B 题　“拍照赚钱”的任务定价

摘要

随着科技的快速发展，一种“拍照赚钱”的 APP 流行起来，人们可以通过该 APP 接受一些拍照任务来获得报酬。但在这个运行模式中，会员能否找到合适的任务、每个任务能否被尽快完成取决于 APP 对任务的定价，所以科学合理的定价标准才可以使 APP 高效运行并获利。本文结合任务完成数据以及会员信息数据建立新的的定价模型，并对其进行评估。

首先，我们通过图形展示对任务数据以及会员信息的分布进行分析，发现完成与未完成的任务呈现一定的聚集特征；通过在特定区域通过 pearson 相关系数来分析影响价格的因素，我们得出定价主要与该地区会员的信誉值、任务数量以及会员的数量呈现一定程度的负相关，任务失败的主要原因为任务的定价过低；同时，我们计算任务与会员位置间的两两距离，并结合任务中会员密度对任务成功与失败的影响，发现任务在一定距离内的会员密度越大，则任务有更高的概率被完成。

然后，为了对价格进行评估，我们基于近邻思想建立了定价评估模型。我们又基于回归学习的思想建立了岭回归定价模型、在近邻基础上结合已有信息构建了近邻定价模型，并利用这两个定价模型对任务数据进行重新定价，再利用评估模型对它们进行评估，其中近邻定价模型对任务定价达到了很好的效果。

接着，我们利用一种基于密度的 DBSCAN 聚类模型将任务划分为核心任务点、边界任务点以及噪音点。将核心任务点打包，优化近邻定价模型并构建近邻打包定价模型，通过对其应用评估，发现该模型具有很好的实际应用价值。

最后，我们分析了一套新的任务位置数据并利用近邻打包定价模型对任务进行定价与打包，并采用评估模型对其进行评估。

此外，我们评价各个模型的优缺点，并提出了改进方案以及其应用展望。

关键词：相关性分析　　岭回归　　近邻定价　　DBSCAN 聚类

一．问题重述

随着科技的快速发展，手机的性能和功能有了质的飞跃，手机不仅可以拍照，甚至可以替代照相机拍出各种高清优美的照片，而且大家可以随时随地在移动互联网上分享自己的照片。

于是一种“拍照赚钱”的 APP 应运而生，手机用户只需要下载 APP 凭注册成为会员，就可以在地图上找例如拍商铺门面或大厦信息的任务，上传照片完成任务就会获得该任务对应的酬金。这种自助式劳务众包平台，不但可以为企业提供各种商业检查和信息搜集，而且可以为很多爱好拍照的人带来额外的收入解决一些人的就业问题。 在这个运行模式中，会员能否找到合适的任务、每个任务能否被尽快完成决定了 APP 的效率和利润，然而影响这些问题的核心因素就是 APP 对任务的定价。科学合理的定价可以使 APP 高效运行并获利，本文主要解决如下问题：

1. 分析附件一中的数据找出任务定价规律并分析任务未完成的原因。

2. 设计新的定价方案并对附件一中的项目定价，和原来的定价方案进行比较。

3. 任务的分布位置和密度会影响任务完成情况，考虑任务高密度地区联合打包发布任务，修改新制定的定价方案，分析对任务完成情况的影响。

4. 用指定的方案对附件三中的任务进行定价，并分析和评价效果 。

二．问题分析

针对问题一，我们首先想到将问题中任务点与会员位置的 GPS 坐标映射到实际的地理位置中，看它们大致的分布情况；然后在同一分区中考察任务的定价规律以及任务未完成的原因；此外我们根据任务与会员位置的信息使用一种基于近邻思想的方法分析这两个问题。

针对问题二，我们首先打算采用一个岭回归模型在各个分区中对价格与信誉值、任务数量以及会员数量这三个因素进行回归来对实现对任务定价；之后我们利用近邻思想建立一个对定价的一个预测与评估的模型；在定价评估模型的基础上我们可以衍生出一种基于近邻的定价模型，并对这两个模型进行评估。

针对问题三，我们拟采用一种基于密度的 DBSCAN 聚类算法，对任务的位置进行聚类，进行打包的分析；之后建立近邻打包模型，并对该模型的结果进行

评估。

针对问题四，我们则利用如上建立的岭回归定价模型、近邻定价模型以及近邻打包模型对新数据进行任务定价，并利用价格评估模型对这些结果进行评估与分析。所有问题的解决流程如下图：

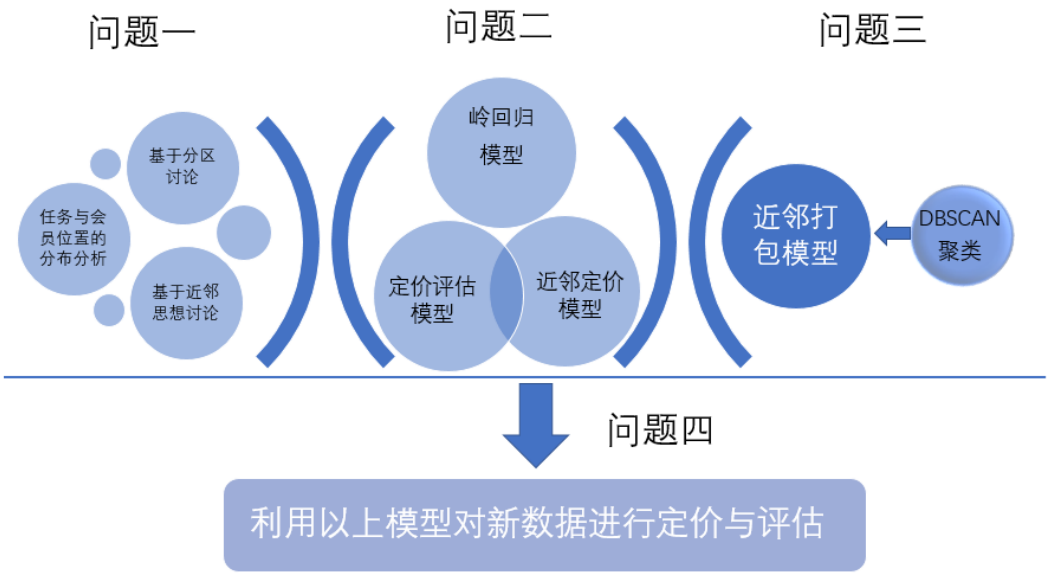

图 1、本文流程图

以上所有模型算法详细的分析过程、具体描述、应用过程、结果分析以及评估见第五节。

三．模型假设

1、假定会员在选择任务时，更趋向于选择距离最近的任务，价格更高的任务。这能使得 5.2.2 的近邻定价模型中会员对α值选择具有意义。

2、假定会员将概率性的选择可接受的任务，选择过程中不受心情、天气原因等其他因素的干扰。

3、假定会员在完成任务时受信誉值的影响，高于总体会员信誉值一定分位点的人将肯定完成任务，信誉低的人将以一定概率完成任务，完成任务过程中不受其它因素的干扰。这与假设 2 共同决定了价格评价模型中任务完成的概率的有效性。

4、假定所有会员可以接受的任务距离相同。

四．符号说明

符号	说明
A	任务集合
B	会员集合
q	会员的预订任务限额
p	任务定价
K	Kruskal 统计量
α	可选任务对会员的吸引程度
τ	打包定价系数

注：以上表列出了重要符号及其说明，其余符号在公式中有详细说明。

五．模型的建立及求解

5.1、定价规律分析

5.1.1、分布总体描述

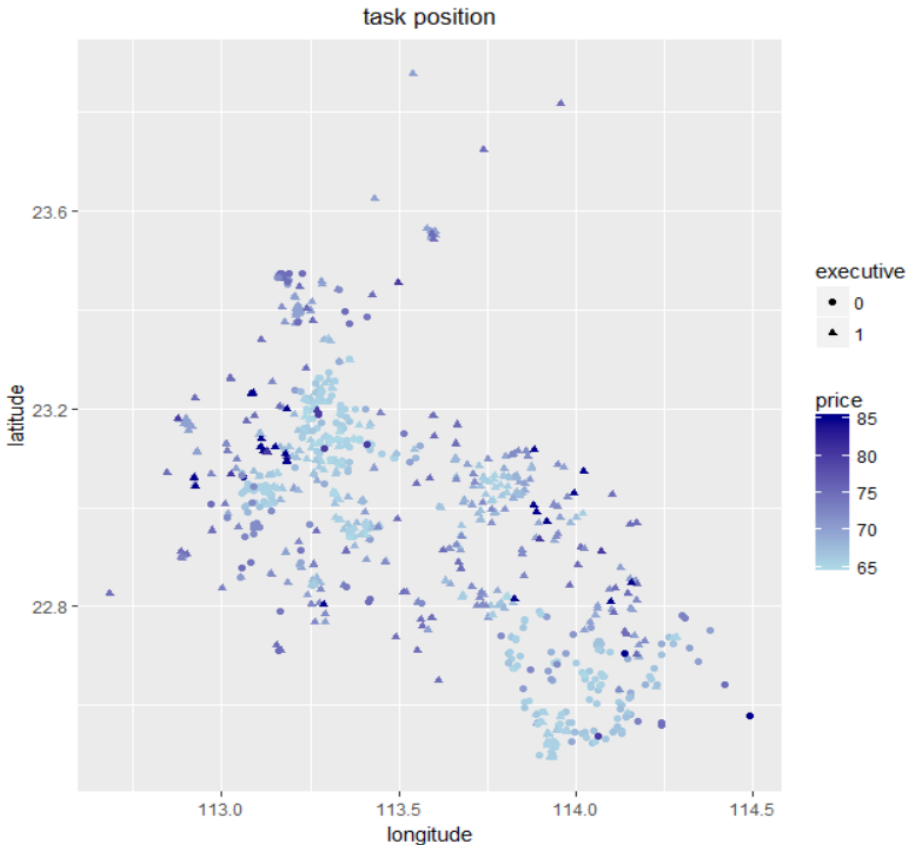

图 2、任务的分布情况

我们首先利用 R 语言[1][3]根据附件一的任务位置信息绘制出任务分布图，并依照标价的大小赋予一定梯度的颜色。分布图如图 2。图中深蓝色表示标价高的任务，浅蓝色表示标价低的任务；圆圈表示未完成任务，三角形表示已完成任务。

由图中可以看出颜色深或浅的任务点趋向于聚集在某一固定的区域，且完成与未完成点也称聚集分布。为获得更精确的地理位置，我们采用 R 的 REmap 包[2]中的函数来将任务位置的经纬度信息映射到地图上，得图 3，可以看出这些任务在某些区域集中分布。

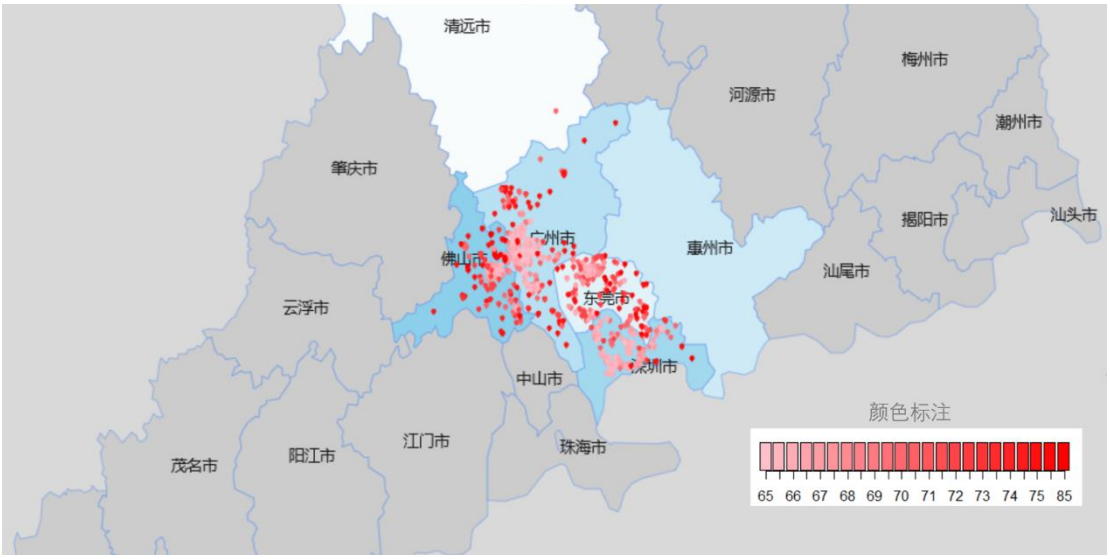

图 3、任务在地图上的分布情况

5.1.2、基于分区分析

为得更确切区域进行后续分析，我们使用了 XGeocoding 工具将数据中的经纬度解析为具体地址，数据详见附录。

设任务共n个，分布在m个地区，在第i个地区有n_i个。为研究地域对任务定价的影响，我们对任务价格在各个地区之间做差异性分析。由于价格在各个地区不服从正态分布，我们采用完全随机设计的 Kruskal-Wallis 秩和检验[4]进行分析。

将这n个任务对应的标价w按高低排序得到各自的秩，记第i个地区的n_i个任务的秩为

$$R_{i1}, R_{i2}, \cdots, R_{in_i} \quad (i = 1,2, \cdots m) \qquad \text{（公式 1-1）}$$

假设：H_0:各地区的标价无显著差异；秩平均值如下：

$$R_{i.} = \frac{R_{i1}+R_{i2}+\cdots+R_{in_i}}{n_i}(i = 1,2,\cdots m) \qquad \text{（公式 1-2）}$$

$$R_{..} = \frac{1}{n}\sum_{i=1}^{m}\sum_{j=1}^{n_i} R_{ij} = \frac{n+1}{2} \qquad \text{（公式 1-3）}$$

其中$R_{i.}$是第i组个体的秩的平均值，$R_{..}$是总的平均值。

构造如下统计量：

$$K = \frac{12}{n(n+1)}\sum_{i=1}^{m} n_i(R_{i.} - \frac{n+1}{2})^2 \qquad \text{（公式 1-4）}$$

经 R 语言 kruskal.test()函数计算得$k = 363.76, p < 2.2 \times 10^{-16}$。所以拒绝$H_0$认为价格在各区域分布是有差异的。

绘制任务价格在各地区的分布如图 4 及地区的编号如表 1：

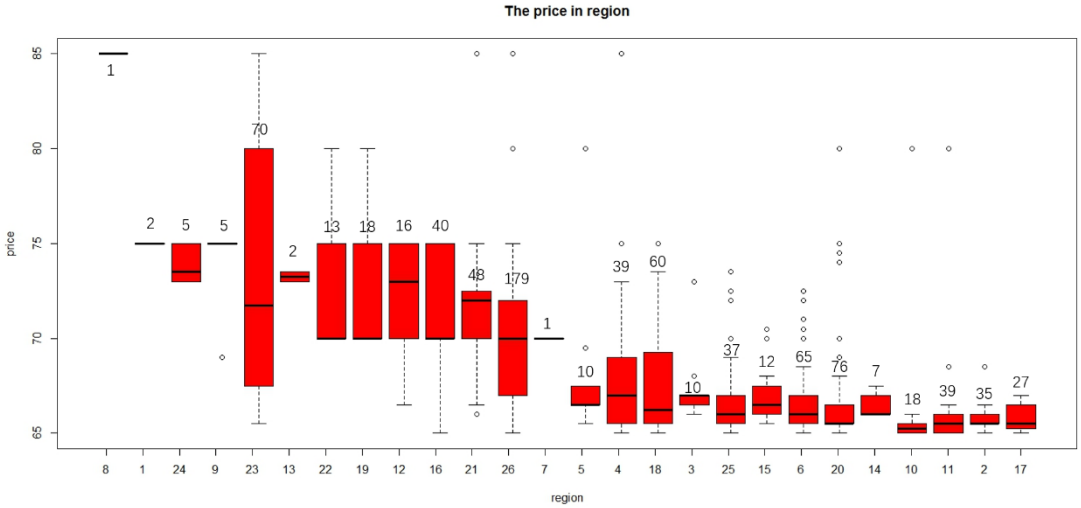

图 4、任务定价在各地区的分布

该图每个箱体由下到上分别表示数据集中的最小值，第一四分位数，中值，第三四分位数和最大值，箱体旁边的数字表示在该区域的任务数量。由于某些地区任务量较少，某些箱体可能呈现一条线。

由上述图表我们易得如下规律：一般小城市或市郊区的任务较少，而且价格较高，如惠州市博罗县和深圳市盐田区；而像一些发达区域，因为商机较多，所以会有较多的任务，但标价也相对较低些，如广州海珠区和深圳南山区。

下面我们用上述同样的方法来分析会员的分布信息，绘制如下图：

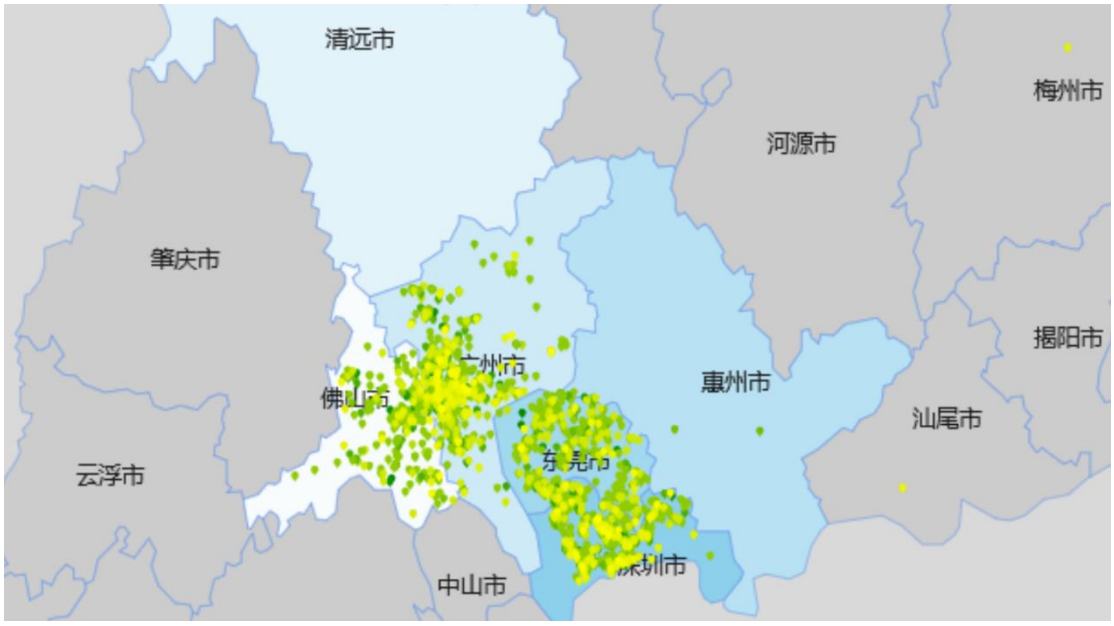

图 5、会员在地图上的分布情况

该图中的点表示附件二会员的位置坐标信息，点的颜色由黄色到绿色渐变，标识会员的信誉值由低至高。

表 1、各地区标号对应位置

地区	该地区编号	地区	该地区编号
广东省深圳市盐田区	1	广东省广州市荔湾区	14
广东省深圳市南山区	2	广东省广州市黄埔区	15
广东省深圳市罗湖区	3	广东省广州市花都区	16
广东省深圳市龙岗区	4	广东省广州市海珠区	17
广东省深圳市福田区	5	广东省广州市番禺区	18
广东省深圳市宝安区	6	广东省广州市从化市	19
广东省清远市佛冈县	7	广东省广州市白云区	20
广东省惠州市博罗县	8	广东省佛山市顺德区	21
广东省广州市增城市	9	广东省佛山市三水区	22
广东省广州市越秀区	10	广东省佛山市南海区	23
广东省广州市天河区	11	广东省佛山市高明区	24
广东省广州市南沙区	12	广东省佛山市禅城区	25
广东省广州市萝岗区	13	广东省东莞市	26

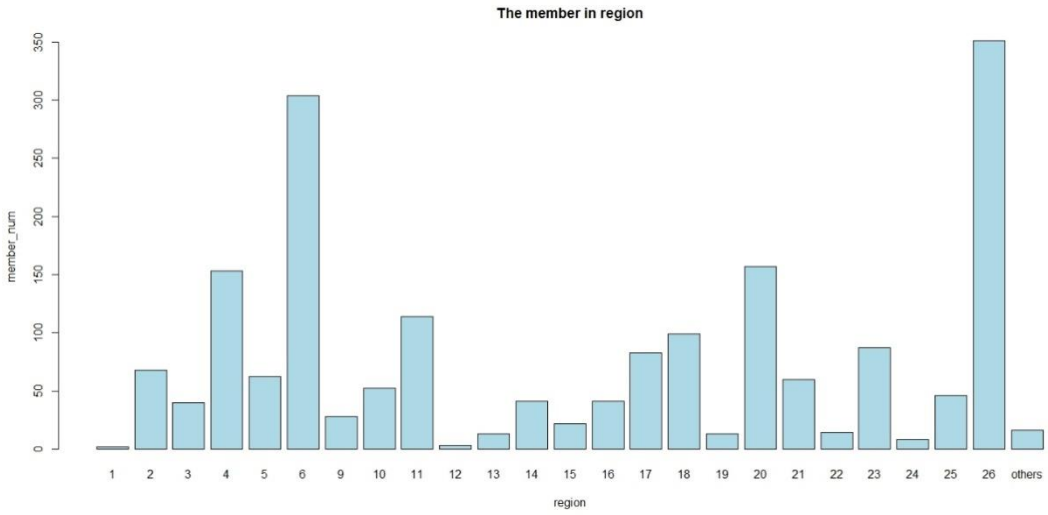

图 6

图 6 中横轴地区编号所对应的地区与表 1 相同，纵轴为相应地区会员的数量。从上图可以看出队员数量在地区上有一定的差异。

为更详细的了解任务定价规律以及分析任务未完成原因，我们整合附件一与附件二的信息，计算出每个地区的任务数量，会员数量，会员信誉平均值，任务定价平均值及任务完成比率。整理结果如表 2：（表中仅列出任务量大于 10 的地区）

我们对每个地区的任务数量，会员数量，会员信誉平均值，任务定价平均值及任务完成比率这 5 个观测两两做相关性分析，计算他们之间的 pearson 相关系数[5]，计算方法如下：

$$cov(X,Y) = \frac{\sum_n^i (X_i - \bar{X})(Y_i - \bar{Y})}{n-1} \quad \text{（公式 1-5）}$$

$$Pearson\_cor = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \quad \text{（公式 1-6）}$$

式中X, Y代表进行相关性分析的两个向量，$cov(X,Y)$表示二者之间的协方差，σ_X, σ_Y表示两个向量的方差，$Pearson_cor$表示二者的 pearson 相关系数，反应二者的相关性情况。

经 R 语言中的 cor()函数计算得到了 5 各向量的相关性矩阵，为更好的观察两两向量间的相关性情况，根据该矩阵绘制热图如图 7.

由热图可以看出某地区任务数量与该地区的会员数量有很强的正相关

（cor=0.8637）,而定价主要与该地区的信誉值、任务数量以及会员数量呈现一定程度的负相关。从而我们可以推断出若某地区会员信誉值高，则任务流水量就越大，且若任务数量以及会员数量越多，竞争机会也就相应增加，导致定价偏低。

此外我们还能看出任务完成率与任务定价具有相关性，为确定其统计学意义，我们对附件一中完成与未完成的任务标价做 Wilcoxon 秩和检验，判断两类任务标价是否有差异。得$p = 1.772 \times 10^{-10}$,即完成与未完成任务的标价具有显著性差异。从而推断出可能是由于任务的标价过低，而此地区的任务量、会员数量以及会员平均的信誉值也不高，会员积极性不高导致的任务失败。

表 2、每个地区的整合信息

region	task_num	mem_num	reputation_m	task_price	exe_condition
26	179	351	372.6549	70.3352	0.988827
20	76	157	558.3835	66.80921	0.565789
23	70	87	60.72029	73.45714	0.885714
6	65	304	100.7498	66.94615	0.123077
18	60	99	185.3325	67.66667	0.783333
21	48	60	30.52227	71.41667	0.5625
16	40	41	47.91678	71.6	0.575
4	39	153	96.60526	68.11538	0.128205
11	39	114	344.7394	65.92308	0.25641
25	37	46	109.2304	67.12162	0.189189
2	35	68	789.3613	65.84286	0.571429
17	27	83	80.41539	65.81481	0.592593
10	18	52	324.1704	66.13889	0.555556
19	18	13	39.47079	72.22222	1
12	16	3	159.5765	72.21875	0.9375
22	13	14	32.20929	72.30769	1
15	12	22	696.2113	67.08333	0.416667

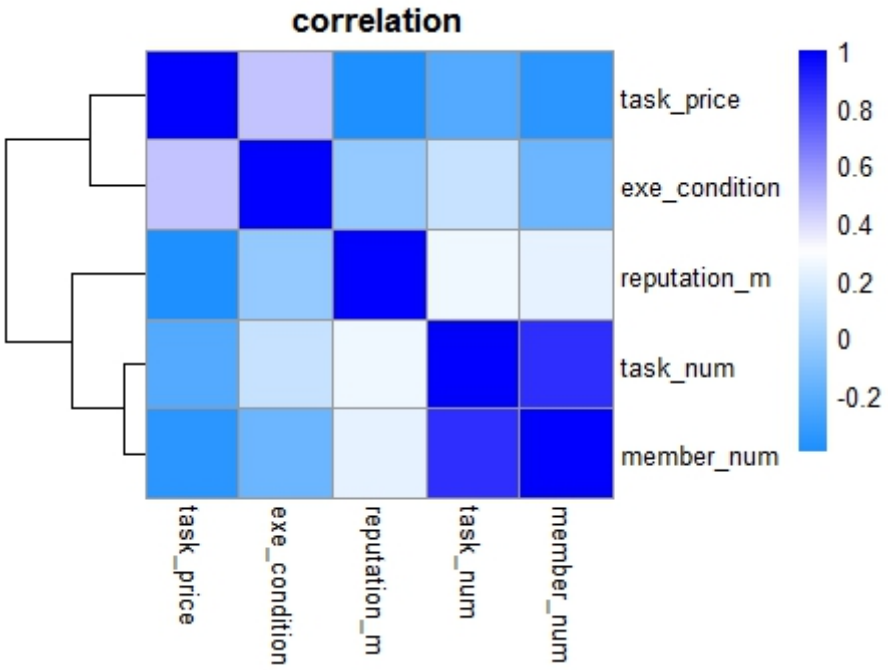

图 7、相关性热图

5.1.3、基于近邻的分析

我们还根据任务与会员所处位置的距离进行分析，通过找某一任务点A_i周围距离d内的会员人数k_i来衡量该任务的分布情况。我们对k与任务标价进行pearson 相关性分析，得$cor = -0.4493, p < 2.2 \times 10^{-6}$，两者具有一定程度上的负相关，即可以看出任务可被接受的人越多，价格也相对较低些。

根据此我们发现任务的完成情况与任务的可接受会员的密度决定，密度越大，完成的较好。我们便可以利用此来从任务完成情况的角度评判任务的定价是否合理。该思想详见 5.2.2。

5.2、新定价方案的设定

5.2.1、岭回归定价模型

由 5.1 的分析可知，任务定价与多种因素有关，包括与某地区的信誉值、任务数量以及会员数量呈现一定程度的负相关，与任务执行的成功率呈现一定的负相关。为设定新的定价方案，我们拟对标价建立一个回归模型来作为定价的新方案。

我们以某地区的信誉值、任务数量以及会员数量这三个因素利用岭回归[6]来制定定价方案，岭回归最初提出是为了解决回归中的多重共线性问题。定义如下回归模型：

$$y = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \varepsilon \quad \text{（公式 2-1）}$$

式中β为未知参数，ε是不可测的随机误差，服从一定的分布，假定其服从正态分布$N(0,\sigma^2)$，y表示某一地区的平均定价，x_1, x_2, x_3分别表示该地区任务数量、会员数量和会员信誉值。根据最小二乘法可得上述β值：

$$\hat{\beta} = arg\,\min_{\beta} \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{3} x_{ij}\beta_j)^2 \quad \text{（公式 2-2）}$$

该式表示使得误差平方达到最小值时的β即为该参数的估计值$\hat{\beta}$。但由于自变量间存在着一定得线性相关，便在最小二乘的估计上对估计值的大小用正则化方法增加一个约束，即得到岭回归的参数估计值：

$$\hat{\beta}_{ridge} = arg\,\min_{\beta} \left\{ \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{3} x_{ij}\beta_j\right)^2 + k \sum_{j=1}^{3} \beta_j^2 \right\} \quad \text{（公式 2-3）}$$

式中$k \sum_{j=1}^{3} \beta_j^2$为$L_2$正则化惩罚，$k$为岭参数。

采用 R 中的 glmnet 包来求解该参数的估计。使用 cv.glmnet()函数对训练样本采取 5 倍交叉验证计算岭参数变化时平方误差的变化，绘制如下图：

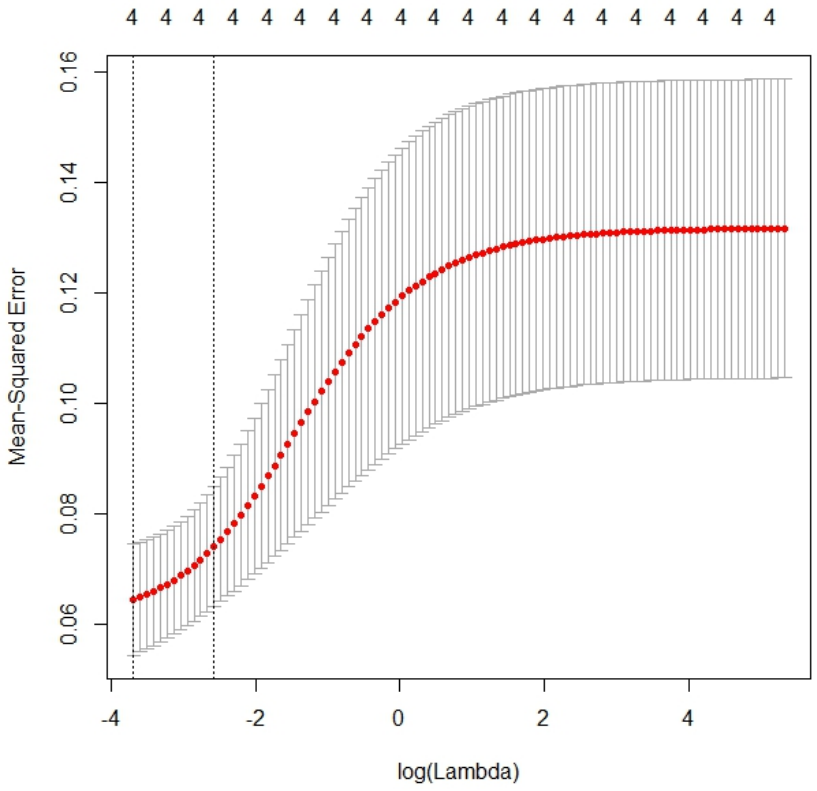

图 8、岭参数随误差变化图

图中横坐标为岭参数的对数值，纵坐标为平方误差，左侧竖直虚线所处的位置为最优岭参数，上方对应数字为变量的数量。我们利用该岭参数获得模型：

$$y = 69.683 + 0.032x_1 - 0.013x_2 - 0.005x_3 \quad \text{（公式 2-4）}$$

式中y表示某一地区任务的平均定价，x_1, x_2, x_3分别表示该地区任务数量、会员数量和会员信誉值。

由该模型可以看出某一地区的任务平均定价y受多种因素的混合影响，且x_1系数为正、x_2, x_3前系数为负，与 5.1 相关性分析的结果一致。我们可以用改该模型从新计算出任务的标价，为比较新标价与原方案的关系，我们计算了两者的相关性系数得$r = 0.7482926$，可以看出与原方案具有相关性。我们可以用该模型来计算出任务标价。

5.2.2、执行情况评价模型

为衡量新标价的执行情况，我们采用基于近邻的思想来设计如下任务执行情况评价模型：

设某一任务点A_i周围距离d内有B_i个会员，而每个会员周围距离d内有μ_j个可选任务，则定义各可选任务对会员的吸引程度为：

$$\alpha_{ij} = log_{10} \frac{price_j}{distance_j} \quad \text{（公式 2-5）}$$

式中$price_j$表示第j个任务的标价，$distance_j$表示该会员与任务点的距离。由两者比例关系我们可以分析出某会员将会更倾向于α值更大的任务。由此，我们给出某任务A_i完成概率的计算方式：

步骤一、找出任务点A_i周围距离d内的B_i个会员；

步骤二、对每个会员找出其附近的距离d内的可选任务；

步骤三、对每个会员计算出其各个可选任务的α值；

步骤四、由会员的预订任务限额与各个可选任务的α值计算选择任务A_i的概率：

若会员的预订任务限额q大于可选任务的数量μ_j：

$$p = \frac{1}{\mu_j + \delta} \quad \text{（公式 2-6）}$$

若A_i的α值的秩次小于预订任务限额q：

$$p = \frac{1}{q + \delta} \quad \text{（公式 2-7）}$$

若A_i的α值的秩次大于预订任务限额q，则该任务将很低概率被选到：

$$p = \frac{1}{(\mu_j - q)\mu_j} \quad \text{（公式 2-8）}$$

由于某些任务被完成的概率并没有预期的高，所以我们在估计时采取一种悲观估计的方式，在上式中加入悲观惩罚因子δ，我们在运行时取 0.5。在算法实行过程中我们发现会员的信誉值对任务完成的情况有很大的影响，于是我们计算预测得分：

$$S = \frac{\omega}{\omega_{0.8}} * P \quad \text{（公式 2-9）}$$

式中ω为会员的信誉值，$\omega_{0.8}$为全体会员信誉值的80%分位数，若$\frac{\omega}{\omega_{0.95}} > 1$，则取1。

因为有些会员的信誉值极低，得出的预测得分过小，对预测的影响极大，所以由该式得出的预测得分为较悲观得分。理论上，若会员的信誉度足够大，S则等于任务被选到的概率。

步骤五、将各会员选择任务A_i概率的最大值做为任务A_i的完成概率。

该算法的概念图如下：

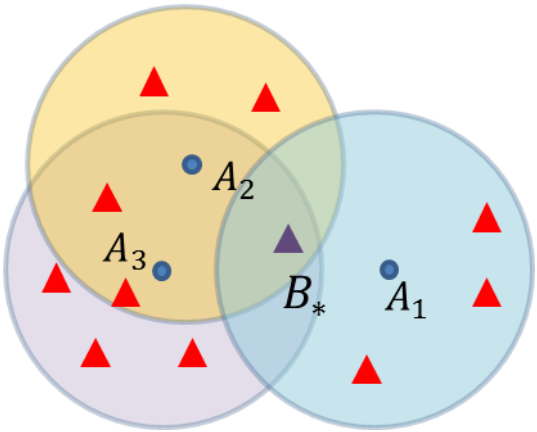

图 9、近邻模型概念图

在图 9 中蓝色点表示任务，三角形表示会员，圆圈表示d范围内。以任务A_1为例，首先搜索其d范围内的会员，如B_*，找到其能接受的任务(A_1, A_2, A_3)，并根据任务的定价与距离计算接受A_1的概率。如此计算任务A_1周围所有会员的接受情况，取最大值作为A_1的接受概率。

由以上的描述我们可知该方法就有很强的实际意义，即可以在理论上预测实

际的任务完成情况，也可以对已给的定价进行实际意义的评估。

采取合适的d值根据上述算法便可评价任务的定价情况。我们利用该预测模型对附件一的任务定价进行评价（$d = 0.01$），得分的分布如图 10 左侧的小提琴图。该图外侧轮廓为各得分的分布情况，内部黑色部分为箱线图。

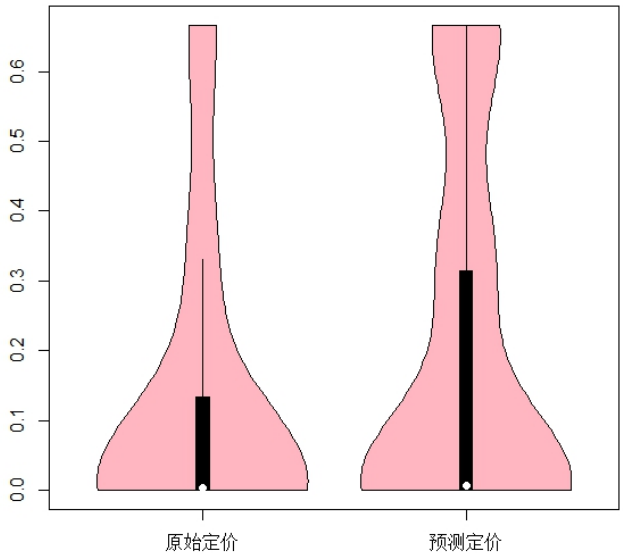

图 10、预测价格与原定价比较

可以看出仅25%以下的任务具有较高的完成概率。而这结果与附件一给出的已完成任务情况有一定的偏差，在某些低得分的任务被完成而某些高得分的任务也没有被完成。主要原因可能有：虽某些会员的信誉值极低，由模型给出的完成概率是个很小的值，但其仍可能去完成该任务；我们的结果还受会员可接受的任务距离影响，不同会员可接受的任务距离可能不同，有些任务点在该模型给定的距离下虽没有可接受的会员，但可能某些会员可接受任务的距离较大，从而完成这些任务等。为减弱低信誉会员的影响，我们可以删除低的得分，使用截断得分来衡量定价情况。

下面我们将依照此算法衍生出一个新的定价模型。

5.2.3、基于近邻的定价模型

由该模型我们也可以看出若想使任务的实施更有效，我们需要采取制定合适的价格使得α较高来鼓励周围的会员完成该任务。我们基于会员与周围各任务点的α均相等的假设来获得全部地区的新定价，让一些被完成概率小的。建立基于近邻的定价模型。

在算法实现中，我们先给定某一地区的任务价格，接着再d范围内搜寻会员，并对每个会员搜寻其可选任务，然后根据α相等原则来推算各个可选任务的标价，最后对已推算出的任务点递归调用该算法，即可得到所有任务的标价信息。若有些任务点与已推算出价格的任务点经会员不连通，则需在不连通区域继续调用该算法。在定价过程中，总会有任务被重新更新订价值，我们则取原定价与新定价的平均值进行更新。

我们利用该定价模型为附件一中的任务重新定价。

我们对该模型得到的任务新定价用 5.2.2 的评价模型进行评估，结果如图 10 右侧的小提琴图。可以看出新预测价格的得分分布在较高区域，且分布较均匀，截断得分的均值显著高于原始定价。所以可以看出定价模型达到了比较好的效果。

5.3、打包定价方案修改

5.3.1、基于密度对任务聚类

由图可以看出有些任务趋于聚集出现，导致周围会员的争抢。所以我们想修改上述基于距离近邻的定价方案，打算将聚集的任务打包分配给会员。我们采用一种基于密度的聚类算法——DBSCAN[7]，该算法将具有有足够高密度的区域划分为簇，可以实现对紧邻任务的打包，具体算法描述如下：

我们首先定义 Eps 和 MinPts，前者为定义密度时的邻域半径，后者为定义核心任务点时的阈值，记为 ε 和 M。

给出核心任务点、边界任务点、噪音点的定义及数学描述：

考虑任务集合$A = \{A^{(1)}, A^{(2)}, \dots A^{(N)}\}$,设任务$x \in A$,称：

$$N_\varepsilon(x) = \{y \in A: distance(y, x) \leq \varepsilon\} \qquad \text{（公式 3-1）}$$

为任务x的ε邻域。称：

$$\rho(x) = |N_\varepsilon(x)| \qquad \text{（公式 3-2）}$$

为x的任务密度,其依赖于半径ε。

若$\rho(x) \geq M$，则称任务x为A任务集合的核心任务点,记由A中所有核心任务点构成的集合为A_c，并记$A_{nc} = A \backslash A_c$表示由$A$中的所有非核心任务点构成的集合；若$x \in A_{nc}$，且$\exists y \in A$，满足$y \in N_\epsilon(x) \cap A_c$,即$x$的$\varepsilon$邻域中存在核心任务点，则称$x$为$A$的边界任务点,记由$A$中所有边界任务点构成的集合为$A_{bd}$,记$A_{noise} =$

$A\backslash(A_c \cup A_{bd})$;若$x \in A_{noise}$，则称$x$为噪音点。

从上述数学描述中可知核心任务点对应任务分布稠密区域内部的点，任务边界点对应任务分布稠密区域边缘的点，而噪音点对应分布稀疏区域中的点。

密度相连的定义及数学描述：

若$x \in A_c, y \in N_\varepsilon(x)$，则称$y$是从$x$直接密度可达的；设$p^1, p^2, \ldots p^{(m)} \in A$，其中$m \geq 2$，若他们满足$p^{(i+1)}$是从$p^{(i)}$直接密度可达的，则称$p^{(m)}$是从$p^{(1)}$密度可达的；设$x, y, z \in A$,若$y$和$z$均是从$x$密度可达的，则称$y$和$z$是密度相连的。

算法流程如下：

扫描整个任务数据集，找到任意一个核心任务点，对该核心任务点进行扩充。扩充的方法是寻找从该核心任务点出发的所有密度相连的数据点。遍历该核心任务点的邻域内的所有核心任务点，寻找与这些任务点密度相连的点，直到没有可以扩充的任务点为止。最后聚类成的簇的边界节点都是非核心任务点。之后就是重新扫描数据集（不包括之前寻找到的簇中的任何任务点），寻找没有被聚类的核心任务点，再重复上面的步骤，对该核心任务点进行扩充直到数据集中没有新的核心任务点为止。数据集中没有包含在任何簇中的任务点就构成异常点。

该算法的概念图如下，图中红色的点为核心任务点，蓝色的点为边界任务点，黄色点为噪音点，由该图可以更好的理解该聚类算法：

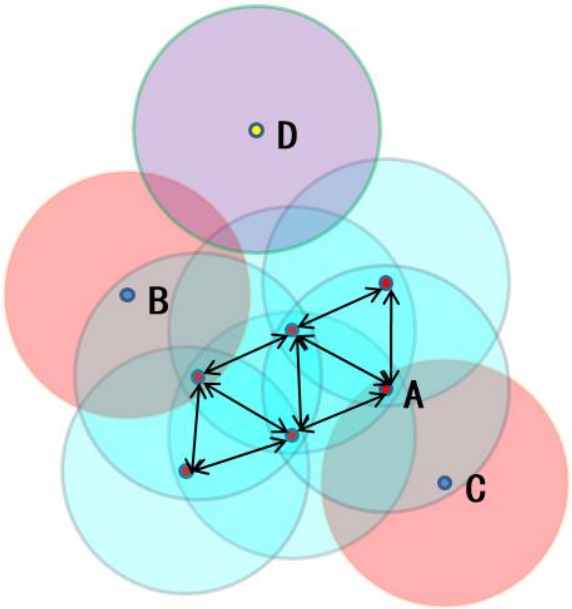

图 11、DBSCAN 聚类算法概念图

我们用该算法对附件一中的任务进行聚类，采用 R 中的 dbscan()函数实现该算法。为了使每个任务包包含的任务不是太多，我们将邻域半径与核心任务点阈值两个参数设定为$Eps = 0.01, MinPts = 3$，聚类后画出聚类图：

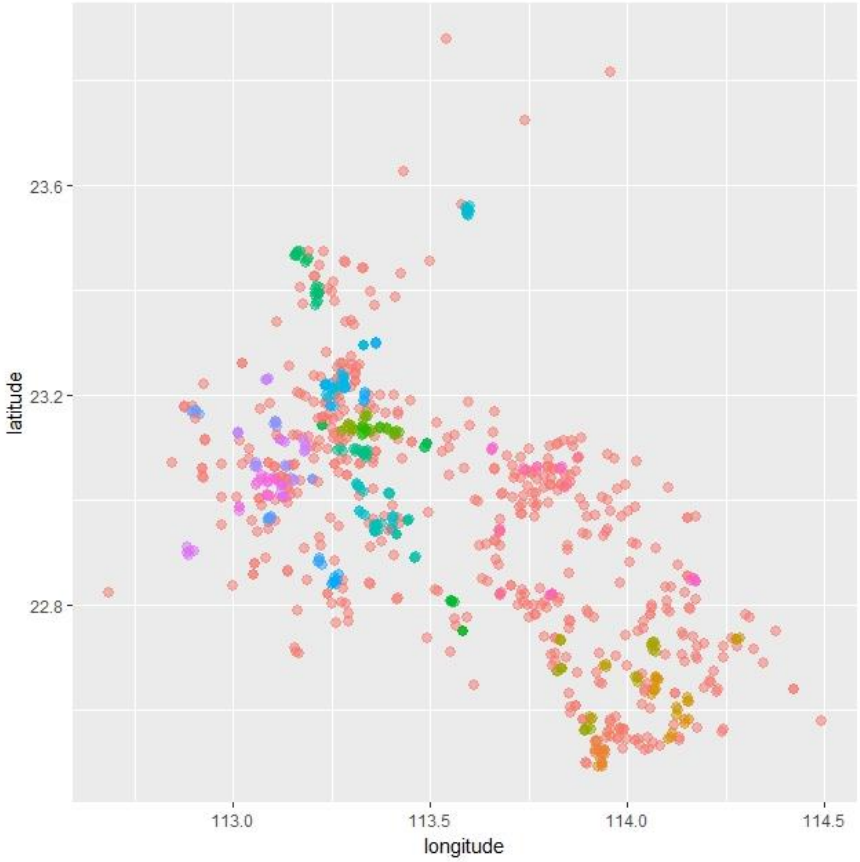

图 12、对任务的聚类结果

聚类后得到 87 个小类，每类包含3~10个任务点。之后我们将每类任务中所有任务包含的会员取并集，得到每类任务中会员的数量情况。我们发现在这些会员中有些人的任务限额小于该类中的任务数量，即他们不能接这个打包任务，我们便将这些会员删除。根据以上方法得到一些打包的任务及可接受的人数，几个任务包的例子如表，可以看出聚类得到了较好的结果。

5.3.2、构建近邻打包模型

我们便依照如上方法建立对距离较近的任务的打包模型。

将该模型应用到 5.2 中的近邻定价模型，可以得到修改后的近邻打包定价模型。在确定打包任务可被多少会员接受时，将该任务包中所有任务的可接受会员

取并集，再结合各个会员的任务限额看是否有权利接受这任务包。如没有人能够接受该任务包，则取消打包。其中各个任务的定价$p_.$由有权利接受的会员依照近邻定价决定的，打包任务的定价以下式确定：

$$p_* = \tau \frac{\sum_* p_.}{n} \qquad \text{（公式 3-3）}$$

式中p_*表示n个任务组成的任务组的定价，$p_.$表示各个任务的定价，τ为打包定价系数，可人为给定。一般可认为打包任务会增加些难度，所以可取$\tau = 1.05$，也可根据不同情况筛选不同的值。

即在近邻打包定价模型中，某些任务会以一定的任务包的形式存在，任务将会被更有效率完成。编写该程序并在附件一的任务集中运行得到结果，结果如表：

表 3、任务打包情况

任务	会员	标价均值	完成情况	会员平均信誉值
A0008 等 3 个	B0002 等 3 个	66.5	010	18464.73
A0698 等 3 个	B0014 等 4 个	72.67	110	7006.156
A0189 等 3 个	B0018 等 6 个	68.67	111	16372.3
A0682 等 5 个	B0014 等 4 个	72.6	11111	7006.156
A0520 等 6 个	B0149	66.92	111111	199.2313
A0200 等 7 个	B0018 等 6 个	70.21	1111111	12830.72
A0422 等 7 个	B0018 等 8 个	69.29	1110010	11533.65
A0169 等 9 个	B0018 等 8 个	67.28	000001100	9383.534
A0009 等 10 个	B0002 等 3 个	66.05	0111101101	18464.73

由表中数据在采用该模型进行标价时，任务点更加集中，由于低信誉度的会员的任务限额较低，会对其产生一定的限制，会使得任务的完成情况会更好。

5.4、新任务定价

我们利用分别用如上介绍的近邻定价模型和近邻打包定价模型对附件三进行价格预测，并利用任务执行情况预测模型来预测其执行情况。具体预测的定价见附录。

我们首先绘制出附件三的任务位置的分布情况如图 13，由此我们可以看出这些任务有些任务较散，而有些任务分布则较为紧密。所以我们在进行近邻打包

聚类分析时，取$Eps = 0.002, MinPts = 2$。得到了 74 个任务包，具体的信息见附表 2，其中标识相同数字的为一个任务包，表示 0 的为非打包任务。

由结果整理出来的可接受会员的信息中我们发现这些会员具有很高的信誉值，所以应具有很好的完成情况。

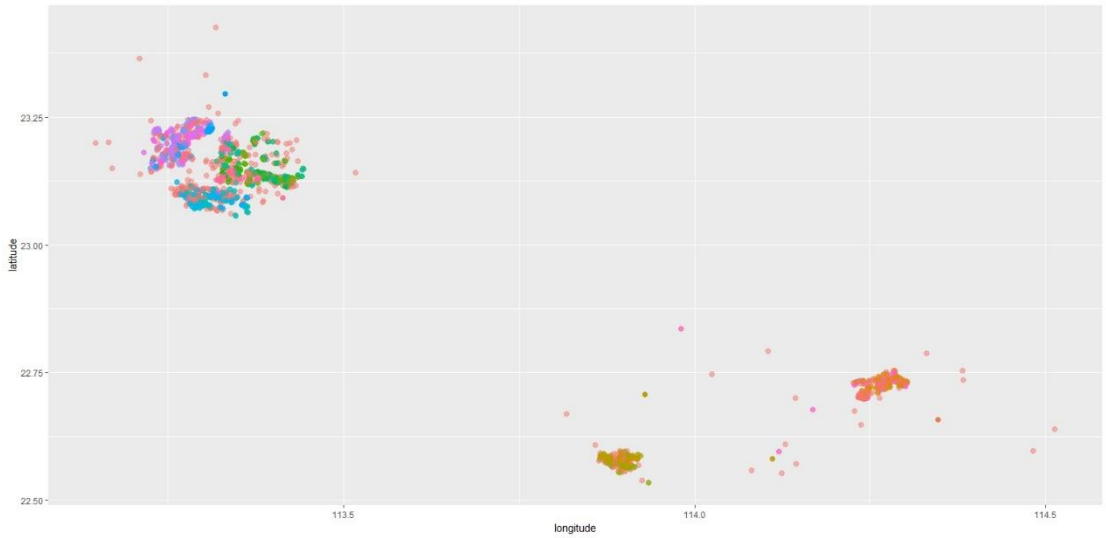

图 13、新任务的位置分布图

六．模型评价

6.1、各模型的优缺点

岭回归定价模型是一种基于机器学习方法的回归预测模型。该模型应用简单，可以从系数上直观看到价格与影响因素的关系，并给出价格预测，适合于单一样本的预测；但在训练时依赖于原样本数据，模型的可解释性差且在应用时局限于其所处的区域。

近邻定价模型是我们依照近邻思想，结合实际情况构造的模型。该模型不同于回归学习模型，其与实际紧密相关，模型的解释性好，结果可应用性强，适合于地区的批量定价；而该模型在定价时依赖于如会员可接受的任务距离d、会员选择任务的概率等因素，易受会员的主观因素干扰。

近邻打包定价模型是结合任务的聚类信息对近邻定价模型的改进。在改进之后，使得任务的完成效果将会更好，任务进行效率高，是一个很理想的定价模型；但该模型仍会受会员的主观因素干扰。

定价评估模型也是一种基于近邻思想并结合实际构造的，结合会员信息考察任务完成情况的模型。该模型可解释性强，可信度高，可以对多任务整体定价设计的好坏进行评估；但其受低信誉会员的影响较大，以及依赖于会员可接受的任务距离d。

总的来说，在使用结合实际情况构造的模型时可应用性强，但总会受到实际情况下干扰因素的影响；而使用回归学习模型时岁考虑了多因素的情况，但模型的解释性差，不适合批量使用。

6.2、模型改进

在使用岭回归定价模型时，若增加训练样本量，多结合些已有知识，则会达到一个比较好的效果，但是仍依赖于原定价，最好应用于单一任务或任务包的定价。

在进行对批量任务进行定价时，采用近邻打包模型达到了非常好的效果，对该模型进行继续优化具有很强的实际应用意义，我们可以从如下方面进行继续优化：

1、在受会员主观因素影响方面，我们可以收集会员选择哪些任务的信息，将该信息训练模型并结合到我们的定价模型中，将会减轻模型受随机效应的影响；

2、在选择会员可接受的任务距离d时，我们也根据会员选择哪些任务的信息来估算出合理的d值；

3、此外也可以采用d自适应上升的办法来确定定价，即若某任务d范围内没有一个信誉值较高的会员，则增大d扩大搜索范围，来寻找高信誉值的会员来继续定价。该改进方案可以很大程度上减弱低信誉值会员的影响；

4、在处理重复定价的过程中，我们采用的是均值作为新定价，我们也可以根据α来取加权和作为新定价，会使得定价的解释性更高。

总之，若要更好的定价效果，我们收集需要更多的信息来丰富改进我们的模型，从多角度优化模型，以让其发挥更大的作用。

6.3、模型应用展望

应用以上模型得到的结果取得了很理想的效果，任务的完成效果好，截断评价得分趋于相同，任务的价格分布不出现偏置。所以我们可以将以上模型嵌入到 APP 当中，为新任务实现定价。而且在 APP 在运行过程中，可以不断地收集会

员的接受任务情况以及完成情况，根据这些信息可以自适应的调整模型，使系统达到完美的应用效果。

七．参考文献

[1]Robert I. Kabacoff，《R 语言实战第 2 版》，黄小宁，刘撷蕊，黄俊文等译.北京：邮电出版社，2016 年。

[2]出门请左拐.数据挖掘：R 语言地图可视化 Remap 包.

https://zhuanlan.zhihu.com/p/23247602，2017 年 9 月 16 日。

[3]Hadley Wickham，《ggplot2 Elegant Graphics for Data Analysis》，Springer-Verlag New York Inc，2009 年。

[4]卢淑化.《社会统计学》.北京：北京大学出版社，2009 年。

[5]盛骤，谢式潘，承毅.《概率论与数理统计》.北京：高等教育出版社，2008 年。

[6]万丽颖.岭回归分析及其应用[J]许昌学院学报，2016,35（02）：19-23。

[7]邢冬丽，赵美红，陈文成.基于密度的 DBSCAN 算法[D]辽宁，辽宁工商技术学院.2015：1-5。

八．附录

1．附表：新任务的预测情况（仅显示前 200 行，详表见支撑材料）

任务号码	任务 GPS 纬度	任务 GPS 经度	近邻预测定价	任务打包情况	打包修正价格
C1844	23.15633901	113.255376	82.31563475	352	78.96963
C2013	23.15511852	113.2553496	80.16180562	352	78.96963
C2018	23.15381021	113.2550439	74.43143833	352	78.96963
C1557	23.18429338	113.327348	86.24876683	344	87.45148
C2011	23.18450212	113.3266313	88.65420018	344	87.45148
C1420	23.14646718	113.3454528	46.01409076	334	48.22726
C1486	23.14709751	113.3441724	50.44042658	334	48.22726
C1175	22.73391699	114.2351752	45.76512435	326	45.25267
C1180	22.73434232	114.2359449	44.74022059	326	45.25267
C1027	22.70179996	114.2462511	46.50573488	317	68.66658
C1077	22.70011627	114.2462028	74.38210118	317	68.66658
C1078	22.70011627	114.2462028	85.11189592	317	68.66658
C0952	23.15999994	113.2399971	75.16562142	306	57.6197
C0958	23.16092701	113.239809	50.23525216	306	57.6197
C2026	23.16045484	113.2412345	47.01085687	306	57.6197
C2028	23.16003203	113.2401767	79.61497338	306	57.6197
C2029	23.16107389	113.2397168	47.42247261	306	57.6197
C2031	23.16196003	113.2401761	46.26900653	306	57.6197
C0939	23.20617703	113.278833	71.35362801	302	76.73127
C2008	23.20617703	113.278833	82.10891213	302	76.73127
C0935	23.14935422	113.233401	46.31580649	301	57.61427
C0937	23.14803299	113.2340485	68.91272427	301	57.61427
C0930	23.17809539	113.2469203	73.49352775	300	77.96098
C1808	23.17880966	113.2460895	82.42842376	300	77.96098

C0888	23.23628282	113.30108	47.52665604	292	67.64146
C0993	23.23756506	113.3024524	81.96777102	292	67.64146
C1955	23.23812868	113.3024835	73.42996122	292	67.64146
C0865	23.22234285	113.2980538	74.11008056	285	62.10754
C0866	23.22220281	113.2988216	71.66364929	285	62.10754
C0869	23.22320441	113.2952196	45.36583354	285	62.10754
C0873	23.22286001	113.299456	46.62302013	285	62.10754
C1942	23.22282685	113.2959984	44.20854541	285	62.10754
C1943	23.22233557	113.2976418	45.50156092	285	62.10754
C1945	23.22324771	113.2958978	84.58634138	285	62.10754
C1946	23.22324771	113.2958978	84.80131918	285	62.10754
C0823	23.21232983	113.3325744	46.27320015	276	57.9089
C0826	23.21039927	113.331032	81.75122273	276	57.9089
C1841	23.20946083	113.3309223	44.92435185	276	57.9089
C1891	23.21324038	113.3333756	45.24954165	276	57.9089
C1892	23.21174194	113.3315996	83.94357433	276	57.9089
C1896	23.21157833	113.3318833	45.31153359	276	57.9089
C0781	23.15994989	113.2278135	45.51159386	267	45.50655
C1869	23.15945918	113.2266535	44.9932814	267	45.50655
C1870	23.15877019	113.2276922	46.01476508	267	45.50655
C0769	23.220519	113.2340276	81.42303768	262	70.55628
C0795	23.22038167	113.2321281	74.97219121	262	70.55628
C0819	23.21972176	113.2312636	83.47777176	262	70.55628
C0918	23.22242701	113.237711	45.94269797	262	70.55628
C0919	23.22244292	113.2342204	45.79257397	262	70.55628
C0920	23.22114324	113.2319099	78.2096058	262	70.55628
C0923	23.22082641	113.2298735	44.99280219	262	70.55628
C1910	23.22155066	113.2359355	79.56285719	262	70.55628
C1989	23.22114401	113.231859	86.14030199	262	70.55628

C1990	23.22259501	113.237466	85.0489535	262	70.55628
C0751	23.24077183	113.2899343	75.45091818	254	64.57409
C1834	23.24172154	113.2884332	74.58381445	254	64.57409
C1835	23.2398074	113.2909599	44.68781963	254	64.57409
C1836	23.2398074	113.2909599	83.47841718	254	64.57409
C1837	23.2420317	113.289767	44.66949865	254	64.57409
C0749	23.22975601	113.293487	88.22660255	252	65.12657
C1817	23.2287019	113.2925815	46.06710495	252	65.12657
C1820	23.22843595	113.293573	80.84979844	252	65.12657
C1974	23.22818832	113.2926425	45.3627611	252	65.12657
C0742	23.22596441	113.272686	81.9450351	249	64.10291
C0925	23.22332543	113.2744367	46.31580649	249	64.10291
C0927	23.22450282	113.2720301	75.58601507	249	64.10291
C0928	23.22129673	113.2727296	45.18412331	249	64.10291
C0929	23.22288001	113.271505	77.73409067	249	64.10291
C0931	23.22202087	113.2716705	45.3580408	249	64.10291
C0996	23.22323709	113.2751832	46.55094239	249	64.10291
C1821	23.2258322	113.2730093	45.82490311	249	64.10291
C1826	23.22628842	113.2726572	84.53496131	249	64.10291
C1988	23.22392548	113.2709844	80.57048418	249	64.10291
C1994	23.22284057	113.2749262	74.81192434	249	64.10291
C1996	23.22294691	113.2714389	81.44543314	249	64.10291
C1997	23.22171187	113.272405	71.49704486	249	64.10291
C1998	23.22140682	113.2744787	68.55801117	249	64.10291
C2000	23.22589883	113.2729862	72.14756351	249	64.10291
C2005	23.22332543	113.2744367	46.10514778	249	64.10291
C2065	23.22158101	113.273016	45.57992732	249	64.10291
C0688	23.15365534	113.2342723	45.14027307	223	58.23888
C0850	23.15268275	113.23529	79.57345106	223	58.23888

C0853	23.15292102	113.2342537	74.30215262	223	58.23888
C1761	23.15181322	113.2323533	46.868493	223	58.23888
C1929	23.15176041	113.233343	45.31003719	223	58.23888
C0670	23.10041876	113.333484	45.28893038	219	62.41319
C1613	23.1018158	113.3326661	79.53745319	219	62.41319
C0617	23.07618942	113.2946005	45.93719189	210	45.89066
C1697	23.07499516	113.295539	45.84413634	210	45.89066
C0611	23.07122717	113.2953972	74.70347597	209	78.8947
C1686	23.0716011	113.2948965	83.085919	209	78.8947
C0561	23.08216973	113.3399428	83.02110067	204	56.36329
C0562	23.08267459	113.3418352	51.18060863	204	56.36329
C0592	23.08308279	113.3399863	46.08981505	204	56.36329
C0596	23.08308279	113.3399863	45.16162429	204	56.36329
C0528	23.06997542	113.2871034	74.21686567	196	70.31343
C0612	23.07708809	113.2899715	88.65420018	196	70.31343
C0641	23.0718287	113.2890075	75.3307222	196	70.31343
C0655	23.07540145	113.2892083	74.1295653	196	70.31343
C0657	23.07401136	113.2887508	79.69442122	196	70.31343
C1645	23.07239962	113.2879002	46.55819943	196	70.31343
C1685	23.07714796	113.2900116	77.20591958	196	70.31343
C1712	23.07128795	113.288387	46.41252627	196	70.31343
C1713	23.07130722	113.2895576	70.61849061	196	70.31343
C0512	23.10909927	113.2863779	83.46391027	189	85.50202
C0514	23.10904524	113.2867449	87.54013963	189	85.50202
C0511	23.10608204	113.3150025	80.8817919	188	68.13308
C0532	23.10572662	113.3155669	45.78817215	188	68.13308
C0541	23.1045085	113.3148747	71.24819526	188	68.13308
C1631	23.10393906	113.3143144	74.61416933	188	68.13308
C0501	23.10483155	113.3448499	49.86031746	182	66.67207

C1623	23.10339376	113.3439109	83.48381856	182	66.67207
C0496	23.09458181	113.2923934	45.41972002	180	50.8987
C0524	23.09443038	113.2912593	45.23369132	180	50.8987
C0529	23.09395878	113.2918453	46.35181351	180	50.8987
C0530	23.09395878	113.2918453	44.59656303	180	50.8987
C0654	23.09312304	113.2935605	46.61597952	180	50.8987
C1632	23.09447782	113.2911732	77.17444906	180	50.8987
C0488	23.08619006	113.325538	79.40769809	177	65.09359
C0521	23.08626228	113.3252833	46.46681564	177	65.09359
C0535	23.08663504	113.3251015	46.52595174	177	65.09359
C0536	23.08619006	113.325538	81.81940941	177	65.09359
C0540	23.0885492	113.3250723	71.24809801	177	65.09359
C0483	23.114957	113.2814447	44.46598335	173	59.47615
C0491	23.11506401	113.281722	87.71674469	173	59.47615
C1691	23.11451284	113.2830336	46.24573246	173	59.47615
C0482	23.09919654	113.3296033	77.40040501	172	77.53712
C1609	23.09982701	113.3295697	81.35024782	172	77.53712
C1755	23.09922949	113.3293998	73.86071484	172	77.53712
C0476	23.0742034	113.3630844	44.59800125	169	59.99279
C0478	23.0742034	113.3630844	88.78856025	169	59.99279
C0517	23.07414653	113.3633138	46.59181786	169	59.99279
C0475	23.1076787	113.3386013	75.03895802	168	81.74287
C1646	23.1084114	113.3373186	88.44678687	168	81.74287
C0464	23.10335599	113.2791487	46.33373934	164	64.68081
C1701	23.10260563	113.2800968	83.02788841	164	64.68081
C0437	23.17843093	113.3484722	72.34023194	160	63.49514
C1369	23.18105596	113.3487012	77.90305345	160	63.49514
C1382	23.17719024	113.3495078	79.59921158	160	63.49514
C1389	23.17719024	113.3495078	46.02064997	160	63.49514

C1396	23.18035058	113.3501187	76.83063264	160	63.49514
C1400	23.17869957	113.3504804	46.70794758	160	63.49514
C1457	23.18220065	113.3507131	45.06423162	160	63.49514
C0434	23.11665299	113.370997	78.13715793	158	63.77466
C0435	23.11770618	113.3701314	49.42906562	158	63.77466
C0439	23.11751722	113.3707793	44.73585294	158	63.77466
C1371	23.11561808	113.3698543	82.79657784	158	63.77466
C0418	23.15532417	113.3310176	83.71911924	155	58.56319
C0421	23.15512634	113.3308469	46.1472737	155	58.56319
C1494	23.15617501	113.329269	72.91522784	155	58.56319
C1563	23.15532417	113.3310176	44.0251649	155	58.56319
C1565	23.15532417	113.3310176	46.00918926	155	58.56319
C0410	23.12579085	113.4293179	46.85067383	153	63.05958
C0422	23.12565121	113.4288337	79.26848899	153	63.05958
C0406	23.1968891	113.3339044	68.82851284	151	65.29268
C0469	23.19628164	113.3347933	79.210045	151	65.29268
C0739	23.19797648	113.3331576	88.66716645	151	65.29268
C1559	23.1968891	113.3339044	45.41248954	151	65.29268
C1806	23.19701417	113.3327588	74.30215262	151	65.29268
C1807	23.19856499	113.3331124	47.41105416	151	65.29268
C2025	23.19977345	113.3341445	46.0983244	151	65.29268
C2060	23.19628164	113.3347933	72.41168114	151	65.29268
C0360	23.15263157	113.3319798	70.59595973	139	63.94737
C1489	23.15255891	113.3322385	75.17154346	139	63.94737
C1491	23.15263157	113.3319798	46.07461644	139	63.94737
C0356	23.20200824	113.3923275	79.21229671	138	78.82547
C0358	23.20157236	113.3932173	81.50130183	138	78.82547
C1502	23.20200824	113.3923275	75.76282129	138	78.82547
C0355	23.15243901	113.335162	75.16088985	137	74.7715

C1497	23.15180501	113.335155	74.38210118	137	74.7715
C0346	23.20819295	113.3782438	83.40082562	134	64.99157
C1381	23.2065107	113.3774832	46.58230935	134	64.99157
C0338	23.15972056	113.4046289	50.32978332	130	46.35526
C0438	23.15891441	113.403394	44.4614196	130	46.35526
C1474	23.1596788	113.4061139	44.27456883	130	46.35526
C0299	23.13746237	113.3939663	44.4614196	120	54.23232
C0304	23.13612309	113.3963694	45.88664992	120	54.23232
C0764	23.13699912	113.3948015	81.39182351	120	54.23232
C1438	23.13746237	113.3939663	45.18939511	120	54.23232
C0294	23.13708468	113.3341858	73.82757371	116	70.82653
C0351	23.13509985	113.3314282	48.20638305	116	70.82653
C1407	23.13654075	113.3327505	75.44129896	116	70.82653
C1410	23.13680577	113.3315495	85.83085311	116	70.82653
C0284	23.14811461	113.3527602	45.57250436	109	66.73183
C0286	23.14744653	113.3511514	45.74477634	109	66.73183
C1397	23.14542487	113.3508006	83.05337356	109	66.73183
C1401	23.14846156	113.3521911	85.61726136	109	66.73183
C1417	23.14607776	113.3514674	44.57401661	109	66.73183
C1518	23.14567525	113.353414	87.89013334	109	66.73183
C1519	23.14567525	113.353414	74.67077177	109	66.73183
C0256	23.16659194	113.3382389	44.99280219	103	65.27325
C0269	23.16825306	113.3381455	46.6667428	103	65.27325
C1383	23.16611314	113.3381535	82.42913504	103	65.27325
C1385	23.16611314	113.3381535	71.7113719	103	65.27325
C1388	23.16764912	113.3373636	80.56621868	103	65.27325
C0255	23.21850801	113.38547	47.35258048	102	63.15855
C0263	23.21719106	113.3842337	78.96451472	102	63.15855
C0252	23.1630241	113.3350317	78.55263613	101	80.17285

C1362	23.16294539	113.3351079	81.79305418	101	80.17285

2. 本文全部程序段：包含文中图、表的数据及各个模型的 R 程序。

```
###数据读取
data<-read.table("data2.txt",header=T,sep="\t",fill=T,stringsAsFactors=F,
encoding="UTF-8")
mem<-read.table("mem.txt",header=T,stringsAsFactors=F)
data_0<-split(data[,1:4],data[,5])$"0"
data_1<-split(data[,1:4],data[,5])$"1"
### kruskal 秩检验
kruskal.test(data$price~data$Add_num)

############绘制任务分布##############
library(ggplot2)
data$executive<-factor(data$executive)
p<-ggplot(data,aes(y,x))+                              #任务位置分布
        geom_point(aes(color=price,shape=executive))+
        scale_colour_gradient(low = 'lightblue', high = 'darkblue')+
        labs(title="task position",x="longitude",y="latitude")+
        theme(plot.title = element_text(hjust = 0.5))

############绘制任务地图################
library(REmap)
```

```
colors <- colorRampPalette(c("pink", "red"))(23)
data_color<-sapply(data[,4],function(x)
colors[match(x,unique(sort(data[,4])))])
barplot(rep(1,23),xaxt="n",yaxt="n",col=colors)     ##绘制颜色 bar
axis(1,seq(0.8,27.2,length.out=23),labels=unique(sort(data[,4])))
loc<-mapNames('guangdong')[c(1,7,12,16,19,20)]
guangdong_map<-data.frame(country = loc,value =
50*sample(6)+200,stringsAsFactors = F)
guangdong_map$value<-as.numeric(guangdong_map$value)
mapdata<-data.frame(lon=data$y,lat=data$x,name=data$num)
pointData<-data.frame(mapdata$name,color = data_color)
map<-remapC(guangdong_map,maptype = "guangdong",color =
'skyblue',
        markPointData = pointData,
        markPointTheme = markPointControl(symbol = 'pin',symbolSize
= 1,effect = F),
geoData = mapdata)

######分区信息
loc<-read.table("sub.txt",header=T,sep="\t")
loc[,2]<- as.numeric(factor(loc[,2]))
apply(loc,1,function(x) match(x[3],data[,2]))
```

```
#######会员分布地图######
colors <- colorRampPalette(c("orange", "green4"))(472)
mem[,6]<-as.character(mem[,6])
data_color<-sapply(mem[,6],function(x)
colors[match(x,unique(sort(mem[,6])))]
barplot(rep(1,472),xaxt="n",yaxt="n",col=colors)      ##绘制颜色 bar
axis(1,seq(0.8,27.2,length.out=23),labels=unique(sort(data[,4])))
loc<-mapNames('guangdong')[c(1,7,12,16,19,20)]
guangdong_map<-data.frame(country = loc,value =
50*sample(6)+200,stringsAsFactors = F)
guangdong_map$value<-as.numeric(guangdong_map$value)
mapdata<-data.frame(lon=mem$y,lat=mem$x,name=mem$m_num)
pointData<-data.frame(mapdata$name,color = data_color)
map<-remapC(guangdong_map,maptype = "guangdong",color =
'skyblue',
            markPointData = pointData,
            markPointTheme = markPointControl(symbol = 'pin',symbolSize
= 1,effect = F),
geoData = mapdata)

######地区价格箱线图#######
```

```
a<-tapply(data$price,data$Add_num,mean)
b<-as.numeric(names(sort(a,decreasing=T)))
c<-split(data,data$Add_num)
d<-c()
for(i in 1:26){
      cc<-c[[b[i]]]
      cc$Add_num<-i
      d<-rbind(d,cc)
}
boxplot(d$price~d$Add_num,xaxt="n",xlab="region",ylab="price",
main="The price in region",col="red")
axis(1,seq(0.8,26,length.out=26),labels=b)
lapply(c,function(x) dim(x)[1])

########会员分布######
mem_loc<-c(table(mem$Add_num)[1:24],others=sum(table(mem$Add_
num)[25:39]))
barplot(mem_loc,col="lightblue",
main="The member in region",xlab="region",ylab="member_num")

######整合信息######
options(stringsAsFactors=F)
```

```
task_num<-table(data$Add_num)
task_num["others"]<-0
member_num<-c(table(mem$Add_num)[1:24],others=sum(table(mem$A
dd_num)[25:39]))
member_num[c("7","8")]<-0
reputation_m<-tapply(mem$reputation,mem$Add_num,mean)
reputation_m<-c(reputation_m[1:24],others=mean(reputation_m[25:39]))
reputation_m[c("7","8")]<-0
task_price<-tapply(data$price,data$Add_num,mean)
task_price["others"]<-0
task_exe.condition<-tapply(data$executive,data$Add_num,function(x)
length(which(x==1))/length(x))
task_exe.condition["others"]<-0
re<-c()
for(i in c(1:26,"others")){

re<-rbind(re,c(i,task_num[i],member_num[i],reputation_m[i],task_price[i
],task_exe.condition[i]))
}
re<-as.data.frame(re)
colnames(re)<-c("region","task_num","member_num","reputation_m","ta
sk_price","exe_condition")
```

```
re<-re[order(as.numeric(re$task_num),decreasing=T),]
write.table(re,"re.txt",quote=F,sep="\t",row.names=F)

########做相关性分析#######
re_d<-matrix(as.numeric(as.matrix(re[-27,-1])),nrow=26)
re_d<-re_d[1:19,]
colnames(re_d)<-colnames(re)[-1]
cor<-cor(re_d,method="pearson")

####相关性热图绘制######
library(pheatmap)
pheatmap(cor, show_colnames= T, show_rownames= T, fondsize= 6.5,
         col = colorRampPalette(c("dodgerblue","white","blue"), alpha
= T)(100),
         main = "correlation",treeheight_col=0)

#######秩和检验#######
wilcox.test(data_1[,4],data_0[,4])   #对完成与未完成任务的标价作秩
和检验
boxplot(data_1[,4],data_0[,4])    #完成与未完成任务的标价箱线图

###########岭回归定价模型#####
```

```
library(glmnet)
price_model<-cv.glmnet(re_d[,1:3],re_d[,4],type.measure="mse",nfolds =
5,a=0)
pre<-predict(price_model, newx = re_d[,1:3], s = 'lambda.min')
cor(pre,re_d[,4])
plot(price_model)
coef(price_model)
con_model<-cv.glmnet(re_d[,1:4],re_d[,5],type.measure="mse",nfolds =
5,a=0)
p_pre<-predict(con_model, newx = re_d[,1:4], s = 'lambda.min')
cor(p_pre,re_d[,5])
coef(con_model)
new_re<-cbind(re_d[,1:3],pre)
new_pre<-predict(con_model, newx = re_d[,1:4], s = 'lambda.min')

############近邻分析##############
dis<-matrix(nrow=dim(data)[1],ncol=dim(mem)[1])
for(i in 1:dim(data)[1]){
      for(j in 1:dim(mem)[1]){
            dis[i,j]<-dist(rbind(data[i,2:3],mem[j,2:3]))
      }
}
```

```
rownames(dis)<-rownames(data)
colnames(dis)<-rownames(mem)
write.table(dis,"dis.txt",sep="\t",quote=F)
dis<-read.table("dis.txt",sep="\t",header=T,check.names=F)
d<-0.05
k<-apply(dis,1,function(x) length(which(x<d)))
k2<-apply(dis,2,function(x) length(which(x<d)))

#########会员优先选择##
mem<-mem[order(mem[,6],decreasing=T),]
d_num<-dim(data)[1]
q_all<-sum(mem[,4])
dis_<-dis
for(i in 1:dim(mem)[1]){
    d_n<-floor((mem[i,4]/q_all)*d_num)
    dis_<-dis_[order(dis_[,i]),]
    count<-0
    while(count<=d_n){
        if(dis_[1,i]<d){
            dis_<-dis_[-1,]
        }
        count<-count+1
```

```
      }
}
data[match(rownames(dis_),rownames(data)),5]


#######任务价格评价模型 ######
repu<-quantile(mem[,6],c(0.8))
d<-0.01
access<-function(data=data,d=0.01,repu=repu,dis=dis,mem=mem,q_all=
q_all,d_num=d_num){
          task_p<-c()
          for(i in rownames(data)){
                m_task<-colnames(dis)[which(dis[i,]<d)]
        p<-c()
                if(length(m_task)>0){
        for(j in m_task){
                    task_m<-rownames(dis)[which(dis[,j]<d)]
                    d_n<-ceiling((mem[j,4]/q_all)*d_num)
                    alpha<-log10(sapply(task_m,function(x)
data[x,4]/dis[x,j]))
                  if(length(task_m)<d_n)
{p[j]<-1/(length(task_m)+0.5);    next()}
                  aa<-sort(alpha,decreasing=T)
```

```
                    if(alpha[as.character(i)]<aa[d_n])
p[j]<-1/((length(task_m)-d_n)*(length(task_m)))          else
                            p[j]<-
1/(length(which(aa>=alpha[as.character(i)]))+0.5)
                    repu_j<-mem[j,6]/repu
                      p[j]<-ifelse(repu_j<1,repu_j,1)*p[j]
                  }
            task_p[i]<-max(p)
          }       else    task_p[i]<-0
        print(i)
     }
   return(task_p)
}
task_s<-access(data=data,d=0.01,repu=repu,dis=dis,mem=mem,q_all=q_
all,d_num=d_num)

#######基于近邻定价#######
dis_r<-dis
dis<-exp(dis_r)
d<-exp(0.1)
find_member<-function(x){
                  return(colnames(dis)[dis[x,]<d])
```

```
}
find_task<-function(x){
                return(rownames(dis)[dis[,x]<d])
}
getprice<-function(mem,task,alpha){
                return(10^alpha*dis[task,mem])
}
options(stringsAsFactors=F)
newprice<-data.frame(task=rownames(data),price="")
qq<-function(l,pr){
      m<-find_member(l)
      for(i in m){
            alpha<-log10(pr/dis[l,i])
            task<-find_task(i)
            for(j in task){
                  newprice[newprice[,1]==j,2]<-getprice(i,j,alpha)
            }
      }
}
l<-1;pr<-1;count<-1
#######近邻定价模型########
near_price<-function(newprice,l="1",pr=1){
```

```
count<-1
while(any(newprice$price=="")&&count<200){
  m<-find_member(l)
                          for(i in m){
                                  alpha<-log10(pr/dis[l,i])
                                  task<-find_task(i)
                                  for(j in task){

newprice[newprice[,1]==j,2]<-getprice(i,j,alpha)
                            }
                    }
if(length(which(newprice[,2]!=""))>0){
i<-sample(which(newprice[,2]!=""),1)
l<-newprice[i,1]
pr<-as.numeric(newprice[i,2])
}else      next()
count<-count+1
print(count)
}
newprice[newprice[,2]=="",2]<-1
return(newprice)
}
```

```
near_price(newprice,l="1",pr=1)
write.table(newprice,"new_price.txt",row.names=F,sep="\t")

########DBscan##########
library(cluster)#做聚类的包
library(fpc)
library(ggplot2)
x<-data[,2:3]
ds <- dbscan(x,0.01,3)
colnames(x)<-c("latitude","longitude")
p <- ggplot(x,aes(longitude,latitude))+
geom_point(size=2.5, aes(colour=factor(ds$cluster)),alpha=I(0.5))
p+theme(legend.position='none')+scale_fill_brewer(palette=2)

########打包模型#########
clu<-predict(ds)
class<-tapply(rownames(data),clu,function(x) x)
m_class<-list()
for(i in 2:length(class)){
    t<-class[[i]]
    m<-c()
    for(j in t){
```

```
            c<-find_member(j)
              d_n<-ceiling((mem[c,4]/q_all)*d_num)
              mm<-mem[c[d_n>length(t)],1]
              m<-c(m,mm)
        }
        m_class[[i]]<-unique(m)
}
names(m_class)<-1:length(m_class)

pa<-m_class[unlist(lapply(m_class,function(x)
length(x)>0&&all(!is.na(x))))]
length(pa)

######小提琴图########
library("vioplot")
new_data<-cbind(data[,1:3],price=as.numeric(newprice[,2])*70)
near_acc<-access(data=new_data,d=0.01,repu=repu,dis=dis)
www<-data
data<-new_data
task_p<-c()
for(i in rownames(data)){
         m_task<-colnames(dis)[which(dis[i,]<d)]
```

```
        p<-c()
        if(length(m_task)>0){
        for(j in m_task){
              task_m<-rownames(dis)[which(dis[,j]<d)]
              d_n<-ceiling((mem[j,4]/q_all)*d_num)
              alpha<-log10(sapply(task_m,function(x) data[x,4]/dis[x,j]))
              if(length(task_m)<d_n)      {p[j]<-1/(length(task_m)+0.5);
next()}
              aa<-sort(alpha,decreasing=T)
              if(alpha[as.character(i)]<aa[d_n])
p[j]<-1/((length(task_m)-d_n)*(length(task_m)))          else
                        p[j]<-
1/(length(which(aa>=alpha[as.character(i)]))+0.5)
              repu_j<-mem[j,6]/repu
              p[j]<-ifelse(repu_j<1,repu_j,1)*p[j]
        }
           task_p[i]<-max(p)
         }        else    task_p[i]<-0
        print(i)
     }
vioplot(raw_acc,near_acc,)
```

```
############新结果#########
new_data<-read.table("new_task.txt",header=T,sep="\t",fill=T,stringsAsF
actors=F,encoding="UTF-8")
new_dis<-read.table("dis_2.txt")
colnames(new_dis)<-1:dim(new_dis)[2]
##运行上述的预测模型程序段进行预测与评估
```