Creating Efficient Workflows with `pins`

When your workflow is clunky

- Does your workflow contain `read.csv()` for a data file that your colleague emails to you after every update?
- Are you working with ephemeral data that refreshes frequently?
- Do you need to redeploy your app every time the supporting data is updated?
- Do multiple apps you've created need to call a model that you developed?
- Are you saving `.R` or `.RData` objects to be called later?

These types of workflows can slow you down unnecessarily. Data scientists can struggle to share and reuse content; knowing how and where to store resources, especially resources that can't be put into a database, is a persistent problem.

This is where `pins` come in. Here at RStudio, we developed `pins` to make the process of discovering, caching, and sharing resources simpler, all with the ultimate aim of helping you create efficient data workflows.

If you're asking your colleagues to download files before running your code, 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, you have a use case for `pins`.

What are `pins`?

These `pins` are pretty much what they sound like: they're a way to store and then remotely access R and Python objects, just like you'd pin a sticky note to a (virtual) cork board.

You can take an object, like a model output, and `pin` it to your board, which can be on RStudio Connect, S3, Google Cloud, or even your website. A `pin` is a resource that is relatively small (a few hundred megabytes at most), perhaps reused across multiple pieces of content, and only needed in their most current form.

Once you've `pin`ned a resource, other folks can discover them. This is great for two reasons: first, `pins` on sites like Kaggle 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. Secondly, `pins` also enhance the discoverability of internally-created datasets with the same access controls you're accustomed to for any content published to Connect, which is great for learning more about what your colleagues are up to.

In the next section, you'll learn how to `pin` your first resource using RStudio Connect. From there, you can check out this website https://pins.rstudio.com for further inspiration on how to use `pins` to make your day-to-day life easier.

Walk me through my first `pin`

You can think of this process in four main steps: First, you install the package. Second, get your R session to connect with your board. Third, and most excitingly, you Pin a resource. Finally, you can access resources that you or others have Pinned.

Let's walk through each step using RStudio Connect as an example.

Before you begin, make sure you have all the resources you'll need to hand. Go ahead and pick out a data set you want to pin. Make sure you have the API key you'll need to use, as well as Publisher permissions on Connect.

That's all you need to get started!

First step: go ahead and install the `pins` package.

```r
install.packages("pins")
```

Next, connect your R session to a board. Before you can publish your first pin to the RStudio Connect board, you have to get an API key from Connect and save this to your system environment variables. [1] This is the best practice, instead of hard coding the API key into your script. Remember, your keys should be kept secure! This is not as intimidating as it sounds - let's walk through the steps.

Generate an API key in RStudio Connect.[2] Give this key any name you like, like `RSC_API_KEY` and be certain to copy the value to your clipboard.

Save this API key to your system environment variables. You can do this by switching back to your RStudio IDE and run the following in the Console, pasting your API key value into the command:

```r
Sys.setenv("RSC_API_KEY" = "key value")
```

"Register" the board.[3] Add the following to your script, inserting your own server address:

```r
pins::board_register(
    "rsconnect",
    server = "https://rstudioconnect.example.com/",
    key = Sys.getenv("RSC_API_KEY")
    )
```

The third step: you're going to **pin** your first resource. Put your object on the rsconnect board with:

```r
pin(my_data, description = "Super cool data set",
    board = "rsconnect")
```

And now for the fourth, step, you can retrieve a **pin**.

Data science is all about sharing your findings with others, so let's get into how you can do this with **pins**.

If you head on over to Connect, you'll notice your **pin** has some useful header information. You can copy that code into your analysis, then assign the retrieved **pin** as a variable.

[1] API keys can differ from user to user – they're just a way to tell Connect that you're allowed to access it, whether you're Pinning or retrieving a Pin

[2] for full instructions, see https://docs.rstudio.com/connect/user/api-keys/

[3] Don't let the vocabulary throw you. This just means you're identifying a location where you can store resources

```r
# Register RStudio Connect
library(pins)
board_register("rsconnect",
 server = "https://yourconnectserver.example.com")

# Retrieve Pin
your_data <- pin_get("yourdata",
                     board = "rsconnect")

#Check out the pin
head(your_data)
```

The first time you publish content that calls a **pin**, it will fail saying "[Connect] Message: 'Invalid API key, the API key is empty.'" This is just because there is no key on the environment variable on Connect. All you need to do is to put RSC_API_KEY in environment variables and refresh.

That environment variable that you specify doesn't have to be the same API key that you created when you created a **pin** - for example, User A might create the pin, and User B might download it. The API keys are different per user, and are just a way of telling Connect that you're allowed to **pin** resources.

You have now **pin**ned and retrieved a **pin**!

Where do I go for help?

At this point, you know what a **pin** is, whether **pins** will be useful for your workflow, and how to get started with your first pin. What next?

If you want more hands-on advice with data you can play with yourself, we have a downloadable example that accompanies this Pro Tip, see:

Go try **pins** on your own!

We also have a ton of resources available for **pins** right at the **pins** website here: http://pins.rstudio.com/. Check out the source code here https://github.com/rstudio/pins. Any issues? Let us know here: https://github.com/rstudio/pins/issues.

Last but not least, let us know how you get on with pins! Get in touch with the Customer Success team at sales@rstudio.com.