Lecture 18: Regular expressions in R

STAT598z: Intro. to computing for statistics

Vinayak Rao

Department of Statistics, Purdue University

We have seen the `print` function:

In [48]:
```
x <- 1
print(x)
y <- list('Hello', TRUE, c(1,2,3))
print(y)
```

```
[1] 1
[[1]]
[1] "Hello"

[[2]]
[1] TRUE

[[3]]
[1] 1 2 3
```

print is a *generic* function:

- looks at class of input and calls appropriate function

`print` and `cat`

`print` can only print its first term

In [56]:

```
print('Right now it is', date())
```

```
Warning message in print.default("Right now it is", date()):
“NAs introduced by coercion”

Error in print.default("Right now it is", date()): invalid 'digits
' argument
Traceback:

1. print("Right now it is", date())
2. print.default("Right now it is", date())
```

For this we need the cat (concatenate) function

In [3]:

```
cat('Right now it is', date())
```

```
Right now it is Mon Mar 27 21:39:50 2017
```

```
cat(..., file = '' , sep = ' ' , fill = FALSE, labels = NULL,
     append = FALSE)
```

...: Inputs that R concatenates to print

`sep`: What to append after each input (default is space)

`file`: Destination file (default is stdout)

Use `paste()` to store the concatenated output (a string)

In [58]:
```
cat(1:5)
```
```
1 2 3 4 5
```
In [59]:
```
cat(1:5,sep= ',' )
```
```
1,2,3,4,5
```
In [60]:
```
cat(1:5,sep= '\n' )
```
```
1
2
3
4
5
```
In [61]:
```
cat('[' ,1:5, ']' ,sep=(',' ))
```
```
[,1,2,3,4,5,]
```
In [62]:
```
cat('[',1:5, ']' ,sep=c('', rep(',' ,4), '' ))
```
```
[1,2,3,4,5]
```
In [63]:
```
cat('Hello','World','New para',sep='\n',file='new_file.txt')
```
In [64]:
```
my_cmd <- paste('[' ,1:5, ']',sep=c('', rep(',' ,4),''))
```

R needs a newline at end of string (not RStudio) Section 8.1.22 in *The R Inferno*, Patrick Burns:

- `print` outputs all characters in the string
- `cat` outputs what the string represents

Compare:

```
In [15]: print('Hello\n')
```

```
[1] "Hello\n"
```

```
In [16]: cat('Hello\n')
```

```
Hello
```

- '\' escapes the following character (indicating it is special)

What if we want to output ‘\n’ using cat ?

Escape \ with another \

```
In [65]: cat('Hello\\n')
```

```
Hello\n
```

Regular expression: representation of a collection of strings

Useful for searching and replacing patterns in strings

Composed of a grammar to build complicated patterns of strings

R has functions, which coupled with regular expressions allow powerful string manipulation

`E.g. grep, grepl, regexpr, gregexpr, sub, gsub`

Matching simple patterns

In [17]:
```
cities <- c('lafayette', 'indianapolis' , 'cincinnati')
grep('in', cities)
```

```
2 3
```

In [66]:
```
grepl('in', cities)
```

```
FALSE TRUE TRUE
```

Usage:
`grep(pattern, x, ignore.case = FALSE,  perl = FALSE, value = FALSE)`

In [67]:
```
grep('in',cities,value=TRUE) #Return values instead of indices
```

```
'indianapolis'  'cincinnati'
```

Where in each element did the match occur?

In [68]:
```
regexpr('in', cities)
```

```
-1  1  2
```

What if more than one match occured?

In [69]:
```
gregexpr('in', cities)
```

1. -1
2. 1
3. 2 5

What if we want to match

- any letter followed by ’n’?
- any vowel followed by ’n’?
- two letters followed by ’n’?
- any number of letters followed by ’n’?

Regular expressions!

- allow us to match much more complicated patterns
- build patterns from a simple vocabulary and grammar

R supports two flavors of regular expressions, we will always use perl (set option `perl = TRUE`)

'.' (period) represents any character except empty string '””'

```
In [71]: vec<-c('ct','at', 'cat', 'cart', 'dog', 'rat', 'carert', 'bet')
```

```
In [72]: grep('.at', vec, perl = TRUE)
```

```
3 6
```

```
In [73]: grep('..t', vec, perl = TRUE)
```

```
3 4 6 7 8
```

+ represents one or more occurrences

```
In [74]: grep( 'c.+t', vec, perl = TRUE)
```

```
3 4 7
```

* represents zero or more occurrences

```
In [75]: grep('c.*t', vec, perl = TRUE)
```

```
1 3 4 7
```

Group terms with parentheses '(' and ')'

```
In [76]: grep('c(.r)+t', vec, perl = TRUE)
```

```
4 7
```

```
In [78]: grep('c(.r)*t', vec, perl = TRUE)
```

```
1 4 7
```

‘.’ ‘,’ ‘+’ ‘*’ are all metacharacters

Other useful ones include:

- ^ and $ (start and end of line)

```
In [79]: grep('r.$', vec, perl = TRUE)
```

```
4 7
```

| (logical OR)

```
In [80]: grep('(c.t)|(c.rt)', vec, perl = TRUE)
```

```
3 4
```

`[` and `]` (create special character classes)

`[a-z]`: lowercase letters

`[a-zA-Z]`: any letter

`[0-9]`: any number

`[aeiou]`: any vowel

`[0-7ivx]`: any of 0 to 7, i, v, and x

Inside a character class `^` means "anything except the following characters". E.g.

`[^0-9]`: anything except a digit

What if we want to match metacharacters like `.` or +?

In [82]:
```
vec <- c('ct', 'cat', 'caat', 'caart', 'caaaat', 'caaraat',
         'c.t')
grep('c.t', vec, perl = TRUE) #Is this what we want?
```

```
2 7
```

Escape them with \

WARNING: a single \ doesn't work. Why?

In [83]:
```
cat('c\.t')
```

```
Error: '\.' is an unrecognized escape in character string starting
"'c\."
Traceback:
```

R thinks `\.` is a special character like `\n`.

Use two \'s

```
In [84]: cat('c\\.t')
```

```
c\.t
```

```
In [85]: grep('c\.t', vec, perl = TRUE)
```

```
Error: '\.' is an unrecognized escape in character string starting
"'c\."
Traceback:
```

```
In [86]: grep('c\\.t', vec, perl = TRUE)
```

7

To match a \, our pattern must represent \\

```
In [88]: my_var <- '\n'
         grep('\\n', my_var)
```

```
1
```

```
In [90]: my_var <- ('\\')
         grep('\\\\', my_var)
```

```
1
```

Search and replace

The `sub` function allows search and replacement:

In [91]:
```
vec <-c('ct','cat','caat','caart','caaaat','caaraat','c.t')
sub('a+', 'a', vec, perl = TRUE)
```

```
'ct'  'cat'  'cat'  'cart'  'cat'  'caraat'  'c.t'
```

`sub` replaces only first match, `gsub` replaces all

Use backreferences \1, \2 etc to refer to first, second group etc

In [92]:
```
gsub('(a+)r(a+)', 'b\\1brc\\2c', vec, perl = TRUE)
```

```
'ct'  'cat'  'caat'  'caart'  'caaaat'  'cbaabrcaact'  'c.t'
```

Use \U, \L, \E to make following backreferences upper or lower case or leave unchanged respectively

```
In [45]: gsub('(a+)r(a+)', '\\U\\1r\\2', vec, perl = TRUE)
```

```
'ct'  'cat'  'caat'  'caart'  'caaaat'  'cAArAAt'  'c.t'
```

```
In [47]: gsub('(a+)r(a+)', '\\U\\1r\\E\\2', vec, perl = TRUE)
```

```
'ct'  'cat'  'caat'  'caart'  'caaaat'  'cAAraat'  'c.t'
```