```
################################################
# Shaun Purcell, PhD
#
# Human Heredity 2011.
# Modifiers and subtype-specific analyses in whole-genome association
studies: a likelihood framework
#
# Last Updated: 2011-Mar-04
################################################


################################################
# NCP given C/C frequencies and sample size
################################################

cc_ncp <- function( case_A, control_A,  n_case, n_control )
{

      # make into case-control units
      n_case_A <- 2 * n_case * case_A
      n_case_a <- 2 * n_case * (1 - case_A)
      n_control_A <- 2 * n_control * control_A
      n_control_a <- 2 * n_control * (1- control_A)

      sum_A <- n_case_A + n_control_A
      sum_a <- n_case_a + n_control_a
      sum <- sum_A + sum_a

      sum_case <- n_case_A + n_case_a
      sum_control <- n_control_A + n_control_a

      exp_case_A <- (sum_A * sum_case) / sum
      exp_control_A <- (sum_A * sum_control) / sum
      exp_case_a <- (sum_a * sum_case) / sum
      exp_control_a <- (sum_a * sum_control) / sum

      ncp <- (n_case_A - exp_case_A)^2 / exp_case_A +
            (n_case_a - exp_case_a)^2 / exp_case_a +
            (n_control_A - exp_control_A)^2 / exp_control_A +
            (n_control_a - exp_control_a)^2 / exp_control_a

      ncp
}


################################################
# Power given NCP and alpha
################################################

power <- function( ncp, alpha )
{
  1-pchisq( qchisq(1-alpha,df=1) , df=1, ncp )
}


################################################
# Main function
################################################

main <- function(nCase, nControl, s, a, b, alpha)
```

```
{

################################################
# Models
################################################

# D     = { d, d* }
# d*    = affected w/ subtype of interest (e.g. psychosis)
# d     = affected w/out subtype
# U     = unaffected

# s     = subtype frequency = P(d*|D)
# a     = baseline = P(A|U)
# b     = effect = P(A|D) - P(A|U)

# alpha = type I error rate


################################################
# Models
################################################

#                          U,   d,        d*

# m0    null               a    a         a

# m1    basic              a    a+b       a+b
# m2    subset             a    a         a+b
# m3    inverse-subset     a    a+b         a
# m4    modifier           a    a-b(1-s)  a+bs,
# m5  gradient       a    a+b/2       a+b


# (i.e. the modifier model specifies that the risk
# allele in no way increases risk for disease; it is
# a pure modifier


################################################
# Tests, for each model
################################################

# Test takes four allele frequencies:
#   { U, D, d, d* }

p1 <- tests(a, a+b                          , a+b    , a+b       , nCase,
nControl, s , alpha )
p2 <- tests(a, a*(1-s)+(a+b)*s              , a      , a+b       , nCase,
nControl, s , alpha )
p3 <- tests(a, (a+b)*(1-s)+a*s              , a+b    , a         , nCase,
nControl, s , alpha )
p4 <- tests(a, (a-b*s)*(1-s)+(a+b*(1-s))*s , a-b*s  , a+b*(1-s) , nCase,
nControl, s , alpha )
p5 <- tests(a, (a+b/2)*(1-s)+(a+b)*s        , a+b/2  , a+b       , nCase,
nControl, s , alpha )

############################
# Record allele freqs

res <- c( a, a+b , a+b  ,a, a  , a+b ,a, a+b , a ,a, a-b*s  , a+b*(1-s)
```

```
, a, a+b/2  , a+b  )

############################
# Power from standard tests

res <- c( res, p1,p2,p3,p4,p5 )

################################################
# Fit LL models to expected values (sanity check)

res <- c( res, fitLLmodels(a, a+b, a+b, nCase, nControl, s, alpha=alpha)
,
             fitLLmodels(a,a,a+b, nCase, nControl, s, alpha=alpha) ,
             fitLLmodels(a, a+b, a, nCase, nControl, s, alpha=alpha) ,
             fitLLmodels(a, a-b*s , a+b*(1-s) , nCase, nControl, s,
alpha=alpha) ,
             fitLLmodels(a, a+b/2 , a+b, nCase, nControl, s,
alpha=alpha) )


################################################
# Fit LL models to simulated datasets

R <- 1
res <- c( res , simulateData(a, a+b, a+b, nCase, nControl, s, R, alpha )
)
res <- c( res , simulateData(a, a,   a+b, nCase, nControl, s, R, alpha )
)
res <- c( res , simulateData(a, a+b, a, nCase, nControl, s, R, alpha ) )
res <- c( res , simulateData(a, a-b*s , a+b*(1-s) , nCase, nControl, s,
R, alpha ) )
res <- c( res , simulateData(a, a+b/2 , a+b, nCase, nControl, s, R,
alpha ) )

res

}



################################################
# Apply the basic tests
################################################

tests <- function(fu,fd,fn,fs,nCase,nControl,s,alpha)
{

  r <- numeric(3)

# Test 1: U versus d, d*
#  ( basic case/control )

  r[1] <- power(cc_ncp( fu, fd, nControl, nCase ),alpha)

# Test 2: U versus d*
#   (subset test)

  r[2] <- power(cc_ncp( fu, fs, nControl, nCase * s ),alpha)

# Test 3: d versus d*
```

```
#   (case-only test)

  r[3] <- power(cc_ncp( fn, fs, nCase*(1-s), nCase * s ),alpha)

# Return vector of power for these three tests
  r
}

################################################
# Simulate data
################################################

simulateData <- function(fu, fn, fs, nCase, nControl, s, R, alpha )
{
 res <- numeric(0)

 for (r in 1:R)
 {
  # draw frequencies from sample
  sfu <- mean( rbinom( 2 * nControl      , 1 , fu ) )
  sfn <- mean( rbinom( 2 * nCase * (1-s) , 1 , fn ) )
  sfs <- mean( rbinom( 2 * nCase * s     , 1 , fs ) )
  tmpres <- fitLLmodels(sfu, sfn, sfs, nCase, nControl, s, pval=T,
alpha=alpha)
  res <- rbind( res, tmpres )
 }


# Return: what proportion of times each model is selected
# Res has 33 cols:
# AIC model, all tests
# BIC model, all tets
# AIC model, L0 < alpha
# BIC model, L0 < alpha
# L0-L8 counts p<alpha

# Return
# 6*4 + 9 values = 33

rtmp <- c(
   length( res[ res[,1]==0 , 1 ] ) ,
   length( res[ res[,1]==1 , 1 ] ) ,
   length( res[ res[,1]==2 , 1 ] ) ,
   length( res[ res[,1]==3 , 1 ] ) ,
   length( res[ res[,1]==4 , 1 ] ) ,
   length( res[ res[,1]==5 , 1 ] ) ,

   length( res[ res[,2]==0 , 1 ] ) ,
   length( res[ res[,2]==1 , 1 ] ) ,
   length( res[ res[,2]==2 , 1 ] ) ,
   length( res[ res[,2]==3 , 1 ] ) ,
   length( res[ res[,2]==4 , 1 ] ) ,
   length( res[ res[,2]==5 , 1 ] ) ,

   length( res[ res[,3]<=alpha & res[,1]==0 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,1]==1 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,1]==2 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,1]==3 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,1]==4 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,1]==5 , 1 ] ) ,
```

```
   length( res[ res[,3]<=alpha & res[,2]==0 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,2]==1 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,2]==2 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,2]==3 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,2]==4 , 1 ] ) ,
   length( res[ res[,3]<=alpha & res[,2]==5 , 1 ] ) ,

   length( res[ res[,3]<=alpha , 1 ] ) ,
   length( res[ res[,4]<=alpha , 1 ] ) ,
   length( res[ res[,5]<=alpha , 1 ] ) ,
   length( res[ res[,6]<=alpha , 1 ] ) ,
   length( res[ res[,7]<=alpha , 1 ] ) ,
   length( res[ res[,8]<=alpha , 1 ] ) ,
   length( res[ res[,9]<=alpha , 1 ] ) ,
   length( res[ res[,10]<=alpha , 1 ] ) ,
   length( res[ res[,11]<=alpha , 1 ] ) )

rtmp
}



################################################
# Global variables that hold data
################################################

uA <- 0
uB <- 0
d1A <- 0
d1B <- 0
d2A <- 0
d2B <- 0
Gs <- 0
alpha <- 0

################################################
# Helper functions
################################################

l2p <- function(x)
{
# if(x<-50) return(0.001)
# if(x>50) return(0.999)
 y <- 1/(1+exp(-x))
}

p2l <- function(x)
{
#return(x)
log(x/(1-x))
}
library(stats4)


################################################
# Fit series of loglinear models
################################################

fitLLmodels <- function(fu, fn, fs, nCase, nControl, s, verbose = F,
```

```
pval = F, alpha = 0.05 , retFreqs = F)
{

# do not try to fit model if variant is monomorphic in any group

if ( fu*fn*fs==0 )
 return( rep(NA, 14) )


# populate actual data

uA <<- 2*nControl * fu
uB <<- 2*nControl * (1-fu)
d1A <<- 2*nCase * (1-s) * fn
d1B <<- 2*nCase * (1-s) * (1-fn)
d2A <<- 2*nCase * s * fs
d2B <<- 2*nCase * s * (1-fs)

Gs <<- s

# FIX #Gs <<- 0.9


# 1 params
# m0 a,a,a

# 2 params
# m1 a,b,b
# m2 a,a,b
# m3 a,b,a
# m4 a,b,c w/ constraint a=mean(b,c)

# 3 params
# m5 a,b,c

m0 <- mle(llm0)
m1 <- mle(llm1)
m2 <- mle(llm2)
m3 <- mle(llm3)
m4 <- mle(llm4)
m5 <- mle(llm5)

# Get allele frequencies

f0 <- as.numeric( c( l2p(coef(m0))   , l2p(coef(m0))   , l2p(coef(m0))
) )
f1 <- as.numeric( c( l2p(coef(m1)[1]), l2p(coef(m1)[2]), l2p(coef(m1)
[2]) ) )
f2 <- as.numeric( c( l2p(coef(m2)[1]), l2p(coef(m2)[1]), l2p(coef(m2)
[2]) ) )
f3 <- as.numeric( c( l2p(coef(m3)[1]), l2p(coef(m3)[2]), l2p(coef(m3)
[1]) ) )
tm1 <- l2p(coef(m4)[1])
tm2 <- l2p(coef(m4)[2])
tm <- tm1 * (1-s) + tm2 * s
f4 <- as.numeric(c( tm, tm1, tm2 ) )
f5 <- as.numeric( c( l2p(coef(m5)[1]), l2p(coef(m5)[2]), l2p(coef(m5)
[3]) ) )
```

```
# Get log-likelihoods

tll <- c( logLik(m0) ,
        logLik(m1) ,
        logLik(m2) ,
        logLik(m3) ,
        logLik(m4) ,
        logLik(m5) )


# Calculate AIC

aic <- c( AIC(logLik(m0)) ,
        AIC(logLik(m1)) ,
        AIC(logLik(m2)) ,
        AIC(logLik(m3)) ,
        AIC(logLik(m4)) ,
        AIC(logLik(m5)) )

nobs <- 2 * ( nCase + nControl )

bic <- c( AIC(logLik(m0),k=log(nobs)) ,
        AIC(logLik(m1),k=log(nobs)) ,
        AIC(logLik(m2),k=log(nobs)) ,
        AIC(logLik(m3),k=log(nobs)) ,
        AIC(logLik(m4),k=log(nobs)) ,
        AIC(logLik(m5),k=log(nobs)) )

# Return p-values, not power

if ( ! pval )
{

}

# Verbose display
if ( verbose )
{
cat("\nFrequency estimates\n")
print( rbind( f0,f1,f2,f3,f4,f5 ) )
cat("\nLRT\n")
print(tll)
cat("\nAIC\n")
print(aic)
cat("\nBIC\n")
print(bic)
cat("Best models (AIC, BIC) ",which.min(aic)-1,which.min(bic)-1,"\n")
}


####################################
# LRTs

# note -- below, abs() function replaces minus in -2


# comparisons with general model

 chi0 <- 2 * as.numeric( abs(logLik(m0) - logLik(m5)) )
 chi1 <- 2 * as.numeric( abs(logLik(m1) - logLik(m5)) )
```

```
 chi2 <- 2 * as.numeric( abs(logLik(m2) - logLik(m5)) )
 chi3 <- 2 * as.numeric( abs(logLik(m3) - logLik(m5)) )
 chi4 <- 2 * as.numeric( abs(logLik(m4) - logLik(m5)) )

# Standard case/control test

 chi5 <- 2 * as.numeric( abs(logLik(m0) - logLik(m1)) )

# modifier test

 chi6 <- 2 * as.numeric( abs(logLik(m0) - logLik(m4)) )

# subtype 'n', 's' test

 chi7 <- 2 * as.numeric( abs(logLik(m0) - logLik(m2)) )
 chi8 <- 2 * as.numeric( abs(logLik(m0) - logLik(m3)) )


pret <- numeric(0)

if ( pval )
{
pret <- c( 1-pchisq(chi0,2) ,
           1-pchisq(chi1,1) ,
         1-pchisq(chi2,1) ,
         1-pchisq(chi3,1) ,
           1-pchisq(chi4,1) ,
           1-pchisq(chi5,1) ,
           1-pchisq(chi6,1) ,
           1-pchisq(chi7,1) ,
           1-pchisq(chi8,1) )
}
else
{
pret <- c(
1-pchisq( qchisq(1-alpha,df=2) , ncp = chi0 , df=2) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi1 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi2 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi3 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi4 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi5 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi6 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi7 , df=1) ,
1-pchisq( qchisq(1-alpha,df=1) , ncp = chi8 , df=1) )
}

# return vector of main results, and keep track of allele freqs
pfreq <- numeric(0)

if ( retFreqs )
 {
   pfreq <- c(fu, fn, fs)
 }

 c( pfreq,
    which.min(aic)-1 ,
    which.min(bic)-1 ,
    pret )
}
```

```
###################################
# Assume counts are known globally

# 1 params
# m0 a,a,a

llm0 <- function( a = 0 )
{
p <- l2p(a)
-sum( uA    * log( p ) ,
      uB    * log( 1-p ) ,
      d1A   * log( p ) ,
      d1B   * log( 1-p ) ,
      d2A   * log( p ) ,
      d2B   * log( 1-p ) )
}



# 2 params
# m1 a,b,b

llm1 <- function( a = 0 , b = 0 )
{
p <- l2p(a)
q <- l2p(b)
-sum( uA    * log( p ) ,
      uB    * log( 1-p ) ,
      d1A   * log( q ) ,
      d1B   * log( 1-q ) ,
      d2A   * log( q ) ,
      d2B   * log( 1-q ) )
}

# m2 a,a,b

llm2 <- function( a = 0 , b = 0 )
{
p <- l2p(a)
q <- l2p(b)
-sum( uA    * log( p ) ,
      uB    * log( 1-p ) ,
      d1A   * log( p ) ,
      d1B   * log( 1-p ) ,
      d2A   * log( q ) ,
      d2B   * log( 1-q ) )
}


# m3 a,b,a

llm3 <- function( a = 0 , b = 0 )
{
p <- l2p(a)
q <- l2p(b)
-sum( uA    * log( p ) ,
      uB    * log( 1-p ) ,
      d1A   * log( q ) ,
```

```
      d1B   * log( 1-q ) ,
      d2A   * log( p ) ,
      d2B   * log( 1-p ) )
}

# m4 a,b,c w/ constraint a=mean(b,c)

llm4 <- function( a = 0 , b = 0 )
{
p <- l2p(a)
q <- l2p(b)
r <- p * (1-Gs) + q * Gs

-sum( uA    * log( r ) ,
      uB    * log( 1-r ) ,
      d1A   * log( p ) ,
      d1B   * log( 1-p ) ,
      d2A   * log( q ) ,
      d2B   * log( 1-q ) )
}


# 3 params
# m5 a,b,c

llm5 <- function( a = 0 , b = 0 , c = 0)
{
p <- l2p(a)
q <- l2p(b)
r <- l2p(c)

-sum( uA    * log( p ) ,
      uB    * log( 1-p ) ,
      d1A   * log( q ) ,
      d1B   * log( 1-q ) ,
      d2A   * log( r ) ,
      d2B   * log( 1-r ) )
}


runall <- function()
{

# Fix at 1000 cases, 1000 controls

#1 Analytic power calculations: basic tests
#2 Analytic power calculations: loglinear models
#3 Simulations (to assess 'confusability')

res <- numeric(0)

b <- 0.15
x <- 0.05

# uncorrected alpha = 1e-3
#    2-test         = 5e-4
#    3-test         = 3.33e-4

#for (b in c(0.15,0.5,0.85))
#for (x in c(0.05,0.1))
```

```
for (s in c(0.1,0.2,0.3,0.4,0.5,0.6,0.7,0.8,0.9))
{
cat("model",b,x,s,"\n")
res <- rbind( res, c( s, b, x , main(nCase=1000,nControl=
1000,s,b,x,alpha=1e-3) ) )
}

res <- as.data.frame(matrix(res,ncol=3+15+15+(5*11)+ (5*33) ))

names(res) <- c("s","b","x",
                "m1fu","m1fn","m1fs",
                "m2fu","m2fn","m2fs",
                "m3fu","m3fn","m3fs",
                "m4fu","m4fn","m4fs",
                "m5fu","m5fn","m5fs",
                "m1t1","m1t2","m1t3",
            "m2t1","m2t2","m2t3",
            "m3t1","m3t2","m3t3",
            "m4t1","m4t2","m4t3",
            "m5t1","m5t2","m5t3",


      "m1AIC","m1BIC","m1L0","m1L1","m1L2","m1L3","m1L4","m1L5","m1L6","
m1L7","m1L8",

      "m2AIC","m2BIC","m2L0","m2L1","m2L2","m2L3","m2L4","m2L5","m2L6","
m2L7","m2L8",

      "m3AIC","m3BIC","m3L0","m3L1","m3L2","m3L3","m3L4","m3L5","m3L6","
m3L7","m3L8",

      "m4AIC","m4BIC","m4L0","m4L1","m4L2","m4L3","m4L4","m4L5","m4L6","
m4L7","m4L8",

      "m5AIC","m5BIC","m5L0","m5L1","m5L2","m5L3","m5L4","m5L5","m5L6","
m5L7","m5L8",


      "sm1AICm0","sm1AICm1","sm1AICm2","sm1AICm3","sm1AICm4","sm1AICm5",

      "sm1BICm0","sm1BICm1","sm1BICm2","sm1BICm3","sm1BICm4","sm1BICm5",

      "sm1AIC2m0","sm1AIC2m1","sm1AIC2m2","sm1AIC2m3","sm1AIC2m4","sm1AI
C2m5",

      "sm1BIC2m0","sm1BIC2m1","sm1BIC2m2","sm1BIC2m3","sm1BIC2m4","sm1BI
C2m5",

      "sm1L0","sm1L1","sm1L2","sm1L3","sm1L4","sm1L5","sm1L6","sm1L7","s
m1L8",


      "sm2AICm0","sm2AICm1","sm2AICm2","sm2AICm3","sm2AICm4","sm2AICm5",

      "sm2BICm0","sm2BICm1","sm2BICm2","sm2BICm3","sm2BICm4","sm2BICm5",

      "sm2AIC2m0","sm2AIC2m1","sm2AIC2m2","sm2AIC2m3","sm2AIC2m4","sm2AI
C2m5",

      "sm2BIC2m0","sm2BIC2m1","sm2BIC2m2","sm2BIC2m3","sm2BIC2m4","sm2BI
```

```
C2m5",

      "sm2L0","sm2L1","sm2L2","sm2L3","sm2L4","sm2L5","sm2L6","sm2L7","s
m2L8",


      "sm3AICm0","sm3AICm1","sm3AICm2","sm3AICm3","sm3AICm4","sm3AICm5",

      "sm3BICm0","sm3BICm1","sm3BICm2","sm3BICm3","sm3BICm4","sm3BICm5",

      "sm3AIC2m0","sm3AIC2m1","sm3AIC2m2","sm3AIC2m3","sm3AIC2m4","sm3AI
C2m5",

      "sm3BIC2m0","sm3BIC2m1","sm3BIC2m2","sm3BIC2m3","sm3BIC2m4","sm3BI
C2m5",

      "sm3L0","sm3L1","sm3L2","sm3L3","sm3L4","sm3L5","sm3L6","sm3L7","s
m3L8",


      "sm4AICm0","sm4AICm1","sm4AICm2","sm4AICm3","sm4AICm4","sm4AICm5",

      "sm4BICm0","sm4BICm1","sm4BICm2","sm4BICm3","sm4BICm4","sm4BICm5",

      "sm4AIC2m0","sm4AIC2m1","sm4AIC2m2","sm4AIC2m3","sm4AIC2m4","sm4AI
C2m5",

      "sm4BIC2m0","sm4BIC2m1","sm4BIC2m2","sm4BIC2m3","sm4BIC2m4","sm4BI
C2m5",

      "sm4L0","sm4L1","sm4L2","sm4L3","sm4L4","sm4L5","sm4L6","sm4L7","s
m4L8",


      "sm5AICm0","sm5AICm1","sm5AICm2","sm5AICm3","sm5AICm4","sm5AICm5",

      "sm5BICm0","sm5BICm1","sm5BICm2","sm5BICm3","sm5BICm4","sm5BICm5",

      "sm5AIC2m0","sm5AIC2m1","sm5AIC2m2","sm5AIC2m3","sm5AIC2m4","sm5AI
C2m5",

      "sm5BIC2m0","sm5BIC2m1","sm5BIC2m2","sm5BIC2m3","sm5BIC2m4","sm5BI
C2m5",

      "sm5L0","sm5L1","sm5L2","sm5L3","sm5L4","sm5L5","sm5L6","sm5L7","s
m5L8" )

res
}


##############################################
# Helper function for raw data
# Given three sets of genotypic counts,
# fit LL models and return what that returns

runData <- function(g)
{

nControl <- g[1]+g[2]+g[3]
```

```
nCase <- sum(g)-nControl
s <- (g[7]+g[8]+g[9]) / nCase
sfu <- (2*g[1]+g[2]) / (2*nControl)
sfn <- (2*g[4]+g[5]) / (2*nCase*(1-s))
sfs <- (2*g[7]+g[8]) / (2*nCase*s)

fitLLmodels(sfu, sfn, sfs, nCase, nControl, s, pval=T, alpha=1e-
4,retFreqs=T)
}


#######################################
# Main driver function for raw data

runAllData <- function(u, n, s, labels)
{
# vectors of genotype counts:
# e.g. u = l x 3 matrix;
#      l SNPs, 3 genotypes, for controls

res <- numeric(0)

# one SNP at a time, fit LL models
res <- cbind( res , apply( cbind(u,n,s) , 1 , runData ) )


res <- as.data.frame(matrix(res,byrow=T,nrow=length(u[,1])))

# add labels and header
#res <- cbind( labels, res )
#names(res) <- c
("CHR","SNP","A1","A2","FU","FN","FS","AIC","BIC","mL0","mL1","mL2","mL3
","mL4","mL5","mL6","mL7","mL8")

# return all results
res
}


rapid <- function()
{

# expect format: CHR SNP A1 A2   G11 G12 G22   (counts)for each dataset
# l <- read.table("../step-data/control.dat",header=F)[ ,1:4]
# u <- as.matrix(read.table("../step-data/control.dat",header=F)[ ,-
(1:4)])
# n <- as.matrix(read.table("../step-data/case-non-rapid.dat",header=F)
[ ,-(1:4)])
# s <- as.matrix(read.table("../step-data/case-rapid.dat",header=F)[ ,-
(1:4)])

 l <- read.table("../scratch/c.dat",header=F)[ ,1:4]
 u <- as.matrix(read.table("../scratch/c.dat",header=F)[ ,-(1:4)])
 n <- as.matrix(read.table("../scratch/nr.dat",header=F)[ ,-(1:4)])
 s <- as.matrix(read.table("../scratch/r.dat",header=F)[ ,-(1:4)])

 runAllData(u,n,s,l)
}

# k <- numeric(0)
```

```
# for(i in 1:372193)
# {
# k <- c(k, as.numeric(r[[1]][[i]]))
# }
# k <- matrix(r3,byrow=T,nrow=14)

#NA in 11 SNPs

# [1] 243091 243096 367604 367605 367608 367611 367612 367622 367638
367640
#[11] 371526


# r2 <- r[[1]]
# r3 <- r2[ which(as.numeric(lapply(r[[1]],length))==14 ) ]
# k <- matrix(unlist(r3),byrow=F,nrow=14)
# k <- t(k)
# l <- l[ which(as.numeric(lapply(r[[1]],length))==14 ) , ]
# k <- cbind( l, k )
# names(k) <- c
("CHR","SNP","A1","A2","FU","FN","FS","AIC","BIC","mL0","mL1","mL2","mL3
","mL4","mL5","mL6","mL7","mL8")

##############################################
# Notes on loglinear model LRTs

#  m*L0  2df test of anything versus null

#  m*L1  1df general versus standard
#  m*L2  1df general versus subset   (test inverse-subset; not assuming
subset==control)
#  m*L3  1df general versus inverse-subset (test subset; not assuming
inverse-subset==control)
#  m*L4  1df general versus modifier

#  m*L5  1df basic versus null
#  m*L6  1df modifier versus null
#  m*L7  1df subset versus null
#  m*L8  1df inverse-subset versus null

#L4 == L5  (basic)
#L6 == L1  (modifier)


##
# Example to run data
# r <- rapid()
# write.table(k, file="../results/rapid-v3.dat", quote=F, sep=" ",
row.names = F, col.names = T )



tabResults <- function(r,b)
{

# write
# First, just concentrate on the analytic results
# Two classes: standard tests, & LRTs

# For simulation:
```

```
#  Standard tests
#  LRT tests
#  Model selection

# Plot as a function of
# 45 conditions: 3*3*5

# Make graphs

m1 <- r[ , c(1, 19:21, 36:44 ) ]
m2 <- r[ , c(1, 22:24, 47:55 ) ]
m3 <- r[ , c(1, 25:27, 58:66 ) ]
m4 <- r[ , c(1, 28:30, 69:77 ) ]
m5 <- r[ , c(1, 31:33, 80:88 ) ]

b1 <- b[ , c(1, 19:21, 36:44 ) ]
b2 <- b[ , c(1, 22:24, 47:55 ) ]
b3 <- b[ , c(1, 25:27, 58:66 ) ]
b4 <- b[ , c(1, 28:30, 69:77 ) ]
b5 <- b[ , c(1, 31:33, 80:88 ) ]

# Insert best-fit models (b)

m1 <- cbind( m1 , b1[,5] , b1[,2] )
m2 <- cbind( m2 , b2[,5] , b2[,3] )
m3 <- cbind( m3 , b3[,5] , b3[,1] )
m4 <- cbind( m4 , b4[,5] , b4[,4] )
m5 <- cbind( m5 , b5[,5] , b5[,1] )

par(mfcol=c(1,1))

m <- m1

plot( m[,1] , m[,2] , type="b" , col="black" , xlim=range(0,1),
ylim=range(0,1) ,
      xlab="Subtype frequency (s)", ylab="Power",lwd=2 ,pch=1,cex=1)
points( m[,1] , m[,3] , type="b" , col="blue",lwd=2 ,pch=4,cex=1)
points( m[,1] , m[,4] , type="b" , col="red",lwd=2 ,pch=3,cex=1)
points( m[,1] , m[,14] , type="b" , col="darkgreen",lwd=2 ,pch=8,cex=1)
points( m[,1] , m[,15] , type="b" , col="gray",lwd=2,lty=2 ,pch=1,cex=1)

dev.copy2eps(file="plot-analytic-power-m5.eps")

}
```