Les expressions régulières sous R

Text Mining

Ricco Rakotomalala

1. Expression régulières – Principes et utilisation
2. Syntaxe des expressions sous R (POSIX étendu)
3. Exemples
4. Bibliographie

Principe et utilisation

EXPRESSIONS RÉGULIÈRES

Définition : Une expression rationnelle ou expression normale ou expression régulière, est, en informatique, une chaîne de caractères, que l'on appelle parfois un motif, qui décrit, selon une syntaxe précise, un ensemble de chaînes de caractères possibles (Wikipédia).

Ex. Le motif « t.t. » peut couvrir les chaînes de caractères « tata », « toto », « tBtU », « t1tw », etc… Il signifie : lettre « t » en minuscule suivie d'un caractère quelconque, puis d'un autre « t », encore suivi d'un autre caractère.

Intérêt : Dans l'analyse des données textuelles (text mining), une expression régulière permet de recherche des documents correspondant à un motif requête, d'effectuer des vérifications (ex. adresse mail valable), des substitutions (ex. harmoniser les formats de date), des suppressions, des découpages, etc. C'est également un outil privilégié pour l'analyse des fichiers logs.

Il y a principalement deux normes :

- PCRE (Perl-Compatible Regular Expressions). Associé au langage de programmation PERL, elle est notamment exploitée par la bibliothèque re de Python.
- POSIX étendu, fruit d'un effort de normalisation, elle est censée être plus simple, mais est en revanche plus lente. L'intérêt pour nous est que R s'appuie sur cette norme (sauf à indiquer explicitement l'option *perl=TRUE*).

Les fonctions de R qui exploitent les expressions régulières sont (cf. 1, 2) :

- Localisation de texte : grep, grepl, regexpr, gregexpr
- Substitution de texte : sub, gsub
- Découpage de texte : strsplit

Un exemple simple avec grep et gsub

```
#un vecteur de chaînes de caractères
textes <- c("toto","gota","tatane","hata","tut")

print(grep("t.t.",textes)) # n°1 et 3

print(grep("t.t.?",textes)) # ? Pour dire que le dernier caractère est optionnel
# correspondent : n°1, 3 et 5 aussi

resultat <- gsub("t.t.","bingo",x=textes)
print(resultat)
# « bingo », « gota », « bingone », « hata », « tut »
```

R ne remplace que la partie correspondant au motif.

Quelques notions importantes (Voir zytrax, un outil de test est dispo)

- **Littéral** (*literal*) : tout caractère que l'on peut utiliser dans une expression (ex. « t » est un caractère littéral ; « to » est une chaîne littérale)
- **Métacaractère** (*metacharacter*) : un caractère spécial symbolisant un ensemble de caractères (ex. « **.** » signifie tout caractère ; « **?** » signifie : le caractère précédent est optionnel)
- **Chaînes cibles** (*target string*) : chaîne de caractère à laquelle s'applique la recherche (ex. les chaînes dans le vecteur **textes** page précédente)
- **Expression régulière** (*search expression, pattern*) : le motif requête qui sert à la recherche (ex. « **t.t.?** » dans les exemples précédents)
- **Séquence d'échappement** (*escape sequence*) : **(1)** « \ » permet d'indiquer que le métacaractère qui suit doit être considéré comme un littéral (ex. pour « **t.t.\?** », « toto » ne correspond pas, « toto? » oui ; Remarque : sous R spécifiquement, il faut faire plutôt « t.t.\\? ») ; **(2)** « \ » permet aussi de désigner des caractères spéciaux (ex. « **\t** » signifie tabulation, « **\n** » désigne le saut de ligne, etc.)

Syntaxe des métacaractères

SYNTAXE SOUS R

Syntaxe - Voir http://informatique-mia.inra.fr/r4ciam/node/148 et http://www.zytrax.com/tech/web/regex.htm

Métacaractère	Signification
[...]	un des caractères indiqués entre les crochets. Par exemple, « t[aeiouy]t[aeiouy] » autorise « toto » mais pas « tbtb » c.-à-d. « t » doit être suivi d'une voyelle
[^...]	tous les caractères sauf ceux indiqués après le ^. Par exemple, « t[^aeiouy]t[^aeiouy] » ne veut pas que « t » soit suivi d'une voyelle
[x-y]	les caractères compris entre x à y inclus. Par exemple, « t[a-z]t[a-z] » veut pas « t » soit suivi d'un caractère compris entre « a » et « z » en minuscule
[:alnum:]	équivalent à a-zA-Z0-9 avec en plus les caractères spéciaux que l'on retrouve suivant les langues utilisées comme les éèùçà. Par exemple, A,B,C, 0, 1, etc.
[:alpha:]	équivalent à a-zA-Z avec en plus les caractères spéciaux que l'on retrouve suivant les langues utilisées
[:digit:]	équivalent à 0-9. Par exemple : « t[[:digit:]]t[[:digit:]] » indique que « t » doit être suivi d'un nombre (attention aux crochets)
[:lower:]	équivalent à a-z avec en plus les caractères spéciaux que l'on retrouve suivant les langues utilisées
[:upper:]	équivalent à A-Z avec en plus les caractères spéciaux que l'on retrouve suivant les langues utilisées comme les ÂÛÔ...
[:xdigit:]	équivalent à 0-9a-fA-F
[:graph:]	tout caractère graphique
[:print:]	tout caractère affichable
[:punct:]	tout caractère de ponctuation
[:blank:]	espace, tabulation
[:space:]	espace, tabulation, nouvelle ligne, retour chariot
[:cntrl:]	tout caractère de contrôle

Syntaxe (suite 1)

Métacaractère	Signification
.	n'importe quel caractère.
^	en dehors du crochet, indique le début du texte. Par exemple, ^WIN localise le "WIN" dans "Windows", mais non le "WIN" dans "MS Windows".
$	en fin de motif, indique la fin du texte.
+	1 ou plusieurs occurrences du motif qui précède le +.
*	0 ou plusieurs occurrences du motif qui précède le *.
?	0 ou 1 occurrence du motif précédant du ?.
{n}	n occurrences du motif qui précède. Par exemple, [A-C]{2} donne AA, AB, AC, BB, BA, BC, etc.
{x,}	x ou plus occurrences du motif qui précède.
{,y}	y occurrences au plus du motif qui précède.
{x,y}	x à y occurrences du motif qui précède.
(...)	définition d'une sous-expression. Par exemple, pa(pa)? donne pa, papa.
\N	Référence en arrière. N est un chiffre pouvant aller de 1 à 9 pour désigner les sous-expressions précédents. \1 correspond à la dernière sous-expression, \2 l'avant dernière expression. Par exemple, (pa|ba)\1 donne papa, baba
|	alternative. Par exemple, (pa|ba) donne pa, ba

Syntaxe (suite 2)

Métacaractère	Signification
\b	expression Perl, indique le début ou la fin d'un mot
\B	expression Perl, indique ni le début ni la fin d'un mot
\d	expression Perl, équivalent à [0-9] dans POSIX
\D	expression Perl, équivalent à [^0-9] dans POSIX
\n	expression Perl, indique une nouvelle ligne
\r	expression Perl, indique le retour chariot
\t	expression Perl, indique la tabulation
\s	expression Perl, équivalent à [:space:] dans POSIX
\S	expression Perl, équivalent à [^[:space:]] dans POSIX
\w	expression Perl, équivalent à [:alnum] dans POSIX
\W	expression Perl, équivalent à [^[:alnum:]] dans POSIX

UN EXEMPLE SOUS R

Recherche dans un ensemble de SMS (Disponible en ligne - UCI)

N°	Message
1	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
2	Ok lar... Joking wif u oni...
3	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
4	U dun say so early hor... U c already then say...
5	Nah I don't think he goes to usf, he lives around here though
6	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, £1.50 to rcv
7	Even my brother is not like to speak with me. They treat me like aids patent.
8	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
9	WINNER!! As a valued network customer you have been selected to receivea £900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
10	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL

Importation des données

```
library(xlsx)
sms <- read.xlsx("sms regular expression.xlsx",sheetIndex=1)
print(sms)
```

Exemples

```
#contenant "crazy" en minuscule  ==> 1
print(grep("crazy",sms$message))

#contenant une référence http ==> 10
print(grep("http",sms$message))

#contenant le symbole £ ==> 6 et 9
print(grep("£",sms$message))

#contenant le terme "call" min. ou maj. ==> 8 et 9
print(grep("call",sms$message,ignore.case=TRUE))

#contenant ? ou ! ==> 6, 9 et 10
print(grep("[\\?\\!]",sms$message))

#contenant au moins 1 chiffre ==> 3, 6, 8 et 9
print(grep("[[:digit:]]",sms$message))

#contenant un numéro de tel. ==> 3 et 9
print(grep("[[:digit:]]{9}",sms$message))

#contenant xxx en min ou maj ==> 6 et 10
print(grep("[x]{3}",sms$message,ignore.case=TRUE))

#commençant par xxx min ou maj ==> 10
print(grep("^[x]{3}",sms$message,ignore.case=TRUE))

#contenant 3 points de suspensions ==> 1, 2 et 4
print(grep("[\\.]{3}",sms$message))

#contenant une référence à un des mois du printemps ==> 3
print(grep("(march|april|may|june)",sms$message,ignore.case=TRUE))
```

Sites web

Zyntrax-Info, « Regular Expressions – User Guide », http://www.zytrax.com/tech/web/regex.htm

Excellent site, avec de surcroit un outil en ligne pour s'exercer

Jwang – « Utilisation des expressions régulières sous R », http://informatique-mia.inra.fr/r4ciam/node/148

Excellent travail de simplification de la référence précédente, il m'a beaucoup inspiré, au moins dans sa partie description des métacaractères

R Manual, « Regular Expressions as used in R », https://stat.ethz.ch/R-manual/R-devel/library/base/html/regex.html

Documentation de R, elle fait référence mais est un peu ardue quand même