PRO TIPS

from RStudio Customer Success

RStudio

Title

Why You Should Use Pins

As a data scientist, it's likely that a recurring challenge is to easily share content with others in your organization. One of the most persistent problems is knowing how and where to store resources, especially resources that can't be easily put in a database for others to access without needing to email files back and forth - for example model outputs, csv files, and r objects.

This is where Pins come in. RStudio developed Pins to make the process of discovering, caching, and sharing resources simpler.

If you're always asking colleagues to download files before running your code, or are redeploying an entire app just to update the model or data in it, or if many pieces of your content are pointed to the same dataset, read on.

What are Pins?

Pins are almost exactly what they sound like: a way to store and then remotely access R and Python objects. You can take an object and "pin" to a number of boards, including RStudio Connect, S3, Google Cloud, even or integrate them to your website.

Once resources are pinned, you can also discover them. Pins let you search well-known data repositories that are guaranteed to contain datasets, instead of searching on the internet and hoping to find what you need, as well as enhancing the discoverability of internally-created datasets.

The use cases are too many to count, so in the next section, you'll learn how to pin your first resource using RStudio Connect. From there, you can check out this website for further inspiration on how to use Pins to make your day-to-day life easier. How to Use Pins There are four main steps to using Pins. First, install the package, get your R session to connect with your board, and then Pin a resource. Finally, you can get resources. Let's walk through each step using RStudio Connect as an example.

Before you can publish a pin to the RSC board, you have to get an API key from Connect and save this to your sys environment variables.

In RSC, generate an API key. You can name this something like "forusingpins". Copy the value to the clipboard. Now in R, you have to save this API key to your system environment variables. Since you want this to persist and not evaporate every time R restarts, write this in your .Rprofile. Think about if you want this to be in your Project .Rprofile file or your home directory .Rprofile depending on whether you'll use it just for this project, or for several.

Wherever you're setting it, you'll add a line like: `Sys.setenv("RSC_API_KEY" = "<Key Value>")`

Now you're going to register the board. This isn't initializing a new board, it's just telling your session that there's a place you can store resources. "Register" here means "I see you and know you're there."

Use `pins::board_register( "rsconnect", server = "https://colorado.rstudio.com/rsc/", key = Sys.getenv("RSC_API_KEY") )`

Now you're going to Pin your first resource. Put your object on the rsconnect board with `pin(my_data, description = "Super Cool data set", board = "rsconnect")`

Code to retrieve the pin will now be provided on the deployed content in RStudio Connect: `setsData <- pin_get("username/my_data", board = "rsconnect")`

You're ready to publish your Pin. The first time you publish, it will fail saying "[Connect] Message: 'Invalid API key, the API key is empty.' " because there is no key on the environment variable on connect. Go put RSC_API_KEY in environment variables and refresh.