R 语言编程：基于 tidyverse

第 29 讲 (附录)　Excel 任务, 非等连接,R 爬虫

张敬信

2022 年 12 月 6 日

哈尔滨商业大学

附录 C R 实现 Excel 中的 VLOOKUP 与透视表

在用 Office 系列日常办公中，经常需要批量重复地做的一些事情，都值得用 R 或 Python 写成程序代码实现自动化，这样可以一劳永逸为你节省大量的时间，还有就是 R 或 Python 代码所能处理的数据量和处理速度都远远超过 Excel.

C.1 VLOOKUP 查询

Excel 中的 VLOOKUP，Index + Match 是让很多人头痛的话题。

实际上 VLOOKUP 在**数据思维**下来看，无非就是：筛选行、选择列，数据连接(若从较大的表查询)，如果查询完还涉及修改，就再加上修改列。

数据思维，才是解决数据问题的正确思维、简洁思维，一看就懂，一用就会。

比如有如下的 Excel 数据表：

No	销售员	性别	销量	地区
1	王东	男	100	北京
2	小西	男	56	上海
3	小南	女	98	苏州
4	小北	女	66	上海
5	小中	男	87	天津
6	小王	女	99	上海
7	小李	男	20	上海

图 1： Excel 查询示例数据

先加载 tidyverse 包并读入数据：

```
library(tidyverse)
library(readxl)
df = read_xlsx("data/VLOOKUP 综合.xlsx")
```

先来看各种查询，其实就是构建筛选条件筛选行，想保留哪些列再选择列。

- 单条件查询：根据销售员查找销量

```
df %>%
  filter(销售员 == " 王东") %>%                    # 筛选行
  select(销售员, 销量)                             # 选择列
#> # A tibble: 1 x 2
#>   销售员  销量
#>   <chr>  <dbl>
#> 1 王东     100
```

注：查询多个销售员姓名，还可以用%in% （属于）写查询条件。

根据另一个表（比如 df2）里的销售员姓名查询，把筛选行改成右连接即可：

```
df %>%
  right_join(df2, by = "销售员") %>%     # 右连接
  select(销售员, 销量)                    # 选择列
```

- 多条件查询：查询销售员在某地区的销量

```
df %>%      # 根据两个条件筛选行
  filter(销售员 == " 王东", 地区 == " 北京") %>%
  select(销售员, 地区, 销量)
#> # A tibble: 1 x 3
#>   销售员 地区   销量
#>   <chr>  <chr> <dbl>
#> 1 王东   北京    100
```

- 多列查询：查询销售员的所有信息

```
df %>%
  filter(销售员 == " 王东")
#> # A tibble: 1 x 6
#>      No 销售员 性别    销量 地区   备注
#>   <dbl> <chr>  <chr> <dbl> <chr>  <lgl>
#> 1     1 王东   男      100 北京   NA
```

- 从右向左查询：查询某销量的销售员

```
df %>%     # 数据思维不用分左右
  filter(销量 == 66) %>%
  select(销量, 销售员)
#> # A tibble: 1 x 2
#>    销量 销售员
#>   <dbl> <chr>
#> 1    66 小北
```

- 使用通配符查询：查询姓名包含”中”

```
df %>%     # 是否检测到" 中" 字，支持正则表达式
  filter(str_detect(销售员, " 中")) %>%
  select(销售员, 销量)
#> # A tibble: 1 x 2
#>   销售员  销量
#>   <chr>  <dbl>
#> 1 小中      87
```

- 划分区间等级[1]：按销量划分等级

```
df %>%
  mutate(销量等级 = case_when(      # 修改列
    销量 < 60  ~ " 不及格",
    销量 < 85  ~ " 及格",
    TRUE          ~ " 优秀"))
#> # A tibble: 7 x 7
#>      No 销售员 性别    销量 地区   备注   销量等级
#>   <dbl> <chr>  <chr> <dbl> <chr> <lgl> <chr>
#> 1     1 王东   男      100 北京  NA    优秀
#> 2     2 小西   男       56 上海  NA    不及格
#> 3     3 小南   女       98 苏州  NA    优秀
#> # ... with 4 more rows
```

[1]对变量重新编码、连续变量离散化

C.2 数据透视表

数据透视表就是透过数据看到汇总的信息，其实就是分组汇总。比如有如下 Excel 数据表：

地区	城市	公司名称	类别名称	产品名称	订购日期	数量	单价	销售额
华北	天津	高上补习	饮料	苹果汁	1996-08-20	45	14.4	648
华东	温州	学仁贸易	饮料	苹果汁	1996-08-30	18	14.4	259
华北	天津	正太实业	饮料	苹果汁	1996-09-30	20	14.4	288
华北	天津	凯旋科技	饮料	苹果汁	1996-11-07	15	14.4	216
华北	天津	就业广兑	饮料	苹果汁	1996-11-14	12	14.4	173
华北	天津	浩天旅行	饮料	苹果汁	1996-12-03	15	14.4	216
华北	北京	留学服务	饮料	苹果汁	1997-01-07	10	14.4	144
华北	天津	池春建设	饮料	苹果汁	1997-01-14	24	14.4	346
华北	张家口	康毅系统	饮料	苹果汁	1997-03-17	15	14.4	216

图 2： Excel 透视表示例数据

想透过数据得到各年份分地区的销售额，这就是按年份、地区分组，对销售额做加和汇总。

- **方法一**：分组汇总 + 长变宽

```
df = read_xlsx("data/数据透视表.xlsx")
library(lubridate)
pt = df %>%
  group_by(年份 = year(订购日期), 地区) %>%
  summarise(销售额 = sum(销售额))
pt
#> # A tibble: 36 x 3
#> # Groups:   年份 [6]
#>    年份 地区   销售额
#>   <dbl> <chr>  <dbl>
#> 1  1996 东北   16984
#> 2  1996 华北   95935
#> 3  1996 华东   51792
#> # ... with 33 more rows
```

Excel 透视表一般不是这样的整洁长表，而是更具可读性的宽表，再对地区列来个长变宽：

```
pt = pt %>%
  pivot_wider(names_from = 地区, values_from = 销售额)
pt
#> # A tibble: 6 x 7
#> # Groups:   年份 [6]
#>    年份  东北   华北   华东   华南  西北  西南
#>   <dbl> <dbl>  <dbl>  <dbl>  <dbl> <dbl> <dbl>
#> 1  1996 16984  95935  51792  38761  1883 20950
#> 2  1997 36984 260084 141214 129679  7683 93889
#> 3  1998  2847 157555  99536  87778  2965 94574
#> # ... with 3 more rows
```

Excel 透视表一般还增加按行、按列的加和：

```
library(janitor)
pt %>%
  adorn_totals(where = c("row", "col"))   # 也可以只用一个
#>    年份   东北   华北   华东   华南  西北   西南   Total
#>    1996  16984  95935  51792  38761  1883  20950  226305
#>    1997  36984 260084 141214 129679  7683  93889  669533
#>    1998   2847 157555  99536  87778  2965  94574  445255
#>    1999    861  65997 160557 170917   178   6271  404781
#>    2000    861 110237  28557 181917   178  86271  408021
#>    2001 100461  12625 104357 117102   178 106326  441049
#>   Total 158998 702433 586013 726154 13065 408281 2594944
```

以上是为了展示中间结果，也可以四步管道操作直接到最终透视表。

方法二：tidyquant::pivot_table()

tidyquant 包是做量化金融的包，顺便实现了大部分的 Excel 函数，pivot_table 就是其中之一，它类似于 Excel 中做透视表，选择行分组、列分组的变量或表达式，以及汇总函数。

- 相当于将方法一中的前三步一步到位，只是要增加行和、列和，仍需要接 adorn_totals()：

```
library(tidyquant)
df %>%
  pivot_table(.rows = ~ YEAR(订购日期), .columns = 地区,
              .values = ~ SUM(销售额)) %>%
  adorn_totals(where = c("row", "col"))  # 结果同上（略）
```

方法三：pivottabler 包

pivottabler 包是专为做数据透视表而生，强大到无以复加：可以任意精细定制、导出各种格式。

```
library(pivottabler)
df %>%
  mutate(年份 = year(订购日期)) %>%
  qpvt(rows = " 年份", columns = " 地区",
       calculations = "sum(销售额)")   # 结果同上 (略)
```

默认增加行和、列和，相当于参数 totals = c(" 年份", " 地区")，可以只留一个，或 totals = "" 两个都不留。

数据透视表的进一步定制格式和美化导出可借助 openxlsx 包，或者在 Excel 中再做。

D.1 非等连接

通常的数据连接是相等连接，即只有所匹配列的值相等，才认为匹配成功，再将匹配成功的其他列连接进来。

很多时候需要非等连接，相当于按条件连接，即匹配列的值不要求必须相等，只要满足一定条件就认为是匹配成功，再将匹配成功的其他列连接进来。

```
library(data.table )
Houses = fread("data/Houses.csv")
Renters = fread("data/Renters.csv")
Deals = fread("data/Deals.csv")
```

```
Houses        # 房源信息
#>      id district           address bedrooms rent
#>   1:  1    South   Rose Street, 5        4 3000
#>   2:  2    North  Main Street, 12        3 2250
#> ---
#>   6:  6    South Little Street, 7        4 3000
#>   7:  7    North   Main Street, 8        3 2100
```

```
Renters      # 租客信息
#>    id          name preferred min_bedrooms min_rent max_
#> 1:  1    Helen Boss     South            3     2500
#> 2:  2  Michael Lane      West            2     1500
#> 3:  3 Susan Sanders      West            4     2500
#> 4:  4     Tom White     North            3     2200
#> 5:  5   Sofia Brown     North            3     1800
Deals        # 已租赁信息
#>    id       date renter_id house_id agent_fee
#> 1:  1 2020/1/30         1        1       600
#> 2:  2  2020/2/3         2        4       350
#> 3:  3 2020/3/12         3        5       700
#> 4:  4 2020/4/10         4        2       450
```

找出可以合租的人，即考虑租客具有相同的首选区域这是 Renters 表的自连接，根据首选区域进行相等匹配，通过非等匹配 (id $<$ id) 避免相同租客组合的重复出现：

```
Renters[Renters, on = .(preferred, id < id)][,
  .(name, preferred, i.name)
] %>% na.omit()
#>              name preferred        i.name
#> 1: Michael Lane       West Susan Sanders
#> 2:    Tom White      North   Sofia Brown
```

同样的操作，还可以用来识别重复，比如找出重复登记房屋（即地址相同）的情况，可以通过不同的房屋 id 来判断。

再比如，找出满足租客要求的可用房源，需要满足以下条件：

- 是租客的首选区域
- 在租客接受的价格范围内
- 卧室数量满足租客所需
- 房屋还没有被租赁（即不在 `Deals` 表中）。

以 Renters 为左表，连接表是未被租赁筛选后的 Houses 表，租客要求的条件体现在非等匹配中，即房屋地区等值匹配租客首选地区，房屋租金介于租客能接受的最低租金和最高租金之间，房屋卧室数大于租客需要的最小数值。

```
Houses[! id %in% Deals$house_id][
  Renters, on = .(district == preferred, rent >= min_rent,
                  rent <= max_rent, bedrooms >= min_bedrooms)
  ][, -(5:6)] %>% na.omit()
#>    id district           address bedrooms i.id          na
#> 1:  3    South   Rose Street, 5        3    1  Helen Bos
#> 2:  6    South Little Street, 7        3    1  Helen Bos
#> 3:  7    North   Main Street, 8        3    5 Sofia Brou
```

D.2 滚动连接

滚动连接往往涉及日期时间，常用来处理在时间上有先后关联的两个事件。基本语法如下：

```
x[y, on = .(id=id, date=date), roll = TRUE]   # 同 roll=inf
```

根据 id 等值匹配行，date 是滚动匹配，匹配与左表中 y 日期最接近的前一个日期，把匹配成功的列合并进来，roll = -inf 则用于匹配最接近的后一个日期。

```
website = fread("data/website.csv")
paypal = fread("data/paypal.csv")
# 为了便于观察, 增加分组 id 列
website[, session_id := .GRP, by = .(name, session_time)]
paypal[, payment_id := .GRP, by = .(name, purchase_time)]
website        # 网页会话数据
#>       name        session_time session_id
#>  1: Isabel 2016-01-01 03:01:00          1
#>  2: Isabel 2016-01-02 00:59:00          2
#> ---
#> 15: Vivian 2016-01-01 01:10:00         15
#> 16: Vivian 2016-01-08 18:15:00         16
```

```
paypal          # 支付数据
#>        name        purchase_time payment_id
#>  1: Isabel 2016-01-08 11:10:00          1
#>  2:  Sally 2016-01-03 02:06:00          2
#> ---
#>  7:  Erica 2016-01-03 00:02:00          7
#>  8:    Mom 2015-12-02 09:58:00          8
# 创建用于连接的单独时间列
website[, join_time:=session_time]
paypal[, join_time:=purchase_time]
# 设置 key
setkey(website, name, join_time)
setkey(paypal, name, join_time)
```

前滚连接：查看有哪些客户在每次支付前进行了网页会话

由于已经设置了键，故可以省略 on 语句。以 paypal 为左表，先等值匹配 id，若同一 id 的支付时间之前有网页会话时间，则匹配成功，合并右表的新列进来

```
website[paypal, roll = TRUE] %>% na.omit()
#>        name        session_time session_id           jo
#>  1: Francis 2016-01-03 11:22:00          8 2016-01-03 1
#>  2: Francis 2016-01-08 12:22:00         10 2016-01-08 1
#> ---
#>  5:   Sally 2016-01-03 02:00:00          6 2016-01-03 0
#>  6:   Sally 2016-01-03 02:00:00          6 2016-01-03 0
#>          purchase_time payment_id
#>  1: 2016-01-03 11:28:00          4
#>  2: 2016-01-08 12:33:00          5
#> ---
#>  5: 2016-01-03 02:06:00          2
```

后滚连接：哪些客户网页会话之后产生了支付

以 website 为左表，先等值匹配 id，若同一 id 在网页会话时间之后有支付时间，则匹配成功，把右表的新列合并进来

```
paypal[website, roll = -Inf] %>% na.omit()
#>        name       purchase_time payment_id           jo
#>  1: Francis 2016-01-03 11:28:00          4 2016-01-02 05
#>  2: Francis 2016-01-03 11:28:00          4 2016-01-03 11
#> ---
#> 10:  Isabel 2016-01-08 11:10:00          1 2016-01-08 11
#> 11:   Sally 2016-01-03 02:06:00          2 2016-01-03 02
#>             session_time session_id
#>  1: 2016-01-02 05:09:00          7
#>  2: 2016-01-03 11:22:00          8
#> ---
#> 10: 2016-01-08 11:01:00          5
#> 11: 2016-01-03 02:00:00          6
```

附录 E R 网络爬虫

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、解析并提取想要的数据、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。

E.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 3: 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	红楼梦	[清] 曹雪芹 著 / 人民文学出版社 / 1996-12 / 59.70元	9.6	343998	都云作者痴，谁解其中味？
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	栗树荫下，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....	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 4: 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[2]。

[2]注意直接根据/分割是不行的，作者不一定几个.

```r
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：

	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 5: 豆瓣读书 Top250 爬虫数据（已清洗）

E.2 `httr` 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。

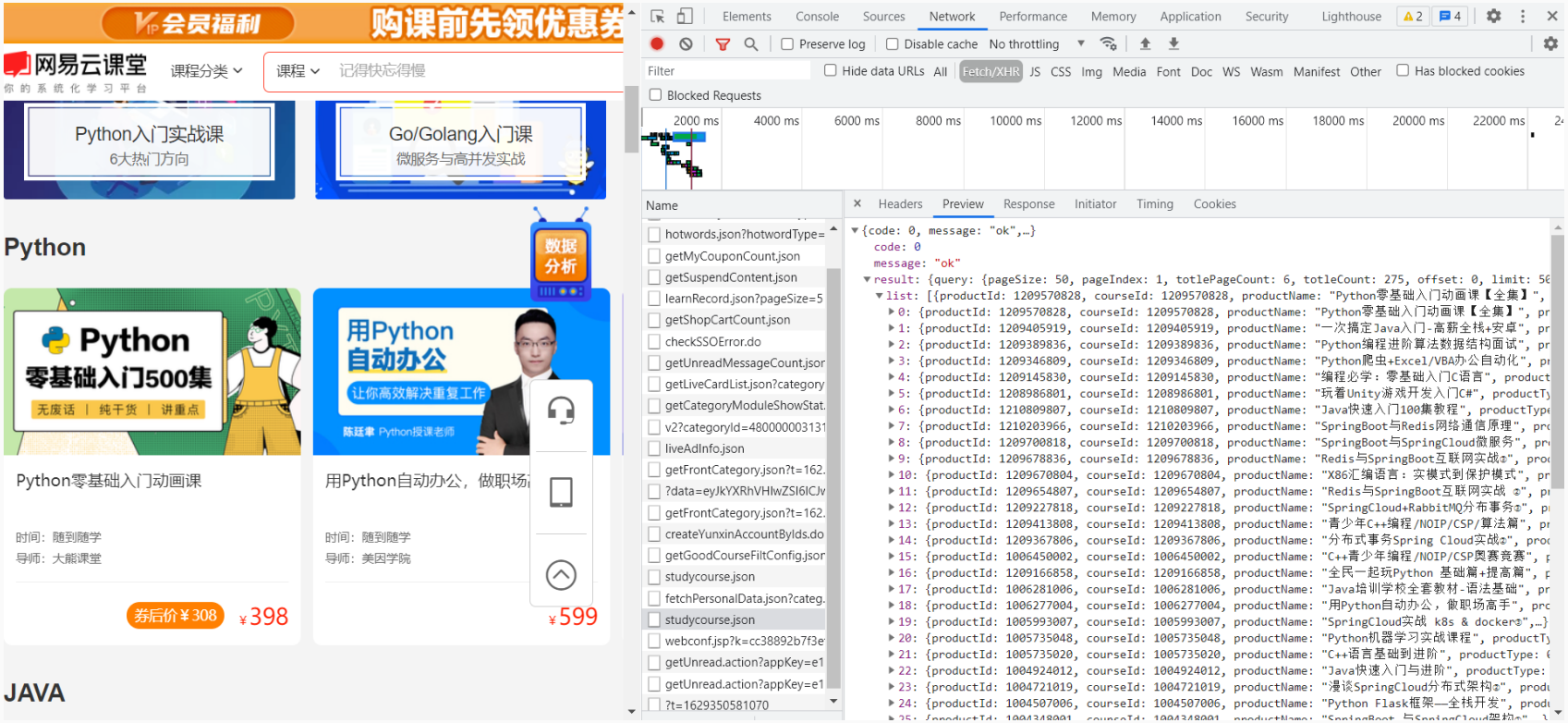
爬取动态网页就需要先发送请求，对请求到的结果再做解析、提取、保存，`rvest` 包就无能为力了。`RCurl` 包或者其简化版的 `httr` 包可以爬取动态网页。

案例：爬取网易云课堂编程与开发类课程

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。

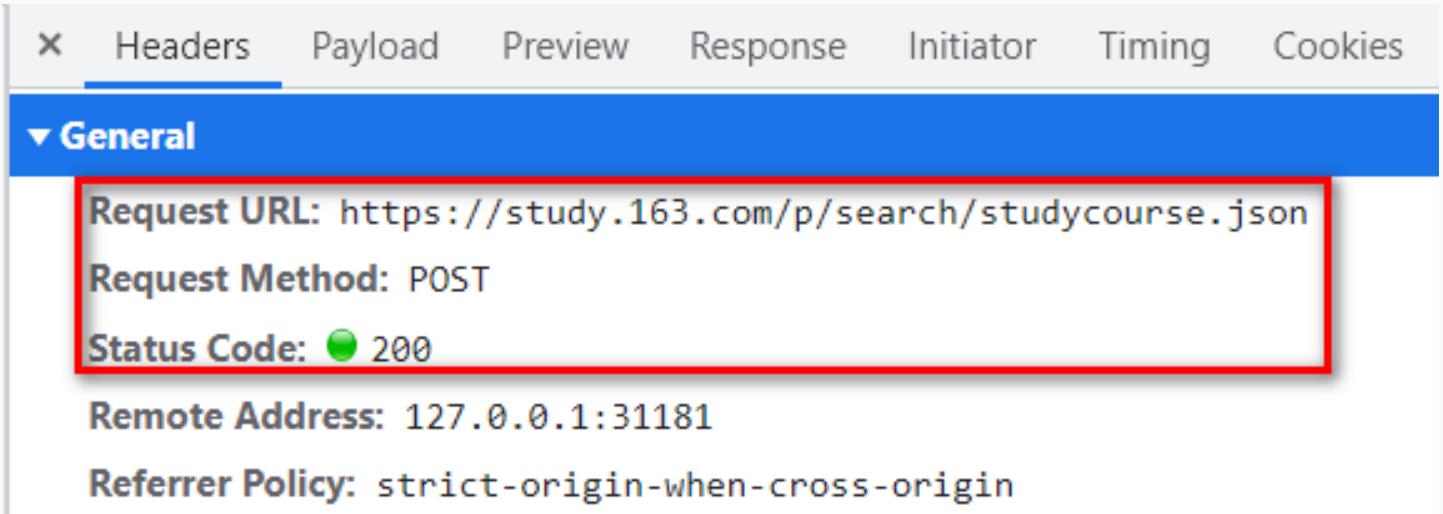
右键“检查”，依次点击“Network”，“fetch/HXR”，刷新网页，则右下窗口出现很多内容，浏览找到 `studycourse.json`，点开，在 `preview` 下可以找到想要抓取的内容：


会员福利
购课前先领优惠券
网易云课堂
你的系统化学习平台
课程分类
课程
记得快忘得慢
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
Python
零基础入门500集
无废话 | 纯干货 | 讲重点
用Python
自动办公
让你高效解决重复工作
陈廷聿 Python授课老师
Python零基础入门动画课
时间：随到随学
导师：大能课堂
券后价¥308
¥398
用Python自动办公，做职场
时间：随到随学
导师：美因学院
¥599
JAVA
Elements
Console
Sources
Network
Performance
Memory
Application
Security
Lighthouse
Preserve log
Disable cache
No throttling
Filter
Hide data URLs
All
Fetch/XHR
JS
CSS
Img
Media
Font
Doc
WS
Wasm
Manifest
Other
Has blocked cookies
Blocked Requests
2000 ms
4000 ms
6000 ms
8000 ms
10000 ms
12000 ms
14000 ms
16000 ms
18000 ms
20000 ms
22000 ms
Name
hotwords.json?hotwordType=
getMyCouponCount.json
getSuspendContent.json
learnRecord.json?pageSize=5
getShopCartCount.json
checkSSOError.do
getUnreadMessageCount.jsor
getLiveCardList.json?category
getCategoryModuleShowStat.
v2?categoryId=480000003131
liveAdInfo.json
getFrontCategory.json?t=162.
?data=eyJkYXRhVHlwZSI6ICJv
getFrontCategory.json?t=162.
createYunxinAccountByIds.do
getGoodCourseFiltConfig.jsor
studycourse.json
fetchPersonalData.json?categ.
studycourse.json
webconf.jsp?k=cc38892b7f3e
getUnread.action?appKey=e1
getUnread.action?appKey=e1
?t=1629350581070
Headers
Preview
Response
Initiator
Timing
Cookies
{code: 0, message: "ok",…}
code: 0
message: "ok"
result: {query: {pageSize: 50, pageIndex: 1, totlePageCount: 6, totleCount: 275, offset: 0, limit: 50
list: [{productId: 1209570828, courseId: 1209570828, productName: "Python零基础入门动画课【全集】",
0: {productId: 1209570828, courseId: 1209570828, productName: "Python零基础入门动画课【全集】", pr
1: {productId: 1209405919, courseId: 1209405919, productName: "一次搞定Java入门-高薪全栈+安卓", pr
2: {productId: 1209389836, courseId: 1209389836, productName: "Python编程进阶算法数据结构面试", pr
3: {productId: 1209346809, courseId: 1209346809, productName: "Python爬虫+Excel/VBA办公自动化", pr
4: {productId: 1209145830, courseId: 1209145830, productName: "编程必学：零基础入门C语言", product
5: {productId: 1208986801, courseId: 1208986801, productName: "玩着Unity游戏开发入门C#", productTy
6: {productId: 1210809807, courseId: 1210809807, productName: "Java快速入门100集教程", productType
7: {productId: 1210203966, courseId: 1210203966, productName: "SpringBoot与Redis网络通信原理", pro
8: {productId: 1209700818, courseId: 1209700818, productName: "SpringBoot与SpringCloud微服务", pro
9: {productId: 1209678836, courseId: 1209678836, productName: "Redis与SpringBoot互联网实战②", proc
10: {productId: 1209670804, courseId: 1209670804, productName: "X86汇编语言：实模式到保护模式", pr
11: {productId: 1209654807, courseId: 1209654807, productName: "Redis与SpringBoot互联网实战 ②", pr
12: {productId: 1209227818, courseId: 1209227818, productName: "SpringCloud+RabbitMQ分布事务②", pr
13: {productId: 1209413808, courseId: 1209413808, productName: "青少年C++编程/NOIP/CSP/算法篇", pr
14: {productId: 1209367806, courseId: 1209367806, productName: "分布式事务Spring Cloud实战②", proc
15: {productId: 1006450002, courseId: 1006450002, productName: "C++青少年编程/NOIP/CSP奥赛竞赛", p
16: {productId: 1209166858, courseId: 1209166858, productName: "全民一起玩Python 基础篇+提高篇", p
17: {productId: 1006281006, courseId: 1006281006, productName: "Java培训学校全套教材-语法基础", pr
18: {productId: 1006277004, courseId: 1006277004, productName: "用Python自动办公，做职场高手", pro
19: {productId: 1005993007, courseId: 1005993007, productName: "SpringCloud实战 k8s & docker②",…}
20: {productId: 1005735048, courseId: 1005735048, productName: "Python机器学习实战课程", productTy
21: {productId: 1005735020, courseId: 1005735020, productName: "C++语言基础到进阶", productType: 6
22: {productId: 1004924012, courseId: 1004924012, productName: "Java快速入门与进阶", productType:
23: {productId: 1004721019, courseId: 1004721019, productName: "漫谈SpringCloud分布式架构②", produ
24: {productId: 1004507006, courseId: 1004507006, productName: "Python Flask框架—全栈开发", produ

2. 构造请求 Headers，用 POST 方法请求网页内容

点开 Headers，重点关注：

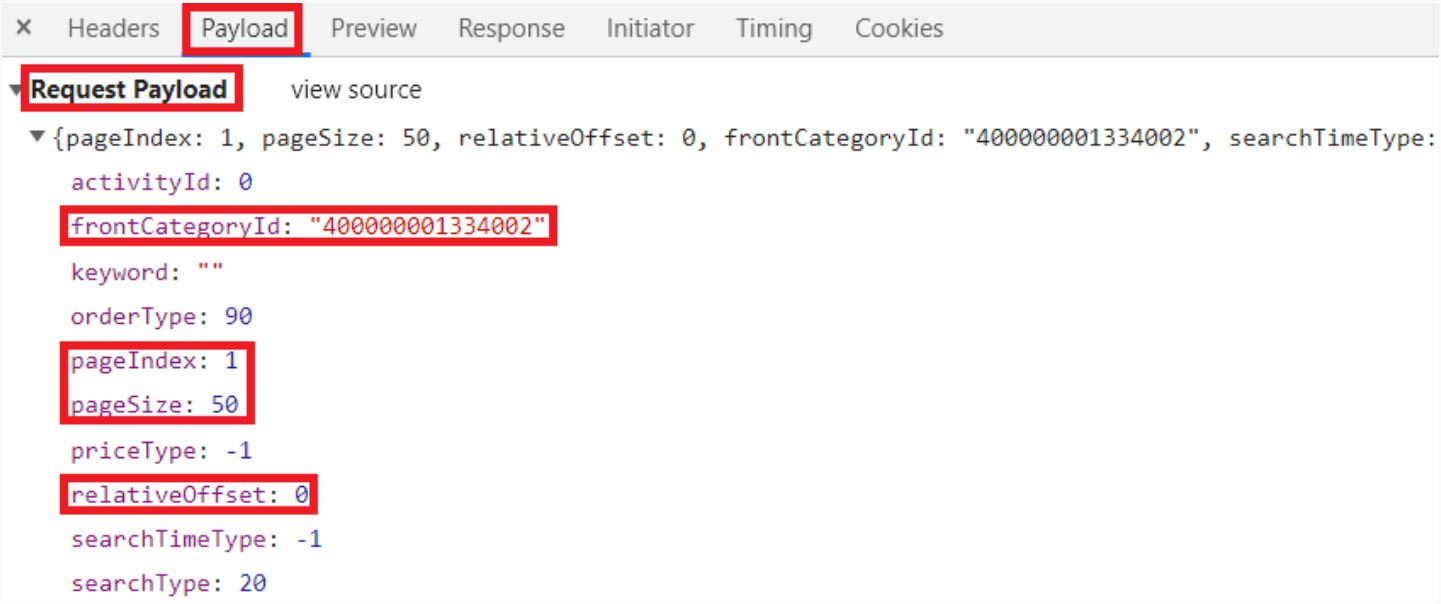
- General 下的：Request URL，Request Method，Status Code



- `Request Headers` 下的：accept, edu-script-token, cookie[3], user-agent

[3]注意，Cookie 代表您账号登录信息，是有时效性的.

- Request Payload 下的：pageIndex，pageSize，relativeOffset，rontCategoryId



获取这些信息之后，就可以在 R 中构造 Headers：

```r
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：

```r
# 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' = "48000000003131009")
  # 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：

	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	高淇 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 6: 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)，以及包文档，以及包文档，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*.