Alex Motter
# R Server Best Practices

## High Level Overview

There are performance limitations to loading big data in RStudio, regardless of whether the server or a VDI are used. Files over 1GB in size (1.43 precisely) present issues as files larger than this size cannot be sent from a local R session to the server session. Below are my findings as well as best practices for using an R server on computing-heavy projects.

## Reading Data into Server Environment

This process depends on whether or not the desired file is greater than or equal to the 1.43GB number. It can be assumed, however, that there are many instances in which files will be, as the computing power of an R server is necessary over that of a local RStudio instance. In this project use case, the data source contained 8921483 observations of 83 fields, and attempting to read in the file was halted once the threshold value was reached.

Regardless of data source size, the first step is connecting with the R Server using the remoteLogin function (Note: for explanations for the functions and packages referenced in the following sections, please section 5 “Helpful Packages”)

If the file is less than 1GB, it can be read into RStudio local environment using read.csv, fread, or another read function. After pausing the server session, passing the putLocalObject function will send the data frame to the server session. Once resume() is executed, the data source will be available under the same name as that given by the putLocalObject call.

In cases like this where the file is greater than or equal to 1.43 GB in size, the file will need to be saved in the C drive of the server environment. If the full file path is used in the fread function, in this case fread("C:/Data/microsoft-malware-prediction/train.csv"), the data can be loaded into the server session’s working directory.

## Interacting With Server Data

Working with objects in the server session presents slight issues, as the Global Environment in the top right displays only objects in the VDI’s local session. However, these files can be viewed with a simple view() statement or by printing the head. In order to view the data source as a separate tab, the user must getRemoteObject to pull into the local session, which will fail if the file is larger than the aforementioned threshold.

## Saving Results of Server Session

As the requisite modeling is conducted on the data frame, users will ultimately need to save the results of their efforts. If the desired file is greater than or equal to 1.43GB, i.e. a cleaned data set, the data frame needs to be saved in the same server folder in which the original dataset was placed. Using the write.csv function with the full specified file path will write the data source to the same folder, where it can be retrieved and utilized for future modeling.

If the size is less than 1GB, however, the mrsdeploy package contains useful functions which allows users to locally save a copy of this data source themselves. In this example, I created a table which listed each of the data frame’s fields and a count of the NAs contained within them, an 80x2 table in total. After pausing the server session using pause(), the getRemoteObject function grabs an object from the workspace of the remote R session and loads it into the workspace of the local R session. From here, write.csv can easily save the object to a VDI.

## Helpful Packages

data.table: Contains functions specifically for dealing with big data and reading large data files

Useful Functions:
**fread()** Efficiently reads large delineated text files of sizes larger than the capabilities of the base R's read.csv. If certain columns are already known to be unnecessary, in my use case 3 of the columns containing more than 98% NA values, the 'drop' argument allows for the omitting of columns before data is read in.

mrsdeploy: Used to not only connect to R Server but also communicate between server environment and local environment.

Useful Functions:
**remoteLogin()** Allows the user to connect to a RServer instance using pre-specified login credentials

**pause(), resume(), and exit** Navigation functions which allow the user to pause their server session to work in the local environment, resume working off the server, or log out of the server session respectively.

**getRemoteObject()** Allows the user to grab an object from the workspace of the remote R session and loads it into the workspace of the local R session

**putLocalObject()** Allows the user to place an object from the workspace of the local R session it into the workspace of the remote R session

More information about using mrsdeploy with R Server enviorments can be found at: https://docs.microsoft.com/en-us/machine-learning-server/r-reference/mrsdeploy/mrsdeploy-package