`rquery` and `rqdatatable`: `R` tools for data manipulation

John Mount
Win-Vector LLC

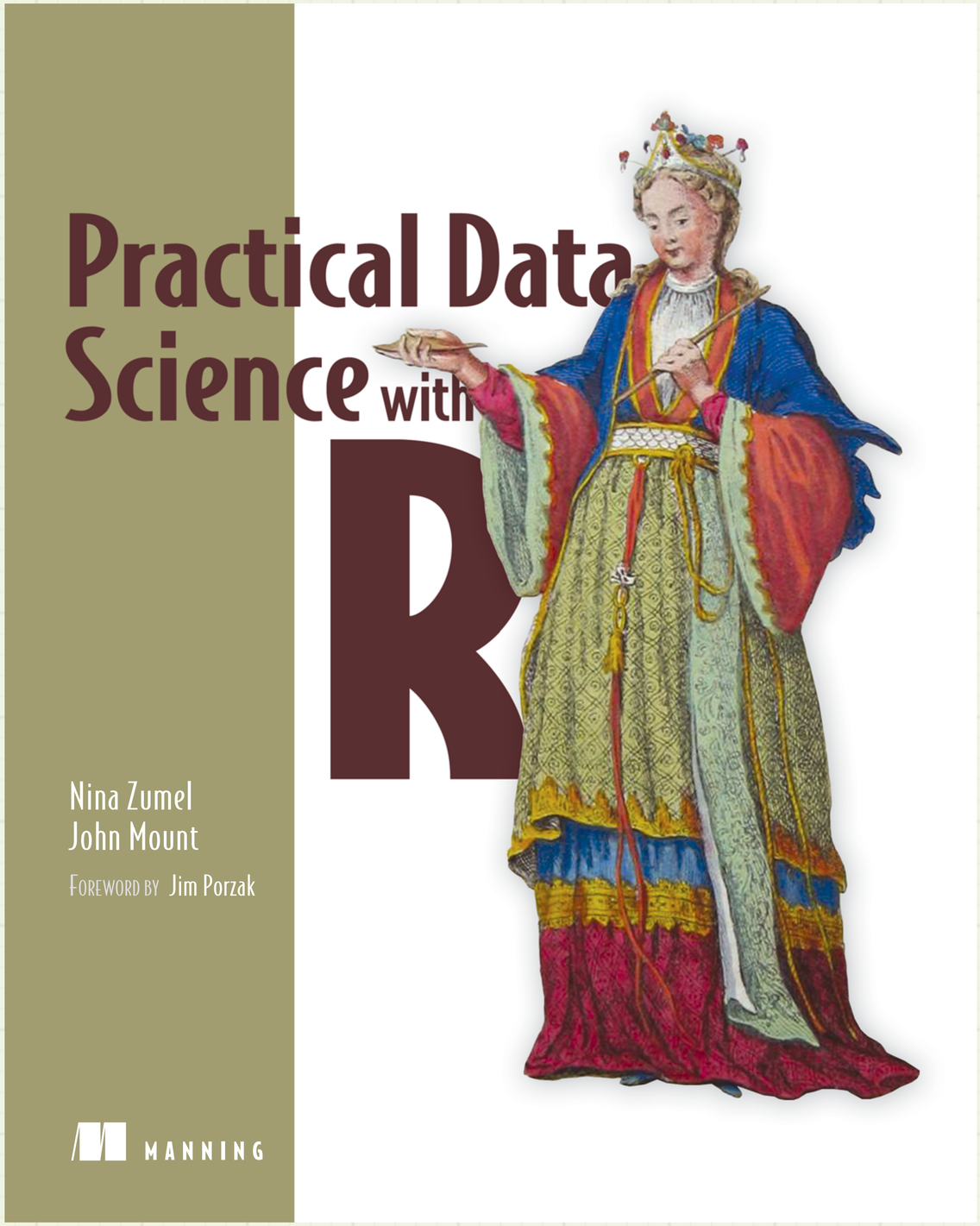
Outline

- Who I am.
- What is the unmet need for **`R`** users working with big data?
- Solution context: a potted history of data manipulation concepts.
- The solution: **`rquery`** and **`rqdatatable`**.
- Performance comparisons.
- Conclusion.

John Mount

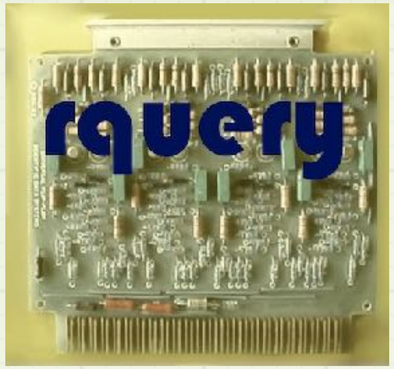
Win-Vector LLC
http://www.win-vector.com/

- One of the authors of the book *Practical Data Science with R*, Zumel Mount, (Manning 2014).
- We (at Win-Vector LLC) provide **`R`**, statistics, and data science training (both live and pre-recorded) and consulting.
- One of the authors of **`vtreat`**: statistically sound preparation of data for predictive modeling.
- Frequent contributor to the Win-Vector blog and conference speaker.



What I Want to Share

- How to wrangle data using the **`rquery`** `SQL` query generator and the **`rqdatatable`** implementation.



Not going to be able to demonstrate/explain *everything*.

If you want to build a ship, don't drum up the [~~men~~ people] to gather wood, divide the work and give orders. Instead, teach them to yearn for the vast and endless sea.
Antoine de Saint-Exupery, "The Wisdom of the Sands"

Win-Vector LLC

R is about ~~statistics~~ ~~programming~~ data

Adapted from “Excursionist Drama 2”, p. 72 of Ben Katchor’s “Julius Knipl Real Estate Photographer”, Little Brown and Company, 1996.

The trouble with data

- You go to a lot of trouble to acquire it, and it gets large and unwieldy.
- One needs tools to work with it at scale.

Michael Pangrazio matte painting of the final warehouse in “Raiders of the Lost Arc” (Paramount Pictures, 1981).

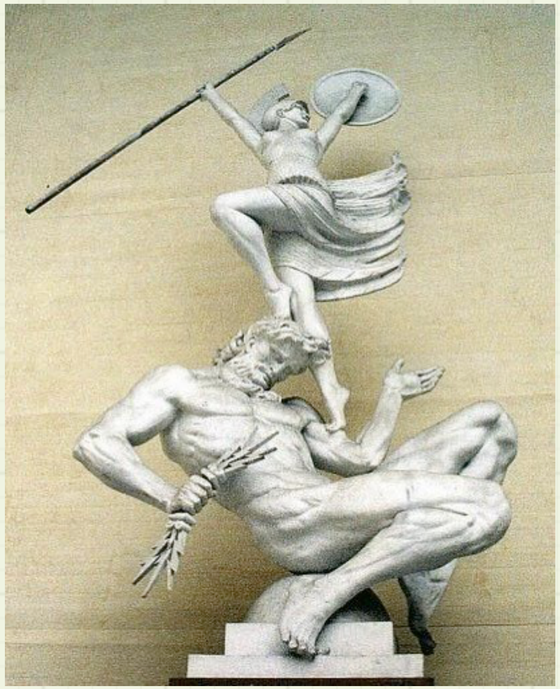
The problem

- Need an **`R`** grammar of data transforms that works well and works the same the same both in-memory and on large data systems (e.g., **`Apache Spark`**).
- Candidates
  - **`SQL`** (run in-memory with **`sqldf`** or by other round-tripping through the data store).
    - Too verbose and hard to maintain.
  - **`dplyr`** / **`dbplyr`**
    - **`dbplyr`** (despite claims) does not work the same as **`dplyr`**.
    - User-facing lazy-evaluation semantics break user expectations and make cross-team development difficult.

The solution

- Go back to influential sources for ideas.
- Implement Codd’s relational algebra in a piped notation.
- Explicitly manage a separation between specification and execution.

Zeus giving birth to Athena, Rudolph Tegner



Athena may have leaped from Zeus's head, fully grown and armed; but even she didn’t appear out of nowhere.

Data Manipulation

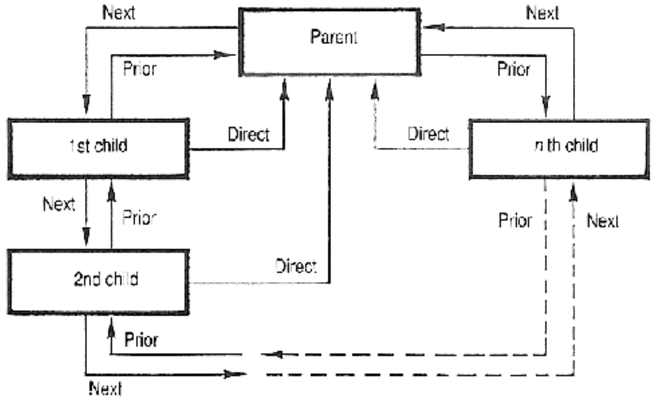
- First big idea: pointer chasing
  - **`CODASYL`** (Conference/Committee on Data Systems Languages, 1959).
  - **`MongoDB`** / NoSQL
  - **`JSON`**
  - **`ORM`** (Object Relational Mapping)

CODASYL

- Data is found by chasing around unidirectional references or pointers.
- Hierarchical or network data model.
- Influential *to this day*.
- Data is navigated by a cursor in an imperative style.

```
OBTAIN CALC CUSTOMER.
PERFORM ORDER-LOOP UNTIL END-OF-SET.
   ORDER-LOOP.
   OBTAIN NEXT ORDER WITHIN CUSTOMER-ORDER.
   MOVE ORDER-NO TO OUT-REC.
   WRITE OUT-REC.
```

From: http://www.dba-oracle.com/data_warehouse/codasyl_generation.html



A closed chain of records in a navigational database model (e.g. CODASYL), with ***next pointers***, ***prior pointers*** and ***direct pointers*** provided by keys in the various records.

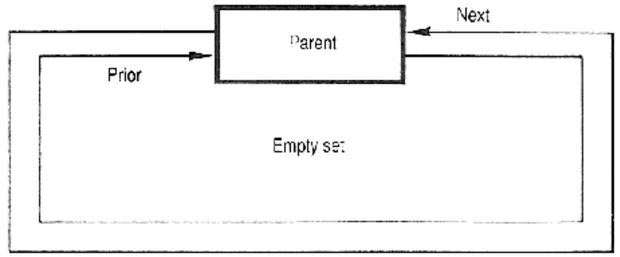


Illustration of an ***empty set***

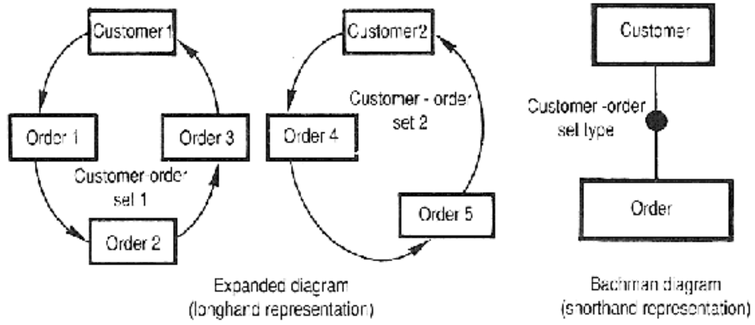
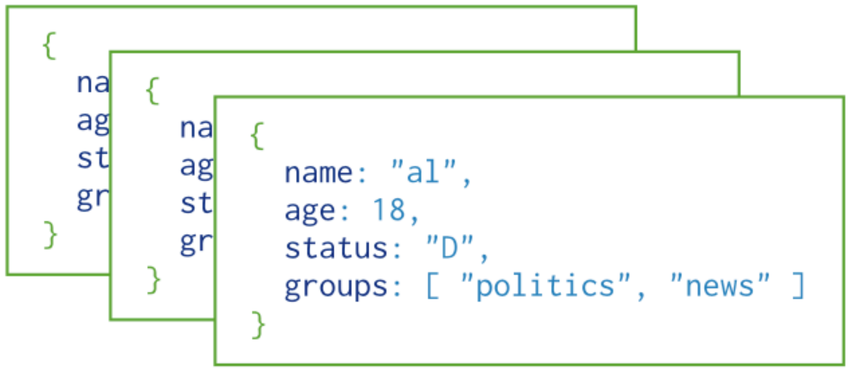


Illustration of a set type using a ***Bachman diagram***

The record set, basic structure of navigational (e.g. CODASYL) database model. A set consists of one parent record (also called "the owner"), and n child records (also called members records)

MongoDB

MongoDB stores BSON documents, i.e. data records, in collections; the collections in databases.



https://docs.mongodb.com/manual/core/databases-and-collections/

JSON

- Nested arrays and maps.

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "office",
      "number": "646 555-4567"
    },
    {
      "type": "mobile",
      "number": "123 456-7890"
    }
  ],
  "children": [],
  "spouse": null
}
```

https://en.wikipedia.org/wiki/JSON

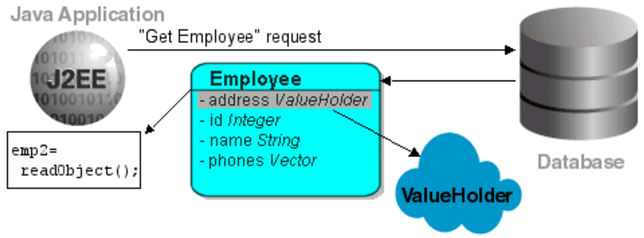
"Do you want *nested for-loops*?! Because that's how you get *nested for-loops*!"

Adapted from Malory Archer, in "Archer" pilot episode.

ORM

Figure 7-9 shows the `Employee` object being read from the database. The `Address` object is not read and will not be created unless it is accessed.

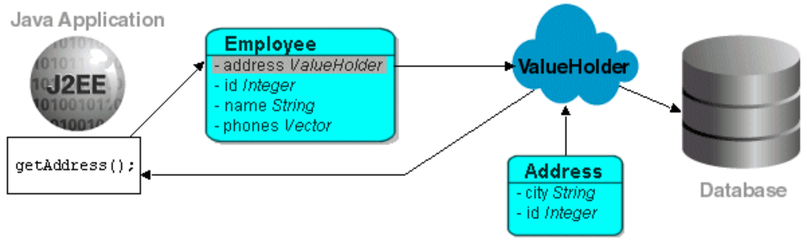
Figure 7-9 Address Object Not Read



Description of "Figure 7-9 Address Object Not Read"

The first time the address is accessed, as in Figure 7-10, the `ValueHolder` reads and returns the `Address` object.

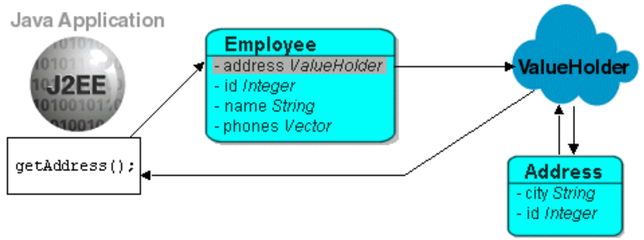
Figure 7-10 Initial Request



Description of "Figure 7-10 Initial Request"

Subsequent requests for the address do not access the database, as shown in Figure 7-11.

Figure 7-11 Subsequent Requests



Description of "Figure 7-11 Subsequent Requests"

http://www.eclipse.org/eclipselink/documentation/2.4/concepts/mappingintro002.htm

"Well that just sounds like pointer chasing with extra steps."

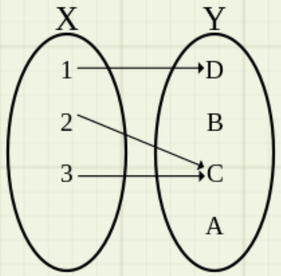
Adapted from: Rick and Morty "The Ricks Must Be Crazy."

A *very* big idea

- Relational algebra
  - Codd, E.F., “A Relational Model of Data for Large Shared Data Banks”, Communications of the ACM 13, (June 1970).
  - Delegate all of your data wrangling tasks to a small set of powerful operators.
  - Example archetypal powerful operator for data scientists: left join (we will return this after deriving the relational model).

Abstract and Generalize

- Pointers are an example of functions.



https://en.wikipedia.org/wiki/Function_(mathematics)

- Functions can be written as tables with the condition that the domain column has unique entries.

domain	range
1	D
2	C
3	C

- Relax the table conditions to have merely every row be unique (or a set of rows) and you have what is called a relation.

X	Y
1	D
2	C
3	C
ω	A
ω	B

Same data structure (table) can be used to represent data or relations between tables!

ω (lower-case omega) represents no-value, similar to **`NULL`**, **`NA`**, or, ⊥.

Win-Vector LLC

An entire theory of data wrangling

- Codd could define complex operators as equivalent to a sequence of simpler operators.
  - Example: in Codd's theory the full outer join (⟗) is defined as:

    R ⟗ S := (R ⟕ S) ∪ (R ⟖ S).

    (where ⟕ and ⟖ are the left and right joins respectively).
- Could also prove different re-arrangements of operators were formally equivalent.
  - Basis for optimizing query planners to this day.
    - Example: pivoting a row selection prior to an expensive operation such as a join.

The realization

- **`SQL`** (Structured query language, 1974).
- *Far* better system than the ugly syntax would suggest.
- Essentially made Oracle Corporation
  - Oracle released a commercial **`SQL`** offering in 1979. This is shortly after IBM's **`System R`** (first customer: Pratt & Whitney in 1977) and *before* IBM's general commercial offering: **`DB2`** (1983).

SQL

- Further relaxed the mathematics from tables that represent sets of rows to arbitrary tables (collection of rows now a "bag" or "multiset").
    - Improves the ability to represent data and makes some operations faster.
        - Don't have to de-duplicate rows.
    - Makes joins more confusing.
        - Joins longer defined in terms of set-operations such as union.
            - Invites tons of ugly questions on how many rows each join situation generates.
        - No longer the case that all tables are relations.
    - Some optimizations no longer possible due to some theorems not carrying over.
        - Example failing theorem: distributive law of intersection over union fails

          $$R \cap (S \cup T) \neq (R \cap S) \cup (R \cap T)$$

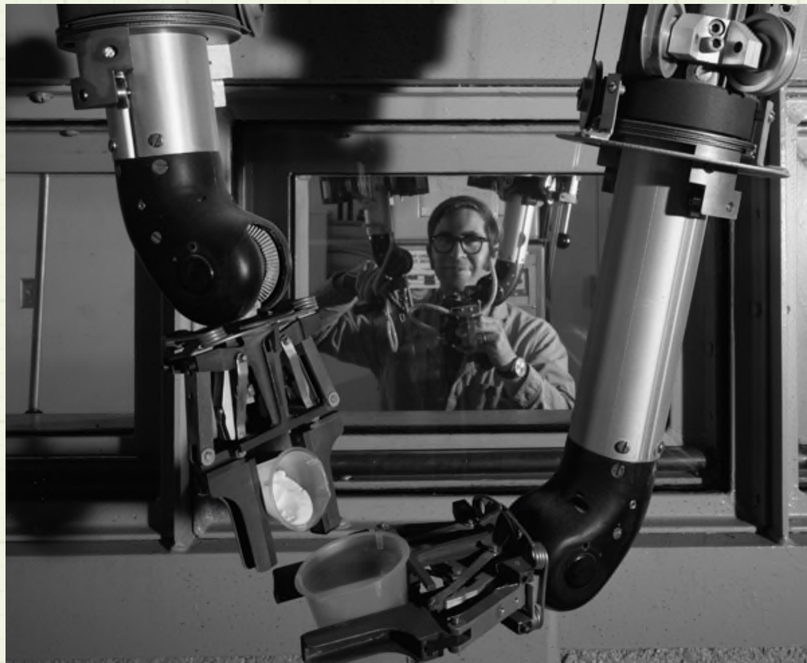
          (mostly due to "∪" being re-defined as essentially `rbind()`).

- Good description of the extended theory: Hector Garcia-Molina; Jeffrey D. Ullman; Jennifer Widom, *Database systems: the complete book (2nd ed.),* Pearson Prentice Hall, (2009).

Relational Databases

- Essentially known by the join operation.
    - In a relational database you assume you have an effective join operator and write many other tasks in terms of join (and other steps).
- Declarative.
    - Instead specifying how to walk through the data the user declares the desired transformation. Like working with thick gloves on, awkward but safe and stong.
- For data scientists the natural left-join is *very* useful.
    - Example task: add columns from a new table **S** to matching rows in our left table **R**.

https://www.jpl.nasa.gov/blog/2012/7/remote-controlled-manipulators



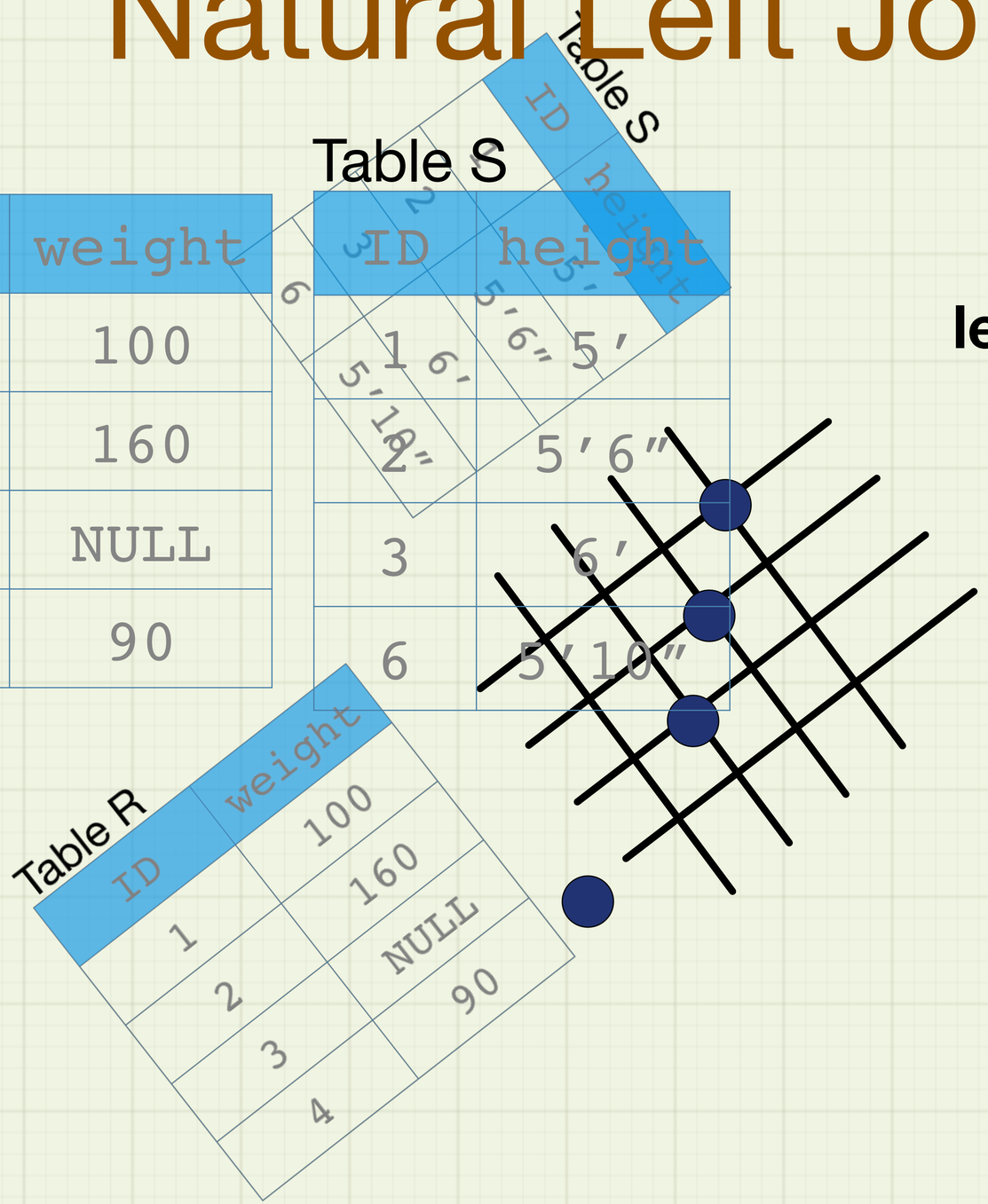
Natural Left Join Example

Table R

ID	weight
1	100
2	160
3	NULL
4	90

Table S

ID	height
1	5′
2	5′6″
3	6′
6	5′10″



left_join(R, S)

ID	weight	height
1	100	5′
2	160	5′6″
3	NULL	6′
4	90	**NULL**

joins

- Joins replace for-loops and pointer chasing.
- We now think in bulk: annotate all rows with this new column (instead of for each row find an annotation).

Codd relational operators

- Small set of operators that most data-wrangling tasks can be decomposed into.
    - **`join`** (we just saw)
    - **`project/aggregate`** (delete columns and also produces group-summaries).
    - **`extend`** (add new calculated columns such as **`x+y`**).
    - **`select_rows`** (take a subset of rows based on a criterion).
- Complex transforms are expressed as a sequence of simpler transforms.
- Notable gaps:
    - Window-functions (such as ranking or grouped ranking).
        - Standardized in **`SQL92`**, commonly available.
    - Transitive closure / graph reachability.
        - Part of **`SQL3`** (1999), mostly still non-standard and not usually offered as part of non-graph databases.
    - Common table expressions and correlated sub-queries.

rquery

- **rquery** is a grammar for data wrangling based on Codd's relational algebra and experience working with **SQL** and **dplyr** at big data scale.
  - **rquery** uses the **wrapr** "dot arrow" to supply legible left to right pipe notation.
    - Consider `x %.>% f(.)` as an approximate synonym for `f(x)`.
    - Doesn't seem like much but it turns out `x %.>% f(.) %.>% g(.)` is a easier to build up piece by piece than `g(f(x))` (and doesn't require reading backwards).
  - **rquery** is primarily a **SQL** query generator.
  - **rquery** depends on external systems (such as **SparklyrR**, **SparkR**, **PostgreSQL**) for implementation.
- **rqdatatable** is an in-memory implementation of the **rquery** grammar based on **data.table**.

What do we mean by **SQL** query generation?

Work through

rquery/extras/NarrowExample.Rmd

Rosetta Stone

Details on pipe-ready **R**-equivalents at https://github.com/WinVector/wrapr/blob/master/extras/pipe_base.R

Primary **dplyr** verbs can be found at https://dplyr.tidyverse.org

relational algebra	SQL	rquery	R / data.table	dplyr
π project (column restriction)	SELECT expressions	select_columns()	.[, cols, drop = FALSE]	select()
aggregation	GROUP BY	project_nse() / project_se()	tapply() / aggregate() / "j=" (data.table)	group_by() summarize() ungroup()
extend (extended projection)	SELECT expressions	extend_nse() / extend_se()	transform() / := (data.table)	mutate()
	ORDER BY	orderby()	.[order(), , drop = FALSE]	arrange()
σ select (row restriction)	WHERE	select_rows_nse() / select_rows_se()	subset()	filter()
$\ltimes$ (left outer join)	LEFT JOIN	natural_join(jointype = "LEFT")	merge(, all.x = TRUE)	left_join()

The **rquery *_nse()** forms are the expected expression capturing interfaces, and the ***_se()** are the value oriented interfaces needed for effective abstraction and programming.
An example of the benefit can be found here https://github.com/WinVector/rquery/blob/master/extras/CollectExprs.md .

Win-Vector LLC

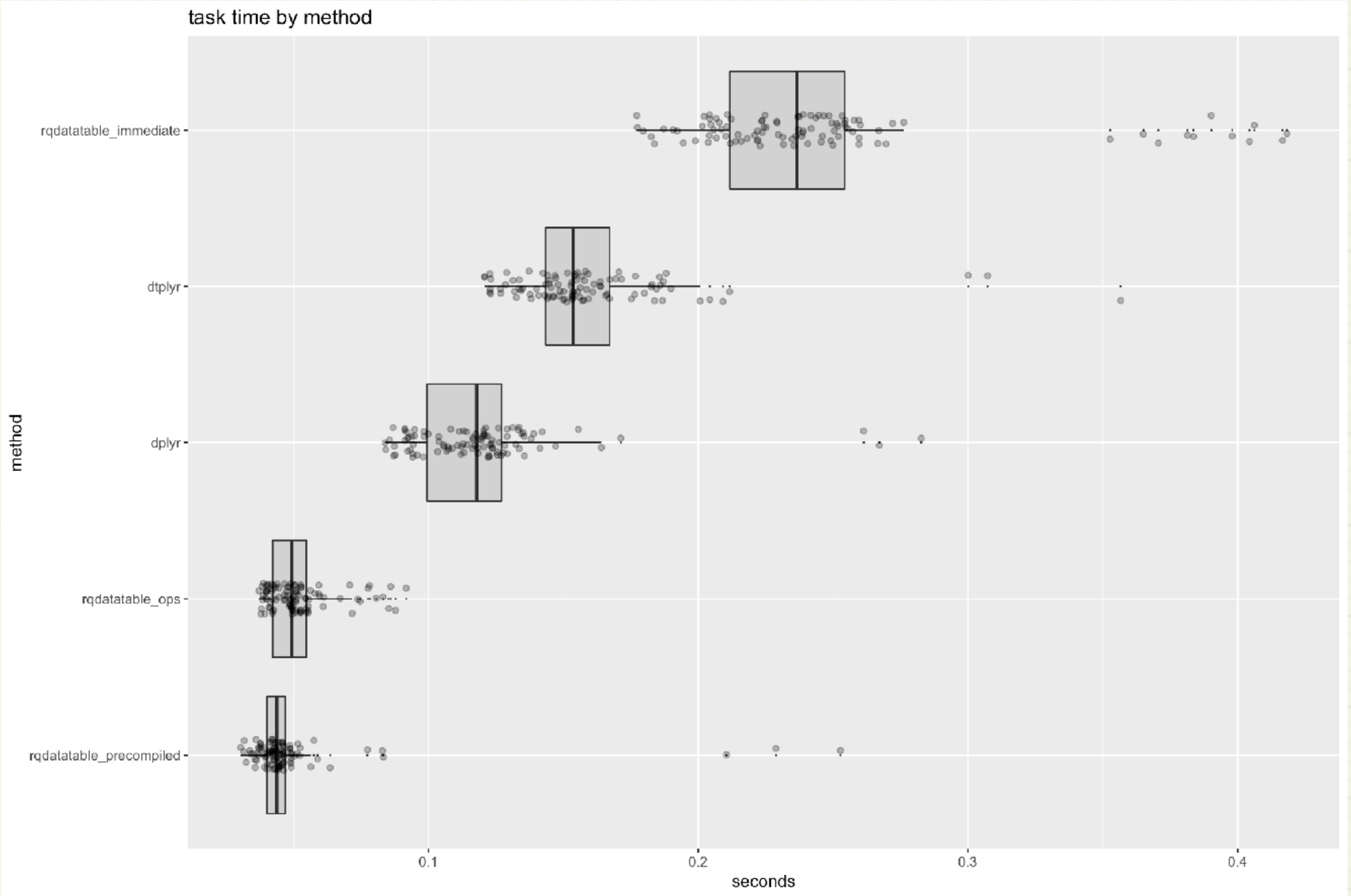
A substantial example

Work through

`rquery/extras/IrisExample.Rmd`

Immediate Mode

- **`rqdatatable`** in immediate mode is myopic (only can see one stage at a time) and fighting to bridge the difference between **`data.table`** reference semantics and expected **`R`** value semantics.
    - Fully avoidable by building an **`rquery`** operator tree object and then piping data into that object.
- **`dtplyr`** documents having similar issue (though no current way to avoid it).

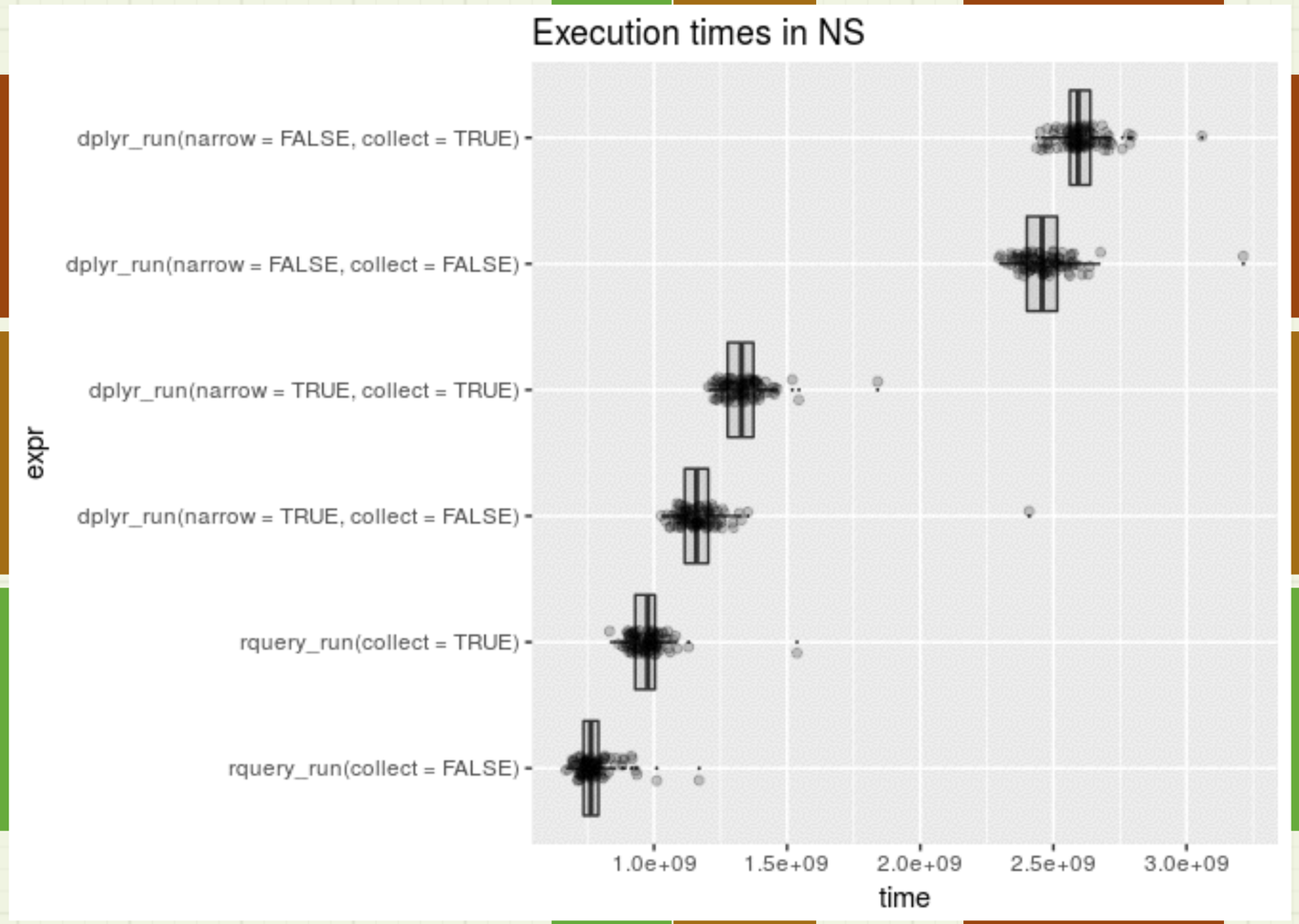


https://github.com/WinVector/rqdatatable/blob/master/extras/ImmediateIssue.md

Win-Vector LLC

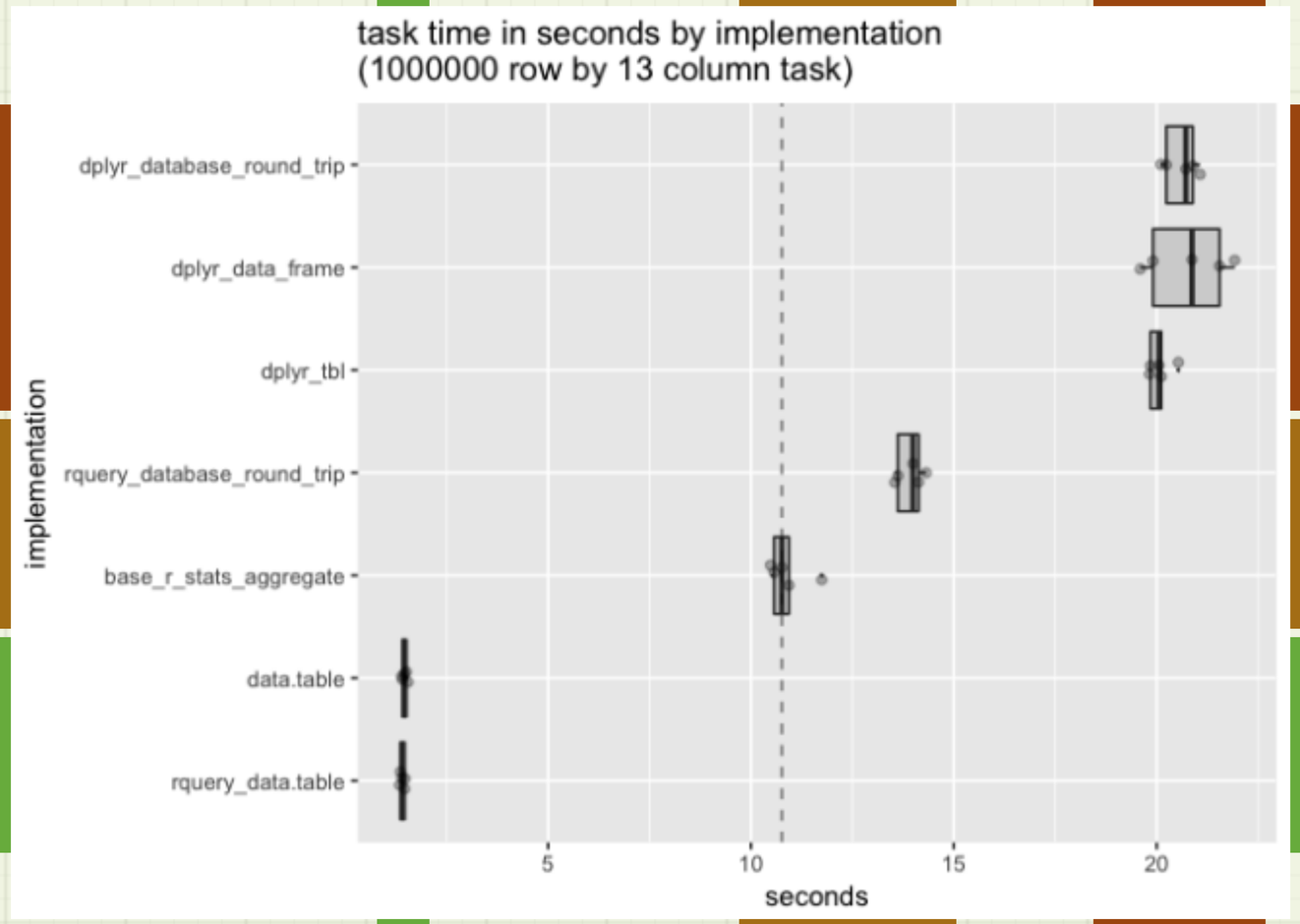
More Results

Performance (`sparklyr`, 40000 rows 1003 columns)



https://github.com/WinVector/rquery/blob/master/extras/PerfTest.md

rqdatatable



Notice *both* **`rquery`** database round-trip *and* base **`R`** are much faster than **`dplyr`**. This is common, but contrary to many unfounded claims.

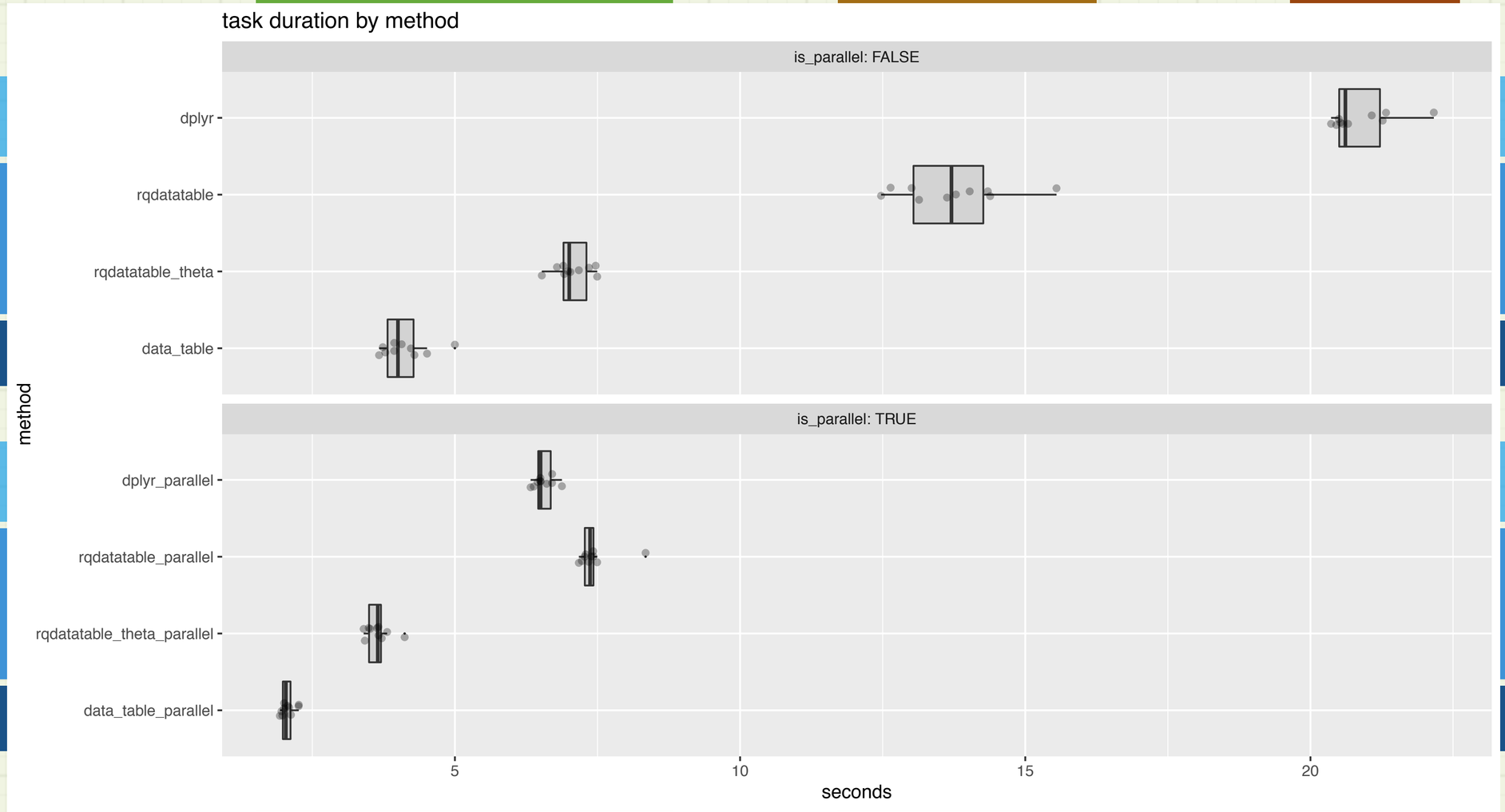
Notice *nothing* prior to these rows is in fact fast. Both these results are due to **`data.table`**.

rquery_data.table == rqdatatable, database is **`PostgreSQL`**

http://www.win-vector.com/blog/2018/06/rqdatatable-rquery-powered-by-data-table/

Win-Vector LLC

wrapr::execute_parallel()!!



Not shown: **multidplyr** and **dtplyr**, as they both error-out for this example.

http://www.win-vector.com/blog/2018/07/speed-up-your-r-work/

Win-Vector LLC

rquery on SparkR (with DataBricks)!

databricks

PRODUCT APACHE SPARK SOLUTIONS CUSTOMERS TRAINING EVENTS TRY DATABRICKS

Search Blog

COMPANY BLOG

Announcements
Customers
Events
Partners
Product
Security

ENGINEERING BLOG

Apache Spark
Ecosystem
Machine Learning
Platform
Streaming

SEE ALL

rquery: Practical Big Data Transforms for R-Spark Users

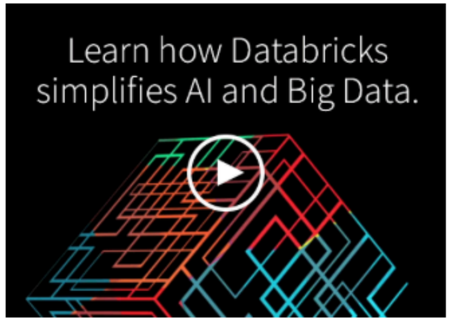
How to use rquery with Apache Spark on Databricks

by Nina Zumel and John Mount
Posted in **ENGINEERING BLOG** | July 26, 2018

This is a guest community blog from Nina Zumel and John Mount, data scientists and consultants at Win-Vector. They share how to use rquery with Apache Spark on Databricks

Try this notebook in Databricks

SHARE POST



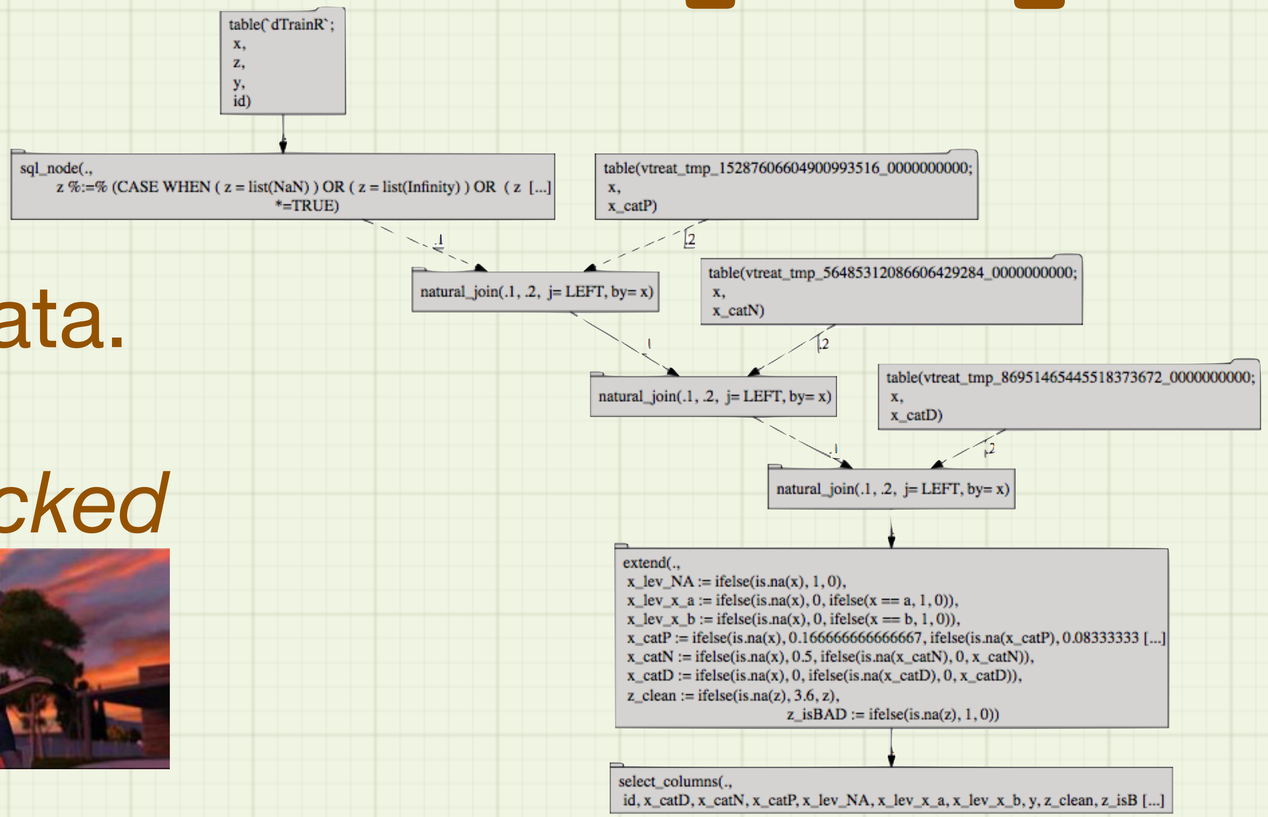
Introduction

In this blog, we will introduce rquery, a powerful query tool that allows R users to implement powerful data transformations using Apache Spark on Databricks. *rquery* is based on Edgar F. Codd's relational algebra, informed by our experiences using SQL and R packages such as *dplyr* at big data scale.

https://databricks.com/blog/2018/07/26/rquery-practical-big-data-transforms-for-r-spark-users.html

Win-Vector LLC

`vtreat` hosted on `rquery`



- Enables `vtreat` on big data.
- Makes for some *totally wicked* op diagrams.

- https://github.com/WinVector/vtreat/blob/master/extras/rquery_vtreat.md
- https://github.com/WinVector/vtreat/blob/master/extras/vtreatOnSpark.md

Conclusion

- **`rquery`** is an excellent query generator for **R** in terms of performance and usability. Using it can increase your team's productivity on **`R`** projects.
    - **`rquery`** is a best of breed solution in terms of:
        - Error Checking
        - Correctness
        - Usability
        - Performance.
    - Building up experience with it mostly with **`PostgreSQL`** and **`Spark`**.
- **`rqdatatable`** is a fast in-memory realization of **`rquery`** supplied by **`data.table`**.
- I would *love* to explore ways to collaborate and get further introductions.
    - **Please** reach out to me at **`jmount@win-vector.com`** . I would especially like to meet with groups considering working with **`R`** and databases or **`Spark`**.

Thank You