Update #1

Shukry Zablah and Emily Lachtara

2019-10-03

Setup

```
library(dplyr)
library(readr)
library(tidyr)
library(lubridate)
library(ggplot2)
library(leaflet)
```

Ingestion

We import the slim version of the data that contains only the start and end entries for any given route, throughout all available data.

```
Routes <- read_tsv("../data/slim.tsv",
                   col_types = cols(
                       route_id = col_character(),
                       bike = col_number(),
                       date = col_datetime(format = ""),
                       latitude = col_double(),
                       longitude = col_double(),
                       user_id = col_character()
                   ),
                   na = c("-", "NA"))

Routes %>% head()
```

```
## # A tibble: 6 x 6
##   route_id          bike date                latitude longitude user_id
##   <chr>            <dbl> <dttm>                 <dbl>     <dbl> <chr>
## 1 route_06_2018@d~   924 2018-06-28 14:09:32     42.3     -72.6 1cc1e858-8~
## 2 route_06_2018@d~   924 2018-06-28 17:33:57     42.3     -72.6 1cc1e858-8~
## 3 route_06_2018@3~   984 2018-06-28 16:43:09     42.3     -72.6 72491657-3~
## 4 route_06_2018@3~   984 2018-06-28 17:11:54     42.3     -72.6 72491657-3~
## 5 route_06_2018@8~   935 2018-06-28 16:43:35     42.3     -72.6 72491657-3~
## 6 route_06_2018@8~   935 2018-06-28 16:54:15     42.3     -72.6 72491657-3~
```

Cleaning

Two things.

1) Our function to slim down the data actually took care of badly downloaded data so will get some lines that are only NAs.
2) The data actually has duplicate entries.

```
Routes <- Routes %>%
    drop_na(-date)  %>% # 9 entries are NA only
    drop_na() %>%  # around 400 entries have no date info
    distinct()
```

We removed a tiny amount of data but it will simplify our lives to have data without NAs.

So far the assumption is that there is two entries for every route (the start and the end). Let's verify.

```
Routes %>%
    count(route_id, name = "num_entries") %>%
    count(num_entries, name = "count")
```

```
## # A tibble: 3 x 2
##   num_entries count
##         <int> <int>
## 1           1  2401
## 2           2 93253
## 3           3     1
```

Less than 3% of the bikes have only one entry, while a single one has 3. For now we will drop these.

```
Inconsistent <- Routes %>%
    count(route_id) %>%
    filter(n != 2)

Routes <- Routes %>%
    anti_join(Inconsistent, by = "route_id")
```

These leaves 93253 routes with start and end entries (the trims of data have been almost none).

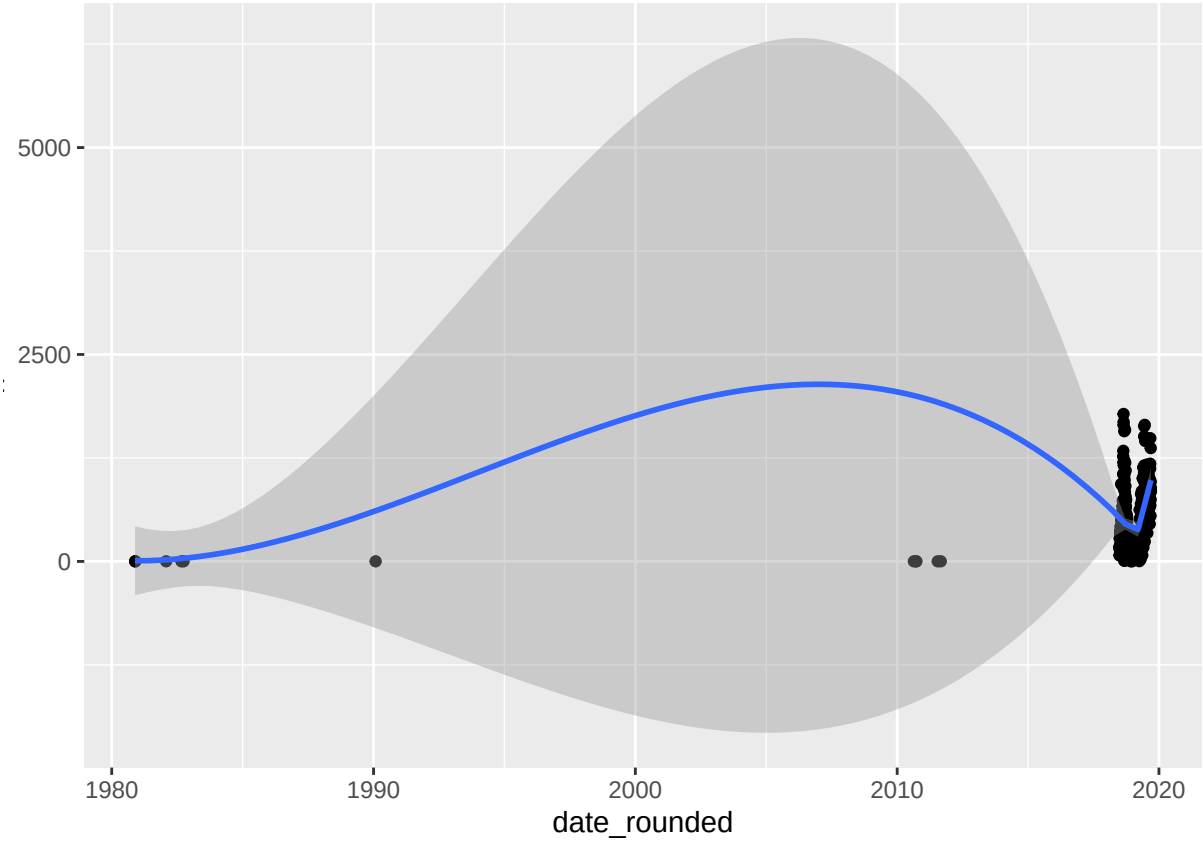
Exploratory Data Analysis and Further Cleanup

We have data for what days?

We know Valley Bike has only been a thing since 2017. Let's look at the data.

```
Routes <- Routes %>%
    mutate(date_rounded = round_date(date, unit="day"))

Routes %>%
    count(date_rounded) %>%
    ggplot(aes(x = date_rounded, y = n)) +
    geom_point() +
    geom_smooth(method = "loess")
```



Let's take a closer look. It seems there are extraneous dates.

```
Routes %>%
    arrange(date) %>%
    head(15)
```

```
## # A tibble: 15 x 7
##    route_id  bike date                latitude longitude user_id
##    <chr>    <dbl> <dttm>                 <dbl>     <dbl> <chr>
##  1 route_1~  1381 1980-11-21 14:24:06    42.3      -72.6 1630f2~
##  2 route_1~  1206 1980-11-26 02:01:04     4.38     -72.6 355a57~
##  3 route_0~  1165 1982-01-28 21:50:09    42.1      -72.6 802aae~
##  4 route_0~  1128 1982-08-30 15:52:09     4.45     -72.6 97db28~
##  5 route_0~  1399 1982-09-29 02:53:08    42.1      -72.6 2edbc5~
##  6 route_0~  1291 1990-01-29 14:00:01    42.1      -72.6 2f5144~
##  7 route_0~  1260 2010-08-22 00:21:35    42.1      -72.6 0557dc~
##  8 route_0~  1294 2010-09-20 20:17:26    42.1      -72.6 e8e316~
##  9 route_0~  1339 2010-09-25 13:43:12    42.4      -72.5 5c3475~
## 10 route_0~  1000 2011-07-20 16:47:45    42.3      -72.6 d81efa~
## 11 route_0~  1019 2011-09-01 00:57:56    42.1      -72.6 46dd58~
## 12 route_0~   924 2018-06-28 14:09:32    42.3      -72.6 1cc1e8~
## 13 route_0~   984 2018-06-28 16:43:09    42.3      -72.6 724916~
## 14 route_0~   935 2018-06-28 16:43:35    42.3      -72.6 724916~
## 15 route_0~   965 2018-06-28 16:44:19    42.3      -72.6 724916~
## # ... with 1 more variable: date_rounded <dttm>
```

Great... 1980-2011 observations. There's only 11 of them so we will just filter those. The fix of the plot:

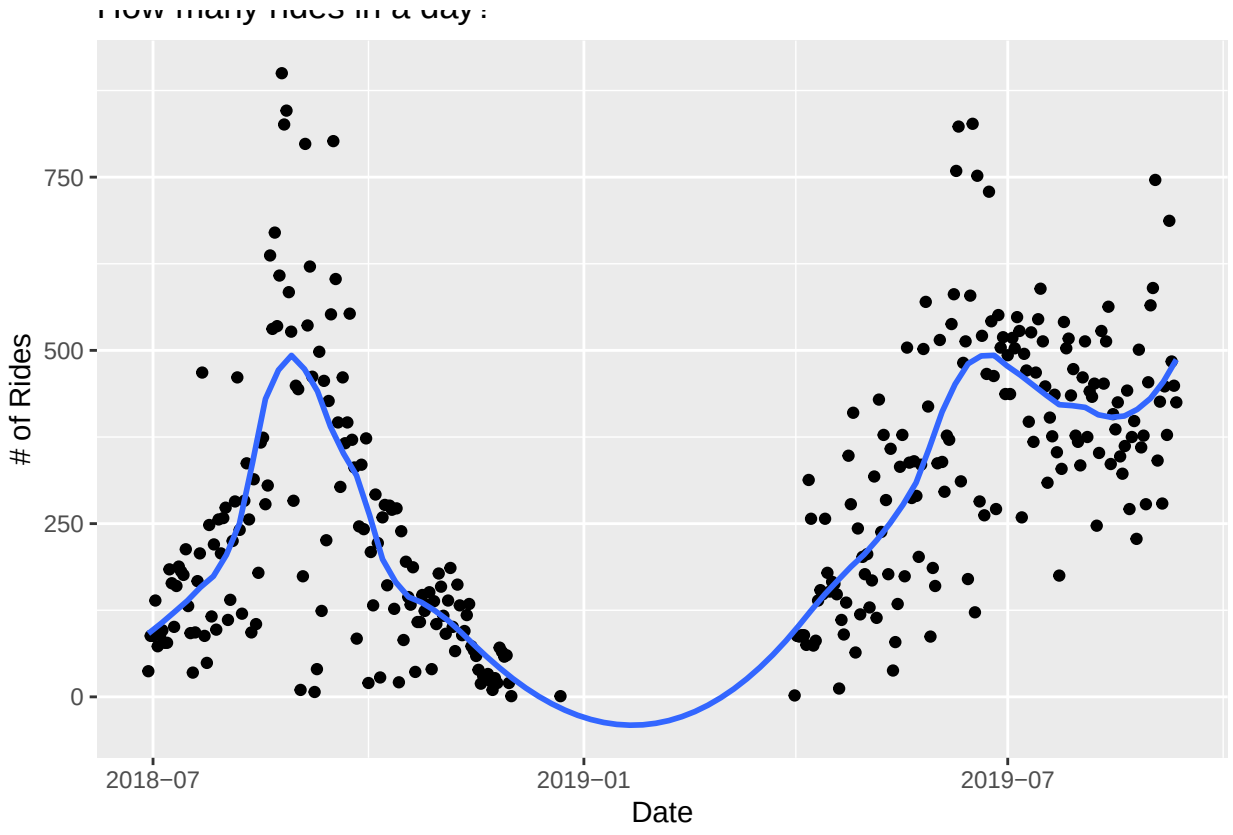
```
Routes <- Routes %>%
    filter(date_rounded >= "2017-01-01")
```

The command above introduced inconsistencies like before. We again want to make our routes have two entries each.

```
Inconsistent2 <- Routes %>%
    count(route_id) %>%
    filter(n != 2)

Routes <- Routes %>%
    anti_join(Inconsistent2, by = "route_id")
```

```
Routes %>%
    count(route_id, date_rounded) %>%
    count(date_rounded, name = "num_routes") %>%
    ggplot(aes(x = date_rounded, y = num_routes)) +
    geom_point() +
    geom_smooth(method = "loess", span=0.2, se = FALSE) +
    labs(x = "Date", y = "# of Rides",
         title = "How many rides in a day?")
```



General usage seems to be predictable. The summers and the start of the semester are the most popular.

2) How many unique rides are there?

```
Routes %>%
  count(route_id) %>%
```

```
  count(name = "num_rides")
```

```
## # A tibble: 1 x 1
##   num_rides
##       <int>
## 1     93242
```

3) Where are the stations?

Our method right now is to round off the coordinates to three decimal places. Then count them up and filter by the arbitrarily chosen value of 500. This means that there are more than 500 entries in that "location".

```
Routes <- Routes %>%
    mutate(lat_rounded = round(latitude, 3),
           lon_rounded = round(longitude, 3))

Stations <- Routes %>%
    count(lat_rounded, lon_rounded, sort = T) %>%
    filter(n > 500) %>%
    arrange(desc(n)) %>%
    mutate(name = paste("Station", row_number())) %>%
    select(name, latitude = lat_rounded, longitude = lon_rounded)

glimpse(Stations)
```

```
## Observations: 56
## Variables: 3
## $ name      <chr> "Station 1", "Station 2", "Station 3", "Station 4", ...
## $ latitude  <dbl> 42.385, 42.376, 42.318, 42.097, 42.394, 42.202, 42.3...
## $ longitude <dbl> -72.531, -72.520, -72.627, -72.582, -72.526, -72.620...
```

We don't have the names right now but there seems to be a similar number of stations as the 56 we have here. We need to find a more sophisticated way of determining whether something is close to something or not.

```
Stations %>%
    leaflet() %>%
    addTiles() %>%
    addMarkers(lng = ~longitude, lat = ~latitude)
```

We save for later our stations, from most to least "popular".

```
write_tsv(Stations, path = "../data/stations.tsv")
```

4) What are the most popular stations for checkout/returns? Least?

Let's identify returns and checkouts. Remember we have two entries, the start and the end of a route.

```
Routes <- Routes %>%
    group_by(route_id) %>%
    mutate(action = if_else(date == min(date), "checkout", "return")) %>%
    ungroup()
```

And let's verify our data makes sense.

```
Routes %>%
    count(action)
```

```
## # A tibble: 2 x 2
```

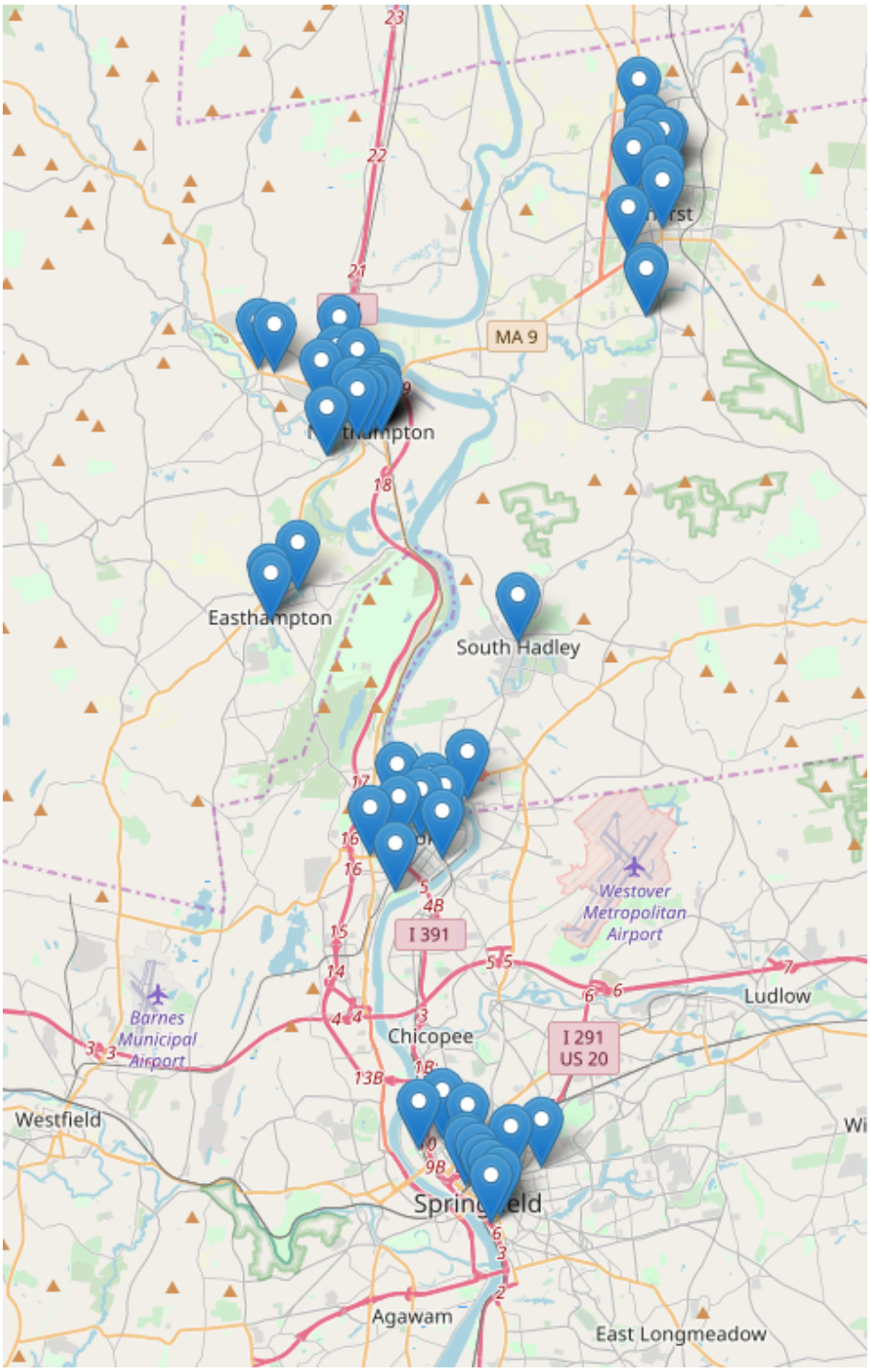


Figure 1: Map of Valley Bike stations.

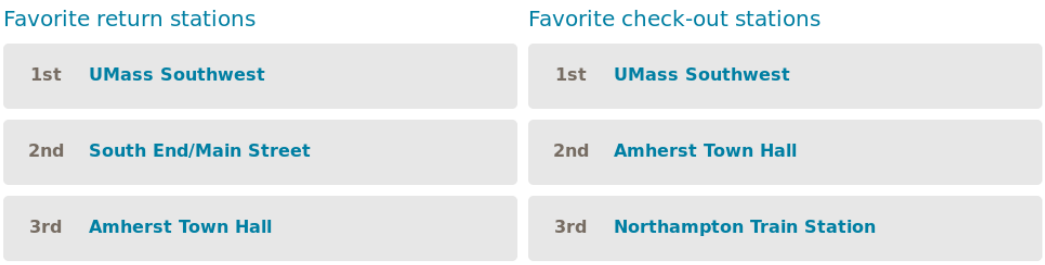


Figure 2: Top stations for checkout and returns from ValleyBike.org

```
##   action       n
##   <chr>    <int>
## 1 checkout 93242
## 2 return   93242
```

Perfect! Now we can find the popularity for checkout/returns.

```
Popularity <- Routes %>%
    count(lat_rounded, lon_rounded, action, sort = T)
```

```
Popularity %>%
    filter(action == "checkout") %>%
    head(3) %>%
    glimpse()
```

```
## Observations: 3
## Variables: 4
## $ lat_rounded <dbl> 42.385, 42.376, 42.097
## $ lon_rounded <dbl> -72.531, -72.520, -72.582
## $ action      <chr> "checkout", "checkout", "checkout"
## $ n           <int> 4131, 3891, 3734
```

```
Popularity %>%
    filter(action == "return") %>%
    head(3) %>%
    glimpse()
```

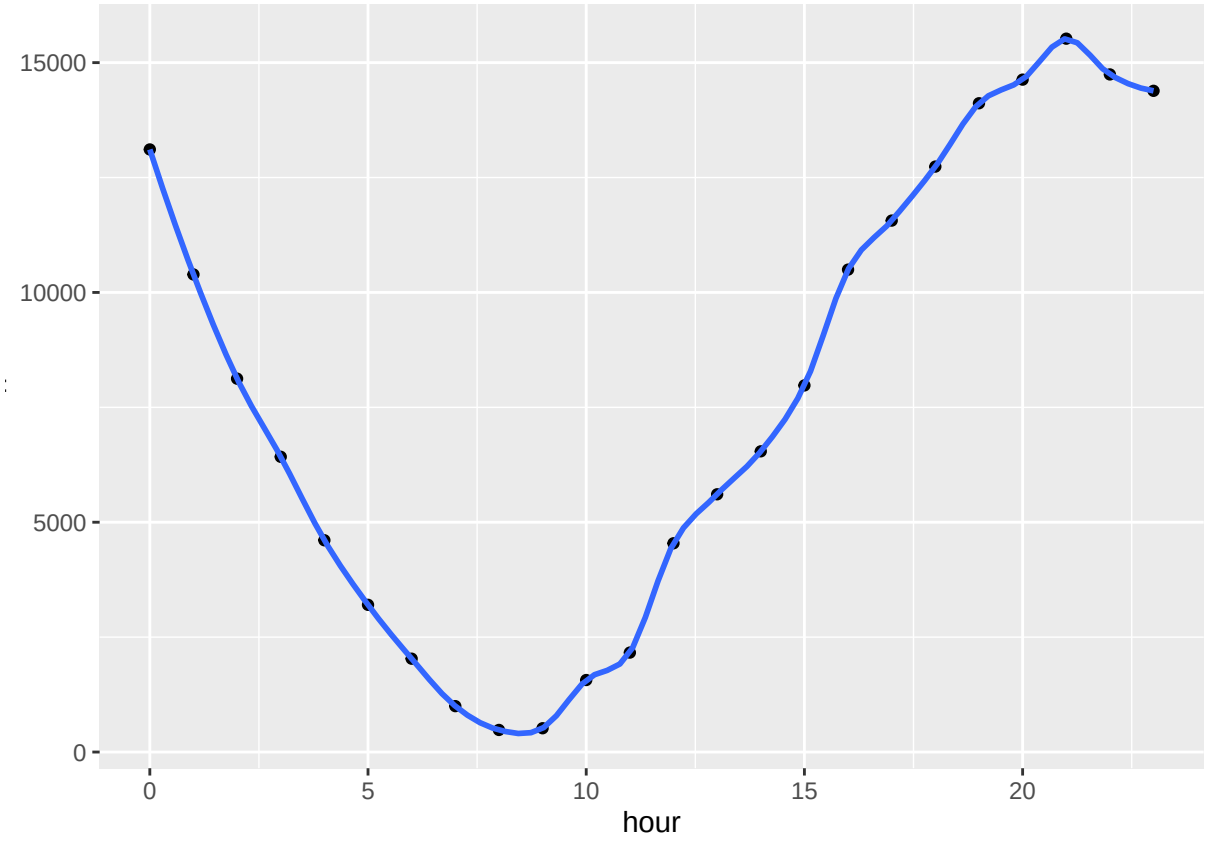
```
## Observations: 3
## Variables: 4
## $ lat_rounded <dbl> 42.385, 42.318, 42.376
## $ lon_rounded <dbl> -72.531, -72.627, -72.520
## $ action      <chr> "return", "return", "return"
## $ n           <int> 4053, 3979, 3714
```

Looking carefully (we could plot them too) we see that our most popular locations match the ones listed on the official site.

4) How does usage vary across the day? (Aggregate version)

The most bike usage occurs in the evening- with 9pm racking in the most with 15688 users returning or leaving from a trip. There does not appear to be much usage during the work-day, from 7am-5pm, as well as the early morning. There is a suprising amount of usage during the night, with 11pm and midnight rides tallying 14582 and 14582 users respectively.

```
Routes %>%
    mutate(hour = hour(date)) %>%
    count(hour, sort = T) %>%
    ggplot(aes(x=hour, y=n)) +
    geom_point() +
    geom_smooth(method = "loess", span=0.2)
```



The above is a good start but we will have to break it down because the hourly usage probably depends on the seasonal trends too.

Save Our Progress

We have made some changes we would like to keep.

```
Routes %>% names()
```

```
##  [1] "route_id"     "bike"         "date"         "latitude"
##  [5] "longitude"    "user_id"      "date_rounded" "lat_rounded"
##  [9] "lon_rounded"  "action"
```

Write it out.

```
write_tsv(Routes, path = "../data/slim_clean.tsv")
```