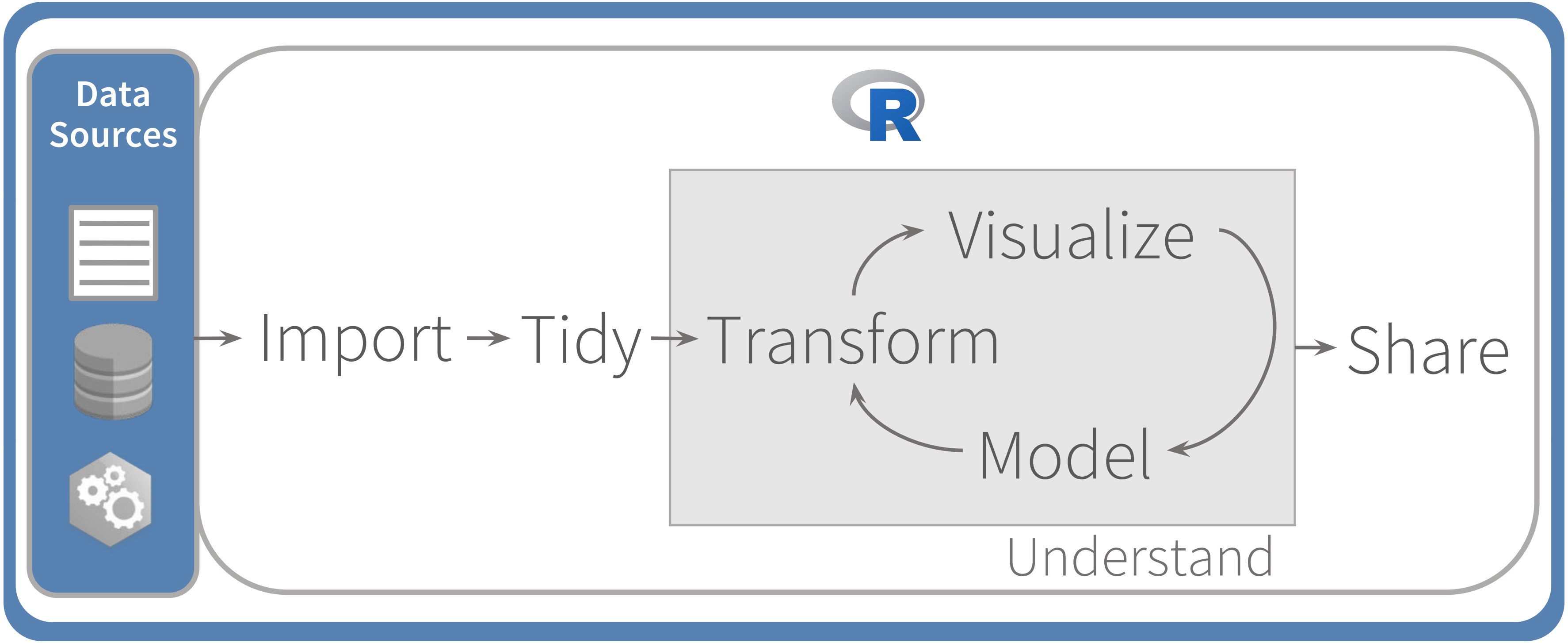
# Data Science with Spark & R

## Edgar Ruiz

Solutions Engineer

*linkedin.com/in/edgararuiz*
*twitter.com/theotheredgar*
*github.com/edgararuiz*

# Data Science

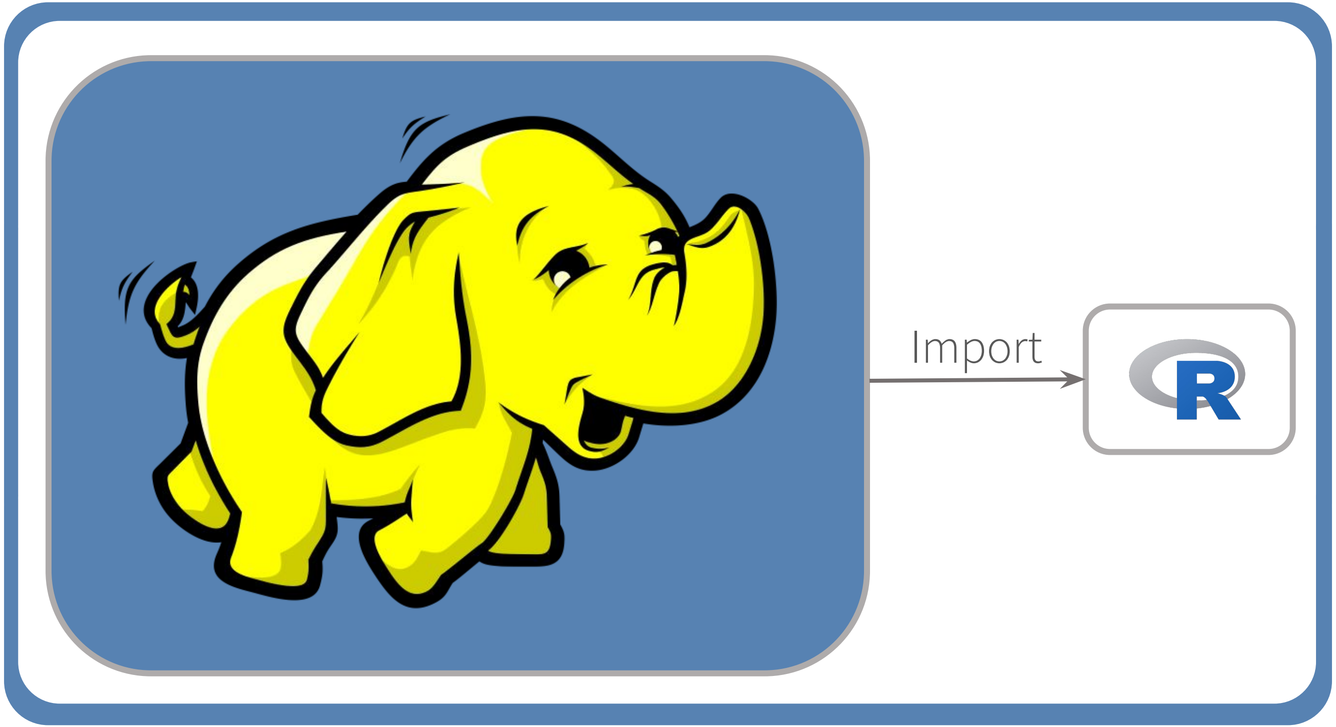


R for Data Science, Wickham & Grolemund

RStudio

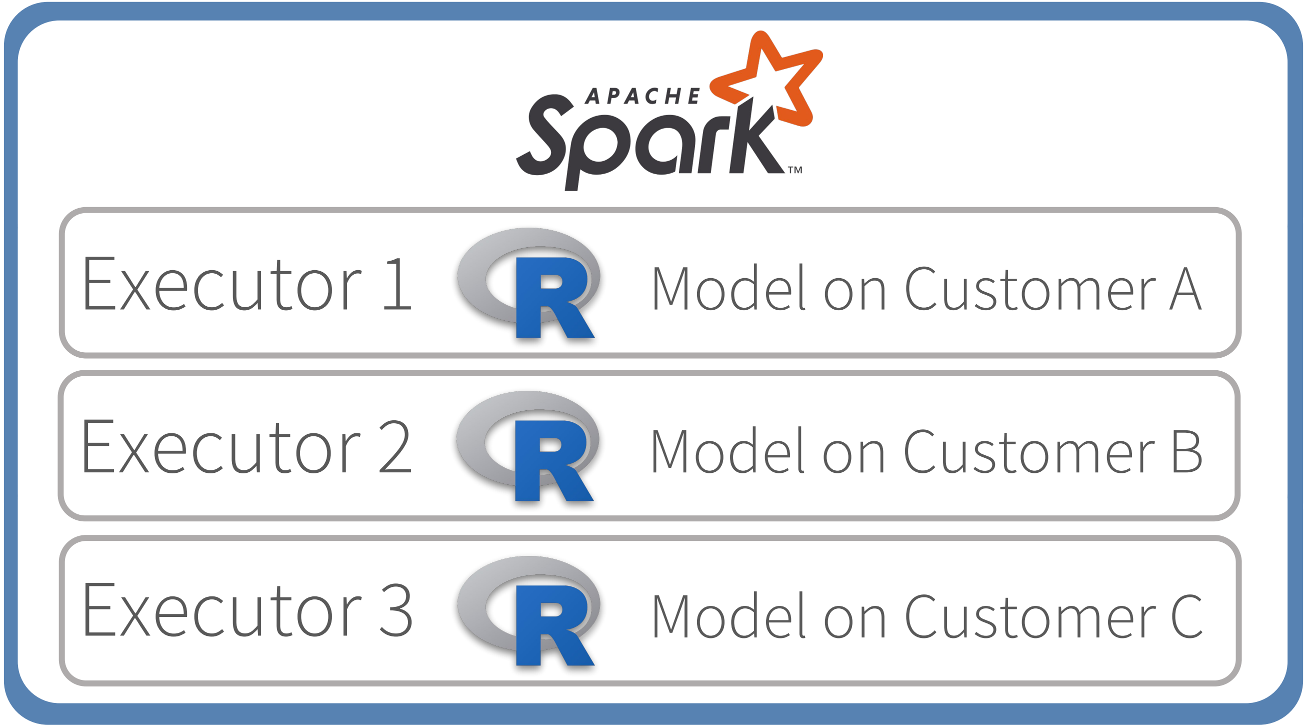
# Big Data as a Data Source

Too large to download into memory



# R inside Spark

Only appropriate for Embarrassingly Parallel cases



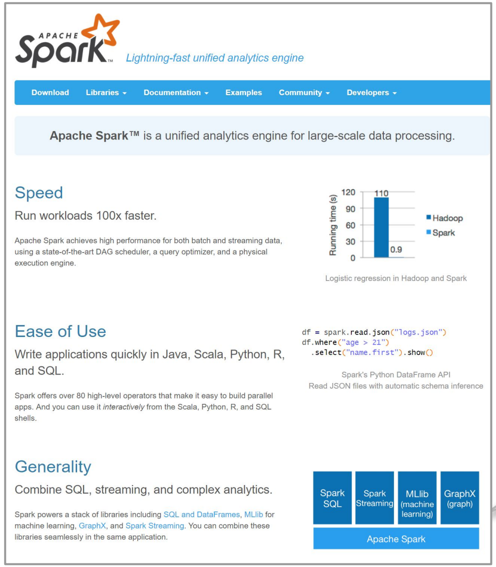
Executor 1 Model on Customer A

Executor 2 Model on Customer B

Executor 3 Model on Customer C

# We keep forgetting…

“Spark is a unified analytics engine for large-scale data processing”



APACHE Spark Lightning-fast unified analytics engine

Download | Libraries ▾ | Documentation ▾ | Examples | Community ▾ | Developers ▾

**Apache Spark™** is a unified analytics engine for large-scale data processing.

## Speed

Run workloads 100x faster.

Apache Spark achieves high performance for both batch and streaming data, using a state-of-the-art DAG scheduler, a query optimizer, and a physical execution engine.

Logistic regression in Hadoop and Spark

## Ease of Use

Write applications quickly in Java, Scala, Python, R, and SQL.

Spark offers over 80 high-level operators that make it easy to build parallel apps. And you can use it *interactively* from the Scala, Python, R, and SQL shells.

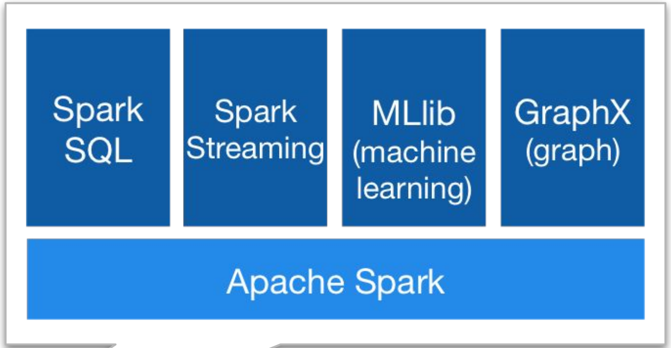
```
df = spark.read.json("logs.json")
df.where("age > 21")
  .select("name.first").show()
```

Spark's Python DataFrame API
Read JSON files with automatic schema inference

## Generality

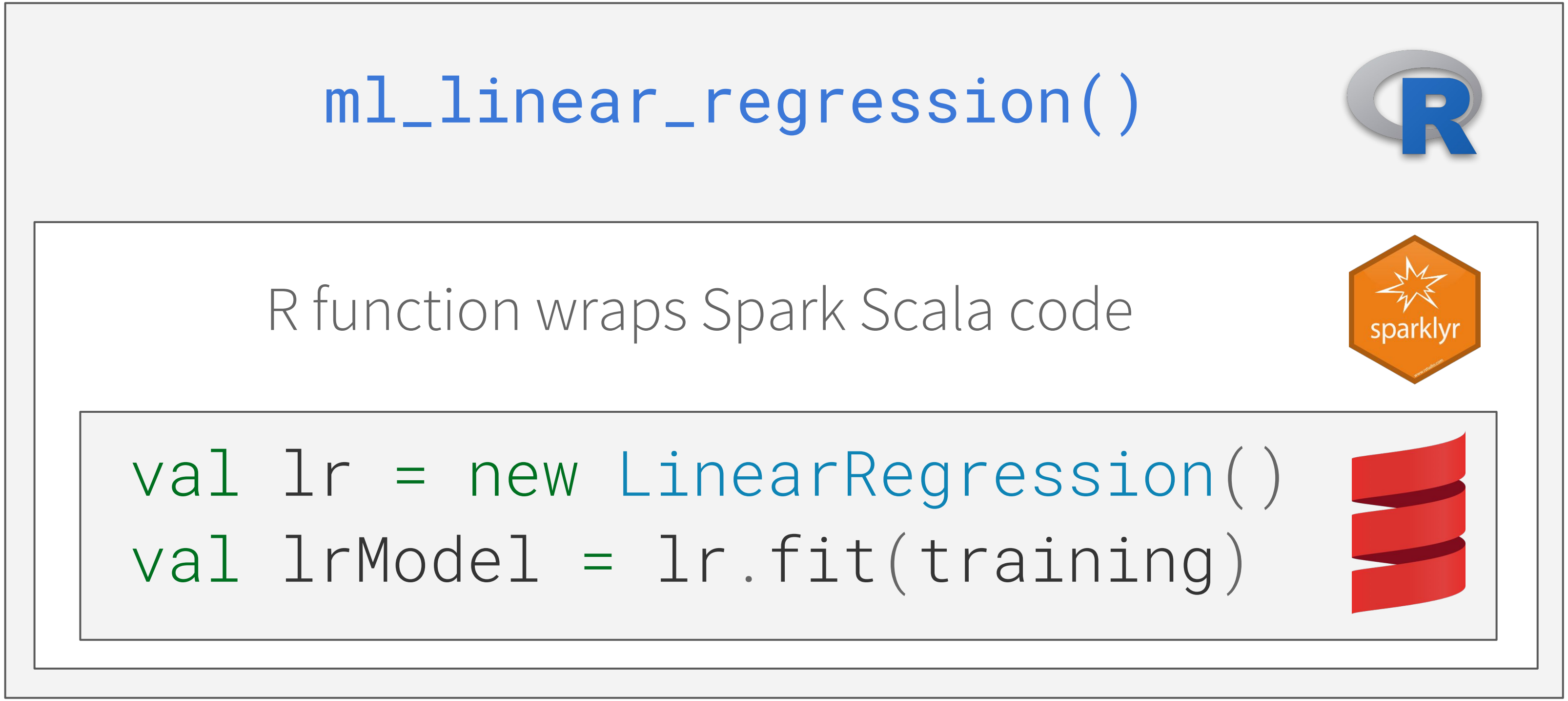
Combine SQL, streaming, and complex analytics.

Spark powers a stack of libraries including SQL and DataFrames, MLlib for machine learning, GraphX, and Spark Streaming. You can combine these libraries seamlessly in the same application.



Source: Spark official website

# A better way… R as an ***interface*** for Spark

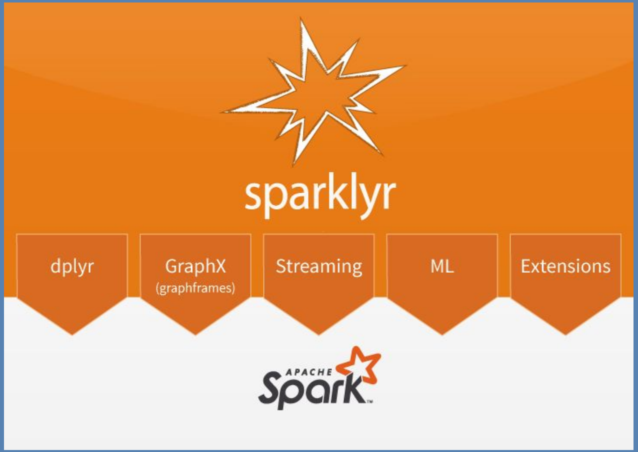


```r
ml_linear_regression()
```

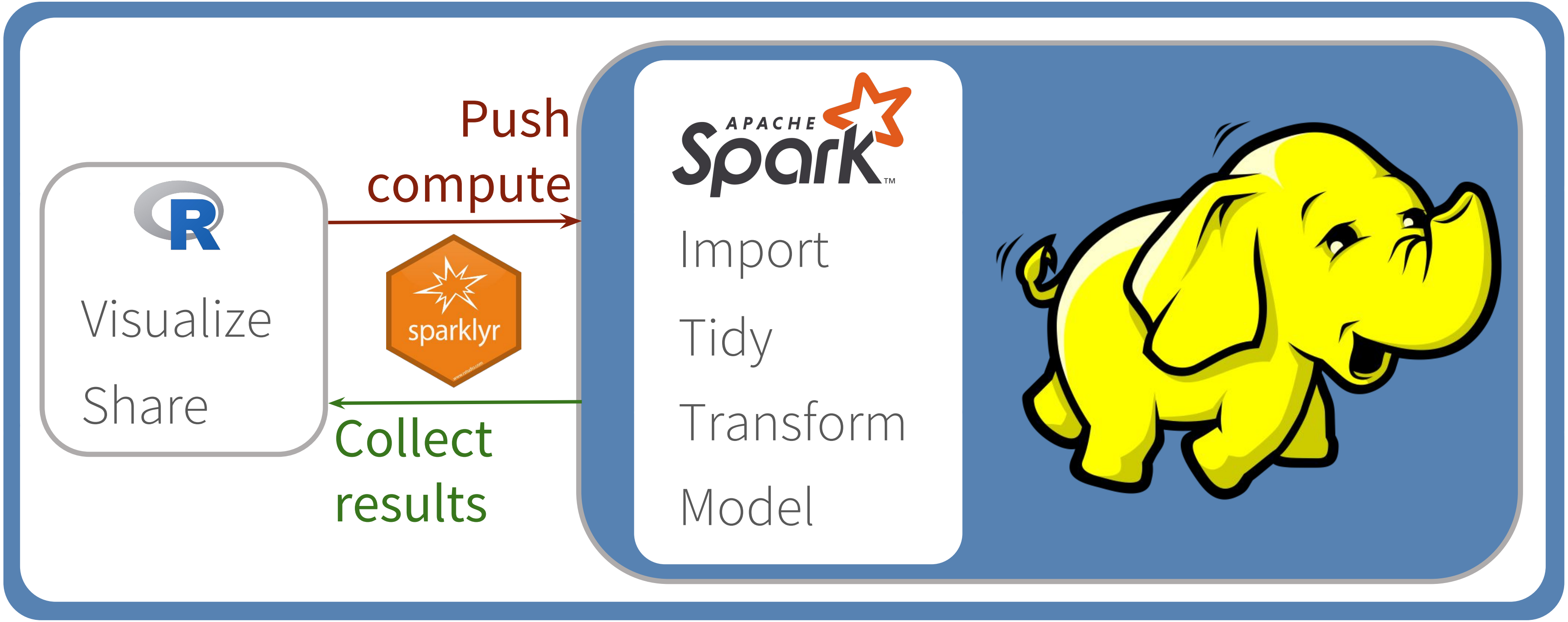
R function wraps Spark Scala code

```scala
val lr = new LinearRegression()
val lrModel = lr.fit(training)
```

# ~~Possibilities~~ Realities of using R as an interface for Spark



# 1. Most Data Science steps can run inside Spark



Visualize

Share

Push compute

Collect results

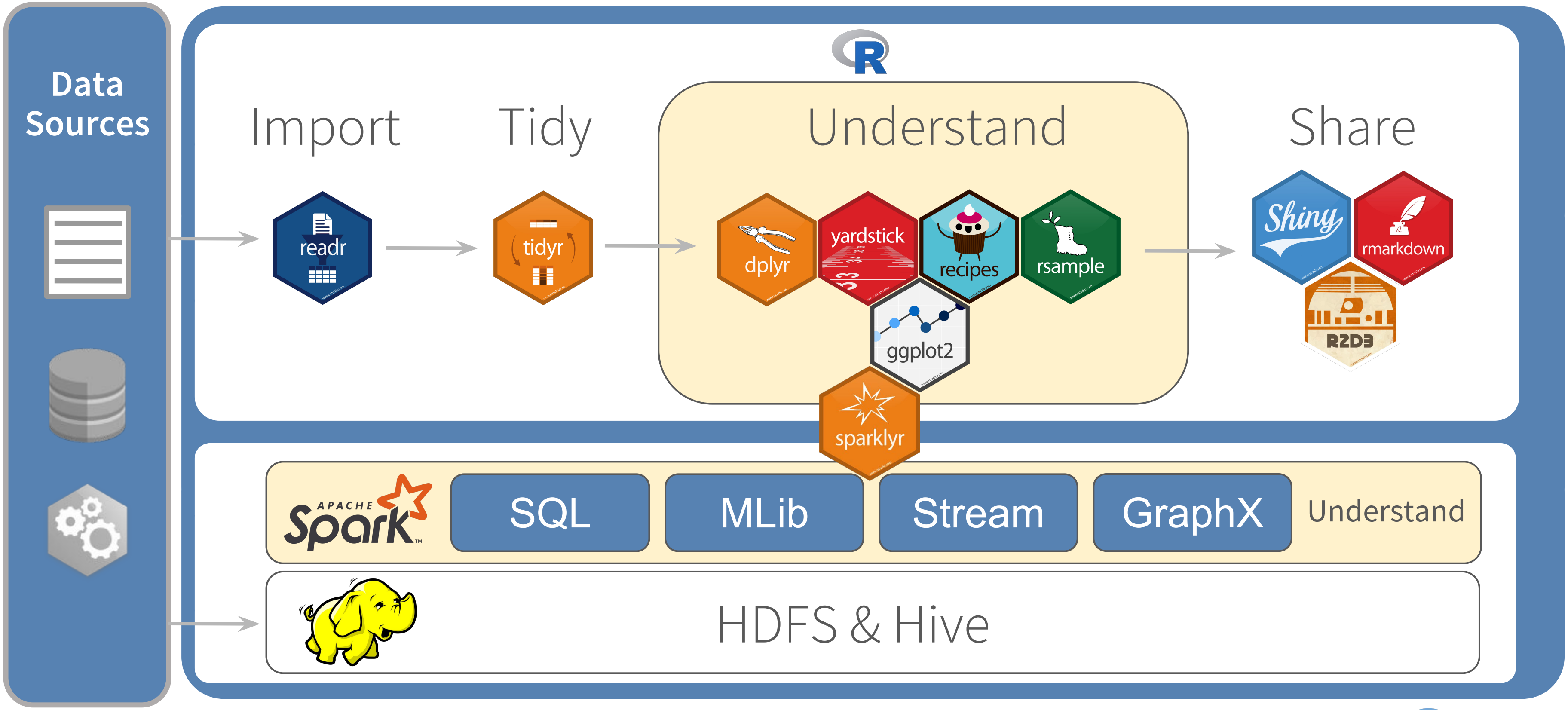
Import

Tidy

Transform

Model

# 2. Best of both worlds: Spark API + R ecosystem



RStudio

# 3. Hand off to Production: ML Pipelines built in R

Steps built in R fashion

```
new <- ml_pipeline(sc) %>%
  ft_dplyr_transformer(
    tbl = pipeline_df
  ) %>%
  ft_binarizer(
    input_col = "arrdelay",
    output_col = "delayed",
    threshold = 15
  ) %>%
  ft_bucketizer(
    input_col = "crsdeptime",
    output_col = "dephour",
    splits = c(0, 400, 800)
  ) %>%
  ft_r_formula(
    delayed ~ arrdelay + dephour
  ) %>%
  ml_logistic_regression()
```

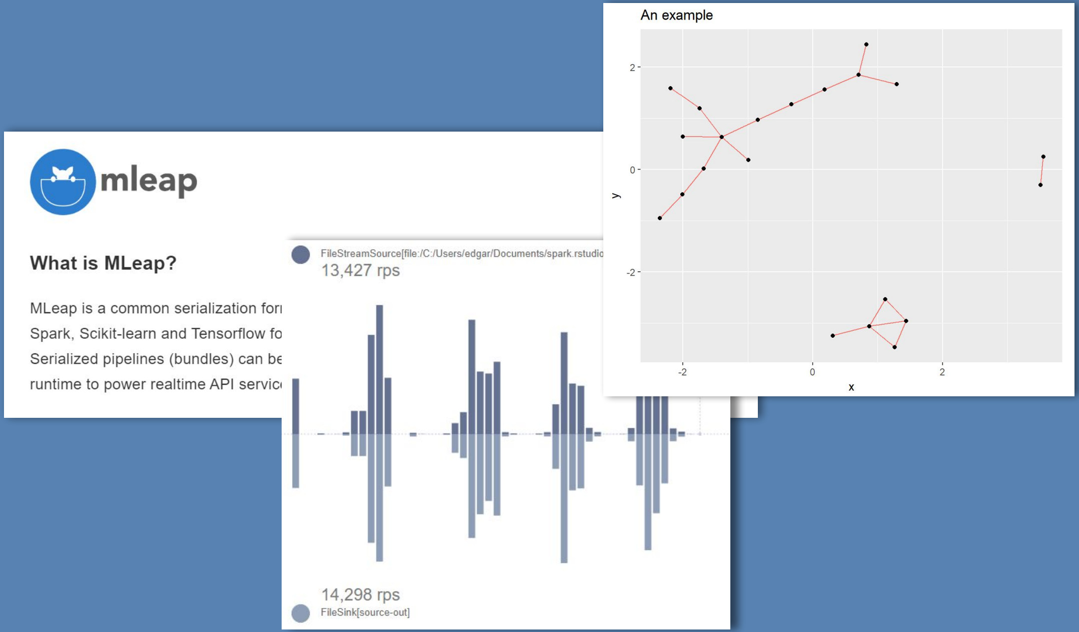
Pipeline and PipelineModel are saved in Scala code

```
ml_save(new)
```

```
ml_fit(new) %>%
  ml_save()
```

Run Pipelines in Spark with **0** **dependencies** on R

# Interface highlights



mleap

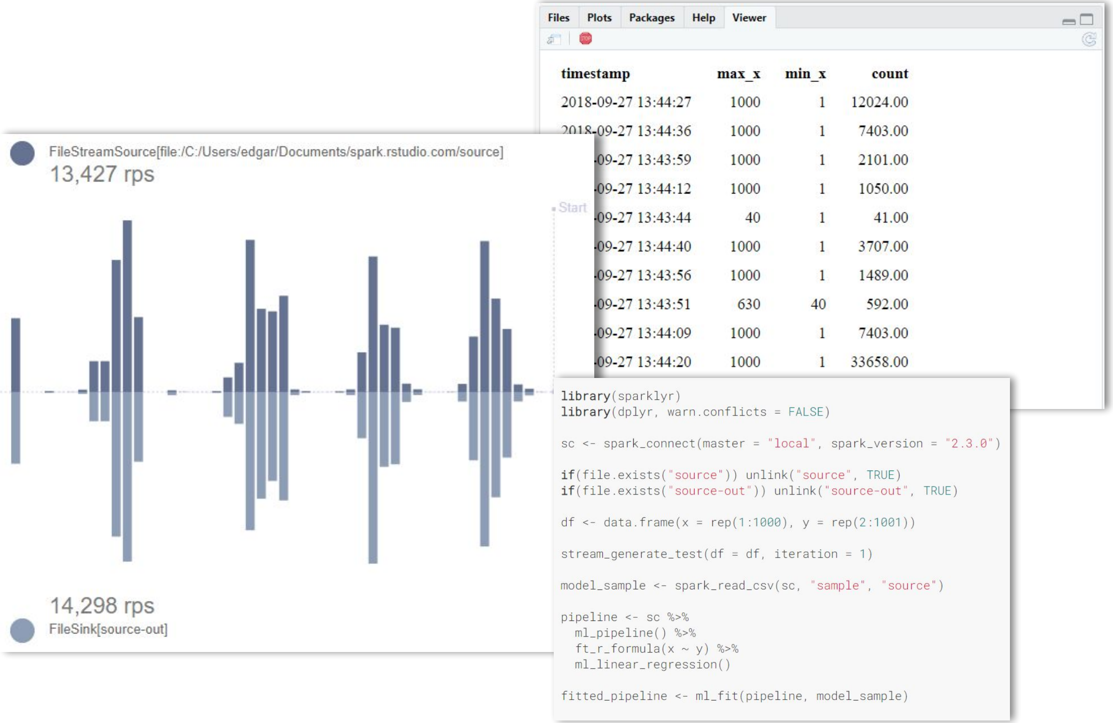
What is MLeap?

MLeap is a common serialization fori
Spark, Scikit-learn and Tensorflow fo
Serialized pipelines (bundles) can be
runtime to power realtime API servic

FileStreamSource[file:/C:/Users/edgar/Documents/spark.rstudio
13,427 rps
14,298 rps
FileSink[source-out]
An example
2
0
-2
y
-2
0
2
x

# 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


Files
Plots
Packages
Help
Viewer
FileStreamSource[file:/C:/Users/edgar/Documents/spark.rstudio.com/source]
13,427 rps
Start
14,298 rps
FileSink[source-out]

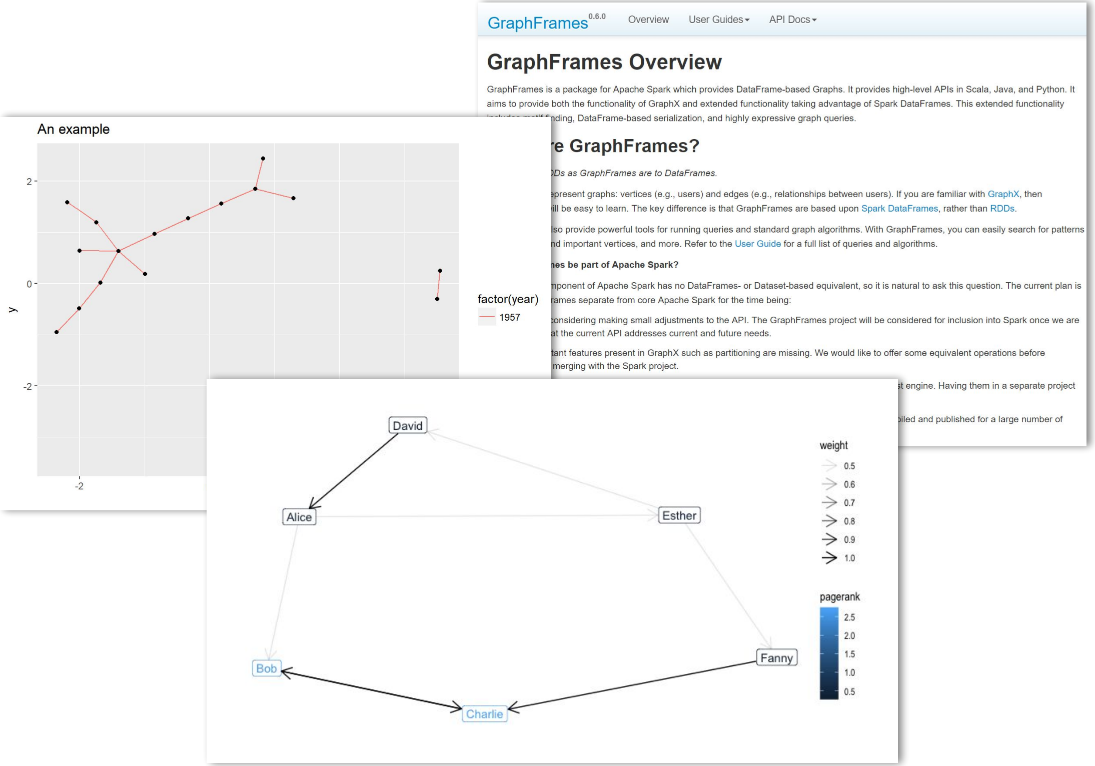
| timestamp | max_x | min_x | count |
|---|---|---|---|
| 2018-09-27 13:44:27 | 1000 | 1 | 12024.00 |
| 2018-09-27 13:44:36 | 1000 | 1 | 7403.00 |
| 09-27 13:43:59 | 1000 | 1 | 2101.00 |
| 09-27 13:44:12 | 1000 | 1 | 1050.00 |
| 09-27 13:43:44 | 40 | 1 | 41.00 |
| 09-27 13:44:40 | 1000 | 1 | 3707.00 |
| 09-27 13:43:56 | 1000 | 1 | 1489.00 |
| 09-27 13:43:51 | 630 | 40 | 592.00 |
| 09-27 13:44:09 | 1000 | 1 | 7403.00 |
| 09-27 13:44:20 | 1000 | 1 | 33658.00 |

```r
library(sparklyr)
library(dplyr, warn.conflicts = FALSE)

sc <- spark_connect(master = "local", spark_version = "2.3.0")

if(file.exists("source")) unlink("source", TRUE)
if(file.exists("source-out")) unlink("source-out", TRUE)

df <- data.frame(x = rep(1:1000), y = rep(2:1001))

stream_generate_test(df = df, iteration = 1)

model_sample <- spark_read_csv(sc, "sample", "source")

pipeline <- sc %>%
  ml_pipeline() %>%
  ft_r_formula(x ~ y) %>%
  ml_linear_regression()

fitted_pipeline <- ml_fit(pipeline, model_sample)
```

https://spark.rstudio.com/guides/streaming/

# 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



## GraphFrames Overview

GraphFrames is a package for Apache Spark which provides DataFrame-based Graphs. It provides high-level APIs in Scala, Java, and Python. It aims to provide both the functionality of GraphX and extended functionality taking advantage of Spark DataFrames. This extended functionality nding, DataFrame-based serialization, and highly expressive graph queries.

## e GraphFrames?

*Ds as GraphFrames are to DataFrames.*

epresent graphs: vertices (e.g., users) and edges (e.g., relationships between users). If you are familiar with GraphX, then ill be easy to learn. The key difference is that GraphFrames are based upon Spark DataFrames, rather than RDDs.

lso provide powerful tools for running queries and standard graph algorithms. With GraphFrames, you can easily search for patterns nd important vertices, and more. Refer to the User Guide for a full list of queries and algorithms.

**nes be part of Apache Spark?**

mponent of Apache Spark has no DataFrames- or Dataset-based equivalent, so it is natural to ask this question. The current plan is rames separate from core Apache Spark for the time being:

considering making small adjustments to the API. The GraphFrames project will be considered for inclusion into Spark once we are at the current API addresses current and future needs.

tant features present in GraphX such as partitioning are missing. We would like to offer some equivalent operations before merging with the Spark project.

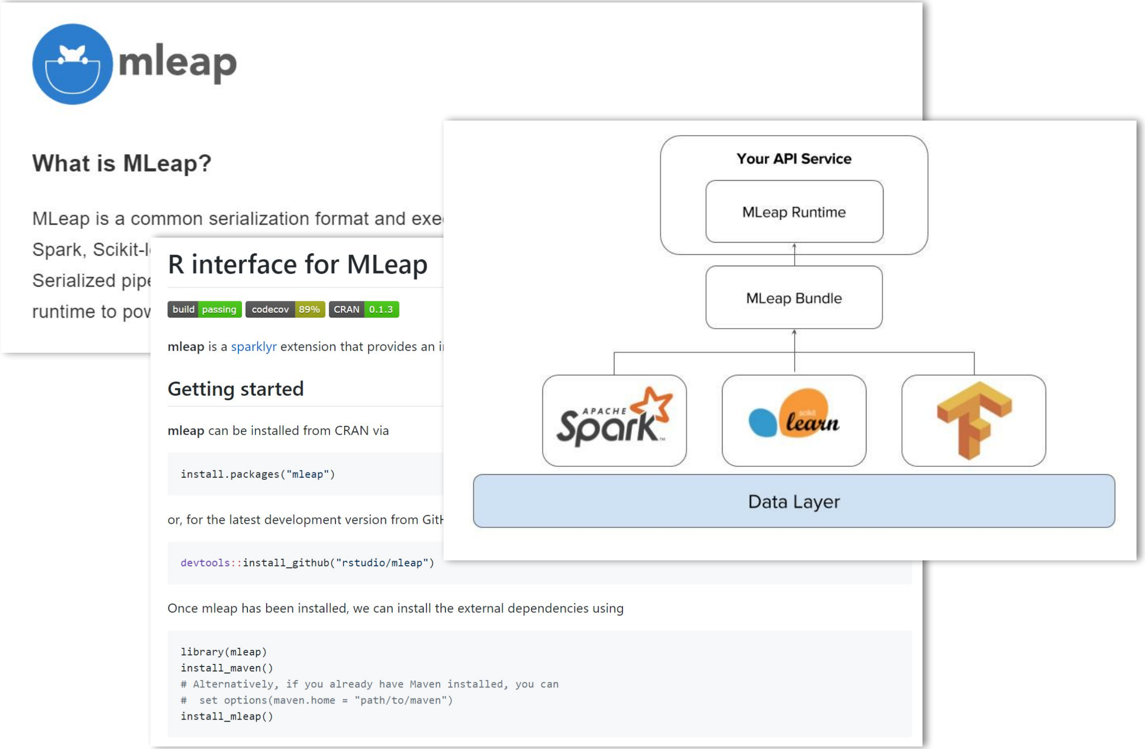
t engine. Having them in a separate project

oiled and published for a large number of

https://spark.rstudio.com/graphframes/

# Production pipelines with Mleap

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



mleap

### What is MLeap?

MLeap is a common serialization format and exe
Spark, Scikit-l
Serialized pipe
runtime to pov

## R interface for MLeap

build passing | codecov 89% | CRAN 0.1.3

**mleap** is a sparklyr extension that provides an i

### Getting started

**mleap** can be installed from CRAN via

```
install.packages("mleap")
```

or, for the latest development version from Gith

```
devtools::install_github("rstudio/mleap")
```

Once mleap has been installed, we can install the external dependencies using

```
library(mleap)
install_maven()
# Alternatively, if you already have Maven installed, you can
#  set options(maven.home = "path/to/maven")
install_mleap()
```

https://spark.rstudio.com/guides/mleap/