

R 语言编程：基于 tidyverse

第 29 讲 (附录) R6 面向对象, R 爬虫

张敬信

2022 年 3 月 30 日

哈尔滨商业大学

R6 包为 R 提供了一个封装的面向对象编程的实现，相较于 S3，S4 类，R6 是更新的面向对象的类，支持引用语义。

面向对象的类语法都是类似的，只是语法外形不同，都有构造函数、属性（公共和私有）、方法。

以定义一个银行账户的类为例。

```
library(R6)
```

```
BankAccount = R6Class(  
  classname = "BankAccount",  
  public = list(  
    name = NULL,  
    age = NA,  
    initialize = function(name, age, balance) {  
      self$name = name  
      self$age = age  
      private$balance = balance  
    },  
    printInfo = function() {  
      cat(" 姓名: ", self$name, "\n", sep = "")  
      cat(" 年龄: ", self$age, " 岁\n", sep = "")  
    }  
  )  
)
```

```
    cat(" 存款: ", private$balance, " 元\n", sep = "")
    invisible(self)
},
deposit = function(dep = 0) {
  private$balance = private$balance + dep
  invisible(self)
},
withdraw = function(draw) {
  private$balance = private$balance - draw
  invisible(self)
}),
private = list(balance = 0)
)
```

- 用银行账户的类创建对象，并简单使用：

```
account = BankAccount$new(" 张三", age = 40,  
                           balance = 10000)
```

```
account$printInfo()
```

```
#> 姓名：张三
```

```
#> 年龄：40 岁
```

```
#> 存款：10000 元
```

```
account$balance
```

```
#> NULL
```

```
account$age
```

```
#> [1] 40
```

```
account$  
  deposit(5000)$  
  withdraw(7000)$  
  printInfo()  
#> 姓名: 张三  
#> 年龄: 40 岁  
#> 存款: 8000 元
```

- 定义**继承类**：可透支银行账户

```
BankAccountCharge = R6Class(  
  classname = "BankAccount",  
  inherit = BankAccount,  
  public = list(  
    withdraw = function(draw = 0) {  
      if (private$balance - draw < 0) {  
        draw = draw + 100  
      }  
      super$withdraw(draw = draw)  
    })  
))
```

- 用可透支银行账户创建对象，并简单使用：

```
charge_account = BanckAccountCharge$new(" 李四", age = 35,  
                                          balance = 1000)  
  
charge_account$withdraw(2000)$  
  printInfo()  
#> 姓名： 李四  
#> 年龄： 35 岁  
#> 存款： -1100 元
```


网络爬虫，简单来说就是通过编程让机器批量地从网页获取数据，主要分为三步：批量请求和抓取目标网页、解析并提取想要的信息、保存为本地数据文件。但是越来越多的网站都有了各种反爬机制能够识别、禁止机器浏览网页，所以又需要破解各种反爬虫，这涉及设置代理 IP、cookie 登录、伪装 Headers、GET/POST 表单提交、Selenium 模拟浏览器等复杂技术。

在网络爬虫领域，Python 无疑是更强大、资料也更多。但上述各种爬虫与反爬虫技术，在 R 里也都能实现。

B.1 rvest 爬取静态网页

打开一个目标网页，右键**查看网页源代码**可以在 HTML 结构中原原本本地看到想要抓取的数据，这就是**静态网页**。

对于静态网页，rvest 包提供了一套简洁和完整的数据抓取方案，主要函数：

- read_html(): 下载并解析网页
- html_nodes(): 定位并获取节点信息
- html_elements(): 提取节点元素信息
- html_text2(): 提取节点文本信息
- html_attr(): 提取节点的属性信息, 比如链接
- html_table(): 提取表格代码转化成数据框

另外，爬虫往往都是批量爬取若干网页，这就涉及循环迭代；对提取的文本数据做进一步的解析和提取，这就涉及正则表达式。

案例：爬取豆瓣读书 Top250。

1. 获取要批量爬取的网址

搜索并打开目标网页<https://book.douban.com/top250>, 先观察网页规律以构建要批量爬取的网址。总共 10 页，这是首页，依次点开第 2,3,...页观察网址规律，发现网址分别多了后缀：`?start=25`, `?start=50`, ...数值是等间隔。

想要批量爬取的网址都是有规律的（或者网页源码是按同样标签结构存放能够全部提取出来），有规律就能构造：

```
library(tidyverse)
suffix = str_c("?start=", seq(25,225, by = 25))
urls = str_c("https://book.douban.com/top250",
             c("", suffix))
```

2. 批量下载并解析网址

批量下载并解析这 10 个网页，用 `map` 循环迭代依次将 `read_html()` 作用在每个网址上。

但是直接这样做（同一 IP 瞬间打开 10 个网页）太容易触发网站的反爬虫机制，最简单（反爬机制稍强就会失效）的做法是增加一个随机等待时间：

```
library(rvest)
read_url = function(url){
  Sys.sleep(sample(5,1))    # 休眠随机 1~5 秒
  read_html(url)
}
htmls = map(urls, read_url)
```

3. 批量提取想要的内容并保存为数据框

这步是爬虫的最关键步骤：从 HTML 源码结构中找到相应位置、提取并保存想要的内容。只要对 HTML 有一点点粗浅了解，再结合浏览器插件 SelectorGadget 就足够。

在浏览器打开其中一个网址，点击 SelectorGadget，则页面处于等待选择状态，用鼠标点击想要提取的内容之一，比如书名“人间词话”，则该内容被标记为绿色，同时所有同类型的内容都被选中并被标记为黄色，但有些内容是识别错误的，点击它（变成红色）取消错误的黄色选择，浏览整个页面，确保只有你想要的书名被选中。



图 1: 用 SelectGadget 识别网页元素

右下角 CSS 选择器显示内容 `.pl2 a` 就是我们想要提取的书名所对应的节点，于是写代码提取它们：

```
book = html_nodes(html, ".pl2 a") %>%  
  html_text2()
```

同样的操作，分别对“作者/出版社/出版日期/定价”、“评分”、“评价数”、“描述”进行识别、提取、存放为向量，再打包到数据框。

注意，该过程是需要逐个调试的，提取的文本内容可能需要做简单的字符串处理和解析成数值等。

把从一个网页提取保存各个内容到保存为数据框的过程，定义为函数：

```
get_html = function(html) {  
  tibble(  
    book = html_nodes(html, ".pl2 a") %>%  
      html_text2(),  
    info = html_nodes(html, "p.pl") %>%  
      html_text2(),  
    score = html_nodes(html, ".rating_nums") %>%  
      html_text2() %>%  
      parse_number(),  
    comments = html_nodes(html, ".star .pl") %>%  
      html_text2() %>%  
      parse_number(),  
  )  
}
```



```
description = html_elements(html, "td") %>%  
  html_text2() %>%  
  stringi::stri_remove_empty() %>%  
  str_extract("(?<=\\)\\n\\n).*")  
}
```

注意，“描述”不是每本书都有的，所以不能像其它内容那样写代码（因行数对不上而报错），改用从更大的结构标签提取，再进行一系列的字符串处理。

然后用 `map_dfr` 依次将该函数应用到每个网页同时按行合并到一个结果数据框：

```
books_douban = map_dfr(htmls, get_html)
```

这就将 250 本书的信息都爬取下来，并保存在一个数据框（部分）：

book	info	score	comments	description
1 红槐梦	[清] 曹雪芹 著 / 人民文学出版社 / 1996-12 / 59.70元	9.6	343998	郁云作青痴，连解其中味？
2 活着	余华 / 作家出版社 / 2012-8-1 / 20.00元	9.4	615919	生的苦难与伟大
3 百年孤独	[哥伦比亚] 加西亚·马尔克斯 / 范晔 / 南海出版公司 / 2011-6 ...	9.3	345237	魔幻现实主义文学代表作
4 1984	[英] 乔治·奥威尔 / 刘绍铭 / 北京十月文艺出版社 / 2010-4-1 / ...	9.4	189192	果树荫下，我出卖你，你出卖我
5 飘	[美国] 玛格丽特·米切尔 / 李美华 / 译林出版社 / 2000-9 / 40.00元	9.3	181660	革命时期的爱情，随风而逝
6 三体全集：地球往事三部曲	刘慈欣 / 重庆出版社 / 2012-1-1 / 168.00元	9.4	102780	地球往事三部曲
7 三国演义（全二册）	[明] 罗贯中 / 人民文学出版社 / 1998-05 / 39.50元	9.3	139908	是非成败转头空

图 2: 豆瓣读书 Top250 爬虫数据（未清洗）

4. 进一步清洗数据框，并保存到数据文件

爬虫总是伴随着文本数据清洗，而这通常要用到正则表达式。

前面得到的数据框，info 列包含作者、出版社、出版日期、定价信息，它们在网页识别的时候是一个整体没办法区分开。

现在用字符串函数 + 正则表达式来做¹。

¹注意直接根据/分割是不行的，作者不一定几个。

```
books_douban = books_douban %>%  
  mutate(author = str_extract(info, ".*(?=/.*/ \\d{4})"),  
         press = str_extract(info, "(?<=/ )[^/]*(?=/ \\d{4})"),  
         Date = str_extract(info, "(?<=/ )[\\d-].*(?= /)"),  
         price = str_extract(info, "(?<=/)[^/]*$") %>%  
    parse_number()) %>%  
  select(-info)  
write_csv(books_douban, file = " 豆瓣读书 TOP250.csv")
```

最终的数据表（部分）如下：

1	book	score	comments	description	author	press	Date	price
2	红楼梦	9.6	343875	都云作者痴，谁解其中味？	[清] 曹雪芹 著	人民文学出版社	1996-12	59.7
3	活着	9.4	615690	生的苦难与伟大	余华	作家出版社	2012-8-1	20
4	百年孤独	9.3	345117	魔幻现实主义文学代表作	[哥伦比亚] 加西亚·马尔克斯 / 范	南海出版公司	2011-6	39.5
5	1984	9.4	189097	栗树荫下，我出卖你，你出卖我	[英] 乔治·奥威尔 / 刘绍铭	北京十月文艺出版	2010-4-1	28
6	飘	9.3	181609	革命时期的爱情，随风而逝	[美国] 玛格丽特·米切尔 / 李美华	译林出版社	2000-9	40
7	三体全集	9.4	102683	地球往事三部曲	刘慈欣	重庆出版社	2012-1-1	168

图 3: 豆瓣读书 Top250 爬虫数据（已清洗）

B2. http 爬取动态网页


动态网页，是基于 AJAX（异步 JavaScript 和 XML）技术动态加载内容，浏览到的内容是由服务器端根据时间、环境或数据库操作结果而动态生成，直接查看网页源码是看不到想要爬取的信息的。

爬取动态网页就需要先发送请求，对请求到的结果再做解析、提取、保存，`rvest` 包就无能为力了。`RCurl` 包或者其简化版的 `http` 包可以爬取动态网页。

1. 找到要爬取的内容


打开网易云课堂, 登录账号, 选择编程与开发, 进入目标页面。


右键“检查”, 依次点击“Network”, “fetch/HXR”, 刷新网页, 则右下窗口出现很多内容, 浏览找到 `studycourse.json`, 点开, 在 `preview` 下可以找到想要抓取的内容:



网易云课堂

会员福利

购课前先领优惠券


课程分类


课程


记得快去看看

Python入门实战课

6大热门方向

Go/Golang入门课

微服务与高并发实战

Python

数据
分析

用Python自动办公

让办公效率翻倍提升工作

第1课 Python零基础入门

Python零基础入门动画课

时间：随到随学

导师：大瓶课堂

券后价 ¥308

¥398

用Python自动办公，做职场

时间：随到随学

导师：美国学院

¥599

The screenshot displays the Chrome DevTools Network tab. The top toolbar includes filters for 'Blocked Requests' and 'Has blocked cookies'. The main area shows a list of network requests. The first request, 'hotwords.json?hotwordType=...', is selected, and its 'Headers' tab is active. The 'Preview' pane shows a JSON response with a 'message' field containing 'ok'. The 'Response' pane shows a list of 23 items, each representing a product entry with fields like productId, courseId, and productLine.

2. 构造请求 Headers, 用 POST 方法请求网页内容

点开 Headers, 重点关注:

- General 下的: Request URL, Request Method, Status Code
- Request Headers 下的: accept, edu-script-token, cookie, user-agent
- Request Payload 下的: pageIndex, pageSize, relativeOffset, rontCategoryId

获取这些信息之后², 就可以在 R 中构造 Headers:

²注意, Cookie 代表您账号登录信息, 是有时效性的.

```
library(httr)
## 构造请求头
myCookie = '您的最新 Cookie'
myUserAgent = 'Mozilla/5.0 (Windows NT 10.0; Win64; x64)
               AppleWebKit/537.36 (KHTML, like Gecko)
               Chrome/92.0.4515.159 Safari/537.36'
headers = c('accept' = 'application/json',
            'edu-script-token' = '38830026a471405eb9327d14d51eeda4',
            'User-Agent' = myUserAgent,
            'cookie' = myCookie)
```

```
# 二次实际请求到的 url
url = "https://study.163.com/p/search/studycourse.json"
# 构造请求 Payload
payload = list('pageIndex' = 1, 'pageSize' = 50,
               'relativeOffset' = 0,
               'frontCategoryId' = "480000003131009")
```

然后，就可以伪装成浏览器，发送 POST 请求获取数据：

```
## POST 方法执行单次请求
result = POST(url, add_headers(.headers = headers),
              body = payload, encode = "json")
```

3. 提取想要的结果

前面爬虫得到的 `result` 是 `json` 数据生成的复杂嵌套列表，需要把想要的数
据提取出来，并创建成数据框。

批量从一系列相同结构的列表提取某个成分下的内容，非常适合用 `map` 映射
成分名，对于内容为空的，设置参数 `.null = NA`，以保证数据框各列等长：

50 个课程信息列表的列表

```
lensons = content(result)$result$list  
df = tibble(ID = map_chr(lensons, "courseId"),  
            title = map_chr(lensons, "productName"),  
            provider = map_chr(lensons, "provider"),  
            score = map_dbl(lensons, "score"),  
            learnerCount = map_dbl(lensons, "learnerCount"),  
            lessonCount = map_dbl(lensons, "lessonCount"),  
            lector = map_chr(lensons, "lectorName",  
                             .null = NA))
```

4. 批量爬取所有页

这次不同页面是通过修改 Payload 参数实现的，总共 11 页，同样将爬取一页并保存到数据框过程定义为函数，自变量为第几页的序号：

```
get_html = function(p) {  
  Sys.sleep(sample(5, 1))  
  payload = list('pageIndex' = p, 'pageSize' = 50,  
                 'relativeOffset' = 50*(p-1),  
                 'frontCategoryId' = "480000003131009")  
  # POST 方法执行单次请求  
  result = POST(url, add_headers(.headers = headers),  
                body = payload, encode = "json")  
  lensons = content(result)$result$list
```

```
tibble(  
  ID = map_chr(lensons, "courseId"),  
  title = map_chr(lensons, "productName"),  
  provider = map_chr(lensons, "provider"),  
  score = map_dbl(lensons, "score"),  
  learnerCount = map_dbl(lensons, "learnerCount"),  
  lessonCount = map_dbl(lensons, "lessonCount"),  
  lector = map_chr(lensons, "lectorName", .null = NA))  
}
```

用 `map_dfr` 依次将该函数应用到每页序号向量，同时按行合并到一个结果数据框，再根据学习人数递降排序，保存到数据文件：

```
wy_lessons = map_dfr(1:11, get_html) %>%  
  arrange(-learnerCount)  
write.csv(wy_lessons,  
          file = " 网易云课堂编程开发类课程.csv")
```


最终，共爬取到 550 个课程的信息，结果（部分）如下：

1	ID	title	provider	score	learnerCount	lessonCount	lector
2	1003425004	老九零基础学编程系列之C语言	老九学堂	5	376440	102	徐嵩 等
3	302001	疯狂的Python：零基础小白入门	pythonercn	4.7	283680	101	邹琪鲜
4	1004987028	免费Python全系列教程全栈工程师	北京图灵学院	5	205746	100	图灵学院刘英
5	343001	Java课程 Java300集大型视频教程	北京尚学堂	4.9	177489	350	高洪 等
6	271005	面向对象程序设计-C++	翁恺	4.9	175049	41	翁恺
7	1367011	C/C++黑客编程项目实战课程	长沙择善教育	4.8	139907	75	Tony老师

图 4：网易云课堂编程与开发类课程的爬虫结果

另外，动态网页还可以用 `RSelenium` 包模拟浏览器行为爬取，或者 `V8` 包能将 `rvest` 包提取的 `JavaScript` 代码渲染出来得到想要爬取的数据。

本篇主要参阅(张敬信, 2022), R6 包文档, 以及博客文章, 模板感谢(黄湘云, 2021), (谢益辉, 2021).

参考文献

张敬信 (2022). *R 语言编程：基于 tidyverse*. 人民邮电出版社, 北京.

谢益辉 (2021). *rmarkdown: Dynamic Documents for R*.

黄湘云 (2021). *Github: R-Markdown-Template*.