

正则表达式简明指南



< R, Perl, JavaScript >

sanchufy

2021-10-10

参考来源

- Ben Forta (福达). 杨涛等译. 2015. [正则表达式必知必会（修订版）](#). 人民邮电出版社.
- Jan Goyvaerts, Steven Levithan. 2012. [Regular Expressions Cookbook](#), Second edition. O'Reilly Media.
- Jeffrey E. F. Friedl. 余晟译. 2012. [精通正则表达式](#). 电子工业出版社.
- 资源合集: [Awesome Regex](#) | [Awesome RegEx](#)
- 其它学习入口:
 - [正则表达式30分钟入门教程](#)
 - [Learn regex the easy way 中文版](#) by ziishaned 等 | [学习正则表达式的简单方法](#) by cdoco 等
 - [Reference - What does this regex mean?](#)
 - [Regular Expressions Quick Start](#)
 - Wikipedia: [Comparison of regular-expression engines](#)

Regular Expression

R、Perl、JavaScript

POSIX vs. **Perl** (Perl-like)

R 正则表达式

- 官方教程: [Regular Expressions as used in R](#)
- 中文教程:
 - 谢益辉 《R 语言忍者秘笈》 4.2 正则表达式
 - 黄湘云 《数据科学与 R 语言》 第 10 章 正则表达式
 - 战立侃: [R 使用的正则表达式](#)
 - 王敏杰 《数据科学中的 R 语言》 数据科学中的 R 语言
 - 李东风 《R语言教程》 36 R 语言的文本处理

R 中的设定与实现

- ERE (Extended Regular Expressions) + **Perl-like Regular Expressions**
 - **POSIX** 1003.2 + Perl 5.x
 - **TRE**^[wiki] 与 **PCRE** 引擎（库）的区别，可参考：
regular expressions in base R: 'perl=TRUE' vs. the default (PCRE vs. TRE)
- 在不同的 **locales**（R 进程的语言环境，如 POSIX）与 implementation（实现）中，解释将有所差异
- 默认贪婪匹配

PCRE: Perl Compatible Regular Expressions

POSIX: Portable Operating System Interface for uniX

R 包和扩展

- R 包 **stringi**: **Regular Expressions**
 - R 包 **stringr**: **Regular expressions**
 - R 包 **stringx**: **Drop-in replacements for base R string functions powered by stringi** 基础 R 的增强版
 - R 包 **qdapRegex**: **trinker/qdapRegex** 常用匹配模式打包集合
 - R 包 **rex**: **kevinushey/rex** 封装为友好可读的语言
- RStudio 扩展 **RegExplain**: 预览匹配结果

R 模式匹配与替换

- `sub()` 替换首个匹配, `gsub()` 全局替换
- `grep()` 匹配的 (或不被匹配的) 索引或值, `grep1()` 返回其逻辑向量
- `regexpr()` 首个匹配的起始位置及其匹配的长度, `gregexpr()` 向量构成的列表 [1]
- `regexexec()` 同 [1], 匹配位置和长度, 但返回列表, `gregexec()` 矩阵构成的列表 [2]
- `regmatches()` 从 [1][2] 获得的匹配数据中提取或替换匹配的子字符串
- `agrep()`, `agrep1()` 近似 (approximate) 匹配
- `match()` 完全匹配, `charmatch()`、`pmatch()` 部分匹配, `startsWith()` 匹配字符串的初始部分
- `strsplit()` 拆分字符串

命名捕获组 Named Capturing Groups

输入文本:

```
"Camellia sinensis (L.) O. Ktze. var. sinensis"
```

匹配模式:

```
"(^[A-Z][a-z]+) ([a-z]+)+ (.*) (var. [a-z]+)"
```

对捕获组进行反向引用 (backreferences):

```
"\1, \2, \4, \3"
```

返回结果:

```
"Camellia, sinensis, var. sinensis, (L.) O. Ktze."
```


PCRE 与 Perl 正则表达式

- PCRE - Perl Compatible Regular Expressions
- Perl regular expressions tutorial
- 其它非官方：
 - PCRE 正则表达式
 - Perl 正则表达式语法总结

环视 Lookaround

零宽断言 zero-length assertions, 即不消耗字符串中的字符, 而只是断言是否有可能匹配。

Assertion	Lookbehind	Lookahead
Positive	(?<=pattern)	(?=pattern)
Negative	(?<!pattern)	(?!pattern)

JavaScript 正则表达式

JavaScript 的正则表达式风格是该语言的 **ECMA-262 标准**的一部分。

- **MDN 中文教程**
- **微软：正则表达式语言 - 快速参考**
- **现代 JavaScript 教程中文版**
- 本站中的示例参考：**科技文章自定义词语高亮显示**

在线学习、测试

- 在线交互学习: [RegexOne](#)
- 小游戏: [Regex Cross-word](#)
- 再来一个小游戏: [Regex Golf](#)

- [RegExr](#): JavaScript & PHP/PCRE | 中译版
- [Regex101](#): PCRE, JavaScript, Golang, Python
- [RegExplain app demo](#): R
- [在线正则表达式测试工具](#)
- [菜鸟工具: 正则表达式在线测试](#)
- [在线正则表达式测试器](#)

速查表 Cheatsheets

- Basic Regular Expressions in **R**
- **R** **stringr** cheatsheet
- DaveChild: Regular Expressions Cheat Sheet
- William Marble: Regular Expressions Cheat Sheet
- **JavaScript** MDN: Regular expression syntax cheatsheet
- **JavaScript** Regex Cheatsheet
- **Python** 正则表达式速查表
- **JavaScript** 正则表达式 CheatSheet 速查表

可视化与常用模板

- 可视化：
 - [Debuggex](#): JavaScript, Python, PCRE
 - [Regulex](#) : JavaScript
 - [RegExper](#): JavaScript
 - [RegViz](#): JavaScript
- 常见匹配模式模板：
 - [常用正则大全](#)
 - [常用正则表达式](#)
 - [正则表达式大全](#)
 - [The Regex Cheat Sheet](#)
 - [RegexHub](#)

不懂就问查 Stack Overflow

聊聊几个常见的问题

中文字符

- Unicode Scripts
 - `\p{Han}` (Wiktionary: **Category:Han script**)
- Unicode Blocks
 - `\p{InCJK_Unified_Ideographs}` 等同于 `U+4E00-U+9FFF` (20977) **[1]**
 - `\p{InCJK_Symbols_and_Punctuation}` 等同于 `U+3000-U+303F` (64)
- **Unicode 编码**, 如 `[\u4E00-\u9FA5]` 比 **[1]** 字符总数稍少

<https://www.regular-expressions.info/unicode.html>

What's the complete range for Chinese characters in Unicode?

Index of Unicode Version 13.0.0 character properties in Perl

Simplified Chinese Unicode table

Delimiters ("foo" vs. /foo/)

"字符串" vs. /斜杠定界符/

以字符串作为输入时，反斜杠 \ 需要在字符串字面量中转义，即成倍出现。

```
var re = /[a-z]\s/i;  
var re = new RegExp("[a-z]\\s", "i");
```

```
var re = /[a-z]:\\\/i;  
var re = new RegExp("[a-z]:\\\\\\", "i");
```

贪婪 greedy vs. 懒惰 lazy

*、 +、 ?

VS.

*?、 +?、 ??

What do 'lazy' and 'greedy' mean in the context of regular expressions?

经典问答

1. What is a non-capturing group in regular expressions?
2. `\d` less efficient than `[0-9]`
3. Regular Expressions: Is there an AND operator?
4. How do I deal with special characters like `\^$.?*[+(){}[]` in my regex?
5. How to use Regular Expressions (Regex) in Microsoft Excel both in-cell and loops

应用场景示例

- 高级文本编辑器：文本查找与替换
- Everything：搜索
- Ditto：复制、粘贴
- userscript：文本高亮
- chrome extensions: **Gooreplacer**
- Total Commander：批量重命名
- R 爬虫之静态页面

Ditto

移除换行符*:

```
clip.AsciiTextReplaceRegex("([a-zA-Z0-9]\\b) *[\r\n]+", "$1 ");
clip.AsciiTextReplaceRegex("[\r\n]+ *(\\b[a-zA-Z0-9])", " $1");
clip.AsciiTextReplaceRegex(" *[\r\n]+ *", "");
clip.AsciiTextReplaceRegex(" {2,}", " ");

return false;
```

* <https://github.com/sabrogden/Ditto/wiki/Scripting>

附录：术语

- metacharacter 元字符
- character classes 字符类
 - POSIX character classes/bracket expressions
- wildcard characters 通配符
- delimiters 定界符; 分隔符
- atoms 原子
- pattern 模式
- character set 字符集
 - encoding 编码, 如: ASCII、Unicode
- 优先级: * (repetition) > concatenation > | (alternation)
 - concatenation 连接; 并置: foobar
 - alternation 交替; 置换: foo|bar
 - Kleene star 克林星号/Kleene closure 克林闭包: foo*

正则表达式是个投入很小回报很大的工具

— — old9

写正则还好，读正则才是痛苦的事。 正则表达式是可写不可读的。

— — Patrick95

在大多数形式中，如果存在至少一个匹配特定集合的正则表达式，那么就存在无数个同样匹配它的其他正则表达式。

— — Wikipedia