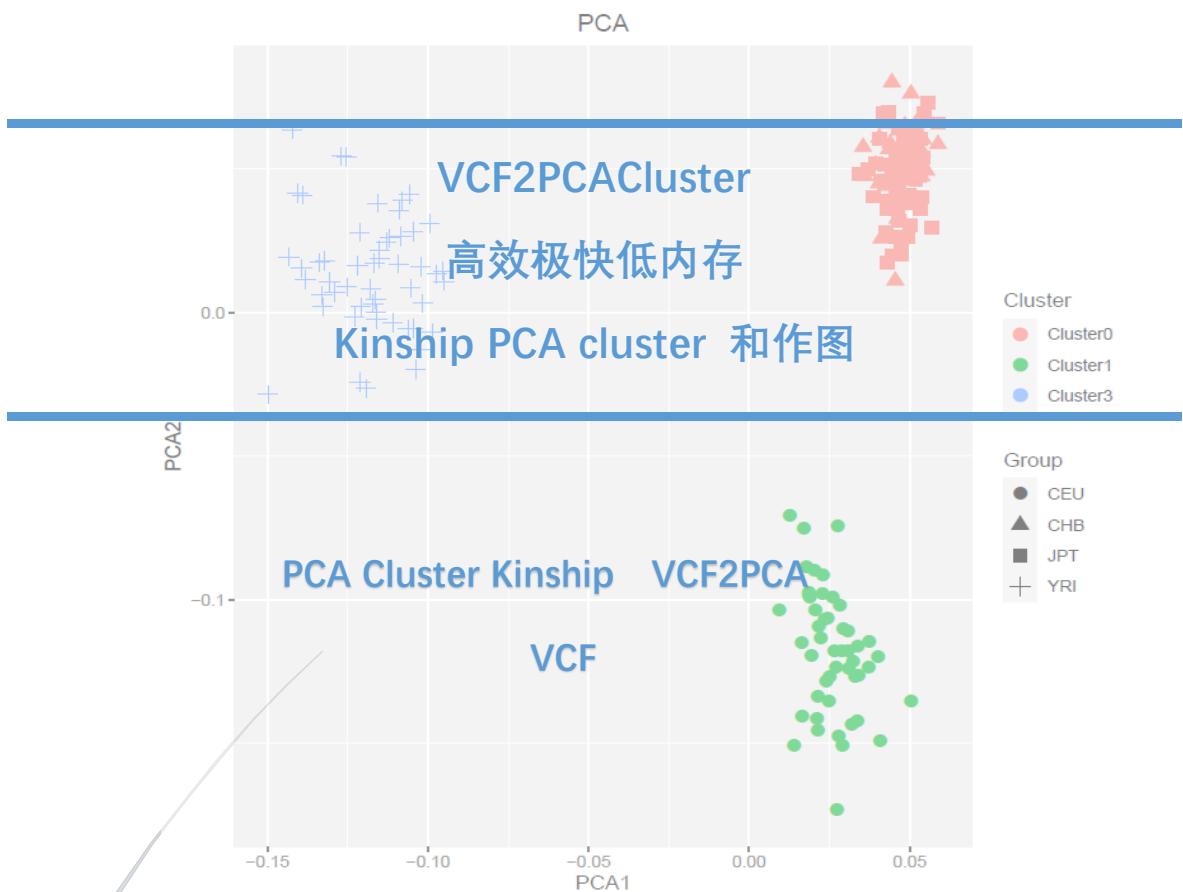


使用手册

VCF2PCAtools

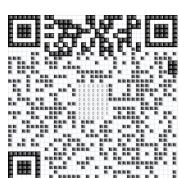


Version 1.38 使用说明文档

2023-06-01

hewm2008@gmail.com / hewm2008@qq.com

微信 打赏



QQ 入群: 125293663



微信公众号



群名称Reseqtools (tools)
群号:125293663

目录

VCF2PCATools.....	0
1.简介	1
各版本新功能和计划	2
2.应用场景示例	2
2.1 千人 VCF 重测序 SNP 基因型	2
2.1.1 EM_Gaussian_cluster 聚类	3
2.1.2 KMeans 聚类	4
2.1.3 DBSCAN 聚类	5
2.2 千人 VCF 重测序的 chr22 所有样品所有位点.....	5
2.3 3K 水稻.....	6
2.4 big Data (K Human all chr)	7
3. 下载与安装	7
3.1 下载网址	7
3.2 预先安装	7
3.3 安装	8
4. 用法和参数说明	8
4.1 VCF2PCACluster 参数	8
4.1.1 更多详细参数	10
4.1.2 其它脚本参数	10
4.2 输入文件	12
4.2.1 数据文件.....	12
4.2.2 样品分群信息(可选)	13
4.3 输出文件	13
5.实例和测评	13
5.1 准确性	13
5.1.1 Kinship : Normalized_IBS	14
5.1.2 Kinship : Centered_IBS	14
5.1.3 PCA 数值	15
5.2 性能比较	16
5.2.1 比较表	17
5.2.2 测评脚本	19
6.优势	21
7.模型及公式	21
7.1 亲缘关系矩阵	21
7.1.1 BNormalized_IBS(Yang/BaldingNicol'sKinship)	22
7.1.2 Centered_IBS(VanRaden)	22
7.1.3 IBSKinship	22
7.1.5 IBSKinshipImpute	23
7.1.5 p_dis (Distance Matrix).....	24

7.2 聚类算法	24
7.2.1 EM_Gaussian_cluster	24
7.2.2 DBSCAN.....	24
7.2.3 KMeans	24
8.常见问题	25
8.1 为什么要重复开发这个 PCA 分析软件.....	25
8.2 VCF2PCACluster 和准确性如何？	25
8.3 联系与打赏	25

1.简介

VCF2PCACluster 是基于群体 SNP 数据 VCF 格式开发的 **PCA 分析和聚类软件**，同时兼并了 Genotype 等格式软件，即只要对应的一个输入文件进来，这 PCA 和作图分组等一步到位。简单、易用和高效。

其中主要功能有：

- 1 SNP 位点过滤：如三碱基,MAF 等
- 2 5 种算法计算亲缘关系矩阵 kinship
- 3 基于 kinship 进行 PCA 分析
- 4 3 种聚类算法对 PCA 的结果进行聚类分析
- 5 基于 PCA 的结果和聚类结果进行可视化

主要亮点：

A:

一步高效生成 PCA 和聚类图。其中为了强调核心是高效低内存，一步操作，一个输入 到 PCA 结果和出图，对用户友好。

简易强调说明：

- 1 结果和 tassel 和 gcta 这三个软件相比，**结果是一样，仅精度差别**。见后面测评
- 2 功能包括：(1) 5 种 kinship 矩阵；(2) PCA 分析；(3) 聚类分析 和 (4) cluster 类别结果可视化。
- 3 一个 VCF 输入，一步到位，方便用户使用，同时可以计算子群体。
- 4 边读边算，内存剥离受位点多少的影响，即内存只受样品量影响，故上 10k 的样品的内存也不过 1~2G 左右。
- 5 聚类算法
 - 5.1 **EM-Gaussian : EM 算法与混合高斯模型**，[具体见这儿介绍](#)。
 - 5.1 Kmean 聚类分析，并找出最佳 K 值，和 Structure 和 K 值一样。作图以此染色。
 - 5.3 DBSCAN 聚类算法。参数可传。[见这儿](#)。
- 6 提作作图小脚本，可以用这个脚本优化作图细节等。

注：程序提供了两个实例数据，

Example1：小数据，提取重测序千人 VCF 部分位点，作为输入文件。具体可以见 example1 里面的用法。

Example2：在数据，重测序千人的整条 chr22 染色体 VCF 作为输入文件。具体可以见 example2 里面的用法，主要用于测评等待时间和内存。

各版本新功能和计划

- A. 作图初步依赖于 R 和 ggplot2 包, **后面看须要可以开发独立的 svg 画图功能和软件**
- B. **1.20 的版本** 添加了新的聚类算法: DBSCAN 聚类算法, 相关[算法介绍可以网上搜](#) Kmean 和 DBSCAN 算法的差别, 网上也有很多教程, 可以搜看, 如[这儿页址](#):。
- C. **1.22 的版本** 添加了新的聚类算法: **EM-Gaussian**: EM 算法与混合高斯模型, [相关说明](#)
- D. **1.35 的版本**, 核心在于提高效率速度, 改造 genotype 的数据结构, 同时使用 openmp 多线程提速
- E. **1.36 的版本**, 添加了 新的计算 kinship 的方法 2:Centered_IBS(VanRaden)
- F. **1.38 的版本**, 添加了纯 SVG 作图, 响应 A, 可以不依赖 R 也可以直接出图。

2. 应用场景示例

初步开发主要针对群体重测序 vcf 文件进行 PCA 分析, 聚类和作图。

2.1 千人 VCF 重测序 SNP 基因型

本程序测试数据集取自千人基因组测序 dbSNP 数据库, 从 22 号染色体中随机挑 **1194** 个位点, 包括 CEU (49), CHB (46), JPT (56) 和 YRI (52) 共 **203** 个样品。

用法如下:

```
VCF2PCACluster -InVCF      Khuman.vcf -OutPut OUT
```

其中可以用 **-InSampleGroup** 输入已知样品和分群信息, 查看是否和聚类结果一致。

```
VCF2PCACluster -InVCF Khuman.vcf -OutPut OUT -InSampleGroup  
pop.info
```

其中, 文件 pop.info 的格式为两列, 第一样为样品名, 第二列为分群名。

结果: pca 和用 **gatc** 的结果是一致的(见后面), 这儿不深研究, 已经确保同一数据, pca 的结果是相同的了。

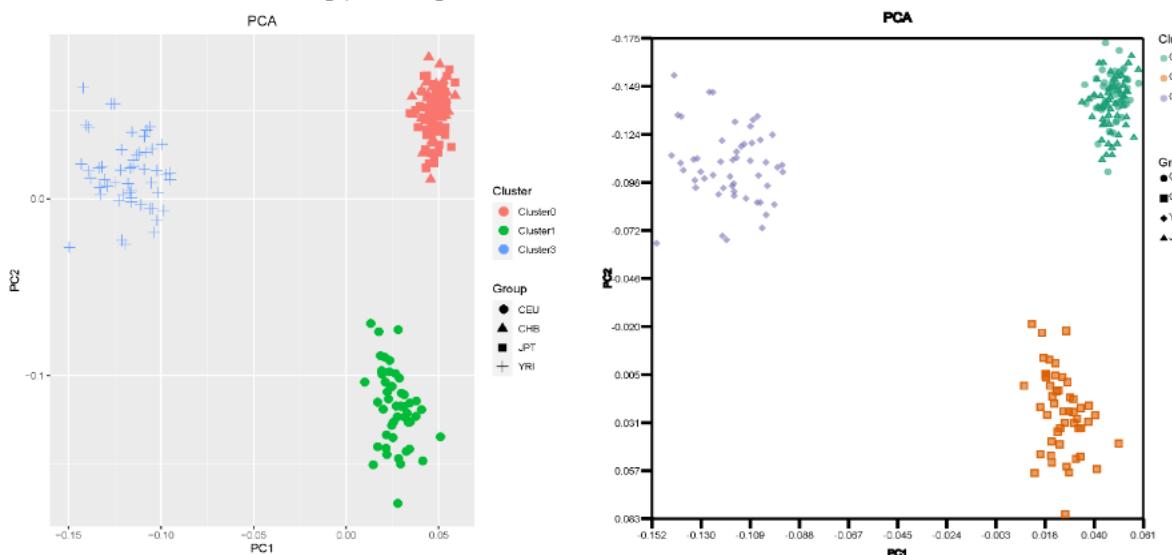
2.1.1 EM_Gaussian_cluster 聚类

EM 的算法[见这](#):

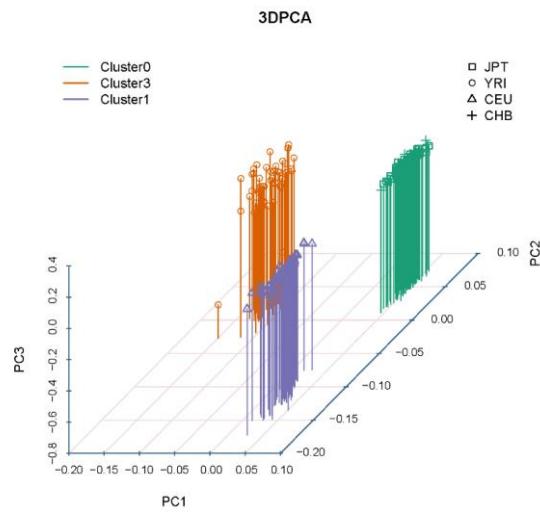
- 1 算法先用调用了 k-mean 算法找到 best K, 用户也可以给参过参数 -BestKManually 指定具体的 K 值
- 2 默认初始值, 迭代次数 (默认 1000, 可以通过参数 -Iterations 传), 和判断收敛的参数 (默认自动, 不大于 1e-10, 用户可以通过-Epsilon 传递)
- 3 调用 EM 算法, 找到最值聚类结果。

同理在 K=4 的情况下, Kmean 和 DBSCAN 是一定会分成 4 组的, 但通过 EM 算法, 结果是 3 组, 这儿也是我感觉 EM 算法更好的一个原因, 故设这个聚类为默认的方法。下面是 example1 聚类的结果比较

PCA and EM Gaussian clustering plot using PC1 and PC2



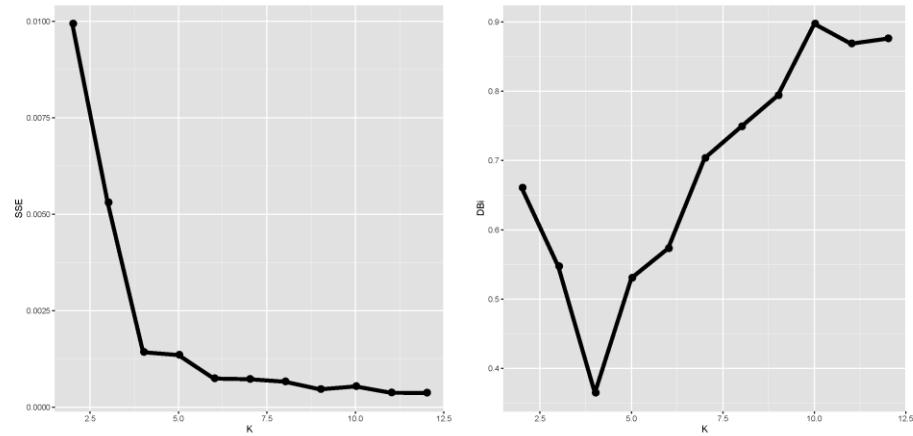
PCA and EM Gaussian clustering plot using PC1,PC2, and PC3



2.1.2 KMeans 聚类

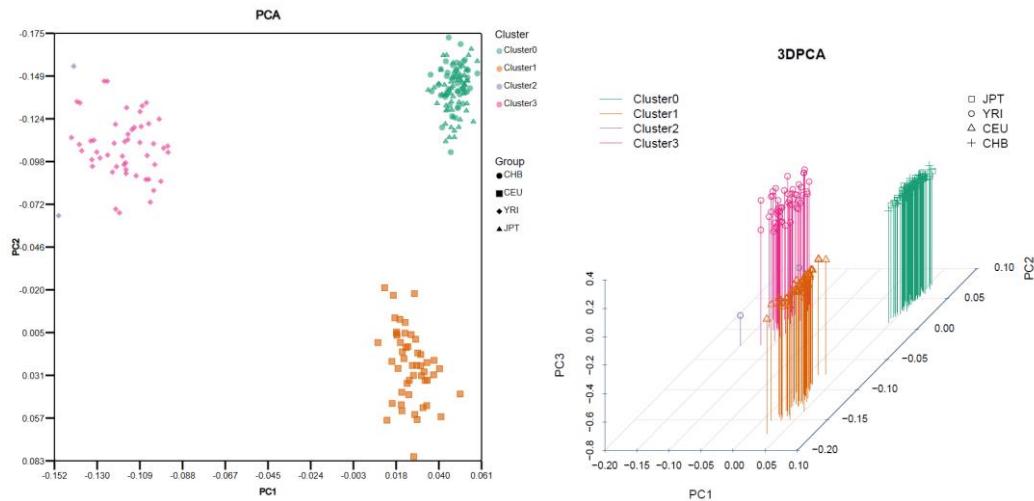
KMeans 聚类算法可以网上搜学习之。

结果： 软件根据 K 和 SSE 的关系，认为最佳的聚类为 K=4, 4 后面之后就平缓了 or 高了，软件可以取 best K=4. 软件默认输出了 K3-12 的结果，用户若认为聚类不好，要用 K=3 可以从文件 OUT.cluster 里面挑 K=3 的结果出来即可。



其中 best K 一般认为 是 SSE 第一次开始平滑 变缓那时对应的 K. 并不是最小值，在 1.07 的版本后更新多计算 DBI 的功能，软件根据 K 和 Davies–Bouldin index (DBI) 的关系, DBI 最小值就是 best K. 其中 [SSE 和 DBI 的相关说明和介绍可以点击打开查看](#)。

那么 K=4 的结果和 PCA 画图如下：



如图 xxx, 主要多出来的两个聚类主要是在 YRI, 相当 wild 种 聚类看起来较乱 (其实当在 pca3 时 YRI 能分出来)。若要手动把 K=3 拿出来重新画图 (软件提供了 ploteig 画图软件, 有很多参数, 对默认的图不满意, 可以单独在 kinship 的结果上重新用这个软件快速作图), 脚本



如下：

```
./../bin/VCF2PCA      -InKinship    OUT1.Normalized_IBS.Kinship   -InSampleGroup  
sample.group      -OutPut OUT2  -BestKManually 5  
perl  ./../bin/ploteig  -InPCA    NewOUT.eigenvec   -OutPrefix NewFig  
(aa.pl 见 example1)
```

另外 **ploteig** 有 **-ColShap** 的参数，可以把上面的 颜色和 形状 对换一下。

有时边缘点在聚类分析中有时因为初始散点质心而有所影响，即找 best K 时会有所变化，在 1.08 后的版本提供了可以 repeat 多次聚类的参数 (**-RandomCenter**) 查看聚类结果。用户可以多次使用 看看各种的效果。

```
./VCF2PCACluster  -InKinship  in.BaldingNicolis.Kinship      -RandomCenter  
-OutPut  NewOUT
```

2.1.3 DBSCAN 聚类

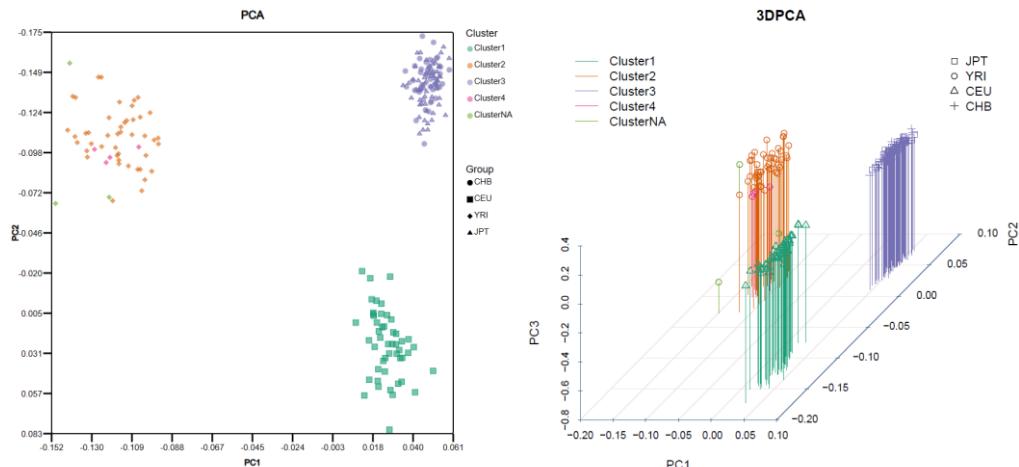
DBSCAN 算法主要有两个参数：

领域半径：**Eps**；

成为核心对象的在领域半径内的最少点数：**MinPts**。

这两个参数均可以用户自己传入，程序默认 Eps 采用了拐点法找最优 Eps 的值，**MinPts** 默认为 4。[具体可以见这儿说明。](#)

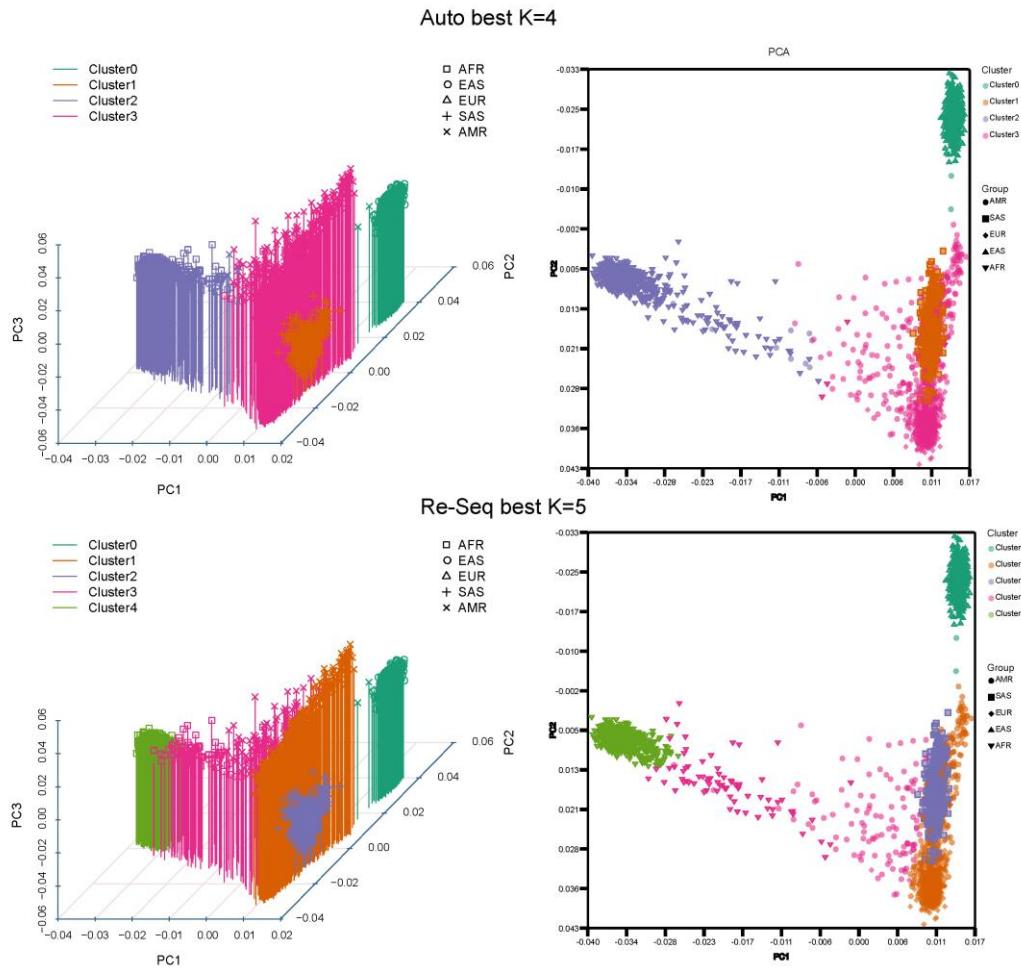
同上的数据，聚类共得到 4 类，同时多了几个 outlier 的样品，结果如下：



2.2 千人 VCF 重测序的 chr22 所有样品所有位点

应用了千人 chr22 的所有位点所有样品后作图如下(见 example2)：在默认 best K=4

和自己设定 best K=5 的结果如下：



2.3 3K 水稻

此外，我们还对包含 3k 个样本和 29 M 个 SNPs 的大规模水稻进行了测试。我们展示了 VCF2PCACluster 能够高效地产生结果，仅需 181 分钟和 0.1 GB 内存。相比之下，PLINK2 花费了 100 分钟，并消耗了大量的 257 GB 内存。

Software	Peak memory	Wait time	CPU
VCF2PCACluster	~0.1G	181min	40 threads
Plink2	~257G	100min	40 threads
Gcta64	~257G	283min	40 threads /1 threads

See detail as follows:

```
### download the 3K Rice 29M data ###
#wget -c https://s3.amazonaws.com/3kricegenome/reduced/NB_final_snp.fam.gz ./ 
#wget -c https://s3.amazonaws.com/3kricegenome/reduced/NB_final.snp.bim.gz ./
```

```

# wget -c https://s3.amazonaws.com/3kricegenome/reduced/NB_final.snp.bed.gz ./  

### change bhe bed to vcf ###  

#gzip -d *.gz  

#plink --bfile NB_final.snp --recode vcf-iid --out NB_final.snp # bed back to vcf 259G mem  

### Run VCF2PCACluster with 40 cpu (0.1G 181min)###  

time ./bin/VCF2PCACluster -InVCF NB_final.snp.vcf -OutPut VCF2PCA -BestKManually 4 #0.1G  

181min 40cpu  

### Run plink2 with 40 cpu (257G mem 100min)  

time plink2 --vcf NB_final.snp.vcf -out plink --allow-extra-chr --pca ## 257G mem 100min 40cpu  

### Run gcta (Result same with VCF2PCACluster) ###  

time plink2 --vcf NB_final.snp.vcf -out Rice3K --allow-extra-chr --make-bed ## 257G  

mem 60min 40cpu  

time gcta64 --bfileRice3K --make-grm --out out.grm ## 3.9G 223min 1cpu  

time gcta64 --grm out.grm --pca 10 --out gctaPCA ## 0.1G 1min 1cpu

```

2.4 big Data (K Human all chr)

我们对来自 1000 基因组计划的染色体 1-22 的 SNP 数据进行了一个包含 81.2 百万个 SNP 的大型数据集的测试。结果显示，使用 VCF2PCACluster 运行时，内存使用极低（约为 0.1 GB），在使用 8 个线程的情况下，成功地完成了约 610 分钟的计算。相反，PLINK2 需要更大的内存使用量 (>200 GB)，并未能完成任务。

3. 下载与安装

3.1 下载网址

这主要是防止大家还在用 beta 版本，还在用可能存在 bug 的程序，强迫大家定期更新。
/hwfssz4/BC_PUB/Software/08.Centos7/VCF2PCACluster-*

里面有 example1-2

持续更新在这：<https://github.com/hewm2008/VCF2PCACluster> 下载时记得加个星星哦

3.2 预先安装

VCF2PCACluster 适用于 Linux/Unix/macOS 系统。在安装之前，请先安装以下使用条件：

- 1) R : [R](#) with [ggplot](#) and [scatterplot3d](#) package is recommended(没有 R 只能出 2D 图)
- 2) g++ : g++ with [-std=c++11](#) > 4.8+ is recommended
- 3) openmp 库要先安装，多线程。 [-fopenmp](#)
- 4) zlib : [zlib](#) > 1.2.3 is recommended

3.3 安装

使用者可采用以下直接 chmod 755 运行:

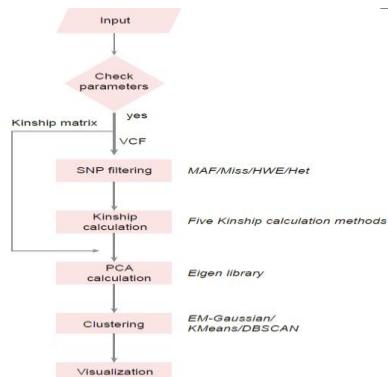
1)

```
git clone https://github.com/hewm2008/MingPCACluster  
cd MingPCACluster; chmod 755 bin/*  
.bin/MingPCACluster -h # 直接运行
```

4. 用法和参数说明

4.1 VCF2PCACluster 参数

程序 VCF2PCACluster 很简单，一个输入和一个输出。具体如下。



```
[heweiming@cngb-ologin-25 bin]$ ./MingPCACluster

Usage: MingPCACluster -InVCF in.vcf.gz -OutPut outPrefix [options]

      -InVCF      <str>      Input SNP VCF Format
      -InKinship   <str>      Input SNP K Kinship File Format
      -OutPut     <str>      OutPut File Prefix(Kinship PCA etc)

      -KinshipMethod <int>    Method of Kinship [1-5],default [1]
                               1:Normalized_IBS(Yang/BaldingNicolsKinship)
                               2:Centered_IBS(VanRaden)
                               3:IBSKinshipImpute 4:IBSKinship 5:p_dis

      -ClusterMethod <str>    Method For Cluster[EM/Kmean/DBSCAN/None] [EM]

      -help          v1.38     Show more Parameters and help [hewm2008]
```

只要一个 VCF 文件作为输入即要可
一个输出文件前缀。

其中对 VCF 可以传多一个文件，只计算里面部分样品的 pca 的结果。

-Method 是生成样品之间 kinship 的算法，也是 pca 依赖的计算矩阵。

4.1.1 更多详细参数

上面是简要输入，更多详情可以 **-h** 查看。

```
./bin/VCF2PCA -h

More Help document please see the pdf/doc help Para [-i] is show for [-InVCF], Para [-o] is show for [-OutPut]

Usage: MingPCACluster -InVCF in.vcf.gz -OutPut outPrefix [options]

    -InVCF      <str>      Input SNP VCF Format
    -InKinship   <str>      Input SNP K Kinship File Format
    -OutPut     <str>      OutPut File Prefix(Kinship PCA etc)

    -KinshipMethod <int>    Method of Kinship [1-5],default [1]
                            1:Normalized_IBS(Yang/BaldingNicolisKinship)
                            2:Centered_IBS(VanRaden)
                            3:IBSKinshipImpute 4:IBSKinship 5:p_dis

    -ClusterMethod <str>    Method For Cluster[EM/Kmean/DBSCAN/None] [EM]

    -help        v1.38      Show more Parameters and help [hewm2008]

    InFile:
        -InGenotype   <str>      InPut Genotype File for no VCF file
        -InSubSample   <str>      Only keep samples from subsample List for PCA[ALLsample]
        -InSampleGroup <str>      InFile of sample Group info,format(sample groupA)

    SNP Filtering:
        -MAF          <float>    Min minor allele frequency filter [0.001]
        -Miss         <float>    Max ratio of miss allele filter [0.25]
        -Het           <float>    Max ratio of het allele filter [1.00]
        -HWE          <float>    Exact test of Hardy-Weinberg Equilibrium for SNP Pvalue[0]
        -Fchr          <str>      Filter the chrX chr[chrX,chrY,X,Y]
        -KeepRemainVCF          keep the VCF after filter

    Clustering:
        -RandomCenter          Random diff-center to Re-Run Cluster for Kmean
        -BestKManually <int>    manually set the Best K (Num of Cluster) (auto)
        -BestKRatio    <float>    Get the best K Cluster by data-SSE Ratio[0.15]
        -MinPointNum   <int>    Minimum point number of D-cluster[4]
        -Epsilon        <float>    Epsilon for DBSCAN_Distance/EM_convergence (auto)
        -Iterations     <int>    iterations number for EM clustering[1000]

    OutPut:
        -PCnum       <int>      Num of PC eig [10]
```

参数一看即明，不过多说明。中间主要是过滤 **SNP** 的参数 和聚类相关参数

下面一般参数 如设 **K** 的最大值，**cluster**，可以理解 **structer** 的 **K** 值

4.1.2 其它脚本参数

程序也提供了作图软件 perl 作图脚本（这个脚本后面将会优化更动较大，主要是最近时间较忙），作图脚本的简要参数说明如下：



```
[heweiming@cngb-ologin-25 bin]$ ./bin/ploteig

Version:1.38          hewm2008@gmail.com

Usage: ploteig -InPCA pca.eigenvec -OutPrefix Fig

Options

-InPCA      <s> : InPut File of PCA
-OutPrefix   <s> : OutPut file prefix

-help        : Show more help [hewm2008]
```

即输入 VCF2PCACluster 的 pca 结果即可。更详细 help 可以-h 查看，主要作图参数，如

```
[heweiming@cngb-ologin-25 bin]$ ploteig -h

Version:1.38          hewm2008@gmail.com

Usage: ploteig -InPCA pca.eigenvec -OutPrefix Fig

Options

-InPCA      <s> : InPut File of PCA
-OutPrefix   <s> : OutPut file prefix

-help        : Show more help [hewm2008]

-columns    <s> : the columns to plot a:b [3:4]
-ColShap    <s> : colour <=> shape for cluster or subpop
-border     <i> : how to plot the border (1,2,4,8,3,31) [3]
-title      <s> : title (legend) [PCA]
-keystyle   <s> : put key at top right default(in) [outside]box [outside]
-pointsize   <i> : point size for plot [3]

-BinDir     <s> : The Bin Dir of gnuplot/R/ps2pdf/convert [$PATH]
```

画图的点的大小等



4.2.2 样品分群信息(可选)

有时已经知道样品的分群信息了，要看已知分群信息和聚类的差异。传入和各聚类结果比较。
格式很简单，就两列。 第一列 样品名 第二列 分群名。 如下截图

NA06984	CEU
NA06986	CEU
NA06994	CEU
NA07048	CEU
NA07051	CEU
NA07347	CEU
NA07357	CEU

用 -InSampleGroup 输入上面的文件

另顺说 -InSubSample 只要样品名一列就行，这儿是计算某些样品之间的 pca，只算子群体的 PCA

4.3 输出文件

主要结果有如下文件：

输出文件	说明
out.kinship	输出的亲缘矩阵，各样品的两两关系
out.eigenvec	输出最优聚类和 pca 结果
out.eigenval	输出 pca 结果的特征向量
out.PCA1_PCA2.pdf	输出按 cluster 染色后的 pca 1 2 图
out.K.pdf	输出 cluster K 图
out.cluster	输出的各种 K 的 cluster 聚类结果

示例图见上面应用场景给的图。示例图和格式当一看即明，相关图可以见 example 1 和 2

5. 实例和测评

5.1 准确性

PCA 的结果核心在于算 kinship 的结果，我们这儿主要查看 kinship 的结果和 PCA 的结果是否准确。在以 example1 里面的 Khuman.vcf.gz 测试数据。(同时我们也用更多数据 3K Rice 共)

kinship 以 另外两个软件 **tassel** 和 **gcta64** 的这两个软件作比较,结论是 kinship 是一致的。

PCA 和 **gcta64** 的结果比较。结论是 PCA 是一致的.

其实方法等也是经得住考验的，在这找不到相关的软件作比较，不列。



5.1.1 Kinship : Normalized_IBS

结论是： 本软件的 Normalized_IBS 和 **tassel** 的 Normalized_IBS 是一样的，和 **gcta64** 的 Yang 方法算出来的 kinship 是一样的。

A 本软件的：按默认的参数结果,如下截图：

203					
NA06984	0.951454	0.207497	0.046003	0.071316	
NA06986	0.207497	1.469973	0.112982	0.028815	
NA06994	0.046003	0.112982	0.822367	0.047398	
NA07048	0.071316	0.028815	0.047398	0.663251	
NA07051	0.080176	0.016679	0.064628	0.075370	
NA07347	0.027569	0.035410	0.057977	0.080284	
NA07357	0.033770	0.018793	0.066452	0.062673	
NA10851	0.053527	0.026677	0.064728	0.095548	
NA11829	0.080034	0.055461	0.095494	0.045353	
NA11831	0.078578	0.113936	0.108653	0.130886	
NA11843	0.090061	0.006014	0.045330	0.097095	
NA11881	0.099786	0.012721	0.052846	0.034093	
NA11893	0.048694	0.003612	0.074860	0.035699	
NA11919	0.038211	0.144765	0.073593	0.069128	
NA11930	0.150488	0.116373	0.047459	0.014196	
NA11932	0.116248	0.081502	0.027814	0.107138	
NA11992	0.062731	0.045931	0.057637	0.094080	
NA12003	0.067879	0.062158	0.023858458	0.10984979	

VCF2PCA -InVCF Khuman.vcf.gz -OutPut OUT -KinshipMethod 1

而 **tassel** 的 Normalized_IBS 和 **gcta64** 的 Yang 方法

##Matrix_Type=Normalized_IBS	203				
NA06984	0.95145607	0.20749767	0.046003226	0.07131609	
NA06986	0.20749767	1.4699764	0.11298213	0.028815197	
NA06994	0.046003226	0.11298213	0.82236797	0.047398437	
NA07048	0.07131609	0.028815197	0.047398437	0.6632524	
NA07051	0.08017568	0.0166794	0.064628385	0.07537005	
NA07347	0.027568767	0.035409804	0.057976812	0.08028415	
NA07357	0.033770025	0.01879272	0.06645211	0.06267319	
NA10851	0.05352711	0.026677167	0.064728	0.09554797	
NA11829	0.08003405	0.055461053	0.09549438	0.045352943	
NA11831	0.0785781	0.11393575	0.108653106	0.13088617	
NA11843	0.090060584	0.0060143895	0.04532959	0.09709576	
NA11881	0.099785596	0.0127210645	0.052845903	0.034092586	
NA11893	0.048693813	0.003611813	0.07486027	0.035699457	
NA11919	0.038210884	0.14476547	0.07359314	0.06912766	
NA11930	0.15048817	0.116373464	0.047458738	0.014195871	
NA11932	0.11624839	0.081501536	0.02781394	0.10713782	
NA11992	0.06273149	0.045930885	0.057637252	0.09407976	
NA11994	0.06787799	0.06215866	0.023858458	0.10984979	
NA12003	0.1274933	0.0580304	0.060805053	0.054646336	

tassel-5.2.52/run_pipeline.pl -fork1 -vcf Khuman.vcf.gz -KinshipPlugin -method **Normalized_IBS** -endPlugin -
export kinship.txt -exportType SqrMatrix
gcta64 --make-grm-alg 0 ## Yang

5.1.2 Kinship : Centered_IBS

结论是： 本软件的 Centered_IBS 和 **tassel** 的 Centered_IBS 是一样的，和 **gcta64** 的 VanRaden 方法算出来的 kinship 是一样的。

A 本软件的：按默认的参数结果,如下截图：

203					
NA06984	1.039789	0.282751	0.089736	0.261001	
NA06986	0.282751	1.076777	0.196358	0.182553	
NA06994	0.089736	0.196358	1.228332	0.068854	
NA07048	0.261001	0.182553	0.068854	0.989213	
NA07051	0.266601	0.117650	0.171396	0.148778	
NA07347	0.089692	0.116999	0.153119	0.218629	
NA07357	0.052400	0.106146	0.256833	0.075583	
NA10851	0.136101	0.172221	0.120212	0.300289	
NA11829	0.199571	0.174001	0.236559	0.143438	
NA11831	0.149038	0.185158	0.238903	0.339665	
NA11843	0.166013	0.096378	0.185375	0.083441	
NA11881	0.230047	0.107535	0.187719	0.147475	
NA11893	0.153075	0.048189	0.101935	0.096942	
NA11919	0.188457	0.295080	0.084439	0.220453	
NA11930	0.161888	0.224447	0.101935	0.044065	
NA11932	0.223144	0.188761	0.075062	0.272765	
NA11992	0.173045	0.182727	0.271724	0.231480	
NA11994	0.224360	0.163538	0.076278	0.203478	

VCF2PCA -InVCF Khuman.vcf.gz -OutPut OUT -KinshipMethod 2

而 tassel 的 Centered_IBS 和 gcta64 的 VanRaden 方法

##Centered_IBS.SumPk=113.47039499176373					
##Matrix_Type=Centered_IBS					
203					
NA06984 1.0397892	0.28275055	0.08973562	0.26100054		
NA06986 0.28275055	1.0767771	0.19635832	0.18255298		
NA06994 0.08973562	0.19635832	1.2283326	0.068853885		
NA07048 0.26100054	0.18255298	0.068853885	0.98921293		
NA07051 0.26660082	0.1176503	0.17139576	0.14877753		
NA07347 0.08969222	0.116999105	0.15311885	0.21862932		
NA07357 0.052400317	0.10614583	0.2568329	0.07558296		
NA10851 0.13610087	0.17222063	0.12021166	0.30028948		
NA11829 0.19957092	0.17400058	0.23655891	0.1434377		
NA11831 0.149038	0.18515775	0.23890325	0.3396652		
NA11843 0.16601254	0.096377864	0.18537481	0.08344071		
NA11881 0.23004694	0.10753501	0.18771914	0.14747511		
NA11893 0.15307543	0.048189238	0.10193472	0.09694221		
NA11919 0.18845718	0.29507992	0.08443922	0.22045268		
NA11930 0.16188832	0.22444667	0.10193473	0.044065014		
NA11932 0.22314425	0.18876107	0.07506197	0.27276552		
NA11992 0.17304549	0.18272662	0.2717236	0.2314796		
NA11994 0.22435984	0.16353801	0.07627755	0.20347811		
NA12003 0.2600889	0.15520267	0.17369668	0.21276852		
NA12005 0.23556042	0.3774346	0.2989871	0.19705296		
NA12043 0.048579946	0.11995119	0.023877863	0.1246398		
NA12045 0.13966076	0.22865778	0.21190026	0.18928201		

tassel-5.2.52/run_pipeline.pl -fork1 -vcf Khuman.vcf.gz -KinshipPlugin -method Centered_IBS -endPlugin - export kinship.txt -exportType SqrMatrix
gcta64 --make-grm-alg 1 ## VanRaden

5.1.3 PCA 数值

都在默认的参数数据，跑出来的结果，主要比较 PCA1, PCA2 PCA3 前三个主要成分。

结论是：PCA 的结果是一致的。

下面是本软件在默认的情况下跑出的结果：



SampleName	Group	Cluster	PCA1	PCA2	PCA3	PCA4	PCA5	PCA6	PCA7	PCA8
NA06984	CEU	Cluster1	0.021051	-0.141331	0.00170528	-0.00326334				
NA06986	CEU	Cluster1	0.0503198	-0.134854	-0.00612967	0.00826446				
NA06994	CEU	Cluster1	0.0265172	-0.117603	-0.00314609	0.0124698				
NA07048	CEU	Cluster1	0.0373148	-0.114575	0.00107168	0.0109161				
NA07051	CEU	Cluster1	0.02924	-0.110075	0.00745119	0.00683107				
NA07347	CEU	Cluster1	0.0270689	-0.123653	0.000800091	0.00411394				
NA07357	CEU	Cluster1	0.019393	-0.119408	-0.0257179	0.00185087				
NA10851	CEU	Cluster1	0.0259301	-0.0990199	0.00424188	0.00498169				
NA11829	CEU	Cluster1	0.0213705	-0.145019	-0.000675612	-0.00110612				
NA11831	CEU	Cluster1	0.0339703	-0.126462	0.0104739	0.027259				
NA11843	CEU	Cluster1	0.0164779	-0.140548	0.0191187	-0.0037902				
NA11881	CEU	Cluster1	0.0228064	-0.0981855	0.00803481	-0.00153922				
NA11893	CEU	Cluster1	0.028189	-0.101798	-0.000516194	0.00697281				
NA11919	CEU	Cluster1	0.0336541	-0.141849	0.00791679	0.0125857				
NA11930	CEU	Cluster1	0.018898	-0.0994452	-0.00214388	0.0163885				
NA11932	CEU	Cluster1	0.0239667	-0.128468	0.00265258	-0.00487042				
NA11992	CEU	Cluster1	0.0323722	-0.121468	-0.00901025	0.0171458				
NA11994	CEU	Cluster1	0.0215832	-0.109444	-0.000131104	-0.00852011				
NA12003	CEU	Cluster1	0.0139342	-0.150668	0.0174059	0.00588741				

```

0 NA06984 0.021051 -0.141331 0.00170528 -0.00326334
0 NA06986 0.0503198 -0.134854 -0.00612967 0.00826446
0 NA06994 0.0265172 -0.117603 -0.00314609 0.0124698
0 NA07048 0.0373148 -0.114575 0.00107168 0.0109161
0 NA07051 0.02924 -0.110075 0.00745119 0.00683107
0 NA07347 0.0270689 -0.123653 0.000800091 0.00411394
0 NA07357 0.019393 -0.119408 -0.0257179 0.00185087
0 NA10851 0.0259301 -0.0990199 0.00424188 0.00498169
0 NA11829 0.0213705 -0.145019 -0.000675611 -0.00110613
0 NA11831 0.0339703 -0.126462 0.0104739 0.027259
0 NA11843 0.0164779 -0.140548 0.0191187 -0.00379019
0 NA11881 0.0228064 -0.0981855 0.00803481 -0.00153922
0 NA11893 0.028189 -0.101798 -0.000516194 0.00697281
0 NA11919 0.0336541 -0.141849 0.00791679 0.0125857
0 NA11930 0.018898 -0.0994452 -0.00214388 0.0163885
0 NA11932 0.0239667 -0.128468 0.00265258 -0.00487042
0 NA11992 0.0323722 -0.121468 -0.00901025 0.0171458
0 NA11994 0.0215832 -0.109444 -0.000131104 -0.00852011

```

gcta64 --bfile file --make-grm GRM

gcta64 --grmGRM --pca 4

画出来的图的位置也全是一致的。

此外，example2 的 chr22 来自 1000 基因组计划（从 <ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502> 下载）具有 1,055,401 个 SNP。example3 水稻 (<https://s3.amazonaws.com/3kricegenome>)，共有 29M 个 SNP。它还表明，VCF2PCA 和 GCTA64 在亲属关系和 PCA 结果方面具有一致性。

5.2 性能比较

性能测评中，由于在不同系统的环境不同，使之编译时的优化程序不同，特别是 gcc 的版本和 opemmp 的版本的不同，会影响编译时的优化程度，同时 opemmp 会根据当前系统可以用的线程，动态用到不同的线程。下面我都是在同一节点 test 的，但总体来说，我们主要比较 是否 使用到的等待时间当在同一个数量级里面。

To test the accuracy and the efficiency of VCF2PCACluster, we used data of this



web site(<https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502>) to test follow software, only used the **chr22** (minimal chromosome SNP database) (2504 sample with 1,055,401 SNP numbers) to test these software.

5.2.1 比较表

下面是用了千人数据 chr22 号数据(2504 个样品 1M 多位点)来测评来进行比较，其中 Tassel 和 gapit3 用 chr22 的实际数据运行时间，内存远大于 150G，同时等待时间超过 300min 以上，可以认为这两个软件不适于做大样本的 kinship 和 PCA 的分析。

另外 gcta 不支持 vcf 直接读入，须要对 vcf 进行过滤和转格式为 plink 的输入，这儿用到了 plink 把 vcf 转格式，多线程等待时间约这 2.27min

软件	语言	过滤和转格式	子群		计算机	等待时间	结果	聚类	可视化结果
			个体	内存					
tassel	Java	没（要额外处理重 复点）	没	>180G	>300min+	Kinship/PCA	否	否	
gapit3	R	没（要额外处理重 复点）	没	>150G	>300min	PCA	否	有	
gcta (1.93.0)	c/c++	没，结合 plink 过滤： (maf)	有	~1.5G	~7min (16 threads)	Kinship/PCA	否	否	
VCF2PCA 1.38	c/c++	有 (maf miss hwe)	有	~0.1G	~12min (8 threads)	Kinship/PCA Cluster/Fig	是	是	
Plink2 (v2.00a2LM)	c/c++	有 (maf miss hwe)	有	~1.5G	~2.47min (16 threads)	Kinship/PCA	否	否	

注 等待时间不是 cpu 时间,是用户等待的时间。

上面是只用了 chr22 的数据来，我们合并 chr1-22 的 vcf (81271745 位点) 到一起得到非压缩的 vcf 文件 (大小为 728G)，重新运行了 plink 和 VCF2PCA，plink 必须重新 bed fam bim 文件，但这一步 Error: File write failure. 无法进行 pca。

结果时间和内存如下

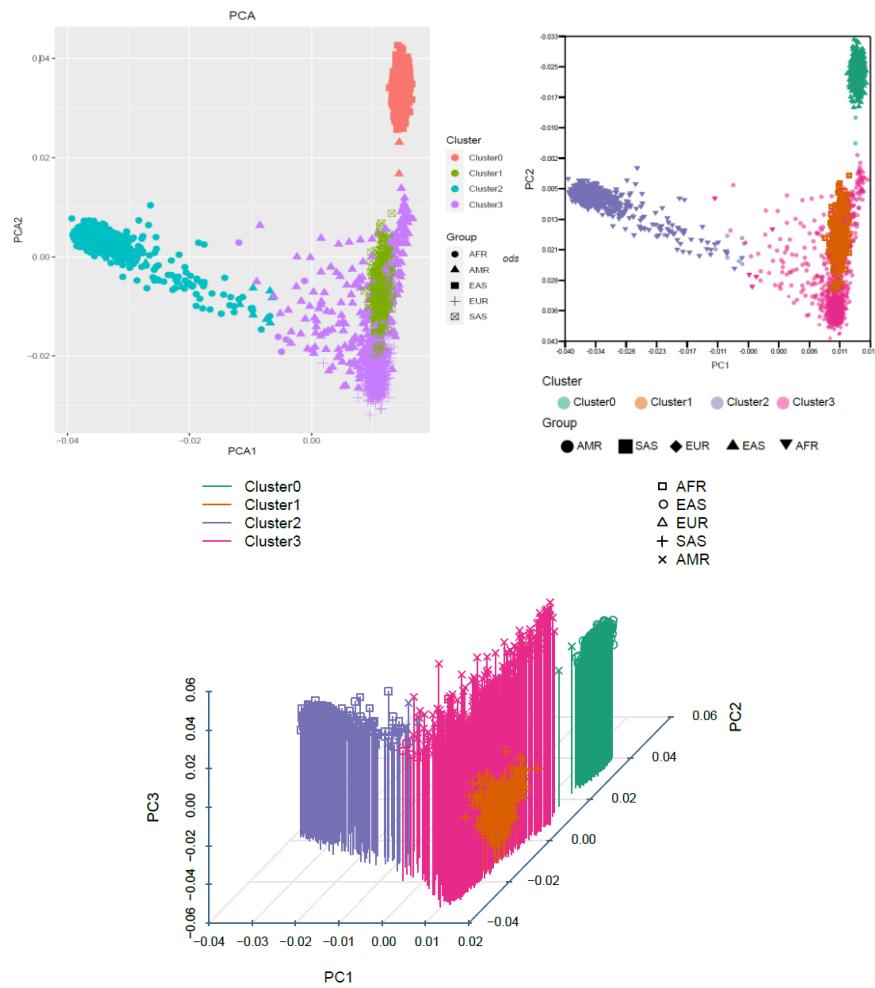
程序	内存	等待时间



Plink	200G	Error: File write failure.
VCF2PCA 1.38	0.1G	~600min : 8 thread

结合 测评的相关脚本如下，VCF2PCA 在计算时用到了多线程，但在读 vcf 时没有用到多线程，所以等待时间多了一点。另外 VCF2PCA 的内存已经剥离了位点的多少的影响，即位点再多内存的 peak 值也不会增加。结合 认为 VCF2PCA 一步到位，聚类画图等，更适用新人使用，时间和内存上也十分有可以接受。

一步到位，到如下图：



测评后的结果图如下：聚类结果和已知的样品分类是一致的(一致性为 0.995)。

5.2.2 测评脚本

主了方便，其它人测评比较，下面是我在我这儿跑的主要脚本：

5.2.2.1 VCF2PCA

VCF2PCA : <2m ~12.5min 之间(8 线程)

```
echo Start Time :  
date  
#wget -c https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/ALL.chr22.phase3_shapeit2_mvncall_integrated_v5b.20130502.genotypes.vcf.gz  
#wget -c https://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502/integrated_call_samples_v3.20130502.ALL.panel  
cut -f 1,3 integrated_call_samples_v3.20130502.ALL.panel > sample.group  
time MingPACluster-1.30/bin/VCF2PCA -InVCF ALL.chr22.phase3_shapeit2_mvncall_integrated_v5b.20130502.genotypes.vcf.gz -InSampleGroup  
sample.group -OutPut OUT  
echo End Time :  
date
```

测评后的结果图如下：聚类结果和已知的样品分类是一致的(一致性为 0.995)。

```
cut -f 2,3 OUT.eigenvec |sed -e 's/Cluster//g' |sed -e 's/EUR/3/g' | sed -e 's/AFR/2/g' | sed -e 's/AMR/3/g' | sed -e 's/EAS/0/g' | sed -e 's/SAS/1/g' |grep -v Group  
R: cor 0.995
```

5.2.2.2 plink 做 pca

脚本： plink : PCA 2m47.502s 内存 peak 1.5G (16 threads)

```
plink2 --vcf ALL.chr22.phase3_shapeit2_mvncall_integrated_v5b.20130502.genotypes.vcf.gz -out  
plink --allow-extra-chr --pca
```

5.2.2.3 gcta64 做 pca

gcta64 : (结合 plink) (16 threads)

步骤 1 先用 plink 转格式和过滤 2.47 min 内存 peak 1.5G

```
plink2 --vcf ALL.chr22.phase3_shapeit2_mvncall_integrated_v5b.20130502.genotypes.vcf.gz -out plink --  
allow-extra-chr --make-bed
```

步骤 2 先用 gcta64 生成 grm 4.21 min 内存 peak 0.5G

```
gcta64 --bfile plink --make-grm--out out.grm  
gcta64 --grm out.grm --pca 10 --out outPCA
```



5.2.2.4 tassel 做 pca

VCF 文件要特别处理，去掉重的位点且 vcf 必须先排序。小数据可以正常运行完成。

时间大于 400min 内存大于 180g

```
Perl remove_repeatSie.pl ALL.chr22.phase3_shapeit2_mvncall_integrated_v5b.20130502.genotypes.vcf.gz out.vcf
perl tassel-5.2.52/run_pipeline.pl -Xms180g -Xmx180g -fork1 -vcf ./chr22.vcf.gz -PrincipalComponentsPlugin
-covariance true -endPlugin -export output -runfork1

## same with the BaldingNicolisKinship ##
perl tassel-5.2.52/run_pipeline.pl -fork1 -vcf Khuman.vcf.gz -KinshipPlugin -method Normalized_IBS -endPlugin
-export kinship2.txt -exportType SqrMatrix
```

5.2.2.5 gapit3 做 pca

主要做 gaws 才用到该软件，若只做 pca 不建议用。

同时一般先安 maf 0.05 过滤减少内存，在这儿内存同理超 150g，跑不动，不建议

```
source("http://www.zzlab.net/GAPIT/GAPIT.library.R")#载入 gapit3 安装包
source("http://www.zzlab.net/GAPIT/gapit_functions.txt")#载入 gapit3 安装包
setwd('D:/GWAS_gapit3')#设置工作路径到文件所在位置，这里记得把盘符 D 后面的斜杠方向改为"/"
myG <- read.table("snp220.hapmap.hmp.txt", head = FALSE)#导入基因型数据
myY <- read.table("220_pheno.txt", head = TRUE, sep = "\t")#导入表型数据
myGAPIT <- GAPIT(G = myG, output.numerical = TRUE, file.output = FALSE) #转换 hapmap 格式到数值型格式，方便下一步计算
myGD = myGAPIT$GD #转换得到数值型基因型数据
myGM = myGAPIT$GM #转换得到.snp 信息数据 (染色体编号, 染色体位置)
myGAPIT <- GAPIT( #这块是 GWAS 正式开始
  Y = myY,
  GD = myGD,
  GM = myGM,
  model = c("GLM"), #model = c("GLM", "MLM", "SUPER", "MLMM", "FarmCPU")这里可以选择不同的分析方法，根据自己需要改变，如果不知道哪个好，用默认就好啦
  PCA.total = 3, # 设置主成分个数，这里是 PC1,PC2,PC3
  file.output = T
)
```

Example1 是抽了小样品的，可以当实例

更多实例 随时更新，见 website 网页，具体见这：

<https://github.com/Hewm2008/VCF2PCACluster> 里面的

Example3 是真正大数据，可以当实例

example3 3K rice (<https://s3.amazonaws.com/3kricegenome>) with a total of 29M SNPs.
It also showed that

VCF2PCA	~0.1G	181min	40CPU
Plink2	~257G	100min	40CPU

脚本在这：

```
#wget -c https://s3.amazonaws.com/3kricegenome/reduced/NB_final.snp.fam.gz /
#wget -c https://s3.amazonaws.com/3kricegenome/reduced/NB_final.snp.bim.gz /
# wget -c https://s3.amazonaws.com/3kricegenome/reduced/NB_final.snp.bed.gz /
#gzip -d *.gz
# plink --bfile NB_final.snp --recode vcf-iid --out NB_final.snp    # bed back to vcf 259G mem
time ./VCF2PCACluster/bin/VCF2PCACluster -InVCF NB_final.snp.vcf -OutPut Rice3K -BestKManually 4 #0.1G
181min
time plink2 --vcf NB_final.snp.vcf -out plink --allow-extra-chr --pca ## 257G mem      100min
```

6.优势

- 1 快速少内存，更多位点，动态多线程等待时间极少
- 2 用法简易，对小白用户十分友好。
- 3 应用场景广泛，应用场景多种多样，自主性较大，用户可以结合自己的数据画图。
- 4 免安装，使用方便
- 5 更多种计算 kinship 的方法

7.模型及公式

在这儿主要列出几种 kinship 的计算公式

7.1 亲缘关系矩阵

Kinship 的公式主要来自这论文：[How to estimate kinship](#)



7.1.1 BNormalized_IBS(Yang/BaldingNicolsKinship)

When information is combined over loci by weighting with sample heterozygosities, we write a common kinship estimator as $r_{jj'}^w$:

$$r_{jj'}^w = \frac{\sum_{l=1}^L (X_{j_l} - 2\tilde{p}_l)(X_{j'_l} - 2\tilde{p}_l)}{2 \sum_{l=1}^L \tilde{p}_l (1 - \tilde{p}_l)} \quad (2)$$

The weighted estimator in Equation (2) is the first estimator discussed by VanRaden (2008). It estimates $(1 + F_j)/2$ when $j = j'$ and $\theta_{jj'}$ when $j \neq j'$. There is no simple translation from these estimates to those we propose in Equation (1).

It is common to refer to $(X_{j_l} - 2\tilde{p}_l) / \sqrt{[2\tilde{p}_l(1 - \tilde{p}_l)]}$ as a standardized genotype measure on the basis that the expected value of X_{j_l} is twice the allele frequency ($2p_l$) in the reference population. However, the variance of X_{j_l} is $2p_l(1 - p_l)(1 + F_j)$ rather than $2p_l(1 - p_l)$.

7.1.2 Centered_IBS(VanRaden)

When information over loci is combined as an unweighted average, we write a common kinship estimator as $r_{jj'}^u$:

$$r_{jj'}^u = \frac{1}{L} \sum_{l=1}^L \frac{(X_{j_l} - 2\tilde{p}_l)(X_{j'_l} - 2\tilde{p}_l)}{2\tilde{p}_l(1 - \tilde{p}_l)} \quad (4)$$

These terms correspond to the second estimator of VanRaden (2008), and they form the off-diagonal elements of the genetic relatedness matrix in GCTA (Yang, Lee, Goddard, & Visscher, 2011). We note that VanRaden (2008) called this estimator “weighted,” because in his matrix notation, the diagonal matrix \mathcal{D} of locus variances comes between the dosage matrices \mathcal{X} and \mathcal{X}' (\mathcal{M} and \mathcal{M}' in the notation of VanRaden 2008, respectively).

7.1.3 IBSKinship

IBS (identity by state) defined as the probability that alleles drawn at random from two individuals at the same locus are the same. The calculation is based on the definition. For a bi-allelic locus with alleles A and C, $\text{probabilityIBS}(AA, AA) = 2$, $pIBS(AA, CC) = 0$, $pIBS(AC, xx) = 1$,

IBSKinship matrix and **skip missing genotype**.



```

    /**
     * IBSKinship matrix and skip missing genotype
     * Kinship for marker j
     *      0   1   2
     * 0   2   1   0
     * 1   1   2   1
     * 2   0   1   2
    */

```

```

double table[4][4];
table[0][0] = table[1][1] = table[2][2] = 2;
table[0][1] = table[1][0] = table[1][2] = table[2][1] = table[1][3] = table[3][1] = 1;
table[0][2] = table[2][0] = 0;

```

[Sum() /L] *0.5

7.1.5 IBSKinshipImpute

IBS (identity by state) defined as the probability that alleles drawn at random from two individuals at the same locus are the same. The calculation is based on the definition. For a bi-allelic locus with alleles A and C, $\text{probability } IBS(AA,AA) = 2$, $pIBS(AA,CC) = 0$, $pIBS(AC,xx) = 1$, for miss alleles, we use the **probability(p)** to Impute the it. the related formula is as follows :

```

    /**
     * IBSKinship matrix and use probability to impute kinship
     * Kinship for marker j
     *      0       1       2       missing
     * 0       2       1       0       2(1-p)
     * 1       1       2       1       1
     * 2       0       1       2       2p
     * missing 2(1-p)   1       2p     2(p^2+q^2)
    */

```

```

double table[4][4];
table[0][0] = table[1][1] = table[2][2] = 2;
table[0][1] = table[1][0] = table[1][2] = table[2][1] = table[1][3] = table[3][1] = 1;
table[0][2] = table[2][0] = 0;

```

```

double p =NowMAF;
table[0][3] = table[3][0] = 2.0 * (1.0 - p);
table[2][3] = table[3][2] = 2.0 * p;
table[3][3] = 2.0 - 4.0 * p * (1 - p);

```

[Sum /L (deal miss)] *0.5



7.1.5 p_dis (Distance Matrix)

We calculate P_Distance as $1 - 0.5 * \text{IBS}$ (identity by state) similarity, with IBS defined as the probability that alleles drawn at random from two individuals at the same locus are the same. For clustering, the distance of an individual from itself is set to 0.

The calculation is based on the definition. For a bi-allelic locus with alleles A and C, **pIBS(AA,AA) = 2, pIBS(AA,CC) = 0, pIBS(AC,xx) = 1**, where xx is any other genotype. For two taxa, pIBS is averaged over all non-missing loci.

$$\text{P_Distance} = 1 - 0.5 * \text{pIBS}.$$

Where L is the length of regions where SNPs can be identified, and given the alleles at position l are A/C:

$d(l)_{ij}=0.0$	if the genotypes of the two individuals were AA and AA;
$d(l)_{ij}=0.5$	if the genotypes of the two individuals were AA and AC;
$d(l)_{ij}=0.0$	if the genotypes of the two individuals were AC and AC;
$d(l)_{ij}=1.0$	if the genotypes of the two individuals were AA and CC;
$d(l)_{ij}=0.0$	if the genotypes of the two individuals were CC and CC;

7.2 聚类算法

聚类算法，网上已经有很多教程，我这儿只是随便搜出来的，大家要学习可以网上搜出更多详细的认真看，在这不细讲

7.2.1 EM_Gaussian_cluster

See more at this website:

https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm

中文见这儿：<https://zhuanlan.zhihu.com/p/616053904>

7.2.2 DBSCAN

See more at this website: <https://en.wikipedia.org/wiki/DBSCAN>

中文：<https://zhuanlan.zhihu.com/p/597075459>

7.2.3 KMeans

See more at this website: https://en.wikipedia.org/wiki/K-means_clustering

8. 常见问题

8.1 为什么要重复开发这个 PCA 分析软件

A. PCA 分析很多软件，但很多须要原始数据 VCF 进一步过滤和转格式等操作，对小白不友好。如几个软件都是要先对原始 vcf 进行过滤（如三碱基）等。如 转格式：gatc64 要用 bed fam bim。

B 另外 tassle gapit 的软件是：n*m (n 是样品，m 是位点) n*n 的矩阵去算 kinship 和 pca 的 由于用到 m 位点一多 内存极大。我这是连读连算 只是两个 n*n 的内存。速度更快 结果比较会 除了精度问题 kinship 是一致的。和 gcta 是一样的。

C 相对 plink 软件和 gatc64 的软件，虽然这两个软件速度极快（也是用了 openmp 多线程的），但其中也是把所有位点读进去，内存极大，并且没有聚类功能和作图。

D 本软件我这添加了三种 聚类算法，可以根据聚类结果进行 染色作图。

E 有这个开发 PCA 的能力，想证明自己，同时也希望能被更多人引用

8.2 VCF2PCACluster 和准确性如何？

用同一数据做 pca 分析,同 gcta 和 tassle gapit 的 kinship 是一致的, pca 的结果也是一样的。再次 Example 也是很好把各人群分开，说明软件的准确性不用质疑，效率从开发时就是考虑要高效低内存的，具体后面有空再给出测评比较图表。

By the way 其中 tassle gapit 主要做 gwas, 是过滤 maf 0.05 后去做的 pca, 对特异的样品（离群的）影响大，mingPCA 默认 maf 为 0.001，几不过滤 maf, PCA 能把离群样品找到。

8.3 联系与打赏

随意 是缘是福，一切随风

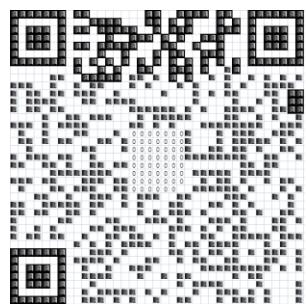
- ✉ hewm2008@gmail.com / hewm2008@qq.com
- join the **QQ Group : 125293663**

微信 打赏

QQ 入群: **125293663**

微信公众号





群名称:Reseqtools (iTools)
群号:125293663

