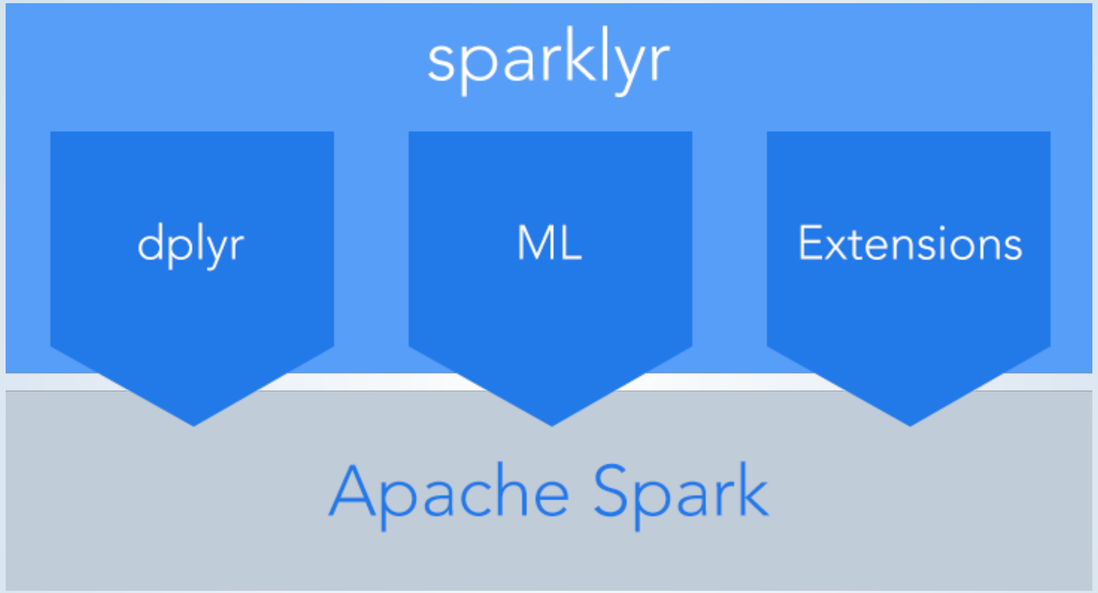


SPARKLYR

USING SPARK WITH RMARKDOWN

RStudio

Analyzing data with R

Laptop

1's Gigabytes

Server

10's Gigabytes

Cluster

100+ Gigabytes

Apache Spark

Fast and general engine for large-scale data processing

- *Can integrate with the Hadoop ecosystem*
- *Supports Spark SQL (HiveQL)*
- *Built-in machine learning*
- *Designed for performance*
- *Extensible*

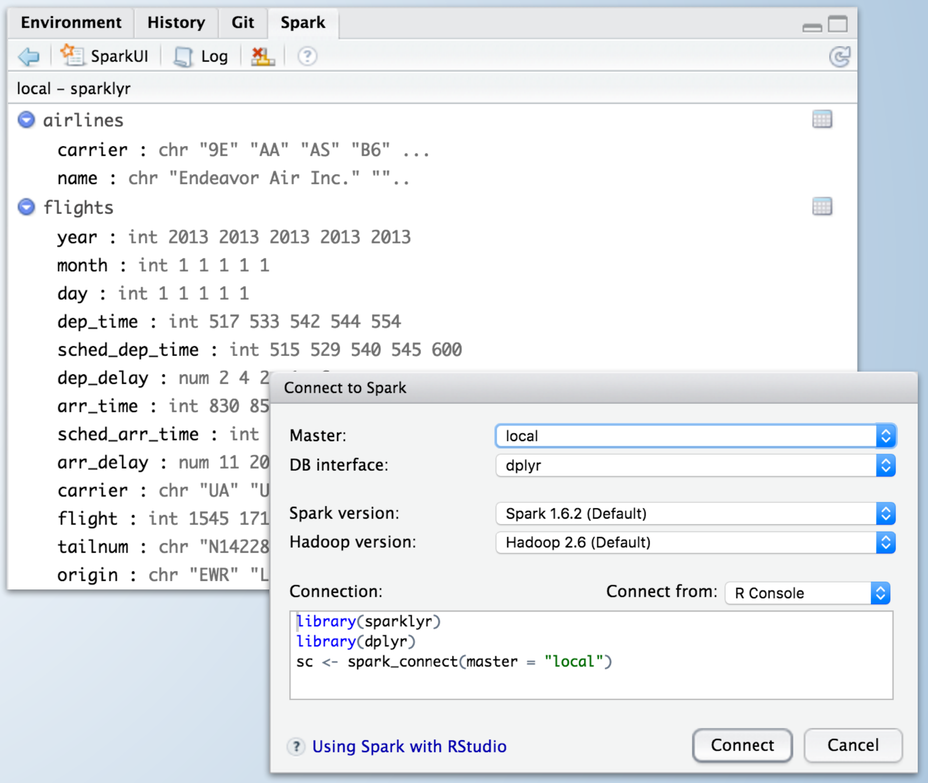
Spark

sparklyr

http://spark.rstudio.com/

New! Open-source R package from RStudio

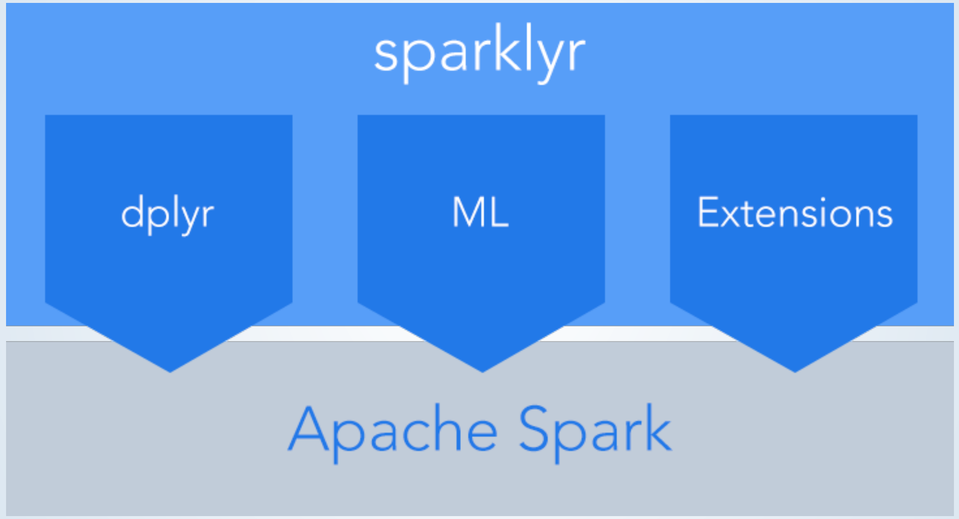
- *Integrated with the RStudio IDE*
- *Sparklyr is a dplyr back-end for Spark*
- *Extensible foundation for Spark applications and R*



Using R with Spark

If you are investing in Spark, then there is nothing stopping you from using it with the full power of R!

sparklyr Interface for Apache Spark



Use dplyr to write spark sql

A fast, consistent tool for working with data frame like objects, both in memory and out of memory.

```
my_tbl %>%
 filter(Petal_Width < 0.3) %>%
 select(Petal_Length, Petal_Width)
```

R Notebooks

Interactive code chunks and inline output

R Markdown from RStudio

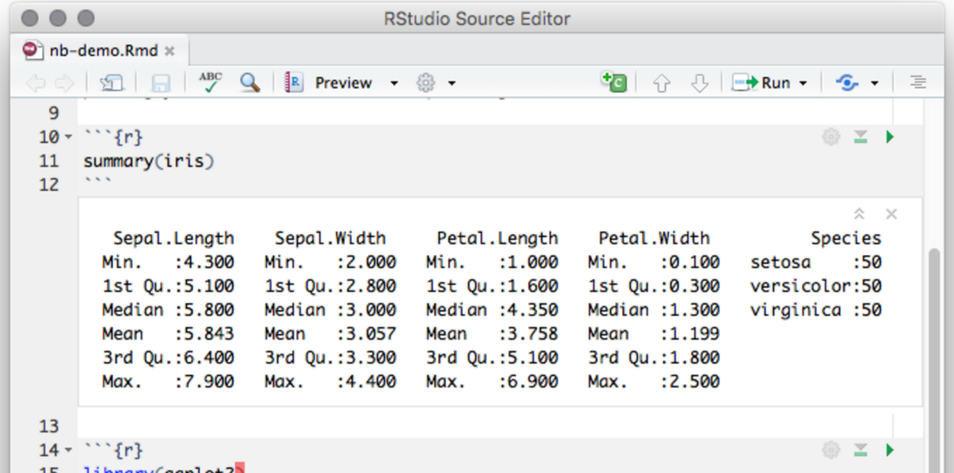
Get Started | Gallery | Formats | Articles

R Notebooks

NOTE: R Notebooks are new feature of RStudio, and are currently available only in the RStudio Preview Release. If you want to try out the features described below please install the preview release.

Overview

An R Notebook is an R Markdown document with chunks that can be executed independently and interactively, with output visible immediate beneath the input.



sparklyr + dplyr

https://beta.rstudioconnect.com/content/2025/

Code

Manipulating Data with dplyr

Overview

dplyr is an R package for working with structured data both in and outside of R. dplyr makes data manipulation for R users easy, consistent, and performant. With dplyr as an interface to manipulating Spark DataFrames, you can:

- Select, filter, and aggregate data
- Use window functions (e.g. for sampling)
- Perform joins on DataFrames
- Collect data from Spark into R

Statements in dplyr can be chained together using pipes defined by the magrittr R package. dplyr also supports non-standard evalution of its arguments. For more information on dplyr, see the introduction, a guide for connecting to databases, and a variety of vignettes.

Flights Data

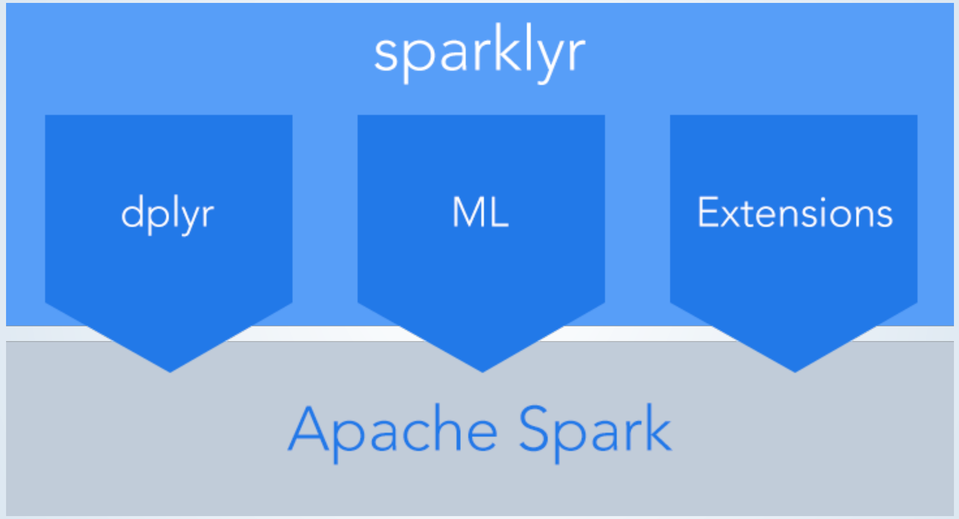
This guide will demonstrate some of the basic data manipulation verbs of dplyr by using data from the `nycflights13` R package. This package contains data for all 336,776 flights departing New York City in 2013. It also includes useful metadata on airlines, airports, weather, and planes. The data comes from the US Bureau of Transportation Statistics, and is documented in `?nycflights13`

Connect to the cluster and copy the flights data using the `copy_to` function. Caveat: The flight data in `nycflights13` is convenient for dplyr demonstrations because it is small, but in practice large data should rarely be copied directly from R objects.

Hide

```
library(sparklyr)
library(dplyr)
library(nycflights13)
library(ggplot2)
sc <- spark_connect(master = "local", version = "2.0.0")
flights <- copy_to(sc, flights, "flights")
airlines <- copy_to(sc, airlines, "airlines")
```

Spark MLlib and Spark Extensions



Spark MLlib

MLlib is Apache Spark's scalable machine learning library.

- *Easy to use*
- *Easy to deploy*
- *100x faster than MapReduce*

Extensions

sparklyr makes it really easy to invoke spark applications

Creating Extensions for sparklyr

Introduction

The sparklyr package provides a dplyr interface to Spark DataFrames as well as an R interface to Spark's distributed machine learning pipelines. However, since Spark is a general-purpose cluster computing system there are many other R interfaces that could be built (e.g. interfaces to custom machine learning pipelines, interfaces to 3rd party Spark packages, etc.).

The facilities used internally by sparklyr for its dplyr and machine learning interfaces are available to extension packages. This guide describes how you can use these tools to create your own custom R interfaces to Spark.

Examples

Here's an example of an extension function that calls the text file line counting function available via the SparkContext:

```
library(sparklyr)
count_lines <- function(sc, file) {
  spark_context(sc) %>%
    invoke("textFile", file, 1L) %>%
    invoke("count")
}
```

The `count_lines` function takes a `spark_connection` (`sc`) argument which enables it to obtain a reference to the `SparkContext` object, and in turn call the `textFile().count()` method.

You can use this function with an existing sparklyr connection as follows:

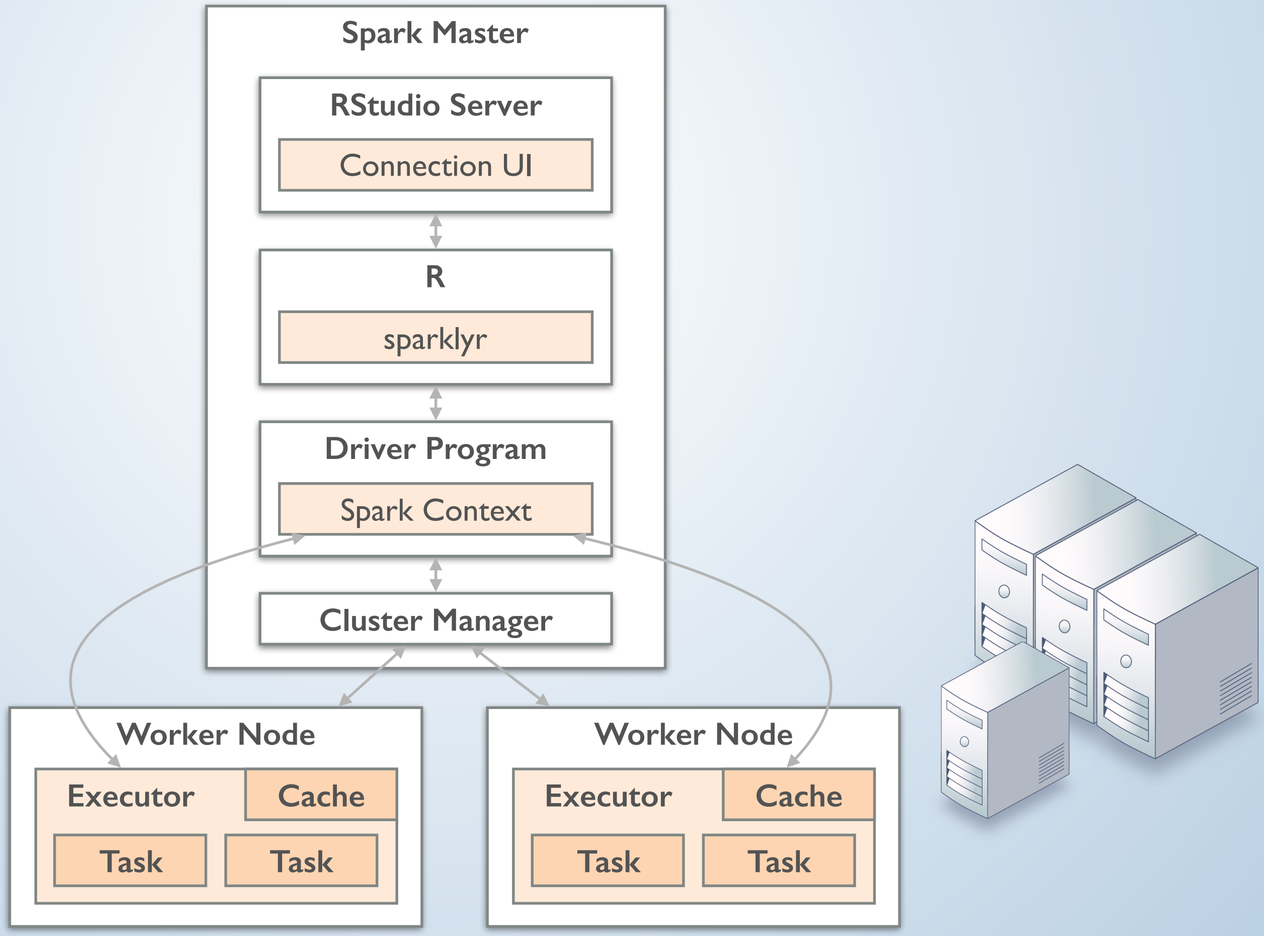
```
library(sparklyr)
sc <- spark_connect(master = "local")
count_lines(sc, "hdfs://path/data.csv")
```

Here are links to some additional examples of extension packages:

Spark Deployment

Cluster Mode

```
spark_connect("spark://spark.company.org:7077")
spark_connect(master = "yarn-client")
```



sparklyr + 1 billion records

https://beta.rstudioconnect.com/content/1705/

Code

Analyzing a billion NYC taxi trips in Spark

Access your data

We analyze the full taxi data as described by Todd Schneider in using R and sparklyr. We load the billion record trips table into Apache Spark and then use sparklyr and dplyr to manipulate the data and run machine learning algorithms at scale.

The data represent 200 GB of uncompressed data in CSV format. When converted and compressed in the parquet format, the data are 70 GB. The data are stored in HDFS and pre-loaded in a Hive table.

The Hadoop cluster runs on Elastic Map Reduce (EMR) in AWS and has 12 worker nodes and one master node. The master node has R, RStudio Server Pro, and sparklyr loaded onto it.

Connect to spark

Use sparklyr to create a new connection to Apache Spark.

Hide

```
# Load libraries
library(ggplot2)
library(leaflet)
library(geosphere)
library(tidyr)
library(shiny)
library(sparklyr)
library(dplyr)
library(miniUI)
library(DT)
# Configure cluster
Sys.setenv(SPARK_HOME="/usr/lib/spark")
config <- spark_config()
config$spark.driver.cores   <- 32
config$spark.executor.cores <- 32
config$spark.executor.memory <- "40g"
# Connect to cluster
sc <- spark_connect(master = "yarn-client", config = config, version = '1.6.1')
```

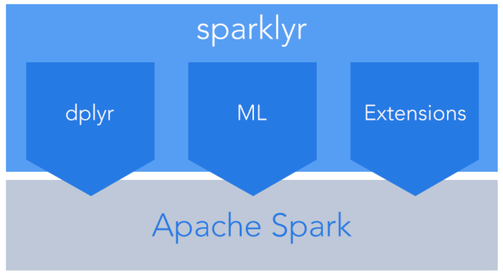
Questions

http://spark.rstudio.com/

Home | dplyr | ML | Extensions | Deployment | Examples | Reference

sparklyr — R interface for Apache Spark

- Connect to Spark from R — the sparklyr package provides a complete dplyr backend.
- Filter and aggregate Spark datasets then bring them into R for analysis and visualization.
- Orchestrate distributed machine learning from R using either Spark MLlib or H2O Sparkling Water.
- Create extensions that call the full Spark API and provide interfaces to Spark packages.



Installation

You can install **sparklyr** from CRAN as follows:

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

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

```
library(sparklyr)
spark_install(version = "1.6.2")
```

If you use the RStudio IDE, you should also download the latest preview release of the IDE which includes several enhancements for interacting with Spark (see the RStudio IDE section below for more details).