RStudio®

Big Data with R

Nick Rohrbaugh
RStudio Customer Success
nick@rstudio.com

Thanks to James Blair @ RStudio for slides and materials!

Overview

1. Big Data
2. dplyr
3. dplyr + friends
4. Best practices
5. Resources

Photo by Mat Reding on Unsplash

Data > RAM

Photo by James Hammond on Unsplash

Big data forms

1. Flat file
    a. csv
    b. parquet
    c. tsv
    d. json
2. Database
3. Datalake

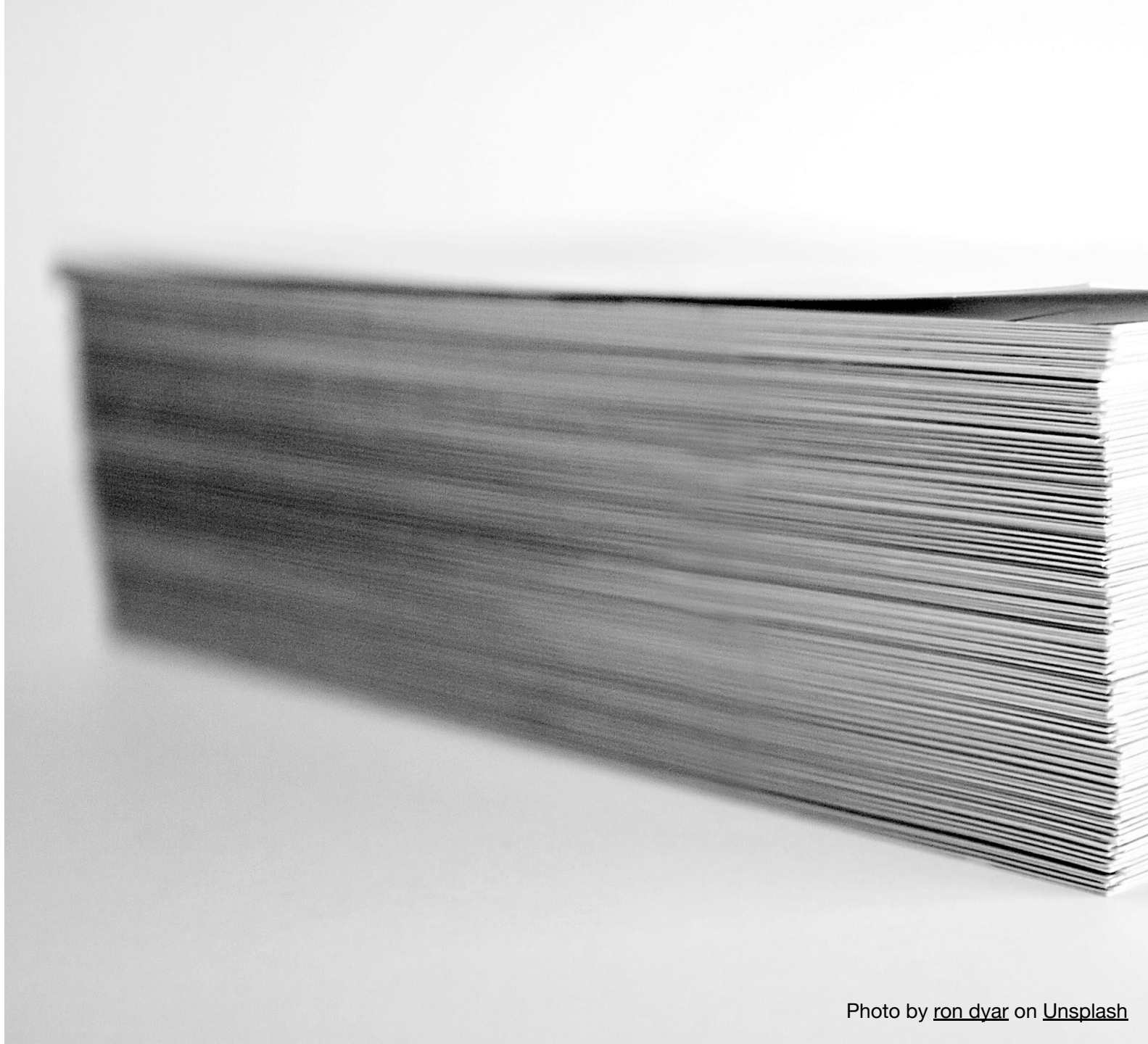
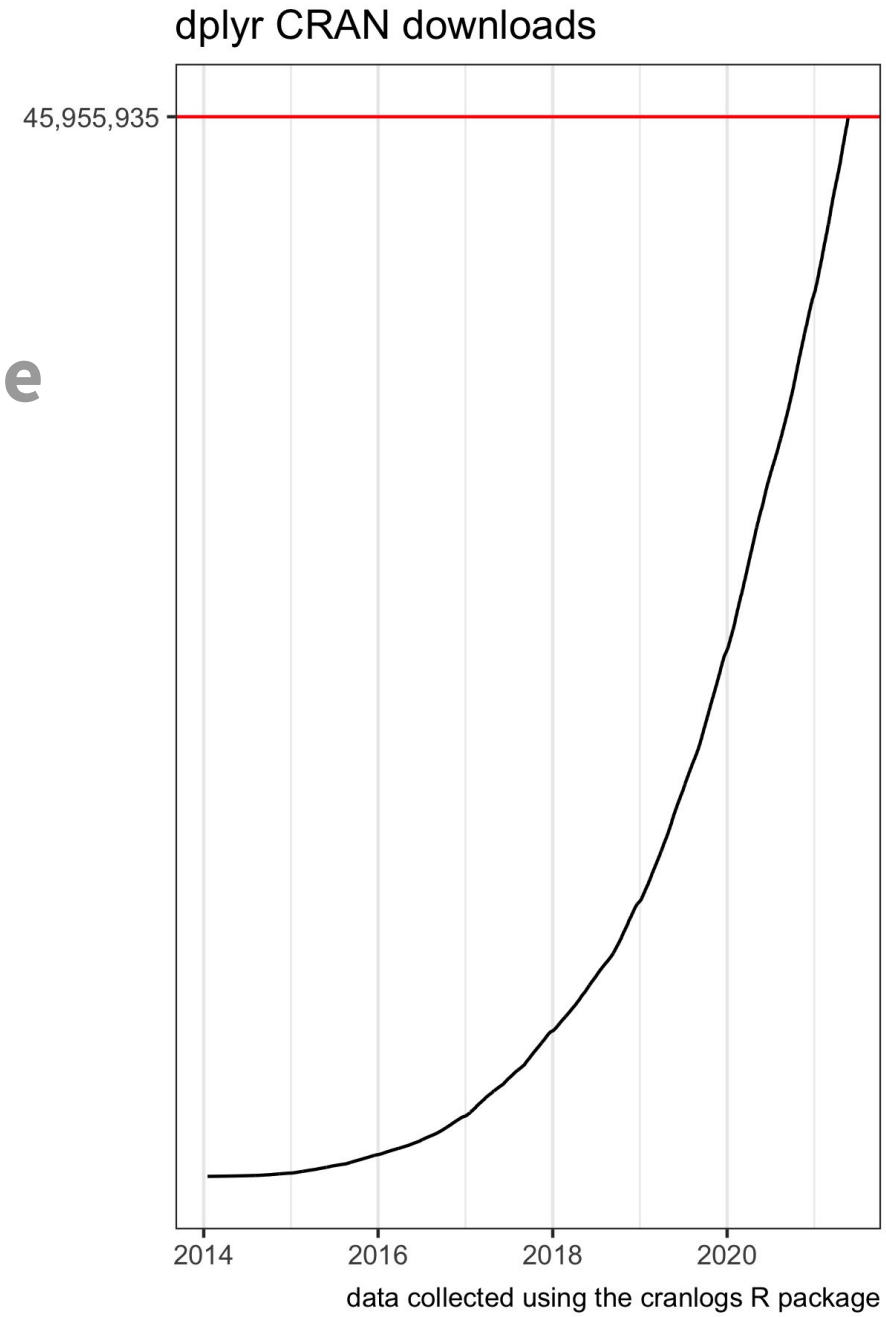


Photo by ron dyar on Unsplash

`dplyr` package

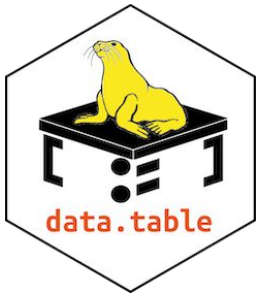
1. A grammar of data manipulation
2. Designed to **abstract over how the data is stored**
3. Consistent function interface



`dplyr` package

Dplyr "abstracts away how your data is stored, so that you can work with data frames, data tables and remote databases using the same set of functions. ***This lets you focus on what you want to achieve, not on the logistics of data storage.***"

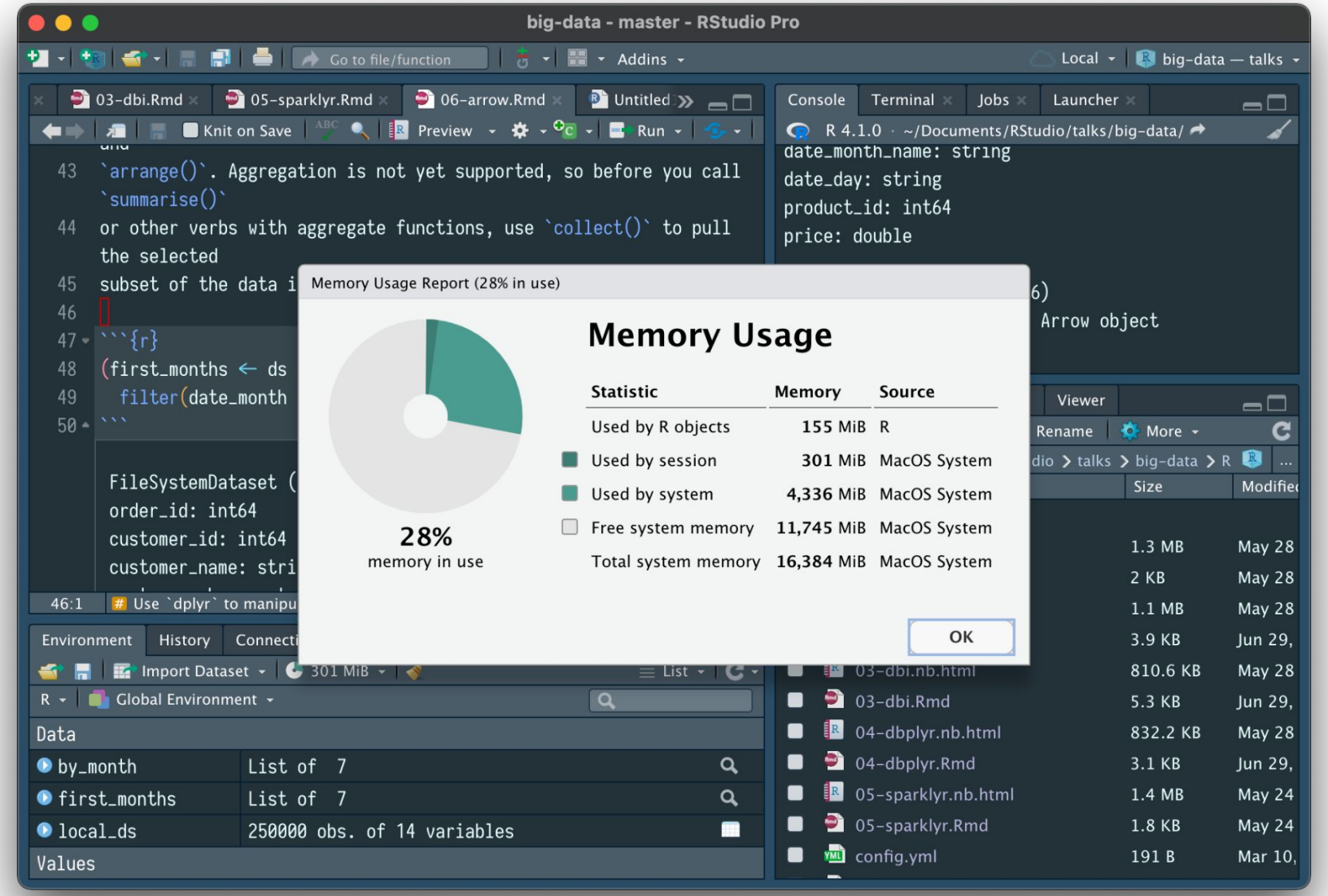
dplyr backends



Memory management

- In most cases, modifying an object in R creates a copy of that object
- Recommended to have 4x largest object of available RAM
- Solutions:
  - Minimize unnecessary object copies
  - Work with data outside of memory

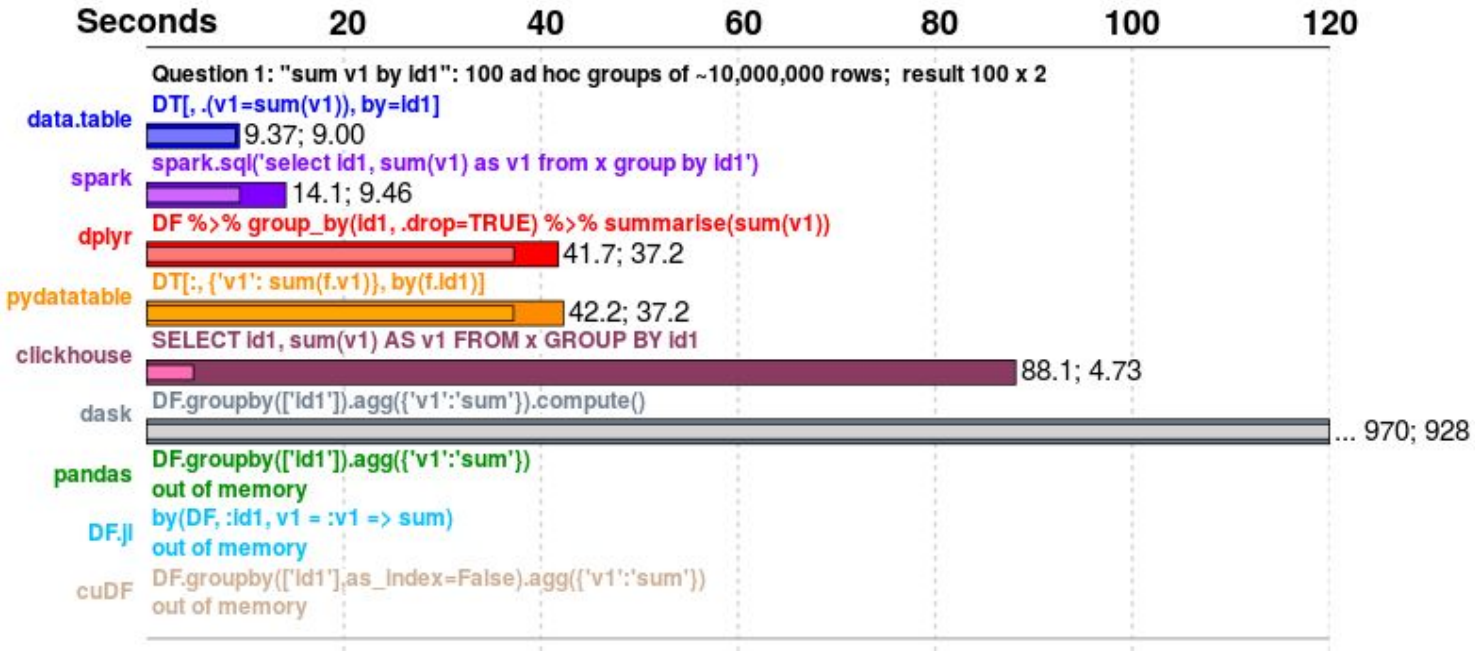
Memory management



RStudio

`data.table` package

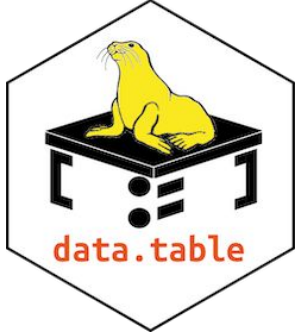
1. High performance version of base R `data.frame`
2. Fast file reader `fread`
3. Concise syntax `DT[i, j, by]`



rdatatable.gitlab.io/data.table

`dtplyr` package

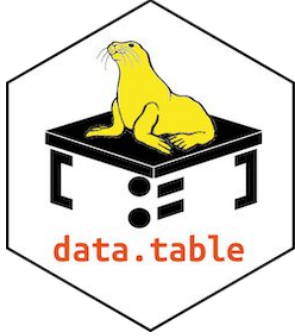
+



The goal of `dtplyr` is to allow you to write `dplyr` code that is automatically translated to the equivalent, but usually much faster, data.table code.

`dtplyr` package

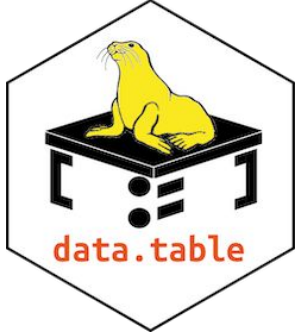
+



1. Provides a `data.table` backend for `dplyr`
2. Combine the syntax of `dplyr` with the speed of `data.table`
3. Lazy evaluation
4. Converts `dplyr` syntax to `data.table` syntax

`dtplyr` package

+



A word about copying…

In `data.table` parlance, all `set*` functions change their input by reference. That is, no copy is made at all, other than temporary working memory, which is as large as one column.

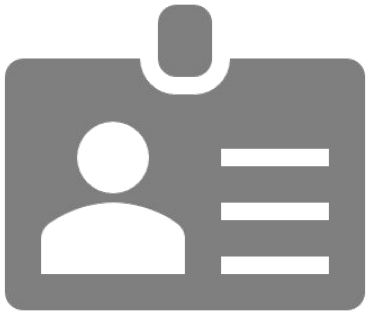
Use `lazy_dt(x, immutable = FALSE)` to prevent `dtplyr` from making copies.

`arrow` package

- High-performance reading and writing of data files with multiple file formats and compression codecs, including built-in support for cloud storage
- **Analyzing and manipulating bigger-than-memory data with dplyr verbs**
  - *Summarization functions are not currently supported*

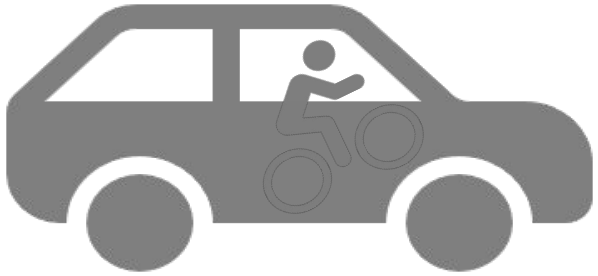
Databases

Connection requirements



Credentials

Location

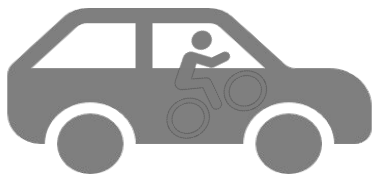


Driver

Requirement definitions

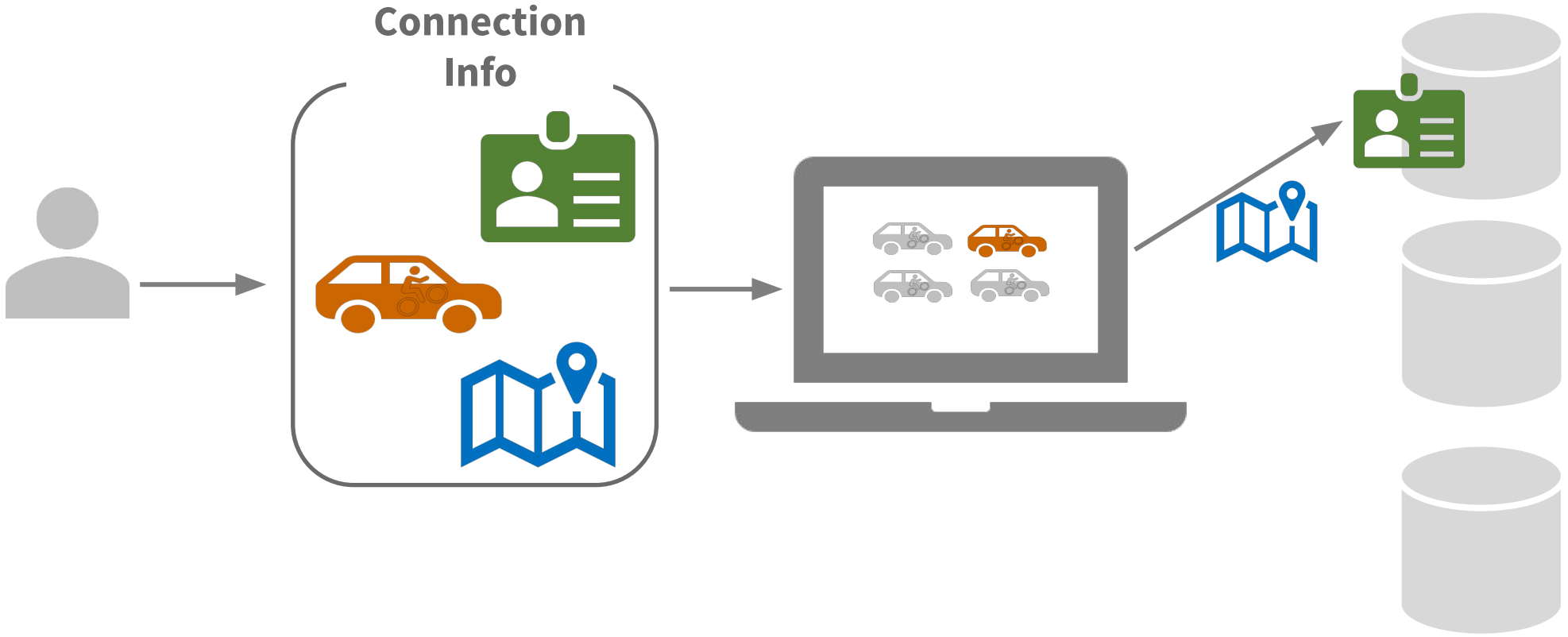
- User name & password
- Token

- URL
- IP Address

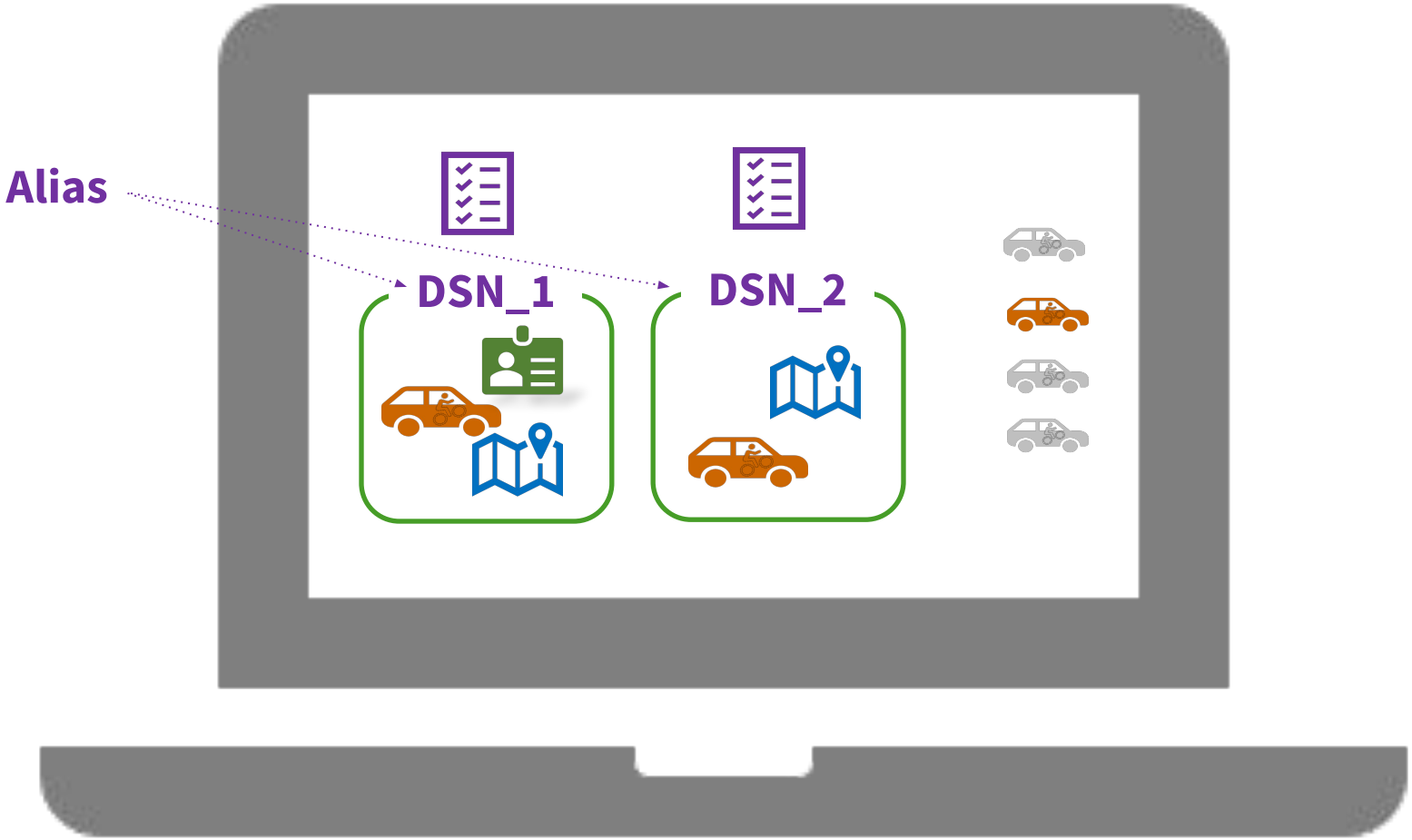


- ODBC (Used by **ADO** & **OLE DB**)
- JDBC

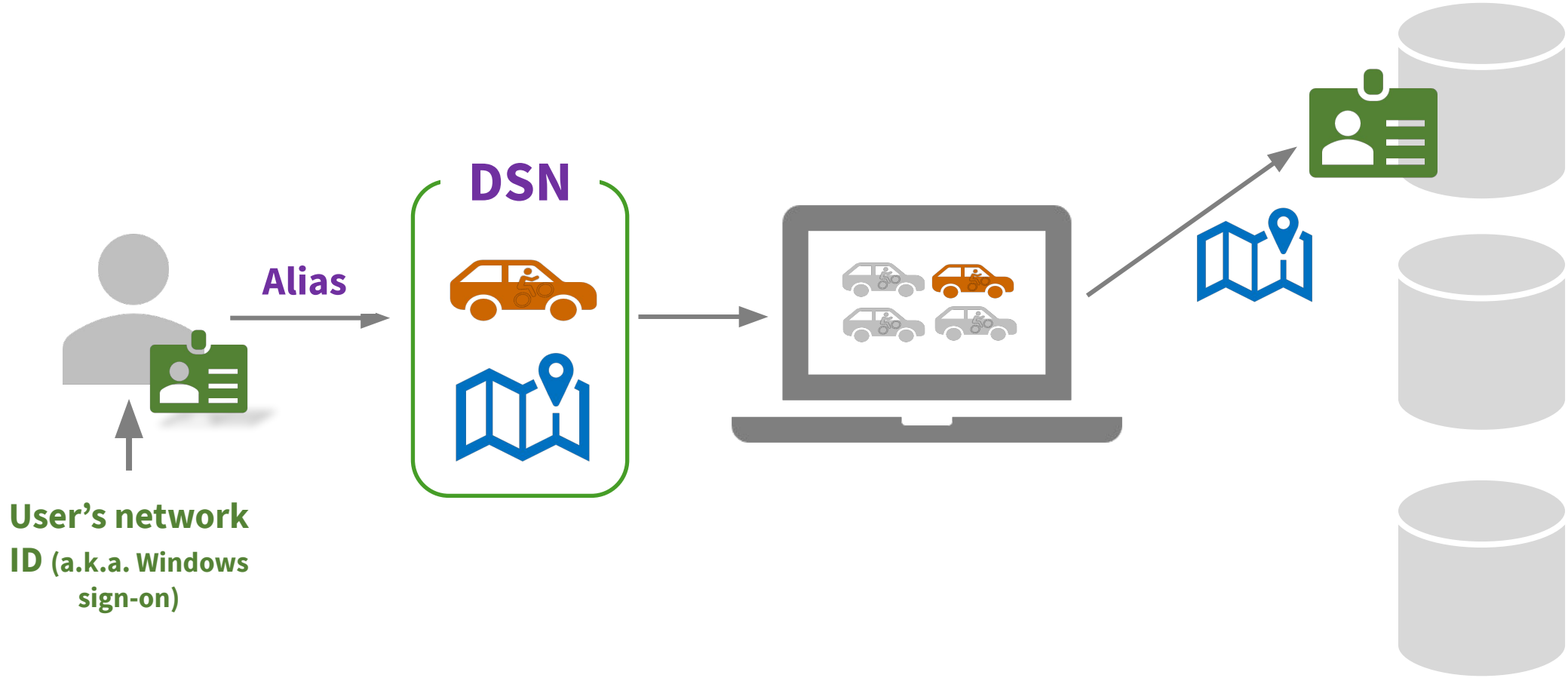
Connection info



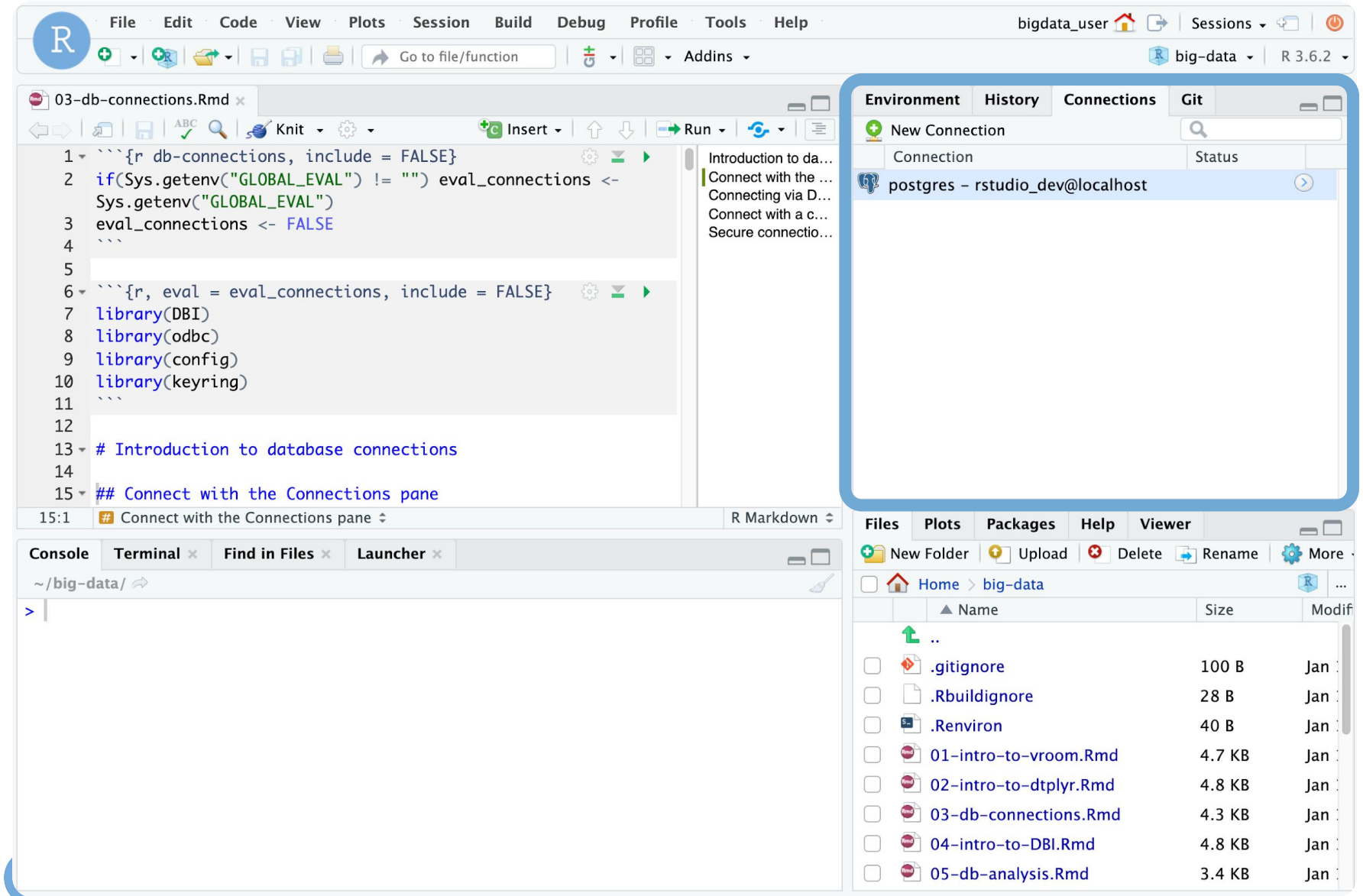
Data Source Name (DSN)

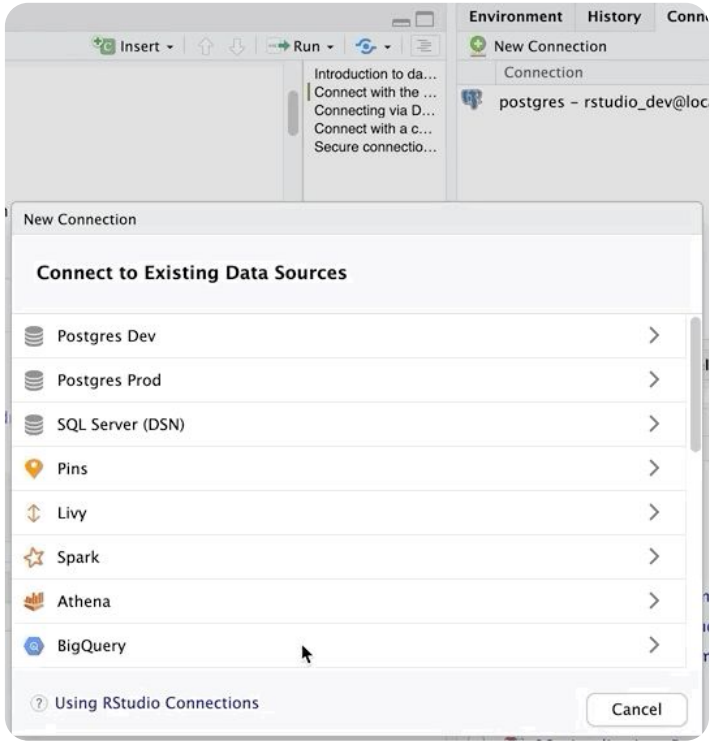


The ideal connection

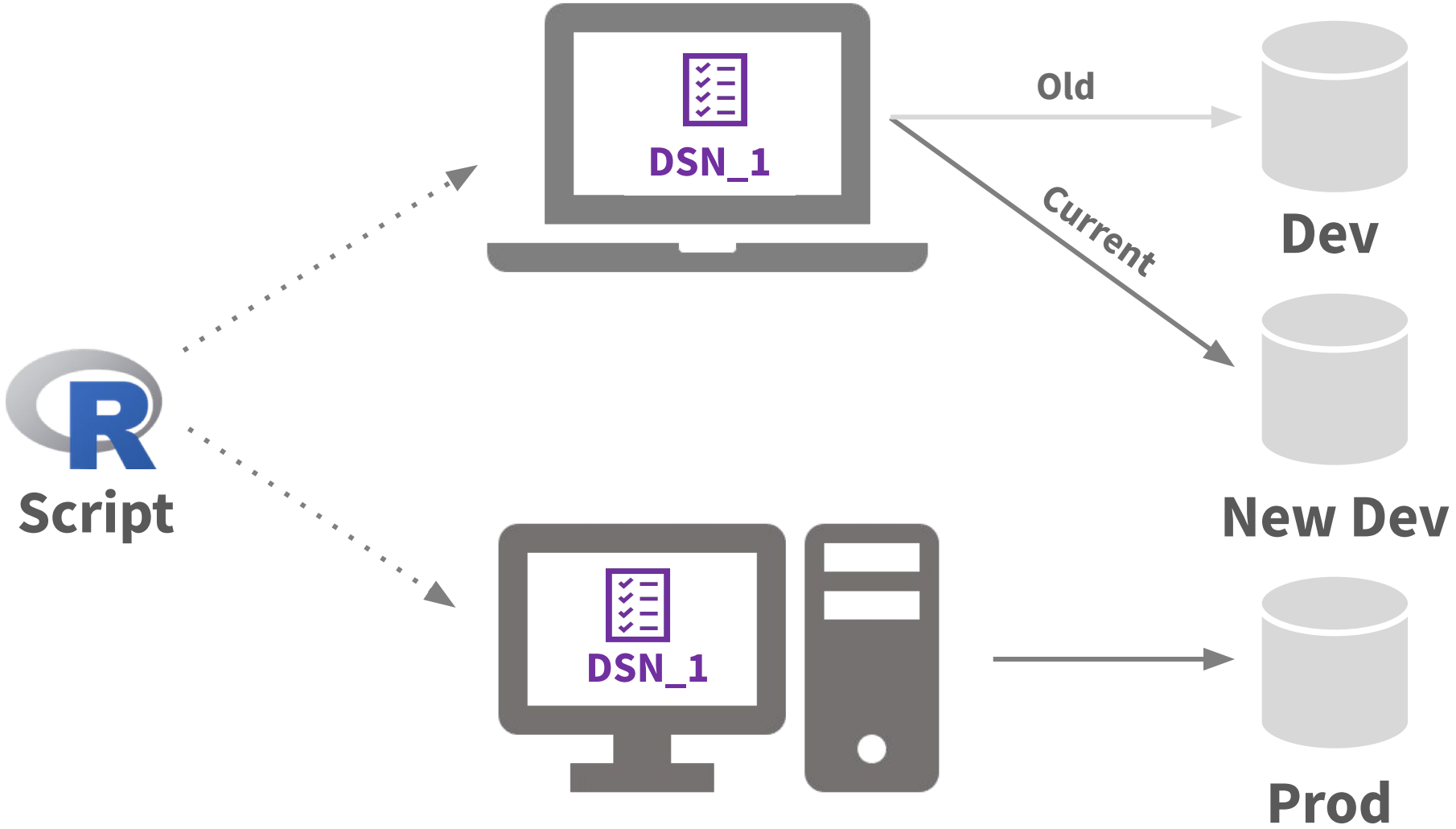


The connections pane





Why DSN?



Alternatives for securing connections

1. `config`
2. `keyring`
3. Environment variables
4. `options()`
5. Prompt for credentials

R packages

General connections

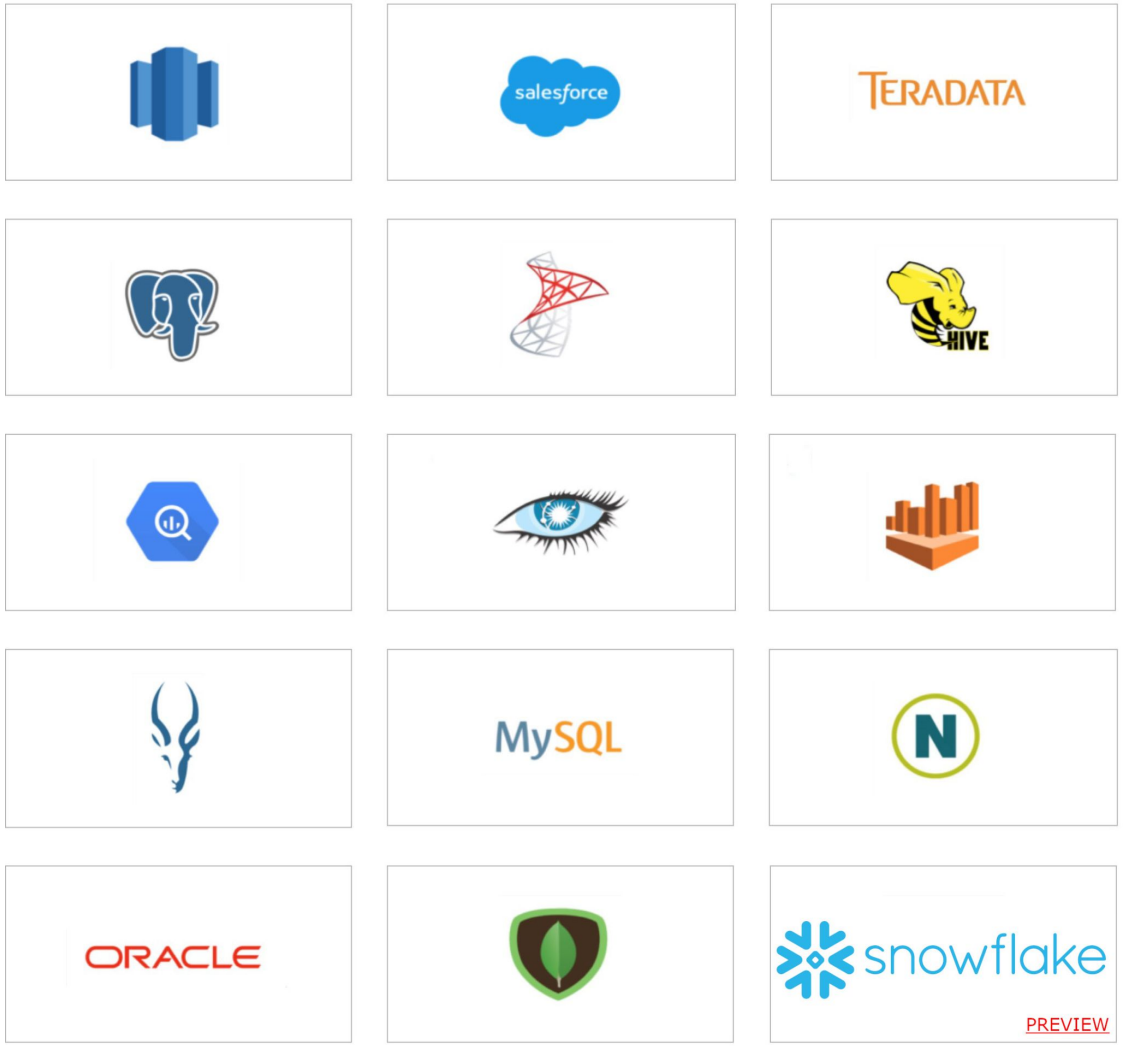
- DBI
- odbc
- connections

Specific Connections

- sparklyr
- RPostgres
- RSQLite
- ...

github.com/r-dbi

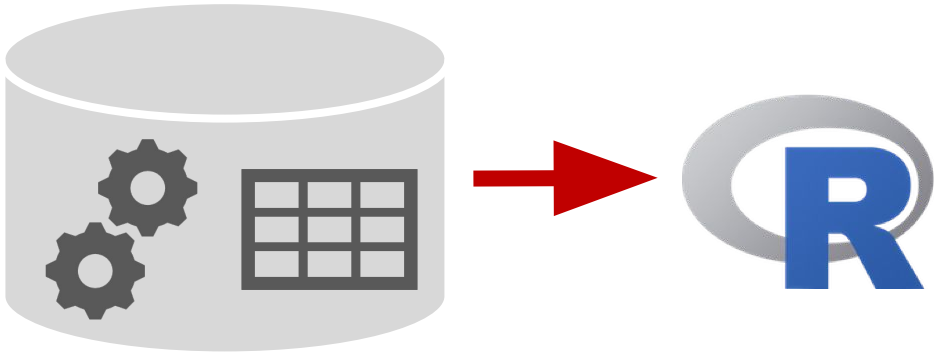
ODBC Drivers



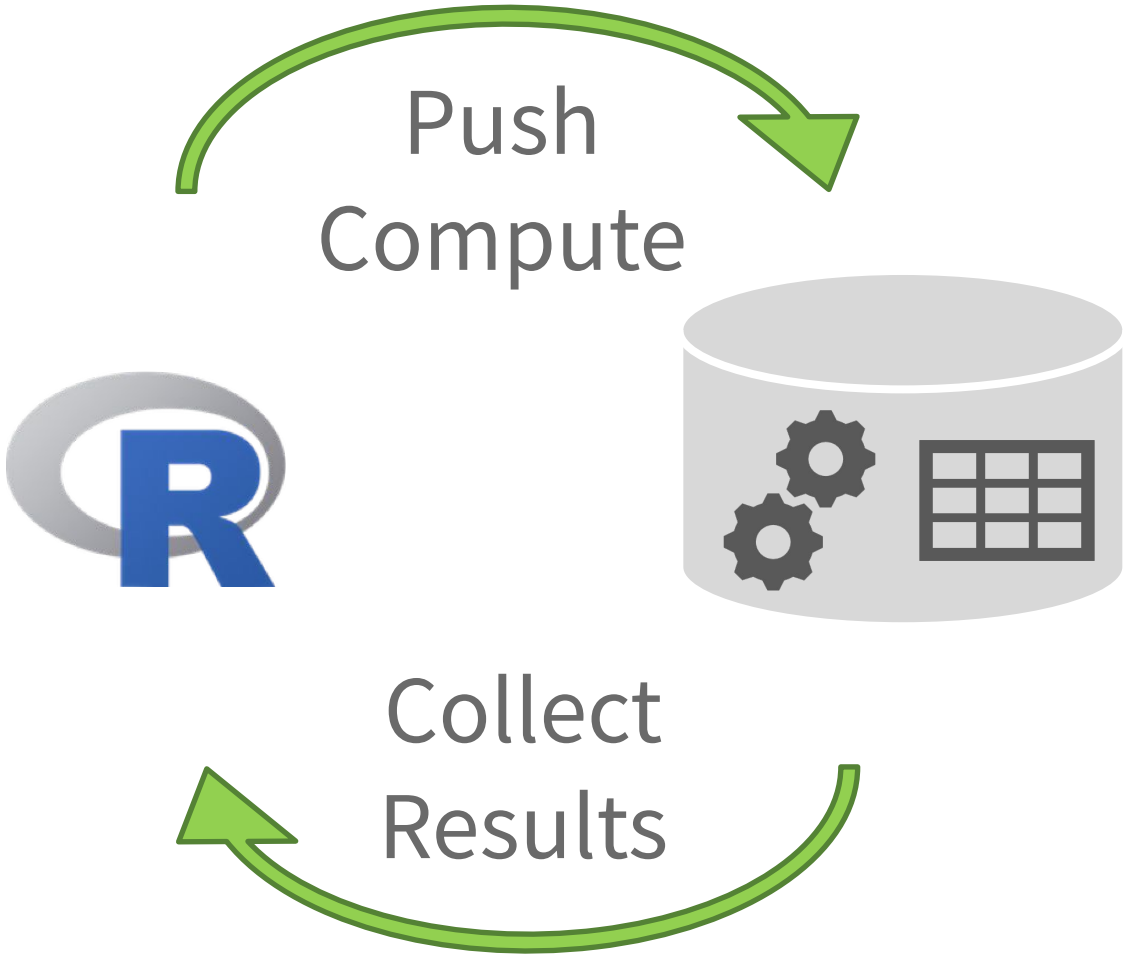
rstudio.com/products/drivers/

Wrangle inside the DB

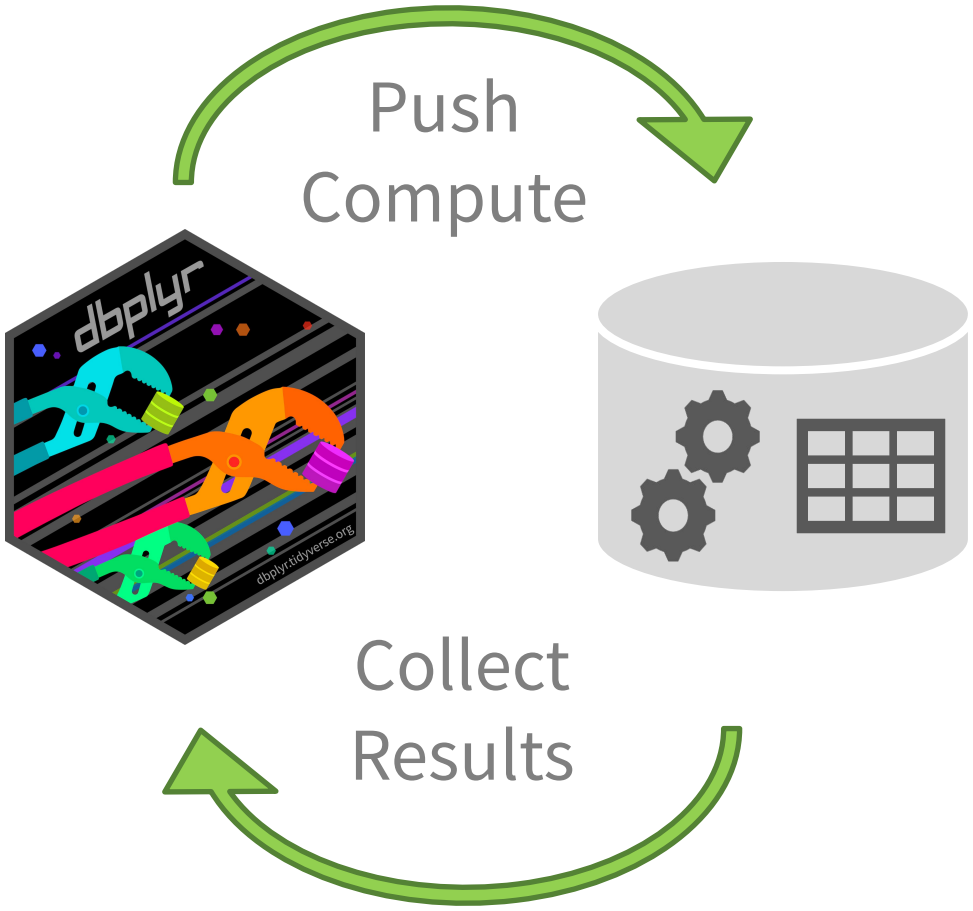
Time Consuming



Extract Data



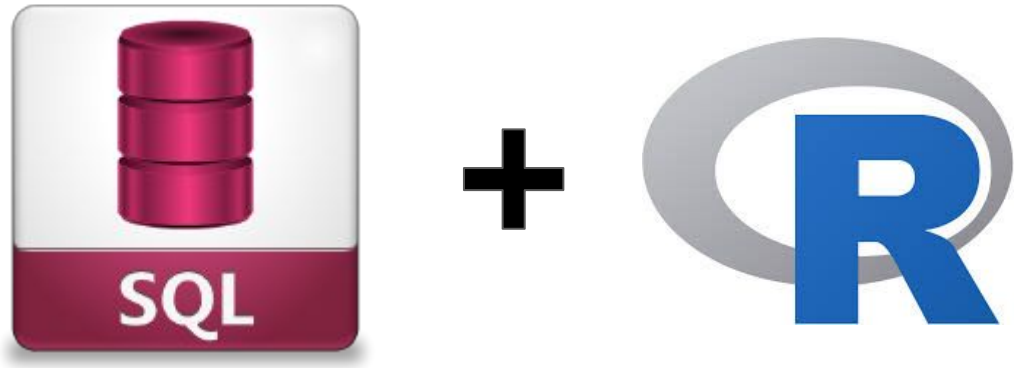
Advantages



1. dplyr translates to SQL
2. Take advantage consistent syntax
3. All your code is in R!

DBI package

1. Stands for **d**ata**b**ase **i**nterface
2. Helps connect R to various database management systems
3. Used for connecting to and interacting with various databases
4. Execute SQL commands against the database



dbi.r-dbi.org/

DBI common functions

Connecting

- dbConnect
- dbDisconnect

Tables

- dbListTables
- dbWriteTable
- dbReadTable

Queries

- dbSendQuery
- dbGetQuery
- dbExecute

Options to Push Compute

Write SQL statements

```
SELECT "customer_id",
COUNT(*) AS "n"
FROM "retail.orders"
GROUP BY "customer_id"
```

Use dplyr verbs

```
orders %>%
 count(customer_id)
```

`sparklyr` package

1. Leverage the power of Spark from R
2. Take advantage of distributed, in-memory processing
3. Create extension that leverage the Spark API

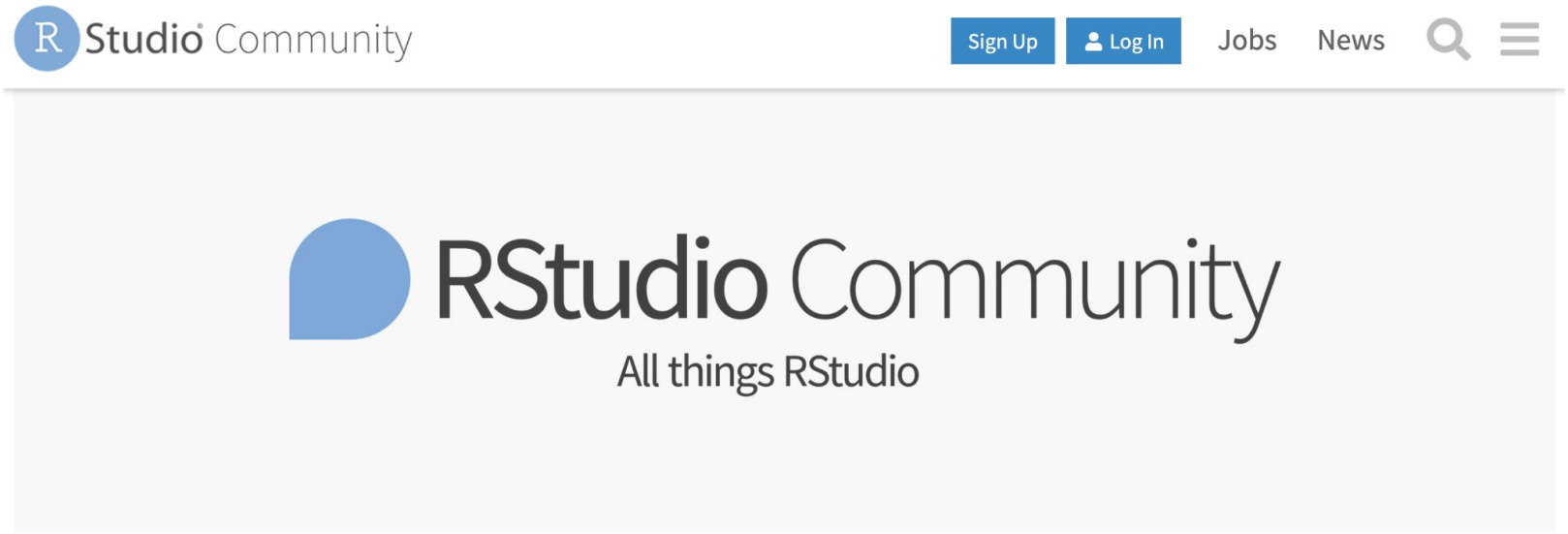
Best Practices

- Use `dplyr` for consistent syntax and tooling
- Rely on each tool for its strengths
- Leave heavy computation to tools designed for heavy computation (Databases, Spark)
- Use R for final analysis
- When possible, avoid bringing **all** data into R
- Create extracts and views to speed things up

Bookmark and check regularly

- http://db.rstudio.com/
- http://spark.rstudio.com/
- https://www.tidyverse.org/
- https://arrow.apache.org/docs/r/
- https://rviews.rstudio.com/
- https://rviews.rstudio.com/categories/databases
- https://blog.rstudio.com/

Join the community!



https://community.rstudio.com/

Familiarize yourself with the repos

If I need to…	Check out
Report an issue or see if others are having the same problem	Issues
See if an feature exists or if it's coming up in future releases	NEWS
See the basics about the package	README

- https://github.com/tidyverse/dplyr
- https://github.com/tidyverse/dbplyr
- https://github.com/tidyverse/ggplot2
- https://github.com/r-dbi/odbc
- https://github.com/r-dbi/DBI
- https://github.com/edgararuiz/dbplot
- https://github.com/tidymodels/tidypredict
- https://github.com/apache/arrow/tree/master/r
- https://github.com/rstudio/sparklyr