SKAT Package

Seunggeun (Shawn) Lee

October 8, 2012

1 Overview

SKAT package contains functions to 1) test an association between SNP sets and continuous/discrete phenotypes and 2) compute power/sample size for future sequence association studies. This document provides tutorial for using SKAT.

2 Testing association between SNP sets and outcome phenotypes.

2.1 Example Dataset

SKAT package provides an example dataset (SKAT.example) which has a genotype matrix (Z) of 2000 individuals and 67 SNPs, a vector of continuous phenotypes (y.c), a vector of binary phenotypes (y.b) and a covariates matrix (X).

```
> library(SKAT)
> data(SKAT.example)
> names(SKAT.example)
```

```
[1] "Z"   "X"   "y.c" "y.b"
```

```
> attach(SKAT.example)
```

To test association, you first need to run SKAT_Null_Model function to obtain parameters and residuals from the null model of no association, and run SKAT to get a p-value.

```
> # continuous trait
> obj<-SKAT_Null_Model(y.c ~ X, out_type="C")
> SKAT(Z, obj)$p.value
```

```
[1] 0.002877041
```

```
> # dichotomous trait
> obj<-SKAT_Null_Model(y.b ~ X, out_type="D")
> SKAT(Z, obj)$p.value
```

```
[1] 0.1401991

>
```

When the trait is binary and the sample size is small, SKAT can produce conservative results. To address this, we recently developed a small sample adjustment method that adjusts the asymptotic null distribution by estimating small sample moments. By default, SKAT ($>=$ ver 0.7) will conduct a small sample adjustment when the sample size $<$ 2000. In the following code, we only use 200 samples to run SKAT.

```
> IDX<-c(1:100,1001:1100)
> # With-adjustment
> obj.s<-SKAT_Null_Model(y.b[IDX] ~ X[IDX,],out_type="D")

Sample size = 200, which is < 2000. The small sample adjustment is applied!

> SKAT(Z[IDX,], obj.s, kernel = "linear.weighted")$p.value

[1] 0.1327999

>
```

If you don't want to use the adjustment, please set Adjustment=FALSE when you run the SKAT_Null_Model function.

```
> # Without-adjustment
> obj.s<-SKAT_Null_Model(y.b[IDX] ~ X[IDX,],out_type="D", Adjustment=FALSE)
> SKAT(Z[IDX,], obj.s, kernel = "linear.weighted")$p.value

[1] 0.147093
```

2.2 Assign weights for each SNP

It is generally assumed that rarer variants have larger effect sizes. To incorporate it, the linear weighted kernel is formulated as ZWZ', where Z is a genotype matrix, and $W = diag\{w_1, \ldots, w_m\}$ is a weight matrix. In the previous examples, we use the default beta(1,25) weight that is $\sqrt{w_i} = dbeta(p_i, 1, 25)$, where $dbeta$ is the beta density function, and p_i is the minor allele frequncy (MAF) of the i^{th} SNP. If you want to use the beta weight with different parameters, you can change the weights.beta parameter. For example, if you want to use Madsen and Browning type of weight, use weight.beta=c(0.5,0.5).

```
> SKAT(Z, obj, kernel = "linear.weighted", weights.beta=c(0.5,0.5))$p.value

[1] 0.4931639
```

If you want to use different types of weights, you should make your own weight vector and use it as weights parameter. Keep in mind that each elements of the weight vector should be $\sqrt{w_i}$, not w_i. For the logistic weight, we provide a function that generates it.

```
> # Shape of the logistic weight
>
> MAF<-1:1000/1000
> W<-Get_Logistic_Weights_MAF(MAF, par1=0.07, par2=150)
> par(mfrow=c(1,2))
> plot(MAF,W,xlab="MAF",ylab="Weights",type="l")
> plot(MAF[1:100],W[1:100],xlab="MAF",ylab="Weights",type="l")
> par(mfrow=c(1,2))
> # Use logistic weight
> weights<-Get_Logistic_Weights(Z, par1=0.07, par2=150)
> SKAT(Z, obj, kernel = "linear.weighted", weights=weights)$p.value

[1] 0.3293643
```

2.3 Unified Test

The test statistic of the unified test is

$$Q_\rho = (1-\rho)Q_S + \rho Q_B,$$

where Q_S is a test statistic of SKAT, and Q_B is a score test statistic of weighted burden test. Thus, $\rho = 0$ results in the original weighted linear kernel SKAT, and $\rho = 1$ results in the weighted burden test. You can specify ρ value using the r.corr parameter (default: , r.corr=0).

```
> # Shape of the logistic weight
>
> #rho=0
> SKAT(Z, obj, r.corr=0)$p.value

[1] 0.1401991

> #rho=0.9
> SKAT(Z, obj, r.corr=0.9)$p.value

[1] 0.06031026
```

If method="optimal", ρ is selected from an equal sized grid of 11 points (from 0 to 1) to maximize the power.

```
> #Optimal Test
> SKAT(Z, obj, method="optimal")$p.value

[1] 0.1037306

>
```

2.4 Imputing missing genotypes.

If there are missing genotypes, SKAT automatically imputes them based on Hardy-Weinberg equilibrium. You can choose either "random" or "fixed" imputation (default="fixed"). For "random" imputation, SKAT generates binomial(2,p_i) random numbers to impute missing values, where p_i is the MAF of the i^{th} SNP calculated from non-missing genotypes. In the SKAT paper, we used the "random" imputation. Since imputed values are randomly genereated, SKAT can produce different p-values for different runs. For "fixed" imputation, SKAT uses the mean genotype value, $2p_i$, to impute missing values.

```
> # Assign missing
> Z1<-Z
> Z1[1,1:3]<-NA
> # random imputation
> SKAT(Z1,obj,impute.method = "random")$p.value

[1] 0.1401991

> # fixed imputation
> SKAT(Z1,obj,impute.method = "fixed")$p.value

[1] 0.1401982
```

2.5 Resampling

SKAT package provides functions to conduct resampling methods to compute resampling p-values and to control family wise error rate. Two different resampling methods are implemented. "bootstrap" conducts the parametric bootstrap to resample residuals from H_0 with considering covariates. When there is no covariate, "bootstrap" is equivalent to the permutation method. "perturbation" perturbs the residuals by multiplying mean zero and variance one normal random variables. The default method is "bootstrap". From ver 0.7, we do not provide the "perturbation" method.

```
> # parametric boostrap.
> obj<-SKAT_Null_Model(y.b ~ X, out_type="D", n.Resampling=5000,
+ type.Resampling="bootstrap")
> # SKAT p-value
> re<- SKAT(Z, obj, kernel = "linear.weighted")
> re$p.value        # SKAT p-value

[1] 0.1401991

> Get_Resampling_Pvalue(re)        # get resampling p-value

$p.value
[1] 0.1339732
```

```
$is_smaller
[1] FALSE
```

When there are many genes/SNP sets to test, resampling methods can be used to control family-wise error rate. You can find an example in the next section.

2.6 Plink Binary format files

SKAT package can read plink binary format files for genome-wide data analysis. To use plink files, you need plink bed, bim and fam files, and your own setid file that contains information of SNP sets. Example files can be found on the SKAT webpage. You need to generate the SSD and Info files first.

```
> # To run this code, first download and unzip example files
>
> ##############################################
> #         Generate SSD file
>
> # Create the MW File
> File.Bed<-"./Example1.bed"
> File.Bim<-"./Example1.bim"
> File.Fam<-"./Example1.fam"
> File.SetID<-"./Example1.SetID"
> File.SSD<-"./Example1.SSD"
> File.Info<-"./Example1.SSD.info"
> # To use binary ped files, you have to generate SSD file first.
> # If you already have a SSD file, you do not need to call this function.
> Generate_SSD_SetID(File.Bed, File.Bim, File.Fam, File.SetID, File.SSD, File.Info)
```

```
1000 Samples, 10 Sets, 984 Total SNPs
[1] "SSD and Info files are created!"
```

Now you can open SSD and Info file and run SKAT. After finishing using it,
you must call close function to clse SSD file.

```
> FAM<-Read_Plink_FAM(File.Fam, Is.binary=FALSE)
> y<-FAM$Phenotype
> # To use a SSD file, please open it first. After finishing using it, you must close it.
>
> SSD.INFO<-Open_SSD(File.SSD, File.Info)
```

```
1000 Samples, 10 Sets, 984 Total SNPs
Open the SSD file
```

```
> # Number of samples
> SSD.INFO$nSample
```

```
[1] 1000

> # Number of Sets
> SSD.INFO$nSets

[1] 10

> obj<-SKAT_Null_Model(y ~ 1, out_type="C")
> out<-SKAT.SSD.All(SSD.INFO, obj)
> out

$results
     SetID    P.value N.Marker.All N.Marker.Test
1  GENE_01 0.77747880           94            94
2  GENE_02 0.06245208           84            84
3  GENE_03 0.38416582          108           108
4  GENE_04 0.46179268          101           101
5  GENE_05 0.18548863          103           103
6  GENE_06 0.93255760           94            94
7  GENE_07 0.18897220          104           104
8  GENE_08 0.73081683           96            96
9  GENE_09 0.67366458          100           100
10 GENE_10 0.40310682          100           100

$P.value.Resampling
NULL

attr(,"class")
[1] "SKAT_SSD_ALL"
```

If you have more than one gene/SNP set to test an association, you should adjust multiple testing. It can be done either by conducting bonferroni correction or by estimating false discovery rate. However, if gene/SNP sets are correlated, these approaches would produce conservative results. Alternatively, you can directly control family wise error rate (FWER) using the resampling method. Example code is given in following.

```
> obj<-SKAT_Null_Model(y ~ 1, out_type="C", n.Resampling=1000, type.Resampling="bootstrap")
> out<-SKAT.SSD.All(SSD.INFO, obj)
> # No gene is significant with controling FWER = 0.05
> Resampling_FWER(out,FWER=0.05)

$result
NULL

$n
[1] 0
```

```
$ID
NULL

> # 1 gene is significnat with controling FWER = 0.5
> Resampling_FWER(out,FWER=0.5)

$result
    SetID    P.value N.Marker.All N.Marker.Test
2 GENE_02 0.06245208           84            84

$n
[1] 1

$ID
[1] 2
```

If you want to test a single gene/SNP set, not all genes/SNP sets, you can use either "SKAT.SSD.OneSet" or "SKAT.SSD.OneSet_SetIndex". Or you can get a genotype matrix using "Get_Genotypes_SSD" function and then run SKAT. If you want to use different types of weights (ex. logistic weights), you should use this approach.

```
> obj<-SKAT_Null_Model(y ~ 1, out_type="C")
> # test the second gene
> id<-2
> SetID<-SSD.INFO$SetInfo$SetID[id]
> SKAT.SSD.OneSet(SSD.INFO,SetID, obj)$p.value

[1] 0.06245208

> SKAT.SSD.OneSet_SetIndex(SSD.INFO,id, obj)$p.value

[1] 0.06245208

> # test the second gene with the logistic weight.
> Z<-Get_Genotypes_SSD(SSD.INFO, id)
> weights = Get_Logistic_Weights(Z, par1=0.07, par2=150)
> SKAT(Z, obj, weights=weights)$p.value

[1] 0.7227001

>
```

After finishing, please close the SSD file.

```
> Close_SSD()

Close the opened SSD file: /private/var/folders/zs/nf_6qpd12r1dm4v3y2y298fr0000gn/T/Rtmp64dA
```

3 Power/Sample Size calculation.

3.1 Dataset

SKAT package provides a haplotype dataset (SKAT.haplotypes) which contains a haplotype matrix of 10,000 haplotypes over 200kb region (Haplotype), and a dataframe with informations of each SNP. These haplotypes were simulated using a calibrated coalescent model with mimicking linkage disequilibrium structure of European ancestry. If you don't have any haplotype information, please use this dataset to compute power/sample size.

```
> data(SKAT.haplotypes)
> names(SKAT.haplotypes)
```

```
[1] "Haplotype" "SNPInfo"
```

```
> attach(SKAT.haplotypes)
```

3.2 Power/Sample Size calculation

SKAT package provides functions to compute the power/sample size for future sequence association studies. In the following example, we conducted sample size calculation using the haplotypes in SKAT.haplotypes with the following parameters.

1. Subregion length = 3k bp
2. Causal percent = 20%
3. Negative percent = 20%
4. For continuous traits, $\beta = c|log_{10}(MAF)|$ (BetaType = "Log") with $\beta = 2$ at MAF = 10^{-4}
5. For binary traits, $log(OR) = c|log_{10}(MAF)|$ (OR.Type = "Log") with OR = 2 at MAF = 10^{-4}, and 50% of samples are cases and 50% of samples are controls

```
> set.seed(500)
> out.c<-Power_Continuous(Haplotype,SNPInfo$CHROM_POS, SubRegion.Length=5000,
+ Causal.Percent= 20, N.Sim=10, MaxBeta=2,Negative.Percent=20)
```

```
[1] "10/10"
```

```
> out.b<-Power_Logistic(Haplotype,SNPInfo$CHROM_POS, SubRegion.Length=5000,
+ Causal.Percent= 20, N.Sim=10 ,MaxOR=7, Negative.Percent=20)
```

```
[1] "10/10"
```

```
> out.c
```

```
$Power
          0.01     0.001     1e-06
500  0.5601495 0.4507543 0.2745436
1000 0.6983510 0.6372979 0.4477310
1500 0.7393476 0.6978347 0.5840998
2000 0.7741144 0.7169529 0.6649380
2500 0.8041370 0.7386689 0.6938517
3000 0.8224103 0.7660432 0.6997755
3500 0.8349515 0.7896737 0.7015918
4000 0.8484832 0.8037123 0.7049269
4500 0.8647970 0.8109526 0.7122846
5000 0.8834324 0.8165985 0.7253563

$R.sq
[1] 0.0693529

attr(,"class")
[1] "SKAT_Power"

> out.b

$Power
          0.01     0.001     1e-06
500  0.3894872 0.2757429 0.1330505
1000 0.5888308 0.4573657 0.2436726
1500 0.7021843 0.5859396 0.3485361
2000 0.7763091 0.6650800 0.4668508
2500 0.8234240 0.7280271 0.5483447
3000 0.8516985 0.7775865 0.5943673
3500 0.8718116 0.8108489 0.6269605
4000 0.8899993 0.8317031 0.6603647
4500 0.9081573 0.8464714 0.6968862
5000 0.9262225 0.8594656 0.7324297

attr(,"class")
[1] "SKAT_Power"

> Get_RequiredSampleSize(out.c, Power=0.8)

$`alpha = 1.00e-02`
[1] 2431.102

$`alpha = 1.00e-03`
[1] 3867.782

$`alpha = 1.00e-06`
[1] "> 5000"
```

```
> Get_RequiredSampleSize(out.b, Power=0.8)

$`alpha = 1.00e-02`
[1] 2251.417

$`alpha = 1.00e-03`
[1] 3336.919

$`alpha = 1.00e-06`
[1] "> 5000"

>
```

In this example, we used N.Sim=10 to get results quickly. When you do the power calculation, please increase it to more than 100. When BetaType = "Log" or OR.Type = "Log", the effect size of continuous trait and the log odds ratio of binary traits are $c|log_{10}(MAF)|$, where c is determined by Max_Beta or Max_OR. For example, $c = 2/4 = 0.5$ when the Max_Beta = 2. In this case, a causal variant with MAF=0.01 has $\beta = 1$. For binary traits, $c = log(7)/4 = 0.486$ with MAX_OR=7. And thus, a causal variant with MAF=0.01 has log OR = 0.972.

If different r.corr (ρ) values to be used to compute the power, Power_Continuous_R or Power_Logistic_R functions can be used instead. For example, r.corr=0 is SKAT and r.corr=1 is a burden test. Since they use slightly different method to compute the power, the powers from Power_Continuous_R and Power_Logistic_R can be slightly different from the powers from Power_Continuous and Power_Logistic although r.corr=0.

If you want to computer the power of SKAT-O by estimating the optimal r.corr, use r.corr=2. The estimated optimal r.corr is

$$r.corr = p_1^2(2p_2 - 1)^2,$$

where p_1 is the proportion of nonzero βs, and p_2 is the proportion of negative (or positive) βs among the non-zero βs.

```
> set.seed(500)
> out.c<-Power_Continuous_R(Haplotype,SNPInfo$CHROM_POS, SubRegion.Length=5000,
+ Causal.Percent= 20, N.Sim=10, MaxBeta=2,Negative.Percent=20, r.corr=2)

[1] "10/10"

> out.c

$Power
          0.01     0.001     1e-06
500  0.5584048 0.4465557 0.2700370
1000 0.6980094 0.6374870 0.4403217
1500 0.7367947 0.6977547 0.5830013
```

```
2000 0.7707641 0.7148115 0.6664808
2500 0.8032711 0.7341910 0.6946357
3000 0.8253110 0.7606592 0.6998229
3500 0.8407660 0.7863270 0.7011542
4000 0.8569269 0.8038311 0.7035340
4500 0.8759197 0.8137950 0.7089662
5000 0.8968032 0.8214246 0.7192218

$R.sq
[1] 0.0693529

$r.corr
[1] 0.0144

attr(,"class")
[1] "SKAT_Power"

> Get_RequiredSampleSize(out.c, Power=0.8)

$`alpha = 1.00e-02`
[1] 2449.686

$`alpha = 1.00e-03`
[1] 3890.566

$`alpha = 1.00e-06`
[1] "> 5000"

>
```