膳食减轻体重法研究

哈尔滨工业大学 李耕

**摘要:** 本文为膳食减轻体重法研究的数据分析部分。本文借助python语言从excel表格中读取预先处理好的各类食物营养信息，通过规范格式，构建lp问题文本文件。再借助线性规划处理库CPLEX，采取barrier算法对lp文件分析处理，获取对应的数据信息。

**关键词：** 膳食减轻体重；python；读取excel；CPLEX；lp

问题简述

本课题来自于生活，从对不同年龄层的人群进行的调查分析中得到。伴随着生活水平的不断提高，社会各年龄段人群的健康水平被体重过高所带来的弊端所影响的现象不断加深。为研究一个切实可行的解决措施，我们小组通过分析决意从膳食改善面入手，通过数据分析人体所必需的营养以及日常食物中的营养成分，借助线性规划手段求得在满足人体所必需的营养供给情况下，将总摄入热量在一定可控范围内减至最小的可行食物搭配。为简化问题，显著结果，本次研究采取的数据皆为针对人体一天内的营养需要。食物方面数据得到的值是以100g为单位的质量。

食物营养表

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 食物 | 热量（千卡) | 蛋白质（克） | 膳食纤维（克) | 维生素C（毫克) | 钠（毫克) | 钙（毫克) | |
| 大米 | 347 | 6.7 | 0.6 | 0 | 527.5 | 46 |  |
| 小米 | 258 | 9.7 | 0.3 | 0 | 172 | 88 |  |
| 黑豆 | 381 | 36 | 0.2 | 0 | 3.3 | 265 |  |
| 玉米 | 336 | 8 | 0.1 | 0 | 2.5 | 10 |  |
| 燕麦片 | 367 | 5 | 0.8 | 0 | 1.6 | 18 |  |
| 鲜牛奶 | 59 | 3.3 | 0 | 0 | 3.2 | 23 |  |
| 山药 | 67 | 1.9 | 3 | 20 | 37.3 | 32 |  |
| 土豆 | 81 | 2 | 2.2 | 25 | 2.1 | 30 |  |
| 黄豆芽 | 44 | 2 | 1.1 | 10 | 39.1 | 14 |  |
| 西兰花 | 40 | 4.1 | 1.2 | 61 | 28.3 | 23 |  |
| 香菜 | 38 | 2.8 | 1.5 | 47 | 19.1 | 14 |  |
| 木瓜 | 31 | 1 | 0.8 | 56 | 14.9 | 27 |  |
| 南瓜 | 26 | 0.7 | 0.6 | 72 | 56.2 | 26 |  |
| 白萝卜 | 21 | 0.8 | 1.2 | 57 | 103.3 | 332 |  |
| 黄瓜 | 16 | 0.7 | 0.5 | 75 | 8.3 | 6 |  |
| 冬瓜 | 14 | 0.4 | 0.3 | 28 | 42.9 | 22 |  |
| 莲藕 | 79 | 1.9 | 0.2 | 56 | 77.4 | 23 |  |
| 韭菜 | 29 | 2.4 | 1.5 | 52 | 175.2 | 167 |  |
| 丝瓜 | 27 | 1 | 0.6 | 47 | 4924 | 22 |  |
| 竹笋 | 30 | 3 | 0.6 | 76 | 7.9 | 42 |  |
| 西红柿 | 20 | 4.5 | 1.2 | 23 | 6.2 | 30 |  |
| 芦笋 | 20 | 1 | 0.5 | 77 | 1.9 | 20 |  |
| 生菜 | 19 | 1 | 1.6 | 52 | 242 | 117 |  |
| 葡萄干 | 341 | 4 | 0.8 | 10 | 73 | 17 |  |
| 干红枣 | 338 | 9 | 6.2 | 8 | 9.9 | 144 |  |
| 椰子 | 700 | 1 | 0.2 | 44 | 2.2 | 72 |  |
| 板栗 | 370 | 4.2 | 0.9 | 76 | 484.9 | 125 |  |
| 香蕉 | 154 | 1.4 | 2.2 | 241 | 3.7 | 22 |  |
| 黄桃 | 58 | 0.9 | 0.8 | 58 | 2.4 | 4 |  |
| 柚子 | 59 | 1 | 1.2 | 87 | 2.3 | 2 |  |
| 西瓜 | 42 | 2 | 0.1 | 8 | 1 | 6 |  |
| 瓜子 | 1332 | 22.4 | 9.1 | 0 | 1.5 | 4 |  |
| 苹果 | 69 | 0.2 | 1.2 | 12 | 1 | 18 |  |
| 橙子 | 64 | 1 | 1.2 | 90 | 1.3 | 15 |  |
| 芒果 | 53 | 1 | 0.9 | 68 | 12.8 | 20 |  |
| 炒花生 | 581 | 20 | 9.5 | 16 | 1.5 | 17 |  |
| 杏仁 | 514 | 22.5 | 9.1 | 30 | 6.4 | 54 |  |
| 肥猪肉 | 816 | 16.9 | 0 | 0 | 26.2 | 554 |  |
| 猪肝 | 130 | 20 | 0 | 0 | 782.9 | 23 |  |
| 牛肉 | 106 | 20.1 | 0 | 0 | 618 | 28 |  |
| 猪大排 | 338 | 14.6 | 0 | 0 | 1421 | 24 |  |
| 胸脯羊肉 | 135 | 11.1 | 0 | 0 | 471 | 27 |  |
| 紫菜 | 207 | 14 | 2 | 23 | 0.3 | 6 |  |
| 金针菇 | 26 | 2.1 | 1.2 | 9 | 87 | 22 |  |
| 黑木耳 | 205 | 15 | 2.4 | 23 | 28 | 98 |  |
| 鸭蛋黄 | 378 | 13 | 0 | 0 | 227 | 877 |  |
| 鸭蛋白 | 47 | 9.6 | 0 | 0 | 229 | 78 |  |
| 基围虾 | 168 | 18.7 | 0 | 0 | 297 | 667 |  |
| 带鱼 | 167 | 15.8 | 0 | 0 | 0 | 1720 |  |
| 桂鱼 | 192 | 15.1 | 0 | 0 | 0 | 1876 |  |

读取Excel文件

# 库文件

Python中设有读取写入excel文件的库，本次操作中只涉及读取操作。

故只需调用xlrd库



导入库文件

# 操作

# 打开文件

workbook = xlrd.open\_workbook(filename=name)

通过调用xlrd中库open\_workbook来实现打开excel文件

#获取sheet页

sheet1 = workbook.sheet\_by\_index(0)

对excel对象调用其sheet\_by\_index(num)来按照索引号获取sheet页

# sheet的名称，行数，列数

print sheet2.name,sheet2.nrows,sheet2.ncols

sheet对象自带行数列数属性

sheet1.cell(i,j).value

通过cell获取unit对象，再通过value获取值

在其他文件处，我们定义了一个python类为food类，以食物表中涉及的营养信息和姓名为属性。在读取excel文件中，我们将食物信息转移到food类中，再构建food数组FOOD返回。

构建LP文件

# LP文件格式

\反斜杠为单行注释符

Maximize\开头关键词表明需要求解的线性规划问题种类，Maximize或Minimize

obj: x1 + 2 x2 + 3 x3\求值表达式

Subject To\约束条件

c1: - x1 + x2 + x3 <= 20

c2: x1 - 3 x2 + x3 <= 30

Bounds\范围限制，不类的线性规划问题往往有不同的节，混合整数线性规划（MIP）就有

0 <= x1 <= 40\General或者Binary节，而Bounds节不是必需的

End\表示文件结束

针对我们需要的LP文件格式来讲。

每一个变量对应着一种食物，求值表达式中的是热量计算公式，而约束条件则是各项营养要求需要达标。

# 本文LP文件

MINIMIZE

Objective: 347.0x1+258.0x2+381.0x3+336.0x4+367.0x5+59.0x6+67.0x7+81.0x8+44.0x9+40.0x10+38.0x11+31.0x12+26.0x13+21.0x14+16.0x15+14.0x16+79.0x17+29.0x18+27.0x19+30.0x20+20.0x21+20.0x22+19.0x23+341.0x24+338.0x25+700.0x26+370.0x27+154.0x28+58.0x29+59.0x30+42.0x31+1332.0x32+69.0x33+64.0x34+53.0x35+581.0x36+514.0x37+816.0x38+130.0x39+106.0x40+338.0x41+135.0x42+207.0x43+26.0x44+205.0x45+378.0x46+47.0x47+168.0x48+167.0x49+192.0x50

Subject To:

C1: 6.7x1+9.7x2+36.0x3+8.0x4+5.0x5+3.3x6+1.9x7+2.0x8+2.0x9+4.1x10+2.8x11+1.0x12+0.7x13+0.8x14+0.7x15+0.4x16+1.9x17+2.4x18+1.0x19+3.0x20+4.5x21+1.0x22+1.0x23+4.0x24+9.0x25+1.0x26+4.2x27+1.4x28+0.9x29+1.0x30+2.0x31+22.4x32+0.2x33+1.0x34+1.0x35+20.0x36+22.5x37+16.9x38+20.0x39+20.1x40+14.6x41+11.1x42+14.0x43+2.1x44+15.0x45+13.0x46+9.6x47+18.7x48+15.8x49+15.1x50>=45

C2: 0.6x1+0.3x2+0.2x3+0.1x4+0.8x5+0.0x6+3.0x7+2.2x8+1.1x9+1.2x10+1.5x11+0.8x12+0.6x13+1.2x14+0.5x15+0.3x16+0.2x17+1.5x18+0.6x19+0.6x20+1.2x21+0.5x22+1.6x23+0.8x24+6.2x25+0.2x26+0.9x27+2.2x28+0.8x29+1.2x30+0.1x31+9.1x32+1.2x33+1.2x34+0.9x35+9.5x36+9.1x37+0.0x38+0.0x39+0.0x40+0.0x41+0.0x42+2.0x43+1.2x44+2.4x45+0.0x46+0.0x47+0.0x48+0.0x49+0.0x50>=30

C3: 0.0x1+0.0x2+0.0x3+0.0x4+0.0x5+0.0x6+20.0x7+25.0x8+10.0x9+61.0x10+47.0x11+56.0x12+72.0x13+57.0x14+75.0x15+28.0x16+56.0x17+52.0x18+47.0x19+76.0x20+23.0x21+77.0x22+52.0x23+10.0x24+8.0x25+44.0x26+76.0x27+241.0x28+58.0x29+87.0x30+8.0x31+0.0x32+12.0x33+90.0x34+68.0x35+16.0x36+30.0x37+0.0x38+0.0x39+0.0x40+0.0x41+0.0x42+23.0x43+9.0x44+23.0x45+0.0x46+0.0x47+0.0x48+0.0x49+0.0x50>=1000

C4: 527.5x1+172.0x2+3.3x3+2.5x4+1.6x5+3.2x6+37.3x7+2.1x8+39.1x9+28.3x10+19.1x11+14.9x12+56.2x13+103.3x14+8.3x15+42.9x16+77.4x17+175.2x18+4924.0x19+7.9x20+6.2x21+1.9x22+242.0x23+73.0x24+9.9x25+2.2x26+484.9x27+3.7x28+2.4x29+2.3x30+1.0x31+1.5x32+1.0x33+1.3x34+12.8x35+1.5x36+6.4x37+26.2x38+782.9x39+618.0x40+1421.0x41+471.0x42+0.3x43+87.0x44+28.0x45+227.0x46+229.0x47+297.0x48+0.0x49+0.0x50>=400

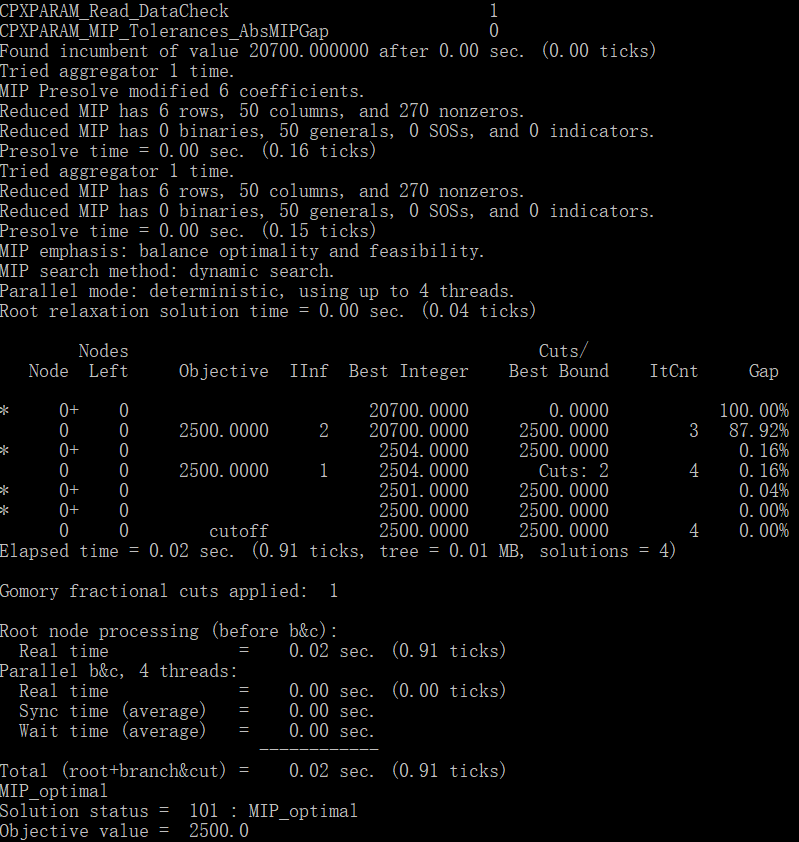
C5: 46.0x1+88.0x2+265.0x3+10.0x4+18.0x5+23.0x6+32.0x7+30.0x8+14.0x9+23.0x10+14.0x11+27.0x12+26.0x13+332.0x14+6.0x15+22.0x16+23.0x17+167.0x18+22.0x19+42.0x20+30.0x21+20.0x22+117.0x23+17.0x24+144.0x25+72.0x26+125.0x27+22.0x28+4.0x29+2.0x30+6.0x31+4.0x32+18.0x33+15.0x34+20.0x35+17.0x36+54.0x37+554.0x38+23.0x39+28.0x40+24.0x41+27.0x42+6.0x43+22.0x44+98.0x45+877.0x46+78.0x47+667.0x48+1720.0x49+1876.0x50>=1000

C6: 347.0x1+258.0x2+381.0x3+336.0x4+367.0x5+59.0x6+67.0x7+81.0x8+44.0x9+40.0x10+38.0x11+31.0x12+26.0x13+21.0x14+16.0x15+14.0x16+79.0x17+29.0x18+27.0x19+30.0x20+20.0x21+20.0x22+19.0x23+341.0x24+338.0x25+700.0x26+370.0x27+154.0x28+58.0x29+59.0x30+42.0x31+1332.0x32+69.0x33+64.0x34+53.0x35+581.0x36+514.0x37+816.0x38+130.0x39+106.0x40+338.0x41+135.0x42+207.0x43+26.0x44+205.0x45+378.0x46+47.0x47+168.0x48+167.0x49+192.0x50>=4000

END

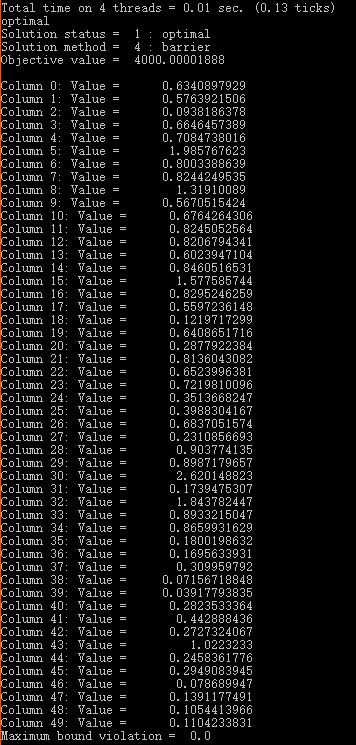
CPLEX处理程序

Cplex读取lp文件将自动分析，通过自己构建的调用cplex的python包分析生成lp。



调用cplex分析得到输出

除此以外，cplex还有自带的python调用示例，lpex2.py。通过lpex2.py可以将生成的lp文件根据不同的算法实现求解。下图为采取barrier算法得到的食物解



# 数据

Total time on 4 threads = 0.01 sec. (0.13 ticks)

optimal

Solution status = 1 : optimal

Solution method = 4 : barrier

Objective value = 4000.00001888

Column 0: Value = 0.6340897929

Column 1: Value = 0.5763921506

Column 2: Value = 0.0938186378

Column 3: Value = 0.6646457389

Column 4: Value = 0.7084738016

Column 5: Value = 1.985767623

Column 6: Value = 0.8003388639

Column 7: Value = 0.8244249535

Column 8: Value = 1.31910089

Column 9: Value = 0.5670515424

Column 10: Value = 0.6764264306

Column 11: Value = 0.8245052564

Column 12: Value = 0.8206794341

Column 13: Value = 0.6023947104

Column 14: Value = 0.8460516531

Column 15: Value = 1.577585744

Column 16: Value = 0.8295246259

Column 17: Value = 0.5597236148

Column 18: Value = 0.1219717299

Column 19: Value = 0.6408651716

Column 20: Value = 0.2877922384

Column 21: Value = 0.8136043082

Column 22: Value = 0.6523996381

Column 23: Value = 0.7219810096

Column 24: Value = 0.3513668247

Column 25: Value = 0.3988304167

Column 26: Value = 0.6837051574

Column 27: Value = 0.2310856693

Column 28: Value = 0.903774135

Column 29: Value = 0.8987179657

Column 30: Value = 2.620148823

Column 31: Value = 0.1739475307

Column 32: Value = 1.843782447

Column 33: Value = 0.8933215047

Column 34: Value = 0.8659931629

Column 35: Value = 0.1800198632

Column 36: Value = 0.1695633931

Column 37: Value = 0.309959792

Column 38: Value = 0.07156718848

Column 39: Value = 0.03917793835

Column 40: Value = 0.2823533364

Column 41: Value = 0.442888436

Column 42: Value = 0.2727324067

Column 43: Value = 1.0223233

Column 44: Value = 0.2458361776

Column 45: Value = 0.2949083945

Column 46: Value = 0.078689947

Column 47: Value = 0.1391177491

Column 48: Value = 0.1054413966

Column 49: Value = 0.1104233831

Maximum bound violation = 0.0

此处Column后的序号加2即为对应食物表中的食物序号。

Value值为食物摄入量。

总结叙述

本文调用cplex得到了barrier求得的解，而需要得到实际应用的结论需要进一步对数据进行比对，比如从高热食物和低热食物的摄入比对，从蔬菜类食物到肉类食物比对，结合中国人的饮食习惯等获取实际有价值的结论。