中国海关总署统计月报网站抓取

胡华平

2020-07-13

Table of Contents

[1.数据信息概要 3](#_Toc45545466)

[1.1资料来源： 3](#_Toc45545467)

[1.2指标和变量 3](#_Toc45545468)

[1.3原始数据形态 3](#_Toc45545469)

[2.数据整理过程 4](#_Toc45545470)

[2.1数据抓取策略 4](#_Toc45545471)

[2.2抓取环节1：中国海关总署统计月报 4](#_Toc45545472)

[2.2.1.下载并保存主站网页源代码 5](#_Toc45545473)

[2.2.2.自动抓取url的R代码实现 5](#_Toc45545474)

[2.2.3. 批量化下载url静态页面到本地 8](#_Toc45545475)

[2.2.4. 把html转换为对应的csv，并提取整合有效数据 11](#_Toc45545476)

# 1.数据信息概要

## 1.1资料来源：

中国海关总署《统计月报》。

* [中国海关总署](http://www.customs.gov.cn/customs/302249/302274/302277/index.html)
  + 文件标题：“2020年出口主要商品量值表”"2020年进口主要商品量值表"
  + 可下载年限范围：2014/6-2020/5
  + 文件格式：html-制式表格
* 搜数网[人大权限](https://libproxy.ruc.edu.cn/ermsClient/eresourceInfo.do?rid=136)
  + 搜数标题：“2000年5月中国出口主要商品量值表统计”/“2000年5月中国进口主要商品量值表统计”
  + 可下载年限范围：2000/1-2019/12
  + 文件格式：html-形式表格

## 1.2指标和变量

农产品类章进/出口量值：

* 指标（数据列）：数量/金额【按人民币】
* 样本单位（数据行）：商品类章
* 数据频率：月度

## 1.3原始数据形态

* 中国海关总署：html表格，在线网址
* 搜数网：下载的html静态页面

# 2.数据整理过程

## 2.1数据抓取策略

同时对两个数据源进行互补式抓取：

* 2014/6-2020/5：抓取中国海关总署，在线网址形式的html表格数据
* 2000/1-2014/5：抓取搜数网，下载好的html静态页面数据

对于中国海关总署网站数据，有两个抓取办法：

1. 理想方法：通过Rselenium+docker的办法自动化抓取到所需表格的所有月份页面的实际网址url，然后批量下载保存成静态页面。
2. 手工办法：进入海关网站，手动下载每个月份页面的实际静态网页到本地。工作量初步估计：进出口量值的页面数=7年x12月x2类=168

## 2.2抓取环节1：中国海关总署统计月报

基本过程：

1. 以源文件方式保存[主站](http://www.customs.gov.cn/customs/302249/302274/302277/index.html)。
2. 第一步是需要获得指定“主题”（例如“进口主要商品量值表”）下所有月份的实际网址url。具体细节见后面的R代码（见节2.2.2）。
3. 根据获取的实际url，下载全部静态表格网页到本地，并批量命名。
4. 通过下载所有月份的静态网页html（含有数据表），然后再进行数据整合。

主要关注如下几类网页表：

* 表4：**“进出口商品类章总值表”**
* 表13/14：**“出口/进口主要商品量值表”**。主要商品较具体，重点关注：肉及杂碎（其子类中的牛肉、猪肉、羊肉）、谷物及谷物粉（其子类中的稻谷和大米、玉米、豆类）等。
* 表15/16：**“对部分国家（地区）出口/进口商品类章金额表”**。按类章、并区分与部分国家的进出口贸易关系。

### 2.2.1.下载并保存主站网页源代码

注意： > - 具体操作：chrome浏览器打开网址 查看源代码 复制所有源代码到 notepad++ 保存为UTF-8格式下的txt文件。 > - 说明：因为网站采用了js框架，不能直接静态拉取页面的所有信息（例如read\_html()）。而采用动态拉取方法也会失效（例如RSelenium + docker的方式）。

### 2.2.2.自动抓取url的R代码实现

我们可以直接分析网页的源代码文件[[1]](#footnote-1)。

**说明**：“2015年进口主要商品量值表” 1月页面链接无效

以下为R代码具体实现过程：

# step 1: save the source code web as .txt file  
  
# step 2: read the txt file with correct encoding type  
the\_page <- readLines("00-web-hack/page-all-index.txt", encoding = "UTF-8")  
  
# step 3: specify the needed table title  
the\_title <- "进口主要商品量值表"  
  
# step 4: create the whole url scrape function  
  
get\_url <- function(page=the\_page, title=the\_title){  
 # 4.1 locate the line number  
 id\_currency <- grep("表名 （", page)  
 id\_head <- grep(title, page)  
 id\_url <- id\_head+1  
 # 4.2 choose the useful lines  
 tbl\_choose <- tibble(id\_currency = grep("表名 （", page),  
 id\_head = grep(title, page),  
 id\_url = id\_head+1,  
 txt\_currency=page[id\_currency],  
 txt\_head= page[id\_head],  
 txt\_url=page[id\_url])  
 # 4.3 extract the currency/year/variables  
 tbl\_extract <- tbl\_choose %>%  
 # extract currency  
 mutate(currency= str\_extract\_all(txt\_currency, "(?<=（)(.+)(?=）)")) %>%  
 mutate(currency= str\_replace(currency, " ", "")) %>%  
 # extract year  
 mutate(year=as.numeric(str\_extract\_all(txt\_head, "(?<=\\))(\\d{4})"))) %>%  
 # extract variables  
 mutate(variables=str\_extract(txt\_head, "进口|出口|进出口"))  
 # 4.4 map function which we can extract the month and its real url  
 extract\_url <- function(txt){  
 text\_tbl <- tibble(text=unlist(str\_split(txt,pattern ="</a>"))) %>%  
 filter(str\_detect(text,"月")) %>%  
 mutate(text=str\_replace\_all(text,"><a", "<a")) %>%  
 # extract month  
 mutate(month=str\_extract(text, pattern = "(?<=>)(.+)(?=月)")) %>%  
 # extract url  
 mutate(url=str\_extract(text, pattern = "(?<=http://)(.\*.html)"))  
 return(text\_tbl)  
 }  
 # 4.5 map and unnest   
 tbl\_map <- tbl\_extract %>%  
 mutate(tbl=map(.x=txt\_url, .f = extract\_url )) %>%  
 unnest(cols = tbl) %>%  
 select(currency,variables,year,month, url )   
   
   
 return(tbl\_map)  
}  
  
# tbl\_show <- get\_url(title = "出口主要商品量值表")

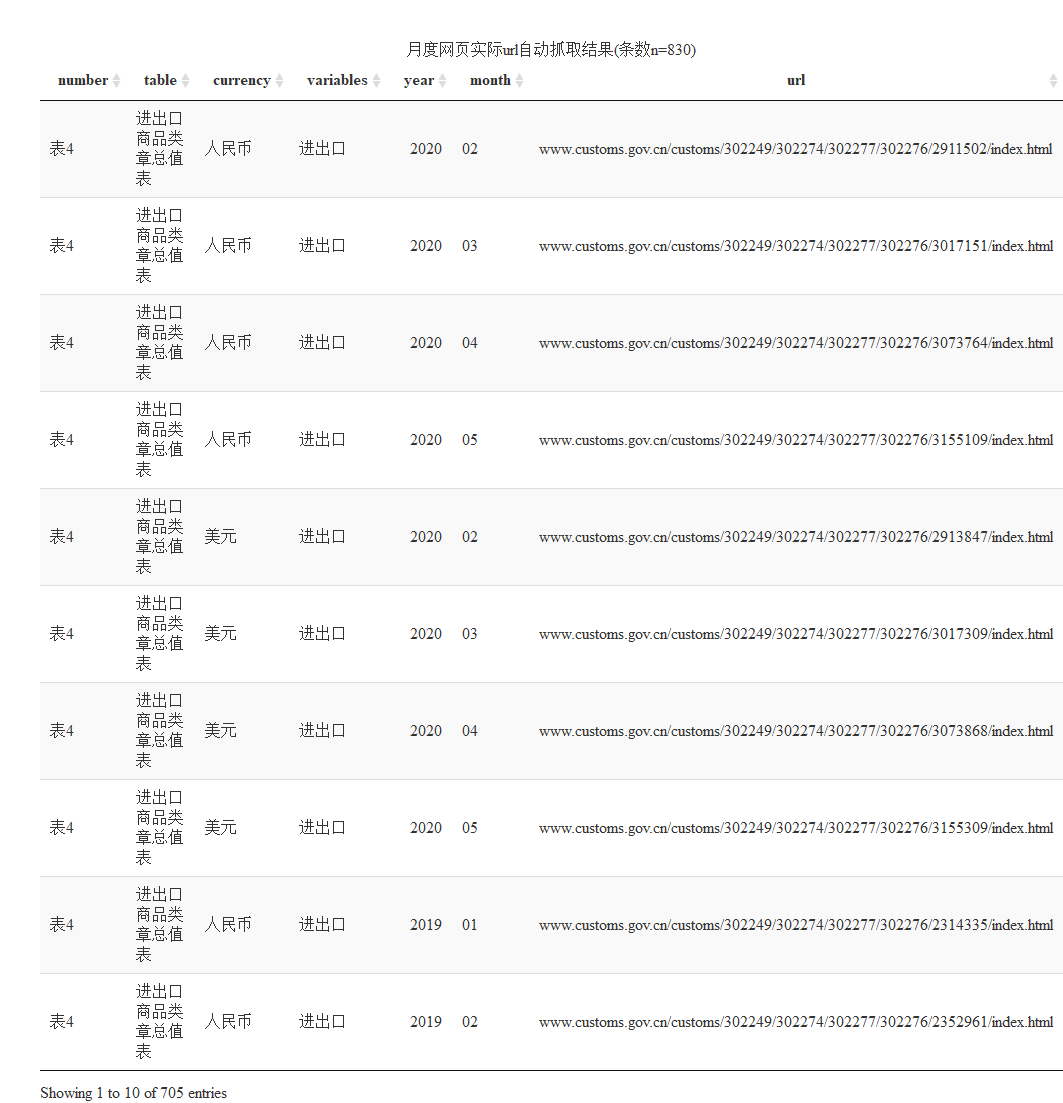
下面，我们把所有感兴趣的主体的url地址全部抓取下来：

# set topic we need  
all\_title <- c("进出口商品类章总值表", # 表4   
 "出口主要商品量值表", # 表13  
 "进口主要商品量值表", # 表14  
 "出口商品类章金额表", # 表15  
 "进口商品类章金额表") # 表16  
number\_tbl <- paste0("表", c(4,13:16))  
# loop for all table  
### i <-1  
tbl\_all <- NULL  
for (i in 1:length(all\_title)){  
 tbl\_show <- get\_url(page=the\_page, title = all\_title[i]) %>%  
 add\_column(table=all\_title[i], .before = "currency") %>%  
 add\_column(number=number\_tbl[i], .before = "table")   
 tbl\_all <- bind\_rows(tbl\_all, tbl\_show)  
 print(number\_tbl[i])  
}

## [1] "表4"  
## [1] "表13"  
## [1] "表14"  
## [1] "表15"  
## [1] "表16"

tbl\_all <- tbl\_all %>%  
 # handle the month with leading zero  
 mutate(month= str\_replace(month, "1-2", "2")) %>%  
 mutate(month=str\_trim(month, side = "both")) %>%  
 mutate(month=str\_pad(month, width = 2, pad = "0"))

抓取的最后结果如下：



最后我们把抓取结果保存在本地：

# write out files  
file\_name<- paste0("00-web-hack/auto-url-",Sys.Date(),".csv")  
write.csv(tbl\_all, file = file\_name, row.names = FALSE)

### 2.2.3. 批量化下载url静态页面到本地

**第一步**：读取并整理所有url。

# set topic we need  
all\_title <- c("进出口商品类章总值表", # 表4   
 "出口主要商品量值表", # 表13  
 "进口主要商品量值表", # 表14  
 "出口商品类章金额表", # 表15  
 "进口商品类章金额表") # 表16  
number\_tbl <- paste0("表", c(4,13:16))  
  
# read the csv file on all urls  
tbl\_all <- read.csv("00-web-hack/auto-url-2020-07-13.csv")  
  
title\_match <- tibble(file\_dir = c("01-tbl04-cat",   
 "02-tbl13-cat-export","03-tbl14-cat-import",  
 "04-tbl15-cat-country-export","05-tbl16-cat-country-import"),  
 table = all\_title)  
  
# tidy url and create file path for each url  
tbl\_urls <- tbl\_all %>%   
 dplyr::filter(!is.na(url)) %>% # filter NULL urls  
 dplyr::filter(currency=="人民币") %>% # only need the RMB table  
 mutate(url=paste0("http://", url)) %>% # fill the actual urls  
 # for the name with leading zero  
 mutate(month=str\_pad(month, width = 2, pad = "0")) %>%   
 # the file name  
 mutate(file\_name = paste0(year, "-", month ,".html")) %>%  
 # the dir name  
 left\_join(x=., y=title\_match, by="table") %>%  
 # the file path  
 mutate(file\_path = paste0(file\_dir, "/html/", file\_name))

**第二步**：利用docker+RSelenium下载所有url对应的静态html到本地。

* 环境准备。安装docker desktop，进行配置（主要是registry mirror设置为国内服务器；确保bios开启虚拟化服务）。拉取pull安装selenium/chrome以及firefox到本地。
* 设计Rselenium包的server-client交互过程。docker启用container时，浏览器的选择很重要，chrome无法进入网页，但是firefox可以。
* 利用其他R包配合，自动化下载静态html到本地硬盘。其中的关键就是利用RSelenium读取源文件，然后用xml2::read\_html()转化为html，最后再由xml2::write\_xml()写出到本地。

说明：

* “表14：2015年进口主要商品量值表（人民币）1月份”的url为空。

具体R代码如下：

# 0. load R pkgs  
library("RSelenium")  
library("xml2")  
  
# 1. run docker service and container  
#### you should run and start docker desktop first.  
#### then run code in 'window power shell': docker run -d -p 4445:4444 selenium/standalone-firefox  
  
# 2. create the remote driver  
remDr <- remoteDriver(remoteServerAddr = "192.168.31.135", port = 4445L,   
 browserName = "firefox")  
## open the connect  
remDr$open()  
  
# you should set the table number to download all html for each url  
url\_list <- tbl\_urls %>%  
 filter(number == "表13")   
  
# i<- 2  
  
for (i in 1:nrow(url\_list) ){  
 # navigate the url  
 remDr$navigate(url\_list$url[i])  
 # wait seconds  
 Sys.sleep(3)  
 # confirm you got there  
 # print(remDr$getTitle())  
 # check on there  
 #remDr$screenshot(display = TRUE)  
 # get the page source  
 webpage <- remDr$getPageSource()[[1]]  
 # wait seconds  
 Sys.sleep(2)  
 # read as html object  
 html\_obj <- xml2::read\_html(webpage)  
 # write out the html file  
 xml2::write\_xml(html\_obj, file=url\_list$file\_path[i])  
}  
  
#close the driver  
remDr$close()  
  
#close the server  
remDr$server$stop()

### 2.2.4. 把html转换为对应的csv，并提取整合有效数据

具体的R操作在每一类表（表4、表13、……）的文件夹里【也可见视频 step3】。

**第一步**：利用R编程，自动把本地的静态html文件，分别转换为csv文件（xlsx就能打开查看的。）。

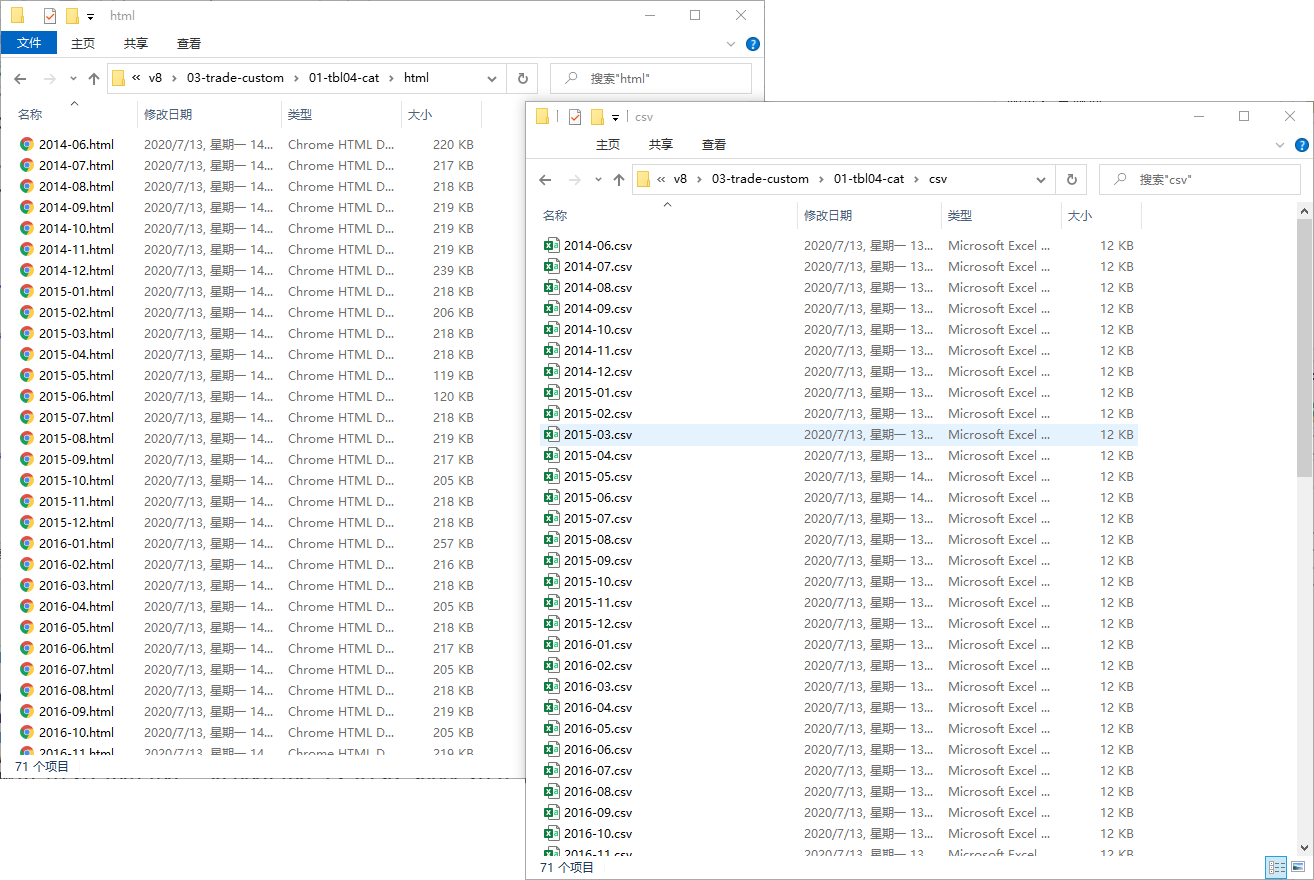


Figure 1: 把所有html文件分别转换成对应的csv文件

**第二步**：从所有的csv文件中提取有用的信息。合并并保存到本地csv。

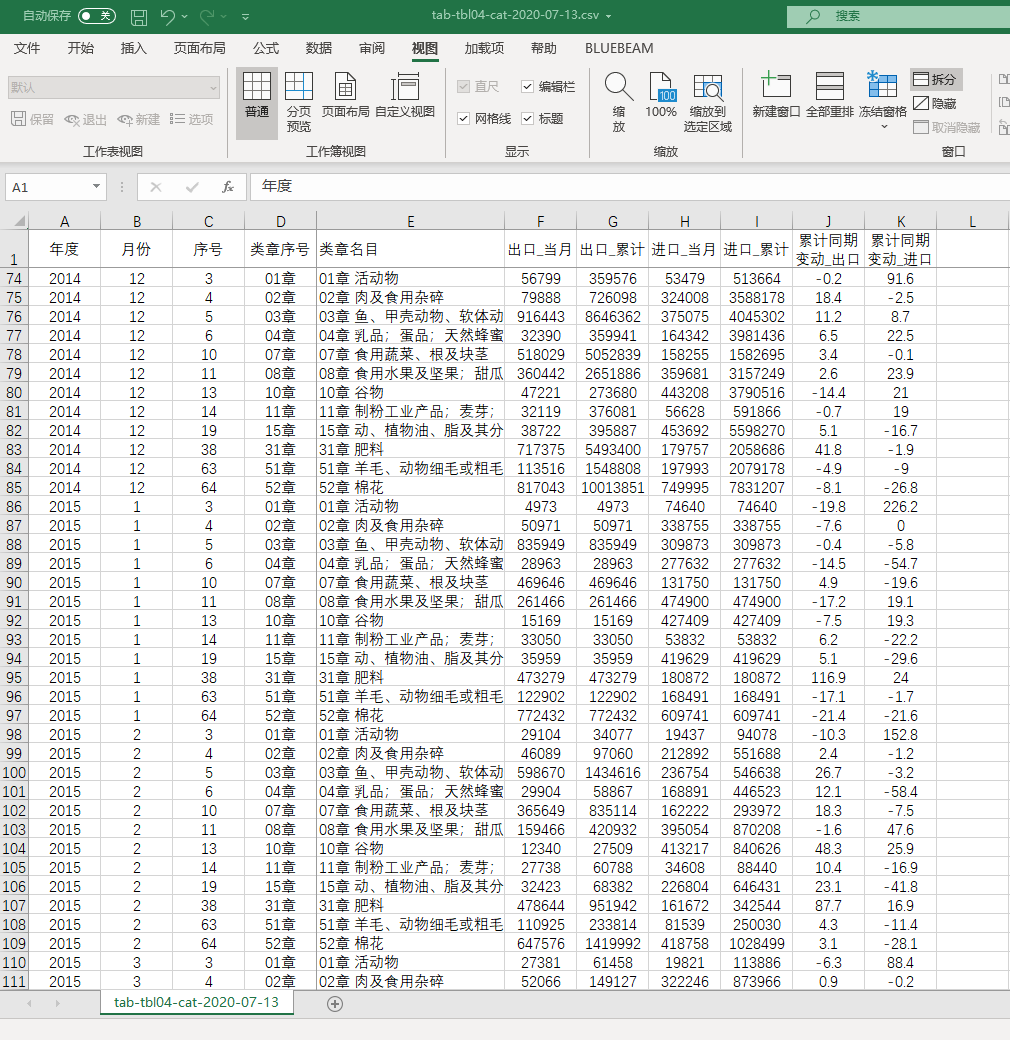


Figure 2: 提取csv文件中的有用数据，并整合成一个数据文件

1. 动态化抓取的办法见[参考资料](https://statistics.berkeley.edu/computing/r-reading-webpages) [↑](#footnote-ref-1)