Ride Membership Analytics

Toyin Olapade

2021-11-08

This is a capstone project undertaken as part of the Google Data Analytics course offered on Coursera. It involves working with Cyclistic, a bike-share company in Chicago, as a Junior Data Analyst. The director of marketing at Cyclistic believes the company's future success depends on maximizing the number of annual memberships. Therefore, the marketing team wants to understand how casual riders and annual members use Cyclistic bikes differently. From these insights, the team will design a new marketing strategy to convert casual riders into annual members. In order to proceed with this goal the executives at Cyclistic require compelling data insights and professional data visualizations, and must approve recommendations resulting from this analysis.

Introduction

In 2016, `Cyclistic`[1] launched a successful bike-share offering. Since then, the program has grown to a fleet of 5,824 bicycles that are geotracked and locked into a network of 692 stations across Chicago. The bikes can be unlocked from one station and returned to any other station in the system anytime.

Until now, Cyclistic's marketing strategy relied on building general awareness and appealing to broad consumer segments. One approach that helped make these things possible was the flexibility of its pricing plans: single-ride passes, full-day passes, and annual memberships. Customers who purchase single-ride or full-day passes are referred to as `casual` riders. Customers who purchase annual memberships are Cyclistic `members`[2].

Cyclistic's finance analysts have concluded that `annual` members are much more profitable than `casual` riders. Although the pricing flexibility helps Cyclistic attract more customers, Moreno[3] believes that maximizing the number of annual members will be key to future growth. Rather than creating a marketing campaign that targets all-new customers, she believes there is a very good chance to convert casual riders into members. She notes that casual riders are already aware of the Cyclistic program and have chosen Cyclistic for their mobility needs.

Moreno has set a clear goal: Design marketing strategies aimed at converting casual riders into annual members. In order to do that, however, my team[4] needs to better understand how annual members and casual riders differ, why casual riders would buy a membership, and how digital media could affect their marketing tactics. Moreno

[1] `Cyclistic` is bike-share company resident in Chicago.

[2] For the purpose of clear description during data analysis we have relabelled the `member` to `annual`.

[3] Lily Moreno is the director of marketing at Cyclistic

[4] The Marketing Analytics Team

and her team are interested in analyzing the Cyclistic historical bike trip data to identify trends.

Three questions will guide the future marketing program: How do annual members and casual riders use Cyclistic bikes differently? Why would casual riders buy Cyclistic annual memberships? How can Cyclistic use digital media to influence casual riders to become members? I have been assigned the first question, and this note[5] reports the following deliverable:

1. A clear statement of the business task
2. A description of all data sources used
3. Documentation of any cleaning or manipulation of data
4. A summary of your analysis
5. Supporting visualizations and key findings
6. Your top three recommendations based on your analysis

[5] This note is prepared in RStudio. See Irene Steve's Using SQL in RStudio at https://irene.rbind.io/post/using-sql-in-rstudio/#passing-variables-tofrom-sql-chunks and Creating Dynamic Documents with RMarkdown and Knitr at https://rstudio-pubs-static.s3.amazonaws.com/180546_e2d5bf84795745ebb5cd3be3dab71fca.html# for guidance. For the backend DBMS we used Postgres (pgAdmin 14).

Business Task

In order to convert casual riders to annual members, Cyclistic wants to understand how annual members and casual riders differs.

Data Collection

This data is collected from Cyclistic's marketing strategy at Google case study, from 05/2020 to 04/2021.

Connecting to Postgres

We used the `DBI` package to facilitate database connectivity. It has the core functionality of connecting R to database servers. We load, `RPostgres`, a package that implements the core functionality of `DBI` for `PostgreSQL`, create and test a `dbConnect` object to hold the connection to `Postgres` database.[6]

[6] It should be noted that this code was generated on my local machine connected to a local copy of the database. Your connection details will be different. I also have permissions to modify this database.

Data Dictionary

We use Cyclistic's historical trip data[7] to analyze and identify trends. This is public data that you can use to explore how different customer types are using Cyclistic bikes. But note that data-privacy issues prohibit you from using riders' personally identifiable information. This means that you won't be able to connect pass purchases to credit card numbers to determine if casual riders live in the Cyclistic service area or if they have purchased multiple single passes.

[7] See data at https://divvy-tripdata.s3.amazonaws.com/index.html. The datasets have a different name because Cyclistic is a fictional company. For the purposes of this case study, the datasets are appropriate and will enable you to answer the business questions. The data has been made available by Motivate International Inc. under this license

Table 1: Cyclistic Data Dictionary.

var_nm	description	data_type
ride_id	Unique identifier for each ride	text
rideable_type	Bike types rideable (classic, docked, electric)	text
started_at	Time at start of trip	datetime
ended_at	Time at end of trip	datetime
start_station_name	Trip-start station name	text
start_station_id	Unique trip-start station identifier	text
end_station_name	Trip-end station name	text
end_station_id	Unique trip-end station identifier	text
start_lat	Latitude of trip-start geolocation	text
start_lng	Longitude of trip-start geolocation	text
end_lat	Latitude of trip-end geolocation	numeric
end_lng	Longitude of trip-end geolocation	numeric
member_typ	Type of riders (casual, annual)	text

Data Preparation

We want to take a look at 12 months of data ranging from `2020-04-01` to `2021-03-31`. We notice that for easier description it may be better to rename the `member_casual` field to `member_typ` and its values from `'member'` to `'annual'`[8].

[8] See Appendix for SQL queries used to achieve this.

```
select ride_id, rideable_type, started_at, member_typ
from divvy_tripdata_
where date(started_at) >= cast('2020-04-01' as date)
  and date(started_at) <= cast('2021-03-31' as date)
```

```
## # A tibble: 6 x 4
##   ride_id          rideable_type started_at          member_typ
##   <chr>            <chr>         <dttm>              <chr>
## 1 A847FADBBC638E45 docked_bike   2020-04-26 17:45:14 annual
## 2 5405B80E996FF60D docked_bike   2020-04-17 17:08:54 annual
## 3 5DD24A79A4E006F4 docked_bike   2020-04-01 17:54:13 annual
## 4 2A59BBDF5CDBA725 docked_bike   2020-04-07 12:50:19 annual
## 5 27AD306C119C6158 docked_bike   2020-04-18 10:22:59 casual
## 6 356216E875132F61 docked_bike   2020-04-30 17:55:47 annual
```

Data Analysis

Questions to explore are:

- Which station is patronized the most, and what type of members

patronized that station? In other words, does a member type tend to prefer a station over another?
- Which station has the most rides per day?
- Do members tend to drop their bike at the same place they picked it from?

Furthermore, one question that is more striking is how many riders patronizes Cyclistics within some period[9] and how is membership type distributed among these riders? However, the data collected by Cyclistic only describe rides and not riders. Each ride has a unique ride id.

[9] We are assuming that a ride day is identified by the ride's start datetime. The period covers `2020-04-01` to `2021-03-31`.

```
select member_typ, count(distinct ride_id) as n_ride
from divvy_tripdata_
where date(started_at) >= cast('2020-04-01' as date)
  and date(started_at) <= cast('2021-03-31' as date)
group by member_typ
```

```
dt_01 %>%
  group_by(member_typ) %>%
  summarise (n=n()) %>%
  mutate(pct=paste0(round(100 * n/sum(n), 0), "%"))
```

```
## # A tibble: 2 x 3
##   member_typ       n pct
##   <chr>        <int> <chr>
## 1 annual     2059372 59%
## 2 casual     1430376 41%
```

```
## # A tibble: 6 x 4
##   ride_id          rideable_type started_at          member_typ
##   <chr>            <chr>         <dttm>              <chr>
## 1 A847FADBBC638E45 docked_bike   2020-04-26 17:45:14 annual
## 2 5405B80E996FF60D docked_bike   2020-04-17 17:08:54 annual
## 3 5DD24A79A4E006F4 docked_bike   2020-04-01 17:54:13 annual
## 4 2A59BBDF5CDBA725 docked_bike   2020-04-07 12:50:19 annual
## 5 27AD306C119C6158 docked_bike   2020-04-18 10:22:59 casual
## 6 356216E875132F61 docked_bike   2020-04-30 17:55:47 annual
```

The data suggests that about 40% of the rides are taken by the casual riders. This is surprising though as I would expect casual riders to be more than the annual members. Perhaps there are some benefits that are interesting to the annual members which the casual riders are not utilizing. To look at a similar aspects, we ask if annual members are more motivated than casual riders to ride. We would expect annual members to have more rides per day[10] than the casual riders if this were true.

[10] Regardless of seasonality

```
select member_typ,
  to_char(date(started_at), 'yy-mm') as ride_mth,
  date(started_at) as ride_dt,
  count(distinct ride_id) as ride_cnt
from divvy_tripdata_
where date(started_at) >= cast('2020-04-01' as date)
  and date(started_at) <= cast('2021-03-31' as date)
group by member_typ, to_char(date(started_at), 'yy-mm'), date(started_at)
```

```
## # A tibble: 726 x 4
##    member_typ ride_mth ride_dt    ride_cnt
##    <chr>      <chr>    <date>      <int64>
##  1 annual     20-04    2020-04-01     1895
##  2 annual     20-04    2020-04-02     2060
##  3 annual     20-04    2020-04-03     2570
##  4 annual     20-04    2020-04-04     1758
##  5 annual     20-04    2020-04-05     1982
##  6 annual     20-04    2020-04-06     1935
##  7 annual     20-04    2020-04-07     3374
##  8 annual     20-04    2020-04-08     1765
##  9 annual     20-04    2020-04-09     1563
## 10 annual     20-04    2020-04-10     2056
## # ... with 716 more rows
```

We now have ride counts per day but we want to visualize its trend by ride membership type so we can compare patterns between them.

```
dt_01 %>%
  mutate(ride_cnt_=as.numeric(ride_cnt)) %>%
  ggplot(mapping=aes(x=ride_mth, y=ride_cnt_, color=member_typ)) +
  geom_boxplot() +
  theme(legend.position = 'bottom')
```

From Figure 1 it looks like annual members rides more per day than casual riders. To further explore this we compute and plot the average ride per day for each month in Figure 2.

```
select member_typ, to_char(ride_dt, 'yy-mm') as ride_mth,
  avg(ride_cnt) as ride_avg
from (
  select member_typ, date(started_at) as ride_dt,
    count(distinct ride_id) as ride_cnt
  from divvy_tripdata_
  where date(started_at) >= cast('2020-04-01' as date)
    and date(started_at) <= cast('2021-03-31' as date)
  group by member_typ, date(started_at)
```

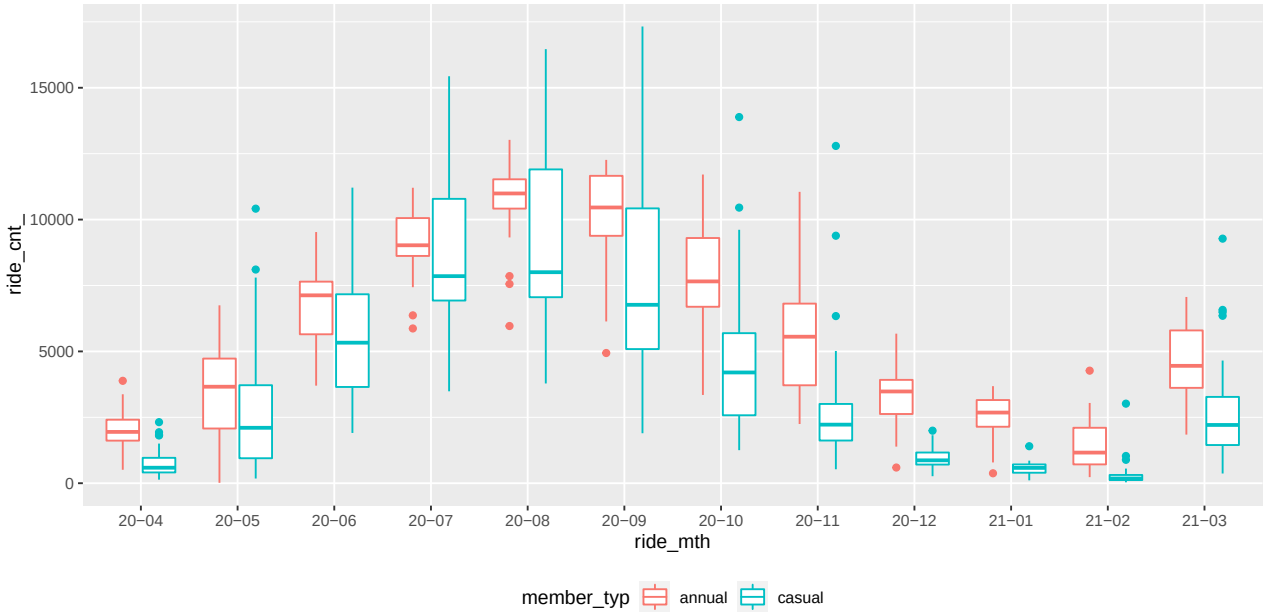

Figure 1: MoM Rides by Member Type

```
) as a
group by member_typ, to_char(ride_dt, 'yy-mm')
order by member_typ, ride_mth


dt_01 %>%
  mutate(ride_cnt_=as.numeric(ride_cnt)) %>%
  ggplot(mapping=aes(x=ride_mth, y=ride_cnt_, fill=member_typ)) +
  stat_summary(fun.y=median, geom="point", shape=21, size=4)+
   labs(title="divvy_cyclist:ride_cnt_ vs.ride_mth",
     subtitle = "sample of the three rideable_type",
     caption = "Data collected by divvy cyclyist")+
  annotate("text", x=5,y=40,label="The annual member are the highest user")
```

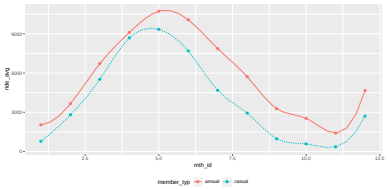

Figure 2: MoM Rides by Member Type

```
## Warning: `fun.y` is deprecated. Use `fun` instead.


select member_typ, rideable_type,
  to_char(date(started_at), 'yy-mm') as ride_mth,
  date(started_at) as ride_dt,
  count(distinct ride_id) as ride_cnt
from divvy_tripdata_
where date(started_at) >= cast('2020-04-01' as date)
  and date(started_at) <= cast('2021-03-31' as date)
group by member_typ, rideable_type, to_char(date(started_at), 'yy-mm'),
  date(started_at)
```

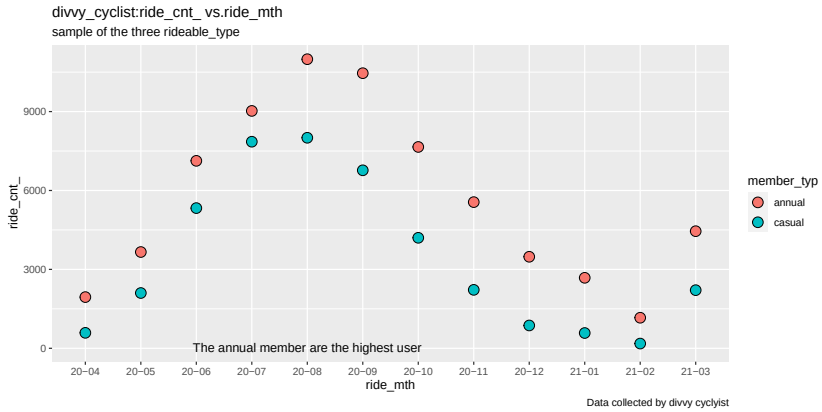

Figure 3: MPG vs horsepower, colored by transmission.

```
dt_03 %>%
  mutate(ride_cnt_=as.numeric(ride_cnt)) %>%
  ggplot(., mapping=aes(x=ride_mth, y=ride_cnt_, color=member_typ)) +
  geom_boxplot() +
  facet_wrap(~rideable_type)
```

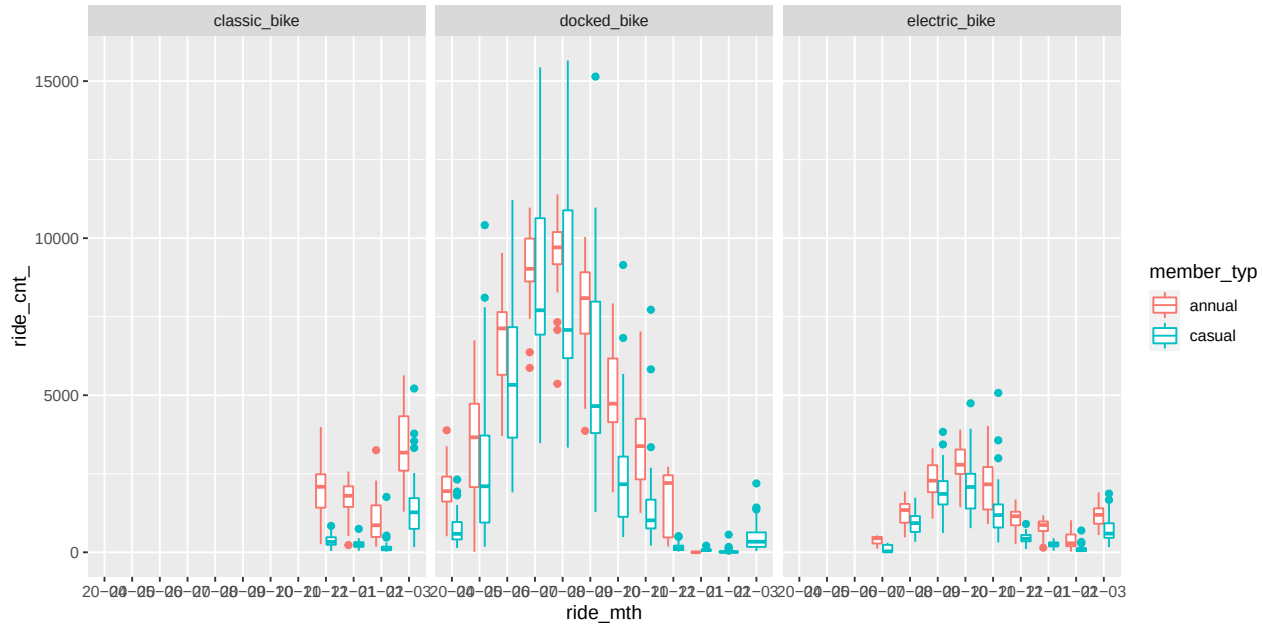

Figure 4: MPG vs horsepower, colored by transmission.

From the plot, the annual member has the high ride per day.

```
dt_03 %>%
  mutate(ride_cnt_=as.numeric(ride_cnt)) %>%
  ggplot(., mapping=aes(x=ride_mth, y=ride_cnt_, color=member_typ, fill=member_typ)) +
```

```
  stat_summary(fun.y=median, geom="point", shape=21, size=4) +
  facet_wrap(~rideable_type)
```

```
## Warning: `fun.y` is deprecated. Use `fun` instead.
```

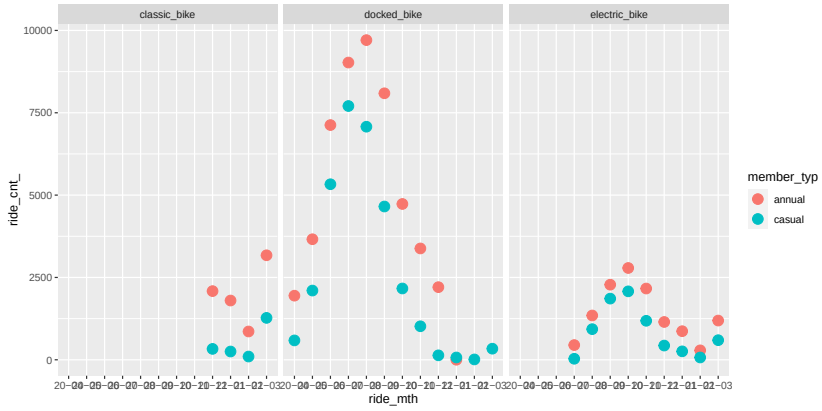

Figure 5: MPG vs horsepower, colored by transmission.

Summary of Key Findings and Recommendations

We report that annual members appear to be more motivated to ride as seen in their higher average rides per day.

Appendix

Code to import data to PostgreSQL pgAdmin 14

```
for ? in {202004, ..., 202110}
  create table if not exists divvy_tripdata_? (
    ride_id text not null,
    rideable_type text,
    started_at timestamp,
    ended_at timestamp,
    start_station_name text,
    start_station_id text,
    end_station_name text,
    end_station_id text,
    start_lat numeric,
    start_lng numeric,
    end_lat numeric,
    end_lng numeric,
    member_casual text,
    primary key (ride_id)
```

```
  );

  copy public.divvy_tripdata_?
  from 'C:\..\bike_rental_analysis\?-divvy-tripdata.csv'
  delimiter ',' csv header;

create table divvy_tripdata as
  select * from divvy_tripdata_202004 union all
  select * from divvy_tripdata_202005 union all
  select * from divvy_tripdata_202006 union all
  select * from divvy_tripdata_202007 union all
  select * from divvy_tripdata_202008 union all
  select * from divvy_tripdata_202009 union all
  select * from divvy_tripdata_202010 union all
  select * from divvy_tripdata_202011 union all
  select * from divvy_tripdata_202012 union all
  select * from divvy_tripdata_202101 union all
  select * from divvy_tripdata_202102 union all
  select * from divvy_tripdata_202103 union all
  select * from divvy_tripdata_202104 union all
  select * from divvy_tripdata_202105 union all
  select * from divvy_tripdata_202106 union all
  select * from divvy_tripdata_202107 union all
  select * from divvy_tripdata_202108 union all
  select * from divvy_tripdata_202109 union all
  select * from divvy_tripdata_202110
;

create table divvy_tripdata_ as
  select ride_id, rideable_type, started_at, ended_at, start_station_name,
    start_station_id, end_station_name, end_station_id, start_lat,
    start_lng, end_lat, end_lng,
    case when member_casual='member' then 'annual' else member_casual end as member_typ
  from divvy_tripdata
```

```
xfun::session_info(c("rmarkdown", "tidyverse", "tufte", "DBI"))
```

```
## R version 4.1.0 (2021-05-18)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 19042)
##
## Locale:
##   LC_COLLATE=English_United States.1252
##   LC_CTYPE=English_United States.1252
##   LC_MONETARY=English_United States.1252
```

```
##   LC_NUMERIC=C
##   LC_TIME=English_United States.1252
##
## Package version:
##   askpass_1.1         assertthat_0.2.1    backports_1.3.0
##   base64enc_0.1.3     bit_4.0.4           bit64_4.0.5
##   blob_1.2.2          broom_0.7.10        callr_3.7.0
##   cellranger_1.1.0    cli_3.1.0           clipr_0.7.1
##   colorspace_2.0.2    cpp11_0.4.2         crayon_1.4.2
##   curl_4.3.2          data.table_1.14.2   DBI_1.1.1
##   dbplyr_2.1.1        digest_0.6.29       dplyr_1.0.7
##   dtplyr_1.1.0        ellipsis_0.3.2      evaluate_0.14
##   fansi_0.5.0         farver_2.1.0        fastmap_1.1.0
##   forcats_0.5.1       fs_1.5.1            gargle_1.2.0
##   generics_0.1.1      ggplot2_3.3.5       glue_1.5.1
##   googledrive_2.0.0   googlesheets4_1.0.0 graphics_4.1.0
##   grDevices_4.1.0     grid_4.1.0          gtable_0.3.0
##   haven_2.4.3         highr_0.9           hms_1.1.1
##   htmltools_0.5.2     httr_1.4.2          ids_1.0.1
##   isoband_0.2.5       jquerylib_0.1.4     jsonlite_1.7.2
##   knitr_1.36          labeling_0.4.2      lattice_0.20.44
##   lifecycle_1.0.1     lubridate_1.8.0     magrittr_2.0.1
##   MASS_7.3.54         Matrix_1.3.3        methods_4.1.0
##   mgcv_1.8.35         mime_0.12           modelr_0.1.8
##   munsell_0.5.0       nlme_3.1.152        openssl_1.4.5
##   pillar_1.6.4        pkgconfig_2.0.3     prettyunits_1.1.1
##   processx_3.5.2      progress_1.2.2      ps_1.6.0
##   purrr_0.3.4         R6_2.5.1            rappdirs_0.3.3
##   RColorBrewer_1.1.2  Rcpp_1.0.7          readr_2.1.1
##   readxl_1.3.1        rematch_1.0.1       rematch2_2.1.2
##   reprex_2.0.1        rlang_0.4.12        rmarkdown_2.11
##   rstudioapi_0.13     rvest_1.0.2         scales_1.1.1
##   selectr_0.4.2       splines_4.1.0       stats_4.1.0
##   stringi_1.7.6       stringr_1.4.0       sys_3.4
##   tibble_3.1.6        tidyr_1.1.4         tidyselect_1.1.1
##   tidyverse_1.3.1     tinytex_0.35        tools_4.1.0
##   tufte_0.10          tzdb_0.2.0          utf8_1.2.2
##   utils_4.1.0         uuid_1.0.3          vctrs_0.3.8
##   viridisLite_0.4.0   vroom_1.5.7         withr_2.4.3
##   xfun_0.28           xml2_1.3.3          yaml_2.2.1
##
## Pandoc version: 2.11.4
```