

Lenguajes y APIs en Spark

2021

Contenidos

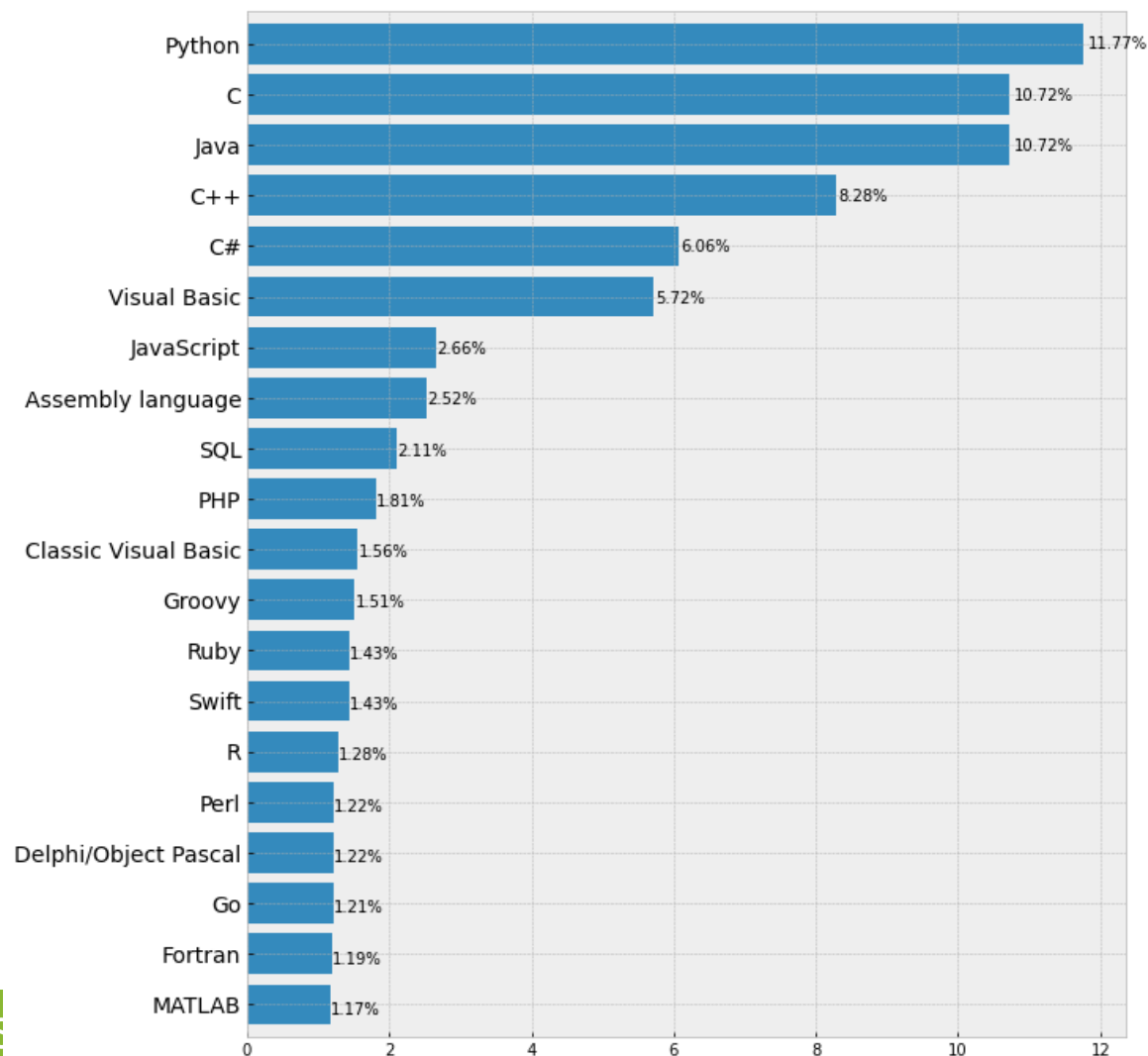
- Introducción
- Spark en Java
- Spark en Scala
 - Introducción a Scala
 - API de Scala en Spark
- Spark en R
 - SparkR
 - sparklyr

Introducción

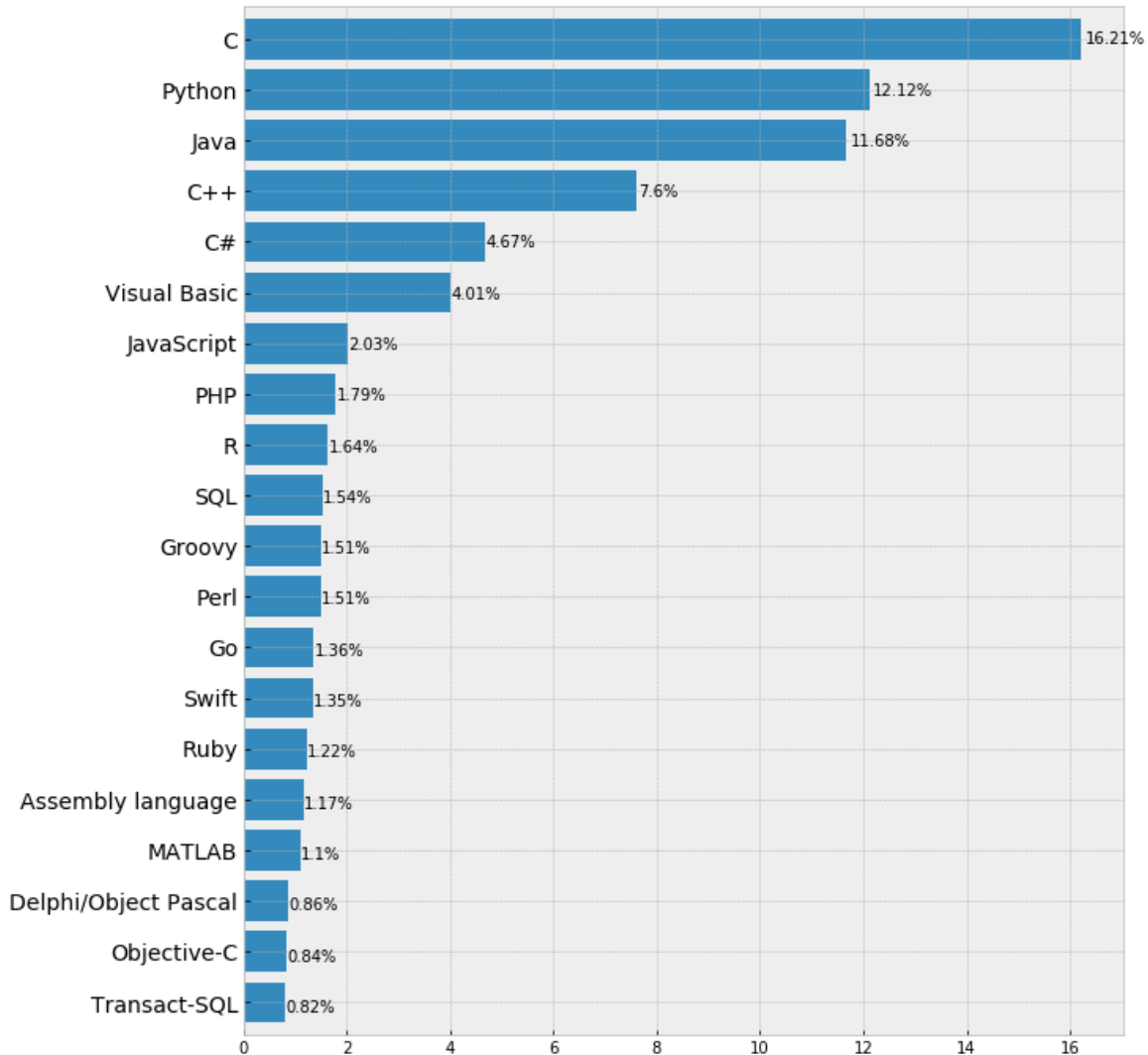
Algunas estadísticas (no concluyentes) sobre lenguajes de programación

Número de resultados
de la consulta
+"xxx programming"
en 25 buscadores

TIOBE Index, noviembre 2021



TIOBE Index, noviembre 2020



Tamaño de las comunidades de desarrolladores

Fuente: [slashdata developer nation](https://slashdata.com/developer-nation)

Encuesta a desarrolladores

Size of programming language communities in Q3 2021

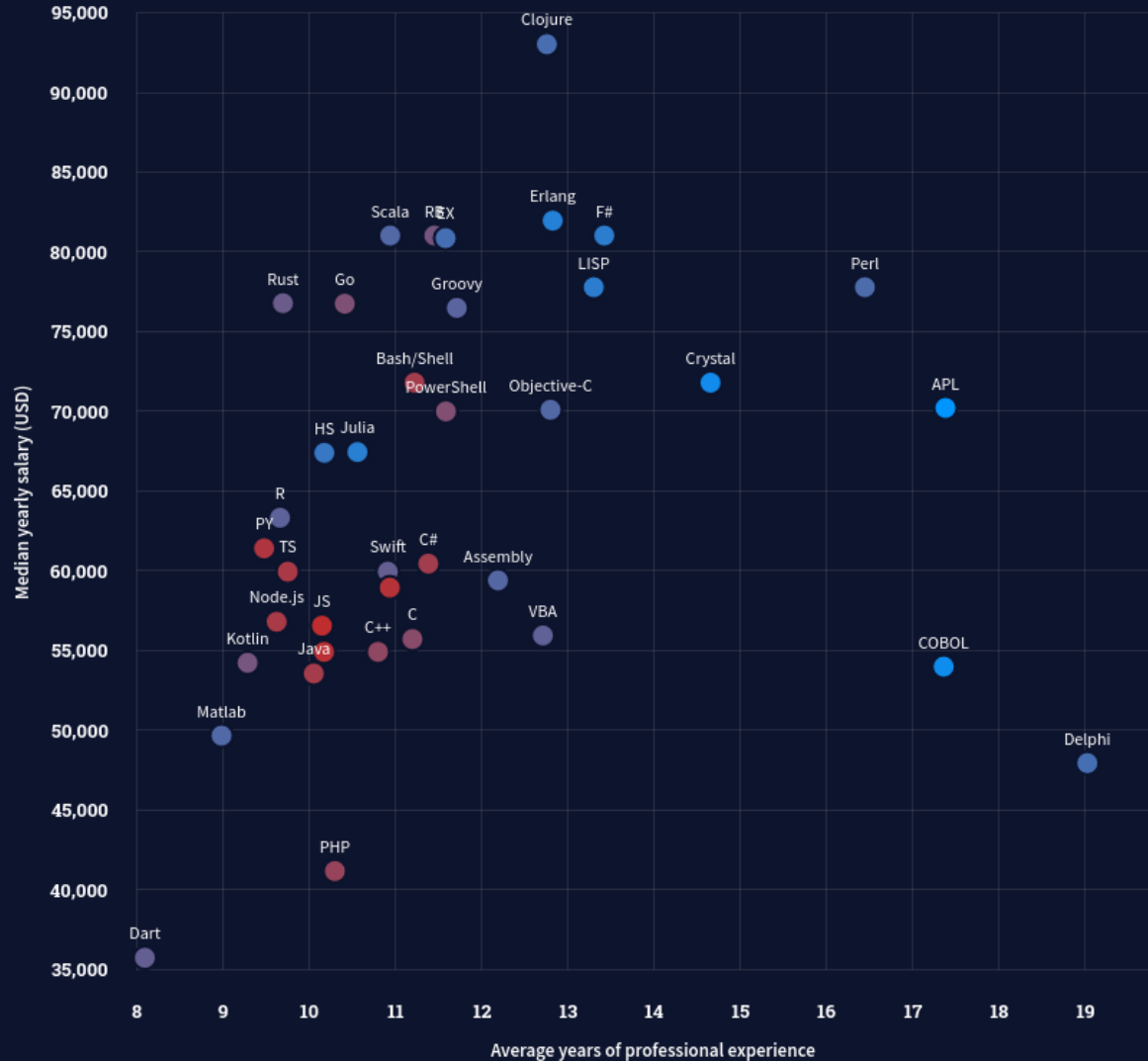
Active software developers, globally, in millions (n=12,506)

		Most popular in	Least popular in
Javascript*	16.4 M	Web, backend	DS/ML, embedded
Python	11.3 M	DS/ML, IoT apps	Mobile, AR/VR
Java	9.6 M	Mobile, desktop	DS/ML, web
C/C++	7.5 M	Embedded, IoT apps	Web, mobile
PHP	7.3 M	Web, backend	DS/ML, mobile
C#	7.1 M	AR/VR, desktop, games	DS/ML, mobile
Visual development tools	3.6 M	Desktop, AR/VR	Cloud, web
Kotlin	2.9 M	Mobile, AR/VR	DS/ML, desktop
Swift	2.5 M	Mobile, AR/VR	Backend, desktop
Go	2.0 M	Backend, apps for 3rd-party ecosystems	Games, web
Dart	1.4 M	Mobile	Web
Objective C	1.4 M	AR/VR	Desktop, games
Ruby	1.4 M	IoT, backend	DS/ML, web
Rust	1.1 M	AR/VR, embedded	Mobile, web
Lua	0.8 M	AR/VR, IoT, games	Mobile, desktop

25 Most In-Demand Data Science Skills in 2021



StackOverflow developer survey, 2021



¿Y en Spark?

Spark permite desarrollar aplicaciones en

- Scala
- Java
- Python
- R
- ... y SQL

¿Y en Spark?

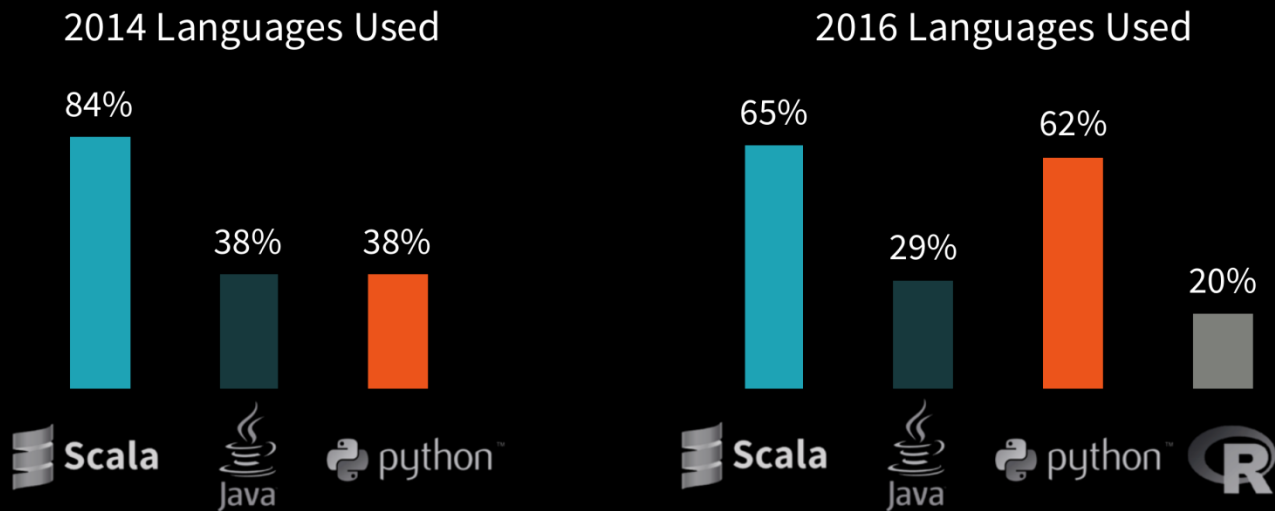
Spark permite desarrollar aplicaciones en

- Scala
- Java
- Python
- R
- ... y SQL
- Más exóticos: [Julia](#), [C#](#), o procesos externos mediante [rdd.pipe\(\)](#)

¿Cuánta gente lo usa?

Matei Zaharia, *What to Expect for Big Data and Apache Spark in 2017*, Spark Summit East 2017

Languages Used for Spark

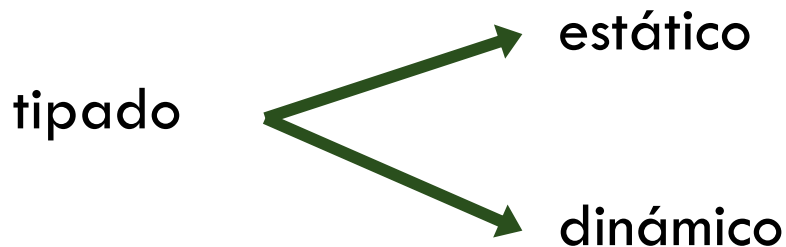
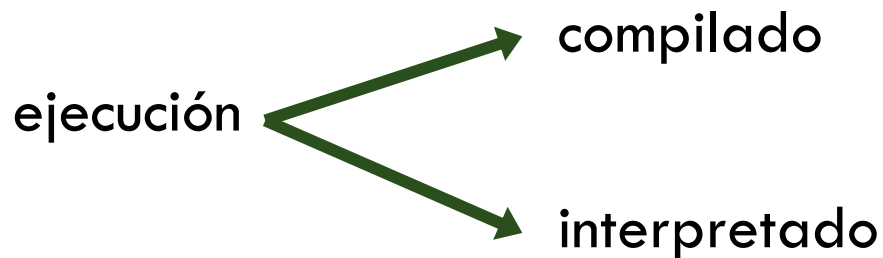


+ Everyone uses SQL (95%)

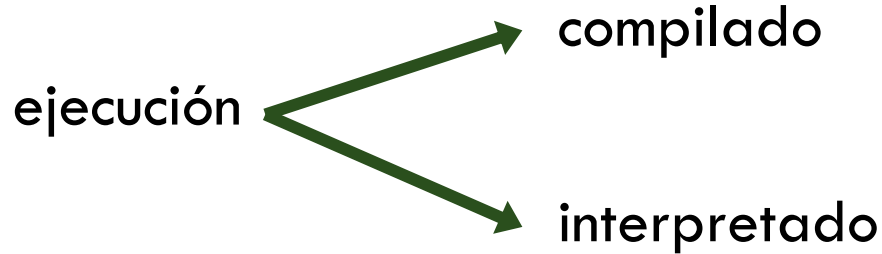
databricks

Clasificación de lenguajes de programación

Lenguajes de programación



Lenguajes de programación

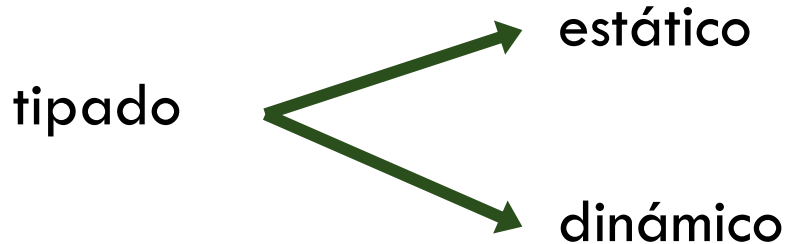


Java

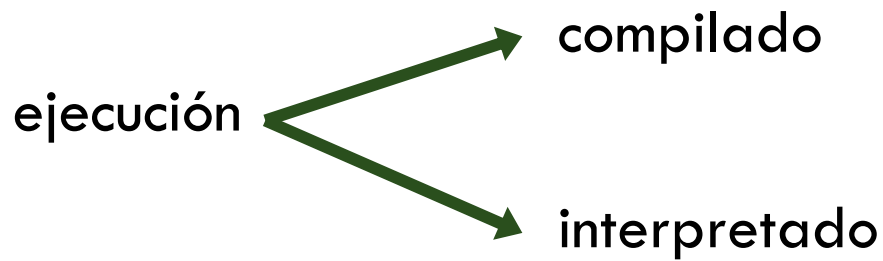
Scala

Python

R



Lenguajes de programación

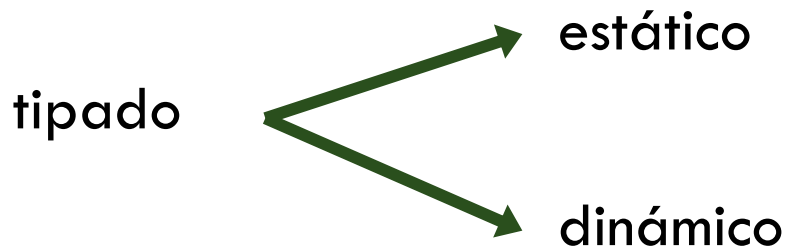


Java

Scala

Python

R



Java

Scala

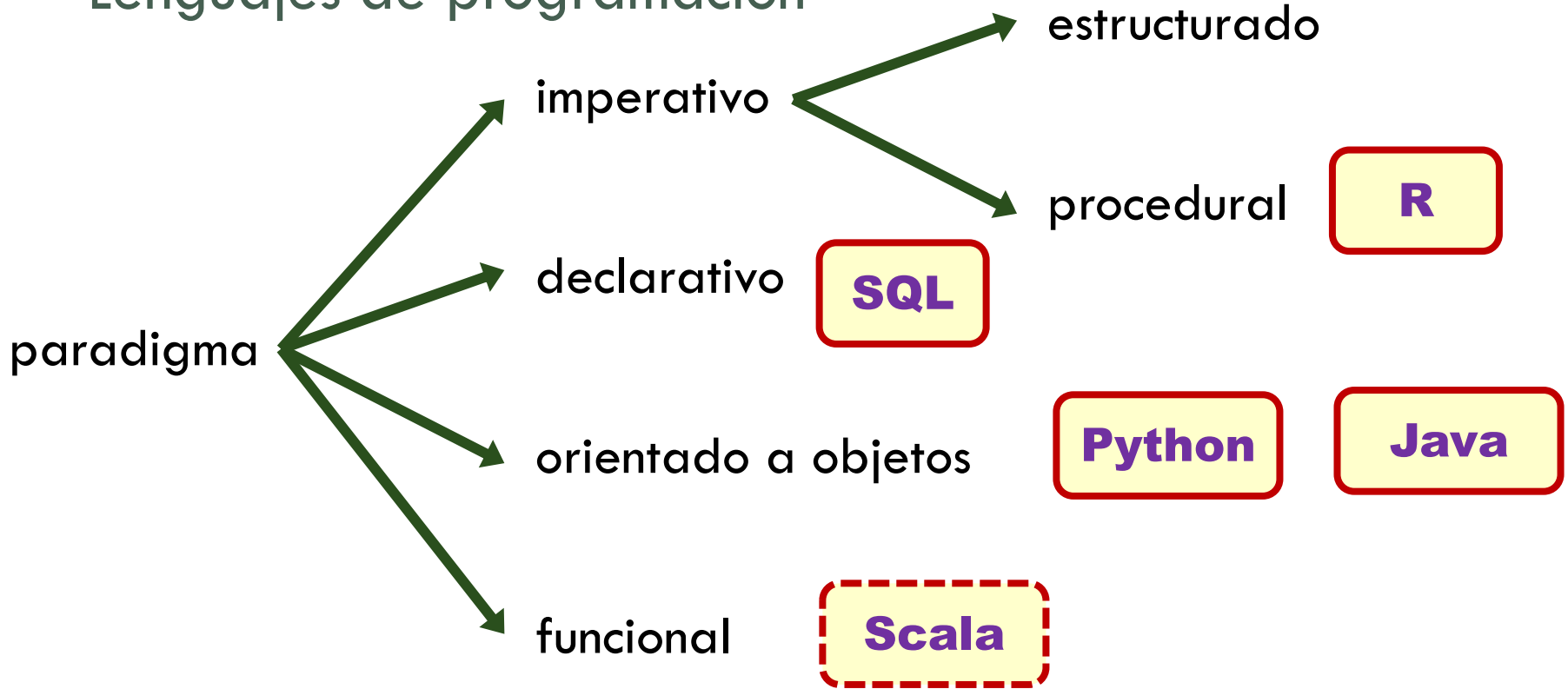
Python

R

Lenguajes de programación



Lenguajes de programación



Recapitulación

Contador de palabras en Python

```
lines = sc.textFile ("hdfs://user/data/example.txt",4)
words = lines.flatMap( lambda l : l.strip().split() )
wordc = words.map( lambda w : (w,1) )
counts = wordc.reduceByKey( lambda a,b : a + b )
result = counts.collect()
```

Spark & Java

- Características de Java
 - Compilado
 - Tipado estático
 - Las características del lenguaje Java hacen que el desarrollo tienda a exigir una gran cantidad de código de relleno (*boilerplate*)
 - problema aliviado a partir de Java 8 (interfaces funcionales, lambda)
- Java en Spark
 - [Documentación](#)
 - [Especificación](#)

➤ Características de Java

- Compilado

- Tipado estático

- Las características del lenguaje Java hacen que el desarrollo tienda a exigir una gran cantidad de código de relleno (*boilerplate*)

 - problema aliviado a partir de Java 8 (interfaces funcionales, lambda)

➤ Java en Spark

- [Documentación](#)

- [Especificación](#)

Spark \geq 2.3 requiere Java 8
Spark \geq 3.0 admite Java 8 & Java 11

Cómo ejecutar una aplicación Java para Spark

1. Conseguir una herramienta de gestión de proyectos

Usaremos Maven (<https://maven.apache.org/>)

Para instalar en la máquina virtual, `vagrant provision --provision-with mvn`

2. Crear un fichero de descripción, indicando las dependencias

Para Maven, un **POM** (*Project Object Model*)

3. Crear la aplicación Java (compilar y empaquetar)

```
mvn package
```

4. Enviar la aplicación al servidor Spark

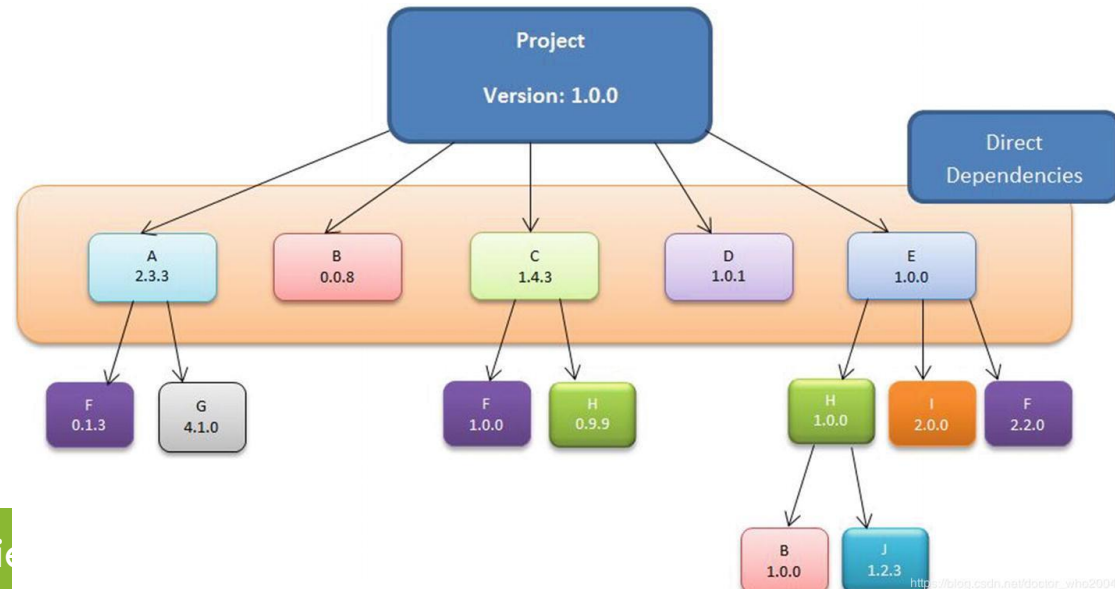
```
spark-submit [--class <entrypoint>] <jarfile> -- <param>
```

p. ej.

```
spark-submit --class main.JavaWordCount \  
target/spark-word-count-1.0.jar ../../DATA/DonQuijote.txt.bz2
```


Resolución de dependencias

- El fichero de proyecto `pom.xml` define las dependencias directas del proyecto
- Esas dependencias pueden a su vez generar otras *dependencias transitivas*
- Maven resuelve la cadena de dependencias (*dependency mediation*) con dos criterios:
 - *Nearest first*
 - *First found*



spark-submit

La expresión general de la ejecución de una tarea Spark es:

```
spark-submit [spark-options] <jarfile> [program options]
```



```
--class <entrypoint>  
punto de entrada a la ejecución
```

Mini-introducción a Scala

Tipología general

- Scala = “*Scalable language*”
- Lenguaje compilado
 - Comprobación estricta de tipos
- Basado y compatible con Java (Java 8 & Java 11))
- Se ejecuta en la JVM
- REPL/scripting
- <https://www.scala-lang.org/documentation/>

El lenguaje

➤ Rasgos principales

➤ Orientado a objetos

➤ Funcional

➤ Interoperable con Java

➤ *las funciones son objetos*

➤ Rasgos especiales

➤ Preferencia hacia la inmutabilidad

➤ Sintaxis simplificada

➤ Programación concurrente: primitivas para sincronización

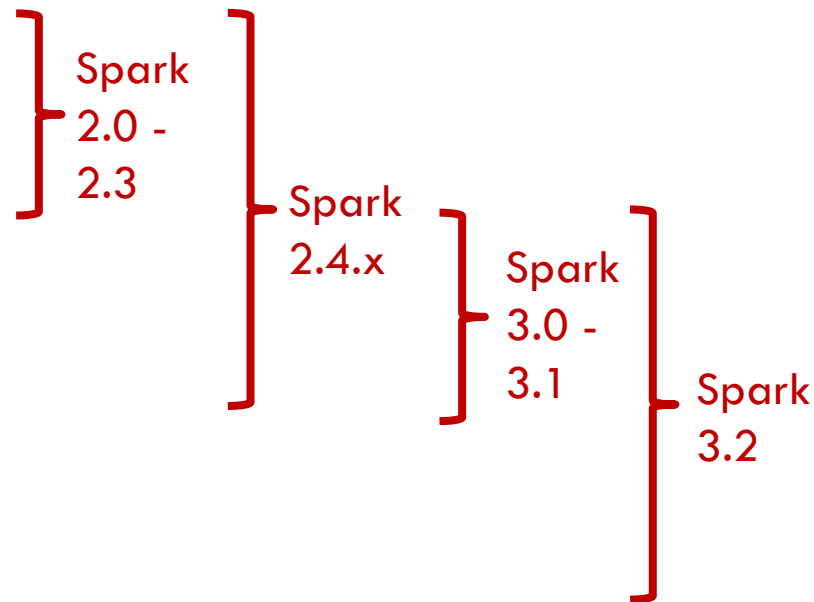
➤ Programación distribuida

Versiones de Scala

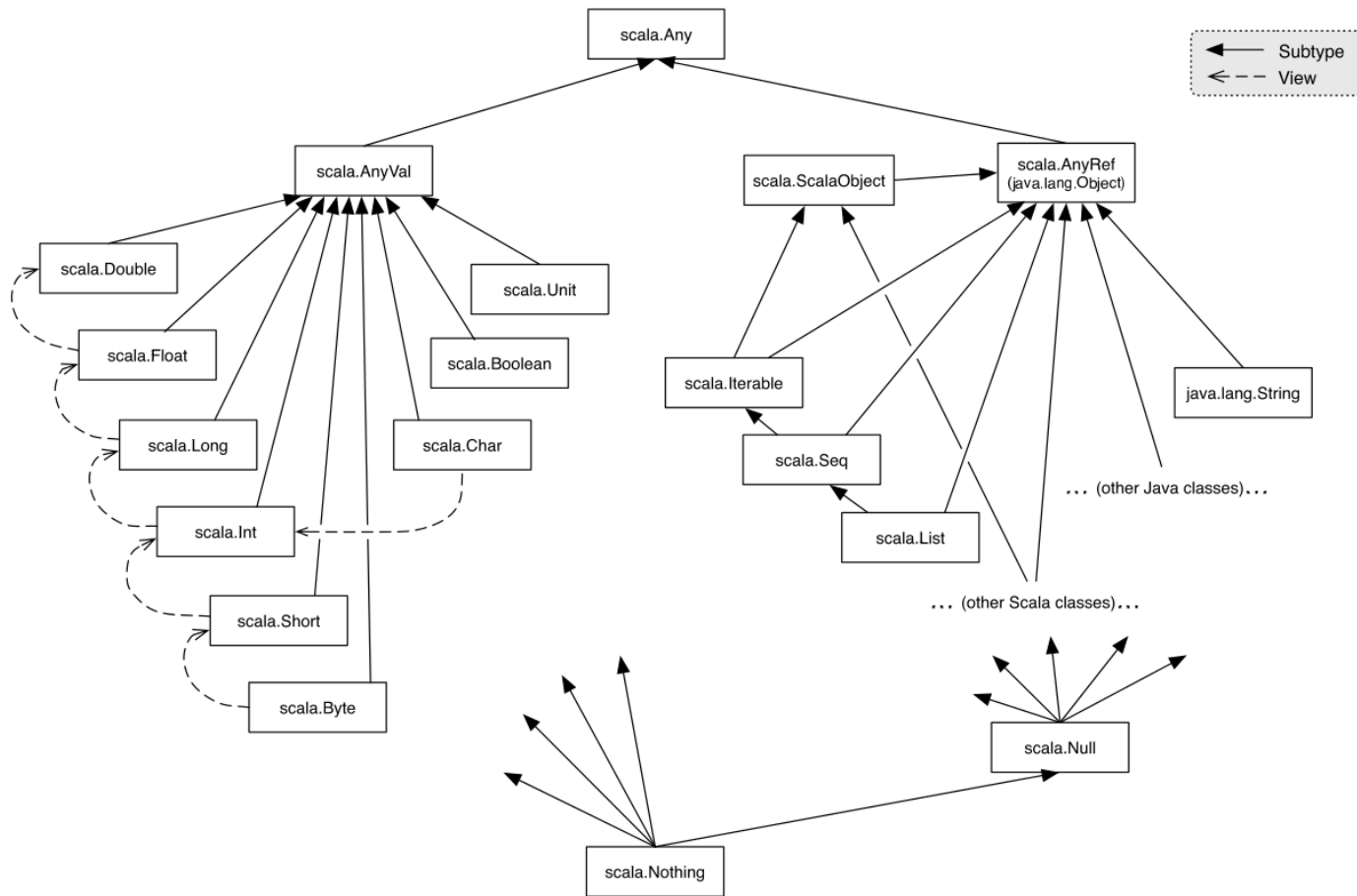
- Scala 2.11
 - abril 2014
 - Última versión: 2.11.12 (noviembre 2017)
- Scala 2.12
 - noviembre 2016
 - Última versión: 2.12.15 (septiembre 2021)
- Scala 2.13
 - Junio 2019
 - Última versión: 2.13.7 (noviembre 2021)
- Scala 3.0
 - Mayo 2021
 - Última versión: 3.1.0 (octubre 2021)

Versiones de Scala

- Scala 2.11
 - abril 2014
 - Última versión: 2.11.12 (noviembre 2017)
- Scala 2.12
 - noviembre 2016
 - Última versión: 2.12.15 (septiembre 2021)
- Scala 2.13
 - Junio 2019
 - Última versión: 2.13.7 (noviembre 2021)
- Scala 3.0
 - Mayo 2021
 - Última versión: 3.1.0 (octubre 2021)



Scala es un lenguaje fuertemente tipado



Tipos de datos

- **Simples:** `Int`, `Double`, `Boolean`, `Char`, etc
- **Referencias:** colecciones, clases
- **Especiales:** `Unit`, `Nothing`, `Null`, `Nil`, `None`, `Any`

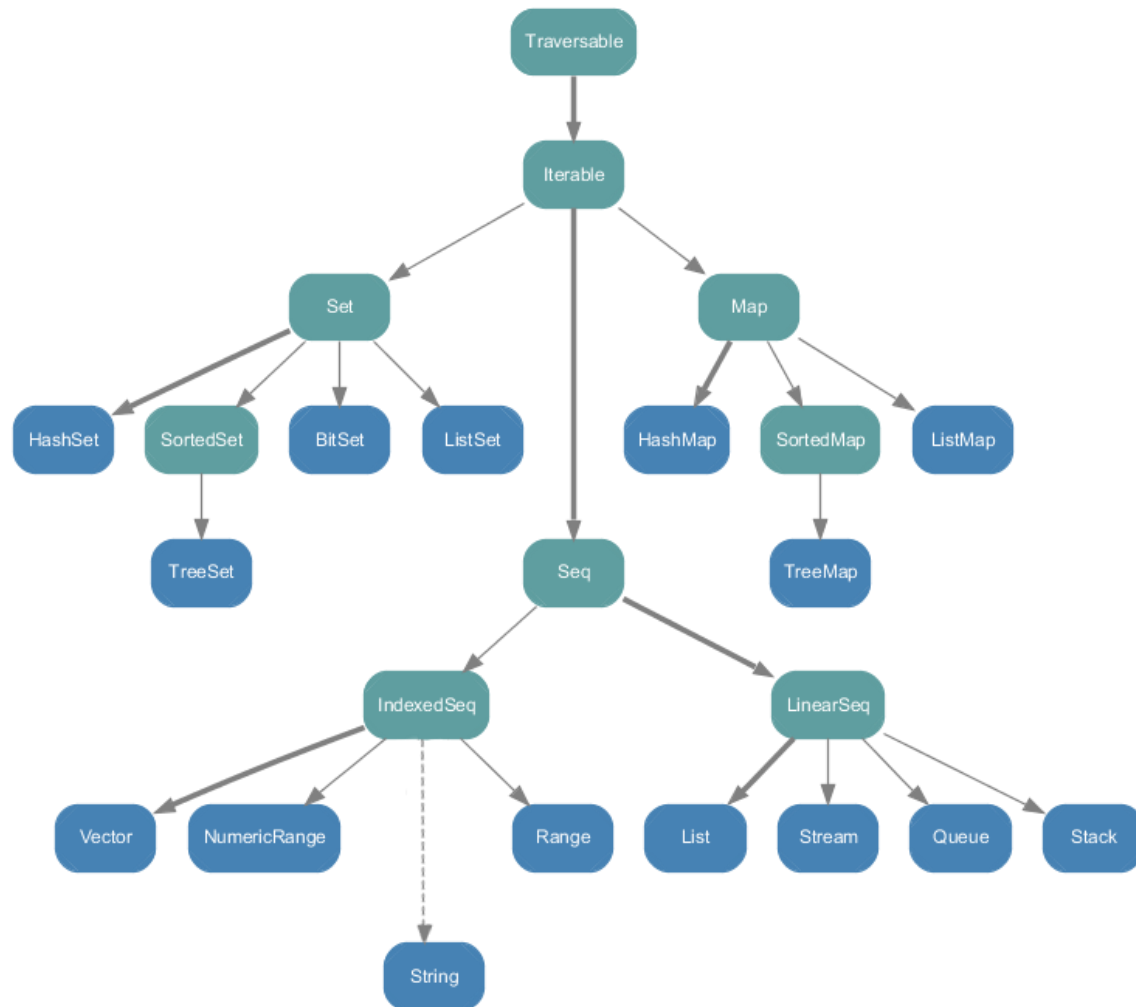
Funciones

- Las funciones son uno de los elementos principales del lenguaje.
- Son objetos de primer orden: pueden asignarse a variables y pasarse libremente
- Scala permite variantes especiales:
 - Funciones anónimas (*function literals*)
 - Funciones de orden superior (*higher level functions*)

Clases y objetos

- Las clases de Scala son muy parecidas a Java. Tienen métodos y campos
 - Se instancian con **new**
 - A diferencia de Java, no tienen un método constructor específico
 - Un objeto de una clase de Scala que define el método **apply** puede llamarse como una función (similar al método `__call__` en Python)
- Las interfaces (especificación de clases en Java) en Scala se realizan mediante **Traits**
- Un tipo especial de clase Scala son las *case classes*

Colecciones



- Las colecciones son una parte oficial del lenguaje
- Hay colecciones mutables e inmutables
- Las superclases de `scala.collections` definen operaciones abstractas genéricas
 - Por ejemplo, `Traversable` define `foreach`

Programación concurrente

- Scala posee funcionalidades orientadas a facilitar la programación concurrente
- La librería base soporta el uso de *Futures & Promises*
- A más alto nivel, paralelismo basado en paso de mensajes y *actores* se proporciona mediante akka

Spark & Scala

¿Qué relación hay entre Spark y Scala?

- Spark está desarrollado primariamente en Scala
 - Las aplicaciones en Scala corren de manera nativa en Spark
 - Java también, Python y R lo hacen por medio de *gateways*
- Las APIs de Spark se desarrollan inicialmente en Scala
 - Más tarde son portadas a otros lenguajes
 - El ritmo de portado es desigual
- Las aplicaciones tienden a ejecutarse más rápido en Scala
 - Compilado vs. interpretado
 - Nativo vs. conectado
 - El procesado por *DataFrames* atenúa esta desigualdad

APIs de Scala - I

- **RDD API**
- La de más bajo nivel
- Basada en RDDs: datos distribuidos inmutables
 - [Documentación](#)
 - [Especificación](#)
- Objeto de gestión: **SparkContext**
- Objeto de datos: **RDD**

APIs de Scala - II

- **DataFrame API (SparkSQL)**
- Basada en tablas
- Estandariza las operaciones posibles
 - [Documentación](#)
 - [Especificación](#)
- Objeto de gestión: **SparkContext/HiveContext**
- Objeto de datos: **DataFrame**

APIs de Scala - III

- **DataSet API (SparkSQL)**
- Combinación de DataFrames y RDDs
- Disponible a partir de Spark 2.0
- Combina el rendimiento de DataFrames con la flexibilidad de RDDs
 - [Documentación](#)
 - [Especificación](#)
- Objeto de gestión: **SparkSession**
- Objeto de datos: **DataSet**
 - `DataFrame == DataSet<Row>`

Ejecución de código Scala en Spark

1. REPL: la shell de Spark en Scala

`spark-shell`

2. Notebooks: [Toree](#), [SPylon](#), [Almond](#)

3. Aplicaciones compiladas: `spark-submit`

- a) Herramienta de gestión de proyectos ([sbt](#), Maven también es posible)
- b) Fichero de definición de proyecto (`build.sbt`)
- c) Empaquetado: `sbt package`
- d) Enviado: `spark-submit`
- e) Si incluye dependencias adicionales, una de estas dos:
 - En empaquetado, `sbt assembly` ([sbt-assembly](#)) para construir un “*fat jar*”
 - En ejecución, usar opciones de `spark-submit`: `--jars --packages`

Ejecución de código Scala en Spark

1. REPL: la shell de Spark en Scala

`spark-shell`

2. Notebooks: [Toree](#), [SPylon](#), [Almond](#)

3. Aplicaciones compiladas: `spark-submit`

- a) Herramienta de gestión de proyectos ([sbt](#), Maven también es posible)
- b) Fichero de definición de proyecto (`build.sbt`)
- c) Empaquetado: `sbt package`
- d) Enviado: `spark-submit`
- e) Si incluye dependencias adicionales, una de estas dos:
 - En empaquetado, `sbt assembly` ([sbt-assembly](#)) para construir un “*fat jar*”
 - En ejecución, usar opciones de `spark-submit`: `--jars --packages`

Para instalar en la máquina virtual,
`vagrant provision`
`--provision-with scala`

Spark & R

SparkR

- SparkR es el API oficial de Spark para procesado en R
- No proporciona APIs para RDDs
- Está basado en DataFrames (Spark SQL)
 - Mapea DataFrames de Spark a DataFrames de R
 - Usa una interfaz similar a [dplyr](#)

SparkR provides a distributed data frame implementation that supports operations like selection, filtering, aggregation etc. (similar to R data frames, dplyr) but on large datasets. SparkR also supports distributed machine learning using MLlib.

SparkR: Scaling R Programs with Spark, Shivaram Venkataraman, Zongheng Yang, Davies Liu, Eric Liang, Hossein Falaki, Xiangrui Meng, Reynold Xin, Ali Ghodsi, Michael

SparkR: Scaling R Programs with Spark

Shivaram Venkataraman¹, Zongheng Yang¹, Davies Liu², Eric Liang², Hossein Falaki²
Xiangrui Meng², Reynold Xin², Ali Ghodsi², Michael Franklin¹, Ion Stoica^{1,2}, Matei Zaharia^{2,3}
¹AMPLab UC Berkeley, ²Databricks Inc., ³MIT CSAIL

ABSTRACT

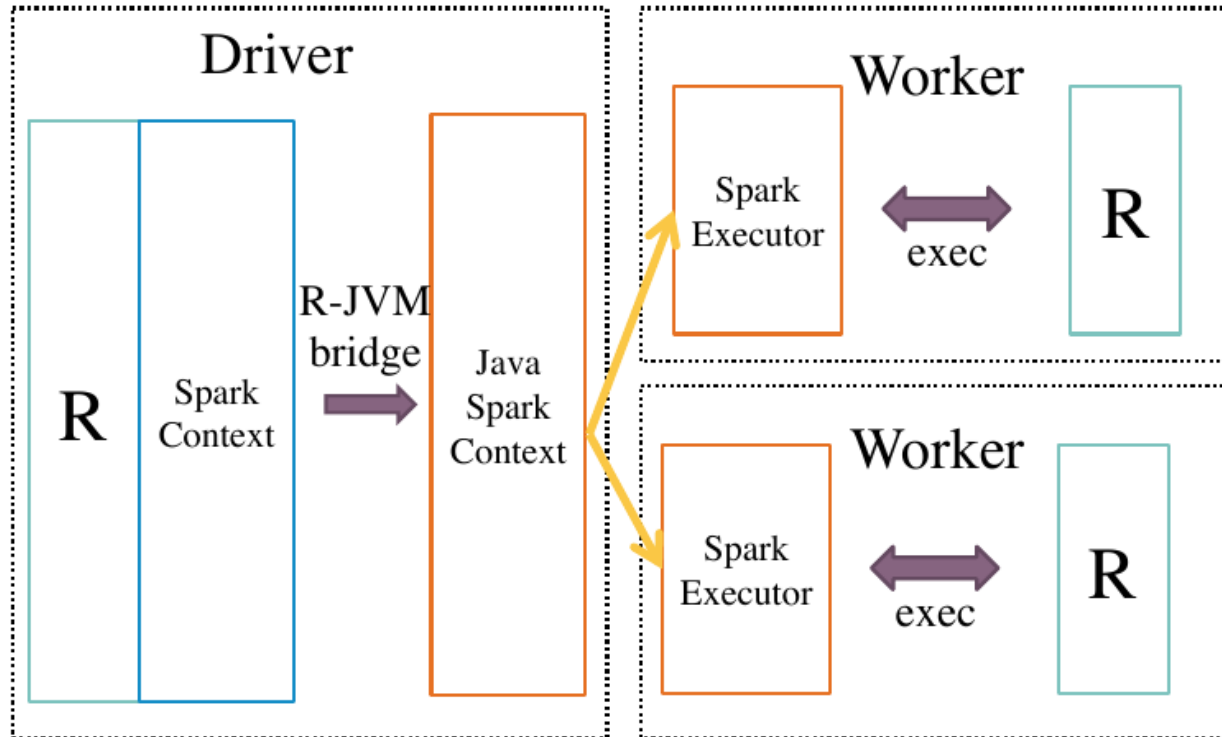
R is a popular statistical programming language with a number of extensions that support data processing and machine learning tasks. However, interactive data analysis in R is usually limited as the R runtime is single threaded and can only process data sets that fit in a single machine's memory. We present SparkR, an R package that provides a frontend to Apache Spark and uses Spark's distributed computation engine to enable large scale data analysis from the R shell. We describe the main design goals of SparkR, discuss how the high-level DataFrame API enables scalable computation and present some of the key details of our implementation.

support for reading input from a variety of systems including HDFS, HBase, Cassandra and a number of formats like JSON, Parquet, etc. Integrating with the data source API enables R users to directly process data sets from any of these data sources.

Performance Improvements: As opposed to a new distributed engine, SparkR can inherit all of the optimizations made to the Spark computation engine in terms of task scheduling, code generation, memory management [3], etc.

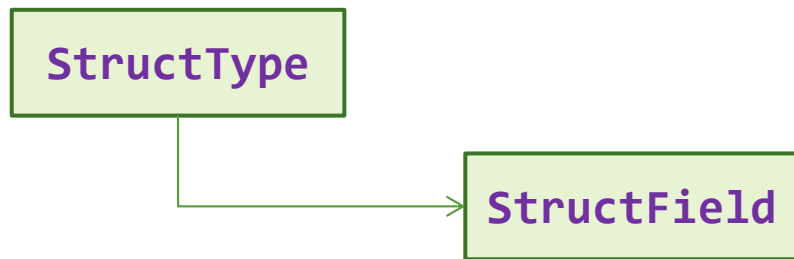
SparkR is built as an R package and requires no changes to R. The central component of SparkR is a distributed data frame that enables structured data processing with a syntax familiar to R users [31]/Figure 1). To improve performance over large datasets

Arquitectura



SparkDataFrame (*)

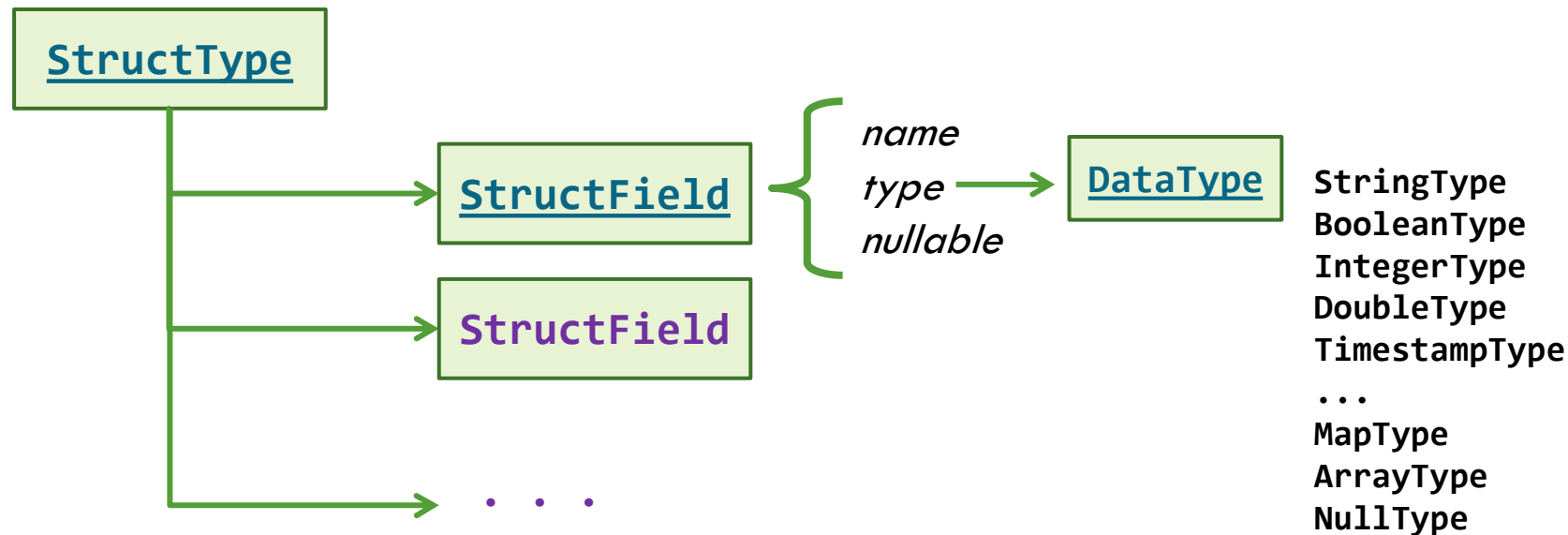
- Una estructura en formato de tabla
- La tabla tiene un esquema
- Todos los elementos de una columna tienen el mismo tipo
- Similar a los DataFrames nativos de R



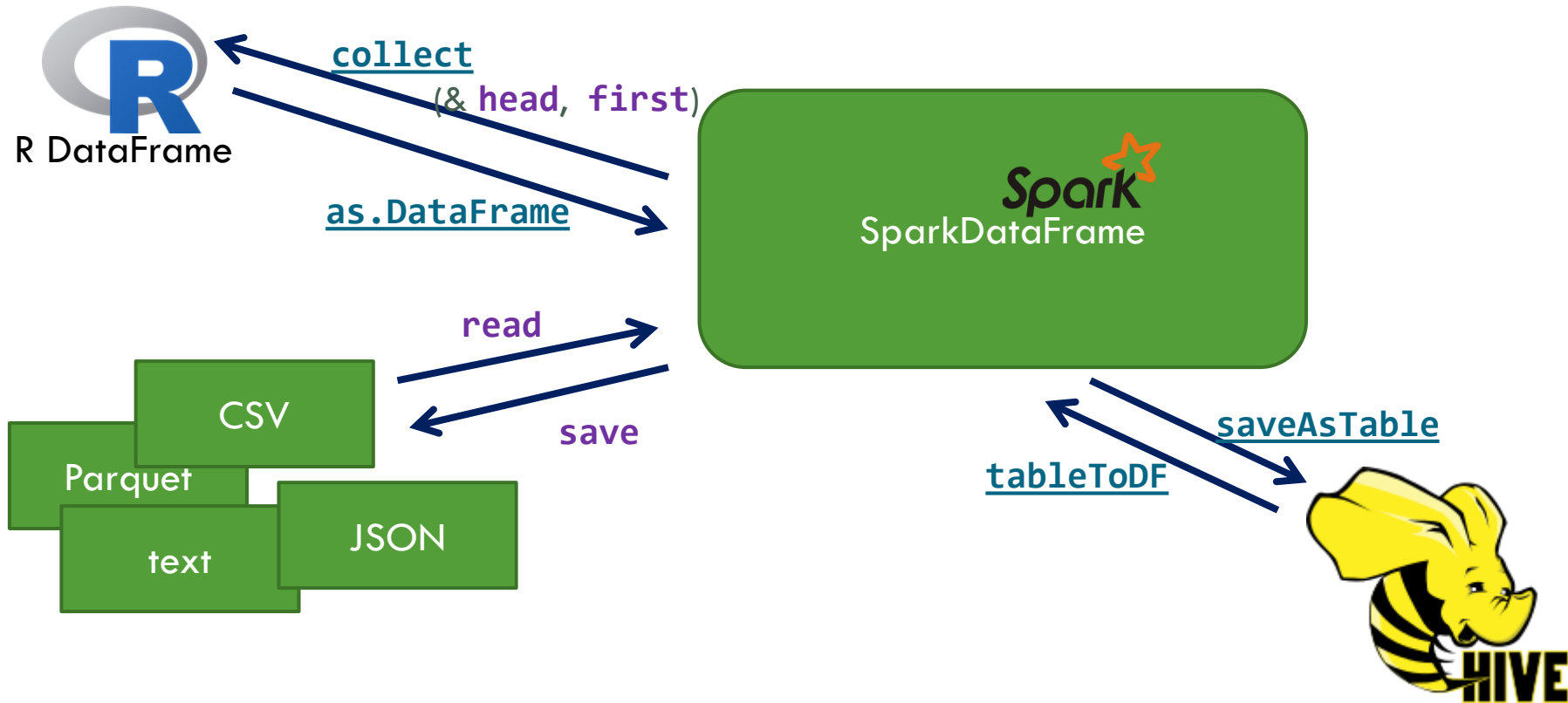
(*) **SparkDataFrame** en Spark 2.x, **DataFrame** en Spark 1.x

SparkDataFrame

- Creación de un **Schema** para un **SparkDataFrame**



Creación de SparkDataFrames



APIs disponibles

- No hay API de RDDs
- Se permiten la mayoría de las operaciones de DataFrames
 - Usando verbos típicos de dplyr
- No incorpora DataSets
- Se están incorporando las funcionalidades de ML
 - (suele estar algo menos actualizado que Scala o Python)

I – dplyr API: Verbos disponibles

SparkR	SQL
select	SELECT
filter	WHERE
arrange	ORDER
summarise (con groupBy)	GROUP BY + agregaciones: sum, min, sd, etc.
mutate	operaciones: +, *, log, etc.
distinct	DISTINCT

II – combinaciones de SparkDataFrames

SparkR	SQL
<code>join</code>	JOIN
<code>intersect</code>	INTERSECT
<code>union</code>	UNION ALL

III – varios

SparkR	significado
<code>sql</code>	consulta mediante sintaxis SQL
<code>sample</code> & <code>sampleBy</code>	obtiene una muestra aleatoria de las filas
<code>randomSplit</code>	divide aleatoriamente un DataFrame en varios trozos

IV – uso de UDFs nativas de R sobre SparkDataFrames

SparkR	significado
<code>dapply</code>	aplica una función por particiones
<code>dapplyCollect</code>	aplica una función por particiones; recoge el resultado
<code>gapply</code>	agrupa por columnas y aplica una función a cada grupo
<code>gapplyCollect</code>	agrupa columnas, aplica a cada grupo, recoge resultado
<code>spark.lapply</code>	itera una función sobre una lista, dentro de Spark

V – Machine Learning

- SparkR proporciona APIs en R para poder ejecutar un conjunto de los algoritmos disponibles en Spark ML
- La lista varía de versión en versión, para saber qué hay disponible en una versión concreta es necesario consultar la [documentación](#)

R closures

- Las UDFs son funciones de R nativas
 - se envían a los nodos para su ejecución sobre datos nativos de R
 - junto con la función se envían los datos referidos por ella (*closure*)
- Aspectos por considerar
 - **¡cuidado con el tamaño de los datos incorporados!**
 - los paquetes locales al driver **no** se incorporan a la *closure*

Machine learning en SparkR

- Proporciona un API R para usar los algoritmos en Spark MLlib
Regresión, clasificación, análisis estadístico
- El API sigue las convenciones de R, incluyendo la expresión mediante fórmulas
$$y \sim x1 + x2 + x1:x3$$
- `spark.glm`, `spark.logit`, `spark.randomForest`, `spark.gbt`,
`spark.kmeans`, etc

La cobertura de Spark MLlib es ya significativa en versiones recientes de Spark (2.x o superior)

sparklyr

sparklyr

- API alternativo para trabajar con Spark desde R
- Creado por [RStudio](#)
- Permite utilizar la interfaz de [dplyr](#) con SparkDataFrames

Verbos disponibles

dplyr / sparklyr	
select	columnas
slice	filas
filter	filas
arrange	filas
distinct	filas
summarise (con group_by)	columnas & filas, con agregaciones: sum, mean, min, max, sd, median, IQR, n , etc.
mutate & transmute	columnas
sample_n & sample_frac	filas

API

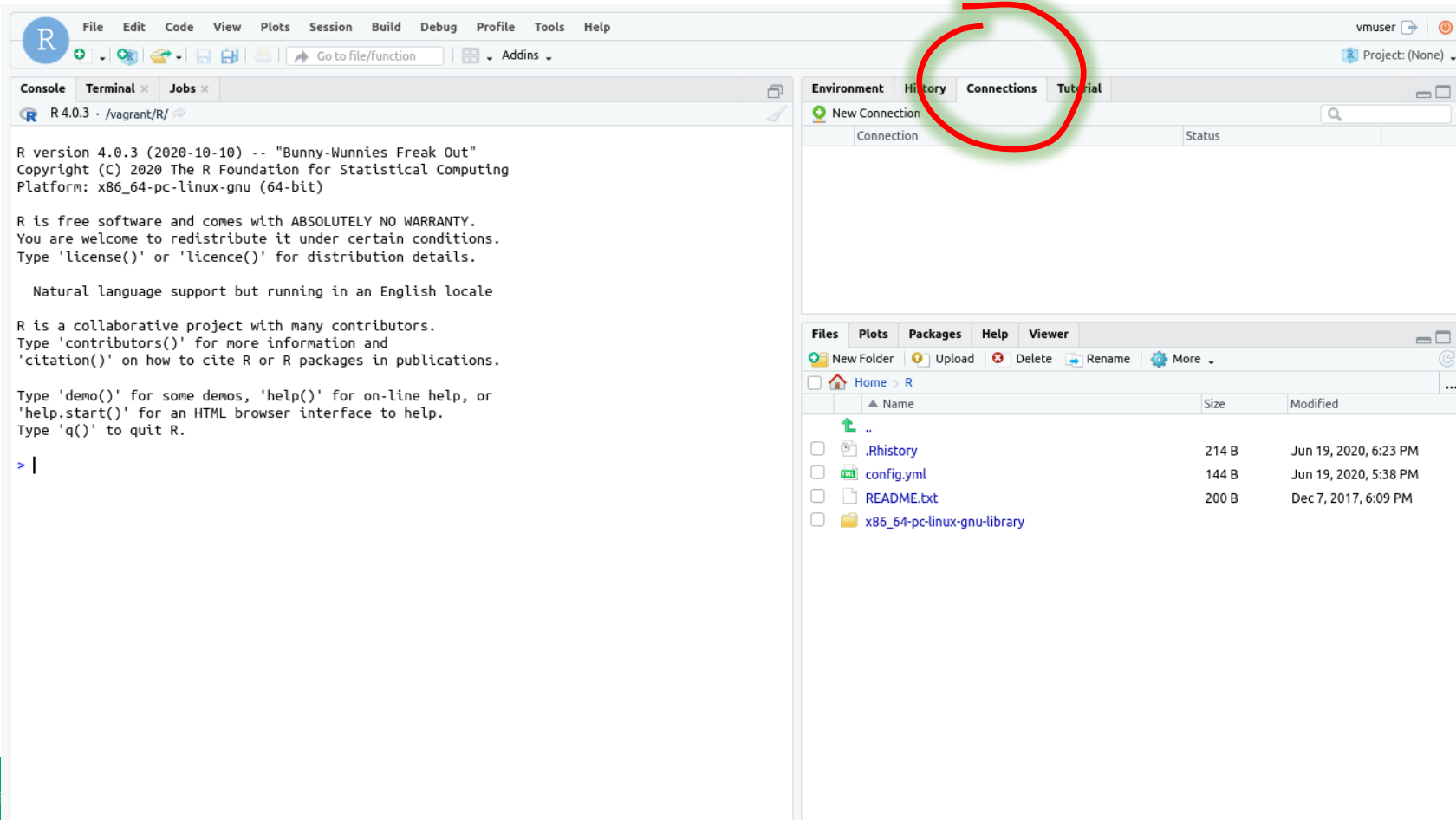
- dplyr: Verbos disponibles en dplyr
- MLlib: acceso al API de DataFrames de Spark.MLlib
- Es posible crear extensiones para sparklyr que exponen funcionalidad de Spark

Spark & RStudio

Uso de Spark en RStudio

- RStudio a partir de la versión 0.99 (aprox) ofrece soporte nativo para Spark
- Permite conectarse a un cluster de Spark mediante el API de **sparklyr**

Uso de Spark en RStudio



The screenshot shows the RStudio interface. The top menu bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Profile, Tools, and Help. The top right corner shows the user 'vmuser' and the project 'Project: (None)'. The left pane contains the Console, Terminal, and Jobs tabs. The Console shows the R version 4.0.3 (2020-10-10) and the R Foundation for Statistical Computing logo. The right pane contains the Environment, History, Connections, and Tutorial tabs. The 'Connections' tab is highlighted with a red circle. Below the tabs, there is a 'New Connection' button and a search bar. The bottom pane shows the Files, Plots, Packages, Help, and Viewer tabs. The Files tab is active, showing a file explorer view of the 'Home > R' directory. The file explorer shows a list of files and folders with columns for Name, Size, and Modified.

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

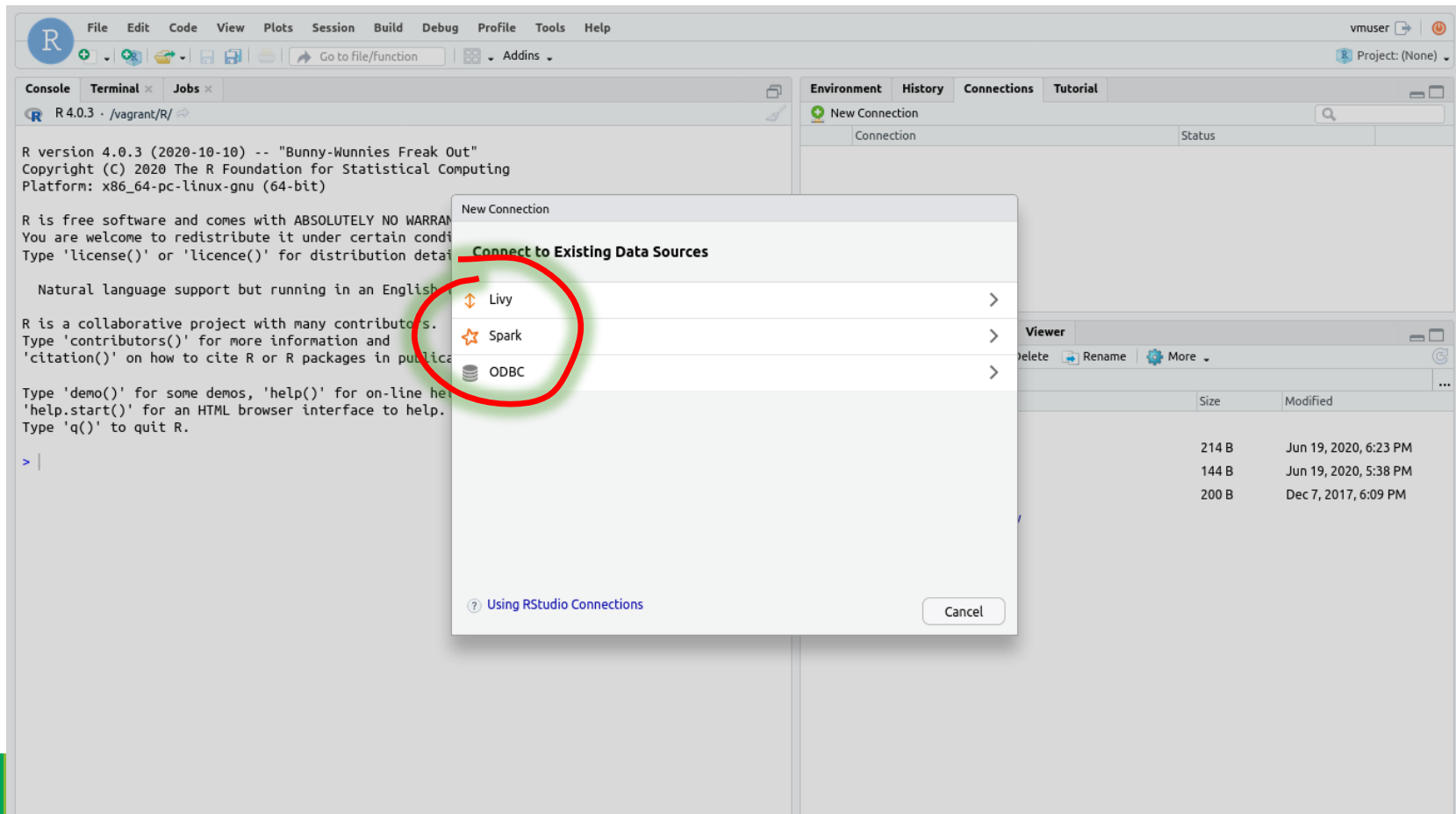
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |

	Name	Size	Modified
	..		
<input type="checkbox"/>	.Rhistory	214 B	Jun 19, 2020, 6:23 PM
<input type="checkbox"/>	config.yml	144 B	Jun 19, 2020, 5:38 PM
<input type="checkbox"/>	README.txt	200 B	Dec 7, 2017, 6:09 PM
<input type="checkbox"/>	x86_64-pc-linux-gnu-library		

Uso de Spark en RStudio



The screenshot shows the RStudio interface with the 'New Connection' dialog box open. The dialog box has a title bar 'New Connection' and a section 'Connect to Existing Data Sources'. Under this section, three options are listed: 'Livy', 'Spark', and 'ODBC'. The 'Spark' option is highlighted with a red circle. The background shows the R console with the R version 4.0.3 (2020-10-10) and the 'Environment' pane on the right.

R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |

New Connection

Connect to Existing Data Sources

- Livy
- Spark**
- ODBC

Using RStudio Connections

Cancel

Environment History Connections Tutorial

New Connection

Connection Status

Viewer

Delete Rename More

Size	Modified
214 B	Jun 19, 2020, 6:23 PM
144 B	Jun 19, 2020, 5:38 PM
200 B	Dec 7, 2017, 6:09 PM

Cuadro final

Resumen de APIs y lenguajes

	RDD	DataFrames	Dataset
Scala	org.apache.spark.rdd.RDD RDD[T]	org.apache.spark.sql.DataFrame DataFrame = Dataset[Row]	org.apache.spark.sql.Dataset Dataset[T]
Java	org.apache.spark.api.java.JavaRDD JavaRDD<type>	org.apache.spark.sql.Dataset Dataset<Row>	org.apache.spark.sql.Dataset Dataset<T>
Python	pyspark.RDD RDD	pyspark.SQL.DataFrame DataFrame	<i>no disponible</i>
R	<i>no disponible</i>	SparkDataFrame	<i>no disponible</i>