

Here we are at the last post in Blogging A to Z! Today, I want to talk about adding additional axes to your ggplot, using the options for fill or color. While these aren't true z-axes in the geometric sense, I think of them as a third, z, axis.

Some of you may be surprised to learn that fill and color are different, and that you could use one or both in a given plot.

Color refers to the outline of the object (bar, piechart wedge, etc.), while fill refers to the inside of the object. For scatterplots, the default shape doesn't have a fill, so you'd just use color to change the appearance of those points.

Let's recreate the pages read over 2019 chart, but this time, I'll just use fiction books and separate them as either fantasy or other fiction; this divides that dataset pretty evenly in half. Here's how I'd generate the pages read over time separately by those two genre categories.

```
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()
## )

fantasy <- reads2019 %>%
  filter(Fiction == 1) %>%
  mutate(date_read = as.Date(date_read, format = '%m/%d/%Y'),
```

```

Fantasy = factor(Fantasy, levels = c(0,1),
                  labels = c("Other Fiction",
                             "Fantasy")) %>%

group_by(Fantasy) %>%
mutate(GenreRead = order_by(date_read, cumsum(Pages))) %>%
ungroup()

```

Now I'd just plug that information into my ggplot code, but add a third variable in the aesthetics (aes) for ggplot – color = Fantasy.

```

library(scales)

##
## Attaching package: 'scales'

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

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

myplot <- fantasy %>%
  ggplot(aes(date_read, GenreRead, color = Fantasy)) +
  geom_point() +
  xlab("Date") +
  ylab("Pages") +
  scale_x_date(date_labels = "%b",
               date_breaks = "1 month") +
  scale_y_continuous(labels = comma, breaks = seq(0,30000,5000)) +
  labs(color = "Genre of Fiction")

```

This plot uses the default R colorscheme. I could change those colors, using an existing colorscheme, or define my own. Let's make a fivethirtyeight style figure, using their theme for the overall plot, and their color scheme for the genre variable.

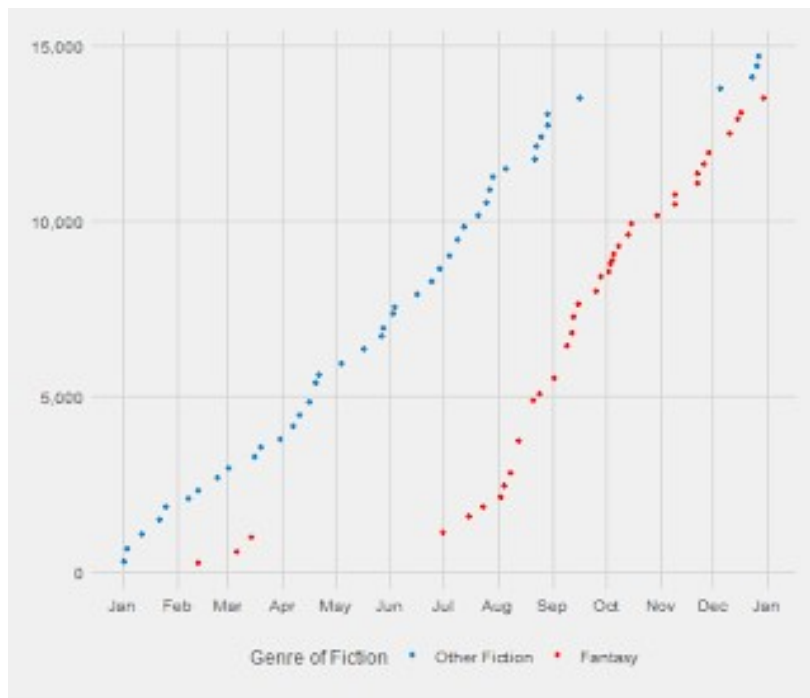
```

library(ggthemes)

## Warning: package 'ggthemes' was built under R version 3.6.3

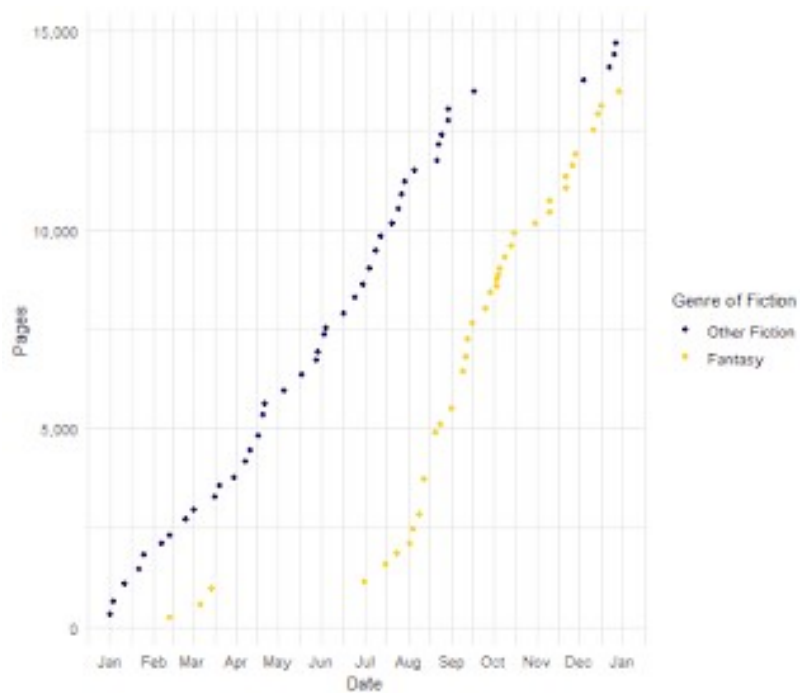
myplot +
  scale_color_fivethirtyeight() +
  theme_fivethirtyeight()

```



I can also specify my own colors.

```
myplot +
  scale_color_manual(values = c("#4b0082", "#ffd700")) +
  theme_minimal()
```



The `geom_point` offers many point shapes; 21-25 allow you to specify both color and fill. But for the rest, only use color.

```
library(ggpubr)

## Warning: package 'ggpubr' was built under R version 3.6.3

## Loading required package: magrittr

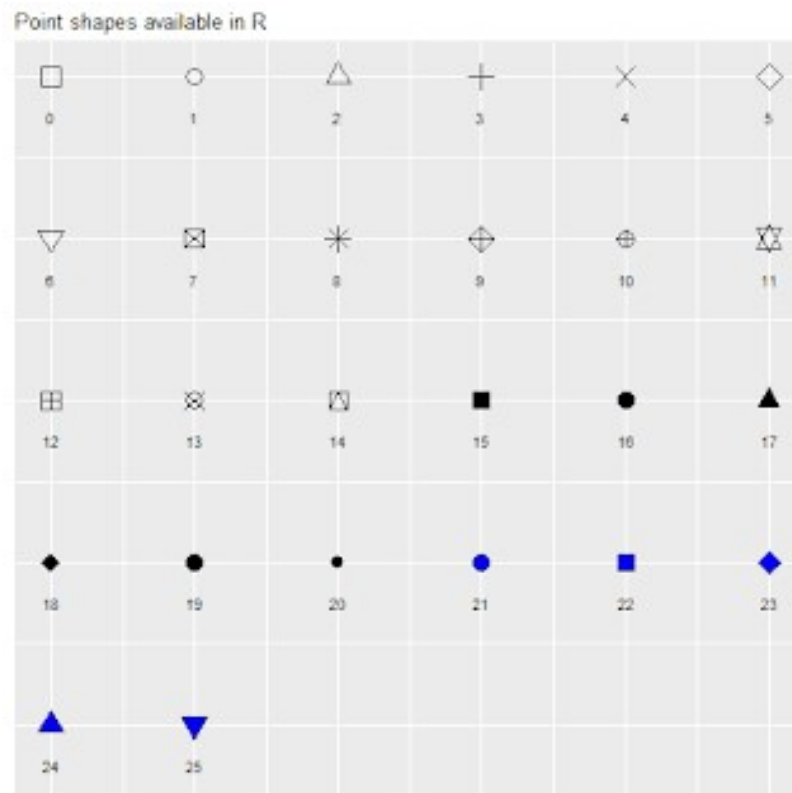
##
## Attaching package: 'magrittr'
```

```
## The following object is masked from 'package:purrr':
##
##   set_names

## The following object is masked from 'package:tidyr':
##
##   extract

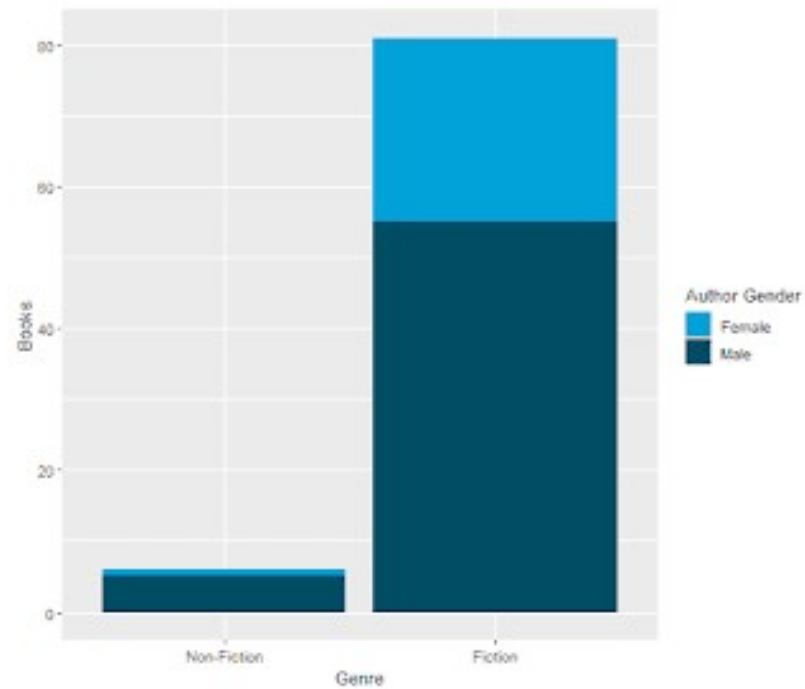
ggpubr::show_point_shapes()

## Scale for 'y' is already present. Adding another scale for 'y', which will
## replace the existing scale.
```



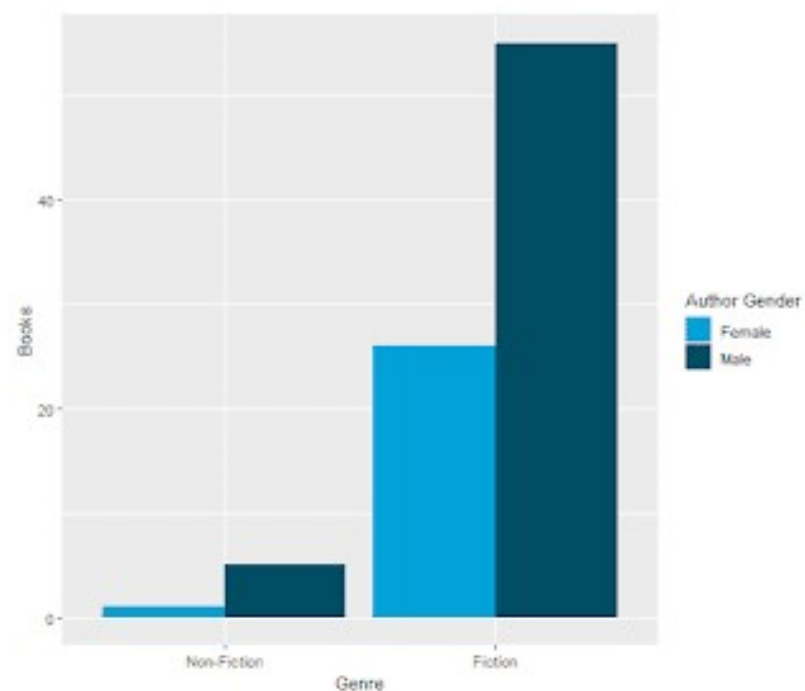
Of course, you may have plots where changing fill is best, such as on a bar plot. In my summarize example, I created a stacked bar chart of fiction versus non-fiction with author gender as the fill.

```
reads2019 %>%
  mutate(Gender = factor(Gender, levels = c(0,1),
                        labels = c("Male",
                                   "Female")),
         Fiction = factor(Fiction, levels = c(0,1),
                        labels = c("Non-Fiction",
                                   "Fiction"),
                        ordered = TRUE)) %>%
  group_by(Gender, Fiction) %>%
  summarise(Books = n()) %>%
  ggplot(aes(Fiction, Books, fill = reorder(Gender, desc(Gender)))) +
  geom_col() +
  scale_fill_economist() +
  xlab("Genre") +
  labs(fill = "Author Gender")
```



Stacking is the default, but I could also have the bars next to each other.

```
reads2019 %>%
  mutate(Gender = factor(Gender, levels = c(0,1),
                        labels = c("Male",
                                   "Female")),
         Fiction = factor(Fiction, levels = c(0,1),
                        labels = c("Non-Fiction",
                                   "Fiction"),
                        ordered = TRUE)) %>%
  group_by(Gender, Fiction) %>%
  summarise(Books = n()) %>%
  ggplot(aes(Fiction, Books, fill = reorder(Gender, desc(Gender)))) +
  geom_col(position = "dodge") +
  scale_fill_economist() +
  xlab("Genre") +
  labs(fill = "Author Gender")
```

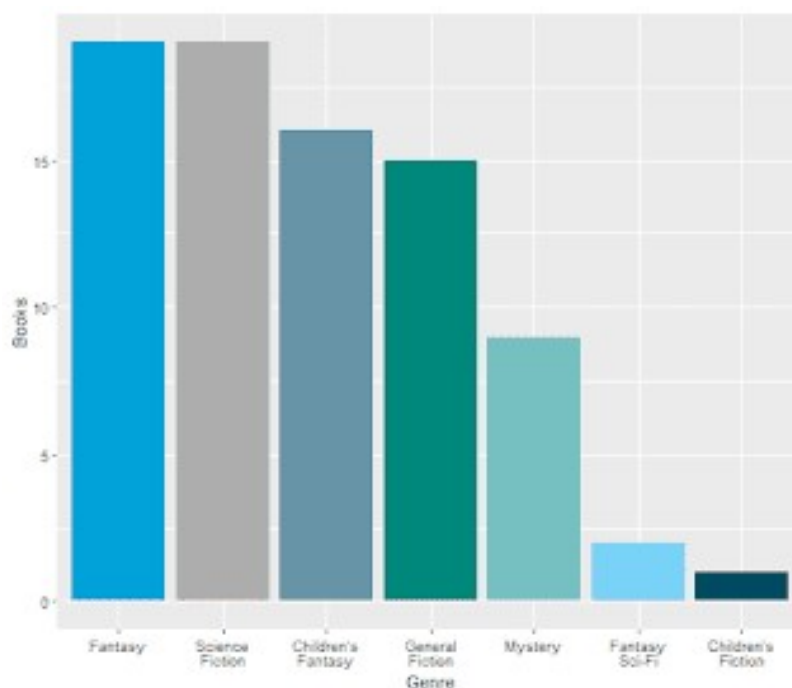


You can also use fill (or color) with the same variable you used for x or y; that is, instead of having it be a third scale, it could add some color and separation to distinguish categories from the x or y variable. This is especially helpful if you have multiple categories being plotted, because it helps break up the wall of bars. If you do this, I'd recommend choosing a color palette with highly complementary colors, rather than highly contrasting ones; you probably also want to drop the legend, though, since the axis will also be labeled.

```
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 %>%
  filter(Genre != "Non-Fiction") %>%
  ggplot(aes(reorder(Genre, -Books), Books, fill = Genre)) +
  geom_col() +
  xlab("Genre") +
  scale_x_discrete(labels=function(x){sub("\\s", "\n", x)}) +
  scale_fill_economist() +
  theme(legend.position = "none")
```



If you only have a couple categories and want to draw a contrast, that's when you can use contrasting shades: for instance, at work, when I plot performance on an item, I use red for incorrect and blue for correct, to maximize the contrast between the two performance levels for whatever data I'm presenting.