strings

November 9, 2015

Today we'll practice some basic string manipulations.

Converting from Fahrenheit Degrees

Here are four functions that convert from Fahrenheit degrees to other temperature scales:

```
to_celsius <- function(x) {
  (x - 32) * (5/9)
}

to_kelvin <- function(x) {
  (x + 459.67) * (5/9)
}

to_reaumur <- function(x) {
  (x - 32) * (4/9)
}

to_rankine <- function(x) {
  x + 459.67
}
```

We can use the previous functions to create a more general function `convert()`:

```
convert <- function(x, to = "celsius") {
  switch(to,
         "celsius" = to_celsius(x),
         "kelvin" = to_kelvin(x),
         "reaumur" = to_reaumur(x),
         "rankine" = to_rankine(x))
}

convert(30, 'celsius')

## [1] -1.111111
```

`convert()` works fine when the argument `to = 'celsius'`. What happens if you try `convert(30, 'Celsius')` or `convert(30, 'CELSIUS')`?

Your turn. Rewrite `convert()` such that the argument `to` can be given in upper or lower case letters. For instance, the following three calls should be equivalent:

```r
convert(30, 'celsius')
convert(30, 'Celsius')
convert(30, 'CELSIUS')

# your function convert
convert <- function(x, to = "celsius") {
  to = tolower(to)
  switch(to,
         "celsius" = to_celsius(x),
         "kelvin" = to_kelvin(x),
         "reaumur" = to_reaumur(x),
         "rankine" = to_rankine(x))
}
convert(30, 'celsius')

## [1] -1.111111

convert(30, 'Celsius')

## [1] -1.111111

convert(30, 'CELSIUS')

## [1] -1.111111
```

Names of files

Imagine that you need to generate the names of 10 data files (with .csv extension). All the files have the same prefix name but each of them has a different number: `file1.csv`, `file2.csv`, … , `file10.csv`.

How can you generate a character vector with these names in R? Come up with at least three different ways to get such a vector:

```r
# vector of file names
library('stringr')
paste0("file", 1:10, ".csv")

##  [1] "file1.csv"  "file2.csv"  "file3.csv"  "file4.csv"  "file5.csv"
##  [6] "file6.csv"  "file7.csv"  "file8.csv"  "file9.csv"  "file10.csv"

paste("file", sep = "", 1:10, ".csv")

##  [1] "file1.csv"  "file2.csv"  "file3.csv"  "file4.csv"  "file5.csv"
##  [6] "file6.csv"  "file7.csv"  "file8.csv"  "file9.csv"  "file10.csv"

str_c("file", 1:10, ".csv")

## Warning in stri_c(..., sep = sep, collapse = collapse, ignore_null =
TRUE):
## longer object length is not a multiple of shorter object length
##  [1] "file1.csv"  "file2.csv"  "file3.csv"  "file4.csv"  "file5.csv"
##  [6] "file6.csv"  "file7.csv"  "file8.csv"  "file9.csv"  "file10.csv"
```

Now imagine that you need to rename the characters `file` into `dataset`. In other words, you want the vector of file names to look like this: `dataset1.csv`, `dataset2.csv`, … , `dataset10.csv`. Take the previous vector of file names and rename its elements:

```
# rename vector of file names
file_names <- paste0("file", 1:10, ".csv")
str_sub(file_names, 1, 4) <- 'dataset'
file_names
```

```
##  [1] "dataset1.csv"  "dataset2.csv"  "dataset3.csv"  "dataset4.csv"
##  [5] "dataset5.csv"  "dataset6.csv"  "dataset7.csv"  "dataset8.csv"
##  [9] "dataset9.csv"  "dataset10.csv"
```

Using function `cat()`

Run the following code:

```
# name of output file
outfile <- "output.txt"

# writing to 'outfile.txt'
cat("This is the first line", file = outfile)
# insert new line
cat("\n", file = outfile, append = TRUE)
cat("A 2nd line", file = "output.txt", append = TRUE)
# insert 2 new lines
cat("\n\n", file = outfile, append = TRUE)
cat("\nThe quick brown fox jumps over the lazy dog\n",
    file = outfile, append = TRUE)
```

After running the previous code, look for the file `output.txt` in your working directory and open it. One of the uses of `cat()` is to write contents to a text file. Note that the first call to `cat()` does not include the argument `append`. The rest of the calls do include `append = TRUE`.

Your turn. Modify the script such that the content of `output.txt` looks like the header of an `.Rmd` file with your information:

```
---
title: "Some title"
author: "Your name and SID"
date: "today's date"
output: html_document
---

This is the first line
```

```
A 2nd line


The quick brown fox jumps over the lazy dog
```

```r
output_file = "output.txt"
cat("---\n", file = output_file)
cat("title: hello world\n", file = output_file, append = TRUE)
cat("author: Junyu Wang 24153759\n", file = output_file, append = TRUE)
cat("output: html_document\n", file = output_file, append = TRUE)
cat("---", file = output_file, append = TRUE)
cat("\n\n", file = output_file, append = TRUE)
cat("This is the first line\n", file = output_file, append = TRUE)
cat("A 2nd line\n", file = output_file, append = TRUE)
cat("\n\nThe quit brown fox jumps over the lazy dog", file = output_file,
append = TRUE)
```

Valid Color Names

Write a function `is_color()` to test if a given name---in English---is a valid R color. If the provided name is a valid R color, `is_color()` returns `TRUE`. If the provided name is not a valid R color `is_color()` returns `FALSE`.

```r
# your is_color() function
is_color <- function(col) {
  col %in% colors()
}


# test it:
is_color('yellow')  # TRUE

is_color('blu')     # FALSE

is_color('turkuiose') # FALSE
```

Plot with a valid color

Use `is_color()` to create the function `colplot()` that takes one argument `col` (the name of a color) to produce a simple scatter plot. If `col` is a valid name---say "blue"---, the scatterplot should show a title `"testing color blue"`. If the provided `col` is not a valid color name, e.g. "blu", then the function must stop, showing an **error message** `"invalid color blu"`

```r
# your coloplot() function
colplot <- function(col) {
  if (is_color(col)) {
    plot.new()
    title(main = paste0("testing color ", col))
  }else {
    cat(paste0("invalid color ", col))
  }
}

# this should plot
colplot('tomato')

# this stops with error message
colplot('tomate')
```

Counting number of vowels

Consider the following vector `letrs` which contains various letters:

```r
# vector of letters
set.seed(1)
letrs <- sample(letters, size = 100, replace = TRUE)
head(letrs)

## [1] "g" "j" "o" "x" "f" "x"
```

If you were to count the number of vowels in `letrs` you would get the following counts:

- a: 2
- e: 2
- i: 6
- o: 2
- u: 8

Write some instructions in R to count the number of vowels in vector `letrs`.

```r
# count number of vowels
count_letters <- function(vec, letter) {
  length(which(tolower(vec) == letter))
}

number_of_letters_for_each_vowel_and_this_is_the_proper_function_name <-
function(letrs) {
  c(paste0("a: ", count_letters(letrs, "a")), paste0("e: ",
count_letters(letrs, "e")), paste0("i: ", count_letters(letrs, "i")),
paste0("o: ", count_letters(letrs, "o")), paste0("u: ", count_letters(letrs,
"u")))
}
```

```
number_of_letters_for_each_vowel_and_this_is_the_proper_function_name(letrs)

## [1] "a: 2" "e: 2" "i: 6" "o: 2" "u: 8"
```

Test your script with `letrs` and verify that you get the same counts for each vowel.

Number of letters, vowels, and consonants

Write a script that given a vector of letters (e.g. `letrs`) computes the total number of letters, the number of vowels, and the number of consonants. For instance, given the vector `letrs`, the script will print on console:

- `"letters: 100"`
- `"vowels: 20"`
- `"consonants: 80"`

```
# your script
paste0("letters: ", length(letrs))

## [1] "letters: 100"

paste0("vowels: ", sum(sapply(letrs, function(letter) {
  length(grep("[aeiou]", x = letter)) > 0
})))

## [1] "vowels: 20"

paste0("consonants: ", sum(sapply(letrs, function(letter) {
  length(grep("[^aeiou]", x = letter)) > 0
})))

## [1] "consonants: 80"
```