

# An introduction to Spark with R

Edgar Ruiz

[github.com/edgararuiz](https://github.com/edgararuiz)

[twitter.com/theotheredgar](https://twitter.com/theotheredgar)

[linkedin.com/in/edgararuiz](https://linkedin.com/in/edgararuiz)

# Data Science Workflow



Begin

End

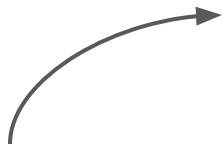
Import



Tidy



Transform



Visualize



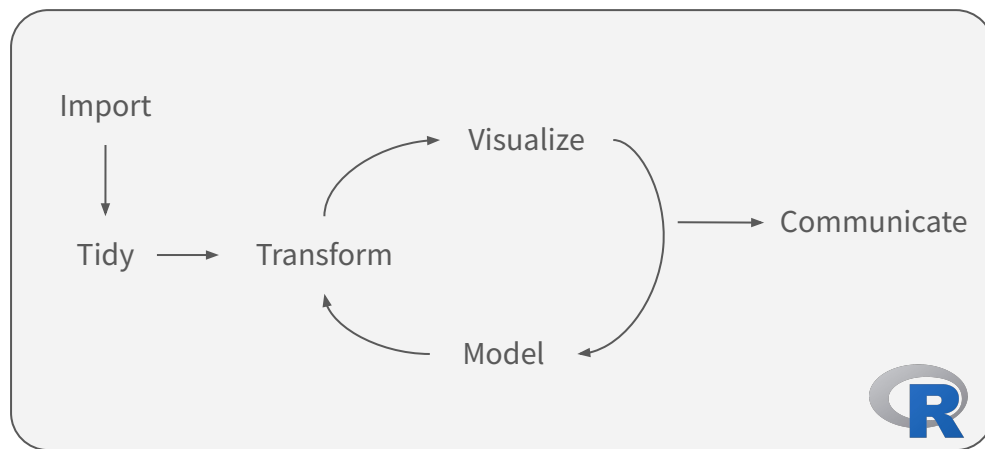
Model



Communicate

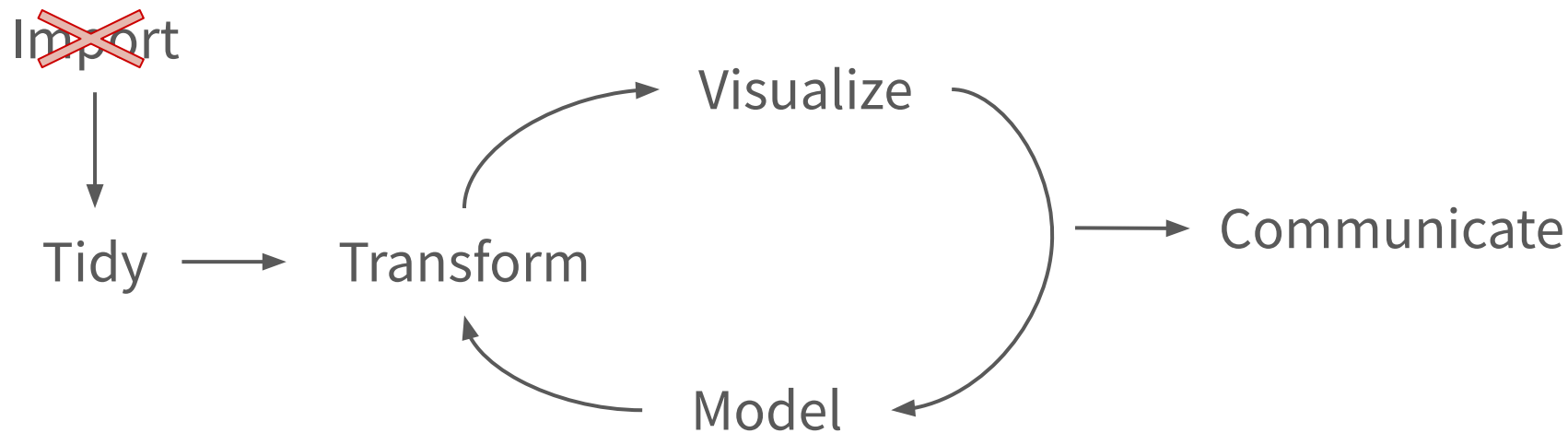
*Understand*

# A single laptop handles “small data” easy



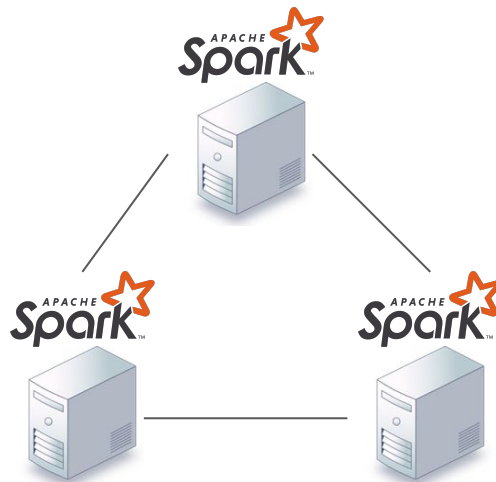
My laptop

# What about data **larger** than memory or disk?



# What is Spark?

- Allows the ability to work with data larger than disk or memory
- Interact with and manipulate data via SQL
- Has additional data transformers that go beyond SQL
- Has a robust set of Machine Learning routines
- Allows the creation of formal modeling pipelines
- Unlike DB's, Spark allows for easy in-memory data caching

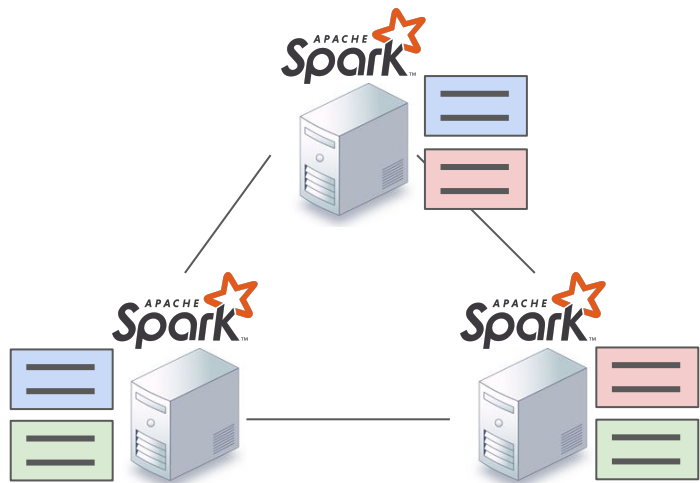
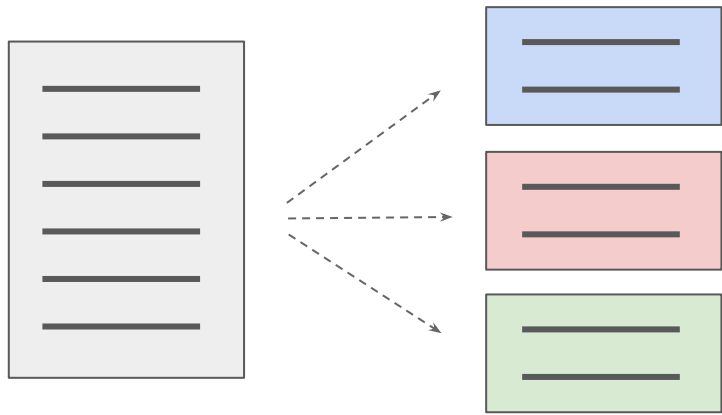


# How does Spark work?

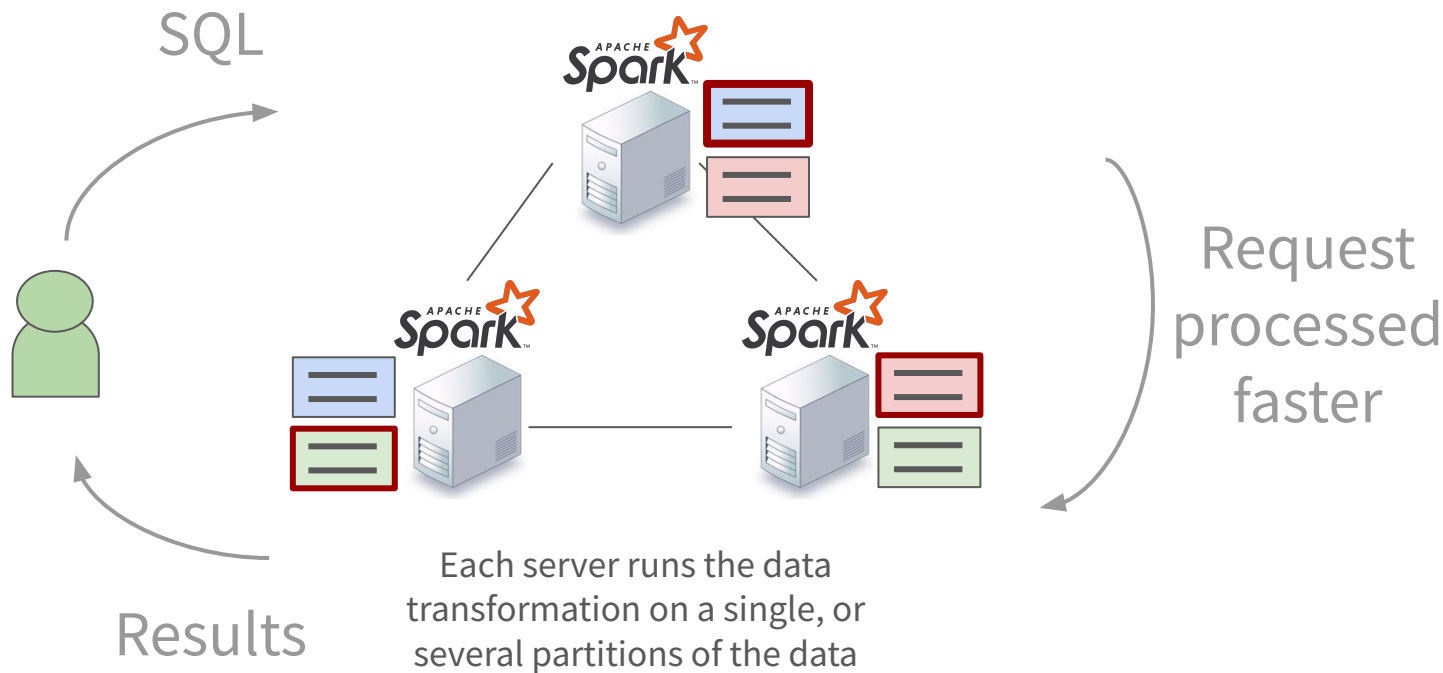
Original large  
data file

File is partitioned  
logically

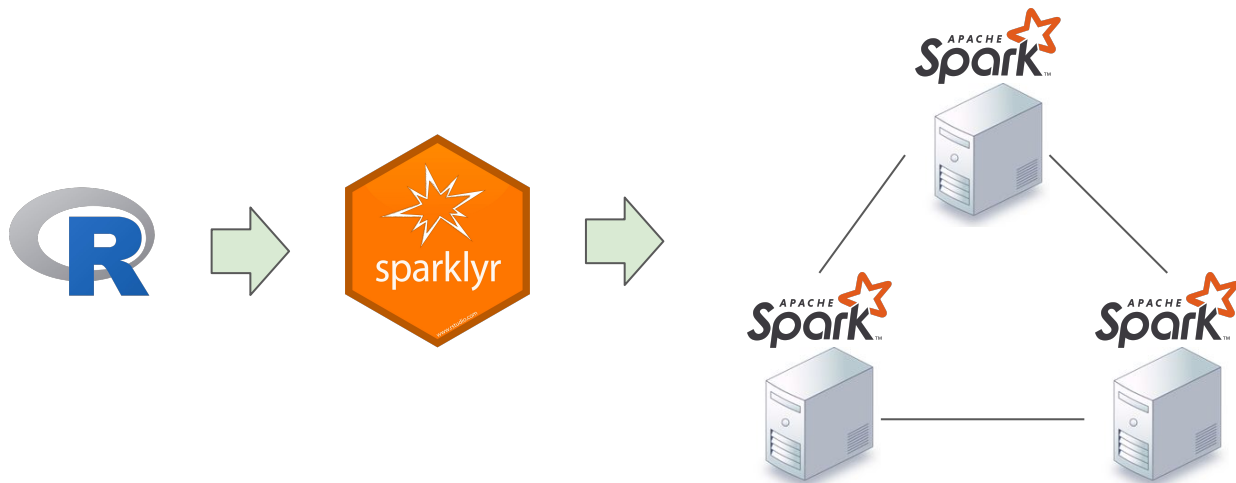
One or several partitions are sent to  
each server's **memory** in the cluster



# How does Spark work?



# How do I interact with Spark?





# Data Science with R, and Spark



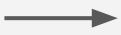
Begin

End

Import 



Tidy 



Transform 



  
Visualize

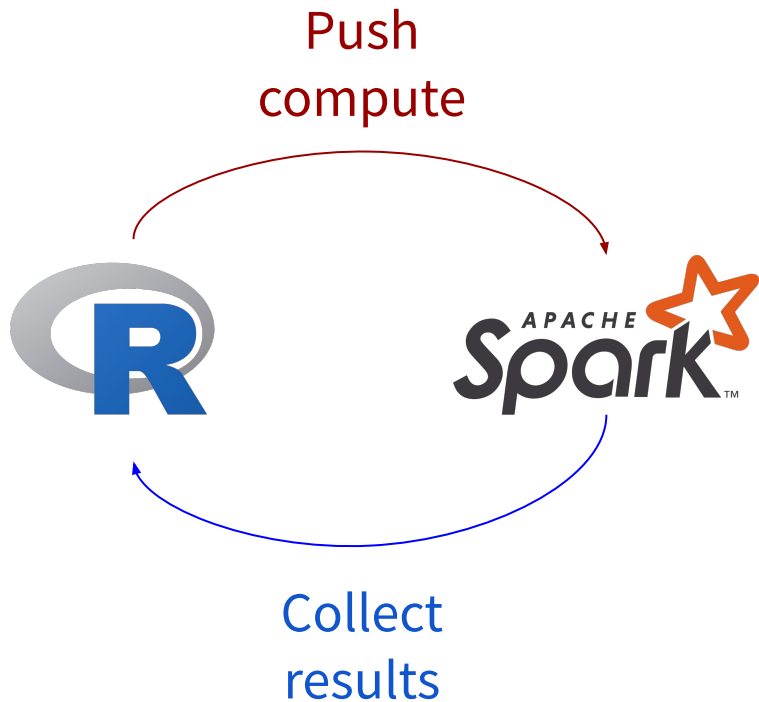


Model 

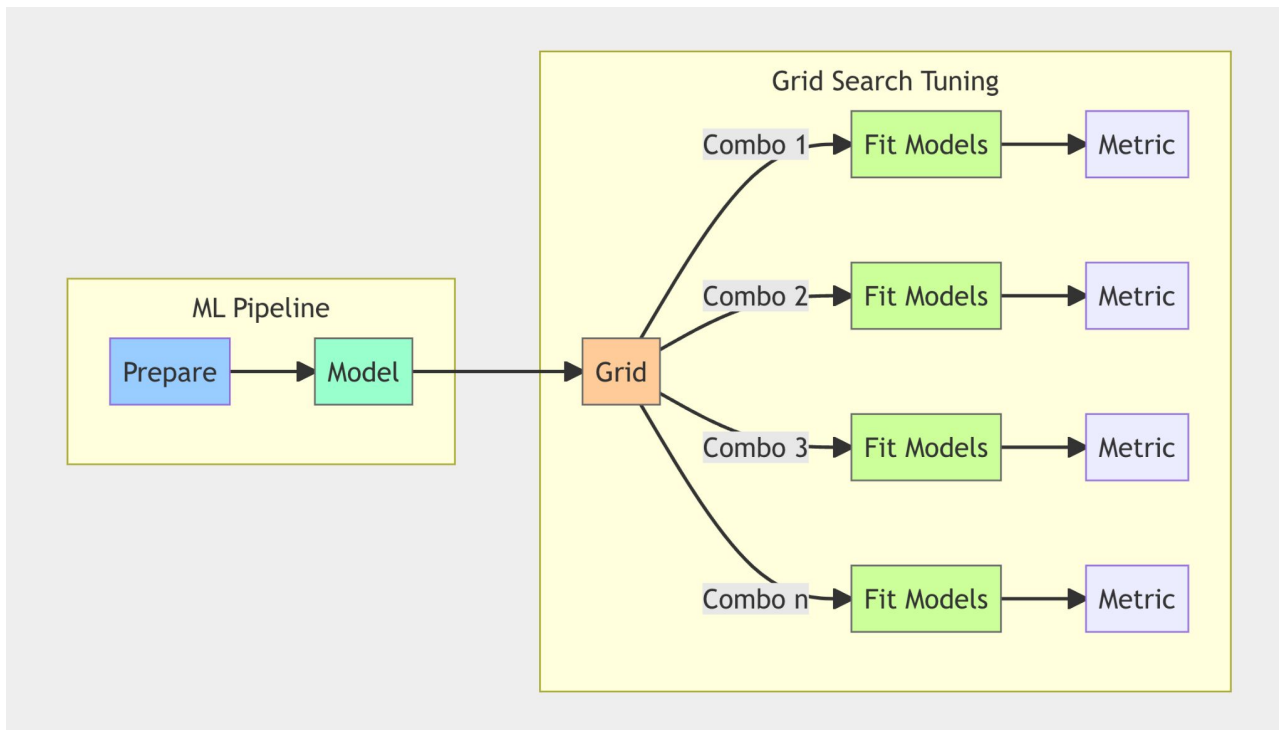


Communicate 

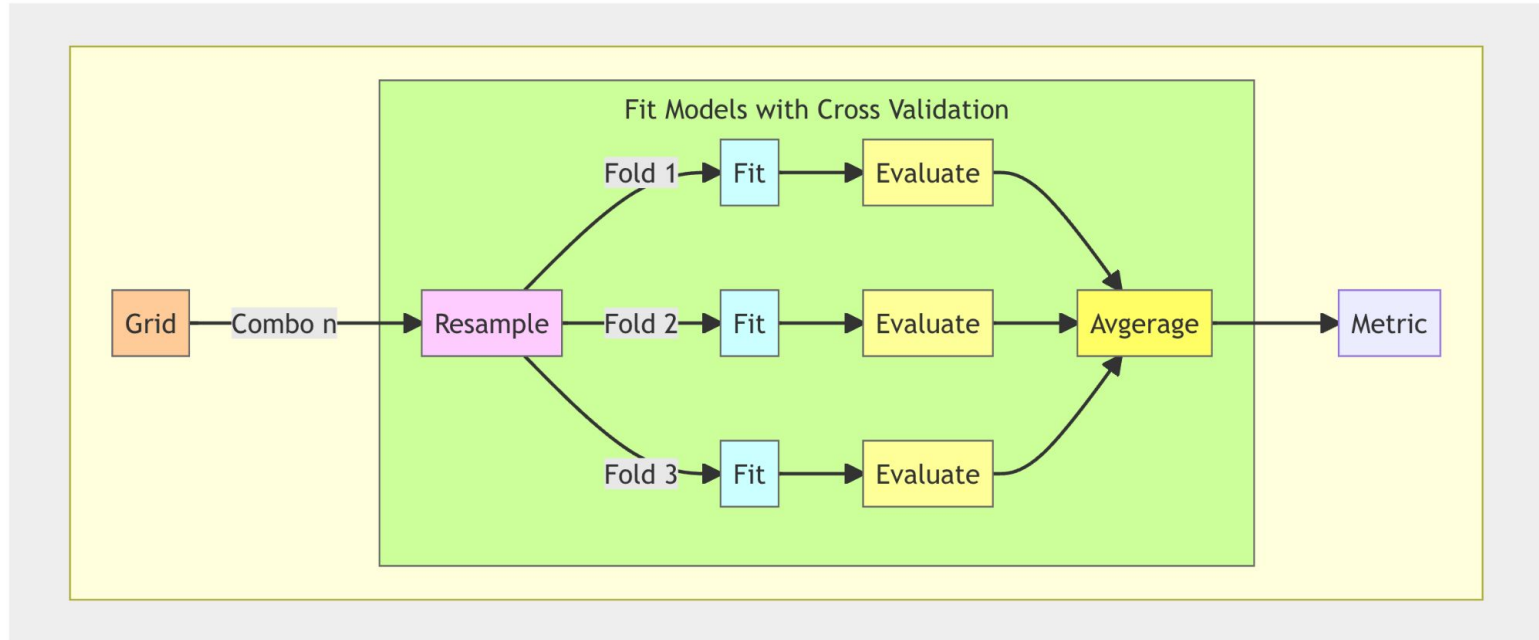
# Main takeaway when working with R and Spark



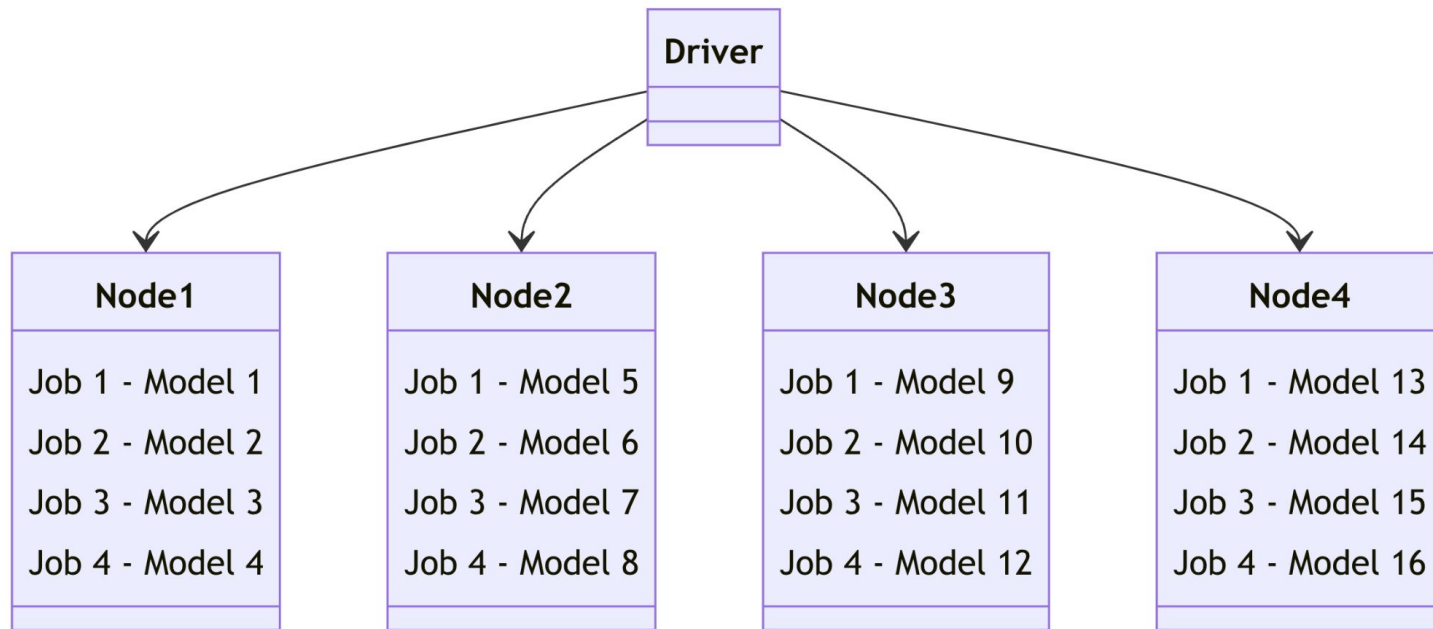
# Grid Tuning



# Cross Validation



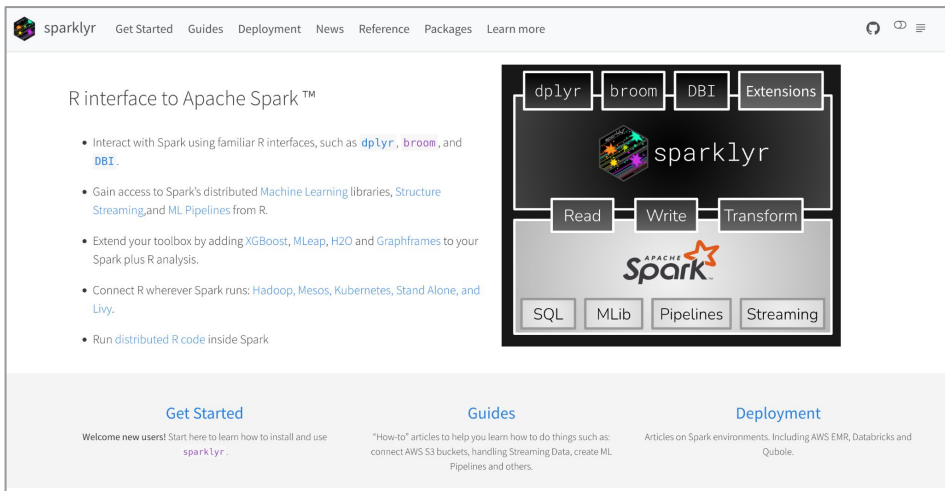
# Spark logistics of tuning



# Learn more...

spark.rstudio.com

therinspark.com



The screenshot shows the sparklyr website. At the top is a navigation bar with links: sparklyr, Get Started, Guides, Deployment, News, Reference, Packages, and Learn more. The main heading is "R interface to Apache Spark™". Below it is a list of bullet points: "Interact with Spark using familiar R interfaces, such as dplyr, broom, and DBI.", "Gain access to Spark's distributed Machine Learning libraries, Structure Streaming, and ML Pipelines from R.", "Extend your toolbox by adding XGBoost, MLeap, H2O and Graphframes to your Spark plus R analysis.", "Connect R wherever Spark runs: Hadoop, Mesos, Kubernetes, Stand Alone, and Livy.", and "Run distributed R code inside Spark". To the right of the text is a diagram showing the sparklyr ecosystem. It includes boxes for dplyr, broom, DBI, and Extensions at the top, connected to a central sparklyr box. Below this are boxes for Read, Write, and Transform, which connect to the Apache Spark logo. At the bottom are boxes for SQL, MLib, Pipelines, and Streaming. At the bottom of the page are three sections: "Get Started" with a "Welcome new users!" message, "Guides" with "How-to" articles, and "Deployment" with articles on Spark environments like AWS EMR, Databricks, and Qubole.

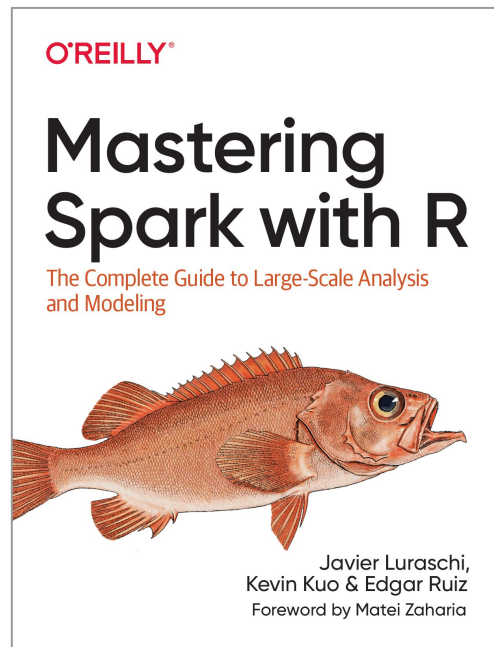
R interface to Apache Spark™

- Interact with Spark using familiar R interfaces, such as [dplyr](#), [broom](#), and [DBI](#).
- Gain access to Spark's distributed [Machine Learning](#) libraries, [Structure Streaming](#), and [ML Pipelines](#) from R.
- Extend your toolbox by adding [XGBoost](#), [MLeap](#), [H2O](#) and [Graphframes](#) to your Spark plus R analysis.
- Connect R wherever Spark runs: [Hadoop](#), [Mesos](#), [Kubernetes](#), [Stand Alone](#), and [Livy](#).
- Run [distributed R code](#) inside Spark

**Get Started**  
Welcome new users! Start here to learn how to install and use sparklyr.

**Guides**  
"How-to" articles to help you learn how to do things such as: connect AWS S3 buckets, handling Streaming Data, create ML Pipelines and others.

**Deployment**  
Articles on Spark environments. Including AWS EMR, Databricks and Qubole.



# sparklyr cheatsheet

## Data Science in Spark with *sparklyr* : : CHEAT SHEET



### Intro

*sparklyr* is an R interface for Apache Spark™. It enables us to write all of our analysis code in R, but have the actual processing happen inside Spark clusters. Easily manipulate and model large-scale using R and Spark via *sparklyr*.

**Import**

- From R (`copy_to()`)
- Read a file (`spark_read()`)
- Read Hive table (`tbl()`)

**Wrangle**

- `dplyr` verb
- `tidyr` commands
- Feature transformer (`ft_*`)
- Direct Spark SQL (`DBI`)

**Visualize**

- Collect result, plot in R

**Model**

- Spark MLlib (`ml_*`)
- H2O Extension

**Communicate**

Collect results into R  
share using RMarkdown

[More R packages](#)  
[Contributors & Sponsors](#)

### Import



Import data into *Spark*, not R

#### READ A FILE INTO SPARK

Arguments that apply to all functions:  
sc, name, path, options=list(), repartition=0,  
memory=TRUE, overwrite=TRUE

CSV	<code>spark_read_csv()</code> (header = TRUE, column=NA, infer_schema=TRUE, delimiter = "\t", quote = "\"", escape = "\\", charset = "UTF-8", null_value = NULL)
JSON	<code>spark_read_json()</code>
PARQUET	<code>spark_read_parquet()</code>
TEXT	<code>spark_read_text()</code>
ORC	<code>spark_read_orc()</code>
LIBSVM	<code>spark_read_libsvm()</code>
DELTA	<code>spark_read_delta()</code>
AVRO	<code>spark_read_avro()</code>

#### R DATA FRAME INTO SPARK

`dplyr::copy_to(dest, df, name)`

Apache Arrow accelerates data transfer between R and Spark. To use, simply load the library

`library(sparklyr)`  
`library(arrow)`

#### FROM A TABLE IN HIVE

`dplyr::tbl(sc, ...)` - Creates a reference to a table without loading it into memory



### Wrangle

#### DPLYR VERBS

Translates into Spark SQL statements

```
copy_to(sc, mtcars) %>%  
mutate(tr = ifelse(am == 0,  
  "auto", "man")) %>%  
group_by(tr) %>%  
summarise_all(mean)
```

#### TIDYR

`pivot_longer()` - Collapse several columns into two.

`pivot_wider()` - Expand two columns into several.

`nest()` / `unnest()` - Convert groups of cells into list-columns, and vice versa.

`unite()` / `separate()` - Split a single column into several columns, and vice versa.

`fill()` - Fill NA with the previous value

#### 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 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 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_robust_scaler()` - Removes the median and scales according to standard scale.

`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

### Visualize



#### DPLYR + GGPLYT2

```
copy_to(sc, mtcars) %>%  
group_by(cyl) %>%  
summarise(mpg_m = mean(mpg)) %>%  
collect() %>%  
ggplot() +  
geom_col(aes(cyl, mpg_m))
```

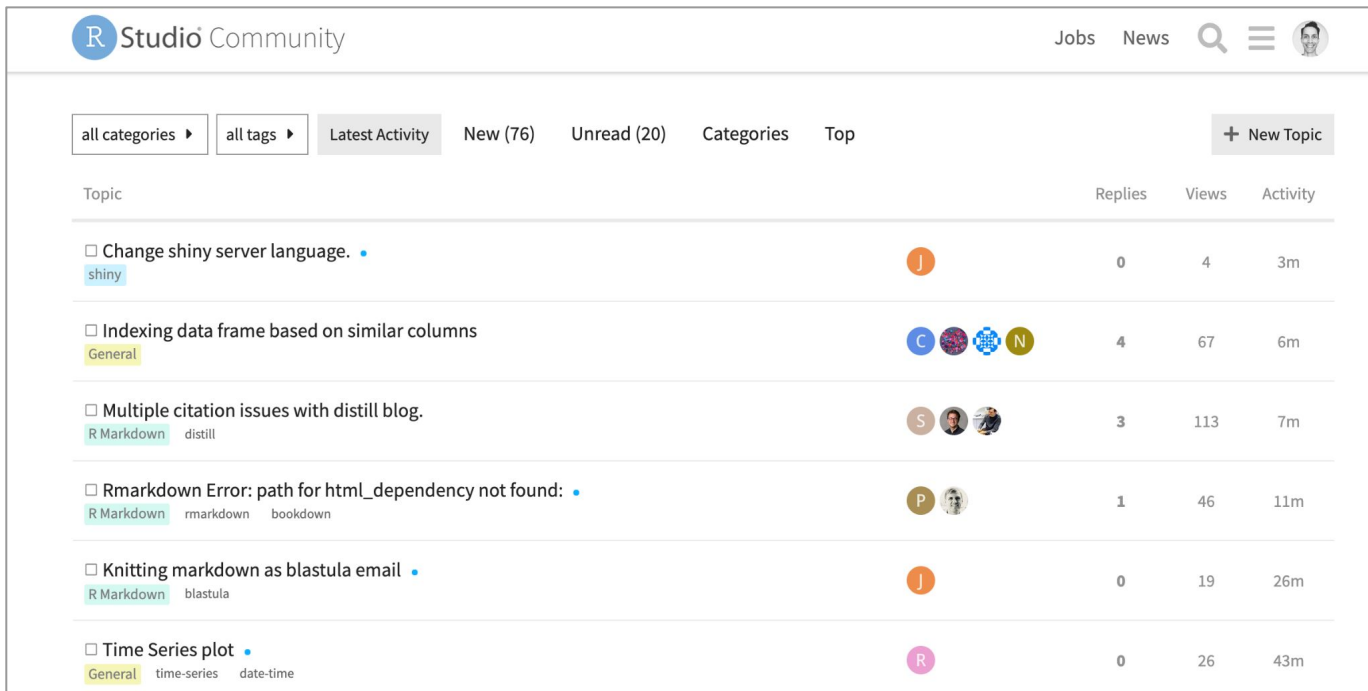
Summarize in Spark

Collect results in R

Create plot

# Join the community...

community.rstudio.com



The screenshot shows the R Studio Community website interface. At the top, there's a navigation bar with the R Studio logo, 'Community' text, and links for 'Jobs', 'News', a search icon, a menu icon, and a user profile icon. Below the navigation bar, there's a filter section with 'all categories' and 'all tags' dropdowns, followed by tabs for 'Latest Activity', 'New (76)', 'Unread (20)', 'Categories', and 'Top'. A '+ New Topic' button is on the right. The main content area displays a list of topics with columns for 'Topic', 'Replies', 'Views', and 'Activity'. Each topic entry includes a checkbox, a title, tags, a user profile picture, and numerical data for replies, views, and activity time.

Topic	Replies	Views	Activity
<input type="checkbox"/> Change shiny server language. <span>•</span> <a href="#">shiny</a>	0	4	3m
<input type="checkbox"/> Indexing data frame based on similar columns <a href="#">General</a>	4	67	6m
<input type="checkbox"/> Multiple citation issues with distill blog. <a href="#">R Markdown</a> <a href="#">distill</a>	3	113	7m
<input type="checkbox"/> Rmarkdown Error: path for html_dependency not found: <span>•</span> <a href="#">R Markdown</a> <a href="#">rmarkdown</a> <a href="#">bookdown</a>	1	46	11m
<input type="checkbox"/> Knitting markdown as blastula email <span>•</span> <a href="#">R Markdown</a> <a href="#">blastula</a>	0	19	26m
<input type="checkbox"/> Time Series plot <span>•</span> <a href="#">General</a> <a href="#">time-series</a> <a href="#">date-time</a>	0	26	43m



# Thank you!

<https://github.com/edgararuiz/talks/tree/main/sparklyr-intro>