

# New way of creating and standardizing phenotype data with variance partitioning

KE Lotterhos

8/18/2020

$$V_P = V_G + V_E + V_{GE} + Cov_{GE}$$

In a reciprocal transplant experiment, there are  $g$  genotypes transplanted into  $e$  environmental patches, for a total of  $g * e = n_{GE}$  genotype-environment combinations.

$g$  is levels of genotypes for  $i = 1, 2, \dots, g$   $e$  is levels of environments for  $j = 1, 2, \dots, e$

Assuming the equal sample sizes  $r$  (1, 2, ... $k$ ) within each genotype-environment combination, could partitioning the variance be as simple as:

$$V_G = re \sum_{i=1}^g (\bar{y}_i - \bar{y})^2$$

$$V_E = rg \sum_{j=1}^e (\bar{y}_j - \bar{y})^2$$

$$V_{GE} = r \sum_{i=1}^g \sum_{j=1}^e (\bar{y}_{ij} - \bar{y}_i - \bar{y}_j + \bar{y})^2$$

$$V_{Cov_{GE}} = (g-1)er \sum_{i=1}^g \sum_{j=1}^e (\bar{y}_i - \bar{y})(\bar{y}_j - \bar{y})I$$

- $I$  is an indicator variable that is 1 when the genotype  $i$  originated from environment  $j$  and 0 otherwise
- basically, each summation needs to count the same number of times to be comparable as a SS on the same scale. If we have a 2x2 reciprocal transplant, we have 2 of 4 treatment combinations that count toward the sum (and 2 that should but do not). If we have a 4x4 reciprocal transplant, we have 4 of 16 reciprocal transplants that count toward the sum (and 12 that should but do not). We can get the SS to the right amount by multiplying  $(g-1)*e$  to get the same number of observations as the other SS calculations. At this point I'm only confident this will work in a reciprocal transplant case, would have to test  $g=4$  and  $e=2$  level cases etc.

$$V_{error} = \sum_{i=1}^g \sum_{j=1}^e \sum_{k=1}^r (y_{ijk} - \bar{y}_{ij})^2$$

Some evidence that this is behaving as expected:

if you set  $\text{beta1} = 1$  and  $\text{beta2} = 1$  and  $\text{int} = 0$ , you get equal amounts of variance explained for  $G$ ,  $E$ , and  $Cov_{GE}$  for 2 pops, but  $Cov_{GE}$  increases with more pops

if you set  $\text{beta1} = 1$  and  $\text{beta2} = 0.1$  and  $\text{int} = 0$ , you get more variance explained for  $V_E$ , a little bit for  $V_G$ , and a decent amount for  $cov_{GE}$

if you set  $\text{beta1} = 0.1$  and  $\text{beta2} = 1$  and  $\text{int} = 0$ , you get the same pattern as above but  $V_E$  and  $V_G$  switched

if you set  $\text{beta1}$  or  $\text{beta2}$  negative and the other positive, you get negative  $cov_{GE}$

try adding noise and see if outputs make sense \* Adding noise can sometimes "create" some  $Cov_{GE}$  when there are enough pops and  $\text{beta1}=0$  or  $\text{beta2}=0$  \* But for most part seems to work (?)

try increasing  $n_{pop}$  and see if outputs make sense \* increasing  $n_{pop}$  can lead to a greater amount of variance explained by  $Cov_{GE}$  and less by  $V_G$  or  $V_E$

## Step 1: create true phenotype data without error

```

beta1 = 10 # the amount the phenotype changes across 1 value of the environment (i.e., the slope). This
n_genotypes = 10 # the number of genotypes collected from different locations
n = 10 # sample size
beta2 = 10 # the amount the phenotype changes from one genotype to the next. This is essentially the inc

scale = 0.01 # the scale for sd_noise within pops compared to sd_means among pops
# I'll let you explore the space, but I think the upper would be scale = 1
# I'd start with scale = 1 and scale = 1/2 and see what that looks like

sd_int = 0#n_genotypes # sd of the interaction terms that will be drawn
# beta1 <- 1
# beta2 <- 1

# Generate data

n_environments = n_genotypes # the number of common garden environments that the genotypes are grow

## Create levels of genotypes and environments ####
G = rep(1:n_genotypes, each=n*n_genotypes) # '0' n times, '1' n times
E = rep(1:n_environments,times=n*n_genotypes) # '0'x50, '1'x50, '0'x50, '1'x50
#set.seed(97)

## Create interaction effect for each level of both factors ####
# In this case we assume the GxE interactions are a
# normally distributed random variable with a mean of 0
# As the sd increases, so does the GxE among treatments
if(sd_int == 0){
  int = 0
}else{
  int <- rnorm(n_genotypes * n_environments, 0, sd=sd_int)
}
# no GxE

# this sd determines the amount of GxE
int_df <- data.frame(expand.grid(G=1:n_genotypes, E=1:n_environments),
                     int)

### Create the model dataframe ###
model_df <- data.frame(G, E)
model_df$GE_factor <- paste0("G",model_df$G, "E", model_df$E)
model_df <- merge(model_df, int_df)
model_df <- model_df[order(model_df$G, model_df$E),]
dim(model_df)

## [1] 1000    4

head(model_df, 30)

##      G E GE_factor int
## 1  1 1      G1E1    0
## 2  1 1      G1E1    0
## 3  1 1      G1E1    0

```

```
## 4 1 1      G1E1  0
## 5 1 1      G1E1  0
## 6 1 1      G1E1  0
## 7 1 1      G1E1  0
## 8 1 1      G1E1  0
## 9 1 1      G1E1  0
## 10 1 1     G1E1  0
## 21 1 2     G1E2  0
## 22 1 2     G1E2  0
## 23 1 2     G1E2  0
## 24 1 2     G1E2  0
## 25 1 2     G1E2  0
## 26 1 2     G1E2  0
## 27 1 2     G1E2  0
## 28 1 2     G1E2  0
## 29 1 2     G1E2  0
## 30 1 2     G1E2  0
## 31 1 3     G1E3  0
## 32 1 3     G1E3  0
## 33 1 3     G1E3  0
## 34 1 3     G1E3  0
## 35 1 3     G1E3  0
## 36 1 3     G1E3  0
## 37 1 3     G1E3  0
## 38 1 3     G1E3  0
## 39 1 3     G1E3  0
## 40 1 3     G1E3  0
```

```
# Generate phenotype data using the regression equation ####
```

```
model_df$GE_true = beta1*model_df$E + beta2*model_df$G + model_df$int
```

```
G_true <- data.frame(G=1:n_genotypes, G_true=tapply(model_df$GE_true, model_df$G, mean))
```

```
E_true <- data.frame(E=1:n_genotypes, E_true=tapply(model_df$GE_true, model_df$E, mean))
```

```
model_df1 <- merge(model_df, G_true)
```

```
model_df2 <- merge(model_df1, E_true)
```

```
model_df <- model_df2
```

```
model_df$mean_true <- mean(model_df$GE_true)
```

```
model_df$int_true <- model_df$mean_true + model_df$GE_true -
  model_df$G_true - model_df$E_true
```

```
head(model_df)
```

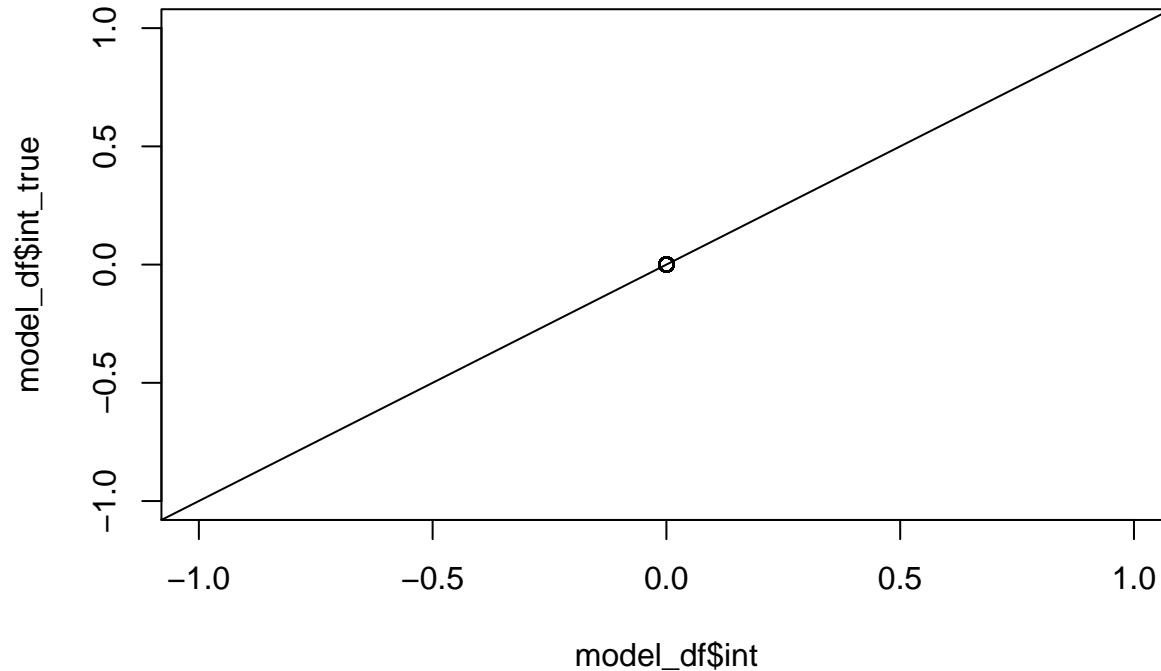
```
##   E G GE_factor int GE_true G_true E_true mean_true int_true
## 1 1 1      G1E1  0      20      65      65      110      0
## 2 1 1      G1E1  0      20      65      65      110      0
## 3 1 1      G1E1  0      20      65      65      110      0
## 4 1 1      G1E1  0      20      65      65      110      0
## 5 1 1      G1E1  0      20      65      65      110      0
## 6 1 1      G1E1  0      20      65      65      110      0
```

```
tail(model_df)
```

```
##       E  G GE_factor int GE_true G_true E_true mean_true int_true
## 995 10 10   G10E10  0     200    155    155      110      0
```

```
## 996 10 10 G10E10 0 200 155 155 110 0
## 997 10 10 G10E10 0 200 155 155 110 0
## 998 10 10 G10E10 0 200 155 155 110 0
## 999 10 10 G10E10 0 200 155 155 110 0
## 1000 10 10 G10E10 0 200 155 155 110 0
```

```
par(mfrow=c(1,1))
plot(model_df$int, model_df$int_true)
abline(0,1)
```



Note that the way we add “int” to create phenotypic variation is not the same way that the actual interaction is calculated from the data. Our simulations create an interaction, but as we create an interaction we also change the G-means and E-means, and so the calculated interaction is a bit different.

## Step 2: standardize true phenotype data

```
GE_true_means <- tapply(model_df$GE_true, model_df$GE_factor, mean)

model_df$GE_stn_true <- (model_df$GE_true - mean(GE_true_means))/sd(GE_true_means)

GE_stn_true_means <- tapply(model_df$GE_stn_true, model_df$GE_factor, mean)

G_stn_true <- data.frame(G=1:n_genotypes, G_stn_true=tapply(model_df$GE_stn_true, model_df$G, mean))
E_stn_true <- data.frame(E=1:n_genotypes, E_stn_true=tapply(model_df$GE_stn_true, model_df$E, mean))

model_df1 <- merge(model_df, G_stn_true)
model_df2 <- merge(model_df1, E_stn_true)
model_df <- model_df2

model_df$mean_stn_true <- mean(model_df$GE_stn_true)

model_df$int_stn_true <- model_df$mean_stn_true + model_df$GE_stn_true -
```

```

model_df$G_stn_true - model_df$E_stn_true

head(model_df)

```

```

##   E G GE_factor int GE_true G_true E_true mean_true int_true GE_stn_true
## 1 1 1      G1E1  0      20      65      65      110      0 -2.204541
## 2 1 1      G1E1  0      20      65      65      110      0 -2.204541
## 3 1 1      G1E1  0      20      65      65      110      0 -2.204541
## 4 1 1      G1E1  0      20      65      65      110      0 -2.204541
## 5 1 1      G1E1  0      20      65      65      110      0 -2.204541
## 6 1 1      G1E1  0      20      65      65      110      0 -2.204541
##   G_stn_true E_stn_true mean_stn_true int_stn_true
## 1 -1.10227 -1.10227      0      0
## 2 -1.10227 -1.10227      0      0
## 3 -1.10227 -1.10227      0      0
## 4 -1.10227 -1.10227      0      0
## 5 -1.10227 -1.10227      0      0
## 6 -1.10227 -1.10227      0      0

```

```

tail(model_df)

```

```

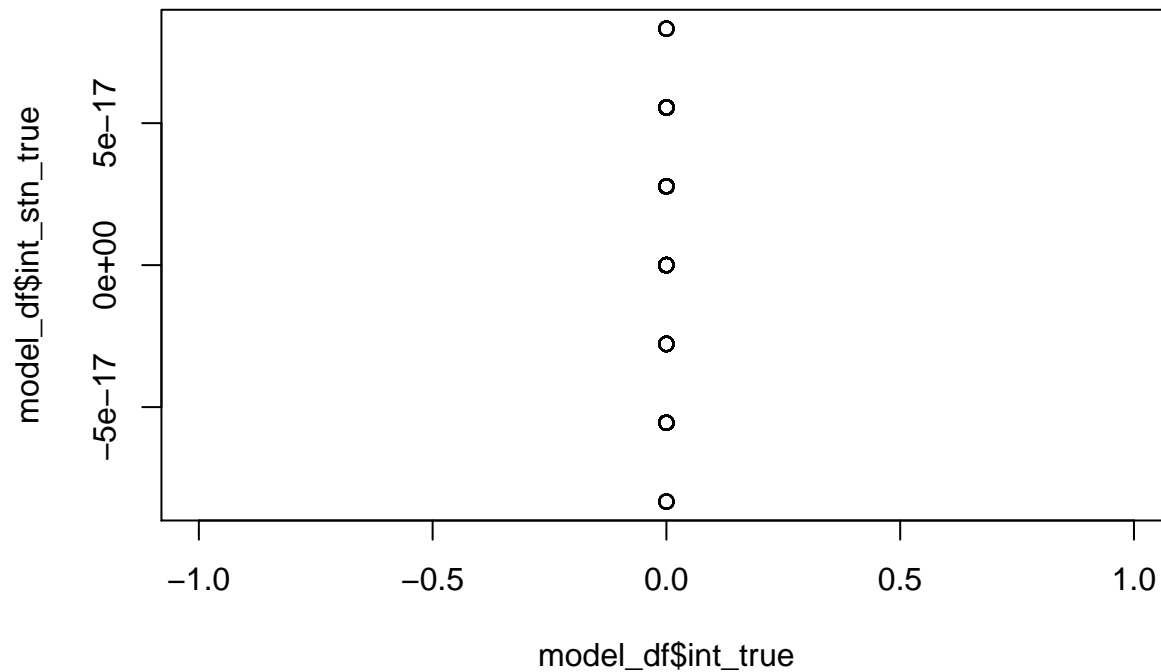
##   E G GE_factor int GE_true G_true E_true mean_true int_true GE_stn_true
## 995 10 9      G9E10  0      190      145      155      110      0  1.9595918
## 996 10 7      G7E10  0      170      125      155      110      0  1.4696938
## 997 10 6      G6E10  0      160      115      155      110      0  1.2247449
## 998 10 3      G3E10  0      130      85      155      110      0  0.4898979
## 999 10 8      G8E10  0      180      135      155      110      0  1.7146428
## 1000 10 6      G6E10  0      160      115      155      110      0  1.2247449
##   G_stn_true E_stn_true mean_stn_true int_stn_true
## 995  0.8573214  1.10227      0      0
## 996  0.3674235  1.10227      0      0
## 997  0.1224745  1.10227      0      0
## 998 -0.6123724  1.10227      0      0
## 999  0.6123724  1.10227      0      0
## 1000 0.1224745  1.10227      0      0

```

```

par(mfrow=c(1,1))
plot(model_df$int_true, model_df$int_stn_true)

```



```
# awesome
```

### Step 3: add error to stnd phenotype data

scaled to the variation in the phenotype means

```
sd(GE_stn_true_means)
```

```
## [1] 1
```

```
# I guess this will always be 1 because we standardize.
```

```
sd_noise <- sd(as.numeric(GE_stn_true_means))*scale
```

```
model_df$e = rnorm(n*n_genotypes*n_environments, 0, sd=sd_noise) # Random noise
```

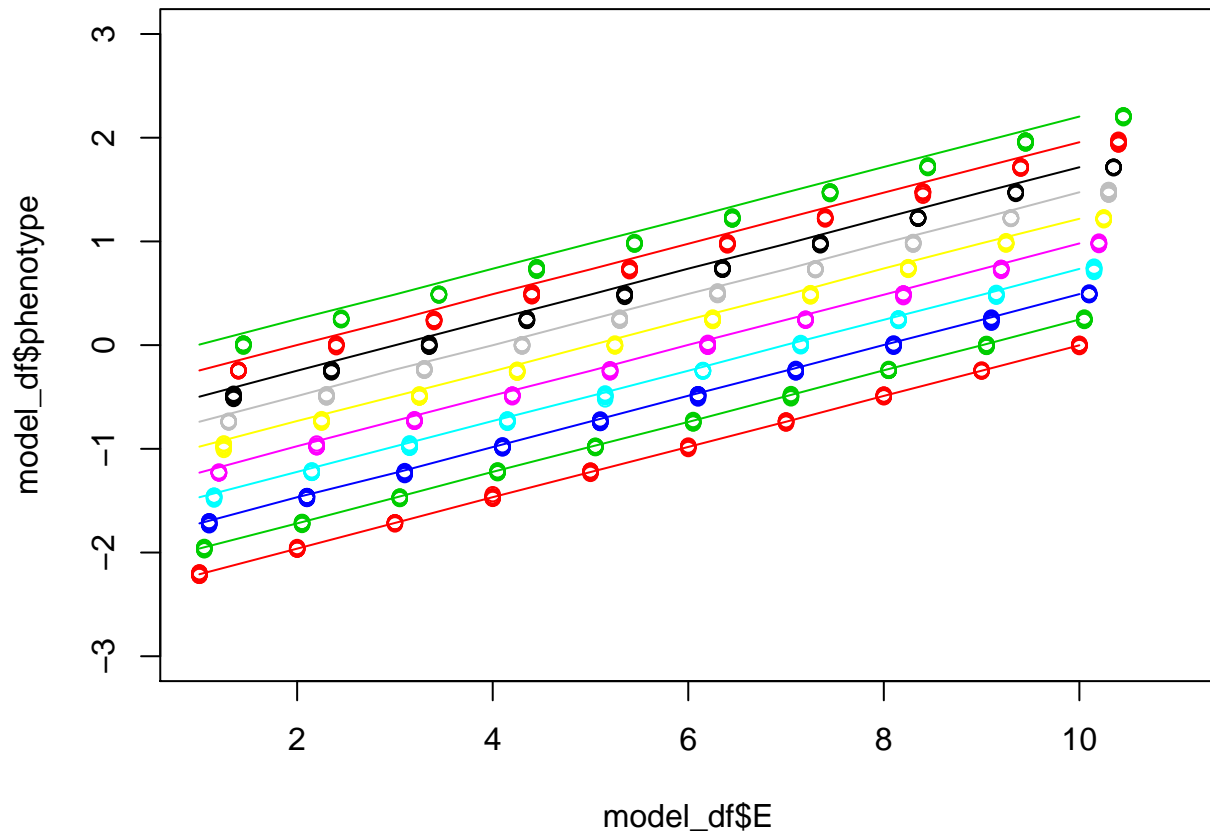
```
model_df$phenotype <- model_df$GE_stn_true + model_df$e
```

```
head(model_df)
```

```
##   E G GE_factor int GE_true G_true E_true mean_true int_true GE_stn_true
## 1 1 1      G1E1   0      20     65     65      110         0 -2.204541
## 2 1 1      G1E1   0      20     65     65      110         0 -2.204541
## 3 1 1      G1E1   0      20     65     65      110         0 -2.204541
## 4 1 1      G1E1   0      20     65     65      110         0 -2.204541
## 5 1 1      G1E1   0      20     65     65      110         0 -2.204541
## 6 1 1      G1E1   0      20     65     65      110         0 -2.204541
##   G_stn_true E_stn_true mean_stn_true int_stn_true      e phenotype
## 1  -1.10227  -1.10227              0              0 -0.020174463 -2.224715
## 2  -1.10227  -1.10227              0              0 -0.010773675 -2.215314
## 3  -1.10227  -1.10227              0              0  0.006140117 -2.198401
## 4  -1.10227  -1.10227              0              0 -0.011312425 -2.215853
## 5  -1.10227  -1.10227              0              0 -0.013353474 -2.217894
## 6  -1.10227  -1.10227              0              0 -0.011807912 -2.216349
```

## Step 4: plot the pattern so we can see what it looks like

```
par(mfrow=c(1,1), mar=c(4,4,1,1))
plot(model_df$phenotype~model_df$E, col=0, xlim=c(1,n_genotypes+1), ylim=c(-3,3))
for (m in 1:n_genotypes){
  x <- model_df$E[model_df$G==m]+0.05*(m-1)
  points(model_df$phenotype[model_df$G==m]~x, col=m+1)
  p_means <- tapply(model_df$phenotype[model_df$G==m],model_df$E[model_df$G==m], mean)
  points(p_means~as.numeric(names(p_means)),
        col=m+1, type="l")
}
```

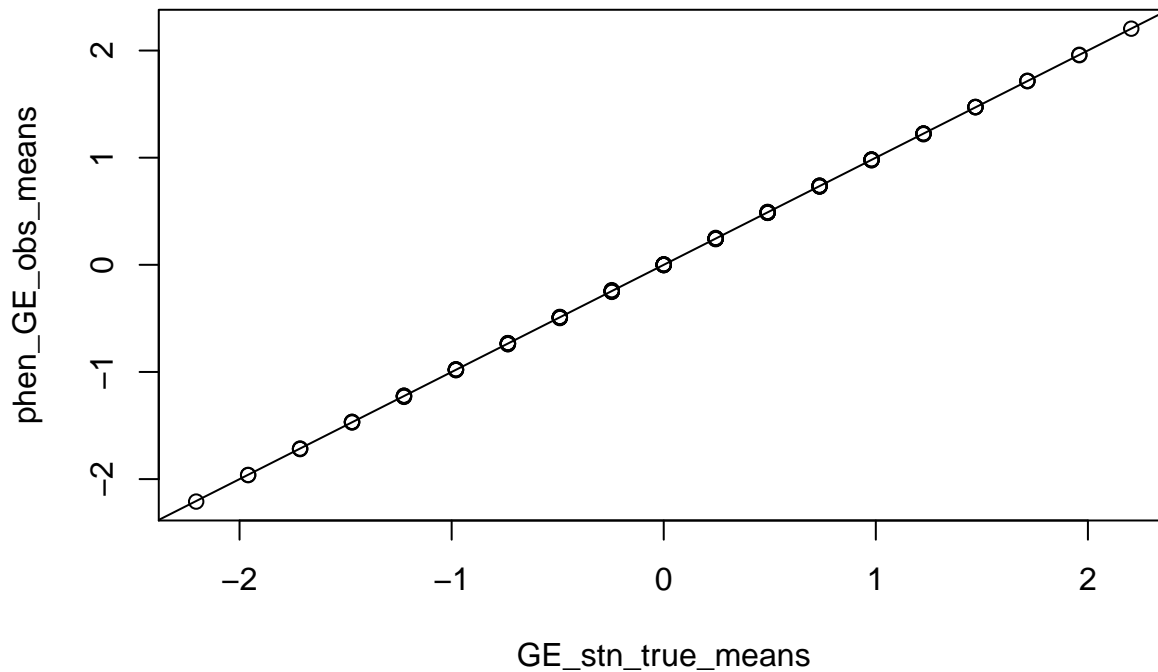


## Step 5: pretend we start with noisy simulated data

Go through the steps we will go through in the study

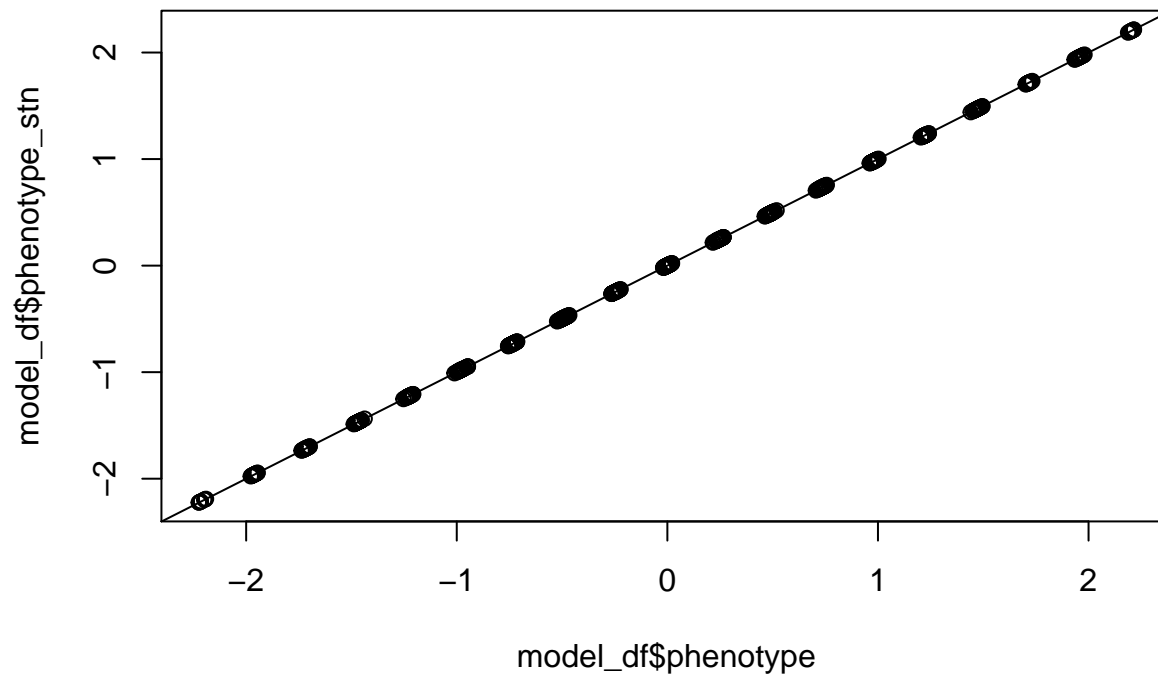
```
# Step A: standardize by mean(GE_means) and sd(GE_means)
phen_GE_obs_means <- tapply(model_df$phenotype, model_df$GE_factor, mean)

plot(GE_stn_true_means, phen_GE_obs_means) #sanity check
abline(0,1)
```



```
model_df$phenotype_stn <- (model_df$phenotype - mean(phen_GE_obs_means)) / sd(phen_GE_obs_means)

plot(model_df$phenotype, model_df$phenotype_stn)
abline(0,1)
```



```
# Step B: calculate observed G_means and E_means and interaction
G_stn_est <- data.frame(G=1:n_genotypes, G_stn_est=apply(model_df$phenotype_stn, model_df$G, mean))
E_stn_est <- data.frame(E=1:n_genotypes, E_stn_est=apply(model_df$phenotype_stn, model_df$E, mean))

GE_stn_est_means <- apply(model_df$phenotype_stn, model_df$GE_factor, mean)
GE_stn_est_means_df <- data.frame(GE_factor=names(GE_stn_est_means), GE_stn_est=GE_stn_est_means)
```



```

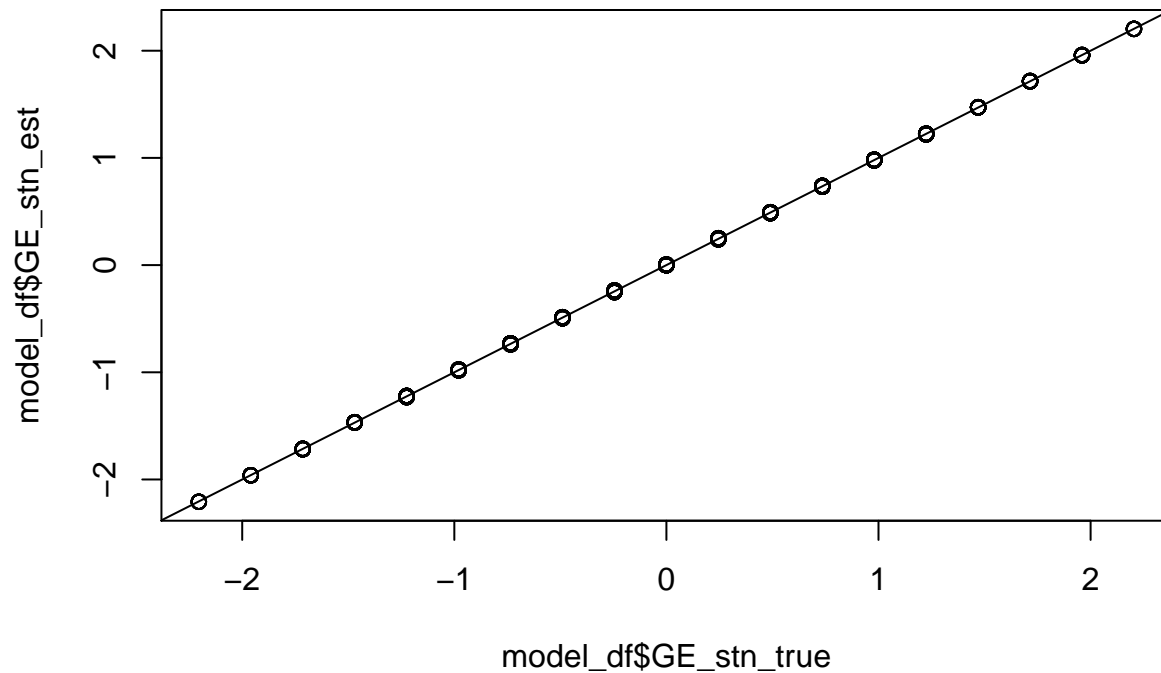
model_df1 <- merge(model_df, G_stn_est)
model_df2 <- merge(model_df1, E_stn_est)
model_df3 <- merge(model_df2, GE_stn_est_means_df)
model_df <- model_df3

```

```

plot(model_df$GE_stn_true, model_df$GE_stn_est)
abline(0,1) # looks good

```



```

model_df$mean_stn_est <- mean(model_df$phenotype_stn)

# Calculate interaction
model_df$int_stn_est <- model_df$mean_stn_est + model_df$GE_stn_est -
                        model_df$G_stn_est - model_df$E_stn_est

```

## Step 6: Compare true values to estimated values

```
head(model_df)
```

```

##   GE_factor E  G int GE_true G_true E_true mean_true int_true GE_stn_true
## 1   G10E1 1 10  0   110   155   65     110         0           0
## 2   G10E1 1 10  0   110   155   65     110         0           0
## 3   G10E1 1 10  0   110   155   65     110         0           0
## 4   G10E1 1 10  0   110   155   65     110         0           0
## 5   G10E1 1 10  0   110   155   65     110         0           0
## 6   G10E1 1 10  0   110   155   65     110         0           0
##   G_stn_true E_stn_true mean_stn_true int_stn_true      e      phenotype
## 1   1.10227  -1.10227          0          0 -6.073255e-04 -6.073255e-04
## 2   1.10227  -1.10227          0          0  1.423871e-02  1.423871e-02
## 3   1.10227  -1.10227          0          0  1.364169e-02  1.364169e-02
## 4   1.10227  -1.10227          0          0  1.226333e-02  1.226333e-02
## 5   1.10227  -1.10227          0          0  5.721586e-03  5.721586e-03

```

```
## 6      1.10227   -1.10227           0           0 -4.673862e-05 -4.673862e-05
##      phenotype_stn G_stn_est E_stn_est  GE_stn_est mean_stn_est int_stn_est
## 1 -0.0001974582    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 2  0.0146421836    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 3  0.0140454142    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 4  0.0126676471    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 5  0.0061287256    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 6  0.0003628870    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823

(true_int <- mean(abs(model_df$int_stn_true)))

## [1] 1.221245e-17

(obs_int <- mean(abs(model_df$int_stn_est)))

## [1] 0.002389785
```

## New step 6.5: make ANOVA table with CovGE

Here, I didn't write loops to calculate the equations, so the code will look different from the equations.

TO DO: check to code the loops as written in the equations and make sure it gives the same answer as what I calculated here

```
head(model_df)

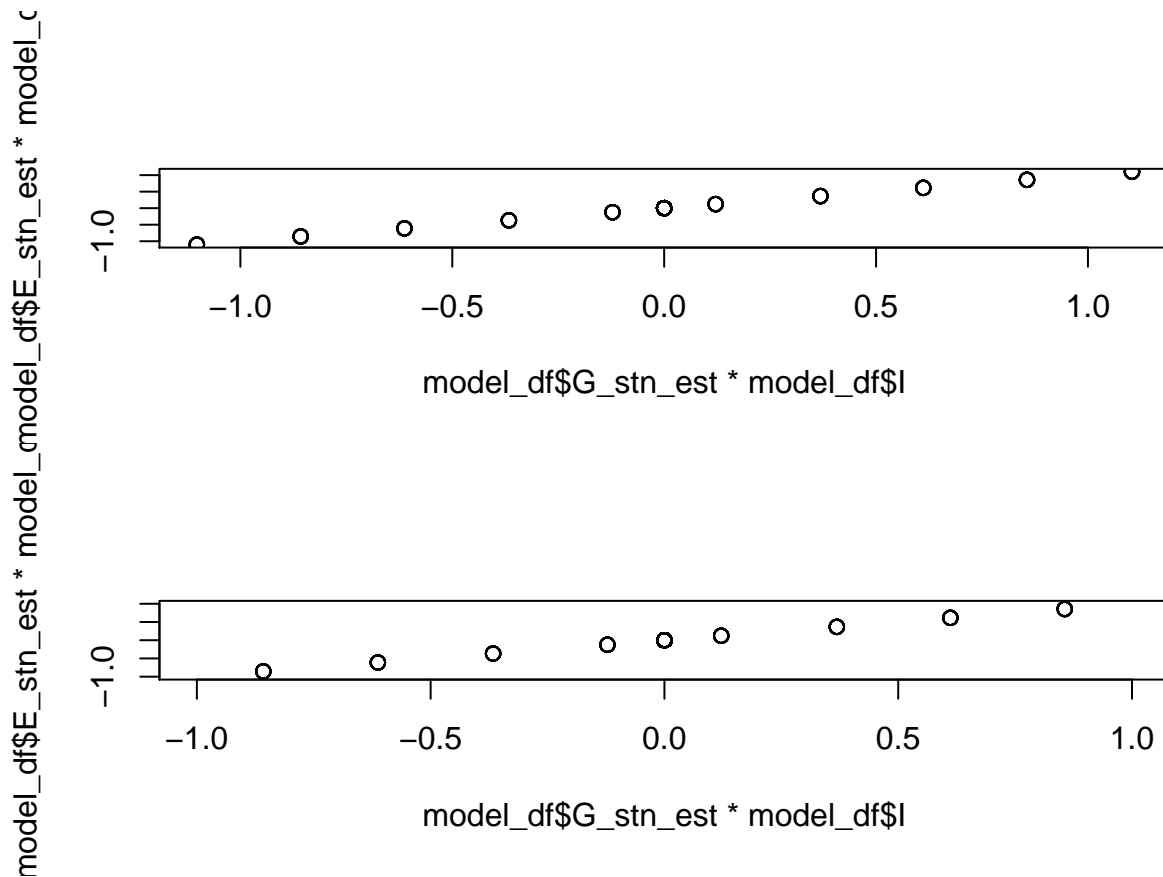
##      GE_factor E  G int GE_true G_true E_true mean_true int_true GE_stn_true
## 1      G10E1 1 10  0      110      155      65      110      0              0
## 2      G10E1 1 10  0      110      155      65      110      0              0
## 3      G10E1 1 10  0      110      155      65      110      0              0
## 4      G10E1 1 10  0      110      155      65      110      0              0
## 5      G10E1 1 10  0      110      155      65      110      0              0
## 6      G10E1 1 10  0      110      155      65      110      0              0
##      G_stn_true E_stn_true mean_stn_true int_stn_true          e      phenotype
## 1      1.10227   -1.10227           0           0 -6.073255e-04 -6.073255e-04
## 2      1.10227   -1.10227           0           0  1.423871e-02  1.423871e-02
## 3      1.10227   -1.10227           0           0  1.364169e-02  1.364169e-02
## 4      1.10227   -1.10227           0           0  1.226333e-02  1.226333e-02
## 5      1.10227   -1.10227           0           0  5.721586e-03  5.721586e-03
## 6      1.10227   -1.10227           0           0 -4.673862e-05 -4.673862e-05
##      phenotype_stn G_stn_est E_stn_est  GE_stn_est mean_stn_est int_stn_est
## 1 -0.0001974582    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 2  0.0146421836    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 3  0.0140454142    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 4  0.0126676471    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 5  0.0061287256    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823
## 6  0.0003628870    1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823

V_G_SS = sum((model_df$G_stn_est-model_df$mean_stn_est)^2)
V_E_SS = sum((model_df$E_stn_est-model_df$mean_stn_est)^2)
V_GE_SS = sum(model_df$int_stn_est^2)
V_error = sum((model_df$phenotype_stn - model_df$GE_stn_est)^2)

model_df$I = model_df$E==model_df$G

# Covariance pattern (ignore the 0,0 points)
par(mfrow=c(2,1))
```

```
plot(model_df$G_stn_est*model_df$I, model_df$E_stn_est*model_df$I)
plot(model_df$G_stn_est*model_df$I, model_df$E_stn_est*model_df$I, xlim=c(-1,1), ylim=c(-1,1))
```



```
V_Cov_GE <- (nlevels(factor(model_df$G))-1)*
  nlevels(factor(model_df$E))*
  sum((model_df$G_stn_est-model_df$mean_stn_est)*(model_df$E_stn_est-model_df$mean_stn_est)*model_df$I)

SS <- round(rbind(V_G_SS, V_E_SS, V_GE_SS, V_Cov_GE, V_error),2)
omega2 <- round(abs(SS)/sum(abs(SS)),2)
data.frame(SS, abs(SS), omega2)
```

```
##          SS abs.SS. omega2
## V_G_SS    495.30  495.30   0.09
## V_E_SS    494.69  494.69   0.09
## V_GE_SS     0.01   0.01   0.00
## V_Cov_GE 4454.95 4454.95   0.82
## V_error     0.09   0.09   0.00
```

## Step 7: pretend we start with noisy empirical data

Let's back transform the noisy data, then go through the steps and see what happens

```
# what we did to standardize: model_df$GE_stn_true <- (model_df$GE_true - mean(GE_true_means))/sd(GE_true_means)
model_df2 <- model_df[,1:3]
model_df2$phenotype2 <- model_df$phenotype*sd(GE_true_means)+mean(GE_true_means) #backtransform
```

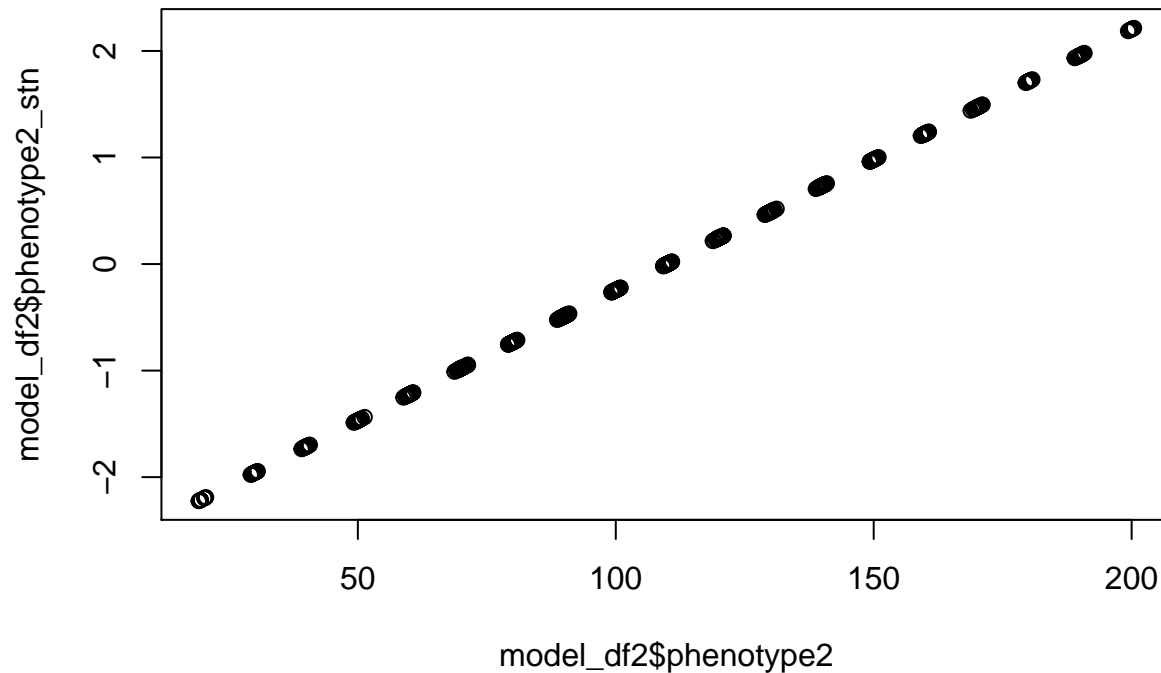
```

# Step A: standardize by mean(GE_means) and sd(GE_means)
phen_GE_obs_means <- tapply(model_df2$phenotype2, model_df2$GE_factor, mean)

model_df2$phenotype2_stn <- (model_df2$phenotype2 - mean(phen_GE_obs_means)) / sd(phen_GE_obs_means)

plot(model_df2$phenotype2, model_df2$phenotype2_stn) # straight line check

```



```

# Step B: calculate observed G_means and E_means and interaction
G_stn_est <- data.frame(G=1:n_genotypes, G_stn_est=tapply(model_df2$phenotype2_stn, model_df2$G, mean))
E_stn_est <- data.frame(E=1:n_genotypes, E_stn_est=tapply(model_df2$phenotype2_stn, model_df2$E, mean))

GE_stn_est_means <- tapply(model_df2$phenotype2_stn, model_df2$GE_factor, mean)
GE_stn_est_means_df <- data.frame(GE_factor=names(GE_stn_est_means), GE_stn_est=GE_stn_est_means)

model_dfa <- merge(model_df2, G_stn_est)
model_dfb <- merge(model_dfa, E_stn_est)
model_dfc <- merge(model_dfb, GE_stn_est_means_df)
head(model_dfc)

```

```

##   GE_factor E  G phenotype2 phenotype2_stn G_stn_est E_stn_est  GE_stn_est
## 1    G10E1 1  10    110.5569    0.0140454142    1.10397 -1.103204  0.005632742
## 2    G10E1 1  10    109.9752   -0.0001974582    1.10397 -1.103204  0.005632742
## 3    G10E1 1  10    110.5813    0.0146421836    1.10397 -1.103204  0.005632742
## 4    G10E1 1  10    110.6155    0.0154790149    1.10397 -1.103204  0.005632742
## 5    G10E1 1  10    110.0082    0.0006110880    1.10397 -1.103204  0.005632742
## 6    G10E1 1  10    109.3813   -0.0147391211    1.10397 -1.103204  0.005632742

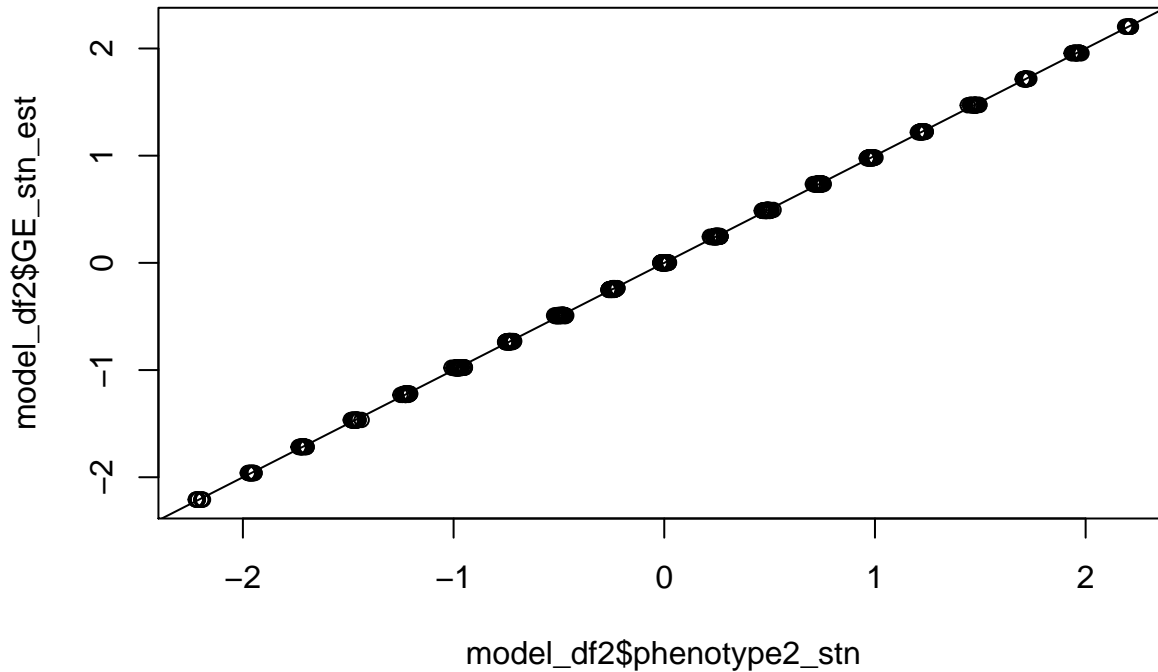
```

```
model_df2 <- model_dfc
```

```

plot(model_df2$phenotype2_stn, model_df2$GE_stn_est)
abline(0,1) # looks good

```



```
model_df2$mean_stn_est <- mean(model_df2$phenotype2_stn)

# Calculate interaction
model_df2$int_stn_est <- model_df2$mean_stn_est + model_df2$GE_stn_est -
  model_df2$G_stn_est - model_df2$E_stn_est
```

## Step 8: Compare true values to estimated values

```
head(model_df)
```

```
##   GE_factor E   G int GE_true G_true E_true mean_true int_true GE_stn_true
## 1   G10E1 1 10   0   110   155   65   110       0         0
## 2   G10E1 1 10   0   110   155   65   110       0         0
## 3   G10E1 1 10   0   110   155   65   110       0         0
## 4   G10E1 1 10   0   110   155   65   110       0         0
## 5   G10E1 1 10   0   110   155   65   110       0         0
## 6   G10E1 1 10   0   110   155   65   110       0         0
##   G_stn_true E_stn_true mean_stn_true int_stn_true      e      phenotype
## 1   1.10227  -1.10227          0          0 -6.073255e-04 -6.073255e-04
## 2   1.10227  -1.10227          0          0  1.423871e-02  1.423871e-02
## 3   1.10227  -1.10227          0          0  1.364169e-02  1.364169e-02
## 4   1.10227  -1.10227          0          0  1.226333e-02  1.226333e-02
## 5   1.10227  -1.10227          0          0  5.721586e-03  5.721586e-03
## 6   1.10227  -1.10227          0          0 -4.673862e-05 -4.673862e-05
##   phenotype_stn G_stn_est E_stn_est  GE_stn_est mean_stn_est int_stn_est   I
## 1 -0.0001974582  1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823 FALSE
## 2  0.0146421836  1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823 FALSE
## 3  0.0140454142  1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823 FALSE
## 4  0.0126676471  1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823 FALSE
## 5  0.0061287256  1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823 FALSE
## 6  0.0003628870  1.10397 -1.103204 0.005632742 2.716078e-17 0.004866823 FALSE
```

```
(true_int <- mean(abs(model_df$int_stn_true)))

## [1] 1.221245e-17
# Although we call this "true_int", when int=0 this should be 0
# note this is a shortcut that only works with equal sample sizes

(obs_int_sim <- mean(abs(model_df$int_stn_est)))

## [1] 0.002389785
# when int=0, this will increase as the within-population mean gets less accurate

(obs_int_emp <- mean(abs(model_df2$int_stn_est)))

## [1] 0.002389785
```

Drop the mic.