生物信息学 原始文件
第一卷。 30号。 2014年9月，第1266-1272
页doi：10.093/生物信息学/btu014

遗传学和人口分析

提前查阅出版物2014年1月9日

使用位置Burrows-Wheeler变换(PBWT)进行有效的单倍型匹配和存储)

理查德·杜宾
威康信托桑格研究所，威康信托基因组校园，剑桥CB101SA，英国

副编辑：杰弗里·巴雷特

摘要
动机：近几年来，基于后缀数组的Burrows-Wheeler变换方法被广泛应用于DNA序列读取匹配和组装。 这些算法提供了非常快速的搜索算法，线性的搜索模式大小，在一个高度可压缩的表示数据集被搜索。同时，基因型数据的算法发展主要集中在基于对参考数据的隐马尔可夫模型表示的概率匹配的分阶段和归责统计方法上，这种方法虽然强大，但计算效率要低得多。 在这里，提出了一种使用后缀数组思想的单倍型匹配理论，它应该比目前基因型算法处理的数据集要大得多。
结果：给定具有N个双等位基因变量位点的M序列，给出了一种基于位置前缀阵列的数据表示O(NM)算法，称为位置Burrows-Wheeler变换(PBWT)。 在大型数据集上，这种压缩的运行长度编码比在原始数据上使用gzip小了100多倍。 使用这种表示，给出了一种在O(NM)时间而不是O(NM)时间内找到集合中所有最大单倍型匹配的方法[2])，如预期的那样，从朴素的两两比较，也是一种快速算法，经验上独立于M给定索引的足够内存，以找到新序列和集合之间的最大匹配。 讨论中包括一些关于如何将这些方法用于估算和分阶段的建议。
可用性：http://github.com/richarddurbin/pbwt
联系人：理查德·德宾@sanger.ac.uk

于2013年9月14日收到；2013年12月9日修改；
于2014年1月4日接受

1 导言

给定大量对齐的遗传序列或单倍型集合，通常感兴趣的是在集合中的序列之间，或在新的测试序列和集合中的序列之间找到长的匹配。 例如，足够长的相同子串是候选的区域，它们通过来自共同祖先的下降(IBD)而相同。 （我将使用“子串”一词来表示连续子序列，这在计算机科学文本匹配文献中是标准的。） 当使用估算方法推断缺失值时，您希望识别与被估算位置周围的测试序列尽可能接近的序列，例如那些是IBD的序列，或者至少与测试序列共享长匹配。 最大限度地利用这种长匹配的数量也可以形成基因型分期的基础。

原始子字符串匹配测试需要ON²每个测试序列的M时间，其中有N个可变位点和M序列，因此有ON²m² 在一组序列中完成所有对比较的时间。 通过保持一个运行的匹配分数来找到最大匹配，就像在BLAST中一样，可以很容易地将这一点降低到每次测试的O(NM)，因此ONM²Langmead等人，2009年李和杜宾，2009年等人，2009年 在整个集合中，但对于大型M来说，这仍然很大。最近，基于后缀数组的方法在标准序列匹配中被证明是强大的，例如Bowtie()、BWA()和SOAP2(Li)。 在这里，描述了一种基于后缀数组的方法，它可以在O(Nm)时间内在一组序列中找到最佳匹配，在O(Nm)时间内对数据集进行预处理后，也可以在O(Nm)时间内找到最佳匹配，并在O(N)时间内找到经验上最佳的单倍型匹配。

这里描述的算法和基于标准后缀数组的序列匹配之间的差异是由于有许多序列是所有的
相同的长度和已经对齐。 所以一方面没有
需要考虑测试序列相对于的偏移量
序列在集合中，但另一方面，测试序列是长的，我们正在寻找最大匹配的
测试序列的任意子字符串，而不是整个测试序列的任意子字符串。

2 方法

当观察来自人类或其他二倍体生物的遗传数据时，每个人有两个潜在的基因组序列，一个来自他们的父亲，一个来自他们的母亲。 这些被称为“单倍型”序列。 在这里，我考虑的情况是，我们分别给出了这两个序列，而不是非相位二倍体‘基因型’序列，其中两个单倍型序列已经被观察到在一起。

考虑一组M单倍型序列x_i我¼0，。。。 ðM-1ÞN个变量站点，由k索引，编号从0到(N-1)。 我们可以把所有的站点都用0或0的值进行双等位 1， 所以 a 典型的 地点 x_i½k]2f0， 1g： 为了任何 顺序 S， 让 我们写作 $s½k_1, k_2þ$ 去 代表 的 半开放的 子字符串 的 s 开始吧 在k_1 在ðk完成。 —1Þ：我们会说，从k来的s和t之间有一个“匹配$_1$ 敬k_2 如果是 k_1, k_2 t k_1, k_2 如果没有扩展也是匹配的，则此匹配是“本地最大值。e. 如果K_1 ¼0或$s½k_1 - 1]6¼t½k_1 - 1]$， 和k_2 ¼N或$s½k_2$6¼t½k_2当将s与序列X的集合进行比较时，我们说s与x有一个集合最大匹配$_i$ 来自K_1敬k_2 如果匹配是局部最大的，并且不再来自匹配 s 去 任何 其他 x_j 那个 包括 的 间隔 $k_1, k_2þ$:

对于一些应用程序，我们将对集合最大值感兴趣
在X内匹配，即。 每个x的集合最大匹配$_i$ 去
Xn$\mathfrak{x}_i$ g:

我们的方法的基础是考虑序列集合X的子串上的一种特定形式的排序。 下面是对这种排序的解释，并给出了为什么它是重要的动机。 我们将考虑对0到N之间的每个位置k进行单独的排序。对于给定的k，让我们在X中对序列x进行排序，使它们的反向前缀x½0，kÞ是有序的，我的意思是，从(k-1)返回到0的前缀的反向序列是以自然的方式排序的，如果前缀是相同的，则根据它们在X中的索引i排序。

让我们考虑两个序列之间的集合最大匹配
从k^0 敬k。如果我们在k处按这个相反的顺序排序，那么最大匹配序列必须是相邻的，因为如果它们之间有另一个前缀排序，那么 它
将不得不匹配两者从k^0 到k，因为排序顺序，它将必须匹配两个位置中的一个ðk^0 − 1Þ，因为最大匹配序列必须采取不同的

值在那里，只有两个可能的值。 新的
因此，序列将形成一个更长的匹配，这与原对之间的匹配被设置为最大值的假设相矛盾。 严格地说，这个参数要求原始匹配在两个方向上都是最大的。 我们将看到这一点在下面很重要。 然而，这只是一个激励段落，所以没有必要考虑它在一个方向上是最大的情况。

那些事先接触后缀数组算法的人会注意到，我在这里谈论的是前缀和反向排序，而不是后缀和词典排序。 在本文中，标准理论的方向被逆转，使得算法通过从0到0的序列自然地向前处理 (N-1)，而不是向后；这对任何算法或结果都没有实质性影响，但将使我们能够按照它们自然出现的顺序处理非常大的数据集，并使一些表示法更自然。

2.1 前缀数组表示的派生

前一段中的论点表明，在k位置按反向前缀的顺序排序将有助于找到最大匹配。 在每个位置k找到所有前缀的排序顺序似乎在计算上是昂贵的，但事实并非如此。 如果我们知道k位置的排序顺序，算法1显示了如何导出排序

每个序列的k值 因此，我们可以计算整个
在位置(kþ1)通过一个简单的过程只看
所有k在一次通过所有se-的排序集-
数量，时间与NM成正比。 让a_k½i 是 的 序列x的索引
m5M$_m$ 从中导出k处反向排序中的第一前缀。 数组a_k 是数字0的排列，。..ðM-1Þ：因为我们经常想讨论按前缀顺序排序的序列，

我定义你^k_i 成为这个排序顺序中的第一个序列
y^k_i ¼x$_{a}$ $_{k}$当前位置关键观察是，条件是
值y^{k}½k】，元素的顺序$_{kþ1}$ 与他们在a中的顺序相同$_k$。 文中给出了一个例子。图1

算法1构建前缀数组-构建位置前缀数组$a_{kþ1}$
从a_k

```
u0, v0, 创建空数组a½], b½]
因为我¼0！ M—1do
  if y^k½k]然后¼0
    a½u]      a_k½i],uþ1
  其他
    b½v]      a_k½i],vþ1
a_kþ1 后面接a的级联 b
```

为了识别最大匹配从哪里开始，我们需要跟踪相邻前缀之间匹配的开始位置。 形式上，对于i40定义d_k½i] 去 是的 最小值j
在这里和下面的y的k后缀，
时 间 上 是 隐 式 的 k
y^k_{i-1} 然后设置d^k_k 然后可以显
任何最大匹配的开始以k结
y_i，是 是由 马克
有效地扩展算法1来更新d_k
如算法2所示。

算法2构建前缀和发散阵列
来自D_k 还有 a_k

```
u0, v0, pkþ1, qkþ1为我创
空数组a½], b½], d½], e½]
0! M-1是
  如果是的话,
    a½u]      a_k½i], d½u]p,
  其他
    b½v]      a_k½i], e½v]q,
a_kþ1  a后面接b的级联
d_kþ1       后面跟着d的级联 e
```

因为我们在处理双等数据， 只要d_k½i]40，y的值$_{i-1}$ d_k 我1
和y_i d_k 我1必须分别为0和1，因为它们按定义不同，并且按排序顺序排列。 这意味着作为推论，d是不可能的$_k$½i
等于d_k½i Þ 只要它们大于0，因为否则y_i d_k 我1需要同时是1和0，这是不可能的½i]－]

我将数组的集合称为a_k 对于所有k，X的“位置前缀数组。 这些与标准后缀数组有关，但除了作为前缀而不是后缀数组，因为排序是相反的方向，它们的不同是因为它们形成了一组N个数组，每个数组的大小为M，而不是单个大小为NM的数组。 医生$_k$ 在标准后缀数组算法中包含相当于“最长公共前缀”值。

2.2 在X长于最小长度L的 找到所有匹配

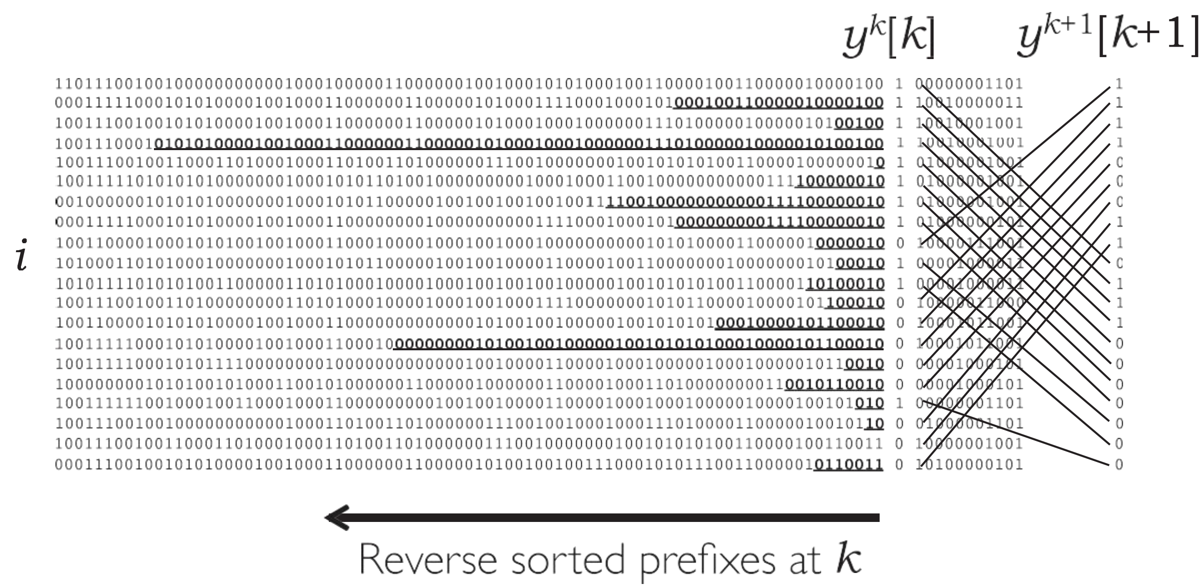
现在我们可以用a了$_k$ 和d_k 数组以有效地找到匹配。 为了只计数一次匹配，我们将随着k的增加扫过序列，并且只在每个k处报告匹配
最后在k，我。e. 对于哪个
要比L长，我们必须定义
为了所有的¯m 因此 对(i, j)
将发生在排序列表中的块中，由位置分隔

署长
2015-10-25 03:17:57
两个相邻prefix的最长match

已经建好的索引上，一个区间内在k点的最长的两个序列match由区间最大的d值决定。

署长
2017-10-28 08:27:21
这里关键是通过循环，跨迭代次数地记载两个序列之间的最小match，这个最小match的长度由多个串之间的最小d决定.

署长
2015-10-25 03:03:46
k+1位置的排序由k位置的排序和k+1位置的0和1决定，因此O(NM)时间可以排完序。



无花果。 1. 一组单倍型序列，在k位置按反向前缀的顺序排序，显示与前后分离的k处的值集，以及在右侧，位置(k+1)的顺序如何从此处导出，如算法1所示。 与前面以k结尾的序列共享的最大子字符串显示为粗体下划线；这些子字符串从

署长
2015-10-29 01:17:38

因此，我们在k位置继续排序列表，跟踪上次d_k[i]大于(k-L)。 考虑到这一点，我们的算法看起来非常像生成a的算法$_k$ 数组。

算法3报告长匹配-比L长 报告匹配在X以k结尾

u0，v0，创建空数组a½]，b½]

因为我¼0！ M—
1
d_k½i]4k-那么
如果u40和v40
所有0CI$_v$5u和0c i$_v$5v做
从a½i报告匹配$_u$]到b½i$_v$以ku0，v0结尾
如果是的话
a½u] a_k
其他
b½v] a_k v vþ1

该算法0ðmaxðNM，matchesÞÞ数； 报告大小为u和v的子集a[]和b[]对为O(NM)。

请注意，报表长匹配可以在用于计算a和d数组的数据中以相同的扫描方式运行，因此，如果我们乐于丢弃a的以前值$_j$ 和d$_j$ 对于j5k，我们可以在O(M)空间中进行。

此方法的一个变体可以传递所有扩展的匹配从一个位置的两个方向至少有一个最小长度1/2。 在这种情况下，我们考虑的是 d$_k$ði，jÞcðk-ð2LÞþ1ÞÞ 找到以位置k为中心的这类匹配集- ðL- 1Þ，并且不分离为y的子集，0或1。 这种方法可能与在一个位置寻找类似序列有关，也许是为了归责。 长匹配将在这个公式中重复多次，因此如果可能的话，最好使用类似的子集，因为它们是在扫描期间识别的，而不是存储它们以供将来使用。

2.3 在线性时间内找到X内的所有集合最大匹配

考虑一个序列y_i 在k的排序列表中……个集最大匹配结束在k？

显然，匹配必须直接对一个或……按排序顺序在其前面或后面。

候选人 间隔 [m,

刘波
2018-10-06 00:03:29

匹配block结束的标志

署长
2015-10-25 0

在yi的前……中所有序……yi+1或yi……区间找到……在k处于y……素具有se……间以外，……短。因此……结束于k

刘波
2018-10-06 00:06:15

匹配block结束的标志

这些j然后y_i 有……KN)，如果有y_j k……以向前扩展，并且……以k结束，迭代所

算法4报告集最大匹

为了k¼0！ 不知道
d_k½0]kþ1，$d_k$kþ1
因为我¼0！ M-
m¼i—1，n¼iþ
如果$D_k$cd$_k$½iþ
而$d_k$cd$_k$½i
我是-1
如果$D_k$cd$_k$½iþ1]然后
而$d_k$cd$_k$½iþ1]做
下一步我nþ1
对于j¼mþ1！ 我-1点
报告匹配的a_k½i]到a_k½j
为了j¼我þ1！ 恩-1做
报告 匹配 的 a_k 去 a_k

署长
2015-10-25 05:02:22

这条降低了时间复杂度，因为如果两个序列在k处也相等，说明不是k处结束的，因此不是set-maximal匹配。

形成approximate匹配

尽管有内部循环，这个算法……½k]限制搜索，以便每个位置最多从每个位置进行一次比较

方向。 完全准确，因为必须匹配
在序列的开始和结束时终止，这最后一条语句依赖于不存在任意大组从0到N相同的序列。在这些条件下，这也证明了X内的集合最大匹配的总数是由NM的固定倍数界的。

与报表长匹配一样，报表集最大匹配可以在用于计算a和d数组的数据中相同的扫描中运行，因此如果我们乐于丢弃以前的值
在我们去的a和d中，它也可以在O(M)空间中进行。

j *j*

2.4 从新序列z找到所有集最大匹配到X

接下来让我们考虑一个新序列z的情况，

并希望在它和集合X之间找到集合最大匹配。我们将再次扫描序列，并在这种情况下跟踪开始 e_k 最长的匹配z与一些 y_i 以k位结束，间隔½f_k，$g_k$þc½0，. . .，的MÞ

我们有z ...]z½e_k −1]6¼y_i½$_k$ —1]：我们允许 g_k 如果是的话$_{m-1}$ 包含在最长匹配集中。

当我们从k移动到(K1)时，我们想要一个有效的过程来更新e、f和g)。 首先，þ让我们想象一下，我们有一个程序来更新f和g到f^0 和g^0 给定固定的起始位置e_k。 如果$f^0$5g^0 然后至少有一些原始匹配开始吧在e_k 还有结束在k可以扩展到(kþ1)，所以$e^0_{kþ1}$ ¼e_k 通过 定义和我们 都完成了。 如果在另一只手
f^0 ¼g^0 然后，没有一个匹配可以扩展，因此
以k结尾的匹配到f之间的序列$_k$ 和g_k 都是集最大值，可以报告。 然后我们需要找到一个新的$e_{kþ1}$4e_k 以及相应的新$f_{kþ1}$5$g_{kþ1}$:

为了有效地更新f和g，我们需要u和v的值算法1。 我们当时没有存储它们，但现在让我们假设我们这样做了，在数组u中$_k$ 和五$_k$并跟踪序列中k位置的零值的总数作为值c_k它等价于算法1中阵列a[]的长度。 如果我们定义了$w_k$ði，0Þ¼你$_k$½i]和 $w_k$ði，1Þ¼c_k þ v_k½i] 那么 算法 1 告诉我 我们 那个$_{kþ1}$¼ði，y_i½k]Þ]¼a_k½i]：此外，如果α$_k$ 是逆的
置换a_k然后α$_{kþ1}$ *I* ½]¼$_k$ ðx_i k½]Þ这最后一个状态-蒙蒂给了我们一个关于如何更新f和g的线索。 如果我们
f^0 ¼wðf，z½k]Þ 定义，那么这将是a中的索引$_{kþ1}$ 第一个Se-昆斯Y_j 用jf表示y_j 这就是想要的½] 同样的g^0 我们想要更新g，所以我们现在也可以通过简单地从存储值中查找来更新f和g。

现在，正如我们上面所看到的，如果$f^0$5g^0 那么 我们都是 完成了。 上 另一方面，如果f^0¼g^0 然后没有匹配的扩展从e开始$_k$。 在k位置，我们知道z排序要么在自然前缀排序的块[f，g]之前，要么就在它之后。 所以它要么在f之前排序，要么在g之前排序
可以推断出y ... 在颠倒的自然顺序中
前缀。所 kþ ... 让我们考虑一下 ...
... 然 ... ¼g找$f_{kþ1}$ ... 如果是1，那么z匹配
y ... 比你好 ... 我们会设定1 ... 去寻找

$g_{kþ1}$4g当前位置无论是哪种情况，我们都需要从d搜索回来$_{kþ1}$找$e_{kþ1}$当前位置我们需要注意0和M的边界。

算法5更新Z匹配，报告任何集合最大匹配的，列X

其 他 g^0 f^0 þ1
当z½e的时候 -1]¼y^{kþ1}½ 做e^0 e^0 − 1
而$g^0$5M和D$_{kþ1}$½c e^0 做g^0 g^0 þ1
$f_{kþ1}$ f^0，$g_{kþ1}$ $g^0$$e_{kþ1}$ e^0

算法为O(N)并不立即明显)。 在f中的while循环^0 或g^0 这是不可避免的，因为它只需要尽可能多的迭代，因为有匹配，以报告下一次的f^0 ¼g^0*根据需要，集合最大匹配的总数以NA为界。 更多很复杂是while循环 那个
减少。 这些被使用的次数之和由N的常数倍数有界。

2.5 紧凑表示X

上述算法都使用a_k 和d_k 数组以查找匹配。 然而，这些是整数数组，元素总数等于原始数据集中的二进制值数，因此存储它们将需要更多的空间，而不是起始数据的位表示。 一些算法可以在空中应用，在相同的扫描数据中，用于生成a和d的值，但出于其他目的，特别是为了分析新的序列，我们希望更有效地存储相关信息。 下面是如何做到这一点的描述。

首先，我们注意到在匹配过程中，我们实际上没有直接使用a_k½i]索引，而是y_i... 这些 都是 a 排列 的 的 x... 价值 已确定 通过
a_k 排列指示前缀k处 ...
Burrows-Wheeler变换 ...
请参阅更接近这里给出的解释）。布伦斯和惠勒，1994年
李和杜宾（2009） ... 我们打电话
一组有序的y序列的 ... Burrows-Wheeler
变换(PBWT)。 与BW ... 一样，PBWT ...
成，因此我们可以将 ...
此外，我们还可以以期望 ...
是因为
群体遗传结构意味着存在 ...
在值中，由于连锁不平衡，...
排序顺序中具有类似前缀 ...
相同的等位基因值在下一个 ...
在y数组中的相同值。 因此 ...
在比原始数据更小的空间 ...
原始数据是运行长度编码的 ...
会反映由于连锁不平衡而共 ...

署长
2015-10-26 00:51:28
f'=w(f，z[k])和g=w(g，z[k])特别重要，这说明，要找到k+1位置的match，需要将z[k]上的值（0或1），和当前已经找到的区间[f，g)做一个alignment，对[f，g)形成进一步的过滤。过滤的是[f，g)区间上所有的prefix的k位置的值，如果有和z[k]相同的值，则会出现一个新的非0区间，否则区间的宽度将是0，即f‘=g’，这时就形成了set-maximal.

刘波
2018-10-08 00:03:44
f和g是两个指针，指的都是两行之间，而不是具体的行。如果f和g之间有z[k]，则f‘和g’不会相等，因为必然g‘>f’，否则会g‘=f’

署长
2015-10-26 12:32:32
每列因为只需要yi[k]来计算ak[i]，因此只需要记录yi[k]，这和BWT的性质是一样的。
2016-01-25 04:00:51

署长
2015-10-26 12:30:00
通过上一个性质、可以得出、z至少在dk+1[f‘]以前和yf’-1及yf‘一样，而yf’-1和yf在dk+1[f‘]-1处，分别是0和1.因此，如果z在dk+1[f‘]-1处是

要找到与新序列的匹配，如算法5中的匹配，我们还需要存储的数组u_k 和五$_k$为了评估扩展-
离子函数w_kðÞ：这些数组对应于信息
存储在所描述的索引中，Ferragina和Manzini（2000年）俗称FM指数。 考虑到PBWT，我们可以
存储所需的信息，以便以与正常字符串完全类似的方式有效地生成它们。

我们确实需要a的值$_k$ 对于报告，但是考虑到PBWT和位置FM索引中的y值，我们只能通过存储a来有效地做到这一点$_k$ 对于k的值的子集，例如每32或64个位置。 报告的匹配将比此长，因此通过使用扩展函数w扩展到a的下一个存储值$_k$ðÞ相对便宜。

最后，我们需要d的紧凑表示$_k$ 数组。
目前，提出了对赫夫曼编码的差异-
也许 只是 商店 他们 在 a 子集 的 k 至于a_k。 这里可能还有进一步改进的余地。

3 结果

在这里，我给出了模拟数据的初始结果。 一个10万个单倍型序列的数据集，覆盖20Mb部分
基因组序列使用顺序马尔可夫聚结模拟器MACS进行模拟，基本上使用命令MACS1000002e7-t0.001-r0.001(事实上，进行了更大的模拟，它崩溃了一点超过20Mb，其余的材料被削减到这一组)。陈（2009） 该数据集中有370264个隔离站点。 原始的MACS输出本质上包含以0‘s和1’s编写的单倍型序列，因此大小约为37GB。 用gzip压缩到1.02GB。

对关键算法进行了初步实现。 这对PBWT使用单字节运行长度编码，顶部位编码值，接下来的两位选择长度是以1、64还是2048为单位，其余5位给出单位数。 为了跑464次但是
这通常需要2个字节，对于运行42048但是
564k这通常需要3个字节。 所有实验都进行了
表1在苹果MacAir笔记本电脑与2.13GHz英特尔核心2Duo处理器使用一个单一的核心。 对上述10万个序列的数据集进行编码需要1070秒（用户加系统），生成一个PBWT表示，其大小为7.7MB，比原始数据的gzip压缩小130倍以上。 进一步的结果包括对数据子集的应用，表明相对增益随序列数的增加而增加，清楚地表明了该算法的非线性优点。 这一点可以通过观察清楚地看到，对于序列数中10倍的每增加一倍，PBWT用于将单倍型值存储在站点上的平均字节数只有大约两倍。 作为对真实数据的测试，对来自1000个基因组项目第一阶段数据发布1000)的染色体1数据应用了类似的措施，其中包括3007196个位点的2184个单倍型。基因组计划(2012年 该数据的gzip文件花费了303MB，而PBWT使用了51.1MB，几乎是6个较小的因子，与基于模拟数据的预期因子不远。

接下来的算法4被实现，以找到所有的集合最大匹配在模拟数据集。 正如预期的时间

表1.pbwt在不断增大的数据集上的压缩性能

序列数目	1000	10 000	100 000
序列.gz大小(KB)	10 515	105 559	1 024 614
PBWT大小(KB)	1686	3372	7698
比率.gz/PBWT	6.2	31.3	133.1
PBWT字节/站点	4.6	9.1	20.8

表2在不断增大的数据集上，pbwt的集最大匹配性能

序列数目	1000	10 000	100 000
集-最大时间)	12.1	120.3	1213.7
集-最大平均长度(Mb)	0.27	1.48	3.98

署长
2015-10-29 09:42:53
只存一部分ak、即每32或64个k存一个ak、然后对需要的ak利用uk和vk做一个动态的生成。

在序列数（2）中，取的序列是线性的，只需20分钟就能找到10万个序列中的所有最大共享子串。表

最后，评估了三种不同的方法将新序列匹配到预索引参考面板的比较性能，找到了每个新序列与参考集的所有集合最大匹配。 对于这一评估，我对模拟序列数据集进行了亚采样，以近似基因型阵列实验中的典型数据，只保留了等位基因频率为45%的位点的一小部分(10%。 减少了
站点数为5940个，约一个前3.4kb，cor-
响应于人类基因组中的850K，与标准基因分型阵列的含量相当。 表3给出了从模拟中匹配1000个序列的结果，并将它们与大小在1000到50000之间的非重叠子集进行了比较。

首先实现了一种“朴素”算法，将每个序列与面板中的所有序列进行一次比较，保持覆盖测试序列中每个碱基的最佳匹配段。 正如预期的那样，这种方法在面板的大小上需要线性的时间，在面板中每个序列~0.05s。 第二种算法5被实现，称为“索引。表3 这需要0.9个用户秒的恒定用户时间来匹配1000个序列到10000个大小的参考面板，但对于5万个参考序列，存储的u、v和d数组（在本实现中没有压缩）的大小比可用内存大，导致系统时间从0.2s增加到15s，用户时间减少到1.7s。因此，我得出结论，基于PBWT的方法可以比直接搜索方法快数百倍，并在与参考面板大小无关的时间内找到匹配，正如上面推测的那样，只要相关的索引数组适合内存。

对于索引不适合内存的情况，我们仍然可以使用PBWT数据结构来提供第三个“批处理”匹配过程，这仍然比天真的要快得多

表3。 以秒为单位匹配1000个新序列的时间，分为用户(U)和系统(S)对索引和批处理方法的贡献

序列数目	1000 000	5000	10 000	50
Naı¨有	52.1	258.9	519.2	2582.6
索引	0.9uþ0.1s	0.9uþ0.1s	0.9uþ0.2s	1.7uþ15s
批次	2.3uþ0.1	3.5uþ0.1	4.8uþ0.1	12.1uþ0.1

方法。 这使用了一个修改版本的内集算法4，它联合通过面板和组合的新序列集在一起，只是考虑新序列和旧序列之间的匹配。 如3所示，这种批处理方法所花费的时间，其内存需求较低，与站点数量无关，随着参考面板大小的增加而增加，但仍然比简单方法中的直接搜索效率高许多倍。 表在渐近上，当前实现所花费的时间将线性地依赖于M，但在不需要PBWT压缩的情况下，可以通过小心避免PBWT压缩来减少这一点。

4 讨论

在这里，我提出了一系列算法，从单倍型序列生成位置前缀数组数据结构，并将它们用于非常强的单倍型数据压缩，以及时间和空间高效的单倍型匹配。 特别是，匹配算法从直接成对比较方法所花费的搜索时间中去除M的一个因子，即被匹配的单倍型序列集的大小。 就这样
可能在数万以内找到所有最好的匹配
序列以分钟为单位，并产生实用的潜力
扩展到数百万序列的软件。 尽管
提出了二进制数据的算法，它们可以扩展
对多个等位基因的数据有点小心。

这些算法与基于后缀数组的类似算法共享其设计的各个方面，用于一般的字符串匹配，但它们是按字符串的位置构造的，从而产生了很大的差异。 一个结果是，与suf-固定阵列方法不同的是，线性时间排序算法是非平凡的，使用算法1在线性时间中构建排序位置前缀数组是简单的。这里使用的方法让人联想到)从非常大的短字符串集合生成字符串BWT。Bauer等人。 (2011

布伦斯和惠勒(1994)关于高效表示，有趣的是，原始BWT是由字符串数据压缩而不是搜索引入的，它实际上构成了bzip压缩算法的基础。 Valimaki. 等人（2007年） 以前曾探索过使用BWT压缩自指数方法来有效地压缩和搜索来自许多个体的遗传序列数据，但这不需要像这里介绍的工作中那样对可变位点进行固定的对齐，而且本质上是不同的。

这里描述的所有算法都需要精确匹配，没有错误或丢失数据。 至于序列匹配，如果更多
敏感搜索需要允许错误，仍然可以使用精确匹配算法来查找种子匹配，然后
通过直接测试加入或扩展这些内容。 这通常是
由生产软件采取的方法，但具有强大的功能
识别种子的方法是性能的关键。

顺序使用后缀/前缀数组方法的替代方法
匹配是建立一个哈希表来识别精确的种子匹配。 类似于这里描述的位置前缀数组的创建，可以为单倍型序列中的每个位置构建一组位置哈希表。 当良好调整时，基于哈希的方法可以比基于后缀数组的方法更快，因为基本操作更简单，但它们通常需要更大的内存，特别是在后缀表示可以压缩的情况下。 在标准序列匹配中不存在基因型数据的一个问题是，位置的信息含量变化很大，大多数罕见的位置信息很少，这意味着哈希词的长度需要根据序列中的位置而变化。 另一种选择是基于等位基因频率大于某些值（如10%）或在某些频率范围内的位点子集构建散列，但这将丢失导致错误种子匹配的信息。

大多数关于分析大量单倍型或基因型数据的算法的研究都集中在统计方法上，这些方法对推断很有帮助，但只扩展到几个单位-
砂位和序列；例如。 参见最近开发的加速方法可以处理多达数万个序列的数据，例如。马奇尼和豪伊（2010年）. Williams等人。(2012) Delaneau等人。 (2012 或)。 然而，这些方法提供了统计匹配方法的近似，并且仍然比这里提出的算法重得多。 已有100多万人进行了基因分型，虽然在汇集该比例的数据集方面存在后勤问题，但4100000人的基因型数据正在提供
Hoffman et al., 2011()。 一种提高效率的方法
相位和归责可能是使用计算效率高的方法，如位置前缀数组方法来播种
布朗宁和布朗宁（2007年） 匹配统计基因型算法，或在其他计算瓶颈。 例如，在他们的BEAGLE软件中，从局部单倍型序列的可变长度马尔可夫模型建立了一个概率隐藏马尔可夫模型，该模型本质上是从位置前缀数组的动态截断导出的。 虽然他们使用这个概率模型的算法将他们带到了一个不同的方向，但我建议这里描述的方法可以大大加快BEAGLE的模型构建阶段。

或者，也可以采取更直接的办法。 大多数分阶段和归责算法从整个数据集建立一个模型，然后依次线程每个序列，以提供一个新的分阶段有效地基于一系列匹配。 相反，位置前缀算法沿着所有序列共同前进。 如果我们从数据的两端开始，那么在某个位置k，我们有基于当前相位的双向匹配的信息，并且可以在一个步骤中为k处的所有序列在增量之前提出等位基因的分配
基于这一思想的k.方法可能是快速的，并与目前的方法相辅相成。

确认

我感谢HengLi和Jared Simpson给了我对基于Burrows-Wheeler变换和后缀数组的数据结构和算法的见解，以及TomislavIlicic为算法5的早期实现的开发提供了见解。这项工作是从2010年对NCBI的一次休假访问中发现的想法发展而来的。

资金来源：惠康信托（赠款098051）；ORAU。

利益冲突：无申报。

参考资料

鲍尔，M。 j.等人。 （2011年）轻型BWT结构，用于非常大的字符串集合。 In：詹卡洛，R。 和曼齐尼，G。 组合模式匹配。 斯普林格，柏林，pp。 219–231.

布朗宁，S。 R.和布朗宁，B。 L.（2007）利用局部单倍型聚类，快速、准确地进行全基因组关联研究的单倍型分期和缺失数据推断。 *Am。 J. 哼。 Genet。*，81，1084–1097.

布伦斯，M。 和惠勒，D。 j. （1994年）块排序无损数据压缩al-算法。 *技术报告124，数字设备公司，帕洛阿尔托，加利福尼亚州。*

陈，G。 k.等人。 （2009年）快速灵活地模拟DNA序列数据。 *基因组研究。*，19，136–42.

德莱诺，奥。 等人。 （2012年）数千个基因组的线性复杂性分阶段方法。 *纳特。 方法，9，179-181。*

费拉吉纳，P。 和曼齐尼，G。 （2000年）机会数据结构与应用-*问题。 In：第41届计算机科学基础研讨会论文集(FOCS2000)，加利福尼亚州雷东多海滩)。 IEEE计算机学会，pp。 390–398.*

1000基因工程。 （2012年）1092个人类基因组的遗传变异综合图谱。 自然，491，56-65。

霍夫曼，T。 j.等人。 （2011年）设计和覆盖高通量基因分型阵列优化的个人东亚，非裔美国人和拉丁美洲种族/族裔使用估算和一种新的混合SNP选择算法。 基因组学，98，422-430。

朗米德，B。 等人。 （2009年）短DNA的超快和记忆效率比对人类基因组序列。 *基因组生物。*，10，r25。

李，H和杜宾，R。 （2009）快速和准确的短读对齐与Burrows-Wheeler变换。 *生物信息学，25，1754-1760。*

李，R。 等人。 （2009）SOAP2：一种改进的用于短读对齐的超快工具。 *生物信息学，1966-1967年，25。*

马奇尼，J。 和豪伊，B。 （2010年）全基因组关联研究的基因型估算。 *纳特。 Rev。 Genet。*，11，499–511.

威廉姆斯，A。 l.等人。 （2012年）逐步收集数千个基因分型样本。 *Am。 J. 哼。 Genet。*，91，238–251.

瓦利马基，N。 等人。 （2007）压缩后缀树-基因组规模序列分析的基础。 *生物信息学，23,629-630。*