**Intro:**

For a long time I have found it difficult to appreciate the benefits of “cloud compute” in my R model builds. This was due to my initial lack of understanding and the setting up of R on cloud compute environments. When I noticed that AWS was bringing out a new product [AWS Sagemaker](https://aws.amazon.com/sagemaker/), the possiblities of what it could provide seemed like a dream come true.

Amazon SageMaker provides every developer and data scientist with the ability to build, train, and deploy machine learning models quickly. Amazon SageMaker is a fully-managed service that covers the entire machine learning workflow to label and prepare your data, choose an algorithm, train the model, tune and optimize it for deployment, make predictions, and take action. Your models get to production faster with much less effort and lower cost. (<https://aws.amazon.com/sagemaker/>)

A question about AWS Sagemake came to mind: *Does it work for R developers???* Well…not exactly. True it provides a simple way to set up an R environment in the cloud but it doesn’t give the means to access other AWS products for example [AWS S3](https://aws.amazon.com/s3/) and [AWS Athena](https://aws.amazon.com/athena/) out of the box. However for Python this is not a problem. Amazon has provided a Software Development Kit (SDK) for Python called [boto3](https://boto3.amazonaws.com/v1/documentation/api/latest/index.html), which comes pre-installed on AWS Sagemaker.

It isn’t all bad news, RStudio has developed a package called [reticulate](https://rstudio.github.io/reticulate/) that lets R interfaced into Python. So using reticulate in combination with boto3 gives R full access to all of AWS products from Sagemaker similar to Python. However are there any other methods for R user to connect to AWS?

**AWS interfaces for R:**

[**paws**](https://paws-r.github.io/) **an R SDK:**

Paws is a Package for Amazon Web Services in R. Paws provides access to the full suite of AWS services from within R.(<https://github.com/paws-r/paws>)

When I want to connect to AWS I usually turn to Python. AWS’s boto3 is an excellent means of connecting to AWS and exploit its resources. However R now has it’s own SDK into AWS, paws. This came as a little surprise to me as I started to accept that R might never have an SDK for AWS. How wrong I was.

What’s pleasing to me was how well developed and easy the package was to use. It felt natural to switch between boto3 and paws. Almost like it was a long lost brother.

*Here is a quick example to show the comparison between boto3 and paws. Returning a list of all objects in S3 inside a prefix:*

**Python**

import boto3

s3 = boto3.Session().client("s3")

obj = s3.list\_objects(Bucket = 'mybucket', Prefix = "prefix\_1/")

[x.get("Key") for x in obj.get("Contents")]

**R**

s3 <- paws::s3()

obj <- s3$list\_objects(Bucket = 'mybucket', Prefix = "prefix\_1/")

lapply(obj$Contents, function(x) x$Key)

From this quick example it is clear that the paws SDK’s syntax is extremely similar to boto3, although with an R twist. This can only a good thing, as hundreds of people know boto3 already and therefore they will be familiar with paws by association. I can’t express the potential the package paws gives R users. A good project that utilises the paws sdk is the package [noctua](https://cran.r-project.org/web/packages/noctua/index.html). noctua creates a wrapper of the paws connection to AWS Athena and developes a DBI interface for R users. We will go into the package noctua in the next blog. First here is an example how of to work with AWS Athena when using paws.

*Querying to AWS Athena using paws*

# create an AWS Athena object

athena <- paws::athena()

# Submit query to AWS Athena

res <- athena$start\_query\_execution(

QueryString = "show Databases",

ResultConfiguration =

list(OutputLocation = "s3://mybucket/queries/"))

# Get Status of query

result <- athena$get\_query\_execution(QueryExecutionId = res$QueryExecutionId)

# Return results if query is successful

if(result$QueryExecution$Status$State == "FAILED") {

stop(result$QueryExecution$Status$StateChangeReason, call. = FALSE)

} else {output <-

athena$get\_query\_results(

QueryExecutionId = res$QueryExecutionId,

MaxResults = 1)}

From an initial view it might look daunting however this is exactly the same interface that boto3 provides when working with AWS Athena. The good news is that noctua wraps all of this and creates the DBI method dbGetQuery for paws.

paws is an excellent R SDK into AWS, so please download paws and give it ago, I am sure you will be pleasantly surprised like myself.

install.packages("paws")

**Note:** For more examples, the developers of paws have created some code examples <https://github.com/paws-r/paws/tree/master/examples> and a documentation website <https://paws-r.github.io/>.

[**botor**](https://daroczig.github.io/botor/) **:**

This R package provides raw access to the ‘Amazon Web Services’ (‘AWS’) ‘SDK’ via the ‘boto3’ Python module and some convenient helper functions (currently for S3 and KMS) and workarounds, eg taking care of spawning new resources in forked R processes. (<https://daroczig.github.io/botor/>)

When using botor on AWS Sagemaker, R users can easily interact with all of AWS products in the exact same manner as a Python user. However botor’s convenient helper functions certainly does make the experience working on AWS Sagemaker easier. Here is a quick example to demostrate how easy/ useful these helper function are:

*Upload iris data.frame to s3 bucket*

library(botor)

write\_s3(iris, data.table::fwrite, "s3://mybucket/iris.csv")

*Read s3 file back into R as a data.frame*

read\_s3("s3:://mybucket/iris.csv", data.table::fread)

These convenient helper functions are not limited to just reading/writing data in csv format. They can also be used to upload R models, which can be really useful when wanted to store pre-built models. Here is a quick example of what I like to call a *crap* model.

train <- iris[1:20,1:4]

test <- iris[21:40,1:4]

model <- lm(Petal.Width ~., train)

*Uploading and downloading R models to S3*

s3\_write(model, saveRDS, "s3://mybucket/crap\_model.RDS")

s3\_model <- s3\_read("s3://mybucket/crap\_model.RDS", readRDS)

It is clear to see how useful botor is when working with AWS S3.

**Cloudyr Project:**

I personally haven’t used the AWS cloudyr packages, however I don’t want to leave them out. The [cloudyr project](https://cloudyr.github.io/) aim is to bring R onto the cloud compute:

The goal of this initiative is to make cloud computing with R easier, starting with robust tools for working with cloud computing platforms.([https://cloudyr.github.io/](https://cloudyr.github.io/" \t "_blank))

As I haven’t utilised the wide range of packages that the cloudyr project provides I won’t give examples. Please go to the cloudyr github <https://github.com/cloudyr> as a lot of work has gone into making R easier to work with cloud computing. They have a lot of documentation plus they are actively developing R packages to make user experience better.

**Summary:**

I believe that all of these packages have advantages in working with AWS when using R. As R has a SDK paws for AWS it would be great if it was added to the base image, as it allows R developers to utilise AWS products in their AWS Sagemaker environments. Alternatively the botor package would be another package for AWS to consider putting in their AWS Sagemaker image.