Stata网页表格爬取示例

程振兴

2018年10月4日

本文以爬取东方财富网CPI数据为例，讲解如何使用Stata进行网页表格数据爬取。

Stata虽非数据爬取利器，但是能够轻松解决一些小的数据爬取任务。数据爬取的本质无非是数据请求和数据处理，因此熟练使用Stata进行数据爬取往往也是很好的数据处理能力的象征。在实际应用中，我们经常需要爬取一些公开数据。这些数据一种常见展示方式是通过HTML表示呈现。一个简单的HTML表格示例如下(这个表格的第一行的三个格子应该分别是红黄蓝色的，PDF文件没能显示出来)：

1	2	3
1	2	3

HTML代码为：

```
<table>
    <tr><td bgcolor="red">1</td><td bgcolor="yellow">2</td><td bgcolor="blue">3<
    <tr><td>1</td><td>2</td><td>3</td></tr>
</table>
```

东方财富网CPI数据的表格是这样的：

月　份	全国				城市				农村			
	当月	同比增长	环比增长	累计	当月	同比增长	环比增长	累计	当月	同比增长	环比增长	累计
2018年08月份	102.3	2.3%	0.7%	102.0	102.3	2.3%	0.6%	102.0	102.3	2.3%	0.8%	102.0
2018年07月份	102.1	2.1%	0.3%	102.0	102.1	2.1%	0.4%	102.0	102.0	2.0%	0.1%	101.9
2018年06月份	101.9	1.9%	-0.1%	102.0	101.8	1.8%	0.0%	102.0	101.9	1.9%	-0.1%	101.9
2018年05月份	101.8	1.8%	-0.2%	102.0	101.8	1.8%	-0.2%	102.0	101.7	1.7%	-0.1%	101.9
2018年04月份	101.8	1.8%	-0.2%	102.1	101.8	1.8%	-0.2%	102.1	101.7	1.7%	-0.3%	101.9
2018年03月份	102.1	2.1%	-1.1%	102.1	102.1	2.1%	-1.1%	102.2	101.9	1.9%	-1.2%	102.0
2018年02月份	102.9	2.9%	1.2%	102.2	103.0	3.0%	1.3%	102.2	102.7	2.7%	1.1%	102.1
2018年01月份	101.5	1.5%	0.6%	101.5	101.5	1.5%	0.6%	101.5	101.5	1.5%	0.6%	101.5
2017年12月份	101.8	1.8%	0.3%	101.6	101.9	1.9%	0.3%	101.7	101.7	1.7%	0.4%	101.3
2017年11月份	101.7	1.7%	0.0%	101.5	101.8	1.8%	0.0%	101.6	101.5	1.5%	0.0%	101.2
2017年10月份	101.9	1.9%	0.1%	101.5	101.9	1.9%	0.1%	101.6	101.7	1.7%	0.2%	101.2
2017年09月份	101.6	1.6%	0.5%	101.5	101.7	1.7%	0.5%	101.6	101.4	1.4%	0.6%	101.1
2017年08月份	101.8	1.8%	0.4%	101.5	101.9	1.9%	0.4%	101.6	101.5	1.5%	0.5%	101.1
2017年07月份	101.4	1.4%	0.1%	101.4	101.5	1.5%	0.1%	101.5	101.0	1.0%	0.0%	101.0
2017年06月份	101.5	1.5%	-0.2%	101.4	101.7	1.7%	-0.1%	101.5	101.0	1.0%	-0.2%	101.0
2017年05月份	101.5	1.5%	-0.1%	101.4	101.7	1.7%	-0.1%	101.5	101.1	1.1%	-0.1%	101.1
2017年04月份	101.2	1.2%	0.1%	101.4	101.3	1.3%	0.1%	101.5	100.8	0.8%	0.0%	101.1
2017年03月份	100.9	0.9%	-0.3%	101.4	101.0	1.0%	-0.3%	101.5	100.6	0.6%	-0.4%	101.1
2017年02月份	100.8	0.8%	-0.2%	101.7	100.9	0.9%	-0.2%	101.8	100.6	0.6%	-0.1%	101.4
2017年01月份	102.5	2.5%	1.0%	102.5	102.6	2.6%	1.0%	102.6	102.2	2.2%	0.9%	102.2

上一页 1 2 3 4 5 ... 7 下一页 转到 Go

下面我将一步步讲解如何爬取这个表格。

准备工作

1. Stata14.0 以上版本的；
2. Chrome浏览器；
3. 想把数据爬下来的你。

网页分析

首先讲解如何爬取一个页面的表格。这个网页的网址是：

http://data.eastmoney.com/cjsj/cpi.html

月 份	全国				城市				农村			
	当月	同比增长	环比增长	累计	当月	同比增长	环比增长	累计	当月	同比增长	环比增长	累计
2018年08月份	102.3	2.3%	0.7%	102.0	102.3	2.3%	0.6%	102.0	102.3	2.3%	0.8%	102.0
2018年07月份	102.1	2.1%	0.3%	102.0	102.1	2.1%	0.4%	102.0	102.0	2.0%	0.1%	101.9
2018年06月份	101.9	1.9%	-0.1%	102.0	101.8	1.8%	0.0%	102.0	101.9	1.9%	-0.1%	101.9
2018年05月份	101.8	1.8%	-0.2%	102.0	101.8	1.8%	-0.2%	102.0	101.7	1.7%	-0.1%	101.9
2018年04月份	101.8	1.8%	-0.2%	102.1	101.8	1.8%	-0.2%	102.1	101.7	1.7%	-0.3%	101.9
2018年03月份	102.1						-1.1%	102.2	101.9	1.9%	-1.2%	102.0
2018年02月份	102.9						1.3%	102.2	102.7	2.7%	1.1%	102.1
2018年01月份	101.5						0.6%	101.5	101.5	1.5%	0.6%	101.5
2017年12月份	101.8						0.3%	101.7	101.7	1.7%	0.4%	101.3
2017年11月份	101.7						0.0%	101.6	101.5	1.5%	0.0%	101.2
2017年10月份	101.9						0.1%	101.6	101.7	1.7%	0.2%	101.2
2017年09月份	101.6						0.5%	101.6	101.4	1.4%	0.6%	101.1
2017年08月份	101.8						0.4%	101.6	101.5	1.5%	0.5%	101.1
2017年07月份	101.4						0.1%	101.5	101.0	1.0%	0.0%	101.0
2017年06月份	101.5						-0.1%	101.5	101.0	1.0%	-0.2%	101.0
2017年05月份	101.5						-0.1%	101.5	101.1	1.1%	-0.1%	101.1
2017年04月份	101.2	1.2%	0.1%	101.4	101.3	1.3%	0.1%	101.5	100.8	0.8%	0.0%	101.1
2017年03月份	100.9	0.9%	-0.3%	101.4	101.0	1.0%	-0.3%	101.5	100.6	0.6%	-0.4%	101.1
2017年02月份	100.8	0.8%	-0.2%	101.7	100.9	0.9%	-0.2%	101.8	100.6	0.6%	-0.1%	101.4
2017年01月份	102.5	2.5%	1.0%	102.5	102.6	2.6%	1.0%	102.6	102.2	2.2%	0.9%	102.2

上一页 1 2 3 4 5 ... 7 下一页 转到 Go

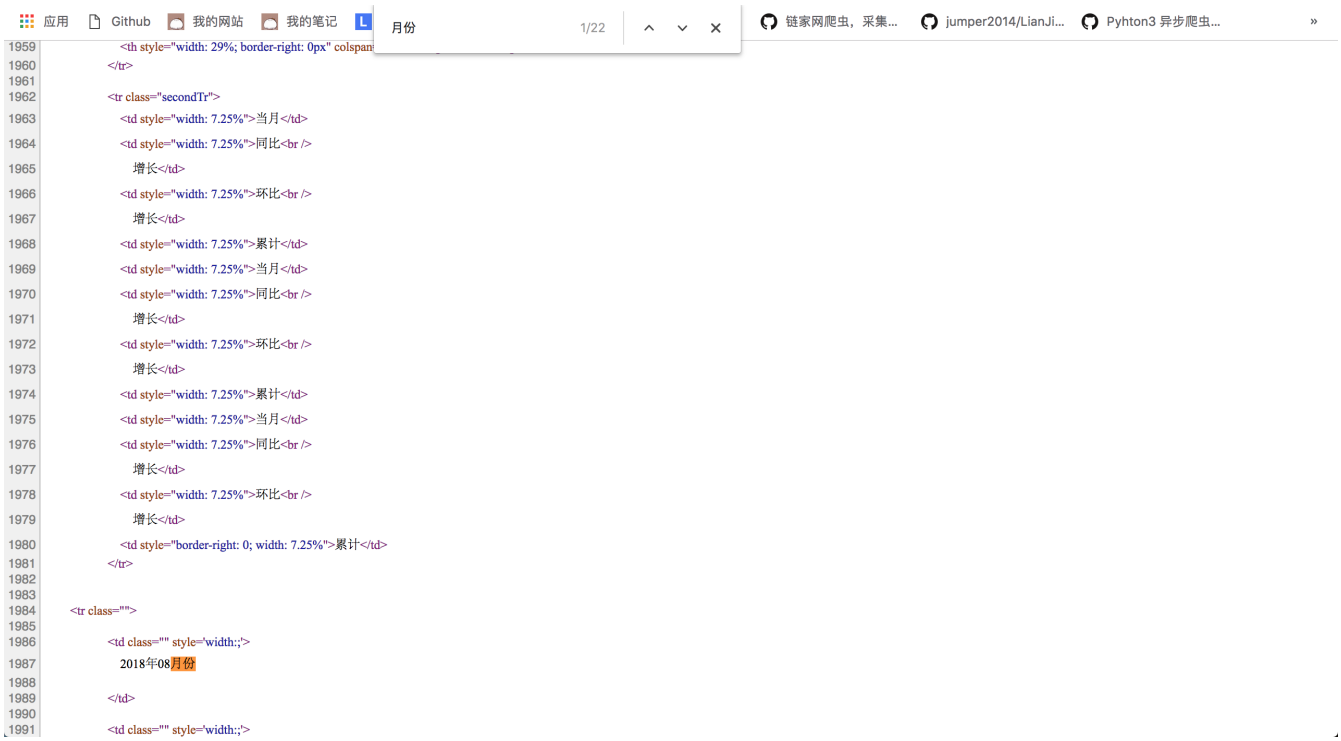
在页面上右键选择显示网页源代码，很多浏览器都有查看网页源代码的功能，但是我还是最喜欢谷歌浏览器的。点击之后即可跳转至网页源代码界面：

```
1
2
3  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-t
4  <!--published at 2018-10-02 18:15:49-->
5  <html xmlns="http://www.w3.org/1999/xhtml">
6  <head>
7      <meta http-equiv="Content-Type" content="text/html; charset=gb2312" />
8      <title>居民消费价格指数（CPI） _ 数据中心 _ 东方财富网</title>
9      <base target="_blank" />
10     <script type="text/javascript" src="http://emcharts.dfcfw.com/ec/2.5.2/emcharts.min.js" charset="utf-8"></script>
11     <script src="/js/chart_common.js"></script>
12     <script type="text/javascript">
13         var swf_line = "http://g1.dfcfw.com/g1/201012/20101214085507.swf";
14         var swf_pie = "http://g1.dfcfw.com/g1/201104/20110412125826.swf";
15         var swf_column = "http://g1.dfcfw.com/g1/201104/20110412130313.swf";
16     </script>
17
18 </head>
19 <body>
20
21 <link href="/css/MacroData/ss1.css" rel="stylesheet" type="text/css" />
22 <link href="../css/page_zjlxnew.css?201606021831" rel="stylesheet" type="text/css" />
23 <script src="/js/MacroData/jquery-1.7.js" type="text/javascript"></script>
24 <script src="/js/MacroData/JScript1.js" type="text/javascript"></script>
25
26 <!-- 微信分享img --><img id='weixin-share' src="http://wap.eastmoney.com/weixin-share.png" />
27 <div id="header">
28     <script> var NavCache = { Page: "经济数据", current_n: 6 };</script>
29
30
31 <link rel="shortcut icon" type="image/ico" href="http://www.eastmoney.com/favicon.ico" />
32 <link rel="stylesheet" type="text/css" media="all" href="/css/default.css?rt=20160616" />
33 <link rel="stylesheet" type="text/css" media="all" href="/css/layer2012.css?rt=20160721" />
34 <link href="/css_001/header950.css" rel="stylesheet" />
35 <script type="text/javascript" src="http://cmsjs.eastmoney.com/channel/jquery-1.8.3.min.js?rt=20151113"></script>
36 <script type="text/javascript">
37     $.noConflict();
38 </script>
39 <script src="http://emcharts.dfcfw.com/suggest/stocksuggest2017.min.js" charset="utf-8"></script>
40 <script type="text/javascript" src="/js_001/base.js?201806021831"></script>
41 <script type="text/javascript" src="/js_001/pluginNoBind.js?201806021831"></script>
42 <script type="text/javascript" src="/js/layer2012.js?rt=201410711"></script>
43 <script type="text/javascript" src="/js_001/ht_web.js"></script>
```

也就是说你刚刚看到的网页的本质实际上是这些源代码，之所以我们能看到各种炫彩的页面，那是因为浏览器帮我们翻译了源代码。

下一步我们要做的事情就是找到这个表格在源代码的哪一块儿了，然后分析表格的特点，已方便在后面的Stata处理源代码的时候进行过滤。

一个经常被用来寻找目标的方法是使用 `查找` 功能。Ctr+F（Mac是Command + F）即可打开搜索框，我们注意到表格里面有 `月份` 两个字，所以我们就用 `月份` 进行查找。



我们首先在第1987行发现了这个词，仔细一看，这附近的代码就是表格的代码。

下一步就是我们先分析一下这部分代码的特点：

1. 所有表格中的数据在源代码中都是单独位于一行的，所以我们不能从其所在行入手了；
2. 表格数据的上一行要么有字符串 `<td class=` 要么有 `<span` 。也可以发现，span标签是控制文字颜色的。

经过以上的网页分析我们就可以开始进行网页表格爬取了。

开始爬取

总的来说，Stata进行网页表格爬取分为3个步骤：

1. 请求：把含有所需数据的源代码下载下来；
2. 转码：很多网页并非使用UTF-8编码，直接读入Stata会出现乱码，因此可以预先进行UTF-8转码；
3. 处理：这里主要是对字符串进行处理，常用操作有分割(split)、转置(sxpose)、提取(正则表达式或直接字符串提取)等等。

请求

由于这个网页没有设置反爬机制，所以可以直接使用 `copy` 命令进行下载，copy命令不仅可以下载网页，还可以下载文件（当然网页其实就是一个html文件）。更多用法可以 `help copy` 。

我们这里把要爬取的页面保存成一个名叫 `temp.txt` 的txt文件，为什么要起这个名字呢，因为爬完之后它就要被删除了，所以只是一个临时的文件。

```
clear all
/* 设定工作目录 */
cd "你自己的工作目录（一个文件夹的路径）"
copy "http://data.eastmoney.com/cjsj/pmi.html" temp.txt, replace
```

转码

在我的Stata命令包——finance包中，我编写了一个简单的转码命令，这个命令包的安装办法为：

```
/* 首先你需要安装github命令，这个命令是用来安装github上的命令的 */
* net install github, from("https://haghish.github.io/github/")
/* 然后就可以安装这个命令了 */
* github install czxa/finance, replace
```

安装成功之后，使用下面的命令就可以直接对temp.txt文件进行转码了：

```
utrans temp.txt
```

如果返回的结果是 `转码成功` ，则表示 `转码成功` 了！（感觉像在说废话。。。）

如果你不幸的因为各种各样的原因没能成功安装这个小命令，可以直接使用下面三句命令进行转码：

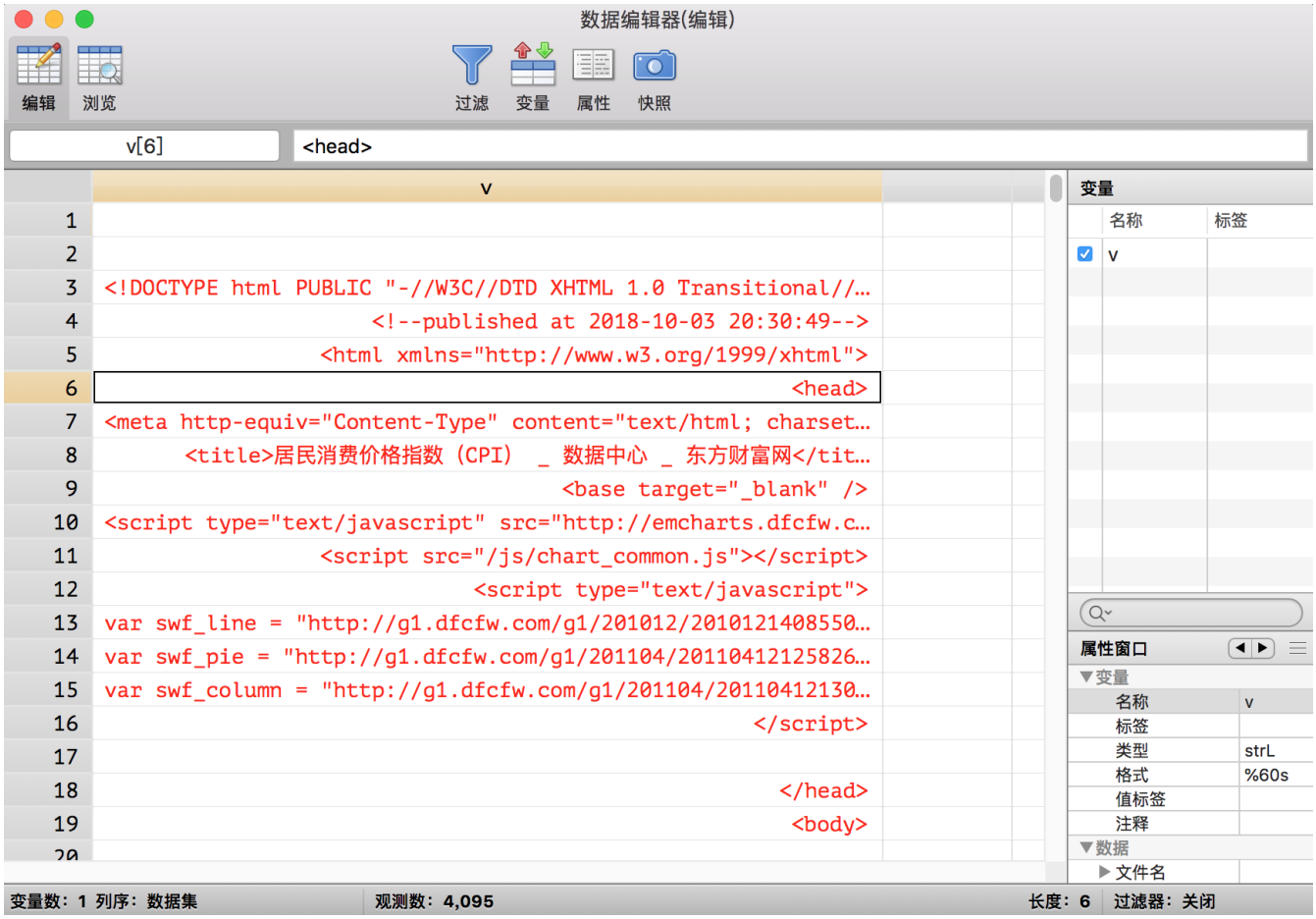
```
unicode encoding set gb18030
unicode translate temp.txt
unicode erasebackups, badidea
```

读入

下面我们就要把temp.txt文件读入Stata进行处理了，一个非常常用的读取方法是使用infix命令：

```
infix strL v 1-20000 using temp.txt, clear
* 把变量v的显示格式变成 %60s （这样看起来更宽）
format v %60s
```

这句命令的含义是创建一个格式为strL的变量v，然后把temp.txt文件的每一行的前1-20000个字符（因为我们注意到temp.txt的每一行都没有超过20000个字符）读入变量v的每一个观测值。读入之后是这样的：



处理

首先根据上面我们进行网页分析发现的结论，我们保留符合表格数据的上一行要么有字符串"<td class="要么有"<span"规律的观测值：

```
keep if index(v[_n-1], "<td class=") | index(v[_n-1], "<span")
/* 再删除空观测值 */
drop if v == ""
/* 再删除一些显然不是表格中数据的观测值，而这些无用观测值中都含有斜线 */
drop if index(v, "/")
```

这一步处理后的结果：

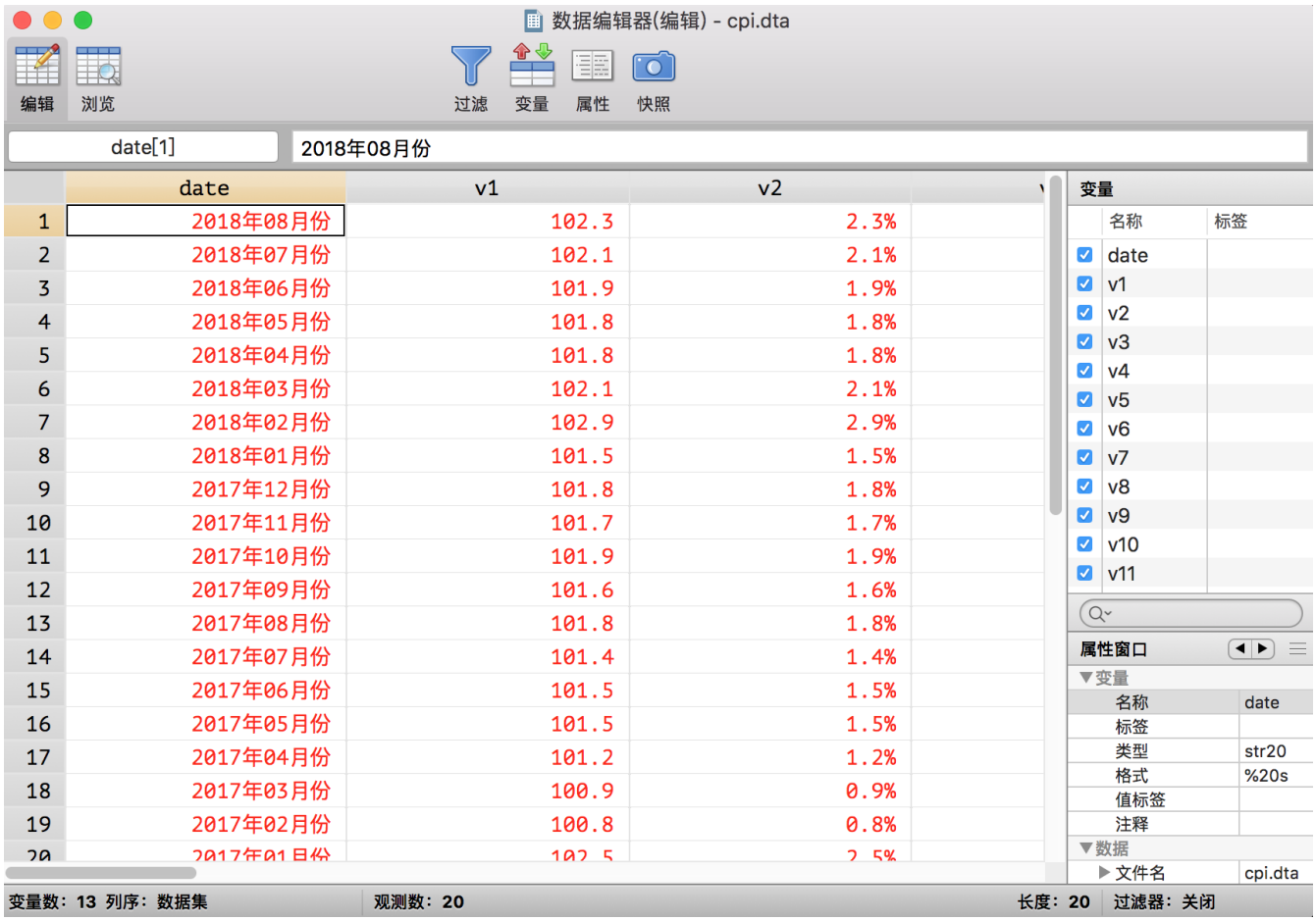
	v
1	2018年08月份
2	102.3
3	2.3%
4	0.7%
5	102.0
6	102.3
7	2.3%
8	0.6%
9	102.0
10	102.3
11	2.3%
12	0.8%
13	102.0
14	2018年07月份
15	102.1
16	2.1%
17	0.3%
18	102.0
19	102.1
20	2.1%

量数：1 列序：数据集　　观测数：260

实际上到这一步，我们已经把表格整理的挺干净了，不过这不像一个表格，我们需要进行这样的一个操作：我们已经注意到了1-13行实际是表格的第一行，14-26行实际是表格的第二行。我们该怎么完成这样的一个操作呢？一个非常好用的命令是`post命令`。这个命令的功能就像它的名字一样——邮局。它可以实现把v的每个观测值发送到我们想要得到的表格的指定位置。

```
* 第一步postfile建立一个“邮局”，同时设定“收件人”（变量date、v1、v2、v3、v4）：
postfile mypost str20 date str20 v1 str20 v2 str20 v3 ///
        str20 v4 str20 v5 str20 v6 str20 v7 str20 v8 str20 v9 ///
        str20 v10 str20 v11 str20 v12 using cpi.dta, replace
* 然后循环将v的值对应发送给各个收件人
forval i = 1(13)`=_N'{
        post mypost (v[`i']) (v[`i' + 1]) (v[`i' + 2]) (v[`i' + 3]) ///
                                (v[`i' + 4]) (v[`i' + 5]) (v[`i' + 6]) (v[`i' + 7]) ///
                                (v[`i' + 8]) (v[`i' + 9]) (v[`i' + 10]) (v[`i' + 11]) ///
                                (v[`i' + 12])
}
* 关闭邮局mypost
postclose mypost

* 打开 cpi.dta 就能看到整理好的数据了
use cpi, clear
```

经过这个“邮局操作”，数据现在变成这个样子的了：

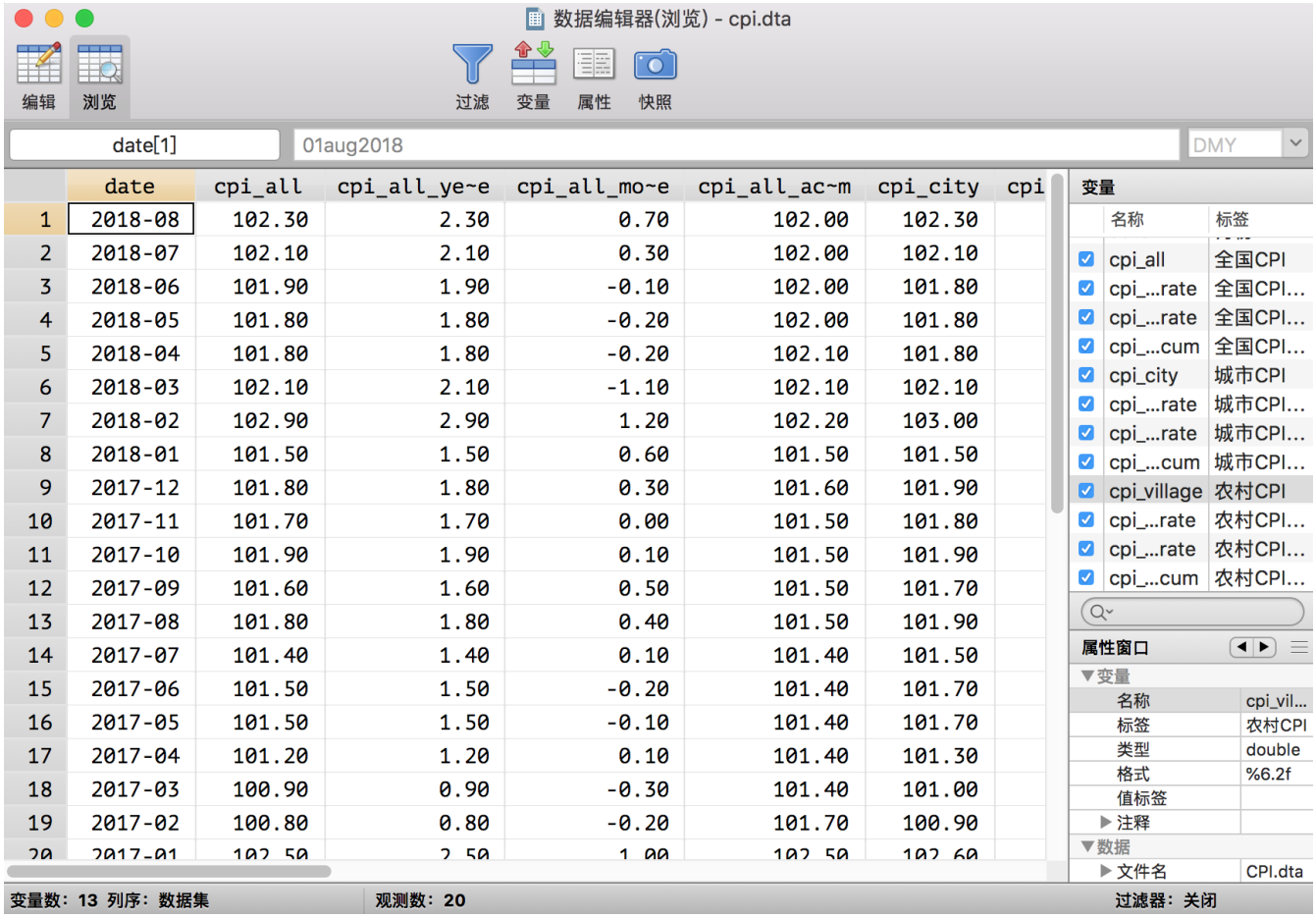


到这一步我们实际上已经完成了这个表格的爬取了，不过接下来我们再把数据整理成更加规整的Stata数据。

```
* 整理date变量
replace date = subinstr(date, "年", "", .)
replace date = subinstr(date, "月份", "", .)
* date()函数把字符串日期变成Stata日期
gen date1 = date(date, "YM")
* format一下便于我们人类理解
format date1 %tdCY-N
* 把date1变量放在第一列
order date1
* 删除date
drop date
* 重命名date1为date
ren date1 date
* 循环所有变量，把%删除
foreach i of varlist _all{
        cap replace `i' = subinstr(`i', "%", "", .)
}
* 把所有能被转换为数值型变量的字符串变量转换成数值型变量
destring, replace
* 循环所有变量，如果变量名不是date就把显示格式变成%6.2f
foreach i of varlist _all{
        if "`i'" != "date" {
                format `i' %6.2f
        }
}
* 添加变量标签
label var date "月份"
label var v1 "全国CPI"
label var v2 "全国CPI年率"
label var v3 "全国CPI月率"
label var v4 "全国CPI累计"
label var v5 "城市CPI"
label var v6 "城市CPI年率"
label var v7 "城市CPI月率"
label var v8 "城市CPI累计"
label var v9 "农村CPI"
label var v10 "农村CPI年率"
label var v11 "农村CPI月率"
label var v12 "农村CPI累计"
* 数据集标签
label data "消费者价格指数"

* 变量重命名
ren v1 cpi_all
ren v2 cpi_all_year_rate
ren v3 cpi_all_month_rate
ren v4 cpi_all_accum
ren v5 cpi_city
ren v6 cpi_city_year_rate
ren v7 cpi_city_month_rate
ren v8 cpi_city_accum
```

```
ren v9 cpi_village
ren v10 cpi_village_year_rate
ren v11 cpi_village_month_rate
ren v12 cpi_village_accum
save CPI_final, replace
```

这样整理之后的数据集是这样的：



	date	cpi_all	cpi_all_ye~e	cpi_all_mo~e	cpi_all_ac~m	cpi_city	cpi
1	2018-08	102.30	2.30	0.70	102.00	102.30	
2	2018-07	102.10	2.10	0.30	102.00	102.10	
3	2018-06	101.90	1.90	-0.10	102.00	101.80	
4	2018-05	101.80	1.80	-0.20	102.00	101.80	
5	2018-04	101.80	1.80	-0.20	102.10	101.80	
6	2018-03	102.10	2.10	-1.10	102.10	102.10	
7	2018-02	102.90	2.90	1.20	102.20	103.00	
8	2018-01	101.50	1.50	0.60	101.50	101.50	
9	2017-12	101.80	1.80	0.30	101.60	101.90	
10	2017-11	101.70	1.70	0.00	101.50	101.80	
11	2017-10	101.90	1.90	0.10	101.50	101.90	
12	2017-09	101.60	1.60	0.50	101.50	101.70	
13	2017-08	101.80	1.80	0.40	101.50	101.90	
14	2017-07	101.40	1.40	0.10	101.40	101.50	
15	2017-06	101.50	1.50	-0.20	101.40	101.70	
16	2017-05	101.50	1.50	-0.10	101.40	101.70	
17	2017-04	101.20	1.20	0.10	101.40	101.30	
18	2017-03	100.90	0.90	-0.30	101.40	101.00	
19	2017-02	100.80	0.80	-0.20	101.70	100.90	
20	2017-01	102.50	2.50	1.00	102.50	102.60	

名称	标签
cpi_all	全国CPI
cpi_...rate	全国CPI...
cpi_...rate	全国CPI...
cpi_...cum	全国CPI...
cpi_city	城市CPI
cpi_...rate	城市CPI...
cpi_...rate	城市CPI...
cpi_...cum	城市CPI...
cpi_village	农村CPI
cpi_...rate	农村CPI...
cpi_...rate	农村CPI...
cpi_...cum	农村CPI...

属性窗口	
▼变量	
名称	cpi_vil...
标签	农村CPI
类型	double
格式	%6.2f
值标签	
▶注释	
▼数据	
▶文件名	CPI.dta

这样我们就爬好了单页面的表格。另外我们也注意到完整的表格有7页。我们点击下一页可以看到第二页的网址是：

```
http://data.eastmoney.com/cjsj/consumerpriceindex.aspx?p=2
```

显然url中的最后一个参数就是页数。每页的结构都是一致的，所以可以循环运行刚刚的代码把剩下6个页面的表格依次爬下来然后合并。

多页面爬取

纵向拼接示例

作为示例，我们再尝试用刚刚的代码爬第二页：

```
clear
copy "http://data.eastmoney.com/cjsj/consumerpriceindex.aspx?p=2" temp.txt, replace
utrans temp.txt
infix strL v 1-20000 using temp.txt, clear
keep if index(v[_n-1], "<td class=") | index(v[_n-1], "<span")
drop if v == ""
drop if index(v, "/")
postfile mypost str20 date str20 v1 str20 v2 str20 v3 ///
        str20 v4 str20 v5 str20 v6 str20 v7 str20 v8 str20 v9 ///
        str20 v10 str20 v11 str20 v12 using CPI_temp.dta, replace
forval i = 1(13)`=_N'{
        post mypost (v[`i']) (v[`i' + 1]) (v[`i' + 2]) (v[`i' + 3]) ///
                                (v[`i' + 4]) (v[`i' + 5]) (v[`i' + 6]) (v[`i' + 7]) ///
                                (v[`i' + 8]) (v[`i' + 9]) (v[`i' + 10]) (v[`i' + 11]) ///
                                (v[`i' + 12])
}
postclose mypost
use CPI_temp, clear
replace date = subinstr(date, "年", "", .)
replace date = subinstr(date, "月份", "", .)
gen date1 = date(date, "YM")
format date1 %tdCY-N
order date1
drop date
ren date1 date
foreach i of varlist _all{
        cap replace `i' = subinstr(`i', "%", "", .)
}
destring, replace
foreach i of varlist _all{
        if "`i'" != "date" {
                format `i' %6.2f
        }
}
ren v1 cpi_all
ren v2 cpi_all_year_rate
ren v3 cpi_all_month_rate
ren v4 cpi_all_accum
ren v5 cpi_city
ren v6 cpi_city_year_rate
ren v7 cpi_city_month_rate
ren v8 cpi_city_accum
ren v9 cpi_village
ren v10 cpi_village_year_rate
ren v11 cpi_village_month_rate
ren v12 cpi_village_accum
```

这些代码运行之后可以得到一个和第一页爬取结果类似的表格，然后可以用append命令把这个表格纵向拼接到第一页爬取的数据集 `CPI_final.dta` 上：

```
append using CPI_final
save CPI_final, replace
```

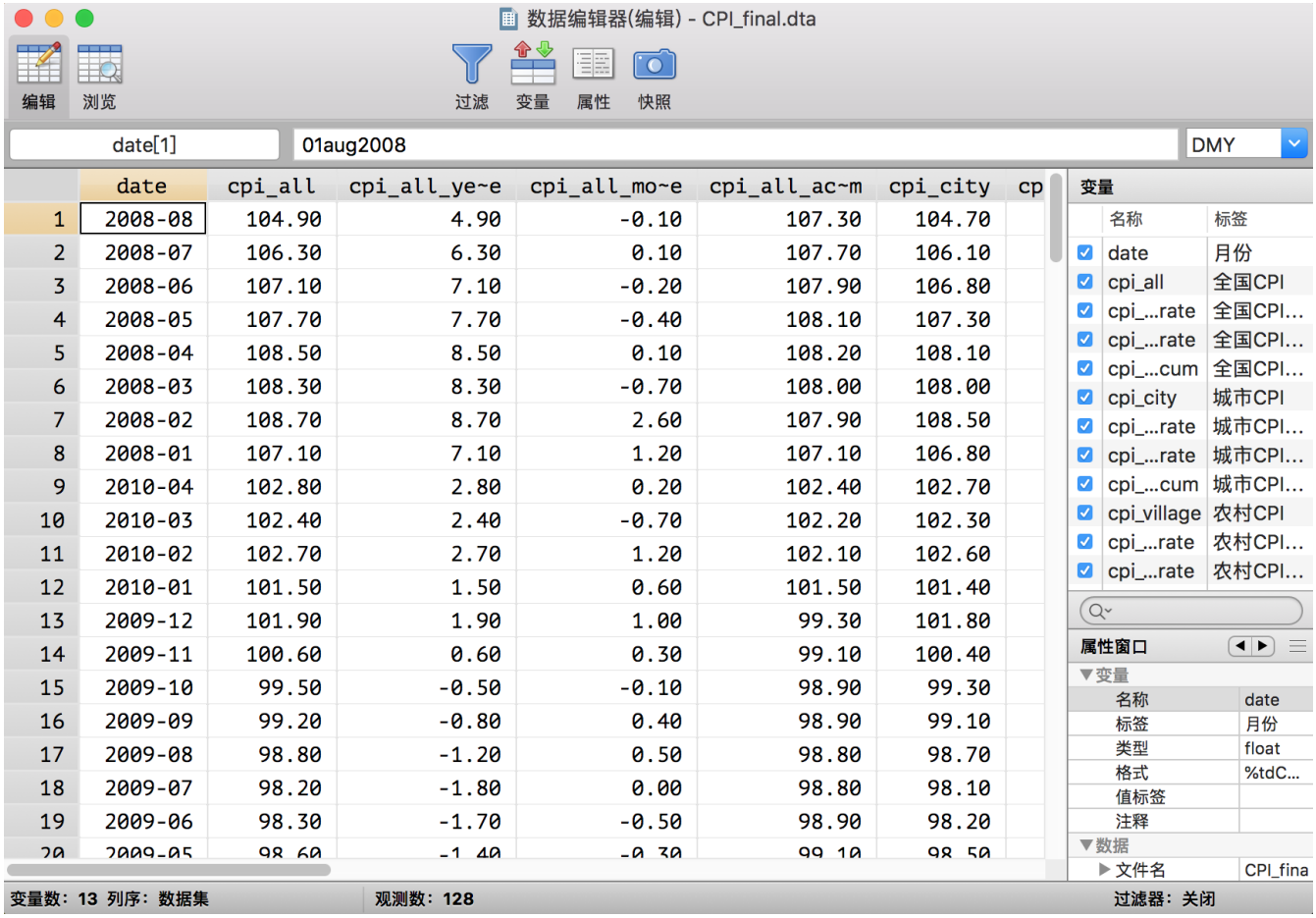
循环拼接

为了连贯，我再把上面的代码重新写，因为有些代码可以在放在循环之后再运行，例如变量标签、变量名等。

```
*================================*
*        东方财富网消费者价格指数爬取
*================================*
* 下载第一页
clear all
cd "~/Desktop"
copy "http://data.eastmoney.com/cjsj/cpi.html" temp.txt, replace
utrans temp.txt
infix strL v 1-20000 using temp.txt, clear
keep if index(v[_n-1], "<td class=") | index(v[_n-1], "<span")
drop if v == ""
drop if index(v, "/")
postfile mypost str20 date str20 v1 str20 v2 str20 v3 ///
        str20 v4 str20 v5 str20 v6 str20 v7 str20 v8 str20 v9 ///
        str20 v10 str20 v11 str20 v12 using CPI_temp.dta, replace
forval i = 1(13)`=_N'{
        post mypost (v[`i']) (v[`i' + 1]) (v[`i' + 2]) (v[`i' + 3]) ///
                (v[`i' + 4]) (v[`i' + 5]) (v[`i' + 6]) (v[`i' + 7]) ///
                (v[`i' + 8]) (v[`i' + 9]) (v[`i' + 10]) (v[`i' + 11]) ///
                (v[`i' + 12])
}
postclose mypost
use CPI_temp, clear
save CPI_final, replace

* 接下来循环第2到第7页，把每一页纵向拼接
forval i = 2/7{
        clear
        copy "http://data.eastmoney.com/cjsj/consumerpriceindex.aspx?p=`i'" temp.txt, replace
        utrans temp.txt
        infix strL v 1-20000 using temp.txt, clear
        keep if index(v[_n-1], "<td class=") | index(v[_n-1], "<span")
        drop if v == ""
        drop if index(v, "/")
        postfile mypost str20 date str20 v1 str20 v2 str20 v3 ///
                str20 v4 str20 v5 str20 v6 str20 v7 str20 v8 str20 v9 ///
                str20 v10 str20 v11 str20 v12 using CPI_temp.dta, replace
        forval i = 1(13)`=_N'{
                post mypost (v[`i']) (v[`i' + 1]) (v[`i' + 2]) (v[`i' + 3]) ///
                        (v[`i' + 4]) (v[`i' + 5]) (v[`i' + 6]) (v[`i' + 7]) ///
                        (v[`i' + 8]) (v[`i' + 9]) (v[`i' + 10]) (v[`i' + 11]) ///
                        (v[`i' + 12])
        }
        postclose mypost
        use CPI_temp, clear
        append using CPI_final
        save CPI_final, replace
}

use CPI_final, clear
replace date = subinstr(date, "年", "", .)
```

```
replace date = subinstr(date, "月份", "", .)
gen date1 = date(date, "YM")
format date1 %tdCY-N
order date1
drop date
ren date1 date
foreach i of varlist _all{
        cap replace `i' = subinstr(`i', "%", "", .)
}
destring, replace
foreach i of varlist _all{
        if "`i'" != "date" {
                format `i' %6.2f
        }
}
* 添加变量标签
label var date "月份"
label var v1 "全国CPI"
label var v2 "全国CPI年率"
label var v3 "全国CPI月率"
label var v4 "全国CPI累计"
label var v5 "城市CPI"
label var v6 "城市CPI年率"
label var v7 "城市CPI月率"
label var v8 "城市CPI累计"
label var v9 "农村CPI"
label var v10 "农村CPI年率"
label var v11 "农村CPI月率"
label var v12 "农村CPI累计"
* 数据集标签
label data "消费者价格指数"

* 变量重命名
ren v1 cpi_all
ren v2 cpi_all_year_rate
ren v3 cpi_all_month_rate
ren v4 cpi_all_accum
ren v5 cpi_city
ren v6 cpi_city_year_rate
ren v7 cpi_city_month_rate
ren v8 cpi_city_accum
ren v9 cpi_village
ren v10 cpi_village_year_rate
ren v11 cpi_village_month_rate
ren v12 cpi_village_accum
save CPI_final, replace
```

爬取结果：

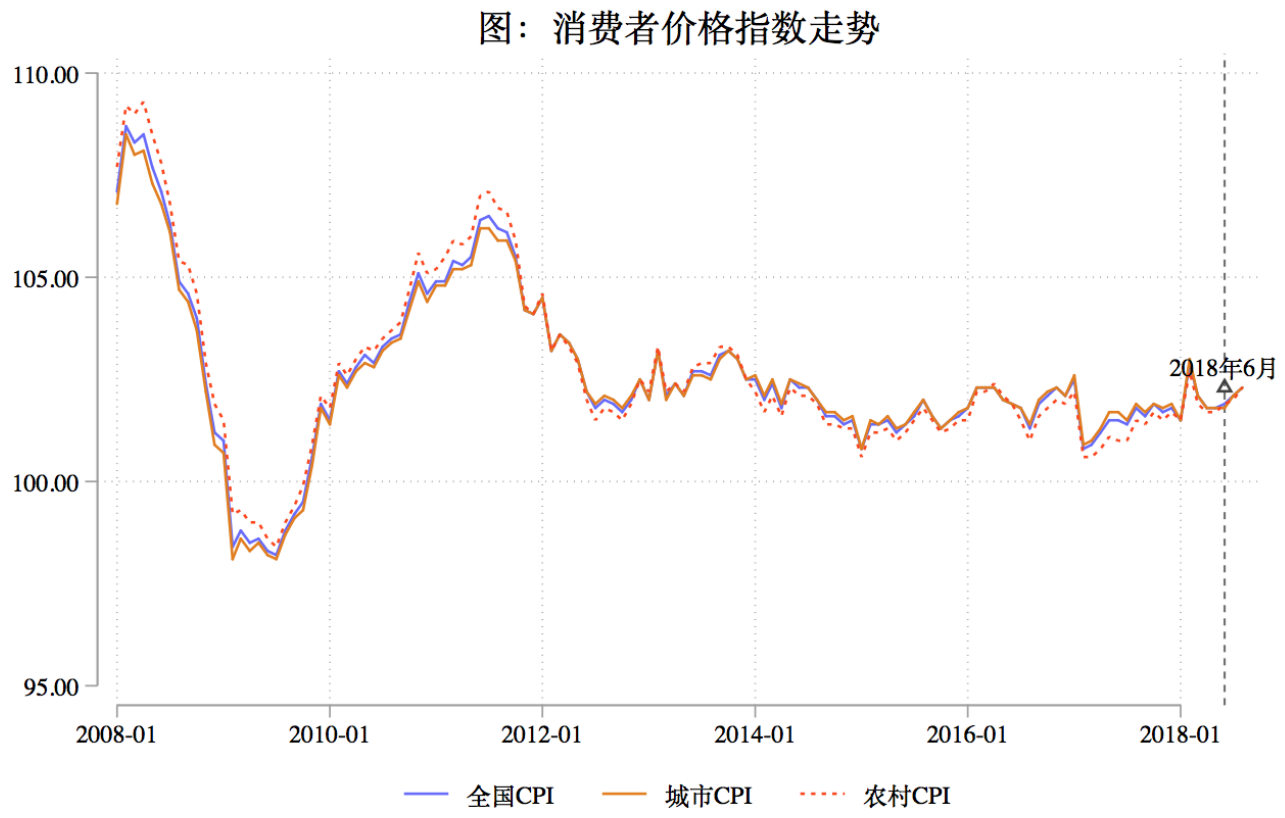


至此，这个爬取人物我们就算完成了。下面我们进行一个简单的应用——数据展示。

数据呈现

假如我想观察CPI的走势，需要绘制一幅线图：

```
use CPI_final, clear
gsort date
* 绘图
* 推荐使用我最喜欢的绘图主题plotplain
* 安装方法
ssc install blindschemes, replace all
* 把绘图主题永久性的设置为plotplain
set scheme plotplain, permanently
* 查看2018年6月对应的Stata日期
di date("2018-06", "YM")
* ///表示代码换行，注意下面几行绘图代码要一起运行
tw ///
line cpi_all date,  ///
        lc(blue*0.6) lp(solid)  ///
        xline(21336) ||  ///
line cpi_city date, lc(dkorange)  ///
        lp(solid) ||  ///
line cpi_village date, lc(orange_red) ||, ///
        ti("图：消费者价格指数走势", size(*1.2)) ///
        leg(pos(6) row(1))  ||  ///
scatteri 102.3 21336 (12) "2018年6月"
// 导出图片为png格式
gr export 20181004a1.png, replace
```



上面一些选项的解释：

- ||: 用于分隔图层。
- line: 用于绘制线图
- tw: 全称是twoway，用于组合多个图层
- lc: 全称为lcolor()用于控制线图的颜色
- lp: 全称是lpattern()用于控制线型
- yline: 在指定位置画一条水平线
- xline: 在指定位置画一条竖直线
- ti: 全称是title()，控制标题，size用于控制标题文字大小，这里是1.2倍
- leg: 全称是legend()，控制图例，pos用于控制图例的位置，
  这里是6点种方向，单行排列。
- scatteri: 用于在指定的坐标处画个点。"2018年6月"是这个点的标签, (12)用于指定这个标签位于点的方向，指定为12点钟方向。