

参赛密码 _____
(由组委会填写)

“华为杯”第十三届全国研究生 数学建模竞赛

学 校 上海交通大学

参赛队号 10248093

队员姓名 1. 钟秋辰

2. 高策

3. 李泽杭

参赛密码 _____
(由组委会填写)



“华为杯”第十三届全国研究生 数学建模竞赛

题 目

具有遗传性疾病和性状的遗传位点分析

摘 要：

近年来，遗传统计学和生物信息学的飞速发展使更多的人投身到探索人体遗传密码的事业中去，据估计人类基因组中每一千个核苷酸就有一个 SNP，人类 30 亿碱基对中共有 300 万以上的 SNP，此类研究由于其本身的特性，通常需要分析和处理大规模数据的能力，动辄几千维的数据给分析和处理带来了一定的困难。本文基于全基因组关联分析（GWAS），主要采用了 AntEpiSeeker 算法，辅助以卡方检验、t 检验、random forest 等算法，解决了一系列层层递进的关联分析问题。

在第一题中，首先对原来的编码方式进行了数值编码转换，理清 SNP 在生物上的性质，如连锁不平衡性，次等位基因频率，边缘效应等，运用合理的数值来代表原来的碱基对（由 A, T, C, G 随机组成），使其在保留生物的一些性质的同时数值化便于对其进行分析和处理。

在第二题中，根据数据的特点，选用 AntEpiSeeker 算法进行求解。AntEpiSeeker 模型使用了卡方检验作为其评价函数，通过蚁群算法在可行解空间中搜索，同时采取了两个阶段的搜索策略来提高蚁群算法在位点的搜索中的效果。此外，利用 t 检验、卡方检验，和随机森林的方法对结果进行检验，t 检验、卡方检验易于操作，随机森林考虑了所有可能。

第三个问题是建立在问题二答案基础上的分类模型，针对数据的分散性，本文分别找到每个位点所在的基因，通过分析单个位点的显著程度，以及基因中显著位点的个数来综合评判基因对性状的调控能力。

第四问是多因素的关联分析问题，对十个相关性状进行综合分析，针对多维的数据，本文首先用 PCA 进行降维，由于十个相关性状之间的相关程度较高，最后可以得到降为一维的主成分数据，为了保证生物学意义，本文针对每个性状单独进行了计算，并将每个结果统计出来得到最终的结论，能够对结果进行合理的生物学意义的解释。

GWAS 局限于解析单个 SNP 位点对疾病易感性的贡献，难以解释大部分复杂疾病的遗传特征。本文通过采取 AntEpiSeeker 算法对致病位点的可能组合进行了

摸索，得出了一些较有价值的结论。

关键词： 蚁群算法；随机森林；遗传统计学；上位性

目录

1	背景回顾.....	- 5 -
1.1	问题引出.....	- 5 -
1.2	全基因组关联性分析 (GWAS) 简述.....	- 5 -
1.3	单核苷酸多态性 (SNP) 简述.....	- 6 -
1.4	基础假设.....	- 7 -
1.1	术语表.....	- 7 -
2	问题分析与建模.....	- 7 -
2.1	问题重述.....	- 7 -
2.2	问题分析与建模.....	- 8 -
2.2.1	位点关联性分析.....	- 9 -
2.2.2	卡方检验.....	- 9 -
2.2.3	双样本 t 检验.....	- 10 -
2.2.4	蚁群算法.....	- 11 -
2.2.5	AntEpiSeeker 模型.....	- 12 -
2.2.6	随机森林.....	- 14 -
3	问题求解.....	- 15 -
3.1	问题一.....	- 15 -
3.2	问题二.....	- 16 -
3.2.1	问题求解.....	- 16 -
3.2.2	卡方检验结果.....	- 17 -
3.2.3	双样本 T 检验结果.....	- 17 -
3.2.4	随机森林检验结果.....	- 19 -
3.3	问题三.....	- 21 -
3.4	问题四.....	- 23 -
3.4.1	相关分析.....	- 23 -
3.4.2	主成分分析.....	- 24 -
3.4.3	问题求解.....	- 26 -
4	模型优缺点.....	- 29 -
5	总结与展望.....	- 30 -
5.1	总结.....	- 30 -
5.2	展望.....	- 31 -
6	参考文献.....	- 32 -
7	附录.....	- 33 -
7.1	代码.....	- 33 -
7.1.1	数据格式相关代码.....	- 33 -
7.1.2	蚁群算法代码.....	- 40 -
7.1.3	分析与作图代码.....	- 61 -

1 背景回顾

1.1 问题引出

人类的遗传物质是 DNA，这是一种带有遗传信息的生物学大分子，储存蛋白的遗传信息。DNA 分子是双螺旋结构，由两条单链互补配对而成。每个核苷酸残基上存在 A,T,C,G 四中不同的碱基，两根单链上的碱基满族 A-T,C-G 的配对关系。人类基因组中大约有 30 亿个碱基对。DNA 长链中具有遗传效应的部分是基因。位点（SNP）为在 DNA 的一些特定位置上的经常发生变异的单个核苷酸，这也是引起 DNA 多态性的原因。在遗传学上，为了区分 SNP 和普通突变，一般把发生概率小于 1% 的认作突变，大于 1% 的变异定义为 SNP。

大量研究表明，人体的许多表型形状差异以及对药物和疾病的易感性都可能与某些位点或包含多个位点的基因相关。如早发性和迟发性老年痴呆症（又称阿尔茨海默病）都受到遗传因素的影响，目前的研究表明，淀粉样前体蛋白基因、早老素 1 基因和早老素 2 基因是家族性早发性阿尔茨海默病的致病基因。慢性阻塞性肺病（COPD）也是一种由遗传因素和环境因素共同导致的疾病。目前普遍认为和其相关性比较大的基因包括肿瘤坏死因子- α 基因、转化生长因子- β 1 基因、 α_1 -抗胰蛋白酶基因、基质金属蛋白酶-9 基因、谷胱甘肽 S 转移酶 P1 基因、微粒体环氧化物水解酶基因和超氧化物歧化酶 3 基因等等。

定位与性状或疾病相关联的位点在染色体或基因中的位置有助于研究人员发现遗传病或性状的遗传机理，有助于控制和防止一些遗传病的发生。[1-3]

1.2 全基因组关联性分析（GWAS）简述

随着人类基因组计划（Human genome project, HGP）和国际人类基因组单体型图计划（The international HapMap project）的完成，以及高通量生物芯片技术的成功研发，人们广泛利用高通量全基因组生物芯片的技术手段，采用全基因组关联性分析（Genome-wide association studies, GWAS）来筛选复杂疾病易感基因。全基因组关联性分析是指在人类全基因组范围内找出存在的序列变异，即单核苷酸多态性（SNP），从中筛选出与疾病相关的 SNPs。

GWAS 研究统计分析原理主要分为两种情况。一是基于无关个体的关联分析，基于此法设计的关联分析包括病例-对照分析法和基于随机群体的关联分析。前者常用于人类疾病易感基因的研究，主要是检测病例组和对照组全基因组中基因型的分布特征和差异，可以通过 4 格表的卡方检验来比较基因频率在研究组和对照组间的差异，若两者之间存在显著性差异则可能表明该遗传差异和疾病有关联，主要关注质量性状。基于随机群体的分析法主要应用于动植物中，主要关注数量性状。比如在研究动物经济性状候选基因时可采用这种方法。

二是基于家系的关联分析，基于无关个体的关联分析可靠性会受到样本群体分层或其他混杂因素的影响。基于家系的关联分析可以有效提高分析的可靠性，避免群体分层对关联分析结果的影响（如果利用多个家系同样有可能产生群体分层）。当选择的样本具有完整的系谱信息时，可以采用传递不平衡检验法（TDT）对 SNP 与所关注数量性状的关联效应进行分析。

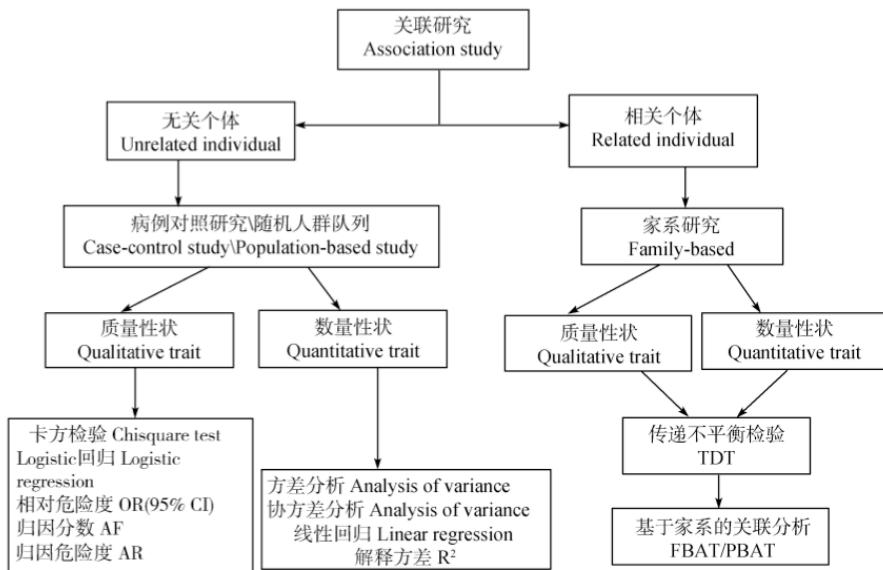


图 1-1 GWAS 的统计分析方法

典型的 GWAS 案例通常包括以下四个部分：

- 1) 建立研究群体，选择尽可能大的群体作研究样本，建立目标性状数据库。如果研究性状为疾病，要同时选择已感染疾病的群体和健康群体作病例-对照组合；
- 2) 提取样本 DNA 并进行质量控制以达到基因分型的要求，对基因型数据进行检测和质量控制以达到后续关联分析的要求；
- 3) 利用合适的统计模型对 SNP 和目标性状进行关联分析；
- 4) 对关联分析的结果进行高级分析及验证。

然而 GWAS 局限在它解析的只是单个 SNP 位点对疾病易感性的贡献，而在大部分情况下，复杂疾病的致病位点不止一个，单个位点的效应微小，GWAS 难以解释大部分复杂疾病的遗传特征。^[4-5]

1.3 单核苷酸多态性（SNP）简述

一般将 DNA 的不同称作 DNA 多态，这些多态导致了人的天生体质、对特定疾病的敏感性、对药物的适应性等等的不同。SNP 是其中一种多态的形式，包括转换、颠换、缺失和插入，形成的遗传标记数量多，多态性丰富。根据对生物的遗传性状的影响，SNP 可以分为同义突变和非同义突变两种，如果一个突变没有引起其他编码的氨基酸变化，这个突变就是同义突变；如果碱基序列的改变使以其为蓝本翻译的蛋白质序列发生改变，从而影响蛋白质的功能，那这个突变就是非同义突变，非同义突变又分为错义突变和无义突变，错义突变改变遗传密码子，导致氨基酸种类变化，无义突变产生一个终止密码子。

每一个 SNP 位点都可以有 4 种不同的变异形式，但实际上发生的只有两种，即转换和颠换，转换的发生率明显高于其他变异，一般是 2: 1。转换是指嘌呤与嘌呤之间或嘧啶与嘧啶之间的相互转换，包括一种情况，C>T (G>A)。颠换指的是嘌呤与嘧啶之间的相互替代，有三种情况，C>A (G>T)、C>G (G>C)、T>A (A<T)。理论上讲，SNP 既可能是二等位多态性，也可能是 3 个或 4 个等

位多态性，但实际上后两者非常少见，基本可以忽略。由于这种二等位性，在检测时无需分析片段长度，有利于快速的检测筛选。

SNP 已经成为了第三代“遗传标志”，在人群多样性、疾病基因定位和药物基因组学的研究中都发挥了重要的作用。国际人类基因组单体型图计划旨在建立一个高分辨率的人类 SNP 位点图谱，为群体遗传学的研究提供数据，为定位遗传性疾病的致病基因提供高密度的 SNP 位点。[6]

1.4 基础假设

为了研究的顺利进行，本文在研究开始之前对一些有争议的内容和基础的遗传学知识进行了必要假设和介绍。本文的研究将基于以下基本假设展开：

- 1) 本文涉及的 9945 个位点位于同一染色体上且该染色体为常染色体。
- 2) 位点之间的变换相互独立。
- 3) 位点决定的信息与其所在的位置无关。

1.5 术语表

符号	含义
iAntCount	蚂蚁数量
iItCountLarge	第一次迭代次数
iItCountSmall	第二次迭代次数
alpha	信息素沉积的权重
iTopModel	第一次迭代的最大数
iTopLoci	第一次迭代的最大位点
rou	信息素挥发速度
phe	初始信息素水平
largehapsize	大单倍型的尺寸
smallhapsize	小单倍型的尺寸
iEpiModel	位点集合中位点个数
pvalue	p 值阈值
SNP	位点

注：在各个问题的探讨中，有一些只与题目有关的术语定义在其中。

2 问题分析与建模

2.1 问题重述

因为染色体具有双螺旋结构，一个位点的信息用两个碱基的组合来表示，如在位点 rs100015 位置共有三种不同的编码方式 TT,TC 和 CC。问题一指出，碱基

(A,T,C,G) 的编码方式不利于进行数据整理和分析，希望能寻求到适当的数值编码转换方式。因此问题一在于如何将位点的数据转化为机器可识别的，便于进行数据分析与处理的格式。

问题二在问题一的基础上，要求给出题目给出的某种疾病最有可能的一个或几个致病位点。在问题一的建模后，数据变为了便于进行数据分析的格式，为该问题做了事前的准备。

在该问题中，一共有 1000 个样例数据，其中每个样例数据有是否患有该疾病，以及 9445 个位点的编码信息。这在全基因组关联性分析中，是典型的病例对照研究（Case-Control Study）。

本题的问题在于找出疾病最有可能的致病位点。而业界目前采取的病例对照研究可以较好地得到问题的结果。

问题三在第二题已经得到最可能的致病位点的基础上，要求我们找出最有可能的致病基因，一个基因是位点的集合，相比于第二问，分析的维度要大一些，由位点为个体变为了位点的集合。

人体的许多遗传疾病和性状是有关联的，如高血压，心脏病、脂肪肝和酒精依赖等，把相关的性状或疾病放在一起研究能提高发现致病位点或基因的能力。问题四在问题二的基础上进行了延伸，提供了十个相关的性状，需要我们找出和这十个性状有关联的位点。

2.2 问题分析与建模

传统的重大疾病的研究模式，是以假说为导向，即首先假设某基因为致病候选基因，再对其进行深入研究，这样的方式无疑不适合本次题目的要求。而现代的、工业化的重大疾病的研究模式，是以数据为导向，对物种或个体进行基因组水平、大规模化研究，即在掌握基因组全局的情况下，发现影响疾病的基因位点。
[9]

目前，基于基因数据的分析有两个方向，一个是连锁分析，一个是关联分析。这两类方法在实验设计、研究对象、采用的技术手段以及对结果的解释等方面均不同。连锁分析是从分析但基因疾病发展起来的，它利用基因组上的微卫星作为遗传标记。在患病家系中观察是否存在某些遗传标记的传递模式显著偏离预期。

而关联分析通常根据某些间接线索、比如基因的位置和基因的功能，选定一个或几个候选基因，通过直接测序或基于 PCR 的方法如等位基因特异性扩增等在病例和对照中比较候选基因的序列差异，然后来确定这些候选基因与患病状态或数量性状之间的关系。

通过分析已有的数据，我们发现，找出疾病最有可能的致病位点，可以重新表述为找到与疾病关联性最大的一个或几个位点。尽管在逻辑上，关联性并不能完全等同于因果性。但是因为该问题是从统计与学习的角度出发，从结果出发进行病例对照的研究，进而分析出最有可能的致病位点。因此疾病与位点之间的因果关系不可能通过已有的数据得出，因此我们将问题转变为对于疾病与位点之间的关联性的分析。这也是病例对照研究的基本思想与思路。

在关联分析与连锁分析之后，我们之所以选择了关联分析作为我们主要的分析方式，一方面因为关联分析能够分析出多个位点或基因相互影响的情况下的多位点致病因素，“多基因假说”认为，一个复杂形状由多个微效基因控制，并存在基因间上位性效应，同时这些单基因和上位性效应对环境敏感。而且同样因为

题目给定的数据不能支撑我们对其进行连锁分析。

因此，我们的重点放在了如何进行病例对照研究中疾病与位点之间的关联性分析。这其中主要涉及了位点的关联分析方法，以及机器学习分类的算法。

2.2.1 位点关联性分析

位点的关联分析方法，是指利用统计学等知识，对位点集合与疾病的关联性进行分析，进而得到与疾病最相关的位点集合。在位点关联性分析中，主要有两方面会影响其结果。一方面是关联性评价方法，另一个方面是位点集合搜索方法。

其中，关联性评价是评价疾病与位点的关联性的函数。位点与疾病的关联性越大，关联性评价就会越高，因此位点就更有可能是疾病的致病位点。目前，有很多被广泛使用的关联性评价方法。比如互信息以及卡方检验。其中卡方检验是一个统计学中最常用的假设检验。

2.2.2 卡方检验

卡方分布是指，如果有 n 个相互独立的随机变量 $X_1, X_2, X_3, \dots, X_n$ ，它们都服从正态分布，即

$$X_i \sim N(0, 1), i = 0, 1, 2, \dots, n$$

则称统计量

$$X^2 = X_1^2 + X_2^2 + \dots + X_n^2$$

服从自由度为 n 的 X^2 分布，即卡方分布，其中自由度为右式中所含有的独立变量的个数。

而 X^2 检验是一种基于卡方分布的假设检验，多用于二阶方差的检验，在分类资料的统计推断中的应用较为广泛。卡方检验针对分类变量统计样本的实际观测值与理论推断值之间的偏离程度，实际观测值与理论推断值之间的偏离程度就决定卡方值的大小，卡方值越大，越不符合；卡方值越小，偏差越小，越趋于符合。

它的基本无效假设是：

H_0 : 行分类变量与列分类变量无关联

H_1 : 行分类变量与列分类变量有关联

$$\chi^2(i) = \sum_{i=1}^k \frac{(A_i - T_i)^2}{T_i}$$

统计量卡方，其中 A_i 是样本资料的计数， T_i 是在 H_0 为真的情况下的理论值，即，期望值。在 H_0 为真时，实际观察数与理论数只差 $A_i - T_i$ 比较接近 0，所以在 H_0 为真时，检验统计量卡方服从自由度为 $k-1$ 的卡方分布。

由于卡方分布是一种连续型分布，而本题目中的数据是离散数据，所以要采用列联表卡方检验的方法，而由此计算的卡方值得抽样分布也是不连续的，本题目中样本量 $n > 40$ ，因此可以采用 Yates 连续性校正卡方值。

列联表卡方检验用于多个率或多个构成比的比较。同一组对象，观察每一个个体

对两种分类方法的表现，结果构成双向交叉排列的统计表就是列联表。因为其基本数据有 R 行 C 列（自由度 $df = (C-1)(R-1)$ ），故通称 $R \times C$ 列联表（contingency table），简称 $R \times C$ 表。将理论频数 $T=nRnC/n$ 代入基本公式可以推导出：

$$\chi^2(i) = n \left(\sum \frac{A^2}{n_R n_C} - 1 \right)$$

本题目中将原始数据分为病例组和对照组，检测两组位点的分布特征和差异，用 2×3 列联表的卡方检验来比对，若两者之间存在显著性差异则可能表明该遗传差异和疾病有关联。

本题目中有两个分类变量 0 和 1，分别代表患遗传病 A 和不患遗传病 A，它们各自对应基因型 AA, AB 和 BB（其中，A、B 分别代表了 A、T、C、G 四种碱基中的两种），其样本频数列联表为：

表格 2-1 联表				
基因型	AA	AB	BB	总计
病例组 (1)	a	b	c	a+b+c
对照组 (0)	d	e	f	d+e+f
总计	a+d	b+e	c+f	a+b+c+d+e+f

假设：

H0：病例组和对照组无关联

H1：病例组和对照组有关联

若要推断的论述为“H1：病例组和对照组有关系”，由表中的数据算出统计变量 K^2 的值（即 K 的平方）， K^2 的值越大，说明“H1：病例组和对照组有关系”成立的可能性越小，进而可以考察两个变量是否有关联。

2.2.3 双样本 t 检验

T 检验，亦称 student t 检验（Student's t test），主要用于样本含量较小（例如 $n < 30$ ），总体标准差 σ 未知的正态分布资料。t 检验是用 t 分布理论来推论差异发生的概率，从而比较两个平均数的差异是否显著。

由于本题目缺少对于目标总体的数据分析，所以不能进行单总体 t 检验，而是要做双总体 t 检验。因为在问题一中我们已经将每个位点对应的碱基对进行了数值编码，因此，题目可以简化为检验两个样本平均数与其各自所代表的总体的差异是否显著。

双总体 t 检验又分为两种情况，一是独立双样本 t 检验（independent two sample t test），一是配对双样本 t 检验（paired two sample t test）。因为本题目中 500 名患有遗传病 A 和 500 名不患遗传病 A 的样本之间，没有在年龄、性别、家族史等因素上进行匹配，所以不能进行配对样本 t 检验，在实际病例分析中，样本选取需要考虑其他无关因素的影响，尽量选取匹配较好的数据进行分析。

独立双样本 t 检验统计量为：

$$t = \frac{\bar{X}_1 - \bar{X}_2}{\sqrt{\frac{(n_1-1)S_1^2 + (n_2-1)S_2^2}{n_1+n_2-2} \left(\frac{1}{n_1} + \frac{1}{n_2} \right)}}$$

S_{12} 和 S_{22} 为两样本方差; n_1 和 n_2 为两样本容量。

独立样本 t 检验适用于已知一个总体均数, 样本来自正态或近似正态总体, 并且能够得到一个样本均数及该样本标准差的分析, 由于 t 检验的前提是原始资料服从正态分布, 针对本题目中的数值结果, 我们尝试采用双样本非参数检验代替 t 检验、与 k-fold crossValidation (交叉验证) 联用的方法进行两组间均值的比较。

基于本题目, 我们做出以下假设:

H0: 假设两组样本所代表的正态分布的总体的均值是相同的

H1: 假设两组样本所代表的正态分布的总体的均值是不同的

在 matlab 中采用 ttest2 方法来实现独立双样本 t 检验, 常用用法:

[H,P,CI, STATS]=ttest2(x,y, ALPHA);)

ALPHA 默认为 0.05, 即计算 x 和 y 在 5% 的显著性水平下假设 H0 是否被接受。当返回的结果 H=0 时, 则表明零假设在 5% 的置信度下不被拒绝, 在本题中即表示, 该位点在病例和对照两组中的双等位碱基序列不存在显著的差异; 如果返回值 H=1, 表明假设 H0 被拒绝, 即该位点在病例和对照两组中的双等位碱基序列存在显著的差异, 即, 位点对遗传病 A 具有显著的调控作用。

结果中 P 值代表显著性 (significance), P 越小, 说明越有理由拒绝 H0 , 越有理由说明两者有差异, CI 为置信区间(confidence interval), 表示达到 $100*(1-\text{ALPHA})\%$ 的置信度的数据区间, 当此区间不跨过 0 时, 表示结果可信度较强。

2.2.4 蚁群算法

若某一数据有两个不同的特征 A 与 B , 这两个特征之间的关联性求法如下。设此两个特征的取值均为离散, 特征 A 的取值范围为 $a_1, a_2, a_3, \dots, a_m$, 特征 B 的取值范围为 $b_1, b_2, b_3, \dots, b_n$ 。假设此数据中, 在 (A, B) 两特征中取值 (a_i, b_j) 的样本数为 t_{ij} , 在特征 A 取值样本 a_i 的样本数为 p_i , 在特征 B 取值样本 b_j 的样本数为 q_j , 即有如下公式:

$$p_i = \sum_j t_{ij}, i = 1, 2, 3, \dots, m$$

$$q_j = \sum_i t_{ij}, j = 1, 2, 3, \dots, n$$

设所有样本总数为 N。此数据的观察值列连表如表所示:

表格 2-2 连表

	b_1	...	b_n	Sum(B)
a_1	t_{11}	...	t_{1n}	q_1
...
a_m	t_{m1}	...	t_{mn}	q_n
Sum(A)	p_1	...	p_m	N

提出零假设 H_0 : 特征 A 与特征 B 无关联性。备择假设 H_1 : 特征 A 与特征 B 有
关联性。如果 H_0 为真, 说明特征 A 与特征 B 相互独立, 则可以进行 p 值检验, p
值越大, 零假设越可信, 两个特征的关联性越小。

SNP 关联分析算法要求的是位点集与样本类标之间的关联性。在应用卡方检
验时, 只需要计算位点集基因型与样本集表现型之间的列连表, 再根据列连表便
可计算出 p 值, 并根据此来评价关联性大小。[6]

卡方分布是常用的评价关联性的函数, 在搜索位点集合的算法中, 常用智能
启发式算法来搜索合适的位点集合。其中的蚁群算法模型是一个实现简单, 效率
尚可的启发式搜索算法。

蚁群算法是意大利学者 Dorigo M 等人于 1991 年提出的一种模拟进化算法,
最初称之为蚁群系统(Ant Colony System, ACS)。其受到蚂蚁觅食时, 总是能找到
巢穴与食物地点之间的最短路径的启发。吸收了真实蚂蚁的行为特征, 通过模拟
蚁群寻找食物的过程来完成对问题的求解。该方法被广泛应用于数据分析、机器
人协作、电力、通信、交通等领域。

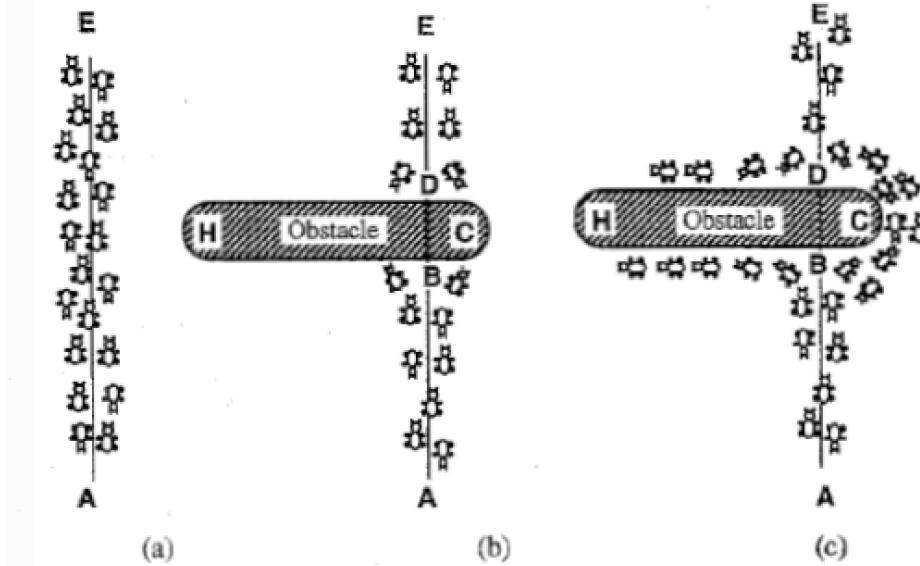


图 2-1 蚁群算法示意图

在图中, 有一群蚂蚁, 假如 A 是蚁巢, E 是食物源 (反之亦然)。这群蚂蚁将
沿着蚁巢和食物源之间的直线路径行驶。假如在 A 和 E 之间突然出现了一个障
碍物 (图 (b)), 那么, 在 B 点或 D 点的蚂蚁将要做出决策, 由于一开始路上没有
前面蚂蚁留下的信息素 (pheromone), 蚂蚁朝着两个方向行进的概率是相等的。
但是当有蚂蚁走过时, 它将会在它行进的路上释放出信息素, 并且这种信息素会
以一定的速率散发。信息素是蚂蚁之间交流的工具之一。它后面的蚂蚁通过路上
信息素的浓度, 做出决策, 而沿着短边的的路径上信息素将会越来越浓(图 1(c)),
从而吸引了越来越多的蚂蚁沿着这条路径行驶。

蚁群算法通过多次迭代, 更新信息素的方式, 模拟蚁群的决策方式, 来搜索
位点集合。

2.2.5 AntEpiSeeker 模型

AntEpiSeeker 模型[11]是在 2010 年由 Yupeng Wang 在 BioInformatics 上提
出的一个分析位点关联性的算法模型, AntEpiSeeker 模型可以用来进行多个位

点的致病因素关联分析。

AntEpiSeeker 模型使用了卡方检验作为其评价函数。通过蚁群算法在可行解空间中搜索。AntEpiSeeker 有对边缘效应鲁棒性高的优点，但它受蚁群算法参数设置的影响较大，而这些参数往往靠经验设置。另外，AntEpiSeeker 还采用了两轮蚁群算法，对这两轮迭代过程，还要分别设置参数。

AntEpiSeeker 需要指定所用蚂蚁数目 $iAntCount$ ，蒸发率 ρ ，初始信息素 τ_0 ，寻找位点集合中位点的数目 n ，第一轮搜索位点数目 N_1 ，第二轮搜索位点数目 N_2 ，以及每轮迭代次数 $iter_max$ 。在一轮迭代中，每只人工蚂蚁会按顺序选择一组位点，这组位点便是一个可行解。假设共有 L 个位点。某只蚂蚁在决策时是否选择位点 k 的概率公式类似于下式：

$$\rho_k(i) = \frac{(\tau_k(i))^{\alpha} \eta_k^b}{\sum_{j=1}^L (\tau_j(i))^{\alpha} \eta_j^b}$$

其中 $\tau_k(i)$ 是第 i 轮第 k 个位点上的信息素。其中 $\eta_k^b = 1$ ，其物理含义为每个蚂蚁在进行新的选择时只考虑信息素的影响，而不关心评价函数。这样是出于提高算法效率的考虑。如果每次蚂蚁在进行新的选择时都会考虑评价函数，则运算的时间复杂度太高，会影响算法的效率，对系统的 CPU 等运算单元有更高的要求。

在迭代结束后，每一个蚂蚁都会从 L 个位点中选择出 n 个位点，这 n 个位点组成了一个集合 S_m ，其中 m 代表蚂蚁的编号。其更新的公式为：

$$\tau_k(i+1) = (1 - \rho)\tau_k(i) + \Delta\tau_k(i)$$

其中 $\Delta\tau_k(i)$ 是 S_m 与疾病之间的 $0.1 \cdot X^2$ 。

为了提高蚁群算法在位点的搜索中的效果，AntEpiSeeker 模型采取了两个阶段的搜索策略。在第一阶段中，AntEpiSeeker 模型用蚁群算法搜索含有大于给定位点数目 n 的位点子集，选出卡方统计量最大的前 M_1 个子集作为候选子集，再选出信息素最大的前 M_2 个位点作为候选位点，在候选子集与候选位点中，通过穷举搜索含有给定数目 n 个位点的集合，评价函数为卡方统计量的 p 值。候选子集的选取在探测边缘作用较小的因素中，起到了缩小解空间的作用。候选位点则是为了探测边缘作用比较大的位点。

重复上面两个阶段的搜索，AntEpiSeeker 采用了两轮搜索策略。在第一轮中，搜索包含位点数为 $largesetSize$ 的候选子集，第二轮中，搜索所含位点数为 $smallsetSize$ 的候选子集。对两轮结果进行统计，选择卡方检验中 p 值小于一个阈值的解为计算结果。

因为两轮搜索后的计算结果中可能包含有相同的位点，所以 AntEpiSeeker 对结果采用了最小化假阳率(Minimizing false positive)处理。将最小化假阳率前的结果用 EI_{all} 表示， EI_{all} 初始化为两轮搜索后的结果，用 EI_m 表示最小化假阳率后的结果， EI_m 初始化为空。对 EI_{all} 中的每一个元素 I_i 执行以下操作：若 EI_m 中没有元素与 I_i 含有同样的位点，则将 I_i 移至 EI_m 中；若 EI_m 中有元素 J_j 与 I_i 含有至少一个相同的位点，则比较两者的 p 值，将较大的保留在 EI_m 中，另外一个将会被丢弃。AntEpiSeeker 模型计算的步骤如下所示。

输入: $iEpiModel$ 位点子集中位点数目, $PvalueSignificance$ 统计显著门限值, 默认 0.01, $Largesetsize$ 两轮搜索子集中 SNP 位点数目, $Smallsetsizes$ 两轮搜索子集中 SNP 位点数目, $iAntCount$ 人工蚂蚁数, $iItCountLarge$ 两轮蚁群算法迭代次数, $iItCountSmall$ 两轮蚁群算法迭代次数, $iTopLoci$ 候选位点数目, $iTopMode$ 位点候选子集数目, τ_0 每个位点上初始化信息素, ρ 蒸发率, α 蚂蚁走过某位点时留下的信息素

输出: P 值小于 $Pvalue - significance$ 的位点集合

```

1: function ANTEPISEEKER( $iEpiModel$ ,  $PvalueSignificance$ ,  $Largesetsize$ ,  $Smallsetsizes$ ,
 $iAntCount$ ,  $iItCountLarge$ ,  $iItCountSmall$ ,  $iTopLoci$ ,  $iTopMode$ ,  $\tau_0$ ,  $\rho$ ,  $\alpha$ )
2:    $setsize \leftarrow Largesetsize$ 
3:    $iItCount \leftarrow iItCountLarge$ 
4:   for  $i = 0 \rightarrow iItCount$  do
5:     蚂蚁选择位点集
6:     计算每只蚂蚁选出子集的卡方统计量, 更新信息素
7:     记录卡方统计量最高的子集, 清除蚂蚁
8:   end for
9:   找出所有含有  $iEpiModel$  个位点的集合并且 p 值小于  $PvalueSignificance$  的所有集合
10:   $setsize \leftarrow Smallsetsizes$ 
11:   $iItCount \leftarrow iItCountSmall$ 
12:  for  $i = 0 \rightarrow iItCount$  do
13:    蚂蚁选择位点集
14:    计算每只蚂蚁选出子集的卡方统计量, 更新信息素
15:    记录卡方统计量最高的子集, 清除蚂蚁
16:  end for
17:  进行最小化假阳率
18: end function

```

代码 2-1 AntEpiSeeker 算法步骤

AntEpiSeeker 模型相比于连锁分析等其他方法, 具有很多优点。首先 AntEpiSeeker 模型充分考虑了上位作用。上位作用是指对等位基因受到另一对等位基因的制约, 并随着后者不同前者的表型有所差异, 后者即为上位基因 (epistatic gene)。这一现象称为上位效应(epistasis)。其他诸如连锁分析等, 都只是将位点作为独立的变量进行分析, 而 AntEpiSeeker 模型则充分考虑了该作用的影响。其实际表现为两个位点各自与疾病的关联并不大, 但是两个位点组合起来就有可能与疾病有很大关联, 而 AntEpiSeeker 模型就可以发现这样的位点组合。

2.2.6 随机森林

随机森林, 是用随机的方式建立一个森林, 森林里面有很多的决策树组成, 随机森林的每一棵决策树之间是没有关联的。在得到森林之后, 当有一个新的输入样本进入的时候, 就让森林中的每一棵决策树分别进行一下判断, 来决定这个样本应该属于哪一类。

在建立每一棵决策树的过程中, 有两点需要注意, 即采样与完全分裂。首先是两个随机采样的过程, 随机森林对输入的数据要进行行、列的采样。对于行采样, 采用有放回的方式, 也就是在采样得到的样本集合中, 可能有重复的样本。

假设输入样本为 N 个，那么采样的样本也为 N 个。这样使得在训练的时候，每一棵树的输入样本都不是全部的样本，使得相对不容易出现过拟合的现象。然后进行列采样，从 M 个 feature 中，选择 m 个 ($m \ll M$)。之后对采样之后的数据使用完全分裂的方式建立出决策树，这样决策树的某一个叶子节点要么是无法继续分裂的，要么里面的所有样本的都是指向的同一个分类。一般很多的决策树算法都一个重要的步骤，那就是剪枝，但是随机森林并不需要剪枝的过程，由于之前的两个随机采样的过程保证了随机性，所以就算不剪枝，也不会出现过度拟合的情况。

随机森林的算法步骤如下所示。

输入: $ntree$ 树的数目
输出: 预测结果

```
1: function ANTEPISEEKER( $ntree$ )
2:   通过自助法 (bootstrap) 构建大小为  $n$  的一个训练集，即重复抽样选择  $n$  个训练样例
3:   对于刚才新得到的训练集，构建一棵决策树  $tree$ 
4:   for  $i = 0 \rightarrow ntree$  do
5:     for  $i = 0 \rightarrow tree.nodes$  do
6:       通过不重复抽样选择  $d$  个特征
7:       利用上面的  $d$  个特征，选择某种度量分割节点
8:     end for
9:   end for
10:  对于每一个测试样例，对  $ntree$  颗决策树的预测结果进行投票。票数最多的结果就是随机森林的预测结果。
11: end function
```

代码 2-2 随机森林算法步骤

随机森林中构建决策树的做法和原始决策树的区别是，在每次分割节点时，不是从所有特征中选择而是在一个小特征集中选择特征。

虽然随机森林模型的可解释性不如决策树，但是它的一大优点是受超参数的影响波动不是很大。我们也不需要对随机森林进行剪枝，因为集成模型的鲁棒性很强。在实际运用随机森林模型时，树的数目需要好好调参。一般，树的数目越大，随机森林的性能越好，当然计算成本也越高。

样本大小 n 能够控制 **bias-variance** 平衡，如果 n 很大，我们就减小了随机性因此随机森林就容易过拟合。另一方面，如果 n 很小，虽然不会过拟合，但模型的性能会降低。大多数随机森林的实现， n 的大小等于原始训练集的大小。

3 问题求解

3.1 问题一

一个位点的信息通常用两个碱基的组合来表示。SNP 样本数据可能有不同的形式，但通常用数字 1 和 2 来表示 SNP 的两个状态，所以 SNP 的特征状态可以

表示为：(1, 1)、(1, 2)、(2, 1)、(2, 2)，由于双螺旋结构，(1, 2) 和 (2, 1) 被认为是一种 SNP 特征状态，这样可以用一个数字来表示一个特征状态。

由于各位点之间相互独立，不同位点所包含的编码方式之间也互相独立，所以本问选用数字 0, 1, 2 分别对应 (1, 1), (1, 2) 或 (2, 1), (2, 2) 这三种特征形态。

题目规定 SNP 的样本分为没有患病和患有遗传病的样本，用数字 0 代表没有患病的样本，用 1 代表患有遗传病的样本。

SNP 的样本序号、类别和 SNP 的特征状态数值共同构成了本文使用的样本数据。

除此之外，我们在分析数据的过程中也使用了很多第三方提供的软件，如 PLINK 等，这些软件有自己定义的格式与方法。

3.2 问题二

3.2.1 问题求解

性状表现型数据可以分为两类，一类是病例对照数据，另一类是数量性状数据。病例对照数据属于二项式分布数据，一般采用 Pearson's 卡方测验和 logistic 回归分析。本题目的研究目标是 1000 个样本上的 9445 个位点，针对 1000 个样本的性状表现可以将其分成两组，分别是包含 500 个患有遗传病 A 的病例组，和 500 个不患有遗传病 A 的对照组。

为了达到找出导致位点的目的，我们需要对病例组和对照组在每个位点的基因型进行分析，分析在两组样本中出现同一基因型的频率是否有显著差异，如果差异较为显著，则可以提出假设，该位点对遗传病 A 的发生具有调控作用。

本题目条件中缺少对样本生存环境和家族遗传的分析，所以我们假设目标样本的性状仅受基因型的影响。

在本题中，我们使用了 AntEpiSeeker 模型分别从单个位点到多个位点进行计算，得到与该疾病关联最大的位点集合。并辅以随机森林和逻辑回归的结果进行综合考量，最后采取随机森林的方法训练分类器，进行结果的检验。

因为 AntEpiSeeker 模型需要给定一些初始参数，而这些参数对于结果会有影响，因此在使用由 AntEpiSeeker 论文作者提供的数据进行参数调优后，我们采取了如下的初始参数：

表格 3-1 初始参数表

变量名	值	备注
iAntCount	2000	蚂蚁数量
iItCountLarge	150	第一次迭代次数
iItCountSmall	300	第二次迭代次数
alpha	1	信息素沉积的权重
iTopModel	1000	第一次迭代的最大数
iTopLoci	200	第一次迭代的最大位点
rou	0.05	信息素挥发速度
phe	100	初始信息素水平
largehapsize	6	大单倍型的尺寸

smallhapsize	3	小单倍型的尺寸
iEpiModel	1-N	位点集合中位点个数
pvalue	0.00001	p 值阈值

修改 iEpiModel，将其从 1 开始，依次得到与疾病最有关联的 n 个位点集合。因为机器配置以及运行时间所限，在本题中只得到了 iEpiModel 为 1 和 2 时的两组解，解如下所示：

表格 3-2 解

iEpiModel	位点集合	卡方	P 值
1	2938(rs2273298)	28.8362	5.47405e-07
2	2938(rs2273298) 6077(rs11573253)	54.3938	5.78981e-09

3.2.2 卡方检验结果

卡方检验关联分析是为了检测病例组和对照组之间的关联，当两组数据之间没有关联时才可以被视为互相独立，通过下表的数据可以看出存在一些数据的病例组和对照组之间有着较强的相关性，建议在之后的分析中将这些数据剔除。

表格 3-3 卡方检验结果

位点 ID	RS 编码	P-value	K2
96	'rs12072310'	6.65E-07	34.24114389
234	'rs12758705'	2.25E-07	36.52674828
4912	'rs2076595'	4.14E-14	68.76584223
4919	'rs3818034'	1.11E-06	33.14844922
6473	'rs12758257'	2.20E-08	41.41826379
7460	'rs3765428'	6.31E-07	34.35231266
7977	'rs4659413'	1.51E-06	32.50895613
8573	'rs4360499'	7.86E-06	28.98720955

3.2.3 双样本 T 检验结果

通过双样本 T 检验，得到结果：在 alpha 设置成 0.05 时，通过对 P 值的设置，得到了以下的计算结果：

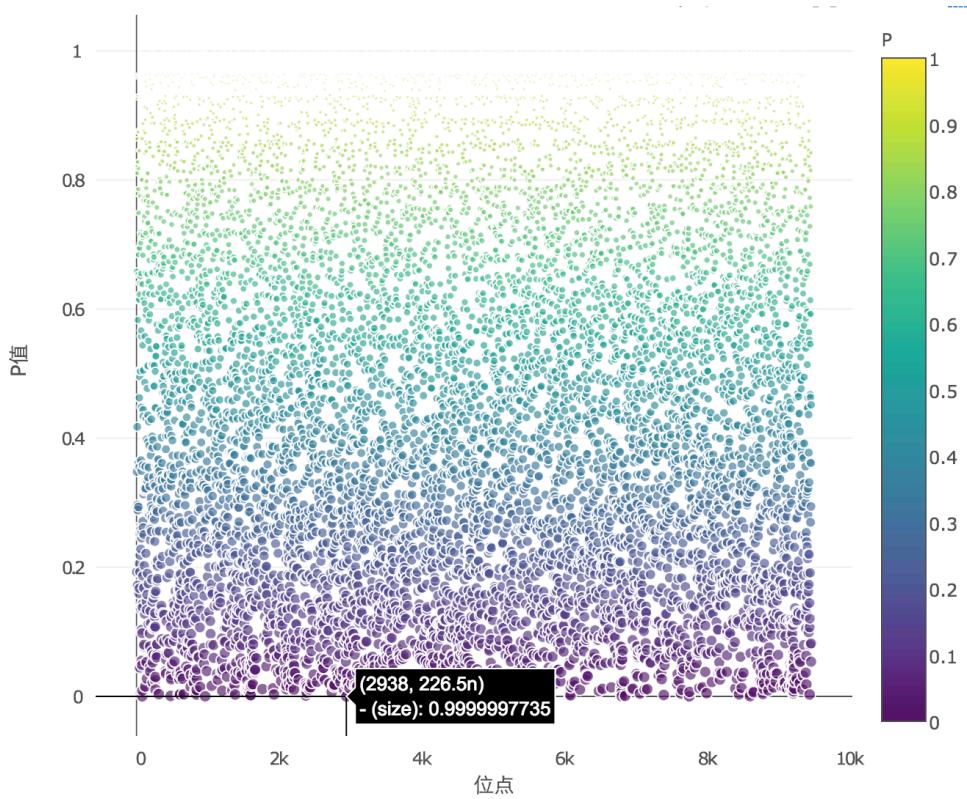


图 3-1 P 值分布

在 $P < 0.01$ 的情况下，结果中包含 96 个位点，即，有 96 个位点的碱基序列在病例组和对照组之间存在显著差异。在本题目中，要得到的位点数量有限， $P < 0.01$ 的条件过于宽松，过滤得到的位点数目过多，因此选取更加严格的过滤条件：在 $P < 0.001$ 的情况下，结果中包含 8 个位点；在 $P < 0.0001$ 的情况下，结果中包含 3 个位点。

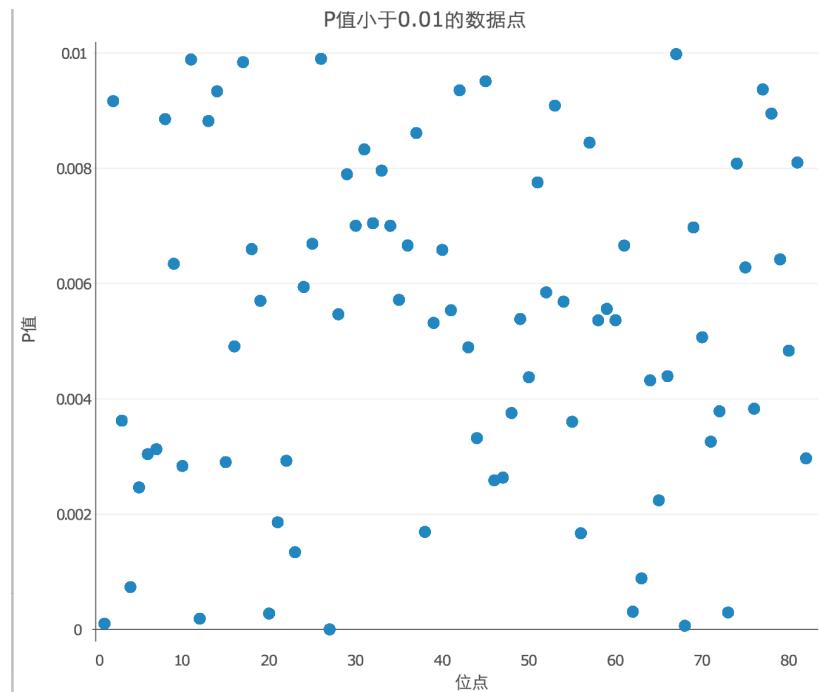


图 3-2 P 值小于阈值的分布

表中的数据包含了在 $P < 0.001$ 的情况下 8 个位点的 ID、rs 编码以及对应的 P 值和 95%CI 区间。

表格 3-4 检验结果

位点 ID	RS 编码	P-value	CI[95%]	
757	'rs880801'	0.000327154	-1.00996365	-0.29803635
1195	'rs1541318'	0.000542335	-0.82344549	-0.22855451
1541	'rs7368252'	0.000168989	-1.41638994	-0.44761006
2938	'rs2273298'	1.05E-06	-1.01602416	-0.43597584
4932	'rs2143810'	0.00055248	0.263684455	0.952315545
5937	'rs1883567'	0.000627427	-1.50916721	-0.41083279
7737	'rs932372'	5.37E-05	-0.78343331	-0.27256669
8380	'rs7543405'	3.87E-05	0.573616116	1.610383884

T 检验在本题目中有较多局限，得到的结果不是最准确最完整的，但是可以较为宽泛地得到大致结果，为之后更准确的计算方法做铺垫和验证。

3.2.4 随机森林检验结果

我们首先将 9445 个位点数据分别作为变量，与样例的患病情况一起进行随机森林训练，并对其进行测试，其中误判率与位点之间的关系如图所示：

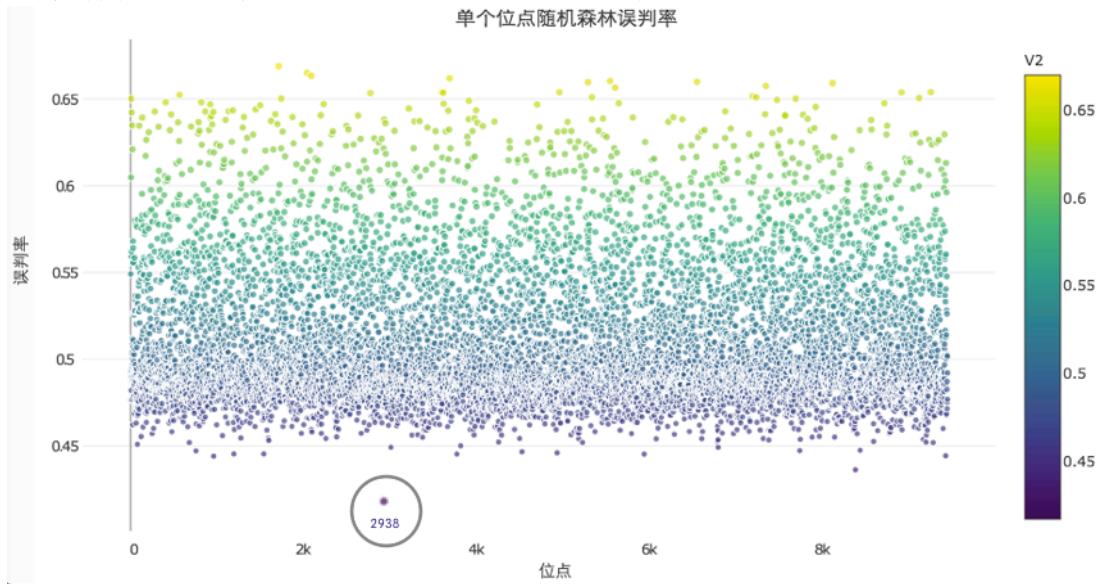


图 3-3 随机森林误判率

在图中，X 轴为位点的 ID，从 0 开始，一直到 9445，而 y 轴是误判率。图中的点代表了由当前位点作为变量，所训练出的随机森林的分类器误判率。

在其中可以看出，误判率最低的是由第 2938 个位点，即 rs2273298 训练出的分类器，其可以达到接近 40% 的误判率，在所有位点中是最低的。

因此可以看出，第 2938 个位点即 rs2273298，是疾病最有可能的致病位点。

而在多位点的分析中，因为 AntEpiSeeker 模型考虑了位点组合的相互影响，因此相比于其他将位点作为独立变量来考虑的算法而言有着更好的效果。而因为位点两两组合的情况一共有 713664200 种可能，因此我们采取了随机采样的方式来进行验证。

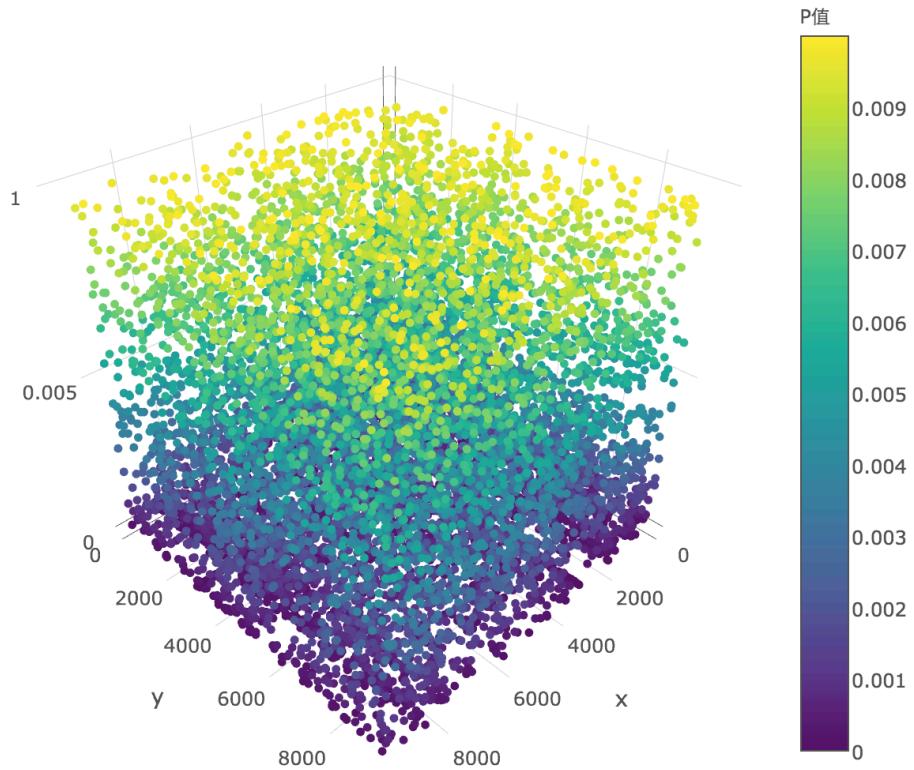


图 3-4 双位点组合 P 值分布

上图是双位点的 AntEpiSeeker 运行结果，其中 X, Y 轴分别是两个位点的 ID，而 Z 轴是 p 值。由数据的随机性可以看出相邻的位点间并没有直接的关联。

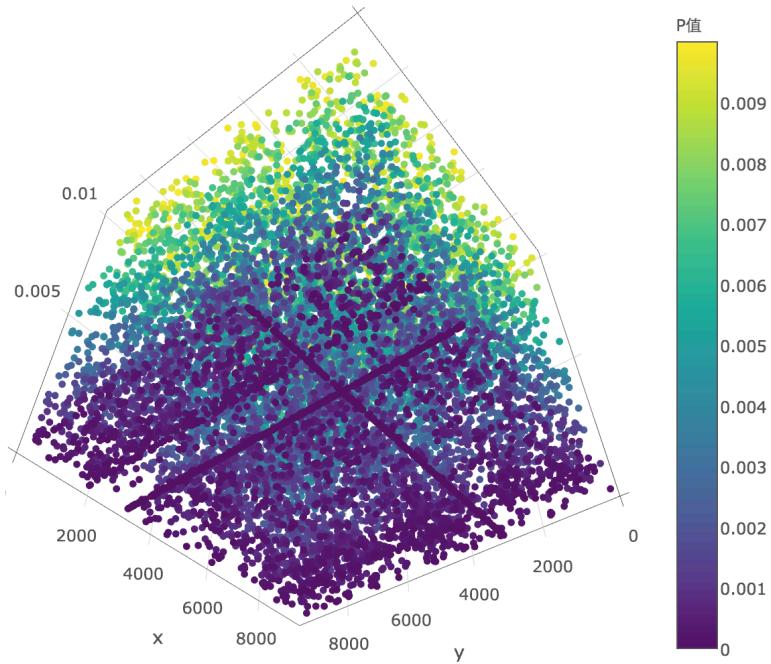


图 3-5 双位点组合 P 值分布

在图中一处值得注意的地方在于，凡是有第 2938 个位点，即 rs2273298 出现的位点集合里，p 值都非常低，这也从另一方面佐证了该位点与疾病有最大的相关性。

3.3 问题三

SNP 标记是 DNA 序列上发生的单个核苷酸碱基之间的变异，人体染色体至少存在 5200 万个 SNP 标记，SNP 标记高密度的优势具有帮助关联分析直接搜索到一个候选基因或者基因片段的可能性。

基因位点是基因在染色体上占有的特定位置。基因位数很多，而染色体的数目较少，因而一条染色体上含有很多基因，位于给定基因位点上相似的 DNA 序列被称为等位基因，基因位点的不同决定每个个体的差异性。

一个基因中会包含多个位点，以题目中给出的 Gene1 和 Gene6 为例，我们可以得到：

Gene1 = { rs3094315, rs3131972, rs3131969, rs1048488, rs12562034, rs12124819, rs4040617};

Gene6 = {rs7290, rs3766180, rs3766178, rs7533, rs7517401, rs3128342, rs7531530, rs3820075, rs3766170, rs3766169, rs2296716, rs9439468, rs6603793, rs7519837, rs6687029, rs6604983, rs4648786};

RS 编码代表了每个基因位点所在的位置，并且不会出现同一个位点出现在两个基因上的情况，因此，我们可以将 gene1 和 gene6 或是其他的任何两个基因看做两个相互独立的集合，每个基因位点是集合中的一个元素，而且任何两个基因集合之间都没有交集，互相独立。

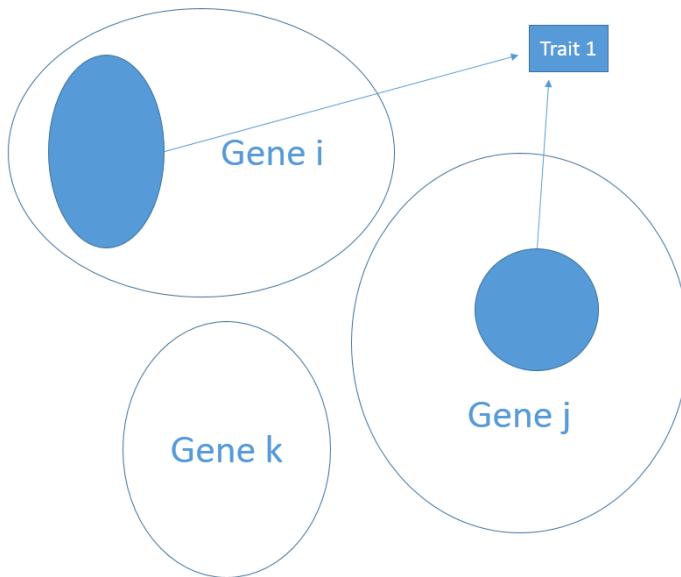


图 3-6 基因和位点关系图

在实际的遗传学问题中，可能调控某一个性状的位点分散在多个基因中，也可能一个基因中有调控多个性状的位点，因此在针对某个性状进行分析时，我们往往采用某基因的子集的形式来表现，例如：

$\text{Trait1} = \{ \text{rs3094315}, \text{rs3131972} \}$ ；则 $\text{Trait1} \subseteq \text{Gene1}$ ，即 Trait1 是 Gene1 的子集。

通过问题二的结果我们可以得到对有无遗传病 A 有显著影响的几个位点，通过搜索 gene_info 文件夹中的 300 个基因对应的位点信息，可以找到有显著影响的位点所在的基因，对于基因的选取我们遵循两个原则：

1. 选取差异性极其显著的位点所在的基因；

2. 选取包含有差异位点数量较多的基因；

针对上述原则 1 我们得到结果：

表格 3-5 结果表

基因 ID	位点 ID	rs 编码	P 值
62	1792	'rs2250358'	P=0.0000712375
102	2938	'rs2273298'	P=0.000000547405

表是问题二结果中显著的 one-way 位点结果中，差异最为显著的两个位点，其所在的基因编码分别是 62 和 102。

针对原则 2，基于问题二 one-way 和 two-way 分析得到的结果，统计得到位点出现频率较高的基因：

表格 3-6 One-Way 结果

基因 ID	位点 ID	rs 编码
55	1541	'rs7368252'
	1593	'rs7522344'
	1599	'rs12036552'
265	8355	'rs4243820'
	8380	'rs7543405'
	8384	'rs556596'
	8390	'rs495223'
293	9164	'rs1201394'
	9196	'rs7555715'

表格 3-7 Two-Way 结果

基因 ID	位点 ID	rs 编码
55	1541	'rs2275741'
	1548	'rs262656'
	1577	'rs4949588'
	1588	'rs1862710'
	1593	'rs7535081'
	1597	'rs10799636'
	1599	'rs11121821'
78	2177	'rs807260'
	2192	'rs2455130'
	2202	'rs4949588'
	2206	'rs10927473'
	2208	'rs11122043'
	2217	'rs35107626'
	2221	'rs2275741'
	2223	'rs2247525'
144	4286	'rs10492939'
	4288	'rs7547618'
	4289	'rs4576609'
	4300	'rs2231863'
	4301	'rs7535816'
	4307	'rs1739822'
	4321	'rs10489156'
265	8355	'rs4576609'
	8356	'rs6673342'

	8357	'rs166604'
	8373	'rs12137141'
	8380	'rs2526830'
	8384	'rs351617'
	8388	'rs28508199'
293	9164	'rs7538876'
	9171	'rs12568609'
	9188	'rs387232'
	9192	'rs10903086'
	9193	'rs10803352'
	9203	'rs387232'
	9211	'rs922114'
	9221	'rs12043302'

分析上述两个表格，可以发现基因编码为 55, 265 和 293 的三个基因中较高频率地出现异常位点。

对以上的基因数据进行综合分析，55, 62, 102, 265, 293 五个基因有可能对遗传病 A 的调控具有显著影响，其中，102 和 265 基因是最有可能的两个基因。

3.4 问题四

SNP标记是一种在全基因组水平上测序得到的高通量数据，高通量测序技术一次可以对几十万到几百万条DNA分析进行碱基序列的检测，由于其高准确性，高通量，高灵敏度的特点，已被广泛应用于基因组学和功能基因组学中。

目前遗传学研究中的复杂性状数据一般包含多个数量性状，这些生物性状往往具有多个不同的环境，而且性状之间呈现较强的相关性，多性状联合分析可以提高统计功效、增加效应估计的准确度和精确度、显著地缩短计算时间和降低研究成本，可以显著地提高对一因多效基因的检测能力和效应估计的准确度。

当今的研究学者已经提出了多种统计分析方法，例如，用多变量回归模型同时分析多个相关的数量性状，采用极大似然比检验影响这些性状的候选位点，但是这种算法的计算量很大，也难以处理同时存在多个数量性状的数据；还有学者提出用半参数统计方法分析多性状数据，即采用广义估计方程拟合多个数量性状的联合分布。此外，还有一些如主成分分析和因子分析等降维方法，也被应用到多性状联合定位的问题中，这些降维方法把原始的多个性状简化为少数几个隐含变量，而后搜索显著的位点。但是这些降维方法都存在一个难点，即如何对隐含变量给出有生物学意义的解释。

3.4.1 相关分析

下图给出的是十个相关性状制成的百分比堆积折线图，通过其中的尖锐的峰值可以直观地推断出这十个性状之间存在显著的一致性。表格是通过 Pearson 相关检验后得到的结果，可以看到这十个性状两两之间都有着很强的相关性。

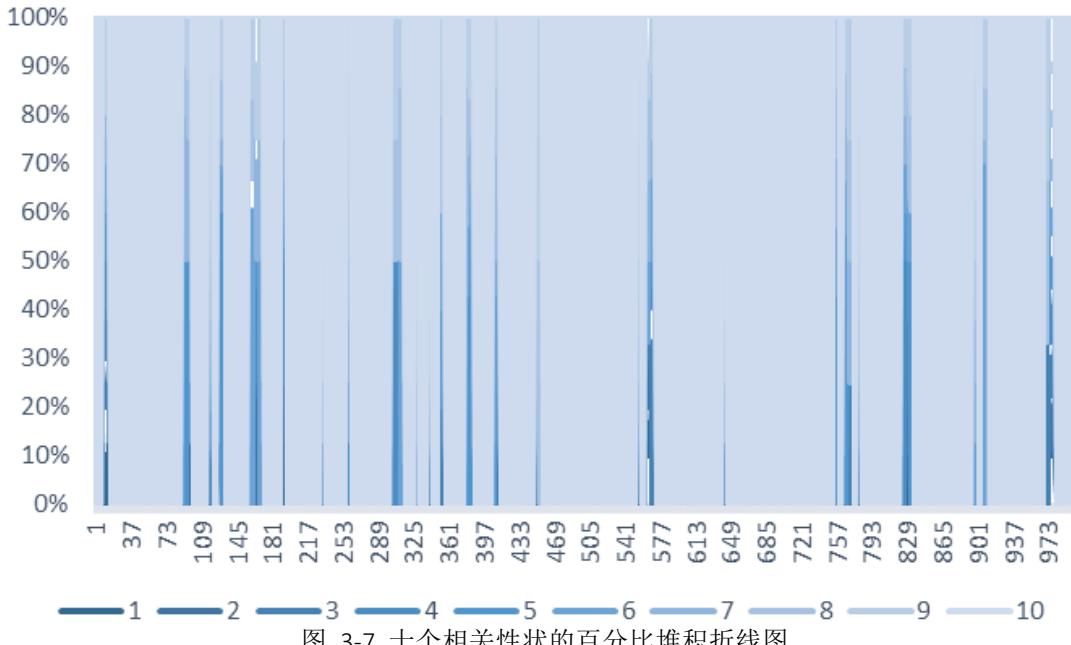


图 3-7 十个相关性状的百分比堆积折线图

	1	2	3	4	5	6	7	8	9	10
1 Pearson 相关性	1	.716	.740	.708	.684	.680	.740	.712	.744	.728
		5.98E-158	4.35E-174	6.16E-153	7.17E-139	1.17E-136	4.35E-174	2.02E-155	5.94E-177	8.39E-166
2 Pearson 相关性		1	.712	.688	.680	.692	.716	.732	.736	.748
			2.02E-155	4.04E-141	1.17E-136	2.09E-143	5.98E-158	1.63E-168	2.82E-171	7.16E-180
3 Pearson 相关性			1	.700	.724	.692	.724	.728	.712	.748
				4.31E-148	3.87E-163	2.09E-143	3.87E-163	8.39E-166	2.02E-155	7.16E-180
4 Pearson 相关性				1	.684	.676	.720	.732	.720	.712
					7.17E-139	1.76E-134	1.60E-160	1.63E-168	1.60E-160	2.02E-155
5 Pearson 相关性					1	.748	.680	.708	.668	.692
						7.16E-180	1.17E-136	6.16E-153	3.17E-130	2.09E-143
6 Pearson 相关性						1	.684	.708	.676	.668
							7.17E-139	6.16E-153	1.76E-134	3.17E-130
7 Pearson 相关性							1	.736	.720	.760
								2.82E-171	1.60E-160	5.72E-189
8 Pearson 相关性								1	.728	.760
									8.39E-166	5.72E-189
9 Pearson 相关性									1	.744
										5.94E-177

图 3-8 结果表

3.4.2 主成分分析

主成分分析 (Principal Component Analysis, PCA)，是一种统计方法，通过正交变换将一组可能存在相关性的变量转换为一组线性不相关的变量，转换后的这组变量叫主成分。本题目中的十个性状是相关性状，互相之间有着一定的相关关系，因此首先通过主成分分析将这十个性状进行降维处理。

下图为通过主成分分析得到的主成分贡献率直方图，可以看出第一主成分的比率高达 73% 左右，第二主成分之后的主成分对结果的贡献率大致相同，在

2%~5%的范围内。因此，我们假设可以用一维数据来代替原题目中的十维数据进行分析，进而简化计算模型。

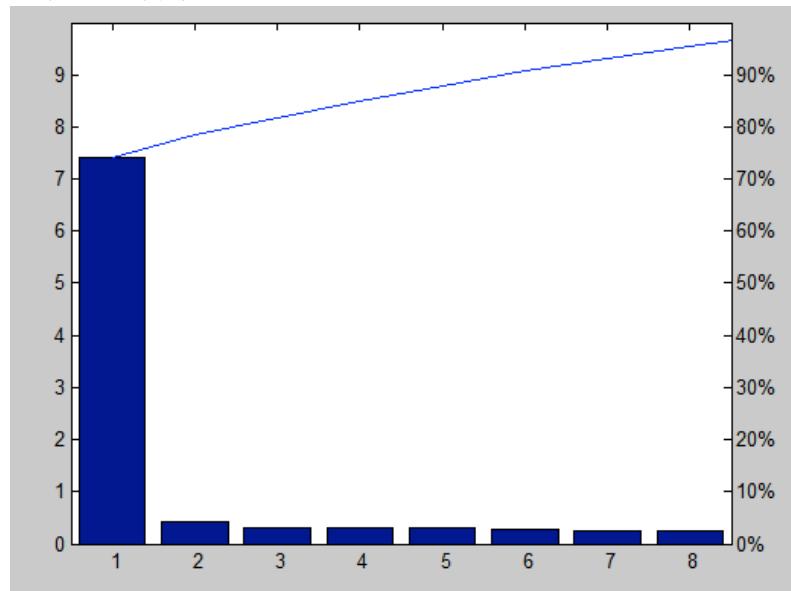


图 3-9 十个相关性状的主成分贡献率直方图

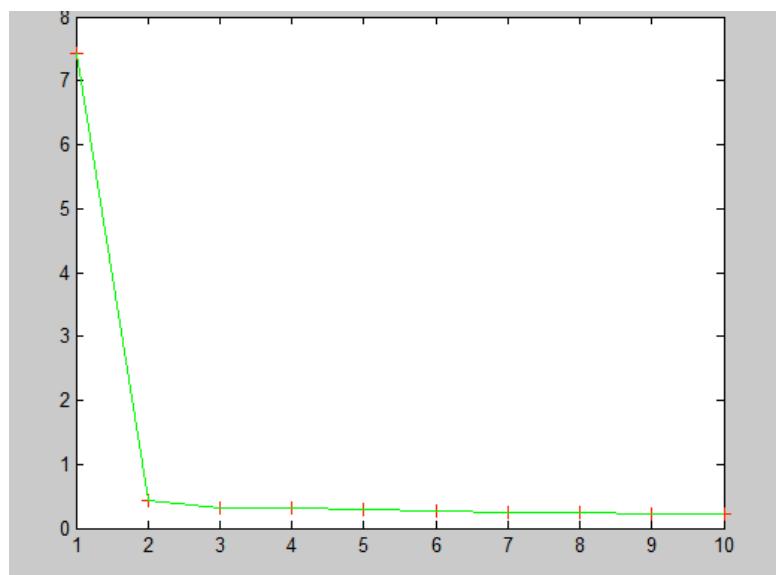


图 3-10 方差贡献山麓图

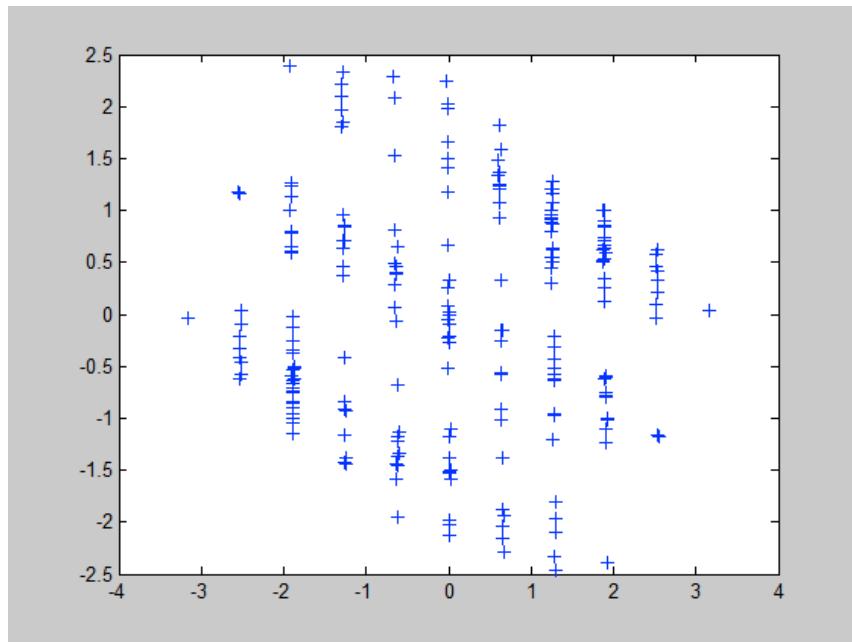


图 3-11 第一主成分和第二主成分的散点分布图

通过上图中所示十个相关性状各自的方差贡献率，可以看到第一个相关性状的方差贡献率最高，验证了第一主成分的贡献率，此外，通过第一主成分和第二主成分的散点分布图，看到没有成云团聚集，说明第二主成分的贡献率可以忽略，因此，我们可以得到简化模型：通过单独分析每一个性状来对十个性状整体的结果进行判断。

3.4.3 问题求解

和第二题类似，我们使用 AntEpiSeeker 模型对十个性状各做了十次计算，得到和性状关联最大的位点集合，参数和第二题一样。

修改 iEpiModel，分别设定为 1 和 2，得到与疾病最有关联的 n 个位点集合。两组解如下所示：

表格 3-8 iEpiModel 为 1 时的结果

性状	位点集合	卡方	p 值
1	1600(rs1775416)	16. 5689	0. 000252409
	1369(rs12754637)	14. 7327	0. 000632165
	5065(rs7538876)	16. 4244	0. 000271329
	1050(rs351617)	21. 7308	1. 91E-05
5	5065(rs7538876)	15. 0343	0. 000543681
	1369(rs12754637)	19. 4978	5. 84E-05
	1339(rs10157835)	17. 9906	0. 00012399
7	8064(rs4360511)	16. 5472	0. 000255162
	4569(rs12746773)	19. 7032	5. 27E-05
	1050(rs351617)	17. 7211	0. 00014188
8	1050(rs351617)	14. 5897	0. 000679041
	5338(rs4920299)	15. 7612	0. 000378014
9	8064(rs4360511)	15. 49	0. 000432907

	1339(rs10157835)	16.3636	0.000279702
	1600(rs1775416)	14.9235	0.00057465
	5338(rs4920299)	14.154	0.000844322
10	4569(rs12746773)	20.0706	4.38E-05
	5338(rs4920299)	13.9207	0.000948763
	1339(rs10157835)	19.982	4.58E-05

表格 3-9 iEpiModel 为 2 时的结果

性状	位点集合	卡方	p 值
1	9300(rs2275741) 4569(rs12746773)	48.9561	6.48E-08
	8064(rs4360511) 2293(rs17031113)	37.8073	8.17E-06
3	8064(rs4360511) 7657(rs10903126)	37.8591	7.99E-06
	5068(rs2526830) 1369(rs12754637)	40.5569	2.52E-06
4	1121(rs922114) 5065(rs7538876)	37.7251	8.46E-06
	9300(rs2275741) 4569(rs12746773)	39.3887	4.16E-06
5	1369(rs12754637) 1165(rs2076883)	38.7281	5.52E-06
	4944(rs2935944) 5065(rs7538876)	45.6027	2.83E-07
6	1050(rs351617) 9440(rs7532525)	43.1828	8.11E-07
	8822(rs4949588) 1050(rs351617)	40.0716	3.11E-06
5	8646(rs2640468) 8113(rs406985)	39.364	4.21E-06
	2319(rs870283) 1369(rs12754637)	39.9497	3.27E-06
6	5065(rs7538876) 3444(rs198411)	42.8117	9.53E-07
	8646(rs2640468) 8113(rs406985)	43.3015	7.71E-07
6	5610(rs17354622) 1369(rs12754637)	46.436	1.96E-07
	2046(rs12043302) 1339(rs10157835)	41.0486	2.04E-06

7	4534(rs1862710) 9097(rs728340)	40.5612	2.52E-06
	4569(rs12746773) 9300(rs2275741)	49.7195	4.63E-08
	4921(rs2057096) 8064(rs4360511)	43.5617	6.88E-07
	8790(rs271376) 5065(rs7538876)	44.1737	5.28E-07
	1050(rs351617) 8038(rs11247925)	40.2902	2.83E-06
8	9300(rs2275741) 4569(rs12746773)	45.7157	2.69E-07
	8822(rs4949588) 1050(rs351617)	40.5809	2.50E-06
	4534(rs1862710) 9097(rs728340)	39.0472	4.82E-06
	5338(rs4920299) 3383(rs11121821)	41.0948	2.00E-06
	1824(rs12240078) 1339(rs10157835)	39.7456	3.57E-06
9	3236(rs1057079) 5338(rs4920299)	38.239	6.80E-06
	2574(rs12123076) 5338(rs4920299)	40.4414	2.65E-06

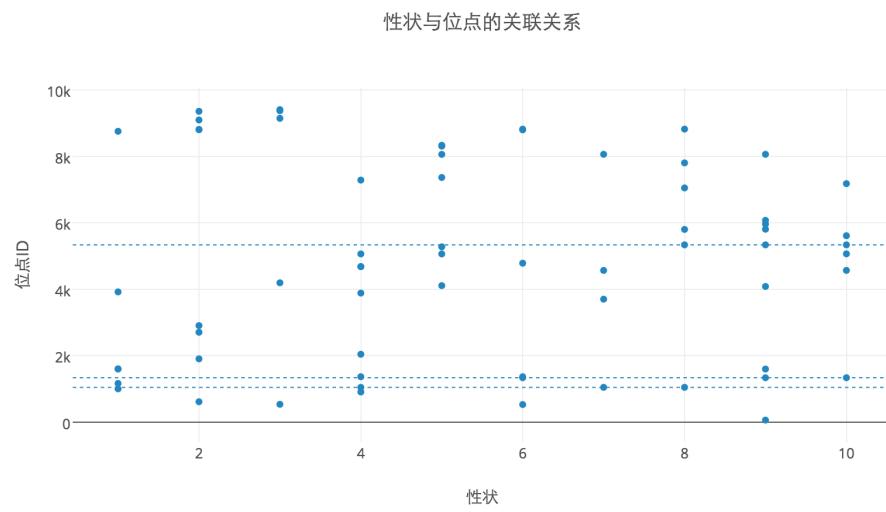


图 3-12 性状与位点的关联关系

由计算结果可知，一般情况下这些位点会同时影响到两个以上的性状，由于十个性状两两之间有很强的相关性，把这十个相关的性状看做一个整体，与其有关联的位点则是与单个性状有关联的位点的集合。从计算结果来看，则是 1600(rs1775416), 1369(rs12754637), 5065(rs7538876), 1050(rs351617),

1339(rs10157835), 8064(rs4360511), 4569(rs12746773), 5338(rs4920299), 位点组合是 9300(rs2275741), 4569(rs12746773), 8064(rs4360511), 2293(rs17031113), 7657(rs10903126), 5068(rs2526830), 1369(rs12754637), 1121(rs922114), 5065(rs7538876), 1369(rs12754637), 1165(rs2076883), 4944(rs2935944), 1050(rs351617), 9440(rs7532525), 8822(rs4949588), 8646(rs2640468), 8113(rs406985), 2319(rs870283), 3444(rs198411), 5610(rs17354622), 2046(rs12043302), 1339(rs10157835), 4534(rs1862710), 9097(rs728340), 4921(rs2057096), 8790(rs271376), 8038(rs11247925), 5338(rs4920299), 3383(rs11121821), 1824(rs12240078), 3236(rs1057079), 2574(rs12123076)。

其中和十个性状最相关的是 1369(rs12754637), 5065(rs7538876), 1050(rs351617), 1339(rs10157835), 8064(rs4360511), 4569(rs12746773), 5338(rs4920299)这几个位点。

4 模型优缺点

本文根据不同的数据集合的特点,选取了最适合的关联分析方法。本文采取的是用 AntEpiSeeker 算法进行求解,用随机森林、t 检验、卡方检验协助检验结果。

AntEpiSeeker 充分考虑了基因的上位性,运用蚁群算法,不但考虑了信息素较高的位点,还考虑了关联性较强的位点集,对边缘效应的鲁棒性更高。但另一方面, AntEpiSeeker 受蚁群算法的参数影响较大,在参数的设置方面,一般需要根据已有的相关研究,归纳后进行推断设置。同时,本文中采用的算法是单线程的,对电脑的配置有比较高的要求,这次由于时间原因没能实现 IEpiModel 大于 3 的情形,在今后的研究分析中,我们会先优化算法,改造成多线程,提高运行效率。

随机森林算法近年来在生物医学和生物信息学领域,特别是在基因表达数据和其他高位数据的分类和回归问题中表现出卓越的性能。随机森林适用于高维小样本数据,能自动实现特征选择功能且对于无关特征不敏感,又能考虑到特征之间的相互作用,对于二分类问题和多分类问题同样适用,不需要复杂的参数选择过程,不容易出现过拟合的情况。然而随机森林在小样本情况下(1k~100k),相对于经典算法(SVM or Boosting)没优势,一般来说效果更差。同时 RF 作为机器学习的一种算法,可以在简化计算步骤方面作进一步优化。

t 检验和卡方检验是通过纯统计学方面的原理,胜在计算简便,易于操作,但是像卡方检验根据统计显著性来判断关联性,容易受罕见基因型的影响。用表 6.1 来举例说明这种影响。

表格 4-1 举例说明罕见基因型对卡方检验的影响

	G1	Gn	
表现型 0	1	N(Gn. 0)	N(0)
表现型 1	0	N(Gn. 1)	N(1)

	1	N(Gn)	N
--	---	-------	-------	---

在表 6.1 中, G1~Gn 代表一个位点集的 n 个基因型, 表现型 0 代表样本没有得病, 表现型 1 代表样本得病, N 为样本总数, N(0)为表现型 0 的样本数, N(1)为表现型 1 的样本数。基因型 G1 非常罕见, 只在一个样本出现过, 并且该样本没有得病, 表现型为 1。则由联合熵的计算公式可得 (17 页), 基因型 G1 表现型 0 对应的卡方统计量为 $\frac{(\frac{N(0)}{N} - 1)^2}{\frac{N(0)}{N}}$ 。由此看出, 如果患病人数和非患病数相差很大的时候, 卡方检验受罕见因素的影响很大。本文中暂时不需要考虑罕见因素的影响。

5 总结与展望

5.1 总结

近年来, 遗传统计学和生物信息学的飞速发展使更多的人投身到探索人体遗传密码的事业中去, 据估计人类基因组中每一千个核苷酸就有一个 SNP, 人类 30 亿碱基对中共有 300 万以上的 SNP, 此类研究由于其本身的特性, 通常需要分析和处理大规模数据的能力, 动辄几千维的数据给分析和处理带来了一定的困难。本文基于全基因组关联分析 (GWAS), 主要采用了 AntEpiSeeker 算法, 辅助以卡方检验、t 检验、random forest 等算法, 解决了一系列层层递进的关联分析问题。

在第一题中, 我们首先对原来的编码方式进行了数值编码转换, 先理清了 SNP 在生物上的性质, 如连锁不平衡性, 次等位基因频率, 边缘效应等, 运用合理的数值来代表原来的碱基对 (由 A,T,C,G 随机组成), 使其在保留生物的一些性质的同时数值化便于对其进行分析和处理。

在第二题中, 我们根据数据的特点, 选用 AntEpiSeeker 算法进行求解。AntEpiSeeker 模型使用了卡方检验作为其评价函数, 通过蚁群算法在可行解空间中搜索, 同时采取了两个阶段的搜索策略来提高蚁群算法在位点的搜索中的效果。因为两轮搜索后的计算结果中可能包含有相同的位点, 所以 AntEpiSeeker 对结果采用了最小化假阳率处理。相比较其他模型, AntEpiSeeker 充分考虑了上位作用, 有对边缘效应鲁棒性高的优点。但它容易受蚁群算法影响, 参数的设置对结果的影响较大, 我们结合了往年学者们的研究结果和经验, 重新合理的设置了参数。此外, 我们用 t 检验、卡方检验, 和随机森林的方法对结果进行检验, t 检验、卡方检验作为纯统计学的方法易于操作, 随机森林考虑了所有可能。

第三个问题是建立在问题二答案基础上的分类模型, 针对数据的分散性, 我们分别找到每个位点所在的基因, 通过分析单个位点的显著程度, 以及基因中显著位点的个数来综合评判基因对性状的调控能力。在问题二中, 有个别位点出现在了多个算法的结果中, 这是我们普遍认为的致病位点, 而此位点所在的基因也有很大可能是致病基因。但形成致病基因的因素肯定不止某一个位点的作用, 其中更复杂的机理还需要进一步深入的探究。

第四问是多因素的关联分析问题, 对十个相关性状进行综合分析, 即把问题

二中的算法应用于十阶的相关数据，针对多维的数据，我们首先采用了 PCA 的方法进行降维，由于十个相关性状之间的相关程度较高，最后可以得到降为一维的主成分数据，为了保证生物学意义，我们并没有使用新生成的一维主成分数据，而是针对每个性状单独进行了计算，并将每个结果统计出来得到最终的结论，能够对结果进行合理的生物学意义的解释。

GWAS 局限于解析单个 SNP 位点对疾病易感性的贡献，而在大部分情况下，复杂疾病的致病位点不止一个，单个位点的效应微小，GWAS 难以解释大部分复杂疾病的遗传特征。本文通过采取 AntEpiSeeker 算法对致病位点的可能组合进行了摸索，得出了一些较有价值的结论。

基于全基因组关联分析，我们拓宽了其适用范围，考虑了更多的因素如特别是基因与基因间的相互作用关系，使得结果更具意义，更符合实际情况。

5.2 展望

在本次分析解题过程中，我们对生物信息学进行了全面的学习了解，利用当前的统计学方法和机器学习，智能启发式算法等对遗传信息进行了计算。在建模求解问题的过程中，由于时间关系，我们没有能对 AntEpiSeeker 算法进行进一步的优化，导致 iEpiModel 设定为 3 以上的结果我们很难得到，在今后的研究过程中，我们希望能将考虑范围扩散到探索 4, 5, 甚至 6 个以上的位点组合与遗传疾病的关联关系。

在基因与位点的映射关系中，利用了集合间的包含关系来分析，基因会收到来自多个位点的共同作用，而且相邻位点之间也存在联系。这些在我们的分析过程中通过假设将其合理建模，在之后运用到实际问题中时需要更加充分地考虑这些因素的影响。

在多性状的关联的分析中，我们将十个相关性状进行了降维处理，可能会造成一定的数据偏差，在之后的研究中，我们会充分考虑性状间的差异性，针对微小差异进行更加详细的分析。

6 参考文献

- [1] 金范莹,张宝荣. 老年痴呆的遗传机制研究进展[J]. 生命科学,2014,01:15-26.
- [2] 方庭正,段蕴铀,欧敏. 慢性阻塞性肺病易感基因相关研究进展[J]. 医学研究生学报,2016,07:764-769.
- [3] 庞裕. 多位点基因关联分析[D].华东师范大学,2012.
- [4] 张鑫,李敏,张学军. 全基因组外显子测序及其应用[J]. 遗传,2011,08:847-856.
- [5] 严卫丽. 复杂疾病全基因组关联研究进展——研究设计和遗传标记[J]. 遗传,2008,04:400-406.
- [6] 高原英. 基于稳定性的多因素 SNP 关联分析算法[D].西安电子科技大学,2012.
- [7] Huang X, Wei X, Sang T, et al. Genome-wide association studies of 14 agronomic traits in rice landraces[J]. Nature genetics, 2010, 42(11): 961-967.
- Burton P R, Clayton D G, Cardon L R, et al. Genome-wide association study of 14,000 cases of seven common diseases and 3,000 shared controls[J]. Nature, 2007, 447(7145): 661-678.
- [8] Newton-Cheh C, Johnson T, Gateva V, et al. Genome-wide association study identifies eight loci associated with blood pressure[J]. Nature genetics, 2009, 41(6): 666-676.
- [9] Li Y, Sidore C, Kang H M, et al. Low-coverage sequencing: implications for design of complex trait association studies[J]. Genome research, 2011.
- [10] 叶道军. SNP 疾病模型的性质研究及检出性能的比较[D].西安电子科技大学,2011.
- [11] Wang Y, Liu X, Rekaya R. AntEpiSeeker2. 0: extending epistasis detection to epistasis-associated pathway inference using ant colony optimization[J]. 2012.

7 附录

7.1 代码

7.1.1 数据格式相关代码

```
#!/usr/local/bin/python3

filename = 'genotype.dat'

class Weidian(object):
    """docstring for weidian"""
    def __init__(self, weidian):
        self.dict = {}
        self.weidian = weidian

    def __str__(self):
        s = "SNPs: " + self.weidian + "\n\t"
        for keys, values in self.dict.items():
            s += ": {0}\n\t".format(values)
        return s

# 是不是病人以及碱基对
class DoubleAndIsPatient(object):
    """docstring for DoubleAndIsPatient"""
    def __init__(self, jianjidui):
        self.patientNum = 0
        self.nonPatientNum = 0
        self.jianjidui = jianjidui
    def __str__(self):
        return "碱基对: " + self.jianjidui + "\n\t\t" + "病人数: "
        " + str(self.patientNum) + "\t 非病人数: " +
        str(self.nonPatientNum)

handle = open(filename, "r")

all_text = handle.read()
lines = all_text.split('\n')
weidians = lines[0].split(' ')

arr = []

for i in range(0, len(weidians) - 1):
    arr.append(Weidian(weidians[i]))
```

```

for i in range(1, len(lines)-1):
    attrs = lines[i].split(' ')
    isPatient = 0
    if i > 501:
        isPatient = 1
    for j in range(0, len(weidians) - 1):
        if attrs[j] not in arr[j].dict:
            arr[j].dict[attrs[j]] =
DoubleAndIsPatient(attrs[j])
        else:
            arr[j].dict[attrs[j]].patientNum += (isPatient ==
1)
            arr[j].dict[attrs[j]].nonPatientNum += (isPatient ==
0)

for i in range(0, len(weidians) - 1):
    w = arr[i]
    for keys,values in w.dict.items():
        if abs(values.patientNum - values.nonPatientNum) > 50:
            print(w)

```

```

#!/usr/local/bin/python3

filename = 'genotype.dat'

outputfilename = 'prepass-genotype.dat'

class Weidian(object):
    """docstring for weidian"""
    def __init__(self, weidian):
        self.arr = []
        self.weidian = weidian

    def __str__(self):
        s = "SNPs: " + self.weidian + "\n\t"
        for e in self.arr:
            s += "\t\t: {0}\n".format(e)
        return s

# 是不是病人以及碱基对
class DoubleAndIsPatient(object):
    """docstring for DoubleAndIsPatient"""
    def __init__(self, jianjidui):
        self.patientNum = 0
        self.nonPatientNum = 0
        self.jianjidui = jianjidui
    def __str__(self):
        return "碱基对: " + self.jianjidui + "\n\t\t" + "病人数:

```

```

"  +  str(self.patientNum)  +  "\t  非 病 人 数 :  "  +
str(self.nonPatientNum)

handle = open(filename, "r")
outputHandle = open(outputfilename, 'w')

all_text = handle.read()
lines = all_text.split('\n')
weidians = lines[0].split(' ')
outputHandle.write(lines[0] + '\n')

arr = []
# 最小字母
weidiansLeast = []

for i in range(0, len(weidians)):
    arr.append(Weidian(weidians[i]))
    weidiansLeast.append('a')

for j in range(0, len(weidians)):
    for i in range(1, len(lines)):
        attrs = lines[i].split(' ')
        if attrs[j][0] != attrs[j][1]:
            if (attrs[j][0] < attrs[j][1]):
                weidiansLeast[j] = attrs[j][0]
            else:
                weidiansLeast[j] = attrs[j][1]
        break

print(weidiansLeast)

for i in range(1, len(lines)):
    attrs = lines[i].split(' ')
    line = ''
    if len(attrs) < 9445:
        continue
    for j in range(0, len(weidians)):
        if attrs[j][0] != attrs[j][1]:
            line += '1 '
        elif attrs[j][0] == weidiansLeast[j]:
            line += '0 '
        else:
            line += '2 '
    line = line[:-1]
    line += '\n'
    outputHandle.write(line)

```

```
#!/usr/local/bin/python3
```

```

filename = 'prepass-for-rt.dat'

outputfilename = 'antdata.txt'

class SNP(object):
    """docstring for weidian"""
    def __init__(self, pheo):
        self.pheo = pheo

    def __str__(self):
        s = "{0} {1} {2} {3} {4} {5}".format(self.familyID,
                                             self.ID, self.pID, self.mID, self.sex, self.isAffected)
        for e in self.arr:
            s += " {0} {1}".format(e[0], e[1])
        s += "\n"
        return s

handle = open(filename, "r")
outputHandle = open(outputfilename, 'w')

all_text = handle.read()
lines = all_text.split('\n')
weidians = lines[0].split(' ')

arr = []

buf = ""
for i in range(1, len(weidians)):
    buf += weidians[i]
    buf += ","
buf += "healthy"
buf += "\n"
outputHandle.write(buf)

for i in range(1, len(lines)):
    attrs = lines[i].split(' ')
    buf = ""
    for j in range(1, len(weidians)):
        buf += attrs[j] + ","
    if i <= 500:
        buf += "0\n"
    else:
        buf += "1\n"
    outputHandle.write(buf)



---


#!/usr/local/bin/python3

filename = 'genotype.dat'

outputfilename = 'prepass-for-rt.dat'

```

```

class Weidian(object):
    """docstring for weidian"""
    def __init__(self, weidian):
        self.arr = []
        self.weidian = weidian

    def __str__(self):
        s = "SNPs: " + self.weidian + "\n\t"
        for e in self.arr:
            s += "\t\t: {0}\n".format(e)
        return s

# 是不是病人以及碱基对
class DoubleAndIsPatient(object):
    """docstring for DoubleAndIsPatient"""
    def __init__(self, jianjidui):
        self.patientNum = 0
        self.nonPatientNum = 0
        self.jianjidui = jianjidui
    def __str__(self):
        return "碱基对: " + self.jianjidui + "\n\t\t" + "病人数: "
    " + str(self.patientNum) + "\t 非病人人数: " +
    str(self.nonPatientNum)

handle = open(filename, "r")
outputHandle = open(outputfilename, 'w')

all_text = handle.read()
lines = all_text.split('\n')
weidiants = lines[0].split(' ')
outputHandle.write("healthy " + lines[0] + '\n')

arr = []
# 最小字母
weidiantsLeast = []

for i in range(0, len(weidiants)):
    arr.append(Weidian(weidiants[i]))
    weidiantsLeast.append('a')

for j in range(0, len(weidiants)):
    for i in range(1, len(lines)):
        attrs = lines[i].split(' ')
        if attrs[j][0] != attrs[j][1]:
            if (attrs[j][0] < attrs[j][1]):
                weidiantsLeast[j] = attrs[j][0]
            else:
                weidiantsLeast[j] = attrs[j][1]
        break

```

```

print(weidiansLeast)

for i in range(1, len(lines)):
    attrs = lines[i].split(' ')
    line = ''
    if i > 501:
        line += '0 '
    else:
        line += '1 '
    if len(attrs) < 9445:
        continue
    for j in range(0, len(weidians)):
        if attrs[j][0] != attrs[j][1]:
            line += '1 '
        elif attrs[j][0] == weidiansLeast[j]:
            line += '0 '
        else:
            line += '2 '
    line = line[:-1]
    line += '\n'
    outputHandle.write(line)

```

```

#!/usr/local/bin/python3

filename = 'genotype.dat'

outputfilename = 'snp.ped'

class Individual(object):
    """docstring for weidian"""
    def __init__(self, i, isAffected):
        self.arr = []
        self.ID = i
        self.familyID = i
        self.isAffected = isAffected
        self.pID = 0
        self.mID = 0
        self.sex = "other"

    def __str__(self):
        s = "{0} {1} {2} {3} {4} {5}".format(self.familyID,
                                             self.ID, self.pID, self.mID, self.sex, self.isAffected)
        for e in self.arr:
            s += " {0} {1}".format(e[0], e[1])
        s += "\n"
        return s

handle = open(filename, "r")
outputHandle = open(outputfilename, 'w')

```

```

all_text = handle.read()
lines = all_text.split('\n')
weidians = lines[0].split(' ')
arr = []

for i in range(1, len(lines)):
    if i <= 500:
        arr.append(Individual(i, 0))
    else:
        arr.append(Individual(i, 1))

for i in range(1, len(lines)):
    attrs = lines[i].split(' ')
    if len(attrs) < 9445:
        continue
    for j in range(0, len(weidians)):
        arr[i - 1].arr.append(attrs[j])

for i in range(0, len(lines) - 1):
    outputHandle.write(str(arr[i]))



---


#!/usr/local/bin/python3

filename = 'genotype.dat'
outputfilename = 'snp.map'

class Weidian(object):
    def __init__(self, weidian):
        self.weidian = weidian
        self.chro = 1
        self.dis = 0
        self.position = 1234555

    def __str__(self):
        return "{0} {1} {2} {3}\n".format(self.chro,
self.weidian, self.dis, self.position)

handle = open(filename, "r")
outputHandle = open(outputfilename, 'w')

all_text = handle.read()
lines = all_text.split('\n')
weidians = lines[0].split(' ')

for i in range(0, len(weidians)):
    outputHandle.write(str(Weidian(weidians[i])))

```

7.1.2 蚁群算法代码

```
#include <gsl/gsl_cdf.h>
#include <ctime>
#include <cstdlib>
#include "model.cpp"
int rnd(int uper)
{
    return (rand()%uper);
}
double rnd(int low, double uper)
{
double p=(rand()/(double)RAND_MAX)*((uper)-(low))+(low);
return (p);
}
int cdf2locus(double x, int start, int end)
{
if(start+1==end)
{
return start;
}
else
{
int temp=(start+end)/2;
//cout<<temp<<endl;
if(SNPdata.cdf[temp]<=x)
{
return cdf2locus(x,temp,end);
}
else
{
return cdf2locus(x,start,temp);
}
}
}
//#include "permute.cpp"
class ant
{
private:
//double* prob;
int m_iLociCount;
//int* AllowedLoci;
public:
int ant_number;
int* tabu;
void initiate();
void start();
void move();
int ChooseNextLocus();
void addlocus(int locus);
```

```

    void destroy();
};

void ant::initiate()
{
tabu=new int[iLociModel+1];
//prob=new double[iLociCount];
//AllowedLoci=new int[iLociCount];
}
void ant::start()
{
m_iLociCount=0;
//int i;
//for(i=0;i<iLociCount;i++)
//{
//AllowedLoci[i]=1;
//prob[i]=0;
//}
}
int ant::ChooseNextLocus()
{
int i,j,k;
j=-1;
//double temp=0;
//int curLocus=tabu[m_iLociCount-1];
//for(i=0;i<iLociCount;i++)
//{
//prob[i]=0;
//if(AllowedLoci[i]==1)
//{
//prob[i]=pow(SNPdata.pheromone[i],alpha);
//temp+=prob[i];
//}
//}
double mRate;
//double mSelect;
//for(i=0;i<2;i++)
//cout<<SNPdata.pheromone[i]<<" "<<SNPdata.cdf[i]<<" ";
while(j== -1)
{
j=0;
srand((unsigned)time(NULL)+rand());
mRate=rnd(0,1);
k=cdf2locus(mRate,0,iLociCount-1);
//cout<<k<<" "<<mRate<<endl;
for(i=0;i<m_iLociCount;i++)
{
if(k==tabu[i])
{
j=-1;
break;
}
}
}
}

```

```

}
//mSelect=0;
//for(i=0;i<iLociCount;i++)
//{
//if(AllowedLoci[i]==1)
//mSelect+=prob[i] ;
//if(mSelect>=mRate)
//{
//j=i;
//break;
//}
//}
}
return k;
}
void ant::addlocus(int locus)
{
tabu[m_iLociCount]=locus;
m_iLociCount++;
//AllowedLoci[locus]=0;
}
void ant::move()
{
int j;
j=ChooseNextLocus();
//cout<<j<<endl;
addlocus(j);
}
void ant::destroy()
{
delete[] tabu;
//delete[] prob;
//delete[] AllowedLoci;
}
class project
{
public:
project();
~project();
ant* ants;
void GetAnt();
void UpdatePheromone();
void StartSearch();
};
project::project()
{
ants=new ant[iAntCount];
int i;
for(i=0;i<iAntCount;i++)
{ants[i].initiate();}
}

```

```

project::~project()
{
delete [] ants;
}
void project::UpdatePheromone()
{
int i,j,k,tag;
for(i=0;i<iLociCount;i++)
SNPdata.pheromone[i]=SNPdata.pheromone[i]*(1-rou);
double eva;
int* locidata;
locidata=new int[iLociModel];
for(i=0;i<iAntCount;i++)
{
for(j=1;j<=iLociModel;j++)
{
locidata[j-1]=ants[i].tabu[j];
}
eva=chi_square(locidata,iLociModel)/100;
for(j=0;j<iLociModel;j++)
{
SNPdata.pheromone[locidata[j]]+=eva;
}
if(eva>eva_TopModel[0])
{
tag=1;
for(j=0;j<iTopModel;j++)
{
if(fabs(eva-eva_TopModel[j])<0.000001)
{
tag=0;
break;
}
}
if(tag)
{
eva_TopModel[0]=eva_TopModel[1];
for(j=0;j<iLociModel;j++)
loci_TopModel[0][j]=loci_TopModel[1][j];
k=1;
while(k<iTopModel && eva>eva_TopModel[k])
{
eva_TopModel[k-1]=eva_TopModel[k];
for(j=0;j<iLociModel;j++)
loci_TopModel[k-1][j]=loci_TopModel[k][j];
k++;
}
eva_TopModel[k-1]=eva;
for(j=0;j<iLociModel;j++)
loci_TopModel[k-1][j]=locidata[j];
}
}

```

```

}
}

//#####
double * prob;
double temp=0;
prob=new double[iLociCount];
for(i=0;i<iLociCount;i++)
{
temp+=pow(SNPdata.pheromone[i],alpha);
prob[i]=temp;
}
for(i=0;i<iLociCount-1;i++)
{
SNPdata.cdf[i+1]=prob[i]/temp;
}
delete [] prob;
delete [] locidata;
}
void project::GetAnt()
{
int i=0;
int locus;
srand((unsigned)time(NULL)+rand());
for (i=0;i<iAntCount;i++)
{
ants[i].start();
locus=rnd(iLociCount);
ants[i].ant_number=i;
ants[i].addlocus(locus);
}
}
void project::StartSearch()
{
int max,i,j;
max=0;
//double temp;
while(max<iItCount)
//while(max<1)
{
cout<<"Iteration: "<<max<<endl;
GetAnt();
for(i=0;i<iAntCount;i++)
{
for(j=1;j<=iLociModel;j++)
{
ants[i].move();
}
}
UpdatePheromone();
//SNPdata.display_pheromone();
max++;
}

```

```

    }
}

void get_toploci()
{
    int i,j,temp1;
    double temp2;
    int* tag;
    tag=new int[iLociCount];
    //int tag[iLociCount];
    double* SortPheromone;
    SortPheromone=new double[iLociCount];
    //double SortPheromone[iLociCount];
    for(i=0;i<iLociCount;i++)
    {
        tag[i]=i;
        SortPheromone[i]=SNPdata.pheromone[i];
    }
    //cout<<"Top ranking pheromones"<<endl;
    for(i=0;i<iTopLoci;i++)
    {
        for(j=i+1;j<iLociCount;j++)
        {
            if(SortPheromone[j]>SortPheromone[i])
            {
                temp1=tag[i];
                tag[i]=tag[j];
                tag[j]=temp1;
                temp2=SortPheromone[i];
                SortPheromone[i]=SortPheromone[j];
                SortPheromone[j]=temp2;
            }
        }
        loci_TopLoci[i]=tag[i];
        phe_TopLoci[i]=SortPheromone[i];
        //cout<<loci_TopLoci[i]<< " "<<phe_TopLoci[i]<<endl;
    }
    delete []tag;
    delete []SortPheromone;
}
char inputfile[200];
char outputfile[200];
double rnd_values[10000];
int ihapsize[2];
int counts[2];
vector<vector< double> > interactions;
vector<vector< double> > mini_interactions;
void postprocessing();
void mini_fp();
void write_result(char* path,int mod);
void loadparameters(char* path);
int main()

```

```

{
    //system("date");
    int i,a;
    iAntCount=1000;
    counts[0]=150;
    counts[1]=300;
    ihapsize[0]=6;
    ihapsize[1]=3;
    alpha=1;
    rou=0.05;
    phe=100;
    iTopModel=1000;
    iTopLoci=200;
    iEpiModel=2;
    pvalue=0.01;
    char* para="parameters.txt";
    loadparameters(para);
    SNPdata.input_data(inputfile);
    //SNPdata.setpheromone(phe);
    iLociCount=SNPdata.iLoci-1;
    char* logpath="AntEpiSeeker.log";
    char* minipath="results_maximized.txt";
    write_result(logpath,0);
    for(a=0;a<2;a++)
    {
        iItCount=counts[a];
        cout<<a+1<<" round search"<<endl;
        iLociModel=ihapsize[a];
        loci_TopModel=new int* [iTopModel];
        for(i=0;i<iTopModel;i++)
            loci_TopModel[i]=new int [iLociModel];
        loci_TopLoci=new int [iTopLoci];
        phe_TopLoci=new double [iTopLoci];
        eva_TopModel=new double [iTopModel];
        for(i=0;i<iTopModel;i++)
            eva_TopModel[i]=0;
    /////////////////////////////////
    //////
        cout<<"---- Initializing parameters----"<<endl;
        cout<<"Number of ants: "<<iAntCount<<endl;
        cout<<"Number of iterations: "<<iItCount<<endl;
        cout<<"Start pheromone level: "<<phe<<endl;
        cout<<"Rou: "<<rou<<endl;
        cout<<"Alpha: "<<alpha<<endl;
        cout<<"Number      of      top      ranking      SNP      sets:
" <<iTopModel<<endl;
        cout<<"Number of top ranking loci: "<<iTopLoci<<endl;
        cout<<"Size of SNP sets: "<<ihapsize[a]<<endl;
        cout<<"Number of SNPs in an epistatic interaction:
" <<iEpiModel<<endl;
    ///////////////////////////////
}

```

```

/////
SNPdata.setpheromone(phe);
project episeeker;
episeeker.StartSearch();
get_toploci();
write_result(logpath,3);
postprocessing();
for(i=0;i<iTopModel;i++)
{
    delete [] loci_TopModel[i];
}
delete [] loci_TopModel;
delete [] loci_TopLoci;
delete [] phe_TopLoci;
delete [] eva_TopModel;

}
mini_fp();
/////////////////////////////permutation///////////////////////////////
///
// cout<<"Permuting genotypes:\n";
// for(i=0;i<10000;i++)
// {
// if(i%500==0)
// cout<<i<<" interactions"<<endl;
// rnd_values[i]=random_chi();
// }
///////////////////////////////
///
write_result(minipath,4);
write_result(outputfile,1);
SNPdata.destroy();
return 1;
}
void mini_fp()
{
int i,j,k,l,find,hit,first_pos;
if(interactions.size()>0)
{
///////////////////////////////
mini_interactions.push_back(interactions[0]);
for(i=1;i<interactions.size();i++)
{
    find=0;
    for(j=0;j<mini_interactions.size();j++)
    {

        hit=0;
        for(k=0;k<iEpiModel;k++)
        {

```

```

for(l=0;l<iEpiModel;l++)
{
if(interactions[i][k]==mini_interactions[j][l])
{
hit=1;
find++;
break;
}
}
if(hit==1)
break;
}
if(find==1 && hit==1)
{
if(interactions[i][iEpiModel]<=mini_interactions[j][iEpiModel])
{
break;
}
else
{
for(k=0;k<iEpiModel+2;k++)
{
mini_interactions[j][k]=interactions[i][k];
}
first_pos=j;
}
}
if(find>1 && hit==1)
{
if(mini_interactions[j][iEpiModel]>mini_interactions[first_pos][iEpiModel])
{
mini_interactions.erase(mini_interactions.begin()+first_pos);
break;
}
else
{
mini_interactions.erase(mini_interactions.begin()+j);
break;
}
}
}
if(find==0)
{
mini_interactions.push_back(interactions[i]);
//cout<<mini_interactions.size()<<endl;
}
}
///////////

```

```

/////
}
}
void loadparameters(char* path)
{
//cout<<"hello"<<endl;
    FILE *f = fopen(path, "r");
    if(f == NULL)
        printf("Cannot open the parameter file
\"%s\"\n", path);
        return;
    }
    int i;
    char tmp[1000];
    while(fgets(tmp, 1000, f) != NULL)
    {
        char str[1000];
        sprintf(str, "%s", tmp);
        str[13]=0;
        if(strcmp(str, "iItCountLarge") == 0)
        {
            for(i = 13; i < (int)strlen(tmp); i++)
                if(tmp[i] >= 48 && tmp[i] < 58)
                    break;
            counts[0]=atoi(&tmp[i]);
        }
        if(strcmp(str, "iItCountSmall") == 0)
        {
            for(i = 13; i < (int)strlen(tmp); i++)
                if(tmp[i] >= 48 && tmp[i] < 58)
                    break;
            counts[1]=atoi(&tmp[i]);
        }
        str[12]=0;
        if(strcmp(str, "largesetsize") == 0)
        {
            for(i = 12; i < (int)strlen(tmp); i++)
                if(tmp[i] >= 48 && tmp[i] < 58)
                    break;
            ihapsize[0]=atoi(&tmp[i]);
        }
        if(strcmp(str, "smallsetsize") == 0)
        {
            for(i = 12; i < (int)strlen(tmp); i++)
                if(tmp[i] >= 48 && tmp[i] < 58)
                    break;
            ihapsize[1]=atoi(&tmp[i]);
        }
        str[9] = 0;
        if(strcmp(str, "iTopModel") == 0)
        {      for(i = 9; i < (int)strlen(tmp); i++)

```

```

                if(tmp[i] >= 48 && tmp[i] < 58)
                        break;
                iTopModel=atoi(&tmp[i]);
            }
        else if(strcmp(str, "iEpiModel") == 0)
        {   for(i = 9; i < (int)strlen(tmp); i++)
            if(tmp[i] >= 48 && tmp[i] < 58)
                break;
            iEpiModel=atoi(&tmp[i]);
        }
        else if(strcmp(str, "iAntCount") == 0)
        {   for(i = 9; i < (int)strlen(tmp); i++)
            if(tmp[i] >= 48 && tmp[i] < 58)
                break;
            iAntCount=atoi(&tmp[i]);
        }
        str[8]=0;
        if(strcmp(str, "iTopLoci") == 0)
        {   for(i = 8; i < (int)strlen(tmp); i++)
            if(tmp[i] >= 48 && tmp[i] < 58)
                break;
            iTopLoci=atoi(&tmp[i]);
        }
        str[7] = 0;
        if(strcmp(str, "INPFILE") == 0)
        {
            for(i = 7; i < (int)strlen(tmp); i++)
                if(tmp[i] != ' ' && tmp[i] != '\t')
                    break;
                int j;
                for(j = i + 1; j < (int)strlen(tmp); j++)
                    if(tmp[j] == ' ' || tmp[j] == '\t' ||
tmp[j] == '"' || tmp[j]=='\n')
                        break;
                    tmp[j] = 0;
                    if(tmp[i] == '"') sprintf(inputfile, "%s",
&tmp[i + 1]);
                        else sprintf(inputfile, "%s", &tmp[i]);
                }
            else if(strcmp(str, "OUTFILE") == 0)
            {
                for(i = 7; i < (int)strlen(tmp); i++)
if(tmp[i] != ' ' && tmp[i] != '\t')
                    break;
                int j;
                for(j = i + 1; j < (int)strlen(tmp); j++)
                    if(tmp[j] == ' ' || tmp[j] == '\t' || tmp[j]
== '"' || tmp[j]=='\n')
                        break;
                    tmp[j] = 0;
                    if(tmp[i] == '"') sprintf(outputfile, "%s",

```

```

&tmp[i + 1]);
else sprintf(outputfile, "%s",
&tmp[i]);
}
str[6]=0;
if(strcmp(str, "pvalue") == 0)
{ for(i = 6; i < (int)strlen(tmp); i++)
    if(tmp[i] >= 48 && tmp[i] < 58)
        break;
pvalue = atof(&tmp[i]);
}

str[5] = 0;
if(strcmp(str, "alpha") == 0)
{
    for(i = 5; i < (int)strlen(tmp); i++)
        if(tmp[i] >= 48 && tmp[i] < 58)
            break;
alpha=atof(&tmp[i]);
}
str[3]=0;
if(strcmp(str, "rou") == 0)
{
    for(i = 3; i < (int)strlen(tmp); i++)
        if(tmp[i] >= 48 && tmp[i] < 58)
            break;
rou = atof(&tmp[i]);
}
else if(strcmp(str, "tau") == 0)
{
    for(i = 3; i < (int)strlen(tmp); i++)
        if(tmp[i] >= 48 && tmp[i] < 58)
            break;
phe = atof(&tmp[i]);
}
}
fclose(f);

}
int **throughout(int m,int n,int** result)
{
int num,mi,ni,i,j,temp;
num=0;
mi=m-1;
ni=n-1;
int* comb=new int[n];
for(i=0;i<=ni;i++)
{
comb[i]=i;
}
temp=0;
while(1)
{

```

```

temp++;
while (comb[ni]==mi)
{
mi--;
ni--;
if (ni===-1)
{
break;
}
}
if (ni===-1)
{
break;
}
if (ni<n-1)
{
comb[ni]++;
for (j=ni+1;j<n;j++)
{
comb[j]=comb[j-1]+1;
}
mi=m-1;
ni=n-1;
}
while (comb[ni]<m)
{
for (i=0;i<n;i++)
{
result[num][i]=comb[i];
}
num++;
comb[ni]++;
}
comb[ni]--;
}
delete comb;
return result;
}
long int comb_num(int m,int n)
{
long int i,p,q;
p=1;
q=1;
for (i=1;i<=n;i++)
{
p=p*i;
q=q*(m-i+1);
}
return q/p;
}
void postprocessing()

```

```

{
long int i,num;
int j,k,l,s,isNew;
double eva,p_value;
int* loci;
loci=new int[iEpiModel];
vector<double> record;
record.resize(iEpiModel+2);
cout<<"Post-processing"<<endl;
cout<<"Dealing with top ranking SNP sets"<<endl;
num=comb_num(iLociModel,iEpiModel);
cout<<"Number of interactions evaluated:
"<<num*iTopModel<<endl;
int** combination;
combination=new int*[num];
for(i=0;i<num;i++)
combination[i]=new int[iEpiModel];
combination=throughout(iLociModel,iEpiModel,combination);
for(i=iTopModel-1;i>=0;i--)
{
for(j=0;j<num;j++)
{
for(k=0;k<iEpiModel;k++)
{
loci[k]=loci_TopModel[i][combination[j][k]];
}
eva=chi_square(loci,iEpiModel);
p_value=1-gsl_cdf_chisq_P(eva,pow(3,iEpiModel)-1);
if(p_value<pvalue)
{
isNew=1;
for(l=0;l<interactions.size();l++)
{
if(fabs(eva-interactions[l][iEpiModel])<0.00000001)
{
isNew=0;
break;
}
}
if(isNew)
{
for(s=0;s<iEpiModel;s++)
{
record[s]=(double)loci[s];
}
record[iEpiModel]=eva;
record[iEpiModel+1]=p_value;
interactions.push_back(record);
}
}
}
}
}

```

```

}

for(i=0;i<num;i++)
delete []combination[i];
delete []combination;
cout<<"Dealing with top ranking loci"<<endl;
num=comb_num(iTopLoci,iEpiModel);
//cout<<"hello"<<endl;
int** combination1;
combination1=new int*[num];
for(i=0;i<num;i++)
combination1[i]=new int[iEpiModel];
combination1=throughout(iTopLoci,iEpiModel,combination1);
cout<<"Number of interactions evaluated: "<<num<<endl;
for(i=0;i<num;i++)
{
for(j=0;j<iEpiModel;j++)
{
loci[j]=loci_TopLoci[combination1[i][j]];
//cout<<loci[j]<<" ";
}
//cout<<endl;
eva=chi_square(loci,iEpiModel);
p_value=1-gsl_cdf_chisq_P(eva,pow(3,iEpiModel)-1);
if(p_value<pvalue)
{
isNew=1;
for(l=0;l<interactions.size();l++)
{
if(fabs(eva-interactions[l][iEpiModel])<0.00000001)
{
isNew=0;
break;
}
}
if(isNew)
{
for(s=0;s<iEpiModel;s++)
{
record[s]=(double)loci[s];
}
record[iEpiModel]=eva;
record[iEpiModel+1]=p_value;
interactions.push_back(record);
}
}
}
for(i=0;i<num;i++)
delete []combination1[i];
delete []combination1;
}
void write_result(char* path,int mod)

```

```

{
int i,j;
std::ofstream result;
if(mod==0)
{
result.open(path,ios::out);
}
if(mod==1)
{
result.open(path,ios::out);
result<<"Epistatic interactions:"<<endl;
result<<"Loci\tChi-square\tP value"<<endl;
for(i=0;i<mini_interactions.size();i++)
{
//////////////////assigning p values/////////////
for(j=0;j<iEpiModel;j++)
{

result<<mini_interactions[i][j]<<" ("<<SNPdata.SNPnames[in
t(mini_interactions[i][j])]<<") "<<" ";
}

result<<"\t"<<mini_interactions[i][iEpiModel]<<"\t"<<mini
_interactions[i][iEpiModel+1]<<endl;
}
}
if(mod==3)
{
result.open(path,ios::app);
result<<"#####Intermediate
results#####"\><endl;
result<<"Top ranking haplotypes"<<endl;
for(i=iTopModel-1;i>=0;i--)
{
//result<<eva_TopModel[i]<<" ";
for(j=0;j<iLociModel;j++)
result<<loci_TopModel[i][j]<<" ";
result<<eva_TopModel[i]<<endl;
}
result<<"Top ranking loci"<<endl;
for(i=0;i<iTopLoci;i++)
{
result<<loci_TopLoci[i]<<" "<<phe_TopLoci[i]<<endl;
}
}
if(mod==4)
{
result.open(path,ios::out);
result<<"Epistatic interactions:"<<endl;
result<<"Loci\tChi-square\tP value"<<endl;
for(i=0;i<interactions.size();i++)
}

```

```

    {
        for(j=0;j<iEpiModel;j++)
        {

result<<interactions[i][j]<<" ("<<SNPdata.SNPnames[int(int
eractions[i][j])]<<") "<<" ";
        }

result<<"\t"<<interactions[i][iEpiModel]<<"\t"<<interacti
ons[i][iEpiModel+1]<<endl;
        }

    }
result.close();
}



---


#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <math.h>
#include <vector>
#include <algorithm>
#include <string>
using namespace std;
int iAntCount;
int iItCount;
int iLociCount;
int iLociModel;
int iEpiModel;
double alpha;
double rou;
double phe;
double pvalue;
int iTopModel;
int** loci_TopModel;
int iTopLoci;
int* loci_TopLoci;
double* eva_TopModel;
double* phe_TopLoci;
vector<vector<double> > sigepi;
int lcn;
class SNP
{
public:
    int iSample;
    int iLoci;
    bool** data;
    char** SNPnames;
    int classvalues[2];
    void input_data(char* path);

```

```

        void destroy();
        void setpheromone(double level);
        double* pheromone;
        double* cdf;
        void display_pheromone();
    };
void SNP::destroy()
{
int i;
for(i=0;i<iLoci;i++)
delete []SNPnames[i];
delete []SNPnames;
for(i=0;i<iSample;i++)
{
delete []data[i];
}
delete []data;
delete []pheromone;
delete []cdf;
}
void SNP::setpheromone(double level)
{
int i;
cdf[0]=0;
for(i=0;i<iLoci-1;i++)
{
pheromone[i]=level;
cdf[i+1]=double(i+1)/double(iLoci-1);
}
}
void SNP::display_pheromone()
{
int i;
for(i=10;i<111;i++)
{
std::cout<<setprecision(5)<<pheromone[i];
cout<<" ";
}
cout<<endl;
}
void SNP::input_data(char* path)
{
    cout<<"----Reading data----"<<endl;
    int i,j,temp;
    string line;
    ifstream in(path);
    getline(in,line);
    istringstream test(line);
    i=0;
    string word;
    while(!test.eof())

```

```

{
    getline(test,word,',');
    i++;
}
iLoci=i;
//cout<<SNPdata.iLoci<<endl;
j=0;
while(!in.eof())
{
    getline(in,line);
        //cout<<line<<endl;
    j++;
}
iSample=j-1;
//cout<<iSample<<endl;
in.close();
pheromone=new double[iLoci-1];
cdf=new double[iLoci];
SNPnames=new char*[iLoci];
for(i=0;i<iLoci;i++)
    SNPnames[i]=new char[20];
data=new bool*[iSample];
for(i=0;i<iSample;i++)
    data[i]=new bool [2*iLoci];
    classvalues[0]=0;
    classvalues[1]=0;
ifstream in1(path);
getline(in1,line);
istringstream test1(line);
i=0;
while(!test1.eof())
{
    if(i==iLoci)
        break;
    getline(test1,word,',');
    strcpy(SNPnames[i],word.c_str());
    i++;
}
i=0;
while(!in1.eof())
{
    if(i==iSample)
    {
        break;
    }
    getline(in1,line);
    istringstream values(line);
    j=0;
    while(!values.eof())
    {
        getline(values,word,',');

```

```

        istringstream int_iss(word);
        int_iss>>temp;
        if(j==2*iLoci-2)
        {
            classvalues[temp]++;
        }
        if(temp==0)
        {
            data[i][j++]=0;
            data[i][j++]=0;
        }
        else if(temp==1)
        {
            data[i][j++]=0;
            data[i][j++]=1;
        }
        else if(temp==2)
        {
            data[i][j++]=1;
            data[i][j++]=0;
        }
        else
        {
            data[i][j++]=1;
            data[i][j++]=1;
        }
    }
    i++;
}
in1.close();
cout<<"----Reading data completed!----<<endl;
cout<<"Number of loci: "<<iLoci-1<<endl;
cout<<"Number of samples: "<<iSample<<endl;
}
SNP SNPdata;



---


#include "basicalg.cpp"
double chi_square(int* selectedSNPSet,int k)
{
//int k = sizeof(selectedSNPSet)/sizeof(selectedSNPSet[0]);
int comb = (int)pow(3.0, k);
double** observedValues;
double* colSumTable;
double** expectedValues;
int i,j,index;
observedValues=new double*[2];
expectedValues=new double*[2];
for(i=0;i<2;i++)
{
observedValues[i]=new double[comb];
expectedValues[i]=new double[comb];

```

```

}

colSumTable=new double[comb];
for(i=0;i<comb;i++)
{
observedValues[0][i] = 0;
observedValues[1][i] = 0;
colSumTable[i] = 0;
}
/*constructing observed freq table*/
bool cont;
int val;
for(i=0;i<SNPdata.iSample;i++)
{
index = 0;
cont = 1;
for(j=0;j<k;j++) {
if(SNPdata.data[i][2*selectedSNPSet[j]] &&
SNPdata.data[i][2*selectedSNPSet[j]+1])
{
cont = 0;
break;
}
else{
val = 0;
if(!SNPdata.data[i][2*selectedSNPSet[j]] &&
SNPdata.data[i][2*selectedSNPSet[j]+1])
{
val =1;
}
if(SNPdata.data[i][2*selectedSNPSet[j]] &&
!SNPdata.data[i][2*selectedSNPSet[j]+1]){
val =2;
}
index = index + val*(int)pow(3.0, (k-1-j));
}
}
if(cont){
if(!SNPdata.data[i][2*(SNPdata.iLoci-1)] &&
!SNPdata.data[i][2*SNPdata.iLoci-1])
{
observedValues[0][index]++;
}
else{
observedValues[1][index]++;
}
colSumTable[index]++;
}
}
/*computing expected freq values and compute chi-square
value*/
double x2 = 0;

```

```

for(i=0;i<comb;i++) {
expectedValues[0][i] = colSumTable[i]*SNPdata.classvalues[0]/(double)SNPdata.iSample;
expectedValues[1][i] = colSumTable[i]*SNPdata.classvalues[1]/(double)SNPdata.iSample;
if(expectedValues[0][i]!=0) {
x2 = (expectedValues[0][i]-observedValues[0][i])*(expectedValues[0][i]-observedValues[0][i])/expectedValues[0][i];
}
if(expectedValues[1][i]!=0) {
x2 = (expectedValues[1][i]-observedValues[1][i])*(expectedValues[1][i]-observedValues[1][i])/expectedValues[1][i];
}
}
for(i=0;i<2;i++)
{
delete [] expectedValues[i];
delete [] observedValues[i];
}
delete [] expectedValues;
delete [] observedValues;
delete [] colSumTable;
return x2;
}

```

7.1.3 分析与作图代码

```

library(randomForest)
rtdata <- read.table("prepass-for-rt.dat", header = TRUE)
rtdata[, 'healthy'] <- as.factor(rtdata[, 'healthy'])

# 42.4
# realdata <- rtdata[, c(1, 758, 963, 1196, 1542, 2939, 3011,
# 4933, 5938, 7738, 8381)]
# 39
#realdata <- rtdata[, c(1, 758, 1196, 1542, 2939, 4933, 5938,
# 7738, 8381)]
# 38
#realdata <- rtdata[, c(1, 758, 963, 1196, 1542, 2939, 3011)]

realdata <- rtdata[, c(1, 2939)]
snp.rf <- randomForest(healthy ~ ., data=realdata,
importance=TRUE, proximity=TRUE)
snp.rf

library(plotly)

```

```

mydata <- read.table("plink.assoc.logistic", header = TRUE)

filteredData = subset(mydata, mydata[, "P"] < 0.01)

p <- plot_ly(filteredData, type = "scattergl", x = c(1:9445),
y = P,
    mode = "markers", marker = list(size = 10)
)

layout(p,                                # all of layout's properties:
/r/reference/#layout
    title = "P 值小于 0.01 的数据点", # layout's title:
/r/reference/#layout-title
    xaxis = list(                  # layout's xaxis is a named list.
List of valid keys: /r/reference/#layout-xaxis
        title = "位 点 ",          # xaxis's title:
/r/reference/#layout-xaxis-title
        showgrid = F              # xaxis's showgrid:
/r/reference/#layout-xaxis-showgrid
    ),
    yaxis = list(                  # layout's yaxis is a named list.
List of valid keys: /r/reference/#layout-yaxis
        title = "P 值 "           # yaxis's title:
/r/reference/#layout-yaxis-title
    )
)


---


library(plotly)
mydata <- read.table("multi-1.txt")

p <- plot_ly(mydata, type = "scatter", x = v1, y = v2,
    mode = "markers", marker = list(size = 1.6)
)


---



```