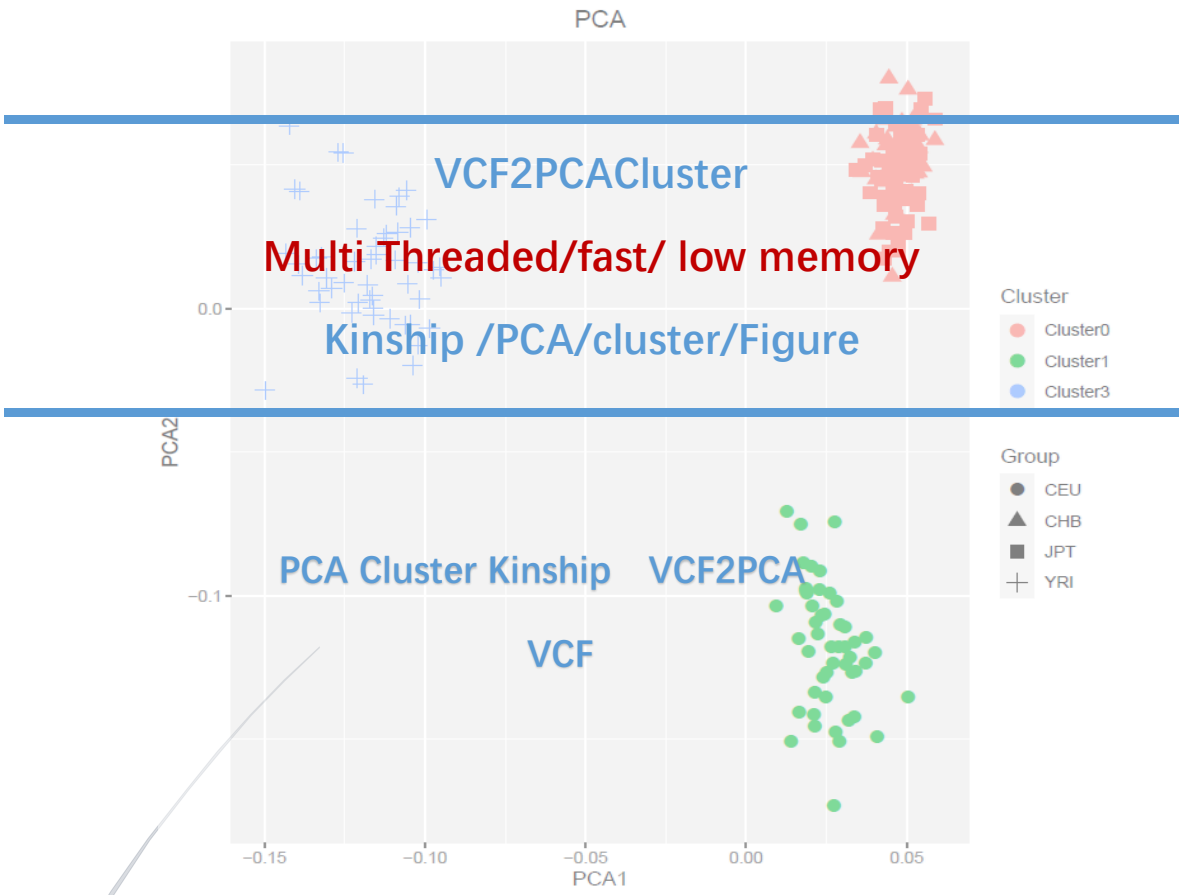
-

Manual VCF2PCACluster



Version 1.38 manual doc

2023-06-01

hewm2008@gmail.com / hewm2008@qq.com

QQ group: *125293663* *WeChat*

Dir

1 Introduction

VCF2PCACluster is a PCA and clustering analysis tool based on population SNPs in a popular VCF format. It requires only one input file and then performs PCA, Clustering and Visualization in a single step, making it very simple, easy-to-use and efficient.

Major highlights:

1. The PCA result is almost identical to those generated by Tassel and Gapit, with only minor differences in precision.

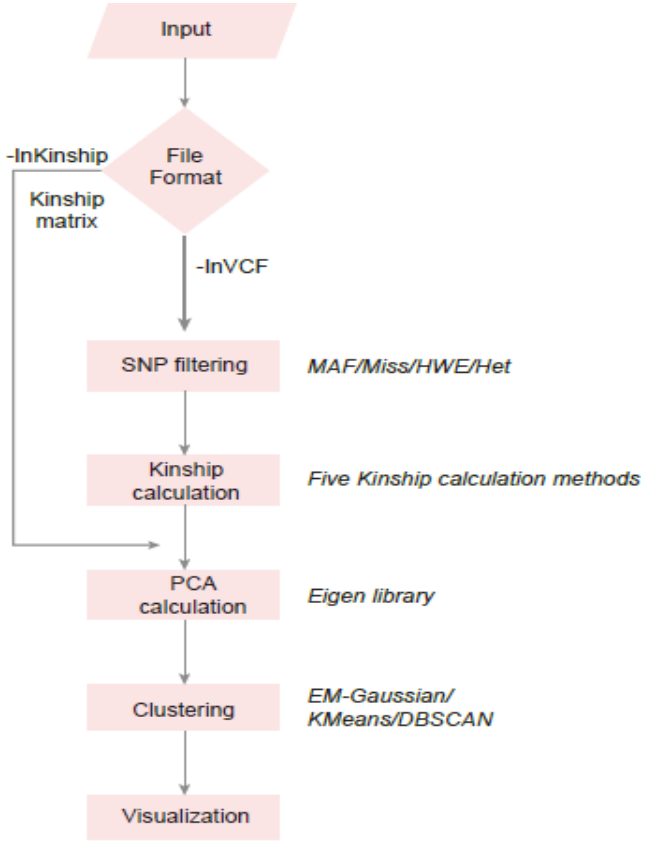
2. Functions include: (1) calculating kinship matrix using five algorithms; (2) PCA analysis; (3) clustering analysis using three algorithms; and (4) visualization of clustering results.

3. One VCF input, one-step process, convenient for users, and capable analysis of subpopulations.

4. Memory usage is only affected by the number of samples, not the number of SNPs, so the memory usage is only around 1-2 GB even up to 10k samples.

5. Three clustering algorithms: 5.1 EM-Gaussian: EM algorithm combined with a Gaussian mixture model. 5.2 K-means clustering analysis, identifies the best K value, similar to Structure and K value. 5.3 DBSCAN clustering algorithm.

6. Two custom scripts were also provided to optimize the plotting details in 2D and/ or 3D manners.



2. Application

2.1 Small Data

To show the accuracy, we provided an example (example 1) with a small SNP dataset. We downloaded the SNPs of the 1000 Genomes Project deposited in dbSNP database, and randomly selected 1194 loci from chromosome (chr) 22, including 203 samples from four populations: CEU (49), CHB (46), JPT (56), and YRI (52). (example1)

Command:

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

Of which, users could add the option ***-InSampleGroup*** for a given of prior classification and compared with PCA and clustering results using the command:

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

Note:

1 The format of ***pop.info*** contains two columns: sample name and cluster name.

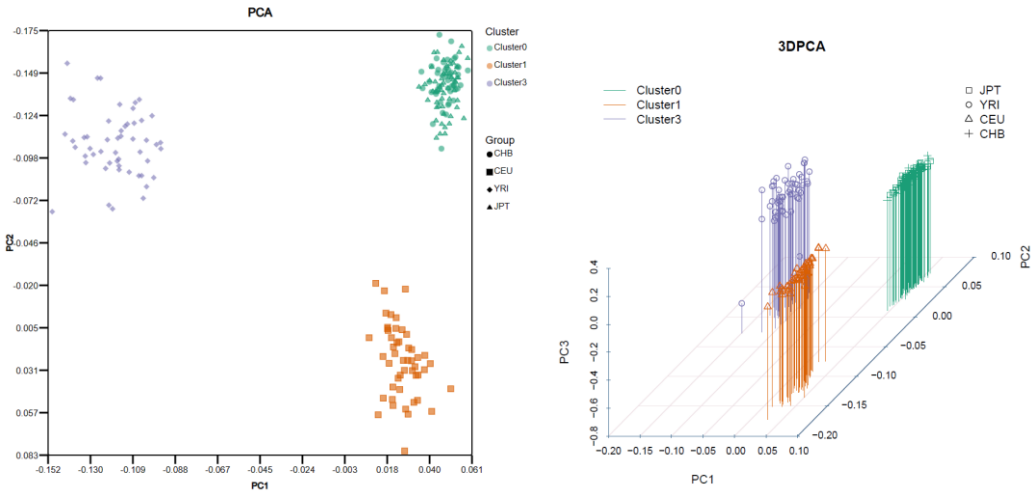
2 Users can choose the clustering method through the ***[-ClusterMethod]*** parameter

3 The user can choose the formula for calculating kinship through ***-KinshipMethod*** parameter

We next demonstrated plots with different clustering methods as the following:

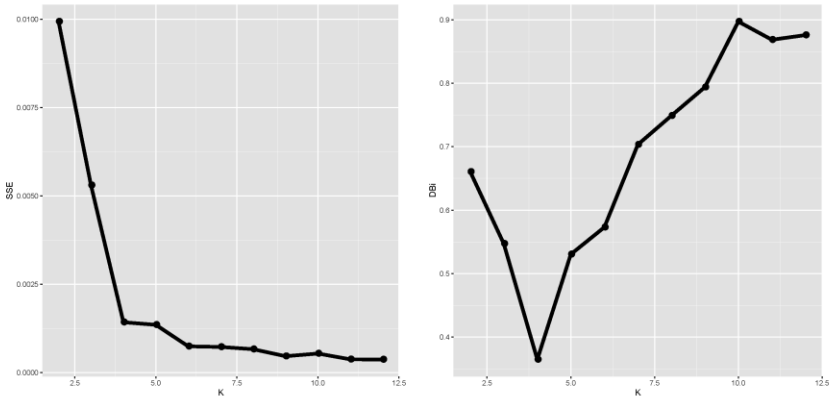
2.1.1 EM

1. The algorithm first uses the K-means algorithm to determine the best K, and users can also specify a specific K value through the parameter "-BestKManually".
2. Default initial values, iteration times (default 1000, can be passed through the parameter -Iterations), and convergence judgment parameters (default automatic, not greater than 1e-10, which can be passed by the user through -Epsilon).
3. Call the EM algorithm to find the maximum value clustering result.

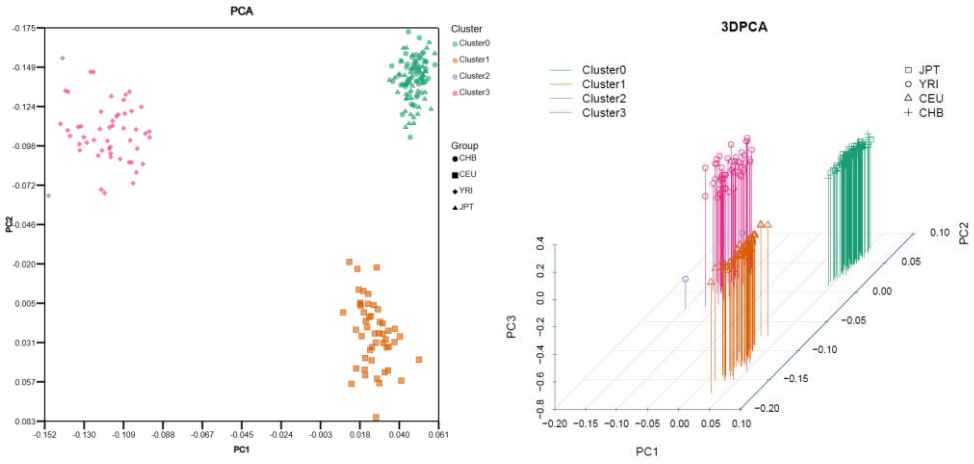


2.1.2 Kmean

According to the relationship between K and SSE, the software considers the best clustering to be K=4, as the SSE levels off or increases after 4. The software can pick the best K as 4. The software outputs results for K3-12 by default.



Our software also has the functionality to calculate the Davies–Bouldin index (DBI). Based on the relationship between K and DBI, the minimum value of DBI corresponds to the best K.
Here is the result and PCA plot for K=4:

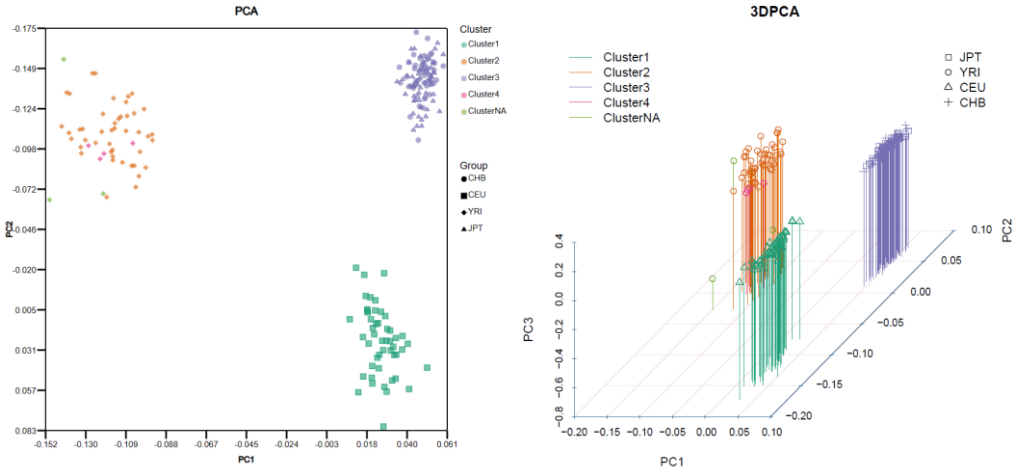


2.1.3 DBSCAN

The DBSCAN algorithm has two main parameters:

- The neighborhood radius: Eps;
- The minimum number of points required to form a core object in the neighborhood radius: MinPts.

Both of these parameters can be manually set by the user, but the program defaults to use the Elbow method to determine the optimal value for Eps, and sets MinPts to 4 in this example.
The below plot is generated by DBSCAN clustering method.



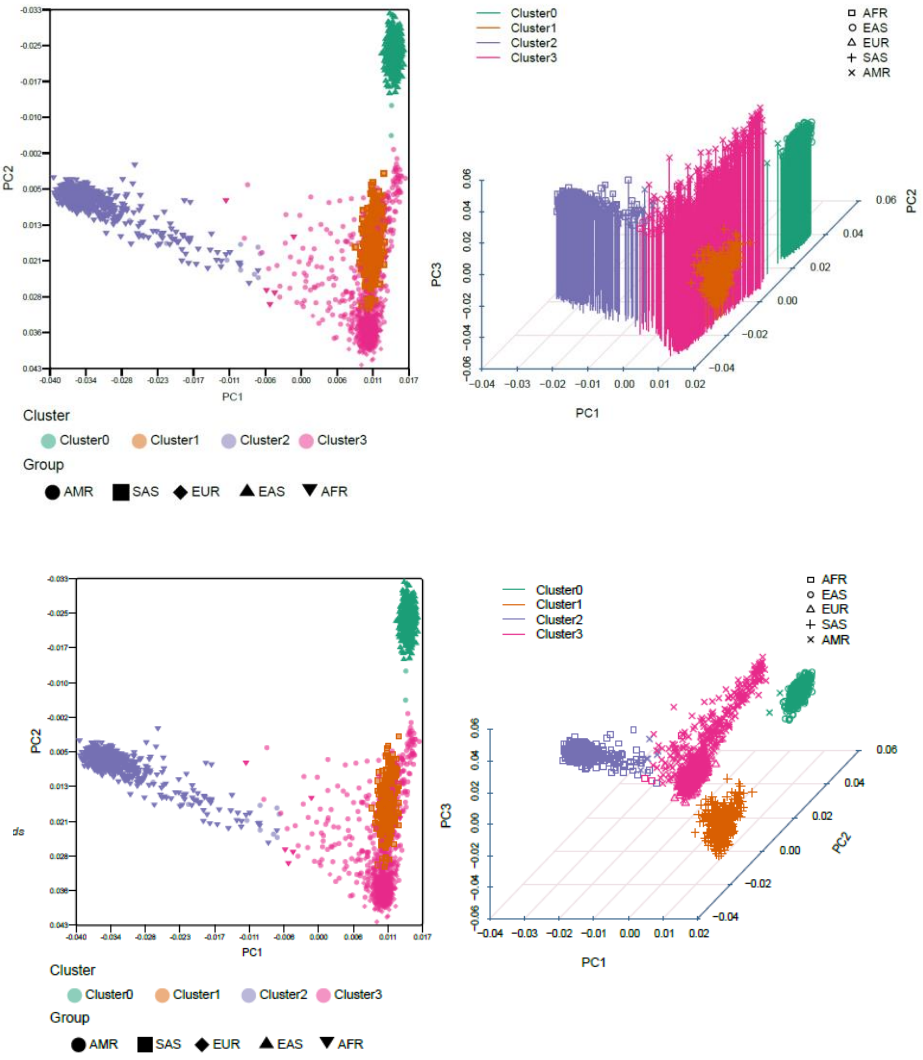
2.2 big Data (K Human chr22)

To test the accuracy and the efficiency of VCF2PCACluster, we used data of SNP data on chr22 from 1000 Genome Project (downloaded from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502) with 1,055,401 SNP in 2504 samples.

VCF2PCACluster: peak memory usage ~0.1 GB; CPU running time: ~13min (8 threading)

```
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_v3.2013050
2.ALL.panel
cut -f    1,3      integrated_call_samples_v3.20130502.ALL.panel    > sample.group
time   MingPCACluster-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
```

the Figure is shown as follows:



2.3 big Data (3K Rice)

In addition, we also tested on a large-scale SNPs in rice with 3k samples and 29 M SNPs. We demonstrated that VCF2PCACluster can efficiently produce results, requiring 181 minutes and only 0.1 GB memory . In comparison, PLINK2 took 100 minutes and consumed a massive 257 GB of memory.

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          --bfile Rice3K        --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)

we conducted a test on a very large dataset with a combination of SNP data on chromosomes 1-22 from the 1000 Genome Project, totaling of 81.2 million (M) SNPs. The results showed extremely low memory usage (~0.1 GB) and successfully finished in about 610 minutes with 8 threads using VCF2PCACluster. Conversely, PLINK2 required a larger memory usage (>200 GB) and failed to complete the job.

3 Download and Installation

3.1 Download

The new version of VCF2PCACluster will be updated and maintained in hewm2008/VCF2PCACluster. Please click below link to download the latest version. hewm2008/VCF2PCACluster

https://github.com/hewm2008/VCF2PCACluster

3.2 Requirements

VCF2PCAtools is capable for Linux/Unix/Mac OS systems. We also provide executable programs for Linux and Mac. Please install the following requirements before compiling and using VCF2PCAtools.

1. OpenMP c/c++ command is recommended to be pre-installed
2. g++ : g++ with --std=c++11 > 4.8+ is recommended
3. zlib : zlib > 1.2.3 is recommended

4. R : R with ggplot2 and scatterplot3d are recommended

3.3 Installation

Users can install it with the following options:

Option 1，we provide a static version for Linux/Unix

```
git clone https://github.com/hewm2008/VCF2PCACluster.git
cd VCF2PCACluster;        chmod 755 -R bin/*
./bin/VCF2PCACluster    -h    ### print help information
```

Option 2: compile from source code for Linux/Unix/macOS

```
git clone https://github.com/hewm2008/VCF2PCACluster.git
cd VCF2PCACluster ; chmod 755 configure    ;    ./configure;
make;      # sh make.sh
mv VCF2PCACluster    bin/;       #         [rm *.o]
```

Note: For macOS, users can run the following command first. Please ensure g++-11 has been installed using the homebrew, we have successfully tested on the macOS Monterey, Apple M1 chip.

```
ln -s    /opt/homebrew/bin/g++-11        /opt/homebrew/bin/g++    ;
export    PATH=/opt/homebrew/bin/:$PATH
```

4 Usage and parameters

4.1 Simplest usage

```
[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],defaut [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]
```

The simplest usage is just given a VCF file via ***-InVCF*** parameter and a prefix for output via ***-OutPut*** parameter. Also, users could provide another parameter ***(-InSubSample)*** for analysis of a given subset.

Users could select other methods for Kinship matrix estimation and clustering via parameters ***-KinshipMethod*** and ***-ClusterMethod*** respectively.
-KinshipMethod, five alternative methods for calculation Kinship matrix. The default is 1 that Normalized_IBS method.
-ClusterMethod, three alternative methods for clustering analysis, EM is set as default.

4.2 Detailed parameters

In addition of parameters for input, output and methods for calculation of kinship and clustering analysis, we also provide other parameters for SNP filtering and clustering. For more details, please add “-h”.

```
./bin/VCF2PCACluster -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],defaut [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]

            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 deta-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]
```

Parameters for other scripts

Bin/Plot2Deig and ***Bin/Plot3Deig*** are PERL scripts for plotting PCA and clustering results in 2D or 3D manners respectively.

```
./Plot2Deig   -h

    Version:1.40              hewm2008@gmail.com

    Usage: Plot2Deig -InFile pca.eigenvec -OutPut Fig


        Options

        -InFile          <s>   : InPut PCA.eigenvec File
        -OutPut          <s>   : OutPut svg file result

        -help                  : Show more help with more parameter

        -ColShap               : colour <=> shape for cluster or group
        -Columns         <s>   : the columns to plot a:b [4:5]
        -ColorBrewer     <s>   : the color brewer for points [Dark2]
        -Title           <s>   : title (legend) [PCA]

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

4.3 Input file

1. VCF format
2. GenoType format

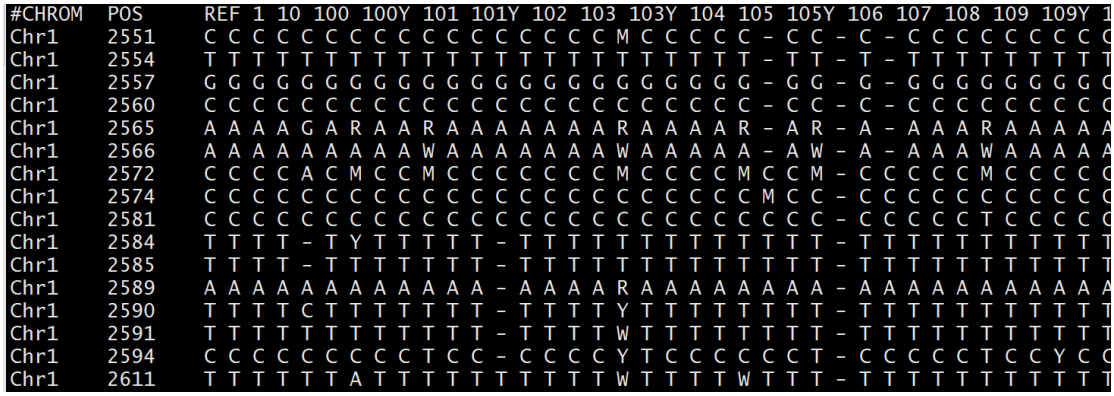
4.3.1 Basic input file format

1. VCF format

1.1 An example

```
##fileformat=VCFv4.2
##fileDate=20090805
##source=myImputationProgramV3.1
##reference=file:///seq/references/1000GenomesPilot-NCBI36.fasta
##contig=<ID=20,length=62435964,assembly=B36,md5=f126cdf8a6e0c7f379d618ff66beb2da,species="Homo sapiens",taxonomy=x>
##phasing=partial
##INFO=<ID=NS,Number=1,Type=Integer,Description="Number of Samples With Data">
##INFO=<ID=DP,Number=1,Type=Integer,Description="Total Depth">
##INFO=<ID=AF,Number=A,Type=Float,Description="Allele Frequency">
##INFO=<ID=AA,Number=1,Type=String,Description="Ancestral Allele">
##INFO=<ID=DB,Number=0,Type=Flag,Description="dbSNP membership, build 129">
##INFO=<ID=H2,Number=0,Type=Flag,Description="HapMap2 membership">
##FILTER=<ID=q10,Description="Quality below 10">
##FILTER=<ID=s50,Description="Less than 50% of samples have data">
##FORMAT=<ID=GT,Number=1,Type=String,Description="Genotype">
##FORMAT=<ID=GQ,Number=1,Type=Integer,Description="Genotype Quality">
##FORMAT=<ID=DP,Number=1,Type=Integer,Description="Read Depth">
##FORMAT=<ID=HQ,Number=2,Type=Integer,Description="Haplotype Quality">
#CHROM POS     ID        REF     ALT     QUAL FILTER INFO                              FORMAT      NA00001        NA00002        NA00003
20     14370   rs6054257 G       A       29   PASS   NS=3;DP=14;AF=0.5;DB;H2           GT:GQ:DP:HQ 0|0:48:1:51,51 1|0:48:8:51,51 1/1:43:5:.,.
20     17330   .         T       A       3    q10    NS=3;DP=11;AF=0.017               GT:GQ:DP:HQ 0|0:49:3:58,50 0|1:3:5:65,3   0/0:41:3
20     1110696 rs6040355 A       G,T     67   PASS   NS=2;DP=10;AF=0.333,0.667;AA=T;DB GT:GQ:DP:HQ 1|2:21:6:23,27 2|1:2:0:18,2   2/2:35:4
20     1230237 .         T       .       47   PASS   NS=3;DP=13;AA=T                   GT:GQ:DP:HQ 0|0:54:7:56,60 0|0:48:4:51,51 0/0:61:2
20     1234567 microsat1 GTC     G,GTCT  50   PASS   NS=3;DP=9;AA=G                    GT:GQ:DP    0/1:35:4       0/2:17:2       1/1:40:3
```

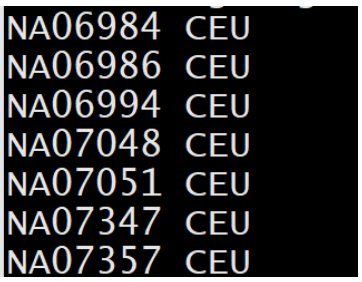
2 Genotype format



#CHROM	POS	REF	1	10	100	100Y	101	101Y	102	103	103Y	104	105	105Y	106	107	108	109	109Y	1
Chr1	2551	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Chr1	2554	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T	T
Chr1	2557	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G	G
Chr1	2560	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Chr1	2565	A	A	A	A	G	A	R	A	A	R	A	A	A	A	A	A	A	R	A	A	A
Chr1	2566	A	A	A	A	A	A	A	A	A	W	A	A	A	A	A	A	A	W	A	A	A
Chr1	2572	C	C	C	C	A	C	M	C	C	M	C	C	C	C	C	C	C	M	C	C	C
Chr1	2574	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Chr1	2581	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C	C
Chr1	2584	T	T	T	T	-	T	Y	T	T	T	T	-	T	T	T	T	T	T	T	T	T
Chr1	2585	T	T	T	T	-	T	T	T	T	T	T	-	T	T	T	T	T	T	T	T	T
Chr1	2589	A	A	A	A	A	A	A	A	A	A	A	-	A	A	A	A	R	A	A	A	A
Chr1	2590	T	T	T	T	C	T	T	T	T	T	T	-	T	T	T	T	Y	T	T	T	T
Chr1	2591	T	T	T	T	T	T	T	T	T	T	T	-	T	T	T	T	W	T	T	T	T
Chr1	2594	C	C	C	C	C	C	C	C	T	C	C	-	C	C	C	C	Y	T	C	C	C
Chr1	2611	T	T	T	T	T	T	A	T	T	T	T	T	T	T	T	T	W	T	T	T	-

4.3.2 Sample cluster information (optional)

In some cases, one may has known about the actual cluster information and want to compare it with clustering results. Then, users could provide sample cluster information by the option (***-InSampleGroup***) with the file contains two columns: **sample name** and **sample cluster**. In addition, users could analyze a subset of samples in the VCF file and then give the subset information with only one column containing sample names by the option (***-InSubSample***).



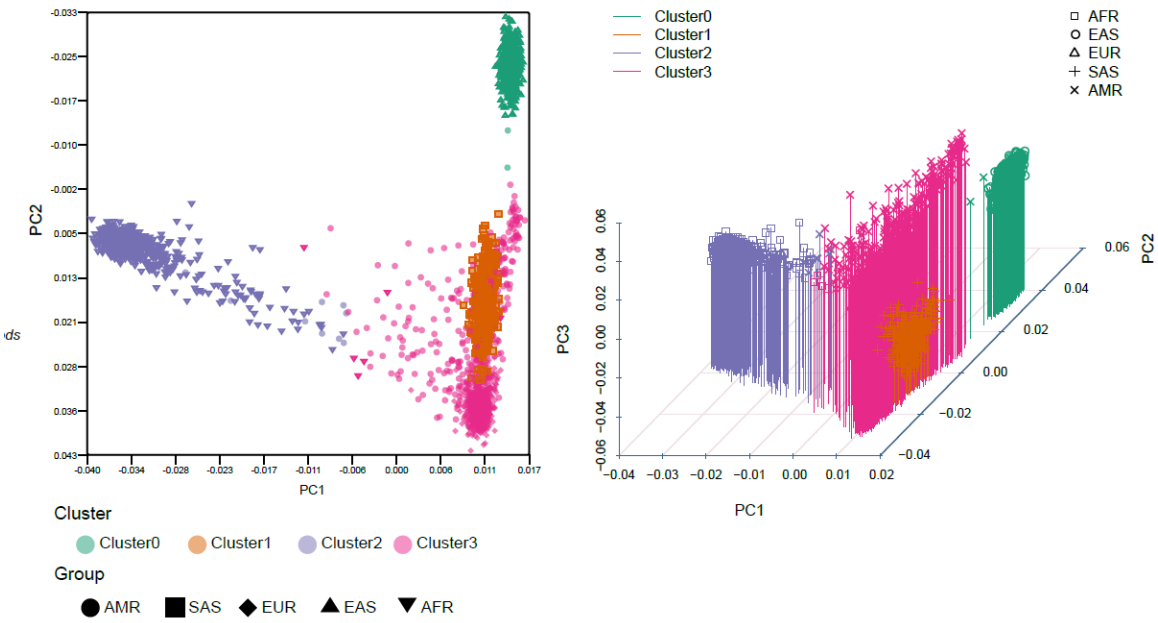
```
NA06984 CEU
NA06986 CEU
NA06994 CEU
NA07048 CEU
NA07051 CEU
NA07347 CEU
NA07357 CEU
```

4.4 Output

VCF2PCAtools generated five output files and was listed bellows:

Output files	explanation
out.kinship	Kinship matrix
out.eigenvec	Best clustering and PCA results
out.eigenval	Eigenvector values of PCA results
out.PCA1_PCA2.pdf	PCA 2D plot with cluster
out.PCA1PCA2PC3.pdf	PCA 3D plot

5 Benchmark of accuracy and performance



We could observe that the consistent between VCF2PCACluster clustering result and prior sample groups was 0.995.

5.1 Accuracy

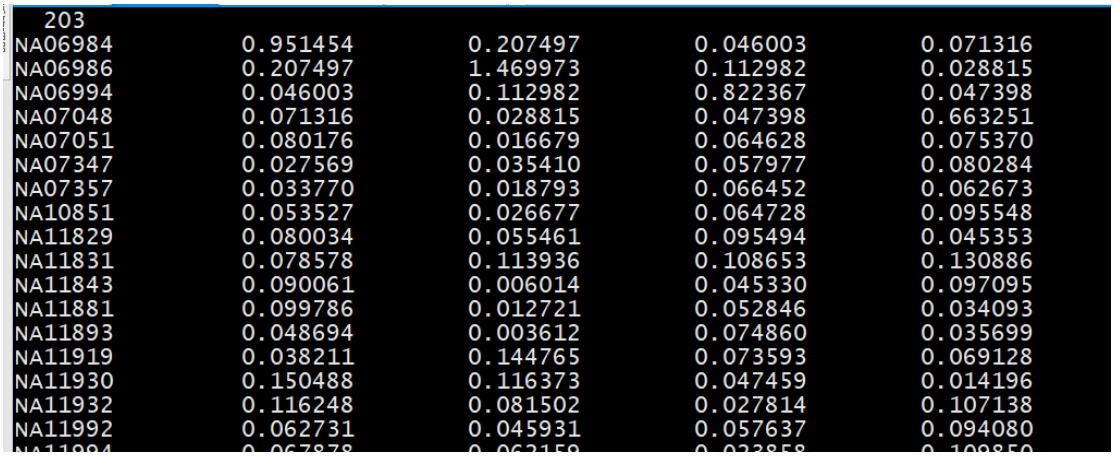
The core result of PCA is the Kinship calculation. Here we mainly compare the Kinship and PCA result using the test data in example 1.
We found the identical result of Kinship calculated by VCF2PCACluster, Tassel and Gcta64. In addition, The PCA result of VCF2PCACluster is also the same with that generated by Gcta64 indicating our high accuracy.

Listed below are Kinship matrix estimated by different methods.

5.1.1 Kinship : Normalized_IBS

Compared with kinship calculated by VCF2PCACluster (Normalized_IBS), Tassel (Normalized_IBS) and Gcta64 (Yang) showed the identical result. (example1)

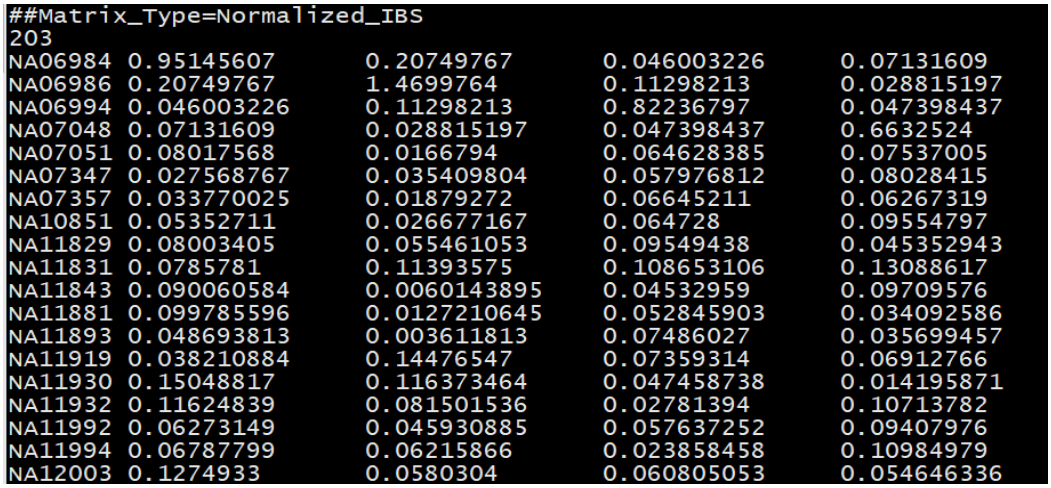
```
VCF2PCA      -InVCF Khuman.vcf.gz   -OutPut OUT     -KinshipMethod 1
```



```
  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
```

Figure above. screenshot of kinship calculated by VCF2PCACluster using Normalized_IBS method.

```
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
```



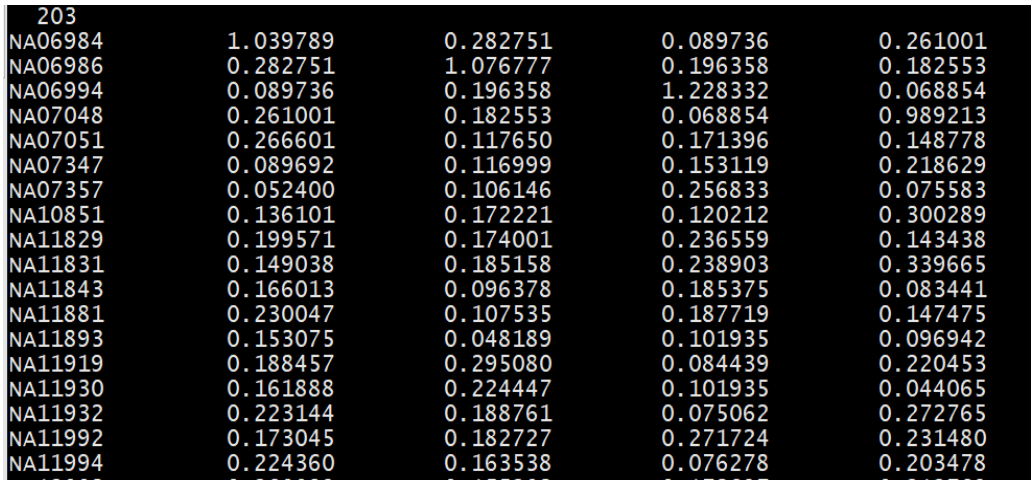
```
##Matrix_Type=Normalized_IBS
203
NA06984 0.95145607      0.20749767      0.046003226     0.07131609
NA06986 0.20749767      1.4699764       0.11298213      0.028815197
NA06994 0.046003226     0.11298213      0.82236797      0.047398437
NA07048 0.07131609      0.028815197     0.047398437     0.6632524
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
```

Figure above, screenshot of kinship calculated by Tassel and Gcta64 using Normalized_IBS and Yang methods respectively.

5.1.2 Kinship : Centered_IBS

The Kinship matrix calculated using Centered_IBS by VCF2PCACluster and Tassel is the sample, as well as the Van method implemented in Gcta64. (example1)

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



```
  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
```

Figure above. screenshot of Kinship calculated by VCF2PCACluster using Centered_IBS method.

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 ## **Van**

```
##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
```

Figure above. Screenshort of Kinship calculated by Tassel and gcta64.

5.1.3 PCA

We run these tools using the default parameters and compared the first three PCs. The result showed the consistent of PCA results produced by these tools, including the PC projection. The following listed the details. (example1)

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		-0.0
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		-0.00487043	
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 0.000800092 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 0.0079168 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
```

```
gcta64 --bfile file --make-grm GRM
gcta64 --grm GRM --pca 4
```

In addition, eaxmple2's chr22 from 1000 Genome Project (downloaded from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502) with 1,055,401 SNP . example3 3K rice (https://s3.amazonaws.com/3kricegenome) with a total of 29M SNPs. It also showed that VCF2PCA and GCTA64 had consistency in Kinship and PCA results.

5.2 Performance

In performance comparison, versions of GCC and OpenMP may affect.
To test the accuracy and the efficiency of VCF2PCACluster, we used data of SNP data on chr22 from 1000 Genome Project (downloaded from ftp://ftp.1000genomes.ebi.ac.uk/vol1/ftp/release/20130502) with 1,055,401 SNP in 2504 samples and run in the same computational node.

We compared different software programs using the chromosome 22 data from the 1,000 Genomes Project, which consisted of over 1 million SNPs and 2,504 samples. Tassel and GAPIT3 took a large amount of time and memory (over 150 GB), and waiting times exceeded 100 minutes. Therefore, these software programs may not be

suitable for large-scale kinship and PCA analyses.

GCTA does not support reading VCF files directly, and VCF files must be filtered and converted to the PLINK format using PLINK. The waiting time for GCTA was approximately 2.27 minutes, with the use of multiple threads.

The VCF2PCACluster software used multiple threads during computations but did not use them while reading the VCF. However, the impact of the number of SNPs on the memory usage of VCF2PCACluster was eliminated. This means that the peak memory usage of VCF2PCACluster will not increase even when dealing with massive amounts of data.

Overall, we believe that VCF2PCACluster is more user-friendly, as it allows for one-step analysis, clustering, and plotting without additional filtering or conversion. Additionally, the time and memory requirements of VCF2PCACluster are very reasonable even for novice users.
The scripts used for the evaluation are attached for reference.

Software	Input	SNP filtering	Functions				performance	
			Kinship	PCA	Clustering	Visualization	memory	time consumption
VCF2PCACluster	VCF	Maf, Missing, HWE	yes	yes	yes	yes	~0.1GB	~12min (8 threads)
GCTA	Plink2	Maf	yes	yes	no	no	~1.5GB	~7min (16 threads)
PLINK2	VCF	Maf, Missing, HWE	yes	yes	no	no	~1.5GB	~2.47min (16 threads)
TASSEL	VCF/hmp	Maf	yes	yes	no	no	>180GB	>400min
GAPIT	hmp	no	no	yes	no	yes	>150GB	>400min

The command listed below was to compare VCF2PCACluster with other tools.

5.2.1 VCF2PCACluster

VCF2PCACluster: peak memory usage <0.1 GB; CPU running time: ~13min (8 threading)

```
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_v3.2013050
2.ALL.panel
cut -f    1,3      integrated_call_samples_v3.20130502.ALL.panel     > sample.group
time   MingPCACluster-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
```

5.2.2 plink

plink : PCA; peak memory usage: 1.5G; CPU running time: 2m47.502s

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

5.2.3 gcta64

step1: use plink2 for format converting and SNP filtering; peak memory usage 1.5G; CPU running time: 2.47 min.

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

step2: use gcta64 for converting plink into grm; peak memory usage 0.5G; CPU running time: 4.21 min.

gcta64 --bfile plink --make-grm --out out.grm

gcta64 --grm out.grm --pca 10 --out outPCA

5.2.4 tassel

Requirement of pre-processing SNPs, including discard repeated sites and sorted. It's ok to test on small dataset but the running time >200 min and peak memory usage >180 GB.

##

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 BaldingNicolsKinship

perl tassel-5.2.52/run_pipeline.pl -fork1 -vcf Khuman.vcf.gz -KinshipPlugin -method Normalized_IBS -endPlugin -export kinship2.txt -exportType SqrMatrix

5.2.5 gapit3

```
source("http://www.zzlab.net/GAPIT/GAPIT.library.R")
source("http://www.zzlab.net/GAPIT/gapit_functions.txt")
myG <- read.table("snp220.hapmap.hmp.txt", head = FALSE)
myY <- read.table("220_pheno.txt", head = TRUE, sep="\t")
myGAPIT <- GAPIT(G=myG, output.numerical=TRUE,file.output =FALSE)
myGD= myGAPIT$GD
myGM= myGAPIT$GM
myGAPIT <- GAPIT(
Y=myY,
GD=myGD,
GM=myGM,
model=c("GLM "),
PCA.total=3,
file.output =T
)
```

Note: We didn't recommend users to employ Gapit for PCA analysis unless for GWAS and should filter low MAF (e.g., 0.05) to avoid large amount of peak memory usage.

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
Gcta64	~257G	283min	1-40CPU

6 Advantages

1. Fast and low memory usage
2. Simple and user-friendly
3. Highly user-defined
4. Free-installation and convenient
5. Five methods calculating Kinship

7. algorithm description

7.1 Kinship matrix

The formula calculating Kinship are referred to: How to estimate kinship

7.1.1 Normalized_IBS (BaldingNicolsKinship/(Yang/Normalized_IBS)

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

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

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 $r^{u}_{jj'}$:

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

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, ***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

7.1.4 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, ***probabilityIBS(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 distance

We calculate P_Distance as 1 – 0.5*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, ***probabilityIBS(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.

P_Distance = 1 -0.5*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 Clustering methods

7.2.1 EM_Gaussian_cluster

Please refers to https://en.wikipedia.org/wiki/Expectation%E2%80%93maximization_algorithm for details.

7.2.2 DBSCAN

Please refers to https://en.wikipedia.org/wiki/DBSCAN for details.

7.2.3 KMeans

Please refers to https://en.wikipedia.org/wiki/K-means_clustering for details.

8.Question and Answer (QA)

8.1 Accuracy about VCF2PCACluster

Answer: For accuracy estimation, we compared with other tools, GCTA, Tassle, Gapit, using the same data for PCA analysis. We found the Kinship matrix is consistent and the PCA result is the same between VCF2PCACluster and other tools. In addition, an example was provided where the different populations were clearly separated, demonstrating that the accuracy of the software is reliable, and its efficiency was considered to be high and low-memory consuming during development. By the way, Tassel GAPIT mainly performs GWAS, and the PCA was done after filtering out MAF 0.05, which had a significant impact on outlier samples. However, the default MAF for VCF2PCACluster is set to 0.001, so PCA can still identify outlier samples.

8.2 Contacts

If any question, please email to hewm2008@gmail.com or hewm2008@qq.com.

QQ group: ***125293663*** ***WeChat***