Basic Regular Expressions in R

Functions for Pattern Matching

Detect Patterns
- `grep(pattern, string)`
- `grep(pattern, string, value = TRUE)`
- `grepl(pattern, string)`

Locate Patterns
- `regexpr(pattern, string)`
- `str_locate_all(string, pattern)`

Extract Patterns
- `regmatches(string, regexp(pattern, pattern, string))`
- `regmatches(string, grexpr(pattern, pattern, string))`

Split a String using a Pattern
- `strsplit(string, pattern)`
- `str::str_split(string, pattern)`

Character Classes
- `[[digit]]` or `\D` Digits: [0-9]
- `[^\d]` Non-digits: [^0-9]
- `[^lower]` Lower-case letters: [a-z]
- `[^upper]` Upper-case letters: [A-Z]
- `[^alpha]` Alphabetic characters: [A-Za-z]
- `[^alnum]` Alphanumeric characters: [A-Za-z0-9]
- `\W` Word characters: [A-Za-z0-9_]
- `\S` Not space: [^[^space:]]
- `[^\S]` Space and tab
- `[^\s]` Space, tab, vertical tab, newline, form feed, carriage return
- `\$` Graphical char.
- `[^\$]` Not graphical char.
- `[^\$]` Not graphical char.
- `[^\$]` Not graphical char.
- `\[\a-zA-Z0-9\]$` Graphical char.
- `[^\$]` Not graphical char.

Special Metacharacters
- `\n` New line
- `\r` Carriage return
- `\t` Tab
- `\v` Vertical tab
- `\f` Form feed

Lookarounds and Conditionals
- `(?=)` Lookahead (requires PERL = TRUE), e.g. `(\b\s*\)?` position followed by `\s*` without including the whitespace
- `(?!)` Negative lookahead (PERL = TRUE), position NOT followed by pattern
- `(?<=)` Lookbehind (PERL = TRUE), e.g. `(\b\s*\)?` position following `\s*` including the whitespace
- `(?<!)` Negative lookbehind (PERL = TRUE), position NOT following pattern
- `(?i)` If-then-condition (PERL = TRUE): use lookheads, optional char. etc in if-clause
- `(?i)` If-then-condition (PERL = TRUE): use lookheads, optional char. etc in if-clause

Greedy Matching
By default R uses POSIX extended regular expressions when PERL = TRUE for base or by wrapping patterns with `perl()` for string.

Escaping Characters
Metacharacters ( . * + etc.) can be used as literal characters by escaping them. Characters can be escaped using `\` or by enclosing them in `\\`.

Case Conversions
Regular expressions can be made case insensitive by specifying `ignore.cases = TRUE`.