Code Review: h2 and rg estimation using Haseman-Elston regression

Martin Mullis

2025-04-09

Load required packages

```r
#nessary for functions to work
library(parallel)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
#used for visualization. not necessary for functions
library(scales)
```

Set working directory and load data

```r
# set working directory here. This should contain the phenotype file and the kinship file
wd <- /PATH/TO/FILES/

# load znorm phenotype data ('phenotypes_non_molecular').
#The first seven columns correspond to ID and covariates.
#The remaining columns (7 onwards) correspond to phenotypes.
load(paste0(wd,"DO_Phenotypes_non_molecular.RData"))
phenotypes <- phenotypes_non_molecular;rm(phenotypes_non_molecular) #rename for convenience

# load kinship data ('kin'). This is a normally list of 20 matrices that contain pairwise kinship...
# ...information among all mice in the data set for each of the 20 chromosomes.
# But in practice, only the first element of the list is actually used.
# So from the github, I've uploaded a list that includes only the first chromosome.
load(paste0(wd,"kinship_loco.RData"))
```

Introduction

Haseman-Elston regression is a method for estimating the heritability (h2) of traits by regressing pairwise phenotype similarity scores against kinship for one or more sets of individuals. This method can also be extended to estimate the genetic correlation (rg) of pairs of traits via a structural equation model. The following functions constitute our implementation of Haseman-Elston for use with z-score normalized phenotype data from DO mice.

Z score normalization

znorm() takes in a vector of phenotype data and zscore-normalizes it. the resulting distribution will have a mean of 0, and a standard deviation and variance of 1.
Arguments:
dist: A vector of numerical phenotypes

```
znorm <- function(dist){((dist - mean(dist))/sd(dist))}
```

Here's how it works:

```
# simulate a normally distributed trait, measured in 1000 individuals...
# ...with a mean of 1 and a standard deviation of 3
n = 1000
mean = 1
sd = 3
trait <- rnorm(n=n,mean = mean,sd=sd)
print("pre-normalization")
```

```
## [1] "pre-normalization"
```

```
print(paste0("mean = ",mean,"; sd = ",sd))
```

```
## [1] "mean = 1; sd = 3"
```

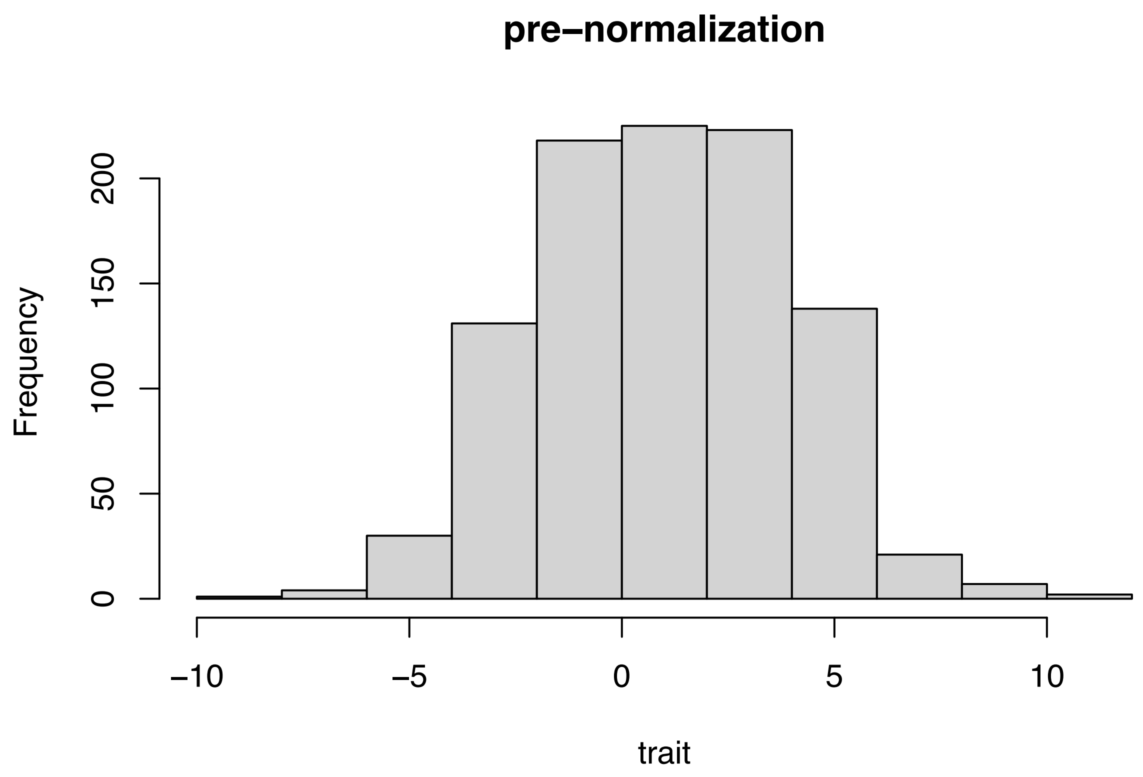
```
norm_trait <- znorm(trait)
print("post-normalizaton")
```

```
## [1] "post-normalizaton"
```

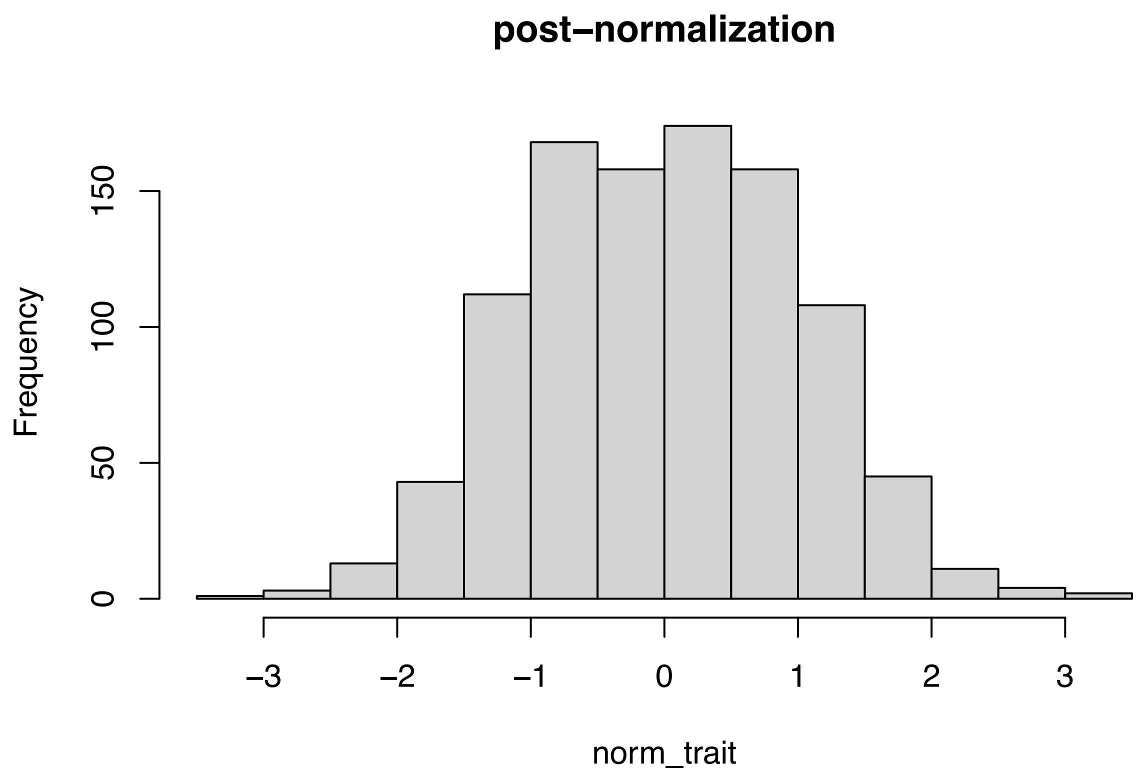
```
print(paste0("mean = ",mean(norm_trait),"; sd = ",sd(norm_trait)))
```

```
## [1] "mean = -1.01030295240889e-17; sd = 1"
```

```
hist(trait,main="pre-normalization")
```



```r
hist(norm_trait,main="post-normalization")
```



Standard error correction

The se_correct() function is used for correcting the standard error (SE) of Haseman-Elston regression heritability (h2) and genetic correlation (rg) estimates. It implements an empirically derived correction factor that increases the SE a bit as sample size and h2 increase.
Arguments: **se**, **h2**, and **n**.
se: a standard error to be corrected.
h2: the corresponding h2 estimate.
n: the number of individuals used to estimate the h2.
output: a corrected standard error estimate.

```
se_correct <- function(se,h2,n){
  factor <- sqrt(n) * (0.020225*h2 + 0.004225) + (-0.2352*h2 +0.9467)
  se_corrected <- se * factor
  return(se_corrected)
}
```

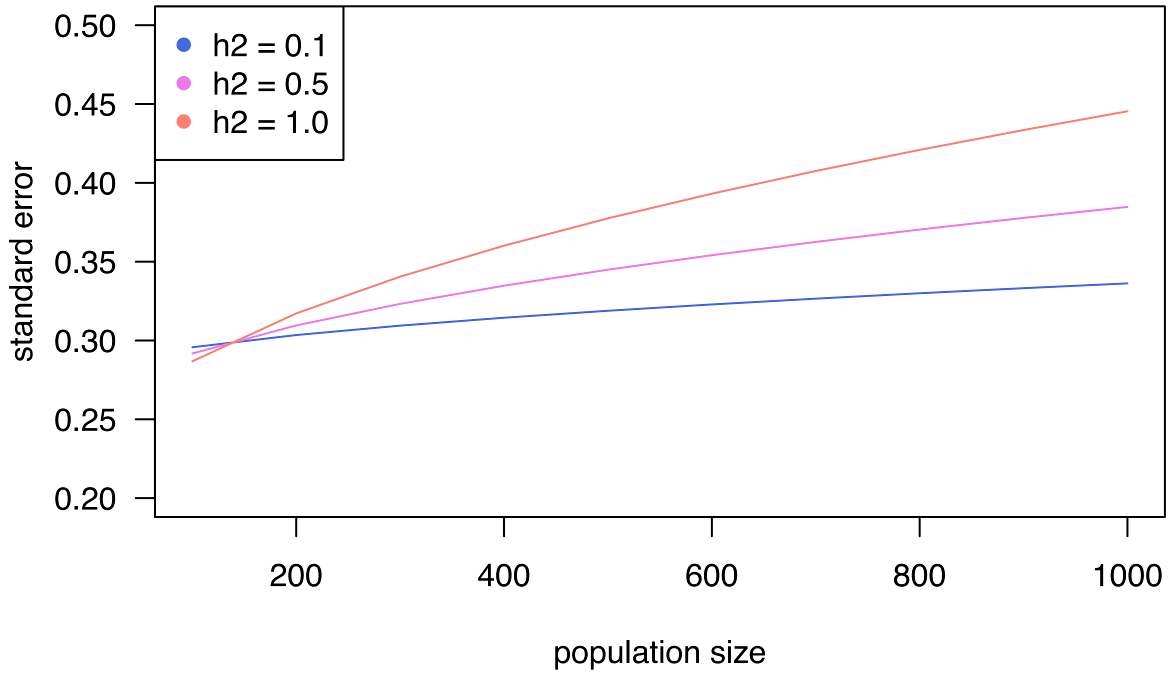
Demonstrate functionality of se_correct() by correcting a particular standard error (0.3) at different h2 levels and population sizes

```
ns <- seq(100,1000,by=100) # set population sizes
se <- 0.3 # set a constant standard error

# get a bunch of corrected standard errors at three different h2 levels as a function of population siz
```

```r
ses <- data.frame(t(sapply(1:length(ns),function(x){
  low_h2 <- se_correct(se,h2=0.1,ns[x])
  med_h2 <- se_correct(se,h2=0.5,ns[x])
  high_h2 <- se_correct(se,h2=1.0,ns[x])
  out <- data.frame("n"=ns[x],"low_h2"=low_h2,"med_h2"=med_h2,"high_h2"=high_h2)
})))
print(ses)
```

```
##       n    low_h2    med_h2   high_h2
## 1   100 0.2956965 0.2917425    0.2868
## 2   200 0.3034599 0.3095589 0.3171826
## 3   300  0.309417 0.3232298 0.3404959
## 4   400  0.314439  0.334755   0.36015
## 5   500 0.3188635 0.3449089 0.3774656
## 6   600 0.3228636 0.3540887 0.3931201
## 7   700  0.326542 0.3625304 0.4075159
## 8   800 0.3299658 0.3703877 0.4209151
## 9   900 0.3331815 0.3777675    0.4335
## 10 1000  0.336223 0.3847475 0.4454031
```

```r
plot1 <- plot(ses$n,ses$low_h2,col="royalblue",type="l",las=1,ylab="standard error",xlab="population si
lines(ses$n,ses$med_h2,col="orchid2",type="l")
lines(ses$n,ses$high_h2,col="salmon",type="l")
legend("topleft",col=c("royalblue","orchid2","salmon"),legend=c("h2 = 0.1","h2 = 0.5","h2 = 1.0"),pch=1
```



Note that as h2 and n grow, so too does the scale of the correction applied to the standard error.

Phenotype similarity scores & relatedness

The pairwise_product2() funciton takes in two sets of trait values and computes all combinations of pairwise products for the sets of values; these are used as phenotypic similarity scores in our implementation of Haseman-Elston regression. The function also takes in a kinship matrix so that it can assign pairwise kinship estimates to the similarity scores. The output is a dataframe that is ready for Haseman-Elston regression.

'NA' values should be removed from the vectors of trait values prior to running this function. This is done automatically in the HEreg() function that calls pairwise_product2().

Arguments: **t1**, **t2**, **k**, **type**
t1: Named vector of z-score normalized phenotype data. Names must correspond to Mouse ID and be present in the kinship matrix.
t2: Second named vector of z-score normalized phenotype data.
k: List of kinship matrices generated by the R/qtl2 package (Broman et al. 2019, 10.1534/genetics.118.301595).
type: Specifies the type of Haseman-Elston regression to be performed.
- `"h"` corresponds to heritability.
- `"rg"` corresponds to genetic correlation.

The output is a four-column dataframe containing the following information:
id1: ID of the first individual that contributed to the phenotypic similarity score.
id2: ID of the second individual that contributed to the phenotypic similarity score.
prod: The phenotypic similarity score (product of the z score normalized phenotypes).
relatedness: The kinship of the two individuals.

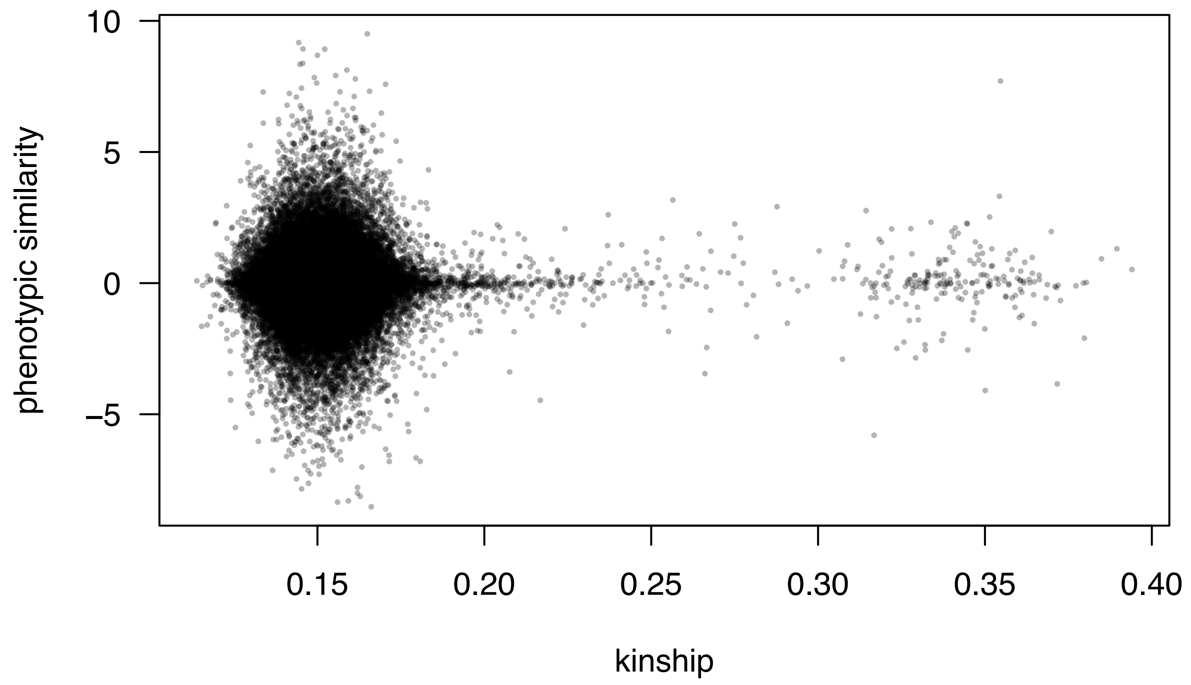
```
pairwise_product2 <- function(t1,t2,k,type=c("h","rg")){
  kin <- k[[1]] # just need the kinship matrix for one autosome

  # make list of combinations to check
  ids1 <- names(t1)
  ids2 <- names(t2)

  kin_sub <- kin[rownames(kin) %in% ids1, colnames(kin) %in% ids2] # subset kinship matrix to only incl
  kin_sub <- kin_sub[match(names(t1),rownames(kin_sub)),]
  kin_sub <- kin_sub[,match(names(t2),colnames(kin_sub))]

  combs <- expand.grid(ids1,ids2)
  temp <- outer(t1,t2,FUN="*")
  if(type == "h"){
    ind <- which(upper.tri(temp,diag=F) , arr.ind = TRUE )

    out_df <- data.frame( col = dimnames(temp)[[2]][ind[,2]] ,
                          row = dimnames(temp)[[1]][ind[,1]] ,
                          prod = temp[ ind ] )

    ind2 <- which(upper.tri(kin_sub,diag=F) , arr.ind = TRUE )


    kin_sub2 <- data.frame( col = dimnames(kin_sub)[[2]][ind[,2]] ,
                            row = dimnames(kin_sub)[[1]][ind[,1]] ,
                            relatedness = kin_sub[ ind ] )

    out_df <- cbind(out_df,kin_sub2$relatedness)
  }else{
    out_df <-cbind(combs,as.vector(temp),as.vector(kin_sub))
  }

  colnames(out_df) <- c("id1","id2","prod","relatedness")
  out_df$id1 <- as.character(out_df$id1)
  out_df$id2 <- as.character(out_df$id2)
  out_df <- out_df[out_df$id1 != out_df$id2,] # exclude rows containing self-self comparisons

  rownames(out_df) <- 1:nrow(out_df)

  #end <- Sys.time()
  #end-start
  return(out_df)
}
```

Demonstrate funcitonality of pairwise_product2() by generating combinatorial products of either one phenotype (for h2 estimation) or two phentypes (for rg estimation). Set working directory path (‘wd’ above) to the phenotype data.

```
# sample first two phenotypes
traits <- colnames(phenotypes)[7:8]

# extract trait data and remove NAs
T1 <- phenotypes[,traits[1]]
names(T1) <- phenotypes$MouseID
T1 <- na.omit(T1)

T2 <- phenotypes[,traits[2]]
names(T2) <- phenotypes$MouseID
T2 <- na.omit(T2)

# compute combinatorial pairwise products and plot out for a single trait
pairwise_prod_h2 <- pairwise_product2(t1=T1,t2=T1,k=kin,type="h")
print(head(pairwise_prod_h2))
```

```
##                 id1              id2        prod relatedness
## 1 cal_int_1025_DFI cal_int_0003_DFI  0.10395919   0.1345693
## 2 cal_int_1049_DFI cal_int_0003_DFI -0.72134147   0.1462931
## 3 cal_int_1049_DFI cal_int_1025_DFI -0.06020430   0.1325282
## 4  cal_int_104_DFI cal_int_0003_DFI  0.84431910   0.1640233
## 5  cal_int_104_DFI cal_int_1025_DFI  0.07046821   0.1408014
## 6  cal_int_104_DFI cal_int_1049_DFI -0.48895763   0.1437497
```

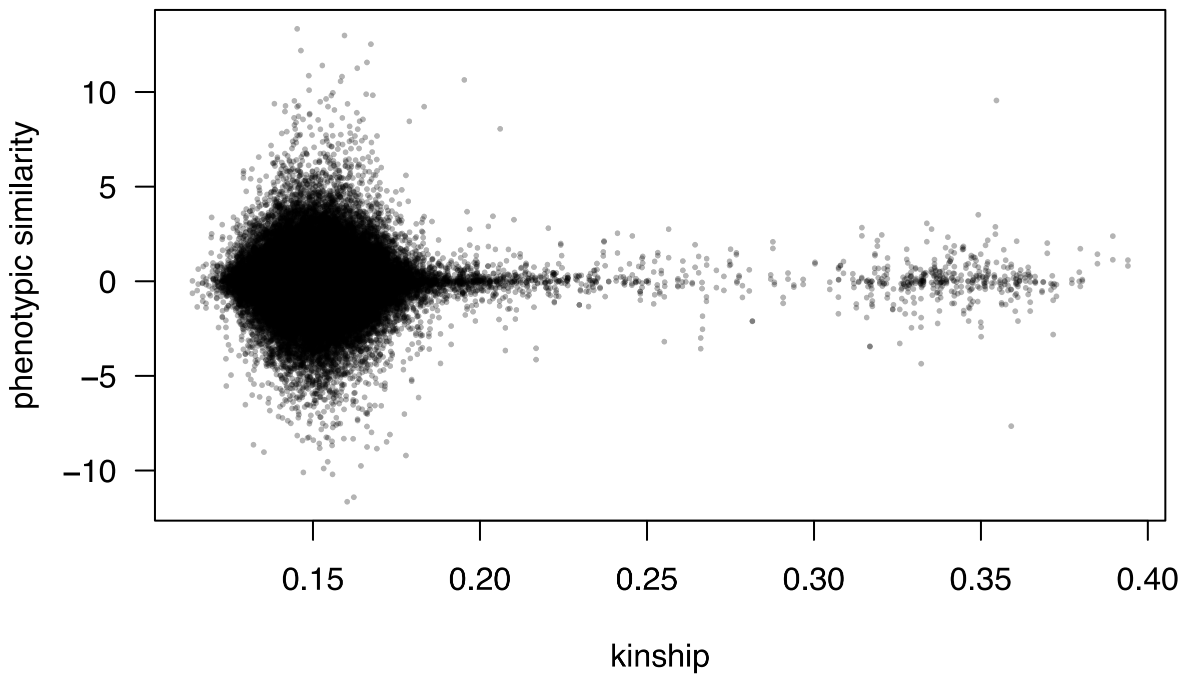
```
plot(pairwise_prod_h2$relatedness,pairwise_prod_h2$prod,col=alpha("black",0.3),pch=16,cex=0.4,las=1,xla
```



```
pairwise_prod_rg <- pairwise_product2(t1=T1,t2=T2,k=kin,type="rg")
print(head(pairwise_prod_rg))
```

```
##                 id1              id2       prod relatedness
## 1  cal_int_1025_DFI cal_int_0003_DFI  0.1775657   0.1345693
## 2  cal_int_1049_DFI cal_int_0003_DFI -1.2320745   0.1462931
## 3   cal_int_104_DFI cal_int_0003_DFI  1.4421242   0.1640233
## 4 cal_int_1061_lung cal_int_0003_DFI  0.9398945   0.1788545
## 5 cal_int_1062_lung cal_int_0003_DFI  4.1389591   0.1449356
## 6 cal_int_1063_lung cal_int_0003_DFI -2.3192318   0.1597254
```

```
plot(pairwise_prod_rg$relatedness,pairwise_prod_rg$prod,col=alpha("black",0.3),pch=16,cex=0.4,las=1,xla
```



Haseman-Elston regression on z score normalized phenotype data

HEreg() performs Haseman-Elston regression on a set of traits and outputs relevant summary statistics. It uses phenotype data that has already been z-score normalized. It calls the pairwise_product2() and se_correct() functions above.

Arguments: **pheno**, **k**, **h2_df**,**type**,**traits**,**cores**,**correct**.
pheno: A dataframe consisting of z-score normalized phenotype values. The first column should be titled "MouseID" and contain ID information for the animals that corresponds to the row and column names of the kinship matrix. Subsequent columns correspond to normalized phenotype data.
k: a list of kinship matrices of the sort generated by the R/qtl2 package (Broman et al. 2019, 10.1534/genetics.118.301595).
h2_df: only necessary if type = "gc". this is a dataframe of heritability values generated by HEreg(). These h2 values are used in the structural equation model to estimate genetic correlations for pairs of traits.
traits: an n-by-2 matrix providing combinations of traits to use in the analysis.
cores: the number of cores to use for parallelization. be mindful of memory constraints.
correct: implement SE correction?

Output: an n-by-14 dataframe.
trait1: The first trait used in the estimation of rg. Will be the same as **trait2** if type="h".
trait2: The second trait used in the estimation of rg. Will be the same as **trait1** if type="h".
gcor: The genetic correlation estimate for the pair of traits. Will be **1** if type="h" as traits are always perfectly genetically correlated with themselves. For type="gc", this may be **NA** if the h2 of one of the traits is too low (e.g. $h2 < 0$).
gcor_p: The p-value associated with the rg estimate.

pcor: The (Pearson) phenotypic correlation of the pair of traits. Will be **1** if type="h". Will be **NA** if a pair of traits were measured in a nonoverlapping set of animals.
pcor_p: The p-value associated with the pcor value.
h2: If type="h", this is the heritability of the trait. If type = "gc", this is **NA**.
se_gc: The standard error of the genetic correlation estimate.
se_h2: The standard error of the h2 estimate. Will be **NA** if type="gc". **ob**: The regression coefficient of Haseman-Elston regression, divided by 2. This is an intermediate value used initially to assess the performance of the function. Not really useful.
h2_trait1: The heritability of **trait1**.
h2_trait2: The heritability of **trait2**.
n_1: The number of individuals from **trait1** used in the regression.
n_2: The number of individuals from **trait2** used in the regression.

```r
HEreg <- function(pheno,k,h2_df,type=c("h","gc"),traits,cores,correct=T){
  names <- colnames(pheno)[2:ncol(pheno)]
  if(is.null(traits)){
    if(type == "h"){
      pairs <- t(sapply(1:length(names),function(x){rep(names[x],times=2)}))
    }else{
      herit_df <- h2_df
      pairs <- t(combn(names,2))
    }
  }else{
    pairs <- traits
    herit_df <- h2_df
  }

  cor_df <- mclapply(1:nrow(pairs),function(x){
    if(x %% 1000 == 0){print(x)}
    start <- Sys.time()
    # retreive names of traits used for HE regression
    trait1 <- pairs[x,1]
    trait2 <- pairs[x,2]

    # access trait values from phenotype dataframe
    T1 <- pheno[is.na(pheno[,trait1]) == F,trait1]
    names(T1) <- pheno[is.na(pheno[,trait1]) == F,1]
    T2 <- pheno[is.na(pheno[,trait2]) == F,trait2]
    names(T2) <- pheno[is.na(pheno[,trait2]) == F,1]

    n_1 <- length(T1)
    n_2 <- length(T2)

    # there are no non-na values, return na for the output
    if(length(T1) == 0 | length(T2) == 0){
      out <- c(trait1,trait2,NA,NA,NA,NA,NA,NA,NA,NA,NA,NA,n_1,n_2)
      return(out)
    }

    # subset to only include individuals in the kinship matrix
    T1 <- T1[which(names(T1) %in% colnames(k[[1]]))]
    T2 <- T2[which(names(T2) %in% colnames(k[[1]]))]

    # pull shared values to calculate phenotypic correlation
    shared <- intersect(names(T1),names(T2))
    if(length(shared) >= 3){
      T1_sub <- T1[`shared`]
      T2_sub <- T2[`shared`]
      pcor <- cor(T1_sub,T2_sub)
      pcor_p <- cor.test(T1_sub,T2_sub)$p.value
    }else{
      pcor <- NA
      pcor_p <- NA
    }

    # get trait product and relatedness for each pair of animals
    data <- pairwise_product2(t1=T1,t2=T2,k=k,type=type)

    fit <- lm(data$prod ~ data$relatedness)
    gcor_p <- summary(fit)$coefficients[2,4]
    b <- fit$coefficients[2]

    ob <- b/(2*var(T1)) # because all traits are z-normalized in advance, the variance of both traits i

    if(type == "h"){
      h2 <- ob
      h2_trait1 <- h2
      h2_trait2 <- h2
    }else{
      h2 <- NA
      h2_trait1 <- herit_df[herit_df$trait1 == trait1,"h2"]
      h2_trait2 <- herit_df[herit_df$trait1 == trait2,"h2"]
      se_h2_trait1 <- herit_df[herit_df$trait1 == trait1,"se_h2"]
      se_h2_trait2 <- herit_df[herit_df$trait1 == trait2,"se_h2"]
    }

    gcor <- ob/(sqrt(h2_trait1)*sqrt(h2_trait2))

    if(type == "gc"){
      b_error <- sqrt(sum(((data$prod - mean(data$prod))^2)) / (nrow(data) - 2)) / sqrt(sum(((data$rela

      ob_error <- b_error/(2*var(T1))

      if(correct==T){
        ob_error <- se_correct(n=sqrt(n_1*n_2),h2=ob,se=ob_error)
      }else{
        ob_error <- ob_error
      }


      # source: http://www.met.rdg.ac.uk/~swrhgnrj/combining_errors.pdf
      se_gc <- sqrt( (ob_error/ob)^2 + ((.5*sqrt(h2_trait1)*se_h2_trait1) / sqrt(h2_trait1))^2 + ((.5*s

      se_h2 <- NA
    }else{
      b_error <- sqrt(sum(((data$prod - mean(data$prod))^2)) / (nrow(data) - 2)) / sqrt(sum(((data$rela
      ob_error <- b_error/(2*var(T1))

      se_h2 <- ob_error

      if(correct==T){
        se_h2 <- se_correct(n=n_1,h2=h2,se=se_h2)
      }else{
        se_h2 <- se_h2
      }

      # source: http://www.met.rdg.ac.uk/~swrhgnrj/combining_errors.pdf
      # https://www.met.reading.ac.uk/~swrhgnrj/
      # https://scholar.google.com/citations?hl=en&user=-vUyGWsAAAAJ&view_op=list_works&sortby=pubdate
      se_gc <- sqrt( (ob_error/ob)^2 + ((.5*sqrt(h2)*se_h2) / sqrt(h2))^2 + ((.5*sqrt(se_h2)*se_h2) / se
    }
    out <- c(trait1,trait2,gcor,gcor_p,pcor,pcor_p,h2,se_gc,se_h2,ob,h2_trait1,h2_trait2,n_1,n_2)

    end <- Sys.time()
    #print(end-start)
    return(out)
  },mc.cores=cores)

  cor_df <- do.call('rbind',cor_df)

  colnames(cor_df) <- c("trait1","trait2","gcor","gcor_p","pcor","pcor_p","h2","se_gc","se_h2","ob","h2

  cor_df <- data.frame(cor_df)
  for(i in 3:ncol(cor_df)){
    cor_df[,i] <- as.numeric(cor_df[,i])
  }
  for(i in 3:(ncol(cor_df)-2)){
    cor_df[,i] <- round(cor_df[,i],3)
  }
  return(cor_df)
}
```

Demonstrate utility of HEreg() function by estimating heritability for a subset of phenotypes

```
#discard covariate data in the phenotypes dataframe
phenotypes <- phenotypes[,c(1,7:ncol(phenotypes))]

#specify a subset of phenotypes to test the function on
traits <- cbind(colnames(phenotypes[,2:20]),colnames(phenotypes[,2:20]))
print("Correct format for 'traits' matrix for h2 estimation:")
```

```
## [1] "Correct format for 'traits' matrix for h2 estimation:"
```

```
print(head(traits))
```

```
##      [,1]                        [,2]
## [1,] "cal_int_pheno_ang"         "cal_int_pheno_ang"
## [2,] "cal_int_pheno_dist"        "cal_int_pheno_dist"
## [3,] "cal_int_pheno_dpa"      "cal_int_pheno_dpa"
## [4,] "cal_int_pheno_gait"     "cal_int_pheno_gait"
## [5,] "cal_int_pheno_wall"     "cal_int_pheno_wall"
## [6,] "cal_int_pheno_bw.delta" "cal_int_pheno_bw.delta"
```

```
# compute h2 for the first 20 traits in our dataset
h2 <- HEreg(pheno=phenotypes,h2_df=NULL,k=kin,type="h",traits=traits,cores=2,correct=T)
print("Heritability of traits:")
```

```
## [1] "Heritability of traits:"
```

```
print(head(h2))
```

```
##                   trait1                 trait2 gcor gcor_p pcor pcor_p     h2
## 1      cal_int_pheno_ang      cal_int_pheno_ang    1  0.260    1      0  0.141
## 2     cal_int_pheno_dist     cal_int_pheno_dist    1  0.682    1      0  0.051
## 3      cal_int_pheno_dpa      cal_int_pheno_dpa    1  0.304    1      0  0.129
## 4     cal_int_pheno_gait     cal_int_pheno_gait    1  0.591    1      0  0.068
## 5     cal_int_pheno_wall     cal_int_pheno_wall  NaN  0.998    1      0  0.000
## 6 cal_int_pheno_bw.delta cal_int_pheno_bw.delta  NaN  0.608    1      0 -0.108
##   se_gc se_h2     ob h2_trait1 h2_trait2 n_1 n_2
## 1 0.892 0.132  0.141     0.141     0.141 396 396
## 2 2.445 0.130  0.051     0.051     0.051 396 396
## 3 0.978 0.132  0.129     0.129     0.129 395 395
## 4 1.867 0.131  0.068     0.068     0.068 394 394
## 5   NaN 0.129  0.000     0.000     0.000 396 396
## 6   NaN 0.211 -0.108    -0.108    -0.108 189 189
```

```
# subset our focal traits to those that are heritable. HEreg can't produce valid estimates if the HE or
traits_subset <- h2$trait1[h2$h2 > 0.1]

# specify combinations of traits to compute rg for
traits2 <- t(combn(traits,2))
print("Correct format for 'traits' matrix for rg estimation:")
```

```
## [1] "Correct format for 'traits' matrix for rg estimation:"
```

```
print(head(traits2))
```

```
##      [,1]                [,2]
## [1,] "cal_int_pheno_ang" "cal_int_pheno_dist"
## [2,] "cal_int_pheno_ang" "cal_int_pheno_dpa"
## [3,] "cal_int_pheno_ang" "cal_int_pheno_gait"
## [4,] "cal_int_pheno_ang" "cal_int_pheno_wall"
## [5,] "cal_int_pheno_ang" "cal_int_pheno_bw.delta"
## [6,] "cal_int_pheno_ang" "cal_int_pheno_circ.f"
```

```
# compute rg among the first 20 traits in our dataset
rg <- HEreg(pheno=phenotypes,h2_df=h2,k=kin,type="gc",traits=traits2,cores=2,correct=T)
print("Genetic correlations among heritable traits:")
```

```
## [1] "Genetic correlations among heritable traits:"
```

```r
head(rg)
```

```
##              trait1                 trait2   gcor gcor_p   pcor pcor_p h2 se_gc
## 1 cal_int_pheno_ang     cal_int_pheno_dist  1.169  0.264  0.783  0.000 NA 0.947
## 2 cal_int_pheno_ang      cal_int_pheno_dpa  0.398  0.546  0.191  0.000 NA 1.726
## 3 cal_int_pheno_ang     cal_int_pheno_gait  0.651  0.474  0.566  0.000 NA 1.460
## 4 cal_int_pheno_ang     cal_int_pheno_wall   -Inf  0.303  0.381  0.000 NA   Inf
## 5 cal_int_pheno_ang cal_int_pheno_bw.delta    NaN  0.409 -0.210  0.007 NA   NaN
## 6 cal_int_pheno_ang   cal_int_pheno_circ.f -0.315  0.568 -0.058  0.458 NA 1.773
##   se_h2     ob h2_trait1 h2_trait2 n_1 n_2
## 1    NA  0.099     0.141     0.051 396 396
## 2    NA  0.054     0.141     0.129 396 395
## 3    NA  0.064     0.141     0.068 396 394
## 4    NA  0.092     0.141     0.000 396 396
## 5    NA  0.102     0.141    -0.108 396 189
## 6    NA -0.071     0.141     0.356 396 189
```