STAT243 Lecture 0.6 Regular Expression

1 概述与核心语法

正则表达式（regex）是一种用于匹配文本模式的领域特定语言（DSL），广泛用于 Python、R、UNIX 工具（如 `sed`、`awk`、`grep`）等环境中。

1.1 主要用途：

- 提取文本（如电话号码）
- 从文本中创建变量
- 清洗和格式化文本
- 文本挖掘
- 网页数据抓取

1.2 正则表达式由三部分组成：

1. **Literal characters**：字面匹配
2. **Character classes**：匹配某一类字符中的任意一个
3. **Modifiers**：修饰符，用于控制重复、位置等

1.3 特殊字符（元字符）：

Shell

```
. ^ $ + ? ( ) [ ] { } | \
```

若要匹配这些字符本身，需使用反斜杠 `\` 进行转义（在 R 中需使用两个反斜杠 `\\`）。

---

2 字符集与字符类

2.1 运算符说明：

表达式	说明
`[abc]`	匹配任意一个列出的字符
`[a-z]`	匹配任意一个范围内的字符
`[^abc]`	匹配任意一个不在列出的字符
`[^a-z]`	匹配任意一个不在范围内的字符
`.`	匹配除换行符外的任意字符
`\`	转义元字符的特殊含义

2.2 示例：

Shell

```
# 匹配包含数字的行
grep -E [0-9] test.txt

# 只输出匹配的数字
grep -E -o [0-9] test.txt
```

2.3 命名字符类（Named Character Classes）：

使用 `[:CLASS:]` 格式，如 `[:digit:]`、`[:punct:]` 等。

```Bash
# 匹配包含标点符号的行
grep -E [[:punct:]] test.txt

# 匹配数字、点或逗号
grep -E [[:digit:].,] test.txt
```

---

3 位置匹配

3.1 运算符说明：

表达式	说明
`^`	匹配行首
`$`	匹配行尾

3.2 示例：

```Bash
# 匹配以数字开头的行
grep -E ^[0-9] test.txt

# 匹配以数字结尾的行
grep -E [0-9]$ test.txt
```

---

4 重复、分组与引用

4.1 修饰符说明：

表达式	说明
`*`	匹配 0 次或多次
`?`	匹配 0 次或 1 次
`+`	匹配 1 次或多次
`{n,m}`	匹配 n 到 m 次
`|`	匹配左边或右边的表达式

4.2 示例：

```Bash
# 匹配 http 或 ftp
grep -E -o "(http|ftp)" test.txt

# 匹配电话号码格式
egrep '(1[-.])?[[:digit:]]{3}[-.][[:digit:]]{3}[-.][[:digit:]]{4}' file2.txt
```

5 贪婪匹配

默认情况下，正则表达式是“贪婪”的，即匹配尽可能长的字符串。

5.1 示例：

```Bash
# 贪婪匹配
grep -o "<.*>" file1.txt
# 输出: <b> in place </b> of <b> one </b>

# 非贪婪匹配 (Perl 语法)
grep -P -o "<.*?>" file1.txt
# 输出: <b> </b> <b> </b>
```

5.2 避免贪婪匹配的技巧：

使用更精确的字符集，避免使用 `.*`，例如：`<[^>]*>`

---

6 ⚠️ 注意：Globbing 与 Regex 的区别

- **Globbing**：用于文件名匹配，`*` 表示任意字符序列
- **Regex**：用于文本模式匹配，`*` 表示前一个字符的重复

如果还有其它要求，请随时告诉我，我们一起调整。