A blue-tinted photograph of the Austin skyline at dusk. In the foreground, the Congress Avenue Bridge spans the Colorado River. A massive swarm of bats is captured in flight against a dark sky above the bridge. In the background, the city's modern skyscrapers, including the Frost Bank Tower and the W Hotel, are silhouetted against the setting sun.

# Welcome to Big Data with R!

# Housekeeping items

- rstudio::conf app
- Wi-fi password
- Access your server

# Wi-fi password

[password]

# Class server

Link: [rstud.io/class](https://rstud.io/class)

Password: [password]

# Schedule

**9am – 10:30am**

Break (30 mins)

**11am – 12:30am**

Lunch (1.5hrs)

**2pm – 3:30pm**

Break (30 mins)

**4pm – 5:30pm**

# The team



**Cole  
Arendt**  
*Infrastructure*



**Mara  
Averick**  
*TA*



**Ron  
Blum**  
*TA*



**Javier  
Luraschi**  
*Guy in the back*



**James  
Blair**  
*Instructor*



**Edgar  
Ruiz**  
*Instructor*



# Pre-class Survey Review

# Class / material overview

- Server
- Database
- Spark
- Deck
- Exercise book

# Unit 1

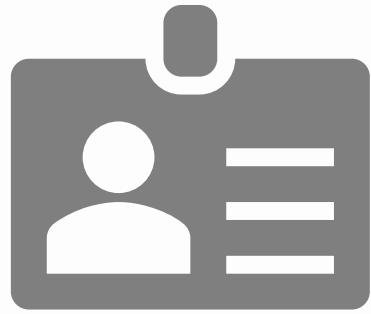
## Accessing databases



Photo by [Florian Pircher](#) on [Unsplash](#)

# Exercise 1.1 - 1.3

# Connection requirements



Credentials

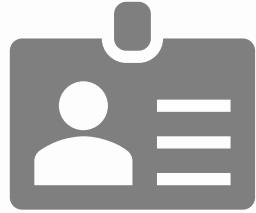


Location



Driver

# Requirement definitions



- User name & password
  - Token
- 

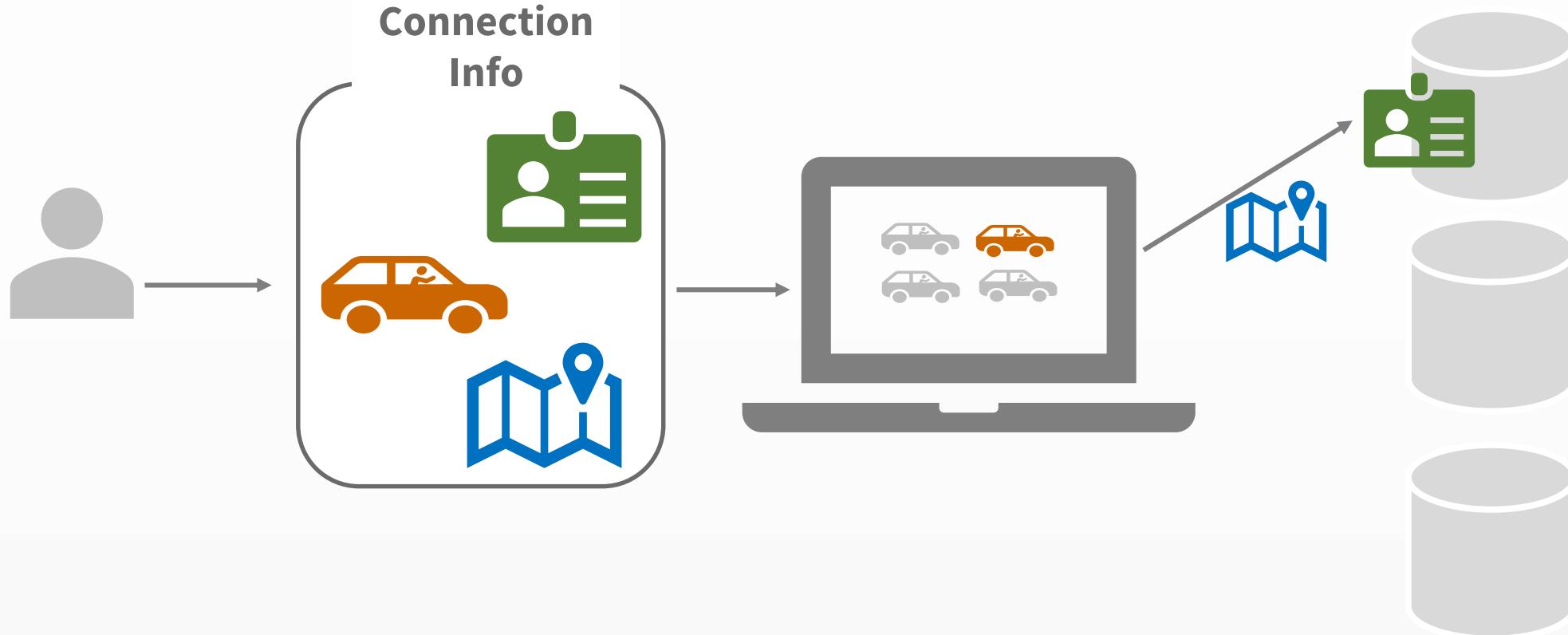


- URL
  - IP Address
- 

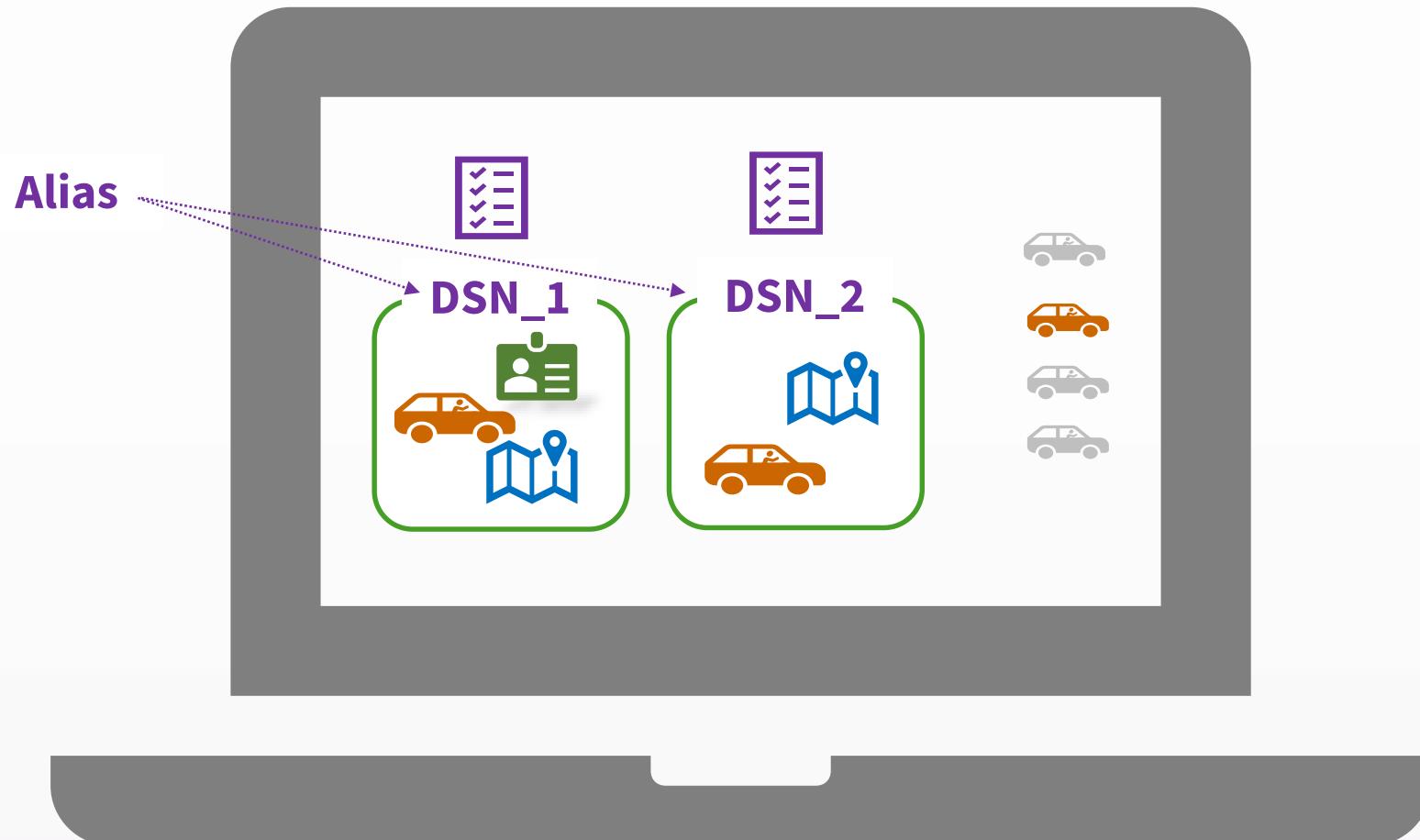


- ODBC (Used by **ADO & OLE DB**)
- JDBC

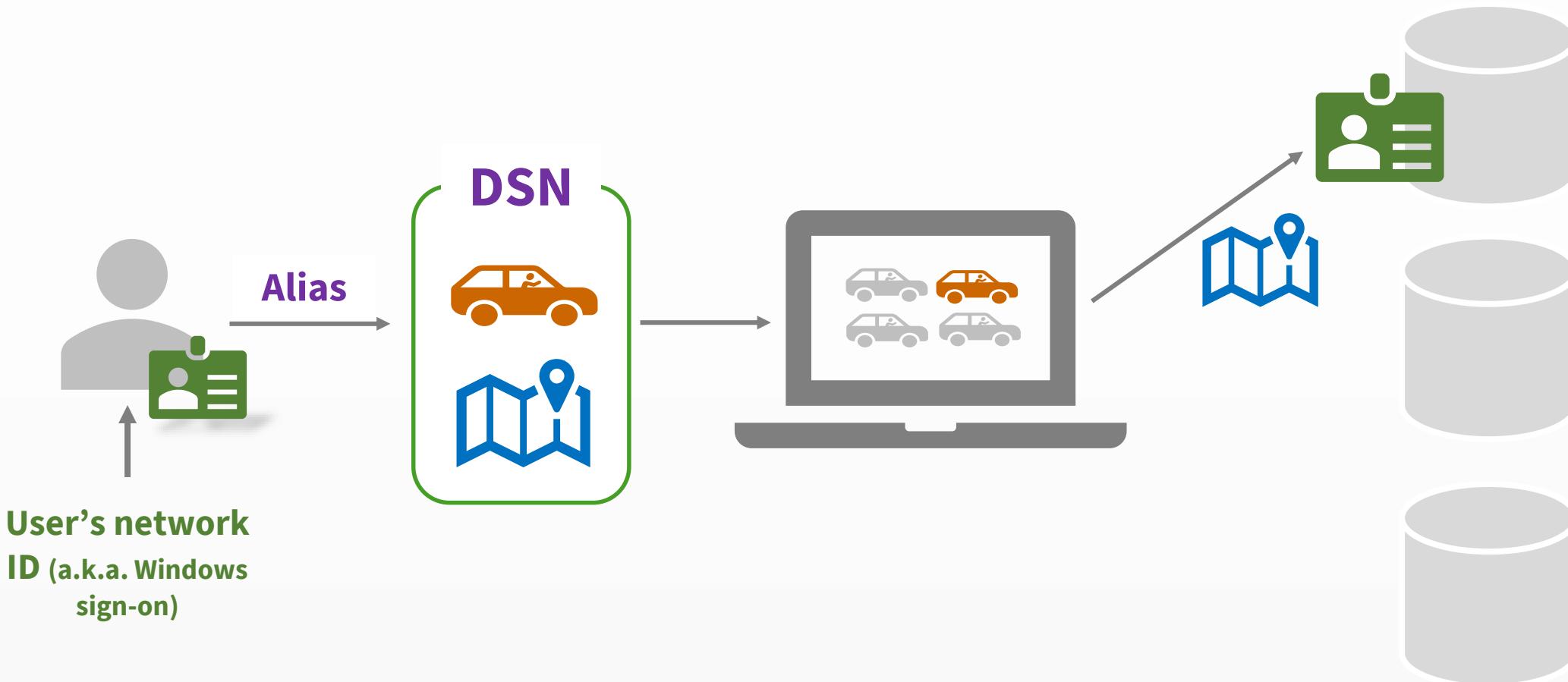
# Connection info



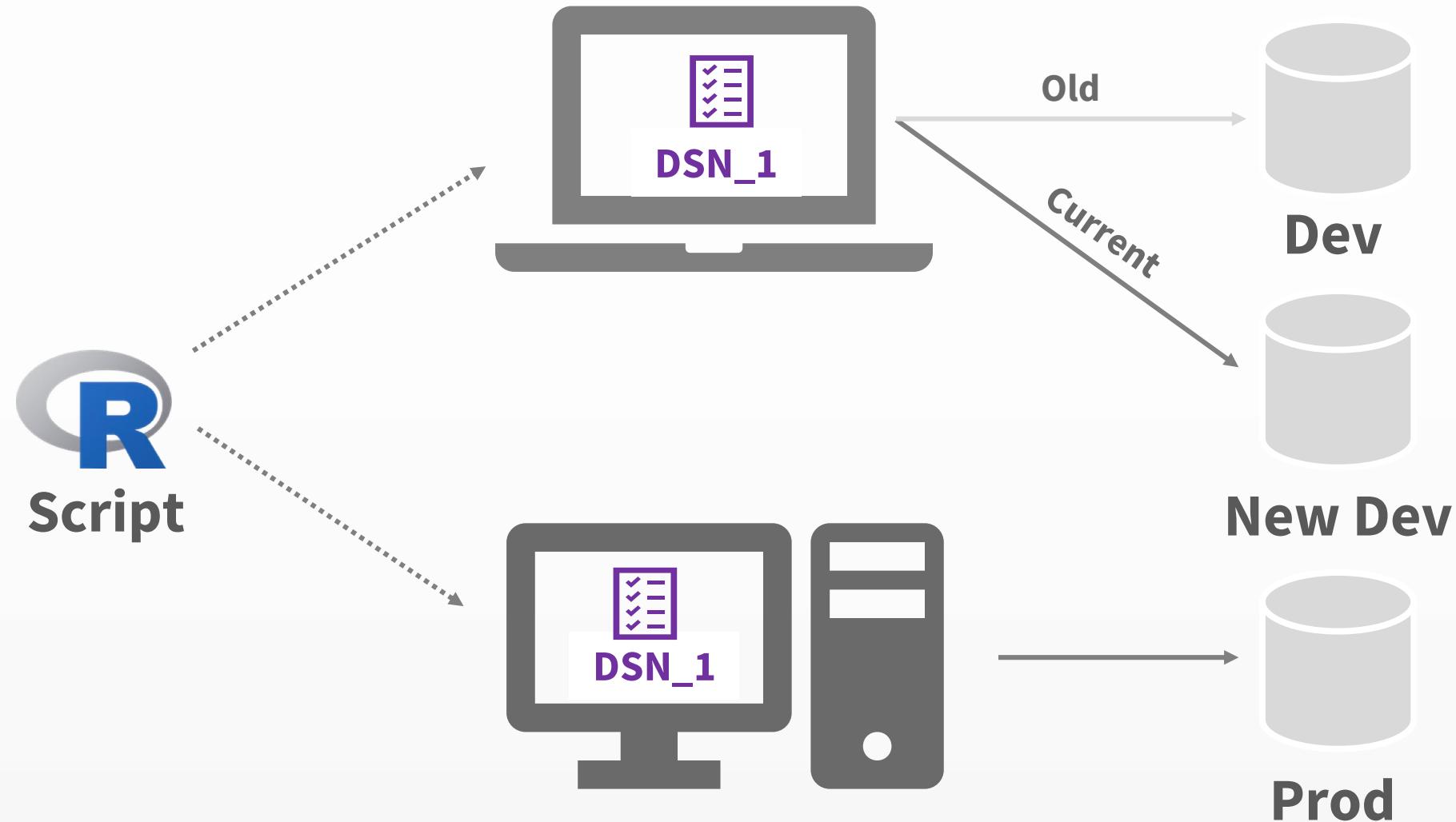
# Data Source Name (DSN)



# The ideal connection



# Why DSN?



# Exercise 1.4

# Alternatives for securing connections

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

# Exercise 1.5 – 1.8

# Let's talk about Big Data



Photo by [Chris Christensen](#) on [Unsplash](#)

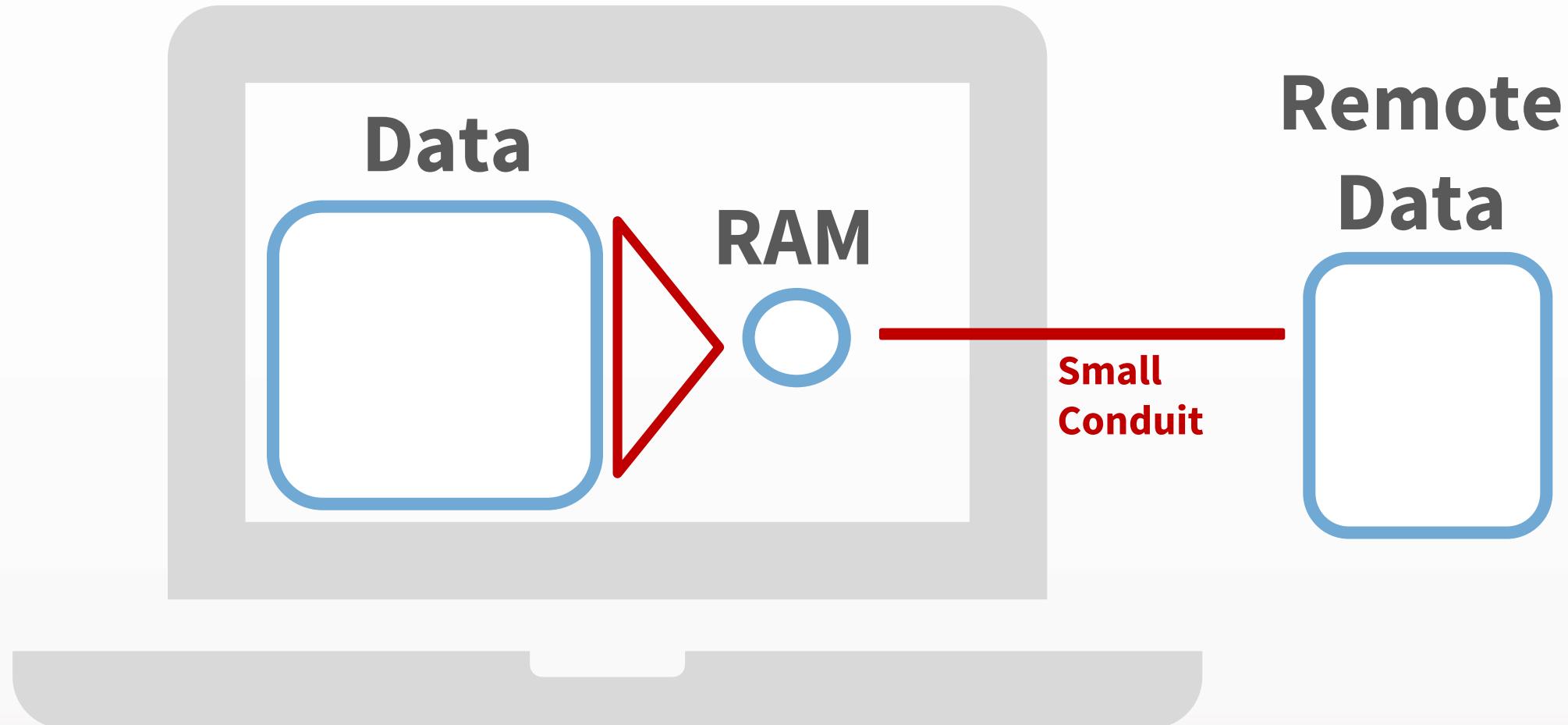
*Velocity*  
**Data > RAM**

*Garrett Grolemund*

*Value*  
**Remote Data**

*Edgar Ruiz (circa 2018)*

# Big Data in R



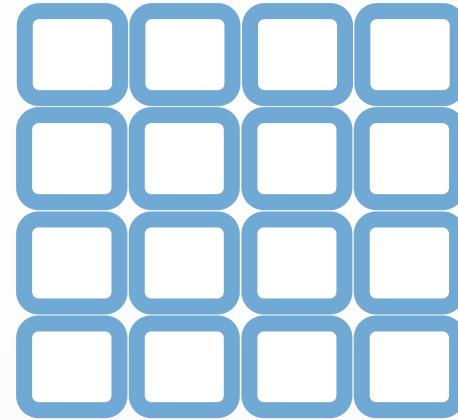
# Big Data Strategies

## Sample



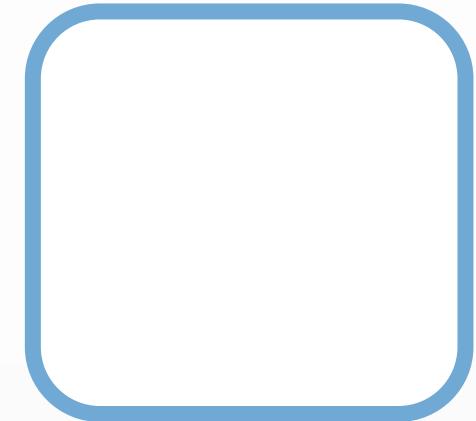
Most common approach for **modeling**

## Parts



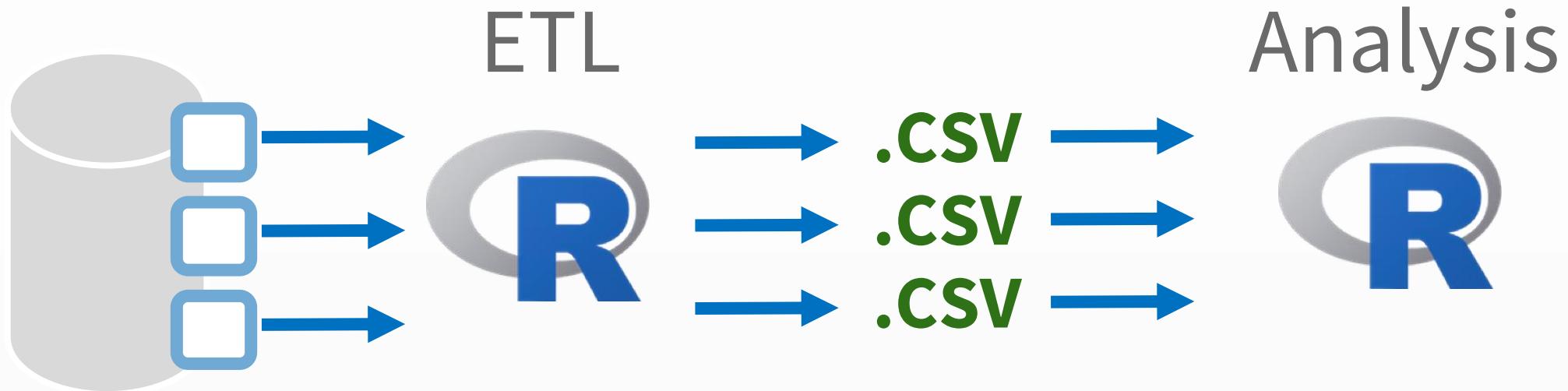
Most common approach for **general analysis**

## Whole

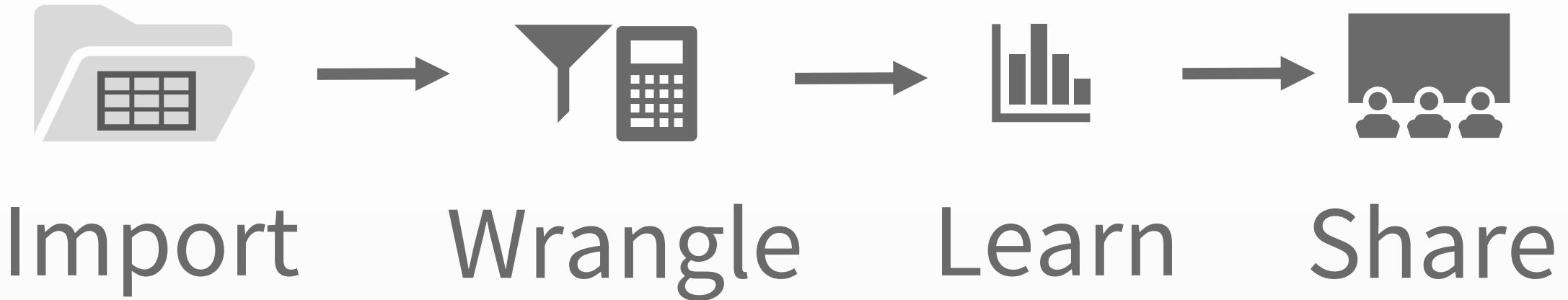


In most cases, **the preferred approach**, it's just not feasible

# Parts - “The Method”



# Typical DS project



# Remote Data Sources

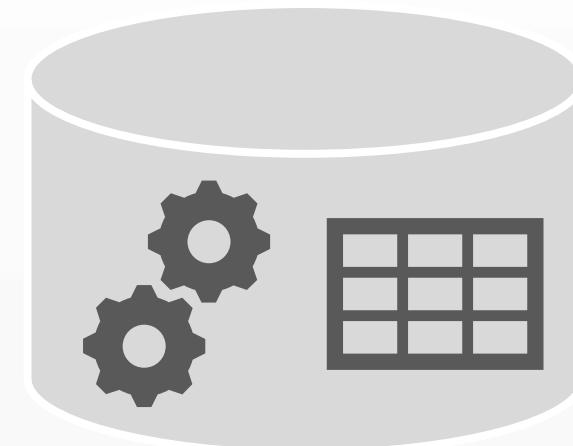
## Flat Files

Only Data



## Remote Sources

Data & Compute engine



# Unit 2 & 3

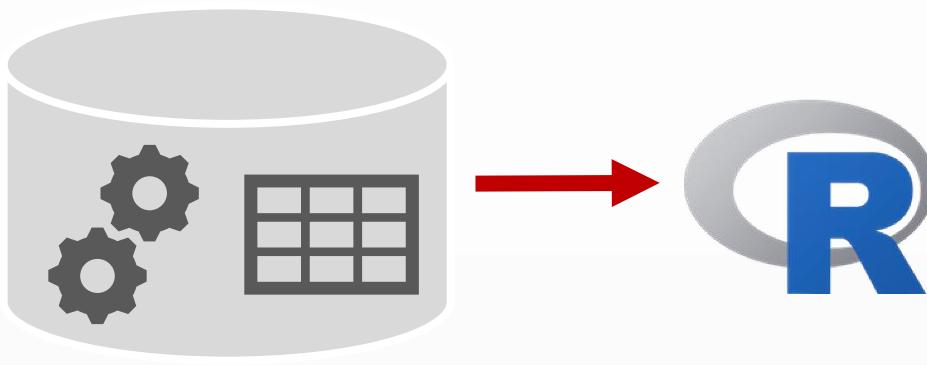
## Using dplyr

/dee-plier/

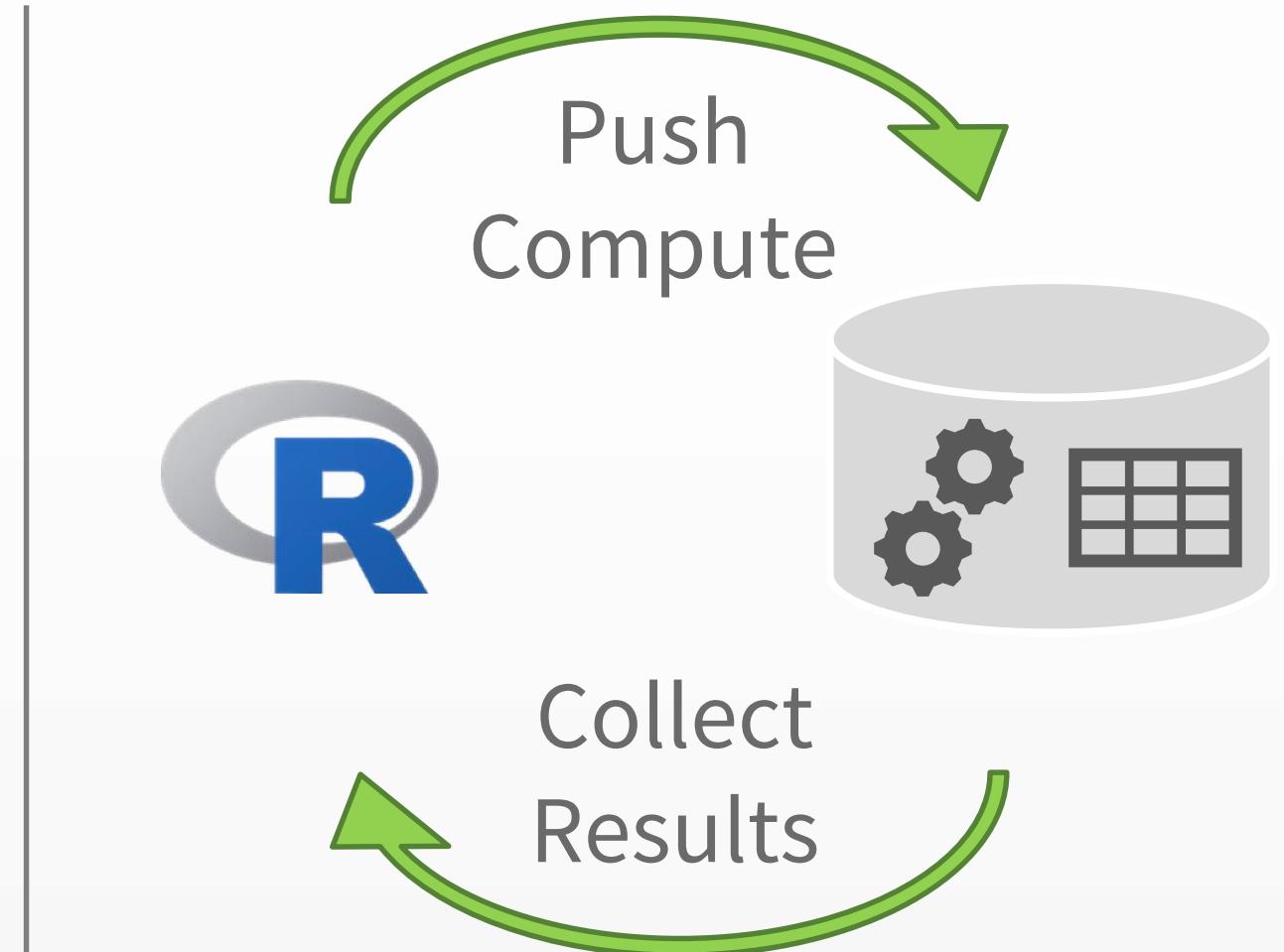


Photo by [Arthur Lambillotte](#) on [Unsplash](#)

# Wrangle inside the DB



Extract Data



# Options to Push Compute

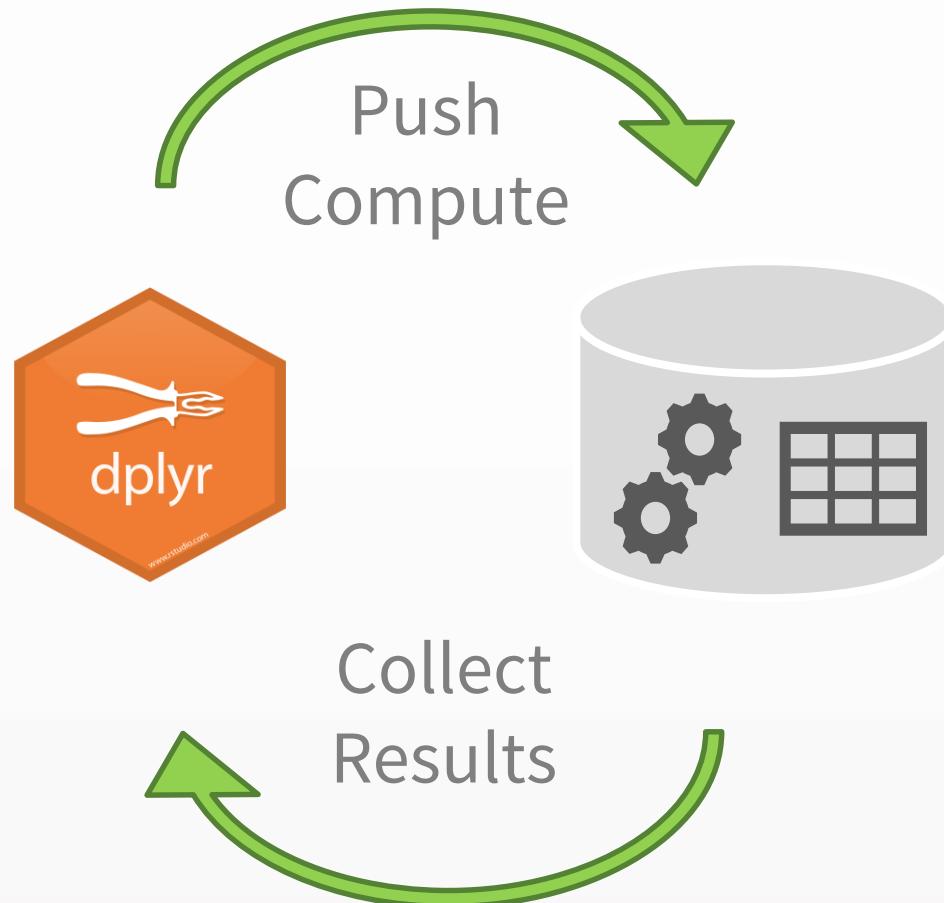
Write SQL statements

```
SELECT "name",  
COUNT(*) AS "n"  
FROM "vwFlights"  
GROUP BY "name"
```

Use dplyr verbs

```
flights %>%  
group_by(name) %>%  
tally()
```

# Advantages



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

# Exercise 2.1 - 2.6

# DS project using DBs



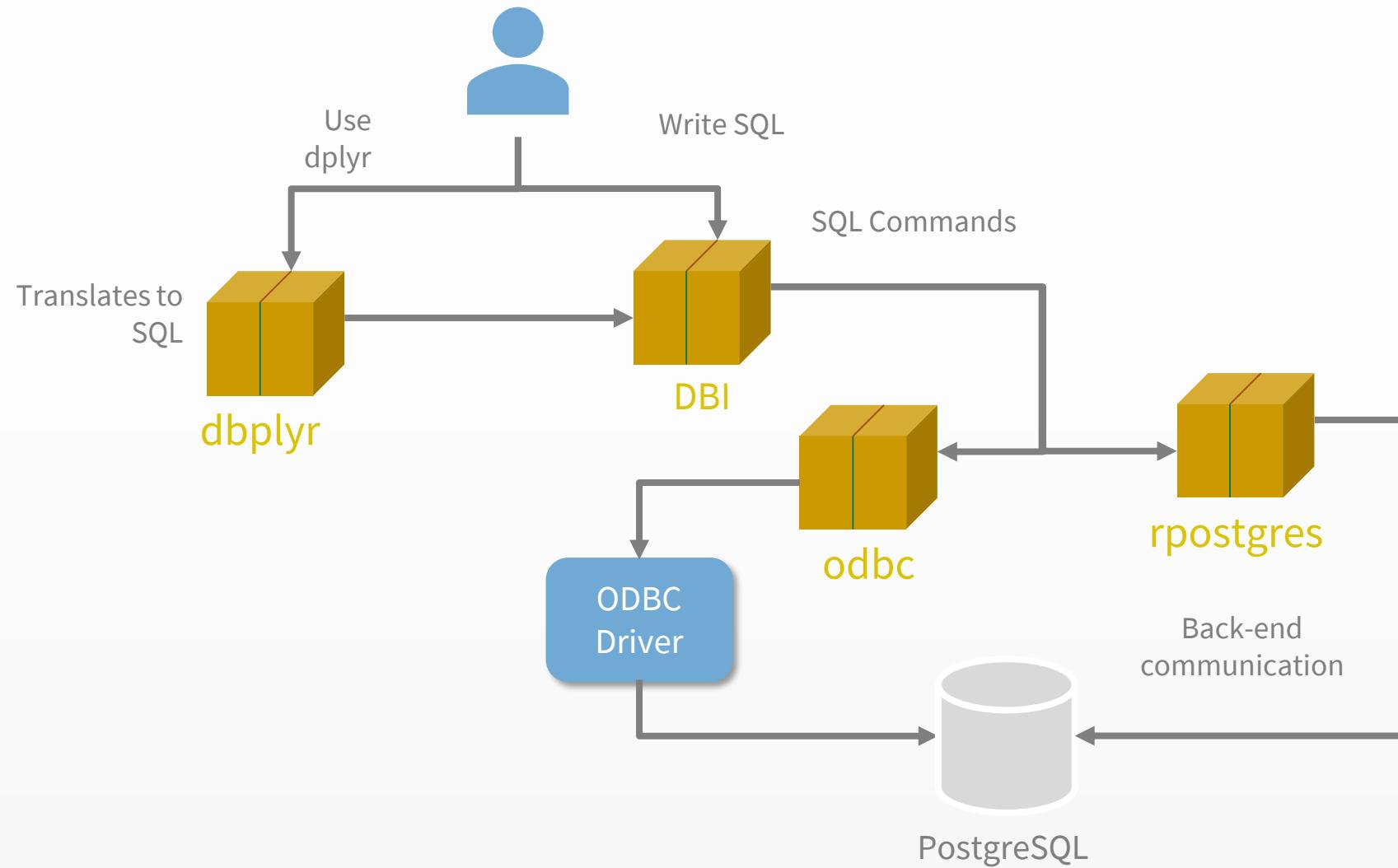
# How to access a database

1. R Package – As implemented by RPostgreSQL and others
2. ODBC - As implemented in odbc package
3. JDBC - As implemented in RJDBC and other

# Packages

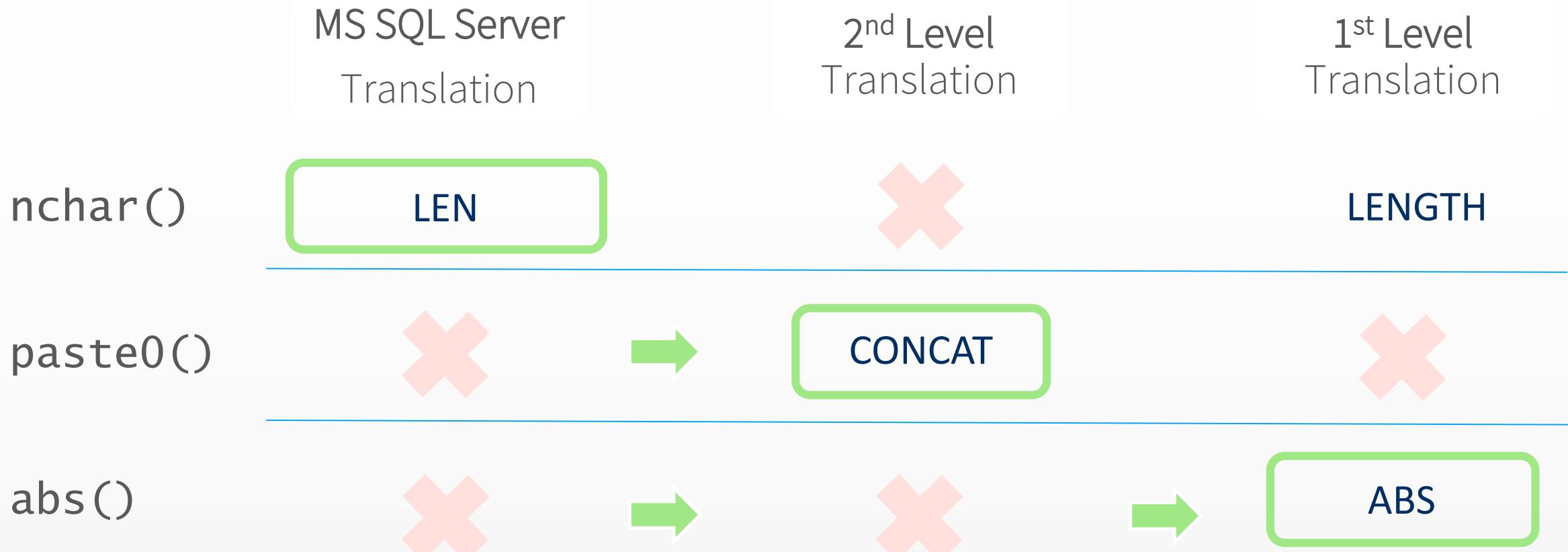
1. `dplyr` – Simplifies data wrangling
2. `dbplyr` – Provides database specific translation
3. `DBI` – Common interface for Databases and R
4. `DB R Package` – Back-end interface for a specific database, such as `RPostgreSQL`
5. `odbc` – Back-end interface to a database using an ODBC driver

# Architecture



# How dbplyr translates

```
class(con)  
[1] "Microsoft SQL Server"
```



# Translations available in *dbplyr*

1. Microsoft SQL Server
2. Oracle
3. Apache Hive
4. Apache Impala
5. PostgreSQL
6. MS Access
7. MariaDB (MySQL)
8. SQLite
9. Amazon Redshift
10. Teradata

# Exercise 3.1 – 3.6

# Some advice...

1. Think before you collect()
2. Just a bit off the top, use head()
3. Be select()ive of fields to bring back
4. `tbl(con, "No SQL statements in tbl")`

# Unit 4

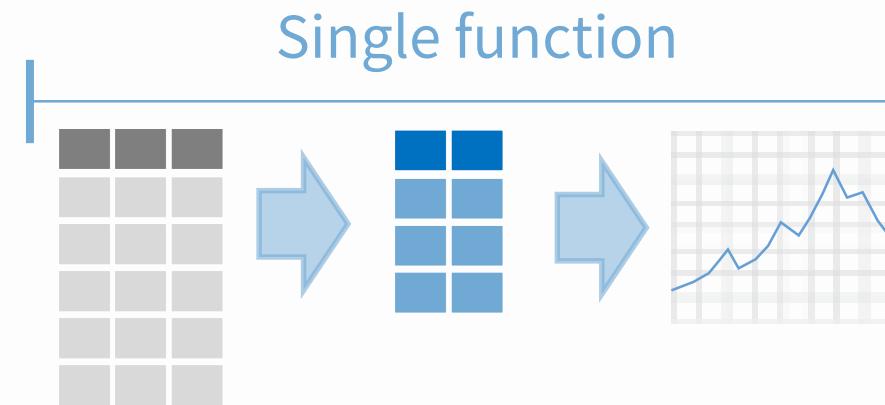
## Visualizations



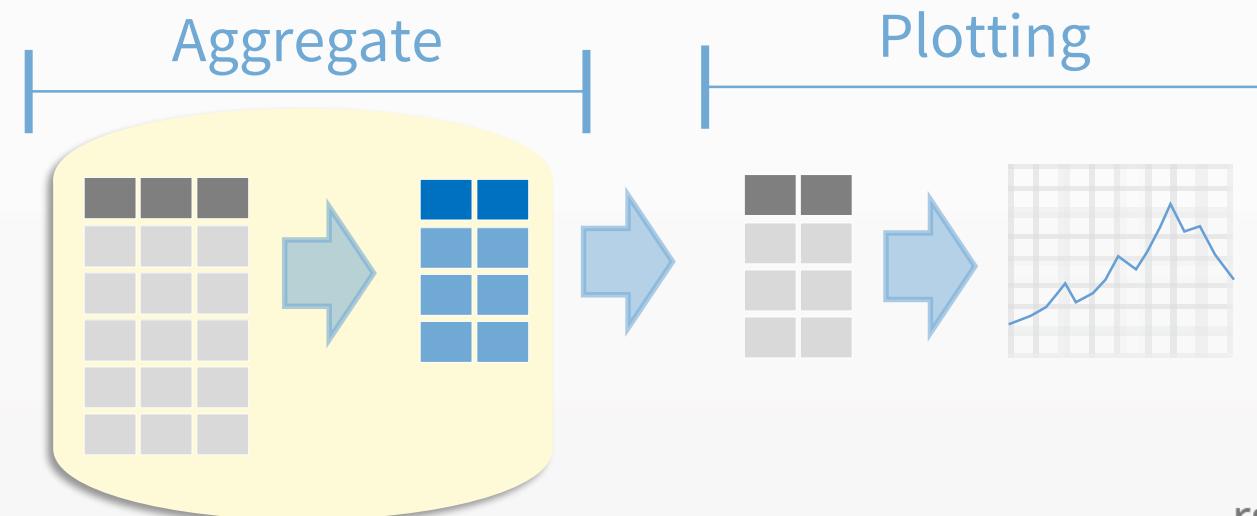
Photo by [Luis Alfonso Orellana](#) on [Unsplash](#)

# Visualizations

Local data

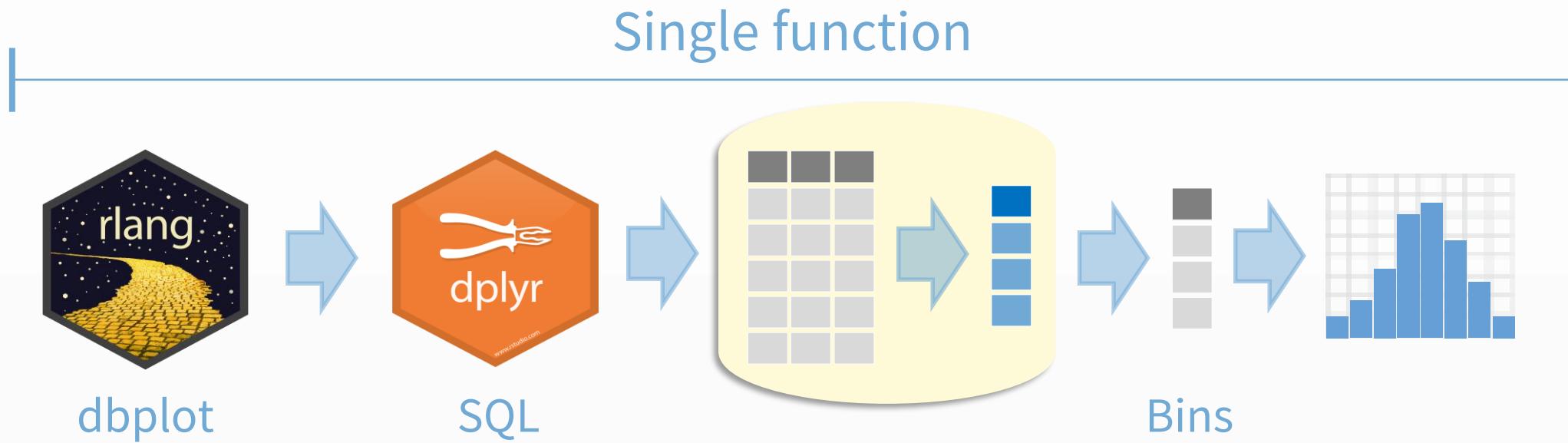


Remote data



# Exercise 4.1 – 4.6

# Complex plots



# Exercise 4.7 – 4.10

# Unit 5 Modeling

A chalkboard with three equations written in chalk. The first equation is  $\frac{dN}{dt} = qV_{act} - \alpha_0(N-N_0)(1-\varepsilon S)S + \mu_e - \frac{\mu}{\tau_n} - \frac{\mu}{\tau_p}$ . The second equation is  $\frac{dS}{dt} = T_b \alpha_0 (\mu N_0) (1-\varepsilon S) S + \mu N - \frac{S}{\tau_p}$ . The third equation is  $\frac{S}{P_t} = \frac{T_p \lambda_0}{T_b + \mu e} = 0$ . To the right of the equations, there is a bracket grouping them with the text  $N = N_0$  above it and  $P_t = (m)$  below it. Below the equations, there is a small diagram of a rectangle with a diagonal line through it, labeled  $S \leq \varepsilon$ .

Photo by [Roman Mager](#) on [Unsplash](#)

# Modeling scenario

1. Training sample



2. Model on sample



3. Testing sample



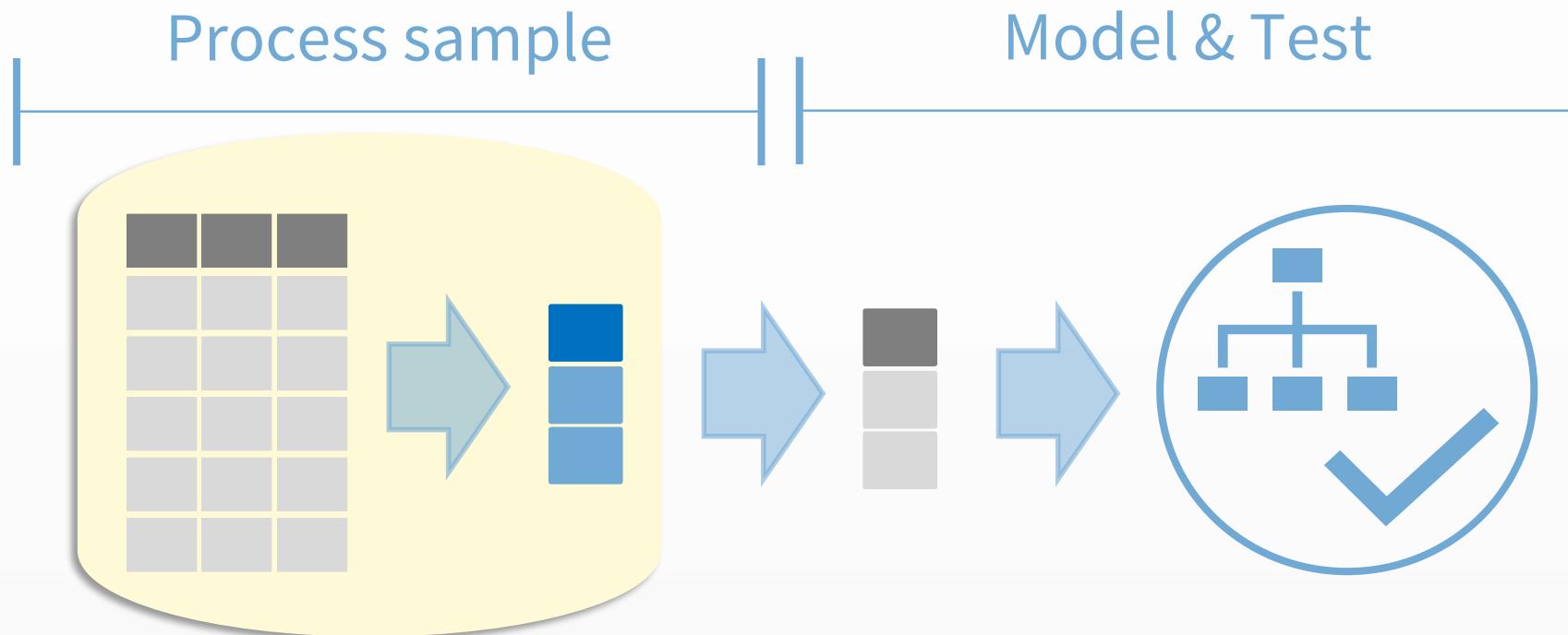
4. Verify model



5. Score data

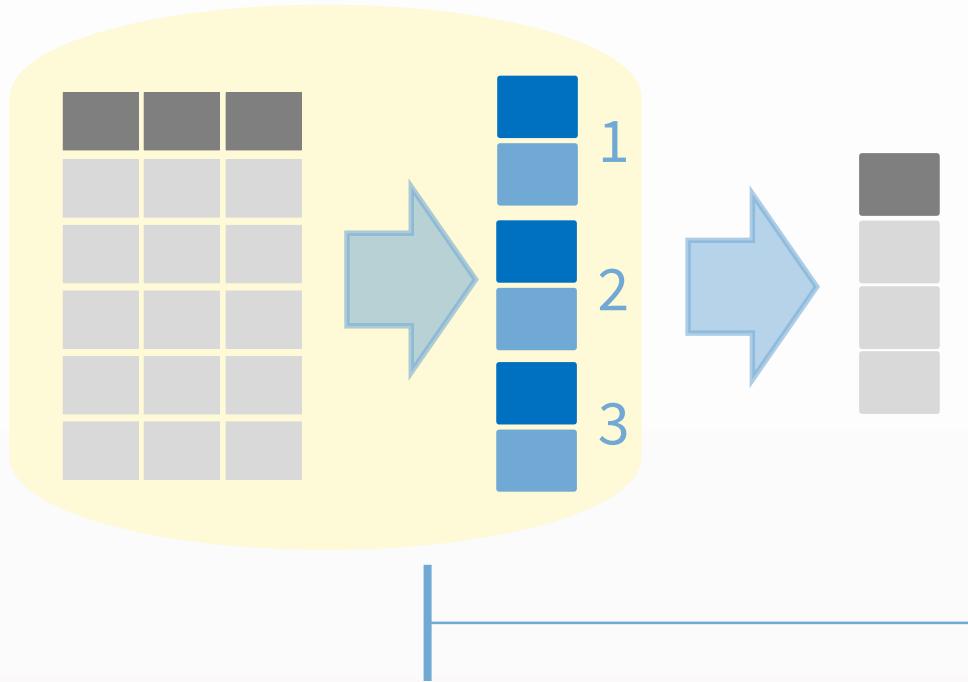


# Modeling with a Database



# Exercise 5.1 – 5.2

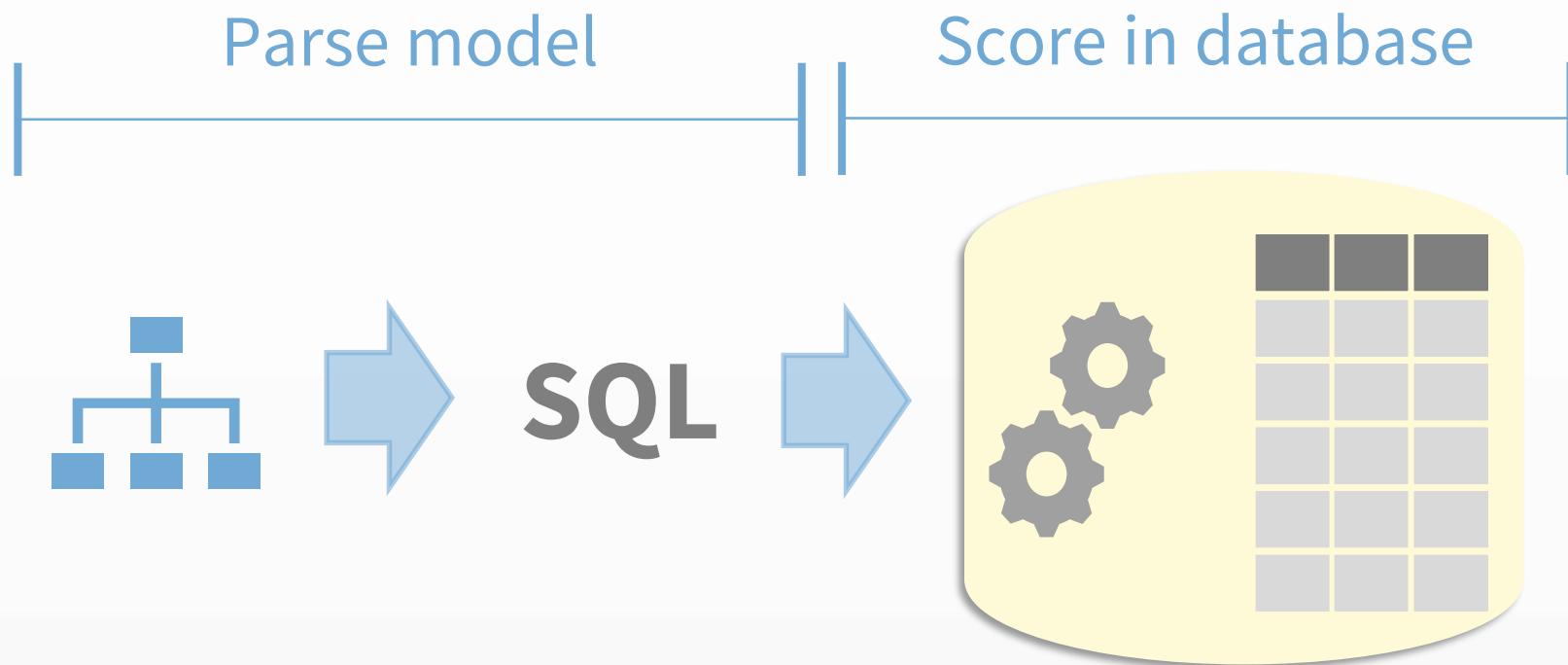
# Multi-step sampling



Multiple SQL requests,  
sample assemble in R

# Exercise 5.3 – 5.4

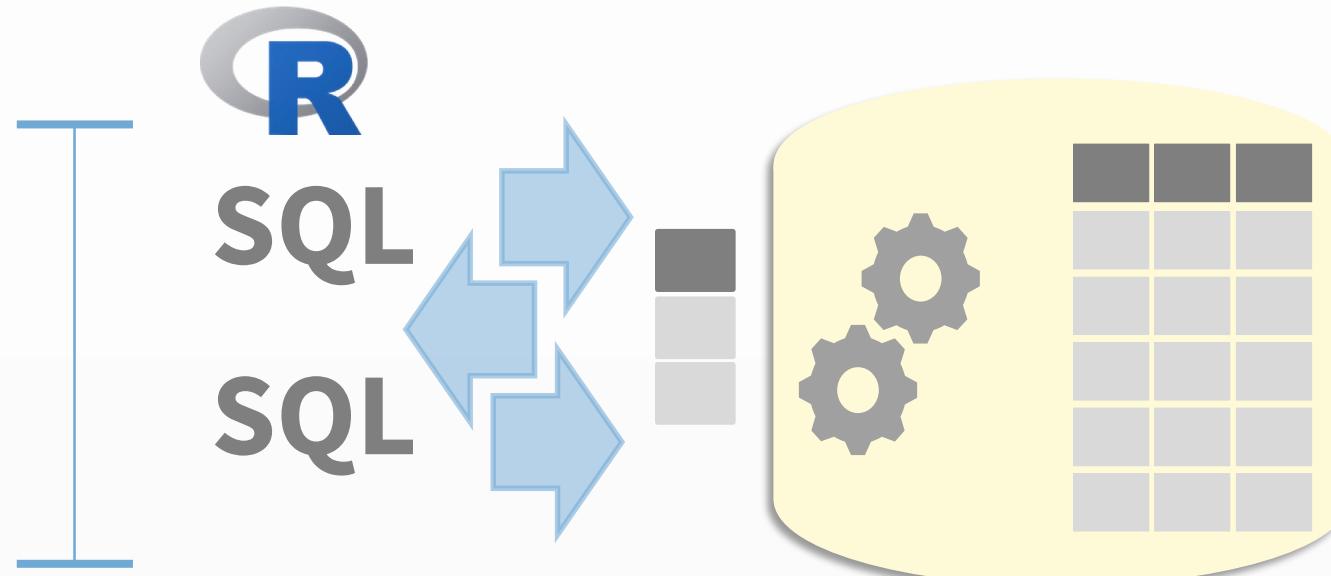
# Score inside the DB



# Exercise 5.5 – 5.6

# Model inside the Database

Runs  
multiple  
queries to fit  
the model



# Exercise 5.7

# Unit 6

## Advanced Operations



Photo by [Holly Stratton](#) on [Unsplash](#)

# Run same code? Create a [tidy] function

```
my_mean("arrtime",  
       flights) 
```

```
flights %>%  
  my_mean("arrtime") 
```

```
flights %>%  
  my_mean(arrtime) 
```

```
flights %>%  
  summarise(  
    m = mean(arrtime)  
)
```

# Tidy eval functions to remember

Prevent  
evaluation

`expr()`

Prevent evaluation  
of arguments

`enquo()`

Evaluate  
expression

`!!`

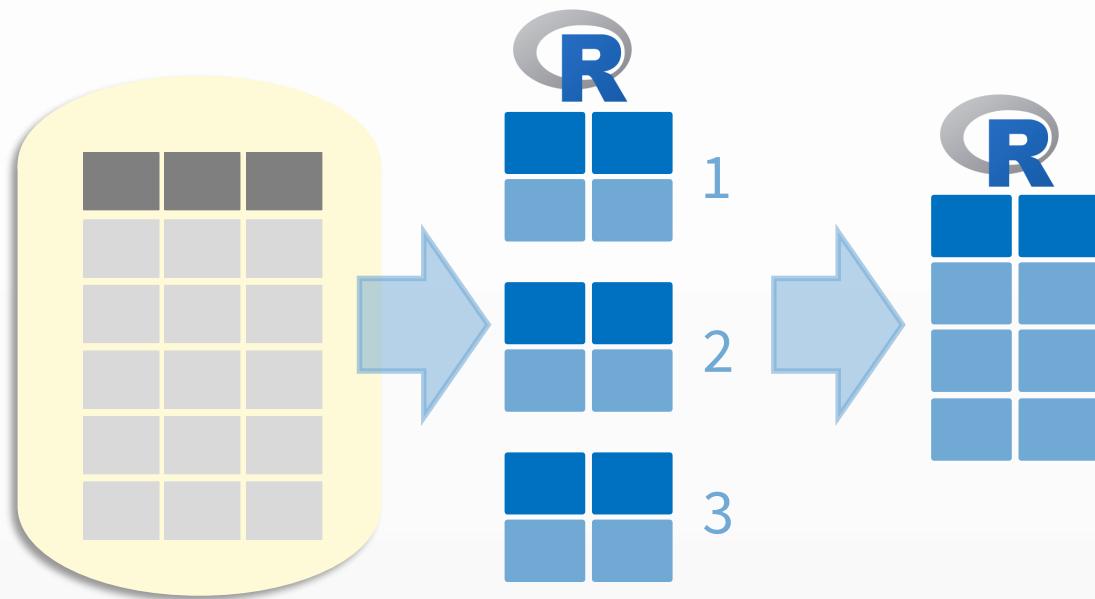
`enquos()`

`!!!`

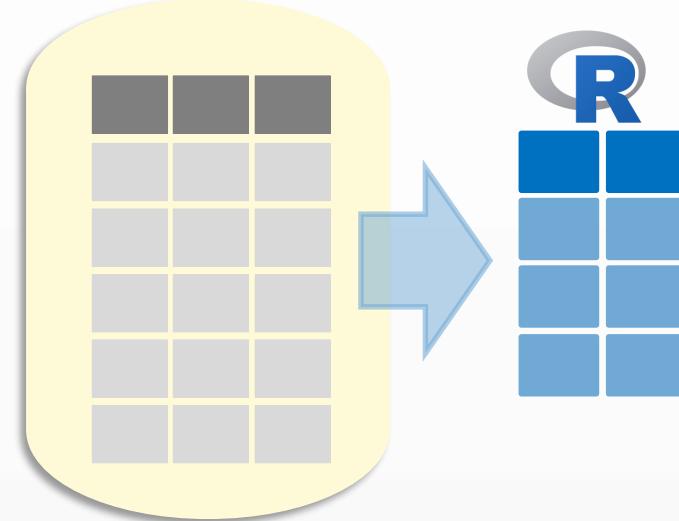
# Exercise 6.1 – 6.2

# Multiple queries

Many trips to the database



One trip to the database



# Map/Reduce ~~data~~ code

Many trips to the database

```
map(  
  dplyr → SQL  
  dplyr → SQL  
  dplyr → SQL  
)
```

One trip to the database

```
map(  
  expr( dplyr ) → SQL  
  expr( dplyr )  
  expr( dplyr )  
  ) %>%  
  reduce()
```

# Exercise 6.3 – 6.4

# Unit 7

## sparklyr

/s-par-klee-r/



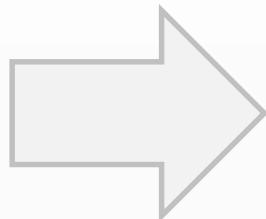
Photo by [Matthew Ronder-Seid](#) on [Unsplash](#)

# What is Spark?

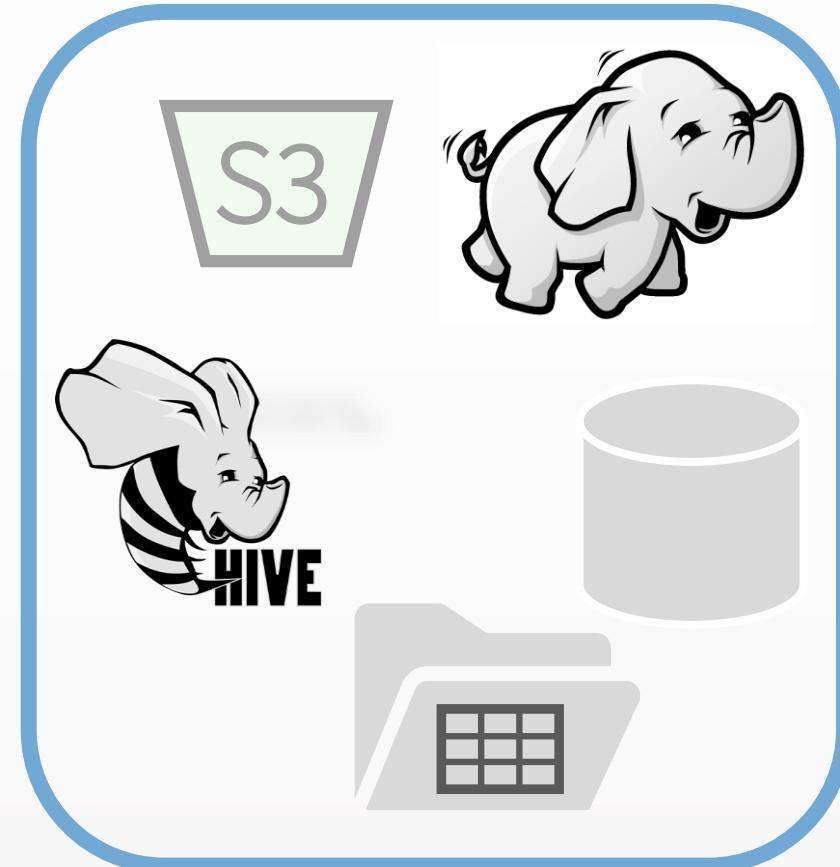
## Processing



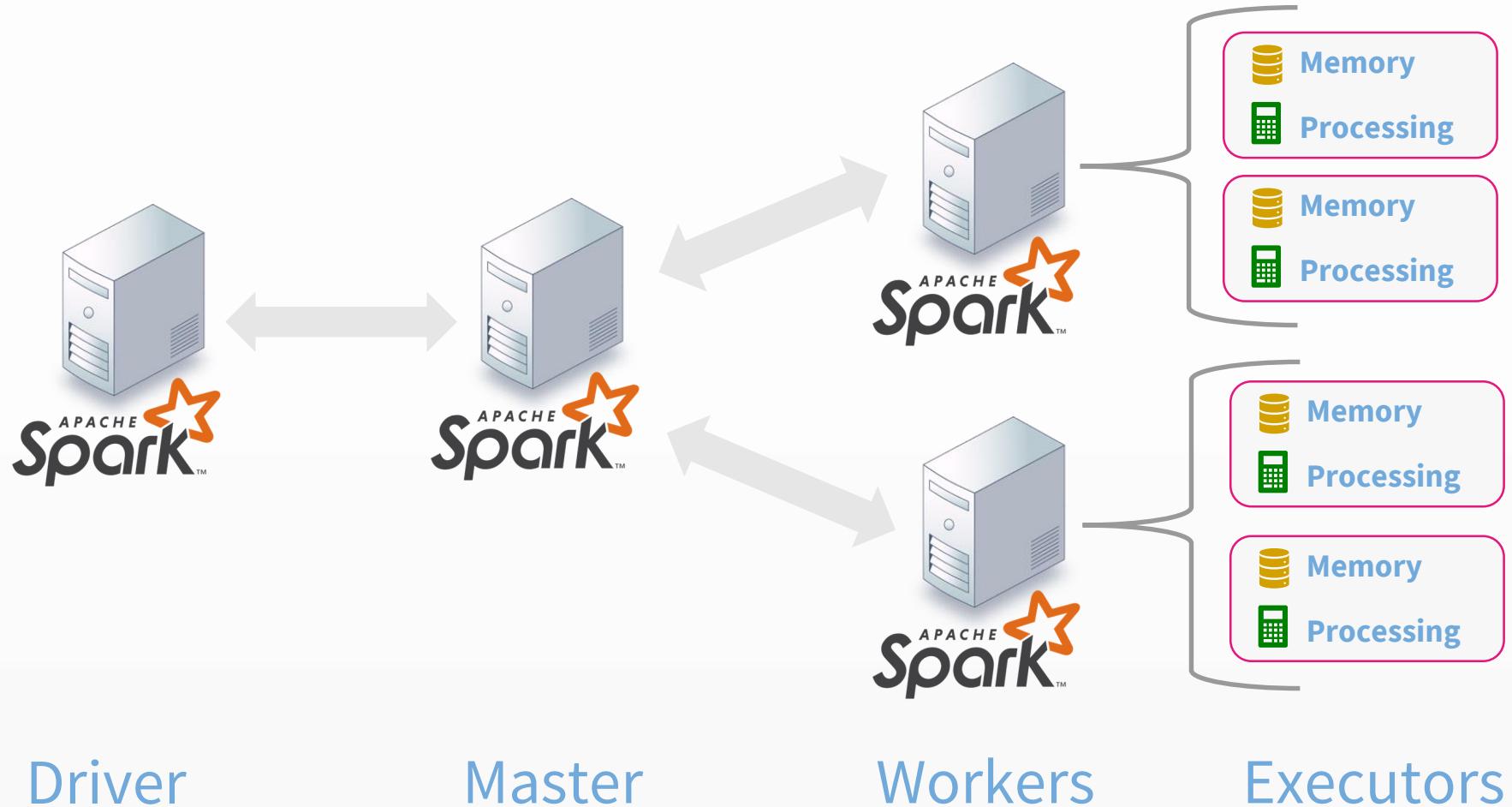
- Cluster Computing
- Machine Learning
- SQL Interface
- Extensible API



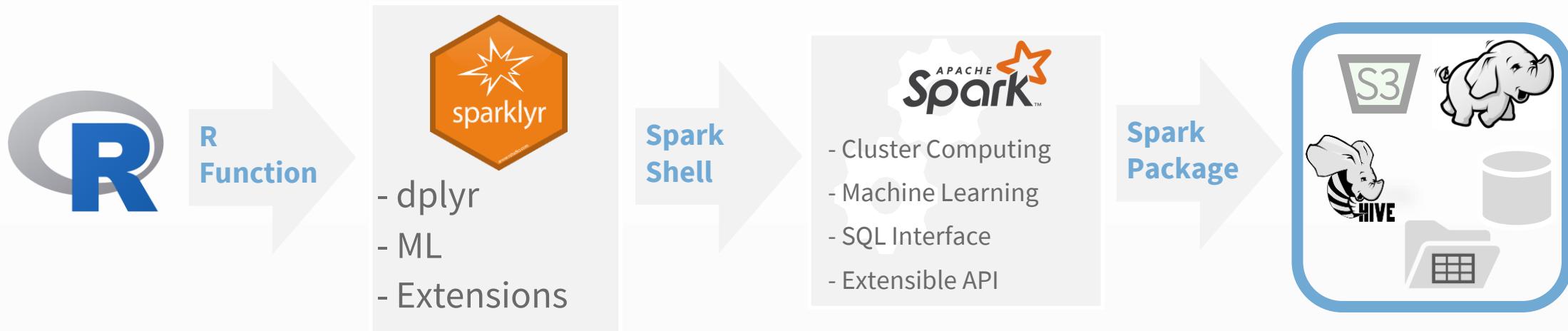
## Storage



# Typical architecture



# sparklyr – An R interface for Spark

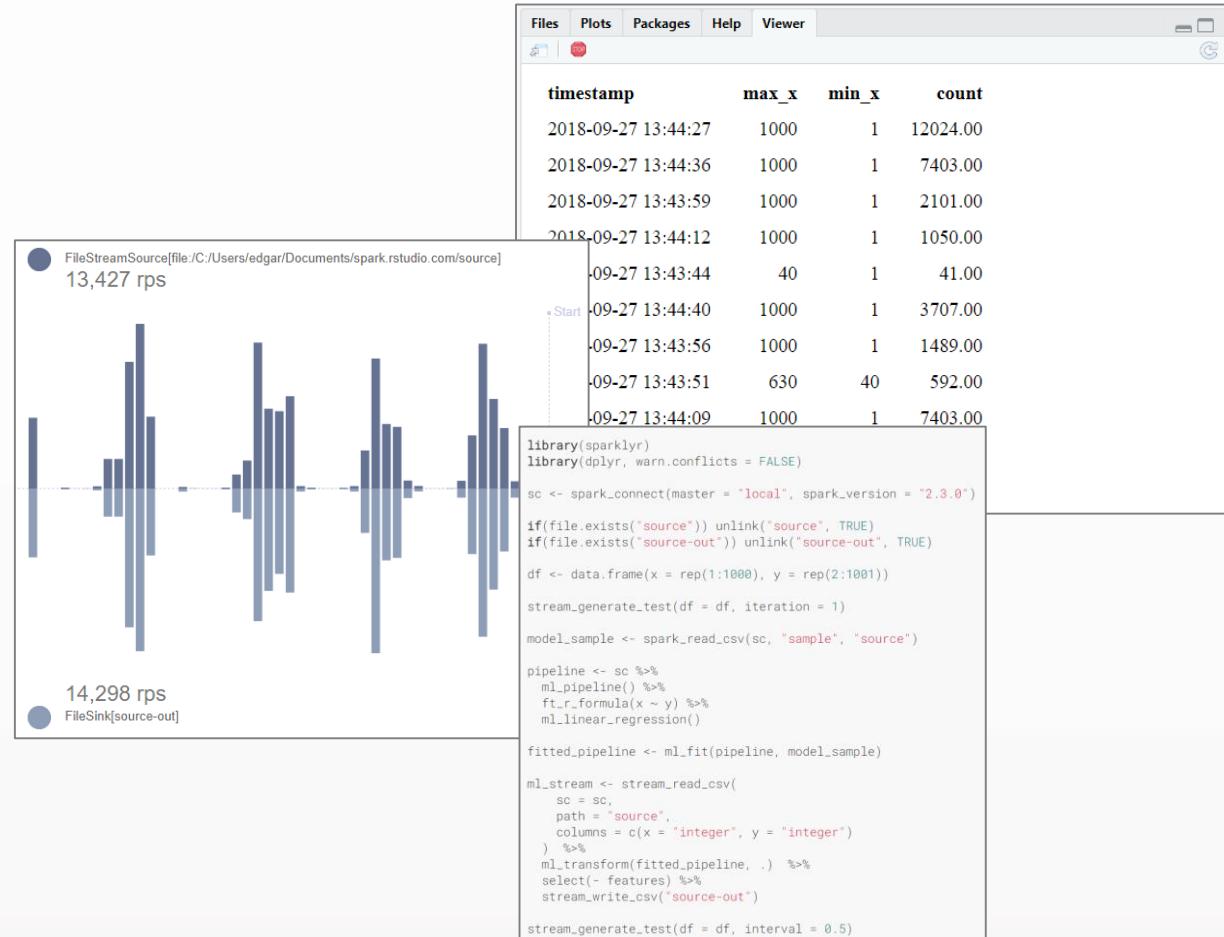


# What does it offer?



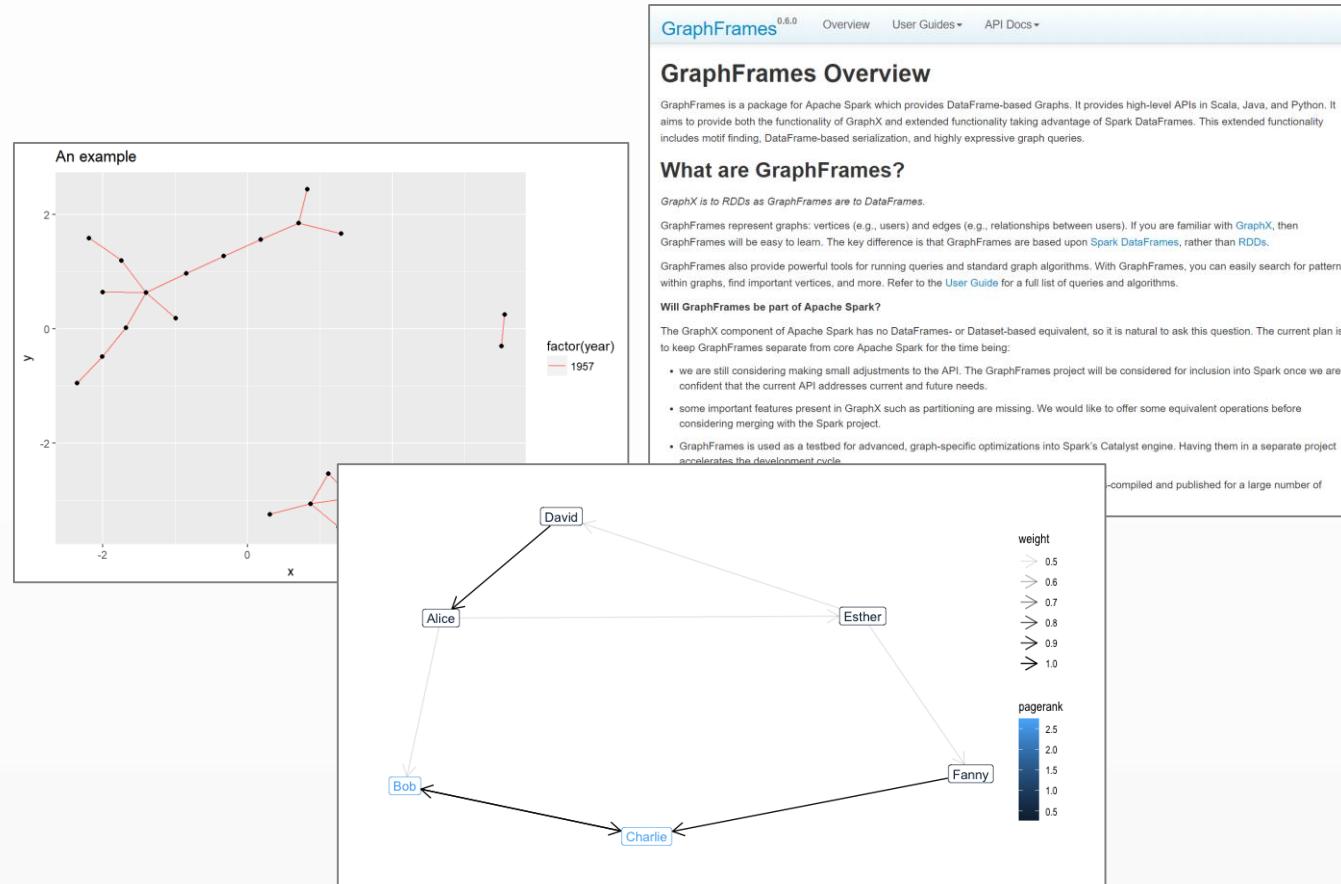
# Streaming

- Ability to run dplyr, SQL, and PipelineModels against a stream
- Read & write stream results to Spark memory and files
- An out-of-the box graph visualization to monitor the stream
- reactiveSpark() function allows Shiny apps to poll the contents of the stream



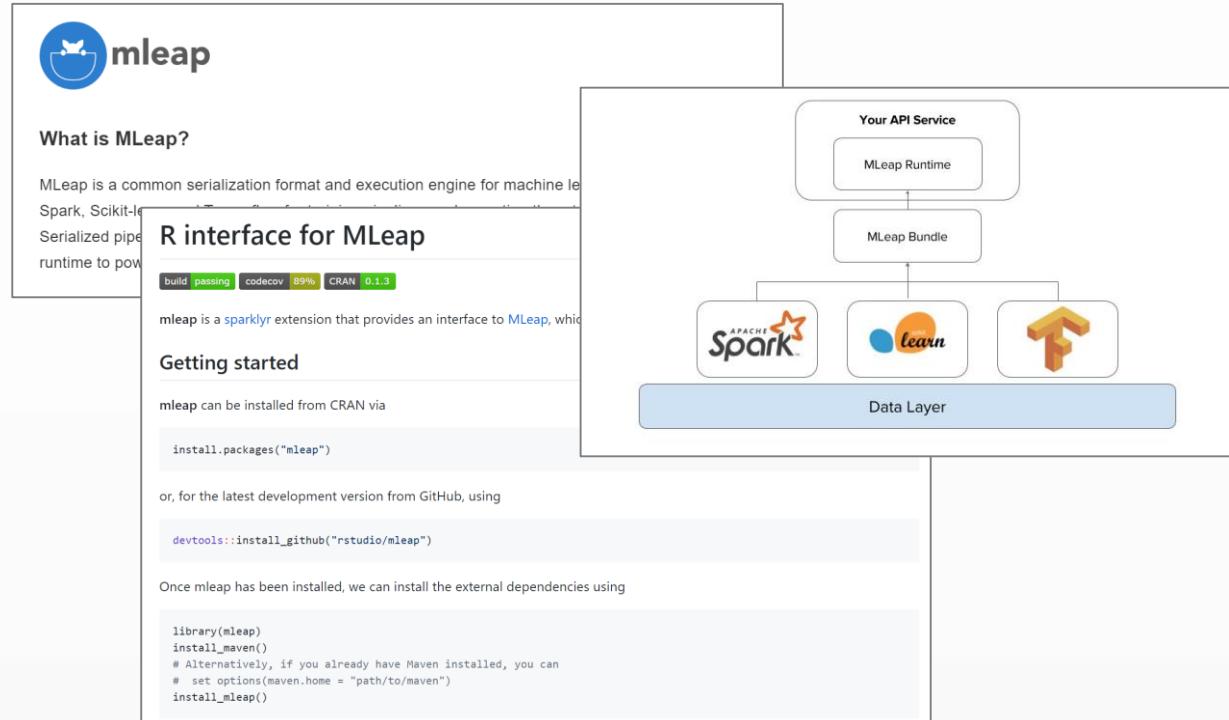
# Graph analysis

- Support for GraphFrames which aims to provide the functionality of GraphX.
- Perform graph algorithms such as: PageRank, ShortestPaths and many others
- Designed to work with sparklyr and the sparklyr extensions



# Productions pipelines with Mleap

A sparklyr extension that provides an interface to MLeap, which allows us to take Spark pipelines to production.



# Coming soon...xgboost

**sparkxgb** is a  
sparklyr extension  
that provides an  
interface to  
XGBoost on Spark.

The image shows two screenshots of a GitHub repository. The top screenshot is the repository page for **sparkxgb**, which is a **XGBoost** eXtreme Gradient Boosting library. It features a header with the repository name, a build status badge (failing), and links to Community, Documentation, Resources, Contributors, and Release Notes. Below the header is a brief description of XGBoost and its purpose. The bottom screenshot is the **README** file for the repository, which contains a heading for **sparkxgb** and a paragraph explaining it is a **sparklyr** extension. It also includes a heading for **Installation** and a code block showing the R command to install the development version.

**sparkxgb** is a [sparklyr](#) extension that provides an interface to [XGBoost](#) on Spark.

## Installation

You can install the development version of sparkxgb with:

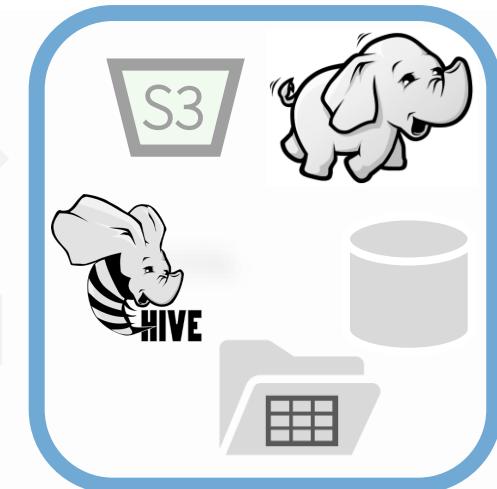
```
# sparkxgb requires the development version of sparklyr
devtools::install_github("rstudio/sparklyr")
devtools::install_github("kevinykuo/sparkxgb")
```

# Exercise 7.1 – 7.3

# Working with data in Spark

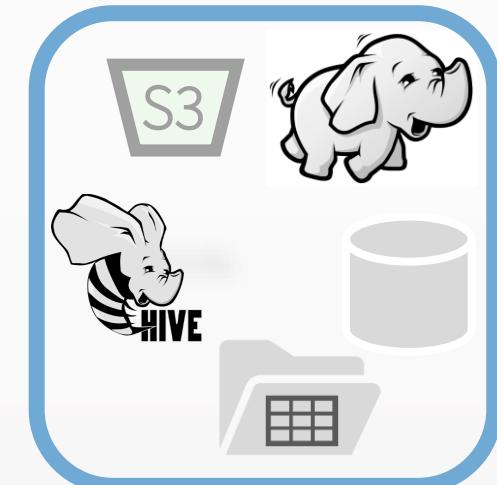
## Option 1

Use Spark as a pass-through for each query



## Option 2

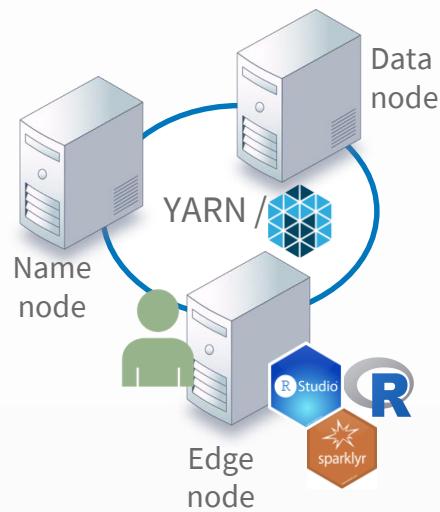
Cache the data into Spark memory & query there



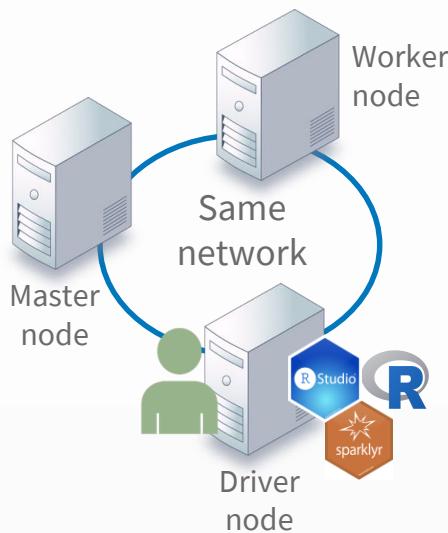
# Exercise 7.4 – 7.9

# Deployment options

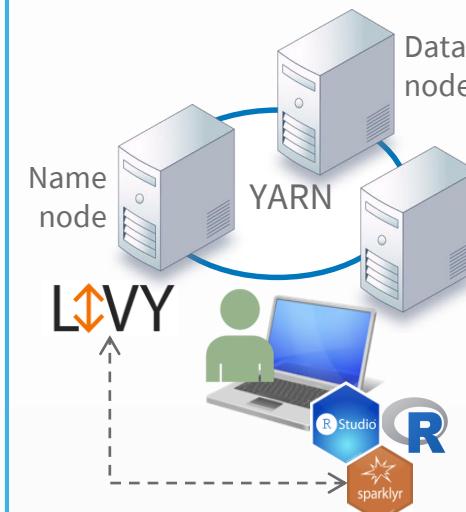
Managed Cluster



Stand Alone



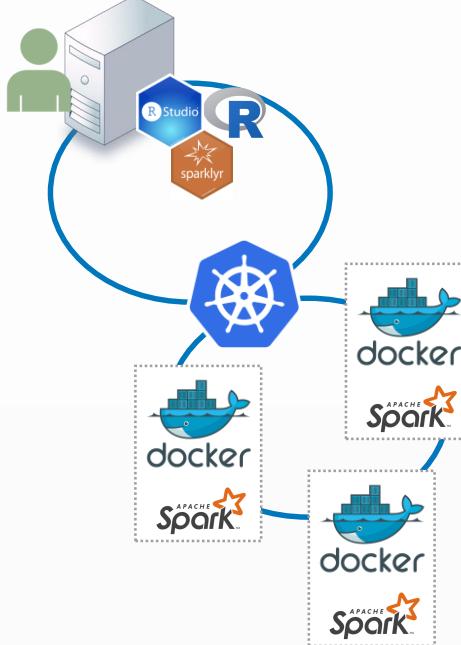
Livy



Local



Kubernetes



- Deployment seen at most business
- Spark version(s) available are limited to what's on the cluster

- Since there's no central data repository, all data has to be either imported or connected to a common shared location (NAS, S3)

- Great for accessing a remote cluster
- Not recommended for Production deployments

- Great for learning
- Works on Windows and Mac too
- Quick and easy way to access multiple cores

# Let's talk about Data Science projects

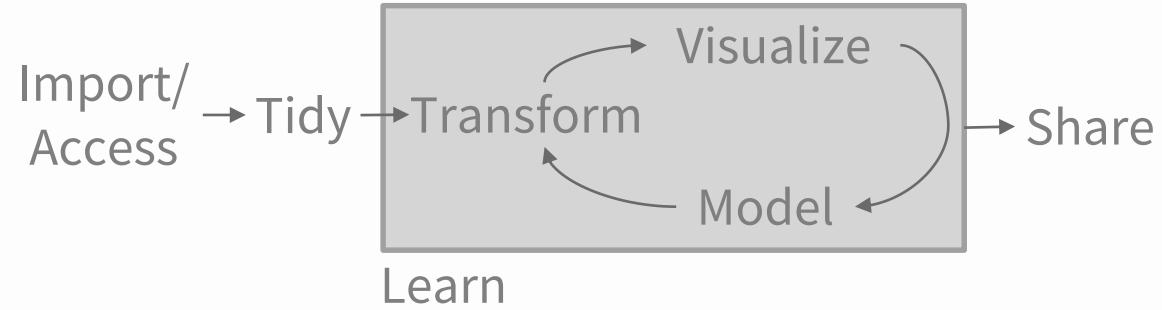


Photo by [Jo Szczepanska](#) on [Unsplash](#)

# Different deliverables

## Data Science

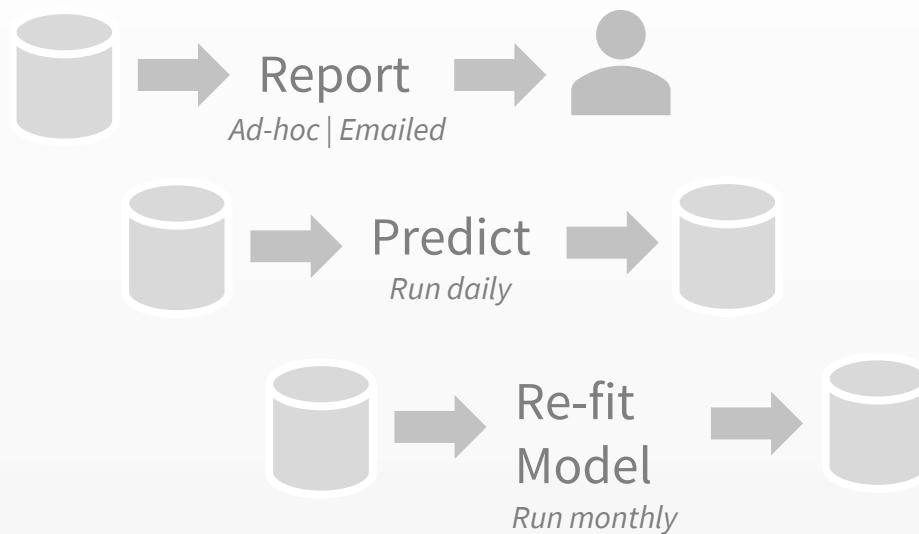
- Deliverable: **Insights**
- Experimental
- Iterative



---

## Production

- Deliverable: **Software**
- Tested
- Automated
- Apply SDLC



# Unit 8

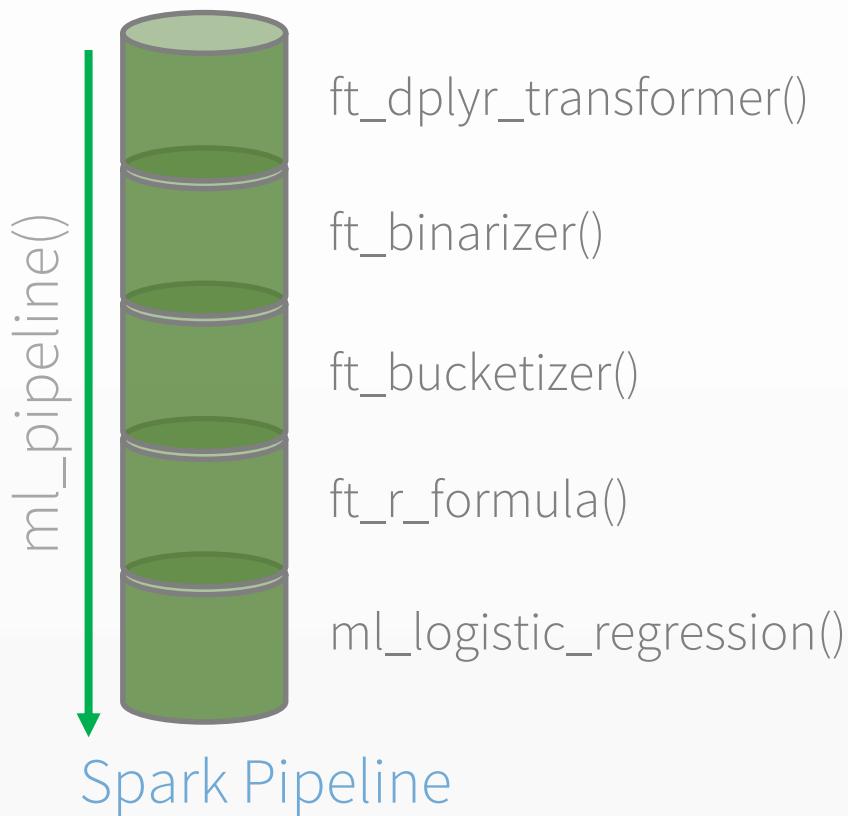
## Spark Pipelines



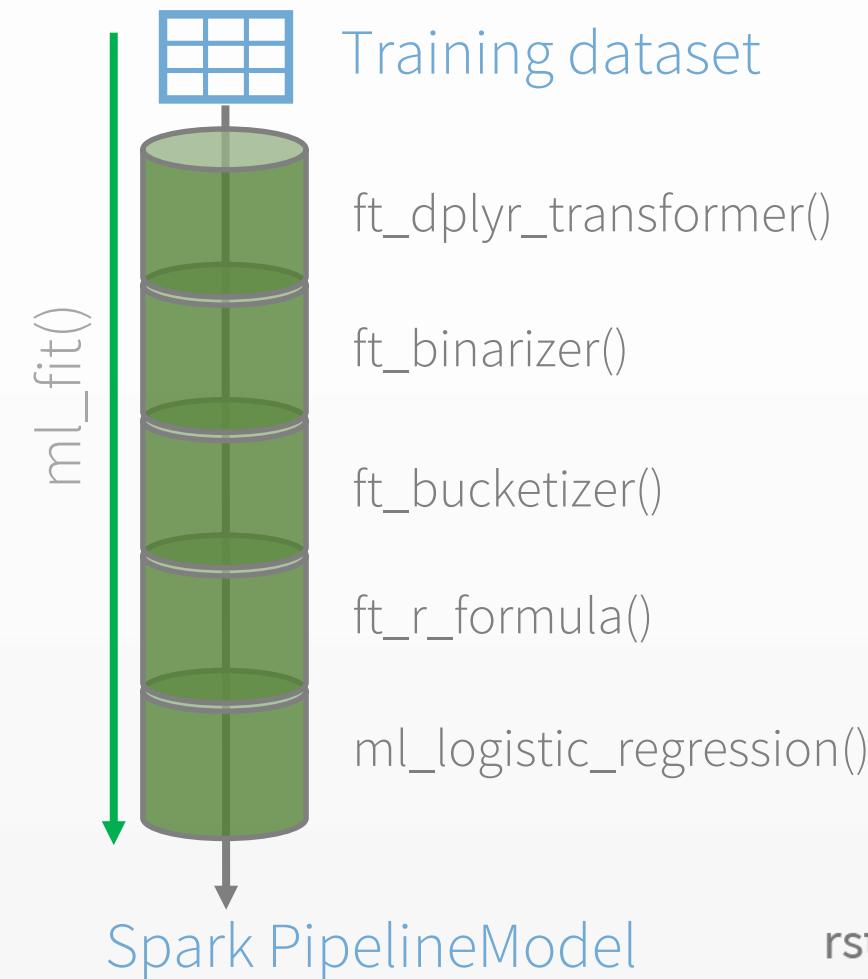
Photo by [Iker Urteaga](#) on [Unsplash](#)

# Spark pipelines types

## Estimator (Plan)

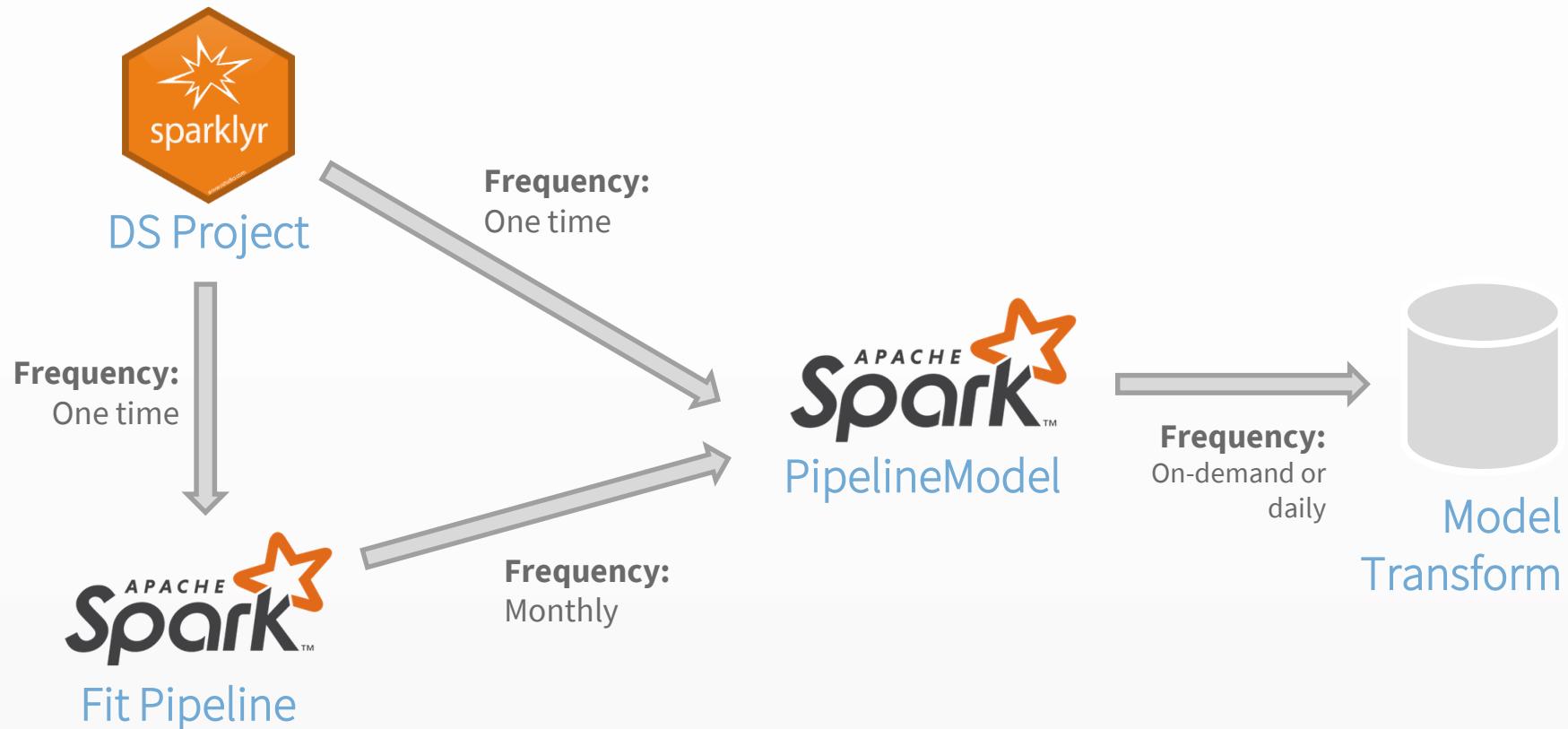


## Transformer (Fit)



# Exercise 8.1 – 8.4

# Production Implementation



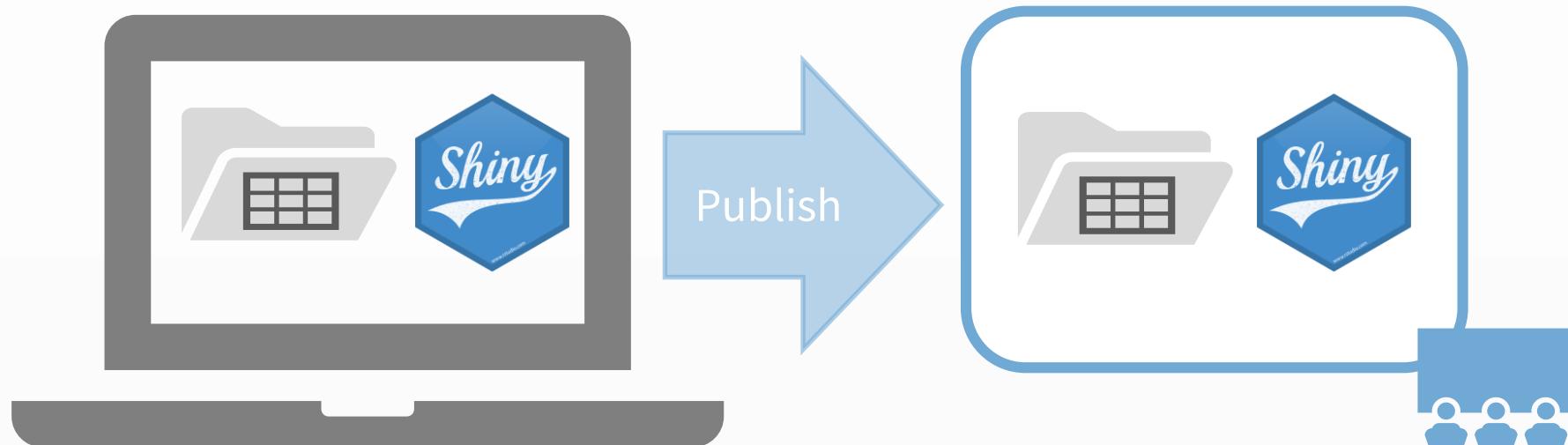
# Units 9 & 10

## Dashboards

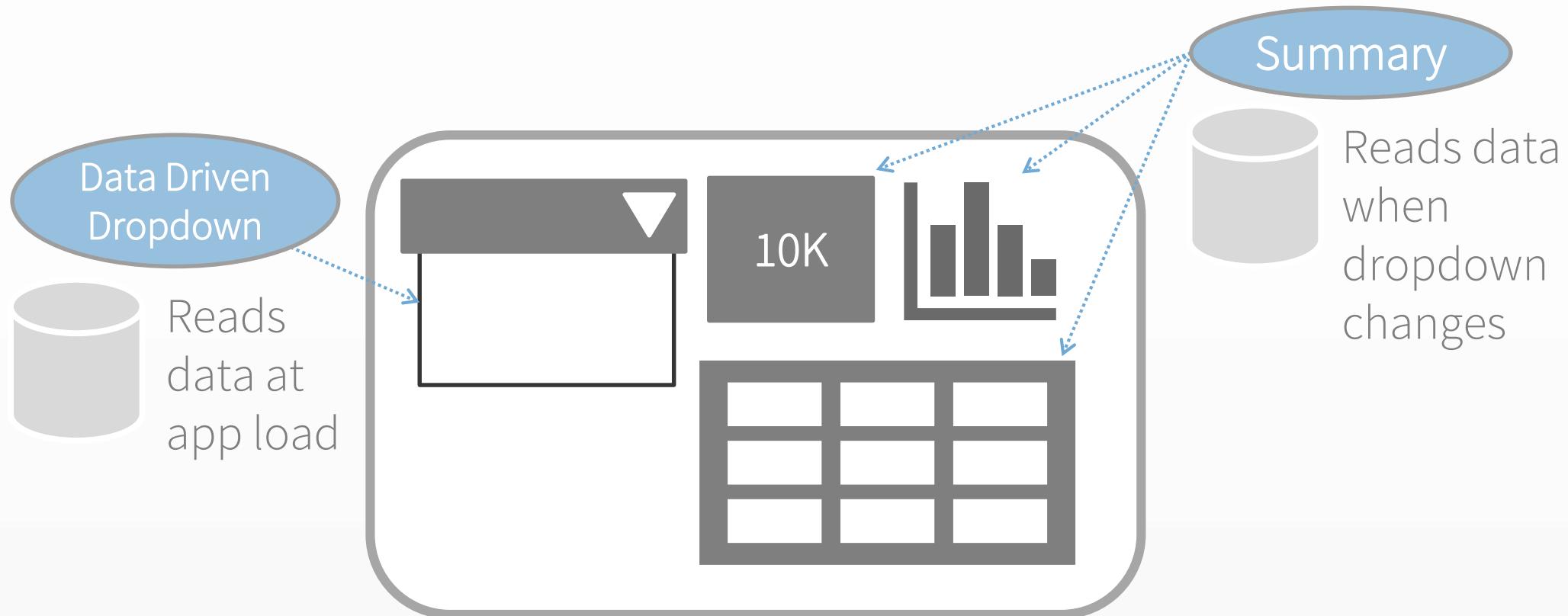


Photo by [Benjamin Child](#) on [Unsplash](#)

# Normal Shiny app

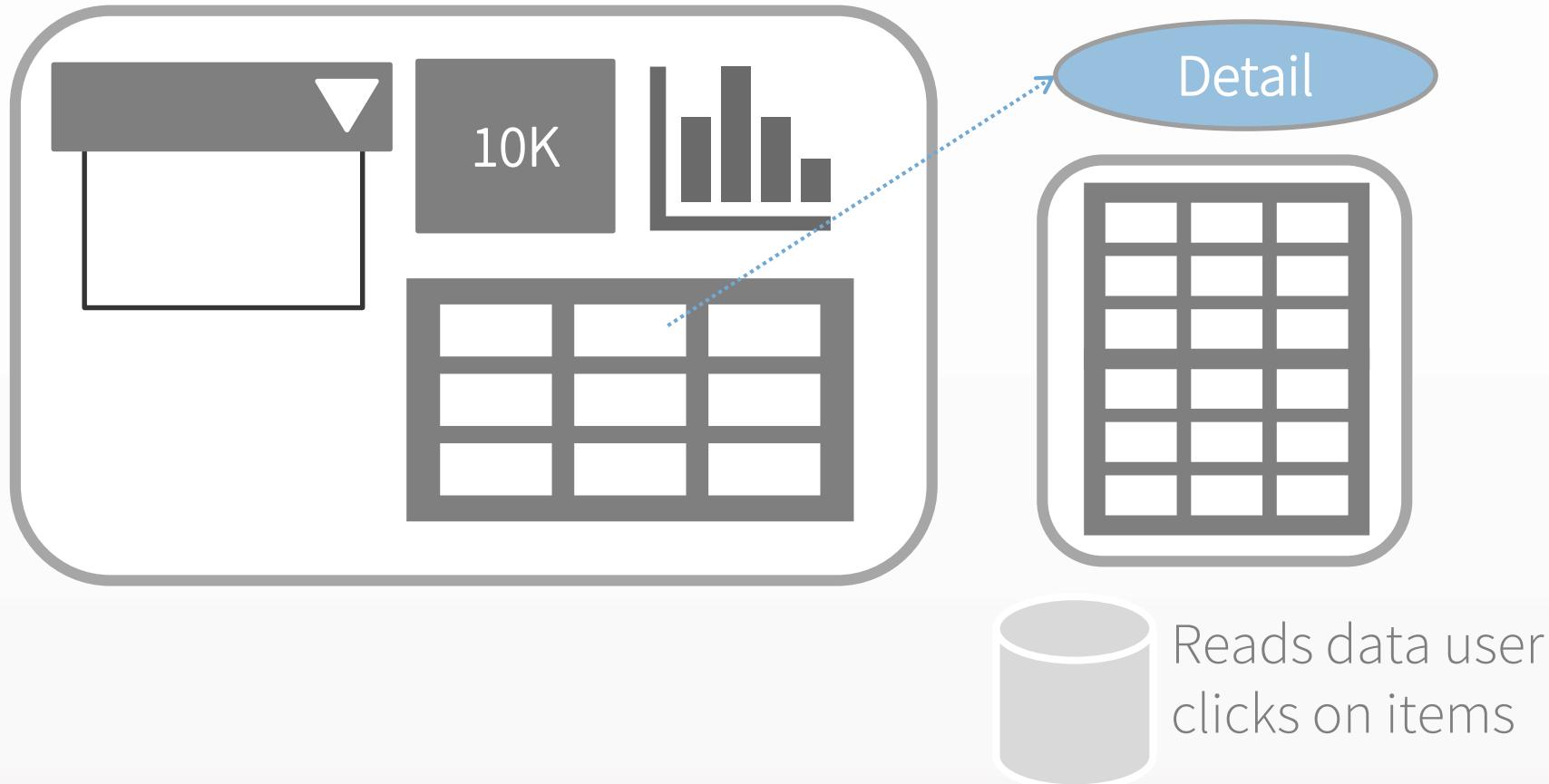


# Database + Dashboard



# Exercise 9.1 – 9.4

# Database + Dashboard



# Exercise 10.1 - 10.4

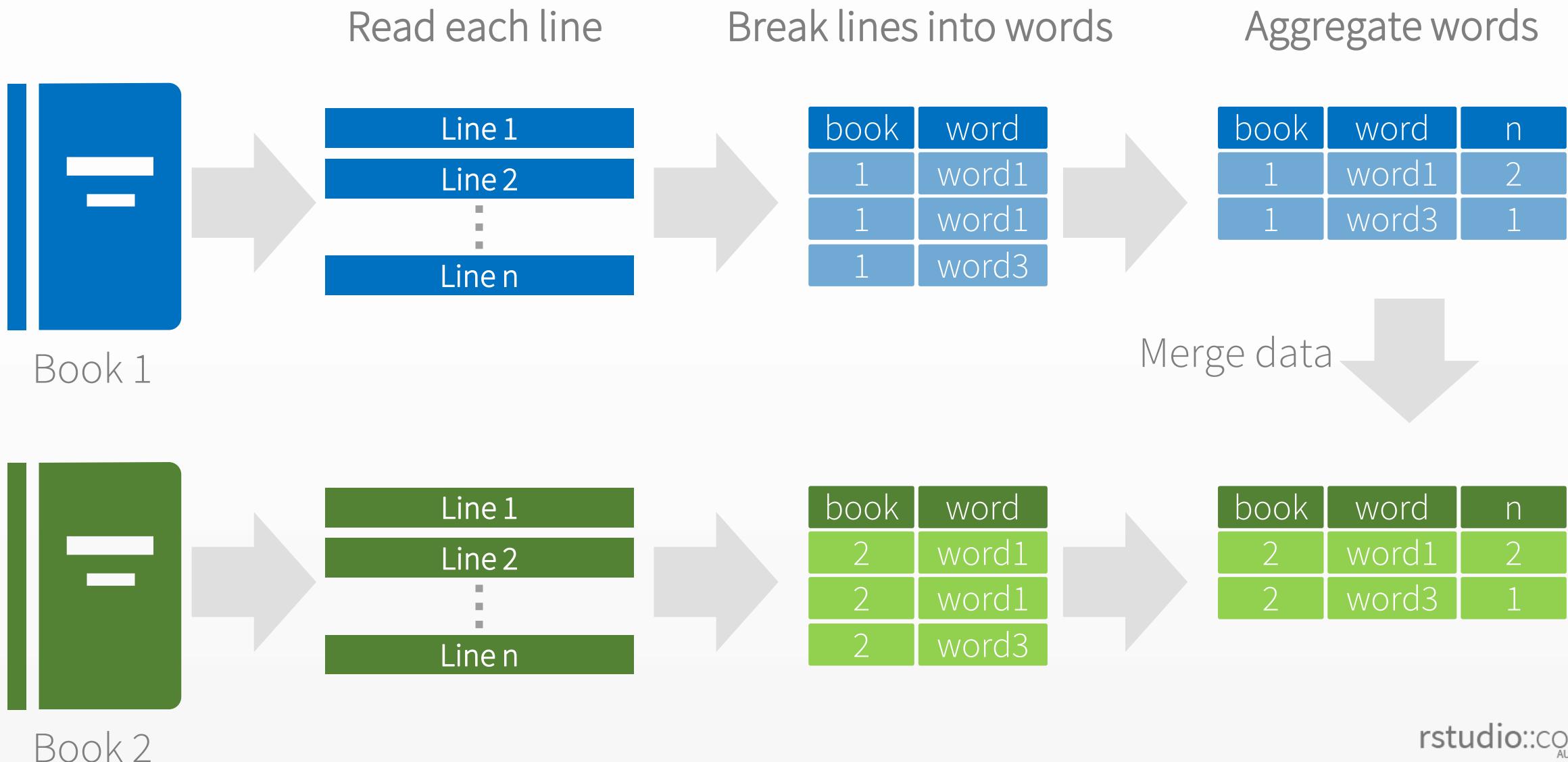
# Unit 11

## Text mining



Photo by [Tucker Good](#) on [Unsplash](#)

# Compare text of two books



# General advice



Photo by [Daria Nepriakhina](#) on [Unsplash](#)

# Bookmark and check regularly

- <http://db.rstudio.com/>
- <http://spark.rstudio.com/>
- <https://www.tidyverse.org/>
- <https://rviews.rstudio.com/>
- <https://rviews.rstudio.com/categories/databases>
- <https://blog.rstudio.com/>

# Join the community!

R Studio Community

all categories ► all tags ► Categories Latest New (12) Unread Top

Category	Topics	Latest
 <b>rstudio::conf 2018</b> This category is for anything and everything related to rstudio::conf.	4 / week 2 new	 How can I connect R with v... application • new rstudio
 <b>tidyverse</b> This category is for anything and everything about the tidyverse.	23 / week	 □ Crash when quitting ■ RStudio IDE bug
 <b>RStudio IDE</b> This category is for discussing the RStudio IDE, both	16 / week 3 new	 □ Is there a way to measure • new

<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/edgararuiz/tidypredict>
- <https://github.com/rstudio/sparklyr>

Thank  
you!!!!



Photo by [Gary Bendig](#) on [Unsplash](#)