

# Processing data sources for analysis

Michael-Philipp Stiebing

2023-03-06

- Start by importing libraries

```
library(tidyverse)
library(lubridate)
library(ggplot2)
library(readr)
```

- Read the csv files into separate dataframes
- Combine all dataframes into a single dataframe

```
setwd("/home/mikiR/remote_transfer/")
X202110_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202110-divvy-tripdata.csv")
X202111_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202111-divvy-tripdata.csv")
X202112_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202112-divvy-tripdata.csv")
X202201_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202201-divvy-tripdata.csv")
X202202_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202202-divvy-tripdata.csv")
X202203_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202203-divvy-tripdata.csv")
X202204_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202204-divvy-tripdata.csv")
X202205_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202205-divvy-tripdata.csv")
X202206_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202206-divvy-tripdata.csv")
X202207_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202207-divvy-tripdata.csv")
X202208_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202208-divvy-tripdata.csv")
X202209_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202209-divvy-tripdata.csv")
X202210_divvy_tripdata <-
  read_csv("./20221205-capstone_datascience-01/002_data/001_csv/202210-divvy-tripdata.csv")

all_trips <- bind_rows(
  X202110_divvy_tripdata, X202111_divvy_tripdata, X202112_divvy_tripdata, X202201_divvy_tripdata,
  X202202_divvy_tripdata, X202203_divvy_tripdata, X202204_divvy_tripdata, X202205_divvy_tripdata,
  X202206_divvy_tripdata, X202207_divvy_tripdata, X202208_divvy_tripdata, X202209_divvy_tripdata,
  X202210_divvy_tripdata)
```

- Create new columns from date field, split into
  - month
  - day
  - year
  - day of week

```
all_trips$date <- as.Date(all_trips$started_at)
all_trips$month <- format(as.Date(all_trips$date), "%m")
all_trips$day <- format(as.Date(all_trips$date), "%d")
all_trips$year <- format(as.Date(all_trips$date), "%Y")
all_trips$day_of_week <- format(as.Date(all_trips$date), "%A")
```

- Create column `ridelength` calculated from the difference of `ended_at` and `started_at`
- Convert column value into numeric

```
all_trips$ride_length <- difftime(all_trips$ended_at, all_trips$started_at)
all_trips$ride_length <- as.numeric(as.character(all_trips$ride_length))
```

- Remove bad data

```
all_trips_v2 <- all_trips[!(all_trips$ride_length < 0),]
```

- Move bad data into a dataframe to doublecheck

```
all_trips_errors <- all_trips[(all_trips$ride_length < 0),]
```

- Trying to calculate the distance between gps coordinates `start_lng` & `start_lat` / `end_lng` & `end_lat`
- Import `geosphere` library

```
library(geosphere)
```

- Calculate the distance between the start and end coordinates using the Haversine formula
- Create new column `geodist` with the calculated distance

```
all_trips_v3 <- all_trips_v2 %>%
  mutate(geodist = distHaversine(
    cbind(all_trips_v2$start_lng, all_trips_v2$start_lat),
    cbind(all_trips_v2$end_lng, all_trips_v2$end_lat)))
```

- Filter all trips with distance = 0 into a dataframe called `round_trips`, the assumption being that when the trip ends where it started

```
round_trips = filter(all_trips_v3, geodist == 0)
```

- While working with the data, I noticed some very long rental periods (> 86400 seconds, aka one day) I think this would be maintenance work, as this would be unlikely to be a rental from a customer
- Also, I filtered out all rentals under 1 minute, since these do not represent useful customer behavior as well

- Similarly, all entries that have a geodist value of NA - because they don't have a return station value, are similarly out for maintenance, or stolen, or in case of very short rental periods of a few seconds, maybe a booking error or system test
- I filtered these sets into dataframes into short\_ride long\_ride and nadist\_ride in order to analyze a little bit

```
short_ride <- (filter(all_trips_v3,ride_length < 60 & !is.na(geodist)))
long_ride <- (filter(all_trips_v3,ride_length > 86400 & !is.na(geodist)))
nadist_ride <- (filter(all_trips_v3,is.na(geodist)))

summary(filter(long_ride)$ride_length)
```

```
##  Min. 1st Qu. Median  Mean 3rd Qu.  Max.
## 86445  89809  97212 222552 207746 2442301
```

```
summary(filter(nadist_ride)$ride_length)
```

```
##  Min. 1st Qu. Median  Mean 3rd Qu.  Max.
##    6  89990  89994 175632  90001 2483235
```

- Create a new dataframe all\_trips\_v4 containing only trips under 86400 s & geodist not NA

```
all_trips_v4 <- (filter(all_trips_v3,ride_length < 86400 & ride_length > 60 & ride_length != 0 & !is.na(geodist)))
summary(filter(all_trips_v4)$ride_length)
```

```
##  Min. 1st Qu. Median  Mean 3rd Qu.  Max.
##  61.0  366.0  634.0  998.6 1129.0 86391.0
```