Extracting Relevant Data

Shukry Zablah and Emily Lachtara

2019-10-01

Setup

```
library(purrr)
library(data.table)
library(vroom)
library(janitor)
library(dplyr)
library(fs)
```

Extracting Relevant Data

The idea here is that gps data contains a lot of information that is not immediately useful. We only care about the start and end locations to do our initial analysis.

Thus we will read in each file, filter out the start and end entries for all the routes and then write them to a larger file. The idea is that then we can perform our analysis in memory.

We create a helper function.

```
extract_from_file <- function(file_path) {
    day <- file_path %>%
        fread(skip = 2) %>%
        clean_names() %>%
        group_by(route_id) %>%
        filter(date == max(date) | date == min(date)) %>%
        ungroup()
    return(day)
}

empty <- tibble(route_id = NA_character_,
                bike = NA_real_,
                date = NA_character_,
                latitude = NA_real_,
                longitude = NA_real_,
                user_id = NA_character_)

possibly_extract_from_file <- possibly(extract_from_file, empty)
```

Don't pay too much attention to the detail. Basically we want to handle the failures that arise from improperly downloaded data and problems during read in.

The following is another helper function that will allow us to perform the readin and filter operation on all files we want.

```
extract_from_matching_files <- function(pattern = NULL) {
    file_names <- dir_ls("../data-raw", regexp = pattern)
    res <- file_names %>%
        map_dfr(possibly_extract_from_file)
```

```
    return(res)
}
```

And then we finally get it. Note you can control what files to take into account by passing a pattern. (You can also leave empty for all the files.)

```
Routes_slim <- extract_from_matching_files("VB_Routes_Data_201._*")
```

Saving

```
Routes_slim %>%
    vroom_write("../data/slim.tsv")
```