Lectures 20 and 21 - Regular Expressions

Professor Sanchez, Stat 133, Fall 2024

Warren Birkman

Table of contents

1 Lecture 20 - `Stringr`

```
library(stringr)
```

2 Regular Expressions

Shift in content - this week and next week we will be focused on text data, then weeks 11 and 12, we will focus on geo-spacial data, finally the last weeks will focus on web technologies. Getting data from the web and doing exploratory analysis.

The whole point of regex is pattern matching - to define a "certain amount of text"

We use letters, digits, and not all but some symbols]

- Literal characters: upper and lower case letters, digits, and a few symbols
- Meta-characters: set of characters (typically symbols) that have some special meaning

2.1 Function from `stringr`

- `str_view()`
- `str_detect()`
- `str.replace()`
- `str.split()`
- `str.match()`

```
txt1 = c('car','bus','bike','train','airplane')

txt1
```

```
[1] "car"      "bus"      "bike"     "train"    "airplane"
```

```
str_view(txt1, pattern = "a")
```

```
[1] | c<a>r
[4] | tr<a>in
[5] | <a>irpl<a>ne
```

```
str_view(txt1, pattern = "ai") # looking for an occurance of "ai"
```

```
[4] | tr<ai>n
[5] | <ai>rplane
```

Say we want to replace certain patterns with other text

```
str_replace(txt1, pattern = 'a', replacement='A')
```

```
[1] "cAr"      "bus"      "bike"     "trAin"    "Airplane"
```

```
str_replace_all(txt1, pattern = 'a', replacement='A')
```

```
[1] "cAr"      "bus"      "bike"     "trAin"    "AirplAne"
```

```
# Another vector
txt2 = c('5','4.0','4-0','500','555','$50')
str_view(txt2, pattern='50')
```

```
[4] | <50>0
[6] | $<50>
```

3 Metacharacter

The **DOT - The Wildcard Meta character: matches any other character**

```
str_view(txt2, pattern='.') # we get the dot but we also get every other character
```

```
[1] | <5>
[2] | <4><.><0>
[3] | <4><-><0>
[4] | <5><0><0>
[5] | <5><5><5>
[6] | <$><5><0>
```

3.1 The backslash \ : escape metacharacter

If we want to match the dot (referencing the cheat sheet: https://evoldyn.gitlab.io/evomics-2018/ref-sheets/R_strings.pdf)

We do \\.

```
str_view(txt2, pattern='\\.')
```

```
[2] | 4<.>0
```

3.2 Make sure you put two \s

```
str_view(txt2, pattern='\.')
```

```
Error: '\.' is an unrecognized escape in character string (<text>:1:26)
```

3.3 $ - a metacharacter

```
str_view(txt2, pattern='$5') # does not run
```

```
str_view(txt2, pattern="\\$5")
```

```
[6] | <$5>0
```

3.4 Character sets

These are a set of characters. A set is defined with `[]` - square brackets

vowels

```
str_view(txt1, pattern='[aeiou]')
```

```
[1] | c<a>r
[2] | b<u>s
[3] | b<i>k<e>
[4] | tr<a><i>n
[5] | <a><i>rpl<a>n<e>
```

Another set: `[AEIOU]`

another: `[0123456789]`

another: `[01234]`

another: `[abcdefghijklmnopqrstuvwxyz]`

```
str_view(txt2, pattern="[01234]")
```

```
[2] | <4>.<0>
[3] | <4>-<0>
[4] | 5<0><0>
[6] | $5<0>
```

3.4.1 Character ranges

- `[0-9]` - a range of values 0 to 9 - `[0123456789]`
- `[5-8]` = `[5678]`
- We can also form ranges of letters: `[a-z]`
- And upper case `[A-Z]`
- Alphanumeric ranges: `[0-9a-fA-F]`
- `[3-7m-xG-M]`

```
str_remove(txt1, pattern = '[p-z]')
```

```
[1] "ca"      "bs"      "bike"    "rain"    "aiplane"
```

```
str_remove_all(txt1, pattern='[p-z]')
```

```
[1] "ca"     "b"      "bike"   "ain"    "ailane"
```

4 Lecture 21 - Men's Long jump world record progression

```
# Import packages
library(tidyverse)
```

```
-- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
v dplyr     1.1.4     v purrr     1.0.2
v forcats   1.0.0     v readr     2.1.5
v ggplot2   3.5.1     v tibble    3.2.1
v lubridate 1.9.3     v tidyr     1.3.1
-- Conflicts ------------------------------------------ tidyverse_conflicts() --
x dplyr::filter() masks stats::filter()
x dplyr::lag()    masks stats::lag()
i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become errors
```

```
# allows us to import data from the web
library(rvest)
```

```
Attaching package: 'rvest'

The following object is masked from 'package:readr':

    guess_encoding
```

```
wiki = "https://en.wikipedia.org/wiki/Men's_long_jump_world_record_progression"

# HTML Doc
doc = read_html(wiki)
class(doc) # we'll return to this
```

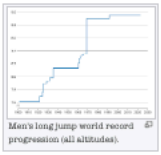
```
[1] "xml_document" "xml_node"
```

```
# extract html tables from html doc
tbl = html_table(doc)[[1]] # This grabs the tables from the html doc
# We only want to work with the first table
```

Tables on the wiki page:

Record progression [edit]

Mark	Wind	Athlete	Place	Date
7.61 m (24 ft 11½ in)		Peter O'Connor (IRE)	Dublin, Ireland	5 August 1901[1]
7.69 m (25 ft 2¾ in)		Edward Gourdin (USA)	Cambridge, United States	23 July 1921[1]
7.76 m (25 ft 5½ in)		Robert LeGendre (USA)	Paris, France	7 July 1924[1]
7.89 m (25 ft 10½ in)		DeHart Hubbard (USA)	Chicago, United States	13 June 1925[1]
7.90 m (25 ft 11 in)		Edward Hamm (USA)	Cambridge, United States	7 July 1928[1]
7.93 m (26 ft 0 in)	0.0	Sylvio Cator (HAI)	Paris, France	9 September 1928[1]
7.98 m (26 ft 2 in)	0.5	Chuhei Nambu (JPN)	Tokyo, Japan	27 October 1931[1]
8.13 m (26 ft 8 in)	1.5	Jesse Owens (USA)	Ann Arbor, United States	25 May 1935[1]
8.21 m (26 ft 11 in)	0.0	Ralph Boston (USA)	Walnut, United States	12 August 1960[1]
8.24 m (27 ft ¼ in)	1.8	Ralph Boston (USA)	Modesto, United States	27 May 1961[1]
8.28 m (27 ft 1¾ in)	1.2	Ralph Boston (USA)	Moscow, Soviet Union	16 July 1961[1]
8.31 m (27 ft 3 in) A	−0.1	Igor Ter-Ovanesyan (URS)	Yerevan, Soviet Union	10 June 1962[1]
8.33 m (27 ft 3¾ in)[2]		Phil Shinnick (USA)	Modesto, United States	25 May 1963[3]
8.31 m (27 ft 3 in)	0.0	Ralph Boston (USA)	Kingston, Jamaica	15 August 1964[note 1][4]
8.34 m (27 ft 4¼ in)	1.0	Ralph Boston (USA)	Los Angeles, United States	12 September 1964[1]
8.35 m (27 ft 4½ in)[5]	0.0	Ralph Boston (USA)	Modesto, United States	29 May 1965[1][6]
8.35 m (27 ft 4½ in) A	0.0	Igor Ter-Ovanesyan (URS)	Mexico City, Mexico	19 October 1967[1]
8.90 m (29 ft 2¼ in) A	2.0	Bob Beamon (USA)	Mexico City, Mexico	18 October 1968[1]
8.95 m (29 ft 4¼ in)	0.3	Mike Powell (USA)	Tokyo, Japan	30 August 1991[1]



Men's long jump world record progression (all altitudes).

Low-altitude record progression 1965–1991 [edit]

The IAAF considers marks set at high altitude as acceptable for record consideration. However, high altitude can significantly assist long jump performances. At the 1968 Summer Olympics in Mexico City, Bob Beamon broke the existing record by a margin of 55 cm (21½ in), and his world record of 8.90 m (29 ft 2¼ in) stood until Mike Powell jumped 8.95 m (29 ft 4¼ in) in 1991. However, Beamon's jump was set at an altitude of 2,292 m (7,520 ft), with a maximum allowable wind, factors which assisted his performance.[7]

This list contains the progression of long jump marks set at low altitude starting with the mark that stood at Beamon's record in 1968 to Powell's 1991 world record.

Mark	Wind	Athlete	Place	Date
8.35 m (27 ft 4½ in)[5]	0.0	Ralph Boston (USA)	Modesto, United States	29 May 1965[1][6]
8.35 m (27 ft 4½ in)	0.8	Josef Schwarz (FRG)	Stuttgart, West Germany	15 July 1970[8]
8.45 m (27 ft 8½ in)	2.0	Nenad Stekić (YUG)	Montreal, Canada	25 July 1975[8]
8.52 m (27 ft 11¼ in)	0.0	Larry Myricks (USA)	Montreal, Canada	26 August 1979[8]
8.54 m (28 ft 0 in)	0.9	Lutz Dombrowski (GDR)	Moscow, Soviet Union	28 July 1980[8]
8.62 m (28 ft 3¼ in)	0.8	Carl Lewis (USA)	Sacramento, United States	20 June 1981[8]
8.76 m (28 ft 8¾ in)	1.0	Carl Lewis (USA)	Indianapolis, United States	24 July 1982[8]
8.79 m (28 ft 10 in)	1.9	Carl Lewis (USA)	Indianapolis, United States	19 June 1983[8]
8.95 m (29 ft 4¼ in)	0.3	Mike Powell (USA)	Tokyo, Japan	30 August 1991[1]

```
tbl
```

```
# A tibble: 19 x 5
   Mark                        Wind   Athlete                 Place        Date
   <chr>                       <chr>  <chr>                   <chr>        <chr>
 1 7.61 m (24 ft 11+½ in)      ""       Peter O'Connor (IRE)    Dublin, Ire~ 5 Au~
 2 7.69 m (25 ft 2+¾ in)       ""       Edward Gourdin (USA)    Cambridge, ~ 23 J~
 3 7.76 m (25 ft 5+½ in)       ""       Robert LeGendre (USA)   Paris, Fran~ 7 Ju~
 4 7.89 m (25 ft 10+½ in)      ""       DeHart Hubbard (USA)    Chicago, Un~ 13 J~
 5 7.90 m (25 ft 11 in)          ""       Edward Hamm (USA)       Cambridge, ~ 7 Ju~
 6 7.93 m (26 ft 0 in)           "0.0"    Sylvio Cator (HAI)      Paris, Fran~ 9 Se~
 7 7.98 m (26 ft 2 in)           "0.5"    Chuhei Nambu (JPN)      Tokyo, Japan 27 O~
 8 8.13 m (26 ft 8 in)           "1.5"    Jesse Owens (USA)       Ann Arbor, ~ 25 M~
 9 8.21 m (26 ft 11 in)          "0.0"    Ralph Boston (USA)      Walnut, Uni~ 12 A~
10 8.24 m (27 ft ¼ in)         "1.8"    Ralph Boston (USA)      Modesto, Un~ 27 M~
11 8.28 m (27 ft 1+¾ in)       "1.2"    Ralph Boston (USA)      Moscow, Sov~ 16 J~
12 8.31 m (27 ft 3 in) A         "-0.1" Igor Ter-Ovanesyan (URS) Yerevan, So~ 10 J~
13 8.33 m (27 ft 3+¾ in)[2] ""       Phil Shinnick (USA)     Modesto, Un~ 25 M~
```

```
14 8.31 m (27 ft 3 in)        "0.0"  Ralph Boston (USA)       Kingston, J~ 15 A~
15 8.34 m (27 ft 4+¼ in)    "1.0"  Ralph Boston (USA)       Los Angeles~ 12 S~
16 8.35 m (27 ft 4+½ in)[5] "0.0"  Ralph Boston (USA)       Modesto, Un~ 29 M~
17 8.35 m (27 ft 4+½ in) A  "0.0"  Igor Ter-Ovanesyan (URS) Mexico City~ 19 O~
18 8.90 m (29 ft 2+¼ in) A  "2.0"  Bob Beamon (USA)         Mexico City~ 18 O~
19 8.95 m (29 ft 4+¼ in)    "0.3"  Mike Powell (USA)        Tokyo, Japan 30 A~
```

Since all of the data in these columns are character vectors, we will need to use regular expressions to extract the values and turn them into integers or double vectors. Graphing this would not work because of this.

4.1 Regular expression time!

```
# Mark column
# Recall: \\ = escape metacharacter
mark1 = as.numeric(str_extract(tbl$Mark, pattern="[0-9]\\.[0-9][0-9]"))
mark1
```

```
 [1] 7.61 7.69 7.76 7.89 7.90 7.93 7.98 8.13 8.21 8.24 8.28 8.31 8.33 8.31 8.34
[16] 8.35 8.35 8.90 8.95
```

Ex: POSIX class for digits:

```
as.numeric(str_extract(tbl$Mark, pattern='[:digit:]\\.[:digit:][:digit:]'))
```

```
 [1] 7.61 7.69 7.76 7.89 7.90 7.93 7.98 8.13 8.21 8.24 8.28 8.31 8.33 8.31 8.34
[16] 8.35 8.35 8.90 8.95
```

4.2 Abbreviated class

- `\\s` - space character
  - `\\S` - not spaces
- `\\w` - "word" character (uppercase, lowercase, letters, digits, and the underscore _)
  - (`[a-zA-Z0-9_]`)
  - `\\W` - not a word character
- `\\d` - for digits (`[0-9]`, `[:digit:]`)
  - `\\D` - not a digit / no digits
- `\\b` - word boundaries
  - `\\B` - not a word boundary

```
as.numeric(str_extract(tbl$Mark, patter='\\d.\\d\\d'))
```

```
 [1] 7.61 7.69 7.76 7.89 7.90 7.93 7.98 8.13 8.21 8.24 8.28 8.31 8.33 8.31 8.34
[16] 8.35 8.35 8.90 8.95
```

4.3 Quantifiers (Modifiers)

Example	Description	Quantity
`"a*"`	any number of "a"	$[0, \infty)$
`"a+"`	at least one "a"	$[1, \infty)$
`"a?"`	optional "a"	$[0, 1]$

Let's change columns

```
str_extract(tbl$Athlete, pattern="\\w\\w\\w")
```

```
 [1] "Pet" "Edw" "Rob" "DeH" "Edw" "Syl" "Chu" "Jes" "Ral" "Ral" "Ral" "Igo"
[13] "Phi" "Ral" "Ral" "Ral" "Igo" "Bob" "Mik"
```

```
str_extract(tbl$Athlete, pattern="\\w*")
```

```
 [1] "Peter"  "Edward" "Robert" "DeHart" "Edward" "Sylvio" "Chuhei" "Jesse"
 [9] "Ralph"  "Ralph"  "Ralph"  "Igor"   "Phil"   "Ralph"  "Ralph"  "Ralph"
[17] "Igor"   "Bob"    "Mike"
```

```
str_extract(tbl$Athlete, pattern='\\w+')
```

```
 [1] "Peter"  "Edward" "Robert" "DeHart" "Edward" "Sylvio" "Chuhei" "Jesse"
 [9] "Ralph"  "Ralph"  "Ralph"  "Igor"   "Phil"   "Ralph"  "Ralph"  "Ralph"
[17] "Igor"   "Bob"    "Mike"
```

```
str_view(tbl$Athlete[1:5],pattern='\\w*')
```

```
[1] | <Peter><> <O><>'<Connor><>{\u00a0}<>(<IRE><>)<>
[2] | <Edward><> <Gourdin><>{\u00a0}<>(<USA><>)<>
[3] | <Robert><> <LeGendre><>{\u00a0}<>(<USA><>)<>
[4] | <DeHart><> <Hubbard><>{\u00a0}<>(<USA><>)<>
[5] | <Edward><> <Hamm><>{\u00a0}<>(<USA><>)<>
```

```
str_view(tbl$Athlete[1:5],pattern='\\w+')
```

```
[1] | <Peter> <O>'<Connor>{\u00a0}(<IRE>)
[2] | <Edward> <Gourdin>{\u00a0}(<USA>)
[3] | <Robert> <LeGendre>{\u00a0}(<USA>)
[4] | <DeHart> <Hubbard>{\u00a0}(<USA>)
[5] | <Edward> <Hamm>{\u00a0}(<USA>)
```

For the two above code chunks the outputs for `str_extract()` will be the same, but the patterns are slightly different. when `w*`, we get any character including blank space, and when we do `w+` we get any string greater than 1 in length.

```
str_view(tbl$Athlete[1:5],pattern='\\w?')
```

```
[1] | <P><e><t><e><r><> <O><>'<C><o><n><n><o><r><>{\u00a0}<>(<I><R><E><>)<>
[2] | <E><d><w><a><r><d><> <G><o><u><r><d><i><n><>{\u00a0}<>(<U><S><A><>)<>
[3] | <R><o><b><e><r><t><> <L><e><G><e><n><d><r><e><>{\u00a0}<>(<U><S><A><>)<>
[4] | <D><e><H><a><r><t><> <H><u><b><b><a><r><d><>{\u00a0}<>(<U><S><A><>)<>
[5] | <E><d><w><a><r><d><> <H><a><m><m><>{\u00a0}<>(<U><S><A><>)<>
```

Every character becomes an individual match in this one.