Workshop: Working with Strings in R

Simon Ress | Ruhr-Universität Bochum

October 18, 2021

Content

Base R

The base R functions for dealing with strings

- *substr() / substring()*: Extract or replace substrings in a character vector by indicies.
- *strsplit()*: Split the elements of a character vector x into substrings according to a given regular expression.
- *paste()*: Concatenate n number of strings.
- *nchar()*: Returns a vector of the number of characters of x.

substr() / substring()

substr()/substring():

Extract or replace substrings in a character vector

```
num <- "12345678"
substr(num, 4, 5)
```

```
## [1] "45"
```

```
substring(num, 1:3, 7)
```

```
## [1] "1234567" "234567"  "34567"
```

strsplit()

strsplit():

Split the elements of a character vector x into substrings according to a given character

```
str = "Splitting sentence into words"
strsplit(str, " ")
```

```
## [[1]]
## [1] "Splitting" "sentence"  "into"      "words"
```

paste()

paste():

Concatenate n number of strings

```
paste("Count number", "of characters")
```

```
## [1] "Count number of characters"
```

nchar()

nchar():

Returns a vector of the number of characters of x

```
nchar("Count number of characters")
```

```
## [1] 26
```

The base R functions for dealing regular expressions

- *grep() / grepl()*: Search for matches of a regular expression/pattern in a character vector ans return the indices/a logical vector.
- *regexpr() / gregexpr()*: Search a character vector for regular expression matches and return the indices of the string where the match begins and the length of the match.
- *sub() / gsub()*: Search the first/all character vector/s for regular expression matches and replace that match with another string.

grep() / grepl()

grep():

Index of vector which matches regex

```
grep("b+", c("abc", "bda", "cca a", "abd"))
```

```
## [1] 1 2 4
```

grepl():

Logical if vector matches regex

```
grepl("b+", c("abc", "bda", "cca a", "abd"))
```

```
## [1]  TRUE  TRUE FALSE  TRUE
```

regexpr()

regexpr():

Search a character vector for regular expression matches and return the indices of the string where the match begins and the length of the match

```
str = "Line 129: O that this too too solid flesh would melt
regexpr("1",str)
```

```
## [1] 6
## attr(,"match.length")
## [1] 1
## attr(,"index.type")
## [1] "chars"
## attr(,"useBytes")
## [1] TRUE
```

sub() / gsub()

sub():

Search **first* match of an regular expression and replace it

```
x <- "<dd>Found on January 1, 2007</dd>"
sub("<dd>[F|f]ound on |</dd>", "", x)
```

```
## [1] "January 1, 2007</dd>"
```

gsub:

Search **all** matches of an regular expression and replace it

```
x <- "<dd>Found on January 1, 2007</dd>"
gsub("<dd>[F|f]ound on |</dd>", "", x)
```

```
## [1] "January 1, 2007"
```

Regular Expressions

Match Characters

MATCH CHARACTERS

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

string (type this)	regexp (to mean this)	matches (which matches this)	example	
	a (etc.)	a (etc.)	see("a")	abc ABC 123 .!?\(){}
\\.	**\.**	.	see("\\.")	abc ABC 123 .!?\(){}
\\!	**\!**	!	see("\\!")	abc ABC 123 .!?\(){}
\\?	**\?**	?	see("\\?")	abc ABC 123 .!?\(){}
\\\\	****	\	see("\\\\")	abc ABC 123 .!?\(){}
\\(	**\(**	(	see("\\(")	abc ABC 123 .!?\(){}
\\)	**\)**	)	see("\\)")	abc ABC 123 .!?\(){}
\\{	**\{**	{	see("\\{")	abc ABC 123 .!?\(){}
\\}	**\}**	}	see("\\}")	abc ABC 123 .!?\(){}
\\n	**\n**	new line (return)	see("\\n")	abc ABC 123 .!?\(){}
\\t	**\t**	tab	see("\\t")	abc ABC 123 .!?\(){}
\\s	**\s**	any whitespace (**\S** *for non-whitespaces*)	see("\\s")	abc ABC 123 .!?\(){}
\\d	**\d**	any digit (**\D** *for non-digits*)	see("\\d")	abc ABC 123 .!?\(){}
\\w	**\w**	any word character (**\W** *for non-word chars*)	see("\\w")	abc ABC 123 .!?\(){}
\\b	***\b***	*word boundaries*	see("\\b")	abc ABC 123 .!?\(){}
	[:digit:][1]	digits	see("[:digit:]")	abc ABC 123 .!?\(){}
	[:alpha:][1]	letters	see("[:alpha:]")	abc ABC 123 .!?\(){}
	[:lower:][1]	lowercase letters	see("[:lower:]")	abc ABC 123 .!?\(){}
	[:upper:][1]	uppercase letters	see("[:upper:]")	abc ABC 123 .!?\(){}
	[:alnum:][1]	letters and numbers	see("[:alnum:]")	abc ABC 123 .!?\(){}
	[:punct:][1]	punctuation	see("[:punct:]")	abc ABC 123 .!?\(){}
	[:graph:][1]	letters, numbers, and punctuation	see("[:graph:]")	abc ABC 123 .!?\(){}
	[:space:][1]	space characters (i.e. \s)	see("[:space:]")	abc ABC 123 .!?\(){}
	[:blank:][1]	space and tab (but not new line)	see("[:blank:]")	abc ABC 123 .!?\(){}
	.	every character except a new line	see(".")	abc ABC 123 .!?\(){}

[1] Many base R functions require classes to be wrapped in a second set of [ ], e.g. **[[:digit:]]**

Alternates

ALTERNATES

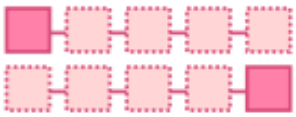
```
alt <- function(rx) str_view_all("abcde", rx)
```

regexp	matches	example	
ab|d	or	alt("ab|d")	abcde
[abe]	one of	alt("[abe]")	abcde
[^abe]	anything but	alt("[^abe]")	abcde
[a-c]	range	alt("[a-c]")	abcde

Anchors

ANCHORS

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

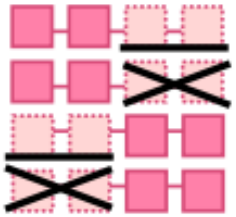


regexp	matches	example	
^a	start of string	anchor("^a")	aaa
a$	end of string	anchor("a$")	aaa

Look Arounds

LOOK AROUNDS

look <- function(rx) str_view_all("bacad", rx)

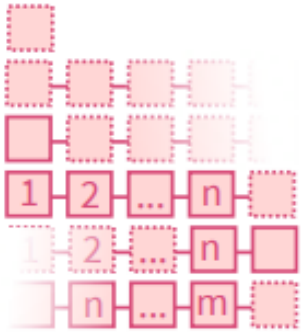


regexp	matches	example	
a(?=c)	followed by	look("a(?=c)")	bacad
a(?!c)	not followed by	look("a(?!c)")	bacad
(?<=b)a	preceded by	look("(?<=b)a")	bacad
(?<!b)a	not preceded by	look("(?<!b)a")	bacad

Quantifiers

QUANTIFIERS

```
quant <- function(rx) str_view_all(".a.aa.aaa", rx)
```



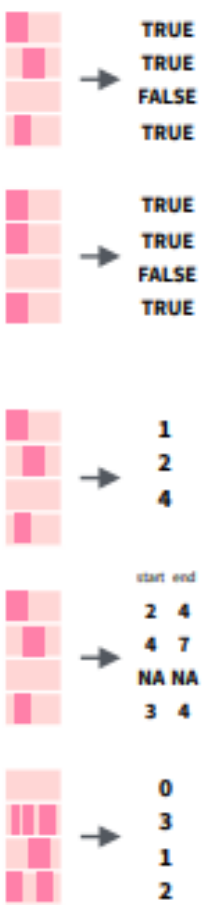
regexp	matches	example	
a**?**	zero or one	quant("a?")	.a.aa.aaa
a*****	zero or more	quant("a*")	.a.aa.aaa
a**+**	one or more	quant("a+")	.a.aa.aaa
a**{n}**	exactly **n**	quant("a{2}")	.a.aa.aaa
a**{n, }**	**n** or more	quant("a{2,}")	.a.aa.aaa
a**{n, m}**	between **n** and **m**	quant("a{2,4}")	.a.aa.aaa

Package: Stringr

String basics

The stringr package provides a series of functions implementing much of the regular expression functionality in R but with a more consistent and rationalized interface.

Detect Matches



str_detect(string, **pattern**, negate = FALSE**)**
Detect the presence of a pattern match in a string. Also **str_like()**. str_detect(fruit, "a")

str_starts(string, **pattern**, negate = FALSE**)**
Detect the presence of a pattern match at the beginning of a string. Also **str_ends()**.
str_starts(fruit, "a")

str_which(string, **pattern**, negate = FALSE**)**
Find the indexes of strings that contain a pattern match. str_which(fruit, "a")

str_locate(string, **pattern**) Locate the positions of pattern matches in a string. Also **str_locate_all()**. str_locate(fruit, "a")

str_count(string, **pattern**) Count the number of matches in a string. str_count(fruit, "a")

Subset Strings

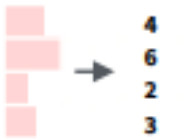
str_sub(string, start = 1L, end = -1L**)** Extract substrings from a character vector.
str_sub(fruit, 1, 3); str_sub(fruit, -2)

str_subset(string, **pattern**, negate = FALSE**)** Return only the strings that contain a pattern match. str_subset(fruit, "p")

str_extract(string, **pattern**) Return the first pattern match found in each string, as a vector. Also **str_extract_all()** to return every pattern match. str_extract(fruit, "[aeiou]")

str_match(string, **pattern**) Return the first pattern match found in each string, as a matrix with a column for each ( ) group in pattern. Also **str_match_all()**.
str_match(sentences, "(a|the) ([^ +])")

Manage Lengths



str_length(string**)** The width of strings (i.e. number of code points, which generally equals the number of characters). str_length(fruit)

str_pad(string, width, side = c("left", "right", "both"), pad = " "**)** Pad strings to constant width. str_pad(fruit, 17)

str_trunc(string, width, side = c("right", "left", "center"), ellipsis = "..."**)** Truncate the width of strings, replacing content with ellipsis. str_trunc(sentences, 6)

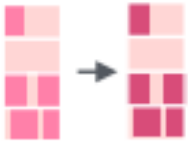
str_trim(string, side = c("both", "left", "right")**)** Trim whitespace from the start and/or end of a string. str_trim(str_pad(fruit, 17))

str_squish(string**)** Trim whitespace from each end and collapse multiple spaces into single spaces. str_squish(str_pad(fruit, 17, "both"))

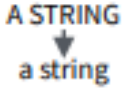
Mutate Strings

str_sub() <- value. Replace substrings by identifying the substrings with str_sub() and assigning into the results.
str_sub(fruit, 1, 3) <- "str"

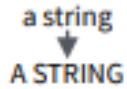
str_replace(string, **pattern**, replacement**)**
Replace the first matched pattern in each string. Also **str_remove()**.
str_replace(fruit, "p", "-")



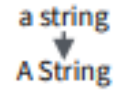
str_replace_all(string, **pattern**, replacement**)**
Replace all matched patterns in each string. Also **str_remove_all()**.
str_replace_all(fruit, "p", "-")



str_to_lower(string, locale = "en"**)**[1]
Convert strings to lower case.
str_to_lower(sentences)



str_to_upper(string, locale = "en"**)**[1]
Convert strings to upper case.
str_to_upper(sentences)



str_to_title(string, locale = "en"**)**[1] Convert strings to title case. Also **str_to_sentence()**.
str_to_title(sentences)

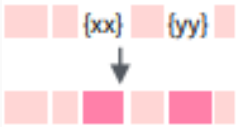
Extra: Join and Split

str_c(..., sep = "", collapse = NULL**)** Join multiple strings into a single string.
str_c(letters, LETTERS)

str_flatten(string, collapse = ""**)** Combines into a single string, separated by collapse.
str_flatten(fruit, ", ")

str_dup(string, times**)** Repeat strings times times. Also **str_unique()** to remove duplicates.
str_dup(fruit, times = 2)

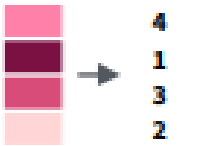
str_split_fixed(string, **pattern**, n**)** Split a vector of strings into a matrix of substrings (splitting at occurrences of a pattern match). Also **str_split()** to return a list of substrings and **str_split_n()** to return the nth substring.
str_split_fixed(sentences, " ", n=3)



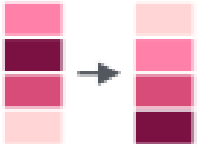
str_glue(..., .sep = "", .envir = parent.frame()**)** Create a string from strings and {expressions} to evaluate. str_glue("Pi is {pi}")

str_glue_data(.x, ..., .sep = "", .envir = parent.frame(), .na = "NA"**)** Use a data frame, list, or environment to create a string from strings and {expressions} to evaluate.
str_glue_data(mtcars, "{rownames(mtcars)} has {hp} hp")

Extra: Order Strings



str_order(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1]
Return the vector of indexes that sorts a character vector. fruit[str_order(fruit)]



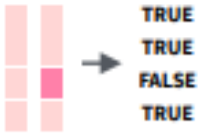
str_sort(x, decreasing = FALSE, na_last = TRUE, locale = "en", numeric = FALSE, ...)[1]
Sort a character vector. str_sort(fruit)

Extra: Helpers

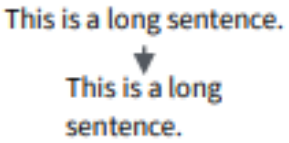
str_conv(string, encoding**)** Override the encoding of a string. str_conv(fruit,"ISO-8859-1")

appl<e>
banana
p<e>ar

str_view_all(string, **pattern**, match = NA**)** View HTML rendering of all regex matches. Also **str_view()** to see only the first match. str_view_all(sentences, "[aeiou]")



str_equal(x, y, locale = "en", ignore_case = FALSE, ...**)**[1] Determine if two strings are equivalent. str_equal(c("a", "b"), c("a", "c"))



str_wrap(string, width = 80, indent = 0, exdent = 0**)** Wrap strings into nicely formatted paragraphs. str_wrap(sentences, 20)

Helpful sources

Helpful sources

- Stringr: Overview
- Stringr: Introduction
- Stringr: Cheatsheet
- Stringr: Reference manual
- Base R String-functions vs Stringr
- Working with strings in R
- Regular expressions
- Primary R functions for dealing with regular expressions

References

All graphics are taken from String manipulaton with stringr Cheatsheet