Package: AssocAFC (via r-universe)

September 6, 2024

Type Package

Title Allele Frequency Comparison

Version 1.0.2

Author Khalid B. Kunji [aut, cre], Mohamad Saad [aut]

Maintainer Khalid B. Kunji <kkunji@hbku.edu.qa>

Description When doing association analysis one does not always have the genotypes for the control population. In such cases it may be necessary to fall back on frequency based tests using well known sources for the frequencies in the control population, for instance, from the 1000 Genomes Project. The Allele Frequency Comparison ('AssocAFC') package performs multiple rare variant association analyses in both population and family-based GWAS (Genome-Wide Association Study) designs. It includes three score tests that are based on the difference of the sum of allele frequencies between cases and controls. Two of these tests, Wcorrected() and Wqls(), are collapsing-based tests and suffer from having protective and risk variants. The third test, afcSKAT(), is a score test that overcomes the mix of SNP (Single-Nucleotide Polymorphism) effect directions. For more details see Saad M and Wijsman EM (2017) <doi:10.1093/bib/bbx107>.

License GPL (>= 3)

URL https://www.r-project.org, https://doi.org/10.1093/bib/bbx107

Imports CompQuadForm

Depends R (>= 3.2.0)

Encoding UTF-8

LazyData true

KeepSource TRUE

NeedsCompilation no

Date/Publication 2018-05-05 09:34:22 UTC

Repository https://kkunji.r-universe.dev

RemoteUrl https://github.com/cran/AssocAFC

RemoteRef HEAD

RemoteSha 82afb6d83d99bc9e2373de41396fe2ca7bf33612

Contents

`afcSKAT` *Allele Frequency Comparison Sequence Kernel Association Test*

Description

Sequence Kernel Association Test, afcSKAT(), for multiple rare variant association test using the difference of sum of minor allele frequencies between cases and controls. This test handles related individuals, unrelated individuals, or both. Plus, it is robust againt the inclusion of both protective and risk variants in the same model.

Usage

```
afcSKAT(MAF, Pheno, Kin, Correlation, Weights)
```

Arguments

`MAF`	matrix (#Snps * 2): First column contains Minor Allele Frequency (MAF) in cases; Second column contains MAF in controls.
`Pheno`	matrix (#subjects * 1): this one-column matrix contains 0's amd 1's: 1 for cases and 0 for controls. No missing values are allowed.
`Kin`	The kinship matrix (#subjects * #subjects): the subjects must be ordered as the Pheno variable.
`Correlation`	Correlation matrix between SNPs (#Snps * #Snps). The user should calculate this matrix beforehand. Either based on own genotype data (in cases, controls, or both) or based on public databases (e.g., 1000 Genomes Projects, ESP, etc.). NA values are not allowed. They have to be replaced by zeros.

`Weights` The weights values that can be used to up-weight or down-weight SNPs. This size of this vector is the number of Snps by 1. By default, the weights are 1 for all Snps.

Value

A vector with two values is returned, the 1st is the p-value calculated by the Satterwaite method and the second is calculated via Davies.

References

Saad M and Wijsman EM, Association score testing for rare variants and binary traits in family data with shared controls, Briefings in Bioinformatics, 2017. Schaid DJ , McDonnell SK , Sinnwell JP , et al. Multiple genetic variant association testing by collapsing and kernel methods with pedigree or population structured data. Genet Epidemiol 2013 ;37 :409 –18.

See Also

`AssocAFC` `Wqls` `Wcorrected`

Examples

```
P_afcSKAT_Satterwaite <- vector("numeric")
P_afcSKAT_Davies <- vector("numeric")
#This data corresponds to what is used in the 1st iteration with the raw data
data("maf.afc")
data("phenotype.afc")
data("kin.afc")
data("cor.afc")
data("weights.afc")
SKAT <- afcSKAT(MAF = maf.afc , Pheno = phenotype.afc, Kin = kin.afc , Correlation=cor.afc,
                Weights = weights.afc)
P_afcSKAT_Satterwaite <- c(P_afcSKAT_Satterwaite, SKAT[1])
P_afcSKAT_Davies <- c(P_afcSKAT_Davies, SKAT[2])
print(P_afcSKAT_Satterwaite)
print(P_afcSKAT_Davies)

## Not run:
#This example shows processing the raw data and uses kinship2,
#which AFC does not depend on

library(kinship2)
library(CompQuadForm)

P_afcSKAT_Satterwaite <- vector("numeric")
P_afcSKAT_Davies <- vector("numeric")

for (j in 1:10)
{
  geno.afc <- read.table(system.file("extdata", "Additive_Genotyped_Truncated.txt",
                         package = "AFC"), header = TRUE)
  geno.afc[ , "IID"] <- paste(geno.afc[ , "FID"]  , geno.afc[ , "IID"]  ,sep=".")
```

```
  geno.afc[geno.afc[,"FA"]!=0 , "FA"] <- paste(geno.afc[geno.afc[,"FA"]!=0 , "FID"],
                                        geno.afc[geno.afc[,"FA"]!=0 , "FA"]  ,sep=".")
  geno.afc[geno.afc[,"FA"]!=0 , "MO"] <- paste(geno.afc[geno.afc[,"FA"]!=0 , "FID"],
                                        geno.afc[geno.afc[,"FA"]!=0 , "MO"]  ,sep=".")
  Kinship <- makekinship(geno.afc$FID , geno.afc$IID , geno.afc$FA, geno.afc$MO)
  kin.afc <- as.matrix(Kinship)
  pheno.afc <- read.table(system.file("extdata", "Phenotype", package = "AFC"))
  phenotype.afc <- matrix(pheno.afc[,j],nc=1,nr=nrow(pheno.afc))
  geno.afc <- geno.afc[,7:ncol(geno.afc)]
  Na <- nrow(pheno.afc[pheno.afc[,j]==1,])
  Nu <- nrow(pheno.afc[pheno.afc[,j]==0,])
  N <- Nu + Na
  maf.afc <- matrix(NA , nr=ncol(geno.afc) , nc=2)
  maf.afc[,1] <- colMeans(geno.afc[phenotype.afc==1,])/2;
  maf.afc[,2] <- colMeans(geno.afc[phenotype.afc==0,])/2;
  P  <- (maf.afc[,1]*Na + maf.afc[,2]*Nu)/N
  Set <- which(P<0.05)
  maf.afc <- maf.afc[c(Set),]
  cor.afc <- cor(geno.afc[,c(Set)])
  cor.afc[is.na(cor.afc)] <- 0
  geno.afc <- as.matrix(geno.afc[,Set])

  weights.afc <- matrix(1/(maf.afc[,2]+1),nc=1,nr=length(Set))
 SKAT <- afcSKAT(MAF = maf.afc , Pheno = phenotype.afc, Kin = kin.afc , Correlation=cor.afc,
                 Weights = weights.afc)
  P_afcSKAT_Satterwaite <- c(P_afcSKAT_Satterwaite, SKAT[1])
  P_afcSKAT_Davies <- c(P_afcSKAT_Davies, SKAT[2])
}
print(P_afcSKAT_Satterwaite)
print(P_afcSKAT_Davies)

## End(Not run)

## The function is currently defined as
function(MAF, Pheno, Kin, Correlation, Weights)
{
  Na     <- length(Pheno[Pheno[,1]==1,])
  Nu     <- length(Pheno[Pheno[,1]==0,])
  N      <- Na + Nu

  # The three following lines: prepare the phenotype variables
  OneN  <- matrix(1, ncol=1, nrow = N)
  Y  <- Pheno
  OneHat <- matrix( Na/N, ncol=1 , nrow=N)

  # Estimate MAF in all subjects
  P  <- (MAF[,1]*Na + MAF[,2]*Nu)/N
  if (is.null(Weights))
  {
    # Variance of SNPs (2p(1-p))
    VarSnps <- sqrt(P*(1-P))
  } else
  {
```

```
    # Variance of SNPs (2p(1-p)) accounting for the prespecified Snp weights
    VarSnps <- Weights*sqrt(P*(1-P))
  }
  VarSnps <- matrix(VarSnps,ncol=1)
  cz <- 2* sum((Y - OneHat) %*% t(Y - OneHat) * Kin )
  Vz <- cz * VarSnps %*% t(VarSnps)*Correlation
  if (is.null(Weights))
  {
   # Quadratic form without Weights
    Q <-  4*Na^2*(sum ( (MAF[,1] - P)^2 ))
  } else
  {
   # Quadratic form with Weights
    Q <-  4*Na^2*(sum ( Weights*(MAF[,1] - P)^2 ))
  }
  # Satterwaite approximation (1)
  E_Q <- sum(diag(Vz))
  # Satterwaite approximation (2)
  V_Q <- 2* sum( diag (Vz %*% Vz))
  # Satterwaite approximation (3)
  Delta <- V_Q/(2*E_Q)
  # Satterwaite approximation (4)
  df<- 2*E_Q^2/V_Q
  # Satterwaite approximation (5)
  Qscaled <- Q / Delta
  # Satterwaite approximation (6)
  Pvalue_Sat <- 1-pchisq(Qscaled, df)

 # Davies approximation (1)
  eig <- eigen(Vz, symmetric = T, only.values = T)
  # Davies approximation (2)
  evals <- eig$values[eig$values > 1e-06 * eig$values[1]]
  # Davies approximation (3)
  Pvalue_Dav <-davies(Q, evals, acc = 1e-5)$Qq
  out <- t(data.frame(c(Pvalue_Sat,Pvalue_Dav)))
  colnames(out)<- c("Satterwaite","Davies")
  rownames(out) <- "Pvalue"
  return(out)
}
```

AssocAFC *Allele Frequency Comparison*

Description

When doing association analysis one does not always have the genotypes for the control population. In such cases it may be necessary to fall back on frequency based tests using well known sources for the frequencies in the control population, for instance, from the 1000 Genomes Project. The Allele Frequency Comparison ('AssocAFC') package performs multiple rare variant association analyses in both population and family-based GWAS (Genome-Wide Association Study) designs. It includes

three score tests that are based on the difference of the sum of allele frequencies between cases and controls. Two of these tests, Wcorrected() and Wqls(), are collapsing-based tests and suffer from having protective and risk variants. The third test, afcSKAT(), is a score test that overcomes the mix of SNP (Single-Nucleotide Polymorphism) effect directions. For more details see Saad M and Wijsman EM (2017) <doi:10.1093/bib/bbx107>.

Author(s)

Khalid B. Kunji [aut, cre], Mohamad Saad [aut]

Maintainer: Khalid B. Kunji <kkunji@hbku.edu.qa>

References

Mohamad Saad, Ellen M. Wijsman; Association score testing for rare variants and binary traits in family data with shared controls, Briefings in Bioinformatics, , bbx107, https://doi.org/10.1093/bib/bbx107

See Also

`Wcorrected` `Wqls` `afcSKAT`

`cor.afc` *AFC Correlation Example*

Description

An example correlation matrix that corresponds to the the one derived in the unrun example's first iteration.

Usage

```
data("cor.afc")
```

Format

The format is: num [1:75, 1:75] 1 0.2153 0.14829 -0.00285 0.0536 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:75] "rs3745120_1" "chr19.22817596_1" "chr19.22817734_1" "chr19.22817955_1" ... ..$: chr [1:75] "rs3745120_1" "chr19.22817596_1" "chr19.22817734_1" "chr19.22817955_1" ...

Source

This is simulated data.

Examples

```
data("cor.afc")
```

geno.afc *AFC Genotype Example*

Description

An example genotype file that corresponds to the the one derived in the unrun example's first iteration.

Usage

```
data("geno.afc")
```

Format

The format is: int [1:1800, 1:75] 0 0 0 0 0 0 0 0 0 0 ... - attr(*, "dimnames")=List of 2 ..$: NULL ..$: chr [1:75] "rs3745120_1" "chr19.22817596_1" "chr19.22817734_1" "chr19.22817955_1" ...

Source

This is simulated data.

Examples

```
data("geno.afc")
```

kin.afc *AFC Kinship Example*

Description

An example kinship matrix that corresponds to the the one derived in the unrun example's first iteration.

Usage

```
data("kin.afc")
```

Format

The format is: num [1:1800, 1:1800] 0.5 0 0.25 0.25 0.25 0.25 0.25 0.25 0.25 0.25 ... - attr(*, "dimnames")=List of 2 ..$: chr [1:1800] "1.1" "1.2" "1.3" "1.4" ... ..$: chr [1:1800] "1.1" "1.2" "1.3" "1.4" ...

Source

This is simulated data.

Examples

```
data("kin.afc")
```

`maf.afc` *AFC MAF Example*

Description

An example MAF (Mean Allele Frequency) file that corresponds to the the one derived in the unrun example's first iteration.

Usage

```
data("maf.afc")
```

Format

The format is: num [1:75, 1:2] 0.00874 0.00541 0.00333 0 0.00291 ...

Source

This is simulated data.

Examples

```
data("maf.afc")
```

`phenotype.afc` *AFC Phenotype Example*

Description

An example phenotype file that corresponds to the the one derived in the unrun example's first iteration.

Usage

```
data("phenotype.afc")
```

Format

The format is: int [1:1800, 1] 1 1 1 0 0 1 1 1 1 1 ...

Source

This is simulated data.

Examples

```
data("phenotype.afc")
```

`Wcorrected` *Corrected Chi Squared Test*

Description

Corrected Chi Squared Test, Wcorrected(), for multiple rare variant association using the difference of the sum of minor allele frequencies between cases and controls. This test handles related individuals, unrelated individuals, or both. Note: this is referred to as X (Chi) Squared Corrected in the reference rather than W Corrected.

Usage

```
Wcorrected(MAF, Pheno, Kin, Correlation, Weights)
```

Arguments

`MAF`	matrix (#Snps * 2): First column contains Minor Allele Frequency (MAF) in cases; Second column contains MAF in controls.
`Pheno`	matrix (#subjects * 1): this one-column matrix contains 0's amd 1's: 1 for cases and 0 for controls. No missing values are allowed.
`Kin`	The kinship matrix (#subjects * #subjects): the subjects must be ordered as the Pheno variable.
`Correlation`	Correlation matrix between SNPs (#Snps * #Snps). The user should calculate this matrix beforehand. Either based on own genotype data (in cases, controls, or both) or based on public databases (e.g., 1000 Genomes Projects, ESP, etc.). NA values are not allowed. They have to be replaced by zeros.
`Weights`	The weights values that can be used to up-weight or down-weight SNPs. This size of this vector is the number of Snps by 1. By default, the weights are 1 for all Snps.

Value

A vector with the following values: the sum of MAF for cases, the sum of MAF for controls, the sum of MAF for all weighted by the phenotype, the numerator of the test, the denominator of the test, the Wcorrected value (the main value calculated by the test), and the P-value.

References

Saad M and Wijsman EM, Association score testing for rare variants and binary traits in family data with shared controls, Briefings in Bioinformatics, 2017. Schaid DJ , McDonnell SK , Sinnwell JP , et al. Multiple genetic variant association testing by collapsing and kernel methods with pedigree or population structured data. Genet Epidemiol 2013 ;37 :409 –18. Choi Y , Wijsman EM , Weir BS. Case-control association testing in the presence of unknown relationships. Genet Epidemiol 2009 ;33 :668 –78.

See Also

AssocAFC Wqls afcSKAT

Examples

```
P_Wcorrected <- vector("numeric")
#This data corresponds to what is used in the 1st iteration with the raw data
data("maf.afc")
data("phenotype.afc")
data("kin.afc")
data("cor.afc")
data("weights.afc")
CORREC <- Wcorrected(MAF = maf.afc , Pheno = phenotype.afc, Kin = kin.afc , Correlation=cor.afc,
                                                                    Weights = weights.afc)
P_Wcorrected <- c(P_Wcorrected, CORREC[7])
print(P_Wcorrected)


## Not run:
#This example shows processing the raw data and uses kinship2,
#which AFC does not depend on

library(kinship2)
library(CompQuadForm)

P_Wcorrected <- vector("numeric")

for (j in 1:10)
{
  geno.afc <- read.table(system.file("extdata", "Additive_Genotyped_Truncated.txt",
                          package = "AFC"), header = TRUE)
  geno.afc[ , "IID"] <- paste(geno.afc[ , "FID"]  , geno.afc[ , "IID"]  ,sep=".")
  geno.afc[geno.afc[,"FA"]!=0 , "FA"] <- paste(geno.afc[geno.afc[,"FA"]!=0 , "FID"],
                                      geno.afc[geno.afc[,"FA"]!=0 , "FA"]  ,sep=".")
  geno.afc[geno.afc[,"FA"]!=0 , "MO"] <- paste(geno.afc[geno.afc[,"FA"]!=0 , "FID"],
                                      geno.afc[geno.afc[,"FA"]!=0 , "MO"]  ,sep=".")
  Kinship <- makekinship(geno.afc$FID , geno.afc$IID , geno.afc$FA, geno.afc$MO)
  kin.afc <- as.matrix(Kinship)
  pheno.afc <- read.table(system.file("extdata", "Phenotype", package = "AFC"))
  phenotype.afc <- matrix(pheno.afc[,j],nc=1,nr=nrow(pheno.afc))
  geno.afc <- geno.afc[,7:ncol(geno.afc)]
  Na <- nrow(pheno.afc[pheno.afc[,j]==1,])
  Nu <- nrow(pheno.afc[pheno.afc[,j]==0,])
  N <- Nu + Na
  maf.afc <- matrix(NA , nr=ncol(geno.afc) , nc=2)
  maf.afc[,1] <- colMeans(geno.afc[phenotype.afc==1,])/2;
  maf.afc[,2] <- colMeans(geno.afc[phenotype.afc==0,])/2;
  P  <- (maf.afc[,1]*Na + maf.afc[,2]*Nu)/N
  Set <- which(P<0.05)
  maf.afc <- maf.afc[c(Set),]
  cor.afc <- cor(geno.afc[,c(Set)])
  cor.afc[is.na(cor.afc)] <- 0
```

```
  weights.afc <- matrix(1/(maf.afc[,2]+1),nc=1,nr=length(Set))
 CORREC <- Wcorrected(MAF = maf.afc , Pheno = phenotype.afc, Kin = kin.afc , Correlation=cor.afc,
                                                                    Weights = weights.afc)
  P_Wcorrected <- c(P_Wcorrected, CORREC[7])
}
print(P_Wcorrected)

## End(Not run)

## The function is currently defined as
function(MAF, Pheno, Kin, Correlation, Weights)
{
  Na     <- length(Pheno[Pheno[, 1] == 1,])
  Nu     <- length(Pheno[Pheno[, 1] == 0,])
  N      <- Na + Nu

  # The three following lines: prepare the phenotype variables
  OneN  <- matrix(1, ncol = 1, nrow = N)
  Y  <- Pheno
  OneHat <- matrix(Na / N, ncol = 1 , nrow = N)

  # Estimate MAF in all subjects
  P  <- (MAF[, 1] * Na + MAF[, 2] * Nu) / N
  if (is.null(Weights))
  {
    # Variance of SNPs (2p(1-p))
    VarSnps <- sqrt(P * (1 - P))
  } else
  {
    # Variance of SNPs (2p(1-p)) accounting for the prespecified Snp weights
    VarSnps <- Weights * sqrt(P * (1 - P))
  }
  VarSnps <- matrix(VarSnps, ncol = 1)
  # This value will account for the correlation between Snps.
  cs <- 2 * t(VarSnps) %*% Correlation %*% VarSnps

  if (is.null(Weights))

  {
    # Numerator of the Xcorrec test statistic
    num <- 4 * (sum (Na * MAF[, 1] - Na * P)) ^ 2
  } else{
    # Numerator of the Xcorrec test statistic
    num <- 4 * (sum (Na * Weights * MAF[, 1] - Na * Weights * P)) ^ 2
  }
  # Denominator of the Xcorrec test statistic
  denom <- 2 * as.numeric(cs) * t(Y - OneHat) %*% Kin %*% (Y - OneHat)
  # Xcorrec test statistic
  W <- num / denom
  # Pvalue from a chi-square proba distribution
  Pvalue <- 1 - pchisq(W, 1)
  out <- t(data.frame(c(sum(MAF[,1]), sum(MAF[,2]), sum(P), num, denom, W, Pvalue)))
```

```
 colnames(out) <- c("Sum MAF Cases", "Sum MAF Controls", "Sum MAF All Weighted", "Numerator",
                    "Denominator", "Wcorrected", "Pvalue")
 rownames(out) <- "Statistics"
 return(out)
}
```

`weights.afc` *AFC Weights Example*

Description

An example weights file that corresponds to the the one derived in the unrun example's first iteration.

Usage

```
data("weights.afc")
```

Format

The format is: num [1:75, 1] 0.996 0.995 0.995 0.999 0.998 ...

Source

This is simulated data.

Examples

```
data("weights.afc")
```

`Wqls` *W Quasi-Likelihood Score*

Description

The W Quasi-Likelihood Score, Wqls(), is a score for multiple rare variant association tests using the difference of the sum of minor allele frequencies between cases and controls. This test handles related individuals, unrelated individuals, or both. This test is expected to be similar to a linear mixed model.

Usage

```
Wqls(Genotypes, MAF, Pheno, Kin, Correlation, Weights)
```

Arguments

`Genotypes`	matrix(#Subjects * #Snps). The genotypes are coded as 0, 1, or 2 copies of the minor allele.
`MAF`	matrix (#Snps * 2): First column contains Minor Allele Frequency (MAF) in cases; Second column contains MAF in controls.
`Pheno`	matrix (#subjects * 1): this one-column matrix contains 0's amd 1's: 1 for cases and 0 for controls. No missing values are allowed.
`Kin`	The kinship matrix (#subjects * #subjects): the subjects must be ordered as the Pheno variable.
`Correlation`	Correlation matrix between SNPs (#Snps * #Snps). The user should calculate this matrix beforehand. Either based on own genotype data (in cases, controls, or both) or based on public databases (e.g., 1000 Genomes Projects, ESP, etc.). NA values are not allowed. They have to be replaced by zeros.
`Weights`	The weights values that can be used to up-weight or down-weight SNPs. This size of this vector is the number of Snps by 1. By default, the weights are 1 for all Snps.

Value

A vector with the following values: the sum of MAF for cases, the sum of MAF for controls, the sum of MAF for all weighted by the phenotype, the numerator of the test, the denominator of the test, the Wqls value (the main value calculated by the test), and the P-value.

References

Saad M and Wijsman EM, Association score testing for rare variants and binary traits in family data with shared controls, Briefings in Bioinformatics, 2017. Bourgain C , Hoffjan S , Nicolae R , et al. Novel case-control test in a founder population identifies P-selectin as an atopy-susceptibility locus . Am J Hum Genet 2003 ;73 :612 –26. Thornton T , McPeek MS. Case-control association testing with related individuals: a more powerful quasi-likelihood score test. Am J Hum Genet 2007 ;81 :321 –37.

See Also

`AssocAFC` `Wcorrected` `afcSKAT`

Examples

```
P_WQLS <- vector("numeric")
#This data corresponds to what is used in the 1st iteration with the raw data
data("geno.afc")
data("maf.afc")
data("phenotype.afc")
data("kin.afc")
data("cor.afc")
data("weights.afc")
QLS <- Wqls(Genotypes = geno.afc, MAF = maf.afc, Pheno = phenotype.afc, Kin = kin.afc,
                                        Correlation = cor.afc, Weights = weights.afc)
```

```
P_WQLS <- c(P_WQLS, QLS[7])
print(P_WQLS)

## Not run:
#This example shows processing the raw data and uses kinship2,
#which AFC does not depend on

library(kinship2)
library(CompQuadForm)

P_WQLS <- vector("numeric")

for (j in 1:10)
{
  geno.afc <- read.table(system.file("extdata", "Additive_Genotyped_Truncated.txt",
                         package = "AFC"), header = TRUE)
  geno.afc[ , "IID"] <- paste(geno.afc[ , "FID"]  , geno.afc[ , "IID"]  ,sep=".")
  geno.afc[geno.afc[,"FA"]!=0 , "FA"] <- paste(geno.afc[geno.afc[,"FA"]!=0 , "FID"],
                                       geno.afc[geno.afc[,"FA"]!=0 , "FA"]  ,sep=".")
  geno.afc[geno.afc[,"FA"]!=0 , "MO"] <- paste(geno.afc[geno.afc[,"FA"]!=0 , "FID"],
                                       geno.afc[geno.afc[,"FA"]!=0 , "MO"]  ,sep=".")
  Kinship <- makekinship(geno.afc$FID , geno.afc$IID , geno.afc$FA, geno.afc$MO)
  kin.afc <- as.matrix(Kinship)
  pheno.afc <- read.table(system.file("extdata", "Phenotype", package = "AFC"))
  phenotype.afc <- matrix(pheno.afc[,j],nc=1,nr=nrow(pheno.afc))
  geno.afc <- geno.afc[,7:ncol(geno.afc)]
  Na <- nrow(pheno.afc[pheno.afc[,j]==1,])
  Nu <- nrow(pheno.afc[pheno.afc[,j]==0,])
  N <- Nu + Na
  maf.afc <- matrix(NA , nr=ncol(geno.afc) , nc=2)
  maf.afc[,1] <- colMeans(geno.afc[phenotype.afc==1,])/2;
  maf.afc[,2] <- colMeans(geno.afc[phenotype.afc==0,])/2;
  P  <- (maf.afc[,1]*Na + maf.afc[,2]*Nu)/N
  Set <- which(P<0.05)
  maf.afc <- maf.afc[c(Set),]
  cor.afc <- cor(geno.afc[,c(Set)])
  cor.afc[is.na(cor.afc)] <- 0
  geno.afc <- as.matrix(geno.afc[,Set])
  weights.afc <- matrix(1/(maf.afc[,2]+1),nc=1,nr=length(Set))
  QLS <- Wqls(Genotypes = geno.afc, MAF = maf.afc, Pheno = phenotype.afc, Kin = kin.afc,
                                         Correlation = cor.afc, Weights = weights.afc)
  P_WQLS <- c(P_WQLS, QLS[7])
}
print(P_WQLS)

## End(Not run)

## The function is currently defined as
function(Genotypes, MAF, Pheno, Kin, Correlation, Weights)
{
  Na     <- length(Pheno[Pheno[,1]==1,])
  Nu     <- length(Pheno[Pheno[,1]==0,])
  N      <- Na + Nu
```

```
# The three following lines: prepare the Phenotype variables
OneN  <- matrix(1, ncol=1, nrow = N)
Y  <- Pheno
OneHat <- matrix( Na/N, ncol=1 , nrow=N)
temp <- Y - OneHat

# Estimate MAF in all subjects
P  <- (MAF[,1]*Na + MAF[,2]*Nu)/N
# Number of Snps
Nsnps   <- nrow(MAF)
if (is.null(Weights))
{
  # Variance of SNPs (2p(1-p))
  VarSnps <- sqrt(P*(1-P))
} else
{
  # Variance of SNPs (2p(1-p)) accounting for the prespecified Snp weights
  VarSnps <- Weights*sqrt(P*(1-P))
}
VarSnps <- matrix(VarSnps,ncol=1)
# This value will account for the correlation between Snps.
cs      <- 2*t(VarSnps) %*% Correlation %*% VarSnps

if (Nsnps==1)
{
  S<-Genotypes
}else
{
  if (is.null(Weights))
  {
    # Rare variant score: sum of minor alleles accross all Snps.
    S <- apply(Genotypes , 1 , sum)
  }else
  {
    # Rare variant score: sum of minor alleles accross all Snps.
    S <- Genotypes %*% Weights
  }
}
S <- matrix(S,ncol=1,nrow=length(S))
Kin <- 2*Kin
KinInv <- solve(Kin)
A <- as.numeric(t(Y) %*% KinInv %*% OneN %*% solve(t(OneN) %*% KinInv %*% OneN))
V <- KinInv %*% Y - A * KinInv %*% OneN
if (is.null(Weights))
{
  num <- (sum((S[Y==1]) * rowSums(KinInv[Y==1,Y==1]))*(1-A) -2*sum(MAF[,2])*(A)*Nu)^2;
}else
{
 num <- (sum((S[Y==1]) * rowSums(KinInv[Y==1,Y==1]))*(1-A) -2*sum(Weights*MAF[,2])*(A)*Nu)^2;
}
denom <- as.numeric(cs) * (t(V)%*% Kin %*% V) ;
W <- num/denom ;
```

```
  Pvalue <- 1-pchisq(W,1) ;
  out <- t(data.frame(c(sum(MAF[,1]), sum(MAF[,2]), sum(P), num, denom, W, Pvalue)))
 colnames(out) <- c("Sum MAF Cases", "Sum MAF Controls", "Sum MAF All Weighted", "Numerator",
                    "Denominator", "Wqls", "Pvalue")
  rownames(out) <- "Statistics"
  return(out)
}
```

Index