# Tracking machine learning models in R with MLflow

In this [story](https://medium.com/@dsml4real/first-impression-on-mlflow-406ab976f22e) I have briefly described what [**MLflow**](https://mlflow.org/) is and how it works. MLflow currently provides APIs in Python language that users can invoke in their machine learning source codes to log parameters, metrics, and artifacts to be tracked by the MLflow tracking server.

Users familiar with R and perform machine learning operations in R may like to track their models and every runs with MLflow. There are several approaches users can take.

* Waiting for [Mlflow](https://github.com/databricks/mlflow) to release the APIs in R, or
* Wrapping MLflow RESTful APIs and logging through curl commands, or
* Calling existing Python APIs with some R packages that can invoke Python interpreter

The last approach is simple and easy enough while allows users to interact with MLflow without waiting for R APIs to be available. I will illustrate how to achieve this with R package [**reticulate**](https://github.com/rstudio/reticulate).

reticulate is an open source R package that allows to call Python from R by embedding a Python session within the R session. It provides seamless and high-performance interoperability between R and Python. The package is available in [CRAN repository](https://cran.r-project.org/web/packages/reticulate/index.html).

Before beginning, you should have Python installed on the environment where R is running. I prefer installing [miniconda](https://conda.io/miniconda.html).

Once the Python is installed, you can create a virtualenv for MLflow and install [mlflow](https://pypi.org/project/mlflow/) package as follow (with conda):

conda create -q -n mlflow python=3.6

source activate mlflow

pip install -U pip

pip install mlflow

Next install [reticulate](https://github.com/rstudio/reticulate) package through R.

install.packages("reticulate")

reticulate allows R to call Python functions seamlessly. The Python package is loaded by the import statement. Calling to a function is through $ operator.

> library(reticulate)

> path <- import("os.path")

> path$isdir("/tmp")

[1] TRUE

As you can see above, it is very simple to call Python functions in os.path module from R with this package. So you can do the same thing with mlflow package by importing it and then call mlflow$log\_param and mlflow$log\_metric to log parameters and metrics for the R script.

Following R script builds a linear regression model with [SparkR](https://spark.apache.org/docs/latest/sparkr.html). You need SparkR package installed for this [example](https://github.com/adrian555/DocsDump/raw/dev/files/mlflow-R/mlflow-r.R).

# load the reticulate package and import mlflow Python module

library(reticulate)

mlflow <- import("mlflow")

# load SparkR package and start spark session

library(SparkR, lib.loc = c(file.path(Sys.getenv("SPARK\_HOME"), "R", "lib")))

sparkR.session(master="local[\*]")

# convert iris data.frame to SparkDataFrame

df <- as.DataFrame(iris)

# parameter for GLM

family <- c("gaussian")

# log the parameter

mlflow$log\_param("family", family)

# fit the GLM model

model <- spark.glm(df, Species ~ ., family = family)

# exam the model

summary(model)

# path to save the model

model\_path <- "/tmp/mlflow-GLM"

# save the model

write.ml(model, model\_path)

# log the artifact

mlflow$log\_artifacts(model\_path)

# stop spark session

sparkR.session.stop()

You can either copy the script to R or [Rstudio](https://www.rstudio.com/) and run interactively, or save it to a file and run with Rscript command. Make sure that the PATH environment variable includes the path to the mlflow Python virtualenv.

Once the script finishes, go to MLflow UI, the run is now showing and so it can be tracked. Here is a snapshot.

In conclusion, this approach lets R users take benefit of MLflow Tracking component and track their R models in a quick way. I will show how R users can use the other two components (Projects and Models) of MLflow in future stories.