用 Julia 编写普通最小二乘法

Cheng Jun

2017 年 7 月 27 日

版本：0.0.1

- 编者信息

实证研究小青年，手艺正在提高中。
经管之余，泛读史哲，关注自由与开源动态。
GitHub：chengjun90
知乎：Cheng Jun

- 笔记说明

本系列笔记最早是分享在知乎专栏：Julia 学习笔记。在这里重新梳理和更新。
笔记是在 Jupyter Notebook 编写，导出 Tex 文件，经手动微调后编译成 pdf 文档。

普通最小二乘法 (OLS) 是最常见的数据分析方法之一，在统计学、应用计量经济学和数据科学中有广泛的应用。

虽然 Julia 目前已经有 GLM 包可以做一些简单的统计与计量分析，但是这里全部采用 Julia 函数来实现 OLS 基本分析。主要目的是增加 Julia 应用的熟练度，纸上得来终觉浅，绝知此事要躬行；其次是温故 OLS 基本算法。

本文的分析是在 Jupyter Notebook 完成。具体环境如下。

```
In [2]: versioninfo()
```

```
Julia Version 0.7.0-DEV.1122
Commit 892768cf52* (2017-07-25 22:26 UTC)
Platform Info:
  OS: Windows (x86_64-w64-mingw32)
  CPU: Intel(R) Core(TM) i5-4210U CPU @ 1.70GHz
  WORD_SIZE: 64
```

```
  BLAS: libopenblas (USE64BITINT DYNAMIC_ARCH NO_AFFINITY Haswell)
  LAPACK: libopenblas64_
  LIBM: libopenlibm
  LLVM: libLLVM-3.9.1 (ORCJIT, haswell)
Environment:
```

主要细节:

- 读取 *csv* 数据，注意数据类型
- 计算回归系数
- 计算 R^2 和调整的 R^2
- 计算 F 值
- 打印参与 *OLS* 计算的样本数量

```
In [3]: auto = readdlm("data/auto.csv", ',',header=true)
```

```
Out[3]: (Any["AMC Concord" 4099 … 3.58 0; "AMC Pacer" 4749 … 2.53 0; … ; "VW Scirocco" 6850
```

```
In [4]: for i in auto[2]
            println(i)
        end
```

```
make
price
mpg
rep78
headroom
trunk
weight
length
turn
displacement
gear_ratio
foreign
```

```
In [5]: auto[1]  # 看一下数据
```

```
Out[5]: 74×12 Array{Any,2}:
 "AMC Concord"      4099  22  3    2.5  11  2930  186  40  121  3.58  0
 "AMC Pacer"        4749  17  3    3.0  11  3350  173  40  258  2.53  0
 "AMC Spirit"       3799  22   ""  3.0  12  2640  168  35  121  3.08  0
 "Buick Century"    4816  20  3    4.5  16  3250  196  40  196  2.93  0
 "Buick Electra"    7827  15  4    4.0  20  4080  222  43  350  2.41  0
 "Buick LeSabre"    5788  18  3    4.0  21  3670  218  43  231  2.73  0
 "Buick Opel"       4453  26   ""  3.0  10  2230  170  34  304  2.87  0
 "Buick Regal"      5189  20  3    2.0  16  3280  200  42  196  2.93  0
 "Buick Riviera"   10372  16  3    3.5  17  3880  207  43  231  2.93  0
 "Buick Skylark"    4082  19  3    3.5  13  3400  200  42  231  3.08  0
 "Cad. Deville"    11385  14  3    4.0  20  4330  221  44  425  2.28  0
 "Cad. Eldorado"   14500  14  2    3.5  16  3900  204  43  350  2.19  0
 "Cad. Seville"    15906  21  3    3.0  13  4290  204  45  350  2.24  0

 "Mazda GLC"        3995  30  4    3.5  11  1980  154  33   86  3.73  1
 "Peugeot 604"     12990  14   ""  3.5  14  3420  192  38  163  3.58  1
 "Renault Le Car"   3895  26  3    3.0  10  1830  142  34   79  3.72  1
 "Subaru"           3798  35  5    2.5  11  2050  164  36   97  3.81  1
 "Toyota Celica"    5899  18  5    2.5  14  2410  174  36  134  3.06  1
 "Toyota Corolla"   3748  31  5    3.0   9  2200  165  35   97  3.21  1
 "Toyota Corona"    5719  18  5    2.0  11  2670  175  36  134  3.05  1
 "VW Dasher"        7140  23  4    2.5  12  2160  172  36   97  3.74  1
 "VW Diesel"        5397  41  5    3.0  15  2040  155  35   90  3.78  1
 "VW Rabbit"        4697  25  4    3.0  15  1930  155  35   89  3.78  1
 "VW Scirocco"      6850  25  4    2.0  16  1990  156  36   97  3.78  1
 "Volvo 260"       11995  17  5    2.5  14  3170  193  37  163  2.98  1
```

如果，现在就进入回归分析会出错。原因是 auto 的数据类型并不是数字 Number 类型，查看了才知道是 Any 类型。

```
In [6]: # 看一下各个变量的数据类型
        typeof(auto[1][:,2]),typeof(auto[1][:,3]),typeof(auto[1][:,5])
```

```
Out[6]: (Array{Any,1}, Array{Any,1}, Array{Any,1})
```

```
In [7]: # auto[1][:,2] 是读取数据中的 price 列，然后转为 Float64 类型
        price = convert(Array{Float64,1}, auto[1][:,2])
```

```
mpg = convert(Array{Float64,1}, auto[1][:,3])
headroom = convert(Array{Float64,1}, auto[1][:,5])
```

```
Out[7]: 74-element Array{Float64,1}:
 2.5
 3.0
 3.0
 4.5
 4.0
 4.0
 3.0
 2.0
 3.5
 3.5
 4.0
 3.5
 3.0

 3.5
 3.5
 3.0
 2.5
 2.5
 3.0
 2.0
 2.5
 3.0
 3.0
 2.0
 2.5
```

现在到了关键的步骤了，就是把 price，mpg 和 headroom 组成矩阵，按照 $y = X\beta + \epsilon$ 进行计算。设 price 为 y，x 是 [c mpg headroom]，c 是常数 1，代表截距项。估计回归系数有两种：按公式是 inv(x′x)*(x′y)，还可以直接 $x\backslash y$ 搞定。

```
In [8]: c = ones(size(price, 1))
        x = [c mpg headroom]
        y = price
```

```
b    = inv(x'x)*(x'y)
b2   = x\y
u    = y - x*b
r2 = 1-(u'*u)/((y-mean(y))'*(y-mean(y)))
r2a = 1-((u'*u)/(size(u,1) - size(x,2)))/(((y-mean(y))'*(y-mean(y)))/(size(u,1)-1))
f = (r2/(size(x,2)-1))/((1-r2)/(size(u,1) - size(x,2)))
```

```
Out[8]: 10.436257971845299
```

调整的 R^2 计算公式，$\overline{R^2} = 1 - \frac{\sum e^2/(n-K)}{\sum (y-\bar{y})^2/(n-1)}$。$n$ 是样本数量，K 是解释变量的个数（含常数项）。

$F = \frac{R^2/(K-1)}{(1-R^2)/(n-K)}$。

上述 Julia 命令都是公式的具体表达。计算成功，下面打印具体的计算结果。

```
In [9]: println("Coef.")
        println(b2)
        println(b)
        println("R-squared: ",r2)
        println("Adj R-squared: ",r2a)
        println("Number of obs: ",size(u,1))
        println("F:",f)
```

```
Coef.
[12683.3, -259.106, -334.021]
[12683.3, -259.106, -334.021]
R-squared: 0.22718998962087344
Adj R-squared: 0.20542069355385573
Number of obs: 74
F:10.436257971845299
```

上面是在 Julia 0.7.0 中计算的结果，下面列出在 Stata 14.2 中回归分析的结果。比较后发现，上面的计算过程基本正确。

```
. use "C:\Program Files (x86)\Stata14\auto.dta"
(1978 Automobile Data)

. reg price mpg headroom
```

```
      Source |       SS           df       MS      Number of obs   =        74
-------------+----------------------------------   F(2, 71)        =     10.44
       Model |   144280501         2  72140250.4   Prob > F        =    0.0001
    Residual |   490784895        71  6912463.32   R-squared       =    0.2272
-------------+----------------------------------   Adj R-squared   =    0.2054
       Total |   635065396        73  8699525.97   Root MSE        =    2629.2

------------------------------------------------------------------------------
       price |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
         mpg |  -259.1057   58.42485    -4.43   0.000    -375.6015   -142.6098
    headroom |  -334.0215   399.5499    -0.84   0.406    -1130.701    462.6585
       _cons |   12683.31   2074.497     6.11   0.000     8546.885    16819.74
------------------------------------------------------------------------------
```