R String Regular Expression (Regex)

Fan Wang

2024-01-01

Contents

1 String Regular Expression

Go to the **RMD**, **R**, **PDF**, or **HTML** version of this file. Go back to fan's REconTools research support package, R4Econ examples page, PkgTestR packaging guide, or Stat4Econ course page.

1.1 Character Class

The regex documentation states that: "A character class is a list of characters enclosed between '[' and ']' which matches any single character in that list"

First, in the example below, we look for strings that contain at a single letter, symbol, or number in the string list enclosed in square brackets.

```
# Fou words with metacharacters
ls_st_regex_charclass <- c(
  '00d',
  'z\\12323_4',
  'pa(_2+\\3',
  'p99.9_sdfasdpf0',
  'k9p.e_d+fd')
# Matches any characters with the letter p
print(grepl("[p]", ls_st_regex_charclass))
# Matches any characters with backslash
print(grepl("[\\]", ls_st_regex_charclass))
# Matches any characters with the number 3
print(grepl("[3]", ls_st_regex_charclass))

# > print(grepl("[p]", ls_st_regex_charclass))
# [1] FALSE FALSE  TRUE  TRUE  TRUE
# > print(grepl("[\\]", ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE FALSE FALSE
# > print(grepl("[3]", ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE FALSE FALSEZ
```

Second, using the same set of words as examples, we now test if the strings contain at least a letter, symbol, or number in the string lis enclosed in square brakets.

```r
# Matches any characters eithr with letter p or d
print(grepl('[pd]', ls_st_regex_charclass))
# Matches any characters eithr with letter p or _
print(grepl('[p_]', ls_st_regex_charclass))
# Matches any characters eithr with letter p or _ or 0
print(grepl('[p_0]', ls_st_regex_charclass))

# > print(grepl('[pd]', ls_st_regex_charclass))
# [1]  TRUE FALSE  TRUE  TRUE  TRUE
# > print(grepl('[p_]', ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE  TRUE  TRUE
# > print(grepl('[p_0]', ls_st_regex_charclass))
# [1] TRUE TRUE TRUE TRUE TRUE
```

Third, using '^', carat, we exclude strings that include characters, letters, and symols. The documentation states that: "unless the first character of the list is the caret '^', when it matches any character not in the list".

```r
# Finds strings that has anything other than d and 0
print(grepl('[^d0]', ls_st_regex_charclass))

# > print(grepl('[^d0]', ls_st_regex_charclass))
# [1] FALSE  TRUE  TRUE  TRUE  TRUE
```

1.2 Repetition Quantifiers

We have the following quantifiers:

- '?': The preceding item is optional and will be matched at most once.
- '*': The preceding item will be matched zero or more times.
- '+':The preceding item will be matched one or more times.
- '{n}': The preceding item is matched exactly n times.
- '{n,}': The preceding item is matched n or more times.
- '{n,m}': The preceding item is matched at least n times, but not more than m times.

Now, we identifier strings where certain characters appear a certain number of times.

```r
# Fou words with metacharacters
ls_st_regex_rep_quantifer <- c(
  '00d',
  'z\\12323_40',
  'ppa(_2+\\3',
  'p99.9_sdfasdpf0',
  'k9p.e_d+fd')
# Matches any characters pp
print(grepl("[p]{2}", ls_st_regex_rep_quantifer))
# Matches any characters with the number 3
print(grepl("[9]{2}", ls_st_regex_rep_quantifer))

# > print(grepl("[p]{2}", ls_st_regex_rep_quantifer))
# [1] FALSE FALSE  TRUE FALSE FALSE
# > print(grepl("[9]{2}", ls_st_regex_rep_quantifer))
# [1] FALSE FALSE FALSE  TRUE FALSE
```

1.3 Matches Strings With Multiple Conditions with Repetition Quantifiers

Now we match string that satisfy multiple conditions jointly. We have the following quantifiers:

- '?': The preceding item is optional and will be matched at most once.
- '*': The preceding item will be matched zero or more times.
- '+':The preceding item will be matched one or more times.
- '{n}': The preceding item is matched exactly n times.
- '{n,}': The preceding item is matched n or more times.
- '{n,m}': The preceding item is matched at least n times, but not more than m times.

First, we define our string array.

```
ls_st_regex_joint <- c(
  '_asdf123p',
  'pz12p323_40_',
  'ppa(_2+\\3',
  'p9_sdfasdpf0',
  'p_k9p.e_d+fd',
  'p123k_dfk')
```

Second, we identify three cases below:

1. Matching words containing just "p_"
2. Matching words containing "p9_" (replace 9 by another other alpha-numeric)
3. Matching words containing either "p_" or "p9_"

```
# Start with p, followed by _
print(grepl("p_", ls_st_regex_joint))
# Start with p, followed by a single alpha-numeric, then _
print(grepl("p[[:alnum:]]_", ls_st_regex_joint))
# Start with p, followed by either:
# 1 single alpha-numeric, then _
# no alpha-numeric, then _
print(grepl("p[[:alnum:]]?_", ls_st_regex_joint))

# > print(grepl("p_", ls_st_regex_joint))
# [1] FALSE FALSE FALSE FALSE  TRUE FALSE
# > print(grepl("p[[:alnum:]]_", ls_st_regex_joint))
# [1] FALSE FALSE FALSE  TRUE FALSE FALSE
# > print(grepl("p[[:alnum:]]?_", ls_st_regex_joint))
# [1] FALSE FALSE FALSE  TRUE  TRUE FALSE
```

Third, we identify cases, where there the word contains substring starting with "p" and ending with "_", with any number (including 0) of alpha-numeric characters in between. Note:

1. In the first string, both "_" and "p" appear, but "p" appears after, so does not match
2. Note in the second word, "p" and "_" appear multiple times
3. Note in the third word, "p" and "_" both appear, but are separated by a non-alpha-numeric character

```
print(grepl("p[[:alnum:]]*_", ls_st_regex_joint))

# > print(grepl("p[[:alnum:]]*_", ls_st_regex_joint))
# [1] FALSE  TRUE FALSE  TRUE  TRUE  TRUE
```

Fourth, we use alternative repetition quantifiers, plus, rather than asterisks, which means we must have at least one alpha-numeric character in between "p" and the "_", in which case, the fifth word no longer satisfies the search condition.

```r
# p and _ separated by at least 1 alpha numerics
print(grepl("p[[:alnum:]]+_", ls_st_regex_joint))

# > print(grepl("p[[:alnum:]]+_", ls_st_regex_joint))
# [1] FALSE  TRUE FALSE  TRUE FALSE  TRUE
```