PRO TIPS from RStudio Customer Success

Creating Efficient Workflows with 'pins' and RStudio Connect

When your Workflow is Clunky

Like that one piece of furniture in your living room that you *need*, but you just can't find a spot for, we often struggle to find a home for the mid-process artifacts of a data analysis.

Think about some of your workflows. Are you:

- Using `read_csv()` to bring in emailed data?
- Saving `.Rds` or `.RData` objects to be called later?
- Sharing your model or data across multiple apps?
- Redeploying your app every time the supporting data is updated?

A "yes" to any of the above indicates a solid use case for `pins`! Here at RStudio, we developed `pins` to make discovering, caching, and sharing resources simpler, all to promote efficient data workflows.

What are Pins?

📌 Just like you'd pin a note (or a picture of your BFF) to a physical cork board, `pins` lets you pin an R or Python object, or a file to a virtual board where you and others access it. The virtual board can be on RStudio Connect, Amazon S3, Azure storage, Microsoft 365 (OneDrive and SharePoint), or Google Cloud, to name a few. Pins can be automatically versioned so you can track changes, re-run analyses on historical data, and undo mistakes.

Pins are best suited for objects up to a few hundred megabytes in size. Often they're made of lightweight or ephemeral data, and multiple assets may rely on them. You can pin datasets, models, plots, files, and more!

It's typically much easier (and safer) to share a pin across multiple assets or across your team than to email files around. Further, if you're using RStudio Connect, you can manage who can access your pins.

In this Pro Tip, you'll learn how to pin and retrieve a resource on RStudio Connect and how to schedule updates to pins so downstream analyses are always current without re-deployment.

Requirements

To utilize `pins` with Connect make sure you:

1. Are a Publisher or Administrator on your Connect server.[1]
2. Have RStudio Connect v 1.7.8 or higher.[2]
3. Have a current version of `pins`. The `pins` package is available on CRAN, and was at version 1.0.0 at the time of this writing. Install `pins` with `install.packages("pins")`.

Got all that? Super! Let's get cooking! 🔍

Pinning to RStudio Connect

Recall that virtual cork board metaphor? Every pin lives on a board, so our first step is to **create a board object** so we can call it. There are multiple board types, but for this example, we will be using a board on RStudio Connect with the following:

```
board <- pins::board_rsconnect()
```

You'll note that we didn't pass any arguments to `board_rsconnect()`. This function is designed to *just work* for most cases. Should you run into issues, you can specify the `auth` method to inform how you'll authenticate to RStudio Connect. Available methods outside of the default of `auth = "auto"` are to specify `auth = "manual"` and provide the server and key arguments; `auth = "envvar"` if you have already defined `"CONNECT_SERVER"` and `"CONNECT_API_KEY"` as environment variables[3]; or `auth = "rsconnect"`, to use the server registered using the `rsconnect` package.

Now that we've got our board, it's time to pin something on it. Select an object and **pin to the rsconnect board with:**

```
my_data <- faithful #for example
pins::pin_write(board,
  my_data,
  name = "faithful_data")
```

Tip: think of `name` like a filename. You'll want to avoid slashes and spaces.

You can optionally specify `type` in the `pin_write()` function. The default is `type = "rds"`, but you can also pin `"csv"`, `"json"`, `"arrow"` for arrow/feather files, or `"qs"` which provides more efficient compression than RDS for faster read/writes.

With this, you've created your first pin! Congratulations! Be sure to adjust the **Access Settings** on your pin in the RStudio Connect content dashboard if you want to share this pin with others.

[1] Viewers can only retrieve pins

[2] If you're not yet at this version but are keen to get started with `pins`, talk to your R Admin and refer to https://docs.rstudio.com/connect/admin/server-management/#upgrading

[3] Easily set environment variables with `usethis::edit_r_profile()` to open your .RProfile for editing, and then insert `Sys.setenv("CONNECT_API_KEY" = "paste key value")` and `Sys.setenv("CONNECT_SERVER" = "paste server value")`. Using environment variables is a best practice, but remember, if you're using git, it's a good idea to add your .RProfile to your .gitignore to ensure you're not publishing your API key to your version control system 😱

PRO TIPS from RStudio Customer Success

Creating Efficient Workflows with 'pins' and RStudio Connect

Retrieving a Pin From Connect

When you view your pin on Connect, you'll notice there is some header information included for pin retrieval. Let's copy that code into our analysis and **retrieve the pin**.

Replace the code section below with the sample from your own pin:

```
library(pins)
board <- board_rsconnect(
  server = Sys.getenv("CONNECT_SERVER"))
my_data <- pin_read(
  board,
  "your_username/my_data")
```

Now you, or anyone you have shared the pin with, can access this information in a secure, versioned manner.

Fun fact! Pins seek to make it both easy and *fast* to share data. Pins will automatically cache remote pins locally so it's fast to access data, but will always check to ensure the that it's up-to-date so you're never using stale data.

Schedule Updates to your Pin

Up to this point, you've found a home for your mid-process artifacts and learned how to share them as pins. Now it's time to put your pins on an update schedule and bask in glory as your analyses automatically refer to the most current data without requiring redeployment.

To do this, create an R Markdown document that pulls your data, does any needed processing, and then creates your pin on RStudio Connect. This will be a supporting ETL (extract, transform, and load) file in your pipeline. **Publish this R Markdown document to Connect**, and go ahead… pat yourself on the back; you've published an ETL document that publishes a pin to Connect, and you've got a pipeline, baby! Hint: If you run into publishing issues at this step, it's likely an environment variable issue.[4]

To finish this little gem, click the **Schedule** button in Connect and establish a schedule for your R Markdown ETL (and resulting pin) to refresh. Now you can point your downstream data analysis at this pin to always have a fresh source of data behind it. Dazzling! ✨

[4] If you are using RStudio v 1.8.8 or higher, RStudio Connect will automatically provide these environment variables for you when you publish. If you are on an older version of Connect, or if this feature has been disabled by your admin, you should reference https://docs.rstudio.com/connect/user/api-keys/ for how to create an API key and https://docs.rstudio.com/connect/user/content-settings/#content-vars for how to input environment variables into the Vars pane of Connect.

Get A Little Deeper

Ready for a little more? In this example, we used `pin_write()` and `pin_read()` to pin and retrieve a dataframe. This works great for any serializable R object (e.g., dataframes, model files, CSVs, feather files). You can use the more broad `pin_upload()` and `pin_download()` to **pin files from disk** to share types of data that would be otherwise unsupported by pins. A great use case here are parquet files.

Need details about that pin? Every pin has **metadata** that you can access with `pin_meta()`. It's particularly helpful to reference the date of the pin using metadata, or even supply your own custom metadata as an argument in the `pin_write()` or `pin_upload()` functions.

Want to keep track of **versions**? Specify `versioned = TRUE` in `pin_write()` or `pin_upload()` and each pin will create a new version. List available versions with `pin_versions()` and call a specific version with its unique hash in `pin_read()` or `pin_download()`.

Where Do I Go From Here?

At this point, you know what a pin is, whether pins will be useful for your workflow, and how to implement them. What next?

Go try `pins` on your own!

- The `pins` website has a comprehensive Getting Started guide, as well as specific information on pinning to RStudio Connect: http://pins.rstudio.com/.
- Looking for inspiration? See this content collection that uses a pinned model and datasets as part of a pipeline to support a Shiny app in Production. The underlying data in the pin is refreshed on a schedule, keeping the Shiny app current: https://solutions.rstudio.com/example/bike_predict/.
- See the Connect User Guide section on pins: https://docs.rstudio.com/connect/user/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! Reach out to your Customer Success Representative, or send a note to us at info@rstudio.com.