Nonlinear Transformation

Karol Buda

7/12/2022

## Simulation Code

set.seed(123) # Constant Seed  
  
# Create a library of 4 mutants  
sim\_df = as.data.frame(gtools::permutations(2, 4, c(-1, 1), repeats = TRUE))  
  
colnames(sim\_df) = c("p1", "p2", "p3", "p4")  
  
geno\_tab = gtools::permutations(2, 4, c(-1, 1), repeats = TRUE)  
geno\_tab[which(geno\_tab == -1)] = 2  
  
geno\_tab = matrix(c("Y", "X")[geno\_tab], nrow = 16, ncol = 4)  
  
geno\_tab = apply(geno\_tab, 1, function(x) paste(x, collapse = ""))  
  
### MAKE SURE GENOTYPES MATCH ORDER OF EFFECT COLUMNS  
  
# Defined epistatic coefficents  
  
pos1 = 2  
pos2 = 4  
pos3 = 0.5  
pos4 = 1  
pos1\_pos2 = 1  
pos1\_pos3 = 1  
pos1\_pos4 = 5  
pos2\_pos3 = 1  
pos2\_pos4 = 0.5  
pos3\_pos4 = 1  
pos1\_pos2\_pos3 = 1  
pos1\_pos2\_pos4 = 3  
pos1\_pos3\_pos4 = 1  
pos2\_pos3\_pos4 = 1  
pos1\_pos2\_pos3\_pos4 = 1  
  
# Define positional combinations  
  
sim\_df$p1\_p2 = sim\_df$p1 == 1 & sim\_df$p2 == 1  
sim\_df$p1\_p3 = sim\_df$p1 == 1 & sim\_df$p3 == 1  
sim\_df$p1\_p4 = sim\_df$p1 == 1 & sim\_df$p4 == 1  
sim\_df$p2\_p3 = sim\_df$p2 == 1 & sim\_df$p3 == 1  
sim\_df$p2\_p4 = sim\_df$p2 == 1 & sim\_df$p4 == 1  
sim\_df$p3\_p4 = sim\_df$p3 == 1 & sim\_df$p4 == 1  
  
sim\_df$p1\_p2\_p3 = sim\_df$p1 == 1 & sim\_df$p2 == 1 & sim\_df$p3 == 1  
sim\_df$p1\_p2\_p4 = sim\_df$p1 == 1 & sim\_df$p2 == 1 & sim\_df$p4 == 1  
sim\_df$p1\_p3\_p4 = sim\_df$p1 == 1 & sim\_df$p3 == 1 & sim\_df$p4 == 1  
sim\_df$p2\_p3\_p4 = sim\_df$p2 == 1 & sim\_df$p3 == 1 & sim\_df$p4 == 1  
  
sim\_df$p1\_p2\_p3\_p4 = sim\_df$p1 == 1 & sim\_df$p2 == 1 & sim\_df$p3 == 1 & sim\_df$p4 == 1  
  
# Simulate effects based on a starting genotype (start)  
  
effect = c()  
for(i in 1:dim(sim\_df)[1]){  
 start = 10  
 if(sim\_df[i,1] == 1) {  
 start = start\*pos1  
 }   
 if(sim\_df[i,2] == 1) {  
 start = start\*pos2  
 }   
 if(sim\_df[i,3] == 1) {  
 start = start\*pos3  
 }   
 if(sim\_df[i,4] == 1) {  
 start = start\*pos4  
 }   
 if(sim\_df[i,5] == 1) {  
 start = start\*pos1\_pos2  
 }   
 if(sim\_df[i,6] == 1) {  
 start = start\*pos1\_pos3  
 }   
 if(sim\_df[i,7] == 1) {  
 start = start\*pos1\_pos4  
 }   
 if(sim\_df[i,8] == 1) {  
 start = start\*pos2\_pos3  
 }   
 if(sim\_df[i,9] == 1) {  
 start = start\*pos2\_pos4  
 }   
 if(sim\_df[i,10] == 1) {  
 start = start\*pos3\_pos4  
 }  
 if(sim\_df[i,11] == 1) {  
 start = start\*pos1\_pos2\_pos3  
 }  
 if(sim\_df[i,12] == 1) {  
 start = start\*pos1\_pos2\_pos4  
 }  
 if(sim\_df[i,13] == 1) {  
 start = start\*pos1\_pos3\_pos4  
 }  
 if(sim\_df[i,14] == 1) {  
 start = start\*pos2\_pos3\_pos4  
 }  
 if(sim\_df[i,15] == 1) {  
 start = start\*pos1\_pos2\_pos3\_pos4  
 }  
 effect = c(effect, start)  
}  
  
# Log transform the effects  
  
sim\_df\_effect = log10(effect)  
  
## Formatting and output to mimic datasets used in epistatic analysis code  
  
out\_df = data.frame("first" = c("WT", "XXXX", "Positions", 1, "Genotype", rep(geno\_tab, each = 1)), "second" = c(rep(NA, 3), 2, "Effect", sim\_df\_effect),  
 "third" = c(rep(NA, 3), 3, rep(NA, length(sim\_df\_effect) + 1)), "fourth" = c(rep(NA, 3), 4, rep(NA, length(sim\_df\_effect) + 1)))  
  
  
## Exporting  
  
write.table(out\_df, "sim.csv", row.names = F, col.names = F, na = "", sep = ",")

## Epistatic Data

We tested the non-linear transformation technique1 from Sailer & Harms on non-idiosyncratic, linear (additive), epistatic data.

We simulated a genotype-phenotype map with 4 mutated positions using the simple\_sim.R script.

Coefficients were as follows:

* p1: 2-fold
* p2: 4-fold
* p3: 0.5-fold
* p4: 1-fold
* p1p4: 5-fold
* p2p4: 0.5-fold
* p1p2p4: 3-fold

First we imported the csv file generated by the simulation script

d = read\_csv("sim.csv",  
 show\_col\_types = F)  
  
d = d[5:dim(d)[1], 1:2]  
colnames(d) = c("Genotype", "Phenotype")  
d$Phenotype = as.numeric(d$Phenotype)  
  
head(d)

## # A tibble: 6 × 2  
## Genotype Phenotype  
## <chr> <dbl>  
## 1 XXXX 1   
## 2 XXXY 1   
## 3 XXYX 0.699  
## 4 XXYY 0.699  
## 5 XYXX 1.60   
## 6 XYXY 1.30

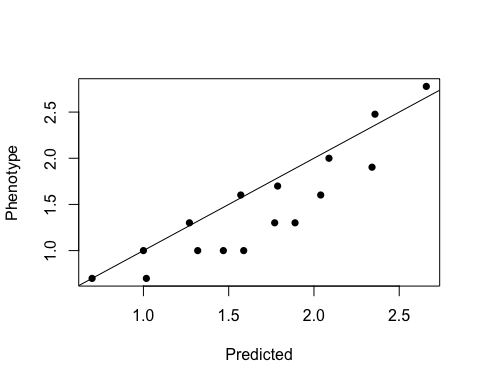
Then we processed the genotypes to 0 and 1 coefficients instead of X’s and Y’s

# Split Genotype into 0s and 1s to take the mean  
  
geno = do.call("rbind", str\_split(d$Genotype, ""))  
geno[geno == "X"] = 0  
geno[geno == "Y"] = 1  
  
geno = as\_tibble(geno)  
colnames(geno) = c("p1", "p2", "p3", "p4")  
geno$p1 = as.numeric(geno$p1)  
geno$p2 = as.numeric(geno$p2)  
geno$p3 = as.numeric(geno$p3)  
geno$p4 = as.numeric(geno$p4)  
  
geno$Phenotype = d$Phenotype  
  
head(geno)

## # A tibble: 6 × 5  
## p1 p2 p3 p4 Phenotype  
## <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 0 0 0 0 1   
## 2 0 0 0 1 1   
## 3 0 0 1 0 0.699  
## 4 0 0 1 1 0.699  
## 5 0 1 0 0 1.60   
## 6 0 1 0 1 1.30

Got the average functional contribution of each position (p1-4) and used an additive model to predict function.

## Average of each position  
dif = function(x){return(x[2]-x[1])}  
params = c()  
  
# Loop ever position and get the parameter for its average effect  
  
for(pos in 1:4) {  
 params = c(params, dif(geno %>%  
 group\_by\_at(pos) %>%  
 summarise(mean\_pheno = mean(Phenotype)) %>%  
 pull(mean\_pheno)))  
}  
  
# Predict  
  
pred\_pheno = c()  
for(row in 1:dim(geno)[1]) {  
 pred\_pheno = c(pred\_pheno, geno$Phenotype[1] + sum(geno[row,1:4] \* params))  
}  
  
# Output is a vector of pred\_pheno  
  
geno$Predicted = pred\_pheno  
  
# Plot of Padd vs Pobs  
  
plot(Phenotype ~ Predicted, geno, pch = 16)  
abline(0, 1)



We see apparent epistasis skewing the average functional effect prediction on the landscape. Keep in mind that Predicted here is the same as predictions stemming from the 1st order coefficients of a linear model.

## Spline Transform

What is the current R2 based on the plot we see above?

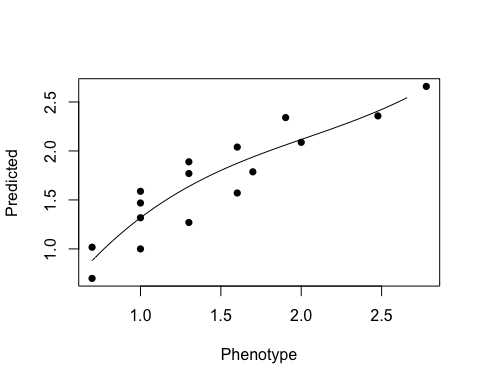
Currently it is 0.6759066

We can try to remove the non-linearity with a spline transform by looping from 1 to 10 knots and determining best fit by AIC

library(splines)  
  
aics = c()  
for(knot in 1:10) {  
 sm = lm(Predicted ~ bs(Phenotype, knots = knot), data = geno)  
 aics = c(aics, AIC(sm))  
}

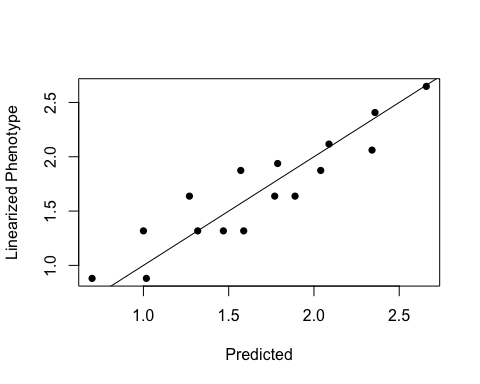
Best performing model is one with 3 knots

sm = lm(Predicted ~ bs(Phenotype, knots = which.min(aics)), data = geno)  
  
plot(Predicted ~ Phenotype, geno, xlab = "Phenotype", ylab = "Predicted", pch = 16)  
nlt <- seq(min(geno$Predicted), max(geno$Predicted), length.out = 200)  
lines(nlt, predict(sm, data.frame(Phenotype = nlt)))



So how does this transformation look like?

plot(predict(sm) ~ geno$Predicted, xlab = "Predicted", ylab = "Linearized Phenotype", pch = 16)  
abline(0, 1)

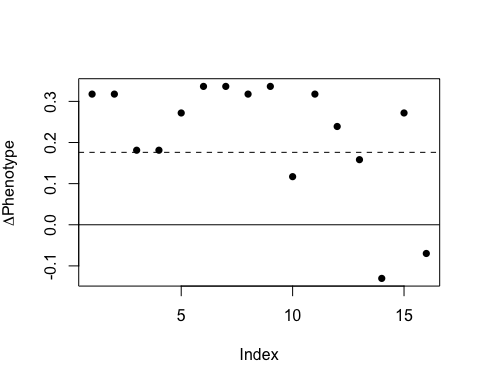


Has the fit improved according to the R2?

Now it is 0.8167167

And how much have we transformed the phenotype? Let’s look at the from original to non-linear transformed

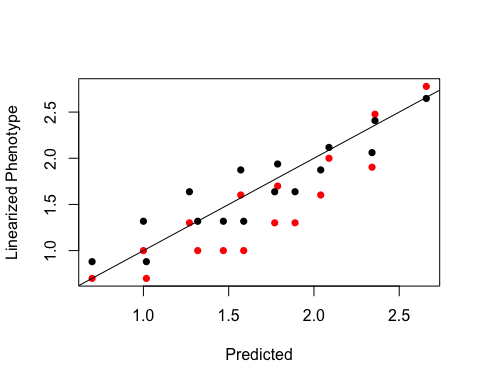
plot(predict(sm) - geno$Phenotype, xlab = "Index", ylab = expression(Delta\*"Phenotype"), pch = 16)  
abline(0, 0)  
abline(log10(1.5), 0, lty = 2)  
abline(log10(1/1.5), 0, lty = 2)



This is a large transformation beyond the 1.5-fold error (shown as dashed line) that is discussed in the main text.

We can also visualize this by plotting the previous pre-linearized phenotype in red.

plot(geno$Phenotype ~ geno$Predicted, xlab = "Predicted", ylab = "Linearized Phenotype", pch = 16, col = "red")  
points(predict(sm) ~ geno$Predicted, pch = 16)  
abline(0, 1)



Since we introduced no inherent non-linearity into the genotype-phenotype map, the non-linear transform is flattening the “true” epistasis we simulated.

Although appreciate the value in applying the non-linear transform to various data in order to reduce the effect of various non-linear properties on the system as much as possible, we chose not to use it for our data in order not to skew or minimize the inferred epistasis.