Handling strings

Andrew Ba Tran

Contents

This is from the third chapter of learn.r-journalism.com.

We're going to use the **stringr** package to manipulate text.

```
#install.packages("stringr")
library(stringr)

library(dplyr)
```

Each function starts with `str_`

Let's load this data in:

```
messy <- data.frame(name=c("Bill Smith", "jane doe", "John Forest-William"),
                    email=c("bsmith@themail.com", "jdoe@themail.com",
                            "jfwilliams$geemail.com"),
                    income=c("$90,000", "$140,000", "E8500"),
                    phone=c("(203) 847-334", "207-999-1122", "2128487345"),
                    activites=c("fishing, sailing, planting flowers", "reading,
                                raising flowers, biking", "hiking, fishing"))

messy
```

```
##                  name                  email   income         phone
## 1          Bill Smith     bsmith@themail.com  $90,000 (203) 847-334
## 2            jane doe       jdoe@themail.com $140,000  207-999-1122
## 3 John Forest-William jfwilliams$geemail.com    E8500    2128487345
##                                                activites
## 1                     fishing, sailing, planting flowers
## 2 reading, \n                     raising flowers, biking
## 3                                        hiking, fishing
```

What problems do you see?

Tasks

1. Split name into First name and Last name
2. Convert names to title case
3. Create a new variable identifying bad email addresses

4. Convert income to a new number in US dollars
5. Create a new variable containing area code
6. Creating a new variable counting how many activities each person is engaged in
7. Break activities into a set of useful dummy codes

Order of elements in date-time	Parse function
`str_length()`	figure out length of string
`str_c()`	combine strings
`str_sub()`	substitute string
`str_detect()`	detect string in string
`str_match()`	does string match
`str_count()`	count strings
`str_split()`	split strings
`str_to_upper()`	convert string to upper case
`str_to_lower()`	convert string to lower case
`str_to_title()`	convert the first letter of each word to upper case
`str_trim()`	eliminate trailing white space

String length

`str_length(string)` counts the number of characters in each element of a string or character vector.

```
x <- c("Bill", "Bob", "William")
str_length(x)
```

```
## [1] 4 3 7
```

Combine strings

`str_c(strings, sep="")`

It's like the equivalent of =concatenate in Excel.

But there are a couple of quirks

```
data <- data.frame(place=c("HQ", "HQ", "HQ"),
                   id=c("A", "B", "C"),
                   number=c("001", "002", "003"))

data
```

```
##   place id number
## 1    HQ  A    001
## 2    HQ  B    002
## 3    HQ  C    003
```

We can add a string to each value in the *number* column this way:

```
data <- data %>%
  mutate(combined=str_c("Num: ", number))

data
```

```
##   place id number combined
## 1    HQ  A    001 Num: 001
## 2    HQ  B    002 Num: 002
## 3    HQ  C    003 Num: 003
```

```
# A couple options that would've done the same thing:

data$combined <- str_c("Num: ", data$number)

# or

data <- data %>%
  mutate(combined=str_c("Num", number, sep=": "))
```

You can also pass the variable `collapse` to `str_c()` if you're turning an array of strings into one.

```
data <- data.frame(place=c("HQ", "HQ", "HQ"),
                   id=c("A", "B", "C"),
                   number=c("001", "002", "003"))

data
```

```
##   place id number
## 1    HQ  A    001
## 2    HQ  B    002
## 3    HQ  C    003
```

```
data %>%
  group_by(place) %>%
  summarize(ids_combined=str_c(number, collapse="-"))
```

```
## # A tibble: 1 x 2
##   place ids_combined
##   <fct> <chr>
## 1 HQ    001-002-003
```

subset strings

`str_sub(strings, start, end)` extracts and replaces substrings

```
x <- "Dr. James"

str_sub(x, 1, 3)
```

```
## [1] "Dr."
```

```
str_sub(x, 1, 3) <- "Mr."
x
```

```
## [1] "Mr. James"
```

Negative numbers count from the right.

```
x <- "baby"
str_sub(x, -3, -1)
```

```
## [1] "aby"
```

```
str_sub(x, -1, -1) <- "ies"
```

detect matches

`str_detect(strings, pattern)` returns T/F

```r
x <- c("Bill", "Bob", "David.Williams")
x
```

```
## [1] "Bill"           "Bob"            "David.Williams"
```

```r
str_detect(x, "il")
```

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

count matches

`str_count(strings, pattern)` count number of matches in a string

```r
x <- c("Assault/Robbery/Kidnapping")
x
```

```
## [1] "Assault/Robbery/Kidnapping"
```

```r
str_count(x, "/")
```

```
## [1] 2
```

```r
# How many offenses
str_count(x, "/") + 1
```

```
## [1] 3
```

extract matches

```r

x <- c("bsmith@microsoft.com", "jdoe@google.com", "jfwilliams@google.com")
str_extract(x, "@.+\\.com$")
```

```
## [1] "@microsoft.com" "@google.com"    "@google.com"
```

split strings

`str_split(string, pattern)` split a string into pieces

```r
x <- c("john smith", "mary todd", "bill holis")

str_split(x, " ", simplify=TRUE)
```

```
##      [,1]   [,2]
## [1,] "john" "smith"
## [2,] "mary" "todd"
## [3,] "bill" "holis"
```

```r
first <- str_split(x, " ", simplify=TRUE)[,1]
last  <- str_split(x, " ", simplify=TRUE)[,2]
```

replace a pattern

`str_replace(strings, pattern, replacement)` replace a pattern in a string with another string

```r
x <- c("john smith", "mary todd", "bill holis")
str_replace(x, "[aeiou]", "-")
```

```
## [1] "j-hn smith" "m-ry todd"  "b-ll holis"
```

```r
str_replace_all(x, "[aeiou]", "-")
```

```
## [1] "j-hn sm-th" "m-ry t-dd"  "b-ll h-l-s"
```

change case

`str_to_upper(strings)` is upper case `str_to_lower(strings)` is lower case `str_to_title(strings)` is title case

```r
x <- c("john smith", "Mary Todd", "BILL HOLLIS")

str_to_upper(x)
```

```
## [1] "JOHN SMITH"  "MARY TODD"   "BILL HOLLIS"
```

```r
str_to_lower(x)
```

```
## [1] "john smith"  "mary todd"   "bill hollis"
```

```r
str_to_title(x)
```

```
## [1] "John Smith"  "Mary Todd"   "Bill Hollis"
```

trim strings

`str_trim(strings)` remove white space at the beginning and end of string

```r
x <- c(" Assault", "Burglary ", " Kidnapping ")
str_trim(x)
```

```
## [1] "Assault"    "Burglary"   "Kidnapping"
```

Your turn

Challenge yourself with these exercises so you'll retain the knowledge of this section.

Instructions on how to run the exercise app are on the intro page to this section.