第九章：字符串处理

王敏杰

2020 年 7 月 30 日

四川师范大学

提问

问题

这是一份关于地址信息的数据

No	address
1	Sichuan Univ, Coll Chem
2	Sichuan Univ, Coll Elect Engn
3	Sichuan Univ, Dept Phys
4	Sichuan Univ, Coll Life Sci
6	Sichuan Univ, Food Engn
7	Sichuan Univ, Coll Phys
8	Sichuan Univ, Sch Business
9	Wuhan Univ, Mat Sci

- 如何提取 `Sichuan Univ` 后面的学院？

正则表达式

什么是正则表达式

正则表达式（Regular Expression），是一种强大、便捷、高效的文本处理工具。它描述了一种字符串匹配的模式（pattern），比如：

- 具有固定格式的文本
- 电话号码
- 网络地址、邮件地址
- 日期格式
- 网页解析
- 等等

stringr 包

- 正则表达式并不是 R 语言特有的，事实上，几乎所有程序语言都支持正则表达式 (e.g. Perl, Python, Java, Ruby, etc).
- R 语言中很多函数都需要使用正则表达式，然而大神 Hadley Wickham 开发的 stringr 包让正则表达式简单易懂，所以今天我们介绍这个包。


```
library(stringr) #install.packages("stringr")
```

stringr 包

- 字符串处理基础
  - 字符串长度
  - 字符串组合
  - 字符串子串
- 使用正则表达式进行模式匹配
  - 基础匹配
  - 锚点 [máo][diǎn]
  - 字符类与字符选项
  - 重复
  - 分组与回溯引用
- 解决实际问题
  - 判断是否匹配
  - 提取匹配内容

字符串处理基础

字符串长度

想获取字符串的长度，使用 str_length() 函数:

```
str_length("R for data science")
#> [1] 18
```

字符串向量，也适用

```
str_length(c("a", "R for data science", NA))
#> [1]  1 18 NA
```

字符串长度

数据框里配合 dplyr 函数，同样很方便

```
data.frame(
  x = c("a", "R for data science", NA)
  ) %>%
  mutate(y = str_length(x))
```

x	y
a	1
R for data science	18
NA	NA

字符串组合

把字符串拼接在一起，使用 str_c() 函数

```
str_c("x", "y")
#> [1] "xy"
```

把字符串拼接在一起，可以设置中间的间隔

```
str_c("x", "y", sep = ", ")
#> [1] "x, y"

str_c(c("x", "y", "z"), sep = ", ")
#> [1] "x" "y" "z"
```

是不是和你想象的不一样，那就试试?str_c

字符串组合

```
str_c(c("x", "y", "z"), c("x", "y", "z"), sep = ", ")
#> [1] "x, x" "y, y" "z, z"
```

用在数据框里

```
data.frame(  x = c("I", "love", "you"),
             y = c("you", "like", "me") ) %>%
  mutate(z = str_c(x, y, sep = "|"))
```

x	y	z
I	you	I|you
love	like	love|like
you	me	you|me

字符串组合

使用 collapse 选项，是先组合，然后再转换成单个字符串，大家对比下

```
str_c(c("x", "y", "z"), c("a", "b", "c"), sep = "|")
#> [1] "x|a" "y|b" "z|c"
```

```
str_c(
  c("x", "y", "z"), c("a", "b", "c"), collapse = "|"
  )
#> [1] "xa|yb|zc"
```

字符串取子集

截取字符串的一部分，需要指定截取的开始位置和结束位置

```
x <- c("Apple", "Banana", "Pear")
str_sub(x, 1, 3)
#> [1] "App" "Ban" "Pea"
```

开始位置和结束位置如果是负整数，就表示位置是从后往前数，比如下面这段代码，截取倒数第 3 个至倒数第 1 个位置上的字符串

```
str_sub(x, -3, -1)
#> [1] "ple" "ana" "ear"
```

字符串取子集

也可以进行赋值，如果该位置上有字符，就用新的字符替换旧的字符

```
x <- c("Apple", "Banana", "Pear")
x
#> [1] "Apple"  "Banana" "Pear"

str_sub(x, 1, 1)
#> [1] "A" "B" "P"

str_sub(x, 1, 1) <- "Q"
x
#> [1] "Qpple"  "Qanana" "Qear"
```

使用正则表达式进行模式匹配

基础匹配

`str_view()` 是查看 string 是否匹配 pattern，

如果匹配，就高亮显示

```
x <- c("apple", "banana", "pear")
str_view(string = x, pattern = "an")
```

```
apple
banana
pear
```

基础匹配

有时候，我们希望在字符 a 前后都有字符（即，a 处在两字符中间，如 rap, bad, sad, wave，spear 等等）

```
x <- c("apple", "banana", "pear")
str_view(x, ".a.")
```

```
apple
banana
pear
```

基础匹配

这里的 . 代表任意字符.

```
c("s.d") %>%
  str_view(".")
```

如果想表达 . 本身呢？

```
c("s.d") %>%
  str_view("\\.")
```

锚点

```
x <- c("apple", "banana", "pear")
x
#> [1] "apple"  "banana" "pear"
```

希望 a 是字符串的开始

```
str_view(x, "^a")
```

```
apple
banana
pear
```

希望 a 是一字符串的末尾

```
str_view(x, "a$")
```

```
apple
banana
pear
```

锚点

```r
x <- c("apple pie", "apple", "apple cake")
str_view(x, "^apple$")
```

```
apple pie
apple
apple cake
```

字符类与字符选项

前面提到，. 匹配任意字符，事实上还有很多这种**特殊含义**的字符：

- `\d`: matches any digit.
- `\s`: matches any whitespace (e.g. space, tab, newline).
- `[abc]`: matches a, b, or c.
- `[^abc]`: matches anything except a, b, or c.

```
str_view(c("grey", "gray"), "gr[ea]y")
```

```
grey
gray
```

重复

控制匹配次数:

- ?: 0 or 1
- +: 1 or more
- *: 0 or more

```
x <- "Roman numerals: MDCCCLXXXVIII"
```

```
str_view(x, "CC?")
```

```
Roman numerals: MDCCCLXXXVIII
```

```
str_view(x, "X+")
```

```
Roman numerals: MDCCCLXXXVIII
```

重复

控制匹配次数:

- `{n}`: exactly n
- `{n,}`: n or more
- `{,m}`: at most m
- `{n,m}`: between n and m

重复

```r
x <- "Roman numerals: MDCCCLXXXVIII"
str_view(x, "C{2}")
```

```
Roman numerals: MDCCCLXXXVIII
```

重复

- 默认的情况，*，+ 匹配都是贪婪的，也就是它会尽可能的匹配更多
- 如果想让它不贪婪，而是变得懒惰起来，可以在 * 或 + 后加个?

```
x <- "Roman numerals: MDCCCLXXXVIII"
```

```
str_view(x, "CLX+")
```

```
Roman numerals: MDCCCLXXXVIII
```

```
str_view(x, "CLX+?")
```

```
Roman numerals: MDCCCLXXXVIII
```

小结一下

character	meaning	e.g.
*	0 or more	ca*t ➔ ct, cat, caat, caaaaaat…
+	1 or more	ca+t ➔ cat, caat, caaaat …
?	0 or 1	ca?t ➔ ct, cat
{m}	m times	ca{2} ➔ caa
{m, n}	At least m, at most n	ca{2,4} ➔ caat, caaat, caaaat

分组与回溯引用

```
ft <- fruit %>% head(10)
ft
#>  [1] "apple"         "apricot"       "avocado"
#>  [6] "bilberry"      "blackberry"    "blackcurrant"
```

我们想看看这些单词里，有哪些字母是重复两次的，比如 aa, pp. 如果用上面学的方法

```
str_view(ft, ".{2}", match = TRUE)
```

```
apple
apricot
avocado
banana
```

分组与回溯引用

所以需要用到新技术 **分组与回溯引用**，

```
str_view(ft, "(.)\\1", match = TRUE)
```

```
apple
bell pepper
bilberry
blackberry
blackcurrant
blood orange
blueberry
```

分组与回溯引用

```
str_view(ft, "(.)\\1", match = TRUE)
```

- . 是匹配任何字符
- (.) 将匹配项括起来，它就用了一个名字，叫\\1；如果有两个括号，就叫\\1 和\\2
- \\1 表示回溯引用，表示引用\\1 对于的 (.)

所以 (.)\\1 的意思就是，匹配到了字符，后面还希望有个**同样的字符**

分组与回溯引用

如果是匹配 `abab`, `wcwc`

```
str_view(ft, "(..)\\1", match = TRUE)
```

```
banana
```

进阶部分

look ahead

想匹配 Windows，同时希望 Windows 右侧是"95"，"98"，"NT"，"2000" 中的一个

```
win <- c("Windows2000", "Windows", "Windows3.1")
str_view(win, "Windows(?=95|98|NT|2000)")
```

```
Windows2000
Windows
Windows3.1
```

look ahead

```
win <- c("Windows2000", "Windows", "Windows3.1")
str_view(win, "Windows(?!95|98|NT|2000)")
```

```
Windows2000
Windows
Windows3.1
```

look behind

```
win <- c("2000Windows", "Windows", "3.1Windows")
str_view(win, "(?<=95|98|NT|2000)Windows")
```

```
2000Windows
Windows
3.1Windows
```

look behind

```
win <- c("2000Windows", "Windows", "3.1Windows")
str_view(win, "(?<!95|98|NT|2000)Windows")
```

```
2000Windows
Windows
3.1Windows
```

小结

有四种情形：

- （?=pattern）要求此位置的**后面**必须匹配 pattern
- （?!pattern）要求此位置的**后面**不能匹配 pattern
- （?<=pattern）要求此位置的**前面**必须匹配 pattern
- （?<!pattern）要求此位置的**前面**不能匹配 pattern

解决实际问题

确定一个字符向量是否匹配一种模式

想判断是否匹配？也可以用到 `str_detect()` 函数

x
apple
banana
pear

```
d %>% mutate(has_e = str_detect(x, "e"))
```

x	has_e
apple	TRUE
banana	FALSE
pear	TRUE

确定一个字符向量是否匹配一种模式

用去筛选也很方便

x
apple
banana
pear

```
d %>% filter(str_detect(x, "e"))
```

x
apple
pear

提取匹配的内容

我们希望能提取第二列中的数值，构成新的一列

x	y
1	wk 3
2	week-1
3	7
4	w#9

```
dt %>% mutate(
    z = str_extract(y, "[0-9]")
  )
```

x	y	z
1	wk 3	3
2	week-1	1
3	7	7
4	w#9	9

提取匹配的内容

回到上课提问：如何提取 `Sichuan Univ` 后面的学院？

No	address
1	Sichuan Univ, Coll Chem
2	Sichuan Univ, Coll Elect Engn
3	Sichuan Univ, Dept Phys
4	Sichuan Univ, Coll Life Sci
6	Sichuan Univ, Food Engn
7	Sichuan Univ, Coll Phys
8	Sichuan Univ, Sch Business
9	Wuhan Univ, Mat Sci

提取匹配的内容

```
d %>% mutate(
  coll = str_extract(address, "(?<=Sichuan Univ,).*")
) %>%
  tidyr::unnest(coll, keep_empty = TRUE)
```

No	address	coll
1	Sichuan Univ, Coll Chem	Coll Chem
2	Sichuan Univ, Coll Elect Engn	Coll Elect Engn
3	Sichuan Univ, Dept Phys	Dept Phys
4	Sichuan Univ, Coll Life Sci	Coll Life Sci
6	Sichuan Univ, Food Engn	Food Engn
7	Sichuan Univ, Coll Phys	Coll Phys
8	Sichuan Univ, Sch Business	Sch Business
9	Wuhan Univ, Mat Sci	NA