



Data Preparation 准备数据

Instructor: Zach Zhizhong ZHOU, Shanghai Jiao
Tong University
主讲教师：周志中，上海交通大学

Data Mining for Business Intelligence

Shmueli, Patel & Bruce



两个样本之间的相似度

$$S_{ij} = e^{-\alpha D_{ij}}$$

S_{ij} : 两个样本之间的相似度

D_{ij} : 两个样本之间的距离

$$\alpha = -\ln(0.5) / D,$$

其中 D 是数据集中样本之间的平均距离。



两个样本之间的距离

标准化的欧氏距离

$$D_{ij} = \left[\sum_{k=1}^n \left((X_{ik} - X_{jk}) / (\max_k - \min_k) \right)^2 \right]^{1/2}$$

n 是维数, \max_k 和 \min_k 是用于第 k 维进行标准化的最大值和最小值。

汉明 (Hamming) 距离和名义变量相似性

$$S_{ij} = \left(\sum_{k=1}^n |X_{ik} - X_{jk}| \right) / n$$

如果 $X_{ik} = X_{jk}$, 则 $|X_{ik} - X_{jk}| = 0$, 否则为1。



基于汉明距离的相似性度量

	喜欢葛优	喜欢宋仲基	喜欢范冰冰	喜欢TFboys
女生1	Y	Y	N	N
女生2	N	Y	N	Y
女生3	N	N	Y	N
女生4	N	Y	N	Y

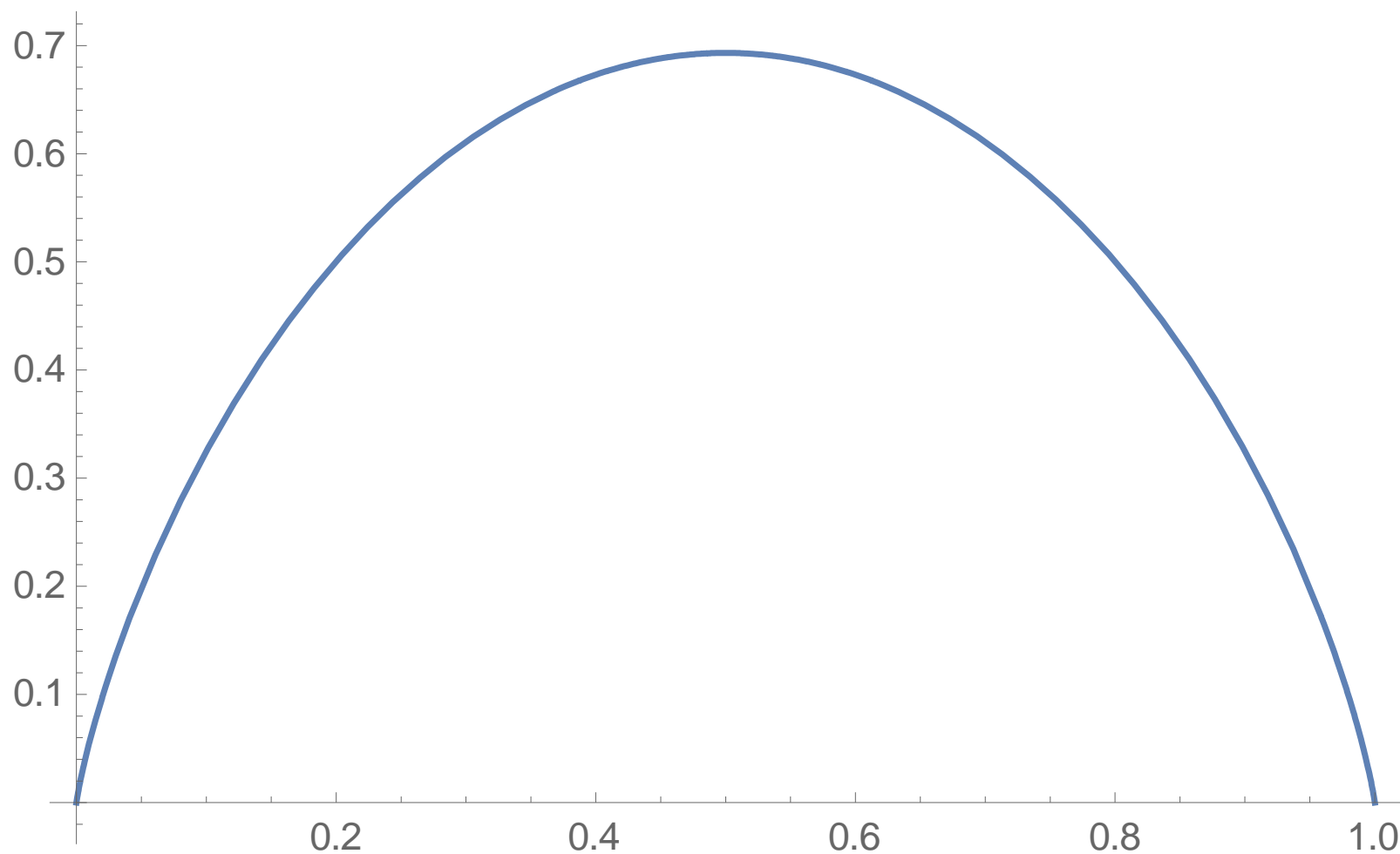
S	1	2	3	4
1	NA	1/2	1/4	1/2
2	1/2	NA	1/4	1
3	1/4	1/4	NA	1/4
4	1/2	1	1/4	NA

□ 从信息论得知熵是一个全局度量，对于有序结构较少，对于无序结构较大。对于一个N个样本的数据集而言，熵为：

$$E = - \sum_{i=1}^{N-1} \sum_{j=i+1}^N \left[S_{ij} \cdot \log S_{ij} + (1 - S_{ij}) \cdot \log(1 - S_{ij}) \right]$$

其中 S_{ij} 是样本 X_i 与 X_j 之间的相似性。

$$-\left[x \cdot \log x + (1-x) \cdot \log(1-x)\right]$$



- 目的：消除数值属性以及为数值属性定义准确的类别。
- 高质量的离散化应该是：区间内一致，区间之间区分明显。
- ChiMerge算法用卡方统计量来决定相邻区间是否一致或者是否区别明显。如果经过验证，类别属性独立于其中一个区间，则这个区间就要被合并。
- ChiMerge算法包括2部分：1、初始化，2、自底向上合并，当满足停止条件的时候，区间合并停止。

□ 卡方计算公式:

$$\chi^2 = \sum_{i=1}^2 \sum_{j=1}^k \frac{(A_{ij} - E_{ij})^2}{E_{ij}}$$

k : 类别数量, A_{ij} : 第 i 区间的类别为 j 的实例的数量

$E_{ij} = (R_i \cdot C_j) / N$: A_{ij} 的期望频率。

$R_i = \sum_{j=1}^k A_{ij}$: 第 i 个区间中的实例个数

$C_j = \sum_{i=1}^2 A_{ij}$: 第 j 类中的实例个数

$N = \text{总实例数} = \sum_{i=1}^2 R_i = \sum_{j=1}^k C_j$

如果 R_i 或者 C_j 等于0, 应该将 E_{ij} 设置为一个较小的值, 如 $E_{ij} = 0.1$ 。避免检验数的分母过小。

ChiMerge数据离散化算法



	Class 1	Class 2	Sum
Interval-1	A_{11}	A_{12}	R_1
Interval-2	A_{21}	A_{22}	R_2
Sum	C_1	C_2	N

ChiMerge数据离散化算法



income type			
1	2	low	1
2	5	low	2
3	10	mid	3
4	10	mid	4
5	20	mid	5
6	60	high	6

InterV	high	low	mid
[2,3.5]	0	1	0
(3.5,7.5]	0	1	0
(7.5,15]	0	0	2
(15,40]	0	0	1
(40,60]	1	0	0

InterV	high	low	mid	InterV
[2,7.5]	0	2	0	[2,7.5]
(7.5,15]	0	0	2	(7.5,15]
(15,40]	0	0	1	(15,40]
(40,60]	1	0	0	(40,60]

InterV	high	low	mid	InterV
[2,7.5]	0	2	0	[2,7.5]
(7.5,40]	0	0	3	(7.5,40]
(40,60]	1	0	0	(40,60]

□ ChiMerge 函数:

- 1. 初始化：生成分割数值的切割点。（InitializeV函数）
- 2. 如果切出来的区间太多，则继续缩减。（Shrink函数）

□ InitializeV函数:

- 对数值进行排序，合并取值相同的点。
- 取最小值为最小切割点，最大值为最大切割点。
- 其他切割点取被切割的两个数值的均值。

□ Shrink函数:

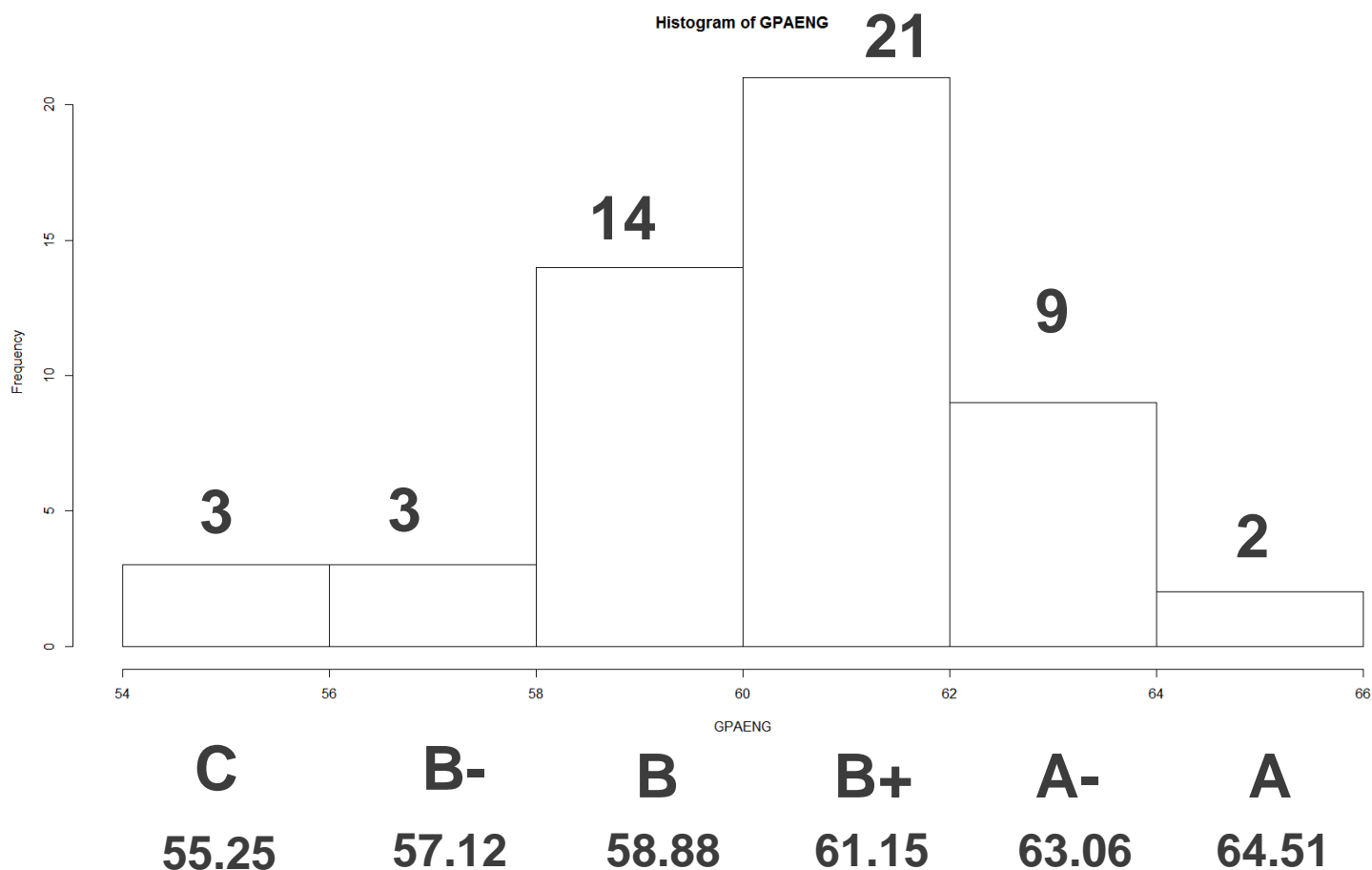
- 根据切割点得到多个区间。
- 统计各个区间内实例属于各个类别的个数。
- 计算相邻两个区间的卡方值。（ChiSquareValue函数）
- 找到取值最小的卡方值，合并该值对应的两个区间。



处理面试评分

- 安泰经管学院本科生出国游学评分体系：GPA+英语成绩占70%，英文面试成绩占30%。
- 有52个学生，3位面试老师进行面试。
- 出现的问题：
 - 面试老师打分把握尺度不同，有的面试老师打分高低分差距很大（比如在60-100分之间分布），而有的面试老师打分高低分差距较小（比如在90-100分之间分布）。考虑以下情况：一个学生得到了100, 90, 90分，那么他的平均分比另外一个得到60, 100, 100分的平均分还要高。但第一个学生有1个老师认为是最好而另外2个老师认为最差；第二个学生2个老师认为最好而只有1个老师认为最差。显然应该是第二个学生平均分比第一个学生平均分高才合理。

□ 假设：学生的GPA+英语成绩分布和学生的面试成绩分布相同。下面是学生的GPA+英语成绩分布柱状图：



□ 每个面试老师需要按照成绩分布图对学生面试情况进行打分，对52个学生的打分必须给出2个A，9个A-，21个B+，14个B，3个B-和3个C。允许有微调。

□ 打出来的分数换算成数值，换算方法如下：如果一个学生得分为（A，A-，B+），则该学生面试平均分数为
 $(64.51+63.06+61.15) / 3$ 。由于面试成绩占30%，因此最终面试成绩是：
 $(64.51+63.06+61.15) / 3 \times (3/7)$



抓取数据：中国银行外汇牌价

- 使用到的程序包：XML
- 抓取中国银行外汇牌价数据：
- `url = 'http://www.boc.cn/sourcedb/whpj/'`
- `Rates = readHTMLTable(url, header=TRUE, which=2, stringsAsFactors=F)` 这条语句抓取网页当中的表格内容。

抓取数据：中国银行外汇牌价



□ `doc = htmlParse(url)` #提取网页源代码

□ `t_heads <- xpathSApply(doc, "//th", xmlValue)` #提取表格头信息。

这里抓取的信息是<th>和</th>之间的取值。//双斜杠号表示th标签可以在HTML源代码中的任何地方。

□ `xpathSApply(doc, "/html/body/div/div/div/div/table/tr/th", xmlValue)`
这里抓取的是指定xpath路径下的取值。

□ `xpathSApply` 的第2个参数如何写，详见《Automated Data Collection with R》的第88页：Table 4.2。



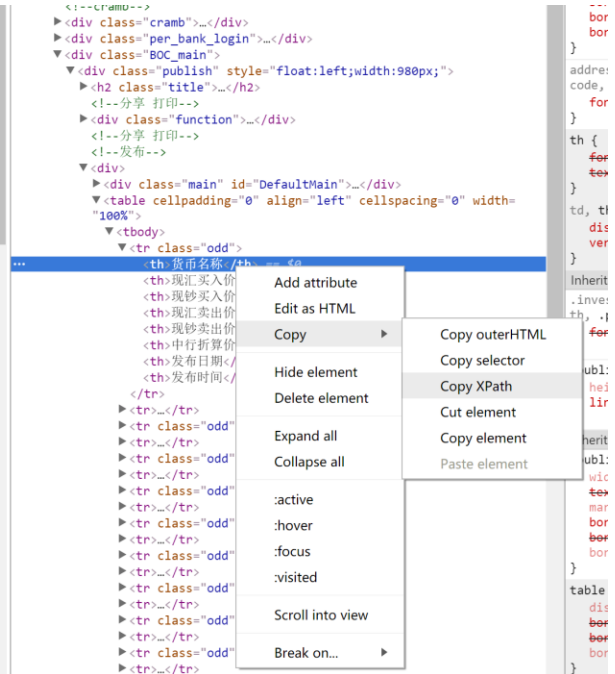
抓取数据：中国银行外汇牌价

中国银行外汇牌价

打印

起始时间: 结束时间: 牌价选择: 选择货币

货币名称	现汇买入价	现钞买入价	现汇卖出价	现钞卖出价	中行折算价	发布日期	发布时间
阿联酋迪拉姆		177.04		189.88	183.5	2016-10-17	20:38:02
澳大利亚元	511.78	495.85	515.22	515.22	512.06	2016-10-17	20:38:02
巴西里亚尔		202.22		221.18	210.08	2016-10-17	20:38:02
加拿大元	511.63	495.45	515.07	515.33	512.13	2016-10-17	20:38:02
瑞士法郎	679.46	658.5	684.24	684.24	680.43	2016-10-17	20:38:02
丹麦克朗	99.25	96.19	100.05	100.05	99.28	2016-10-17	20:38:02
欧元	739.17	716.15	744.13	744.13	738.83	2016-10-17	20:38:02
英镑	816.36	790.94	821.85	821.85	819.82	2016-10-17	20:38:02
港币	86.68	85.99	87.02	87.02	86.85	2016-10-17	20:38:02



□ 在谷歌浏览器中将鼠标移到“货币名称”点击右键，然后点击“查看”，就可以调出“货币名称”的Xpath和selector。不过有时候在谷歌浏览器看到的Xpath不一定和接收到的源代码中元素的Xpath相同，以源代码为准。

抓取数据：微博排行榜



```
□ <dd class="name">谢娜</dd>  
names <- xpathSApply(doc,  
"//dd[@class='name']",xmlValue) #提取人名
```

```
□ <dd class="bio">觉得等等帅的请转发，觉得我帅的请评论，觉得我把等等秒成渣的请点赞！ </dd>  
shortbio <- xpathSApply(doc,  
"//dd[@class='bio']",xmlValue) #提取微博简介。
```

```
□ <section class="list_info" action-type="userList"  
data-url="http://weibo.com/u/5187664653"> ...  
</section>  
WeiboURLs <- xpathSApply(doc, "//dd[@class='bio']",  
xmlGetAttr, "data-url") #提取微博链接地址。
```



提取Email: 正则表达式

□ `pattern <- '[:alnum:]_.-]+@[[:alnum:]_.-]+\.[A-Za-z]+'` #Email的正则表达式。

□ `pattern <- '[:digit:]{17}[:digit:]Xx'` #18位身份证的正则表达式。

□ `pattern <- '[:digit:]{4}-[:digit:]{2}-[:digit:]{2}'` 匹配YYYY-MM-DD格式的日期



提取Email: 匹配正则表达式

□ 找出所有和正则表达式匹配的字符串:

```
str_match_all(line, pattern)
```

□ `match_emails <- unlist(str_match_all(line, pattern))`
对匹配结果进行unlist操作。

□ `email_list <- append(email_list, match_emails)` 将提取出来的Email放入email_list尾部。

使用PhantomJS读取JavaScript生成的数据



```
var url = ' http://hefei.lashou.com/ ';  
var page = new WebPage()  
var fs = require('fs');  
page.open(url, function (status) { just_wait();});  
function just_wait() { setTimeout(function() {  
    fs.write('MyPage.html', page.content, 'w');  
    phantom.exit();  }, 2500); }
```

上面的语句使用PhantomJS读取网页并执行JavaScript, 最后将源代码写入MyPage.html。

```
MyScrapeJS <- function(MyURL){  
  ## change Phantom.js scrape file  
  lines <- readLines("ScrapePage.js")  
  lines[1] <- paste0("var url =", MyURL, ";;")  
  writeLines(lines, "ScrapePage.js") }
```

上面的函数用来修改ScrapePage.js当中的第一行：
var url = MyURL，其中MyURL可以在R Script当中自行定义。

使用PhantomJS读取JavaScript生成的数据



- ❑ `MyScrapeJS('http://hefei.lashou.com/')`
- ❑ `system(“phantomjs ScrapePage.js”)` #相当于在命令行下执行: `phantomjs ScrapePage.js`
- ❑ `doc <- htmlParse(‘MyPage.html’)` #提取出的源代码放在MyPage.html当中。
- ❑ `xpathSApply(doc, “//a[@class= ‘index-goods-name’]” , xmlValue)` #提取需要的数据

- 安装CasperJS和PhantomJS
- 设置环境变量Path，确保CasperJS和PhantomJS可执文件在Path定义的目录当中。
- 在命令行下执行CasperJS SearchBaidu.js 做的动作是在百度搜索SJTU，然后打开第一个搜索结果页面并且下载该页面的源代码并截屏。

使用CasperJS读取JavaScript生成的数据



```
var url
= 'http://v6.bang.weibo.com/czv/caijing?period=month' ; // 此处可以修改成其他URL地址
var casper = require('casper').create();
var fs = require('fs');
var save = 'MyPage.html';
casper.start(url, function() {
    fs.write(save, this.getPageContent(), 'w');    });
casper.run();
```

在命令行模式下执行CasperJS ScrapePage_CasperJS.js
也可以把JavaScript生成数据的页面抓下来存在
MyPage.html当中供后面的分析和数据抓取。

使用Selenium IDE自动生成Python代码



- 安装Java, Selenium和Firefox (低版本比如Firefox 45)。
- 在Firefox安装Selenium IDE
- 按F10在Firefox上跳出Tools, 点击Selenium IDE并点击录制键开始录制。
- 在网页上进行操作, Selenium IDE会记录下你的所有操作并可以将操作转换成Python语句。
- 在Python上继续完善代码, 比如加入提取网页的指令。

□ 在R Studio中打开SearchJenner.R逐行执行代码观察Selenium+Firefox如何自动实现鼠标在网页上的各种操作。这个程序实现的功能是：在Jenner.com上搜索位于纽约的Partners。

□ 在R Studio中打开SearchBaidu.R逐行执行代码。代码实现以下功能：在百度搜索框中搜索SJTU，然后提取出第一个搜索结果的链接（是交大主页），接着访问交大主页，倒回原先的搜索结果（goBack），往前到交大主页（goForward），再回到搜索结果（goBack），抓取搜索结果的网页源代码，通过分析源代码抓取第一页搜索结果的链接地址和网页描述，访问第二页搜索结果，提取第二页搜索结果的链接地址和网页描述。

使用Selenium IDE自动生成Python代码



```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

driver = webdriver.Firefox();
driver.get("https://jenner.com/people");

driver.find_element_by_xpath("//form[@id='new_search']/div[4]/div/h1").click()
driver.find_element_by_id("search_offices_new_york").click()
driver.find_element_by_css_selector("div.filter.roles > div.header > h1").click()
driver.find_element_by_xpath("//form[@id='new_search']/div[6]/div/h1").click()
driver.find_element_by_id("search_roles_partner").click()
driver.find_element_by_css_selector("input.button:nth-child(2)").click()

driver.save_screenshot('page.png');
driver.close();

//这一段程序代码使用Selenium IDE录制在https://jenner.com/people搜索位于纽约
(New York) 的Partners的鼠标点击动作，然后输出成Python代码，最后截图。
```

使用Selenium+Python实现鼠标点击动作





















□ 网页上的一些element, 如<a> 此处包括onclick语句, 连网页代码都是自动生成。这时候使用CasperJS写以下的语句试图点击“1st Period”是无效的:

```
this.click(xPath('//*[@id="bettype-tabs-scope"]/ul/li[2]/a'));
```

Carolina Hurricanes - Ottawa Senators

 Saturday, 25 Jan 2014, 18:00

 Final result 6:3 (1:0, 4:3, 1:0)

1X2	Home/Away	AH	O/U	EH	DC	More bets	▼
Full Time	1st Period	2nd Period	3rd Period				
Bookmakers	1▼	X▼	2▼	Payout▼			
 bet-at-home	2.25	3.80	2.60	91.6%			
 bet365	2.29	3.79	2.64	92.7%			
 bwin	2.30	3.75	2.70	93.3%			
 Marathonbet	2.35	3.80	2.78	95.4%			
 TonyBet	2.35	3.70	2.70	93.8%			
 Unibet	2.35	3.85	2.60	93.5%			
Click to show 38 more bookmakers!							

□ 使用Selenium IDE录制鼠标行为同样无效。

使用Selenium+Python实现鼠标点击动作



```
from selenium import webdriver
driver = webdriver.Firefox();
driver.get("http://www.oddsportal.com/hockey/usa/nhl-2013-2014/carolina-hurricanes-ottawa-senators-80YZhBGC");

driver.find_element_by_css_selector("ul.sub-menu > li:nth-child(2) > a:nth-child(1)").click(); // 点击1st Period
driver.save_screenshot('page1.png');

driver.find_element_by_css_selector(".ul-nav > li:nth-child(3) > a:nth-child(1)").click();
driver.save_screenshot('page2.png');

driver.close();
```

- ❑ PhantomJS: <http://phantomjs.org/>
- ❑ PhantomJS Examples:
<http://phantomjs.org/examples/index.html>
- ❑ CasperJS: <http://casperjs.org/>
- ❑ Code Epicenter:
<http://code-epicenter.com/category/tutorials/>
- ❑ Selenium:
<http://docs.seleniumhq.org/>
- ❑ Scrapy:
<https://scrapy.org/>
- ❑ 极客学院: Python网络爬虫:
<http://wiki.jikexueyuan.com/project/python-crawler-guide/>