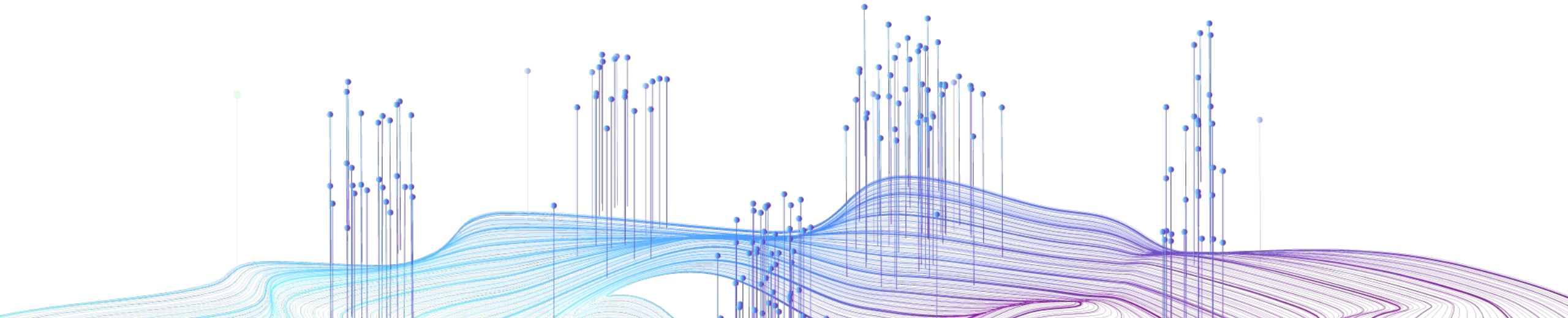
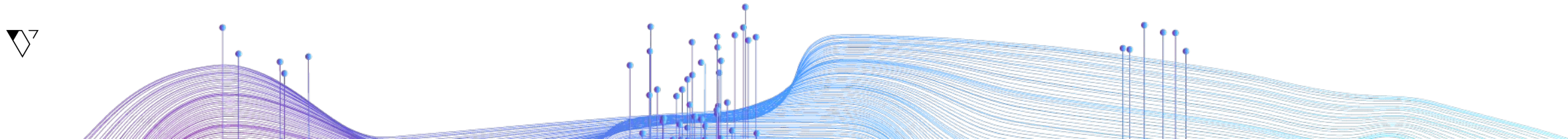
VOLTRON DATA

tidyquery and queryparser: Translating SQL Queries to dplyr Pipelines

Ian Cook
@ianmcook
ian@voltrondata.com
@ianmcook



dplyr ~ SQL



✅ Each variable forms a column

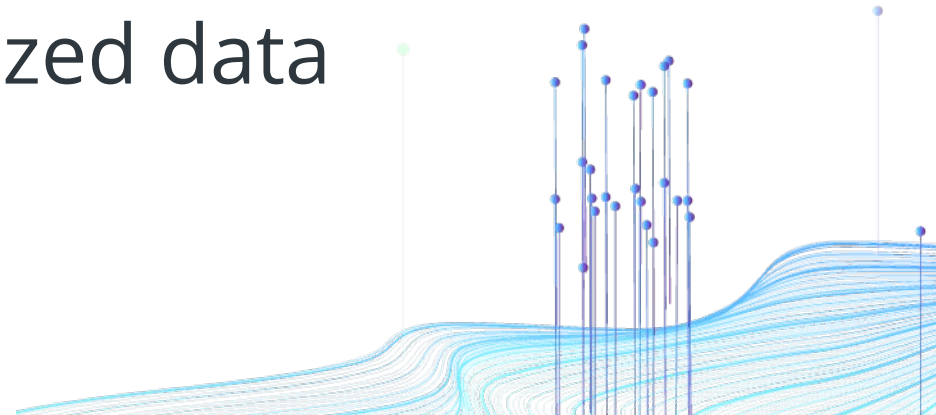
✅ Each observation forms a row

✅ Each value has its own cell

✅ Each type of observational unit forms a table

☝️ Tidy data

☝️ Normalized data

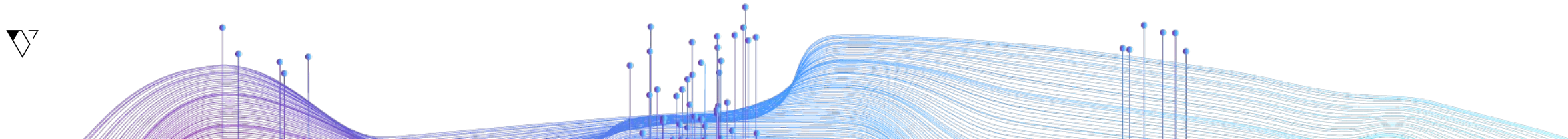


`SELECT`	→	`select()` for columns `mutate()` for expressions `summarise()` for aggregates
`FROM`	→	which data frame
`WHERE`	→	`filter()`
`GROUP BY`	→	`group_by()`
`HAVING`	→	`filter()` after `group_by()`
`ORDER BY`	→	`arrange()`
`LIMIT`	→	`head()`

VOLTRON DATA

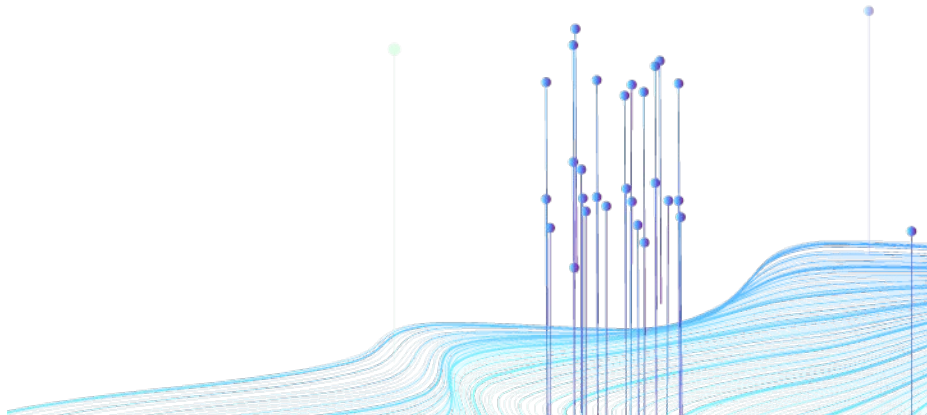
dbplyr

dplyr → SQL



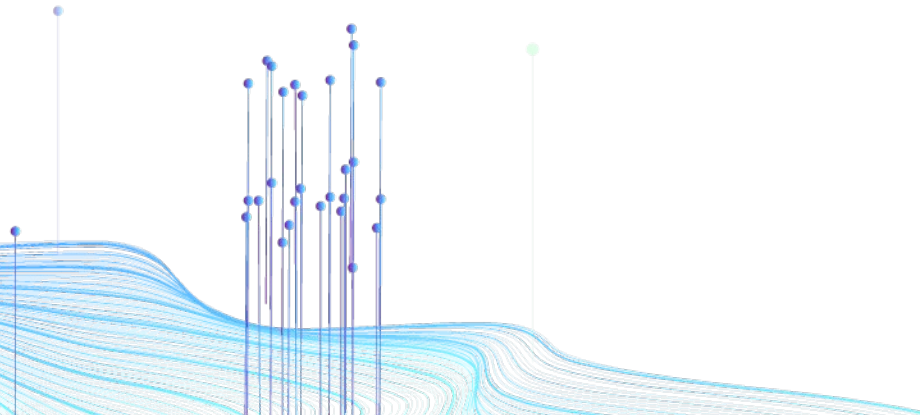
```
library(nycflights13)
library(dplyr)

flights_db <- dbplyr::tbl_memdb(flights, name = "flights_db")

flights_db %>%
  group_by(carrier) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  head(3)
```

```
#> # Source:     lazy query [?? x 2]
#> # Database:   sqlite 3.33.0 [:memory:]
#> # Ordered by: desc(n)
#>   carrier     n
#>   <chr>   <int>
#> 1 UA      58665
#> 2 B6      54635
#> 3 EV      54173
```



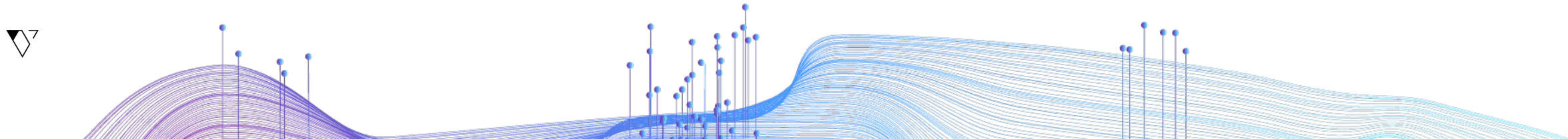
```
flights_db %>%
  group_by(carrier) %>%
  summarise(n = n()) %>%
  arrange(desc(n)) %>%
  head(3) %>%
  show_query()
```

```
#> <SQL>
#> SELECT `carrier`, COUNT(*) AS `n`
#> FROM `flights_db`
#> GROUP BY `carrier`
#> ORDER BY `n` DESC
#> LIMIT 3
```



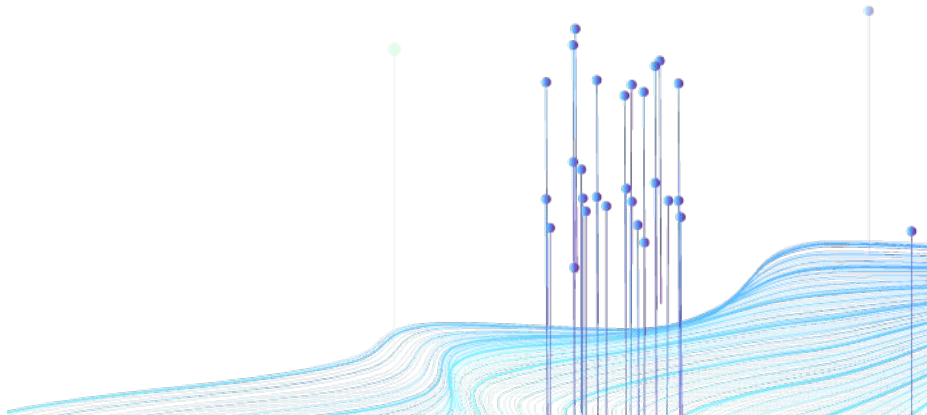
VOLTRON DATA

SQL → dplyr ?



Translate SQL queries into unevaluated R expressions

```
library(queryparser)
parse_query("SELECT carrier, AVG(arr_delay) AS delay FROM flights GROUP BY carrier")

#> $select
#> $select[[1]]
#> carrier

#> $select$delay
#> mean(arr_delay, na.rm = TRUE)

#> attr(,"aggregate")
#> FALSE  TRUE

#> $from
#> $from[[1]]
#> flights

#> $group_by
#> $group_by[[1]]
#> carrier

#> attr(,"aggregate")
#> [1] TRUE
```



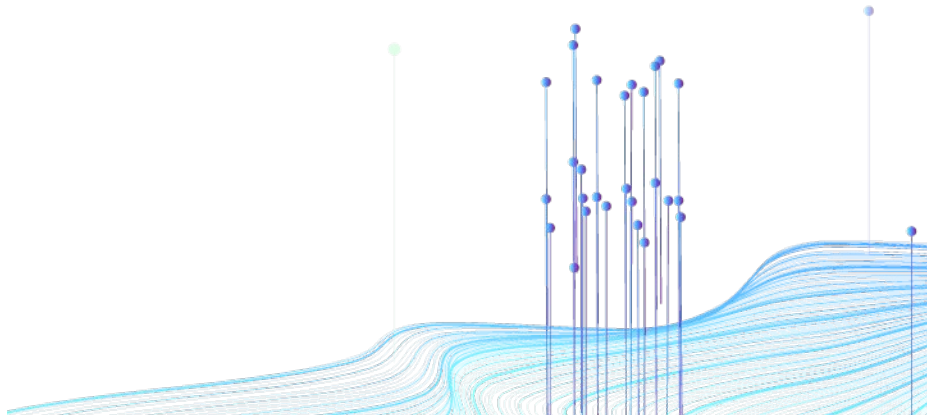
queryparser

- Has no dependencies; implemented as pure R code
- Includes an option to use tidyverse functions in translated R expressions

```
parse_query("SELECT COUNT(*) FROM flights",  tidyverse = TRUE)
#> $select
#> $select[[1]]
#> dplyr::n()
```

- Performs semantic (not literal) translations

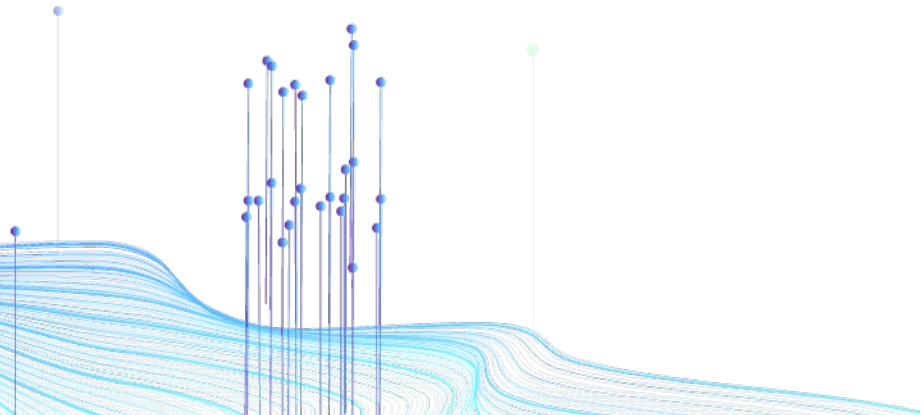
```
parse_query("SELECT SUM(1) FROM flights",  tidyverse = TRUE)
#> $select
#> $select[[1]]
#> dplyr::n() * 1
```



queryparser

- Is secure by default

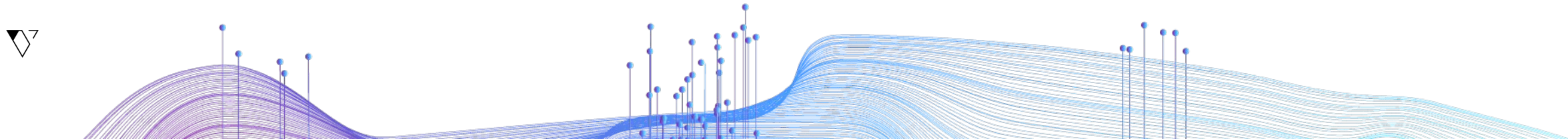
```
parse_query("SELECT * FROM flights WHERE system('rm -rf /')")
#> Error: Unrecognized function or operator: system
```



VOLTRON DATA

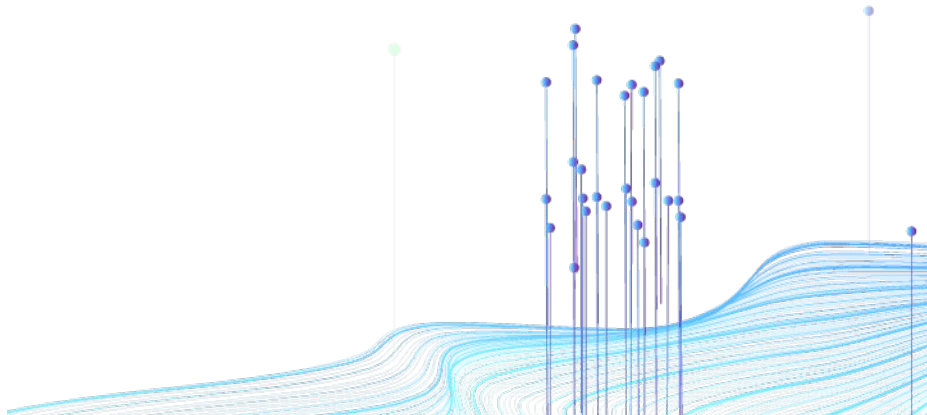
How queryparser works

1. Split query into clauses
2. Split clauses into strings containing SQL expressions
3. Manipulate strings to change from SQL expressions to R-like expressions
4. Parse the R-like expressions with `str2lang()`
5. Quote, substitute, unquote, deparse, manipulate strings, and reparse
6. Manipulate the AST
7. Check for validity
8. Return list



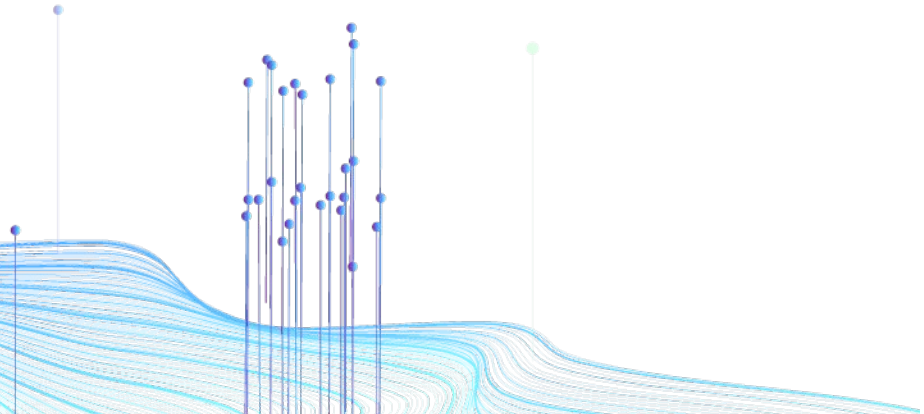
Query R data frames with SQL SELECT statements

```
library(nycflights13)
library(tidyquery)

query("
  SELECT carrier, COUNT(*) AS n
    FROM flights
    GROUP BY carrier
    ORDER BY n DESC
    LIMIT 3
")

#> # A tibble: 3 x 2
#>   carrier     n
#>   <chr>   <int>
#> 1 UA      58665
#> 2 B6      54635
#> 3 EV      54173
```



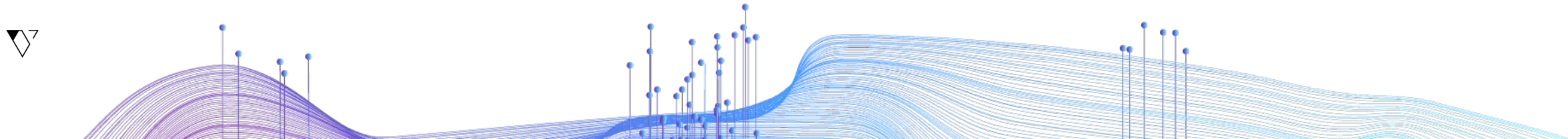
```
planes %>%
  filter(engine == "Turbo-fan") %>%
  query("SELECT manufacturer AS maker, COUNT(*) AS num_planes GROUP BY maker") %>%
  arrange(desc(num_planes)) %>%
  head(5)
```

```
#> # A tibble: 5 x 2
#>   maker             num_planes
#>   <chr>                  <int>
#> 1 BOEING                  1276
#> 2 BOMBARDIER INC           368
#> 3 AIRBUS                   331
#> 4 EMBRAER                  298
#> 5 AIRBUS INDUSTRIE         270
```



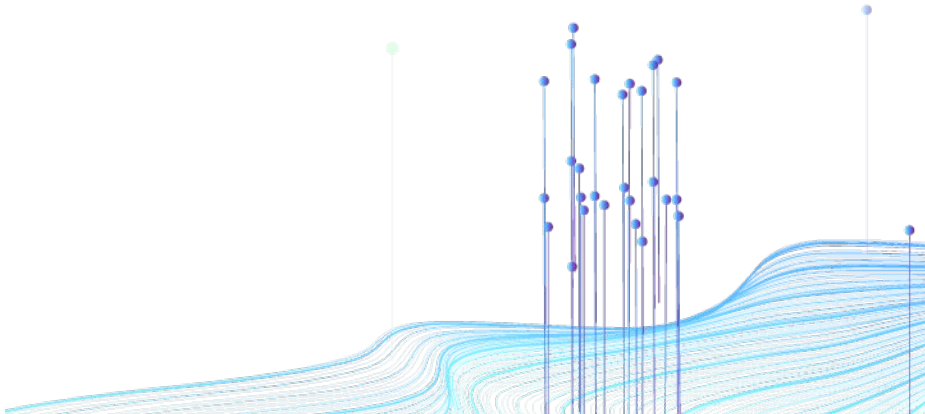
VOLTRON DATA

```
show_dplyr("
  SELECT carrier, COUNT(*) AS n
    FROM flights
    GROUP BY carrier
    ORDER BY n DESC
    LIMIT 3
")

#> flights %>%
#>   group_by(carrier) %>%
#>   summarise(n = dplyr::n()) %>%
#>   ungroup() %>%
#>   arrange(dplyr::desc(n)) %>%
#>   head(3)
```



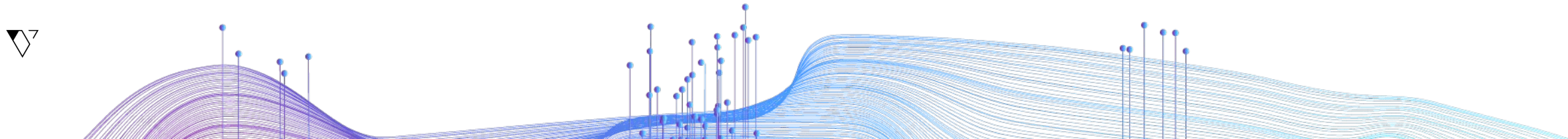
```
query("
  SELECT origin, dest,
      COUNT(flight) AS num_flts,
      round(SUM(seats)) AS num_seats,
      round(AVG(arr_delay)) AS avg_delay
    FROM flights f LEFT OUTER JOIN planes p
      ON f.tailnum = p.tailnum
    WHERE distance BETWEEN 200 AND 300
      AND air_time IS NOT NULL
    GROUP BY origin, dest
    HAVING num_flts > 3000
    ORDER BY num_seats DESC, avg_delay ASC
    LIMIT 2
")
```

```
#> # A tibble: 2 x 5
#>   origin dest  num_flts num_seats avg_delay
#>   <chr>  <chr>    <int>     <dbl>     <dbl>
#> 1 LGA    DCA       4468    712643         6
#> 2 EWR    BOS       5247    611192         5
```



How tidyquery works

1. Call `queryparser::parse_query()`
2. Determine which sequence of dplyr verbs to call
3. Use dplyr to evaluate the expressions in the context of the data frame



Current limitations

queryparser and tidyquery do not yet support:

- Subqueries
- Unions
- SQL-89-style (implicit) join notation
- Joins of three or more tables
- WITH clause (common table expressions)
- OVER expressions (window or analytic functions)