# Welcome to!

# Housekeeping items

- rstudio::conf app
- Wi-fi password

# Wifi password

**ssid:** rstudio20

**pwd:** tidyverse20

# Schedule

**9am – 10:30am**

Break (30 mins)

**11am – 12:30pm**

Lunch (1 hr)

**1:30pm – 3 pm**

Break (30 mins)

**3:30pm – 5 pm**

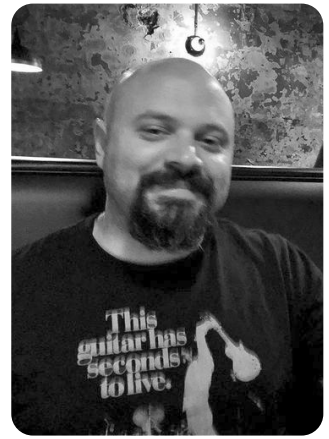
# The team

Marijulie Martinez

*TA*

Alex Gold

*TA*




Taner Alkaya

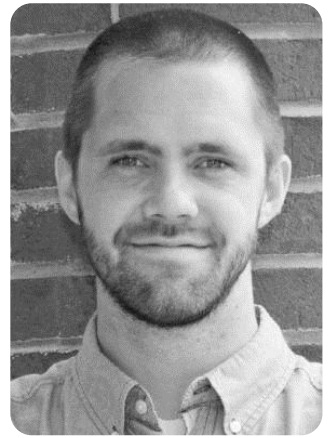
*TA*

James Blair

*Instructor*

Edgar Ruiz

*Instructor*



Cole Arendt

*Infrastructure*

# Introductions

# Class materials

- Server
- Database
- Deck
- Exercise book



github.com/rstudio-conf-2020/big-data

# Asking for help

I'm stuck



I'm not stuck, but I need help on my computer

I need help understanding something (which likely means others do too)

# Class server

**Link:** rstd.io/class

**Identifier:** big_data

# Units 1 & 2

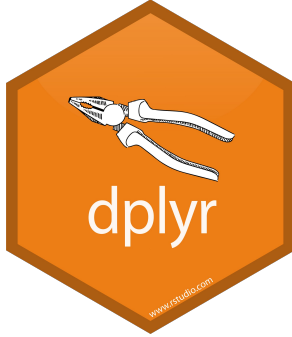
# Large files


Photo by Viktor Talashuk on Unsplash

# `dplyr` package

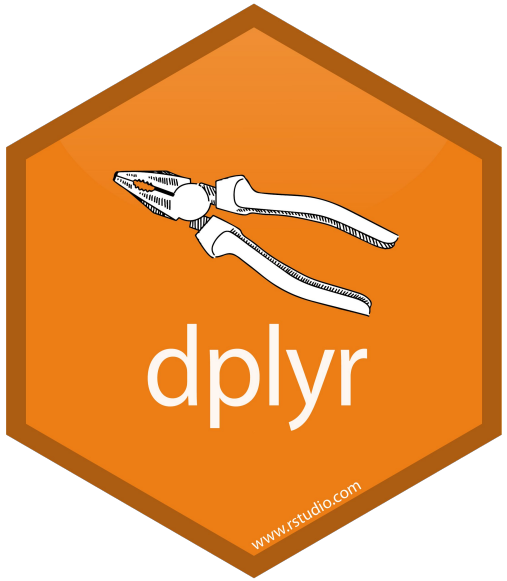
1. A grammar of data manipulation
2. Designed to **abstract over how the data is stored**
3. Consistent function interface

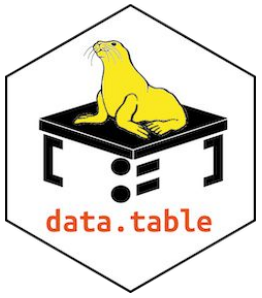
# `dplyr` package



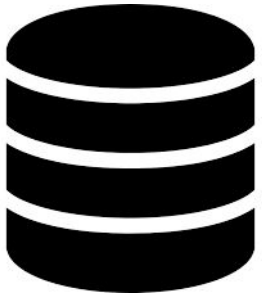

*Dplyr "abstracts away how your data is stored, so that you can work with data frames, data tables and remote databases using the same set of functions.* ***This lets you focus on what you want to achieve, not on the logistics of data storage.***"

# dplyr backends



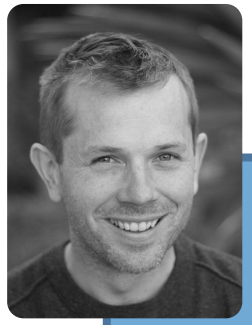







# Apache Arrow

1. Cross-language platform for in-memory data
2. Exciting applications for "big data"
3. `dplyr` compliant



**Accelerating Analytics with Apache Arrow**

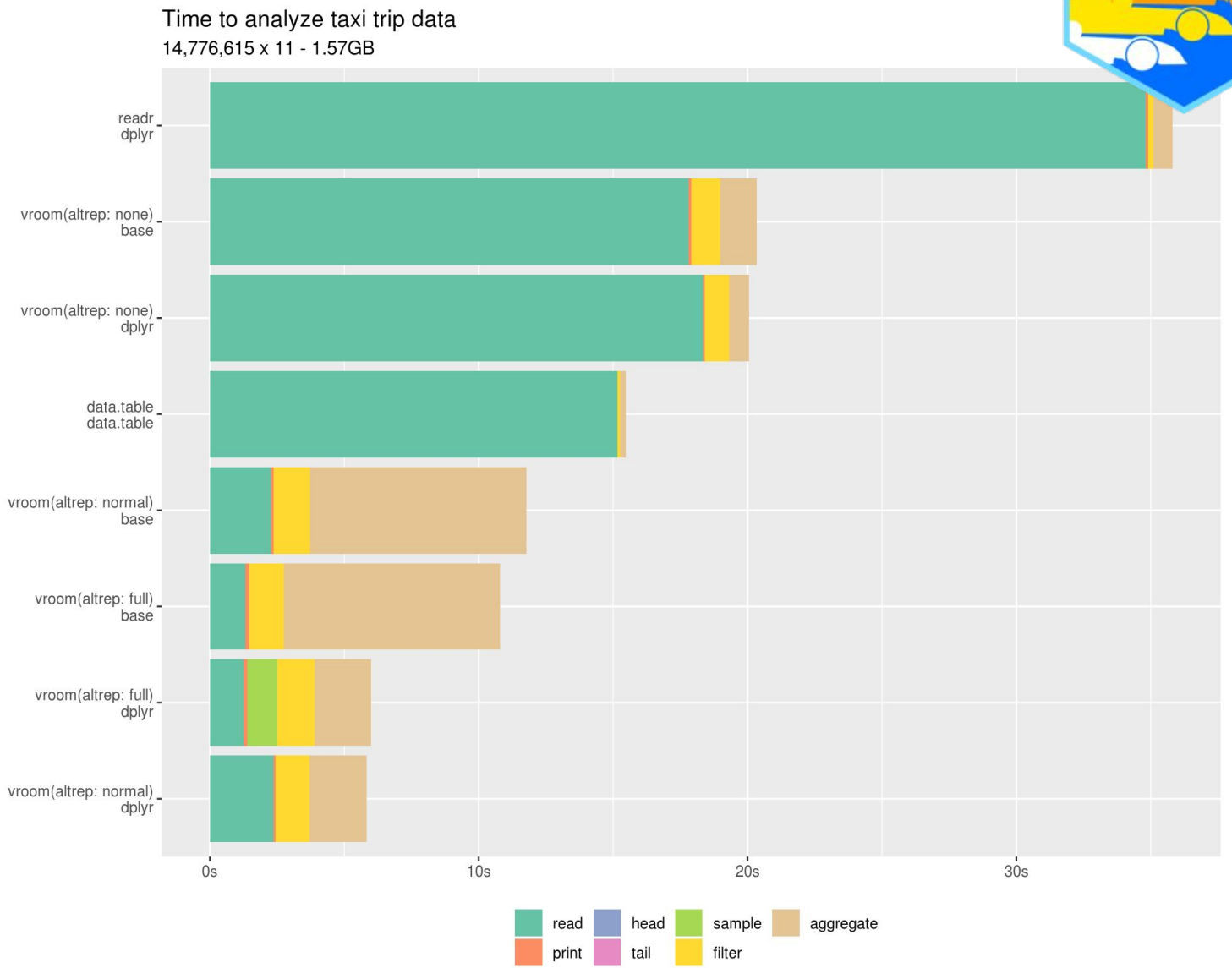
*Neal Richardson*

Wednesday, January 29, 2020
2:15-2:37pm

arrow.apache.org/docs/r/

# `vroom` package

1. Initially indexes data but does not read it
2. Loads the data into R only when needed
3. Super fast! 1.27 GB/sec






vroom.r-lib.org

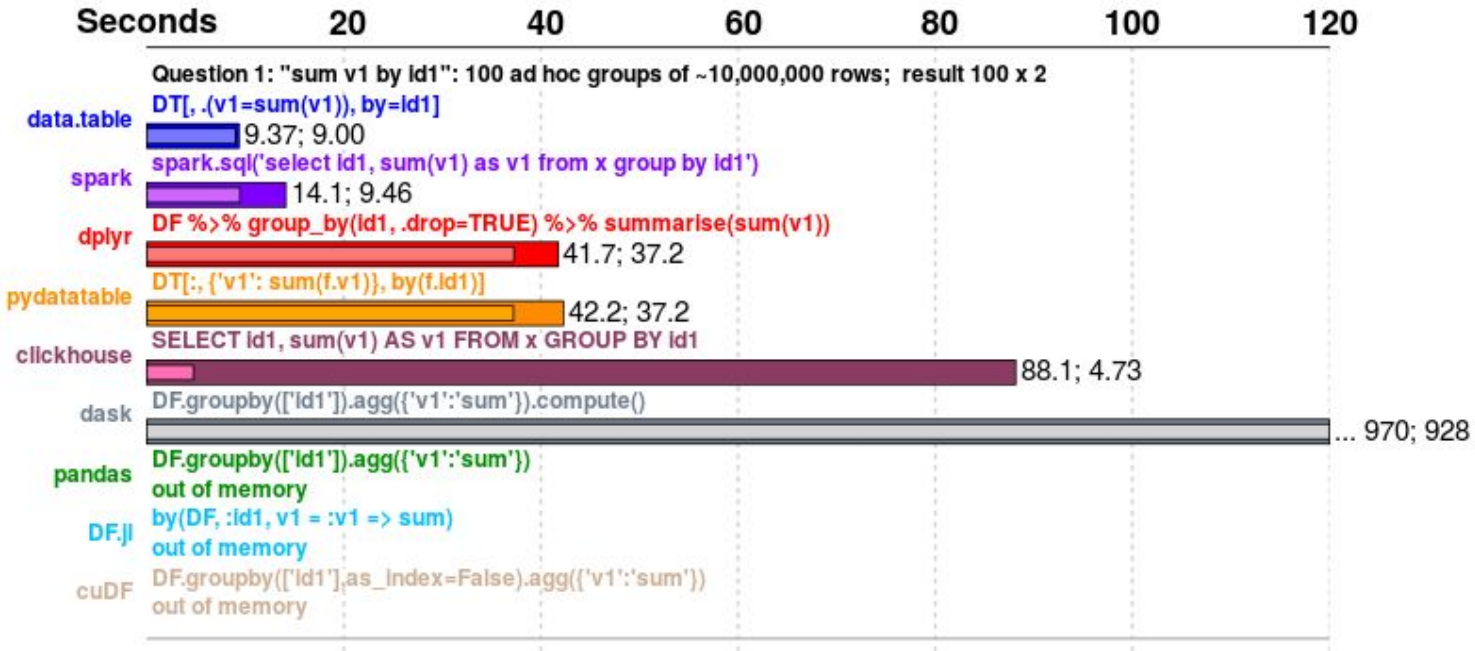
# `vroom` features

1. Nearly all parsing features of `readr`
2. `skip` and `n_max` arguments
3. Column selection
4. Read from multiple files or connections

# Exercise 1.1 – 1.3

# `data.table` package

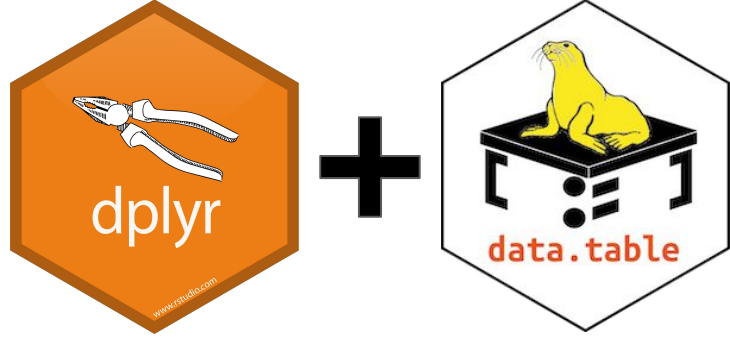
1. High performance version of base R `data.frame`
2. Fast file reader `fread`
3. Concise syntax `DT[i, j, by]`





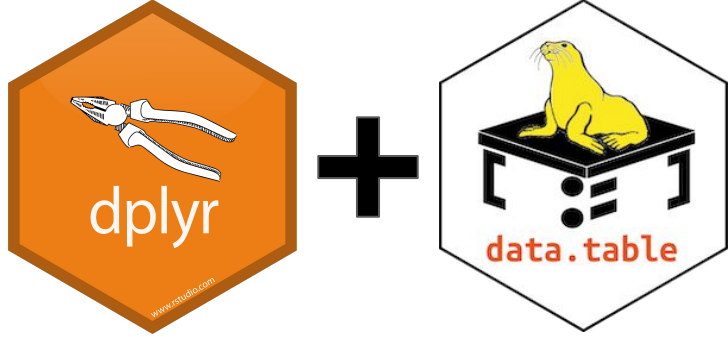

rdatatable.gitlab.io/data.table

# `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.*

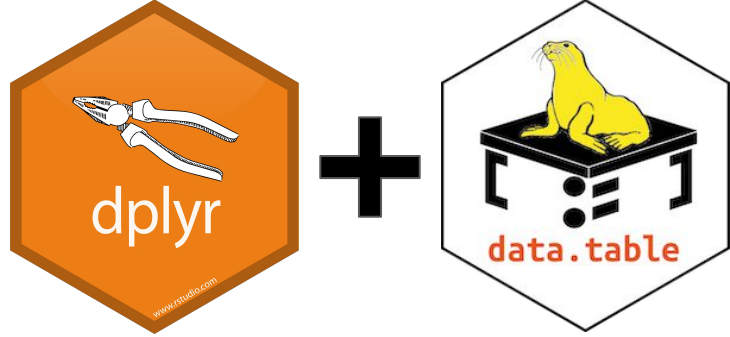
# `dtplyr` package




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

# Exercise 2.1 – 2.2

# `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.

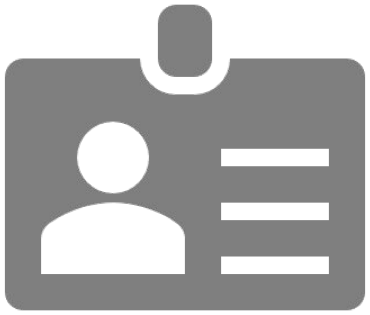
# Exercise 2.3 – 2.5

# Unit 3
# Accessing databases


Photo by Florian Pircher on Unsplash

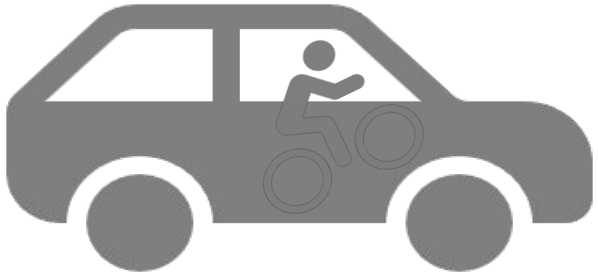
RStudio

# Connection requirements



Credentials

Location

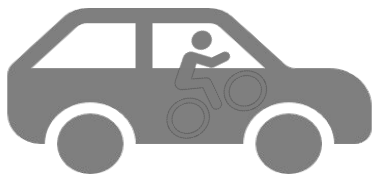


Driver

# Requirement definitions

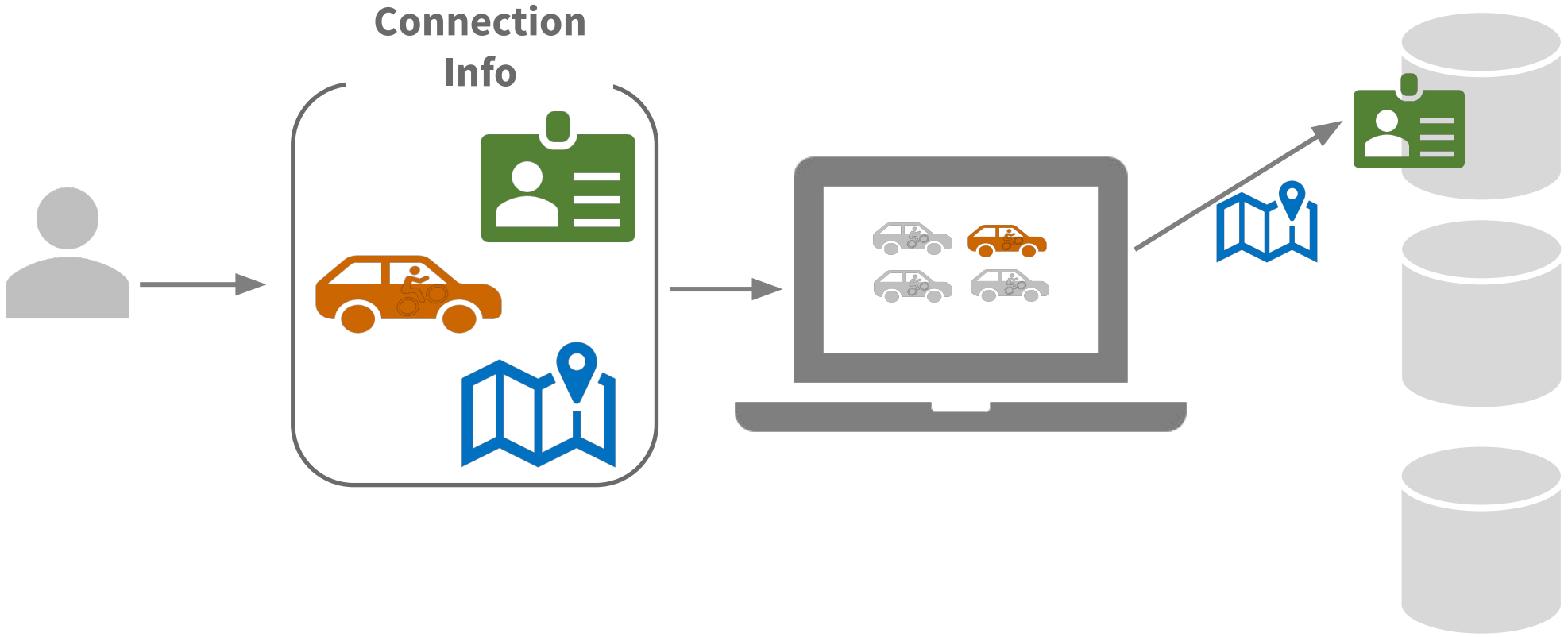
- User name & password
- Token

- URL
- IP Address

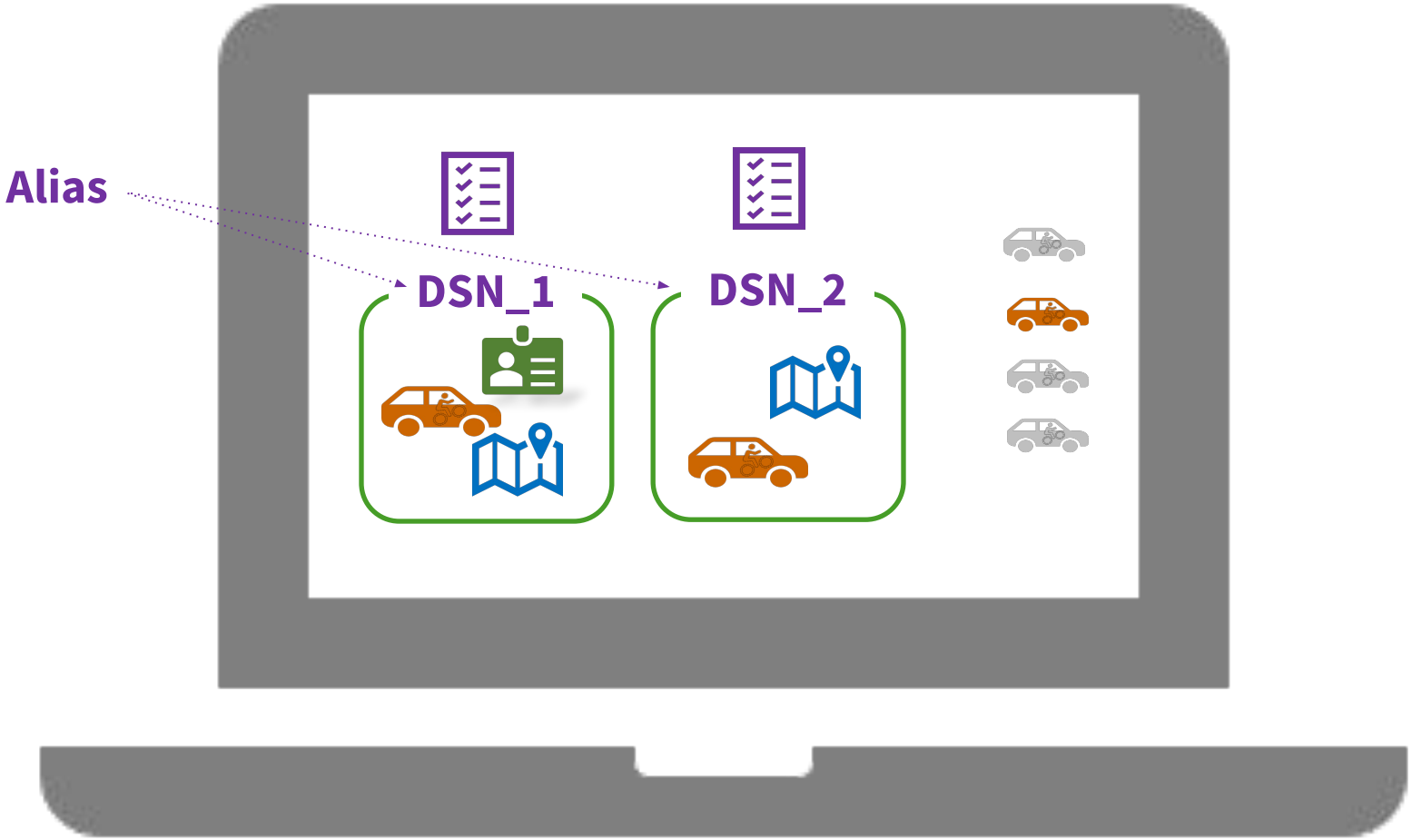


- ODBC (Used by **ADO** & **OLE DB**)
- JDBC

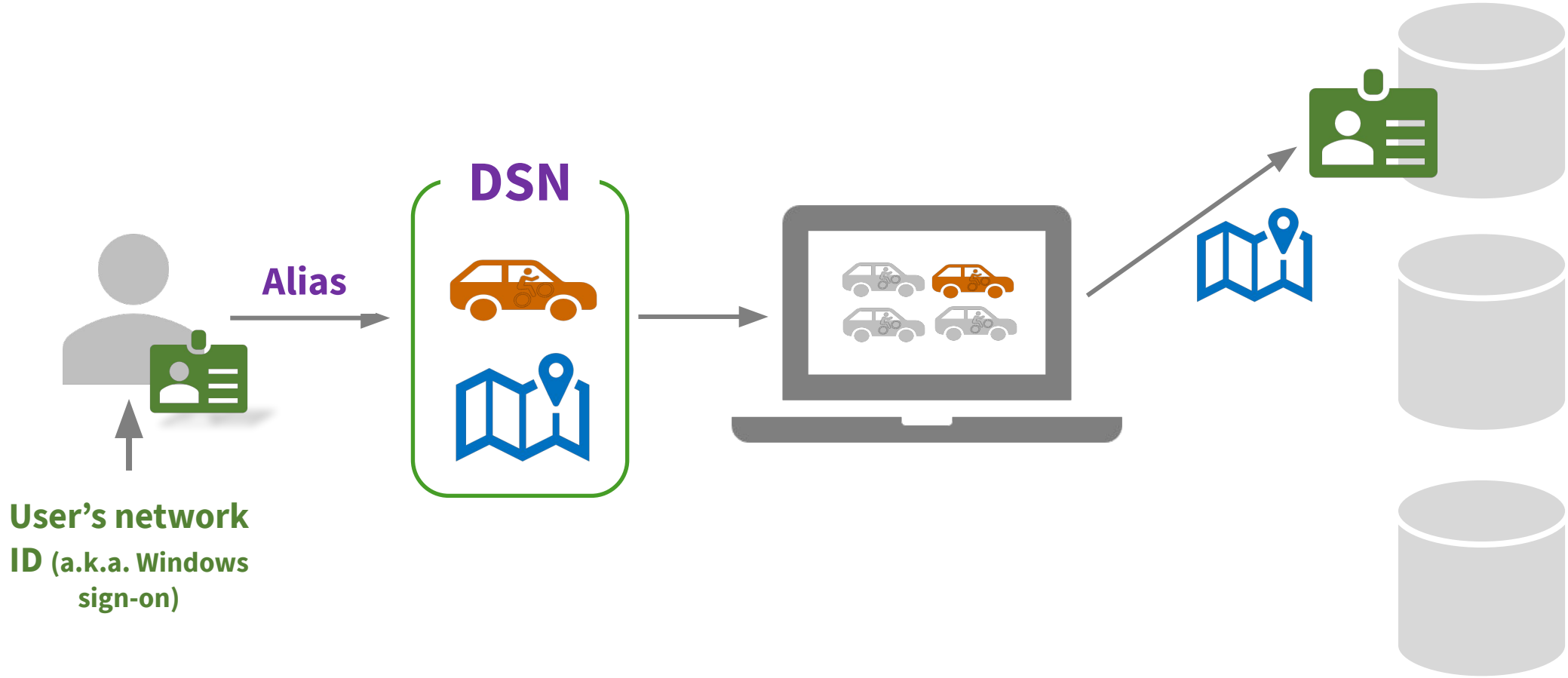
# Connection info



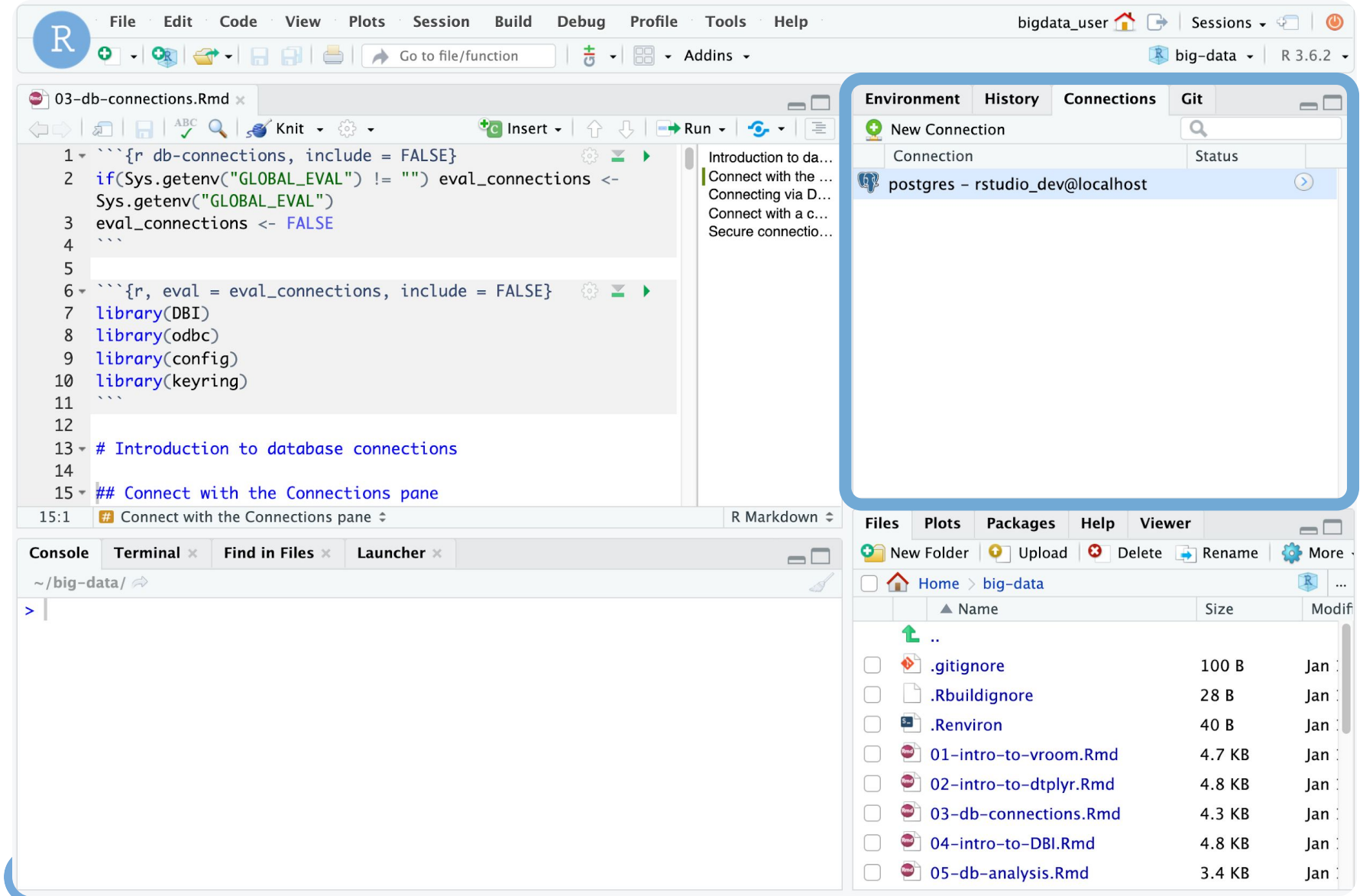
# Data Source Name (DSN)

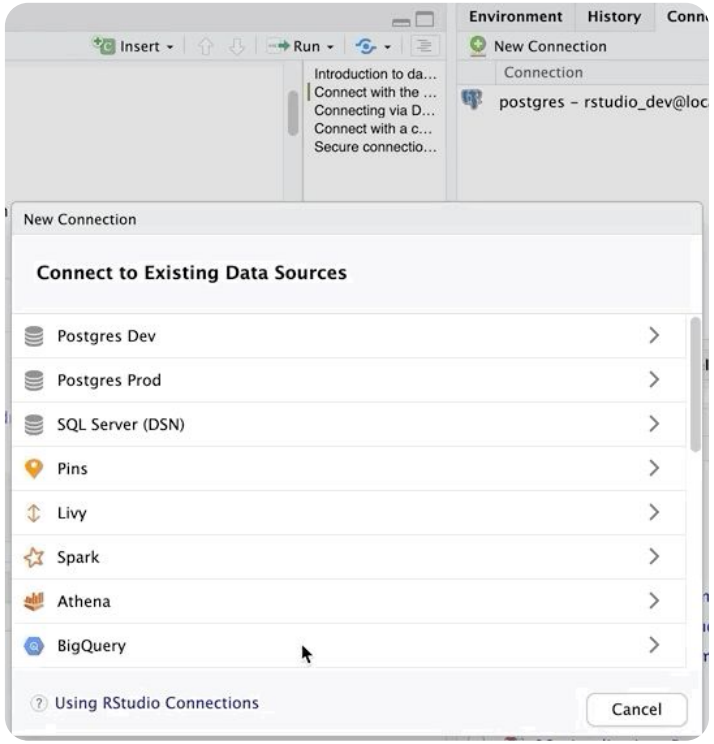


# The ideal connection

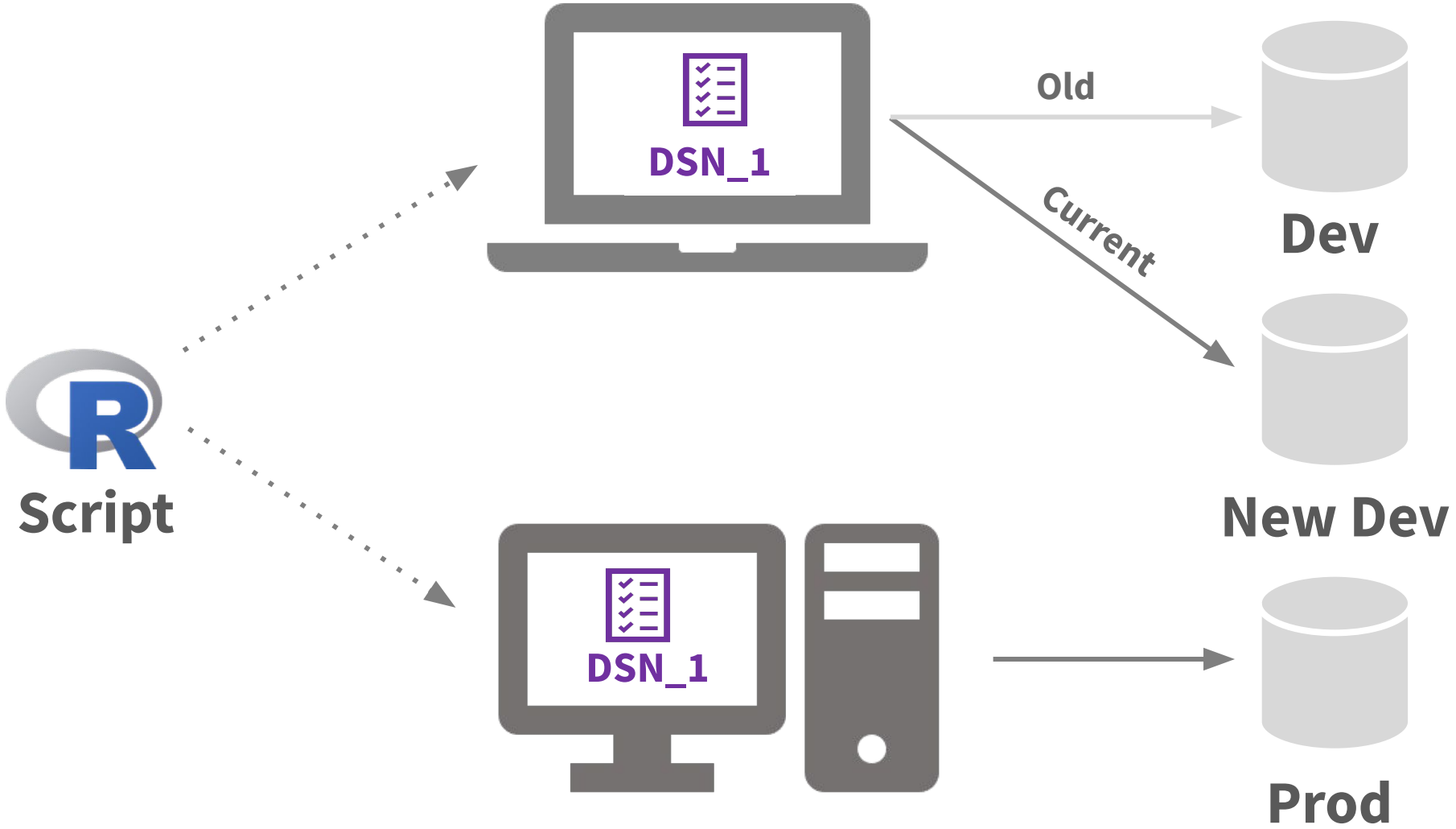


# The connections pane



# Why DSN?



# Alternatives for securing connections

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

# Exercise 3.1 - 3.4

# R packages

## General connections

- DBI
- odbc
- connections

## Specific Connections

- bigrquery
- RPostgres
- RSQLite
- RMariaDB
- sparklyr



github.com/r-dbi

# Unit 4
# Interacting with Databases


Photo by Nick Fewings on Unsplash

RStudio

# 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

# Exercise 4.1 - 4.4

# Unit 5

# Databases with dplyr

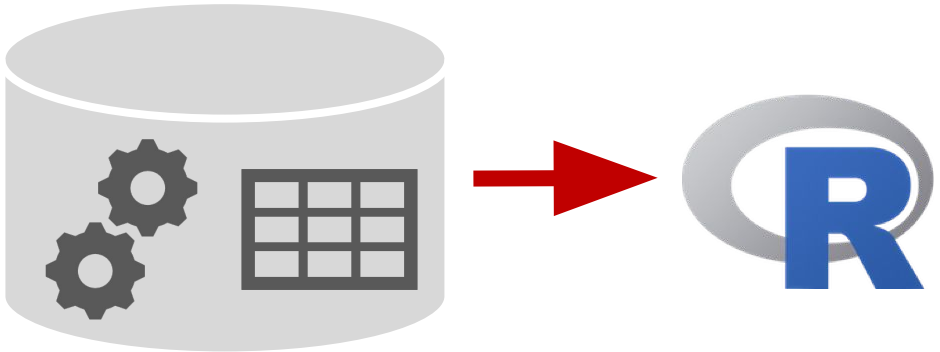
/dee-plier/


Photo by Arthur Lambillotte on Unsplash

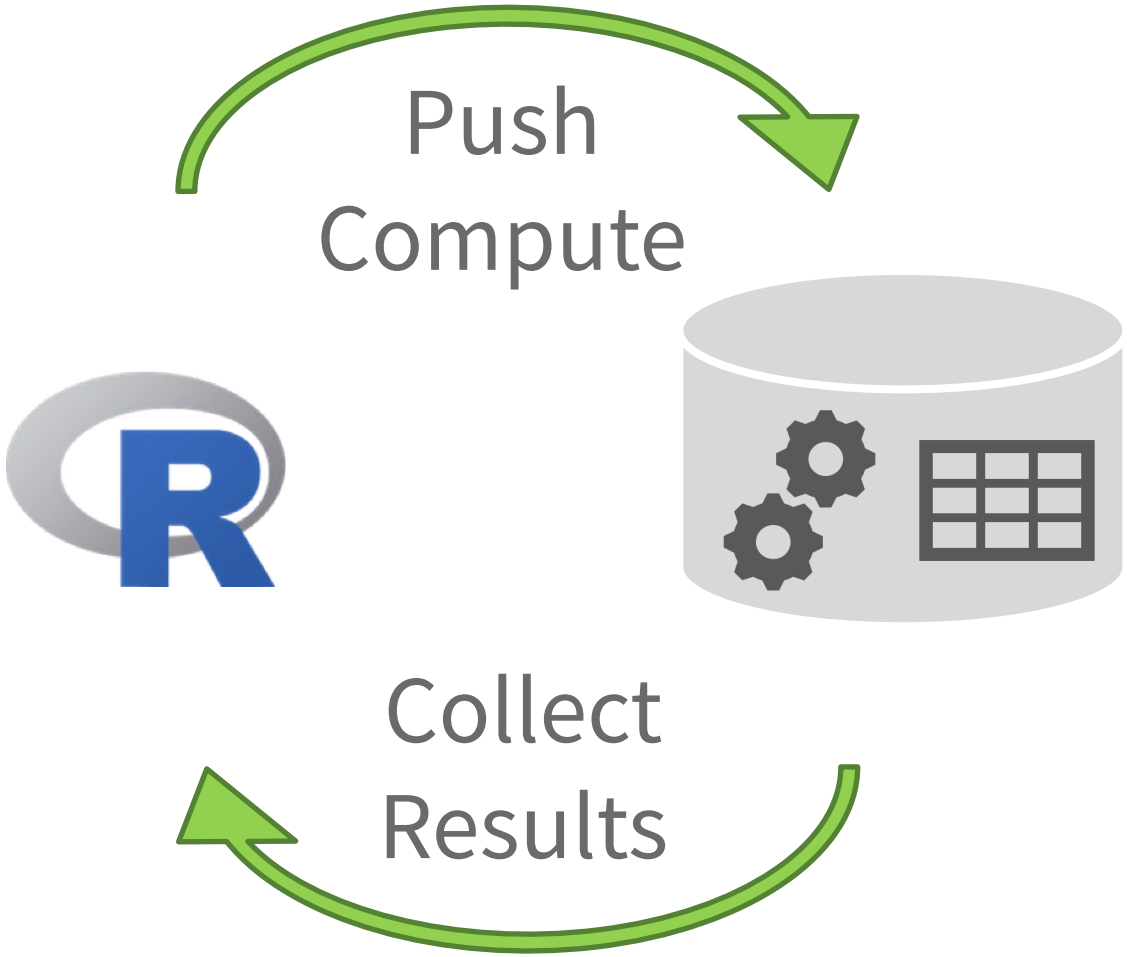
RStudio

# Wrangle inside the DB

Time Consuming



Extract Data



# Options to Push Compute

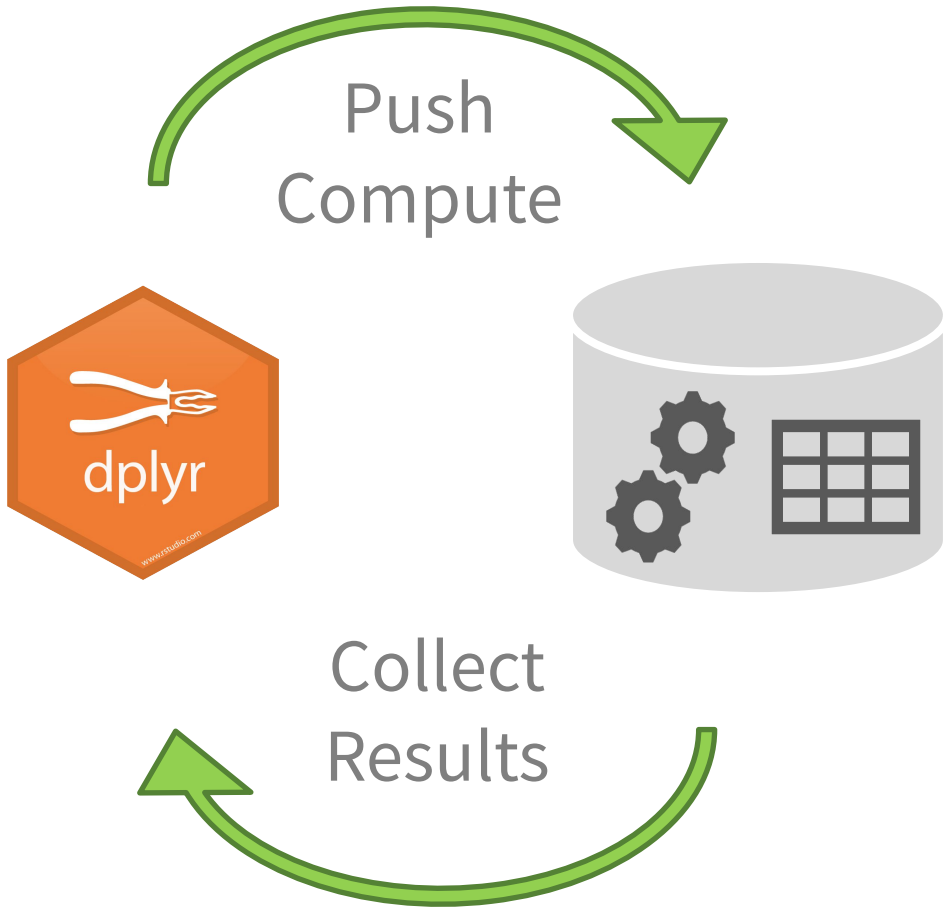
## Write SQL statements

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

## Use dplyr verbs

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

# Advantages




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

# Exercise 5.1 – 5.5

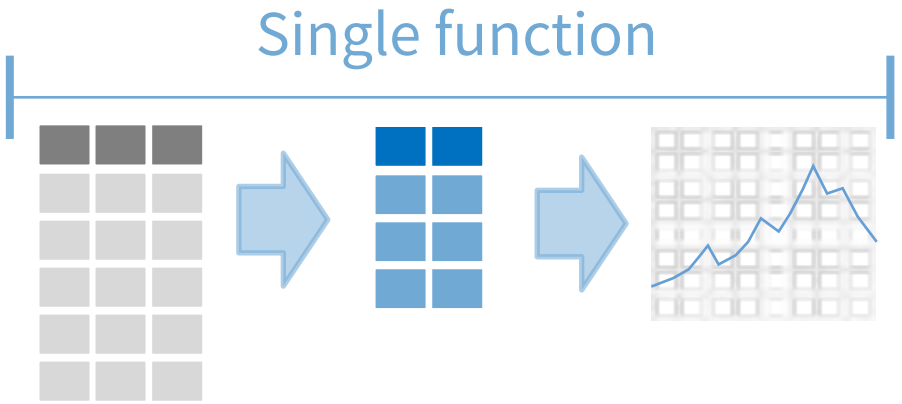
# Unit 6
# Visualizations


Photo by Luis Alfonso Orellana on Unsplash

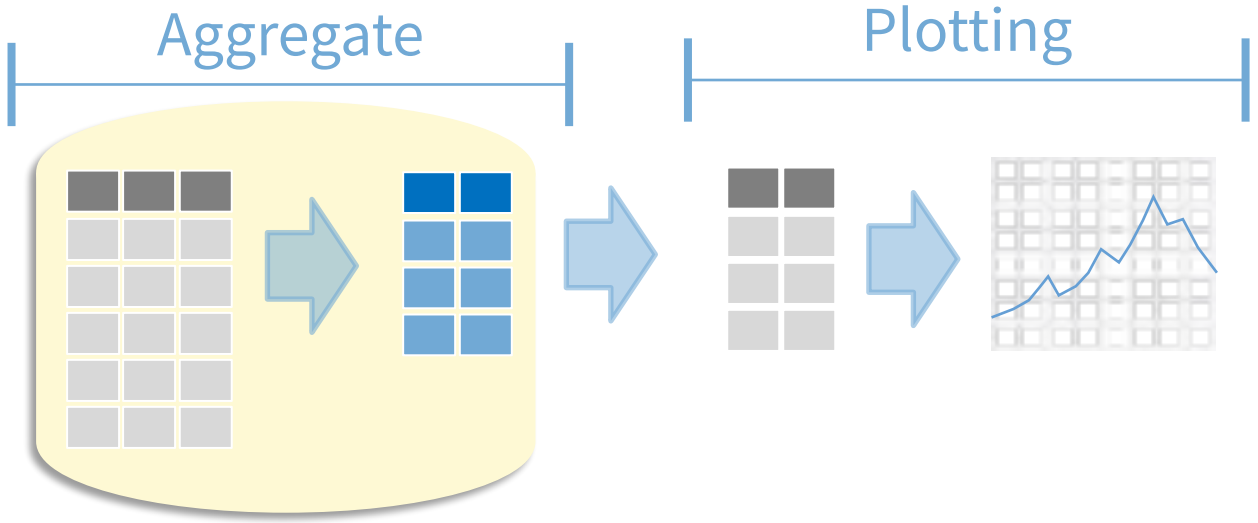
RStudio

# Visualizations

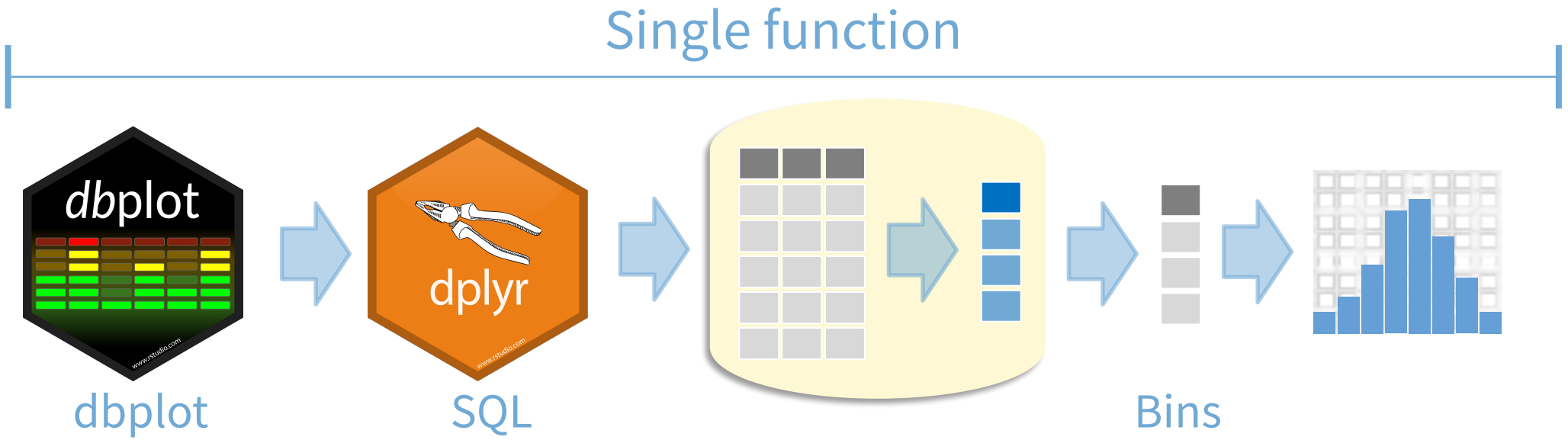
Local data




Remote data



# Complex plots



# Exercise 6.1 – 6.5

# Unit 7
# Modeling

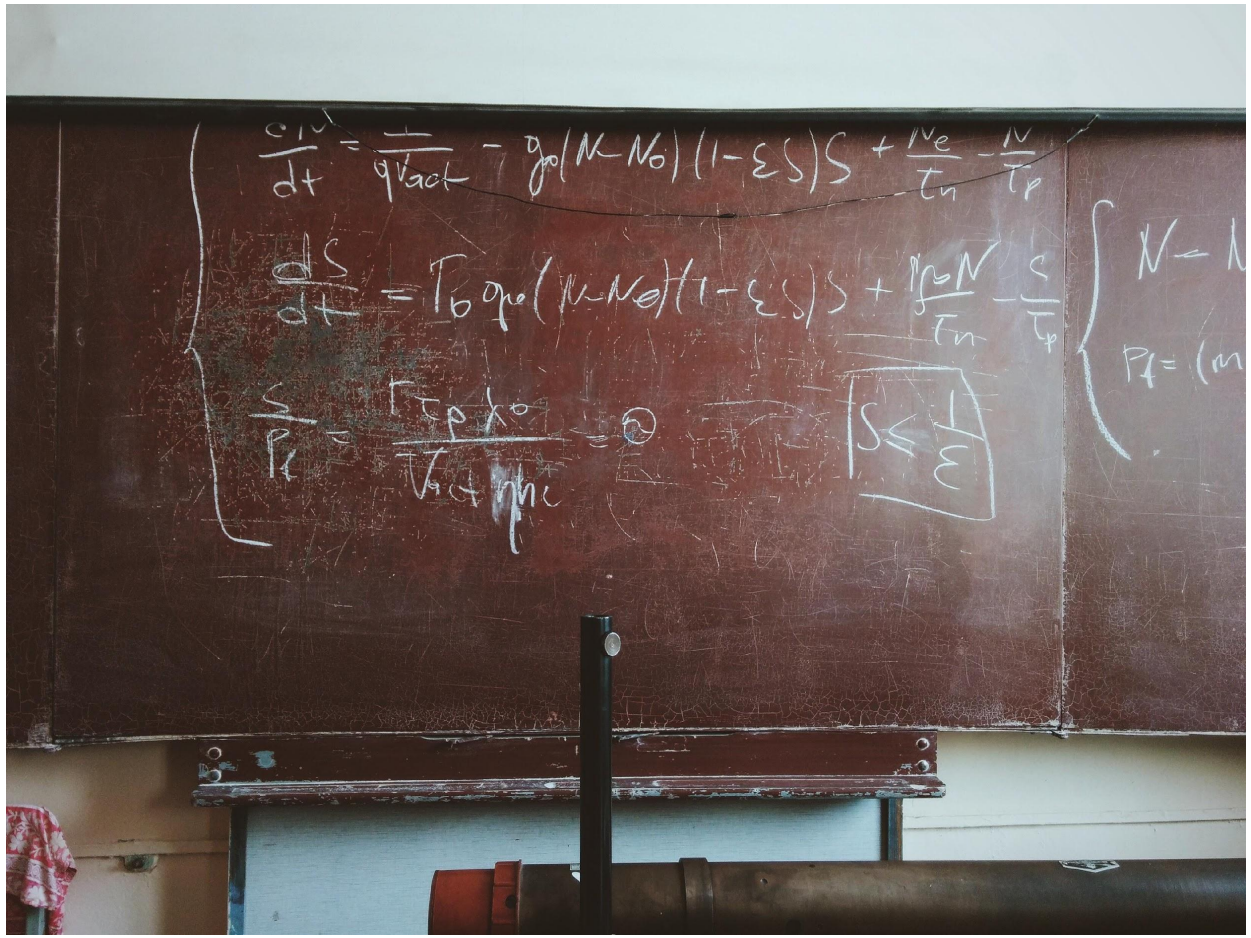
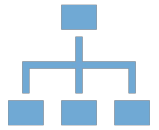
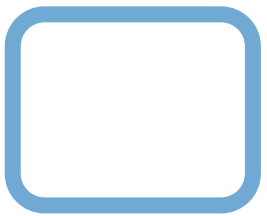


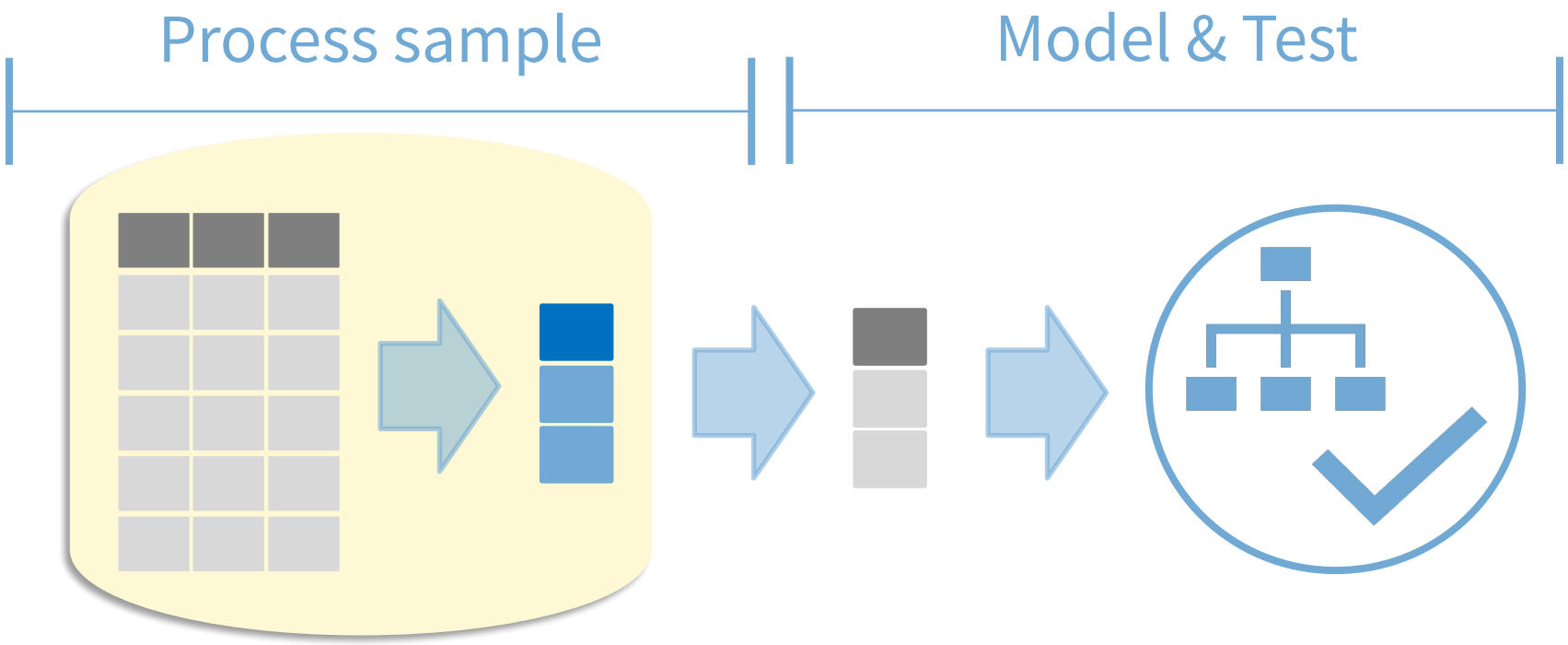
Photo by Roman Mager on Unsplash

RStudio

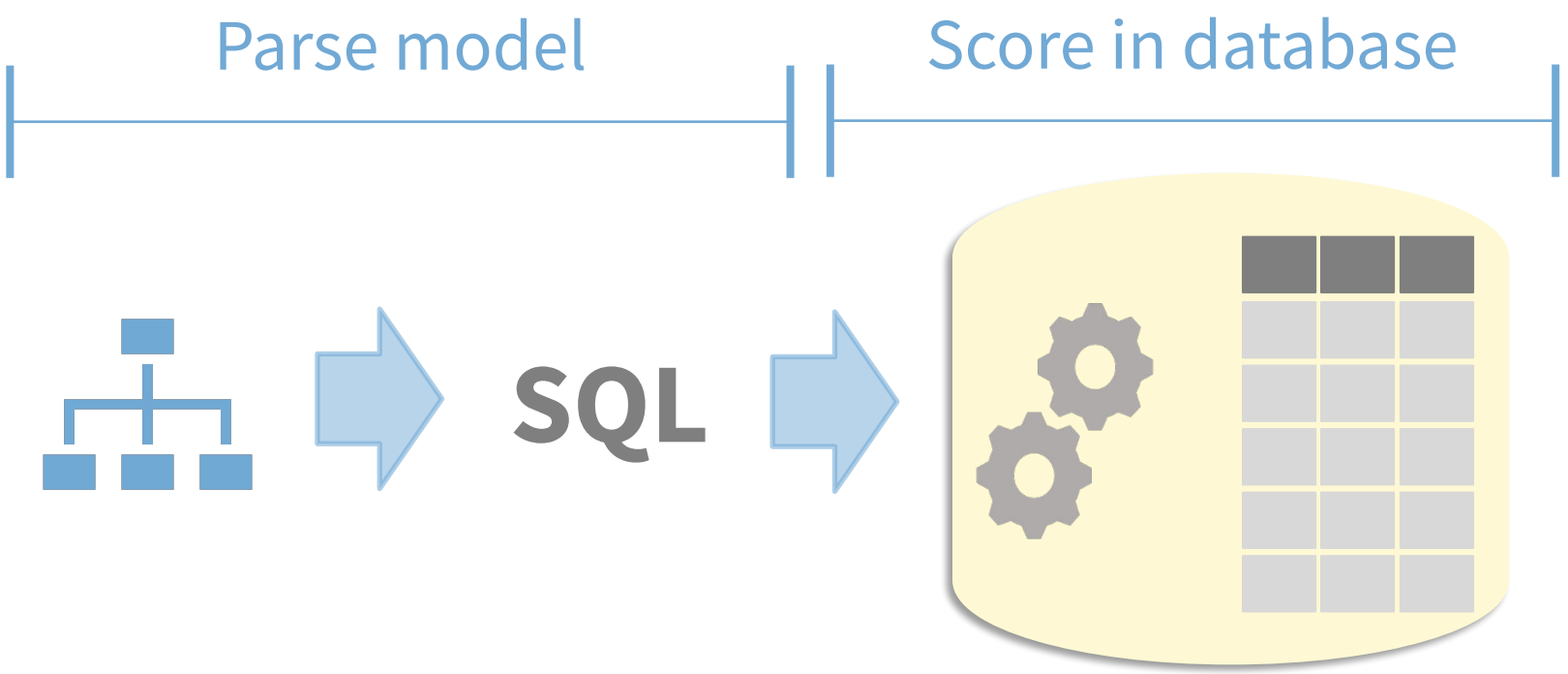
# Modeling scenario

1. Training sample 
2. Model on sample 
3. Testing sample 
4. Verify model 
5. Score data 

# Modeling with a Database

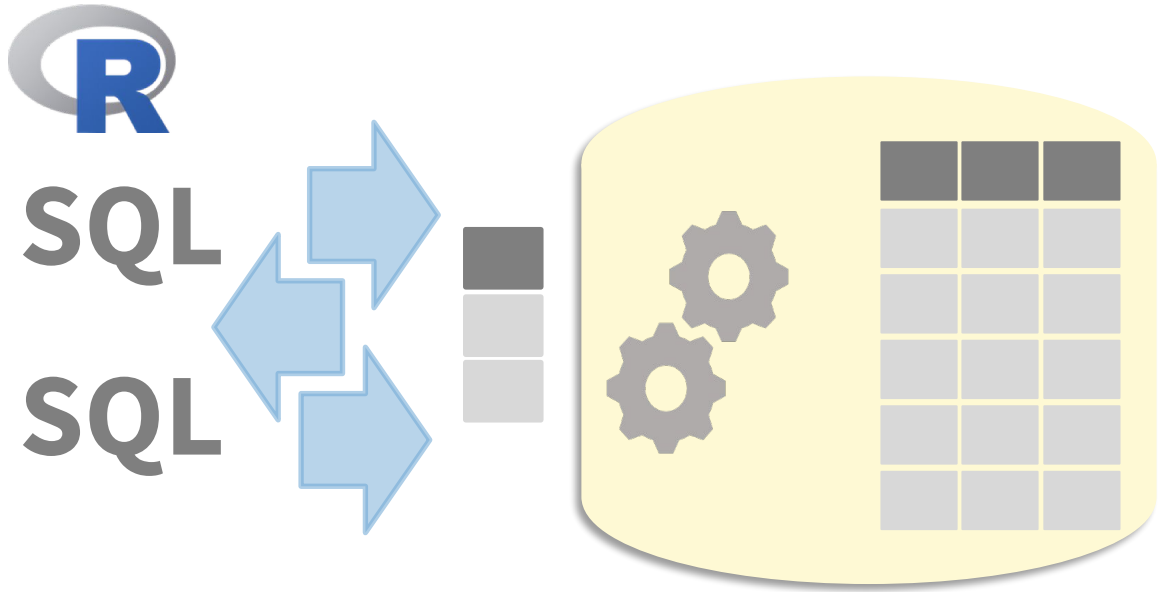


# Score inside the DB



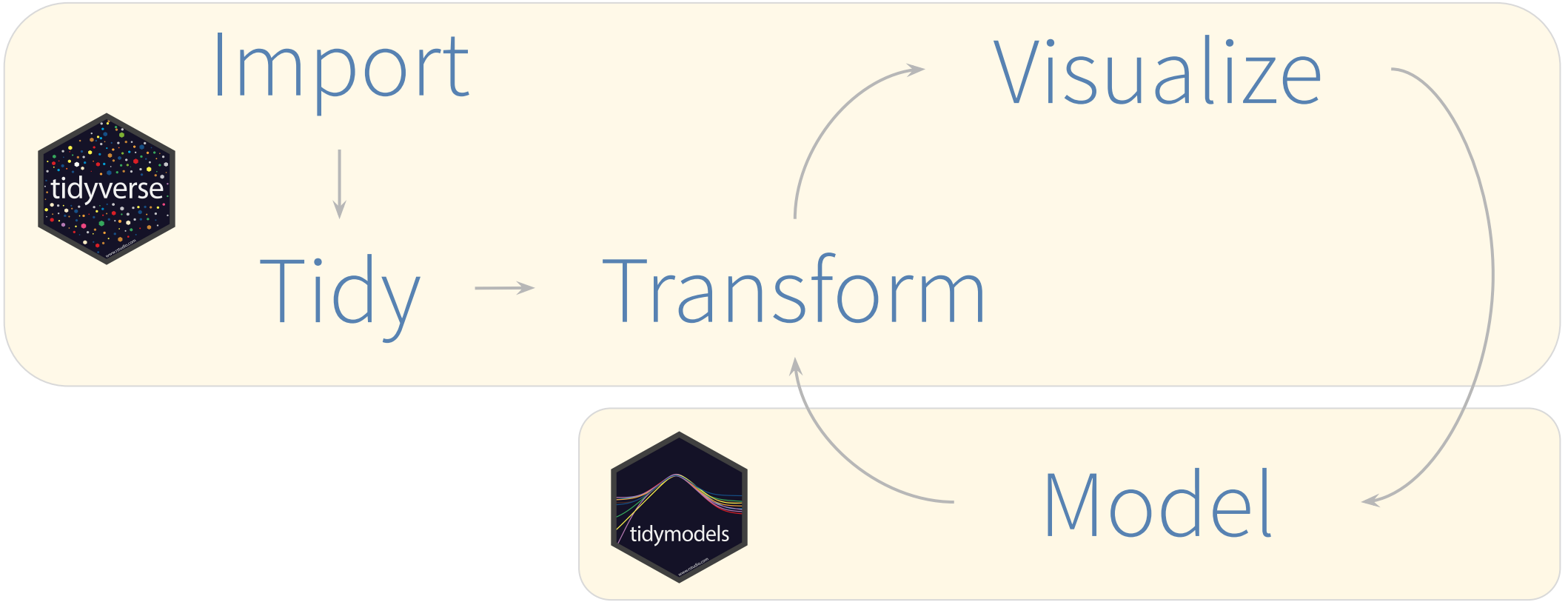
# Model inside the Database

Runs multiple queries to fit the model

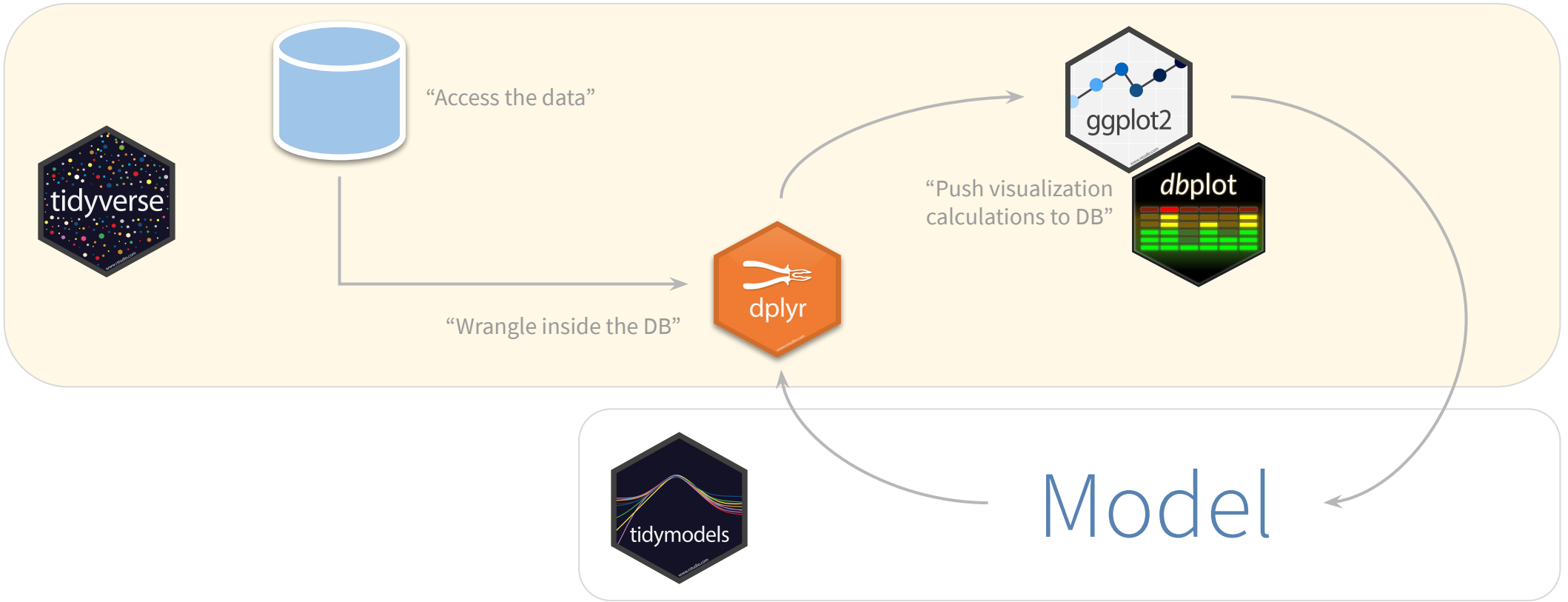


# Focus on three packages

# Where do they fit?



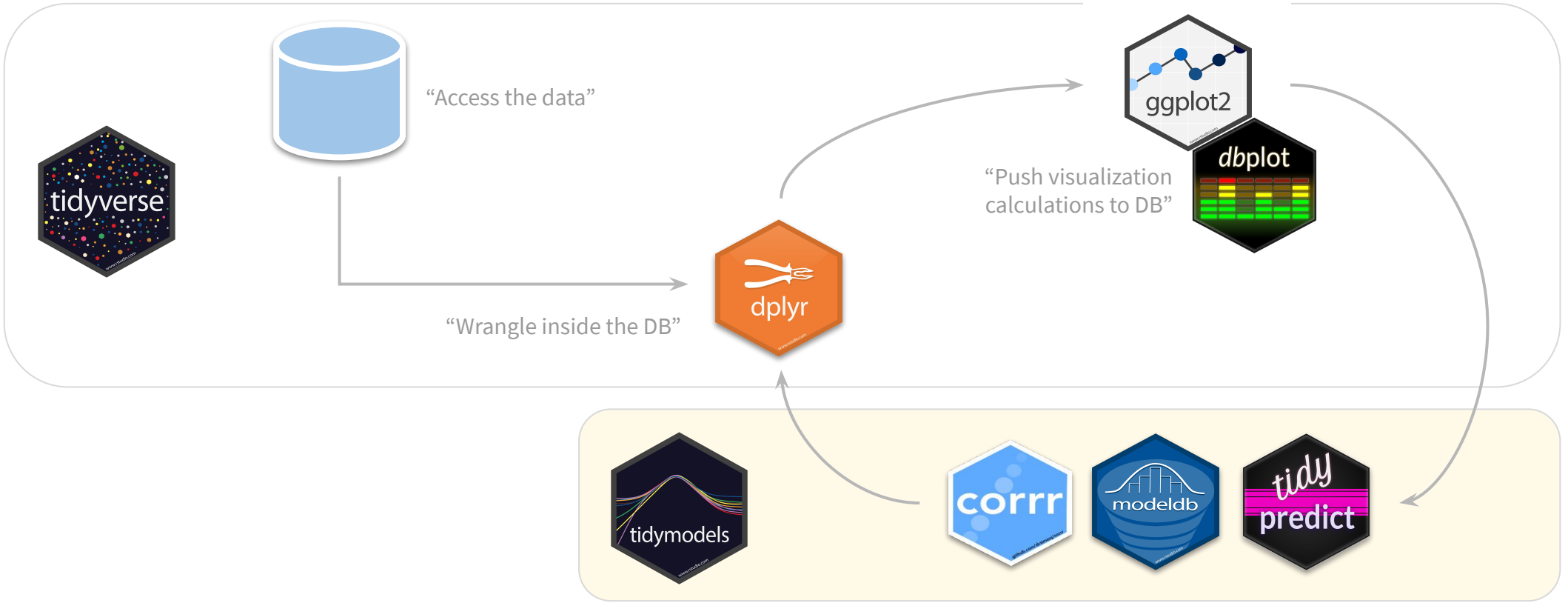
# Tidyverse + Databases




RStudio

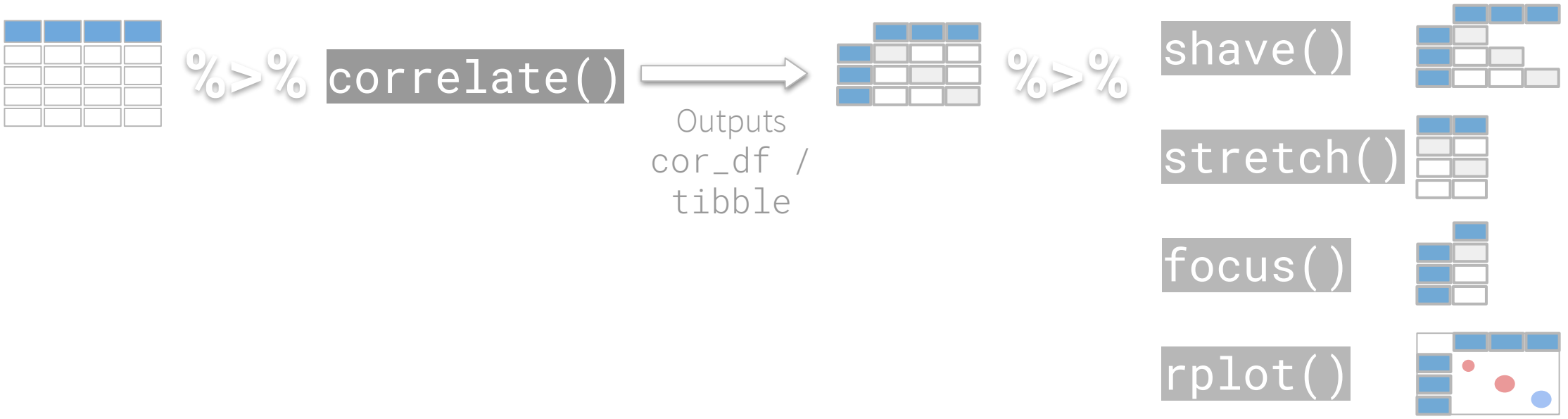
resources.rstudio.com/webinars/best-practices-for-working-with-databases-march-2018-edgar-ruiz

# Tidymodels + Databases

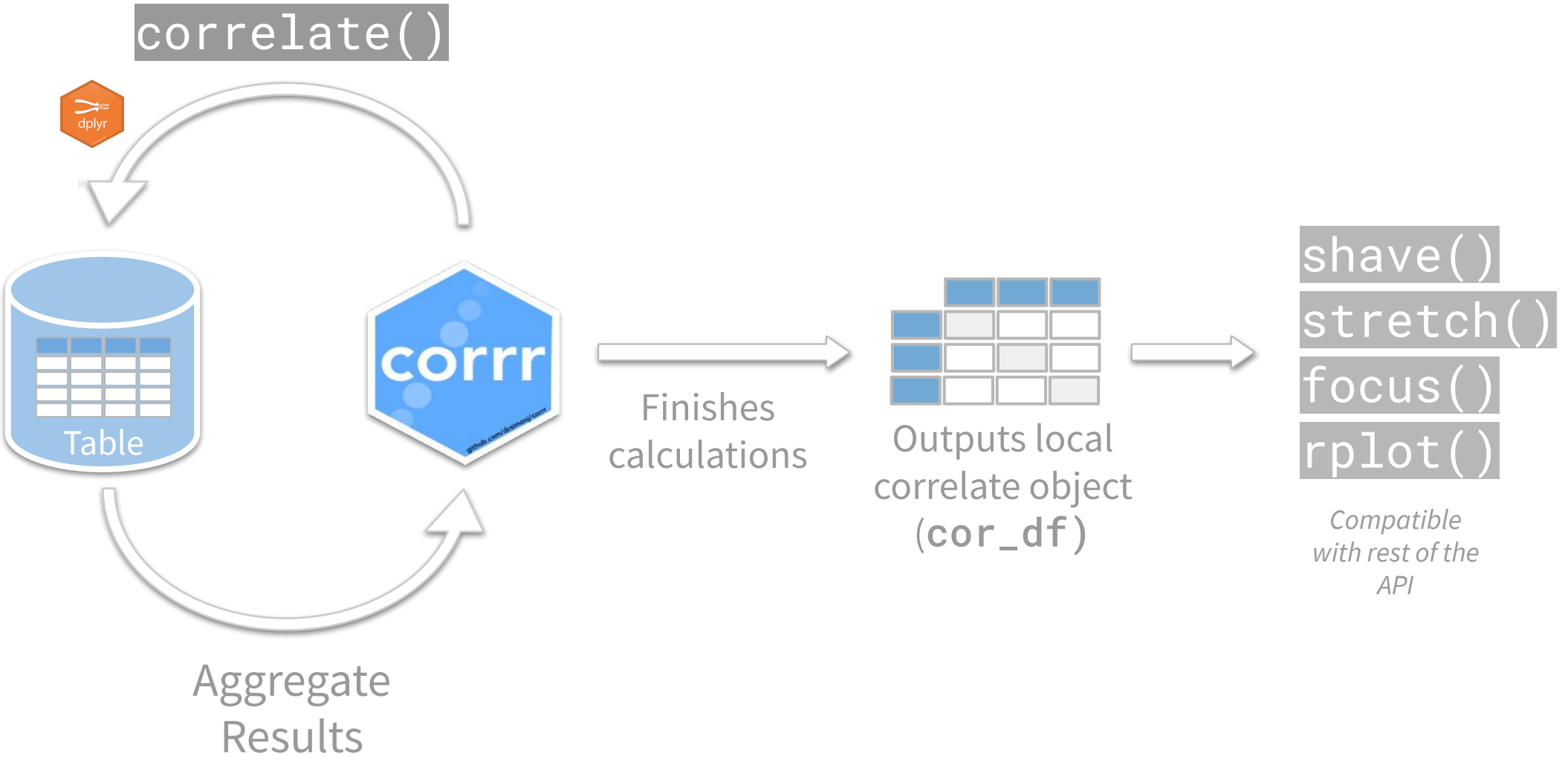


# `corrr` package

`corrr` integrates with the `tidyverse`. It comes with specialized functions that make exploring even easier. **Great for everyday analysis.**



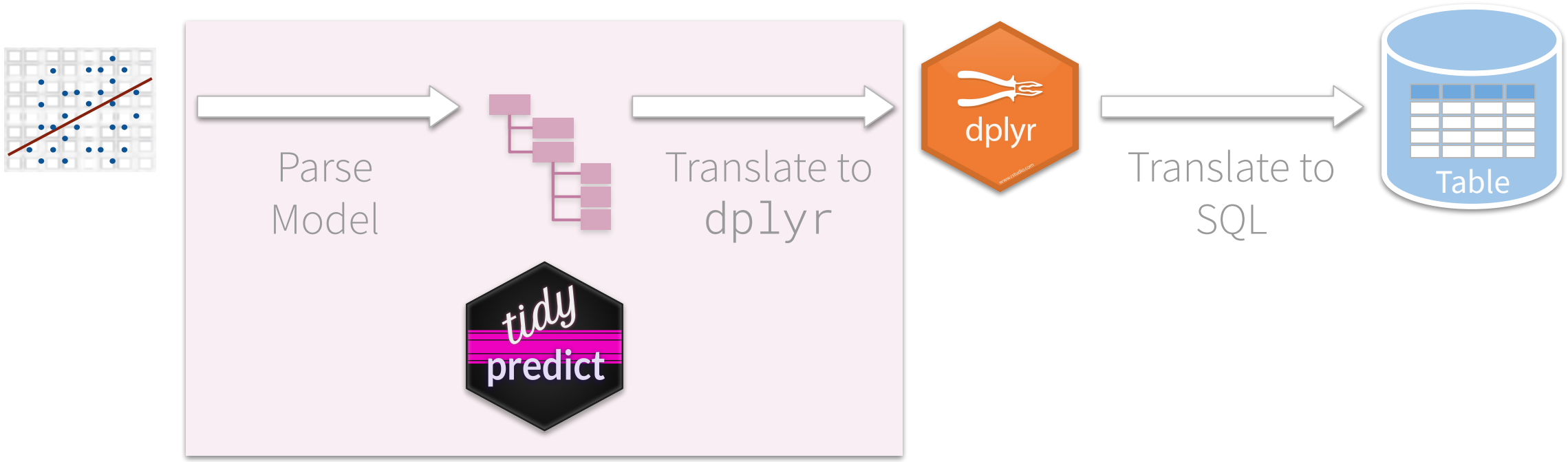
# `corrr` database integration






tidymodels.github.io/corrr/articles/databases.html

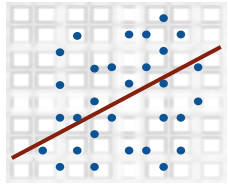
# `tidypredict` workflow






tidymodels.github.io/tidypredict/reference/

# tidypredict API



Parse Model

Translate to dplyr

tidy predict

dplyr

Translate to SQL

Table

tidypredict_to_column()

tidypredict_sql()

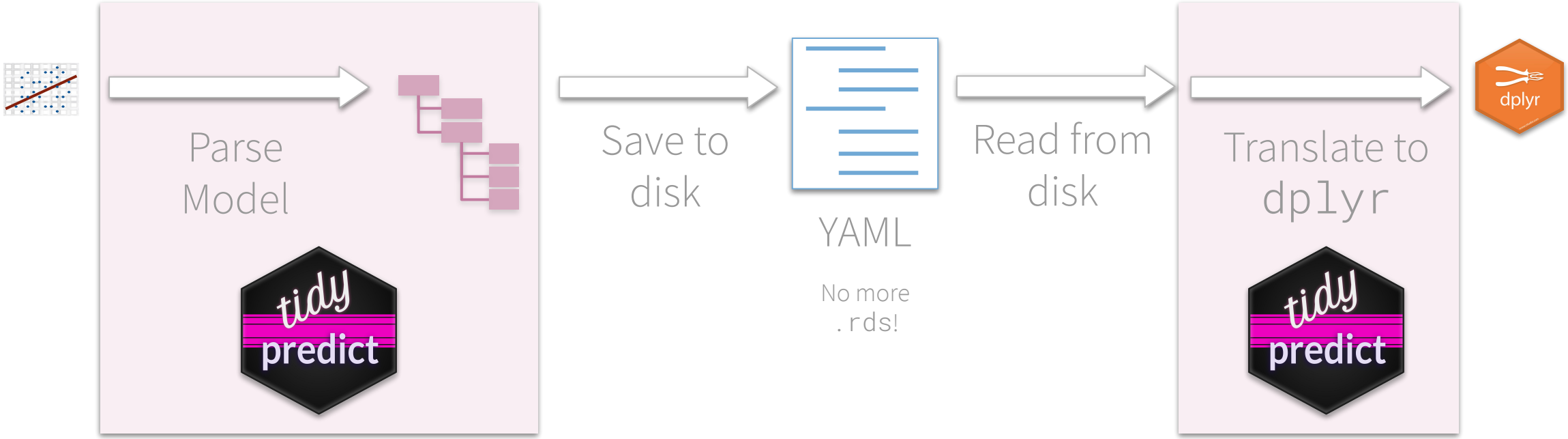
tidypredict_fit()

parse_model()



tidymodels.github.io/tidypredict/reference/

# Save the parsed model

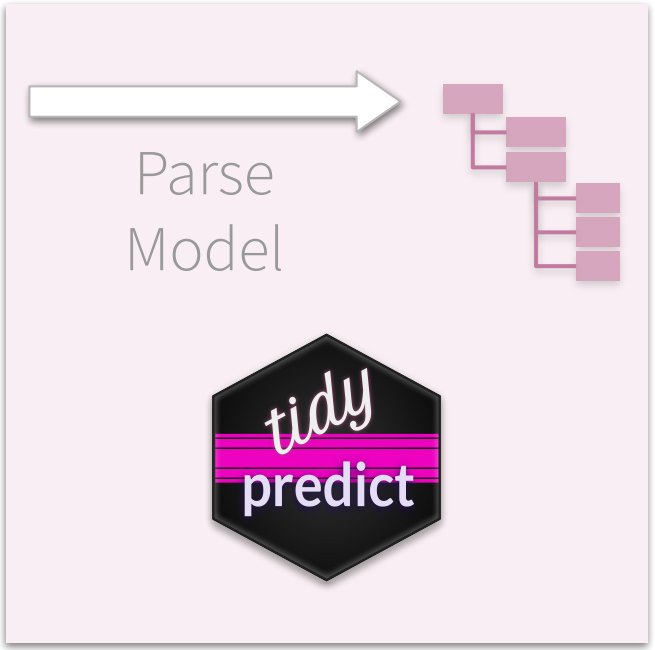





tidymodels.github.io/tidypredict/articles/save.html

# Today, we have parsers for 8 R models

1. Linear Regression - `lm()`
2. Generalized Linear - glm()
3. Random forest - `randomForest::randomForest()`
4. Ranger - `ranger::ranger()`
5. MARS - `earth::earth()`
6. Cubist - `Cubist::cubist()`
7. partykit tree - `partykit::ctree()`
8. XGBoost - `xgboost::xgb.Booster.complete()`

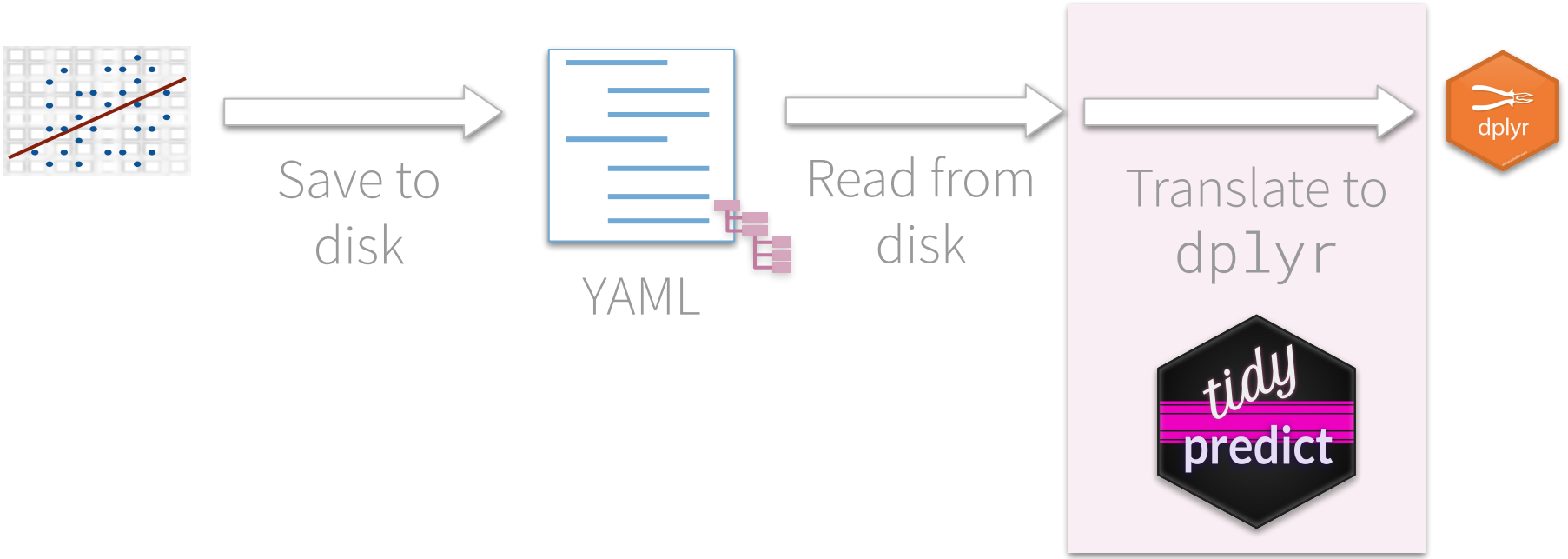





tidymodels.github.io/tidypredict/

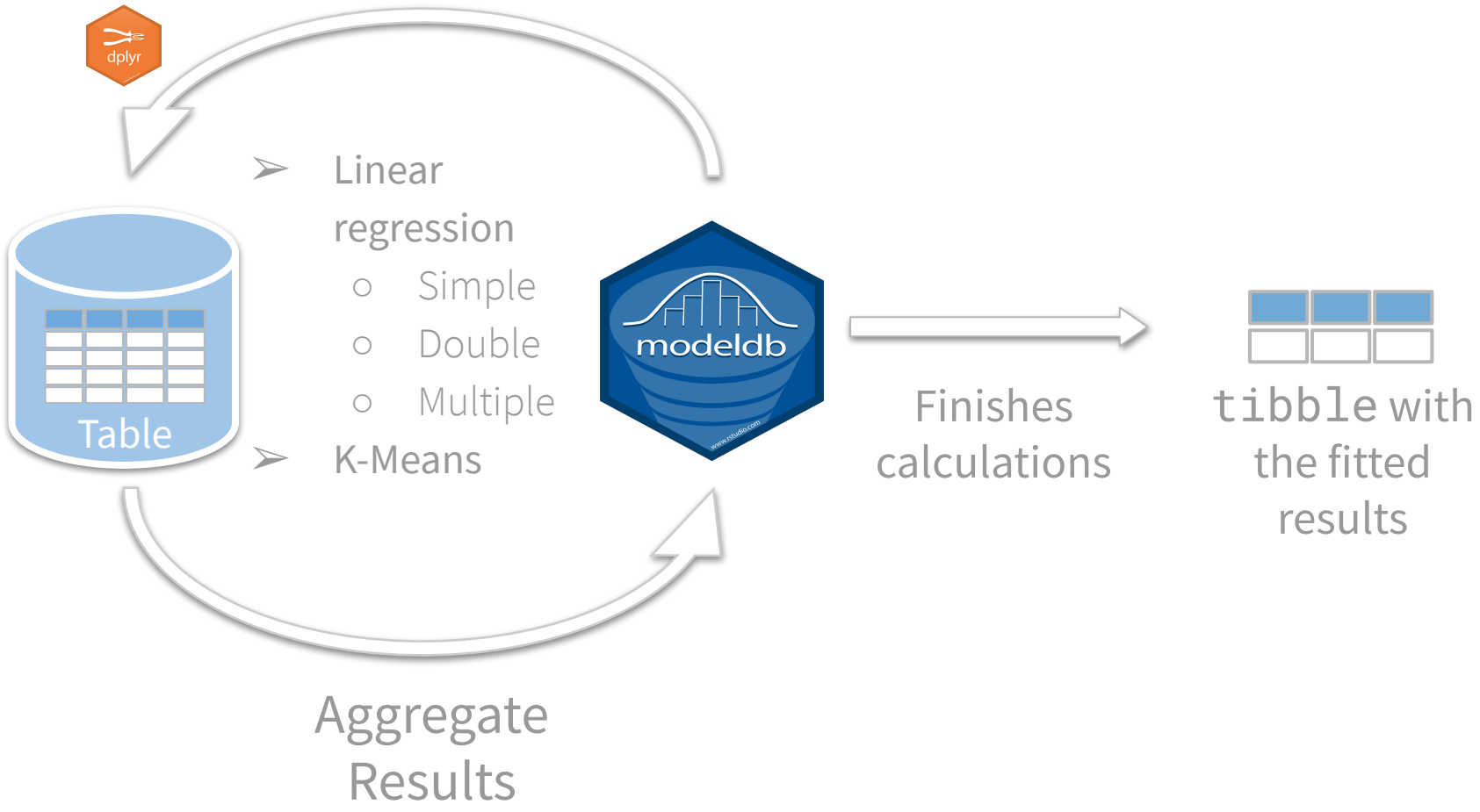
# Integrate models from other languages

`tidypredict` is able to use exported files from *non-R scripts* as long as they follow one of three parse model specs (regression, tree, xgb)




tidymodels.github.io/tidypredict/articles/non-r.html

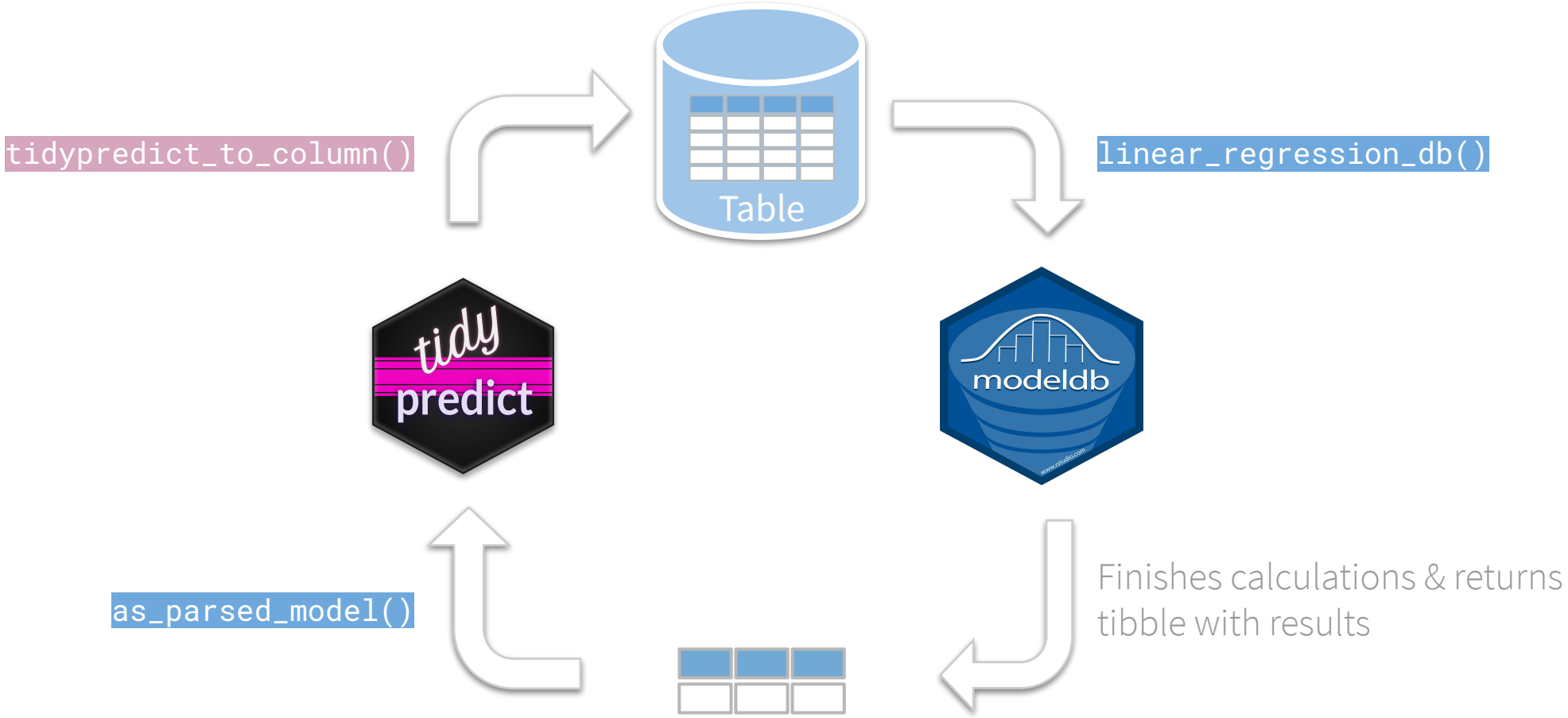
# `modeldb` runs models in database






tidymodels.github.io/modeldb/

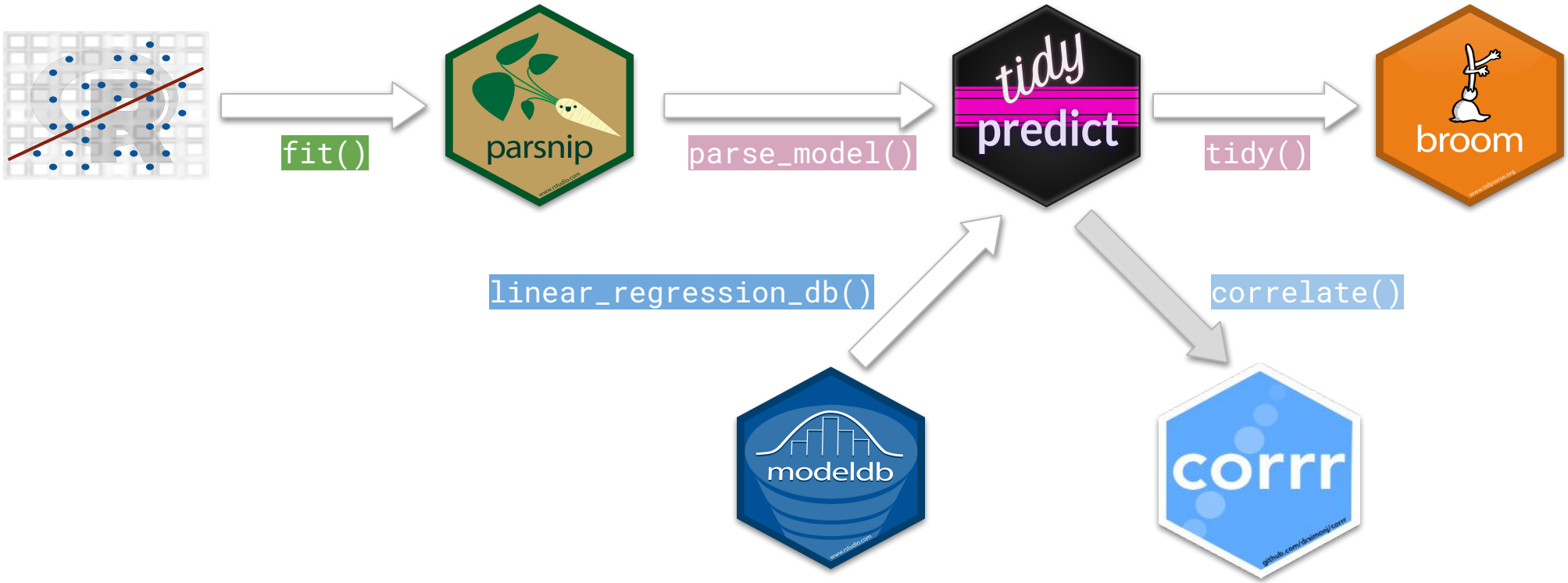
# modeldb + tidypredict



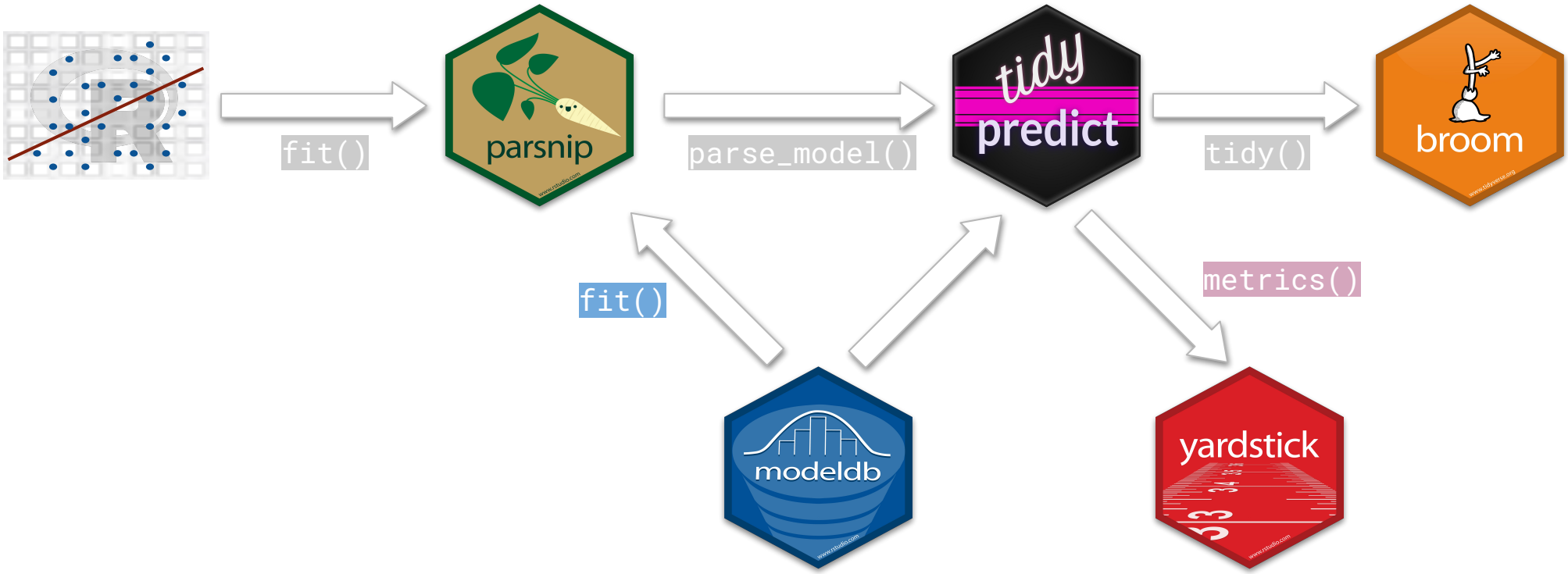



tidymodels.github.io/modeldb/

# Integration today



# Integration in the future



# Exercise 7.1 - 7.4

# Unit 8
# 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)
```

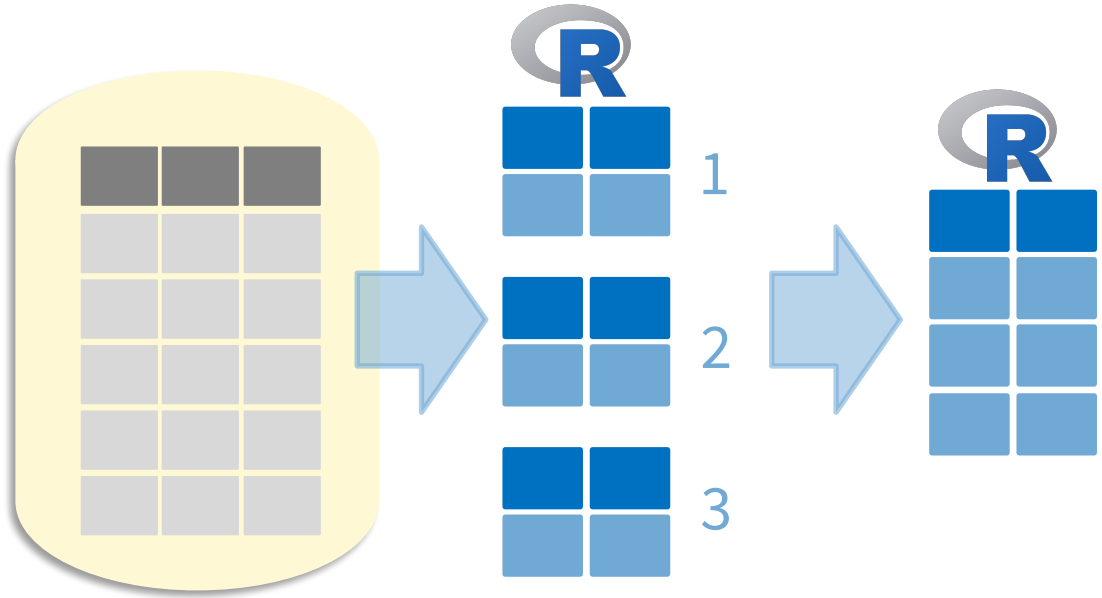
```
flights %>%
  summarise(
    m = mean(arrtime)
  )
```

# Tidy eval functions to remember

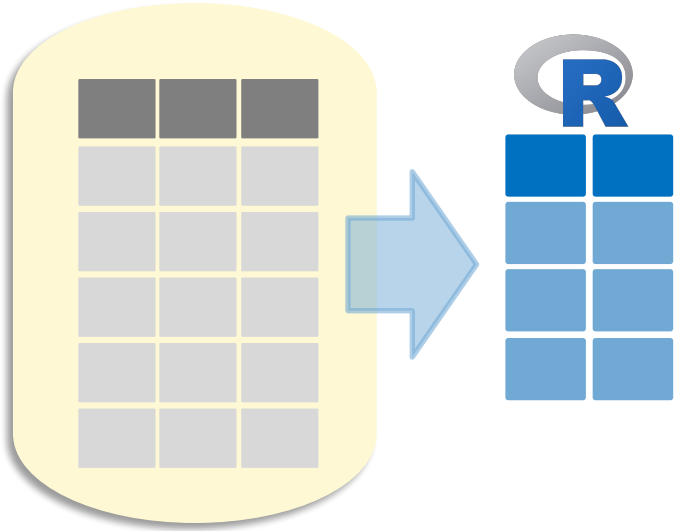
| Pauses evaluation | Pauses evaluation of arguments | Resumes evaluation | Coerces to a *field* name |
| --- | --- | --- | --- |
| `expr()` | `enquo()` | `!!` | `sym()` |
| | `enquos()` | `!!!` | `syms()` |

# Multiple queries

## Many trips to the database




## One trip to the database



# Map/Reduce ~~data~~ code

Many trips to the database

```
map
(
```




```
)
```

One trip to the database

```
map
(
  expr
```


```
  )
  (
  expr
```


```
  )
  (
  expr
```


```
  )
  (
    ) %>%

  reduce()
```

# Exercise 8.1 - 8.5

# Unit 9

# sparklyr

/s-par-klee-r/

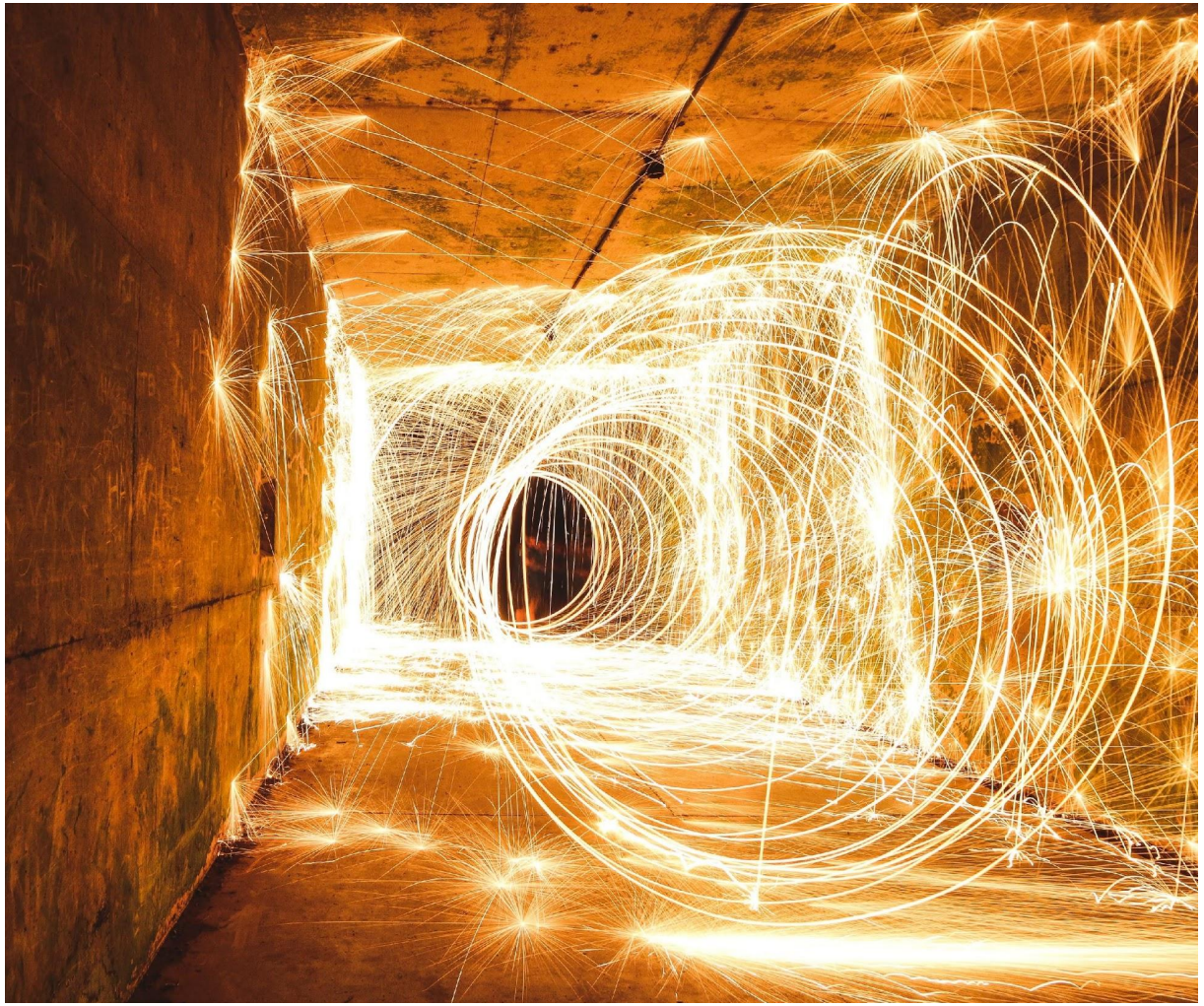


Photo by Matthew Ronder-Seid on Unsplash

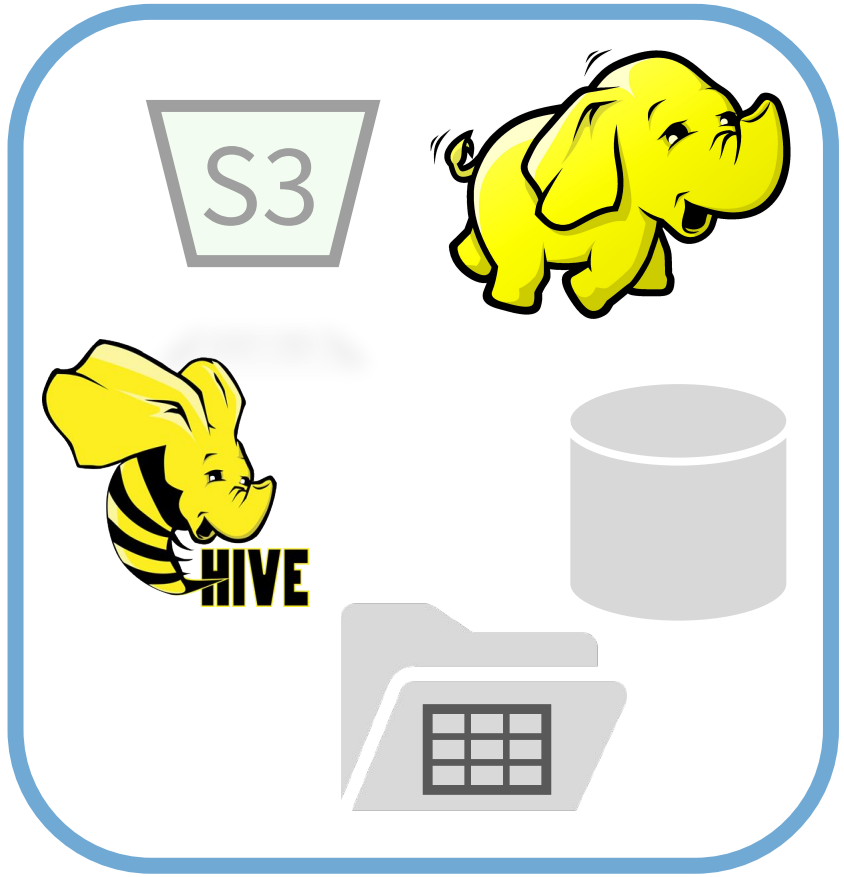
RStudio

# What is Spark?

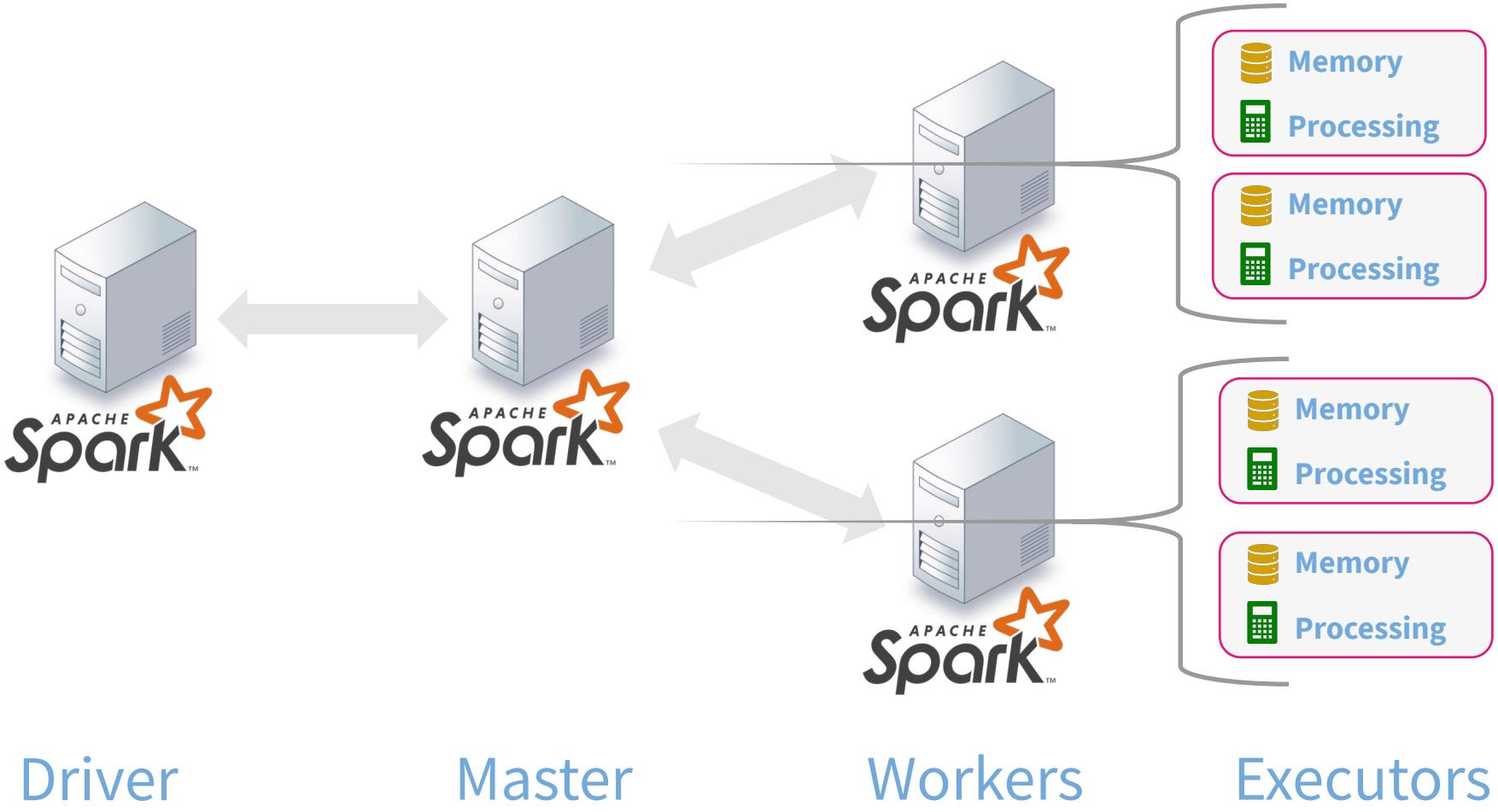
## Processing

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

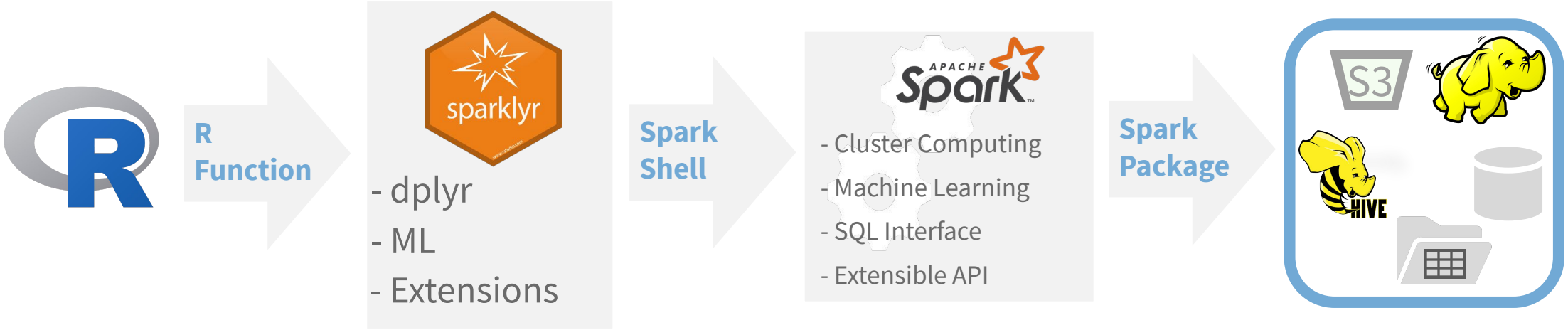
## Storage



# Typical architecture

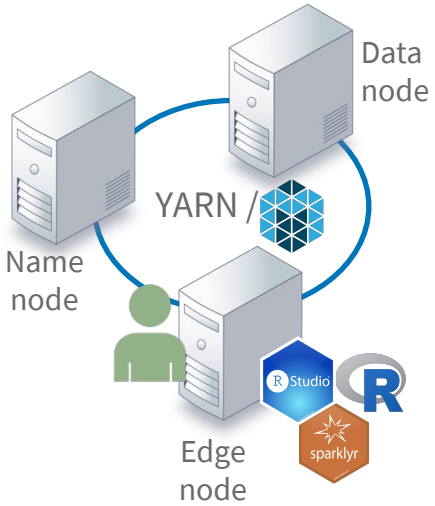


# sparklyr – An R interface for Spark



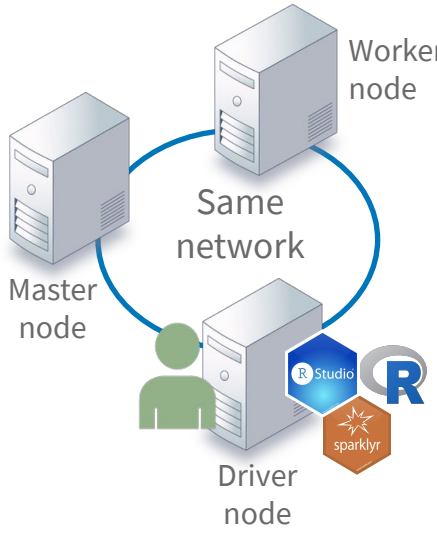
# Deployment options

## Managed Cluster



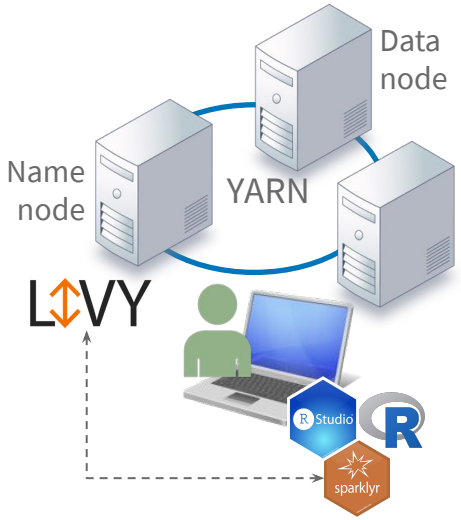

- Deployment seen at most business
- Spark version(s) available are limited to what's on the cluster

## Stand Alone




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

## Livy

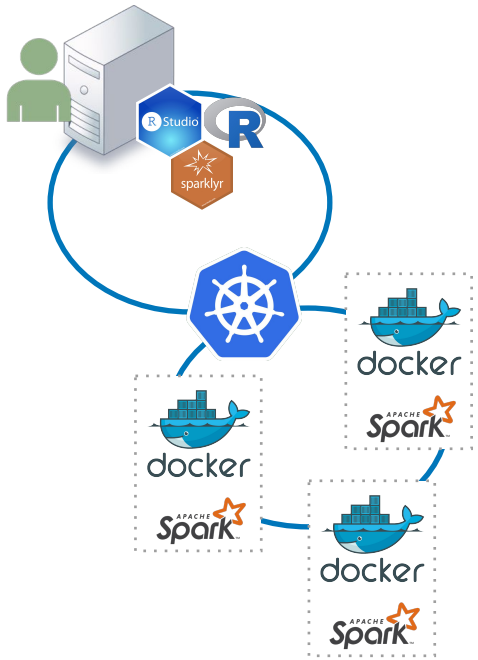



- Great for accessing a remote cluster
- Not recommended for Production deployments

## Local


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

## Kubernetes

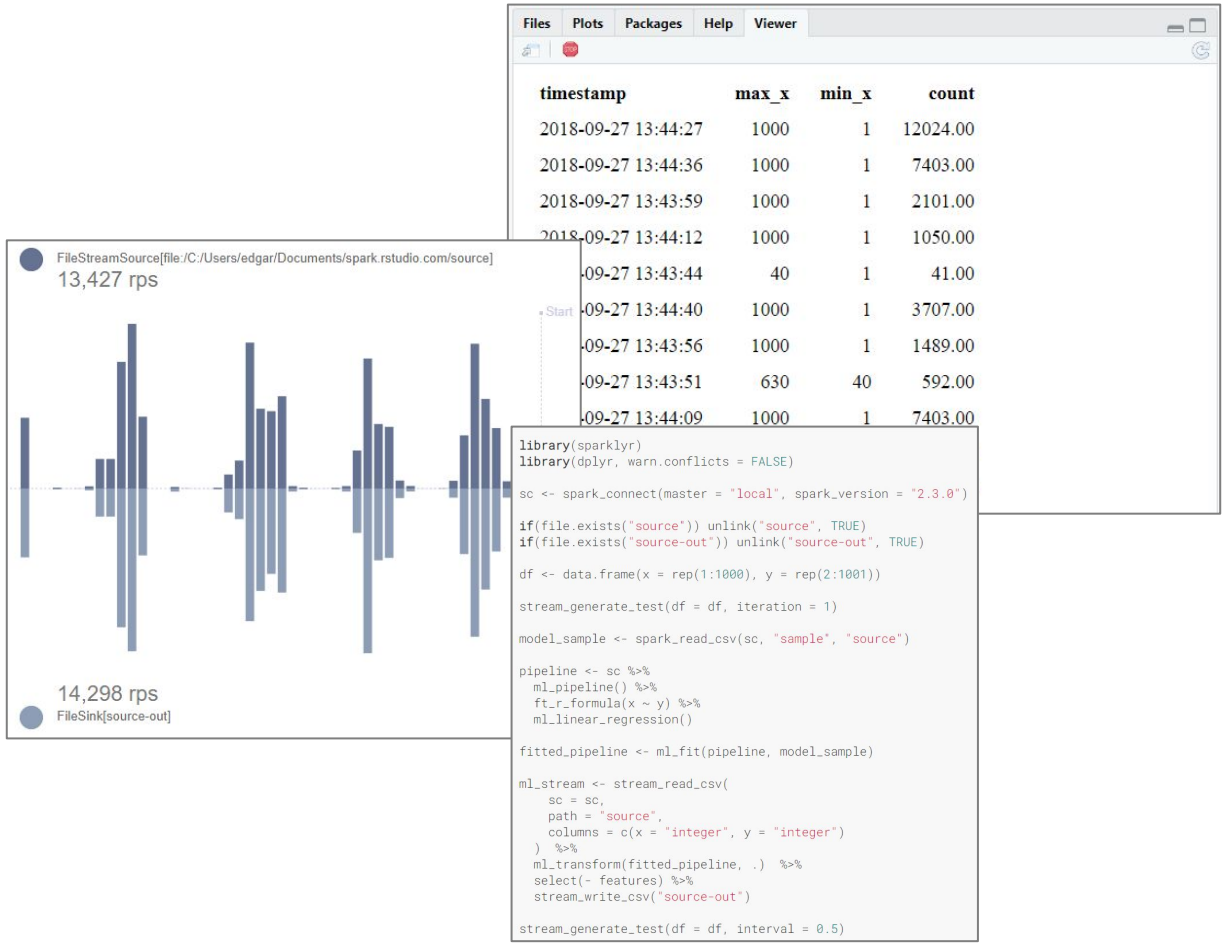



- New – It allows to connect to a Spark cluster inside a Kubernetes cluster

# What does it offer?

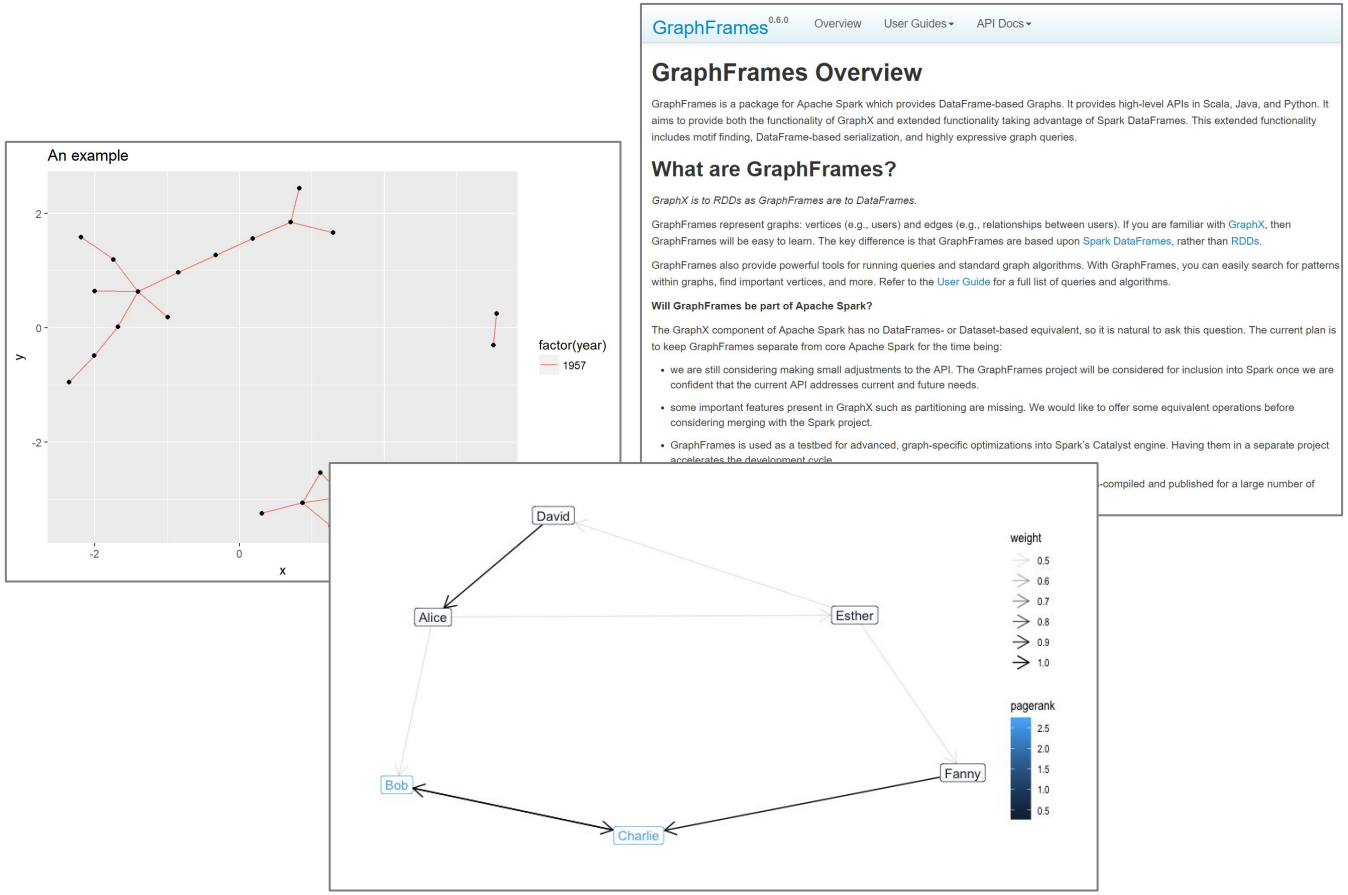
# Streaming

- 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



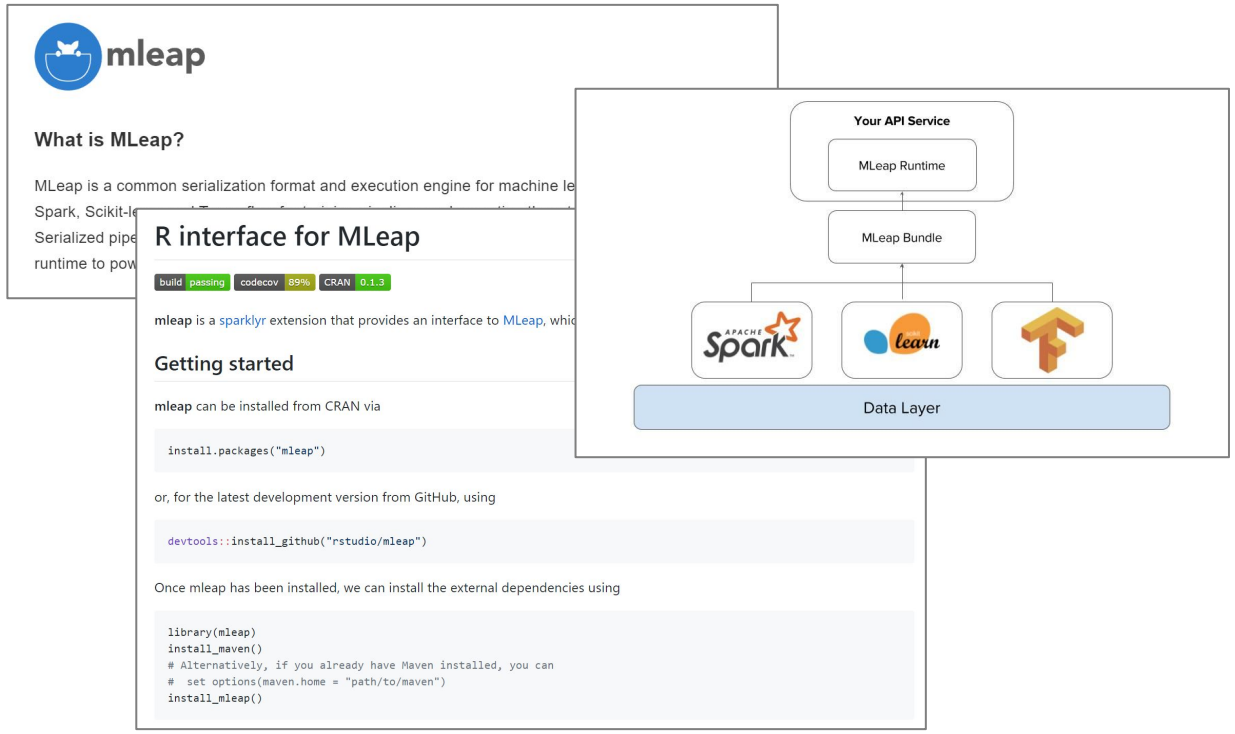
# Graph analysis

- 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



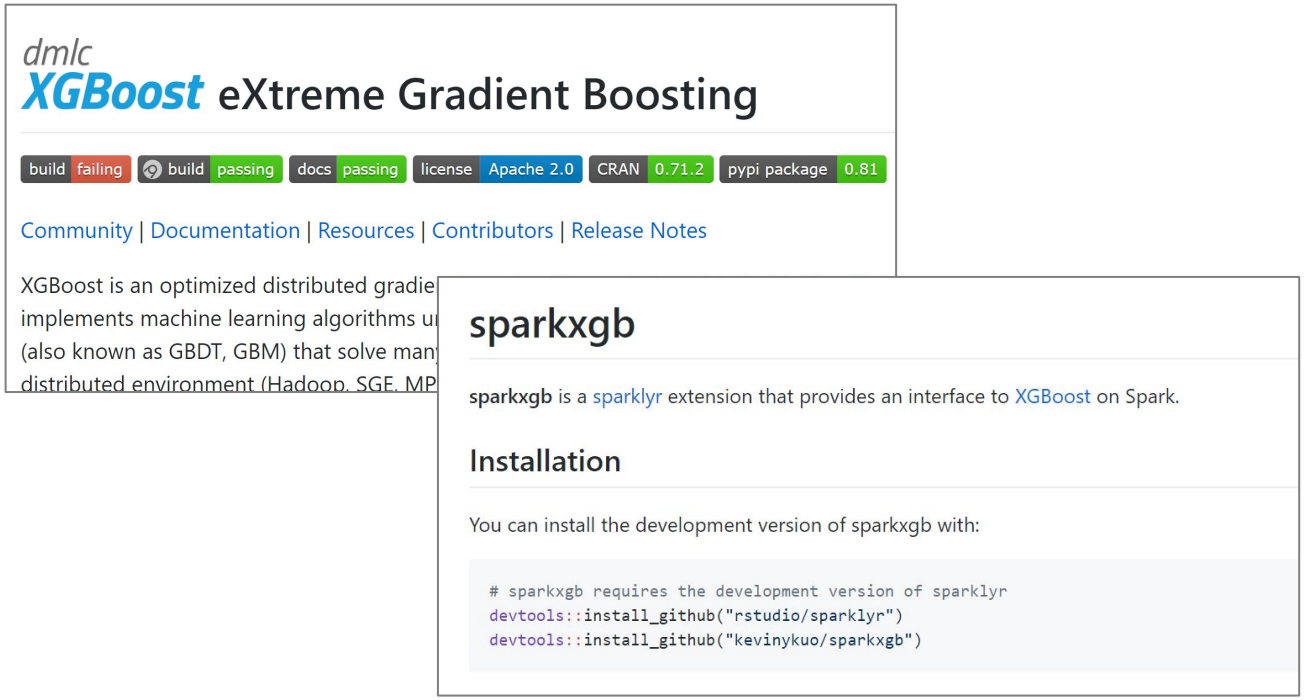
# Productions pipelines with Mleap

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



# XGBoost/Spark integration with sparkxgb

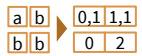
**sparkxgb** is a sparklyr extension that provides an interface to XGBoost on Spark.



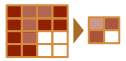
# Feature transformers


**ft_binarizer()** - Assigned values based on threshold

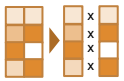

**ft_bucketizer()** - Numeric column to discretized column



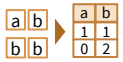

**ft_count_vectorizer()** - Extracts a vocabulary from document



**ft_discrete_cosine_transform()** - 1D discrete cosine transform of a real vector




**ft_elementwise_product()** - Element-wise product between 2 cols




**ft_hashing_tf()** - Maps a sequence of terms to their term frequencies using the hashing trick.

**ft_idf()** - Compute the Inverse Document Frequency (IDF) given a collection of documents

**ft_imputer()** - Imputation estimator for completing missing values, uses the mean or the median of the columns


**ft_index_to_string()** - Index labels back to label as strings


**ft_interaction()** - Takes in Double and Vector type columns and outputs a flattened vector of their feature interactions


**ft_max_abs_scaler()** - Rescale each feature individually to range [-1, 1]


**ft_min_max_scaler()** - Rescale each feature individually to a common range [min, max] linearly


**ft_ngram()** - Converts the input array of strings into an array of n-grams

**ft_bucketed_random_projection_lsh()**
**ft_minhash_lsh()** - Locality Sensitive Hashing functions for Euclidean distance and Jaccard distance (MinHash)


**ft_normalizer()** - Normalize a vector to have unit norm using the given p-norm


**ft_one_hot_encoder()**- Continuous to binary vectors

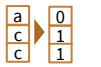
**ft_pca()** - Project vectors to a lower dimensional space of top k principal components


**ft_quantile_discretizer()** - Continuous to binned categorical values


**ft_regex_tokenizer()** - Extracts tokens either by using the provided regex pattern to split the text

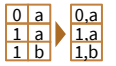

**ft_standard_scaler()** - Removes the mean and scaling to unit variance using column summary statistics


**ft_stop_words_remover()** - Filters out stop words from input

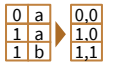



**ft_string_indexer()** - Column of labels into a column of label indices.

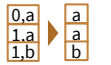

**ft_tokenizer()** - Converts to lowercase and then splits it by white spaces



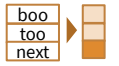

**ft_vector_assembler()** - Combine vectors into single row-vector




**ft_vector_indexer()** - Indexing categorical feature columns in a dataset of Vector




**ft_vector_slicer()** - Takes a feature vector and outputs a new feature vector with a subarray of the original features




**ft_word2vec()** - Word2Vec transforms a word into a code

RStudio

# Modeling with Spark

## REGRESSION

**ml_linear_regression()** - Regression using linear regression.

**ml_aft_survival_regression()** - Parametric survival regression model named accelerated failure time (AFT) model

**ml_generalized_linear_regression()** - Generalized linear regression model

**ml_isotonic_regression()** - Currently implemented using parallelized pool adjacent violators algorithm. Only univariate (single feature) algorithm supported

**ml_random_forest_regressor()** - Regression using random forests.

## CLASSIFICATION

**ml_linear_svc()** - Classification using linear support vector machines

**ml_logistic_regression()** - Logistic regression

**ml_multilayer_perceptron_classifier()** - Classification model based on the Multilayer Perceptron.

**ml_naive_bayes()** - Naive Bayes Classifiers. It supports Multinomial NB which can handle finitely supported discrete data

**ml_one_vs_rest()** - Reduction of Multiclass Classification to Binary Classification. Performs reduction using one against all strategy.

## FEATURE

**ml_chisquare_test(x,features,label)** - Pearson's independence test for every feature against the label

**ml_default_stop_words()** - Loads the default stop words for the given language

## CLUSTERING

**ml_bisecting_kmeans()** - A bisecting k-means algorithm based on the paper

**ml_lda() | ml_describe_topics() | ml_log_likelihood() | ml_log_perplexity() | ml_topics_matrix()** - LDA topic model designed for text documents.

**ml_gaussian_mixture()** - Expectation maximization for multivariate Gaussian Mixture Models (GMMs)

**ml_kmeans() | ml_compute_cost()** - K-means clustering with support for k-means

## FP GROWTH

**ml_fpgrowth() | ml_association_rules() | ml_freq_itemsets()** - A parallel FP-growth algorithm to mine frequent itemsets.

## RECOMMENDATION

**ml_als() | ml_recommend()** - Recommendation using Alternating Least Squares matrix factorization

## TREE

**ml_decision_tree_classifier() | ml_decision_tree() | ml_decision_tree_regressor()** - Classification and regression using decision trees

**ml_gbt_classifier() | ml_gradient_boosted_trees() | ml_gbt_regressor()** - Binary classification and regression using gradient boosted trees

**ml_random_forest_classifier()** - Classification and regression using random forests.

**ml_feature_importances(model,...)ml_tree_feature_importance(model)** - Feature Importance for Tree Models

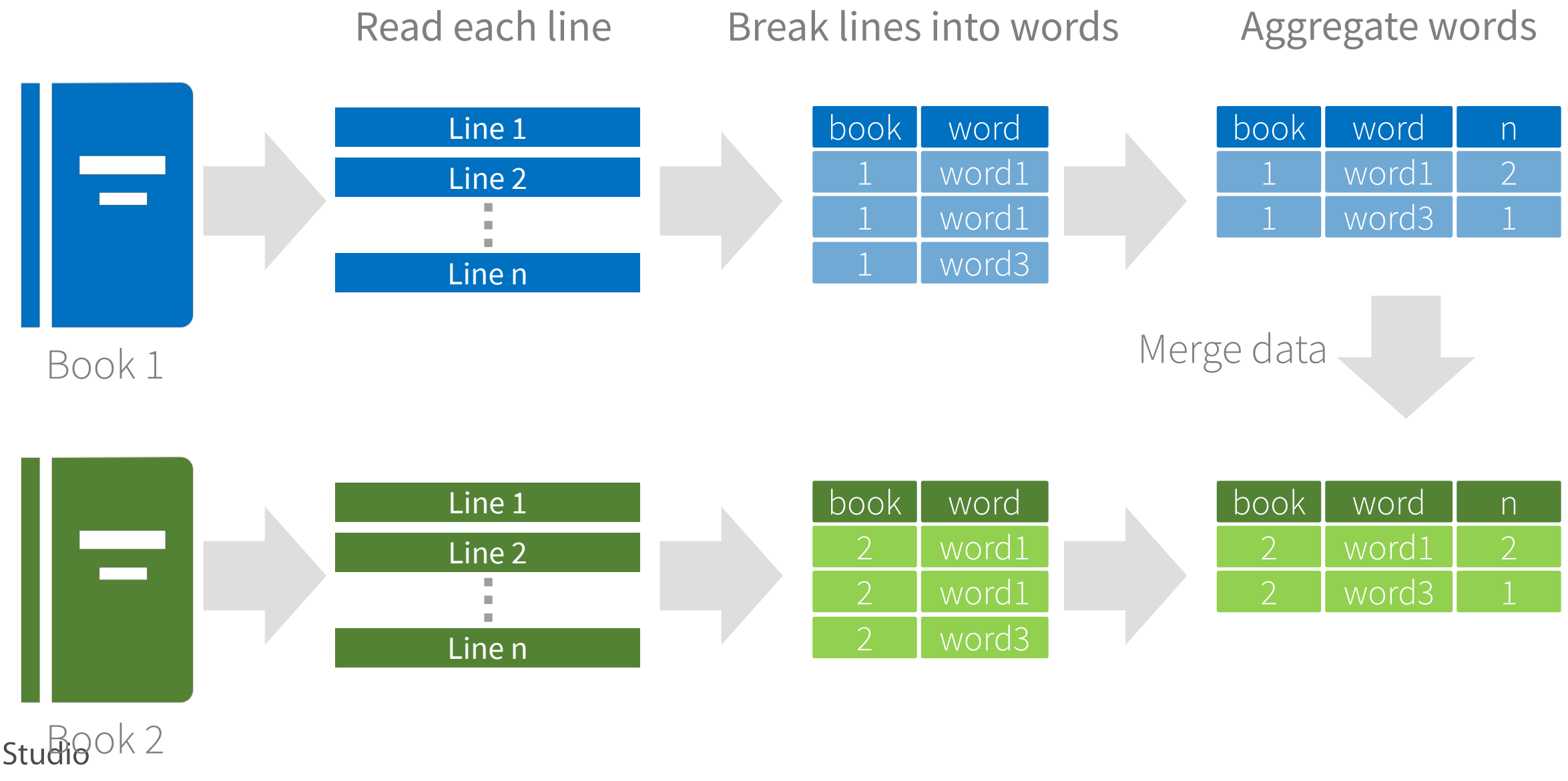
# Exercise 9.1 - 9.3

# Unit 10
# Text mining

Photo by Tucker Good on Unsplash

RStudio

# Compare text of two books



# Exercise 10.1 - 10.4

# Unit 11
# Memory Caching

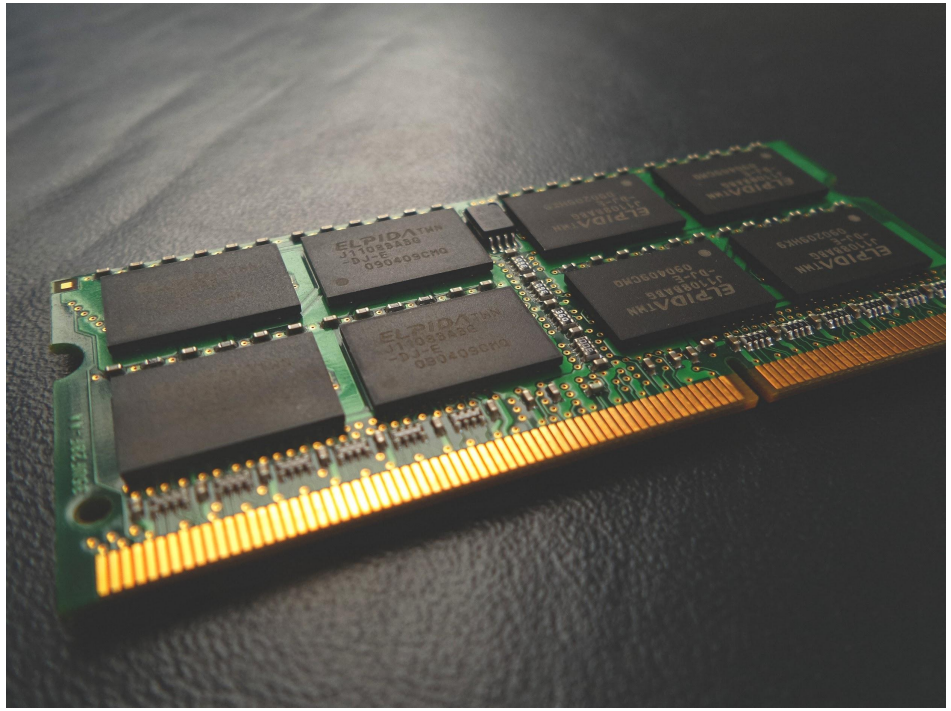
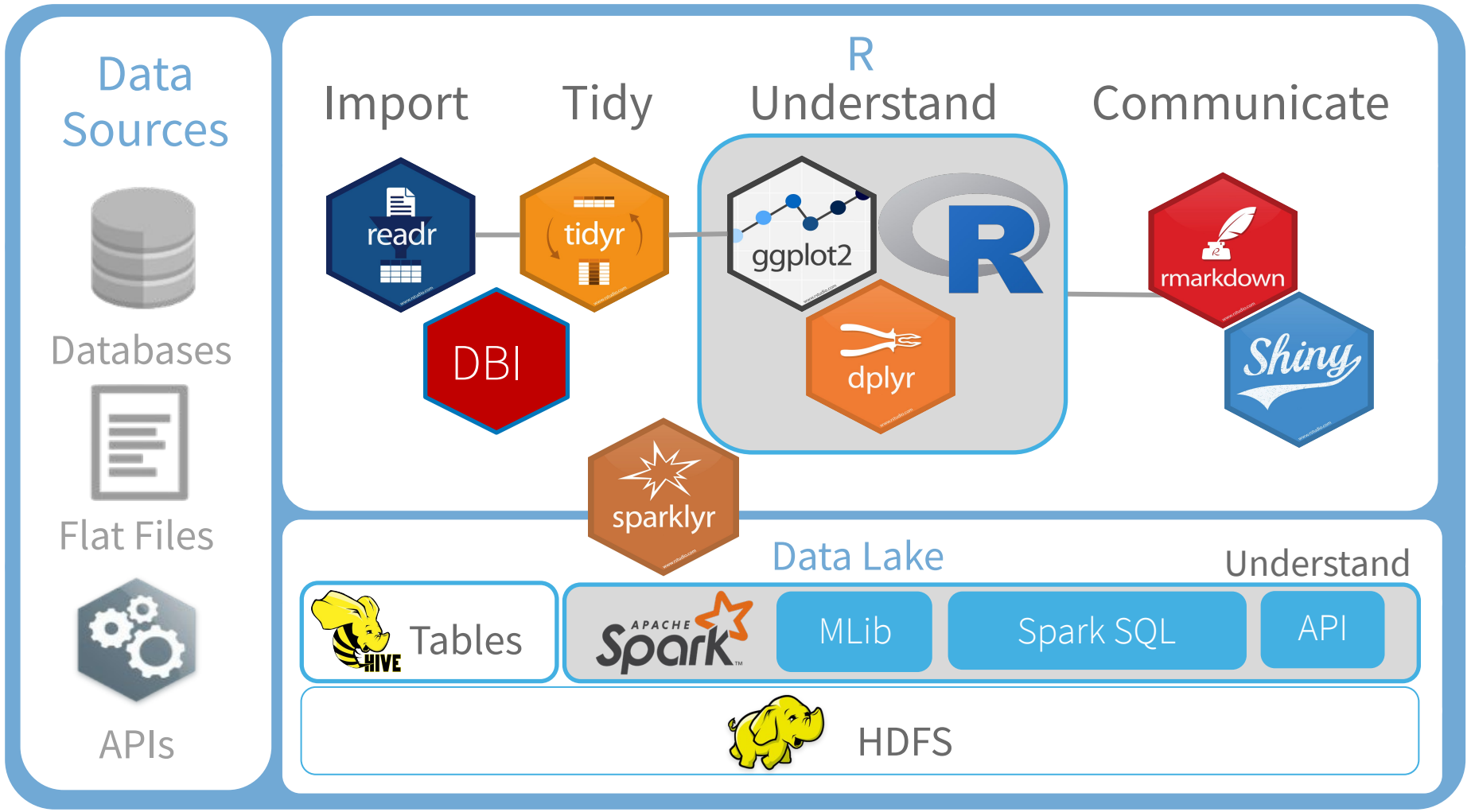


Photo by Franck V. on Unsplash

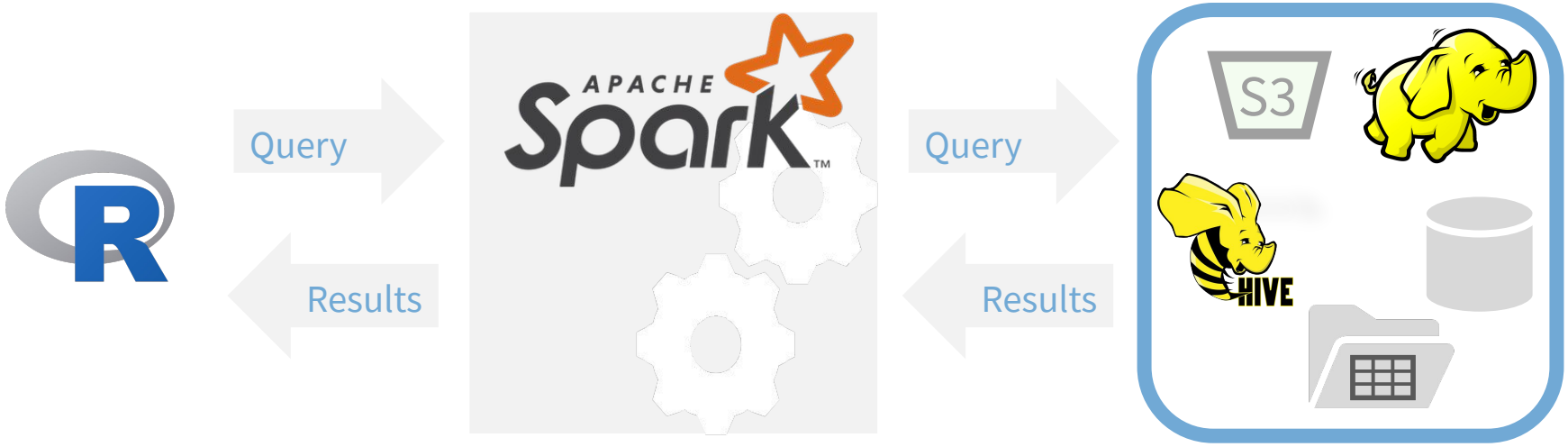
# Toolchain in a Data Lake



# Working with data in Spark

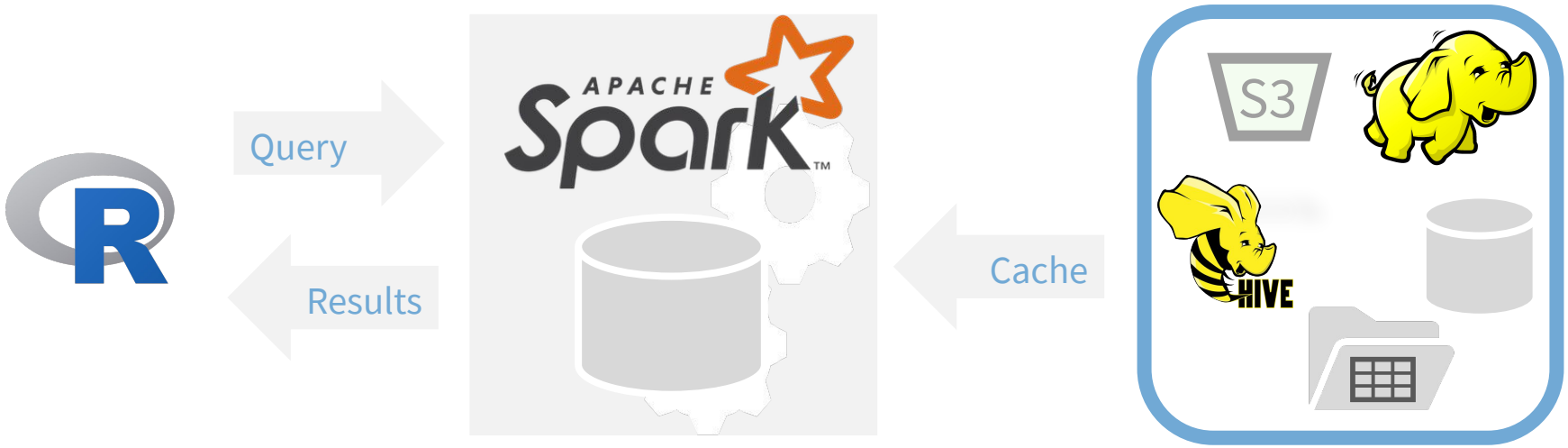
## Option 1

Use Spark as a pass-through for each query




## Option 2

Cache the data into Spark memory & query there



# Exercise 11.1 - 11.2

# Let's talk about Data Science projects

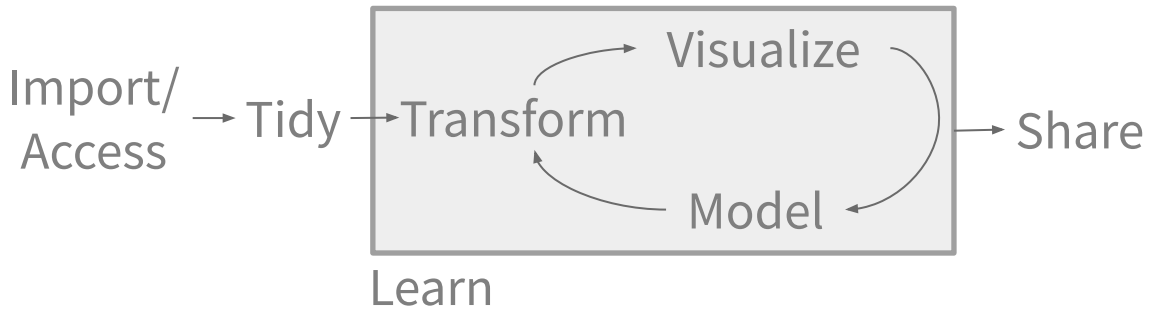

Photo by Jo Szczepanska on Unsplash

RStudio

# Different deliverables

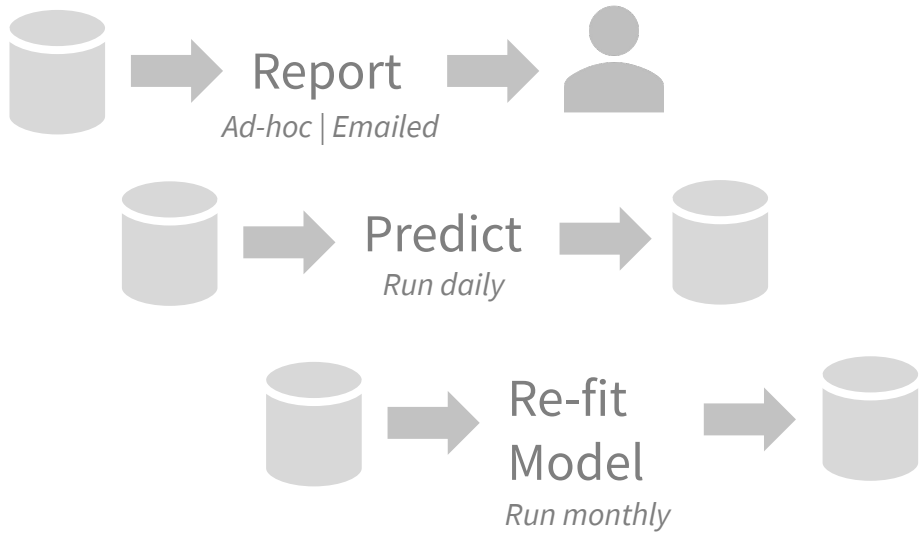
## Data Science

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




## Production

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



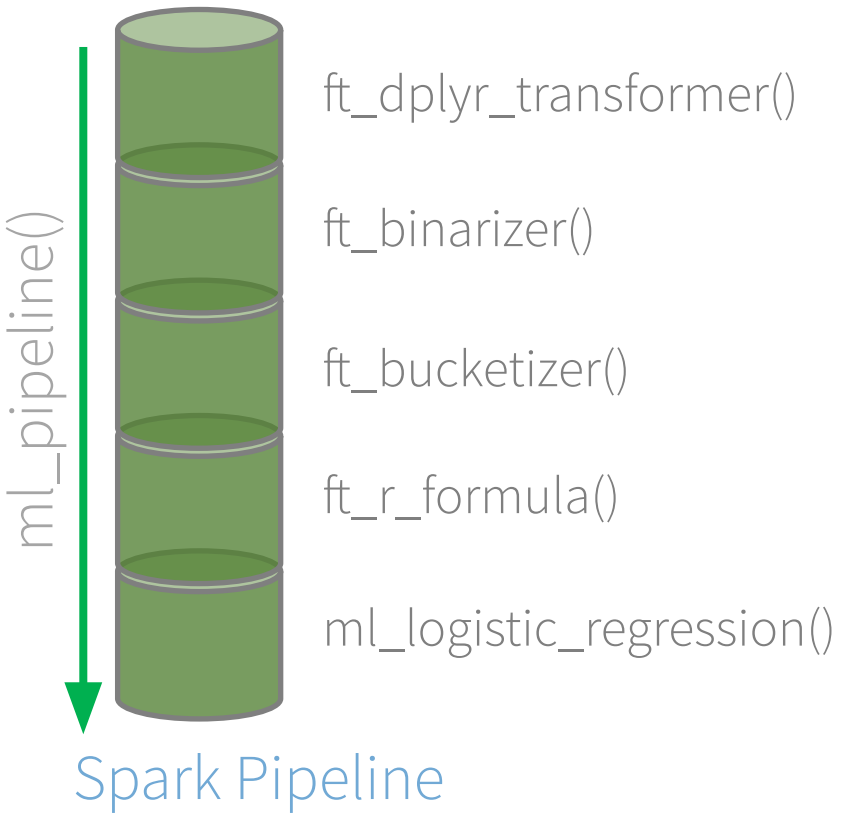
# Unit 12
# Spark Pipelines

Photo by Iker Urteaga on Unsplash

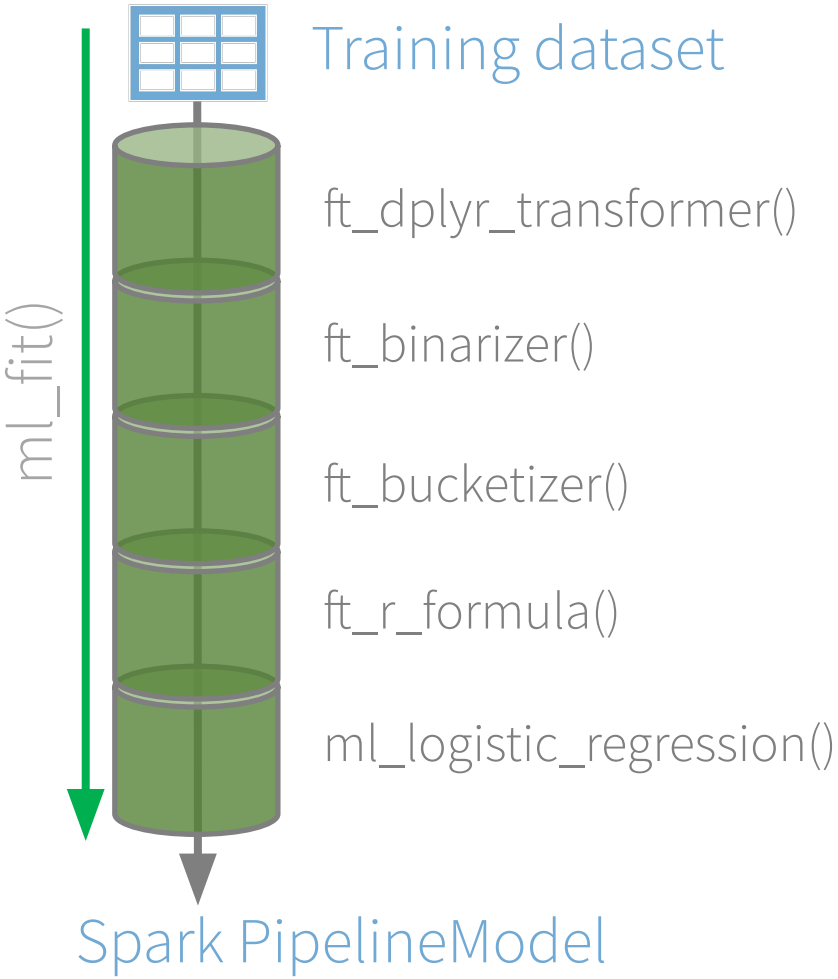
RStudio

# Spark pipelines types

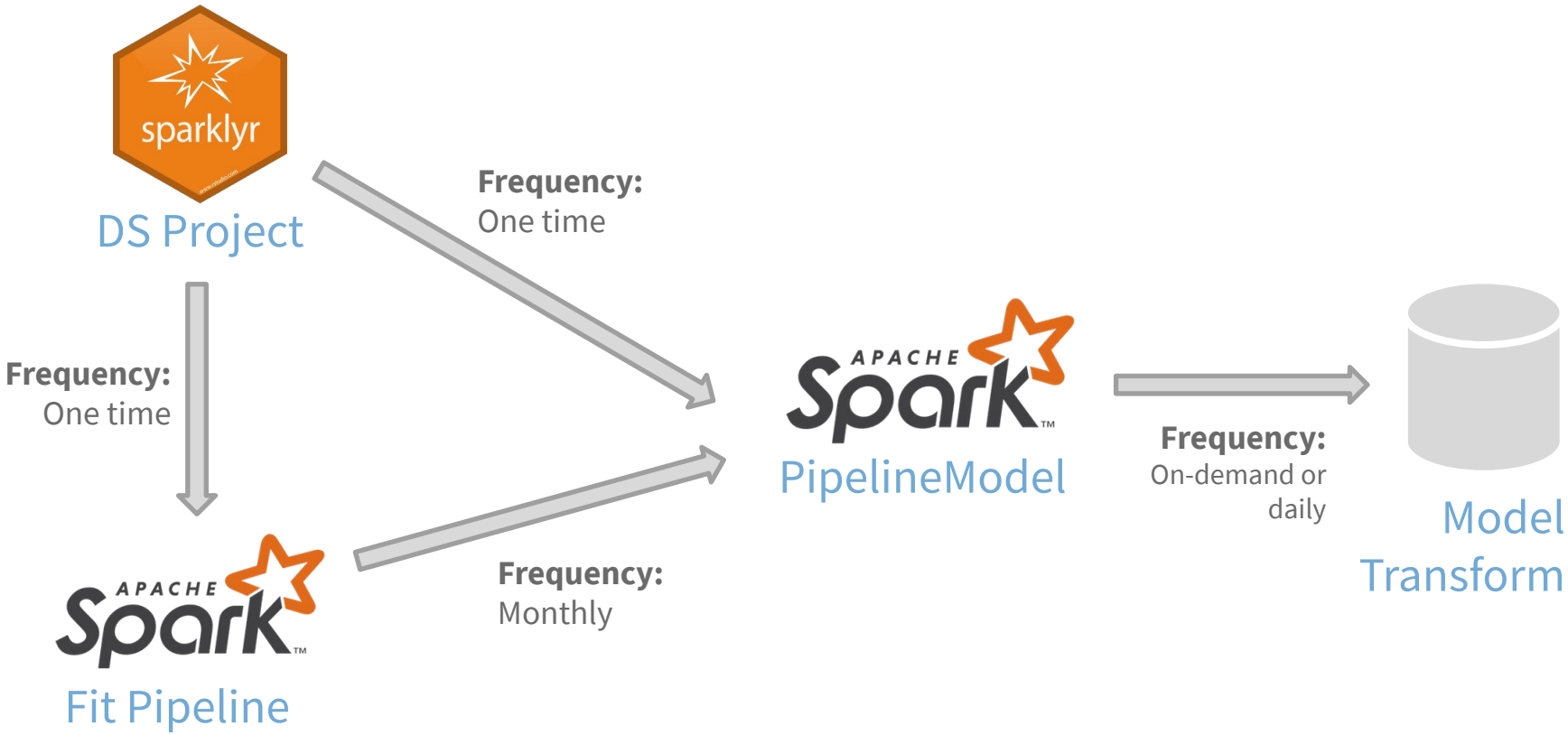
## **Estimator** (Plan)




## **Transformer** (Fit)



# Production Implementation

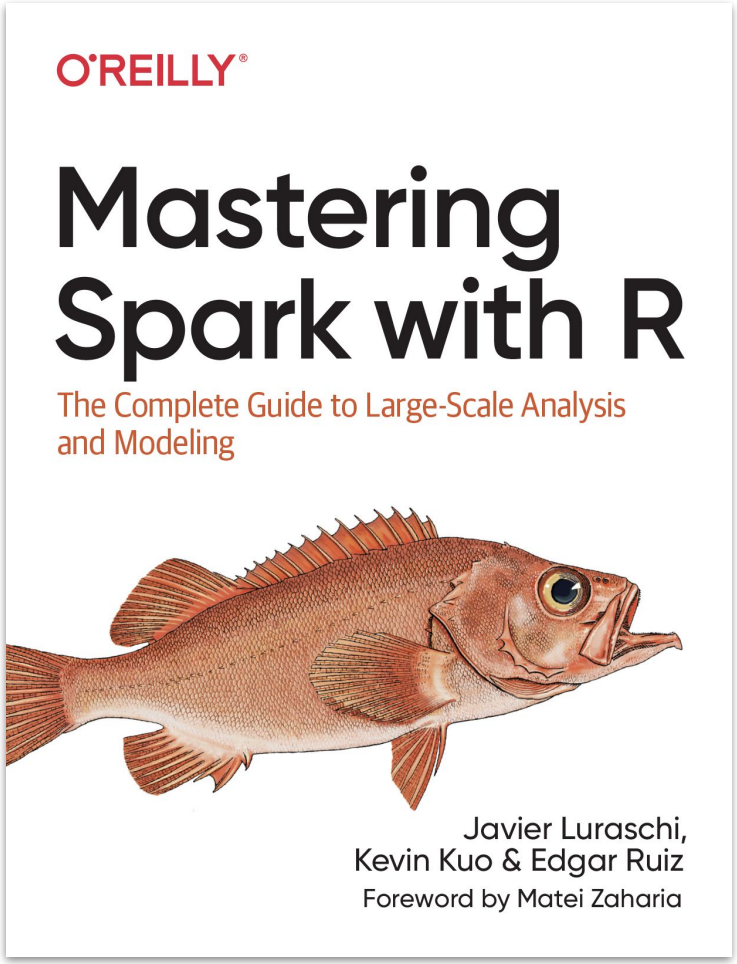


# Exercise 12.1 - 12.4

# General advice


Photo by Daria Nepriakhina on Unsplash

# The book...

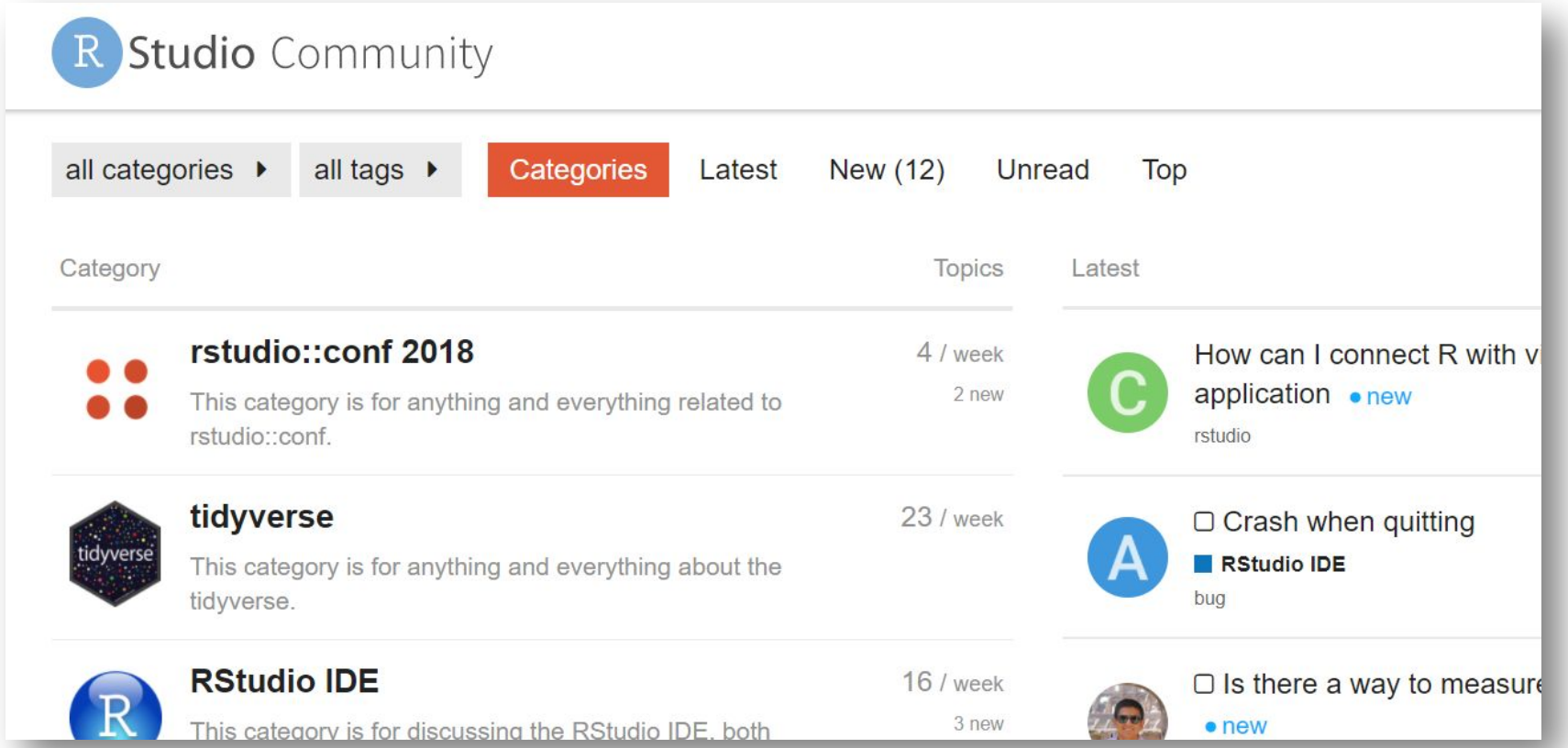



therinspark.com

# 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!




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/rstudio/sparklyr

rstd.io/ws-survey

# Thank you!!!!!

Photo by Gary Bendig on Unsplash