

Getting Started with R and Spark

Edgar Ruiz

github.com/edgararuiz

twitter.com/theotheredgar

linkedin.com/in/edgararuiz

Data Science Workflow



Begin

End

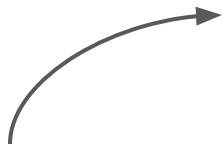
Import



Tidy



Transform



Visualize



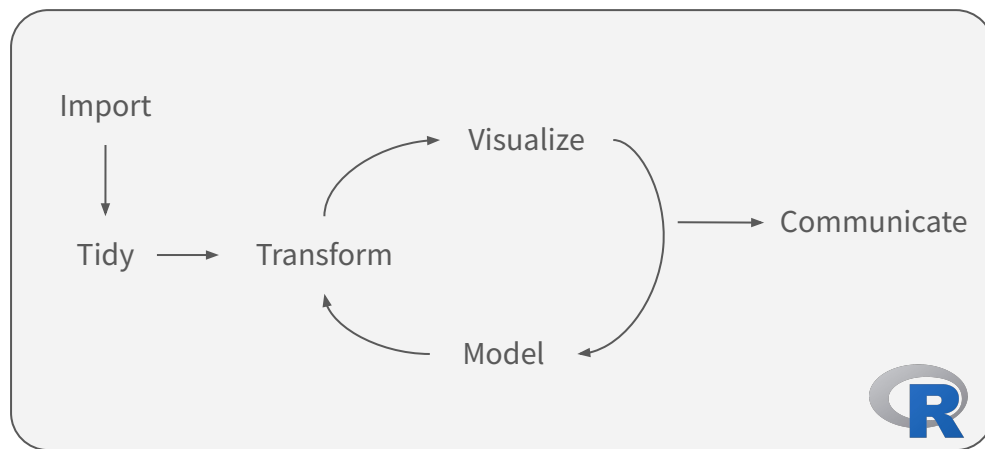
Model



Communicate

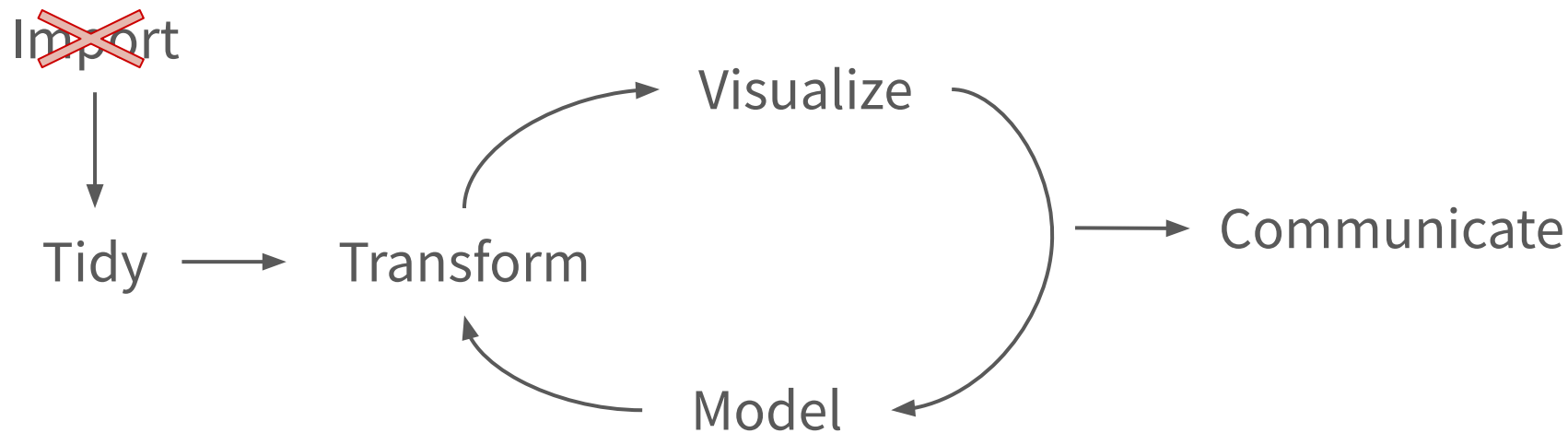
Understand

A single laptop handles “small data” easy



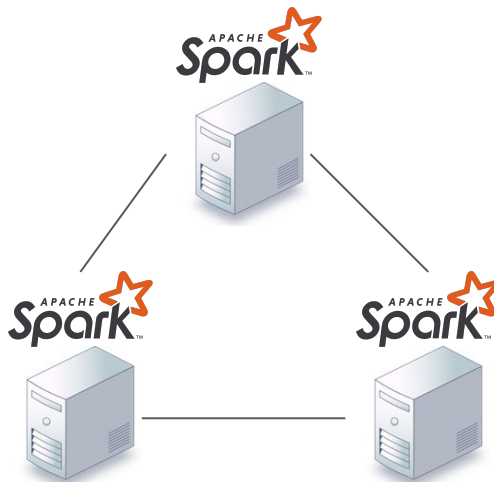
My laptop

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

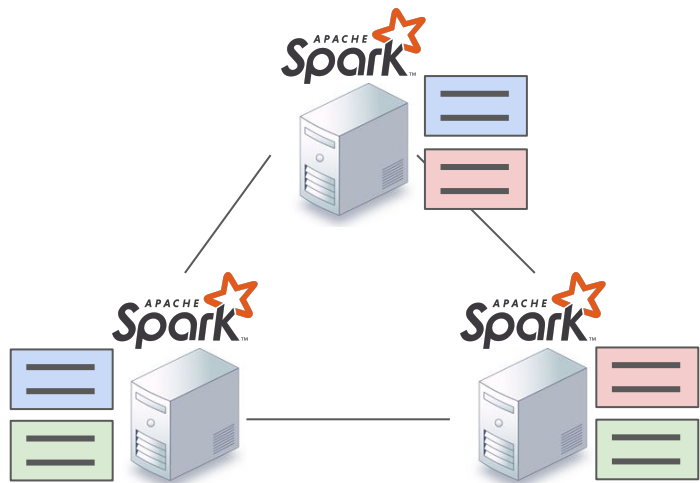
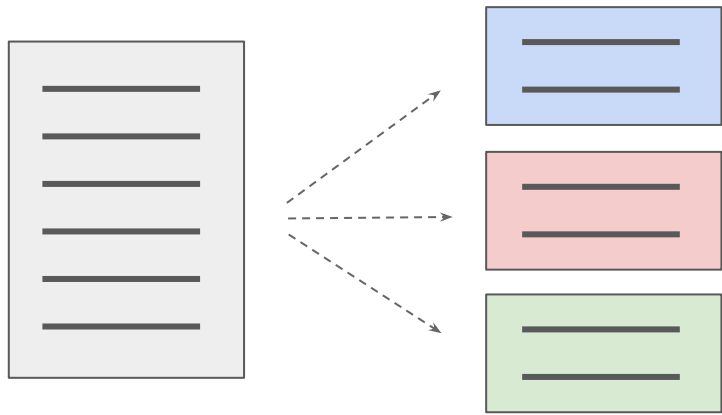


How does Spark work?

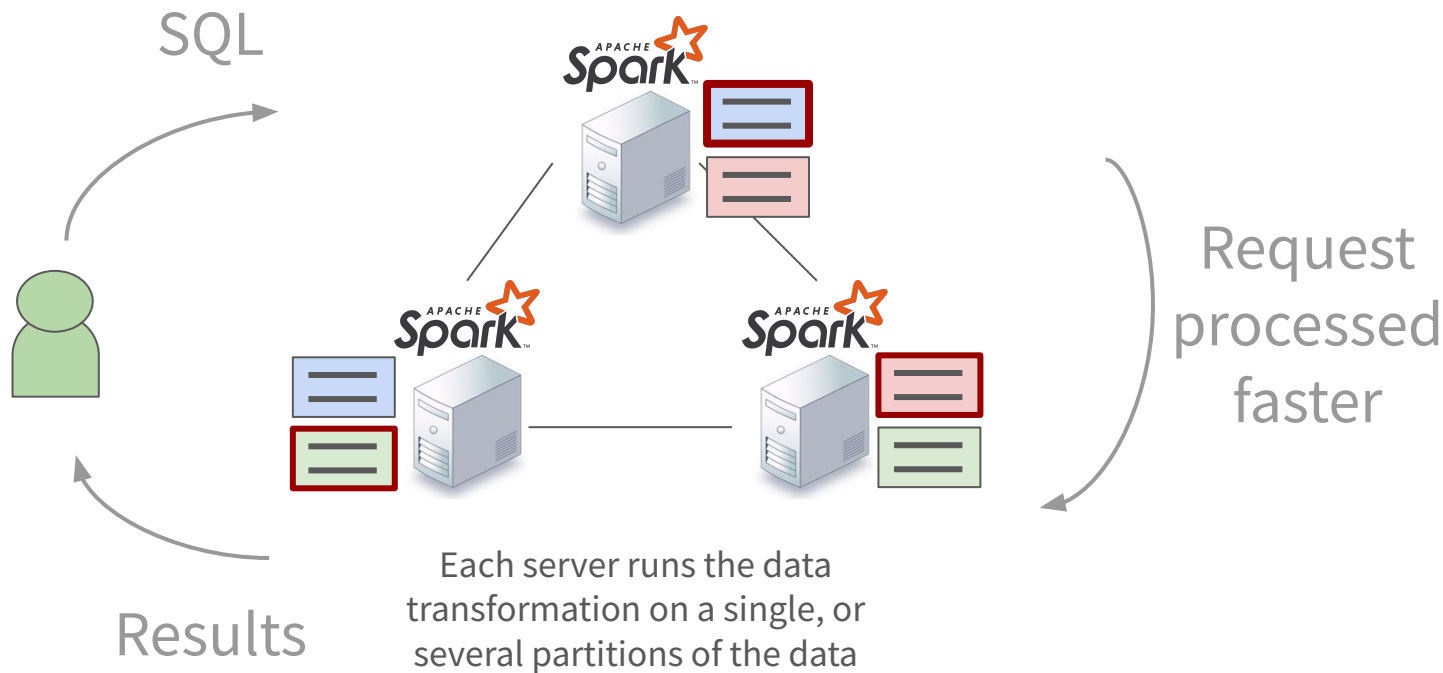
Original large
data file

File is partitioned
logically

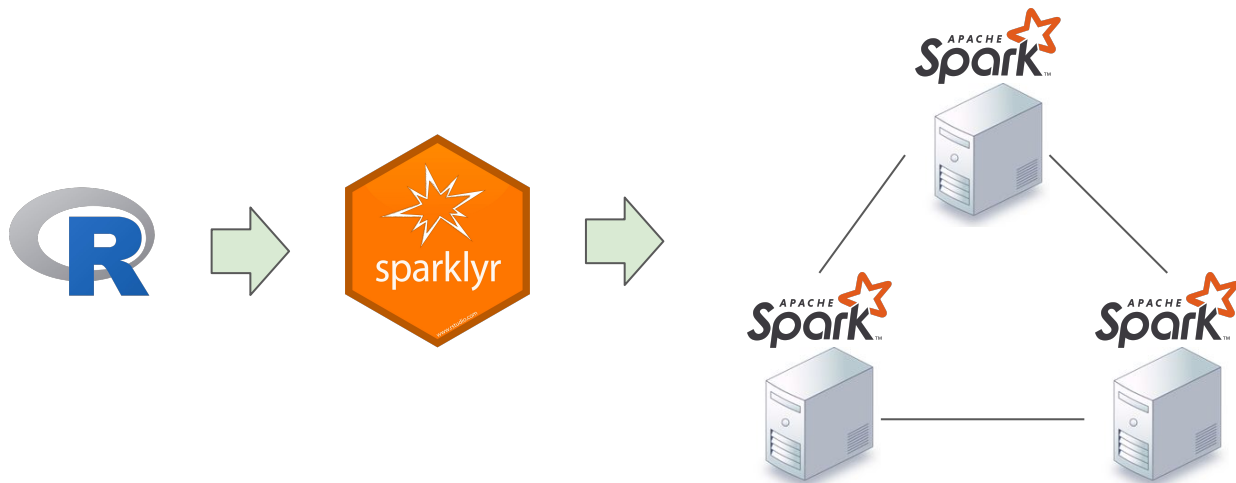
One or several partitions are sent to
each server's **memory** in the cluster



How does Spark work?



How do I interact with Spark?



Data Science with R, and Spark



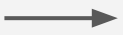
Begin

End

Import 



Tidy 



Transform 




Visualize

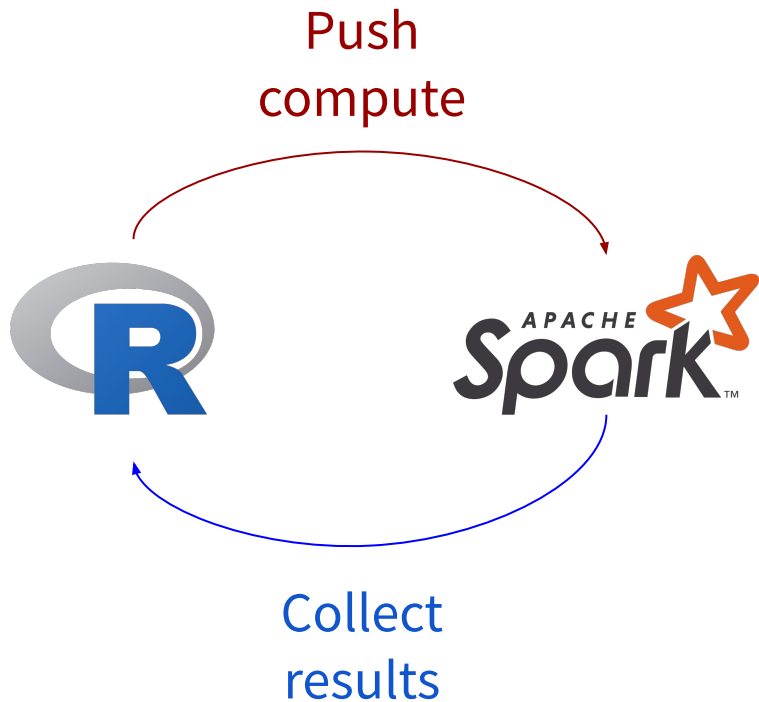


Model 



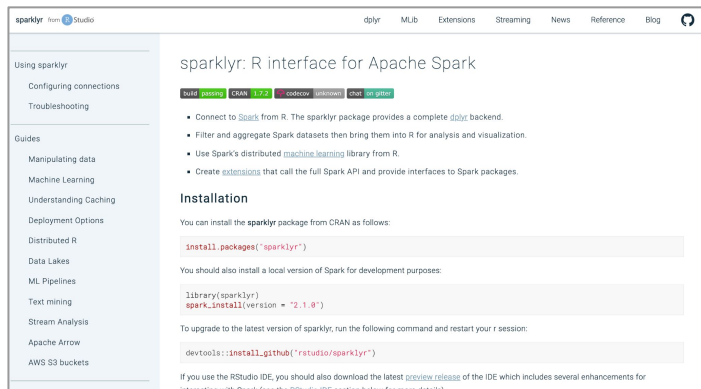
Communicate 

Main takeaway when working with R and Spark

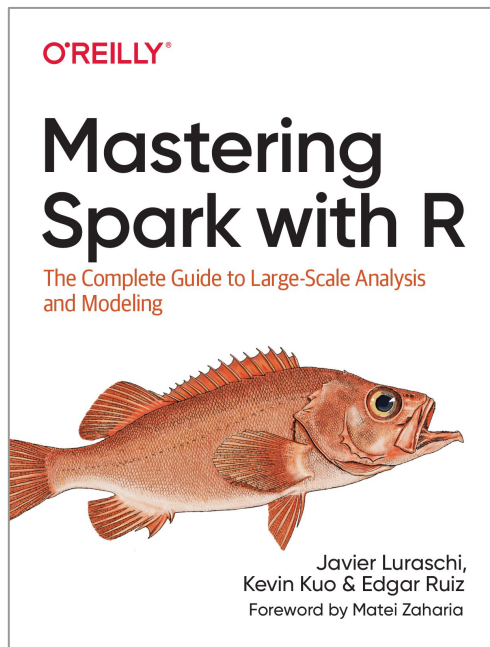


Learn more...

spark.rstudio.com



therinspark.com



sparklyr cheatsheet

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



Intro

sparklyr is an R interface for Apache Spark™. *sparklyr* 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
- Feature transformer (`ft_*`)
- Direct Spark SQL (DBI)

Visualize

- Collect result, plot in R
- Use `dbplot`

Model

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

Communicate

- Collect results into R share using `markdown`

[R for Data Science, Grolemund & Wickham](#)

Import



READ A FILE INTO SPARK

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

CSV

```
spark_read_csv(header = TRUE,
  columns=NULL,
  infer_schema=TRUE, delimiter=";",
  quote="\"", escape="\\", charset =
  "UTF-8", null_value = NULL)
```

JSON

```
spark_read_json()
```

PARQUET

```
spark_read_parquet()
```

TEXT

```
spark_read_text()
```

HIVE TABLE

```
spark_read_table()
```

ORC

```
spark_read_orc()
```

LIBSVM

```
spark_read_libsvm()
```

JDBC

```
spark_read_jdbc()
```

DELTA

```
spark_read_delta()
```

R DATA FRAME INTO SPARK

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

FROM A TABLE IN HIVE

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

Wrangle

DPLYR VERBS

Translates into Spark SQL statements

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

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() - 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
```

Visualize



DPLYR + GGLOT2

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

DBPLOT

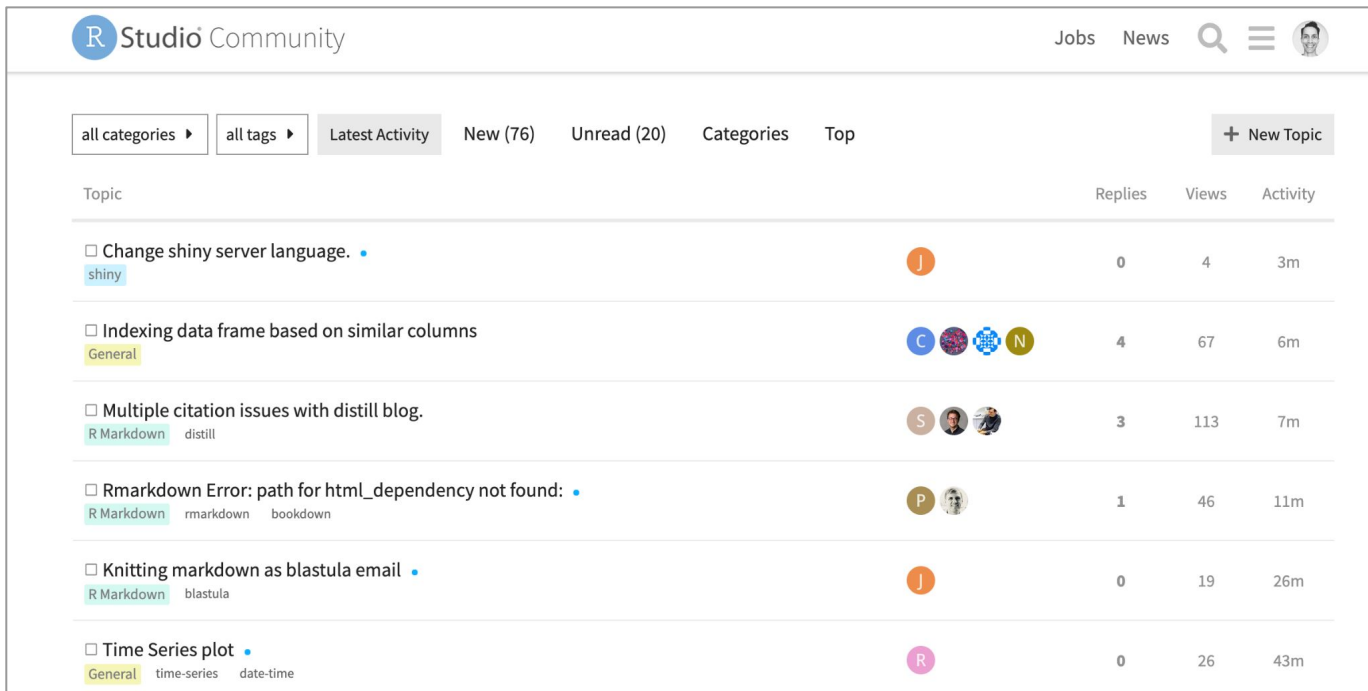
```
copy_to(sc, mtcars) %>%
dbplot_histogram(mpg) +
labs(title = "Histogram of MPG")

dbplot_histogram(data, x, bins = 30, binwidth = NULL) -
Calculates the histogram bins in Spark and plots in ggplot2
dbplot_raster(data, x, y, fill = n0, resolution = 100,
complete = FALSE) - Visualize 2 continuous variables, use
instead of geom_point()
```



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

Thank you!

Edgar Ruiz

github.com/edgararuiz

twitter.com/theotheredgar

linkedin.com/in/edgararuiz