Основы программирования в R

Работа с JSON-файлами. Формат дата-время. Работа с API.

Алла Тамбовцева, НИУ ВШЭ

Содержание

Предполагается, что студенты уже знакомы с форматом JSON и API благодаря обязательному курсу по Python.

Загрузка и обработка файла JSON

Формат JSON расшифровывается как *Java Script Object Notation* и является одним из популярных форматов хранения данных. Чаще всего с этим форматом можно столкнуться при работе с API — результаты запросов обычно возвращаются в виде объекта JSON. На Python мы уже сталкивались с этим форматом и работали с объектами такого вида:

```
[
 {'id': 1, 'name' : 'Anna'},
 {'id' : 2, 'name' : 'Nick'}
]
```

Объект выше — простой список словарей. В R словарей нет, поэтому чаще всего JSON будет распознаваться как список списков или список векторов.

Для работы с JSON-файлами нам понадобится библиотека `jsonlite`. Установим и обратимся к ней:

```
install.packages("jsonlite")
```

```
library(jsonlite)
```

Сегодня мы поработаем с файлом, в котором хранятся предварительные результаты довольно необычных выборов — выборов в Арбитражный комитет Википедии. Зайдем на эту страницу и скачаем файл `allvotes.json` (выйти на список файлов можно через эту страницу, выбрав *Предварительные итоги выборов* и зайдя в машиночитаемые результаты).

Чтобы понимать, с какого рода данными мы будем работать, несколько пояснений:

- пользователи могут голосовать за (+) и против (-), причем за любое число кандидатов;
- пользователи могут менять свой голос за кандидата до окончания голосования;
- в файле указана дата голосования, голос, имя кандидата, имя избирателя.

Теперь загрузим файл `allvotes.json` в R с помощью функции `fromJSON()`:

```
votes <- fromJSON("/Users/allat/Desktop/allvotes.json")
```

Посмотрим на класс `votes`:

```
class(votes)
```

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

Это большой список текстовых векторов. Однако есть небольшая проблема: не все векторы имеют одинаковую длину, встречаются результаты, которые имеют пометку «тестовое голосование». Выглядят они так:

```
votes[[12]]
```

```
## [1] "2020-01-30 00:05:00"  "+"                     "Alexander Roumega"
## [4] "Track13"              "тестовое голосование"
```

Так как тестовое голосование нас не интересует, давайте отберем из списка `votes` только вектора из четырех элементов. Фильтрацию списка по условию можно выполнить с помощью самого условия и функции `sapply()`.

Функция `sapply()` позволит нам определить длину каждого векторе в списке:

```
head(sapply(votes, length), 15)
```

```
##  [1] 4 4 4 4 4 4 4 4 4 4 4 5 5 5 5
```

А теперь сформулируем условие и укажем его в квадратных скобках:

```
votes <- votes[sapply(votes, length) == 4]
```

Теперь в списке `votes` хранятся векторы одной длины. А это значит, что мы свободно сможем превратить его в датафрейм! Подгрузим любимый `tidyverse`:

```
library(tidyverse)
```

Теперь сконвертируем `votes` в датафрейм и транспонируем для более привычного вида (вспомним, что `t()` возвращает матрицу):

```
votes_dat <- votes %>% as.data.frame %>% t %>% as.data.frame
head(votes_dat)
```

```
##                                                                                 V1
## c..2020.01.30.00.01.00.........Alexander.Roumega....Inctructor. 2020-01-30 00:01:00
## c..2020.01.30.00.01.00.........Alexei.Kopylov....Inctructor..   2020-01-30 00:01:00
## c..2020.01.30.00.01.00.........Bolboschoenus....Inctructor..    2020-01-30 00:01:00
## c..2020.01.30.00.01.00.........Drbug....Inctructor..            2020-01-30 00:01:00
## c..2020.01.30.00.01.00.........Kaganer....Inctructor..          2020-01-30 00:01:00
## c..2020.01.30.00.01.00.........Vladimir.Solovjev....Inctructor. 2020-01-30 00:01:00
##                                                                                 V2
## c..2020.01.30.00.01.00.........Alexander.Roumega....Inctructor.  +
## c..2020.01.30.00.01.00.........Alexei.Kopylov....Inctructor..    +
## c..2020.01.30.00.01.00.........Bolboschoenus....Inctructor..     +
## c..2020.01.30.00.01.00.........Drbug....Inctructor..             +
## c..2020.01.30.00.01.00.........Kaganer....Inctructor..           +
## c..2020.01.30.00.01.00.........Vladimir.Solovjev....Inctructor.  +
##                                                                              V3
## c..2020.01.30.00.01.00.........Alexander.Roumega....Inctructor. Alexander Roumega
## c..2020.01.30.00.01.00.........Alexei.Kopylov....Inctructor..      Alexei Kopylov
## c..2020.01.30.00.01.00.........Bolboschoenus....Inctructor..        Bolboschoenus
## c..2020.01.30.00.01.00.........Drbug....Inctructor..                       Drbug
## c..2020.01.30.00.01.00.........Kaganer....Inctructor..                    Kaganer
## c..2020.01.30.00.01.00.........Vladimir.Solovjev....Inctructor. Vladimir Solovjev
##                                                                           V4
## c..2020.01.30.00.01.00.........Alexander.Roumega....Inctructor. Inctructor
## c..2020.01.30.00.01.00.........Alexei.Kopylov....Inctructor..   Inctructor
## c..2020.01.30.00.01.00.........Bolboschoenus....Inctructor..    Inctructor
```

```
## c..2020.01.30.00.01.00.........Drbug....Inctructor..             Inctructor
## c..2020.01.30.00.01.00.........Kaganer....Inctructor..           Inctructor
## c..2020.01.30.00.01.00.........Vladimir.Solovjev....Inctructor. Inctructor
```

Уберем дату из названий строк и переименуем столбцы:

```
rownames(votes_dat) <- 1:nrow(votes_dat)
colnames(votes_dat) <- c("timestamp", "vote", "cand", "voter")
head(votes_dat)
```

```
##             timestamp vote              cand      voter
## 1 2020-01-30 00:01:00    + Alexander Roumega Inctructor
## 2 2020-01-30 00:01:00    +    Alexei Kopylov Inctructor
## 3 2020-01-30 00:01:00    +     Bolboschoenus Inctructor
## 4 2020-01-30 00:01:00    +             Drbug Inctructor
## 5 2020-01-30 00:01:00    +           Kaganer Inctructor
## 6 2020-01-30 00:01:00    + Vladimir Solovjev Inctructor
```

Теперь все красиво. Но для дальнейшей работы нам понадобится еще поправить типы некоторых столбцов. По умолчанию R делает текстовые столбцы факторными, нам это будет мешать, поэтому выполним приведение типов:

```
votes_dat <- votes_dat %>% mutate(timestamp = as.character(timestamp),
                      cand = as.character(cand))
```

Формат дата-время

Довольно часто при работе с данными, особенно когда данные приведены не за год или месяц, а с точностью до минут или секунд, возникает необходимость приводить строки с датами к специальному типу *date-time*. В отличие от обычных строк, с объектами типа *date-time* можно выполнять операции, определенные на датах: считать разницу между датами в секундах, прибавлять к датам некоторые числа, приводить даты в соответствие принятым стандартам и прочее.

Потестируем на первой дате в датафрейме функции `as.Date()` и `as.POSIXlt()`.

```
test <- votes_dat$timestamp[1]
```

Функция `as.Date()` умеет извлекать дату из строки по предложенному шаблону. Вид шаблона очень похож на шаблоны, которые мы использовали для форматирования строк (вспомните `%s`, %i‘ и прочие):

```
as.Date(test, "%Y-%m-%d")
```

```
## [1] "2020-01-30"
```

Здесь мы знаем, что в строке на первом месте стоит год в виде четырехзначного числа (`Y`), месяц в виде числа (`m`) и день тоже в виде числа (`d`). Эти обозначения являются распространенными, встречаются не только в R, но и в других языках программирования. Подробнее о разных сокращениях и форматах можно почитать здесь.

Функция `as.POSIXlt()` умеет извлекать не только дату, но и время. Здесь к шаблону строки добавятся часы (`H`), минуты (`M`) и секунды (`S`):

```
as.POSIXlt(test, format="%Y-%m-%d %H:%M:%S")
```

```
## [1] "2020-01-30 00:01:00 MSK"
```

Подробнее о формате POSIX можно почитать здесь. Важный факт: если вы столкнетесь с датой-временем в виде огромного целого числа, это скорее всего, дата-время в формате POSIX, то есть число секунд, прошедшее с 1 января 1970 года.

Теперь применим функцию `as.POSIXlt()` ко всем значениям в столбце `timestamp`:

```
votes_dat$timestamp <- as.POSIXlt(votes_dat$timestamp)
```

Что нам это дает? Как минимум то, что мы сможем вычислять разности между временными точками или отслеживать динамику голосования, выбрав какой-то момент в качестве точки отсчета. Например, посчитаем разницу во времени голосования в строках 12 и 1:

```
mydiff <- votes_dat$timestamp[12] - votes_dat$timestamp[1]
mydiff
```

```
## Time difference of 53 mins
```

Немного про агрегирование и `spread()`

Результаты голосования с такой детализацией — это интересно, в реальном мире с тайной голоса такого не встретить. Но что если все-таки хочется получить более общую картину, результаты в агрегированном виде? Попробуем сгруппировать датафрейм по имени кандидата и голосу и посчитать число наблюдений в группах:

```
votes_dat %>% group_by(cand, vote) %>% tally %>% head
```

```
## # A tibble: 6 x 3
## # Groups:   cand [3]
##   cand              vote      n
##   <chr>             <fct> <int>
## 1 Николай Эйхвальд  -        46
## 2 Николай Эйхвальд  +        59
## 3 Томасина          -        39
## 4 Томасина          +       121
## 5 Alexander Roumega -        46
## 6 Alexander Roumega +        42
```

Замечательно! Мы видим количество голосов «за» и «против» по каждому кандидату. Но с таблицей такого вида не очень удобно работать. Мы не сможем, например, быстро посчитать проценты голосов «за» и «против» или посмотреть на их распределение, так как пока они у нас в одном столбце `vote`. Хотелось бы получить табличку, где есть отдельный столбец с голосами «за» и отдельный — с голосами «против». Добиться этого можно с помощью функции `spread()`:

```
elect <- votes_dat %>% group_by(cand, vote) %>% tally %>%
  spread(vote, n)

head(elect)
```

```
## # A tibble: 6 x 3
## # Groups:   cand [6]
##   cand                `-`   `+`
##   <chr>             <int> <int>
## 1 Николай Эйхвальд     46    59
## 2 Томасина             39   121
## 3 Alexander Roumega    46    42
## 4 Alexei Kopylov       15   123
## 5 Arsenal.UC           67    18
## 6 Bolboschoenus        72    18
```

Функция `spread()` в буквальном смысле расширяет таблицу, используя уникальные категории в столбце `vote` и вытаскивая значения из столбца `n`. В таких случаях иногда говорят, что мы перевели таблицу

из формата *long* в формат *wide*, то есть из длинного представления данных сделали широкое. Если пытаетесь вспомнить аналоги подобных операций в Python, почитайте про `stack()` и `unstack()` в `pandas`.

Работа с API

Симпатичную таблицу с результатами мы получили. Осталось теперь дополнить ее информацией о самих кандидатах. Такую информацию мы можем получить, используя API Википедии.

Для того, чтобы отправлять запросы к API и обрабатывать ответы, нам понадобится библиотека `httr`. Установим ее и обратимся к ней:

```
install.packages("httr")
```

```
library(httr)
```

Если мы обратимся к документации API Википедии и найдем раздел Userinfo, мы выясним, какие пары *ключ-значение* нужно добавлять в запрос для получения информации о пользователе. Итак, что у нас есть:

- параметр `action` со значением `query` для запроса;
- параметр `format` со значением `json`, если хотим такой результат в JSON;
- параметр `list` со значением `users`, так как ищем в разделе с пользователями;
- параметр `usprop`, в котором через `|` перечисляем нужные характеристики пользователя (`prop` — от *properties*);
- параметр `ususers`, в котором через `|` перечисляем имена пользователей.

Как можно заметить, для отправки запроса в документации предлагается *GET request*. Для отправки такого запроса и получения ответа мы можем воспользоваться функцией `GET()` из `httr`. Запрос со всеми указанными выше параметрами оформляется в виде списка и помещается в аргумент `query`. Выведем информацию по пользователю *Drbug*:

```
r <- GET("https://ru.wikipedia.org/w/api.php",
    query = list(action = "query",
                 format = "json",
                 list = "users",
                 usprop = "blockinfo|groups|editcount|registration|emailable|gender",
                 ususers = "Drbug"))
r
```

```
## Response [https://ru.wikipedia.org/w/api.php?action=query&format=json&list=users&usprop=blockinfo%7Cg
##   Date: 2020-06-11 00:30
##   Status: 200
##   Content-Type: application/json; charset=utf-8
##   Size: 207 B
```

Результат, который мы получили, имеет класс *response*. Но из этого объекта всегда можно извлечь содержимое с помощью функции `content()`. Выгрузим содержимое в виде текста:

```
content(r, as="text")
```

```
## [1] "{\"batchcomplete\":\"\",\"query\":{\"users\":[{\"userid\":47,\"name\":\"Drbug\",\"editcount\":23
```

Выглядит не очень. А если попарсим?

```
content(r, as="parsed")
```

```
## $batchcomplete
## [1] ""
```

```
##
## $query
## $query$users
## $query$users[[1]]
## $query$users[[1]]$userid
## [1] 47
##
## $query$users[[1]]$name
## [1] "Drbug"
##
## $query$users[[1]]$editcount
## [1] 23664
##
## $query$users[[1]]$registration
## NULL
##
## $query$users[[1]]$groups
## $query$users[[1]]$groups[[1]]
## [1] "editor"
##
## $query$users[[1]]$groups[[2]]
## [1] "reviewer"
##
## $query$users[[1]]$groups[[3]]
## [1] "sysop"
##
## $query$users[[1]]$groups[[4]]
## [1] "*"
##
## $query$users[[1]]$groups[[5]]
## [1] "user"
##
## $query$users[[1]]$groups[[6]]
## [1] "autoconfirmed"
##
##
## $query$users[[1]]$emailable
## [1] ""
##
## $query$users[[1]]$gender
## [1] "unknown"
```

Уже лучше, похоже на список, но слишком длинно. Давайте вернемся к первому варианту в виде текста и воспользуемся тем, что функция `fromJSON()` умеет не только считывать файлы в формате JSON, но и строки.

```
fromJSON(content(r, as="text"))
```

```
## $batchcomplete
## [1] ""
##
## $query
## $query$users
##   userid  name editcount registration
## 1     47 Drbug     23664           NA
```

```
##                                                groups emailable  gender
## 1 editor, reviewer, sysop, *, user, autoconfirmed           unknown
```

Это уже похоже на что-то разумное. Давайте запросим информацию по всем кандидатам в `elect`, а потом аналогичным образом получим красивый датафрейм. Для этого нам нужно склеить все имена пользователей в столбце `cand`, используя разделитель `|`. Как мы уже знаем, для склеивания строк существует функция `paste()`. Но если мы применим ее без дополнительных опций, получим немного не то:

```
paste(elect$cand, "|")
```

```
##  [1] "Николай Эйхвальд |"  "Томасина |"          "Alexander Roumega |"
##  [4] "Alexei Kopylov |"    "Arsenal.UC |"        "Bolboschoenus |"
##  [7] "Drbug |"             "Kaganer |"          "Klip game |"
## [10] "Vladimir Solovjev |" "Wanderer777 |"      "Zanka |"
```

Функция приклеила `|` к каждому элементу вектора. Чтобы R понял, что это не вторая часть для склеивания, а разделитель, надо поместить символ в аргумент `collapse`:

```
users_all <- paste(elect$cand, collapse="|")
users_all
```

```
## [1] "Николай Эйхвальд|Томасина|Alexander Roumega|Alexei Kopylov|Arsenal.UC|Bolboschoenus|Drbug|Kagane
```

Теперь возвращаемся к запросу:

```
res <- GET("https://ru.wikipedia.org/w/api.php",
    query = list(action = "query",
                 format = "json",
                 list = "users",
                 usprop = "blockinfo|groups|editcount|registration|emailable|gender",
                 ususers = users_all))
```

И преобразуем содержимое ответа на запрос в датафрейм:

```
wiki_dat <- fromJSON(content(res, as="text")) %>% as.data.frame
```

Давайте выберем столбцы, которые могут быть интересны для дальнейшего анализа: имя пользователя, дата регистрации пользователя, группы (роли) пользователя, пол пользователя. А заодно переменуем столбец `query.users.name` в `cand`, чтобы столбцы с именами кандидатов назывались одинаково в `elect` и `wiki_small`.

```
wiki_small <- wiki_dat %>% dplyr::select("query.users.name",
                                         "query.users.registration",
                                         "query.users.groups",
                                         "query.users.gender") %>%
  dplyr::rename(cand = query.users.name)
```

Теперь остался финальный аккорд: склеим два датафрейма, датафрейм `elect` с результатами выборов и датафрейм `wiki_small` с информацией о кандидатах. Очевидно, будем клеить по столбцу `cand`, не зря же мы его переименовывали:

```
wiki_final <- merge(elect, wiki_small, by = "cand")
head(wiki_final)
```

```
##                cand  -   + query.users.registration
## 1  Николай Эйхвальд 46  59     2015-01-15T14:19:23Z
## 2          Томасина 39 121     2012-05-07T10:15:23Z
## 3 Alexander Roumega 46  42     2010-02-28T06:01:03Z
## 4    Alexei Kopylov 15 123     2006-09-20T19:17:44Z
```

```
## 5        Arsenal.UC 67  18     2012-07-05T11:22:38Z
## 6     Bolboschoenus 72  18     2015-05-12T18:40:00Z
##                                                        query.users.groups
## 1 editor, rollbacker, suppressredirect, uploader, *, user, autoconfirmed
## 2             arbcom, closer, editor, rollbacker, *, user, autoconfirmed
## 3                     closer, editor, rollbacker, *, user, autoconfirmed
## 4                                editor, sysop, *, user, autoconfirmed
## 5 editor, rollbacker, suppressredirect, uploader, *, user, autoconfirmed
## 6 editor, rollbacker, suppressredirect, uploader, *, user, autoconfirmed
##   query.users.gender
## 1            unknown
## 2             female
## 3               male
## 4               male
## 5            unknown
## 6               male
```

Готово!