Основы программирования в R

Парсинг HTML-страниц: часть 2

Алла Тамбовцева, НИУ ВШЭ

Содержание

Продолжим парсить HTML-страницы и посмотрим на то, как быстро извлекать данные из таблиц, которые встречаются в коде HTML.

Подгрузим библиотеки:

```
library(rvest)
library(tidyverse)
```

Зайдем на страницу с результатами опросов Левада-Центра и поставим себе задачу — выгрузить данные из всех таблиц на странице. Считаем код со страницы и сохраним в `Lpage`:

```
Lpage <- read_html("https://www.levada.ru/indikatory/")
```

Найдем все таблицы на странице по тэгу `table`:

```
tabs <- Lpage %>% html_nodes("table")
```

В списке `tabs` 10 элементов, в то время как на странице всего 5 таблиц. Почему так? Для знакомства с устройством таблиц выберем первый элемент в `tabs` и посмотрим, что лежит в ячейках первой таблицы:

```
tabs[1] %>% html_nodes("td")
```

```
## {xml_nodeset (4)}
## [1] <td> </td>
## [2] <td>Одобряю</td>
## [3] <td>Не одобряю</td>
## [4] <td>Нет ответа</td>
```

Таблицы на сайте устроены так, что часть с вариантами ответа («Одобряю», «Не одобряю», «Нет ответа») представляет собой отдельную таблицу. Это нестрашно, мы уже сталкивались с ситуацией, когда элементы внутри списка нужно рассортировать по индексам. В данном случае таблицы с числами хранятся в элементах списка с четными индексами. Но об этом потом. Сейчас посмотрим, как избежать поиска по `td` и сделать выгрузку данных из таблицы проще.

Для примера выберем вторую таблицу и воспользуемся функцией `html_table()`, которая превращает HTML-код для таблицы в датафрейм (а точнее, в список датафреймов):

```
tab1 <- tabs[2]
dat1 <- html_table(tab1)
class(dat1)
```

```
## [1] "list"
```

В нашем случае таблица внутри одна, извлечем ее и получим датафрейм:

```
final <- dat1[[1]]
head(final[1:4, 1:12])
```

```
##        X1      X2      X3      X4      X5   X6   X7   X8   X9  X10  X11  X12
## 1  8.1999  9.1999 10.1999 11.1999 12.1999  1.2  2.2  3.2  4.2  5.2  6.2  7.2
## 2 31.0000 53.0000 65.0000 80.0000 79.0000 84.0 75.0 70.0 77.0 72.0 61.0 72.0
```

```
## 3 33.0000 27.0000 20.0000 12.0000 13.0000 10.0 17.0 21.0 15.0 17.0 26.0 17.0
## 4 37.0000 20.0000 15.0000  8.0000  8.0000  7.0  8.0  9.0  8.0 11.0 13.0 10.0
```

Если посмотреть на эту таблицу внимательно, возникнет два вопроса: почему так много столбцов, и откуда взялись непонятные значения в первой строке? Ответы на них лежат в структуре исходного кода страницы. Дело в том, что в исходном коде страницы хранятся не только данные за последний год, которые мы видим на сайте, но и данные, которые используются для построения графиков с динамикой одобрения. А на графиках мы видим временной промежуток с августа 1999 года. Это объясняет то, почему у нас так много столбцов, и подсказывает, что значения в первой строке — это даты опросов, номер месяца и год. Только из-за точки R распознал даты как обычные числа с плавающей точкой и «обрезал» нули в 2000, 2010 и 2020 годах.

Этот пример иллюстрирует полезность работы с исходным кодом страницы. Сами того не желая и ничего специально не предпринимая, мы собрали больше информации, чем планировали.

Приведем в порядок полученную таблицу. Перенесем значения из первой строки в названия столбцов и удалим эту строку из самой таблицы:

```
colnames(final) <- final[1, ]
final <- final[2:4, ]
```

Чтобы было ясно, какие значения чему соответствуют, добавим названия строк:

```
rownames(final) <- c("Approve",
                     "Not approve",
                     "No answer")
```

Обычно даты идут по строкам таблицы, а названия показателей — по столбцам. Транспонируем полученную таблицу (и превратим в датафрейм, так как функция `t()` возвращает матрицу):

```
survey <- as.data.frame(t(final))
head(survey)
```

```
##         Approve Not approve No answer
## 8.1999       31          33        37
## 9.1999       53          27        20
## 10.1999      65          20        15
## 11.1999      80          12         8
## 12.1999      79          13         8
## 1.2          84          10         7
```

Давайте для удобства создадим столбец с датой опроса:

```
survey$Date <- rownames(survey)
rownames(survey) <- 1:nrow(survey)
```

Поправим урезанные годы с помощью `str_replace_all()` и регулярных выражений:

```
survey$Date <- str_replace_all(survey$Date, "\\.2$", ".2000")
survey$Date <- str_replace_all(survey$Date, "\\.201$", ".2010")
survey$Date <- str_replace_all(survey$Date, "\\.202$", ".2020")
```

Пояснение к регулярным выражениям: мы ищем последовательности символов из точки и чисел 2, 201 и 202, причем такие, где перечисленные числа стоят в конце строки (`$`).

Теперь все красиво. На этом остановимся. При желании можно написать функцию и проделать подобные преобразования со всеми таблицами с данными в `tabs`.