Pins: Three Ways

Ian Lyttle

2022-08-04

Table of contents

Preface **3**
Rest of the book . . . 3
Perspectives . . . 4

1 Using R **5**
1.1 Local board . . . 5
1.1.1 Writing pins . . . 5
1.1.2 Reading pins . . . 7
1.2 Remote board . . . 8
1.2.1 Reading pins . . . 8

Preface

The purpose of this book is to make a brief demonstration of the pins package using R and Python, and to imagine how it might be used with JavaScript.

Pins helps you manage sharing data with yourself, others, or even CI processes. There are two levels of abstraction:

- **pin**: a "thing" to be shared as a file. It could be a data frame, a model, a nested list (dictionary, object). If it can be serialized to a file, it can be pinned. Some serializations, such as CSV, JSON, and arrow, are common to multiple languages (R, Python, JavaScript), so can be used for cross-language collaboration. Other serializations are specific to a language (pickle for Python, rds for R).
- **board**: a collection of pins hosted at a "place". A board could be hosted at Azure Blob Storage, an Amazon S3 Bucket, RStudio (soon to be Posit) Connect, a local filesystem, a remote URL, …

Pins distinguishes itself from straightforward filesharing by:

- storing metadata, including user-defined metadata.
- this metadata allows pins to handle deserialization automatically.
- supporting versioning.
- supporting authentication for boards (e.g. AWS S3).
- caching results locally, so that reading a pin may not require a download.

Rest of the book

In the rest of the book I (plan to):

- use R to:
  - create a board.
  - write a data frame as a pin, using the arrow format.
  - read the pin into a data frame.
- use Python to:
  - read the data-frame pin written using R.

    - write a pandas data-frame as a pin using the arrow format.
- use JavaScript to:
    - read the data-frame pins written using R and Python, using arquero, which supports the arrow format.

> **Quarto implementation**
>
> Every time a pin is written, a new file is created on the board; this supports versioning. I don't want to write a new version of the same file each time this book is rendered (especially on CI).
> To avoid this, for code-blocks where I write pins:
>
> - I include code that I run only manually.
> - I paste the response into the prose manually.

Perspectives

I have some ideas for the conculsions I might come to in the course of writing the rest of this material. That said, I'll want to make some *actual observations* before calling for any action. I'll update this section as I go.

1 Using R

The first implementation of the pins package was made in R. In this chapter, I will:

- create a board on my local filesystem.
- write pins to the board.
- include the board as a part of this book's website.
- read pins from the board.

```
library("pins")
library("here")
library("palmerpenguins")
library("waldo")
library("conflicted")
```

1.1 Local board

The first step is to create a board:

```
board_local <- board_folder(here("pins"))
```

1.1.1 Writing pins

The next step is to write a pin. Let's write the penguins data-frame as a JSON pin:

```
pin_write(
  board_local,
  x = penguins,
  name = "penguins-json",
  type = "json",
  metadata = list(
    authors = c("Allison Horst", "Alison Hill", "Kristen Gorman"),
    license = "CCO",
    url = "https://allisonhorst.github.io/palmerpenguins/"
```

```
  )
)
```

```
Creating new version '20220805T171936Z-fa33e'
Writing to pin 'penguins-json'
```

As you can see, the version number is a combination of the creation time (UTC) and a (shortened) hash of the contents.

I also want to create an arrow version of the pin.

The `pin_write()` function offers `type = "arrow"`, which uses `arrow::write_feather()`. However, the default behavior is to use compression; pins does not offer (so far as I know) a way to supply the `compression` argument to `arrow::write_feather()`. This presents a problem for me because the arrow implementation for JavaScript does not support compression.

It should not surprise you that pins offers an escape hatch, I can wrap `pins_upload()` in a function:

```
pin_write_arrow_uncompressed <- function(board, x, name = NULL, ...) {

  tempfile <- withr::local_tempfile()

  arrow::write_feather(x, tempfile, compression = "uncompressed")

  result <- pins::pin_upload(
    board,
    paths = tempfile,
    name = name,
    ...
  )

  message(glue::glue("Writing to pin '{name}'"))

  invisible(result)
}
```

```
pin_write_arrow_uncompressed(
  board_local,
  x = penguins,
  name = "penguins-arrow",
  metadata = list(
```

```
    authors = c("Allison Horst", "Alison Hill", "Kristen Gorman"),
    license = "CCO",
    url = "https://allisonhorst.github.io/palmerpenguins/"
  )
)
```

```
Creating new version '20220805T175034Z-ef034'
Writing to pin 'penguins-arrow'
```

1.1.2 Reading pins

```
penguins_json <- pin_read(board_local, name = "penguins-json")
compare(penguins, penguins_json)
```

```
`class(old)`: "tbl_df" "tbl" "data.frame"
`class(new)`:                "data.frame"

`old$species` is an S3 object of class <factor>, an integer vector
`new$species` is a character vector ('Adelie', 'Adelie', 'Adelie', 'Adelie', 'Adelie', ...)

`old$island` is an S3 object of class <factor>, an integer vector
`new$island` is a character vector ('Torgersen', 'Torgersen', 'Torgersen', 'Torgersen', 'Torg

`old$sex` is an S3 object of class <factor>, an integer vector
`new$sex` is a character vector ('male', 'female', 'female', NA, 'female', ...)
```

We see some differences between the original ("old") version and "new" version of penguins:

- new version does not have the "tibble" classes.
- new version does not know that some of the columns are factors.

These are not huge differences; in fact, the JSON format has no way of encoding that something is a factor.

Let's look at the arrow version. Because we used a file format (using `pin_upload()`), we need also to write a handler for `pin_download()`:

```
pin_read_arrow_uncompressed <- function(board, name, ...) {

  tempfile <- pins::pin_download(board, name, ...)
```

```
  arrow::read_feather(tempfile)
}
```

```
penguins_arrow <- pin_read_arrow_uncompressed(board_local, "penguins-arrow")
compare(penguins, penguins_arrow)
```

```
v No differences
```

The fact that there are no differences is one of the many cool things about arrow.

1.2 Remote board

1.2.1 Reading pins