Reading: Basic Regular Expressions

Regular Expressions are generic expressions that are used to match patterns in strings and text. It is extremely useful and powerful for processing textual data such as checking if strings matching certain patterns e.g., an email or telephone pattern, or detecting and extracting substrings such as a domain name in an email address.

This reading should give you enough information to understand and start using regular expressions.

Character classes

Character classes are used to matching a single specific character pattern, normally wrapped by a pair of square brackets `[ ]`.

For example, for words `gray` and `grey`, we can see a pattern here that the third character is either `a` or `e` so we can create a regular expression pattern here like `gr[ae]y` or `gr[a|e]y`.

Here we give a list of commonly used character classes with the use `grep(pattern, string)` function to check if a string matches a character class:

- `.` Any character exception for the line break `\n`. For example, `a` or `A`.
- `|` Or. For example, `a|b` means character `a` or `b`
- `[...]` A character from a list, for example `[abc]` means a character to be `a` or `b` or `c`

  Code example:

  ```
  strings <- c("abc", "ABC", "grey", "gray", "grrr")
  pattern <- "gr[ae]y"
  grep(pattern, strings, value=TRUE)
  ```

  ```
  Output: 'grey''gray'
  ```

- `[a-z]` A character range, for example a character in the range from `a` to `z`

  Code example:

  ```
  strings <- c("a", "A", "B", "G", "d")
  pattern <- "[a-z]"
  grep(pattern, strings1, value=TRUE)
  ```

  ```
  Output: 'a''d'
  ```

- `[^...]` A character not in a list, for example, `[^abc]` means a character not in list `abc`

  Code example:

  ```
  strings <- c("abc", "ABC", "grey", "gray", "grrr")
  pattern <- "gr[ae]y"
  grep(pattern, strings, value=TRUE)
  ```

  ```
  Output: 'grey''gray'
  ```

- `[[:digit:]]` or `\\d` Digits `[0-9]`
- `[[:lower:]]` Lower characters `[0]`

- `\\w` Word characters. Word characters; `[A-z0-9_]`
- `\\W` Non-word characters
- `[[:punct:]]` Punctuation characters; !"#$%&'()*+,-./:;<=>?@[]^_`{|}~

Grouping

Often you need to match multiple characters together as a group such as two character patterns `re` in `grey` or `ra` in `gray`, you will need to use `( )` as grouping patterns. For example, `(re)` or `(ra)`

Code example:

```
strings <- c("abc", "ABC", "grey", "gray", "grrr")
pattern <- "(re)|(ra)"
grep(pattern, strings, value=TRUE)
```

```
Output: 'grey''gray'
```

Quantifiers

When you want to define repetitive character patterns such as `aaa` or `rrry`, you need to use quantifiers.

Next we will give some commonly used quantifiers along with `regexpr(pattern, string)` function to find the the index of first matching repetitive pattern in a string:

- `*` Matches at least 0 times
- `+` Matches at least 1 time

  Code example:

  ```
  strings <- c("GARY", "gABC", "grey", "ggryry", "gRyRyRyRy")
  # Matching character `ry` or `Ry` at least once, followed by character `y`
  pattern <- "+([r|R]y)"
  regexpr(pattern, strings)
  ```

  ```
  Output: -1 -1 -1 3 2
  ```

- `?` Matches at most 1 time; optional string
- `{n}` Matches exactly n times

  Code example:

  ```
  strings <- c("GARY", "gABC", "grey", "ggryry", "gRyryRyRy")
  # Matching character `ry` or `Ry` exact 4 times
  pattern <- "([r|R]y){4}"
  regexpr(pattern, strings)
  ```

  ```
  Output: -1 -1 -1 -1 2
  ```

- `{n,}` Matches at least n times
- `{,n}` Matches at most n times
- `{n,m}` Matches between n and m times

More details about Regular Expressions in R can be found in this helper page:

Regular Expressions as used in R

Next, complete the following string and regular expressions lab to have more coding practices.

Author(s)

Yan Luo

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-04-01	1.0	Yan	Created initial version