```
  Course: Exploratory_Data_Analysis
  Lesson: Working_with_Colors

- Class: text
  Output: "Working_with_Colors. (Slides for this and other Data Science courses may be
  found at github https://github.com/DataScienceSpecialization/courses/. If you care to
  use them, they must be downloaded as a zip file and viewed locally. This lesson
  corresponds to 04_ExploratoryAnalysis/Colors.)"


- Class: text
  Output:  This lesson is about using colors in R. It really supplements the lessons on
  plotting with the base and lattice packages which contain functions that are able to
  take the argument col. We'll discuss ways to set this argument more colorfully.

- Class: text
  Output: Of course, color choice is secondary to your data and how you analyze it, but
  effectively using colors can enhance your plots and presentations, emphasizing the
  important points you're trying to convey.

- Class: text
  Output: The motivation for this lesson is that the default color schemes for most
  plots in R are not optimal. Fortunately there have been recent developments to
  improve the handling and specification of colors in plots and graphs. We'll cover
  some functions in R as well as in external packages that are very handy. If you know
  how to use some of these then you'll have more options when you create your displays.

- Class: figure
  Output:  We'll begin with a motivating example - a typical R plot using  3 default
  colors.
  Figure: showBad.R
  FigureType: new

- Class: mult_question
  Output: According to the plot, what is color 2?
  AnswerChoices:  Empty black circles; Red; Green; Blue
  CorrectAnswer:  Red
  AnswerTests: omnitest(correctVal='Red')
  Hint: What color point does the arrow labeled col=2 point to?

- Class: text
  Output: So these are the first 3 default values. If you were plotting and just
  specified col=c(1:3) as one of your arguments, these are colors you'd get. Maybe you
  like them, but they might not be the best choice for your application.

- Class: figure
  Output:  To show you some options, here's a display of two color palettes that come
  with the grDevices package available to you. The left shows you some colors from the
  function heat.colors. Here low values are represented in red and as the values
  increase the colors move through yellow towards white. This is consistent with the
  physical properties of fire. The right display is from the function topo.colors which
  uses topographical colors ranging from blue (low values) towards brown (higher values).
  Figure: showHeat.R
  FigureType: new

- Class: cmd_question
  Output: So we'll first discuss some functions that the grDevices package offers. The
  function colors()  lists the names of 657 predefined colors you can use in any
  plotting function.  These names are returned as strings. Run the R command sample
  with colors() as its first argument and 10 as its second to give you an idea of the
  choices you have.
  CorrectAnswer: sample(colors(),10)
  AnswerTests: omnitest(correctExpr='sample(colors(),10)')
  Hint: Type sample(colors(),10) at the command prompt.

- Class: text
  Output:  We see a lot of variety in the colors, some of which are names followed by
  numbers indicating that there are multiple forms of that particular color.

```

```
- Class: text
  Output: So you're free to use any of these 600+ colors listed by the colors function.
  However, two additional functions from grDevices, colorRamp and  colorRampPalette,
  give you more options. Both of these take color names as arguments and use them as
  "palettes", that is, these argument colors are blended in different proportions to
  form new colors.

- Class: text
  Output: The first, colorRamp, takes a palette of colors (the arguments) and returns a
  function that takes values between 0 and 1 as arguments. The 0 and 1 correspond to
  the extremes of the color palette. Arguments between 0 and 1 return blends of these
  extremes.

- Class: cmd_question
  Output:  Let's see what this means. Assign to the variable pal the output of a call
  to colorRamp with the single argument, c("red","blue").
  CorrectAnswer: pal <- colorRamp(c("red","blue"))
  AnswerTests: expr_creates_var("pal"); omnitest(correctExpr='pal <-
  colorRamp(c("red","blue"))')
  Hint: Type pal <- colorRamp(c("red","blue")) at the command prompt.

- Class: cmd_question
  Output:  We don't see any output, but R has created the function pal which we can
  call with a single argument between 0 and 1. Call pal now with the argument 0.
  CorrectAnswer: pal(0)
  AnswerTests: omnitest(correctExpr='pal(0)')
  Hint: Type pal(0) at the command prompt.

- Class: text
  Output: We see a 1 by 3 array with 255 as the first entry and 0 in the other 2. This
  3 long vector corresponds to red, green, blue (RGB) color encoding commonly used in
  televisions and monitors. In R, 24 bits are used to represent colors. Think of these
  24 bits as 3 sets of 8 bits, each of which represents an intensity for one of the
  colors red, green, and blue.

- Class: text
  Output: The 255 returned from the pal(0) call corresponds to the largest possible
  number represented with 8 bits, so the vector (255,0,0) contains only red (no green
  or blue), and moreover, it's the highest possible value of red.

- Class: mult_question
  Output: Given that you created pal with the palette containing "red" and "blue", what
  color do you think will be represented by the vector that pal(1) returns? Recall that
  pal will only take arguments between 0 and 1, so 1 is the largest argument you can
  pass it.
  AnswerChoices:   red; green; blue; yellow
  CorrectAnswer:  blue
  AnswerTests: omnitest(correctVal='blue')
  Hint: If the one extreme (0) returned red, it makes sense that pal(1) will return a
  vector representing blue.

- Class: cmd_question
  Output:  Check your answer now by calling pal with the argument 1.
  CorrectAnswer: pal(1)
  AnswerTests: omnitest(correctExpr='pal(1)')
  Hint: Type pal(1) at the command prompt.

- Class: mult_question
  Output:  You see the vector (0,0,255) which represents the highest intensity of blue.
  What vector do you think the call pal(.5) will return?
  AnswerChoices:   (0,255,0); (127.5,0,127.5); (255,255,255);(255,0,255)
  CorrectAnswer: (127.5,0,127.5)
  AnswerTests: omnitest(correctVal='(127.5,0,127.5)')
  Hint: The correct answer should be halfway between (255,0,0) and (0,0,255). Which is
  the only choice that is the average (mean) of these two?

- Class: cmd_question
  Output:  The function pal can take more than one argument. It returns one 3-long (or
  4-long, but more about this later) vector for each argument. To see this in action,
```

```
    call pal with the argument seq(0,1,len=6).
  CorrectAnswer: pal(seq(0,1,len=6))
  AnswerTests: omnitest(correctExpr='pal(seq(0,1,len=6))')
  Hint: Type pal(seq(0,1,len=6)) at the command prompt.

- Class: text
  Output: Six vectors (each of length 3) are returned. The i-th vector is identical to
    output that would be returned by the call pal(i/5) for i=0,...5. We see that the i-th
    row (for i=1,...6) differs from the (i-1)-st row in the following way. Its red entry
    is 51 = 255/5 points lower and its blue entry is 51 points higher.

- Class: text
  Output: So pal creates colors using the palette we specified when we called
    colorRamp. In this example none of pal's outputs will ever contain green since it
    wasn't in our initial palette.

- Class: text
  Output: We'll turn now to colorRampPalette, a function similar to colorRamp. It also
    takes a palette of colors and returns a function. This function, however,  takes
    integer arguments (instead of numbers between 0 and 1) and returns a vector of colors
    each of which is a blend of colors of the original palette.

- Class: text
  Output: The argument you pass to the returned function specifies the number of colors
    you want returned. Each element of the returned vector is a 24 bit number,
    represented as 6 hexadecimal characters, which range from 0 to F. This set of 6 hex
    characters represents the intensities of red, green, and blue, 2 characters for each
    color.

- Class: cmd_question
  Output:  To see this better, assign to the variable p1 the output of a call to
    colorRampPalette with the single argument, c("red","blue"). We'll compare it to our
    experiments using colorRamp.
  CorrectAnswer: p1 <- colorRampPalette(c("red","blue"))
  AnswerTests: expr_creates_var("p1"); omnitest(correctExpr='p1 <-
    colorRampPalette(c("red","blue"))')
  Hint: Type p1 <- colorRampPalette(c("red","blue")) at the command prompt.

- Class: cmd_question
  Output:  Now call p1 with the argument 2.
  CorrectAnswer: p1(2)
  AnswerTests: omnitest(correctExpr='p1(2)')
  Hint: Type p1(2) at the command prompt.

- Class: text
  Output: We see a 2-long vector is returned. The first entry FF0000 represents red.
    The FF is hexadecimal for 255, the same value returned by our call pal(0). The second
    entry 0000FF represents blue, also with intensity 255.

- Class: cmd_question
  Output:  Now call p1 with the argument 6. Let's see if we get the same result as we
    did when we called pal with the argument seq(0,1,len=6).
  CorrectAnswer: p1(6)
  AnswerTests: omnitest(correctExpr='p1(6)')
  Hint: Type p1(6) at the command prompt.

- Class: cmd_question
  Output: Now we get the 6-long vector (FF0000, CC0033, 990066, 650099, 3200CC,
    0000FF). We see the two ends (FF0000 and 0000FF) are consistent with the colors red
    and blue. How about CC0033? Type 0xcc or 0xCC at the command line to see the decimal
    equivalent of this hex number. You must include the 0 before the x to specify that
    you're entering a hexadecimal number.
  CorrectAnswer: 0xcc
  AnswerTests: ANY_of_exprs('0xcc','0xCC','0xcC','0xCc')
  Hint: Type 0xcC or 0xcc or 0xCC or 0xCc at the command prompt.

- Class: text
  Output: So 0xCC equals 204 and we can easily convert hex 33 to decimal, as in
    0x33=3*16+3=51. These were exactly the numbers we got in the second row returned from
```

```
    our call to pal(seq(0,1,len=6)). We see that 4 of the 6 numbers agree with our
    earlier call to pal. Two of the 6 differ slightly.

- Class: cmd_question
  Output: We can also form palettes using colors other than red, green and blue. Form a
  palette, p2, by calling colorRampPalette with the colors "red" and "yellow". Remember
  to concatenate them into a single argument.
  CorrectAnswer: p2 <- colorRampPalette(c("red","yellow"))
  AnswerTests: expr_creates_var("p2"); omnitest(correctExpr='p2 <-
  colorRampPalette(c("red","yellow"))')
  Hint: Type p2 <- colorRampPalette(c("red","yellow")) at the command prompt.

- Class: cmd_question
  Output:  Now call p2 with the argument 2. This will show us the two extremes of the
  blends of colors we'll get.
  CorrectAnswer: p2(2)
  AnswerTests: omnitest(correctExpr='p2(2)')
  Hint: Type p2(2) at the command prompt.

- Class: figure
  Output: Not surprisingly the first color we see is FF0000, which we know represents
  red. The second color returned, FFFF00, must represent yellow, a combination of full
  intensity red and full intensity green. This makes sense, since yellow falls between
  red and green on the color wheel as we see here. (We borrowed this image from
  lucaskrech.com.)
  Figure: showColor.R
  FigureType: new

- Class: cmd_question
  Output:  Let's now  call p2 with the argument 10. This will show us how the two
  extremes, red and yellow, are blended together.
  CorrectAnswer: p2(10)
  AnswerTests: omnitest(correctExpr='p2(10)')
  Hint: Type p2(10) at the command prompt.

- Class: text
  Output:  So we see the 10-long vector. For each element, the red component is fixed
  at FF, and the green component grows from 00 (at the first element) to FF (at the
  last).

- Class: cmd_question
  Output:  This is all fine and dandy but you're probably wondering when you can see
  how all these colors show up in a display. We copied some code from the R
  documentation pages (color.scale if you're interested) and created a function for
  you, showMe. This takes as an argument, a color vector, which as you know, is
  precisely what calls to p1 and p2 return to you. Call showMe now with p1(20).
  CorrectAnswer: showMe(p1(20))
  AnswerTests: omnitest(correctExpr='showMe(p1(20))')
  Hint: Type showMe(p1(20)) at the command prompt.


- Class: cmd_question
  Output:  We see the interpolated palette here. Low values in the lower left corner
  are red and as you move to the upper right, the colors move toward blue. Now call
  showMe with p2(20) as its argument.
  CorrectAnswer: showMe(p2(20))
  AnswerTests: omnitest(correctExpr='showMe(p2(20))')
  Hint: Type showMe(p2(20)) at the command prompt.

- Class: cmd_question
  Output:  Here we see a similar display, the colors moving from red to yellow, the
  base colors of our p2 palette. For fun, see what p2(2) looks like using showMe.
  CorrectAnswer: showMe(p2(2))
  AnswerTests: omnitest(correctExpr='showMe(p2(2))')
  Hint: Type showMe(p2(2))) at the command prompt.

- Class: text
  Output: A much more basic pattern, simple but elegant.
```

```
- Class: text
  Output:  We mentioned before that colorRamp (and colorRampPalette) could return a 3
  or 4 long vector of colors. We saw 3-long vectors returned indicating red, green, and
  blue intensities. What would the 4th entry be?

- Class: cmd_question
  Output: We'll answer this indirectly. First, look at the function p1 that
  colorRampPalette returned to you. Just type p1 at the command prompt.
  CorrectAnswer: p1
  AnswerTests: omnitest(correctExpr='p1')
  Hint: Type p1 at the command prompt.

- Class: text
  Output: We see that p1 is a short function with one argument, n. The argument n is
  used as the length in a call to the function seq.int, itself  an argument to the
  function ramp. We can infer that ramp is just going to divide the interval from 0 to
  1 into n pieces.

- Class: cmd_question
  Output: The heart of p1 is really the call to the function rgb with either 4 or 5
  arguments. Use the ?fun construct to look at the R documentation for rgb now.
  CorrectAnswer: ?rgb
  AnswerTests: omnitest(correctExpr='?rgb')
  Hint: Type ?rgb at the command prompt.

- Class: mult_question
  Output: We see that rgb is a color specification function that can be used to produce
  any color with red, green, blue proportions. We see the maxColorValue is 1 by
  default, so if we called rgb with values for red, green and blue, we would specify
  numbers at most 1 (assuming we didn't change the default for maxColorValue).
  According to the documentation, what is the maximum number of arguments rgb can have?
  AnswerChoices: 3;4;5;6
  CorrectAnswer: 6
  AnswerTests: omnitest(correctVal='6')
  Hint: Look at the sample calling line in the Usage section of the documentation.

- Class: cmd_question
  Output: So the fourth argument is alpha which can be a logical, i.e., either TRUE or
  FALSE, or a numerical value. Create the function p3 now by calling colorRampPalette
  with the colors blue and green (remember to concatenate them into a single argument)
  and setting the alpha argument to .5.
  CorrectAnswer: p3 <- colorRampPalette(c("blue","green"),alpha=.5)
  AnswerTests: expr_creates_var("p3"); omnitest(correctExpr='p3 <-
  colorRampPalette(c("blue","green"),alpha=.5)')
  Hint: Type p3 <- colorRampPalette(c("blue","green"),alpha=.5) at the command prompt.


- Class: cmd_question
  Output: Now call p3 with the argument 5.
  CorrectAnswer: p3(5)
  AnswerTests:  omnitest(correctExpr='p3(5)')
  Hint: Type p3(5) at the command prompt.

- Class: text
  Output: We see that in the 5-long vector that the call returned, each element has 32
  bits, 4 groups of 8 bits each. The last 8 bits represent the value of alpha. Since it
  was NOT ZERO in the call to colorRampPalette, it gets the maximum FF value. (The same
  result would happen if alpha had been set to TRUE.) When it was 0 or FALSE (as in
  previous calls to colorRampPalette) it was given the value 00 and wasn't shown. The
  leftmost 24 bits of each element are the same RGB encoding we previously saw.

- Class: text
  Output: So what is alpha? Alpha represents an opacity level, that is, how transparent
  should the colors be. We can add color transparency with the alpha parameter to calls
  to rgb. We haven't seen any examples of this yet, but we will now.

- Class: cmd_question
  Output: We generated 1000 random normal pairs for you in the variables x and y. We'll
  plot them in a scatterplot by calling plot with 4 arguments. The variables x and y
```

```
    are the first 2. The third is the print character argument pch. Set this equal to 19
    (filled circles). The final argument is col which should be set equal to a call to
    rgb. Give rgb 3 arguments, 0, .5, and .5.
    CorrectAnswer: plot(x,y,pch=19,col=rgb(0,.5,.5))
    AnswerTests:  omnitest(correctExpr='plot(x,y,pch=19,col=rgb(0,.5,.5))')
    Hint: Type plot(x,y,pch=19,col=rgb(0,.5,.5)) at the command prompt.

- Class: cmd_question
    Output: Well this picture is okay for a scatterplot, a nice mix of blue and green,
    but it really doesn't tell us too much information in the center portion, since the
    points are so thick there. We see there are a lot of points, but is one area more
    filled than another? We can't really discriminate between different point densities.
    This is where the alpha argument can help us. Recall your plot command (use the up
    arrow) and add a 4th argument, .3,  to the call to rgb. This will be our value for
    alpha.
    CorrectAnswer: plot(x,y,pch=19,col=rgb(0,.5,.5,.3))
    AnswerTests:  omnitest(correctExpr='plot(x,y,pch=19,col=rgb(0,.5,.5,.3))')
    Hint: Type plot(x,y,pch=19,col=rgb(0,.5,.5,.3)) at the command prompt.

- Class: text
    Output: Clearly this is better. It shows us where, specifically, the densest areas of
    the scatterplot really are.

- Class: text
    Output: Our last topic for this lesson is the RColorBrewer Package, available on
    CRAN, that contains interesting and useful color palettes, of which there are 3
    types, sequential, divergent, and qualitative. Which one you would choose to use
    depends on your data.

- Class: figure
    Output: Here's a picture of the palettes available from this package. The top section
    shows the sequential palettes in which the colors are ordered from light to dark. The
    divergent palettes are at the bottom. Here the neutral color (white) is in the
    center, and as you move from the middle to the two ends of each palette, the colors
    increase in intensity. The middle display shows the qualitative palettes which look
    like collections of random colors. These might be used to distinguish factors in your
    data.
    Figure: showBrewer.R
    FigureType: new

- Class: text
    Output: These colorBrewer palettes can be used in conjunction with the colorRamp()
    and colorRampPalette() functions. You would use colors from a colorBrewer palette as
    your base palette,i.e., as arguments to colorRamp or colorRampPalette  which would
    interpolate them to create new colors.

- Class: cmd_question
    Output: As an example of this, create a new object, cols by calling the function
    brewer.pal with 2 arguments, 3 and "BuGn". The string "BuGn" is the second last
    palette in the sequential display. The 3 tells the function how many different colors
    we want.
    CorrectAnswer: cols <- brewer.pal(3, "BuGn")
    AnswerTests:  expr_creates_var("cols"); omnitest(correctExpr='cols <- brewer.pal(3,
    "BuGn")')
    Hint: Type cols <- brewer.pal(3, "BuGn") at the command prompt.

- Class: cmd_question
    Output: Use showMe to look at cols now.
    CorrectAnswer: showMe(cols)
    AnswerTests:  omnitest(correctExpr='showMe(cols)')
    Hint: Type showMe(cols) at the command prompt.

- Class: cmd_question
    Output: We see 3 colors, mixes of blue and green. Now create the variable pal by
    calling colorRampPalette with cols as its argument.
    CorrectAnswer: pal <- colorRampPalette(cols)
    AnswerTests:  omnitest(correctExpr='pal <- colorRampPalette(cols)')
    Hint: Type pal <- colorRampPalette(cols) at the command prompt.
```

```
- Class: cmd_question
  Output: The call showMe(pal(3)) would be identical to the showMe(cols) call. So use
  showMe to look at pal(20).
  CorrectAnswer: showMe(pal(20))
  AnswerTests:  omnitest(correctExpr='showMe(pal(20))')
  Hint: Type showMe(pal(20)) at the command prompt.

- Class: cmd_question
  Output: Now we can use the colors in pal(20) to display topographic information on
  Auckland's Maunga Whau Volcano. R provides this information in a matrix called
  volcano which is included in the package datasets.  Call the R function image with
  volcano as its first argument and col set equal to pal(20) as its second.
  CorrectAnswer: image(volcano, col = pal(20))
  AnswerTests:  omnitest(correctExpr='image(volcano, col = pal(20))')
  Hint: Type image(volcano, col = pal(20)) at the command prompt.

- Class: cmd_question
  Output: We see that the colors here of the sequential palette clue us in on the
  topography. The darker colors are more concentrated than the lighter ones. Just for
  fun, recall your last command calling image and instead of pal(20), use p1(20) as the
  second argument.
  CorrectAnswer: image(volcano, col = p1(20))
  AnswerTests:  omnitest(correctExpr='image(volcano, col = p1(20))')
  Hint: Type image(volcano, col = p1(20)) at the command prompt.

- Class: text
  Output: Not as nice a picture since the palette isn't as well suited to this data,
  but that's okay. It's review time!!!!


- Class: mult_question
  Output:  True or False? Careful use of colors in plots/maps/etc. can make it easier
  for the reader to understand what points you're trying to convey.
  AnswerChoices:  True; False
  CorrectAnswer:  True
  AnswerTests: omnitest(correctVal='True')
  Hint: If this were false, would we have created this great lesson on the subject?

- Class: mult_question
  Output:  Which of the following is an R package that provides color palettes for
  sequential, categorical, and diverging data?
  AnswerChoices:  RColorBrewer; RColorVintner; RColorBluer; RColorStewer
  CorrectAnswer:  RColorBrewer
  AnswerTests: omnitest(correctVal='RColorBrewer')
  Hint: Three of the choices rhyme. Eliminate the one with color in it and the one
  that's not really a real word.

- Class: mult_question
  Output:  True or False? The colorRamp and colorRampPalette functions can be used in
  conjunction with color palettes to connect data to colors.
  AnswerChoices:  True; False
  CorrectAnswer:  True
  AnswerTests: omnitest(correctVal='True')
  Hint: Recall our example using RColorBrewer and colorRampPalette.

- Class: mult_question
  Output:  True or False?    Transparency can NEVER be used to clarify plots with many
  points
  AnswerChoices:  True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
  Hint: Recall our example with the scatterplot and the dense section.

- Class: mult_question
  Output:  True or False?    The call p7 <- colorRamp("red","blue") would work (i.e.,
  not generate an error).
  AnswerChoices:  True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
```

```
  Hint: Recall our reminders to concatenate the colors to form a single argument.

- Class: mult_question
  Output:  True or False?    The function colors returns only 10 colors.
  AnswerChoices:  True; False
  CorrectAnswer: False
  AnswerTests: omnitest(correctVal='False')
  Hint: We only looked at 10 but there were many, many more.

- Class: mult_question
  Output:  Transparency is determined by which parameter of the rgb function?
  AnswerChoices:  alpha; beta; gamma; delta; it's all Greek to me
  CorrectAnswer: alpha
  AnswerTests: omnitest(correctVal='alpha')
  Hint: We only mentioned one Greek letter in this lesson.

- Class: text
  Output: Congratulations! We hope this lesson didn't make you see red. We're green
  with envy that you blue through it.

- Class: mult_question
  Output: "Would you like to receive credit for completing this course on
    Coursera.org?"
  CorrectAnswer: NULL
  AnswerChoices: Yes;No
  AnswerTests: coursera_on_demand()
  Hint: ""
```