

# cloudera

---

dplyr Interfaces to Large-Scale Data

Ian Cook

@ianmcook

ian@cloudera.com

# Context

Mission for Cloudera: Provide a platform for data analysts, data scientists to efficiently query, analyze, model large-scale data in clusters, cloud storage

- By distributing Apache Spark, Apache Impala, other tools
- By enabling productive use of these tools

Python and R users often have difficulty moving from smaller data to large-scale distributed data

- Familiar packages, methods don't work the same way on distributed data



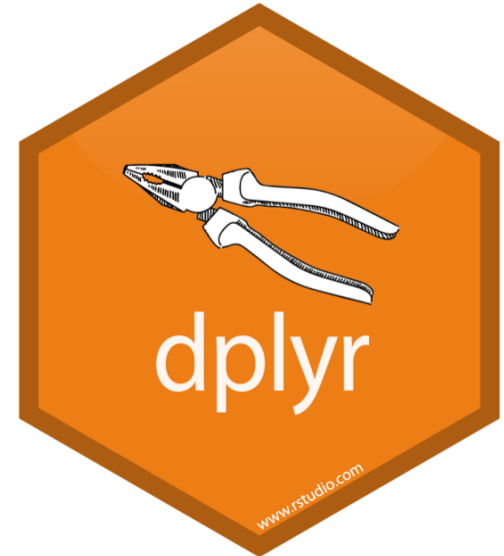
# dplyr

dplyr provides a set of *verbs* that perform common data manipulation steps:

- `select()` to select columns
- `filter()` to filter rows
- `arrange()` to order rows
- `mutate()` to create new columns
- `summarise()` to aggregate
- `group_by()` to perform operations by group

dplyr works on local data and with remote data sources

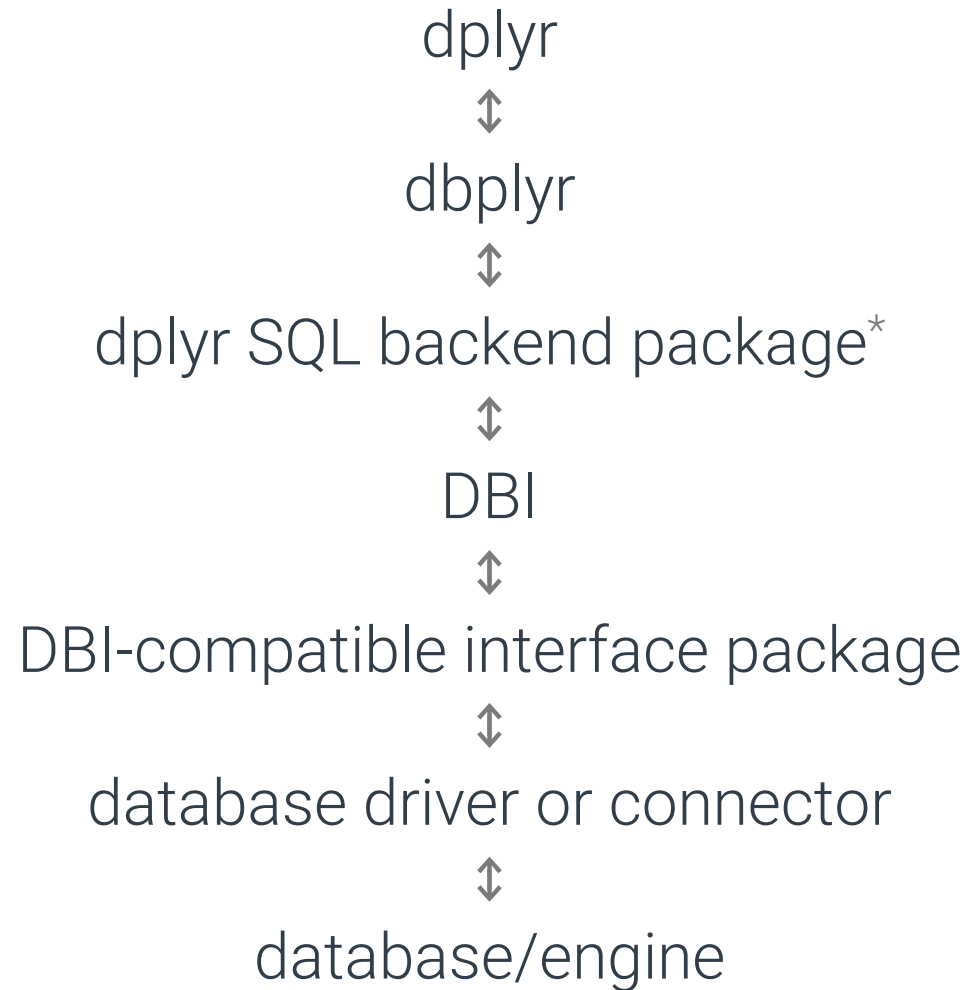
- For remote sources, dplyr commands are translated into SQL



# Demonstration

Example code at  
[github.com/ianmcook/dplyr-examples](https://github.com/ianmcook/dplyr-examples)

# dplyr SQL backends



\* optional

# sparklyr

- Provides a SQL backend to dplyr for Spark
- Also exposes the MLlib API and a subset of the Spark DataFrames API
- Developed by RStudio

[spark.rstudio.com](http://spark.rstudio.com)

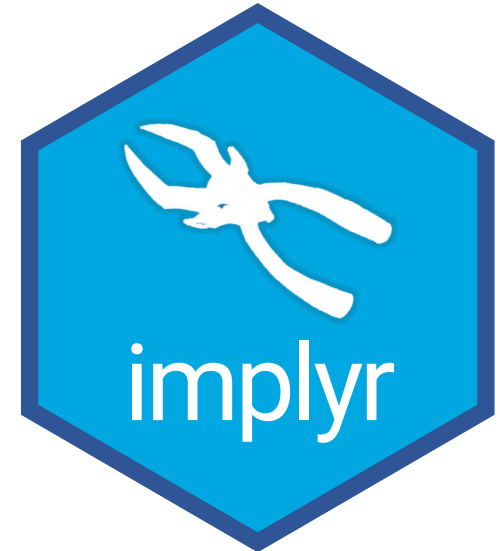
**cloudera**



# implyr

- Provides a SQL backend to dplyr for Impala
- Uses ODBC or JDBC to connect to Impala
- Developed at Cloudera

[tiny.cloudera.com/implyr](http://tiny.cloudera.com/implyr)





---

# Five tips for using dplyr with SQL data sources

---

1

---

Use `show_query()`

---

# 2

---

`filter()` early  
`arrange()` late

---

# 3

---

Check your data types

---

# 4

---

Know your SQL engine

---

5

---

Know when to collect ()

---

# Thank you

---

Ian Cook  
@ianmcook  
ian@cloudera.com