Snp HeRitability Estimation Kit (SHREK) Development Log

Sam Shing Wan, Choi
choishingwan@gmail.com

Johnny Sheung Him, Kwan
shkwan@hku.hk

Henry Chi-Ming, Leung
cmleung2@cs.hku.hk

March 14, 2016

Contents

1 Assumption in implementation

To make life easier, I made a number of assumption when I implement the programme and they are listed as follow

1. The intervals within each individual bed files are non-overlapping
2. SNPs filtered out when reading the p-value file will not be included in the output no matter what (avoid storing unnecessary information)
3. We will remove *all* SNPs that are in perfect LD with each-other
    (a) Realistically, we can not remove *all* SNPs that are in perfect LD with each-other without first computing the LD matrix for the whole genome. However, that will incur a large speed penalty to the algorithm. To compromise, we remove SNPs that are in perfect LD with each-other *within the same window.* Take into consideration of the calculation of variance, we will need to plan forward and consider SNPs within 4 windows. That's definitely overkill but that avoid problem of crazy perfect LD interactions (to be honest, I would consider those as error, yet it is easier for me to just remove them)
4. We do not consider a SNP ambiguous unless we need to perform flipping
5. Only when the direction of effect is presented will we perform the detail variance estimation calculation. The equation for the variance calculation is: $\boldsymbol{R}_{\boldsymbol{sq}}^{-1}\boldsymbol{n}(4\boldsymbol{R}\circ(\boldsymbol{z}\boldsymbol{z}^t)-2\boldsymbol{R}_{\boldsymbol{sq}}^{-1})\boldsymbol{n}\boldsymbol{R}_{\boldsymbol{sq}}^{-1}$ where $\boldsymbol{n}$ is a square diagonal matrix with diagonal as the sample size and $\boldsymbol{z}$ is the vector of z-statistics.
6. Apparently, we also assume that the bed file for the region is non-overlapping

2 Simulation Information

It is important to record whatever version we are using for the simulation. Here are the informations:

Table 1: Version of programmes used for simulation

Programme	version	Download date
LDSC	1.0.0	2016-3-3
GCTA	1.25.2	2016-3-3

3 To Do

1. Compare the condition number of the LD matrix before and after the LD correction
2. Compare the performance of case control estimation with different sample size
3. For LD correction, might also worth testing how the density and total number of SNPs affect the estimation
4. For effect size estimation, don't use the complicated method
5. Compare the speed of the new and old implementation (remember that the complex variance calculation do takes more time to perform)

4 January 25, 2016

We are going to re-write SHREK starting from today. There are a number of main goal in this re-work

1. The whole parameter parsing
2. Better help messages
3. Allow filtering by imputation score (0-1)
4. Filtering of SNPs that have different reference/alternative allele with reference
5. Use Armadillo instead of EIGEN to improve speed
6. Filter *all* SNPs that are in perfect LD
7. Implement the advance variance calculation
8. Start documenting the codes

4.1 Finished Today

1. Finished the error message for different modes
2. Finished the parsing for binary trait
3. Finished the parsing for quantitative trait
4. Disabled risk prediction, should focus on the heritability estimation first

I have not perform debugging, there can be error

5 January 26, 2016

Continue on where I left yesterday Assumptions we made in the programme

1. p-value file has header
2. the numbers return from Command class are index (0 based)

I really want to write a more compacted code. Will try to restructure the command class

5.1 Finished Today

1. Finished refining the usage messages
2. Finished the basic parameter parsing (Still haven't finish the index based parameter though)

6 January 27, 2016

6.1 Aim

We want to at least finish the region parsing and SNP class update today

6.2 Finished Today

1. Completed the update of the Command class (compiled without problem)
2. Completed the update of the Region class (compiled without problem)
3. Completed the update of the Snp class (compiled without problem)

I also *haven't* test run the programme, so there might be additional syntax / logic error that are not picked up at the current stage.

7 January 28, 2016

7.1 Finished Today

1. Finished the checking of reference panel
2. Found a different algorithm for popcount, which should be platform independent
3. Installed armadillo and OpenBLAS
   (a) I have to add the OpenBLAS to the PATH and LD_LIBRARY_PATH for armadillo to detect it
4. Updated the makefile accordingly

    (a) However, I have not updated the decomposition class, therefore there will be problem when we try to compile the programme

5. Updated the r^2 calculation function such that it is using the correct sample size
    (a) The sample size when calculating the r^2 is defined as the number of samples containing *both* SNPs
    (b) Because of how complicated the genotype class is, I really don't want to modify it
    (c) The main bottleneck for the computation is in the decomposition anyway, so it should be fine

8 January 29, 2016

Need to at least finish the getSNP function. Must be careful with it as this is not only complicated, but also extremely important for the whole programme.

I spent whole day in annotating the document of the class to make sure I don't overlook any problem.

9 February 1, 2016

Start to implement the getSNP function. A number of point to note

1. First get all the SNPs that passed all filtering (e.g. MAF)
2. Then for each window, calculate the LD
3. When encountered with perfect LD, we retain the *first* SNP
    (a) The summary statistics will be the mean of all SNPs in perfect LD
    (b) All SNPs except the first are *removed*
    (c) So they will not appear in subsequent analysis

10 February 2, 2016

Try to perform extensive check on codes that has been written

10.1 Finished Today

1. Finished debugging, everything up until the initialization of genotypeFileHandler is ok

11 February 3, 2016

Sat in a meeting with Pak today, now we have a few things to-do.

11.1 Finished Today

We have switched to use list instead of deque. The benefit of list is that it is a double linked list, therefore it is much easier for one to remove the middle part. Such property is ideal for the handling of perfect LD. However, it also means that we cannot access the elements of the list by index. The only way for us to access the elements is to use the iterators and the advance function from the iterator library. Although this makes the implementation more difficult, it should be able to help us to speed up our code even if we want to perform perfect LD removal.

1. Extensive testing of the previous classes
2. Now use iterators for the boundary, this is more dynamic and can work with the perfect LD removal without inducing much complication into the algorithm (however, the implementation itself can be complicated)

11.2 Problem identified today

So assuming that the SNPs A B C are occurring in the genome by alphabetical order, when we calculate the LD of A with B C, we do not observe any perfect LD. This will lead us to continue with the calculation of SNP B and C. However, if SNP B and SNP C is in perfect LD, we will need to go back to the LDs of SNP A to remove the column of SNP C. Not only is this procedure complicates the implementation, it will also significantly slow down the computation because of the constant resizing of the matrix.

To conclude, it might be better for us to use the original "sausage" approach, where we should also locate all the perfect LD location. Then we can remove the perfect LD afterwards.

12 February 4, 2016

Was spending the whole day in implementing the "sausage" approach. However, it was found that the calculated R^2 are terribly wrong. So now converting back to the old method of inclusion for the genotype reading. We will then check the implementation of the R^2 computation right after we have make sure the genotype reading is correct.

13 February 5, 2016

Try to continue on what we have been doing yesterday.

13.1 Finished Today

1. Finished debugging the genotype reading. By using the old inclusion method, we avoid a lot of complication and the programme now works accordingly. We also checked the bit arithmetic and decided to use our original calculation which unlike what I previously thought, isn't platform specific.

13.2 Special Note

The NumberOfBit function in the usefulTools doesn't work in our case because our number is longer than uint32_t. I am rather uncertain how should we change the script accordingly because it involves extreme high level bit arithmetic. However, based on online sources, it seems like the __builtin_popcountll function will work as long as we compile the programme using gcc. So we will stick to that function for now.

14 February 11, 2016

Back from holiday, now taking the perfect LD seriously and start implementing the perfect LD removal algorithm. It is important to note that it is possible for the boundaries to be in perfect LD, therefore potentially changing the boundary. However, take into account of our current procedure, most of the time, only the last block need our attention, unless this is the beginning of the chromosome or decomposition region.

14.1 Finished Today

1. Implemented the slow but simple version of the perfect LD removal, need to test run it.
2. Solved some bugs, e.g. the genotype collection and LD construction bug (use the wrong index)
3. Found that there are still problems with the perfect LD removal. Specifically, when the SNP at the bound is in perfect LD with previous SNPs, the handling are still problematic. Need to change my test case in which I reduce the gap between the two SNPs (e.g. the 4− >7)

15 February 12, 2016

Finishing the first part of debugging. Basically, the LD construction seems to be ok for now. Also, I reused my old code for getting the mean of the test-statistics. Now, we store only the z-score in the Snp object. This reduces the number of information we store in Snp object but slightly increases the number of computation we have to perform.

15.1 Finished Today

1. Finished all the test run for linkage construction (have not test the cross chromosome yet)
2. Implemented the decomposition and variance calculation. (Using Tim's Equation) Will need to perform the simulation to see if the variance estimation works.
3. Updated the Snp class to accommodate the perfect LD computation.

15.2 Note

Unfortunately, I am not used to template usage in c++, therefore I will have to write two separate function for the decomposition of the matrix depending on whether if the sign is provided.

An important finding regarding the linking of the library. You must have the armadillo/usr/lib folder under your LD_LIBRARY_PATH. Also, you will need -larmadillo when you compile and it should come *after* the cpp file.

When performing the test code, it was found that the pinv function of armadillo is actually 100 times faster than my self-implemented pseudo inverse. Therefore, we should use their implementation instead.

What we have to do next is to store the results of the estimated variance and to generate the required output.

16 February 15, 2016

Start working on the variance capturing for the complex variance calculation. The most difficult part is to determine the denominator of the calculation. The best way might be to use median, which is less sensitive to outliers. However, the problem is, how to calculate the median without storing the whole matrix, which is memory intensive?

According to Tim, we only need the *rows* of the target SNPs, therefore we only need to capture each *rectangle* per run. Currently trying to list out all possible scenarios to avoid bugs.

17 February 16, 2016

Continue on what I was doing yesterday.

17.1 Finished Today

1. Finished implementing the whole programme, now need to start the debugging hell

18 February 17, 2016

DEPENDENCY HELL!!! AND BUGS, A LOT OF BUGS!!!

18.1 Today's work

1. Trying to debug the programme
2. Found that gdb doesn't work (too old)
3. Try to compile the gdb
4. Failed, so tried to compile latest gcc and see if that works

5. Need to make sure we recompile everything with the new gcc (otherwise, you will have tones of undefined referencing)

6. The variance is most likely to be wrong

19 February 18, 2016

Early simulation results suggested that the MSE of the new programme doubles that of the old one which is really bad. So now we need to check what's the problem of that.

Using the last simulated data, it was found that by not performing the LD correction, we can get a similar result as the new implementation. Therefore it is likely that the LD correction is accounting for the problem.

19.1 Note

Given the following plink simulation input, we would like to have an empirical variance of around 0.002729694

```
       10 snp 0.05 0.05 0.05 0.05 1 0.001 0
  100 snp2 0.05 0.05 0.05 0.05 0.7 0.001 0
        890 null 0.05 0.05 0.05 0.05 0.8 0 0
```

19.2 Finished Today

1. The variance calculation has been fixed

2. It seems like the programme can produce sensible results for now

20 February 19,2016

The use of List is very smart in a way that it preserves the iterator, however, using the profiling, it was found that the List operations are taking up most time. Therefore, I will now have to change all the List usage back to deque, hopefully make things a bit faster

21 February 22, 2016

We have finished some basic debugging in the past few days. The heritability estimation now work as planned. However, the variance is still rather problematic. The original equation:

$$\mathrm{Var}(h^2) = \boldsymbol{R_{sq}}^{-1}\frac{4\boldsymbol{R}\circ\boldsymbol{z}\boldsymbol{z}^t - 2\boldsymbol{R_{sq}}}{n^2}\boldsymbol{R_{sq}}^{-1} \tag{1}$$

Seems to return negative results. We need to keep working on it.

$$
\begin{aligned}
\mathrm{Var}(H) &= \boldsymbol{1^t R_{sq}^{-1}} \frac{4\boldsymbol{R} \circ \boldsymbol{\mu\mu^t} + 2\boldsymbol{R_{sq}}}{n^2} \boldsymbol{R_{sq}^{-1} 1} \\
&= \boldsymbol{1^t R_{sq}^{-1}} \frac{4\boldsymbol{R} \circ (\boldsymbol{zz^t} - \boldsymbol{R}) + 2\boldsymbol{R_{sq}}}{n^2} \boldsymbol{R_{sq}^{-1} 1} \\
&= \boldsymbol{R_{sq}^{-1}} \frac{4\boldsymbol{R} \circ \boldsymbol{zz}^t - 2\boldsymbol{R_{sq}}}{n^2} \boldsymbol{R_{sq}^{-1}} \qquad (2)
\end{aligned}
$$

22 February 23, 2016

The main problem with our simulation is the speed required. From what we saw, the main bottle neck for the simulation is hapgen. So we might have to figure out a way to make it faster.

23 February 24, 2016

We are trying to tackle the problem of variance estimation. Seems like only when we use the sample genotypes can we correctly estimate the empirical variance.

24 February 25, 2016

Still trying to figure out the problem of the variance estimation. One thing we found is that our current simulation method actually violates the assumption of fixed X in Tim's formular of phenotype simulation. Therefore, we now try to use a large reference panel (e.g. 10000+) samples to calculate the Var($X\beta$) and use this for the calculation of phenotype.

25 February 29, 2016

When working on the ASD samples, we found a segmentation fault, now dealing with it.

We have fixed the segmentation fault, it was actually an old bug that was corrected before in the other function.

We found that the samples simulated from HapGen are actually highly related, therefore we need to figure out a method to properly perform the simulation. Lesson learn: *NEVER ASSUME ANYTHING WITHOUT FIRST TESTING IT!*

So I am now checking on the relationship matrix of the samples, it seems like most of the samples are related. Therefore we need to figure out a way to quickly remove all the related samples and retain the maximum number of independent samples. An idea is the mean distance with others.

26 March 1, 2016

Now trying to use genomeSimla. The installation isn't easy as we will first need to install the boost library and libpng. Both can easily be done as long as we set the prefix when configure. However, when compiling the genomeSimla, we need to first perform configure (specifying the location of boost just in case), then most importantly, go into: /src/pngwriter and /src/genomesim and change their make file such that all instance of libpng are pointing to the libpng we installed. Also, make sure to change the -lpng12 flag into the correct version of libpng we are using (referring to the libpng include directory), for example, in my current build, I've used -lpng16. Then we can make and make install successfully.

Ok, I have absolutely no idea what genomeSimla is doing, their results ain't comprehensible either... Just a bunch of binary files and htmls...

On the other hand, when dealing with the ASD sample, we found yet another segmentation fault. Also, the final output for some reason only contain things on chromosome 6. Currently debugging the programme.

We have now decided that we should use the WTCCC samples for the simulation. By performing QC (geno 0.01 mind 0.95 hwe 0.0000001 genome min 0.125), we have 15929 samples left.

27 March 2, 2016

We need to perform detail QC on the samples. First, we need to check the individual batch of samples (by different WTCCC diseases). The following QCs are required:

1. Genotyping rate
2. Sample Genotyping rate
3. MAF filtering
4. Heterozygousity check
5. PCA for population stratification
6. Hardy Weinberg

The script of QC are as followed (hardcoded though...)

```
    related=0.05;
for i in `ls ~/workspace/inheritance_estimate/rawData/*.fam | grep -v CC`;
do
    arrIN=(${i//\// });
    fileName=(${arrIN[5]});
    disease=(${fileName//./ });
    diseaseName=${disease[0]};
    echo "Working on $diseaseName";
    #Prune samples (Better for heterozygousity check and PCA)
    #We also perform the hwe with 1e-5 as the threshold as suggested by Clara
```

```
 plink --bfile WTCCC17000 --geno 0.01 --mind 0.95 --maf 0.05 --make-bed --
out WTCCC17000.temp.$diseaseName --threads 10 --indep 50 5 2 --keep $i >/
dev/null 2>&1;
 #Extract the Pruned SNPs and also calculate the heterozygousity rate
 plink --bfile WTCCC17000.temp.$diseaseName --extract WTCCC17000.temp.
$diseaseName.prune.in --ibc --out WTCCC17000.temp.$diseaseName --threads
10 >/dev/null 2>&1;
 #Identify problematic samples (Heterozygousity check)
 Rscript heteroCheck.R WTCCC17000.temp.$diseaseName.ibc WTCCC17000.temp.
$diseaseName.remove >/dev/null 2>&1;
 temp=$(wc -l WTCCC17000.temp.$diseaseName.remove | cut -f 1 -d " ");
 echo "${temp[0]} samples failed heterozygousity test";
 #Now perform the filtering and find all related samples
 plink --bfile WTCCC17000.temp.$diseaseName --remove WTCCC17000.temp.
$diseaseName.remove --extract WTCCC17000.temp.$diseaseName.prune.in --
genome --min $related --threads 10 --out WTCCC17000.temp.$diseaseName>/dev
/null 2>&1;
 ~/workspace/inheritance_estimate/programme/GreedyRelativeness WTCCC17000.
temp.$diseaseName.genome >> WTCCC17000.temp.$diseaseName.remove;
 #awk 'NR!=1 {print $1" "$2; print $3" "$4}' WTCCC17000.temp.$diseaseName.
genome | sort | uniq >> WTCCC17000.temp.$diseaseName.remove;
 temp=$(wc -l WTCCC17000.temp.$diseaseName.remove | cut -f 1 -d " ");
 echo "Total of ${temp[0]} samples removed after considering the
relationship";
 #Now try performing PCA with all related samples removed
 plink --bfile WTCCC17000.temp.$diseaseName --remove WTCCC17000.temp.
$diseaseName.remove --extract WTCCC17000.temp.$diseaseName.prune.in --
threads 10 --pca --out WTCCC17000.temp.$diseaseName >/dev/null 2>&1;
 Rscript plotPCA.R WTCCC17000.temp.$diseaseName.eigenvec WTCCC17000.temp.
$diseaseName >/dev/null 2>&1;
 temp=$(wc -l WTCCC17000.temp.$diseaseName.pc | cut -f 1 -d " ");
 echo "${temp[0]} samples seems to be outlier considering PCA plot";
 #Now we will try to also remove the samples from the PCA plot
 #It is important to note that we should always check the PCA plot
beforehand
 #Cannot be too automatic with this
 temp=$(wc -l WTCCC17000.temp.$diseaseName.pc | cut -f 1 -d " ");
 NUMOFLINES=${temp[0]};
 while [ $NUMOFLINES -gt 0 ]
 do
 cat WTCCC17000.temp.$diseaseName.remove WTCCC17000.temp.$diseaseName.pc >
WTCCC17000.temp.$diseaseName.remove.temp;
 mv WTCCC17000.temp.$diseaseName.remove.temp WTCCC17000.temp.$diseaseName.
remove;
 rm WTCCC17000.temp.$diseaseName.pc;
 plink --bfile WTCCC17000.temp.$diseaseName --remove WTCCC17000.temp.
$diseaseName.remove --extract WTCCC17000.temp.$diseaseName.prune.in --
threads 10 --pca --out WTCCC17000.temp.$diseaseName>/dev/null 2>&1;
 Rscript plotPCA.R WTCCC17000.temp.$diseaseName.eigenvec WTCCC17000.temp.
$diseaseName WTCCC17000.temp.$diseaseName>/dev/null 2>&1;
 temp=$(wc -l WTCCC17000.temp.$diseaseName.pc | cut -f 1 -d " ");
 NUMOFLINES=${temp[0]};
 echo "$NUMOFLINES samples seems to be outlier considering PCA plot";
 done
```

```
    #Finally, we want to re-examine the hardy weinberg equilibrium
    plink --bfile WTCCC17000.temp.$diseaseName --remove WTCCC17000.temp.
   $diseaseName.remove --geno 0.01 --mind 0.95 --maf 0.05 --hardy --out
   WTCCC17000.temp.$diseaseName.filter --make-bed >/dev/null 2>&1;
    wc -l WTCCC17000.temp.$diseaseName.filter.fam
done
```

After running all these scripts, we will now have the PCA plot for each samples. We should also have the information as to what samples should we exclude from the final data set.

```
    count=1;
    related=0.05;
for i in `ls *.temp*.fam`;
do
    disease=(${i//./ });
    diseaseName=${disease[2]};
    if [ $count -eq 1 ]
    then
    cat WTCCC17000.temp.$diseaseName.bim | cut -f 1,2 -d " " > WTCCC17000.temp
   .retainSnp;
    cat WTCCC17000.temp.$diseaseName.filter.fam > WTCCC17000.temp.retainSample
   ;
    else
    cat WTCCC17000.temp.$diseaseName.bim | cut -f 1,2 -d " " >> WTCCC17000.
   temp.retainSnp;
    sort WTCCC17000.temp.retainSnp | uniq -d > temp;
    mv temp WTCCC17000.temp.retainSnp;
    cat WTCCC17000.temp.$diseaseName.filter.fam>> WTCCC17000.temp.retainSample
   ;
    fi
    count=$((count+1));
done
    plink --bfile WTCCC17000 --geno 0.01 --mind 0.95 --maf 0.05 --hwe 1e-5 --
   keep WTCCC17000.temp.retainSample --extract WTCCC17000.temp.retainSnp --
   make-bed --out WTCCC17000.filter --indep 50 5 2 --threads 10
    #We also want to lable the sample by batch so that we can see if there is
   any batch problem
    ls *temp*.fam | awk -F "." '{print "awk \x27{print \""$3" \"\$2}\x27 "$0}'
    | bash |  awk 'NR==FNR{a[$2]=$1} NR!=FNR{print a[$2]" "$2" "$3" "$4" "$5"
    "$6}' - WTCCC17000.filter.fam > temp
    mv temp WTCCC17000.filter.fam
    rm -f *temp*
    #To avoid cross batch relatedness, we will again check the relationship
   between the samlpes
    plink --bfile WTCCC17000.filter --extract WTCCC17000.filter.prune.in --
   genome --min $related --out WTCCC17000.filter --threads 10
    ~/workspace/inheritance_estimate/programme/GreedyRelativeness  WTCCC17000.
   filter.genome > WTCCC17000.filter.remove;
    #awk 'NR!=1 {print $1" "$2; print $3" "$4}' WTCCC17000.filter.genome |
   sort | uniq > WTCCC17000.filter.remove;
    # Remove related sample and get the final result
```

```
 plink --bfile WTCCC17000.filter --remove WTCCC17000.filter.remove --
threads 10 --geno 0.01 --mind 0.95 --maf 0.05 --hwe 1e-5 --make-bed --out
WTCCC17000.final
 # We then generate the PCA to check that everything is indeed ok
 plink --bfile WTCCC17000.final --pca --extract WTCCC17000.filter.prune.in
--out WTCCC17000.final --threads 10
 Rscript plotPCA.R WTCCC17000.final.eigenvec  WTCCC17000.final
 cp WTCCC17000.final.fam WTCCC17000.final.bk.fam
```

And the R script for heterozygousity check is

```
 args = commandArgs(trailingOnly=TRUE)
 data=read.table(args[1])
 meanData=mean(data$Fhat2);
 sdData = sd(data$Fhat2);
 write.table(data[data$Fhat2 > (meanData+3*sdData), | data$Fhat2 <(meanData
-3*sdData) ], args[2], quote=F, row.names=F, col.names=F)
```

This is the plotPCA Rscript:

```
 args = commandArgs(trailingOnly=TRUE)
 require(ggplot2)
 data=read.table(args[1])
 png(paste(args[2],".png", sep=""))
 if(length(unique(data$V1))<5){
 ggplot(data, aes(x=V3, y=V4, color=V1))+geom_point()+theme(plot.title =
element_text(lineheight=.8, face="bold",size=24),axis.text=element_text(
size=16),axis.title=element_text(size=16,face="bold"))+labs(x="PC1",y="PC2
")+ggtitle(args[2]);
 }else{
 ggplot(data, aes(x=V3, y=V4))+geom_point()+theme(plot.title = element_text
(lineheight=.8, face="bold",size=24),axis.text=element_text(size=16),axis.
title=element_text(size=16,face="bold"))+labs(x="PC1",y="PC2")+ggtitle(
args[2]);
 }
 dev.off()
 #Here we will also output samples with problem
 pc1Mean = mean(data$V3)
 pc1SD = sd(data$V3)
 pc2Mean = mean(data$V4)
 pc2SD = sd(data$V4)
 problemSample = data[data$V3>pc1Mean+6*pc1SD|data$V3<pc1Mean-6*pc1SD |data
$V4>pc2Mean+6*pc2SD|data$V4<pc2Mean-6*pc2SD,]
 write.table(problemSample[,c(1,2)], paste(args[2], ".pc", sep=""),quote=F,
  col.names=F, row.names=F)
```

28 March 3, 2016

First, we should check that our samples are alright indeed ok and will not introduce any bias into our results. Therefore we need to first simulate a bunch of null traits and see if the resulting QQ plots follows the NULL

```
 data=read.table("WTCCC17000.final.fam")

 emilyQQ<-function(pvector, main=NULL,...){
      m=main
      observed <- sort(pvector)   #column name of the p-value column is "p"
      lobs <- -(log10(observed))
      n <- length(observed)
      j <- 1:n
      x <- (j-0.5)/n
      # We don't want the x-axis be too far
      plot(c(1), c(1), c(0,max(-log10(x)))+0.1, c(0,max(-log10(x),lobs))
+0.1, col="white", lwd=1, type="p", xlab=expression(Expected~~ -log[10](
italic(p))), ylab=expression(Observed~~-log[10](italic(p))),xaxs="i", yaxs
="i", bty="l",cex.lab=1.2, cex.axis=1.2, font.lab=2,las=1,main=m)
      abline(0,1)
      points(-log10(x), lobs, col="red", cex=0.5, lwd=1.5, pc=16)
      lines(-log10(x), -log10(qbeta( 0.05 , j , ( n - j + 1 ), ncp=observed)
 ),col="black",lty=2)
      lines(-log10(x), -log10(qbeta( 0.95 , j , ( n - j + 1 ), ncp=observed)
 ),col="black",lty=2)
 }

 for(i in 1:20){
      data$V6 = rnorm(nrow(data),mean=0, sd=1)
      write.table(data, "WTCCC17000.final.fam", row.names=F, col.names=F,
quote=F)
      system("plink --bfile WTCCC17000.final --assoc --out WTCCC17000.final"
)
      res = read.table("WTCCC17000.final.qassoc", header=T);
      png(paste("NullCheck_",i,".png",sep=""))
      emilyQQ(res$P)
      dev.off()
 }
```

Now once we have the correct NULL, we can start preparing the samples for simulation. The core concept is that because we have the whole population their GRM will not change, we can precompute the GRM for simulation. Therefore we can first calculate the GRM by running:

```
 # We try to reduce the simulation size by using only chromosome 1
 plink --bfile WTCCC17000.final --chr 1 --make-bed --out WTCCC17000.chr1
 gcta64 --bfile WTCCC17000.chr1 --autosome --maf 0.05 --make-grm --out
WTCCC17000.chr1.gcta --make-grm-gz --thread-num 10
```

Now the simulation script section 28. I want to be a bit greedy here where I will perform test on *all* condition together. The basic concept of this simulation will be “draw from population”. So, we first simulate a population given a number of information (e.g. causal SNPs and effect Size), then from this population, we randomly draw N samples for the simulation and repeat this procedure 100 times.

```
# This will likely be a very long code...
# First, we need to get the required programmes
shrek="~/workspace/inheritance_estimate/programme/Shrek/shrek";
gcta="gcta64";
ldsc="~/programmes/genotype/ldsc/";

# Now the simulation meta information
nSituation = 10; # Number of different causal SNP set
repeats = 100; # Number of repeats for each causal SNP set
nRef = 500; # Number of reference use for LDSC and SHREK
nSample =1000;
causalSnps = c(10,50,100,1000);
heritability = seq(0,0.9,0.1);
prevalence = c(0.5,0.25,0.1);
observedPrevalence=0.5;

# Now the required fam file
fam = read.table("WTCCC17000.chr1.fam");

# Now the simulation code:
for(situation in 1:nSituation){
    #First simulate the reference
    ref=sort(sample(1:nrow(fam),size=nRef,replace = FALSE));
}
# Just keep it here as it seems like GCTA can also calculate the LDSC
# gcta64 --bfile test --ld-score --ld-wind 1000 --ld-score-adj --out test
```

29 March 4, 2016

Johnny wants us to use 0.05 as relationship cutoff. This will lead to a large amount of samples being removed, therefore it is better if we use the greedy approach (Try to retain one of the sample in relationship pair).

I wrote a C++ programme for it and it works nicely

```
#include <stdio.h>
#include <string>
#include <fstream>
#include <vector>
#include <boost/ptr_container/ptr_vector.hpp>
#include <assert.h>
#include <map>
#include "usefulTools.h"

class Sample{
public:
    Sample(const std::string name, const std::string fam): m_name(name),m_fam(
   fam){}
    void addSample(std::string target, double pihat ){
        related.push_back(target);
        piHat.push_back(pihat);
        sum+= pihat;
        sizeOfRelatives++;
    };
    size_t getSum() const{ return sum;};
    size_t getSize() const{ return sizeOfRelatives; };
    std::string getName() const {return m_name; };
    std::string getFam() const {return m_fam; };
    void removeSamples(std::map<std::string, int> &index, boost::ptr_vector<
   Sample> &samples){
        for(size_t i = 0; i < related.size(); ++i){
            assert(index.find(related[i])!=index.end());
            int loc = index[related[i]];
            if(loc != -1){
                samples[loc].update(piHat[i]);
            }
        }
    }

    void update(double pihat){
        sum-=pihat;
        sizeOfRelatives--;
    }
private:
    std::vector<std::string> related;
    std::vector<double> piHat;
    std::string m_name="";
    std::string m_fam="";
    double sum=0.0;
    size_t sizeOfRelatives=0;
};

int main(int argc, char *argv[]){
    if(argc !=2){
        fprintf(stderr, "Usage: %s <Relationship File>\n", argv[0]);
        return EXIT_FAILURE;
    }
    // When updating the sequence, we actually know that the samples will only
     be sorted down-wards, so our performance will get better and better
    std::ifstream input;
    input.open(argv[1]);
    if(!input.is_open()){
        fprintf(stderr, "Cannot open input file: %s\n", argv[1]);
        return EXIT_FAILURE;
    }

    std::map<std::string, int> indexOfSample; // For quick identification of
   the sample
    boost::ptr_vector<Sample> samples; // Storing the samples

    std::string line;
    std::getline(input, line);//remove header
    while(std::getline(input, line)){
        line= usefulTools::trim(line);
        std::vector<std::string> token;
        if(!line.empty()){
            usefulTools::tokenizer(line, "\t ",&token);
            if(token.size() > 9){
                double pihat = atof(token[9].c_str());
                std::string fam1 = token[0];
                std::string iid1 = token[1];
                std::string fam2 = token[2];
                std::string iid2 = token[3];
                if(indexOfSample.find(iid1)==indexOfSample.end()){
                    indexOfSample[iid1] = samples.size();
                    samples.push_back(new Sample(iid1, fam1));
                }
                if(indexOfSample.find(iid2)==indexOfSample.end()){
                    indexOfSample[iid2] = samples.size();
                    samples.push_back(new Sample(iid2, fam2));
                }
                // Now both iid1 and iid2 must be present
                samples[indexOfSample[iid1]].addSample(iid2, pihat);
                samples[indexOfSample[iid2]].addSample(iid1, pihat);
            }
        }
    }
    std::vector<std::string> removeSample;
    std::vector<std::string> removeFam;
    size_t sizeSample = samples.size();
    for(size_t i = 0; i < sizeSample; ++i){
        size_t maxSize = 0;
        size_t maxSum = 0;
        size_t index = i;
        std::string maxName = "";
        std::string maxFam = "";
        for(size_t j = i; j < sizeSample; ++j){
            //Find the maximum sample
            size_t currentSize = samples[j].getSize();
            if(currentSize> maxSize){
                maxSize = currentSize;
                maxSum = samples[j].getSum();
                maxName = samples[j].getName();
                maxFam = samples[j].getFam();
                index = j;
            }
            else if(currentSize == maxSize){
                size_t currentSum = samples[j].getSum();
                if(currentSum > maxSum){
                    maxSize = currentSize;
                    maxSum = currentSum;
                    maxName = samples[j].getName();
                    maxFam = samples[j].getFam();
                    index = j;
                }
            }
        }
        // If max size equals 0, we know we have remove most samples already
        if(maxSize==0) break;
        else{
            // here we are going to remove the samples
            samples[index].removeSamples(indexOfSample, samples); //update the
    counts and stuff
            removeSample.push_back(maxName); //store samples
            removeFam.push_back(maxFam); //store samples
            indexOfSample[maxName] = -1; //update
            if(index != i){
                //supposedly swap the location of the i and j element
                samples.replace(index, samples.replace(i, &samples[index]).
   release()).release();
            }
        }
    }
    for(size_t i = 0; i < removeSample.size(); ++i){
        fprintf(stdout, "%s\t%s\n", removeFam[i].c_str(), removeSample[i].
   c_str());
    }
    return EXIT_SUCCESS;
}
```

30 March 8, 2016

As we have moved the codes to a makefile which contain most of the commenting. Therefore, from now on, codes will not be stored here.

Completed the simulation script, now need to check whether if the output is correct.

For SHREK, the random stopping problem is still here, yet we fail to identify the reason behind the problem...

31 March 14, 2016

A note to myself: if we are going to build another server account in the future, put all the libraries in the same folder instead of putting them into their own sub folders for easy linking (can use soft link). e.g. libblas.so in LIB instead of in LIB/BLAS

When installing LAPACK, make sure we use the cmake function:

```
 cmake -DCMAKE_INSTALL_PREFIX=/home/sam/programmes/lib/lapack-3.6.0 -
DCMAKE_BUILD_TYPE=RELEASE -DBUILD_SHARED_LIBS=ON ../
```

32 Useful Resources

Description	URL
Timing the functions in c++ programme	link