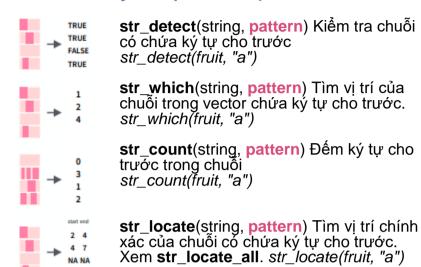
Xử lý ký tự với stringr: : CHEAT SHEET

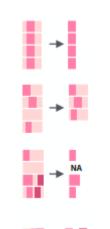
Thư viện stringr cung cấp các công cụ hữu ích để làm việc với kiểu dữ liệu định dạng ký tự.

stringr

Tìm kiếm ký tự phù hợp



Trích xuất ký tự



str_sub(string, start = 1L, end = -1L) Trích xuất ký tự theo vị trí. str_sub(fruit, 1, 3); str_sub(fruit, -2)

str_subset(string, pattern) Trả ra chuỗi trong vector có chứa ký tự cho trước . str subset(fruit, "b")

str_extract(string, pattern) Trả ra ký tự đầu tiên phù hợp với điều kiện lọc dưới dạng vector. Xem thêm str_extract_all. str_extract(fruit, "[aeiou]")

str_match(string, **pattern**) Trả ra chuỗi ký tự đầu tiên phù hợp với điều kiện lọc. Xem thêm **str_match_all**. str match(sentences, "(a|the) ([^]+)")

Quản lý độ dài của chuỗi



str_length(string) Số lượng ký tự có trong
1 chuỗi.
str_length(fruit)

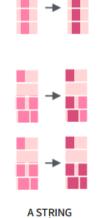
str_pad(string, width, side = c("left", "right", "both"), pad = " ") Chèn thêm ký tự để chuổi có độ dài cố định. str_pad(fruit, 17)

str_trunc(string, width, side = c("right",
"left", "center"), ellipsis = "...") Tinh lược ký
tự trong chuỗi
str_trunc(fruit, 3)

→

str_trim(string, side = c("both", "left",
"right")) Cắt ký tự trắng bắt đầu hoặc kết
thúc của chuỗi
str trim(fruit)

Biến đổi chuỗi



a string

a string

A STRING

a string

▼A String

str_sub() <- value. Thay chuỗi con bằng giá trị mới. str_sub(fruit, 1, 3) <- "str"

str_replace(string, **pattern**, replacement) Thay thế kỹ tự đầu tiên trong chuỗi thỏa mãn điều kiên. str_replace(fruit, "a", "-")

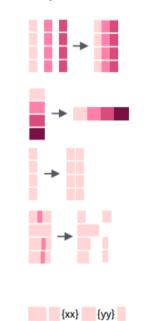
str_replace_all(string, pattern, replacement) Thay the tat ca ky tự trong chuỗi thỏa mãn điều kiện. str_replace_all(fruit, "a", "-")

str_to_lower(string, locale = "en")¹ Chuyên sang chữ thường. str_to_lower(sentences)

str_to_upper(string, locale = "en")¹ Chuyên sang chữ hoa. str_to_upper(sentences)

str_to_title(string, locale = "en")¹ Chuyển
sang định dạng tên.
str_to_title(sentences)

Gôp và tách chuỗi



str_c(..., sep = "", collapse = NULL) Gộp nhiều ký tự thành một str_c(letters, LETTERS)

str_c(..., sep = "", collapse = NULL) Gộp
các ký tự/chuỗi trong một véc-tơ thành một
chuỗi duy nhất
str c(letters, collapse = "")

str_dup(string, times) Lặp lại chuỗi. str_dup(fruit, times = 2)

str_split_fixed(string, **pattern**, n) Tách một véc-tơ chuỗi với điều kiện và số lượng chuỗi con xác định trước. str_split_fixed(fruit, " ", n=2)

glue::glue(..., .sep = "", .envir = parent.frame(), .open = "{", .close = "}") Tạo chuỗi từ chuỗi và sử dụng {} để thực hiện câu lệnh (nếu có) . glue::glue("Pi is {pi}")

glue::glue_data(.x, ..., .sep = "", .envir = parent.frame(), .open = "{", .close = "}") Sử dụng dataframe để tạo chuỗi mới và dung {} để thực hiện câu lệnh. glue::glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")

Sắp xếp ký tự



str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)¹ Trả ra kết quả là một véc-tơ thứ tự ký tự sau khi sắp xếp. *x[str_order(x)]*



str_sort(x, decreasing = FALSE, na_last =
TRUE, locale = "en", numeric = FALSE, ...)¹
Săp xêp lại vị trí ký tự.
str_sort(x)

Các hàm hữu ích khác

apple banana pear **str_conv**(string, encoding) Thay đổi encoding của chuỗi. str_conv(fruit,"ISO-8859-1")

str_view(string, **pattern**, match = NA) Xem kết quả HTML với ký tự đầu điên thỏa mãn điều kiện lọc. str_view(fruit, "[aeiou]")

apple banana pear **str_view_all**(string, **pattern**, match = NA) Xem kết quả HTML với tất cả ký tự thỏa mãn điều kiện lọc str_view_all(fruit, "[aeiou]")

str_wrap(string, width = 80, indent = 0, exdent = 0) Co ngắn độ dài của chuỗi. str_wrap(sentences, 20)

¹ Xem thêm bit.ly/ISO639-1.



Những điều cần biết

Điều kiện lọc trong stringr sẽ được diễn dịch thành biêu thức chính quy (regular expressions – viết tắt là regex).

Trong R, ta viết biết thức chính quy (regex) dưới dạng chuỗi ký tự trong dấu ngoặc kép hoặc ngoặc đơn.

Không phải tất cả các ký tự đều có thể thể hiện trực tiếp trong R dưới dạng chuỗi. Một số trường hợp phải thể hiện dưới dạng ký tự đặc biệt với ý nghĩa riệng.

> Ký tự Ý nghĩa // \n Dòng mới

Thực hiện câu lệnh ?"" để xem thêm.

Do đó, bất cứ khi nào ký tự \ xuất hiện trong regex, ta phải viết dưới dạng hai dấu gạch số \\ trong chuỗi thể hiện của regex.

Sử dụng hàm **writeLines**() để xem R thực hiện chuỗi với các ký tự đặc biệt như thế nào

writeLines("\\.")

writeLines("\\ is a backslash") #\is a backslash

Giải thích

Điều kiện lọc trong stringr được diễn dịch thành biểu thức chính quy (regex). Để thay đổi chế độ mặc định, ta có thể sử dụng các hàm sau: regex(pattern, ignore_case = FALSE, multiline = FALSE, comments = FALSE, dotall = FALSE, ...) Điều chỉnh biểu thức chính quy, cho phép bỏ qua các điều kiện đặc biệt, cho phép R ghi chú bên trong một biểu thức chính str_detect("I", regex("i", TRUE))

fixed() Khớp các ký tự dưới dạng bytes, tuy nhiên có thể sẽ bị sót các ký tự có khả năng biểu diễn dưới nhiều hình thức khác nhau. str_detect("\u0130", fixed("i"))

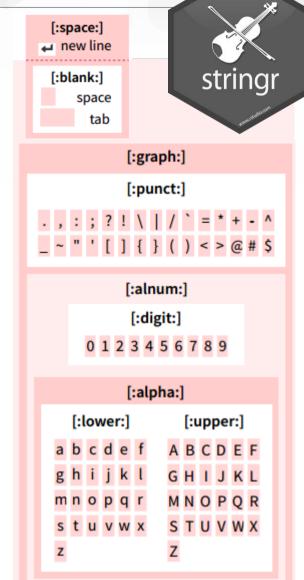
coll() Khớp các ký tự dưới dạng bytes nhưng sử dụng locale để xác định chính xác ký tự. str_detect("\u0130", coll("i", TRUE, locale =

boundary() Khớp với các ký tự phân biệt giữa câu, từ, đồng mới... str_split(sentences, boundary("word"))

Biểu thức chính quy (regex) - Biểu thức chính quy hay regex là phương pháp chính xác để mô tả điều kiện lọc trong chuỗi

see <- function(rx) str_view_all("abc ABC 123\t.!?\\(){}\n", rx)

Ký tự	Regex	Ý nghĩa	Ví dụ	
	a (etc.)	a (etc.)	see("a")	abc ABC 123 .!?\(){}
\\.	\.		see("\\.")	abc ABC 123 .!?\(){}
\\!	\!	1	see("\\!")	abc ABC 123 .!?\(){}
\\?	\?	?	see("\\?")	abc ABC 123 .!?\(){}
////	//	\	see("\\\\")	abc ABC 123 .!?\(){}
\\(\(ì	see("\\(")	abc ABC 123 .!?\ <mark>(</mark>){}
//)	\	ì	see("\\)")	abc ABC 123 .!?\(<mark>)</mark> {}
\\{	\{ 	1	see("\\{")	abc ABC 123 .!?\(){}
\\}	\ }	1	see("\\}")	abc ABC 123 .!?\(){}
\\ n	\n	Xuống dòng	see("\\n")	abc ABC 123 .!?\(){}
\\t	\t	Tab	see("\\t")	abc ABC 123 .!?\(){}
\\s	\s	Khoảng trắng	see("\\s")	abc ABC 123 .!?\(){}
\/d	\d	Số (\D với điều kiện không phải số	see("\\d")	abc ABC 123 .!?\(){}
\\w	\w	Chữ (\W với điều kiện không phải chữ)	see("\\w")	abc ABC 123 .!?\(){}
\\b	\b	Đường biên của một từ	see("\\b")	abc ABC 123 .!?\(){}
	[:digit:]	Số	see("[:digit:]")	abc ABC 123 .!?\(){}
	[:alpha:]	Chữ	see("[:alpha:]")	abc ABC 123 .!?\(){}
	[:lower:]	Viết thường	see("[:lower:]")	abc ABC 123 .!?\(){}
	[:upper:]	Viết hoa	see("[:upper:]")	abc ABC 123 .!?\(){}
	[:alnum:]	Số hoặc chữ	see("[:alnum:]")	abc ABC 123 .!?\(){}
	[:punct:]	Dấu	see("[:punct:]")	abc ABC 123 .!?\(){}
	[:graph:] 1	Chữ, số hoặc dấu	see("[:graph:]")	abc ABC 123 .!?\(){}
	[:space:]	Dấu cách	see("[:space:]")	abc ABC 123 .!?\(){}
	[:blank:]	Khoảng trắng	see("[:blank:]")	abc ABC 123 .!?\(){}
		Tất cả ký tự trừ dòng mới	see(".")	abc ABC 123 .!?\(){}



Điều kiên

alt <- function(rx) str view all("abcde", rx)

regexp	Ý nghĩa	Ví dụ	
abld	Ноặс	alt("ab d")	abcde
abe	Một trong	alt("[abe]")	abcde
[^abe]	Tất cả trừ	alt("[^abe]")	ab <mark>cd</mark> e
[a-c]	Trong khoảng	alt("[a-c]")	abcde

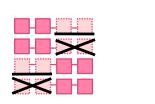
Điều kiện chặn

anchor <- function(rx) str_view_all("aaa", rx)

	regexp	Ý nghĩa	Ví dụ	
Н	^ a	Bắt đầu chuỗi	anchor("^a")	aaa
	a\$	Kết thúc chuỗi	anchor("a\$")	aaa

Tìm mở rộng

look <- function(rx) str view all("bacad", rx)



regexp	Ý nghĩa	Ví dụ	
a(?=c)	Tìm a theo sau là c	look("a(?=c)")	bacad
a(?!c)	Tìm a với ký tự đằng sau không phải là c Tìm a với ký tự đằng	look("a(?!c)")	bacad
(?<=b)a	Tìm a với ký tự đăng trước là b	look("(?<=b)a")	bacad
(? b)a</th <td>Tìm a với ký tự đằng trước không phải b</td> <td>look("(?<!--b)a")</td--><td>bacad</td></td>	Tìm a với ký tự đằng trước không phải b	look("(? b)a")</td <td>bacad</td>	bacad

Đếm số lương

2 - ...-n

regexp

{n} {n, } {n, m}

quant <- function(rx) str view all(".a.aa.aaa", rx) example matches

Không hoặc một quant("a?") .a.aa.aaa Không hoặc nhiều quant("a*") a.aa.aaa Môt hoặc nhiều quant("a+") .a.aa.aaa Chính xác n quant("a{2}") .a.aa.aaa **n** hoặc nhiều hơn quant("a{2,}") .a.aa.aaa Giữa n và m quant("a{2,4}") .a.aa.aaa

Nhóm

ref <- function(rx) str_view_all("abbaab", rx)

Sử dụng dấu ngoặc để tạo ngóm

Điều kiện Ví dụ sets precedence alt("(ab|d)e") abcde

Sử dụng dấu gạch sổ để lặp lại nhóm trong ngoặc đơn. Số thứ tự chính là thứ tự của nhóm.

Ký tự	regex	Diễn giải	Ví dụ
(câu lệnh)	(Ý nghĩa)	(Điều kiện lọc)	(Kết quả tương tự như ref("abba"))
\\1	\1 (etc.)	Nhóm đầu tiên.	ref("(a)(b)\\2\\1") abbaab

