Regular Expressions based programs

Yi Chua

4/17/2020

A small project creating software programs found in everyday life using Regular Expression to match character combinations in strings!

```
library(stringr)
```

Counting Vowels

```
#' @title Counting Vowels
#' @description computes the number of vowels in a character string
#' @param string (character)
#' @return a vector count of the number of vowels
count_vowels <- function(string){
  extract <- str_extract_all(str_to_lower(string), regex('[aeiou]'))
  list_vowels <- c('a','e','i','o','u')
  count <- rep(0,length(list_vowels))
  for (v in (1:length(list_vowels))){
    cum_sum <- cumsum(extract[[1]] == list_vowels[v])
    count[v] <- cum_sum[length(cum_sum)]
  }
  names(count) <- list_vowels
  return(count)
}

#test cases
count_vowels("FIAT LUX")
```

```
## a e i o u
## 1 0 1 0 1
```

```
count_vowels("MaY tHe FoRce Be wIth yOu")
```

```
## a e i o u
## 1 3 1 2 1
```

```
count_vowels("a fox jumps over the lazy dog")
```

```
## a e i o u
## 2 2 0 3 1
```

Counting Consonants

```r
#' @title Counting Consonants
#' @description computes the number of consonants in a character string
#' @param string (character)
#' @return table count of the number of consonants (vector)
count_consonants <- function(string){
  extract <- str_extract_all(str_to_lower(string), regex('[^aeiou]'))
  list_consonants <- c('b','c','d','f','g','h','j','k','l','m','n','p',
                       'q','r','s','t','v','w','x','y','z')
  count <- rep(0, length(list_consonants))
  for (c in (1:length(list_consonants))){
    cum_sum <- cumsum(extract[[1]] == list_consonants[c])
    count[c] <- cum_sum[length(cum_sum)]
  }
  names(count) <- list_consonants
  return(count)
}

#test cases
count_consonants("FIAT LUX")
```

```
## b c d f g h j k l m n p q r s t v w x y z
## 0 0 0 1 0 0 0 0 1 0 0 0 0 0 0 1 0 0 1 0 0
```

```r
count_consonants("MaY tHe FoRce Be wIth yOu")
```

```
## b c d f g h j k l m n p q r s t v w x y z
## 1 1 0 1 0 2 0 0 0 1 0 0 0 1 0 2 0 1 0 2 0
```

```r
count_consonants("a fox jumps over the lazy dog")
```

```
## b c d f g h j k l m n p q r s t v w x y z
## 0 0 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 0 1 1 1
```

Hex verifier

```r
#' @title Hexadecimal checker
#' @description checks whethter the string is a valid color in hexadecimal notation
#' @param string (character)
#' @return boolean value (Logic)
is_hex <- function(string){
  return(str_detect(str_to_upper(string), regex('^#[A-F0-9]{6}$')))
}

#test case
is_hex("#ff0000")
```

```
## [1] TRUE
```

```r
is_hex("#123456")
```

```
## [1] TRUE
```

```r
is_hex("#12Fb56")
```

```
## [1] TRUE
```

```r
is_hex("#1234GF")
```

```
## [1] FALSE
```

```r
is_hex("#1234567")
```

```
## [1] FALSE
```

```r
is_hex("blue")
```

```
## [1] FALSE
```

```r
#' @title Hexadecimal w/ Alpha Transparency checker
#' @description checks whethter the string is a valid hex color with alpha transparency
#' @param string (character)
#' @return boolean value (Logic)
is_hex_alpha <- function(string){
  return(str_detect(str_to_upper(string), regex('^#[A-F0-9]{8}$')))
}

#test cases
is_hex_alpha("#FF000078")
```

```
## [1] TRUE
```

```r
is_hex_alpha("#FF0000")
```

```
## [1] FALSE
```

Hexadecimal-RGB Converter

```r
#' @title Hexadecimal to RGB Values
#' @description takes a hex-color and returns a named vector with the values of the RGB as well as the
#' @param string (character)
#' @return RGB values (vector)
hex_values <- function(string){
  list_rgba <- c("red", "green", "blue", "alpha") #will be used to label the vector with the values of
  rgb <- rep("0",3)
  if (!is_hex(string) & !is_hex_alpha(string)){
  stop("\ninput is not a valid hexadecimal color")
  } else{
    # set pattern depending if it is hex with alpha or without alpha
    if(is_hex_alpha(string)){
    pattern <- '^#([A-F0-9]{2})([A-Fa-f0-9]{2})([A-F0-9]{2})([A-F0-9]{2})$' #pattern for hex w/ alpha
    } else{
      pattern <- '^#([A-F0-9]{2})([A-Fa-f0-9]{2})([A-F0-9]{2})$' #pattern for just hex
    }
    match_group <- str_match(string,regex(pattern)) #pattern matched
    for (m in 2:(length(match_group))){
      rgb[m-1] <-  match_group[m]
      names(rgb)[m-1] <- list_rgba[m-1]
    }
  }
  return(rgb)
}

#test cases
hex_values("#12345655")
```

```
##   red green  blue alpha
##  "12"  "34"  "56"  "55"
```

```r
hex_values("#12Fb56")
```

```
##   red green  blue
##  "12"  "Fb"  "56"
```

```r
hex_values("#1234GF")
```

```
## Error in hex_values("#1234GF"):
## input is not a valid hexadecimal color
```

```r
hex_values("#1234567")
```

```
## Error in hex_values("#1234567"):
## input is not a valid hexadecimal color
```

```r
hex_values("blue")
```

```
## Error in hex_values("blue"):
## input is not a valid hexadecimal color
```

Password verifier

Checks if the string is a “valid” password

We consider a string to be a valid password if it:

- Has a minimum of 8 characters
- Contains at least 1 lowercase letter
- Contains at least 1 uppercase letter
- Contains at least 1 number
- Contains at least 1 of the permitted special characters -!@#$%^&*
- Should not contain pattern *hello* or *123*

```
#' @title Password Checker
#' @description checks if a string is a 'valid' password
#' @param string (character)
#' @return boolean value (logic)
check_pswd <- function(passwords){
return(!(str_detect(passwords, regex("hello"))) & !str_detect(passwords,regex("123")) &str_detect(passw
}


passwords <- c("Hello12!", "Hell@125!", "Hel7o123", "Hel7o123!", "L@R!6373", "Fanny*1", "L@ttie#@*!", "

check_pswd(passwords)
```

```
## [1]  TRUE  TRUE FALSE FALSE FALSE FALSE FALSE FALSE
```

Currency converter

```
#' @title Currency converter
#' @description converts from one currency to another
#' @param amount numeric value (double)
#' @param from type of currency: euro, pound, yen, yuan, rupee, peso, bitcoin (character)
#' @param to type of currency: euro, pound, yen, yuan, rupee, peso, bitcoin (character)
#' @return the amount based on the type of currency one desired (double)
exchange <- function(amount = 10, from = 'us', to = 'euro'){
  conversion_table <- c("us" = 1,"euro" = 0.87, "pound" = 0.68, "yen" = 106.43, "yuan" = 6.47, "rupee" =
  conversion_table
  if ((!(str_extract(from, regex("\\w+"))) %in% names(conversion_table)) | !(str_extract(to, regex("\\w
    stop("currency type must be the following:\nus, euro, pound, yen, yuan, rupee, peso, or bitcoin")
  if (amount < 0){
    stop("\n amount must be nonnegative")
  } else {
    #extract the numerical values that will be used as the currency exchange rate
    f = conversion_table[which(names(conversion_table) == (str_extract(from, regex("\\w+"))))]
    t = conversion_table[which(names(conversion_table) == (str_extract(to, regex("\\w+"))))]
    converted <- (t / f) * amount
    names(converted) <- NULL
    return(converted)
  }
}

#test cases
exchange(amount = 1, from = 'us', to = 'euro')
```

```
## [1] 0.87
```

```
exchange(amount = 10, from = 'us', to = 'euro')
```

```
## [1] 8.7
```

```
exchange(amount = 10, from = 'us', to = 'bitcoin')
```

```
## [1] 0.022
```

```
exchange(amount = 10, from = 'us', to = 'pound')
```

```
## [1] 6.8
```

```
#negative amount
exchange(amount = -50, from = 'us', to = 'euro')
```

```
## Error in exchange(amount = -50, from = "us", to = "euro"):
##  amount must be nonnegative
```

```r
#incorrect currency type/nonexistent
exchange(amount = 10, from = 'us', to = 'ringgit')
```

```
## Error in exchange(amount = 10, from = "us", to = "ringgit"): currency type must be the following:
## us, euro, pound, yen, yuan, rupee, peso, or bitcoin
```

Dataframe Cleaner

```r
#load messy data csv file
dat <- read.csv('USA_crime.csv', header = FALSE, stringsAsFactors = FALSE)
dat
```

```
##                           V1
## 1 Alabama >>+13.2&&236$^58
## 2   Alaska??10.0  263>  48
## 3  Arizona+ 8.1:;  294  80
## 4  Arkansas-  8.8* 190  50
## 5 California? 9.0  276= 91
## 6   Colorado  7.9= 204> 78
```

```r
#' @title Data cleaner
#' @description cleans messy data and return a clean data frame containing information
#' @param df takes in a data frame (data frame)
#' @return a cleaned up data frame (data frame)
clean_data <- function(df){
  pattern <- "^[\\s]?([A-z]+)[+?>]*[- ]*[\\s]?[>+&$^?:;*=]*(\\d{1,2}\\.\\d{1})[>+&$^?:;*=]*[\\s]*(\\d{3
  matched <- str_match(df[[1]],regex(pattern))
  #create empty matrix
  clean_df <- matrix(data= NA, nrow =nrow(matched), ncol = ncol(matched) - 1)
  for (i in 1:nrow(matched)){
    row_df <- matched[i,]
    for (j in 2:ncol(matched)){
      clean_df[i,j-1] <- row_df[j]
    }
  }
  #convert matrix to data frame
  clean_df <- as.data.frame(clean_df,header = FALSE, stringsAsFactors = FALSE)
  names(clean_df) <- c("State", "Murder", "Assault", "UrbanPop")
  return(clean_df)
}

clean_data(dat)
```

```
##        State Murder Assault UrbanPop
## 1    Alabama   13.2     236       58
## 2     Alaska   10.0     263       48
## 3    Arizona    8.1     294       80
## 4   Arkansas    8.8     190       50
## 5 California    9.0     276       91
## 6   Colorado    7.9     204       78
```