

R 语言编程：基于 tidyverse

第 05 讲 (选讲) 数据结构 Iv: 正则表达式, 时间序列

张敬信

2022 年 2 月 21 日

哈尔滨商业大学

九. 正则表达式

正则表达式，是根据字符串规律按一定法则，简洁表达一组字符串的表达式。正则表达式通常就是从貌似无规律的字符串中发现规律性，进而概括性地表达它们所共有的规律或模式，以方便地操作处理它们，这是真正的**化繁为简，以简驭繁**的典范。

几乎所有的高级编程语言都支持正则表达式，正则表达式广泛应用于文本挖掘、数据预处理，例如：

- 检查文本中是否含有指定的特征词
- 找出文本中匹配特征词的位置
- 从文本中提取信息
- 修改文本

- 正则表达式包括：
 - 只能匹配自身的普通字符（如英文字母、数字、标点等）
 - 被转义了的特殊字符（称为“元字符”），用于构造匹配规则
- 正则表达式学习建议：
 - 先学会最常用的三个正则表达式实例
 - 遇到具体问题，查阅基本语法表，尝试构造正则表达式，调试得到结果

1. 常用的元字符

符号	描述
.	匹配除换行符“ <code>\n</code> ”以外的任意字符
\\	转义字符，匹配元字符时，使用“ <code>\\</code> 元字符”
	表示或者，即 前后的表达式任选一个
^	匹配字符串的开始
()	提取匹配的字符串，即括号内的看成一个整体，即指定子表达式
[]	可匹配方括号内任意一个字符
{ }	前面的字符或表达式的重复次数： <code>{n}</code> 表示重复 <code>n</code> 次； <code>{n,}</code> 重复 <code>n</code> 次到更多次； <code>{n, m}</code> 表示重复 <code>n</code> 次到 <code>m</code> 次
*	前面的字符或表达式重复 0 次或更多次
+	前面的字符或表达式重复 1 次或更多次
?	前面的字符或表达式重复 0 次或 1 次

注：其他语言中的转义字符一般是\.

2. 特殊字符类与反义

符号	描述
\\d 与 \\D	匹配数字，匹配非数字
\\s 与 \\S	匹配空白符，匹配非空白符
\\w 与 \\W	匹配字母或数字或下划线或汉字，匹配非\\w 字符
\\b 与 \\B	匹配单词的开始或结束的位置，匹配非\\b 的位置
\\h 与 \\H	匹配水平间隔，匹配非水平间隔
\\v 与 \\V	匹配垂直间隔，匹配非垂直间隔
[^...]	匹配除了...以外的任意字符

- `\\S+`: 匹配不包含空白符的字符串
- `\\d`: 匹配数字, 同 `[0-9]`
- `[a-zA-Z0-9]`: 匹配字母和数字
- `[\u4e00-\u9fa5]` 匹配汉字
- `[^aeiou]`: 匹配除 `aeiou` 之外的任意字符, 即匹配辅音字母

3. POSIX 字符类

符号	描述
<code>[:lower:]</code>	小写字母
<code>[:upper:]</code>	大写字母
<code>[:alpha:]</code>	大小写字母
<code>[:digit:]</code>	数字 0~9
<code>[:alnum:]</code>	字母和数字
<code>[:blank:]</code>	空白符包括空格、制表符、换行符、中文全角空格等
<code>[:cntrl:]</code>	控制字符
<code>[:punct:]</code>	标点符号包括! " # % & ' () * + - . / : ; 等
<code>[:space:]</code>	空格字符：空格, 制表符, 垂直制表符, 回车, 换行符, 换页符
<code>[:xdigit:]</code>	十六进制数字：0-9 A-F a-f
<code>[:print:]</code>	打印字符： <code>[:alpha:]</code> , <code>[:punct:]</code> , <code>[:space:]</code>
<code>[:graph:]</code>	图形化字符： <code>[:alpha:]</code> , <code>[:punct:]</code>

4. 运算优先级

圆括号括起来的表达式最优先，其次是表示重复次数的操作（即 `*` `+` `{ }`）；再次是连接运算（即几个字符放在一起，如 `abc`）；最后是或者运算（`|`）。

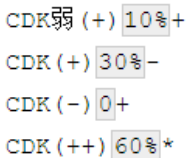
另外，正则表达式还有若干高级用法，常用的有**零宽断言**和**分组捕获**。

- 以上正则表达式语法组合起来使用，就能产生非常强大的匹配效果，对于匹配到的内容，根据需要可以提取它们，可以替换它们。
- stringr 包提供了以下函数及其 `_all` 版本：
 - `str_view()`: 调试和查看正则表达式的匹配效果
 - `str_extract()`: 提取正则表达式匹配到的内容
 - `str_replace()`: 替换正则表达式匹配到的内容
- 使用正则表达式关键是，能够从貌似没有规律的字符串中发现规律性，再将规律性用正则表达式语法表示出来。

例 1.1 直接匹配

- 适合想要匹配的内容具有一定规律性，该规律性可用正则表达式表示出来。
- 比如，数据中包含字母、符号、数值，我们想提取其中的数值，按正则表达式语法规则直接把要提取的部分表示出来：

```
library(stringr)
x = c("CDK 弱 (+)10%+", "CDK(+)30%-",
      "CDK(-)0+", "CDK(++)60%*")
str_view(x, "\\d+%")
str_view(x, "\\d+%?")
```



CDK 弱 (+) 10% +
CDK (+) 30% -
CDK (-) 0 +
CDK (++) 60% *

图 1: Viewer 窗口显示匹配效果

例 1.2 (零宽断言) 匹配两个标志之间的内容

- 适合想要匹配的内容没有规律性，但该内容位于两个有规律性的标志之间，标志也可以是开始和结束。
- 通常想要匹配的内容不包含两边的“标志”，这就需要用**零宽断言**。简单来说，就是一种引导语法告诉既要匹配到“标志”，但又不包含“标志”。
- 左边标志的引导语法是 (`?<= 标志`)，右边标志的引导语法是 (`?= 标志`)，而真正要匹配的内容放在它们中间。

- 比如，来自问卷星“来自 IP”数据，想要提取 IP、省份：

```
x = c("175.10.237.40(湖南-长沙)", "114.243.12.168(北京-北京)"  
      "125.211.78.251(黑龙江-哈尔滨)")
```

提取省份

```
str_extract(x, "\\(.*-") # 对比，不用零宽断言
```

```
#> [1] "(湖南-" "(北京-" "(黑龙江-"
```

```
str_extract(x, "(?<=\\().*?(?=)") # 用零宽断言
```

```
#> [1] " 湖南" " 北京" " 黑龙江"
```

提取 IP

```
# str_extract(x, "\\d.*\\d") # 直接匹配
```

```
str_extract(x, "^.*?(?=\\()") # 用零宽断言
```

```
#> [1] "175.10.237.40" "114.243.12.168" "125.211.78.251"
```

- 用零宽断言提取专业（位于“级”和数字之间）：

```
x = c("18 级能源动力工程 2 班", "19 级统计学 1 班")  
str_extract(x, "(?<= 级).*?(?=[0-9])")  
#> [1] " 能源动力工程" " 统计学"
```

关于懒惰匹配

- 正则表达式正常都是贪婪匹配，即重复直到文本中能匹配的最长范围，例如匹配小括号：

```
str_extract("(1st) other (2nd)", "\\(\\.+\\)")  
#> [1] "(1st) other (2nd)"
```

- 若想只匹配到第 1 个右小括号，则需要懒惰匹配，在重复匹配后面加上？即可：

```
str_extract("(1st) other (2nd)", "\\(\\.+?\\)")  
#> [1] "(1st)"
```

例 1.3 分组捕获

- 正则表达式中可以用圆括号来分组，作用是
 - 确定优先规则
 - 组成一个整体
 - 拆分出整个匹配中的部分内容（称为捕获）
 - 捕获内容供后续引用或者替换。

- 比如，来自瓜子二手车的数据：若型号是中文，则品牌与型号中间有空格；若型号为英文或数字，则品牌与型号中间没有空格。
- 若用正则表达式匹配“字母或数字”并分组，然后捕获该分组并用添加空格替换：

```
x = c(" 宝马 X3 2016 款", " 大众 速腾 2017 款",  
      " 宝马 3 系 2012 款")  
str_replace(x, "([a-zA-Z0-9])", " \\1")  
#> [1] " 宝马 X3 2016 款"      " 大众 速腾 2017 款" " 宝马 3 系 2012 款"
```

注：更多分组的引用还有 \\2, \\3, ...

十. 时间序列

- 为了研究某一事件的规律，依据时间发生的顺序将事件在多个时刻的数值记录下来，就构成了一个时间序列，用 $\{Y_t\}$ 表示。
- 例如，国家或地区的年度财政收入，股票市场的每日波动，气象变化，工厂按小时观测的产量等。另外，随温度、高度等变化而变化的离散序列，也可以看作时间序列。

- base R 下的 ts 数据类型是专门为时间序列设计的，本质上是一个数值型向量，扩展了时刻属性使得每个数都有一个时刻与之对应。
- 用 `ts(data, start, end, frequency, ...)` 生成时间序列

```
ts(data = 1:10, start = 2010, end = 2019)      # 年度数据
#> Time Series:
#> Start = 2010
#> End = 2019
#> Frequency = 1
#> [1]  1  2  3  4  5  6  7  8  9 10
ts(data = 1:10, start = 2010, frequency = 4)    # 季度数据
#>      Qtr1 Qtr2 Qtr3 Qtr4
#> 2010     1     2     3     4
#> 2011     5     6     7     8
#> 2012     9    10
```

参数 frequency 设置时间频率，默认为 1，表示一年有 1 个数据，frequency=12 (月度数据)，frequency=52 (周度数则)，frequency=365 (日度数据)。

- fpp3 生态下的 tsibble 包提供了整洁的时间序列数据结构 tsibble.
- 时间序列数据，无非就是指标数据 + 时间索引（或者再 + 分组索引）¹
- 对于分组时间序列数据，首先是一个数据框，若有分组变量需采用“长格式”作为一列，只需要指定时间索引、分组索引，就能变成时间序列数据结构。

¹多元时间序列，就是包含多个指标列.

- 现有 tibble 格式的 3 家公司 2017 年的日度股票数据，其中存放 3 只股票的 Stock 列为分组索引：

```
library(fpp3)
load("datas/stocks.rda")
stocks
#> # A tibble: 753 x 3
#>   Date      Stock  Close
#>   <date>    <chr> <dbl>
#> 1 2017-01-03 Google  786.
#> 2 2017-01-03 Amazon  754.
#> 3 2017-01-03 Apple   116.
#> 4 2017-01-04 Google  787.
#> 5 2017-01-04 Amazon  757.
#> # ... with 748 more rows
```

- 用 `as_tsibble()` 将数据框转化为时间序列对象 `tsibble`, 只需要指定时间索引 (`index`)、分组索引 (`key`):

```
stocks = as_tsibble(stocks, key = Stock, index = Date)
stocks
```

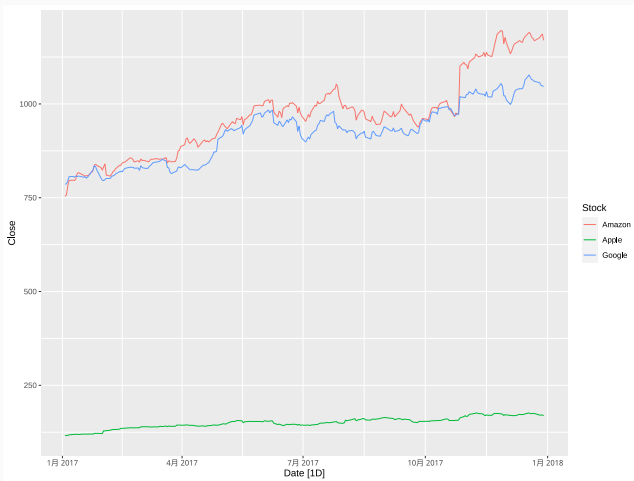
```
#> # A tsibble: 753 x 3 [1D]
#> # Key:      Stock [3]
#>   Date      Stock  Close
#>   <date>    <chr>  <dbl>
#> 1 2017-01-03 Amazon  754.
#> 2 2017-01-04 Amazon  757.
#> 3 2017-01-05 Amazon  780.
#> 4 2017-01-06 Amazon  796.
#> 5 2017-01-09 Amazon  797.
#> # ... with 748 more rows
```

- tsibble 对象非常便于后续处理和探索:

```
stocks %>%  
  group_by_key() %>%  
  index_by(weeks = ~ yearweek(.)) %>%      # 周度汇总  
  summarise(max_week = mean(Close))  
#> # A tsibble: 156 x 3 [1W]  
#> # Key:      Stock [3]  
#>   Stock      weeks max_week  
#>   <chr>    <week>    <dbl>  
#> 1 Amazon  2017 W01      772.  
#> 2 Amazon  2017 W02      805.  
#> 3 Amazon  2017 W03      809.  
#> 4 Amazon  2017 W04      830.  
#> 5 Amazon  2017 W05      827.  
#> # ... with 151 more rows
```


autoplot(stocks)

可视化



本篇主要参阅 (张敬信, 2022), (Hyndman and Athanasopoulos, 2021), (李东风, 2020), 模板感谢 (黄湘云, 2021), (谢益辉, 2021).

参考文献

Hyndman, R. J. and Athanasopoulos, G. (2021). *Forecasting: Principles and Practice*. O Texts, 3 edition.

张敬信 (2022). *R 语言编程：基于 tidyverse*. 人民邮电出版社, 北京.

李东风 (2020). *R 语言教程*.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.