Yesterday, I talked about scale_x. Today, I'll continue on that topic, focusing on the y-axis.

The key to using any of the scale_ functions is to know what sort of data you're working with (e.g., date, continuous, discrete). Yesterday, I talked about scale_x_date and scale_x_discrete. We often put these types of data on the x-axis, while the y-axis is frequently used for counts. When displaying counts, we want to think about the major breaks that make sense, as well as any additional formatting to make them easier to read.

If I go back to my pages over time plot, you'll notice the major breaks are in the tens of thousands. We're generally used to seeing those values with a comma separating the thousands from the hundreds. I could add those to my plot like this (with a little help from the scales package).

```
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)
```

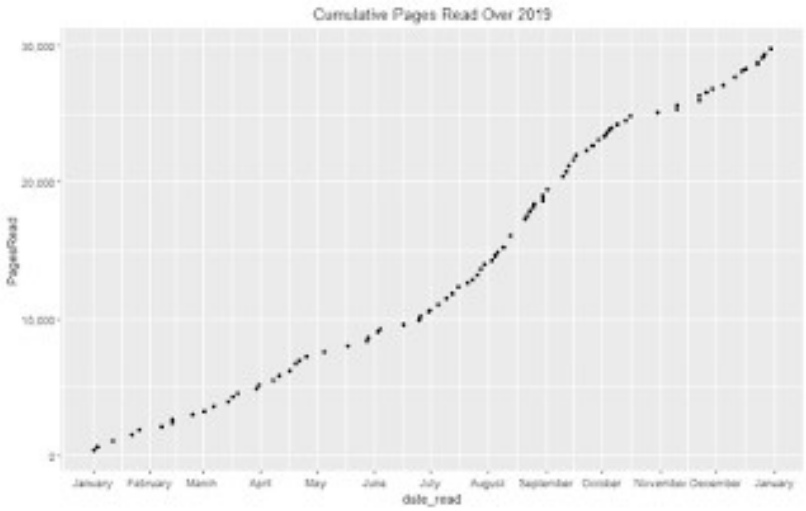
```
## Parsed with column specification:
## cols(
##   Title = col_character(),
##   Pages = col_double(),
##   date_started = col_character(),
##   date_read = col_character(),
##   Book.ID = col_double(),
##   Author = col_character(),
##   AdditionalAuthors = col_character(),
##   AverageRating = 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)))
```

```
library(scales)
```

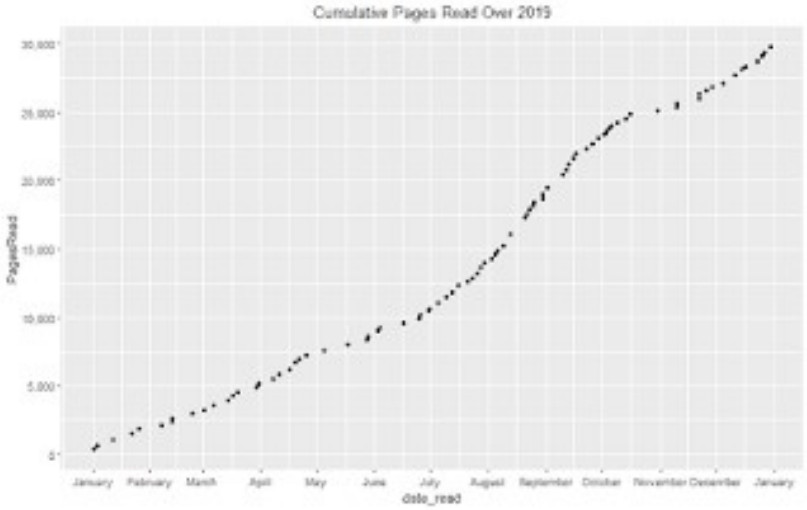
```
##
## Attaching package: 'scales'

## The following object is masked from 'package:purrr':
##
##     discard

## The following object is masked from 'package:readr':
##
##     col_factor
```

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



I could also add more major breaks.

```
reads2019 %>%
  ggplot(aes(date_read, PagesRead)) +
  geom_point() +
  scale_x_date(date_labels = "%B",
               date_breaks = "1 month") +
  scale_y_continuous(labels = comma,
                     breaks = seq(0, 30000, 5000)) +
  labs(title = "Cumulative Pages Read Over 2019") +
  theme(plot.title = element_text(hjust = 0.5))
```



The scales package offers other ways to format data besides the 3 I've shown in this series (log transformation, percent, and now continuous with comma). It also lets you format data with currency, bytes, ranks, and scientific notation.