These next two posts will deal with formatting scales in ggplot2 – x-axis, y-axis – so I'll try to limit the amount of overlap and repetition.

Let's say I wanted to plot my reading over time, specifically as a cumulative sum of pages across the year. My x-axis will be a date. Since my reads2019 file initially formats my dates as character, I'll need to use my mutate code to turn them into dates, plus compute my cumulative sum of pages read.

```
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()
```

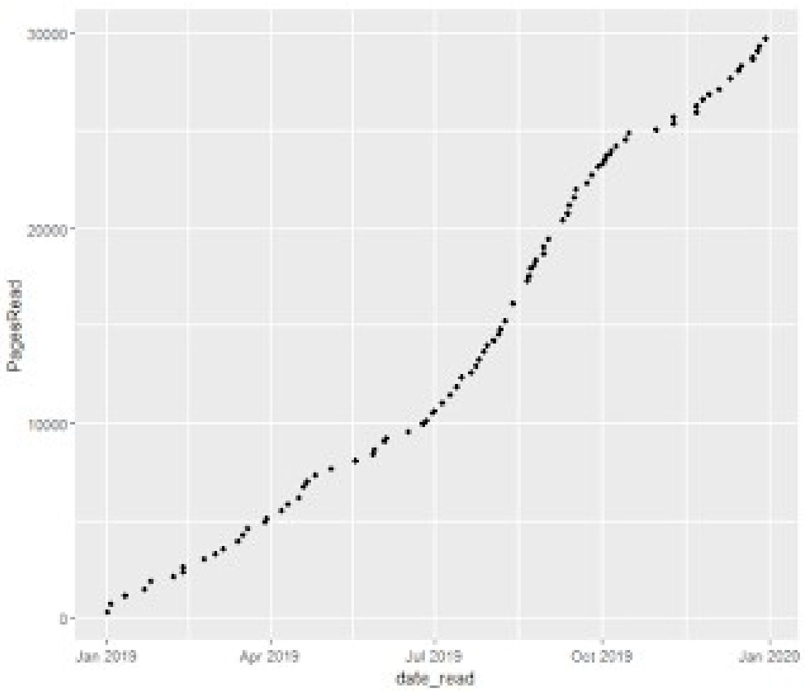
```
reads2019 <- read_csv("~/Downloads/Blogging A to Z/SaraReads2019_allchanges.
csv",
                      col_names = TRUE)
```

```
## 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()
## )
```

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

This gives me the variables I need to plot my pages read over time.

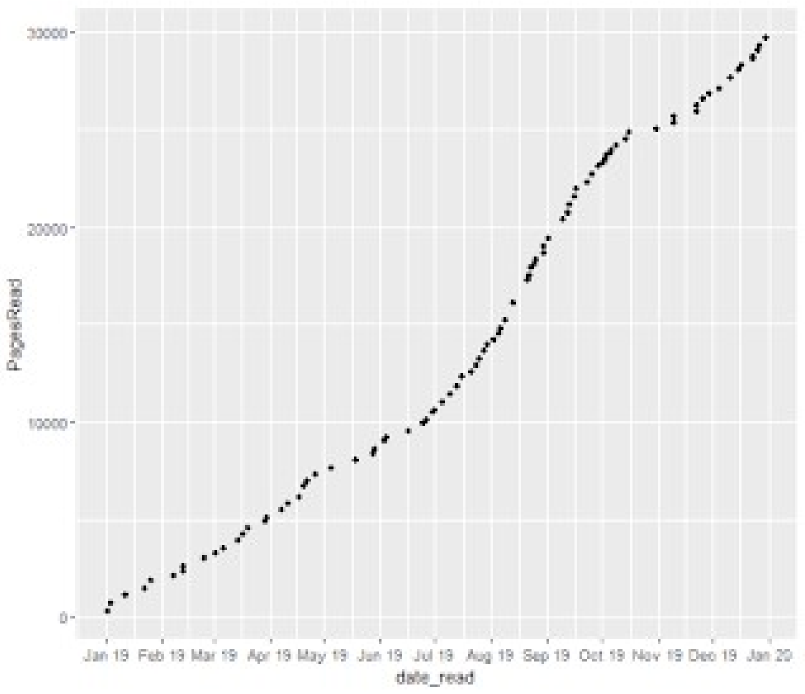
```
reads2019 %>%
  ggplot(aes(date_read, PagesRead)) +
  geom_point()
```



ggplot2 did a fine job of creating this plot using default settings. Since my date_read variable is a date, the plot automatically ordered date_read, formatted as “Month Year”, and used quarters as breaks. But we can still use the scale_x functions to make this plot look even better.

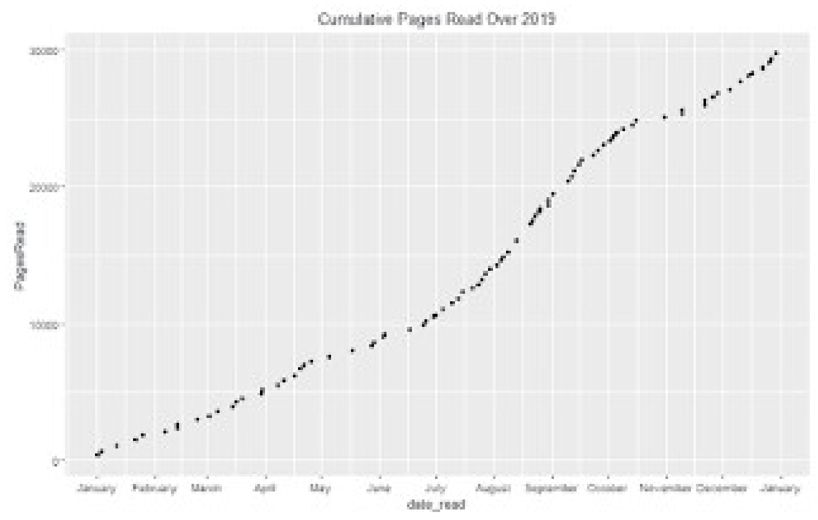
One way could be to format years as 2-digit instead of 4. We could also have month breaks instead of quarters.

```
reads2019 %>%
  ggplot(aes(date_read, PagesRead)) +
  geom_point() +
  scale_x_date(date_labels = "%b %y",
               date_breaks = "1 month")
```



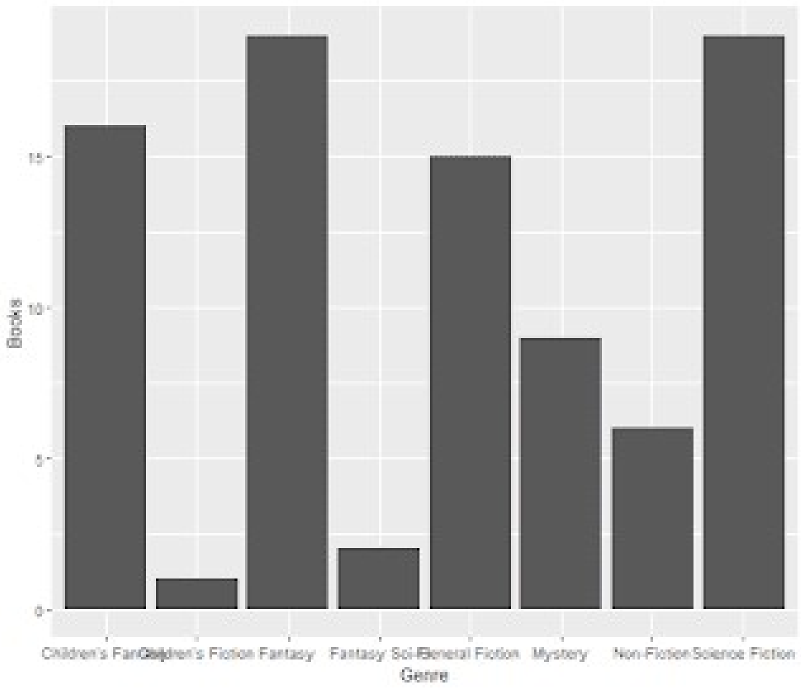
Of course, we could drop year completely and just show month, since all of this data is for 2019. We could then note that in the title instead.

```
reads2019 %>%
  ggplot(aes(date_read, PagesRead)) +
  geom_point() +
  scale_x_date(date_labels = "%B",
               date_breaks = "1 month") +
  labs(title = "Cumulative Pages Read Over 2019") +
  theme(plot.title = element_text(hjust = 0.5))
```



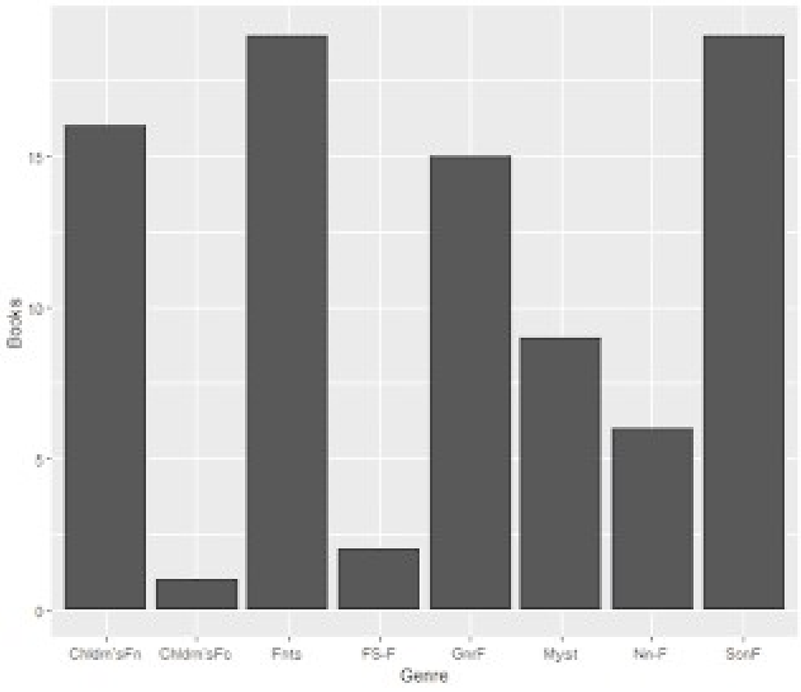
Tomorrow, I'll show some tricks for how we can format the y-axis of this plot. But let's see what else we can do to the x-axis. Let's create a bar graph with my genre data. I'll use the genre names I created for my summarized data last week.

```
genres <- reads2019 %>%
  group_by(Fiction, Childrens, Fantasy, SciFi, Mystery) %>%
  summarise(Books = n())

genres <- genres %>%
  bind_cols(Genre = c("Non-Fiction",
            "General Fiction",
            "Mystery",
            "Science Fiction",
            "Fantasy",
            "Fantasy Sci-Fi",
            "Children's Fiction",
            "Children's Fantasy"))

genres %>%
  ggplot(aes(Genre, Books)) +
  geom_col()
```



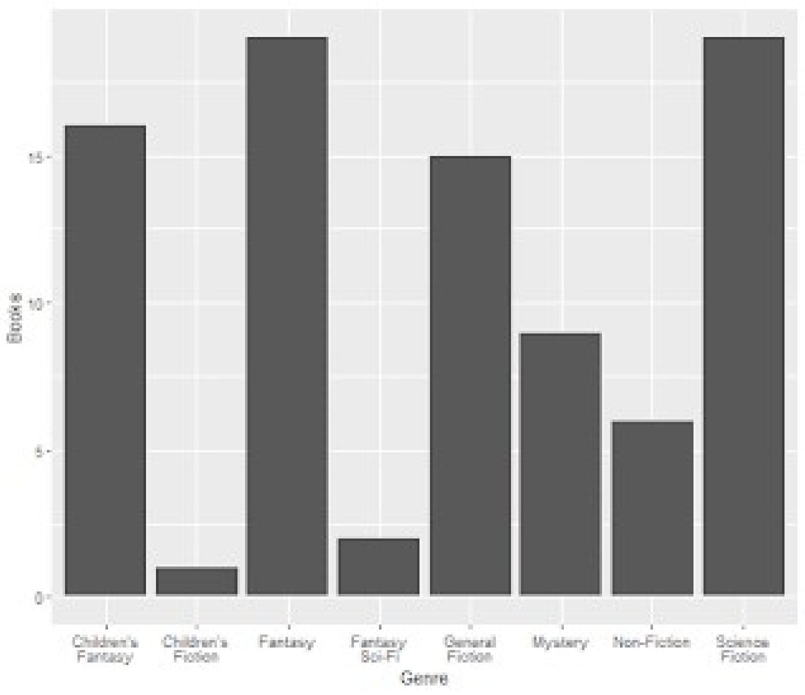
Unfortunately, my new genre names are a bit long, and overlap each other unless I make my plot really wide. There are a few ways I can deal with that. First, I could ask ggplot2 to abbreviate the names.

```
genres %>%
  ggplot(aes(Genre, Books)) +
  geom_col() +
  scale_x_discrete(labels = abbreviate)
```



These abbreviations were generated automatically by R, and I'm not a huge fan. A better way might be to add line breaks to any two-word genres. This Stack Overflow post gave me a function I can add to my scale_x_discrete to do just that.

```
genres %>%
  ggplot(aes(Genre, Books)) +
  geom_col() +
  scale_x_discrete(labels=function(x){sub("\\s", "\n", x)})
```



MUCH better!

As you can see, the scale_x function you use depends on the type of data you're working with. For dates, scale_x_date; for categories, scale_x_discrete. Tomorrow, we'll show some ways to format continuous data, since that's often what you see on the y-axis. See you then!

By the way, this is my 1000th post on my blog!