This will be a quick post on another tidyverse function, order_by. I'll admit, I don't use this one as often as arrange. It can be useful, though, if you don't want to permanently change the order of your dataset but want to use functions that require ordering the data. One example is the cumulative sum function (cumsum).

```
library(tidyverse)
```

```
## -- Attaching packages ------------------------------
--------------------------------------------------- tidyverse 1.3.0 --

##  ggplot2 3.2.1      purrr   0.3.3
##  tibble  2.1.3      dplyr   0.8.3
##  tidyr   1.0.0      stringr 1.4.0
##  readr   1.3.1      forcats 0.4.0

## -- Conflicts -------------------------------------------------------------
----------------------- tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

```
reads2019 <- read_csv("~/Downloads/Blogging A to Z/SaraReads2019_allrated.csv",
                      col_names = TRUE)
```

```
## Parsed with column specification:
## cols(
##   Title = col_character(),
##   Pages = col_double(),
##   date_started = col_character(),
##   date_read = col_character(),
##   Book.ID = col_double(),
##   Author = col_character(),
##   AdditionalAuthors = col_character(),
##   AverageRating = col_double(),
##   OriginalPublicationYear = col_double(),
##   read_time = col_double(),
##   MyRating = col_double(),
##   Gender = col_double(),
##   Fiction = col_double(),
##   Childrens = col_double(),
##   Fantasy = col_double(),
##   SciFi = col_double(),
##   Mystery = col_double(),
##   SelfHelp = col_double()
## )
```

In my reading dataset, I could compute a cumulative sum of my days reading, then plot that against day of the year (days numbered from 1 to 365), to see where I might have had gaps in my reading habits.

```
library(lubridate)
```

```
##
## Attaching package: 'lubridate'

## The following object is masked from 'package:base':
##
##     date
```

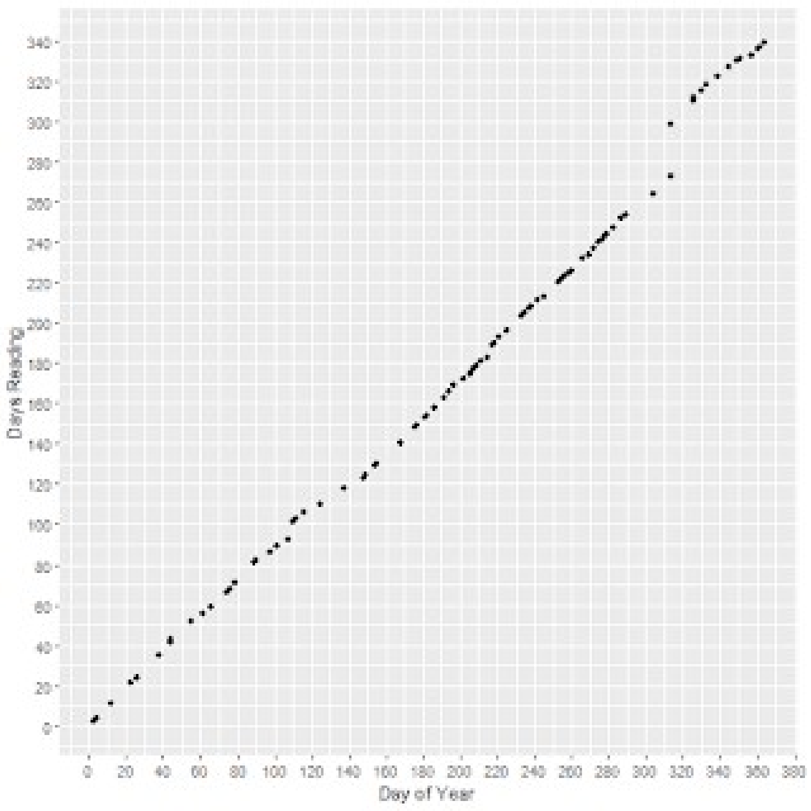
```
reads2019 <- reads2019 %>%
  mutate(date_started = as.Date(reads2019$date_started, format = '%m/%d/%Y'),
         date_read = as.Date(date_read, format = '%m/%d/%Y'),
         days_reading = order_by(date_read, cumsum(read_time)),
         DOY = yday(date_read))
```

A quick look at my data shows one issue though – my first book read in 2019 was actually started in 2018, so the days spent reading is greater than the numbered day of the year. I should probably correct that, then recompute my days_reading variable.

```
reads2019 <- reads2019 %>%
  mutate(read_time = ifelse(read_time > DOY, DOY, read_time),
         days_reading = order_by(date_read, cumsum(read_time)))
```

Now I can plot days reading against day of the year.

```
reads2019 %>%
  ggplot(aes(DOY, days_reading)) +
  geom_point() +
  scale_x_continuous(breaks = seq(0, 380, 20)) +
  scale_y_continuous(breaks = seq(0, 360, 20)) +
  xlab("Day of Year") +
  ylab("Days Reading")
```



Based on this chart, I have few gaps in my reading habits, but overall, read 341 days out of the year. There are also spots where I was clearly reading 2 books at once, as well as places where I finished one book one day, then started and finished another one that same day (some of the books I read were short, though).

Another potential use for order_by is with the fill function, which fills missing values from the next or previous column entry. If you were doing a study where people receive repeated measures and miss one of the measurements, or if your dataset only notes a new value when the previous one has changed, you could use fill to copy previous values into the empty cells.