Code

STAT 209: Lab3

Using Color and Themes in `ggplot2`

Goal

Gain familiarity with the way that color palettes and aesthetic themes are specified in `ggplot2`, so that you can use both in your own visualizations.

Data

The examples will use the `storms` dataset which is included in the `dplyr` package (both `dplyr` and `ggplot2` are loaded by `tidyverse`, so it suffices to do `library(tidyverse)`). The data comes from the NOAA Atlantic hurricane database, and includes positions and attributes of 198 tropical storms measured at six hour intervals during each storm's lifetime.

Exercise 1 Load the `storms` data, and examine the documentation using the `?datasetname` syntax to see the definitions of the variables.

Solution

Hide

```
library(tidyverse)
data(storms)
?storms
# The last line generally wouldn't go in a Markdown document
# since it opens an external window
```

Exercise 2 Make a box plot depicting the distribution of wind speed grouped by the type of storm (tropical depression, tropical storm, or hurricane) and save it to a variable called `wind_boxplot`.

Solution

Hide

```
wind_boxplot <-
  ggplot(
    storms,
    aes(x = status, y = wind)
    ) +
  geom_boxplot()
wind_boxplot
```

Side note: reordering a categorical variable

You might notice that in the boxplot you get, the storm types are ordered alphabetically. In this case the types have a natural ordering based on their intensity: tropical depressions are the least severe, and hurricanes the most. Our plot will be more natural if we respect this natural ordering. We can do this as follows:

Code

Hide

```
storms.modified <-
  mutate(
    storms,
    status = factor(status, levels = c("tropical depression", "tropical storm", "hurri
  cane"))
  )

## Recreate the plot
wind_boxplot <-
  ggplot(
    storms.modified,
    aes(x = status, y = wind)
    ) +
  geom_boxplot()
wind_boxplot
```

Adding color

We can add some color to the plot by adding a redundant mapping: in addition to mapping `status` to the `x` dimension, we'll also map it to the `fill` feature (the interior color of the boxplots).

Code

Hide

```
wind_boxplot <-
  wind_boxplot +
    geom_boxplot(aes(fill = status))
```

Specifying colors manually

The default color palette looks pretty nice, but just for fun let's see how we could change it.

One option is to directly specify the colors we want R to use, using hexadecimal codes for the colors' RGB values. The colors in the following example come from one of my personal favorite palettes (you might recognize it from my website, for example), the Solarized (http://ethanschoonover.com/solarized) palette by Ethan Schoonover. In each code, the first two characters represent the red channel (on a 0-255 scale, encoded in base 16), the next two represent the green channel, and the last two represent the blue channel.

Code

Hide

```
wind_boxplot +
  scale_fill_manual(
    values = c("#268bd2", "#859900", "#dc322f"))
```

Using a predefined palette

Manually specifying colors allows very precise control, but it's tedious, time consuming, and brittle. It is generally better and easier to use predefined color palettes instead.

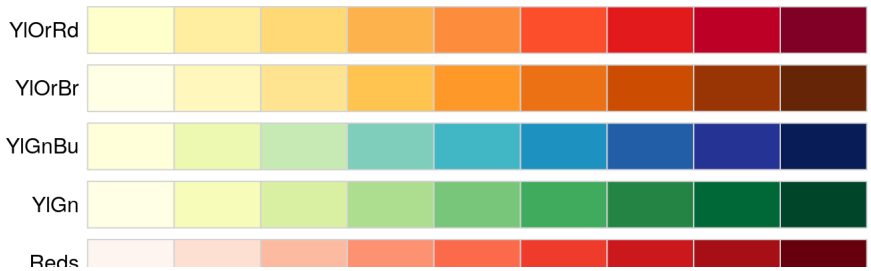
A fairly large and diverse collection of color palettes was developed by Cynthia Brewer (link (http://colorbrewer2.org)) and is available in the `RColorBrewer` package.

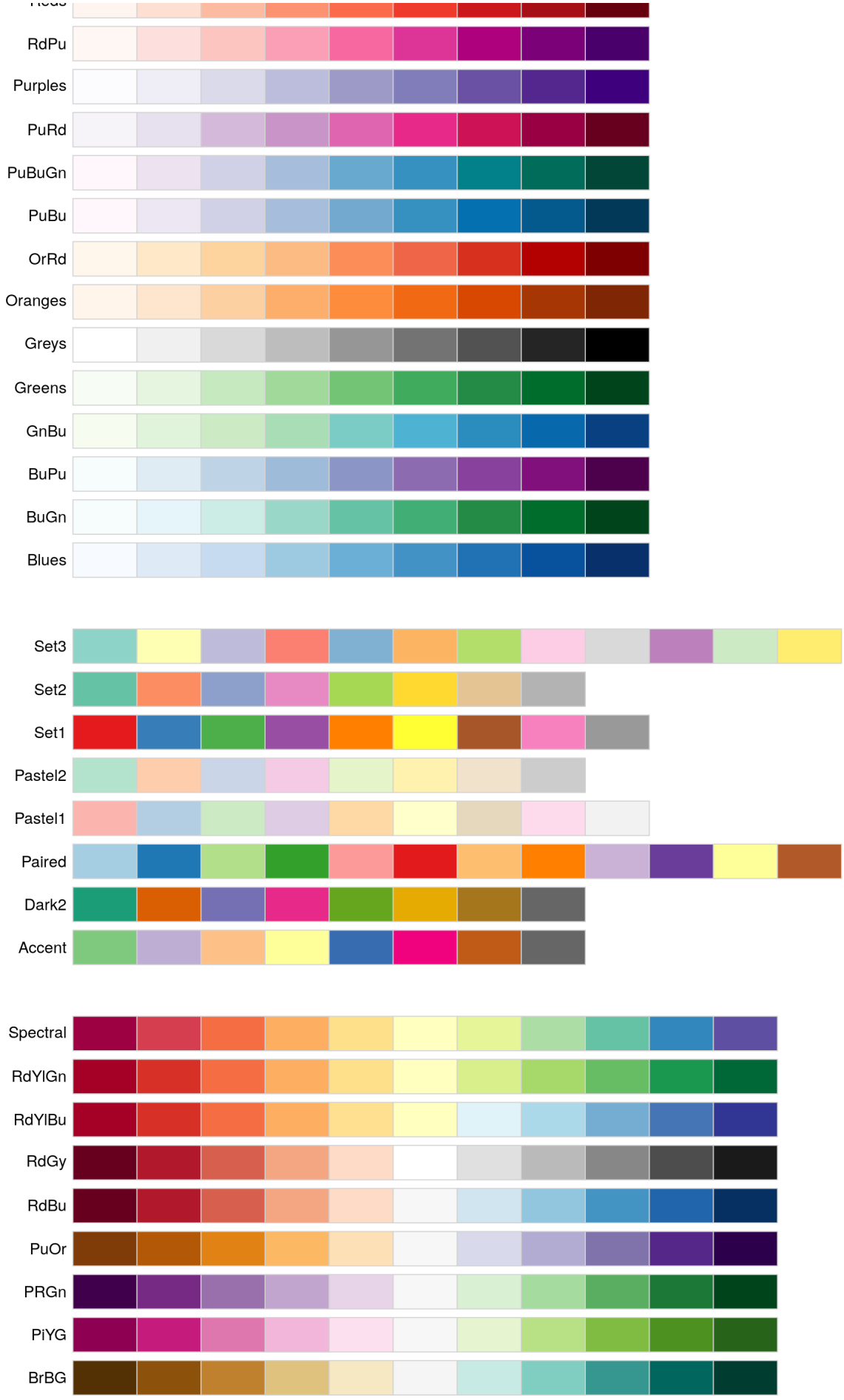
After loading the package, we can view the included palettes with `display.brewer.all()`

Code

Hide

```
library(RColorBrewer)
display.brewer.all()
```




RdPu
Purples
PuRd
PuBuGn
PuBu
OrRd
Oranges
Greys
Greens
GnBu
BuPu
BuGn
Blues
Set3
Set2
Set1
Pastel2
Pastel1
Paired
Dark2
Accent
Spectral
RdYlGn
RdYlBu
RdGy
RdBu
PuOr
PRGn
PiYG
BrBG

We can apply whichever palette we choose to the `fill` feature with the `scale_fill_brewer()` function:

Code

Hide

```
wind_boxplot +
  scale_fill_brewer(palette = "Set3")
```

Makes for a great Easter decoration. Or collection of golf shirts?

Exercise 3 Test out a few other palettes to find one that looks good.

Let's visualize the paths of some of these storms, using color to depict wind speed, so we can see how the intensity changed over the path of each storm.

To make the plot more manageable, we'll create a smaller dataset, `storms.1995` with only those storms that occurred in 1995.

Code

Hide

```
storms.1995 <-
  filter(storms.modified, year == 1995)
```

Exercise 4 With the restricted data, use facets to separate individual storms by their name, and map longitude to the `x` dimension, latitude to the `y` dimension, and wind speed to the `color` dimension.

Solution

Hide

```
trajectory_plot <-
  ggplot(
    storms.1995,
    aes(x = long, y = lat, color = wind)
    ) +
  geom_point() +
  facet_wrap(~name)
trajectory_plot
```

Exercise 5 Choose a different color palette for the plot above. Since we are mapping a quantitative variable to color, we should use either a

sequential or diverging palette. Which one do you think makes more sense here?

Applying global aesthetic themes using `ggthemes`

In addition to controlling the features involved in the aesthetic mapping, we can control the overall look of a plot, through things like font, style of the axes, style and color of the background, etc.

There are several predefined themes made available in the `ggthemes` package, including several that mimic the style of popular publications, like the Wall Street Journal, The Economist, FiveThirtyEight, and others. Nate Silver of FiveThirtyEight used natural disasters as a running example of predictive modeling in his book *The Signal and the Noise*, so let's apply the FiveThirtyEight theme to our hurricane box plot from above. Note that the specification of theme is in many cases separate from the specification of color palette, but many of the themes come with one or more accompanying palettes.

Code

Hide

```
library(ggthemes)
wind_boxplot +
  theme_fivethirtyeight() +
  scale_fill_fivethirtyeight()
```

Getting Credit

Exercise 6 Find an interesting dataset from one of the R packages that's already installed. You can list them by typing `data()` at the console, with no arguments, and examine the documentation for a given dataset with the `?datasetname` syntax. Use `ggplot()` to create a plot of something interesting from the dataset you choose, applying color and theme to style the plot. In a sentence or two, describe what your plot shows. Post your graphic and your short description to Slack – this time, let's post them to the `#lab3` channel so everyone can see each other's plots (since they will presumably all be different)!

This lab was adapted by Colin Dawson for STAT 209: Data Computing and Visualization, at Oberlin College, from a similar lab by Jordan Crouser and Ben Baumer used for SDS 192: Introduction to Data Science, at Smith College.