Regular Expressions

MSDS 597 Data Wrangling & Husbandry

2/10/2020

Pattern matching

Matching, and sometimes replacing patterns, comes up frequently when working with strings.

- Find addresses with “NJ” on them (but not words like “injury”)
- Find “Trump”, but only at the start of a line
- Find “skeptic” or “sceptic”
- Find phone numbers like “123-456-1234” and rewrite them as “(123) 456-1234”
- Extract the latitide and longitude from a string like
    - “120 Cedar Grove Lane\nSomerset, NJ 08873-6462\n(40.521568, -74.521327)”

Regular expressions

- Regular expressions, also known as regex or sometimes grep expressions, are just a sequence of characters that define a search pattern
- The expressions can look impenetrable (e.g., "^0?[1-9]$|[12][0-9]|3[01]" matchs valid days of the month for a month with 31 days), but can mostly be built up out of simpler pieces
- There are a few complications in R having to do with special characters
- You can get help using `help(regex)`; there's also a RegEx cheatsheet as well as the RStudio "Strings" cheatsheet

In R we use regular expressions via base R functions or functions from the `stringr` package (part of the tidyverse). The regular expressions themselves are the same in both cases.

- To understand patterns and matches
  - `str_view(string, pattern)` will show in the viewer where the patten matches the string
- To detect patterns
  - `grep(pattern, string)` gives the index
  - `grep(pattern, string, value = TRUE)` gives the string(s)
  - `grepl(pattern, string)` gives a vector of TRUE and FALSE
  - `str_detect(string, pattern)` gives a vector of TRUE and FALSE *Note the reversed order*

Building patterns

- a regex of ordinary characters just matches those characters
    - `. \\ | ( ) [ ] $ * + ?` are all special characters

```
library(stringr)
regex.ex <- c("It is", "a truth", "universally", "acknowledged", "321")
grep("is", regex.ex)
```

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

```
str_detect(regex.ex, "21")
```

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

- `.` is a wild card that matches any character other than end of line
- characters inside `[]` match any of those characters
  - `[aeiou]` will match any vowel
- a `^` inside the `[]` means any except these
  - `[^aeiou]` will match any character that's not a vowel

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
grep(".a", regex.ex)
```

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

```
grep("[ir]s", regex.ex)
```

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

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
str_detect(regex.ex, "[aeiou]l")
```

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

```
str_detect(regex.ex, "[^aeiou]l")
```

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

Repeats

Following a pattern with

- `+` means zero or more times
- `?` means zero or one times
- `*` means one or more times
- `{2}` means exactly two times
- `{2,4}` means from two to four times
- `{2, }` means two or more times

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
grep("e.+e", regex.ex)
```

```
## [1] 4
```

```
str_detect(regex.ex, "l{2,}")
```

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

Character classes

- `[[:alpha:]]` or `[A-z]` will match alphabetic characters
- `[[:digit:]]` or `[0-9]` or `\d`` will match digits
- `[[:blank:]]` matches space and tab
- `[[:space:]]` or `\\s` matches space, tab, new line

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
grep("[[:blank:]]", regex.ex)
```

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

```
grep("[[:blank:]]t", regex.ex)
```

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

```
str_detect(regex.ex, "[0-9]")
```

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

```
str_detect(regex.ex, "[0-9]{4,}")
```

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

- `|` works as or
- `^` means the start of a string
- `$` means the end of a string
- any of these can be literally matched by starting with `\\`

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
grepl("^a", regex.ex)
```

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

```
str_detect(regex.ex, "^a|s$")
```

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

```
str_detect("$100", "\\$")
```

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

```
pattern31 <- "^0?[1-9]$|[12][0-9]|3[01]"
str_detect(c("3", "03", "19", "31", "32", "99", "1a"), pattern31)
```

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

- To locate patterns
  - `str_locate(string, pattern)` gives the start and end positions of the first match
  - `str_locate_all(string, pattern)` gives the start and end positions of all matches
  - `str_match()` and `str_match_all()` will also show the “capture groups”

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
str_locate(regex.ex, "t")
```

```
##      start end
## [1,]     2   2
## [2,]     3   3
## [3,]    NA  NA
## [4,]    NA  NA
## [5,]    NA  NA
```

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
str_locate_all(regex.ex, "t")
```

```
## [[1]]
##      start end
## [1,]     2   2
##
## [[2]]
##      start end
## [1,]     3   3
## [2,]     6   6
##
## [[3]]
##      start end
##
## [[4]]
##      start end
##
## [[5]]
##      start end
```

- To extract patterns
  - `str_extract(string, pattern)` extracts the first match
  - `str_extract_all(string, pattern)` extracts all matchs, outputs a list

```
strings <- c(" 219 733 8965", "329-293-8753 ", "banana",
  "(387) 287-6718", "239 923 8115 and 842 566 4692", "595 794 7569", "apple", "233.39
   "Work: 579-499-7527", "$1000",
  "Home: 543.355.3679")
phone <- "([2-9][0-9]{2})\\)?[- .]?([0-9]{3})[- .]?([0-9]{4})"
```

```
str_extract(strings, phone)
```

```
##  [1] "219 733 8965"  "329-293-8753"  NA              "387) 287-6718"
##  [5] "239 923 8115"  "595 794 7569"  NA              "233.398.9187"
##  [9] "482 952 3315"  "579-499-7527"  NA              "543.355.3679"
```

```
str_extract_all(strings, phone)
```

```
## [[1]]
## [1] "219 733 8965"
##
## [[2]]
## [1] "329-293-8753"
##
## [[3]]
## character(0)
##
## [[4]]
## [1] "387) 287-6718"
##
## [[5]]
## [1] "239 923 8115" "842 566 4692"
##
## [[6]]
## [1] "595 794 7569"
##
## [[7]]
## character(0)
##
## [[8]]
## [1] "233.398.9187"
```

```
str_match(strings, phone)
```

```
##       [,1]            [,2]  [,3]  [,4]
##  [1,] "219 733 8965"  "219" "733" "8965"
##  [2,] "329-293-8753"  "329" "293" "8753"
##  [3,] NA              NA    NA    NA
##  [4,] "387) 287-6718" "387" "287" "6718"
##  [5,] "239 923 8115"  "239" "923" "8115"
##  [6,] "595 794 7569"  "595" "794" "7569"
##  [7,] NA              NA    NA    NA
##  [8,] "233.398.9187"  "233" "398" "9187"
##  [9,] "482 952 3315"  "482" "952" "3315"
## [10,] "579-499-7527"  "579" "499" "7527"
## [11,] NA              NA    NA    NA
## [12,] "543.355.3679"  "543" "355" "3679"
```

```
str_match_all(strings, phone)
```

```
## [[1]]
##      [,1]           [,2]  [,3]  [,4]
## [1,] "219 733 8965" "219" "733" "8965"
##
## [[2]]
##      [,1]           [,2]  [,3]  [,4]
## [1,] "329-293-8753" "329" "293" "8753"
##
## [[3]]
##      [,1] [,2] [,3] [,4]
##
## [[4]]
##      [,1]            [,2]  [,3]  [,4]
## [1,] "387) 287-6718" "387" "287" "6718"
##
## [[5]]
##      [,1]           [,2]  [,3]  [,4]
## [1,] "239 923 8115" "239" "923" "8115"
## [2,] "842 566 4692" "842" "566" "4692"
##
## [[6]]
##      [,1]           [,2]  [,3]  [,4]
## [1,] "595 794 7569" "595" "794" "7569"
```

Finally, we can replace patterns

- `sub(pattern, replacement, string)` replaces the first match
- `gsub(pattern, replacement, string)` replaces all matches
- `str_replace(string, pattern, replacement)` replaces the first match
- `str_replace_all(string, pattern, replacement)` replaces all matches

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
str_replace(regex.ex, "universally", "rarely")
```

```
## [1] "It is"        "a truth"      "rarely"       "acknowledged" "321"
```

```
regex.ex
```

```
## [1] "It is"        "a truth"      "universally"  "acknowledged" "321"
```

```
str_replace(regex.ex, "l", "!")
```

```
## [1] "It is"        "a truth"      "universa!ly"  "acknow!edged" "321"
```

```
str_replace_all(regex.ex, "l", "!")
```

```
## [1] "It is"        "a truth"      "universa!!y"  "acknow!edged" "321"
```

```
strings
```

```
##  [1] " 219 733 8965"                "329-293-8753 "
##  [3] "banana"                       "(387) 287-6718"
##  [5] "239 923 8115 and 842 566 4692" "595 794 7569"
##  [7] "apple"                        "233.398.9187  "
##  [9] "482 952 3315"                 "Work: 579-499-7527"
## [11] "$1000"                        "Home: 543.355.3679"
```

```
str_replace(strings, phone, "\\1-\\2-\\3")
```

```
##  [1] " 219-733-8965"                "329-293-8753 "
##  [3] "banana"                       "(387-287-6718"
##  [5] "239-923-8115 and 842 566 4692" "595-794-7569"
##  [7] "apple"                        "233-398-9187  "
##  [9] "482-952-3315"                 "Work: 579-499-7527"
## [11] "$1000"                        "Home: 543-355-3679"
```

You can use `(?i)` as a prefix to ignore case; in the base R functions you can also use the `ignore.cases = TRUE` option

```
str_replace_all(regex.ex, "(?i)i", "&")
```

```
## [1] "&t &s"        "a truth"      "un&versally"  "acknowledged" "321"
```

In class exercises:

- Find the strings that *end* with “our” or “or” in `c(“colour”, “food”, “color”, “favour”, “favorite”, “or”, “our”)
- Do the same, but only if “our” or “or” is part of a larger word
- Replace the “our” with “or” if “our” is part of a larger word.
- Replace every occurance of ‘f’,‘v’, and ‘r’ with ‘ff’, ‘vv’, and ‘rr’ by using `\\1` in the replacement.
- Make a vector of babynames (use the function `unique()`). How many have exactly two vowels in a row? Three or more vowels in a row? Any with four or more vowels? What are they?
- Find some babynames that start with consonent vowel consonent vowel consonent vowel consonent vowel consonent vowel