Background Data in ggplot2

Emeka Nwosu

September 4, 2016

I came across this article through the Data Elixir newsletter about plotting background data with ggplot2 in R. The plots reminded me of the GameCube sales data project I did last year and how much better the plots I made for that would have looked using this technique. So I applied the techniques from the article to my data for a bit of practice and to see if I notice anything new seeing the data represented differently.

Obviously, I'm going to load ggplot2. As for the data, I'm just keeping it simple. The only columns I'll need for this example are Game, Metacritic Score, and Sales. I'm also going to create an additional column called "Console" for the console in which the game came out on. I'll need to do this to distinguish the games after I combine everything into a single data frame.

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

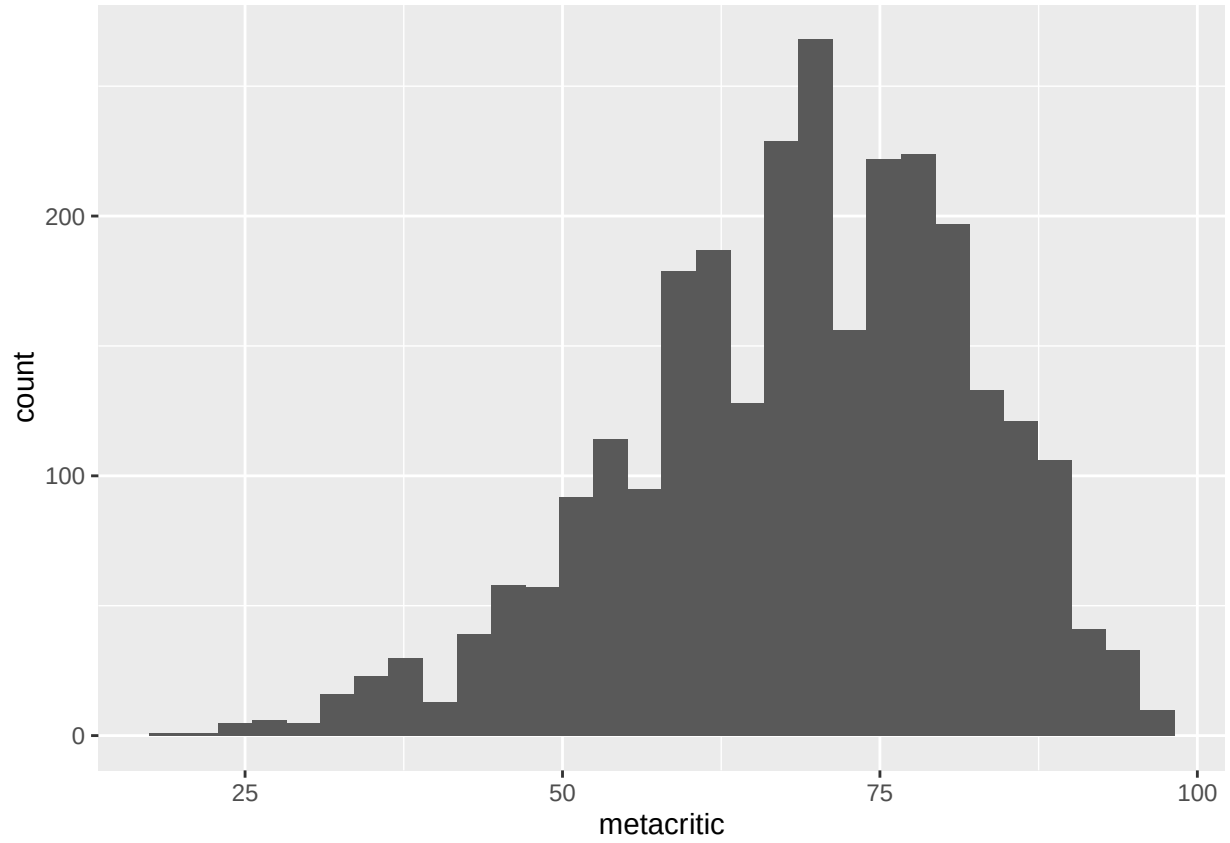
```
library(ggplot2)

gcn <- read.csv("~/Emekas Documents/R-Directory/Don Data/gcn.csv")
console <- rep("GameCube", length = nrow(gcn))
gcn <- data.frame(gcn$game, gcn$metacritic, gcn$sales_total, console)
names(gcn) <- c("game", "metacritic", "sales", "console")

oxb <- read.csv("~/Emekas Documents/R-Directory/Don Data/oxb.csv")
console <- rep("Xbox", length = nrow(oxb))
oxb <- data.frame(oxb$game, oxb$metacritic, oxb$sales_total, console)
names(oxb) <- c("game", "metacritic", "sales", "console")

ps2 <- read.csv("~/Emekas Documents/R-Directory/Don Data/ps2.csv")
console <- rep("Playstation 2", length = nrow(ps2))
ps2 <- data.frame(ps2$game, ps2$metacritic, ps2$sales_total, console)
names(ps2) <- c("game", "metacritic", "sales", "console")

sixgen <- rbind(gcn, oxb, ps2) ## called "sixgen" for sixth console generation
```

First is a regular histogram of Metacritic scores. This is just a very basic plot.

```
ggplot(sixgen, aes(x = metacritic)) +
  geom_histogram()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1691 rows containing non-finite values (stat_bin).
```

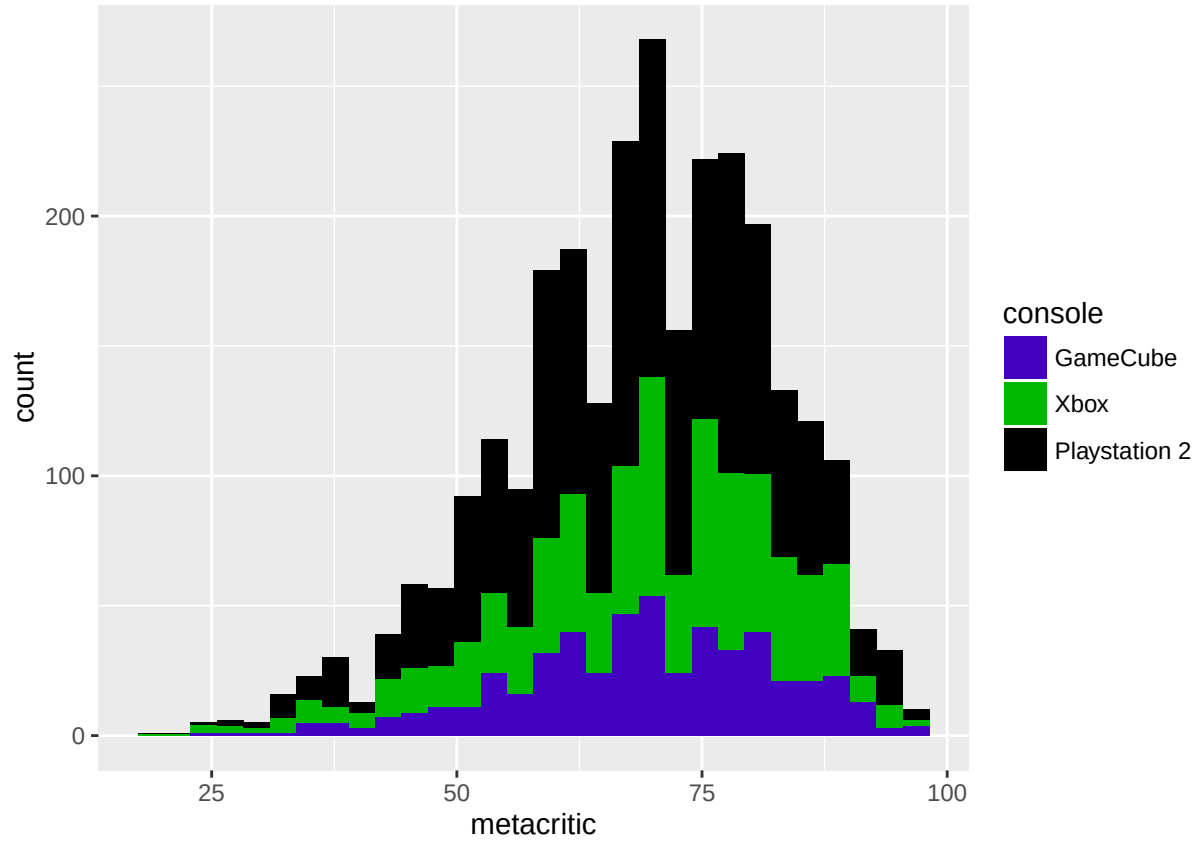


Now his is how I would normally display a histogram while distinguishing between the different groups.

```
ggplot(sixgen, aes(x = metacritic, fill = console)) +
  geom_histogram() +
scale_fill_manual(values = c("Playstation 2" = "#000000", "GameCube" = "#4200C0", "Xbox" = "#00B800"))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1691 rows containing non-finite values (stat_bin).
```

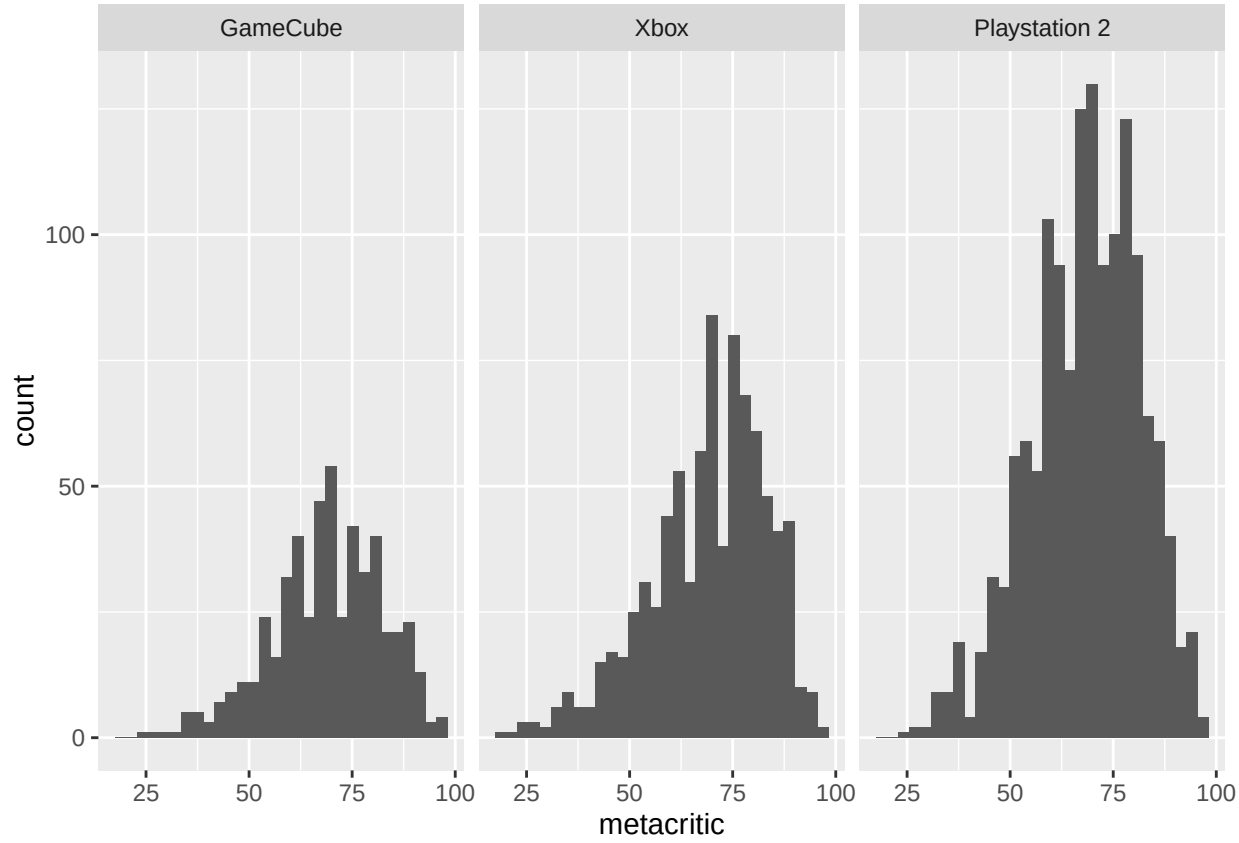


Another way to distinguish between the different groups is to facet by the group variable, making three separate plots.

```
ggplot(sixgen, aes(x = metacritic)) +
  geom_histogram() +
  facet_wrap(~ console)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

```
## Warning: Removed 1691 rows containing non-finite values (stat_bin).
```



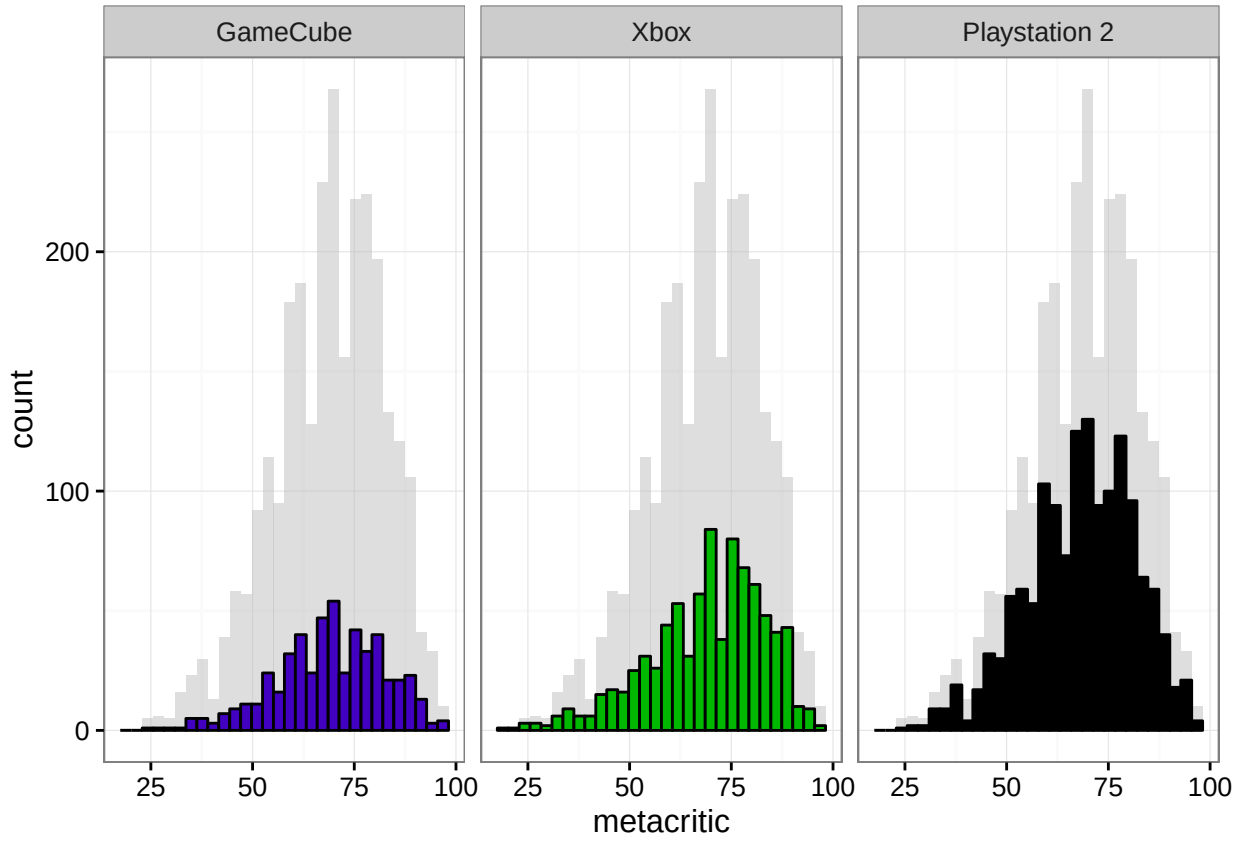
Now in order get fancy and plot background data, I'll need a version of the data frame without the separation by groups. This way, I can make that first histogram I showed.

```
sixgen2 <- sixgen[,-4]
```

Now we can plot the histograms, separated by the categorical "Console" variable in red.

```
ggplot(sixgen, aes(x = metacritic, fill = console)) +
  geom_histogram(data = sixgen2, fill = "grey", alpha = .5) +
  geom_histogram(colour = "black") +
  scale_fill_manual(values = c("Playstation 2" = "#000000", "GameCube" = "#4200C0", "Xbox" = "#00B800")) +
  facet_wrap(~ console) +
  guides(fill = FALSE) +  # to remove the legend
  theme_bw()
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 5073 rows containing non-finite values (stat_bin).

## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.

## Warning: Removed 1691 rows containing non-finite values (stat_bin).
```

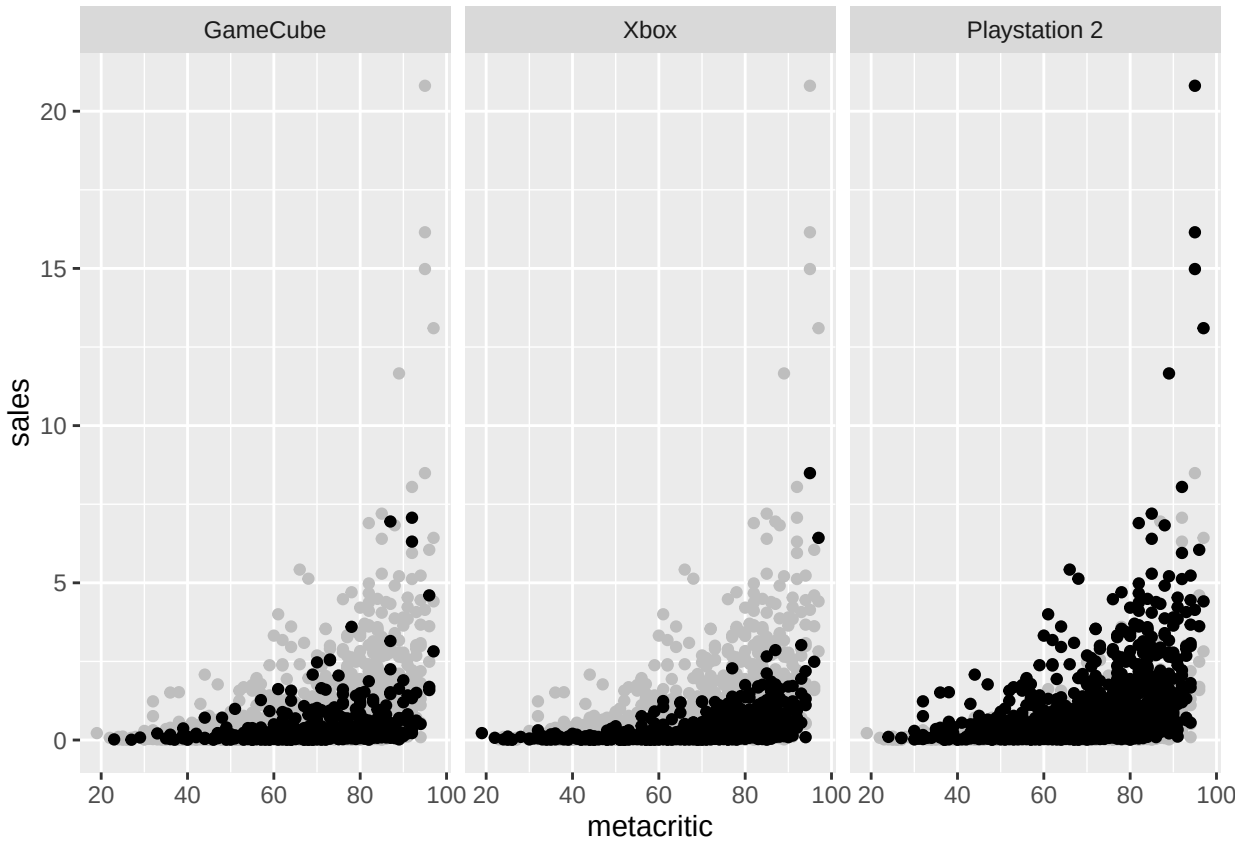


Here's the same technique done with a dot plot

```
ggplot(sixgen, aes(x = metacritic, y = sales)) +
  geom_point(data = sixgen2, colour = "grey") +
  facet_wrap(~ console) +
  geom_point() +
  facet_wrap(~ console)
```

```
## Warning: Removed 5337 rows containing missing values (geom_point).
```

```
## Warning: Removed 1779 rows containing missing values (geom_point).
```

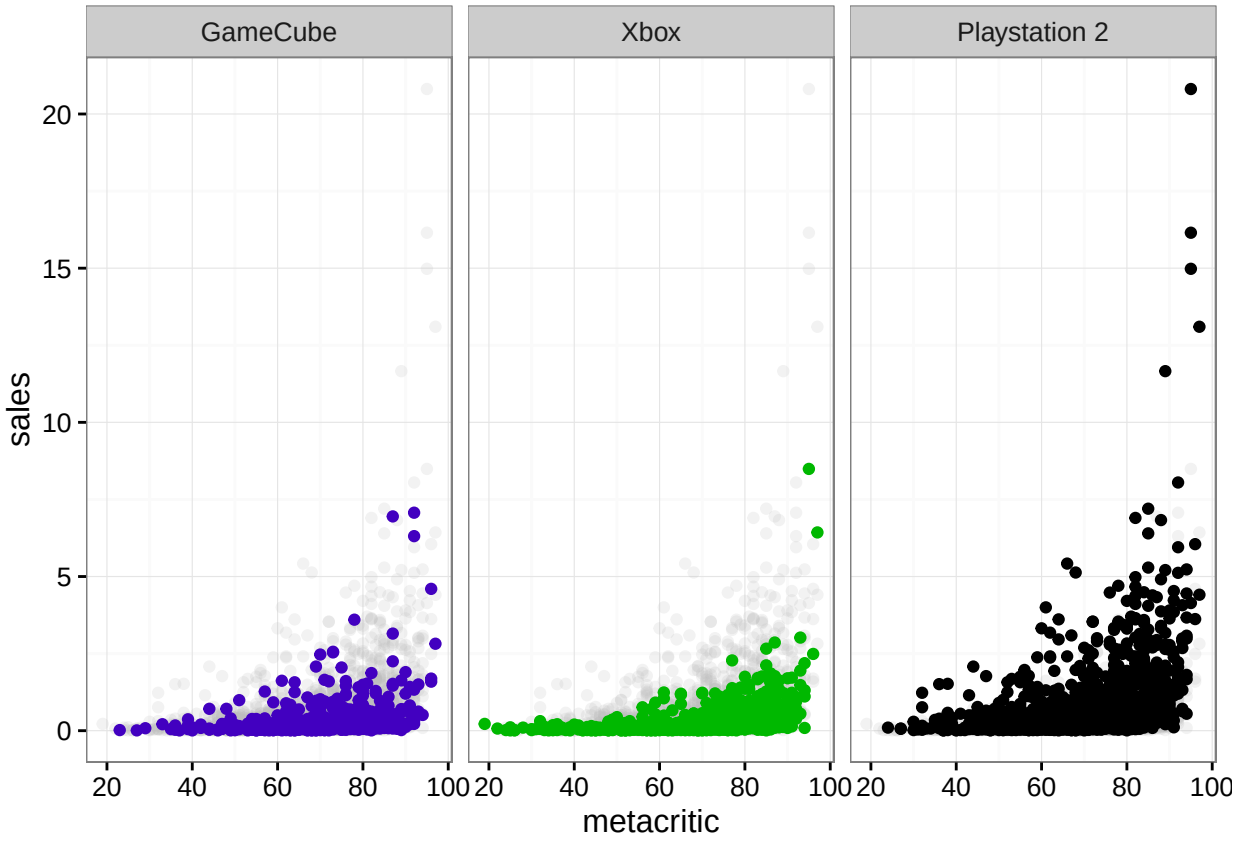


and with the color distinction.

```
ggplot(sixgen, aes(x = metacritic, y = sales, colour = console)) +
  geom_point(data = sixgen2, colour = "grey", alpha = .2) +
  geom_point() +
  scale_color_manual(values = c("Playstation 2" = "#000000", "GameCube" = "#4200C0", "Xbox" = "#00B800"
  facet_wrap(~ console) +
  guides(colour = FALSE) +
  theme_bw()
```

```
## Warning: Removed 5337 rows containing missing values (geom_point).
```

```
## Warning: Removed 1779 rows containing missing values (geom_point).
```



That's all I have to show for now. I'm sure there is a lot more that can be done with this technique. I just needed to test it out myself so I can remember to use it in future projects.