**Intro:**

**paws and AWS Athena:**

Before getting into the next package, lets first look at how the SDK’s interact with AWS Athena.

*For example: return all databases in AWS Athena*

# 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 = 10)}

This isn’t the prettiest code when wanting to query AWS Athena with the SQL, in the above example: SHOW DATABASES. This example only returns the top 10 results. It is even more “interesting” if you wish to return the entire data frame from AWS Athena. This is where noctua comes in.

**noctua**

* **What is noctua?**

noctua is a R package that creates a DBI (Database Interface) for R, using the R package DBI and the R SDK paws as the backend (so basically the same as RAthena)

* **Why was noctua created when there are already methods for connecting to Athena?**

noctua was created to provide an extra method to connect to Athena for R users. Plus it seemed natural to create noctua due to the nature in how it connects to AWS Athena (through a SDK), which is the method RAthena connects to AWS Athena.

* **Why is noctua called noctua?**

This is a tricky one as RAthena was already taken. So I looked for a historic reference to link the new package to AWS Athena. I settled on noctua due to: [Athena/Minerva](https://en.wikipedia.org/wiki/Athena) is the Greek/Roman god of wisdom, handicraft, and warfare. One of the main symbols for Athena is the Owl. Noctua is the latin word for Owl.

**How to install:**

noctua is currently on the CRAN and Github:

*CRAN version:*

install.packages("noctua")

*Github development version:*

remotes::install\_github("dyfanjones/noctua")

**Usage:**

As with all DBI interface packages the key functions are exactly the same. Which means that there is little to no upskilling required. The only difference between each method is how they connect and send data back to the database. So we will focus mainly on those two aspects.

**Connecting to AWS Athena:**

noctua offers a wide range of connection methods from hard coding to using Amazon Resource Name Roles (ARN roles). Which is very similar to the RAthena package.

**Hard-Coding Method:**

This method isn’t recommended as your credentials are hard-coded.

library(DBI)

con <- dbConnect(noctua::athena(),

aws\_access\_key\_id = "YOUR AWS KEY ID",

aws\_secret\_access\_key = "YOUR SECRET ACCESS KEY",

s3\_staging\_dir = "s3://path/to/query/bucket/")

**Note:** *s3\_staging\_dir requires to be in the format of s3 uri for example “s3://path/to/query/bucket/”*

If you do not wish to create AWS Profiles then setting environmental variables would be the recommended method.

**Environment Variable Method:**

noctua supports [AWS credentials](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-envvars.html) when set into the environment variables to avoid hard-coding. From what I have found out, an easy way to set up environment variables (that persists) in R is to use the file.edit function like so:

file.edit("~/.Renviron")

And now you can simply add in your environment variables in the file you are editing for example:

AWS\_ACCESS\_KEY\_ID = YOUR AWS KEY ID

Once you have set your environment variables you can connect to Athena in the following method:

library(DBI)

con <- dbConnect(noctua::athena(),

s3\_staging\_dir = "s3://path/to/query/bucket/")

You can set the s3\_staging\_dir parameter as an environmental variable, to do this you need to set the following environmental variable:

AWS\_ATHENA\_S3\_STAGING\_DIR = s3://path/to/query/bucket/

This allows for the following connection:

library(DBI)

con <- dbConnect(noctua::athena())

**AWS Profile Names:**

Another method is to use AWS Profile Names. AWS profile names can be setup either manually in the ~/.aws directory or by using the [AWS Command Line Interface (AWS CLI)](https://docs.aws.amazon.com/cli/latest/userguide/cli-configure-profiles.html). Once you have setup your profile name you can connect to AWS Athena:

**Using Default Profile Name:**

library(DBI)

con <- dbConnect(noctua::athena())

**Using Non-Default Profile Name:**

library(DBI)

con <- dbConnect(noctua::athena(),

profile\_name = "rathena")

**ARN Roles:**

ARN roles are fairly useful if you need to assume a role that can connect to another AWS account and use the AWS Athena in that account. Or whether you want to create a temporary connection with different permissions than your current role ([AWS ARN Documentation](https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_cross-account-with-roles.html)).

**Assuming ARN role credentials before connecting to AWS Athena:**

library(noctua)

library(DBI)

assume\_role(profile\_name = "YOUR\_PROFILE\_NAME",

role\_arn = "arn:aws:sts::123456789012:assumed-role/role\_name/role\_session\_name",

set\_env = TRUE)

# Connect to Athena using ARN Role

con <- dbConnect(athena(),

s3\_staging\_dir = "s3://path/to/query/bucket/")

**Connect to AWS Athena directly using ARN role:**

library(DBI)

con <- dbConnect(noctua::athena(),

profile\_name = "YOUR\_PROFILE\_NAME",

role\_arn = "arn:aws:sts::123456789012:assumed-role/role\_name/role\_session\_name",

s3\_staging\_dir = 's3://path/to/query/bucket/')

**Note:** *ARN Roles have a duration timer before they will expire. To change the default you can increase the duration\_seconds parameter from the default 3600 seconds (1 hour).*

**Temporary Sessions:**

Finally you can create temporary credentials before connecting to AWS Athena:

library(noctua)

library(DBI)

# Create Temporary Credentials duration 1 hour

get\_session\_token("YOUR\_PROFILE\_NAME",

serial\_number='arn:aws:iam::123456789012:mfa/user',

token\_code = "531602",

set\_env = TRUE)

# Connect to Athena using temporary credentials

con <- dbConnect(athena(),

s3\_staging\_dir = "s3://path/to/query/bucket/")

**Querying:**

To query AWS Athena using the noctua it is very similar to querying any other DBI database method:

library(DBI)

con <- dbConnect(noctua::athena())

dbGetQuery(con, "show databases")

That is it! So if we look back at the initial paws code when working with AWS Athena. The code was very intimidating when wanting to do basic AWS Athena queries. noctua packages all that up and makes it super easy to work with.

**Uploading Data:**

It is all very well querying data from AWS Athena but what is more useful is to upload data as well. noctua has addressed this and implemented a method in dbWriteTable.

dbWriteTable(con, "mtcars", mtcars,

partition=c("TIMESTAMP" = format(Sys.Date(), "%Y%m%d")),

s3.location = "s3://mybucket/data/")

Once you have uploaded data into AWS Athena you can query it in the following:

dbGetQuery(con, "select \* from mtcars")

Here are all variable parameters for the dbWriteTable method:

**conn:** An AthenaConnection object, produced by dbConnect()

**name:** A character string specifying a table name. Names will be automatically quoted so you can use any sequence of characters, not just any valid bare table name.

**value:** A data.frame to write to the database.

**overwrite:** Allow overwriting the destination table. Cannot be ‘TRUE’ if ‘append’ is also ‘TRUE’.

**append:** Allow appending to the destination table. Cannot be ‘TRUE’ if ‘overwrite’ is also ‘TRUE’.

**row.names:** Either TRUE, FALSE, NA or a string. If TRUE, always translate row names to a column called “row\_names”. If FALSE, never translate row names. If NA, translate rownames only if they’re a character vector. A string is equivalent to TRUE, but allows you to override the default name. For backward compatibility, NULL is equivalent to FALSE.

**field.types:** Additional field types used to override derived types.

**partition:** Partition Athena table (needs to be a named list or vector) for example: c(var1 = “2019-20-13”)

**s3.location** s3 bucket to store Athena table, must be set as a s3 uri for example (“s3://mybucket/data/“)

**Conclusion:**

noctua is a package that gives R users the access to AWS Athena using the R AWS SDK paws. Thus no external software is required and it can all be installed from the CRAN. If you are interested in how to connect R to AWS Athena please check out [RAthena](https://cran.r-project.org/web/packages/RAthena/index.html) as well (my other AWS Athena connectivity R package).