**程序编写过程说明**

我们已经安装好了python程序，并且在Github上完成了账号的注册与管理。因此我们可以开始编写程序，从而在实践中学习python的使用。本次作业的内容为《编程导论》中16.4统计方法并不能代替数据。下面作者将在文档中说明该程序编写中的注意事项。

1. **编写前的插件准备**

在编写开始之前，作者并不知道有哪些插件将被使用，在编写过程中发现有需求时才进行安装。下列插件为编写完成后确认被使用到的插件：

xlrd

pylab

numpy

scipy（必须同时有numpy与mkl插件才能顺利完成安装）

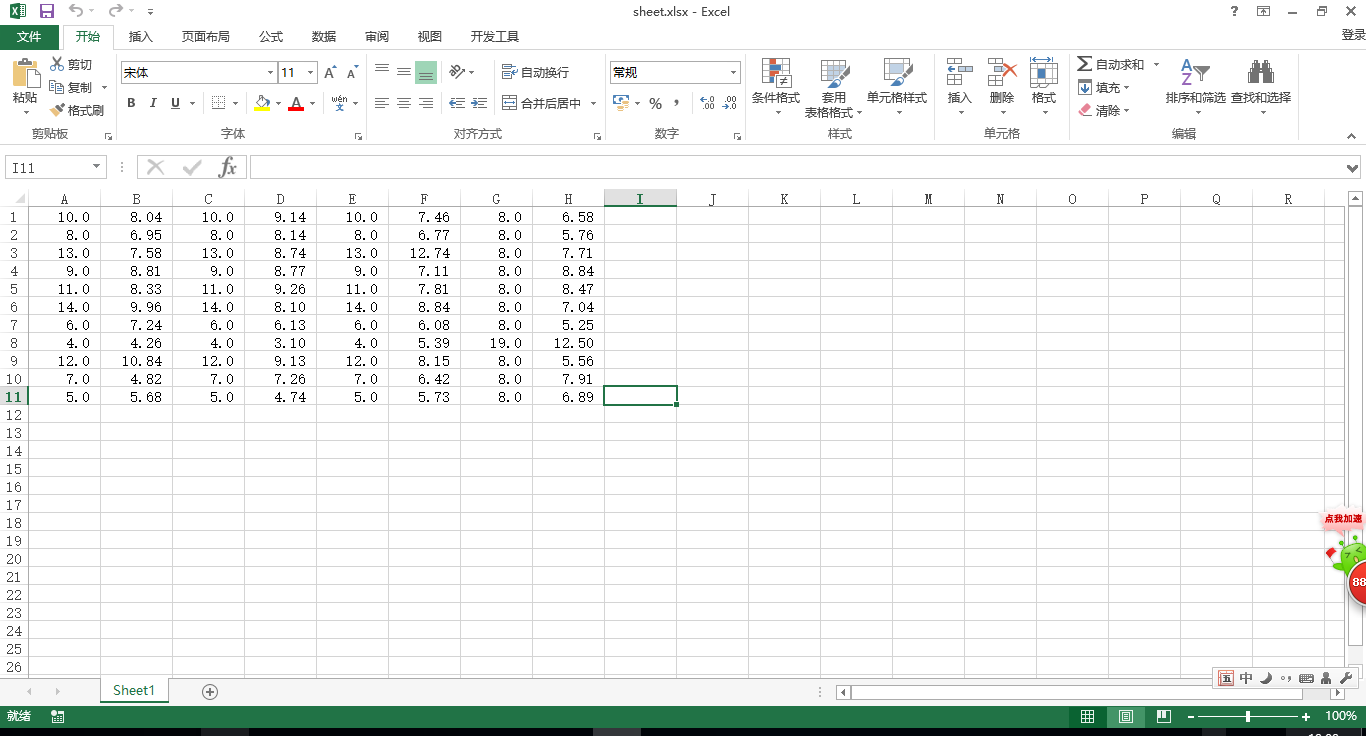
由于numpy以及scipy插件在以前的作业中已经完成安装，此处仅需要完成xlrd与pylab的安装。

1. **程序的编写**

本程序主要包括三个部分：从excel中读取列表数据，对四组数据分别进行数学统计，完成线性回归与图表绘制。

**2.1 从excel中读取列表数据**

首先我们需要将数据写入Excel中，并为该文件取一个英文名。



注：由于python为英文软件，对中文识别能力很差，故不要将此Excel文件放于含中文的目录中。

作者于该网站中学习了从Excel中读取数据的方法：

<http://www.jb51.net/article/42635.htm>

通过学习与理解，作者在eclipse中编写了如下代码：

import xlrd

fname = "D:\homework\sheet.xlsx"

bk = xlrd.open\_workbook(fname)

shxrange = range(bk.nsheets)

try:

sh = bk.sheet\_by\_name("Sheet1")

except:

print( "no sheet in %s named Sheet1" % fname)

#获取行数

nrows = sh.nrows

#获取列数

ncols = sh.ncols

print( "nrows %d, ncols %d" % (nrows,ncols))

col\_list=[]

#获取各行数据

for i in range(0,ncols):

col\_data = sh.col\_values(i)

col\_list.append(col\_data)

注：col\_list为一个无限大的数组，由写入的数据来确定元数及长度。

此处用到了xldr插件，如未安装此插件侧本步无法完成。

**2.2 对四组数据分别进行数学统计**

numpy插件为python的一个数学计算插件，通过其中的下属函数我们可以很方便的进行数学计算：

平均值 numpy.var( )

方差 numpy.mean( )

相关系数 numpy.correlate( , )

由于作者没有想出如何将四组计算用For循环进行表示，故此处代码重复了四次，仅对涉及到的参数进行了改变。因此该部分略显烦杂，代码如下：

import numpy

x1=col\_list[0]

xp1=numpy.var(x1)

xf1=numpy.mean(x1)

print('表1中x平均值=',xp1)

print('表1中x方差=',xf1)

y1=col\_list[1]

yp1=numpy.var(y1)

yf1=numpy.mean(y1)

print('表1中y平均值=',yp1)

print('表1中y方差=',yf1)

xy1=numpy.correlate(col\_list[0],col\_list[1])

print('表1中xy相关系数=',xy1[0])

x2=col\_list[2]

xp2=numpy.var(x2)

xf2=numpy.mean(x2)

print('表2中x平均值=',xp2)

print('表2中x方差=',xf2)

y2=col\_list[3]

yp2=numpy.var(y2)

yf2=numpy.mean(y2)

print('表2中y平均值=',yp1)

print('表2中y方差=',yf1)

xy2=numpy.correlate(col\_list[2],col\_list[3])

print('表2中xy相关系数=',xy2[0])

x3=col\_list[4]

xp3=numpy.var(x3)

xf3=numpy.mean(x3)

print('表3中x平均值=',xp3)

print('表3中x方差=',xf3)

y3=col\_list[5]

yp3=numpy.var(y3)

yf3=numpy.mean(y3)

print('表3中y平均值=',yp3)

print('表3中y方差=',yf3)

xy3=numpy.correlate(col\_list[4],col\_list[5])

print('表3中xy相关系数=',xy3[0])

x4=col\_list[6]

xp4=numpy.var(x4)

xf4=numpy.mean(x4)

print('表4中x平均值=',xp4)

print('表4中x方差=',xf4)

y4=col\_list[7]

yp4=numpy.var(y4)

yf4=numpy.mean(y4)

print('表4中y平均值=',yp4)

print('表4中y方差=',yf4)

xy4=numpy.correlate(col\_list[6],col\_list[7])

print('表4中xy相关系数=',xy4[0])

代码输出结果如下：

表1中x平均值= 10.0

表1中x方差= 9.0

表1中y平均值= 3.75206280992

表1中y方差= 7.50090909091

表1中xy相关系数= 797.6

表2中x平均值= 10.0

表2中x方差= 9.0

表2中y平均值= 3.75206280992

表2中y方差= 7.50090909091

表2中xy相关系数= 797.59

表3中x平均值= 10.0

表3中x方差= 9.0

表3中y平均值= 3.74783636364

表3中y方差= 7.5

表3中xy相关系数= 797.47

表4中x平均值= 10.0

表4中x方差= 9.0

表4中y平均值= 3.74840826446

表4中y方差= 7.50090909091

表4中xy相关系数= 797.58

**2.3 完成线性回归与图表绘制**

该部分需要的插件有scipy下属stats，numpy下属np以及pylab。

此处代码的主要参与来源为：

<http://www.dataguru.cn/thread-183366-1-1.html>

由于作者没有想出如何将四组图表用For循环进行表示，故此处代码重复了四次，仅对涉及到的参数进行了改变，代码如下：

from scipy import stats

import numpy as np

import pylab

x1=np.array(col\_list[0])

y1=np.array(col\_list[1])

slope, intercept, r\_value, p\_value, slope\_std\_error = stats.linregress(x1, y1)

predict\_y1 = intercept + slope \* x1

pred\_error = y1 - predict\_y1

degrees\_of\_freedom = len(x1) - 2

residual\_std\_error = np.sqrt(np.sum(pred\_error\*\*2) / degrees\_of\_freedom)

# Plotting

pylab.figure(1)

pylab.plot(x1, y1, 'o')

pylab.plot(x1, predict\_y1, 'k-')

pylab.show()

x2=np.array(col\_list[2])

y2=np.array(col\_list[3])

slope, intercept, r\_value, p\_value, slope\_std\_error = stats.linregress(x2, y2)

predict\_y2 = intercept + slope \* x2

pred\_error = y2 - predict\_y2

degrees\_of\_freedom = len(x2) - 2

residual\_std\_error = np.sqrt(np.sum(pred\_error\*\*2) / degrees\_of\_freedom)

# Plotting

pylab.figure(2)

pylab.plot(x2, y2, 'o')

pylab.plot(x2, predict\_y2, 'k-')

pylab.show()

x3=np.array(col\_list[4])

y3=np.array(col\_list[5])

slope, intercept, r\_value, p\_value, slope\_std\_error = stats.linregress(x3, y3)

predict\_y3 = intercept + slope \* x3

pred\_error = y3 - predict\_y3

degrees\_of\_freedom = len(x3) - 2

residual\_std\_error = np.sqrt(np.sum(pred\_error\*\*2) / degrees\_of\_freedom)

# Plotting

pylab.figure(3)

pylab.plot(x3, y3, 'o')

pylab.plot(x3, predict\_y3, 'k-')

pylab.show()

x4=np.array(col\_list[6])

y4=np.array(col\_list[7])

slope, intercept, r\_value, p\_value, slope\_std\_error = stats.linregress(x4, y4)

predict\_y4 = intercept + slope \* x4

pred\_error = y4 - predict\_y4

degrees\_of\_freedom = len(x4) - 2

residual\_std\_error = np.sqrt(np.sum(pred\_error\*\*2) / degrees\_of\_freedom)

# Plotting

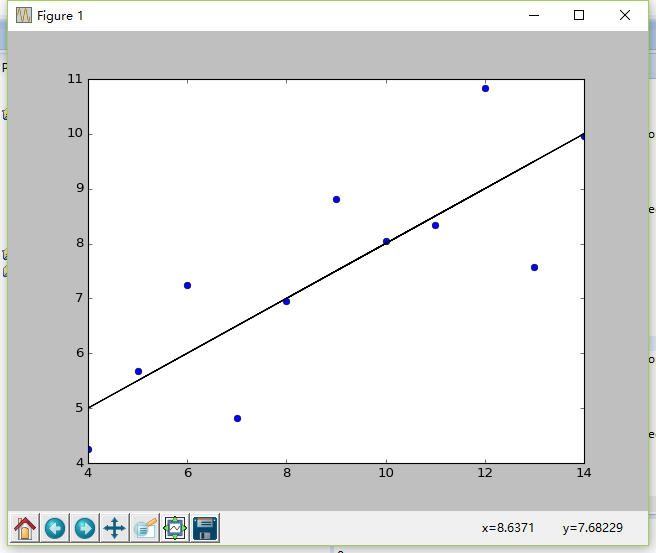
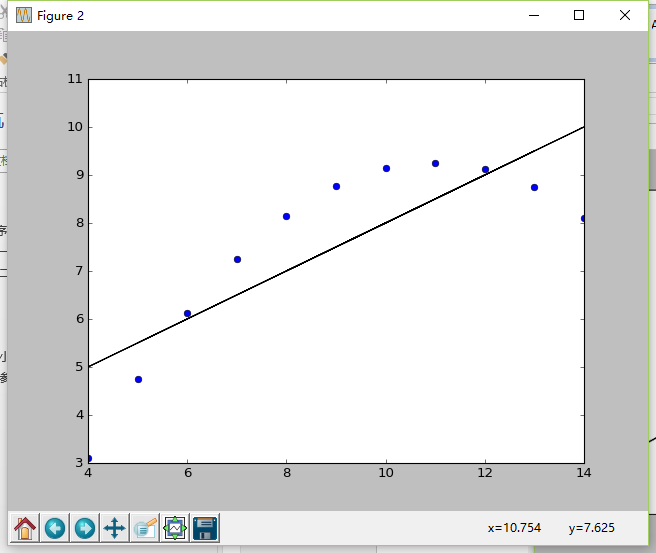
pylab.figure(4)

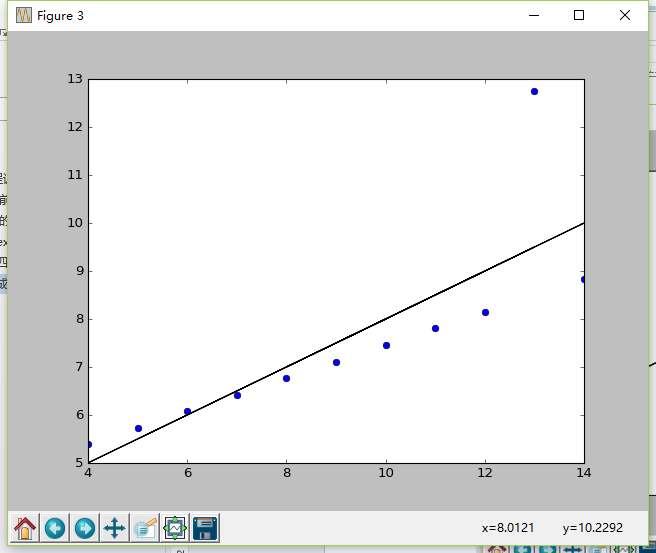
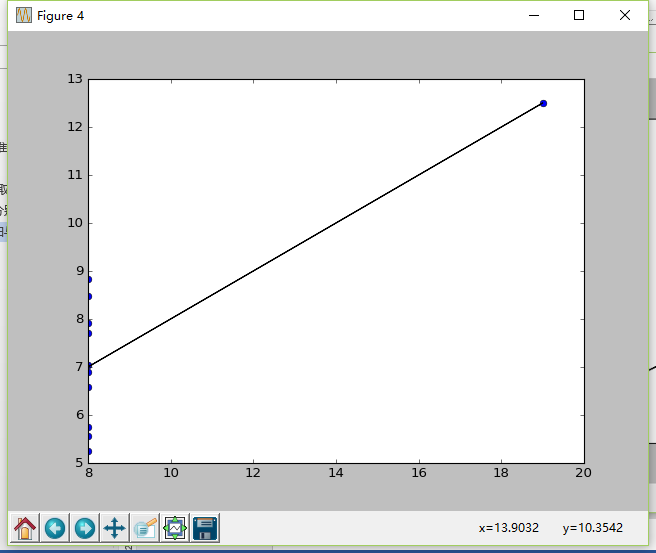
pylab.plot(x4, y4, 'o')

pylab.plot(x4, predict\_y4, 'k-')

pylab.show()

代码完成后运行，可以得到图表结果：

注：此处的pylab.savefig()函数因多次调试无法使用，故要保存图表请手动点击保存。

**小结**

通过本程序的编写，作者对于python代码的编写模式有了一定的认识，并且发现通过网络查找可以帮助我们在编写程序时省下不少的力气，还能少走很多弯路。但是在借用他人的代码时也要注意理解，只有在理解的基础上进行应用，才能真正的掌握相关的知识。

**参考文献**

1. Python读写Excel的实例

[http://www.jb51.net/article/42635.htm](http://www.jb51.net/article/42635.htm%20)

1. 使用python绘制一元线性回归

[http://www.dataguru.cn/thread-183366-1-1.html](http://www.dataguru.cn/thread-183366-1-1.html%20)