具有遗传性疾病和性状的遗传位点分析

摘要：

人体的每条染色体携带一个 DNA 分子，且人的遗传密码由人体中的 DNA 分子携带。DNA 是由分别带有 A, T, C, G 四种碱基的脱氧核苷酸链接组成的双螺旋长链分子。在这条双螺旋的长链中，共有约 30 亿个碱基对，而基因则是 DNA 长链中有遗传效应的一些片段。在组成 DNA 的数量浩瀚的碱基对（或对应的脱氧核苷酸）中，有一些特定位置的单个核苷酸经常发生变异引起 DNA 的多态性，我们称之为位点(Single Nucleotide Polymorphism，SNPs)。染色体、基因和位点的结构关系见图 1。

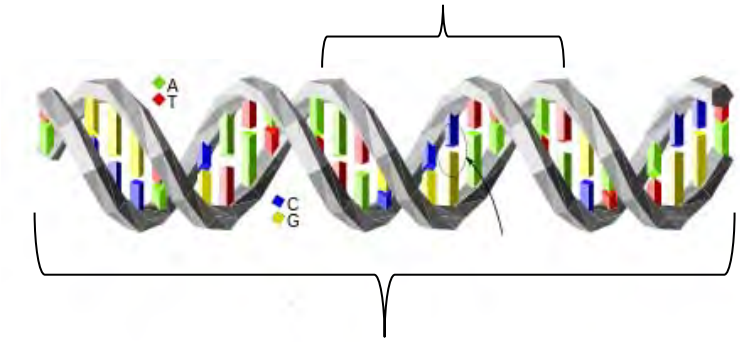

图 1. 染色体、基因和位点的结构关系。

大量研究表明，人体的许多表型性状差异以及对药物和疾病的易感性等都可能与某些位点相关联，或和包含有多个位点的基因相关联。因此，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，能帮助研究人员了解性状和一些疾病的遗传机理，也能使人们对致病位点加以干预，防止一些遗传病的发生。

近年来，研究人员大都采用全基因组关联性分析的方法来确定致病位点或致病基因。全基因组关联性分析(Genome-Wide Association Study，GWAS)，简单的讲，就是从人类全基因组范围内的序列变异(单核苷酸多态，SNP)中，筛选出那些与疾病性状关联的 SNPs。此外，通过应用统计学原理，研究人员通过对样本的健康状况和位点编码的统计，对比分析来确定致病位点，从而发现遗传病或性状的遗传机理。

本题针对某遗传病提供的 1000 个样本信息数据，每个样本信息包含的 9445 个位点编码信息以及包含这些位点的基因信息数据。通过应用统计学，矩阵降维处理及数据挖掘的相关方法，来对本题所包含的四个问题进行研究以及对所得出的具体结果数据进行合理性分析。具体问题如下：

问题一：将 genotype.dat 中每个位点的碱基(A,T,C,G) 编码方式转化成数值编码方式，便于进行数据分析。

问题二：设计或采用一种方法，依据相关的理论依据，通过对附录中提供的 1000 个样本中的可能致病的染色体片段上的 9445 个位点的编码信息进行筛选，判断，找出某种疾病最有可能的一个或几个致病位点。

问题三：同题二中的遗传疾病 A 的样本信息（phenotype.txt 文件）。现有 300 个基因，每个基因所包含的位点名称见文件夹 gene_info 中的 300 个 dat 文件，每个 dat 文件列出了对应基因所包含的位点(位点信息见文件 genotype.dat)。由于可以把基因理解为若干个位点组成的集合，遗传疾病与基因的关联性可以由基因中包含的位点的全集或其子集合表现出来请找出与疾病最有可能相关的一个或几个基因，并说明理由。

问题四：依据题二中的样本中的 9445 个位点的编码信息（见 genotype.dat 文件）。通过将相关的性状或疾病看成一个整体，然后来探寻与它们相关的位点或基因。依据此原理并根据 multi_phenos.txt 文件给出的 1000 个样本的 10 个相关联性状的信息及其 9445 个位点的编码信息(见 genotype.dat)，找出与 multi_phenos.txt 中 10 个性状有关联的位点。

关键词：遗传统计学， 全基因组关联性分析 (GWAS)， 位点 (SNPs)

目录

1. 问题重述

1.1 问题背景

在 DNA 长链中，位点个数约为碱基对个数的 1/1000。由于位点在 DNA 长链中出现频繁，多态性丰富，近年来成为人们研究 DNA 遗传信息的重要载体，被称为人类研究遗传学的第三类遗传标记。大量研究表明，人体的许多表型性状差异以及对药物和疾病的易感性等都可能与某些位点相关联，或和包含有多个位点的基因相关联。因此，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，能帮助研究人员了解性状和一些疾病的遗传机理，也能使人们对致病位点加以干预，防止一些遗传病的发生。

近年来，研究人员大都采用全基因组的方法来确定致病位点或致病基因，具体做法是：招募大量志愿者（样本），包括具有某种遗传病的人和健康的人，通常用 1 表示病人，0 表示健康者。对每个样本，采用碱基(A,T,C,G)的编码方式来获取每个位点的信息(因为染色体具有双螺旋结构，所以用两个碱基的组合表示一个位点的信息)；此外，研究人员可以通过对样本的健康状况和位点编码的对比分析来确定致病位点，从而发现遗传病或性状的遗传机理。

另外，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。科研人员往往把相关的性状或疾病放在一起研究，这样能提高发现致病位点或基因的能力。

1.2 待解决问题

(1) 为了便于进行数据分析，利用适当的方法，把 genotype.dat 中每个位点的碱基(A,T,C,G)编码方式转化成数值编码方式。

(2) 根据附录中 1000 个样本在某条有可能致病的染色体片段上的 9445 个位点的编码信息(见 genotype.dat)和样本患有遗传疾病 A 的信息（见 phenotype.txt 文件）。设计或采用一个方法，找出某种疾病最有可能的一个或几个致病位点，并给出相关的理论依据。

(3) 同问题二中的样本患有遗传疾病 A 的信息（phenotype.txt 文件）。现有 300 个基因（具体信息见 gene_info 中的 300 个 dat 文件）。由于可以把基因理解为若干个位点组成的集合，遗传疾病与基因的关联性可以由基因中包含的位点的全集或其子集合表现出来，请找出与疾病最有可能相关的一个或几个基因，并说明理由。

(4) 在问题二中，已知 9445 个位点，其编码信息见 genotype.dat 文件。在实际的研究中，科研人员往往把相关的性状或疾病看成一个整体，然后来探寻与它们相关的位点或基因。试根据 multi_phenos.txt 文件给出的 1000 个样本的 10 个相关联性状的信息及其 9445 个位点的编码信息(见 genotype.dat)，找出与 multi_phenos.txt 中 10 个性状有关联的位点。

2. 符号说明

符号	说明
$S^{'}$	表示 500 个患遗传疾病 A 的样本中对应于 155 个位点类型所构造的二维矩阵。
P	155 个位点组成的向量。
S	$S^{'}$ 中元素与 P 中元素作‘与’运算后，所构造出的二维矩阵。
N	表示问题三种给出的 300 个样本基因。
Q	为采用 K-means 聚类算法后，产生的类。
Num_N	为 N 中每个样本基因片段中所包含的所有位点的数量。

3. 问题一

3.1 问题分析

由 1.2 (1)可知，问题 1 要求利用适当方法，将 genotype.dat 中每个位点的碱基(A,T,C,G)编码方式转化成数值编码方式，从而便于进行数据分析。

在 genotype.dat 文件中，共有 1000 个样本信息的染色体片段，其中每个染色体片段包含了 9445 个位点的编码信息，且每一个位点则是由 A,T,C,G 组成的碱基对。通过排列组合，我们容易得出位点的所有类型为 16 ($A_4^1 \times A_4^1$)类。同时，利用文本文档对 genotype.dat 进行位点类型统计，验证了样本文件中的位点类型的数目恰好为 16 类。

通过结合计算机中的二进制编码方式，本题中采用两位二进制数来表示一种碱基 (比如：以二进制数‘00’来编码碱基‘A’，具体编码方式见表 1.1)。且采用这样的数值表示方式，可以完整的将 16 种类型的位点表示出来，且数值表示的范围与所有的位点类型恰好一一对应。

表 1.1 四种碱基及对应的二进制数值编码方式

碱基	数值编码	碱基	数值编码
A	00	T	01
C	10	G	11

由表 1.1，我们可以对样本染色体片段中 9445 个位点所包含的 16 中位点类型进行数值编码(表 1.2 列出其中 6 种，具体见附录 1)。

表 1.2 样本位点类型及对应的编码方式

位点类型	编码方式	位点类型	编码方式
AA	0000	AT	0001
AC	0010	AG	0011
TA	0100	TT	0101

通过采用这种数值编码方式，可以有效的利用计算机中的二进制数值之间的运算方法：‘与’，‘或’，‘非’以及‘异或’。而且，采用二进制的编码方式的一大优点是，可以利用已经成熟的二进制检测技术：奇偶校验以及海明码校验的方法，来对某一基因片段中的变异或致病位点进行检测。

3.2 问题解决

利用 Linux 中的脚本技术，将 genotype.dat 导入系统，之后再将 A, T, C, G 分别以 00, 01, 10, 11 替换。具体替换过程如下：

```
#!/bin/bash
sed 's/A/00/g' genotype.dat > genotype_tmp.dat
```

```
sed 's/T/01/g' genotype_tmp.dat > genotype.dat
sed 's/C/10/g' genotype.dat > genotype_tmp.dat
sed 's/G/11/g' genotype_tmp.dat > genotype.dat
```

4. 问题二

4.1 问题分析

由 1.2 (2)可知，问题二要求采用一种方法，对样本信息(genotype.dat)文件中的某遗传疾病 A 的染色体上的位点进行对比筛选，从而找出其中最有可能的一个或几个位点。

针对此问题，我们首先将样本信息文件 genotype.dat 中的整体数据视作一个1000×9445的二维矩阵。因为庞大的数据量，且其中很多的位点为所有样本共有的共性位点，所以，通过全基因组关联性分析的方法以及统计学原理。我们对此矩阵进行降维处理。再从经过处理之后的矩阵中，以按列统计的方式，将那些在有遗传疾病 A 的样本中出现的，且占总体样本比重大于84%的位点确定为致病位点。

4.2 问题解决

(1) 首先对此二维矩阵进行按列分割，即分割后的每一列，为样本总体的染色体中的基因片段上的同一位点的具体编码信息。具体分割操作代码见附录 2.4。

(2) 其次，对分割后的矩阵进行降维处理，降维标准为：对每一列中的 1000 个位点进行统计，假设整体样本中的位点数量分别为 X, Y, Z(其中，有遗传疾病 A 的样本中包含着三类位点的数目设为 X',Y',Z')。若 $X'/X\geq 84\%$ 或者 $Y'/Y\geq 84\%$ 或者 $Z'/Z\geq 84\%$，那么此位点则是高概率异常位点，相应的，其为致病位点的可能性也就高了。相似的，若上面的概率在$[40\%,70\%]$之间，我们则认为这类位点为中间概率异常位点，即此类位点是大部分样本都拥有的异常位点，且认为此位点是非致病位点。若上面计算的概率在$[0,40\%]$，即此类位点是没有遗传疾病 A 的样本中普遍共有的位点，显然，对于这类位点，我们也将其视为非致病位点。

具体的位点统计，筛选操作如下：

通过统计及筛选后，我们得出了附录 2.1，附录 2.2 和附录 2.3 中的具体位点信息。同时，依据附录 2.1 和附录 2.3 中的数据信息以及上面所提的位点比重计算方法，我们得出了如图 4.1 和图 4.2 的位点所占比重的曲线。同时，我们也对高概率，中间概率以及低概率异常位点在曲线中的位置进行标注。

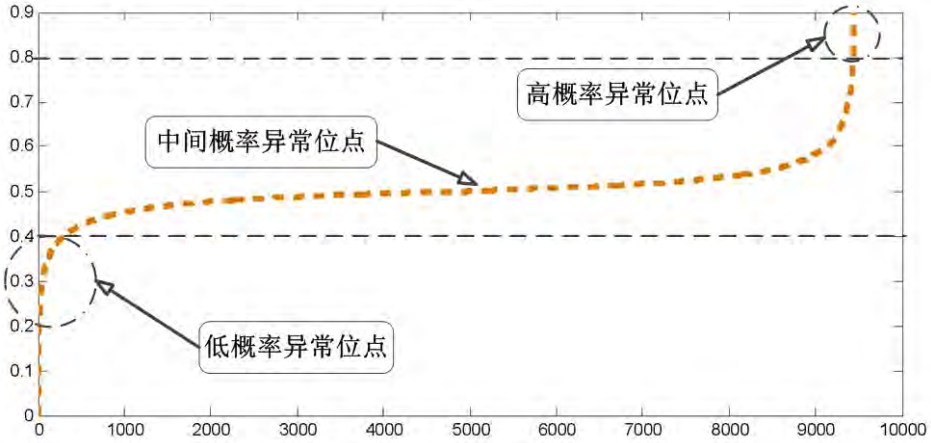

图 4.1 患遗传病 A 的样本位点占总样本位点的比重 1 (附录 2.1)

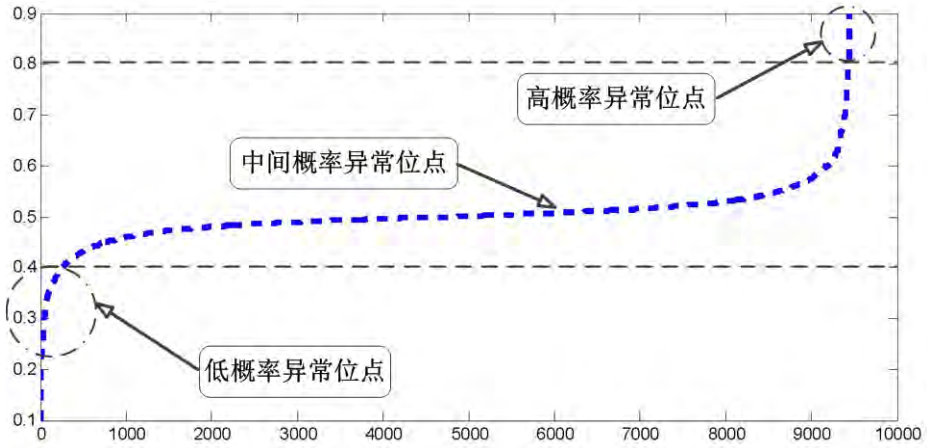

图 4.1 患遗传病 A 的样本位点占总样本位点的比重 2 (附录 2.3)

(3) 最后，依据(2)中统计及计算的结果，我们发现了其中的 8 个最有可能的致病位点，具体见表 4.1。

表 4.1 患遗传病 A 样本中最有可能致病的位点

异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目	所占比重
rs2999878	16	19	84.21%
rs3118505	6	7	85.71%
rs4846212	13	15	86.67%
rs3795263	9	10	90%
rs2480773	11	13	84.62%
rs1256341	12	14	85.71%
rs6429696	7	8	87.50%
rs2235927	9	10	90%

5. 问题三

5.1 问题分析

由 1.3 (3)可知，由于遗传疾病可以由基因中包含的位点的全集或子集合表现出来，因此问题三主要研究遗传疾病与基因的关联性，并给出了 300 个基因(见样本文件夹 gene_info 中的 300 个 dat 文件)，并以此来找出以遗传疾病最有可能相关的一个或几个基因。

由问题一中的思路，我们首先对在问题三中筛选，统计以及合并后的 155 种位点编码与 genotype.dat 样本文件中的 500 个患遗传疾病 A 的样本经过‘与’运算，构造一个500×155的二维矩阵。然后，利用 K-means 聚类算法，将这 155 个位点进行聚类。再利用公式算出 300 个样本文件中的基因与聚类后的每一个位点集合的相似度，并取其中相似度的最大值作为此样本基因是否为致病基因的判断标准。

5.2 问题解决

(1) 构造二维矩阵：假设 500 个患遗传疾病 A 的样本中含有我们已经筛选统计并合并好的 155 个位点所组成的矩阵为：

$$S^{'}=\begin{bmatrix} s^{'}_{1,1}, s^{'}_{1,2}, \cdots\cdots\cdots, s^{'}_{1,155} \\ \cdots\cdots\cdots\cdots\cdots\cdots \\ s^{'}_{500,1}, s^{'}_{500,2}, \cdots, s^{'}_{500,155} \end{bmatrix},$$

然后，对应的 155 个位点所组成的向量设为：$P=(p_1, p_2, \cdots, p_{155})$。之后，对$S^{'}$与$P$中的元素分别做‘与’运算，例如：若$s^{'}_{1,1}$处的位点为 AA，且$p_1$也为 AA，那么$s^{'}_{1,1}$与$p_1$作‘与’运算的结果为 1，则将结果 1 存入矩阵$S_{500\times155}$相应位置。同理，若$s^{'}_{1,1}$处的位点为 AA，$p_1$为其他位点，那么$s^{'}_{1,1}$与$p_1$作‘与’运算的结果为 0，则将结果 0 存入矩阵$S_{500\times155}$相应位置。

(2) K-means 聚类：以(1)中S矩阵中的位点类型为聚类对象，并分别聚成 3 类(设此三个类为集合Q_1, Q_2, Q_3)和 4 类(设此三个类为集合Q_1, Q_2, Q_3, Q_4)。

在数据挖掘中，聚类算法是给出大量的原始数据，然后通过相应的算法将其中具有相似特征的数据聚为一类。其中，K-means 聚类算法，就是实现这样的一个功能。具体算法定义如下：

首先给出未被标记的原始数据$\{x_1, x_2, \cdots, x_n\}$。然后，初始化$k$个随机数据：$\{u_1,$

$u_2,\cdots,u_k\}$，且这些x_n和u_k都是向量。

然后，由公式一和公式二，经过迭代，就能求出最终所有的u，且这些u就是最终所有类的中心位置。

公式一：

$$c^{(i)} = \arg\min_j \| x^{(i)} - u_j \|^2 ;$$

公式二：

$$u_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}} \quad 。$$

具体 matlab 代码见附录 3.1。

(3) 设题中给出的 300 个样本基因为$N = (n_1, n_2, \cdots, n_{300})$，依据计算基因片段与(2)中所聚成的类中的位点的相似度的公式：

$$(N_i \cap Q_j) / Num_{N_i},$$

其中，$i \in [1,300]; j \in [1,3]$(聚成 3 类)或$j \in [1,4]$(聚成 4 类)。

具体结果如图 5.1 所示：

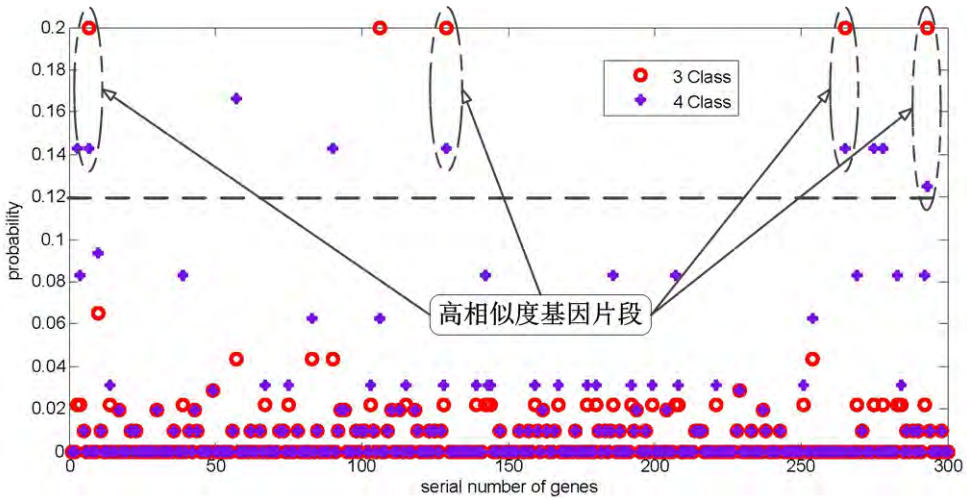

图 5.1 样本基因(300 组)与聚成 3 类和 4 类的相似度

由图 5.1，我们可以知道图中所标注的四个样本基因(即样本文件加 gene_info 中的 gene_7, gene_129, gene_265 和 gene_293)，在采用 K-means 聚类算法，将 155 个位点分别聚成 3 类和 4 类时，其与这些类中的位点集合的相似度分别高达 20%与 14%左右。因此，我们认为此 4 个基因为与遗传疾病 A 相关的基因。

6. 问题四

6.1 问题分析

由 1.3 (4)可知，问题四通过把相关性状或疾病看作一个整体，来探寻与性状或疾病相关的位点或基因的实验方法，要求根据 multi_phenos.txt 文件给出的 1000 个样本的 10 个相关联性状的信息及其 9445 个位点的编码信息(见 genotype.dat)，找出与 multi_phenos.txt 中 10 个性状有关联的位点。

所以，我们首先假设每一个相关性状是多个位点的组合引发的，是一种自然现象。因此，在这种情况下，通过全概率公式，我们有：

P(位点 i 为 j 时能引发相关性状)= $\sum_{z=1}^{Num_{M_{ij}}}$ Pr(z 组合中位点 i 为 j 时能引发相关性状)× Pr(z 组合发生)。

其中，M_{i_j} 是一种组合，该组合 i 位点的编码为 j；

Pr(z 组合发生)=z 组合出现的个数/所有组合出现个数。

通常情况下，Pr(z 组合中位点 i 为 j 时能引发相关性状)是一个自然概率，在有病的人和无病的人中值是相同的，但 Pr(z 组合发生)就不一样了，它在有病的人和无病人中的差距可能很大，因为有病人位点的编码组合与无病人位点的编码组合相差很大。由此我们可得到在有病的人中或无病人中 Pr(位点 i 为 j 时能引发相关性状)的值是不相同的。

我们将有病人的数据被分为了 2 组，每组 250 个，同时也将没病的病人分成 2 组，每组 250 个。根据大数定律，应该有：

(1) Pr(位点 i 为 j 能引发相关性状)有病组 1=Pr(位点 i 为 j 能引发相关性状)有病组 2；

(2)有 Pr(位点 i 为 j 能引发相关性状)无病组 1=Pr(位点 i 为 j 能引发相关性状)无病组 2。

如果其中的一式不等，说明这个位点组合是不稳定的，与我们的假设是不成立，因此只要这两式同时成立，我们便可认为这个位点是能够引发某一些特性的。其中 Pr(位点 i 为 j 引发相关性状)=位点 i 为 j 的人员拥有相关性状/位点 i 为 j 的人员的总个数。

6.2 问题解决

(1) 确定各个位点中编码的顺序（即 X，Y，Z）的顺序。

(2) 构建 3 个 0-1 矩阵，分别位 A，B，C，它们的维数均是 1000×9445，根据给出人员位点的编码信息，A 取各个位点上编码顺序为 1 的位置上的值为 1，其余值为 0；B 取各个位点上编码顺序为 2 的位置上的值为 1，其余值为 0；C 取各个位点上编码顺序为 3 的位置上的值为 1，其余值为 0；

(3) 将给出的 1000 个用户的 10 个相关性状表现样本看作是一个 0-1 矩阵为 D，维数为 1000×10，同时设 $E_{i\text{-}j}^{S}$ 是 10×9445 维数的矩阵，E 的每一行是相同的，且每一行上各个元素的值为对应的 S 矩阵 i 到 j 行的上每一列的元素之和,计算:

$(D_{1\text{-}250}^{T}A_{1\text{-}250}./E_{1\text{-}250}^{A})./(D_{251\text{-}500}^{T}A_{251\text{-}500}./E_{251\text{-}500}^{A})$ ①

$(D_{501\text{-}750}{}^{T}A_{501\text{-}750}./E_{501\text{-}750}{}^{A})./(D_{751\text{-}1000}{}^{T}A_{751\text{-}1000}./E_{751\text{-}1000}{}^{A})$ ②

$(D_{1\text{-}250}{}^{T}B_{1\text{-}250}./E_{1\text{-}250}{}^{B})./(D_{251\text{-}500}{}^{T}B_{251\text{-}500}./E_{251\text{-}500}{}^{B})$ ③

$(D_{501\text{-}750}{}^{T}B_{501\text{-}750}./E_{501\text{-}750}{}^{B})./(D_{751\text{-}1000}{}^{T}B_{751\text{-}1000}./E_{751\text{-}1000}{}^{B})$ ④

$(D_{1\text{-}250}{}^{T}C_{1\text{-}250}./E_{1\text{-}250}{}^{C})./(D_{251\text{-}500}{}^{T}C_{251\text{-}500}./E_{251\text{-}500}{}^{C})$ ⑤

$(D_{501\text{-}750}{}^{T}C_{501\text{-}750}./E_{501\text{-}750}{}^{C})./(D_{751\text{-}1000}{}^{T}C_{751\text{-}1000}./E_{751\text{-}1000}{}^{C})$ ⑥

(4) 步骤三求出的结果均是 10×9445 维的矩阵，其中①, ③, ⑤代表有病的一类，而②, ④, ⑥代表无病的一类。一类有 3 个矩阵数据是由于每个位点有 3 种可能的编码，每一个矩阵代表其中的一种编码。由于我们要寻找 $Pr_{有病组1}$(位点 i 为 j 能引发相关性状) =$Pr_{有病组2}$(位点 i 为 j 能引发相关性状)且 $Pr_{无病组1}$(位点 i 为 j 能引发相关性状) =$Pr_{无病组2}$(位点 i 为 j 能引发相关性状)的位点，对于每一个相关性状，执行如下步骤：找出式①，②，③，④，⑤和⑥与该性状对应的行中的值大于 1-ops 的列序号，并求出这 6 组数据的交集，交集即为我们要确定的位点的位置编号。注：ops 是一个阀值，由于这里的数据量较小，计算概率的时候存在着一些偏差。ops 对于每种相关性状的值是不相同的，一般取值为 0~0.1，具体要看取到的位点集合中元素的个数。下面显示了 2 组实验后的数据，第一组实验数据中阀值偏小，每一个相关性状获得出的位点个数为 1，第二组实验数据中阀值稍大，每一个相关性状获得出的位点个数为 2-3。（详细 matlab 代码见附录 4）

表 6.1 低阀值下的可能的关联位点

相关性状编号	阀值(ops)	可能位点位置编号	位点名称
1	0.0965	3922	rs10737914
2	0.0275	8335	rs7535816
3	0.0475	158	rs4648611
4	0.0565	8318	rs17362501
5	0.0905	3717	rs17038468
6	0.0625	3608	rs28551666
7	0.0245	2794	rs9662275
8	0.0760	5539	rs6698315
9	0.0985	7387	rs4649197
10	0.0730	469	rs4310388

表 6.2 稍高阀值下的可能的关联位点

相关性状编号	阀值(ops)	可能位点位置编号	位点名称
1	0.1040	3922 6790	rs10737914 rs2473253
2	0.0330	8335 1913	rs7535816 rs11122083
3	0.0495	158 3154	rs4648611 rs541695
4	0.0640	8318 6175	rs17362501 rs1076623

5	0.0965	3717 1729 7554	rs17038468 rs7535806 rs3122034
6	0.0700	3608 2106	rs28551666 rs17030322
7	0.0325	2794 4079	rs9662275 rs10928056
8	0.0790	5539 8059	rs6698315 rs4436378
9	0.1010	7387 966	rs4649197 rs10915423
10	0.0785	469 845	rs4310388 rs1181871

7. 总结

本文主要围绕染色体中的基因片段以及位点与遗传疾病之间的联系展开的，为了从大量的样本数据中统计筛选出有用的位点信息，我们应用了全基因组关联性分析技术，统计学原理，数值编码技术，矩阵处理技术以及数据挖掘中的 K-means 聚类算法，主要工作如下：

1. 通过借鉴二进制的数值编码方式，来对样本信息中的位点中的碱基对进行数值编码，从而利用已经完善的‘与，或’等二进制数据计算方法，从而实现简化样本信息的统计筛选的工作量。

2. 通过应用矩阵分割技术以及对矩阵进行降维处理的方法，来去除样本信息中的冗余数据，从而实现高效，准确的找到并分析出样本数据中的最有可能的致病位点。

3. 通过应用数据挖掘中的 K-means 聚类算法思想，可以实现某些相似位点的聚类，再通过算法提供的公式，来计算样本基因片段包含的位点与这些聚类的位点集合的相似度，从而来研究如问题三中要求的寻找可能的致病基因。

参考文献

[1] 严卫丽：复杂疾病全基因组关联研究进展——遗传统计分析。DOI：10.3724/SP.1.1005.2008.00543, 2008 年 5 月，543-549 页。

[2] 陈锋，柏建岭，赵杨，荀鹏程：全基因组关联研究中的统计分析方法。中华流行病学杂志，第 32 卷第 4 期，2011 年 4 月。

[3] 全超，孙文靖，于旸，白静：统计学在肿瘤细胞遗传学中的应用。国际遗传学杂志，第 35 卷第 6 期，DOI：10.3760/cma.j.issn.1673-4386.2012.06.005, 2013 年 2 月。

附录

附录 1

16 种具体位点类型以及对应的编码方式：

位点类型	编码方式	位点类型	编码方式
AA	0000	AT	0001
AC	0010	AG	0011
TA	0100	TT	0101
TC	0110	TG	0111
CA	1000	CT	1001
CC	1010	CG	1011
GA	1100	GT	1101
GC	1110	GG	1111

附录 2.1

经过统计筛选后，得到的 155 个异常位点类型，及其在患有遗传疾病以及在所有样本信息中的数量(部分)：

异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目	异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目
rs10737914	12	17	rs11203280	12	17
rs669410	12	17	rs2075993	12	17
rs2803309	5	7	rs349394	15	21
rs14708	5	7	rs7526401	10	14
rs4661590	20	28	rs2977290	15	21
rs17427202	5	7	rs878235	10	14
rs11589294	23	32	rs3765695	31	43
rs2459984	13	18	rs351609	13	18
rs10788679	13	18	rs12134924	21	29
rs4648648	8	11	rs3820253	8	11

rs2066002	8	11	rs3924436	16	22
rs4509550	19	26	rs1459765	22	30
rs2180183	11	15	rs1459760	17	23
rs3810982	17	23	rs11121242	12	16
rs11800086	3	4	rs12045736	15	20
rs2024724	12	16	rs7556176	9	12
rs2377060	24	32	rs12401792	12	16
rs12036216	10	13	rs10797437	10	13
rs883867	10	13	rs11203327	10	13
rs4912048	20	26	rs437722	10	13
rs350187	7	9	rs2473277	11	14
rs10915577	15	19	rs731024	15	19
rs2250358	38	48	rs4654418	8	10
rs533123	8	10	rs4912019	25	31
rs2999878	16	19	rs3118505	6	7
rs4846212	13	15	rs3795263	9	10

附录 2.2

经过统计筛选后，得到的 155 个异常位点类型，及其在患有遗传疾病以及在所有样本信息中的数量(部分)：

异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目	异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目
rs12119911	111	198	rs746881	111	198
rs4543765	174	310	rs2803309	109	194

rs10927093	109	194	rs9426296	109	194
rs3104434	109	194	rs770720	113	201
rs770720	113	201	rs209692	131	233
rs2816050	158	281	rs7519807	117	208
rs2744677	117	208	rs12044299	180	320
rs4609454	157	279	rs1883567	179	318
rs585160	98	174	rs1188402	118	209
rs2797682	192	340	rs2038903	91	161
rs3806425	13	23	rs10914189	130	230
rs11582551	108	191	rs7543405	278	491
rs16824712	153	270	rs11573221	34	60
rs1188347	160	282	rs2477782	126	222
rs2480772	175	308	rs2143808	129	227
rs3766306	153	269	rs2782810	111	195
rs869179	127	223	rs11586865	131	230
rs909823	129	226	rs2801178	161	282
rs4661526	116	203	rs12401776	109	190
rs13513	123	214	rs2254669	134	233
rs6429804	104	180	rs12090714	111	192
rs11121675	33	57	rs3131419	132	228
rs1924270	148	255	rs11203254	61	105
rs10492940	25	43	rs7512834	103	177

rs707472	148	254	rs12062540	102	175
rs4908443	123	211	rs4508063	56	96
rs11247865	149	254	rs7550997	125	213
rs12758257	110	187	rs66661776	118	199
rs3000851	111	187	rs12036216	130	217
rs427552	8	12	rs12031599	20	28

附录 2.3

经过统计筛选后，得到的 155 个异常位点类型，及其在患有遗传疾病以及在所有样本信息中的数量(部分)：

异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目	异常位点名称	在有遗传病 A 样本中的数目	所有样本中的总数目
rs642696	12	17	rs12726255	5	7
rs6665175	10	14	rs1193227	10	14
rs1280989	10	14	rs12410376	5	7
rs10489442	10	14	rs927727	5	7
rs3820034	23	32	rs3767150	21	29
rs10917404	21	29	rs3813199	8	11
rs6429804	8	11	rs2506969	11	15
rs11588846	14	19	rs478103	14	19
rs6694109	20	27	rs10799257	9	12
rs707582	12	16	rs2072996	6	8
rs6690493	9	12	rs10803320	9	12
rs7547731	6	8	rs7534452	12	16

rs2483679	16	21	rs35232720	13	17
rs1001567	10	13	rs7525571	10	13
rs2070658	14	18	rs2053	11	14
rs7527904	15	19	rs909813	19	24
rs12072310	8	10	rs170629	8	10
rs3753271	8	10	rs2745260	16	20
rs2526833	4	5	rs12746273	8	10
rs12141588	17	21	rs2027508	9	11
rs12032209	14	17	rs3765736	10	12
rs3765964	10	12	rs2480773	11	13
rs1256341	12	14	rs6429696	7	8
rs2235927	9	10			

附录 2.4

此附录内容为问题二中，通过对 genotype.dat 样本文件进行筛选，统计，得到与某种遗传疾病相关的位点的具体信息的操作代码。

首先将 genotype.dat 的第一行提取出来。

其次，通过 Linux Shell 统计出患有遗传病和全部样本信息中每个位点中各种编码出现的次数，并将这些数据写入文件。具体代码如下：

```
#!bin/bash
i=1
touch tmp1.txt tmep2.txt #创建临时文件用来存储结果
while [ $i -le 9445 ]
do
a=$(awk -v i="$i" 'NR==1,NR==500 {a[ $i ]++} END {for (j in a) print j,a[ j ]}' genotype.dat)
b=$(awk -v i="$i" 'NR==1,NR==1000 {a[ $i ]++} END {for (j in a) print j,a[ j ]}' genotype.dat)
echo $a $b >> tmp1.txt
i=$(($i+1))
i=1
```

while [$i -le 9445]

do

read d1 d2 d3 d4 d5 d6 < <(awk -v i="$i" 'NR==i {print $2 $4 $6 $8 $10 $12}' tmp1.txt

echo $d1 $d2 $d3 $d4 $d5 $6 >> tmp2.txt

done

为了进行进一步的数据处理，将 tmp2.txt 中的数据导入到 Matlab 中，并求得各个位点上的各种碱基编码在患有遗传病的样本和所有样本数据信息之间的比例，以下是相应的 script 代码。

Data=importdata('tmp2.txt') %将得到的数据导入到 matlab 中。

注：导入数据后会有两组数据出现错误，由于 shell 在统计每一位点中编码出现次数的时候，会把为 0 的选项删除，在 matlab 中将这两组数据进行添加，接下来，求得各种编码在各个位点上出现的次数有病和所有人的比例。

Persent1=Data(:,1)./Data(:,4);

Persent2=Data(:,2)./Data(:,5);

Persent3=Data(:,3)./Data(:,6);

然后，获取每组比例中较高比例的值所对应的位点编号

Position1=find(Data1>0.7)

Position2=find(Data2>0.56)

Position3=find(Data3>0.7)

从中，可以获得到 155 个位置点，这些位点上的某一种编码对应的比例值较高，由这些位置，提取 155 组数据（由于每个位点有三种可能的编码，这 155 组数据依据编码的次序被分为了 3 组，为 Data1，Data2 和 Data3），其中最后一列为位点编号，前 3 列为位点中的 3 个编码在患有遗传病的样本信息中统计的数据，而后 3 列为在所有样本信息中统计的数据，详见附录 1。

附录 3.1

此附录内容为问题三中，所要研究的寻找与某种遗传疾病相关的基因的具体操作代码。

首先将问题二中获得的 155 个位点上的编码生成与所有样本信息相关的 0,1 矩阵，矩阵的维数为1000×155，其中每一列对应着位点上的一种编码，每一行代表一个参与人员是否存在这些位点上所对应的编码，如果存在，则为 1，不存在，则为 0，具体的 Matlab 代码如下所示。

for i=1:1:size(Data1,1)

position=find(Data4==Data1(i,7)); %position 是将 Position1，Position2 和 Position3 中的值按照从小到达的顺序组成的一个一维数组

for j=1:1:500

if strcmp(result4{Data1(i,7)}(1,1:2),result5{1,position(1,1)}{j,1})==1 %由于每个位点有三种可能的编码，我们在第二题统计各个位点上编码出现的次数的时候已经定义了这三种编码的次序，result4 保存了所有的这些编码次序的信息，其每一行如：AA AG GG。result5 种保存了 1000 个人的 155 个位点的编码信息，是一个 1*155 的 cell，每个 cell 中记录了该位点对应的 1000 人的编码。

result6(j,position)=1;

```
end
end
end
for i=1:1:size(Data2,1)
position=find(Data4==Data2(i,7));
for j=1:1:500
if strcmp(result4{Data2(i,7)}(1,4:5),result5{1,position(1,1)}{j,1})==1
result6(j,position)=1;
end
end
end

for i=1:1:size(Data3,1)
position=find(Data4==Data3(i,7));
for j=1:1:500
if strcmp(result4{Data3(i,7)}(1,7:8),result5{1,position(1,1)}{j,1})==1
result6(j,position)=1;
end
end
end
```

%result6 是我们最终生成的 0-1 矩阵，见附录 2

接下来，我们通过 k-means 算法，将这个 0-1 矩阵进行聚类，其中位点即（列向量）为聚类的对象，最终我们将这些位点聚类为 3 类，见附录 3。

最后，分别求得各个基因与我们获得的 3 类数据的相似度（即基因中位点在某一分类中的个数/基因中位点的总的个数），取 3 个值的最大值为最终致病的概率。

```
function [ persent ] = genocompare( Class1geno, Class2geno, Class3geno) %该函数主要进行基因匹配，找出最有可能致病的基因
%Classigeno 为第 i 类位点致病位点的集合
%由于我们将致病位点分为3类，第一类只要出现就有很大的可能引发疾病，第二类出位点的个数较多能够引发疾病，第三类出现位点的个数较多时才能引发疾病
 persent=zeros(300,1);%机了各个基因引发疾病的概率
 %依次遍列 300 个基因，获取各个基因占据各个类的比例，取其中的最大值，并记录下来
 for i=1:1:300
    Geno=importdata(['/root/                下                载/2016/2016/B/B/gene_info/gene_',num2str(i),'.dat']);%geno 是一个 cell，里面存储的是基因所涉及到的所有位点
    Temp1=intersect(Geno,Class1geno);%获取与第一组位点相同的个数
    Temp2=intersect(Geno,Class2geno);%获取与第二组位点相同的个数
    Temp3=intersect(Geno,Class3geno);%获取与第三组位点相同的个数

persent(i,1)=max([length(Temp1)/length(Class1geno),length(Temp2)/length(Class2ge
```

```
no),length(Temp3)/length(Class3geno)]);
    end
end
```

数据切割的 Linux Shell

```
#！/bin/bash
i=1
while [ $i -le 9445 ]
do
awk -v i="$i" '{print $i}' genotype.dat > 9445/$i.txt
i=$(($i+1))
done
```

附录 3.2

```
%N 是数据一共分多少类
%data 是输入的不带分类标号的数据
%u 是每一类的中心
%re 是返回的带分类标号的数据
function [u re]=KMeans(data,N)
    [m n]=size(data);     %m 是数据个数，n 是数据维数
    ma=zeros(n);          %每一维最大的数
    mi=zeros(n);          %每一维最小的数
    u=zeros(N,n);         %随机初始化，最终迭代到每一类的中心位置
    for i=1:n
        ma(i)=max(data(:,i));      %每一维最大的数
        mi(i)=min(data(:,i));      %每一维最小的数
        for j=1:N
            u(j,i)=ma(i)+(mi(i)-ma(i))*rand();  %随机初始化，不过还是在
每一维[min max]中初始化好些
        end
    end
    while 1
        pre_u=u;                %上一次求得的中心位置
        for i=1:N
            tmp{i}=[];          % 公式一中的 x(i)-uj,为公式一实现做准备
            for j=1:m
                tmp{i}=[tmp{i};data(j,:)-u(i,:)];
            end
        end
        quan=zeros(m,N);
        for i=1:m           %公式一的实现
            c=[];
            for j=1:N
```

```
                c=[c norm(tmp{j}(i,:))];
            end
            [junk index]=min(c);
            quan(i,index)=norm(tmp{index}(i,:));
        end
        for i=1:N                    %公式二的实现
            for j=1:n
                u(i,j)=sum(quan(:,i).*data(:,j))/sum(quan(:,i));
            end
        end
        if norm(pre_u-u)<0.1    %不断迭代直到位置不再变化
            break;
        end
    end
    re=[];
    for i=1:m
        tmp=[];
        for j=1:N
            tmp=[tmp norm(data(i,:)-u(j,:))];
        end
        [junk index]=min(tmp);
        re=[re;data(i,:) index];
    end
end
```

附录 4

第一步 A，B，C 矩阵的生成，A 被分割为 myresult11，myrresult12，myresult13，myresult14，类似的有 B 和 C 的分割为 myresult2*和 myresult3*。

```
function [ myresult ] = data2matrix( result4 )
%该函数是将大量的人的位点信息转换为矩正，其中每列的数值只可能为 1，2，3，其序列由 result4 中定义的序列决定
    myresult=zeros(1000,9445);
    for i=1:1:9445
      result1=zeros(1000,1);%result1 是 1000 行一列
      Data1=importdata(['/root/下载/2016/2016/B/B/9445/',num2str(i),'.txt']);
      for j=1:1:1000%每个 Data1 中有 1000 个信息
        if strcmp(Data1{j,1},result4{i,1}(1,1:2))==1
          result1(j,1)=1;
        elseif strcmp(Data1{j,1},result4{i,1}(1,4:5))==1
          result1(j,1)=2;
        elseif strcmp(Data1{j,1},result4{i,1}(1,7:8))
          result1(j,1)=3;
```

```
            end
        end
        myresult(:,i)=result1;
    end
end
myresult11=zeros(1000,9445);
myresult11(find(myresult(:,1)==1))=1;
.....
result11=multi_phenos1'*myresult11;
result12=multi_phenos2'*myresult12;
result13=multi_phenos3'*myresult13;
result14=multi_phenos4'*myresult14;
result21=multi_phenos1'*myresult21;
result22=multi_phenos2'*myresult22;
result23=multi_phenos3'*myresult23;
result24=multi_phenos4'*myresult24;
result31=multi_phenos1'*myresult31;
result32=multi_phenos2'*myresult32;
result33=multi_phenos3'*myresult33;
result34=multi_phenos4'*myresult34;
test11=result11./repmat(sum(myresult11),10,1);
test12=result12./repmat(sum(myresult12),10,1);
test13=result13./repmat(sum(myresult13),10,1);
test14=result14./repmat(sum(myresult14),10,1);
test21=result21./repmat(sum(myresult21),10,1);
test22=result22./repmat(sum(myresult22),10,1);
test23=result23./repmat(sum(myresult23),10,1);
test24=result24./repmat(sum(myresult24),10,1);
test31=result31./repmat(sum(myresult31),10,1);
test32=result32./repmat(sum(myresult32),10,1);
test33=result33./repmat(sum(myresult33),10,1);
test34=result34./repmat(sum(myresult34),10,1);
final11=test11./test12;
final12=test13./test14;
final21=test21./test22;
final22=test23./test24;
final31=test31./test32;
final32=test33./test34;
function [ finalresult ] = compare( final11,final12,final21,final22,final31,final32)
%该方程用来获取 6 个矩阵的交集
    finalresult=zeros(10,30);
    for i=1:1:10
        for j=1:1:200 %20 对应的 ops 是 0.05
            ops=0.001*j;
```

```
            ans11=find(final11(i,:)>(1-ops)&final11(i,:)<(1+ops));
            ans12=find(final12(i,:)>(1-ops)&final12(i,:)<(1+ops));
            ans21=find(final21(i,:)>(1-ops)&final21(i,:)<(1+ops));
            ans22=find(final22(i,:)>(1-ops)&final22(i,:)<(1+ops));
            ans31=find(final31(i,:)>(1-ops)&final31(i,:)<(1+ops));
            ans32=find(final32(i,:)>(1-ops)&final32(i,:)<(1+ops));
            ans1=intersect(ans11,ans12);
            ans2=intersect(ans21,ans22);
            ans3=intersect(ans31,ans32);
            temp=intersect(ans1,ans2);
            result=intersect(temp,ans3);
            if length(result)>1
               finalresult{i,1}=1-ops;
               finalresult{i,2}=result;
               break;
            end
            disp(length(result));
            finalresult(i,j)=length(result);
         end
      end
end
```