Week 1 Module 6: The RGB Color System (Bonus)

Module Overview and Learning Objectives

In this module, we will learn more about working with colors in R.

- In Section 1, we'll discuss the RGB color system, and how it is implemented in R.
- In Section 2, we'll visit a cool website that will generate lots of interesting color ideas.

When you've finished this module, you should be able to:

- Specify colors using the RGB system, including using explicit hexadecimal codes.

Section 1: Specifying Color

How can we specify a color precisely?

So far, we've used a simple naming system, where each color is associated with a string identifier such as "aquamarine2" or "mediumorchid3".

With the "Rcolor.pdf" file, we have hundreds of colors with associated names, and this is usually more than sufficient to find some good colors.

However, this is a limited system, and sometimes you might want to have a more structured method for specifying colors.

What happens if there is a particular color that we want to work with in our R graph?

For instance, perhaps we are preparing a Powerpoint presentation, and we want to coordinate the colors in the graph with the presentation color theme.

We can try to look in the "Rcolor.pdf" file for a color that matches the presentation color theme, but this is laborious, and there won't always be a good match.

If there isn't a named color in the "Rcolor.pdf" file that matches our desired color, then we need a different way to indicate the color.

The solution to this is to use a more systematic approach rather than a set of arbitrary associations of colors with character strings.

One such system is called the "RGB" color system.

Here, "R" stands for the color red, "G" stands for the color green, and "B" stands for the color blue.

By specifying the amounts of red, green, and blue, we can essentially generate all colors that are visible to the human eye.

Typically, the amounts of red, green, and blue are specified by integers (whole numbers) ranging from 0 to 255.

For instance, let's consider the color named "dodgerblue3".

It turns out that this can be specified by these values:

Color	Value
Red	24
Green	116
Blue	205

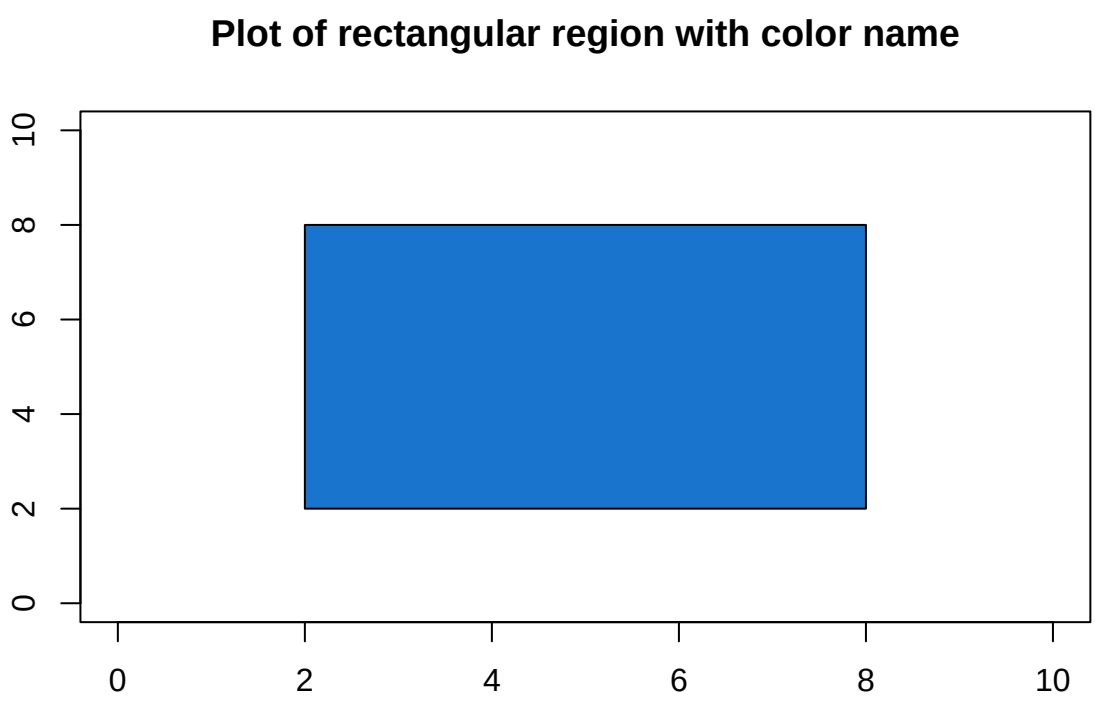
Let's perform an experiment to find out if this is really true.

First, we'll create a rectangle using the `polygon()` function, and fill it with the color "dodgerblue3":

```
# Example 1: Using a color name to specify a color

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 10),
    main = "Plot of rectangular region with color name",
    xlab = "",
    ylab = ""
)

polygon(
    x = c(2, 2, 8, 8),
    y = c(2, 8, 8, 2),
    col = "dodgerblue3"
)
```



Now let's create the same plot again, but this time, instead of specifying the color by using the name "dodgerblue3", we'll explicitly set the red, green, and blue values by using the *rgb()* function:

- The first argument is named "red", and this is a numeric value, ranging from 0 to 255.
- The second argument is named "green", and this is a numeric value, ranging from 0 to 255.
- The third argument is named "blue", and this is a numeric value, ranging from 0 to 255.
- Finally, there is a fourth argument, named *maxColorValue*, and you should set this equal to 255.

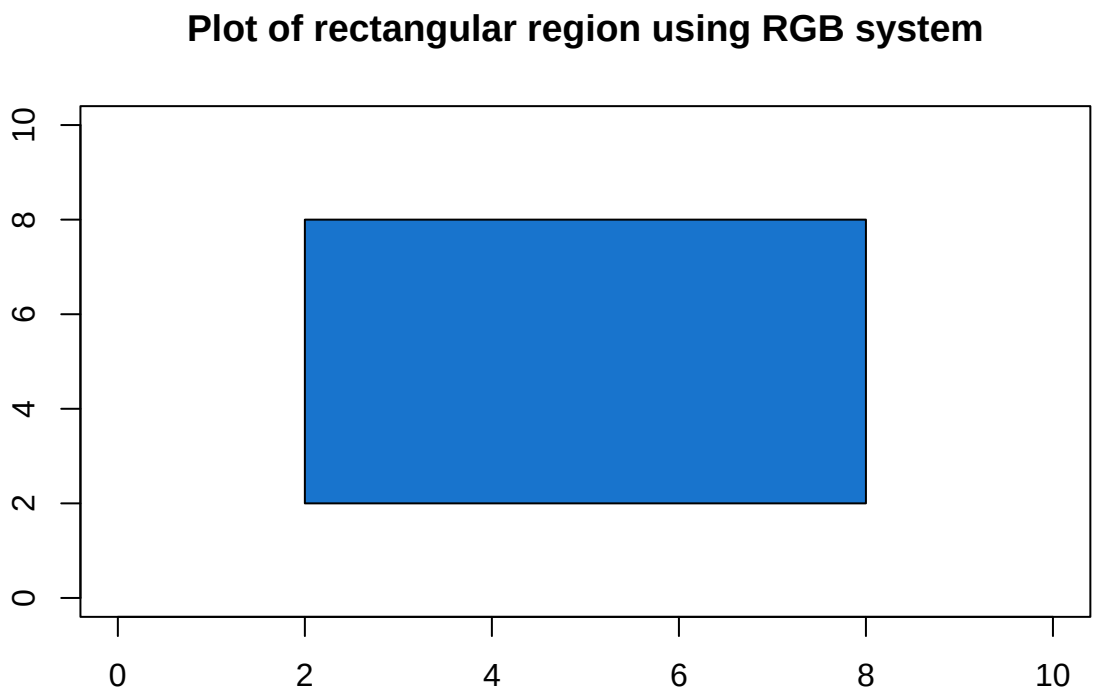
Let's try this out, using the color values I mentioned above:

```
# Example 2: Specifying a color using the RGB system

plot(
    x = NULL,
    xlim = c(0, 10),
    ylim = c(0, 10),
    main = "Plot of rectangular region using RGB system",
    xlab = "",
    ylab = ""
)

polygon(
    x = c(2, 2, 8, 8),
    y = c(2, 8, 8, 2),
```

```
    col =
        rgb(
            red = 24,
            green = 116,
            blue = 205,
            maxColorValue = 255
        )
)
```



So these are exactly the same color.

You might be wondering how I knew that the color "dodgerblue3" was equivalent to an RGB specification of 25, 116, and 205 for red, green, and blue, respectively.

I used the very handy "col2rgb" function, which takes a character string color name and returns the red, green, and blue values for the color:

```
# Example 3: Finding the red, green, and blue values

col2rgb( "dodgerblue3" )
```

```
##       [,1]
## red     24
## green  116
## blue   205
```

Here, the red, green, and blue values are returned in a data structure called a *matrix*, which we'll learn about later on in the course.

However, you don't need to be familiar with this advanced data structure to read off the values for the red, green, and blue values.

If you try to call *col2rgb()* using a character string that is not the name of a color, R will generate an error:

```
# Example 4: Making a mistake

col2rgb( "dodgerblue7" )
```

```
## Error in col2rgb("dodgerblue7"): invalid color name 'dodgerblue7'
```

Section 2: RGB and Hexadecimal Numbers

When we call the *rgb()* function, what sort of object is returned?

Let's find out:

```
#Example 5: Calling the rgb() function

rgb(
    red = 24,
    green = 116,
    blue = 205,
    maxColorValue = 255
)
```

```
## [1] "#1874CD"
```

Wow! What's *that*?!?!?

First of all, notice that the object that was returned by the function is enclosed in quotes, so that means that it's a character string.

```
# Example 6: "#1874CD" is a character string

class( "#1874CD" )
```

```
## [1] "character"
```

Notice that the character string starts with a hashtag character.

That indicates to R that the character string represents a number in what's called *hexadecimal* notation.

"Hexadecimal" is a fancy way of saying base 16.

Because 16 is greater than 10, we need extra symbols to represent the values from 10 to 15, and the usual convention is to use letters:

Value	Hexadecimal Representation
10	A

Value	Hexadecimal Representation
11	B
12	C
13	D
14	E
15	F

The advantage of using hexadecimal notation is that we can use exactly two characters to represent 256 values.

Recall that in the RGB system the red, green, and blue colors are specified using a range from 0 to 255, which gives a total of 256 different colors.

This means that we can specify the numeric value of red using exactly 2 characters, and likewise for green and blue.

Thus, to specify the numeric values of red, green, and blue, we need exactly 6 characters.

The character string "#1874CD" starts with a hashtag in order to indicate that this is a hexadecimal number, and then there are six characters to specify the actual color.

The first two characters "18" specify the value of red.

We can convert from hexadecimal to base 10 like this:

$$1 \times 16 + 8 \;=\; 24$$

If you look back, you'll see that the red value (in base 10) was 24.

The next two characters "74" specify the value of green.

Likewise, we can convert "74" in hexadecimal to base 10:

$$7 \times 16 + 4 \;=\; 116$$

If you look back, you'll see that the value of green for this color was indeed 116.

Finally, the last two characters "CD" indicate the value for blue.

Here, we'll have to use the table above to convert the characters 'C' and 'D' to numeric values:

$$12 \times 16 + 13 \;=\; 205$$

Again, if you check back, you'll find that this is exactly the value for blue that we found.

If you're a little murky about base 16, don't worry – you don't have to understand the details.

The important thing to understand is that this hexadecimal system enables us to completely specify any color using exactly 6 characters, so it's uniform, compact, and practical.

We never perform arithmetic operations on hexadecimal numbers, at least not in CSCI E-5a.

Instead, you can just pretend that these strange character strings are essentially just a weird naming convention, and R will handle all the necessary conversions.

This method of using hexadecimal numbers to specify colors in the RGB system is not some bizarre thing that is specific to R, but instead it's quite universal.

For instance, Microsoft Office uses exactly this system, and it coordinates perfectly with R.

Suppose you're developing a Powerpoint slide deck, and you've selected an Office theme, which sets the Office color palette.

You can find the RGB hexadecimal representation for any of the colors, copy that string, and then paste it into R, and you'll get the exact same color.

Likewise, if there's a color that you're using in R, say for instance "mediumorchid3", then you can obtain the exact same color in your Powerpoint deck by copying and pasting the hexadecimal code.

Section 3: A Cool Website

I used Microsoft Office as an example, just because many working professionals use this suite to develop documents and presentations, but there are many programs and apps available for choosing colors, and these will usually enable you to pick a color and then determine its RGB value.

CSCI E-5a is a course in R programming, not graphic design, so regrettably we can't delve into the fascinating world of color theory.

However, I do want to mention one very nice website:

http://color.adobe.com

This website has some very nice tools for selecting colors, and it automatically displays the hexadecimal code, which you can then just copy and paste into your R code.

Module Review

Now that you've finished this module, you should be able to:

- Specify colors using the RGB system, including using explicit hexadecimal codes.