# EFFICIENT IN-MEMORY NON-EQUI JOINS

## using the #rdatatable package


Arun Srinivasan

DEVELOPER/DATA ANALYST, **OPEN ANALYTICS**


USER'16, JUNE 27-30



@ARUN_SRINIV

# WHO AM I?

1. Bioinformatician / Computational Biologist
2. R and data.table user since 2011
3. data.table developer since late 2013
4. Data analyst @Open Analytics since Feb'15

# THE PROBLEM

For each row in B replace A$z where A$x <= B$x & A$y > B$y with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

# THE PROBLEM

For each row in B replace A$z where A$x <= B$x & A$y > B$y with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | NA |
| 4: | 2 | 6 | NA |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | NA |

How can we accomplish this using data.table?

Before answering that,
a quick detour...

# ROW SUBSETS

Return all rows where x == 4

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

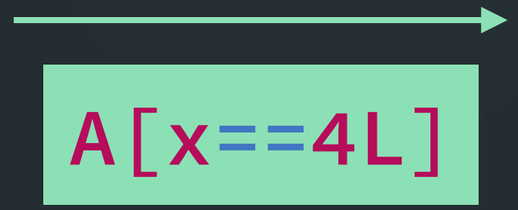
| | x | y | z |
|---|---|---|---|
| 1: | 4 | 5 | 5 |
| 2: | 4 | 5 | 6 |
| 3: | 4 | 10 | 3 |

# ROW SUBSETS

Return all rows where x == 4

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |




| | x | y | z |
|---|---|---|---|
| 1: | 4 | 5 | 5 |
| 2: | 4 | 5 | 6 |
| 3: | 4 | 10 | 3 |

# ROW SUBSETS

Return all rows where x == 4 & y == 5

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |



| | x | y | z |
|---|---|---|---|
| 1: | 4 | 5 | 5 |
| 2: | 4 | 5 | 6 |

```
A[x==4L & y==5L]
```

# SUBSET+UPDATE

Update col z for all rows where x == 4 & y == 5 with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | NA |
| 6: | 4 | 5 | NA |
| 7: | 4 | 10 | 3 |

```
A[x==4L & y==5L, z:=NA]
```

# ROW SUBSET? JOIN?

Return all rows where rows of B matches A on cols x and y

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 1 | 2 |
| 2: | 4 | 5 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 4 | 5 | 5 |
| 3: | 4 | 5 | 6 |

# JOIN...

Return all rows where rows of B matches A on cols x and y

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 1 | 2 |
| 2: | 4 | 5 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 4 | 5 | 5 |
| 3: | 4 | 5 | 6 |

```
merge(A, B, by=c("x", "y"))
```

# JOIN AS SUBSET

Return all rows where rows of B matches A on cols x and y

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 1 | 2 |
| 2: | 4 | 5 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 4 | 5 | 5 |
| 3: | 4 | 5 | 6 |

```
A[B, on=.(x, y)]
```

But why do we need to do joins as subsets?

# JOIN+UPDATE

Replace A$z where B matches A on cols x and y with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 1 | 2 |
| 2: | 4 | 5 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | NA |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | NA |
| 6: | 4 | 5 | NA |
| 7: | 4 | 10 | 3 |

# JOIN+UPDATE

Replace A$z where B matches A on cols x and y with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 1 | 2 |
| 2: | 4 | 5 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | NA |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | NA |
| 6: | 4 | 5 | NA |
| 7: | 4 | 10 | 3 |

```
A[B, on=.(x, y), z:=NA]
```

# SUBSET+UPDATE

Update col z for all rows where x == 4 & y == 5 with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | NA |
| 6: | 4 | 5 | NA |
| 7: | 4 | 10 | 3 |

```
A[x==4L & y==5L, z:=NA]
```

# THE PROBLEM

For each row in B replace A$z where A$x <= B$x & A$y > B$y with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | NA |
| 4: | 2 | 6 | NA |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | NA |

# NON-EQUI JOIN+UPDATE

For each row in B replace A$z where A$x <= B$x & A$y > B$y with NA

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | NA |
| 4: | 2 | 6 | NA |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | NA |

```
A[B, on=.(x<=x, y>y), z:=NA]
```

# Very briefly, how does it work?

# EXTENSION OF NCLIST

Oxford Journals › Science & Mathematics › Bioinformatics › Volume 23 Issue 11 › Pp. 1386-1393.

## Nested Containment List (NCList): a new algorithm for accelerating interval query of genome alignment and interval databases

Alexander V. Alekseyenko[1] and Christopher J. Lee[2,*]

[1]Department of Biomathematics, David Geffen School of Medicine and [2]Molecular Biology Institute, Center for Computational Biology, Institute for Genomics and Proteomics, Department of Chemistry and Biochemistry, University of California Los Angeles, Los Angeles, CA 90095–1570, USA

*To whom correspondence should be addressed.

Received June 16, 2006.
Revision received December 9, 2006.
Accepted December 18, 2006.

« Previous | Next Article »
Table of Contents

This Article

Bioinformatics (2007) 23 (11): 1386-1393.
doi: 10.1093/bioinformatics/btl647
First published online: January 18, 2007



All Versions of this Article:
btl647v1

# HOW DOES IT WORK?

data.table uses binary search for joins. For non-equi joins, we need to create a special id column based on the columns being joined on.

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

# HOW DOES IT WORK?

data.table uses binary search for joins. For non-equi joins, we need to create a special id column based on the columns being joined on.

A

| | x | y | z |
|---|---|---|---|
| 1: | 1 | 2 | 2 |
| 2: | 2 | 3 | 1 |
| 3: | 2 | 4 | 7 |
| 4: | 2 | 6 | 4 |
| 5: | 4 | 5 | 5 |
| 6: | 4 | 5 | 6 |
| 7: | 4 | 10 | 3 |

| id |
|---|
| 1 |
| 1 |
| 1 |
| 1 |
| 2 |
| 2 |
| 1 |

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

# HOW DOES IT WORK?

On a sorted data.table, within each id, all join columns should be in increasing order, independently. Run binary search for each id.

Combine all matched indices.

| | x | y | z | id |
|---|---|---|---|---|
| 1: | 1 | 2 | 2 | 1 |
| 2: | 2 | 3 | 1 | 1 |
| 3: | 2 | 4 | 7 | 1 |
| 4: | 2 | 6 | 4 | 1 |
| 5: | 4 | 10 | 3 | 1 |

A1

| | x | y | z | id |
|---|---|---|---|---|
| 1: | 4 | 5 | 5 | 2 |
| 2: | 4 | 5 | 6 | 2 |

A2

B

| | x | y |
|---|---|---|
| 1: | 2 | 3 |
| 2: | 4 | 8 |

→ indices from `A1[B]`
indices from `A2[B]`

# PERFORMANCE

| nrow(A) ~= 40m, nrow(B) ~= 33k | | |
|---|---|---|
| **Method** | **Run Time(s)** | **Memory used (GB)** |
| dt-non-equi | 4.9 | 1.2 |
| dt-foverlaps | 4.1 | 1.4 |
| findOverlaps | 6.2 | 2.1 |
| RSQLite | 87.0* | - |

* nrow(A) = 100,000

# THANKS TO

Matt for the ideas on extending the on argument for non-equi joins.

Jan Gorecki for extensive testing and feedback during development.

... and to you for listening.

# ADDITIONAL INFO

Homepage: https://github.com/Rdatatable/data.table/wiki

Try v1.9.7: https://github.com/Rdatatable/data.table/wiki/Installation

Vignettes: https://github.com/Rdatatable/data.table/wiki/Getting-started