movie titled "Troy". By using the index obtained, you can update a particular value in another column of the data frame corresponding to that movie (i.e., index).
- We can check if the rating is updated:

```
View(favorite)
```

Calculation with a column

- 1) Once you access a column with a \$ operator, you can use the vector in any *arithmetic* or *logical* operations.

You are asked to create a vector whose elements are 'TRUE' if each of the corresponding movie is longer than 2 hours and 'FALSE' otherwise.

```
> favorite$duration > 120  
[1] TRUE TRUE TRUE FALSE TRUE FALSE TRUE TRUE TRUE FALSE FALSE
```

You are asked to create a new vector by adding 5 minutes to 'duration' of each movie.

```
> favorite$duration + 5  
[1] 132 199 167 114 181 112 137 180 143 104 93
```

- 2) Also, once you access a column with a \$ operator, you can use the vector with any *R function* which takes a vector as its argument.

if you are asked to determine the highest budget, you can use 'max()' as follows:

```
> max(favorite$budget)
```

How to create a new column in an existing data frame

1) You can assign a **vector of values** to a column of the data frame

- Suppose you decide to add two new columns: *is.action* to indicate if the movie is an action movie or not and *is.drama* to indicate if the movie is a drama or not.

```
favorite$is.action <- c(TRUE, FALSE, TRUE, TRUE, TRUE, TRUE, TRUE, FALSE,
                        TRUE, FALSE, FALSE)
favorite$is.drama <- c(FALSE, TRUE, FALSE, FALSE, TRUE, FALSE, FALSE, TRUE,
                      TRUE, FALSE, FALSE)
```

- Let's view the updated favorite data frame (You can either type `View(favorite)` on the Console or click on `favorite` on the Environment pane.

	title	year	duration	budget	rating	is.action	is.drama
1	Spider-Man 2	2004	127	2.00e+08	7.9	TRUE	FALSE
2	Titanic	1997	194	2.00e+08	6.9	FALSE	TRUE
3	Troy	2004	162	1.85e+08	7.2	TRUE	FALSE
4	Terminator 3: Rise of the Machines	2003	109	1.75e+08	6.9	TRUE	FALSE
5	Waterworld	1995	176	1.75e+08	5.4	TRUE	TRUE
6	Wild Wild West	1999	107	1.70e+08	4.0	TRUE	FALSE
7	Van Helsing	2004	132	1.60e+08	5.4	TRUE	FALSE
8	Alexander	2004	175	1.50e+08	5.5	FALSE	TRUE
9	Master and Commander: The Far Side of the World	2003	138	1.50e+08	7.5	TRUE	TRUE
10	Polar Express, The	2004	99	1.50e+08	6.8	FALSE	FALSE
11	Tarzan	1999	88	1.50e+08	7.1	FALSE	FALSE

How to create a new column in an existing data frame

2) The new column can be a **calculated column**.

- Let's coin a concept called `rating_ROI` and define it as the *ratio of rating to budget*.

```
favorite$rating_ROI <- favorite$rating / favorite$budget
```

- Here is the data frame with the new column.

	title	year	duration	budget	rating	is.action	is.drama	rating_ROI
1	Spider-Man 2	2004	127	2.00e+08	7.9	TRUE	FALSE	3.950000e-08
2	Titanic	1997	194	2.00e+08	6.9	FALSE	TRUE	3.450000e-08
3	Troy	2004	162	1.85e+08	7.1	TRUE	FALSE	3.837838e-08
4	Terminator 3: Rise of the Machines	2003	109	1.75e+08	6.9	TRUE	FALSE	3.942857e-08
5	Waterworld	1995	176	1.75e+08	5.4	TRUE	TRUE	3.085714e-08
6	Wild Wild West	1999	107	1.70e+08	4.0	TRUE	FALSE	2.352941e-08
7	Van Helsing	2004	132	1.60e+08	5.4	TRUE	FALSE	3.375000e-08
8	Alexander	2004	175	1.50e+08	5.5	FALSE	TRUE	3.666667e-08
9	Master and Commander: The Far Side of the World	2003	138	1.50e+08	7.5	TRUE	TRUE	5.000000e-08
10	Polar Express, The	2004	99	1.50e+08	6.8	FALSE	FALSE	4.533333e-08
11	Tarzan	1999	88	1.50e+08	7.1	FALSE	FALSE	4.733333e-08

Missing Values

- **Not** all datasets are **complete**.
- They often contain some *missing values*.
- For example, you might have daily mid-day temperature of the cities in your dataset but some of the temperatures were not recorded in a particular city due to some technical issues. That is, there are some values of the temperature are missing for some observations.
- R represents *missing values* with NA, which reads as *Not Available*.
- Missing values can exist in *any data type*. Let's create a simple data frame which has some student information.

```
age <- c(18,21,20,NA)
ethnicity <- c("French", NA, "Malaysian", "Korean")
canadian_resident <- c(TRUE, TRUE, NA, FALSE)
student <- tibble(age, ethnicity, canadian_resident)
```

	age	ethnicity	canadian_resident
1	18	French	TRUE
2	21	NA	TRUE
3	20	Malaysian	NA
4	NA	Korean	FALSE

```
> student$age
[1] 18 21 20 NA
> student$ethnicity
[1] "French"      NA      "Malaysian" "Korean"
> student$canadian_resident
[1] TRUE  TRUE   NA FALSE
```

- As you can see, each of the numeric, character, and logical data types can have missing values.
- For example, the fourth individual's age is not available. That does not mean that he/she does not have any age; just that we do not know his/her age. Similarly, we do not know the ethnicity of the second individual, and the Canadian residency status of the last individual.

Arithmetic/Logical Operations with NA

Rule of Thumb

Any **arithmetic or logical operation** involving **NA** would result in **NA**.

- Arithmetic operations involving NA

```
> 1 + NA
[1] NA
> NA * 3
[1] NA
> NA / 2
[1] NA
```

- Logical operations involving NA

```
> NA > 3
[1] NA
> NA == "German"
[1] NA
> NA != TRUE
[1] NA
> NA == NA
[1] NA
```

Arithmetic/Logical Operations with NA

- Rule of thumb applies to any calculation involving a **column/vector** containing **NA**.

Questions

Using student data frame, answer the following.

- How old would the students be in 10 years from now?
- Has anyone have an Italian ethnicity background?
- Who is paying domestic student tuition and fees?

```
> student$age + 10
[1] 28 31 30 NA
> student$ethnicity == "Italian"
[1] FALSE  NA FALSE FALSE
> student$canadian_resident == TRUE
[1] TRUE TRUE  NA FALSE
```

- Since we do not know the age of the fourth individual, we do not
- Since we do not know the ethnicity of the second individual, we do
- Since the fourth individual is not Canadian resident, he/she is not

NAs with R functions

Question

What is the **average age** of these students?

```
> mean(student$age)
[1] NA
```

- Why? You could have guessed that it is something to do with the fact that there is an NA in the age column.
- Some R functions, like `mean()`, have an argument `na.rm`.
- `na.rm` is a logical value indicating whether NA values should be ignored before the computation proceeds.
- By default, `na.rm` is set to `FALSE` in many functions.
- By explicitly setting `na.rm` to `TRUE`, you can instruct R to do the calculation only with the non-NA values.
- Here is how you could obtain the average age of the three individuals (ignoring the one whose age is not known):

```
> mean(student$age, na.rm = TRUE)
```

is.na() function

Question

Suppose you are asked to identify **NAs** in the **age** column of student data frame

```
> student$age==NA  
[1] NA NA NA NA
```

- This did not work as we know that any arithmetic/logical operations with NA results in NA.

is.na()

is.na() function allows us to check if elements of an R object is NA or not. The output indicates which elements of the object are missing. **TRUE** indicates missing element **FALSE** indicates non-missing element.

```
> is.na(student$age)  
[1] FALSE FALSE FALSE TRUE
```

- If the R object has many elements that make harder to spot if there is any TRUE in the result, the following trick would work.

The End

Thanks for watching

See you in next time!

© 2020 Hasan Cavusoglu - UBC

This content is protected and may not be shared, uploaded, or distributed.