Основы программирования в R

Работа с данными с `tidyverse`: регулярные выражения

Алла Тамбовцева, НИУ ВШЭ

Содержание

Библиотека `tidyverse`

Для удобной работы с данными нам понадобится библиотека `tidyverse`. Это большая библиотека, которая включает в себя другие библиотеки, в том числе библиотеки `dplyr` и `tidyr` для обработки данных, библиотеку `ggplot2` для визуализации данных, библиотеку `stringr` для работы с регулярными выражениями, библиотеку `jsonlite` для загрузки файлов в формате JSON, библиотеку `rvest` для парсинга html-файлов.

Установим эту библиотеку и обратимся к ней:

```
install.packages("tidyverse")
```

```
library(tidyverse)
```

Введение в регулярные выражения

При работе с данными мы часто сталкиваемся с проблемой: необходимо выбрать не просто ячейки в таблице, которые содержат некоторый текст, а выбрать именно те строки, где текст начинается с определенной буквы или заканчивается цифрами. Чтобы решить эту проблему, понадобятся регулярные выражения.

Регулярные выражения — специальные выражения, последовательности символов, которые позволяют искать совпадения в тексте. Выражаясь более формально, они помогают найти подстроки определенного вида в строке. О регулярных выражениях можно думать как о шаблонах, в которые мы можем подставлять текст, и проверить, либо этот текст соответствует шаблону, либо нет. В самом простом случае в качестве регулярного выражения может использоваться обычная строка. В более сложных случаях нам понадобятся специальные символы. Для разбора этих символов в регулярных выражениях, создадим небольшой набор слов, не очень осмысленный, но удобный:

`xa, xaxa, xa-xa, xax, xex`

- Знак `.` соответствует одному любому символу в строке. Так, регулярное выражение `x.x` «поймает» слова `xax`, `xex` и `xaxa`.
- Знак `+` соответствует одному или более вхождению символа(ов), который стоит слева от `+`. Выражение `xa+` «поймает» все слова, кроме `xex`.
- Знак `*` соответствует нулю или более вхождениям символа, который стоит слева от `*`. Выражение `xa*` «поймает» все слова выше.
- Знак `?` соответствует нулю или одному вхождению символа, который стоит слева от `?`. Выражение `xax?` «поймает» все слова, кроме `xex`.

В регулярных выражениях можно фиксировать начало и конец строки. Например, если мы хотим из «смеющейся» строки выше взять только `xax` и `xex`, мы должны зафиксировать, что после буквы `x` строка должна заканчиваться: `x.x$`. Для начала строки используется символ `^`.

Как быть, если с помощью регулярного выражения нужно найти подстроку, содержащую знаки препинания? Те же точки, вопросительные знаки, скобки? Нужно их экранировать — ставить перед ними \, например, `\.`, `\,`, `\?`. Это символ будет сообщать R, что нам нужен именно конкретный символ (точка, запятая, знак вопроса и др.).

А если нам понадобится найти строки, содержащие буквы или цифры? Или и то, и другое, но обязательно с пробелом посередине? Промежутки, заключенные в квадратные скобки, позволяют найти цифры или буквы разных алфавитов и разных регистров:

- `[0-9]` соответствует любой цифре;
- `[A-Z]` соответствует любой заглавной букве английского алфавита;
- `[a-z]` соответствует любой строчной букве английского алфавита;
- `[А-Я]` и `[а-я]` – аналогично для букв русского алфавита.

Для цифр также есть специальный символ `\d` (от *digit*). Добавляя слэш, мы отмечаем, что ищем именно цифру, а не просто букву *d*.

Для пробела тоже существует свой символ – `\s` (от *space*). Этот символ соответствует ровно одному пробелу в тексте.

Любой знак, отличный от пробела, обозначается как `\S`, заглавная буква здесь отвечает за отрицание.

Помимо перечисленных выше символов в регулярных выражениях используются скобки. Круглые скобки объединяют символы в группы, это полезно при поиске последовательностей из нескольких символов. Квадратные скобки означают условие *или*. Например, в последовательности цифр `[0-9]` в выражении выше квадратные скобки означают выбор одной цифры (или 0, или 1, или 2, и так далее). Выражение `x[ae]x` найдет все последовательности вида `x.x` с буквой `a` или `e` посередине. Иногда для условия *или* добавляют уже знакомый нам оператор `|`: `x[a|e]x`.

В регулярных выражениях можно явно задавать число повторений символов. Если мы знаем точное число символов, то его можно указать в фигурных скобках. Так, выражение `a{4}` будет соответствовать четырем буквам `a` подряд. Если точное число повторений нам неизвестно, можно задать диапазон, указав начало и конец интервала через запятую. Например, такое выражение позволит найти от двух до четырех букв a подряд: `a{2,4}`. Если известен только левый или правый конец интервала, то второй конец можно опустить: `a{2,}` (не менее двух) или `a{,4}` (не более 4).

Регулярные выражения и `stringr`

Перейдем к примерам в R. В библиотеке `stringr`, которая загружается вместе с `tidyverse`, есть ряд функций, позволяющих выполнять поиск строк по регулярным выражениям.

Так как дальше мы будем работать с датафреймом по волшебному миру Дж.К.Роулинг, давайте добавим магии уже сейчас — создадим вектор с заклинаниями:

```
spells <- c("expecto patronum", "expelliarmus", "bombarda maxima",
            "lumos", "alohomora", "reducto", "riddikulus",
            "levicorpus", "liberacorpus", "wingardium leviosa",
            "impervius", "accio")
```

Найдем все заклинания, в которых есть буква “e”.

```
str_detect(spells, "e")
```

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

Функция `str_detect()` не возвращает вектор с подходящими строками, а выдает вектор из `TRUE` (буква “e” найдена) и `FALSE` (буква “e” не найдена). Чтобы получить вектор нужных строк, воспользуемся другой функцией — `str_subset()`:

```
str_subset(spells, "e")
```

```
## [1] "expecto patronum"   "expelliarmus"       "reducto"
## [4] "levicorpus"         "liberacorpus"       "wingardium leviosa"
## [7] "impervius"
```

Теперь найдем только те заклинания, которые начинаются с “e”:

```
str_subset(spells, "^e")
```

```
## [1] "expecto patronum" "expelliarmus"
```

Или те, которые заканчиваются на “s”:

```
str_subset(spells, "s$")
```

```
## [1] "expelliarmus" "lumos"        "riddikulus"   "levicorpus"   "liberacorpus"
## [6] "impervius"
```

А теперь те, которые заканчиваются на “us”:

```
str_subset(spells, "(us)$")
```

```
## [1] "expelliarmus" "riddikulus"   "levicorpus"   "liberacorpus" "impervius"
```

Выберем заклинания, которые начинаются на “a” или на “b”:

```
str_subset(spells, "^[ab]")
```

```
## [1] "bombarda maxima" "alohomora"       "accio"
```

То же самое, но с оператором `|`:

```
str_subset(spells, "^[a|b]")
```

```
## [1] "bombarda maxima" "alohomora"       "accio"
```

Возьмем оператор `|` и сделаем поиск по группам символов — выберем заклинания, которые начинаются на “le” или “li”:

```
str_subset(spells, "^(le)|^(li)")
```

```
## [1] "levicorpus"   "liberacorpus"
```

А теперь напишем шаблон: в заклинании должно быть 5 букв, первая “l”, последняя “s”:

```
# ... в середине 3 любых символа
str_subset(spells, "^l...s$")
```

```
## [1] "lumos"
```

Просто все заклинания, начинающиеся на “l” и заканчивающиеся на “s”:

```
# .+ для одного и более символов в середине
str_subset(spells, "^l.+s$")
```

```
## [1] "lumos"        "levicorpus"   "liberacorpus"
```

Если бы у нас кроме букв в словах встречались еще и другие символы, мы могли бы поставить ограничение и указать, что между “l” и “s” должны быть буквы английского алфавита:

```
str_subset(spells, "^l[a-z]+s$")
```

```
## [1] "lumos"        "levicorpus"   "liberacorpus"
```

Посмотрим на поиск повторяющихся символов. Найдем заклинания с двумя буквами “d”:

```
str_subset(spells, "d{2}")
```

```
## [1] "riddikulus"
```

Теперь с числом букв “d” от 1 до 2:

```
str_subset(spells, "d{1,2}")
```

```
## [1] "bombarda maxima"    "reducto"            "riddikulus"
## [4] "wingardium leviosa"
```

В завершение посмотрим на еще одну полезную функцию, которая нам понадобится для извлечения определенных последовательностей из строк. Извлечем все окончания заклинаний с `u`:

```
str_extract(spells, "u.$")
```

```
##  [1] "um" "us" NA   NA   NA   NA   "us" "us" "us" NA   "us" NA
```

Функция `str_extract()` удобна тем, что в случае отсутствия набора символов, описанного регулярным выражением, она автоматически возвращает `NA`. Эту функцию можно использовать, когда из текста нам нужно извлечь данные определенного вида. Например, из строки с датой рождения извлечь год (последовательность из 4 цифр), из строки с адресом извлечь улицу (слова после “ул.”), из строки с номером телефона выбрать только цифры, не затрагивая дефисы, скобки и прочее.