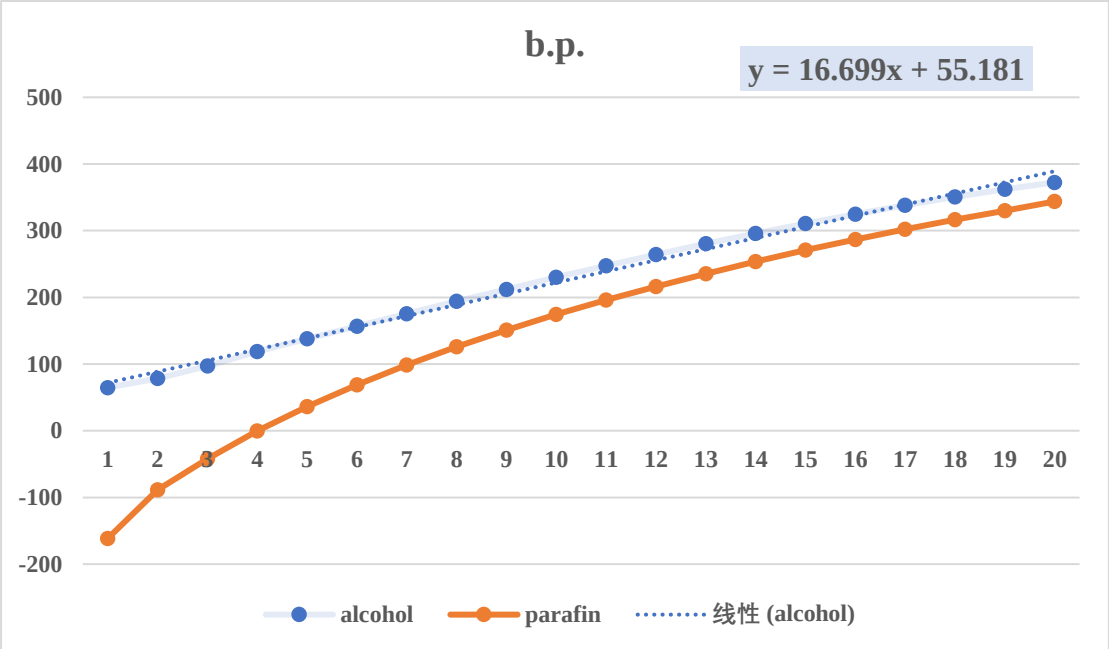
Part 1: Components Transformation Temperature Analysis

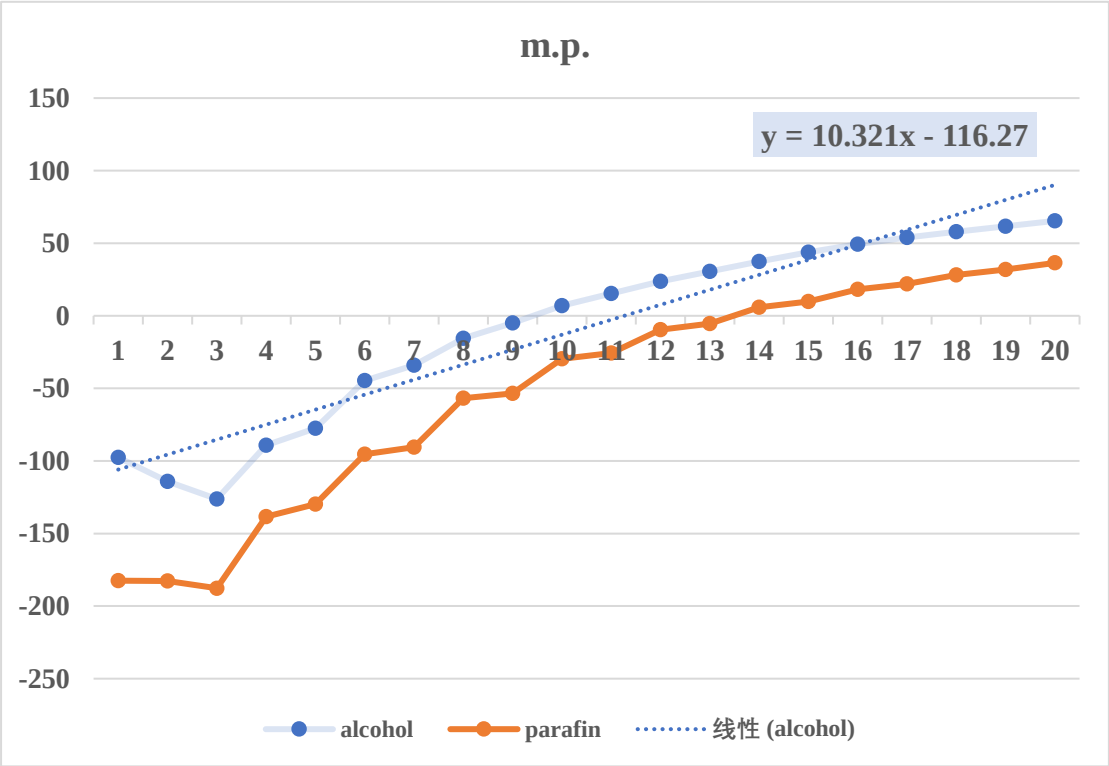
Boiling point of alcohol and paraffin with different C atoms:

Number of C	Alcohol/℃	Paraffin/℃
1	64.7	-161.49
2	78.29	-88.6
3	97.2	-42.04
4	118.75	-0.5
5	137.75	36.07
6	156.75	68.73
7	175.45	98.43
8	193.95	125.68
9	212.05	150.82
10	229.85	174.155
11	247.15	195.928
12	263.95	216.323
13	280.25	235.466
14	295.85	253.577
15	310.75	270.685
16	324.85	286.864
17	338.15	302.15
18	350.45	316.71
19	361.95	329.9
20	372.35	343.78



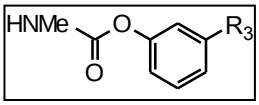
Melting point of alcohol and paraffin with different C atoms:

Number of C	Alcohol/℃	Paraffin/℃
1	-97.68	-182.456
2	-114.1	-182.798
3	-126.2	-187.68
4	-89.3	-138.29
5	-77.59	-129.73
6	-44.6	-95.32
7	-34	-90.58
8	-15.5	-56.77
9	-5	-53.49
10	6.9	-29.64
11	15.3	-25.579
12	23.8	-9.582
13	30.6	-5.39
14	37.5	5.86
15	43.9	9.922
16	49.2	18.158
17	53.9	21.984
18	57.9	28.16
19	61.7	31.89
20	65.4	36.43



（All data from Aspen Properties database）

Part 2　Hansch Model



为方便操作未采用 Excel，使用 python 编了一个小程序实现拟合。

以选取第二个分子为例：

原始数据：

Substituents	IC_{50} (μM)	$1/IC_{50}$	$\log(1/IC_{50})$	π^2	π	σ_m	**MR**
H	**200**	0.01	-2.30	0.00	0.00	0.00	1.03
F	**85**	0.01	-1.93	0.02	0.14	0.34	0.92
Cl	**50**	0.02	-1.70	0.50	0.71	0.37	6.03
Br	**13**	0.08	-1.11	0.74	0.86	0.39	8.88
I	**7.0**	0.14	-0.85	1.25	1.12	0.35	13.94
CH_3	**14**	0.07	-1.15	0.31	0.56	-0.07	5.65
CH_2CH_3	**4.8**	0.21	-0.68	1.04	1.02	-0.07	10.30
$CH(CH_3)_2$	**0.34**	2.94	0.47	2.34	1.53	-0.07	14.98
$CH_2CH(CH_3)_2$	**0.16**	6.25	0.80	3.92	1.98	-0.10	19.62
OCH_3	**22**	0.05	-1.34	0.00	-0.02	0.12	7.87
$OCH(CH_3)_2$	**9.2**	0.11	-0.96	0.13	0.36	0.10	17.06
$N(CH_3)_2$	**8.0**	0.13	-0.90	0.03	0.18	-0.15	15.55

记录为“3-Substituted.mat”

运行记录：

```
In [36]: runfile('C:/Users/neetsaki/Desktop/分子设计与产品工程/regression_1.py', wdir='C:/Users/
neetsaki/Desktop/分子设计与产品工程')

input regression molecule number(2~4): 3
Coefficients:
 [[ 0.08665577  0.7102806  -0.81447406  0.05288364]]
Intercept:
 [-1.96428692]
Mean squared error: 0.04
Variance score: 0.95
(3, 6, [array([-1.69897]), array([-1.14612804])], array([[-1.39877157],
       [-1.18354878]]))
```

说明：input regression molecule number(2~4) 对应于选择“2-Substituted.mat”，“3-Substituted.mat”,“4-Substituted.mat”其中一个母体；

Coefficients 对应拟合的系数

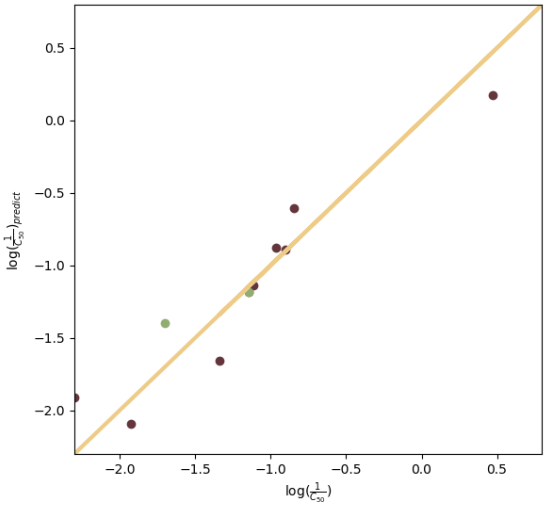
Intercept 对应拟合的截距

Mean squared error 对应拟合的均方差

Variance score 对应拟合的 R 方

最后一列前两个数代表抽出来做 TEST 集的两个集团的编号，此处对应 Br 和 CH_2CH_3

后面两个向量代表此两个数的真值与预测值；



输出图片如右图所示：

对角线为 y=x，点出的两个绿点为 TEST 集预测与实际对比，点出的若干红点为 TRAIN 集预测与实际对比。

据此可知结果为：

$$\log\left(\frac{1}{C_{50}}\right) = 0.05288\pi^2 - 0.814474\pi + 0.71028\sigma_m + 0.086655MR - 1.96428$$

Experimental Activity log(1/IC$_{50}$)	Predicted Activity log(1/IC$_{50}$)
-2.30	-1.87502535
-1.93	-1.756052112
-1.70	-1.730566482
-1.11	-1.579111992
-0.85	-1.353589508
-1.15	-1.963921122
-0.68	-1.897200228
0.47	-1.838266128
0.80	-1.740484668
-1.34	-1.180760918
-0.96	-0.701275092
-0.90	-0.868228758

注：其中 TEST 集合为避免人为选取影响，采用随机种子产生两个数来选取。（由于随机选取可能与其他同学作业相重复，不过此处理已覆盖所有可能性）
本程序成功实现所提供所有分子随机选取一~两个分子作为 TEST 集，其余作为 TRAIN 集最小二乘法多元拟合的任务。
所有结果均已上传至：https://github.com/neetsaki/Molecule-design-homework
其他运行结果截图如下：

In [6]:

```
import matplotlib.pyplot as plt
import numpy as np
from sklearn import linear_model
import scipy.io as scio
from sklearn.metrics import mean_squared_error, r2_score
import random
```

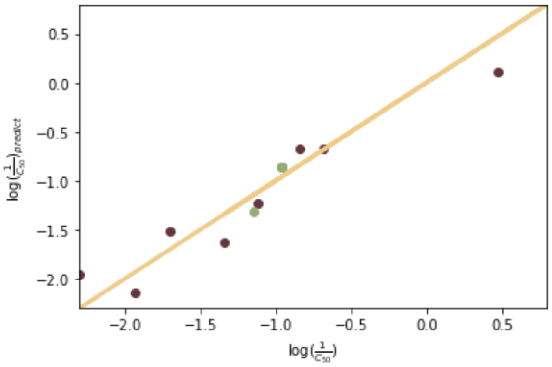
In [10]:

```
def regression(n,m):
    data = scio.loadmat(str(n)+"-substituted.mat")
    reg = linear_model.LinearRegression()
    X=data['X']
    X_o=data['X']
    y=data['y']
    y_o=data['y']
    seed1=random.randint(0,len(y)-1)
    seed2=random.randint(0,len(y)-1)
    X=np.delete(X,seed1,0)
    y=np.delete(y,seed1,0)
    X=np.delete(X,seed2,0)
    y=np.delete(y,seed2,0)
    #reg.fit(X,y)
    #X=np.delete(a,m,axis=0)
    #y=np.delete(b,m,axis=0)
    reg.fit(X,y)
    y_pred0= reg.predict(X)
    y_pred = reg.predict([X_o[seed1],X_o[seed2]])

    #y_pred_o=np.delete(y_pred,m,axis=0)

    #LinearRegression(copy_X=True, fit_intercept=True,n_jobs=None,normalize=False)
    # The coefficients
    print('Coefficients: \n', reg.coef_)
    #intercept
    print('Intercept: \n', reg.intercept_)
    # The mean squared error
    print("Mean squared error: %.2f"% mean_squared_error(y,y_pred0))
    # Explained variance score: 1 is perfect prediction
    print('Variance score: %.2f' % r2_score(y,y_pred0))
    # Plot outputs
    plt.xlim(min(y),max(y))
    plt.ylim(min(y),max(y))
    plt.scatter(y,y_pred0,  color='#64363C')
    plt.scatter([y_o[seed1],y_o[seed2]],y_pred,  color='#91AD70')
    plt.plot(y,y, color='#EECC88', linewidth=3)
    plt.xlabel(r'$\log(\frac{1}{C_{50}})$')
    plt.ylabel(r'$\log(\frac{1}{C_{50}})_{predict}$')
    return seed1+1,seed2+1,[y_o[seed1],y_o[seed2]],y_pred
```

In [3]:

```
n=input("input regression molecule number(2~4): ")
#regression(n,5)
print(regression(n,5))
```

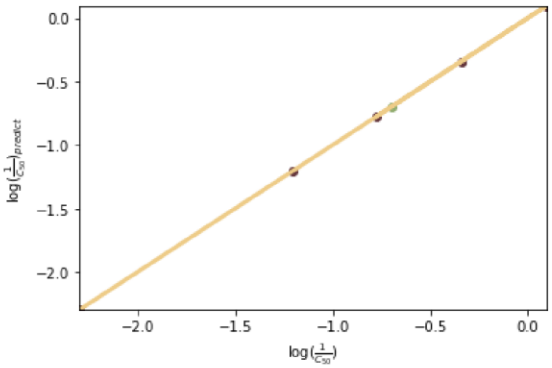
```
input regression molecule number(2~4): 3
Coefficients:
 [[ 0.22076964  0.40987645 -0.71932455  0.06224282]]
Intercept:
 [-2.01989456]
Mean squared error: 0.05
Variance score: 0.94
(6, 11, [array([-1.14612804]), array([-0.96378783])], array([[-1.31910576],
       [-0.8537973 ]]))
```



In [12]:

```
n=input("input regression molecule number(2~4): ")
#regression(n,5)
print(regression(n,5))
```

```
input regression molecule number(2~4): 2
Coefficients:
 [[ 0.63873489 -0.51021355  3.42377692  0.07512225]]
Intercept:
 [-2.37840591]
Mean squared error: 0.00
Variance score: 1.00
(7, 3, [array([0.16115091]), array([-0.69897])], array([[-0.85533952],
       [-0.69888666]]))
```



In [11]:

```
n=input("input regression molecule number(2~4): ")
#regression(n,5)
print(regression(n,5))
```

```
input regression molecule number(2~4): 4
Coefficients:
 [[-0.42145708  0.70352722 -1.18284299  0.00183589]]
Intercept:
 [-2.05970621]
Mean squared error: 0.02
Variance score: 0.60
(7, 1, [array([-1.90308999]), array([-2.30103])], array([[-2.20143807],
       [-2.05781525]]))
```