**Spark, SparkR, and Sparklyr**

Randall Reese

**Introduction.**

Spark is an open-source cluster-computing framework distributed by Apache Software Foundation. Spark falls into the same family of large scale computational tools such as Hadoop and MapReduce. One of the larger advertised benefits of Spark over something such as Hadoop (also distributed by Apache) is that Spark can achieve run-times 10, 20, even 100 times faster than the run-time for the same task completed using Hadoop.

In this document, I will mainly focus on using Spark in R. Spark can also be used in Java, Scala, and Python.

We will answer the following questions in our report:

1. Does it help with storage, access, processing, modelling or visualization of data?

2. What does it do? (This will mostly be demonstrated in R, using the package **sparklyr**).

3. What other tools is it related to and how?

4. Is there a corresponding R package?

5. Provide a link where people could find out more about the tool.

**What aspect of big data is Spark used in?**

The main application of Spark is in processing “big” data sets that are stored in a distributed system (i.e. cluster computing). Holistically, Spark acts as an interface for managing, transforming, and processing data items distributed over a cluster of machines. These distributed data sets that Spark works with are commonly referred to as resilient distributed datasets (RDDs). Here, the term “resilient” means that RDDs record the transformations used to create the current data set at hand (also known as lineage caching). Thus, if a node of the distributed system fails, the RDD has enough information to reconstruct the lost partition of data. Ultimately, however, the minutia of RDDs is not a must-know topic for beginning (or even probably intermediate) users of Spark. For further details on RDDs, see [1] (written by the original developers of Spark).

**What tools are related to Spark?**

As previously discussed, Spark is a tool similar unto MapReduce, Hadoop, and other tools for distributed computing. Beyond availability as a standalone tool, Spark interfaces with a large variety of technologies (some well-known, others not so much): Apache Mesos, Hadoop (and HDFS for RDDs), Cassandra, HBase, Hive, and Amazon S3. Conceptually, Spark acts as alink between a RDD and SQL or **dplyr** (the R package) in that it allows users to select specific rows or columns of a data set based on certain attributes or conditions. (See example code below).

**Using Spark in R: SparkR and sparklyr.**

Spark can be used in R using one of either two packages. The first of these, **SparkR**, is distributed and managed directly by Apache and is discussed in [3]. The second of these packages, **sparklyr**, is distributed via CRAN and will be discussed below. For beginning users of Spark who are nonetheless decently familiar with R (especially **dplyr)**, **sparklyr** will feel more familiar and intuitive, hence our choice of demonstrating its use here. Moreover, **sparklyr** tends to interface better with other CRAN packages. Advanced users of Spark wishing to apply user-defined functions in R to RDDs managed via a Spark connection can implement such usage in **SparkR**.For a discussion on the advantages and disadvantages of each package, see [4].

Before going through some sample R code for **sparklyr**, be warned that users should ideally have the newest version of R, RStudio, and Java installed on their machine. Issues initializing Spark commonly center around one of these three things being out-of-date.

Below I provide some basic example code to get users started in using **sparklyr**. Users can also visit the link at [5].

**R Code for sparklyr.**

#####Taken in main part from http://spark.rstudio.com/  
###install.packages("sparklyr")  
###install.packages("digest")  
  
library(sparklyr)  
library(dplyr)

If you do not have a local version of Spark installed, install it using the line below:

spark\_install(version = "1.6.2")##Verify if you have the newest version.

To upgrade to the very latest version of sparklyr, run the following code. Note that this will require first having installed devtools.

devtools::install\_github("rstudio/sparklyr")

For our purposes, we are going to work with a local Spark connection:

sc <- spark\_connect(master = "local")

If you had access to a remote RDD for Spark, you could access it using code such as the following:

cluster\_url <- system('cat /root/spark-ec2/cluster-url', intern=TRUE)  
 sc <- spark\_connect(master = cluster\_url)

Copy some locally available R data sets over to the Spark cluster.

###install.packages("Lahman")#Sean 'Lahman' Baseball Database  
library(Lahman)  
  
iris\_tbl <- copy\_to(sc, iris)###iris is native to R.   
batting\_tbl <- copy\_to(sc, Lahman::Batting, "batting")  
src\_tbls(sc)####What do we have on the Spark Cluster?

The glorious thing now for beginning Spark users is that we can do much of our manipulation of the data set via dplyer. For example, we can select all players from the Atlanta Braves who have had at least 20 hits in a season.

batting\_tbl %>%  
 select(teamID, playerID, yearID,H) %>%  
 filter(teamID == "ATL") %>%  
 arrange(playerID, yearID) %>%  
 group\_by(playerID) %>%  
 filter(H > 20)  
  
# Source: query [731 x 4]  
# Database: spark connection master=local[8] app=sparklyr local=TRUE  
# Groups: playerID  
#   
# teamID playerID yearID H  
# <chr> <chr> <int> <int>  
# 1 ATL aaronha01 1966 168  
# 2 ATL aaronha01 1967 184  
# 3 ATL aaronha01 1968 174  
# 4 ATL aaronha01 1969 164  
# 5 ATL aaronha01 1970 154  
# 6 ATL aaronha01 1971 162  
# ... with 724 more rows

This ability to use dplyr allows us to work on a dataset without having to get too messy actually using Spark directly.

We can also do basic SQL queries on the databases via Spark:

library(DBI)  
iris\_preview <- dbGetQuery(sc, "SELECT Sepal\_Width FROM iris WHERE Sepal\_Width > 4 ")  
iris\_preview ###We only have three samples with Spepal\_Width > 4  
  
# Sepal\_Width  
# 1 4.4  
# 2 4.1  
# 3 4.2

**References**

Reference [1] deals with RDDs. Reference [2] is a general guide for manipulating data in Spark. This guide does not include material for using Spark with R, however it does cover usage of Spark in Scala, Java, and Python. Thus [2] acts as a good reference for those desiring to employ Spark in a setting outside of R. Reference [3] introduces users to the **SparkR** package. Reference [4] is written by the creator of **sparklyr**, so the author is naturally biased towards **sparklyr**. Nevertheless, I feel like this link explains the difference between the two Spark packages in R quite accurately. Reference [5] provides a tutorial on using **sparklyr**.

[1] “Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing”(Matei Zaharia, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker, Ion Stoica)*Found at:* <https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final138.pdf>

[2] <http://spark.apache.org/docs/latest/programming-guide.html>

[3] <http://spark.apache.org/docs/latest/sparkr.html>

[4] <https://github.com/rstudio/spark.rstudio.com/blob/master/_drafts/faq.Rmd>

[5] <http://spark.rstudio.com/>