

data.table syntax

JOINING DATA WITH DATA.TABLE IN R



Scott Ritchie

Postdoctoral Researcher in Systems
Genomics

Recap of the data.table syntax

General form of `data.table` syntax

```
DT[i, j, by]
  |  |  |
  |  |  --> grouped by what?
  |  -----> what to do?
  -----> on which rows?
```

Joins

General form of `data.table` syntax joins

```
DT[i, on]
  |  |
  |  ----> join key columns
-----> join to which data.table?
```

Right joins

The default join is a right join

```
demographics[shipping, on = .(name)]
```

demographics:

name	gender	age
Trey	NA	54
Matthew	M	43
Angela	F	39
Michelle	F	63

NA

shipping:

name	address
Matthew	7 Mill road
Trey	12 High street
Abdullah	3a Union street
Angela	33 Pacific boulevard



name	gender	age	address
Matthew	M	43	7 Mill road
Trey	NA	54	12 High street
Abdullah	NA	NA	3a Union street
Angela	F	39	33 Pacific boulevard

The on argument

Variables inside `list()` or `.()` are looked up in the column names of both `data.tables`

```
shipping[demographics, on = list(name)]  
shipping[demographics, on = .(name)]
```

Character vectors can also be used

```
join_key <- c("name")  
shipping[demographics, on = join_key]
```

Left joins

Remember, a left join is the same as a right join with the order swapped:

```
shipping[demographics, on = .(name)]
```

shipping:

name	address
Matthew	7 Mill road
Trey	12 High street
Abdullah	3a Union street
Angela	33 Pacific boulevard

demographics:

name	gender	age
Trey	NA	54
Matthew	M	43
Angela	F	39
Michelle	F	63



name	address	gender	age
Trey	12 High street	NA	54
Matthew	7 Mill road	M	43
Angela	33 Pacific boulevard	F	39
Michelle	NA	F	63

Inner joins

Set `nomatch = 0` to perform an inner join:

```
shipping[demographics, on = .(name), nomatch = 0]
```

shipping:

name	address
Matthew	7 Mill road
Trey	12 High street
Abdullah	3a Union street
Angela	33 Pacific boulevard

demographics:

name	gender	age
Trey	NA	54
Matthew	M	43
Angela	F	39
Michelle	F	63



name	address	gender	age
Trey	12 High street	NA	54
Matthew	7 Mill road	M	43
Angela	33 Pacific boulevard	F	39

Full joins

Not possible with the `data.table` syntax, use the `merge()` function:

```
merge(demographics, shipping, by = "name", all = TRUE)
```

demographics:

name	gender	age
Trey	NA	54
Matthew	M	43
Angela	F	39
Michelle	F	63

+

shipping:

name	address
Matthew	7 Mill road
Trey	12 High street
Abdullah	3a Union street
Angela	33 Pacific boulevard



name	gender	age	address
Abdullah	NA	NA	3a Union street
Angela	F	39	33 Pacific boulevard
Matthew	M	43	7 Mill road
Michelle	F	63	NA
Trey	M	NA	12 High street

Anti-joins

Filter a `data.table` to rows that have no match in another `data.table`

```
demographics[!shipping, on = .(name)]
```

demographics:

name	sex	age
Trey	NA	54
Matthew	M	43
Angela	F	39
Michelle	F	63

anti join



shipping:

name	address
Matthew	7 Mill road
Trey	12 High street
Abdullah	3a Union street
Angela	33 Pacific boulevard



name	sex	age
Michelle	F	63

Let's practice!

JOINING DATA WITH DATA.TABLE IN R

Setting and viewing data.table keys

JOINING DATA WITH DATA.TABLE IN R



Scott Ritchie

Postdoctoral Researcher in Systems
Genomics

Setting `data.table` keys

Setting keys means you don't need the `on` argument when performing a join

- Useful if you need to use a `data.table` in many different joins

Sorts the `data.table` in memory by the key column(s)

- Makes filtering and join operations faster

Multiple columns can be set and used as keys

The `setkey()` function

Key columns are passed as arguments

```
setkey(DT, ...)
```

```
setkey(DT, key1, key2, key3)  
setkey(DT, "key1", "key2", "key3")
```

```
# To set all columns in DT as keys  
setkey(DT)
```

The `setkey()` function

Set the keys of both `data.tables` before a join

```
setkey(dt1, dt1_key)
setkey(dt2, dt2_key)
```

Perform an inner, right, and left join:

```
# Inner join dt1 and dt2
dt1[dt2, nomatch = 0]
# Right join dt1 and dt2
dt1[dt2]
# Left join dt1 and dt2
dt2[dt1]
```

Setting keys programmatically

Key columns are provided as a character vector

```
keys <- c("key1", "key2", "key3")  
setkeyv(dt, keys)
```

Getting keys

`haskey()` checks whether you have set keys

```
haskey(dt1)
```

```
TRUE
```

`key()` returns the key columns you have set

```
key(dt1)
```

```
"dt1_key"
```


Getting keys

When no keys are set

```
haskey(dt_no_key)
```

```
FALSE
```

```
key(dt_no_key)
```

```
NULL
```

Viewing all `data.tables` and their keys

```
tables()
```

	NAME	NROW	NCOL	MB	COLS	KEY
[1,]	dt	3	4	1	key1,key2,key3,value	key1,key2,key3
[2,]	dt1	1,000	3	1	dt1_key_column,value,group	dt1_key
[3,]	dt2	1,000	2	1	dt2_key_column,time	dt2_key
[4,]	dt_no_key	5	2	1	id,color	

Total: 4MB

Let's practice!

JOINING DATA WITH DATA.TABLE IN R

Incorporating joins into your data.table workflow

JOINING DATA WITH DATA.TABLE IN R



Scott Ritchie

Postdoctoral Researcher in Systems
Genomics

Chaining data.table expressions

data.table expressions can be chained in sequence:

```
demographics[...][...]
```

General form of chaining a join:

```
DT1[DT2, on][i, j, by]  
  |      |  |  |  |  
  |      |  |  |  --> grouped by what?  
  |      |  |      -----> what to do?  
  |      |      -----> on which rows?  
  |      -----> join key columns  
  -----> join to which data.table?
```

Join then compute

```
customers <- data.table(name = c("Mark", "Matt", "Angela", "Michelle"),  
                        gender = c("M", "M", "F", "F"),  
                        age = c(54, 43, 39, 63))  
  
customers
```

	name	gender	age
1:	Mark	M	54
2:	Matt	M	43
3:	Angela	F	39
4:	Michelle	F	63

Join then compute

```
purchases <- data.table(name = c("Mark", "Matt", "Angela", "Michelle"),  
                        sales = c(1, 5, 4, 3),  
                        spent = c(41.70, 41.78, 50.77, 60.01))  
  
purchases
```

```
   name sales spent  
1:  Mark     1 41.70  
2:  Matt     5 41.78  
3: Angela     4 50.77  
4: Michelle    3 60.01
```

Join then compute

```
customers[purchases,  
          on = .(name)][sales > 1,  
                        j = .(avg_spent = sum(spent) / sum(sales)),  
                        by = .(gender)]
```

```
   gender avg_spent  
1:      M  13.91333  
2:      F  20.00333
```


Computation with joins

Computation with joins:

```
DT1[DT2, on, j]
|      |      |
|      |      ----> what to do on the join result?
|      |      -----> using which columns as keys?
|      |      -----> join to which data.table?
```

Efficient for large `data.tables` !

Joining and column creation

Column creation takes place in the main `data.table` :

```
customers[purchases, on = .(name), return_customer := sales > 1]  
customers
```

	name	gender	age	return_customer
1:	Mark	M	54	FALSE
2:	Matt	M	43	TRUE
3:	Angela	F	39	TRUE
4:	Michelle	F	63	TRUE

Grouping by matches

`by = .EACHI` groups `j` by each row **from** DT2

```
DT1[DT2, on, j, by = .EACHI]
```

```
| | | |
```

```
| | | --> grouped by each match in DT1.
```

```
| | -----> what to do on the join result?
```

```
| -----> using which columns as keys?
```

```
-----> join to which data.table?
```

Grouping by matches

```
shipping[customers, on = .(name),  
  j = .("# of shipping addresses" = .N),  
  by = .EACHI]
```

shipping:

name	type	address
Mark	residential	34 Yarra drive
Matt	residential	2 Sunshine crescent
Matt	business	12 Commercial road
Angela	residential	33 Pacific boulevard

customers:

name	gender	age
Mark	M	54
Matt	M	43
Angela	F	39
Michelle	F	63



name	# of shipping addresses
Mark	1
Matt	2
Angela	1
Michelle	0

Grouping by columns with joins

Grouping by columns in the `by` restricts computation to the main data.table:

```
DT1[DT2, on, j, by]
|      |      |      |
|      |      |      --> grouped by what columns in DT1?
|      |      -----> what to do on columns in DT1?
|      -----> using which columns as keys?
-----> join to which data.table?
```

Grouping by columns with joins

Join and calculate by group in `customers` :

```
customers[shipping, on = .(name),  
          .(avg_age = mean(age)), by = .(gender)]
```

```
   gender avg_age  
1:      M 46.66667  
2:      F 39.00000
```

Let's practice!

JOINING DATA WITH DATA.TABLE IN R