

Upload pins for use with data explorer app

Sadie Wisotsky

2025-09-29

Purpose

This is a walkthrough of how I set up a pins folder and added data to it. The tags and folder structures combine with a coming shiny app that allows you to filter data sets before downloading them. This is essentially my first attempt at an ETL pipeline using R only and prior to pushing for a switch to an SQL data warehouse and true ETL pipeline.

Get your data

first read the data in that you want to upload

```
library(tidyverse)

iris.dat <- data(iris)
```

take a peak at the data and also here is where you'd do any data manipulation to it (add columns, calculations, etc.)

```
summary(iris)
```

connect to the sharepoint

```
library(pins)
#library(Microsoft365R)

site <- "[placeholder]"
folder <- "../pins/tutorial" #change the folder based on where you wish the
pins to end up. can also make a new folder for a new study or project

# only needed if pins are stored remotely on a sharepoint site
# can also change this if data is on google drive, aws, etc
#sp <- get_sharepoint_site(site_id = site)
#doclib <- sp$get_drive()
#board <- board_ms365(doclib, folder)
```

can take a peak at the pins that already exist in this folder. Note that the pins listed here will not necessarily reflect the pins listed in the app because of the Upload = TRUE metadata step we will show later.

```
pin_list(board)
```

create new pin

here we are going to use `pin_write` to create a new pin in the folder. For the app, I've tried to include column names, a short description of the data, and some informative tags you will see how they get added here. additionally in the custom metadata section I have added a flag called "Upload", if "Upload" = TRUE then the data is meant to appear in the data delivery app. If "Upload" is FALSE or missing, it will not appear in the data delivery app.

```
cnames <- colnames(iris) # get column names

desc <- "default IRIS data for pins upload vingette"

pin_write(board,
  iris,
  name = "IRIS",
  #title = meta$title,
  description = desc,
  metadata = list(
    "Column Names" = cnames,
    "Upload" = TRUE #change if you do not wish to share with the app
  ),
  tags = c("example", "IRIS"),
  type = "rds",
  force_identical_write = TRUE # forces rewrite with changed metadata.
  # otherwise only over writes if data itself has changed.
)
```

check out the pin's metadata

```
meta <- pin_meta(board, "IRIS")
meta
```

Add pin to the app

so now the pin is officially in the pins folder and should have the correct metadata.

the next step to get it to show up in the app is to add it to the pin called `metadata_list` that I created for each study.

this exists because mapping over each pin and reading its data in real time was slowing down the app. so we do this before hand; outside of the app. But it means a pin will not show up in the app until this list has been refreshed by running this code

```
list.of.pins <- pin_list(board)
metas <- purrr::map(list.of.pins, purrr::possibly(.f = \(x) pin_meta(board,
x),
                                                    otherwise= "error"))

pin_write( board = board, metas, name = "metadata_list", type = "rds")
```

look at the metadata_list

if you are curious about what is in the metadata_list pin you can find out with:

```
mdl <- pin_read(board, "metadata_list")
mdl |> glimpse()
```

code from the app

if you are wondering why these steps are necessary the code for the app may be good to look at:

```
if ("metadata_list"%in%all.pins == TRUE) {
  metas <- pin_read(values$hi, "metadata_list")
}
else{
  metas <- purrr::map(all.pins,
                      possibly(.f = \(x) pin_meta(values$hi, x)))
}

values$ups <- data.frame(
  names = map(metas, ~pluck(., "name",
                           .default = "skip")) %>% unlist(),
  upload =map(metas, ~pluck(., "user", "Upload",
                           .default = "FALSE"))%>%
    unlist(),
  tags = map(metas, ~pluck(., "tags",
                           .default = "FALSE") %>%
    paste(collapse = ",") %>% unlist(),
  ext = map(metas, ~pluck(., "file",
                           .default = "rds")) |> unlist()
)
```

basically, for every study folder, the code looks for a pin called metadata_list and if it exists, reads it in. If it doesn't exist then it takes a minute to map over all the pins in a folder and read in the

metadata for them while the app is live (this can take a long time depending on the number of pins in a folder.)

and later on in the app the code populates the drop down choices with only the names of pins that have an Upload = TRUE value.