

# THE #RDATATABLE PACKAGE

for fast, flexible and memory efficient data wrangling

Arun Srinivasan

CO-DEVELOPER, DATA.TABLE

# Willem Ligtenberg

FORMER COLLEAGUE, DEVELOPER, DATA.TABLE USER  
ASSISTING ME TODAY

# #RDATATABLE

# #RDATA TABLE

- Main author: Matt Dowle, H2O.ai

# #RDATATABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>

# #RDATATABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>
- Since 2006 on CRAN, >35 releases so far

# #RDATATABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>
- Since 2006 on CRAN, >35 releases so far
- ~6100 unit tests, ~90% coverage (using covr)

# #RDATATABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>
- Since 2006 on CRAN, >35 releases so far
- ~6100 unit tests, ~90% coverage (using covr)
- ~451 packages import/depend/suggest data.table



# #RDATATABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>
- Since 2006 on CRAN, >35 releases so far
- ~6100 unit tests, ~90% coverage (using covr)
- ~451 packages import/depend/suggest data.table
  - ~22 packages per month since Jan'17

# #RDATATABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>
- Since 2006 on CRAN, >35 releases so far
- ~6100 unit tests, ~90% coverage (using covr)
- ~451 packages import/depend/suggest data.table
  - ~22 packages per month since Jan'17
- 10th most starred R package on Github (METACRAN)

# #RDATAABLE

- Main author: Matt Dowle, H2O.ai
- Homepage: <http://r-datatable.com>
- Since 2006 on CRAN, >35 releases so far
- ~6100 unit tests, ~90% coverage (using covr)
- ~451 packages import/depend/suggest data.table
  - ~22 packages per month since Jan'17
- 10th most starred R package on Github (METACRAN)
- >5800 Q on StackOverflow. 3rd amongst R packages

# OBJECTIVE

What is the purpose of the project?

What are the goals and objectives?

What are the expected outcomes?

What are the risks and challenges?

What are the resources and budget?

What are the roles and responsibilities?

What are the timelines and milestones?

What are the communication and reporting mechanisms?

What are the evaluation and feedback mechanisms?

What are the next steps and actions?

What are the conclusions and recommendations?

What are the references and sources?

What are the appendices and annexes?

# OBJECTIVE

- Main: Get you comfortable with data.table's DT[i, j, by] syntax.

# OBJECTIVE

- Main: Get you comfortable with data.table's DT[i, j, by] syntax.
- If time permits, will discuss fread, fwrite, rbindlist, reshape etc.

# BASE R REFRESHER

```
set.seed(20170703L)
DF1 = data.frame(id = sample(1:2, 9L, TRUE),
                  code = sample(letters[1:3], 9, TRUE),
                  valA = 1:9, valB = 10:18,
                  stringsAsFactors = FALSE)

DF2 = data.frame(id = c(3L, 1L, 1L, 2L, 3L),
                  code = c("b", "a", "c", "c", "d"),
                  mul = 5:1, stringsAsFactors = FALSE)
```

# EXERCISE 1A

- 1a. Subset all rows where **id** column equals **1** & code column is not equal to "**c**"
- 1b. Same as (1) but perform the subset using **with()**. See **?with** if necessary
2. Select **valA** and **valB** columns from **DF1** and store it in variable **tmp1**
3. Get **sum(valA)** and **sum(valB)** for **id > 1** as a 1-row, 2-col data.frame
4. Replace **valB** with **valB+1** for all rows where **code == "c"**
5. Add a new column **valC** column with values equal to **valB^2 - valA^2**
6. Get **sum(valA)** and **sum(valB)** grouped by **id** and **code** (i.e., for each unique combination of **id, code**)
7. Get **sum(valA)** and **sum(valB)** grouped by **id** for **id >= 2 & code %in% c("a", "c")**
8. Replace **valA** with **max(valA)-min(valA)** grouped by **code**
9. Create a new col named **valD** with **max(valB)-min(valA)** grouped by **code**



# EXERCISE 1B

10. Subset **DF1** by **DF2** on **id,code** column. That is, for each row of **DF2\$id**, **DF2\$code**, get **valA** and **valB** cols from **DF1**. Include rows that have no matches as well.
11. Same as (10), but fetch just the **first** matching row of **DF1** for each row of **DF2\$id**, **DF2\$code**. Exclude non-matching rows.
12. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valA** with **valA\*mul**.
13. Add a new column **val** to **DF1** with values from **DF2\$mul** where **DF2\$id**, **DF2\$code** matches with **DF1**'s. Rows that don't match should have **NA**.
14. Compute **sum(valA)\*mul** for every row of **DF2\$id**, **DF2\$code** by matching it against **DF1**.
15. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valB** with **valB\*mul**.

**Every question is a good question!**  
**Feel free to interrupt.**

# COMMON OPERATIONS - 1 TABLE

1. **SELECT** - Retrieve data from a table

2. **INSERT** - Add new data to a table

3. **UPDATE** - Modify existing data in a table

4. **DELETE** - Remove data from a table

5. **JOIN** - Combine data from two or more tables

6. **AGGREGATE** - Perform calculations on data

7. **GROUP BY** - Group data by one or more columns

8. **ORDER BY** - Sort data by one or more columns

9. **INDEX** - Improve query performance

10. **VIEW** - Create a virtual table from a query

11. **TRIGGER** - Execute a set of actions when an event occurs

12. **SEQUENCE** - Generate a series of numbers

13. **SYNCHRONIZATION** - Control access to shared resources

14. **REPLICATION** - Copy data from one database to another

15. **BACKUP** - Create a copy of a database

16. **RESTORE** - Recover a database from a backup

17. **COMMIT** - Save changes to a database

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			



# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

A	B	C	D

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

A	B	C	D

A	C

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

A	B	C	D

A	C

A	B

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

A	B	C	D

A	C

A	B

A	B	C	D
3			
2			
1			
1			
3			

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

A	B	C	D

A	C

A	B

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

1

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

2

A	B	C	D

A	C

A	B

3

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

4

# COMMON OPERATIONS - 2 TABLES

1. **SELECT** - retrieve data from the database

2. **INSERT** - add new data to the database

3. **UPDATE** - modify existing data in the database

4. **DELETE** - remove data from the database

5. **JOIN** - combine data from two or more tables

6. **INDEX** - improve the performance of the database

7. **VIEW** - create a virtual table from the data in one or more tables

8. **TRIGGER** - execute a set of statements in response to an event



# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	E
4	
1	

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	E
4	
1	

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	E
4	
1	

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	B	C	D
3			
2			
1			
1			
3			

A	E
4	
1	

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

A	E
4	
1	

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

A	E
4	
1	

A	B	C	D
4	-	-	-
1			

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

A	E
4	
1	

A	B	C	D
4	-	-	-
1			

A	B	E
1		

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

A	E
4	
1	

A	B	C	D
4	-	-	-
1			

A	B	E
1		

A	B	C	D
3			
2			
1			
1			
3			



# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

A	E
4	
1	

A	B	C	D
4	-	-	-
1			

A	B	E
1		

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

1

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

2

A	E
4	
1	

A	B	C	D
4	-	-	-
1			

A	B	E
1		

3

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

4

# COMMON OPERATIONS - 1 TABLE

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
1			
1			

A	C
3	
2	
1	
1	
3	

A	B
1	
1	

1

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

2

A	B	C	D

A	C

A	B

3

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				
2				
1				
1				
3				

4

# COMMON OPERATIONS - 2 TABLES

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D
4	-	-	-
1			
1			

A	B	E
1		
1		

1

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

2

A	E
4	
1	

A	B	C	D
4	-	-	-
1			

A	B	E
1		

3

A	B	C	D
3			
2			
1			
1			
3			

A	B	C	D	E
3				-
2				-
1				
1				
3				-

4

# OUTLINE

1	Introduction	1
2	Background	2
3	Methodology	3
4	Results	4
5	Discussion	5
6	Conclusion	6
7	References	7
8	Appendix	8
9	Glossary	9
10	Index	10

# OUTLINE

- Discuss each group (1-4) one by one and learn data.table with these exercises you just attempted (~10 min for each group)
- You will internalise the concepts by solving exercises immediately after each group (~15-20 min)
- We will go through the answers (~5-10 min)
- If time permits, discuss other functions.

# DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

DATA TABLES

# DATA TABLES

- think in terms of basic units — rows, columns and groups



# DATA TABLES

- think in terms of basic units — **rows**, **columns** and **groups**
- data.table syntax provides *placeholder* for each of them

General form: DT[**i**, **j**, **by**]

# DATA TABLES

- think in terms of basic units — **rows**, **columns** and **groups**
- data.table syntax provides *placeholder* for each of them

General form: DT[**i**, **j**, **by**]

On which rows



What to do?

Grouped by  
what?

# CREATING DATA.TABLE

```
require(data.table)  
DT1 = as.data.table(DF1)  
DT2 = as.data.table(DF2)
```

# Group 1

# EXERCISE 1A

1.  $\frac{1}{2} + \frac{1}{3} = \frac{3}{6} + \frac{2}{6} = \frac{5}{6}$

2.  $\frac{1}{4} + \frac{1}{5} = \frac{5}{20} + \frac{4}{20} = \frac{9}{20}$

3.  $\frac{1}{6} + \frac{1}{8} = \frac{4}{24} + \frac{3}{24} = \frac{7}{24}$

4.  $\frac{1}{10} + \frac{1}{12} = \frac{6}{60} + \frac{5}{60} = \frac{11}{60}$

5.  $\frac{1}{15} + \frac{1}{20} = \frac{4}{60} + \frac{3}{60} = \frac{7}{60}$

6.  $\frac{1}{18} + \frac{1}{24} = \frac{4}{72} + \frac{3}{72} = \frac{7}{72}$

7.  $\frac{1}{25} + \frac{1}{30} = \frac{6}{150} + \frac{5}{150} = \frac{11}{150}$

8.  $\frac{1}{35} + \frac{1}{42} = \frac{6}{210} + \frac{5}{210} = \frac{11}{210}$

9.  $\frac{1}{45} + \frac{1}{54} = \frac{2}{27} + \frac{1}{9} = \frac{2}{27} + \frac{3}{27} = \frac{5}{27}$

10.  $\frac{1}{56} + \frac{1}{63} = \frac{3}{252} + \frac{4}{252} = \frac{7}{252} = \frac{1}{36}$

11.  $\frac{1}{72} + \frac{1}{81} = \frac{5}{324} + \frac{4}{324} = \frac{9}{324} = \frac{1}{36}$

12.  $\frac{1}{90} + \frac{1}{100} = \frac{2}{45} + \frac{1}{10} = \frac{2}{45} + \frac{9}{45} = \frac{11}{45}$

13.  $\frac{1}{108} + \frac{1}{120} = \frac{5}{270} + \frac{3}{270} = \frac{8}{270} = \frac{4}{135}$

14.  $\frac{1}{126} + \frac{1}{140} = \frac{10}{1260} + \frac{9}{1260} = \frac{19}{1260}$

15.  $\frac{1}{144} + \frac{1}{160} = \frac{5}{720} + \frac{4}{720} = \frac{9}{720} = \frac{1}{80}$

16.  $\frac{1}{168} + \frac{1}{180} = \frac{5}{720} + \frac{4}{720} = \frac{9}{720} = \frac{1}{80}$

17.  $\frac{1}{180} + \frac{1}{200} = \frac{2}{90} + \frac{1}{10} = \frac{2}{90} + \frac{9}{90} = \frac{11}{90}$

# EXERCISE 1A

**1a.** Subset all rows where **id** column equals **1** & code column is not equal to "**c**"

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

# EXERCISE 1A

**1a.** Subset all rows where **id** column equals **1** & code column is not equal to "**c**"

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[id == 1 & code != "c"]
```

# EXERCISE 1A

**1a.** Subset all rows where **id** column equals **1** & code column is not equal to "**c**"

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[id == 1 & code != "c"]
```



# EXERCISE 1A

**1a.** Subset all rows where **id** column equals **1** & code column is not equal to "**c**"

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[id == 1 & code != "c"]
```

	id	code	valA	valB
1:	1	b	2	11
2:	1	b	7	16

# EXERCISE 1A

# EXERCISE 1A

2. Select **valA** and **valB** columns from **DF1**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

# EXERCISE 1A

2. Select **valA** and **valB** columns from **DF1**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[, list(valA, valB)]  
  .() alias to list()
```

# EXERCISE 1A

2. Select **valA** and **valB** columns from **DF1**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[, list(valA, valB)]  
  .() alias to list()
```

# EXERCISE 1A

2. Select **valA** and **valB** columns from **DF1**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[, list(valA, valB)]  
  .() alias to list()
```

	valA	valB
1:	1	10
2:	2	11
3:	3	12
4:	4	13
5:	5	14
6:	6	15
7:	7	16
8:	8	17
9:	9	18

# EXERCISE 1A

# EXERCISE 1A

3. Get **sum(valA)** and **sum(valB)** for **id > 1** as a 1-row, 2-col data.frame

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[id > 1,  
     .(valA=sum(valA),  
       valB=sum(valB))]
```



# EXERCISE 1A

3. Get **sum(valA)** and **sum(valB)** for **id > 1** as a 1-row, 2-col data.frame

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[id > 1,  
     .(valA=sum(valA),  
       valB=sum(valB))]
```

# EXERCISE 1A

3. Get **sum(valA)** and **sum(valB)** for **id > 1** as a 1-row, 2-col data.frame

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[id > 1,  
     .(valA=sum(valA),  
       valB=sum(valB))]
```

	valA	valB
1:	19	46

# EXERCISE 1B

# EXERCISE 1B

10. Subset **DF1** by **DF2** on **id,code** column. That is, for each row of **DF2\$id**, **DF2\$code**, get **valA** and **valB** cols from **DF1**. Include rows that have no matches as well.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

# EXERCISE 1B

10. Subset **DF1** by **DF2** on **id,code** column. That is, for each row of **DF2\$id**, **DF2\$code**, get **valA** and **valB** cols from **DF1**. Include rows that have no matches as well.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

# EXERCISE 1B

10. Subset **DF1** by **DF2** on **id,code** column. That is, for each row of **DF2\$id**, **DF2\$code**, get **valA** and **valB** cols from **DF1**. Include rows that have no matches as well.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

```
DT1[DT2, on=.(id, code),  
      .(valA, valB)]
```

# EXERCISE 1B

10. Subset **DF1** by **DF2** on **id,code** column. That is, for each row of **DF2\$id**, **DF2\$code**, get **valA** and **valB** cols from **DF1**. Include rows that have no matches as well.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

```
DT1[DT2, on=.(id, code),  
      .(valA, valB)]
```

	valA	valB
1:	NA	NA
2:	NA	NA
3:	1	10
4:	3	12
5:	4	13
6:	9	18
7:	NA	NA
8:	NA	NA

# EXERCISE 1B

11. Same as (10), but fetch just the **first** matching row of **DF1** for each row of **DF2\$id**, **DF2\$code**. Exclude non-matching rows.

??



# EXERCISE 1B

11. Same as (10), but fetch just the **first** matching row of **DF1** for each row of **DF2\$id**, **DF2\$code**. Exclude non-matching rows.

```
DT1[DT2, on=.(id, code),  
      .(valA, valB),  
      mult="first", nomatch=0L]
```

# Practice

# GROUP 1 WRAP UP

## Easy-moderate:

1. Get all rows where `valA > 5` and `valB` is `<= 16` from DT1.
2. Get all rows where `valA` is in between 5 and 8 (both included) from DT1.
3. Order DT1 by `code` in increasing order, and within that by `valA` in decreasing order.
4. Return the last two rows of DT1.
5. Return a random sample of 4 rows.
6. Get median of `valA` and `valB` cols where `code` is not "a". Name the columns 'mA' and 'mB'.
7. Remove all rows in DT2 where DT2\$code is duplicated. Store the result in DT3. Hint: see ? duplicated.

## Moderate-Hard:

8. Return all unique combinations of `id`, `code` (as a two column data.table) where  $valA^2 > valB$ . Hint: you'll need to use the function ``unique()`` in ``j``.
9. Read ?``.SD`` and check explanation and examples and try to use ``.SD`` in ``j`` to solve (8).
10. For every DT3\$code, return the last matching values of `valA` from DT1 along with 'id' column from DT3. i.e., result should contain `code`, `valA` and `id` (from DT3) columns. Do not remove non-matching rows.

# Group 2

# EXERCISE 1A

# EXERCISE 1A

4. Replace **valB** with **valB+1** for all rows where **code == "c"**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[code == "c",  
      valB := valB+1L]  
(Or)  
DT1[code == "c",  
      `:=`(valB = valB+1L)]
```

# EXERCISE 1A

4. Replace **valB** with **valB+1** for all rows where **code == "c"**

	id	code	valA	valB
1:	1	c	1	11
2:	1	b	2	11
3:	1	c	3	13
4:	1	c	4	14
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	19

```
DT1[code == "c",  
      valB := valB+1L]  
(Or)  
DT1[code == "c",  
      `:=`(valB = valB+1L)]
```

# EXERCISE 1A

??



# EXERCISE 1A

5. Add a new column **valC** column with values equal to **valB<sup>2</sup> - valA<sup>2</sup>**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

??

# EXERCISE 1A

5. Add a new column **valC** column with values equal to **valB<sup>2</sup> - valA<sup>2</sup>**

	id	code	valA	valB	valC
1:	1	c	1	10	99
2:	1	b	2	11	117
3:	1	c	3	12	135
4:	1	c	4	13	153
5:	2	a	5	14	171
6:	2	a	6	15	189
7:	1	b	7	16	207
8:	2	a	8	17	225
9:	1	c	9	18	243

??

# EXERCISE 1A

5. Add a new column **valC** column with values equal to **valB<sup>2</sup> - valA<sup>2</sup>**

	id	code	valA	valB	valC
1:	1	c	1	10	99
2:	1	b	2	11	117
3:	1	c	3	12	135
4:	1	c	4	13	153
5:	2	a	5	14	171
6:	2	a	6	15	189
7:	1	b	7	16	207
8:	2	a	8	17	225
9:	1	c	9	18	243

```
DT1[, valC := valB^2-valA^2]
```

# EXERCISE 1B

# EXERCISE 1B

12. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valA** with **valA\*mul**.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

# EXERCISE 1B

12. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valA** with **valA\*mul**.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

# EXERCISE 1B

12. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valA** with **valA\*mul**.

	id	code	valA	valB
1:	1	c	3	10
2:	1	b	2	11
3:	1	c	9	12
4:	1	c	12	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	27	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

```
DT1[DT2, on=.(id, code),  
      valA := valA*i.mul]
```

# EXERCISE 1B



# EXERCISE 1B

13. Add a new column **val** to **DF1** with values from **DF2\$mul** where **DF2\$id**, **DF2\$code** matches with **DF1**'s. Rows that don't match should have **NA**.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

# EXERCISE 1B

13. Add a new column **val** to **DF1** with values from **DF2\$mul** where **DF2\$id**, **DF2\$code** matches with **DF1**'s. Rows that don't match should have **NA**.

	id	code	valA	valB	val
1:	1	c	1	10	3
2:	1	b	2	11	NA
3:	1	c	3	12	3
4:	1	c	4	13	3
5:	2	a	5	14	NA
6:	2	a	6	15	NA
7:	1	b	7	16	NA
8:	2	a	8	17	NA
9:	1	c	9	18	3

```
DT1[DT2, on=.(id, code),  
    val := i.mul]
```

# Practice

# GROUP 2 WRAP UP

## **Easy-moderate: (unless specified, assume DT1)**

1. On those rows where `id != 2`, replace `valA` and `valB` with `valA+1` and `valB+1` respectively.
2. On those rows where `id == 2`, replace `valA` with `valB` if `valA` is `<= 7`, else with `valB^2`.
3. Create a new column ``tmp`` and assign ``NA`` to it by reference.
4. What's the type (or class) of ``tmp`` column that we just created?
5. Do `DT1[, tmp := NULL]` and observe the output.. What's the difference compared to (3)?

## **Moderate-Hard:**

6. Create a new column named "rank" which takes value 1 where `code == "a"`, 2 where `code == "b"` and 3 where `code == "c"`. Do it in as many different ways you could think of :-).
7. Let `DT3 = DT2[!duplicated(code)]`. Update both `valA` and `valB` columns with `'valA*mul'` and `'valB*mul'` wherever `DT3$code` matches `DT1$code`.. What happens to those rows where there are no matches in `DT1`? Why?
8. Add the column `'mul'` from `DT2` to `DT1` by reference where `DT2$id` matches `DT1$id`. What happens to those values where `DT2$id` has the same value occurring more than once?
9. Replace `DT2$mul` with `NA` where `DT1$id, DT1$code` matches `DT2$id, DT2$code`.

# Group 3

# EXERCISE 1A

6. Get **sum(valA)** and **sum(valB)** grouped by **id** and **code** (i.e., for each unique combination of **id, code**)

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[, .(sum(valA), sum(valB)),  
      by=.(id, code)]
```

	id	code	V1	V2
1:	1	c	17	53
2:	1	b	9	27
3:	2	a	19	46

# EXERCISE 1A

7. Get **sum(valA)** and **sum(valB)** grouped by **id** for **id >= 2 & code %in% c("a", "c")**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	V1	V2
1:	2	a	19	46

```
DT1[id >= 2 & code %in% c("a", "c"),  
     lapply(.SD, sum),  
     by=.(id, code)]
```

# EXERCISE 1B

14. Compute **sum(valA)\*mul** for every row of **DF2\$id**, **DF2\$code** by matching it against **DF1**.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

```
DT1[DT2, on=.(id, code),  
      sum(valA)*mul,  
      by=.EACHI]
```

	id	code	V1
1:	3	b	NA
2:	1	a	NA
3:	1	c	51
4:	2	c	NA
5:	3	d	NA



# GROUP 3 WRAP UP

## Easy-moderate: (unless specified, assume DT1)

1. Get  $\max(\text{valB}) - \min(\text{valA})$  grouped by code. Name the column 'diff'.
2. Get  $\max(\text{valA}) - \min(\text{valB})$  grouped by code and id. Name the column 'diff'
3. Get the median of valA grouped by code.
4. Get the median of valA and  $\log(\text{sum}(\text{valB}))$  grouped by code. Why does it fail? Hint: Read the error message and use `verbose = TRUE` for both (3) and (4) and observe the difference.
5. For each code (i.e., grouped by code) randomly sample one row of the rest of the columns. Hint: you could do it with ``.SD`` and ``.N`` and `sample()`.

## Moderate-Hard:

6. Get the most frequently occurring code grouped by id. This might require multiple steps.
7. Get the count of values where  $\text{valA} > \sqrt{\text{valB}}$  is TRUE and the count of values where the condition isn't TRUE.
8. Get  $\min(\text{valA})$  of DT1 from rows that match id, code from DT2 and NA if it doesn't match.
9. Get  $\max(\text{valB})$  of DT1 from rows that match id, code from DT2. Only keep matching rows
10. Let  $\text{DT3} = \text{DT2}[\text{!duplicated}(\text{id})]$ . For each  $\text{DT3}\$id$ , get sum of valB only where  $\text{valA} \geq \text{mul}$ .

# Group 4

# EXERCISE 1A

# EXERCISE 1A

8. Replace **valA** with **max(valA)-min(valA)** grouped by **code**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[, valA :=  
      max(valA)-min(valA),  
      by=code]
```

# EXERCISE 1A

8. Replace **valA** with **max(valA)-min(valA)** grouped by **code**

	id	code	valA	valB
1:	1	c	8	10
2:	1	b	5	11
3:	1	c	8	12
4:	1	c	8	13
5:	2	a	3	14
6:	2	a	3	15
7:	1	b	5	16
8:	2	a	3	17
9:	1	c	8	18

```
DT1[, valA :=  
      max(valA)-min(valA),  
      by=code]
```

# EXERCISE 1A

# EXERCISE 1A

9. Create a new col named **valD** with **max(valB)-min(valA)** grouped by **code**

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

```
DT1[, valD :=  
      max(valB)-min(valA),  
      by=code]
```

# EXERCISE 1A

9. Create a new col named **valD** with **max(valB)-min(valA)** grouped by **code**

	id	code	valA	valB	valD
1:	1	c	1	10	17
2:	1	b	2	11	14
3:	1	c	3	12	17
4:	1	c	4	13	17
5:	2	a	5	14	12
6:	2	a	6	15	12
7:	1	b	7	16	14
8:	2	a	8	17	12
9:	1	c	9	18	17

```
DT1[, valD :=  
      max(valB)-min(valA),  
      by=code]
```



# EXERCISE 1B

# EXERCISE 1B

15. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valB** with **valB\*mul**.

	id	code	valA	valB
1:	1	c	1	10
2:	1	b	2	11
3:	1	c	3	12
4:	1	c	4	13
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	18

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

```
DT1[DT2, on=.(id, code),  
      valB := valB*i.mul,  
      by=.EACHI]
```

# EXERCISE 1B

15. For every row of **DF2\$id**, **DF2\$code** that matches with **DF1**'s, update **valB** with **valB\*mul**.

	id	code	valA	valB
1:	1	c	1	30
2:	1	b	2	11
3:	1	c	3	36
4:	1	c	4	39
5:	2	a	5	14
6:	2	a	6	15
7:	1	b	7	16
8:	2	a	8	17
9:	1	c	9	54

	id	code	mul
1:	3	b	5
2:	1	a	4
3:	1	c	3
4:	2	c	2
5:	3	d	1

```
DT1[DT2, on=.(id, code),  
      valB := valB*i.mul,  
      by=.EACHI]
```

# GROUP 4 WRAP UP

## **Easy-moderate: (unless specified, assume DT1)**

- 1) Update valB with  $\text{valB} * \text{no. of rows in that group}$  grouped by code
- 2) Update both valA and valB with  $\text{valA} * \max(\text{valA})$  and  $\text{valB} * \max(\text{valB})$  respectively grouped by id, code
- 3) Create two new columns 'A2', 'B2', while grouped by code, by randomly sampling (with replacement) the same number of rows in the group from valA and valB respectively.
4. Add a column named 'uniq\_N' which contains the count of unique 'code' values, while grouped by 'id'.

## **Moderate-Hard:**

5. Update all rows of valB with NA where DT3\$id, DT3\$code \*don't\* match with DT1\$id, DT1\$code.
6. Let  $\text{DT3} = \text{DT2}[\text{!duplicated(id)}]$ . For each DT3\$id, find all rows in DT1\$id that is  $\leq \text{DT3$id}$  and compute  $\text{sum(valA)} * \text{mul}$ .