**An Example of String Manupilation and Function Usage**

In this project, we explore how to generate RGB color codes, manipulate them, and convert them into hexadecimal format. This process involves creating various helper functions to handle different steps of the conversion. Let's walk through the code and understand each part in detail.

**Generating RGB Codes**

First, we need a function to generate RGB codes. Each RGB code consists of three values: red, green, and blue, each ranging from 0 to 255. We use the sample function to generate these values randomly.

**Function: create\_rgb**

The create\_rgb function takes an integer n as input and generates n random RGB color codes.

create\_rgb = function(n) {

set.seed(100)

rgb = vector("character", length = n)

for (i in 1:n) {

r = sample(0:255, 1)

g = sample(0:255, 1)

b = sample(0:255, 1)

# Format RGB code and store it in the vector

rgb[i] = paste("(", r, ",", g, ",", b, ")", sep = "")

}

return(rgb)

}

Q1 = create\_rgb(6)

Q1

* **Explanation**:
  + set.seed(100): Ensures reproducibility of random numbers.
  + rgb = vector("character", length = n): Initializes an empty character vector to store RGB codes.
  + for (i in 1:n): Loops n times to generate n RGB codes.
  + r = sample(0:255, 1), g = sample(0:255, 1), b = sample(0:255, 1): Randomly samples a value between 0 and 255 for red, green, and blue respectively.
  + rgb[i] = paste("(", r, ",", g, ",", b, ")", sep = ""): Formats and stores the RGB code in the vector.
  + return(rgb): Returns the vector of RGB codes.

**Finding Quotients and Remainders**

To convert RGB values to hexadecimal, we need to find the quotient and remainder when dividing each RGB value by 16. We create two helper functions, find\_quotient and find\_remainder, to perform these operations.

**Function: find\_quotient**

The find\_quotient function calculates the quotient of the division of two numbers.

find\_quotient = function(dividend, divisor) {

floor(dividend / divisor)

}

find\_quotient(22, 4)

* **Explanation**:
  + floor(dividend / divisor): Computes the integer part of the division (quotient).

**Function: find\_remainder**

The find\_remainder function calculates the remainder of the division of two numbers.

find\_remainder = function(dividend, divisor) {

dividend - floor(dividend / divisor) \* divisor

# dividend %% divisor can be used as well, but this way is faster

}

find\_remainder(17, 10)

* **Explanation**:
  + dividend - floor(dividend / divisor) \* divisor: Computes the remainder by subtracting the product of the quotient and the divisor from the dividend.

**Extracting RGB Values**

Next, we need to extract the RGB values from the formatted (r, g, b) string. The extract\_rgb function parses these strings and converts them into a dataframe with separate columns for red, green, and blue.

**Function: extract\_rgb**

The extract\_rgb function takes a vector of RGB color codes and extracts the numeric values into a dataframe.

extract\_rgb = function(color\_code) {

b = substr(color\_code, 2, nchar(color\_code) - 1)

bb = unlist(strsplit(b, split = ","))

df = data.frame(matrix(as.numeric(bb), nrow = length(b), byrow = T))

colnames(df) = c("R", "G", "B")

return(df)

}

Q4 = extract\_rgb(Q1)

Q4

* **Explanation**:
  + b = substr(color\_code, 2, nchar(color\_code) - 1): Removes the parentheses from the color code string.
  + bb = unlist(strsplit(b, split = ",")): Splits the string into individual RGB values.
  + df = data.frame(matrix(as.numeric(bb), nrow = length(b), byrow = T)): Converts the values into a numeric matrix and then into a dataframe.
  + colnames(df) = c("R", "G", "B"): Sets the column names of the dataframe to "R", "G", and "B".
  + return(df): Returns the dataframe.

**Converting RGB to Hexadecimal**

To convert RGB values to hexadecimal, we first find the quotient and remainder for each RGB component using the previously defined functions. We apply these functions to each column of the dataframe.

**Applying Quotient and Remainder Functions**

Q5a\_quotient = apply(Q4, 2, find\_quotient, 16)

Q5b\_remainder = apply(Q4, 2, find\_remainder, 16)

Q4

Q5a\_quotient

Q5b\_remainder

* **Explanation**:
  + apply(Q4, 2, find\_quotient, 16): Applies the find\_quotient function to each column of the dataframe Q4 with the divisor 16.
  + apply(Q4, 2, find\_remainder, 16): Applies the find\_remainder function to each column of the dataframe Q4 with the divisor 16.

**Replacing Values with Hexadecimal Characters**

Values between 10 and 15 need to be replaced with the corresponding hexadecimal characters (A to F). The replace\_10\_16 function handles this replacement.

**Function: replace\_10\_16**

The replace\_10\_16 function replaces numeric values 10 to 15 with their corresponding hexadecimal characters.

replace\_10\_16 = function(a) {

nums = 10:15

for (i in 1:6) {

nums = 10:15

a = ifelse(a == nums[i], LETTERS[i], as.character(a))

}

return(a)

}

replace\_10\_16(c(12, 15, 6))

replace\_10\_16(c(3, 4, 5))

replace\_10\_16(c(13, 15, 10))

* **Explanation**:
  + nums = 10:15: Defines the range of numbers to be replaced.
  + for (i in 1:6): Loops through each value in the range.
  + a = ifelse(a == nums[i], LETTERS[i], as.character(a)): Replaces the numeric value with the corresponding letter (A-F) if it matches.

We then apply this function to the quotients and remainders.

Q7a\_quotient = replace\_10\_16(Q5a\_quotient)

Q7a\_quotient

Q7b\_remainder = replace\_10\_16(Q5b\_remainder)

Q7b\_remainder

**Combining Quotients and Remainders**

Finally, we combine the quotients and remainders to form the hexadecimal codes. The hex\_codes function concatenates these values and adds a # at the beginning.

**Function: hex\_codes**

The hex\_codes function takes the quotients and remainders and constructs the hexadecimal color codes.

hex\_codes = function(Q7a\_quotient, Q7b\_remainder) {

m = matrix(paste0(Q7a\_quotient, Q7b\_remainder), nrow(Q7a\_quotient), ncol(Q7a\_quotient))

hexs = c()

for (i in 1:nrow(m)) {

hexs[i] = paste0("#", paste(m[i, ], collapse = ""))

}

return(hexs)

}

Q8 = hex\_codes(Q7a\_quotient, Q7b\_remainder)

Q1

Q8

* **Explanation**:
  + m = matrix(paste0(Q7a\_quotient, Q7b\_remainder), nrow(Q7a\_quotient), ncol(Q7a\_quotient)): Creates a matrix by concatenating the quotients and remainders.
  + hexs = c(): Initializes an empty vector to store the hex codes.
  + for (i in 1:nrow(m)): Loops through each row of the matrix.
  + hexs[i] = paste0("#", paste(m[i, ], collapse = "")): Concatenates the values in each row to form the hex code and adds a # at the beginning.
  + return(hexs): Returns the vector of hex codes.

This function creates the final hexadecimal color codes. The entire process converts randomly generated RGB values into their corresponding hexadecimal format, providing a complete solution for RGB to hex conversion.

**Conclusion**

This project demonstrates the complete process of converting RGB color codes to hexadecimal format through a series of well-defined steps and functions in R. By breaking down the task into manageable parts, we were able to address each aspect of the conversion with clarity and precision.

**Summary of Key Steps**

1. **Generating RGB Codes**:
   * We began by creating a function create\_rgb to generate random RGB codes. This function ensures that the values for red, green, and blue components are within the 0-255 range, formatted as (r, g, b).
2. **Finding Quotients and Remainders**:
   * Two helper functions, find\_quotient and find\_remainder, were developed to compute the quotient and remainder when dividing an RGB value by 16. These are essential steps for the conversion process.
3. **Extracting RGB Values**:
   * The extract\_rgb function was used to parse the RGB codes and convert them into a dataframe with separate columns for each color component. This step facilitated easy manipulation and application of further calculations.
4. **Converting RGB to Hexadecimal**:
   * Applying the quotient and remainder functions to the RGB dataframe allowed us to prepare the values for conversion to hexadecimal. This step was crucial for breaking down each color component into manageable parts.
5. **Replacing Numeric Values with Hexadecimal Characters**:
   * The replace\_10\_16 function converted numeric values between 10 and 15 to their corresponding hexadecimal characters (A-F). This step ensured that the values were in the correct format for hexadecimal representation.
6. **Combining Quotients and Remainders**:
   * Finally, the hex\_codes function combined the processed quotient and remainder values, adding a # prefix to create the final hexadecimal color codes.

**Importance and Applications**

Understanding and implementing the conversion between RGB and hexadecimal color codes is fundamental in various fields, particularly in web development and digital design. Hexadecimal codes are widely used in HTML and CSS to specify colors, making this conversion process highly relevant for creating and styling web pages.

**Learning Outcomes**

* **Problem-Solving**: Breaking down the conversion process into smaller tasks helped in methodically solving the problem and writing efficient code.
* **String Manipulation**: Handling and manipulating strings was key to extracting and formatting the RGB and hexadecimal values correctly.
* **Data Handling**: Converting data between different formats and types (strings, numerics, dataframes) reinforced the importance of data handling skills in programming.

**Future Directions**

This project lays a solid foundation for further exploration in color theory and digital image processing. Future work could involve:

* **Batch Processing**: Extending the functions to handle larger datasets of colors.
* **Inverse Conversion**: Developing functions to convert hexadecimal codes back to RGB.
* **Color Manipulation**: Implementing functions to adjust brightness, contrast, or apply filters to colors.

By completing this project, we have not only addressed the specific task of converting RGB to hexadecimal but also gained valuable insights and skills applicable to broader programming and data manipulation challenges.

**All Code**

#Q1

create\_rgb = function(n) {

set.seed(100)

rgb = vector("character", length = n)

for (i in 1:n) {

r = sample(0:255, 1)

g = sample(0:255, 1)

b = sample(0:255, 1)

# Format RGB code and store it in the vector

rgb[i] = paste("(", r, ",", g, ",", b, ")", sep = "")

}

return(rgb)

}

Q1 = create\_rgb(6)

Q1

#Q2

find\_quotient = function(dividend,divisor){

floor(dividend/divisor)

}

find\_quotient(22,4)

#Q3

find\_remainder = function(dividend,divisor){

dividend - floor(dividend/divisor)\*divisor

# dividend %% divisor can be used as well, but this way is faster

}

find\_remainder(17,10)

#Q4

extract\_rgb = function(color\_code){

b = substr(color\_code,2,nchar(color\_code)-1)

bb= unlist(strsplit(b,split = ","))

df = data.frame(matrix(as.numeric(bb),nrow = length(b),byrow = T))

colnames(df) = c("R","G","B")

return(df)

}

Q4 = extract\_rgb(Q1)

Q4

#Q5

Q5a\_quotient = apply(Q4,2,find\_quotient,16)

Q5b\_remainder = apply(Q4,2,find\_remainder,16)

Q4

Q5a\_quotient

Q5b\_remainder

#Q6

replace\_10\_16=function(a){

nums= 10:15

for (i in 1:6) {

nums=10:15

a = ifelse(a==nums[i],LETTERS[i],as.character(a))

}

return(a)

}

replace\_10\_16(c(12,15,6))

replace\_10\_16(c(3,4,5))

replace\_10\_16(c(13,15,10))

#Q7

Q5a\_quotient

Q7a\_quotient = replace\_10\_16(Q5a\_quotient)

Q7a\_quotient

Q5b\_remainder

Q7b\_remainder = replace\_10\_16(Q5b\_remainder)

Q7b\_remainder

#Q8

hex\_codes = function(Q7a\_quotient,Q7b\_remainder){

m = matrix(paste0(Q7a\_quotient,Q7b\_remainder),nrow(Q7a\_quotient),ncol(Q7a\_quotient))

hexs = c()

for (i in 1:nrow(m)) {

hexs[i] = paste0("#",paste(m[i,],collapse = ""))

}

return(hexs)

}

Q8 = hex\_codes(Q7a\_quotient,Q7b\_remainder)

Q1

Q8