# Notes for this computer

Windows path for R has been set for R version 3.2.1 as “C:\Program Files\R\R-3.2.1\bin\x64”

Spark is installed in “C:\Apache\spark-1.4.1-bin-hadoop2.6” (NOTE- use “/” instead of “\” when writing the path inside R.

# Installing and Starting SparkR Locally on Windows OS and RStudio

*July 26, 2015*

By emaasit



**Introduction**

With the recent release of Apache Spark 1.4.1 on July 15th, 2015, I wanted to write a step-by-step guide to help new users get up and running with SparkR locally on a Windows machine using command shell and RStudio. SparkR provides an R frontend to Apache Spark and using Spark’s distributed computation engine allows R-Users to run large scale data analysis from the R shell. The steps listed here are also documented in my online book title “Getting Started with SparkR for Big Data Analysis” which can be accessed at: http://www.danielemaasit.com/getting-started-with-sparkr/. These steps will get you up and running in less than 5 mins.

**Prerequisite**

Make sure you have Java 6+ installed on your computer and the system environments set.

**Step 1: Download Spark**

Open your web browser and open this web page: http://spark.apache.org/. This is the official website for the Apache Spark project. You should see a large green button to the right of the page that reads “Download Spark”, as shown in Figure 1. Click the green button.

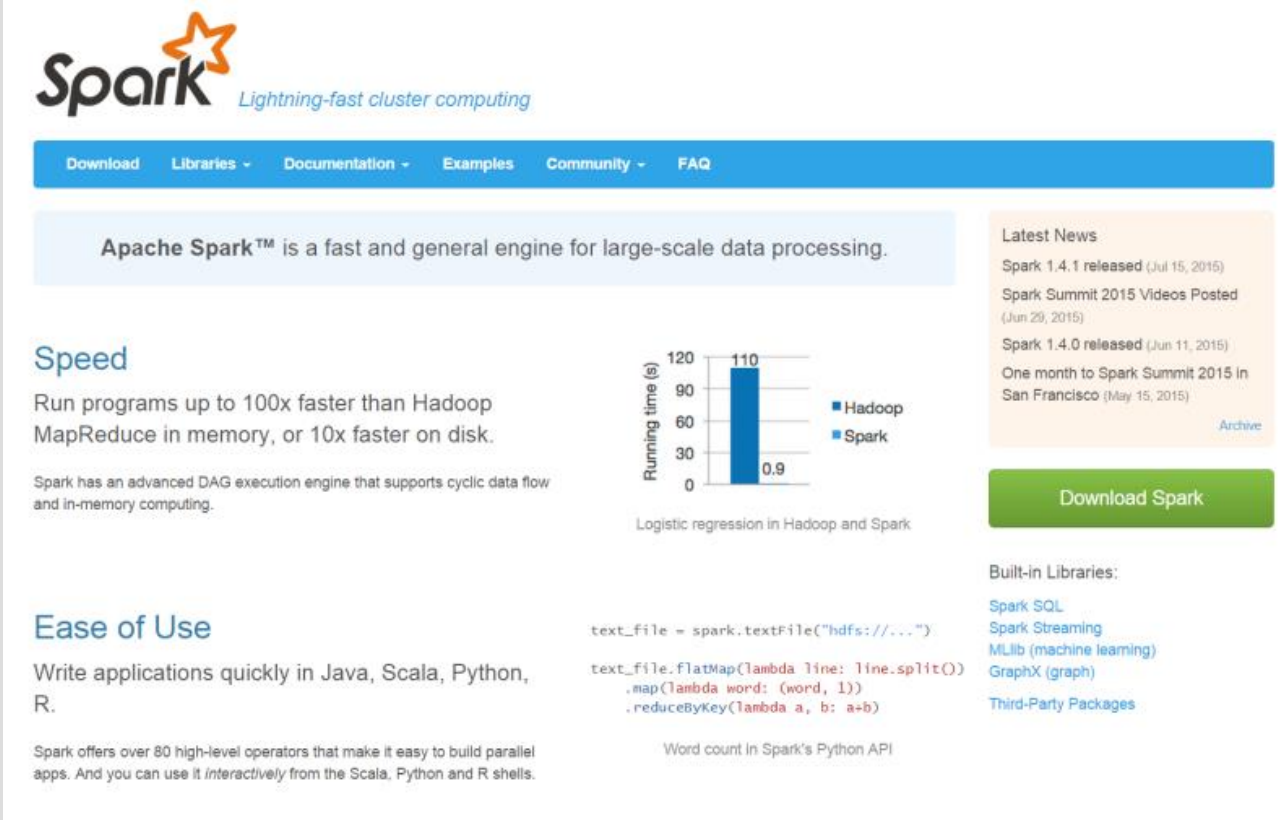


**Figure 1: Apache Spark home page**

Clicking the green button will take you to the download page as shown in Figure 2 below.

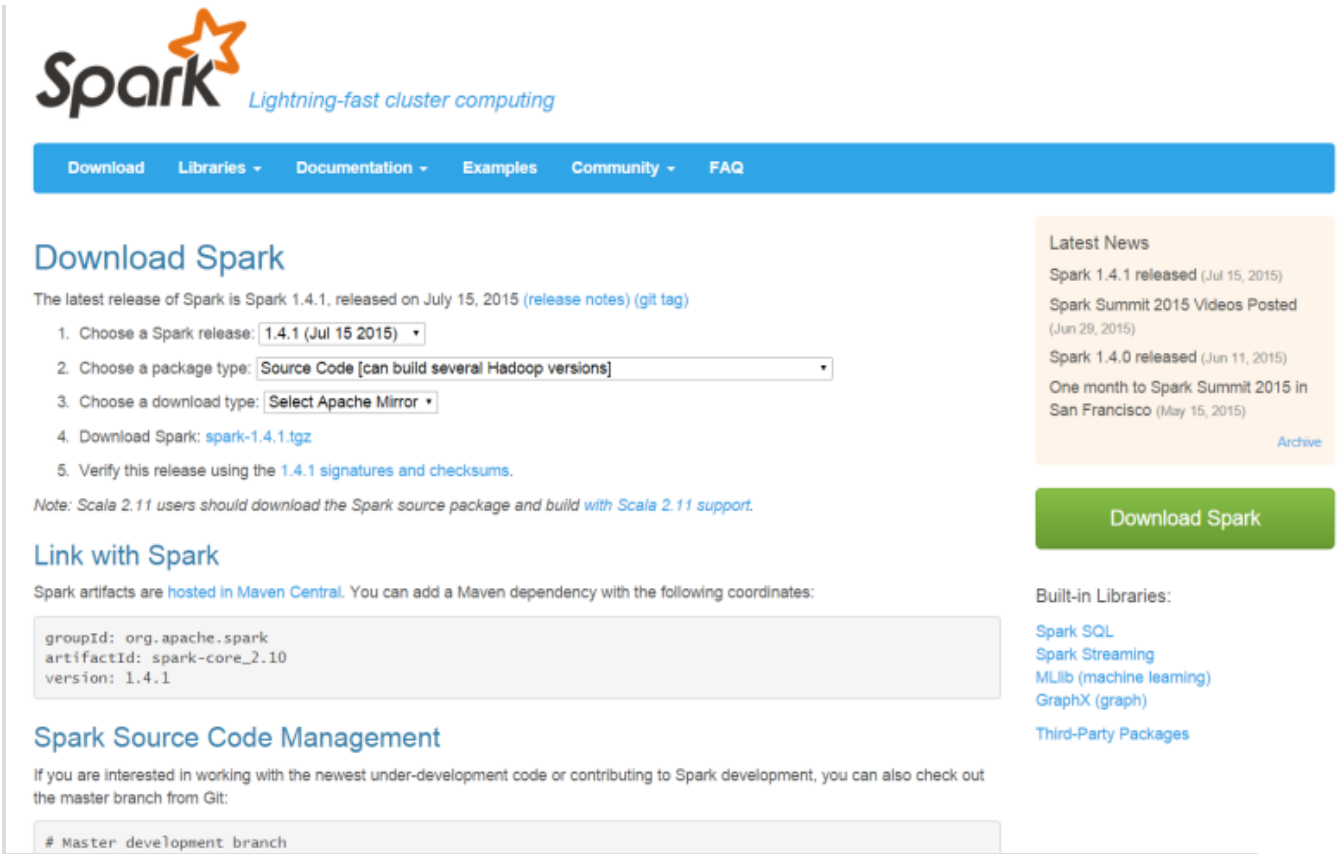


**Figure 2: The download page**

You should follow the steps 1 to 3 to create a download link for a Spark Package of your choice. On the *"2. Choose a package type"* option, select any pre-built package type from the drop-down list (Figure 3). Since we want to experiment locally on windows, a pre-built package for Hadoop 2.6 and later will suffice.

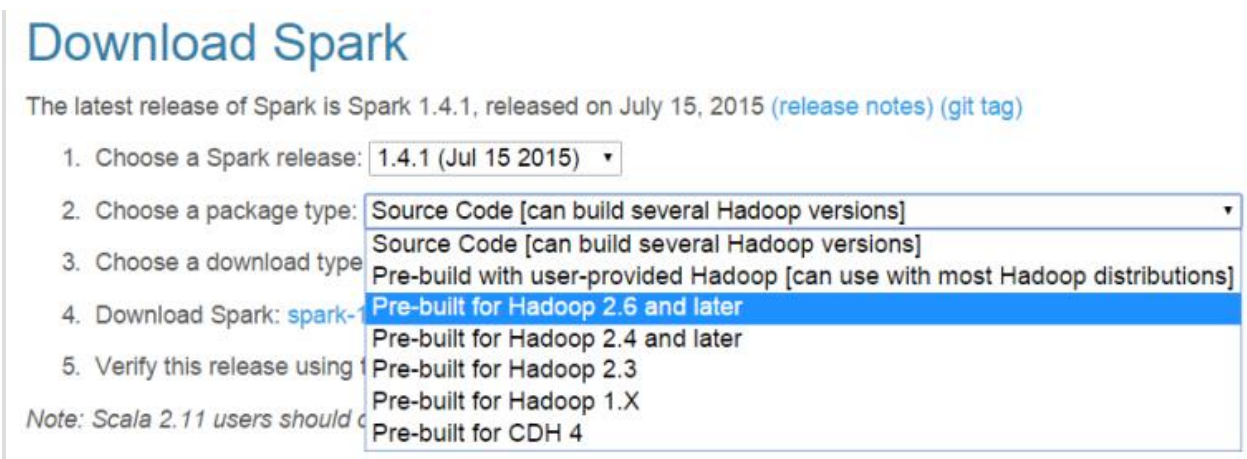


**Figure 3: Choose a package type**

On the *"3. Choose a download type"* option, select "Direct Download" from the drop-down list (Figure 4).


**Figure 4: Choose a download type**

After selecting the download type, a link is created next to the option *"4. Download Spark"* (Figure 5). Click this link to download Spark.

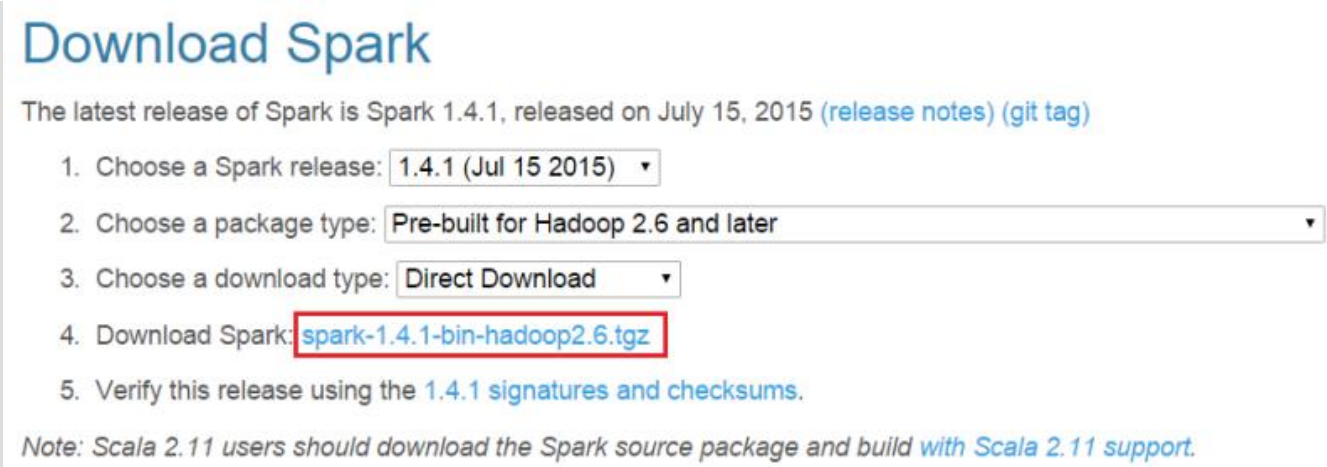


**Figure 5: Click the download link**

Save the zipped file to your computer (Figure 6).

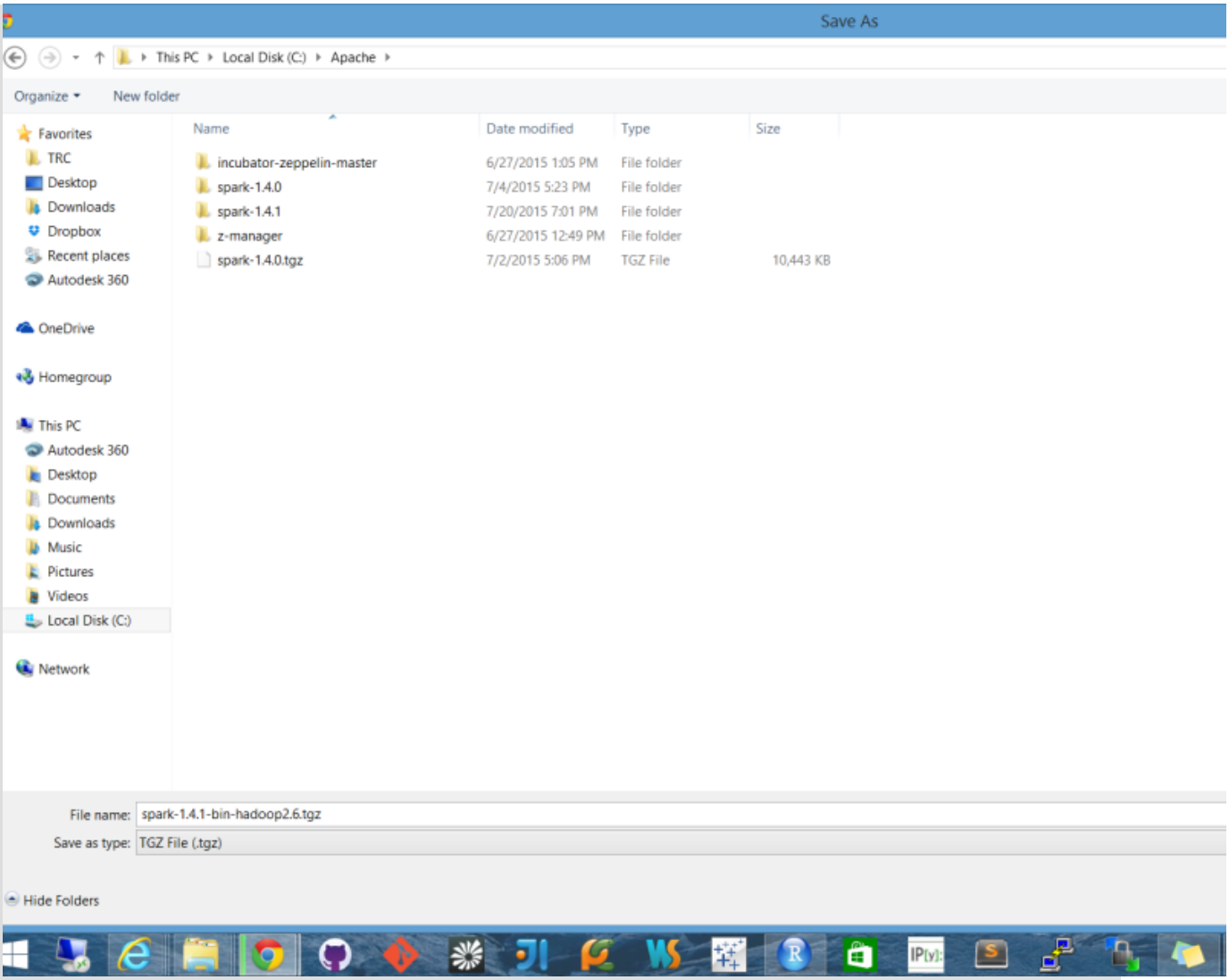


**Figure 6: Save to your computer**

### Step 2: Unzip Built Package

Unzip and save the files to a directory folder of your choice. In Figure 7 below, I chose to save to *"C:/Apache/Spark-1.4.1"*.

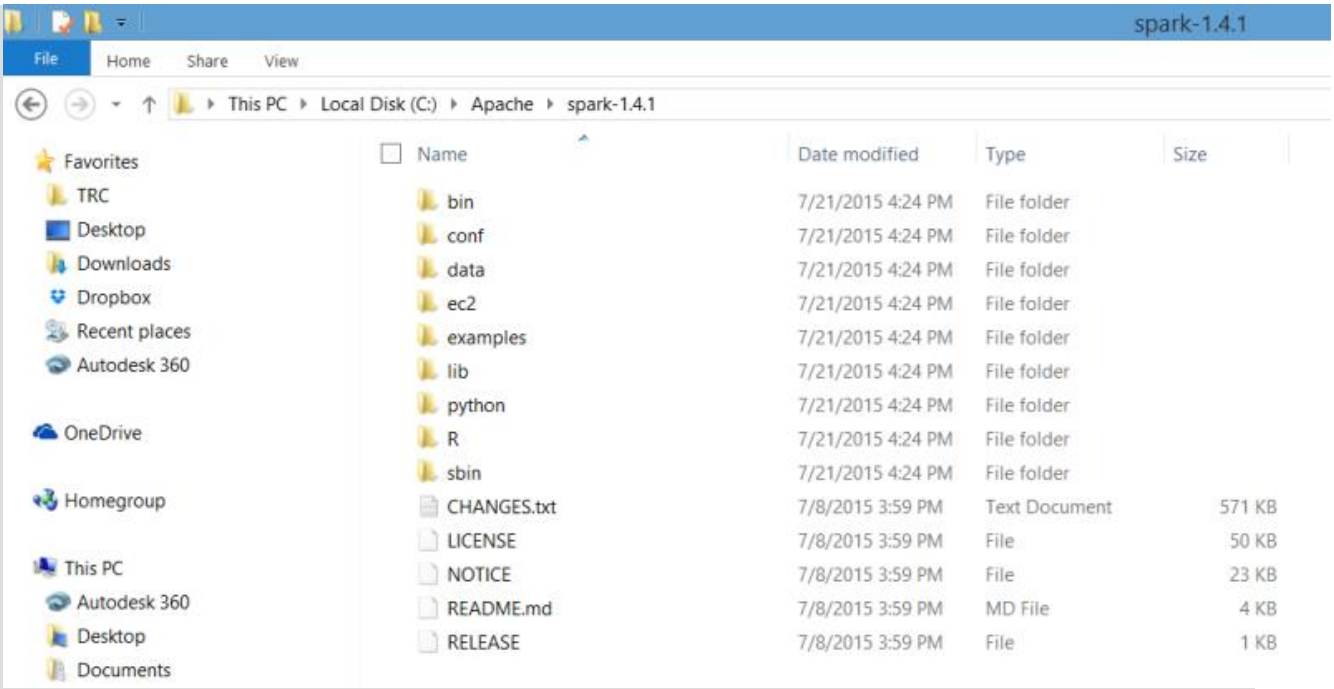


**Figure 7: List of Files in unzipped folder**

### Step 3: Run in Command Prompt

Now start your favorite command shell and change directory to your Spark folder as shown in Figure 8.

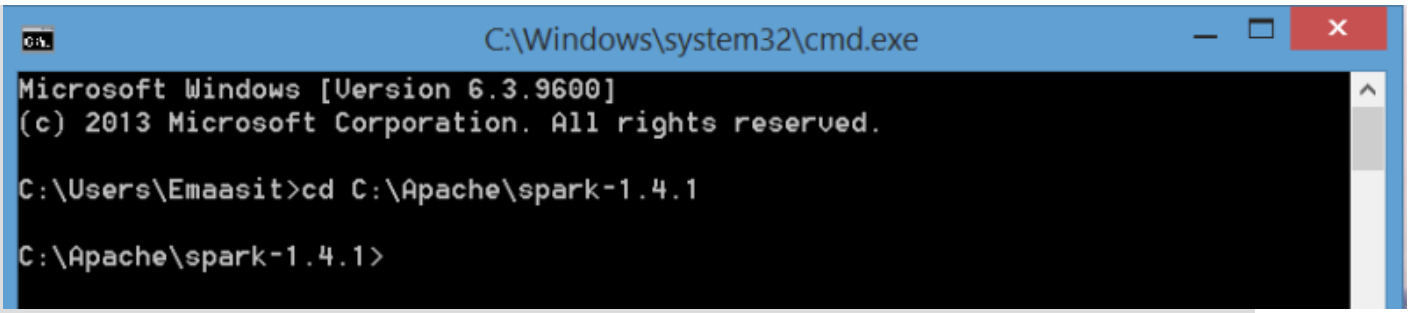


**Figure 8: Start command prompt and change directory**

To start SparkR, simply run the command `".binsparkR"` on the top-level Spark directory as shown in Figure 9 below.


**Figure 9: Start SparkR**

You will see logs on your screen that should take at most 15 seconds to launch SparkR. If everything ran smoothly you should see a welcome message that reads “Welcome to SparkR!” as shown in Figure 10.

```
15/07/21 17:01:20 INFO SparkUI: Started SparkUI at http://192.168.56.1:4040
15/07/21 17:01:20 INFO Executor: Starting executor ID driver on host localhost
15/07/21 17:01:20 INFO Utils: Successfully started service 'org.apache.spark.net
work.netty.NettyBlockTransferService' on port 62447.
15/07/21 17:01:20 INFO NettyBlockTransferService: Server created on 62447
15/07/21 17:01:20 INFO BlockManagerMaster: Trying to register BlockManager
15/07/21 17:01:20 INFO BlockManagerMasterEndpoint: Registering block manager loc
alhost:62447 with 265.4 MB RAM, BlockManagerId(driver, localhost, 62447)
15/07/21 17:01:20 INFO BlockManagerMaster: Registered BlockManager

 Welcome to SparkR!
 Spark context is available as sc, SQL context is available as sqlContext
>
>
```

**Figure 10: Welcome to SparkR!**

At this point you are ready to start prototyping with SparkR on the command shell. Note that a Spark context and a SQL Context have been initialized for you as "*sc*" and "*sqlContext*" respectively. You can now start experimenting using the example shown in **Step 4.5**.

**Running in RStudio**

While using SparkR in the command shell is good for quickly getting started, most R users typically use an Integrated Development Environment (IDE) like RStudio for development and running production ready code. Step 4 below will guide you to get started using SparkR in RStudio.

**Step 4: Run in RStudio**

- **Step 4.1: Set System Environment**

Once you have opened RStudio, you need to set the system environment first. You have to point your R session to the installed version of SparkR. Use the code shown in Figure 11 below but replace the ***SPARK_HOME*** variable using the path to your Spark folder. Mine is "C:/Apache/Spark-1.4.1″.

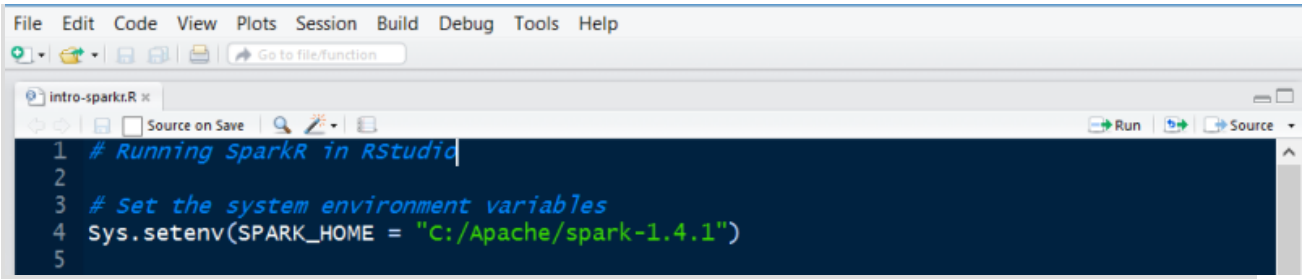


**Figure 11: Set System Environment Variable**

- **Step 4.2: Set the Library Paths**

Second, you have to set the library path for Spark a shown in Figure 12 below.

```
# Set the library path
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))
```

**Figure 12: Set the Library Path**

- **Step 4.3: Load SparkR Library**

Next, you can now load SparkR just as you would any other R library using the `library()` command as shown in Figure 13.

```
# Load the SparkR library
library(SparkR)
```

**Figure 13: Load the SparkR library**

- **Step 4.4: Initialize Spark Context and SQL Context**

Initialize SparkR by creating a Spark context using the command `sparkR.init()`. The argument in this command is *master = "local[N]"*, where *N* stands for the number of threads that you want to use.

Also, you need to create a SQL context to be able to work with DataFrames (the main abstraction in SparkR). Use the command `sparkRSQL.init()` to create a SQL context from your Spark context as shown in Figure 14.

```
# Create a spark context and a SQL context
sc <- sparkR.init(master = "local")
sqlContext <- sparkRSQL.init(sc)
```

**Figure 14: Initialize Spark Context and SQL Context**

When you run the above commands (From step 4.1 to 4.4), this invokes the "spark-submit" script that launches java, as shown in Figure 15. If this runs successfully, your Spark context and SQL context should be created and at this stage you should be able to start experimenting with SparkR.

```
> #load the Sparkr library
> library(SparkR)

Attaching package: 'SparkR'

The following object is masked from 'package:stats':

    filter

The following objects are masked from 'package:base':

    intersect, sample, table

> # Create a spark context and a SQL context
> sc <- sparkR.init(master = "local")
Launching java with spark-submit command C:/Apache/spark-1.4.1/bin/spark-submit.cmd
sparkr-shell C:\Users\Emaasit\AppData\Local\Temp\RtmpQdRVBW\backend_port20886132bb9
> sqlContext <- sparkRSQL.init(sc)
>
```

**Figure 15: All set**

- **Step 4.5: A Quick Example**

You can start experimenting with SparkR on the command shell and in RStudio using the example provided below. You can monitor your Spark jobs using the Spark UI at localhost:4040

```
# Set the system environment variables
Sys.setenv(SPARK_HOME = "C:/Apache/spark-1.4.1")
.libPaths(c(file.path(Sys.getenv("SPARK_HOME"), "R", "lib"), .libPaths()))
#load the Sparkr library
library(SparkR)
# Create a spark context and a SQL context
sc <- sparkR.init(master = "local")
sqlContext <- sparkRSQL.init(sc)
#create a sparkR DataFrame
DF <- createDataFrame(sqlContext, faithful)
head(DF)

# Create a simple local data.frame
localDF <- data.frame(name=c("John", "Smith", "Sarah"), age=c(19, 23, 18))
# Convert local data frame to a SparkR DataFrame
df <- createDataFrame(sqlContext, localDF)
# Print its schema
printSchema(df)
# root
#  |-- name: string (nullable = true)
#  |-- age: double (nullable = true)
# Create a DataFrame from a JSON file
path <- file.path(Sys.getenv("SPARK_HOME"), "examples/src/main/resources/people.json")
peopleDF <- jsonFile(sqlContext, path)
printSchema(peopleDF)
# Register this DataFrame as a table.
registerTempTable(peopleDF, "people")
# SQL statements can be run by using the sql methods provided by sqlContext
teenagers <- sql(sqlContext, "SELECT name FROM people WHERE age >= 13 AND age <= 19")
```

```
# Call collect to get a local data.frame
teenagersLocalDF <- collect(teenagers)
# Print the teenagers in our dataset
print(teenagersLocalDF)
# Stop the SparkContext now
sparkR.stop()
```

**view rawintro-sparkr.R** hosted with by **GitHub**

**Final Remarks**

The purpose of this blog post was to get you up and running quickly with SparkR locally on a personal computer. In the next blog post, I will show you how to use SparkR on a cloud computing framework like Amazon Elastic Compute Cloud (EC2) to manipulate large datasets with millions of records.

Tagged: Apache Spark, Big Data, R, RStudio, SparkR Comments: 8