



Studio[®]

Extending Spark using sparklyr and R

A screenshot of the R Studio interface. The top navigation bar includes File, Edit, Code, View, Plots, Session, Build, Debug, Tools, Help, Go to file/function, Addins, Environment, History, and Git. The main area shows an R script named 'chicagoFood.R' with the following code:

```
1 url <- "http://data.cityofchicago.org/api/views/4ijn-s7e5/rows.csv?c"
2 data <- read.csv(url, header = TRUE) # takes a minute...
3 names(data) <- tolower(names(data))
4 data1 <- subset(data, risk %in% c("Risk 1 (High)", "Risk 2 (Medium)", "Risk 3 (Low)"))
5 data1$risk <- droplevels(data1$risk)
6
7 data1 <- data1[1:50,]
8 library(leaflet)
9 leaflet(data1) %>%
10   addTiles() %>%
11   addMarkers(lat = ~latitude, lng = ~longitude)
```

The console below shows the same code being run.

```
> data1 <- subset(data, risk %in% c("Risk 1 (High)", "Risk 2 (Medium)", "Risk 3 (Low)"))
> data1$risk <- droplevels(data1$risk)
>
> data1 <- data1[1:50,]
> library(leaflet)
> leaflet(data1) %>%
+   addTiles() %>%
+   addMarkers(lat = ~latitude, lng = ~longitude)
```

To the right, the Global Environment panel shows 'data' (118607 obs. of 17 variables) and 'data1' (50 obs. of 17 variables). Below that is a map of Chicago with several blue location pins placed on it, corresponding to the data points. The map includes labels for neighborhoods like Norridge, Elmwood, and Oak Park, and shows various roads and highways.

EXTENDING SPARK USING SPARKLYR AND R

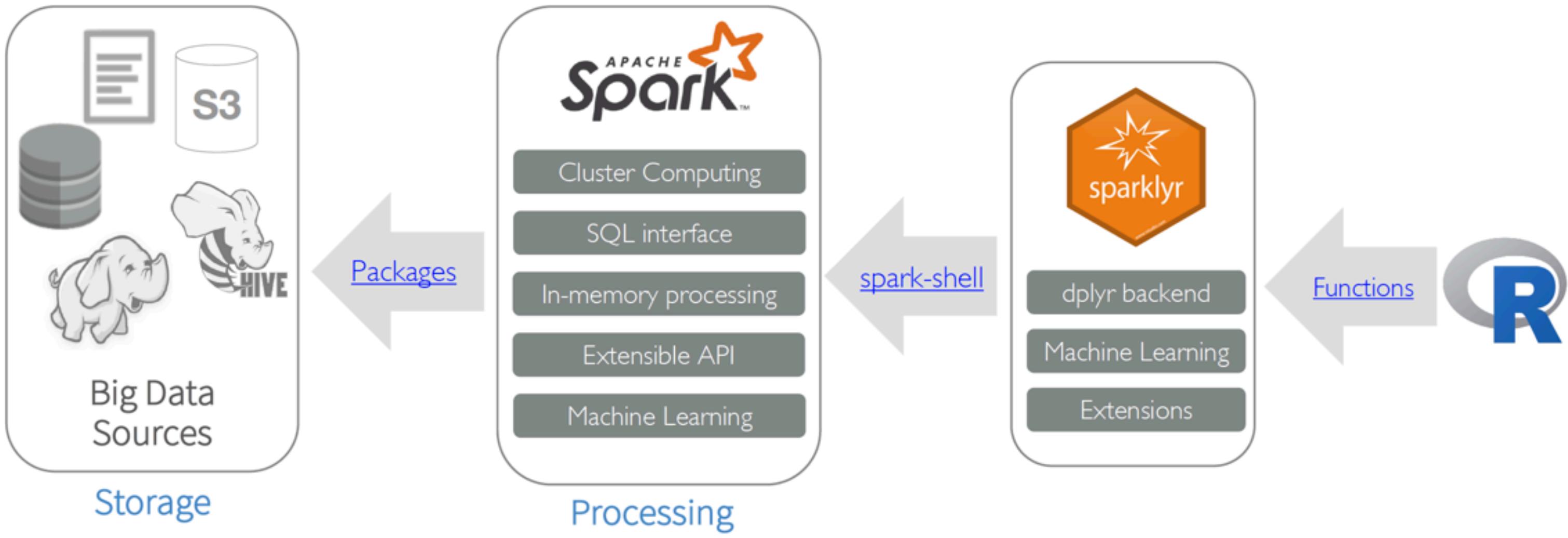
- ✓ Extensions Overview
- ✓ Using Extensions
- ✓ Extensions using R Code
- ✓ Extensions using Scala Code
- ✓ Extensions as R Packages



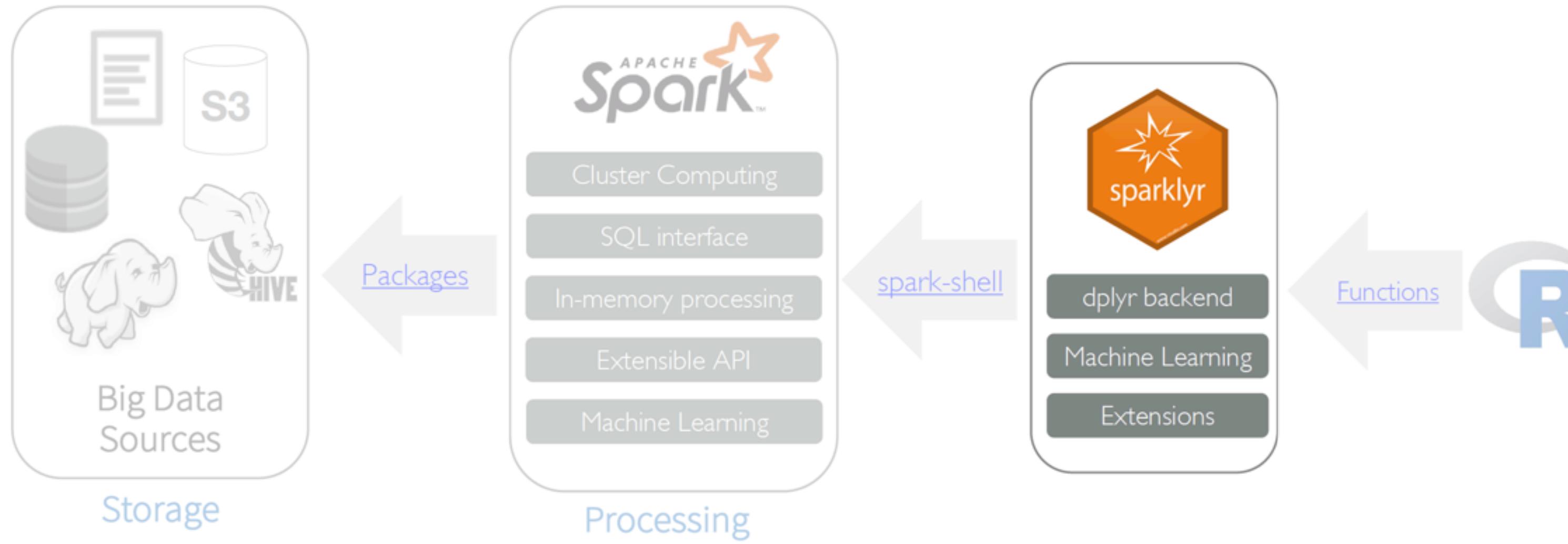
Overview

R

OVERVIEW



OVERVIEW



Creating R Packages

```
1  install.packages(rsparkling)
2
3
```

Using Extensions

```
1 library(rsparkling)
2
3
```

Using R Code

```
1 spark_dataframe(iris_tbl) %>%
2   invoke("isLocal")
3
4
```

Using Scala Code

```
1 package SparkHello
2
3
4 object HelloWorld {
5   def hello(): String = {
6     "Hello, world! - From Scala"
7   }
8 }
```

Extensions

R

USING EXTENSIONS

Secure | https://spark.rstudio.com

sparklyr Home Gallery Guides Deployment Reference

- Connect to Spark from R. The provides a complete `dplyr` backend.
- Filter and aggregate Spark data them into R for analysis and visualization.
- Use Spark's distributed machine learning from R.
- Create `extensions` that call the full Spark API and provide interfaces to Spark packages.

dplyr
mllib
H2O
Extensions
Caching
Distributed R

sparklyr
ML
Extensions

Apache Spark

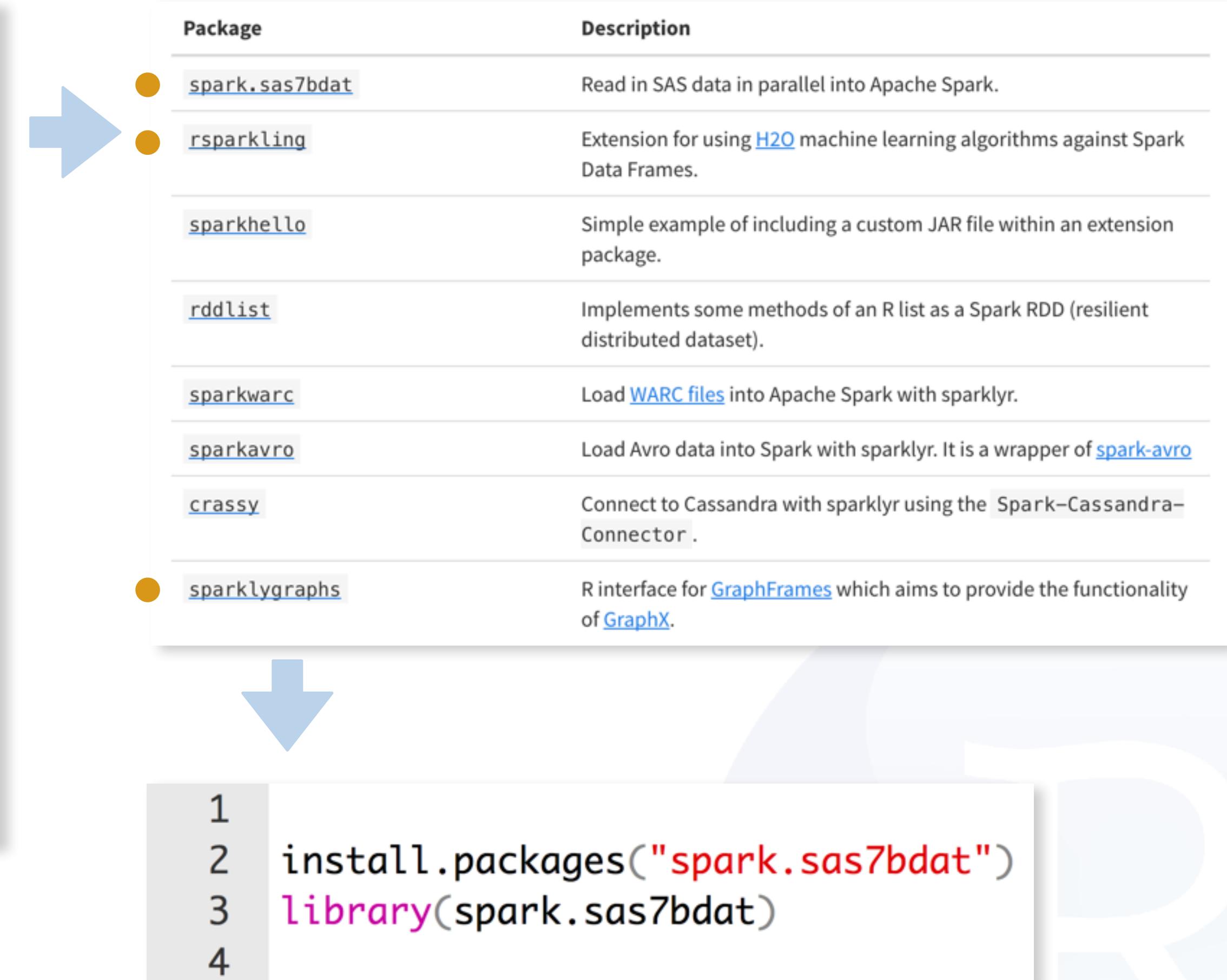
You can install the `sparklyr` package from CRAN as follows:

```
install.packages("sparklyr")
```

You should also install a local version of Spark for development purposes:

```
library(sparklyr)
spark_install(version = "2.1.0")
```

<https://spark.rstudio.com/articles/guides-extensions.html>



USING EXTENSIONS

rpubs.com/juraschi/sparklyr-extensions-webinar-2017

Rpubs brought to you by RStudio

Extending Spark using sparklyr and R

Setup

Initialize all extensions for this webinar, at once and connect to Spark:

```
install.packages(spark.sas7bdat)
install.packages(resparkling)
devtools::install_github("kevinyxu/sparklyrgraphs")
```

```
options(resparkling.sparklingwater.version = "2.1.0")

library(spark.sas7bdat)
library(resparkling)
library(sparklyrgraphs)

library(sparklyr)
sc <- spark_connect(master = "local", version = "2.1.0")
```

Reading SAS files with spark.sas7bdat

```
spark_read_sas(sc, "inst/datasets/datetime.sas7bdat", "sas_data")
```

VAR1 <chr>	VAR2 <chr>	VAR3 <chr>	VAR4 <dbl>	VAR5 <dbl>
2015-02-02 06:42:12.0	2015-02-01	2015-02-01	20121	52932
2014-01-01 02:14:23.0	2013-12-31	2013-12-31	19724	38063
2015-01-14 22:15:22.0	2015-08-14	2015-08-14	20254	22522
1948-09-09 14:32:00.0	1948-09-15	1948-09-15	-4124	77520

Edit Details Delete sparklyr - extensions webinar - 2017 by Javier Luraschi Last updated about 7 hours ago

Comments Share Hide Toolbar

RPubs.com/juraschi/sparklyr-extensions-webinar-2017

RPubs brought to you by RStudio

sparklyr: R Interface for Apache Spark

```
title: "sparklyr: R Interface for Apache Spark"
output:
  github_document:
    fig_width: 9
    fig_height: 5
---
[(https://travis-ci.org/rstudio/sparklyr.svg?branch=master)](https://travis-ci.org/rstudio/sparklyr) [](https://gitter.im/rstudio/sparklyr?utm_medium=badge&utm_content=badge)
```

```
```{r setup, include=FALSE}
knitr::opts_chunk$set(eval = TRUE)
knitr::opts_chunk$set(warning = FALSE)
knitr::opts_chunk$set(fig.path = "tools/readme/", dev = "png")
````
```

```
img src="tools/readme/sparklyr-illustration.png" width=364 height=187 align="right"/>
```

- Connect to [Spark](http://spark.apache.org/) from R. The sparklyr package provides a `dbn` complete [dplyr](https://github.com/hadley/dplyr) backend.
- Filter and aggregate Spark datasets then bring them into R for `dbn` analysis and visualization.
- Use Spark's distributed [Machine Learning](http://spark.apache.org/docs/latest/ml-guide.html) library from R.
- Create [extensions](http://spark.rstudio.com/extensions.html) that call the full Spark API and provide `dbn` interfaces to Spark packages.

```
## Installation
You can install the **sparklyr** package from CRAN as follows:
```

```
```{r, eval=FALSE}
install.packages("sparklyr")
````
```

You should also install a local version of Spark for development purposes:

```
```{r, eval=FALSE}
library(sparklyr)
spark_install(version = "2.1.0")
````
```

To upgrade to the latest version of sparklyr, run the following command and restart your R session:

```
```{r, eval=FALSE}
spark_install()
````
```

43.1 | Chunk 4 |

Console

R Code



SPARK PACKAGES

The screenshot shows the homepage of the Spark Packages website at <https://spark-packages.org>. The page has a blue header with the title "SparkPackages". Below the header, there are navigation links for "Feedback", "Register a package", "Login", and "Find a package". A search bar is also present. The main content area displays a list of packages. At the top of the list is "spark-als", described as "Another, hopefully better, implementation of ALS on Spark (already merged into MLlib)". Below it is "mllib-grid-search", described as "An example project for doing grid search in MLlib". At the bottom of the list is "spark-avro", described as "Integration utilities for using Spark with Apache Avro data". Each package entry includes a link to its details page, a release date, a license, and a star rating.

```
1 library(sparklyr)
2
3 config <- spark_config()
4 config$sparklyr.defaultPackages <- c(
5   "datastax:spark-cassandra-connector:2.0.0-RC1-s_2.11"
6 )
7
8
9 sc <- spark_connect(master = "local", config = config)
10
```

```
1 spark_read_source(
2   sc,
3   "emp",
4   "org.apache.spark.sql.cassandra",
5   list(keyspace = "dev", table = "emp")
6 )
7
8
```

INVOKED() - INTRO

The screenshot shows a web browser displaying the Apache Spark Streaming Programming Guide. The URL is <https://spark.apache.org/docs/latest/streaming-programming-guide.html>. The page content includes Scala code snippets for creating a StreamingContext and processing a DStream of words.

```
import org.apache.spark._  
import org.apache.spark.streaming._  
import org.apache.spark.streaming.StreamingContext._ // not necessary since Spark 1.3  
  
// Create a local StreamingContext with two working thread and batch interval of 1 second.  
// The master requires 2 cores to prevent from a starvation scenario.  
  
val conf = new SparkConf().setMaster("local[2]").setAppName("NetworkWordCount")  
val ssc = new StreamingContext(conf, Seconds(1))  
  
Using this context, we can create a DStream that represents streaming data from a TCP source, specified as hostname (e.g. localhost) and port  
(e.g. 9999).  
  
// Create a DStream that will connect to hostname:port, like localhost:9999  
val lines = ssc.socketTextStream("localhost", 9999)  
  
This lines DStream represents the stream of data that will be received from the data server. Each record in this DStream is a line of text. Next, we want to split the lines by space characters into words.  
  
// Split each line into words  
val words = lines.flatMap(_.split(" "))  
  
flatMap is a one-to-many DStream operation that creates a new DStream by generating multiple new records from each record in the source DStream. In this case, each line will be split into multiple words and the stream of words is represented as the words DStream. Next, we want to count these words.  
  
import org.apache.spark.streaming.StreamingContext._ // not necessary since Spark 1.3  
// Count each word in each batch  
val pairs = words.map(word => (word, 1))  
val wordCounts = pairs.reduceByKey(_ + _)  
  
// Print the first ten elements of each RDD generated in this DStream to the console  
wordCounts.print()
```

The words DStream is further mapped (one-to-one transformation) to a DStream of (word, 1) pairs, which is then reduced to get the frequency of words in each batch of data. Finally, wordCounts.print() will print a few of the counts generated every second.

```
1  ssc <- invoke_new(  
2      sc,  
3      "org.apache.spark.streaming.StreamingContext",  
4      spark_context(sc),  
5      invoke_new(  
6          sc,  
7          "org.apache.spark.streaming.Duration",  
8          10000L  
9      )  
10     )  
11     )  
12     )
```

Invoke() - MAPPINGS

| From R | Scala | To R |
|-------------|----------------|-----------|
| NULL | void | NULL |
| integer | Int | integer |
| character | String | character |
| logical | Boolean | logical |
| double | Double | double |
| numeric | Double | double |
| | Float | double |
| | Decimal | double |
| | Long | double |
| raw | Array[Byte] | raw |
| Date | Date | Date |
| POSIXlt | Time | |
| POSIXct | Time | POSIXct |
| list | Array[T] | list |
| environment | Map[String, T] | |
| jobj | Object | jobj |

```
1 ensure_scalar_integer()
2 ensure_scalar_double()
3 ensure_scalar_boolean()
4 ensure_scalar_character()
```

6

Scala Code

R

COMPILE_PACKAGE_JARS()

The screenshot shows a GitHub repository page for 'javierluraschi/sparkhello'. The repository has 18 commits, 1 branch, 0 releases, and 3 contributors. The latest commit was made on Jul 12. The file list includes .Rbuildignore, .gitignore, DESCRIPTION, LICENSE, NAMESPACE, README.Rmd, README.md, and sparkhello.Rproj.

| Name | Size | Modified |
|------------------|------|----------|
| .Rbuildignore | | |
| .gitignore | | |
| DESCRIPTION | | |
| LICENSE | | |
| NAMESPACE | | |
| README.Rmd | | |
| README.md | | |
| sparkhello.Rproj | | |

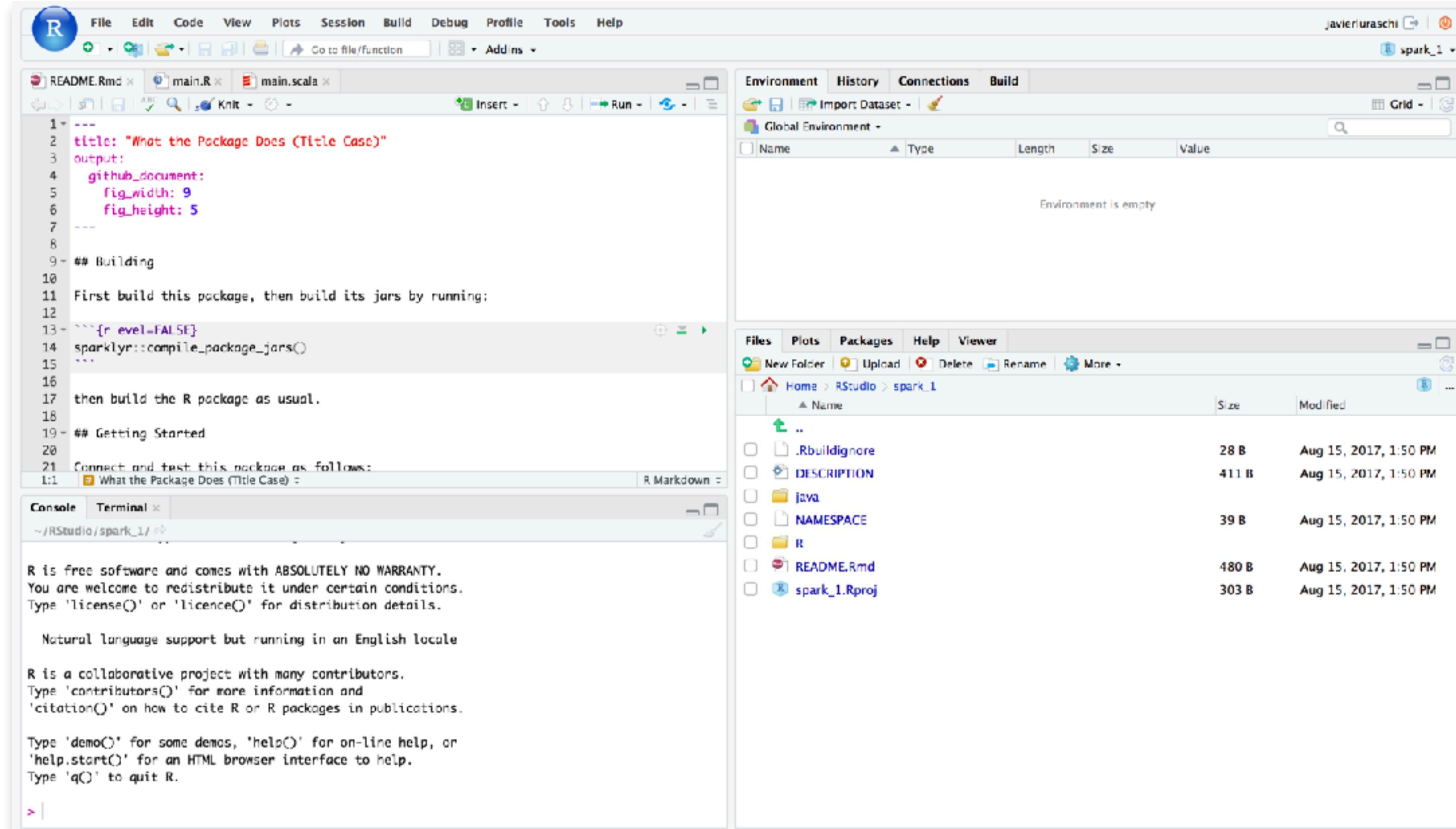
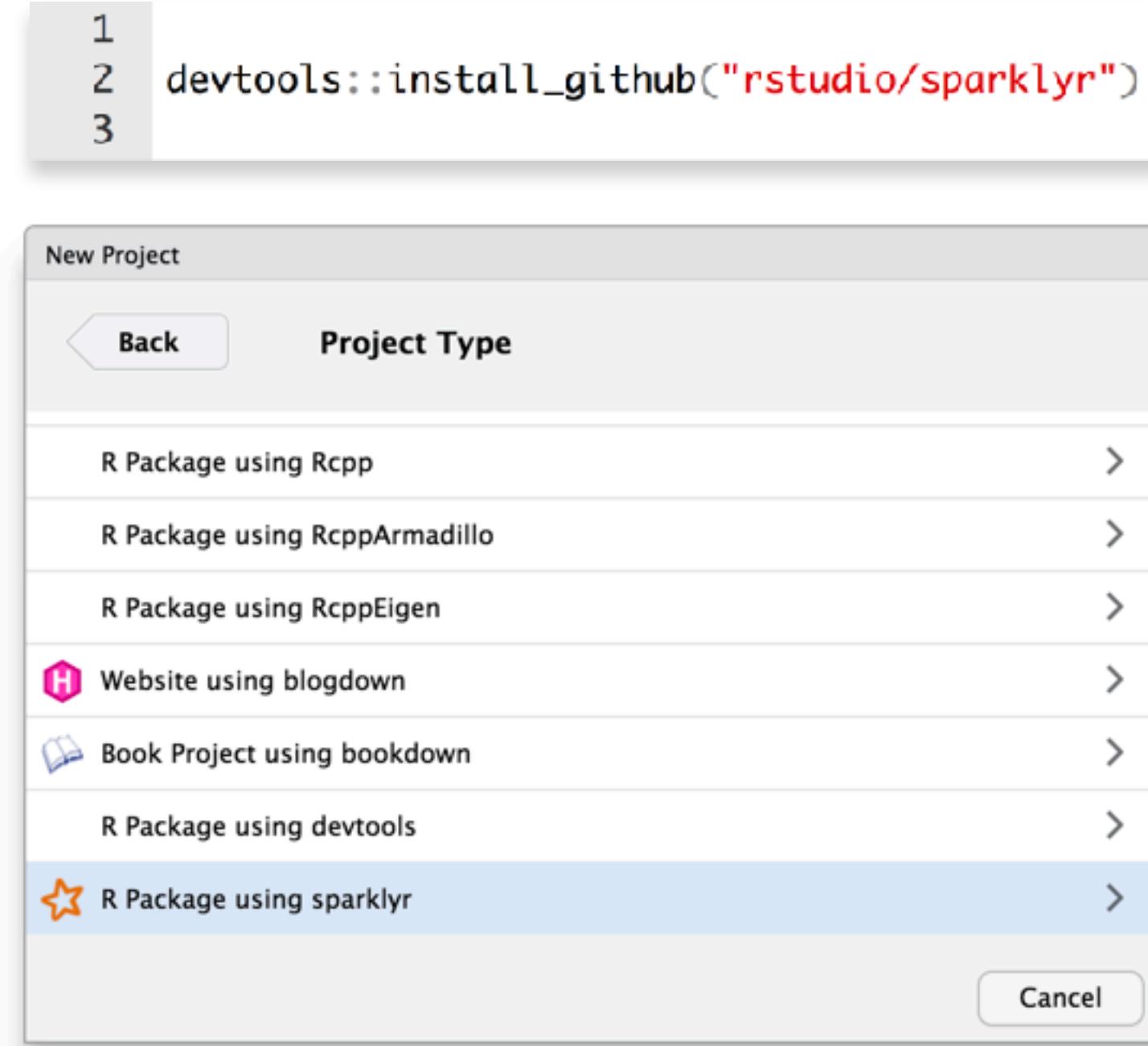
The screenshot shows an RStudio interface with a file browser window. The path is Home > RStudio > sparkhello2 > inst > java. The file list contains five JAR files: sparkhello2-1.5-2.10.jar, sparkhello2-1.6-2.10.jar, sparkhello2-2.0-2.11.jar, sparkhello2-2.1-2.11.jar, and sparkhello2-2.2-2.11.jar. All files are 1.4 KB or 1.5 KB in size and were modified on Aug 7, 2017, at 6:00 PM.

| Name | Size | Modified |
|--------------------------|--------|----------------------|
| sparkhello2-1.5-2.10.jar | 1.4 KB | Aug 7, 2017, 6:00 PM |
| sparkhello2-1.6-2.10.jar | 1.4 KB | Aug 7, 2017, 6:00 PM |
| sparkhello2-2.0-2.11.jar | 1.5 KB | Aug 7, 2017, 6:00 PM |
| sparkhello2-2.1-2.11.jar | 1.5 KB | Aug 7, 2017, 6:00 PM |
| sparkhello2-2.2-2.11.jar | 1.5 KB | Aug 7, 2017, 6:00 PM |

R Packages

R

R PACKAGES

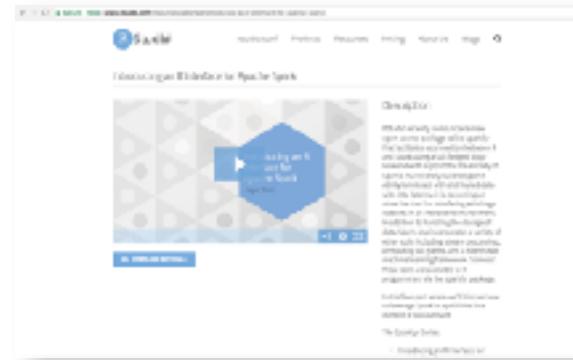


RStudio New Project integration requires sparklyr devel or sparklyr 0.7 (unreleased)

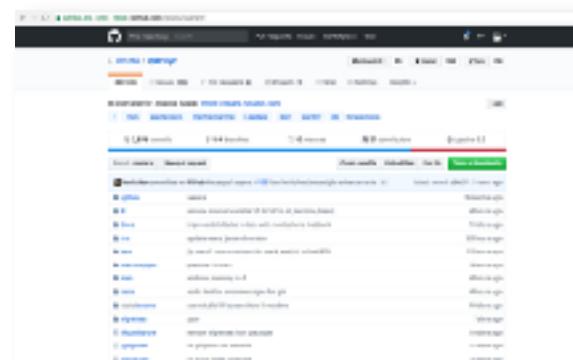
USEFUL WEBSITES



RStudio and Spark:
spark.rstudio.com



RStudio Webinar Resources:
rstudio.com/resources/webinars/



Sparklyr Github:
github.com/rstudio/sparklyr



Spark Documentation:
spark.apache.org/docs/2.1.0/api/scala/

Questions

R



Open Source & Free

Desktop: <http://www.rstudio.com/products/rstudio/download/>

RStudio Server: <http://www.rstudio.com/products/rstudio/download-server/>

Shiny Server: <http://www.rstudio.com/products/shiny/download-server/>

shinyapps.io beta: <https://www.shinyapps.io/admin/#/signup>

45 Day Evaluation of Pro Products

RStudio Server Pro: <http://www.rstudio.com/products/rstudio-server-pro/evaluation/>

Shiny Server Pro: <http://www.rstudio.com/products/shiny-server-pro/evaluation/>

PLEASE STAY IN TOUCH



Blog - <http://rviews.rstudio.com/>



Blog - <http://blog.rstudio.org/>



Twitter - @rstudio #rstats <http://twitter.com/rstudio/>



GitHub - <https://github.com/rstudio/>



LinkedIn - <https://linkedin.com/company/rstudio-inc>



Facebook - <https://www.facebook.com/pages/RStudio-inc>



Google+ - <https://plus.google.com/110704473211154995841/posts>