Stat 243, Fall 2016, Manipulating Strings

Gaston Sanchez

October 3, 2016

Strings Manipulation

The purpose of this lab is to work with regular expressions and string manipulations in R. The packages required to complete the work are: `"XML"`, `"stringr"`, and `"ggplot2"`.

```
library(XML)
library(stringr)
library(ggplot2)
```

Data

You will be working with the data set for the *Men's high jump world record progression*, available in wikipedia:

https://en.wikipedia.org/wiki/Men%27s_high_jump_world_record_progression

Height	Athlete	Venue	Date
2.00 m (6 ft 6¾ in)	George Horine (USA)	Palo Alto, California	18 May 1912[1]
2.022 m (6 ft 7⅝ in)	Edward Beeson (USA)	Berkeley, California	2 May 1914[3]
2.038 m (6 ft 8¼ in)	Harold Osborn (USA)	Urbana, Illinois	27 May 1924[4]
2.04 m (6 ft 8⅜ in)	Walter Marty (USA)	Fresno, California	13 May 1933[1]
2.06 m (6 ft 9⅛ in)	Walter Marty (USA)	Palo Alto, California	28 April 1934[1]
2.07 m (6 ft 9½ in)	Cornelius Johnson (USA)	New York	12 July 1936[1]
2.07 m (6 ft 9½ in)	Dave Albritton (USA)	New York	12 July 1936[1]

Figure 1: screenshot of the html table

Let's start by downloading the html file of the wikipedia page. To do this, we will use the function `paste0()` to assemble the address of the webpage. We can create a base url `wiki`, and then a separate string with the part that has to do with the Men's high jump work record:

```
# Download a copy of the HTML file,
wiki = "https://en.wikipedia.org/wiki/"
mens_high_jump = "Men%27s_high_jump_world_record_progression"

wiki_mhj = paste0(wiki, mens_high_jump)
```

```
download.file(wiki_mhj, "mens-high-jump.html")
```

Once you've downloaded the html content in the file `mens-high-jump.html`, you can use the function `readHTMLTable()` (from "XML") to read the tables:

```
# read in tables with readHTMLTable()
tbls <- readHTMLTable("mens-high-jump.html")

# how many html tables?
length(tbls)
```

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

```
# dimension of tables?
lapply(tbls, dim)
```

```
## $`NULL`
## [1] 40  4
##
## $`NULL`
## [1] 6 2
```

As ou can tell, `tbls` contains two HTML tables. Inspecting their dimensions, you should see that it is the first table the one that we are interested in. So let's re-read just the first table, and tell R to not convert strings as factors:

```
# the first table is the good one
dat <- readHTMLTable("mens-high-jump.html",
                     which = 1,
                     stringsAsFactors = FALSE)
```

Data Cleaning

The main goal is to clean the downloaded data frame in order to produce another data frame with columns:

- `height` (in meters)
- `name` (name of athlete)
- `first_name` (first name of athlete)
- `last_name` (last name of athlete)
- `country` (country abbreviation)
- `date` (in format `"%d %B %Y"`)
- `day` (number of day)
- `month` (abbreviated name, e.g. `Jan`, `Apr`, `Dec`)
- `year` (number of year)

Regex

Here's a table with some of the common regex patterns:

Pattern	Description
`abc`	letters
`123`	digits
`.`	any chracter
`\\.`	period
`[abd]`	only a, b, or c
`[^abc]`	not a, b, nor c
`[a-z]`	characters a to z
`[0-9]`	numbers 0 to 9
`{m}`	m repetitions
`{m,n}`	m to n repetitions
`*`	zero or more repetitions
`+`	one or more repetitions
`?`	optional character
`\\d`	any digit
`\\D`	any Non-digit
`\\w`	any alphanumeric character
`\\W`	any non-alphanumeric character
`\\s`	any whitespace
`\\S`	any Non-whitespace
`^a`	starts with a
`b$`	ends with b

Extracting meters

The column `Height` contains a character string with the record expressed both in meters and feet-inches. We want to extract only the value associated to meters.

My suggestion is to always start small. In this case, we can get a subset of values on which we can play:

```
tmp <- head(dat$Height)
tmp
```

```
## [1] "2.00 m (6 ft 6 ¾ in)"  "2.022 m (6 ft 7 5/8 in)"
## [3] "2.038 m (6 ft 8 ¼ in)" "2.04 m (6 ft 8 3/8 in)"
## [5] "2.06 m (6 ft 9 1/8 in)"  "2.07 m (6 ft 9 ½ in)"
```

With the values in `tmp`, let's try to match the value of meters.

Meters: Option 1. One possibility is to use `str_split()` to **split** the vector using `"m"` as the pattern to separate the values.

```
str_split(tmp, "m")
```

```
## [[1]]
## [1] "2.00 "            " (6 ft 6 ¾ in)"
##
## [[2]]
## [1] "2.022 "           " (6 ft 7 5/8 in)"
##
## [[3]]
## [1] "2.038 "           " (6 ft 8 ¼ in)"
##
## [[4]]
## [1] "2.04 "            " (6 ft 8 3/8 in)"
##
## [[5]]
## [1] "2.06 "            " (6 ft 9 1/8 in)"
##
## [[6]]
## [1] "2.07 "            " (6 ft 9 ½ in)"
```

The output is a list in which each element has two values: the character numbers of the meters, and the characters inside parenthesis of feet-inches.

Then we can use `sapply()` to loop over the list, and retrieve the first element:

```
tmp_split <- str_split(tmp, "m")
sapply(tmp_split, function(x) x[1])
```

```
## [1] "2.00 "  "2.022 " "2.038 " "2.04 "  "2.06 "  "2.07 "
```

Finally, we need to remove the extra spaces at the end of each string. One option to do this is with `str_trim()`. The last step consists in converting the characters into a numeric vector:

```
tmp_meters <- sapply(tmp_split, function(x) x[1])
str_trim(tmp_meters)
```

```
## [1] "2.00"  "2.022" "2.038" "2.04"  "2.06"  "2.07"
```

```
# and
as.numeric(str_trim(tmp_meters))
```

```
## [1] 2.000 2.022 2.038 2.040 2.060 2.070
```

Meters: Option 2. Another alternative to extract the height value of meters, is to use a regular expression that matches those values. Here is my solution with the pattern `"2\\.[0-9][0-9]+"`:

```r
# height in meters
str_extract(tmp, "2\\.[0-9][0-9]+")
```

```
## [1] "2.00"  "2.022" "2.038" "2.04"  "2.06"  "2.07"
```

The pattern `"2\\.[0-9][0-9]+"` indicates: match a `2`, followed by a dot `\\.`, followed by any number `[0-9]`, followed by one or more occurrances of another number `[0-9]+`.

```r
# numeric vector of height (in meters)
height <- as.numeric(str_extract(dat$Height, "2\\.[0-9][0-9]+"))
```

Extracting Athlete Name

The second task involves extracting the name of the athlete. If you inspect the column `Athlete`, you will see that all its values are formed with the first name, the last name, and the country inside parenthesis:

```r
ath <- head(dat$Athlete)
ath
```

```
## [1] "George Horine (USA)"     "Edward Beeson (USA)"
## [3] "Harold Osborn (USA)"     "Walter Marty (USA)"
## [5] "Walter Marty (USA)"      "Cornelius Johnson (USA)"
```

Work with the sample vector `ath` and try to `str_extract()` the first name. You can experiment with the *word* pattern `"\\w+"` (i.e. one or more alphanumeric characters):

```r
# your code (for first name)
```

Now use the patterns *word* `"\\w+"` and whitespace `\\s` to attempt extracting the athlete's full name (first and last names):

```r
# your code for the full name
```

How would you extract just the last name?

```r
# your code for the last name
```

Country

Next to the athlete's name we have the athelete's country (inside parentheses). To match parenthesis, you must escape them: `"\\("` and `"\\)"`.

Here's one pattern that matches zero or more characters inside parentheses:

```
str_extract(ath, "\\(.*\\)")
```

```
## [1] "(USA)" "(USA)" "(USA)" "(USA)" "(USA)" "(USA)"
```

To be more specific about the type of characters inside parenthesis, we can use a character class with three upper case letters:

```
str_extract(ath, "\\([A-Z][A-Z][A-Z]\\)")
```

```
## [1] "(USA)" "(USA)" "(USA)" "(USA)" "(USA)" "(USA)"
```

Since the pattern `"[A-Z]"` is repeated three times, we can specify that with a counter:

```
str_extract(ath, "\\([A-Z]{3}\\)")
```

```
## [1] "(USA)" "(USA)" "(USA)" "(USA)" "(USA)" "(USA)"
```

All of the options described above are able to math the country abbreviations, but they also match the parentheses.

Instead of specifying a regex pattern, we can use the function `str_sub()`. This function requires two indices: one for the `start` position, and the other one for the `end` position. For instance, if you want to substring the first three characters in the vector `ath`, simple specify:

```
str_sub(ath, start = 1, end = 3)
```

```
## [1] "Geo" "Edw" "Har" "Wal" "Wal" "Cor"
```

A very nice feature of `str_sub()` is that it allows you to specify negative indices. Here's how to get the last five characters:

```
str_sub(ath, start = -5)
```

```
## [1] "(USA)" "(USA)" "(USA)" "(USA)" "(USA)" "(USA)"
```

Find out how to play with `start` and `end` in order to get just the abbreviated country name (without the parenthesis)

```
# your code to str_sub() country name
```

Date

The date values are in the column `Date`:

```
dts <- head(dat$Date)
dts
```

```
## [1] "18 May 1912[1]"   "2 May 1914[3]"    "27 May 1924[4]"
## [4] "13 May 1933[1]"   "28 April 1934[1]" "12 July 1936[1]"
```

In addition to the date, there is also one or more numbers inside square brackets. To "remove" these brackets we will use `str_replace()` and a pattern that matches the brackets. Keep in mind that brackets are metacharacters; you must escape them if you want to match them: `"\\["` and `"\\]"`.

Let's start with the pattern `"\\[[0-9]\\]"`, that is, an open bracket, followed by a number, followed by a closing bracket:

```
str_replace(dts, "\\[[0-9]\\]", "")
```

```
## [1] "18 May 1912"   "2 May 1914"    "27 May 1924"   "13 May 1933"
## [5] "28 April 1934" "12 July 1936"
```

For the first six elements, the pattern works. But there are some rows that contain more than one number inside brackets:

```
str_replace(dat$Date[10:13], "\\[[0-9]\\]", "")
```

```
## [1] "27 June 1953[5]" "29 June 1956[6]" "13 July 1957"    "30 April 1960"
```

This means that we need to **repeat** the pattern. To do that, we surround the pattern with parenthesis, and we append a `"+"` to indicate one or more occurrances:

```
str_replace(dat$Date[10:13], "(\\[[0-9]\\])+", "")
```

```
## [1] "27 June 1953"  "29 June 1956"  "13 July 1957"  "30 April 1960"
```

```
# vector of dates
dates <- str_replace(dat$Date, "(\\[[0-9]\\])+", "")
```

Your turn: With the `dates` vector, extract in separate vectors the values of day, month name, and year:

```
# your code for days, months, and years
```

Finally, use the function `as.Date()` to format the vector `dates` as a vector of class `"Date"`. Use the format `""%d %B %Y"`:

```
# your code to format dates
```

Clean Data Frame

Assuming that you have created vectors for all the cleaned components, you should be able to create a data frame `high_jump`, for instance:

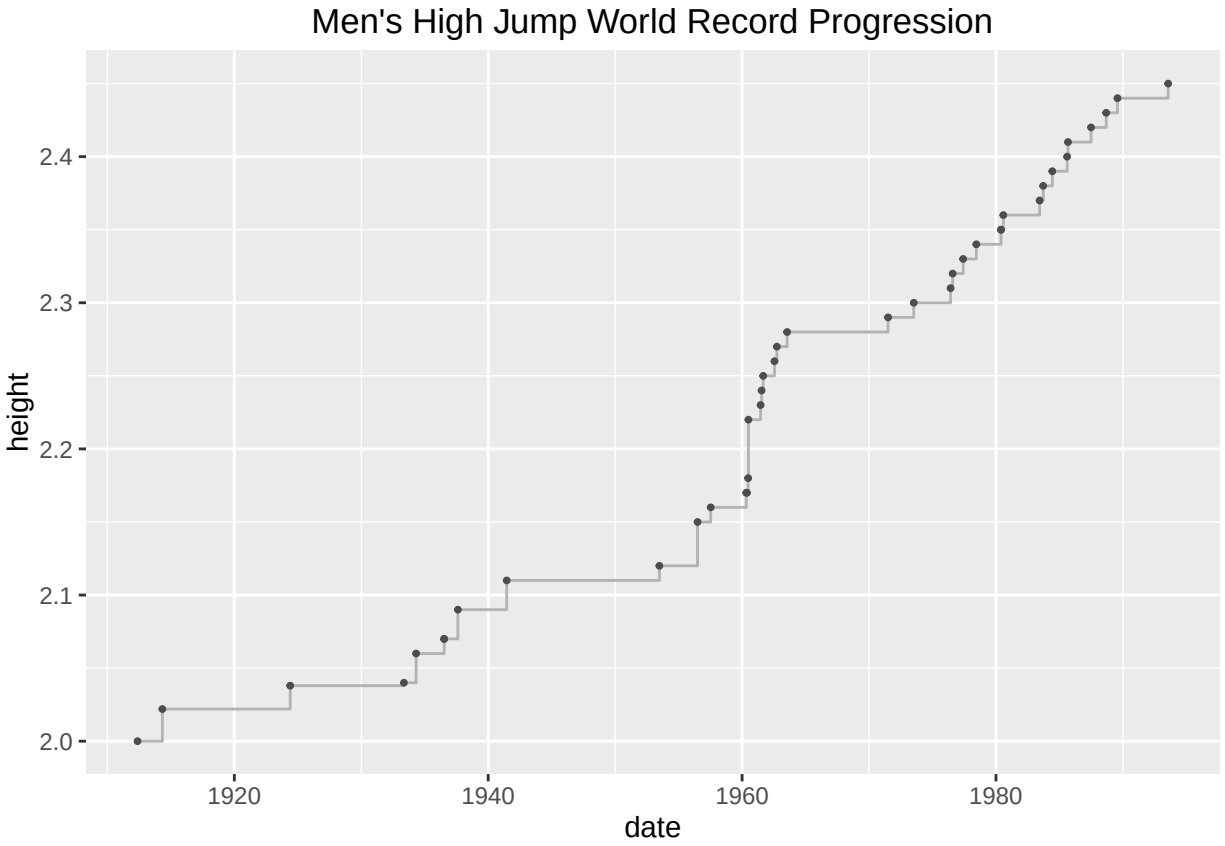
```
# clean data frame
high_jump <- data.frame(
  height = height,
  first_name = first_name,
  last_name = last_name,
  country = country,
  date = dates,
  day = day,
  month = month,
  year = year
)
```

```
##   height first_name last_name country       date day month year
## 1  2.000     George    Horine     USA 1912-05-18  18   May 1912
## 2  2.022     Edward    Beeson     USA 1914-05-02   2   May 1914
## 3  2.038     Harold    Osborn     USA 1924-05-27  27   May 1924
## 4  2.040     Walter     Marty     USA 1933-05-13  13   May 1933
## 5  2.060     Walter     Marty     USA 1934-04-28  28   Apr 1934
## 6  2.070  Cornelius   Johnson     USA 1936-07-12  12   Jul 1936
```

Plot of record progression

With the data frame `high_jump`, we can use the package `"ggplot2"` to plot a step-line chart that shows the progression over time:

```
ggplot(data = high_jump, aes(x = date, y = height)) +
  geom_step(color = "gray70") +
  geom_point(size = 0.7, color = "gray30") +
  ggtitle("Men's High Jump World Record Progression")
```



Use the column `country` as an *aesthetic* attribute to colored the points with ggplot.

```
# your code to add color of points based on 'country'
```

Exporting Clean Data

Use one of the writing table functions (e.g. `read.table()`) to export the data frame `high_jump` in an external file (e.g. CSV file)

```
# your code to save high_jump in a field-separated format file
```

Solutions at the end of this document

Solutions

```
# numeric vector of height (in meters)
height <- as.numeric(str_extract(dat$Height, "2\\.[0-9][0-9]+"))

# all athlete names are in the form "First Last"
# we can extract them using the word "\\w"
first_last = str_extract(dat$Athlete, "\\w+\\s\\w+")

# athlete first name
first_name = str_extract(first_last, "^\\w+")

# athlete last name
last_name = str_extract(first_last, "\\w+$")

# country
country = str_sub(dat$Athlete, -4, -2)

# dates
dates <- str_replace(dat$Date, "(\\[[0-9]\\])+", "")

# extracting day
day = as.numeric(str_extract(dates, "^[0-9]+"))

# extracting month
month = str_trim(str_extract(dates, "\\D+"))
month = str_sub(month, 1, 3)

# extracting year
year = as.numeric(str_extract(dates, "\\d+$"))

# reformat `dates`
dates <- as.Date(dates, "%d %B %Y")
```

```
# clean data frame
high_jump <- data.frame(
  height = height,
  first_name = first_name,
  last_name = last_name,
  country = country,
  date = dates,
  day = day,
  month = month,
  year = year
)
```

```r
# step line with colored points based on country
ggplot(data = high_jump, aes(x = date, y = height)) +
  geom_step(color = "gray70") +
  geom_point(size = 0.8, aes(color = country)) +
  ggtitle("Men's High Jump World Record Progression")
```