# Lifetime fitness through female and male function: the influence of density and genetically effective population size

*Mason W. Kulbaba & Ruth G. Shaw*

*Last Updated April 11, 2019*

## Contents



# Introduction

The following code performs fixed-effects aster analyses on data examining the effects of density and effective genetic population size ($N_e$) on female (seeds seet) fitness. LM analysis of biomass (above and below ground) and number of aborted ovules follows
the aster analyses.

Please send any questions to Mason Kulbaba (mason.kulbaba@gmail.com)

## Preliminaries

Set working directory and load data.

Examine data

```
fin<- read.csv("data/aster.dat.csv")

names(fin)
```

```
##  [1] "Treat"     "plotID"    "Den"       "Gen"       "plantID"
##  [6] "familyID"  "surv"      "flw"       "frt"       "frt.2"
## [11] "seeds"     "aborted"   "rel.seeds" "seed.wt"   "mass.a"
## [16] "mass.b"
```

```
head(fin)
```

```
##   Treat plotID Den Gen plantID familyID surv flw frt frt.2 seeds aborted
## 1  HDHG      5   H  HG       1        4    1   1   3     2    33      14
## 2  HDHG      5   H  HG       2        6    1   1  21     6    69      22
## 3  HDHG      5   H  HG       3       11    1   1  16     7    73      19
## 4  HDHG      5   H  HG       4       14    1   1  18     8   107      23
## 5  HDHG      5   H  HG       5        6    1   1   1     1    16       4
## 6  HDHG      5   H  HG       6       14    1   1   0     0     0       0
##   rel.seeds seed.wt mass.a mass.b
## 1 0.5714286   6.792    5.8    0.8
## 2 1.1948052   6.995    1.9    0.2
## 3 1.2640693   7.413   20.8    1.3
## 4 1.8528139   8.554   18.2    1.1
## 5 0.2770563   7.771   65.8    6.1
## 6 0.0000000   0.000    6.6    1.2
```

Make sure Den (density treatment), Gen ($N_e$ treatment), plotID, plantID, and familyID are all classified as factors. Otherwise

```
fin$Den<- as.factor(fin$Den)
fin$Gen<- as.factor(fin$Gen)
fin$plotID<- as.factor(fin$plotID)
fin$plantID<- as.factor(fin$plantID)
fin$familyID<- as.factor(fin$familyID)
```

## Setting up for aster analyses

Load aster package (make sure you have most current version)

```
library(aster)
```

```
## Loading required package: trust
```

Begin by naming variables that will be used in the graphical model of the aster analyses:

flw - total number of flowers produced frt - total number of fruits produced frt.2 - subsetted number of fruits collected seeds - total number of seeds collected from subsetted fruits

```
vars<- c( "flw", "frt", "frt.2","seeds")
```

Reshape the data so that all response variables are located in a single vector, in a new data set called “redata”

```
redata <- reshape(fin, varying = list(vars), direction = "long",timevar = "varb", times = as.factor(vars
```

Designate the terminal fitness variable “seeds” (make it numberic), and then add it to the reshaped data

```
fit <- grepl("seeds", as.character(redata$varb))
fit<- as.numeric(fit)
```

```
redata$fit <- fit
```

Check

```
with(redata, sort(unique(as.character(varb)[fit == 0])))
```

```
## [1] "flw"   "frt"   "frt.2"
```

```
with(redata, sort(unique(as.character(varb)[fit == 1])))
```

```
## [1] "seeds"
```

Add a variable "root" to redata, where value is 1. This is the "starting point" of the aster graphical model (i.e. a seed planted)

```
redata<- data.frame(redata, root=1)
```

Set up the graphical model and designate the statistical distribution for each node. This graphical model has four nodes (in object `pred`) described earlier. Statistical family for each node is described by object `fam`.

```
pred<- c(0,1,2,3)
fam<- c(1,2,2,2)
```

Show distribution family for each node

```
sapply(fam.default(), as.character)[fam]
```

```
## [1] "bernoulli" "poisson"   "poisson"   "poisson"
```

# Main Aster Analyses for Female Fitness (Seeds set)

First aster analysis with only fitness data. Note, `aster` reads the `redata` version of the data.

```
aout<- aster(resp~varb , pred, fam, varb, id, root, data=redata)

summary(aout, show.graph=T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb, pred = pred, fam = fam,
##     varvar = varb, idvar = id, root = root, data = redata)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -44.6476     0.5507  -81.08   <2e-16 ***
## varbfrt      49.1734     0.5599   87.82   <2e-16 ***
## varbfrt.2    34.2048     0.5538   61.76   <2e-16 ***
## varbseeds    46.9877     0.5507   85.32   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Add density to the model as a fixed effect. Note that we follow the "no naked predictors" rule with `fit:(Den)`

```
aout.d<- aster(resp~varb + fit:(Den), pred, fam, varb, id, root, data=redata)

summary(aout.d, show.graph=T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(Den), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redata)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -39.755770   0.597861 -66.497   <2e-16 ***
## varbfrt      44.287436   0.606131  73.066   <2e-16 ***
## varbfrt.2    29.362530   0.598590  49.053   <2e-16 ***
## varbseeds    42.077712   0.597597  70.412   <2e-16 ***
## fit:DenH      0.041854   0.003121  13.411   <2e-16 ***
## fit:DenL     -0.039658   0.004752  -8.346   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:DenM
```

Do not try to determine the significance `Density` from the `summary` statement. These are not reliable. Instead, perform a liklihood ratio test. The results of the liklihood ratio test show that the model with `Density` explains significantly more variation than the model without.

```
anova(aout, aout.d)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb
## Model 2: resp ~ varb + fit:(Den)
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        4    132606
## 2        6    133112  2   505.61 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Now, add density and $N_e$ (`Gen`) to the model, and perform a liklihood ratio test

```
aout.dg<- aster(resp~varb + fit:(Den + Gen), pred, fam, varb, id, root, data=redata)

summary(aout.dg, show.graph=T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(Den + Gen), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redata)
```

```
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept) -38.436651   0.570184  -67.41   <2e-16 ***
## varbfrt      42.971860   0.578764   74.25   <2e-16 ***
## varbfrt.2    28.072418   0.570263   49.23   <2e-16 ***
## varbseeds    40.775220   0.569845   71.56   <2e-16 ***
## fit:DenH      0.040797   0.003083   13.23   <2e-16 ***
## fit:DenL     -0.046588   0.004664   -9.99   <2e-16 ***
## fit:GenLG    -0.048047   0.003018  -15.92   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:DenM
```

```
anova(aout.d, aout.dg)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb + fit:(Den)
## Model 2: resp ~ varb + fit:(Den + Gen)
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        6    133112
## 2        7    133406  1   294.19 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Include interaction between `Den` & `Gen` and test for significance of interaction with liklihood ratio test.

```
aout.dg2<- aster(resp~varb + fit:(Den + Gen + Den*Gen), pred, fam, varb, id, root, data=redata)

summary(aout.dg2)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(Den + Gen + Den *
##     Gen), pred = pred, fam = fam, varvar = varb, idvar = id,
##     root = root, data = redata)
##
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -39.685580   0.573110 -69.246  < 2e-16 ***
## varbfrt         44.223124   0.581733  76.020  < 2e-16 ***
## varbfrt.2       29.337161   0.574020  51.108  < 2e-16 ***
## varbseeds       42.019372   0.572764  73.362  < 2e-16 ***
## fit:DenH         0.050631   0.003726  13.589  < 2e-16 ***
## fit:DenL        -0.061064   0.005868 -10.407  < 2e-16 ***
## fit:GenLG       -0.076651   0.004327 -17.716  < 2e-16 ***
## fit:DenL:GenLG   0.081842   0.006936  11.799  < 2e-16 ***
```

```
## fit:DenM:GenLG   0.039494   0.006678   5.914 3.34e-09 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:DenM
```

```
anova(aout.dg, aout.dg2)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb + fit:(Den + Gen)
## Model 2: resp ~ varb + fit:(Den + Gen + Den * Gen)
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        7    133406
## 2        9    133542  2   135.58 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Finally, include plotID to model and perform yet anohter liklihood ratio test

```
aoutc<- aster(resp~varb + fit:(plotID + Den + Gen + Den*Gen), pred, fam, varb, id, root, data=redata)

summary(aoutc, show.graph=T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(plotID + Den + Gen +
##     Den * Gen), pred = pred, fam = fam, varvar = varb, idvar = id,
##     root = root, data = redata)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##                Estimate Std. Error z value Pr(>|z|)
## (Intercept)  -37.862132   0.594544 -63.683  < 2e-16 ***
## varbfrt       42.414140   0.602792  70.363  < 2e-16 ***
## varbfrt.2     27.620364   0.594622  46.450  < 2e-16 ***
## varbseeds     40.099155   0.596376  67.238  < 2e-16 ***
## fit:plotID1    0.107193   0.010629  10.085  < 2e-16 ***
## fit:plotID2    0.018053   0.012738   1.417  0.15641
## fit:plotID3   -0.055465   0.018987  -2.921  0.00349 **
## fit:plotID4   -0.028087   0.014469  -1.941  0.05223 .
## fit:plotID5    0.007564   0.012785   0.592  0.55408
## fit:plotID6    0.027366   0.012667   2.160  0.03074 *
## fit:plotID7    0.059572   0.011864   5.021 5.13e-07 ***
## fit:plotID8    0.061736   0.011796   5.233 1.66e-07 ***
## fit:plotID9    0.067894   0.011607   5.849 4.94e-09 ***
## fit:plotID10   0.087198   0.011071   7.876 3.38e-15 ***
## fit:plotID11   0.056926   0.011947   4.765 1.89e-06 ***
## fit:plotID12   0.100938   0.010755   9.385  < 2e-16 ***
```

```
## fit:plotID13   0.180417   0.009807  18.398  < 2e-16 ***
## fit:plotID14   0.076877   0.011345   6.776 1.23e-11 ***
## fit:plotID15   0.033981   0.012566   2.704  0.00685 **
## fit:plotID16   0.014427   0.012752   1.131  0.25793
## fit:plotID17   0.014738   0.012751   1.156  0.24777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:plotID18
##      fit:DenL
##      fit:DenM
##      fit:GenLG
##      fit:DenL:GenLG
##      fit:DenM:GenLG
```

```
anova(aout.dg2, aoutc)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb + fit:(Den + Gen + Den * Gen)
## Model 2: resp ~ varb + fit:(plotID + Den + Gen + Den * Gen)
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        9    133542
## 2       21    134616 12   1074.7 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

That was fun. The results of these aster models (of entire data set) and liklihood ratio tests are presetned in Table 1 of the manuscript. The next steps will produce mean fitness and standard errors for these factors.

## Calculation of Mean Fitness and Standard Errors

As the effects of `GEN` was significant in the above analyses, we can divide the data into high and low $N_e$ data sets, and perform additional analsyis to calcualte mean fitness and standard errors for the density treatmetns. The aster analyses for high and low $N_e$ are performed in parallel below (i.e. each step is performed twice, once for high $N_e$ and once for low $N_e$ analses).

First, isloate high (HG) and low (LG) data from the main redata file, and drop unused levels. Therefore, don't have to do "reshape" data step. `HG`/`hg` = High $N_e$, `LG`/`lg` = Low $N_e$

```
redataHG<- subset(redata, Gen=="HG")
redataLG<- subset(redata, Gen=="LG")

redataHG<- droplevels(redataHG)
redataLG<- droplevels(redataLG)
```

Perform aster analysis on `HG` and `LG` data with just fitness data (no predictors), then add `Den` and perform a liklihood ratio test.

```
aoutHG<- aster(resp~varb, pred, fam, varb, id, root, data=redataHG)
aoutHG2<- aster(resp~varb + fit:(Den), pred, fam, varb, id, root, data=redataHG)

summary(aoutHG, show.graph = T)
```

```
##
## Call:
```

```
## aster.formula(formula = resp ~ varb, pred = pred, fam = fam,
##     varvar = varb, idvar = id, root = root, data = redataHG)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -46.7514     0.8383  -55.77   <2e-16 ***
## varbfrt      51.2764     0.8499   60.33   <2e-16 ***
## varbfrt.2    35.6147     0.8417   42.31   <2e-16 ***
## varbseeds    49.1697     0.8383   58.66   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aoutHG2, show.graph=T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(Den), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataHG)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -37.702589   0.851082  -44.30   <2e-16 ***
## varbfrt      42.242416   0.862010   49.01   <2e-16 ***
## varbfrt.2    26.664567   0.850417   31.36   <2e-16 ***
## varbseeds    40.096837   0.850770   47.13   <2e-16 ***
## fit:DenH      0.048489   0.003650   13.28   <2e-16 ***
## fit:DenL     -0.070015   0.006872  -10.19   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:DenM
```

```
anova(aoutHG, aoutHG2)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb
## Model 2: resp ~ varb + fit:(Den)
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        4     89573
```

```
## 2        6     90187  2   614.18 < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Perform same sequence of analyses on LG dataset

```
aoutLG<- aster(resp~varb, pred, fam, varb, id, root, data=redataLG)
aoutLG2<- aster(resp~varb + fit:(Den), pred, fam, varb, id, root, data=redataLG)

summary(aoutLG, show.graph = T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb, pred = pred, fam = fam,
##     varvar = varb, idvar = id, root = root, data = redataLG)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -42.0953     0.7497  -56.15   <2e-16 ***
## varbfrt      46.6262     0.7638   61.05   <2e-16 ***
## varbfrt.2    32.9264     0.7552   43.60   <2e-16 ***
## varbseeds    44.2799     0.7497   59.06   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
summary(aoutLG2, show.graph=T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(Den), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataLG)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -41.520615   0.776686 -53.459  < 2e-16 ***
## varbfrt      46.051828   0.790274  58.273  < 2e-16 ***
## varbfrt.2    32.355588   0.781521  41.401  < 2e-16 ***
## varbseeds    43.704518   0.776742  56.266  < 2e-16 ***
## fit:DenH      0.010151   0.005359   1.894  0.05822 .
## fit:DenL     -0.015065   0.005258  -2.865  0.00417 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
##
## Original predictor variables dropped (aliased)
##      fit:DenM
```

```
anova(aoutLG, aoutLG2)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb
## Model 2: resp ~ varb + fit:(Den)
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        4     43589
## 2        6     43612  2   23.643 7.345e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The effectds of density (`Den`) was significant in both the high (`HG`) and low (`LG`) effective genetic population size data sets.

First step is to generate MLE of saturated model mean value parameter vector: mu. Again, all step are performed twice, once for high and once for low $N_e$ data. Because we want treatment-level estiamtes of fitness, we generate these estimates from the analyses that included density: `aoutHG` and `aoutLG`

```
pout.HG<- predict(aoutHG, se.fit=TRUE)

pout.LG<- predict(aoutLG, se.fit=TRUE)
```

Make design matrix data.frame of indivudals for each density level (low, med., high), that has a 1 for each element of the matrix. These will eventually be replaced with actual fitness values in later steps.

```
fred.hg <- data.frame( Den=levels(redataHG$Den), flw=1, frt=1,frt.2=1, seeds=1,root = 1)

fred.lg <- data.frame( Den=levels(redataLG$Den), flw=1, frt=1,frt.2=1, seeds=1,root = 1)
```

Reshape the design matrix just as the actual data

```
renewdata.hg <- reshape(fred.hg, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")

renewdata.lg <- reshape(fred.lg, varying = list(vars),
                        direction = "long", timevar = "varb",
                        times = as.factor(vars), v.names = "resp")
```

Make character string from "varb" of renewdata without actual values (i.e., the layers of varb in renewdata), and add it to each renewdata object

```
layer<- gsub("[0-9]", "", as.character(renewdata.hg$varb))


renewdata.hg<- data.frame(renewdata.hg, layer= layer)
renewdata.lg<- data.frame(renewdata.lg, layer= layer)
```

Add "seeds" in new layer column of renewdata as numeric, called fit Note: only need one `fit` object as it is the same for both High and Low $N_e$ data, and add to each renew data file

```
fit<- as.numeric(layer=="seeds")

renewdata.gh<- data.frame(renewdata.hg, fit = fit)
```

```
renewdata.lg<- data.frame(renewdata.lg, fit = fit)
```

Rerun prediction of aster analyses, with the reshaped design matrices

```
pout.hg<- predict(aoutHG2, newdata= renewdata.hg, varvar= varb,
                idvar = id, root = root, se.fit = TRUE)

pout.lg<- predict(aoutLG2, newdata= renewdata.lg, varvar= varb,
                idvar = id, root = root, se.fit = TRUE)
```

Check class of each column in prediction outputs

```
sapply(pout.hg, class)
```

```
##       fit    se.fit  gradient    modmat
## "numeric" "numeric"  "matrix"   "array"
```

```
sapply(pout.lg, class)
```

```
##       fit    se.fit  gradient    modmat
## "numeric" "numeric"  "matrix"   "array"
```

Lengths of fit and se.fit (12) match row number of renewdata (as should be with predict.aster)

```
sapply(pout.hg, length)
```

```
##      fit   se.fit gradient   modmat
##       12       12       72       72
```

```
sapply(pout.lg, length)
```

```
##      fit   se.fit gradient   modmat
##       12       12       72       72
```

Therefore, we can make 12 CIs, one for each of 4 nodes of graphical model, and 3 density treatmetns (4 nodes x 3 treatments =12 estimates).

Put the parameter estimates into a matrix with individuals in rows and nodes in columns

Extract HG resutls, and produce a 3 x 4 matrix (3 density treatments by 4 nodes)

```
nnode<- length(vars)
sally.hg<- matrix(pout.hg$fit, ncol = nnode)
dim(sally.hg)
```

```
## [1] 3 4
```

Name the rows (by `Den` treatments) and columns (as nodes), and view the matrix

```
rownames(sally.hg)<- unique(as.character(renewdata.hg$Den))
colnames(sally.hg)<- unique(as.character(renewdata.hg$varb))

round(sally.hg, 3)
```

```
##     flw    frt  frt.2   seeds
## H 1.000 61.951 36.336 418.028
## L 0.861 34.901  5.659  57.833
## M 1.000 48.425 16.477 180.583
```

Now generate matrix of standard errors, and name rows and columns just as fitness estimates

```
nnode2<- length(vars)
sally2<- matrix(pout.hg$se.fit, ncol = nnode)
dim(sally2)
```

```
## [1] 3 4
```

```
rownames(sally2)<- unique(as.character(renewdata.hg$Den))
colnames(sally2)<- unique(as.character(renewdata.hg$varb))

round(sally2, 3)
```

```
##     flw   frt frt.2  seeds
## H 0.000 1.012 1.253 14.952
## L 0.058 2.412 0.560  5.972
## M 0.000 0.821 0.765  8.874
```

Combine estimates with standard errors for only final node: seeds

```
ests<- sally.hg[,grepl("seeds", colnames(sally.hg))]
se<-  sally2[,grepl("seeds", colnames(sally2))]

HG<- cbind(ests, se)
```

Perform the same steps for `LG` results

```
nnode<- length(vars)
sally.lg<- matrix(pout.lg$fit, ncol = nnode)
dim(sally.lg)
```

```
## [1] 3 4
```

```
rownames(sally.lg)<- unique(as.character(renewdata.lg$Den))
colnames(sally.lg)<- unique(as.character(renewdata.lg$varb))

round(sally.lg, 3)
```

```
##     flw    frt  frt.2   seeds
## H 0.977 45.194 13.701 122.917
## L 0.735 31.994  7.757  67.861
## M 0.927 41.784 11.570 102.750
```

Extract standard errors

```
nnode2<- length(vars)
sally.lg2<- matrix(pout.lg$se.fit, ncol = nnode)
dim(sally.lg2)
```

```
## [1] 3 4
```

```
rownames(sally.lg2)<- unique(as.character(renewdata.lg$Den))
colnames(sally.lg2)<- unique(as.character(renewdata.lg$varb))

round(sally.lg2, 3)
```

```
##     flw   frt frt.2 seeds
## H 0.014 1.188 0.772 7.298
## L 0.068 3.086 0.924 8.279
## M 0.033 1.840 0.825 7.632
```

Combine estimates with standard errors for only final node: seeds

```
ests<- sally.lg[,grepl("seeds", colnames(sally.lg))]
se<-  sally.lg2[,grepl("seeds", colnames(sally.lg2))]

LG<- cbind(ests, se)
```

These are the fitness and standard errors for HG and LG treatments (across densities)

```
HG
```

```
##        ests        se
## H 418.02778 14.951711
## L  57.83334  5.971818
## M 180.58333  8.874330
```

```
LG
```

```
##       ests       se
## H 122.9167 7.298401
## L  67.8611 8.278818
## M 102.7500 7.631958
```

# Comparing Female (seeds set) and Male (seeds sired) Fitness

## Calculate mean seed set for each treat x $N_e$ treatment to "relativize" female fitness

Calculate mean seed set for each individual plot. These values will be used to relativize fitness estimates

```
aggregate(fin$seeds, by=list(fin$plotID), mean)
```

```
##    Group.1          x
## 1        1  250.00000
## 2        2   68.91667
## 3        3   11.08333
## 4        4   25.25000
## 5        5   57.75000
## 6        6   79.33333
## 7        7  123.75000
## 8        8  127.50000
## 9        9  138.91667
## 10      10  183.58333
## 11      11  119.33333
## 12      12  226.41667
## 13      13 1012.75000
## 14      14  157.83333
## 15      15   87.16667
## 16      16   65.00000
## 17      17   65.33333
## 18      18   50.00000
```

Load data with number of seeds sired (male fitness), where sires is the number of seeds sired for each family

```
fin<- read.csv("data/aster.sire.dat.csv")

names(fin)
```

```
##  [1] "ID"        "Treat"     "plotID"    "Den"       "Gen"
```

```
##  [6] "plantID"   "familyID"  "surv"      "flw"       "frt"
## [11] "frt.2"     "seeds"     "rel.seeds" "seed.wt"   "mass.a"
## [16] "mass.b"    "sires"
```

Note that the file `aster.sire.dat.csv` only has data from the High $N_e$ treatment. We are only working with High $N_e$ (2 full-sib individuals from 6 families), as we could not always asign paternity between two full-sibs. So, we assigned paternity to families and not individual plants.

Sum of number of seeds that we successfully assigned paternity to, per density treatment

```
aggregate(fin$sires, by=list(fin$Den), sum)
```

```
##   Group.1   x
## 1       H 175
## 2       L 166
## 3       M 172
```

Make sure class of factor variables for predictors (as was done with female fintess.

```
fin$Den<- as.factor(fin$Den)
fin$Gen<- as.factor(fin$Gen)
fin$plotID<- as.factor(fin$plotID)
fin$plantID<- as.factor(fin$plantID)
fin$familyID<- as.factor(fin$familyID)
```

The following generally follows the steps for performing aster analyses of female fitness (seeds set)

Load aster package

```
library(aster)
```

Set the variables to be included in the graphical model. Notice that these are the same variables: flower number -> total fruit -> subsampled fruit -> seeds set

```
vars<- c( "flw", "frt", "frt.2","seeds")
```

Subset data into High, Medium, and Low density treatments

```
datH<- subset(fin, Den=="H")
datH$familyID<- droplevels(datH$familyID)
datM<- subset(fin, Den=="M")
datM$familyID<- droplevels(datM$familyID)
datL<- subset(fin, Den=="L")
datL$familyID<- droplevels(datL$familyID)
```

Perform the same data reshaping steps as before, reshaping data so that all response variables are located in a single vector in a new data sets called for each density treatment

```
redataH <- reshape(datH, varying = list(vars), direction = "long",timevar = "varb", times = as.factor(va
redataM <- reshape(datM, varying = list(vars), direction = "long",timevar = "varb", times = as.factor(va
redataL <- reshape(datL, varying = list(vars), direction = "long",timevar = "varb", times = as.factor(va
```

Designation of fitness variable, and add to each reshaped data.

```
fit <- grepl("seeds", as.character(redataH$varb))
fit<- as.numeric(fit)

redataH$fit <- fit
redataM$fit <- fit
redataL$fit <- fit
```

Check organization of graphical models

```
with(redataH, sort(unique(as.character(varb)[fit == 0])))
```

```
## [1] "flw"   "frt"   "frt.2"
```

```
with(redataH, sort(unique(as.character(varb)[fit == 1])))
```

```
## [1] "seeds"
```

```
with(redataM, sort(unique(as.character(varb)[fit == 0])))
```

```
## [1] "flw"   "frt"   "frt.2"
```

```
with(redataM, sort(unique(as.character(varb)[fit == 1])))
```

```
## [1] "seeds"
```

```
with(redataL, sort(unique(as.character(varb)[fit == 0])))
```

```
## [1] "flw"   "frt"   "frt.2"
```

```
with(redataL, sort(unique(as.character(varb)[fit == 1])))
```

```
## [1] "seeds"
```

Setting the varible "root" to each redata file differs from the previous aster analyses. Here "root" is given the value 2 here to compliment the male fitness estimates (see section: Male Fitness).

```
redataH<- data.frame(redataH, root=2)
redataM<- data.frame(redataM, root=2)
redataL<- data.frame(redataL, root=2)
```

Set graphical model and family for each node

```
pred<- c(0,1,2,3)
fam<- c(1,2,2,2)
```

Perform aster analyses for three density treatments with `familyID` as fixed-effect

```
aoutH<- aster(resp~varb + fit:(familyID), pred, fam, varb, id, root, data=redataH)
summary(aoutH, show.graph = T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(familyID), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataH)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -36.086858   1.627818 -22.169  < 2e-16 ***
## varbfrt         40.633965   1.643658  24.722  < 2e-16 ***
## varbfrt.2       24.552921   1.623894  15.120  < 2e-16 ***
## varbseeds       38.436899   1.632504  23.545  < 2e-16 ***
```

```
## fit:familyID1    0.069576   0.021249   3.274  0.00106 **
## fit:familyID2    0.153190   0.018685   8.199 2.43e-16 ***
## fit:familyID3    0.119853   0.020460   5.858 4.69e-09 ***
## fit:familyID4   -0.024099   0.033573  -0.718  0.47287
## fit:familyID6   -0.007988   0.030665  -0.260  0.79448
## fit:familyID7    0.138946   0.019048   7.295 2.99e-13 ***
## fit:familyID8    0.180643   0.018792   9.613  < 2e-16 ***
## fit:familyID9    0.113480   0.019597   5.791 7.01e-09 ***
## fit:familyID11   0.154918   0.018671   8.297  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID14
```

```
aoutM<- aster(resp~varb + fit:(familyID), pred, fam, varb, id, root, data=redataM)
summary(aoutM, show.graph = T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(familyID), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataM)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -46.43896    2.02107 -22.977  < 2e-16 ***
## varbfrt         51.25262    2.03479  25.188  < 2e-16 ***
## varbfrt.2       36.53140    2.01786  18.104  < 2e-16 ***
## varbseeds       48.50804    2.05645  23.588  < 2e-16 ***
## fit:familyID1    0.26137    0.06701   3.901  9.6e-05 ***
## fit:familyID2    0.17115    0.06764   2.530 0.011403 *
## fit:familyID4    0.09941    0.07018   1.416 0.156645
## fit:familyID6    0.13194    0.07065   1.868 0.061821 .
## fit:familyID7    0.12400    0.06847   1.811 0.070133 .
## fit:familyID9    0.16082    0.06825   2.356 0.018452 *
## fit:familyID11   0.21592    0.06723   3.212 0.001320 **
## fit:familyID14   0.22927    0.06731   3.406 0.000659 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID15
```

```
aoutL<- aster(resp~varb + fit:(familyID), pred, fam, varb, id, root, data=redataL)
summary(aoutL, show.graph = T)
```

```
##
## Call:
```

```
## aster.formula(formula = resp ~ varb + fit:(familyID), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataL)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  frt.2    frt         poisson
##  seeds    frt.2       poisson
##
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -25.517173   1.105432 -23.083  < 2e-16 ***
## varbfrt         29.740008   1.133509  26.237  < 2e-16 ***
## varbfrt.2       14.839584   1.121753  13.229  < 2e-16 ***
## varbseeds       27.756878   1.109365  25.020  < 2e-16 ***
## fit:familyID1    0.043765   0.027776   1.576    0.115
## fit:familyID2    0.106163   0.025622   4.143 3.42e-05 ***
## fit:familyID4   -0.003372   0.044194  -0.076    0.939
## fit:familyID6    0.015868   0.039996   0.397    0.692
## fit:familyID7    0.143954   0.025101   5.735 9.75e-09 ***
## fit:familyID9    0.028787   0.031055   0.927    0.354
## fit:familyID11   0.006698   0.029933   0.224    0.823
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID14
```

## Generate Fitness Estimates for Each Genetic Family (familyID) in Each Density Treatment

High Density Treatment Fitness Estimate

These steps follow those that produced mean fitness estimates for seed set

```
pout<- predict.aster(aoutH, se.fit=TRUE)

#make design matrix
fred <- data.frame(familyID=levels(redataH$familyID), flw=1, frt=1, frt.2=1, seeds=1,root = 2)

#reshape the design matrix just as the actual data
renewdata <- reshape(fred, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")

#make character string from "varb" of renewdata,
#without actual values (e.g., the layers of varb in renewdata)
layer<- gsub("[0-9]", "", as.character(renewdata$varb))

#add layer to renewdata
renewdata<- data.frame(renewdata, layer= layer)

#add fit (seeds) in new layer col of renewdata as numeric, called fit
fit<- as.numeric(layer=="seeds")
```

```
#add fit to renewdata
renewdata<- data.frame(renewdata, fit = fit)

#rerun prediction of aout, with "made up" renewdata
pout<- predict(aoutH, newdata= renewdata, varvar= varb,
               idvar = id, root = root, se.fit = TRUE)

sapply(pout, class)
```

```
##       fit    se.fit  gradient    modmat
## "numeric" "numeric"  "matrix"   "array"
```

```
sapply(pout, length)
```

```
##      fit   se.fit gradient   modmat
##       40       40      520      520
```

```
#put the parameter estimates into a matrix with individuals in rows
#and nodes along columns
nnode<- length(vars)
sally<- matrix(pout$fit, ncol = nnode)
dim(sally)# makes 10 x 4 matrix: 10 families by 4 nodes
```

```
## [1] 10  4
```

```
#name the rows (by familyID) and columns (as nodes)
rownames(sally)<- unique(as.character(renewdata$familyID))
colnames(sally)<- unique(as.character(renewdata$varb))

#view matrix
round(sally, 3)
```

```
##      flw     frt   frt.2    seeds
## 1  2.000  91.349  25.086  282.000
## 2  2.000 144.347 105.657 1291.333
## 3  2.000 113.357  55.578  657.000
## 4  1.570  60.269   6.057   62.000
## 6  1.762  68.853   8.171   85.000
## 7  2.000 128.513  79.135  953.500
## 8  2.000 194.184 199.728 2509.000
## 9  2.000 109.407  49.761  584.500
## 11 2.000 146.621 109.613 1342.000
## 14 1.834  72.448   9.346   98.000
```

```
#use just totalseeds as predicted (expected) fitneses
herman<- sally[,grepl("seeds", colnames(sally))]

#Generate Standard Errors for these estimates

nFam<- nrow(fred)
nnode<- length(vars)
amat<- array(0, c(nFam, nnode, nFam))
dim(amat)# makes an 10 x 4 x 10 matrix
```

```
## [1] 10  4 10
```

```
foo<- grepl("seeds", vars)
for(k in 1:nFam)
  amat[k, foo, k]<- 1

#use aout object, with renewdata, and amat format
pout.amat<- predict(aoutH, newdata= renewdata, varvar= varb,
                    idvar= id, root = root, se.fit=TRUE, amat = amat)

#pout.amat$fit should be the same as file "herman"
herman
```

```
##        1        2        3        4        6        7        8        9
##  282.000 1291.333  657.000   62.000   85.000  953.500 2509.000  584.500
##       11       14
## 1342.000   98.000
```

```
pout.amat$fit #they are the same.  Good.
```

```
##  [1]  282.000 1291.333  657.000   62.000   85.000  953.500 2509.000
##  [8]  584.500 1342.000   98.000
```

```
#combine std.err with estimates, and then round
#to three decimal places
foo<- cbind(pout.amat$fit, pout.amat$se.fit)
rownames(foo)<- as.character(fred$familyID)
colnames(foo)<- c("High Den Fitness", "SE")
round(foo, 3)
```

```
##    High Den Fitness      SE
## 1           282.000  46.530
## 2          1291.333  97.684
## 3           657.000 110.596
## 4            62.000  35.868
## 6            85.000  39.537
## 7           953.500  98.785
## 8          2509.000 257.770
## 9           584.500  72.707
## 11         1342.000 100.092
## 14           98.000  29.081
```

```
H_estimates<- round(foo,3)

H_estimates
```

```
##    High Den Fitness      SE
## 1           282.000  46.530
## 2          1291.333  97.684
## 3           657.000 110.596
## 4            62.000  35.868
## 6            85.000  39.537
## 7           953.500  98.785
## 8          2509.000 257.770
## 9           584.500  72.707
## 11         1342.000 100.092
## 14           98.000  29.081
```

Medium Density Fitness Estiamtes

```
#Generate MLE of saturated model mean value parameter vector: mu
pout<- predict.aster(aoutM, se.fit=TRUE)


#make design matrix
fred <- data.frame(familyID=levels(redataM$familyID), flw=1, frt=1, frt.2=1, seeds=1,root = 2)

#reshape the "made up data" just as the actual data
renewdata <- reshape(fred, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")

#make character string from "varb" of renewdata,
#without actual values (e.g., the layers of varb in renewdata)
layer<- gsub("[0-9]", "", as.character(renewdata$varb))

#add layer to renewdata
renewdata<- data.frame(renewdata, layer= layer)

#seed seed.ct in new layer col of renewdata as numeric, called fit
fit<- as.numeric(layer=="seeds")

#add fit to renewdata
renewdata<- data.frame(renewdata, fit = fit)

#rerun prediction of aout, with "made up" renewdata
pout<- predict(aoutM, newdata= renewdata, varvar= varb,
               idvar = id, root = root, se.fit = TRUE)

sapply(pout, class)
```

```
##       fit    se.fit  gradient    modmat
## "numeric" "numeric"  "matrix"   "array"
```

```
sapply(pout, length)
```

```
##      fit   se.fit gradient   modmat
##       36       36      432      432
```

```
#put the parameter estimates into a matrix with familyID in rows
#and nodes along columns
nnode<- length(vars)
sally<- matrix(pout$fit, ncol = nnode)
dim(sally)# makes 9 x 4 matrix: 9 families by 4 nodes
```

```
## [1] 9 4
```

```
#name the rows (by Den Treat) and columns (as nodes)
rownames(sally)<- unique(as.character(renewdata$familyID))
colnames(sally)<- unique(as.character(renewdata$varb))

#view matrix
round(sally, 3)
```

```
##      flw     frt  frt.2   seeds
## 1  2.000 154.794 82.858 852.000
```

```
## 2  2.000 112.973 24.906 234.000
## 4  1.936  98.406 11.321  99.000
## 6  1.991 105.206 16.161 146.000
## 7  1.984 103.752 14.839 133.000
## 9  1.999 110.676 22.153 206.000
## 11 2.000 127.203 43.118 423.667
## 14 2.000 133.439 51.618 514.000
## 15 1.107  52.743  2.652  21.000
```

```
#use just totalseeds as predicted (expected) fitneses
herman<- sally[,grepl("seeds", colnames(sally))]

#Generate Standard Errors for these estimates

nFam<- nrow(fred)
nnode<- length(vars)
amat<- array(0, c(nFam, nnode, nFam))
dim(amat)# makes an 9 x 4 x 9 matrix
```

```
## [1] 9 4 9
```

```
foo<- grepl("seeds", vars)
for(k in 1:nFam)
  amat[k, foo, k]<- 1

#use aout object, with renewdata, and amat format
pout.amat<- predict(aoutM, newdata= renewdata, varvar= varb,
                    idvar= id, root = root, se.fit=TRUE, amat = amat)

#pout.amat$fit should be the same as file "herman"
herman
```

```
##        1        2        4        6        7        9       11       14
## 852.0000 234.0000  99.0000 146.0000 133.0000 206.0000 423.6667 514.0000
##       15
##  21.0000
```

```
pout.amat$fit #they are the same.  Good.
```

```
## [1] 852.0000 234.0000  99.0000 146.0000 133.0000 206.0000 423.6667 514.0000
## [9]  21.0000
```

```
#combine std.err with estimates, and then round
#to three decimal places
foo<- cbind(pout.amat$fit, pout.amat$se.fit)
rownames(foo)<- as.character(fred$familyID)
colnames(foo)<- c("Med Den Fitness", "SE")
round(foo, 3)
```

```
##    Med Den Fitness     SE
## 1          852.000 69.047
## 2          234.000 31.201
## 4           99.000 35.036
## 6          146.000 41.421
## 7          133.000 27.974
## 9          206.000 35.445
## 11         423.667 44.712
```

```
## 14         514.000 61.694
## 15          21.000 19.328
```

```
M_estimates<- round(foo, 3)

M_estimates
```

```
##    Med Den Fitness     SE
## 1          852.000 69.047
## 2          234.000 31.201
## 4           99.000 35.036
## 6          146.000 41.421
## 7          133.000 27.974
## 9          206.000 35.445
## 11         423.667 44.712
## 14         514.000 61.694
## 15          21.000 19.328
```

Low Densitiy Estimates

```
#generate MLE of saturated model mean value parameter vector: mu
pout<- predict.aster(aoutL, se.fit=TRUE)


#make data.frame of indivudals for each block (1-8)
fred <- data.frame(familyID=levels(redataL$familyID), flw=1, frt=1, frt.2=1, seeds=1,root = 2)

#reshape the "made up data" just as the actual data
renewdata <- reshape(fred, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")

#make character string from "varb" of renewdata,
#without actual values (e.g., the layers of varb in renewdata)
layer<- gsub("[0-9]", "", as.character(renewdata$varb))

#add layer to renewdata
renewdata<- data.frame(renewdata, layer= layer)

#seed seed.ct in new layer col of renewdata as numeric, called fit
fit<- as.numeric(layer=="seeds")

#add fit to renewdata
renewdata<- data.frame(renewdata, fit = fit)

#rerun prediction of aout, with "made up" renewdata
pout<- predict(aoutM, newdata= renewdata, varvar= varb,
               idvar = id, root = root, se.fit = TRUE)

sapply(pout, class)
```

```
##       fit    se.fit  gradient    modmat
## "numeric" "numeric"  "matrix"   "array"
```

```
sapply(pout, length)
```

```
##      fit   se.fit gradient   modmat
```

```
##       32       32      384      384
```

```
#put the parameter estimates into a matrix with familyID in rows
#and nodes along columns
nnode<- length(vars)
sally<- matrix(pout$fit, ncol = nnode)
dim(sally)# makes 8 x 4 matrix: 8 families by 4 nodes
```

```
## [1] 8 4
```

```
#name the rows (by Den Treat) and columns (as nodes)
rownames(sally)<- unique(as.character(renewdata$familyID))
colnames(sally)<- unique(as.character(renewdata$varb))

#view matrix
round(sally, 3)
```

```
##      flw     frt  frt.2   seeds
## 1  2.000 154.794 82.858 852.000
## 2  2.000 112.973 24.906 234.000
## 4  1.936  98.406 11.321  99.000
## 6  1.991 105.206 16.161 146.000
## 7  1.984 103.752 14.839 133.000
## 9  1.999 110.676 22.153 206.000
## 11 2.000 127.203 43.118 423.667
## 14 2.000 133.439 51.618 514.000
```

```
#use just totalseeds as predicted (expected) fitneses
herman<- sally[,grepl("seeds", colnames(sally))]

#Generate Standard Errors for these estimates

nFam<- nrow(fred)
nnode<- length(vars)
amat<- array(0, c(nFam, nnode, nFam))
dim(amat)# makes an 8 x 4 x 8 matrix
```

```
## [1] 8 4 8
```

```
foo<- grepl("seeds", vars)
for(k in 1:nFam)
  amat[k, foo, k]<- 1

#use aout object, with renewdata, and amat format
pout.amat<- predict(aoutL, newdata= renewdata, varvar= varb,
                    idvar= id, root = root, se.fit=TRUE, amat = amat)

#pout.amat$fit should be the same as file "herman"
herman
```

```
##        1        2        4        6        7        9       11       14
## 852.0000 234.0000  99.0000 146.0000 133.0000 206.0000 423.6667 514.0000
```

```
pout.amat$fit #they are the same.  Good.
```

```
## [1]  83.66667 203.66667  39.00000  54.00000 365.50000  66.50000  46.33333
## [8]  41.33333
```

```
#combine std.err with estimates, and then round
#to three decimal places
foo<- cbind(pout.amat$fit, pout.amat$se.fit)
rownames(foo)<- as.character(fred$familyID)
colnames(foo)<- c("Low Den Fitness", "SE")
round(foo, 3)
```

```
##    Low Den Fitness     SE
## 1           83.667 20.403
## 2          203.667 31.447
## 4           39.000 25.965
## 6           54.000 29.824
## 7          365.500 55.011
## 9           66.500 22.886
## 11          46.333 16.162
## 14          41.333 15.383
```

```
L_estimates<- round(foo, 3)

L_estimates
```

```
##    Low Den Fitness     SE
## 1           83.667 20.403
## 2          203.667 31.447
## 4           39.000 25.965
## 6           54.000 29.824
## 7          365.500 55.011
## 9           66.500 22.886
## 11          46.333 16.162
## 14          41.333 15.383
```

The files present the mean fitness and standard error for each genetic family, in each density treatments under high $N_e$ conditions

```
H_estimates
```

```
##    High Den Fitness      SE
## 1            282.000  46.530
## 2           1291.333  97.684
## 3            657.000 110.596
## 4             62.000  35.868
## 6             85.000  39.537
## 7            953.500  98.785
## 8           2509.000 257.770
## 9            584.500  72.707
## 11          1342.000 100.092
## 14            98.000  29.081
```

```
M_estimates
```

```
##    Med Den Fitness     SE
## 1           852.000 69.047
## 2           234.000 31.201
## 4            99.000 35.036
## 6           146.000 41.421
## 7           133.000 27.974
## 9           206.000 35.445
```

```
## 11         423.667 44.712
## 14         514.000 61.694
## 15          21.000 19.328
```

```
L_estimates
```

```
##    Low Den Fitness     SE
## 1           83.667 20.403
## 2          203.667 31.447
## 4           39.000 25.965
## 6           54.000 29.824
## 7          365.500 55.011
## 9           66.500 22.886
## 11          46.333 16.162
## 14          41.333 15.383
```

## Estimate Male (Seeds Sired) Fitness

This section performs aster analyses with the number of seeds sired as the terminal fitness node, to obtain mean fitness estiames through male reproductive function. Because these analyses use the same data files as above (for relativized female fitness estimates), we can begin the process of setting up the aster analyses at designating the graphical model variables.

The graphical model for male fitness: flower number -> fruit number -> number of seed sired

```
vars<- c( "flw", "frt", "sires")
```

Reshape data, as with female fitness, so that all response variables are located in a single vector in a new data

```
redata <- reshape(fin, varying = list(vars), direction = "long",timevar = "varb", times = as.factor(vars
```

Designation of number of seeds sired ("sires") as fitness variable

```
fit <- grepl("sires", as.character(redata$varb))
fit<- as.numeric(fit)

redata$fit <- fit

with(redata, sort(unique(as.character(varb)[fit == 0])))
```

```
## [1] "flw" "frt"
```

```
with(redata, sort(unique(as.character(varb)[fit == 1])))
```

```
## [1] "sires"
```

Add a variable "root" to redata, where value is 2

```
redata<- data.frame(redata, root=2)
```

Set graphical model and statistical family for each node

```
pred<- c(0,1,2)
fam<- c(1,2,2)

sapply(fam.default(), as.character)[fam]
```

```
## [1] "bernoulli" "poisson"   "poisson"
```

### Main Aster Analysis of Male Fitness

First analysis with only fitness data, then add `familyID`, and finally density treatment. Perform liklihood ratio test to assess significance of `familyID` and `Den`

```
aout<- aster(resp~varb, pred, fam, varb, id, root, data=redata)

summary(aout)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb, pred = pred, fam = fam,
##     varvar = varb, idvar = id, root = root, data = redata)
##
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -47.4511     0.8242  -57.57   <2e-16 ***
## varbfrt      52.2904     0.8361   62.54   <2e-16 ***
## varbsires    45.1290     0.8254   54.68   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
aout2<- aster(resp~varb +fit:familyID, pred, fam, varb, id, root, data=redata)

summary(aout2)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:familyID, pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redata)
##
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -47.24221    0.83073 -56.869   <2e-16 ***
## varbfrt         52.08160    0.84256  61.813   <2e-16 ***
## varbsires       44.87191    0.87952  51.018   <2e-16 ***
## fit:familyID1    0.10745    0.30144   0.356    0.721
## fit:familyID2    0.16941    0.29912   0.566    0.571
## fit:familyID3   -0.38042    0.44555  -0.854    0.393
## fit:familyID4   -0.17411    0.33594  -0.518    0.604
## fit:familyID6   -0.11335    0.33341  -0.340    0.734
## fit:familyID7   -0.08407    0.30978  -0.271    0.786
## fit:familyID8    0.07902    0.39767   0.199    0.842
## fit:familyID9   -0.08407    0.30978  -0.271    0.786
## fit:familyID11   0.18515    0.29897   0.619    0.536
## fit:familyID14  -0.01169    0.30524  -0.038    0.969
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID15
```

Effect of `familyID` is not significant

```
anova(aout, aout2)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb
```

```
## Model 2: resp ~ varb + fit:familyID
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        3     27276
## 2       13     27288 10   11.541     0.317
```

Aster analsyis with Den

```
aout3<- aster(resp~varb +fit:Den, pred, fam, varb, id, root, data=redata)


summary(aout3)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:Den, pred = pred, fam = fam,
##     varvar = varb, idvar = id, root = root, data = redata)
##
##              Estimate Std. Error z value Pr(>|z|)
## (Intercept) -47.44789    0.82427 -57.563   <2e-16 ***
## varbfrt      52.28721    0.83621  62.529   <2e-16 ***
## varbsires    45.12992    0.82696  54.573   <2e-16 ***
## fit:DenH      0.01265    0.09183   0.138    0.890
## fit:DenL     -0.02564    0.09247  -0.277    0.782
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:DenM
```

Effect of Den is not significant

```
anova(aout, aout3)
```

```
## Analysis of Deviance Table
##
## Model 1: resp ~ varb
## Model 2: resp ~ varb + fit:Den
##   Model Df Model Dev Df Deviance P(>|Chi|)
## 1        3     27276
## 2        5     27276  2  0.17861    0.9146
```

## Family-specific Male Fitness Estimates

To compare with family-specific estimates of female fitness, we now generate family-specific male fitness estimates.

Load the data with male fitess (seeds sired), make sure all predictors are factors, and set the graphical model: flower number -> fruit number -> number of seed sired

```
fin<- read.csv("C:data/aster.sire.dat.csv")

fin$Den<- as.factor(fin$Den)
fin$Gen<- as.factor(fin$Gen)
fin$plotID<- as.factor(fin$plotID)
fin$plantID<- as.factor(fin$plantID)
fin$familyID<- as.factor(fin$familyID)

vars<- c( "flw", "frt", "sires")
```

Reshape data so that all response variables are located in a single vector in a new data, and add `fit` and `root` to redata.

```
redata <- reshape(fin, varying = list(vars), direction = "long",timevar = "varb", times = as.factor(vars


fit <- grepl("sires", as.character(redata$varb))
fit<- as.numeric(fit)

redata$fit <- fit

with(redata, sort(unique(as.character(varb)[fit == 0])))
```

```
## [1] "flw" "frt"
```

```
with(redata, sort(unique(as.character(varb)[fit == 1])))
```

```
## [1] "sires"
```

```
redata<- data.frame(redata, root=2)
```

Set graph. model and family for each node

```
pred<- c(0,1,2)
fam<- c(1,2,2)
```

Split reshaped data file into density-specific files for individual aster analsyses. Drop unused levels

```
redataL<-subset(redata, Den=="L")

redataL$familyID<- droplevels(redataL$familyID)

redataM<- subset(redata, Den=="M")

redataM$familyID<- droplevels(redataM$familyID)

redataH<- subset(redata, Den=="H")

redataH$familyID<- droplevels(redataH$familyID)
```

### Low Density Fitness Estiamte for Male Fitness

Aster analysis for low density treatment

```
aoutL<- aster(resp~varb + fit:(familyID), pred, fam, varb, id, root, data=redataL)

summary(aoutL, show.graph = T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(familyID), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataL)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
```

```
##  frt      flw         poisson
##  sires    frt         poisson
##
##                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -27.41995    1.12316 -24.413   <2e-16 ***
## varbfrt         31.67372    1.15114  27.515   <2e-16 ***
## varbsires       25.64046    1.13287  22.633   <2e-16 ***
## fit:familyID1    0.04207    0.20522   0.205    0.838
## fit:familyID2    0.12271    0.20300   0.604    0.546
## fit:familyID4   -0.32100    0.33892  -0.947    0.344
## fit:familyID6   -0.23585    0.32108  -0.735    0.463
## fit:familyID7   -0.05446    0.23407  -0.233    0.816
## fit:familyID9   -0.08830    0.23633  -0.374    0.709
## fit:familyID11   0.04207    0.20522   0.205    0.838
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID14
```

Generate MLE of saturated model mean value parameter vector: mu

```
pout<- predict.aster(aoutL, se.fit=TRUE)
```

Make a design matrix and reshape it just like the origional data

```
fred <- data.frame(familyID=levels(redataL$familyID), flw=1, frt=1, sires=1,root = 2)

renewdata <- reshape(fred, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")
```

Ad `varb` and `fit` objects to reshaped design matrix

```
layer<- gsub("[0-9]", "", as.character(renewdata$varb))

renewdata<- data.frame(renewdata, layer= layer)

fit<- as.numeric(layer=="sires")

renewdata<- data.frame(renewdata, fit = fit)
```

Generate predicted values from aster analsyis (low density data_, with "made up" design matrix

```
pout<- predict(aoutL, newdata= renewdata, varvar= varb,
               idvar = id, root = root, se.fit = TRUE)
```

Put the parameter estimates into object `ests` and standard errors into object `se`, that are both matrices with individuals in rows and nodes along columns (makes 8 x 3 matrix: 8 indiv by 3 nodes).

```
nnode<- length(vars)
ests<- matrix(pout$fit, ncol = nnode)
se<- matrix(pout$se.fit, ncol= nnode)
```

Name the rows (as familID) and columns (as nodes)

```
rownames(ests)<- unique(as.character(renewdata$familyID))
colnames(ests)<- unique(as.character(renewdata$varb))
```

```
rownames(se)<- unique(as.character(renewdata$familyID))
colnames(se)<- unique(as.character(renewdata$varb))

round(ests, 3)
```

```
##      flw    frt  sires
## 1  1.841 56.827 10.000
## 2  1.896 59.414 11.333
## 4  1.396 40.851  5.000
## 6  1.522 45.020  6.000
## 7  1.751 53.198  8.500
## 9  1.714 51.791  8.000
## 11 1.841 56.827 10.000
## 14 1.805 55.317  9.333
```

```
round(se, 3)
```

```
##      flw    frt sires
## 1  0.135  5.572 2.315
## 2  0.097  4.707 2.380
## 4  0.477 15.323 3.319
## 6  0.427 14.211 3.529
## 7  0.215  7.980 2.737
## 9  0.236  8.472 2.699
## 11 0.135  5.572 2.315
## 14 0.156  6.070 2.281
```

Use just seeds sired as predicted fitneses estiamte

```
L_ests<- ests[,grepl("sires", colnames(ests))]
L_se<- se[,grepl("sires", colnames(se))]
```

Combine estimates and standard errors into single object

```
L_fit<- cbind(L_ests, L_se)

round(L_fit, 3)
```

```
##    L_ests  L_se
## 1  10.000 2.315
## 2  11.333 2.380
## 4   5.000 3.319
## 6   6.000 3.529
## 7   8.500 2.737
## 9   8.000 2.699
## 11 10.000 2.315
## 14  9.333 2.281
```

### Medium Density Fitness Estiamte for Male Fitness

Aster analysis for medium density treatment

```
aoutM<- aster(resp~varb + fit:(familyID), pred, fam, varb, id, root, data=redataM)

summary(aoutM, show.graph = T)
```

```
##
## Call:
```

```
## aster.formula(formula = resp ~ varb + fit:(familyID), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataM)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  sires    frt         poisson
##
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.613e+01  1.681e+00 -33.392   <2e-16 ***
## varbfrt         6.116e+01  1.698e+00  36.020   <2e-16 ***
## varbsires       5.356e+01  1.708e+00  31.350   <2e-16 ***
## fit:familyID1   2.953e-01  3.416e-01   0.864    0.387
## fit:familyID2  -9.739e-02  3.567e-01  -0.273    0.785
## fit:familyID4   1.008e-15  4.317e-01   0.000    1.000
## fit:familyID6   7.945e-16  4.317e-01   0.000    1.000
## fit:familyID7  -3.227e-01  3.975e-01  -0.812    0.417
## fit:familyID9  -4.762e-02  3.767e-01  -0.126    0.899
## fit:familyID11  2.230e-01  3.442e-01   0.648    0.517
## fit:familyID14  4.561e-02  3.713e-01   0.123    0.902
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID15
```

Generate MLE of saturated model mean value parameter vector: mu

```
pout<- predict.aster(aoutM, se.fit=TRUE)
```

Make a design matrix and reshape it just like the origional data

```
fred <- data.frame(familyID=levels(redataL$familyID), flw=1, frt=1, sires=1,root = 2)

renewdata <- reshape(fred, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")
```

Ad `varb` and `fit` objects to reshaped design matrix

```
layer<- gsub("[0-9]", "", as.character(renewdata$varb))

renewdata<- data.frame(renewdata, layer= layer)

fit<- as.numeric(layer=="sires")

renewdata<- data.frame(renewdata, fit = fit)
```

Generate predicted values from aster analsyis (low density data_, with "made up" design matrix

```
pout<- predict(aoutM, newdata= renewdata, varvar= varb,
               idvar = id, root = root, se.fit = TRUE)
```

Put the parameter estimates into object **ests** and standard errors into object **se**, that are both matrices with individuals in rows and nodes along columns (makes 8 x 3 matrix: 8 indiv by 3 nodes).

```
nnode<- length(vars)
ests<- matrix(pout$fit, ncol = nnode)
se<- matrix(pout$se.fit, ncol= nnode)
```

Name the rows (as familID) and columns (as nodes)

```
rownames(ests)<- unique(as.character(renewdata$familyID))
colnames(ests)<- unique(as.character(renewdata$varb))

rownames(se)<- unique(as.character(renewdata$familyID))
colnames(se)<- unique(as.character(renewdata$varb))

round(ests, 3)
```

```
##      flw     frt  sires
## 1  1.990 124.143 12.667
## 2  1.924 116.114  8.000
## 4  1.950 118.506  9.000
## 6  1.950 118.506  9.000
## 7  1.833 109.096  6.000
## 9  1.938 117.381  8.500
## 11 1.984 122.910 11.667
## 14 1.959 119.513  9.500
```

```
round(se, 3)
```

```
##      flw    frt sires
## 1  0.015  3.649 2.189
## 2  0.090  6.743 1.809
## 4  0.085  7.888 3.276
## 6  0.085  7.888 3.276
## 7  0.184 11.970 1.997
## 9  0.083  6.806 2.267
## 11 0.022  3.861 2.107
## 14 0.058  5.635 2.366
```

Use just seeds sired as predicted fitneses estiamte

```
M_ests<- ests[,grepl("sires", colnames(ests))]
M_se<- se[,grepl("sires", colnames(se))]
```

Combine estimates and standard errors into single object

```
M_fit<- cbind(M_ests, M_se)

round(M_fit, 3)
```

```
##    M_ests  M_se
## 1  12.667 2.189
## 2   8.000 1.809
## 4   9.000 3.276
## 6   9.000 3.276
## 7   6.000 1.997
## 9   8.500 2.267
## 11 11.667 2.107
## 14  9.500 2.366
```

### High Density Fitness Estiamte for Male Fitness

Aster analysis for high density treatment

```
aoutH<- aster(resp~varb + fit:(familyID), pred, fam, varb, id, root, data=redataH)

summary(aoutH, show.graph = T)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(familyID), pred = pred,
##     fam = fam, varvar = varb, idvar = id, root = root, data = redataH)
##
##
## Graphical Model:
##  variable predecessor family
##  flw      root        bernoulli
##  frt      flw         poisson
##  sires    frt         poisson
##
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)    -5.520e+01  1.677e+00 -32.914   <2e-16 ***
## varbfrt         6.022e+01  1.694e+00  35.544   <2e-16 ***
## varbsires       5.248e+01  1.702e+00  30.838   <2e-16 ***
## fit:familyID1  -1.246e-15  3.301e-01   0.000   1.0000
## fit:familyID2   5.592e-01  2.761e-01   2.025   0.0428 *
## fit:familyID3  -3.155e-01  4.533e-01  -0.696   0.4864
## fit:familyID4  -5.592e-02  4.115e-01  -0.136   0.8919
## fit:familyID6   5.315e-02  3.978e-01   0.134   0.8937
## fit:familyID7   1.983e-01  3.163e-01   0.627   0.5307
## fit:familyID8   2.426e-01  3.763e-01   0.645   0.5192
## fit:familyID9  -1.133e-15  3.301e-01   0.000   1.0000
## fit:familyID11  4.498e-01  2.807e-01   1.603   0.1090
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:familyID14
```

Generate MLE of saturated model mean value parameter vector: mu

```
pout<- predict.aster(aoutH, se.fit=TRUE)
```

Make a design matrix and reshape it just like the origional data

```
fred <- data.frame(familyID=levels(redataL$familyID), flw=1, frt=1, sires=1,root = 2)

renewdata <- reshape(fred, varying = list(vars),
                     direction = "long", timevar = "varb",
                     times = as.factor(vars), v.names = "resp")
```

Ad `varb` and `fit` objects to reshaped design matrix

```
layer<- gsub("[0-9]", "", as.character(renewdata$varb))

renewdata<- data.frame(renewdata, layer= layer)
```

```
fit<- as.numeric(layer=="sires")

renewdata<- data.frame(renewdata, fit = fit)
```

Generate predicted values from aster analsyis (low density data_, with "made up" design matrix

```
pout<- predict(aoutH, newdata= renewdata, varvar= varb,
               idvar = id, root = root, se.fit = TRUE)
```

Put the parameter estimates into object **ests** and standard errors into object **se**, that are both matrices with individuals in rows and nodes along columns (makes 8 x 3 matrix: 8 indiv by 3 nodes).

```
nnode<- length(vars)
ests<- matrix(pout$fit, ncol = nnode)
se<- matrix(pout$se.fit, ncol= nnode)
```

Name the rows (as familID) and columns (as nodes)

```
rownames(ests)<- unique(as.character(renewdata$familyID))
colnames(ests)<- unique(as.character(renewdata$varb))

rownames(se)<- unique(as.character(renewdata$familyID))
colnames(se)<- unique(as.character(renewdata$varb))

round(ests, 3)
```

```
##      flw     frt  sires
## 1  1.919 113.940  7.500
## 2  1.996 124.477 14.333
## 4  1.901 112.460  7.000
## 6  1.934 115.245  8.000
## 7  1.965 118.361  9.500
## 9  1.919 113.940  7.500
## 11 1.991 122.720 12.667
## 14 1.919 113.940  7.500
```

```
round(se, 3)
```

```
##      flw    frt sires
## 1  0.103  7.649 2.142
## 2  0.006  3.443 2.324
## 4  0.147 10.809 2.951
## 6  0.105  8.724 3.105
## 7  0.050  5.210 2.352
## 9  0.103  7.649 2.142
## 11 0.013  3.554 2.185
## 14 0.103  7.649 2.142
```

Use just seeds sired as predicted fitneses estiamte

```
H_ests<- ests[,grepl("sires", colnames(ests))]
H_se<- se[,grepl("sires", colnames(se))]
```

Combine estimates and standard errors into single object

```
H_fit<- cbind(H_ests, H_se)

round(H_fit, 3)
```

```
##    H_ests  H_se
## 1   7.500 2.142
## 2  14.333 2.324
## 4   7.000 2.951
## 6   8.000 3.105
## 7   9.500 2.352
## 9   7.500 2.142
## 11 12.667 2.185
## 14  7.500 2.142
```

# Above and Below Ground Biomass

## Preliminary and Summary of Data

The following code summarizes and performs linear analyses of biomass accumulation (above and below ground). The required data is in the same file as the female fitness data.

Reload data and load package `emmeans`

```
fin<- read.csv("C:data/aster.dat.csv")

library(emmeans)
```

```
## Warning: package 'emmeans' was built under R version 3.5.3
```

Subset the data into High and Low $N_e$ data

```
hi<- subset(fin, Gen=="HG")
lo<- subset(fin, Gen=="LG")
```

For summary purposes, define a function that will calculate standard errors

```
stderr<- function(x) sd(x)/sqrt(length(x))
```

### Generate Summary Statistics (mean & standard error) for above and below biomass, in High and Low $N_e$ plants

High $N_e$

Above ground biomass

```
aggregate(hi$mass.a, by=list(hi$Den), mean)
```

```
##   Group.1        x
## 1       H 12.65000
## 2       L  4.20000
## 3       M 15.12778
```

```
aggregate(hi$mass.a, by=list(hi$Den), stderr)
```

```
##   Group.1        x
## 1       H 2.435705
## 2       L 1.195700
## 3       M 3.220772
```

Below ground biomass

```
aggregate(hi$mass.b, by=list(hi$Den), mean)
```

```
##   Group.1         x
```

```
## 1       H 1.5922500
## 2       L 0.6319444
## 3       M 1.6233611
```

```
aggregate(hi$mass.b, by=list(hi$Den), stderr)
```

```
##   Group.1         x
## 1       H 0.3081703
## 2       L 0.1579752
## 3       M 0.3654846
```

Low $N_e$ Above ground biomass

```
aggregate(lo$mass.a, by=list(lo$Den), mean)
```

```
##   Group.1        x
## 1       H 9.377778
## 2       L 2.750000
## 3       M 6.388889
```

```
aggregate(lo$mass.a, by=list(lo$Den), stderr)
```

```
##   Group.1         x
## 1       H 2.7333975
## 2       L 0.5085694
## 3       M 0.8996541
```

Below ground biomass

```
aggregate(lo$mass.b, by=list(lo$Den), mean)
```

```
##   Group.1        x
## 1       H 1.426111
## 2       L 0.934500
## 3       M 1.361472
```

```
aggregate(lo$mass.b, by=list(lo$Den), stderr)
```

```
##   Group.1         x
## 1       H 0.2243047
## 2       L 0.3218768
## 3       M 0.2908803
```

## Linear Analysis of Biomass

### Above Ground Biomass (`mass.a`)

Sequentially add Density, $N_e$, and Den x $N_e$ interaction. Note all analyses log-transform biomass.

```
f.lm<- lm(log(mass.a) ~ (Den) , data=fin)

f.lm2<- lm(log(mass.a) ~ (Den + Gen) , data=fin)

f.lm3<- lm(log(mass.a) ~ (Den + Gen + Den*Gen) , data=fin)

summary(f.lm)
```

```
##
## Call:
## lm(formula = log(mass.a) ~ (Den), data = fin)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.5864 -1.4014  0.1614  1.3778  3.3957
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.2839     0.2088   6.150 3.78e-09 ***
## DenL         -1.0499     0.2952  -3.556 0.000464 ***
## DenM         -0.1893     0.2952  -0.641 0.521995
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.771 on 213 degrees of freedom
## Multiple R-squared:  0.0632, Adjusted R-squared:  0.0544
## F-statistic: 7.184 on 2 and 213 DF,  p-value: 0.0009563
```

```
summary(f.lm2)
```

```
##
## Call:
## lm(formula = log(mass.a) ~ (Den + Gen), data = fin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.7465 -1.3199  0.2282  1.4551  3.3212
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.4439     0.2406   6.001  8.4e-09 ***
## DenL         -1.0499     0.2947  -3.563 0.000454 ***
## DenM         -0.1893     0.2947  -0.642 0.521248
## GenLG        -0.3201     0.2406  -1.330 0.184831
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.768 on 212 degrees of freedom
## Multiple R-squared:  0.07095,    Adjusted R-squared:  0.05781
## F-statistic: 5.397 on 3 and 212 DF,  p-value: 0.001345
```

```
summary(f.lm3)
```

```
##
## Call:
## lm(formula = log(mass.a) ~ (Den + Gen + Den * Gen), data = fin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.0503 -1.3634  0.1512  1.3541  3.6250
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   1.7477     0.2939   5.947 1.12e-08 ***
## DenL         -1.5223     0.4156  -3.663 0.000316 ***
## DenM         -0.6285     0.4156  -1.512 0.131985
```

```
## GenLG        -0.9278     0.4156  -2.232 0.026643 *
## DenL:GenLG    0.9447     0.5877   1.607 0.109467
## DenM:GenLG    0.8782     0.5877   1.494 0.136612
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.763 on 210 degrees of freedom
## Multiple R-squared:  0.08498,    Adjusted R-squared:  0.06319
## F-statistic: 3.901 on 5 and 210 DF,  p-value: 0.00211
```

Perform liklihood ratio test of models with Density, $N_e$, and Den x $N_e$ interaction

```
anova(f.lm, f.lm2, f.lm3)
```

```
## Analysis of Variance Table
##
## Model 1: log(mass.a) ~ (Den)
## Model 2: log(mass.a) ~ (Den + Gen)
## Model 3: log(mass.a) ~ (Den + Gen + Den * Gen)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    213 668.41
## 2    212 662.87  1    5.5339 1.7800 0.1836
## 3    210 652.87  2   10.0093 1.6098 0.2024
```

Generate Least Square Means (EMeans) for Density treatmetns across High and Low $N_e$

```
emmeans(f.lm2, "Den", type='response', by="Gen")
```

```
## Gen = HG:
##  Den response    SE  df lower.CL upper.CL
##  H       4.24 1.020 212    2.637     6.81
##  L       1.48 0.357 212    0.923     2.38
##  M       3.51 0.844 212    2.182     5.63
##
## Gen = LG:
##  Den response    SE  df lower.CL upper.CL
##  H       3.08 0.740 212    1.915     4.94
##  L       1.08 0.259 212    0.670     1.73
##  M       2.55 0.613 212    1.584     4.09
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```
pairs(emmeans(f.lm2, "Den", type='response', by="Gen"))
```

```
## Gen = HG:
##  contrast ratio    SE  df t.ratio p.value
##  H / L    2.857 0.842 212  3.563  0.0013
##  H / M    1.208 0.356 212  0.642  0.7968
##  L / M    0.423 0.125 212 -2.920  0.0108
##
## Gen = LG:
##  contrast ratio    SE  df t.ratio p.value
##  H / L    2.857 0.842 212  3.563  0.0013
##  H / M    1.208 0.356 212  0.642  0.7968
##  L / M    0.423 0.125 212 -2.920  0.0108
##
```

```
## P value adjustment: tukey method for comparing a family of 3 estimates
## Tests are performed on the log scale
```

```
test(emmeans(f.lm2, "Den", type='response', by="Gen"))
```

```
## Gen = HG:
##  Den response    SE  df t.ratio p.value
##  H       4.24 1.020 212 6.001   <.0001
##  L       1.48 0.357 212 1.637   0.1030
##  M       3.51 0.844 212 5.214   <.0001
##
## Gen = LG:
##  Den response    SE  df t.ratio p.value
##  H       3.08 0.740 212 4.670   <.0001
##  L       1.08 0.259 212 0.307   0.7591
##  M       2.55 0.613 212 3.883   0.0001
##
## Tests are performed on the log scale
```

### Below Ground Biomass (mass.b)

Sequentially add Density, $N_e$, and Den x $N_e$ interaction. Note all analyses log-transform biomass.

```
b.lm<- lm(log(mass.b) ~ (Den) , data=fin)

b.lm2<- lm(log(mass.b) ~ (Den + Gen) , data=fin)

b.lm3<- lm(log(mass.b) ~ (Den + Gen + Den*Gen) , data=fin)

summary(b.lm)
```

```
##
## Call:
## lm(formula = log(mass.b) ~ (Den), data = fin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.3845 -1.3271  0.3035  1.7639  4.7921
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.5232     0.2752  -1.901   0.0586 .
## DenL         -2.0828     0.3893  -5.351 2.26e-07 ***
## DenM         -0.4522     0.3893  -1.162   0.2466
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.336 on 213 degrees of freedom
## Multiple R-squared:  0.1295, Adjusted R-squared:  0.1213
## F-statistic: 15.84 on 2 and 213 DF,  p-value: 3.852e-07
```

```
summary(b.lm2)
```

```
##
## Call:
## lm(formula = log(mass.b) ~ (Den + Gen), data = fin)
```

```
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -6.5072 -1.4498  0.4244  1.7945  4.9148
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -0.4006     0.3181  -1.259    0.209
## DenL         -2.0828     0.3896  -5.346 2.32e-07 ***
## DenM         -0.4522     0.3896  -1.161    0.247
## GenLG        -0.2453     0.3181  -0.771    0.442
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.338 on 212 degrees of freedom
## Multiple R-squared:  0.1319, Adjusted R-squared:  0.1196
## F-statistic: 10.74 on 3 and 212 DF,  p-value: 1.338e-06
```

```
summary(b.lm3)
```

```
##
## Call:
## lm(formula = log(mass.b) ~ (Den + Gen + Den * Gen), data = fin)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -6.404 -1.383  0.520  1.700  5.196
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.50357    0.39000  -1.291  0.19805
## DenL        -1.69884    0.55154  -3.080  0.00235 **
## DenM        -0.52726    0.55154  -0.956  0.34019
## GenLG       -0.03935    0.55154  -0.071  0.94319
## DenL:GenLG  -0.76794    0.78000  -0.985  0.32598
## DenM:GenLG   0.15007    0.78000   0.192  0.84762
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.34 on 210 degrees of freedom
## Multiple R-squared:  0.1385, Adjusted R-squared:  0.118
## F-statistic:  6.75 on 5 and 210 DF,  p-value: 7.395e-06
```

Perform liklihood ratio test of models with Density, $N_e$, and Den x $N_e$ interaction.

```
anova(b.lm, b.lm2, b.lm3)
```

```
## Analysis of Variance Table
##
## Model 1: log(mass.b) ~ (Den)
## Model 2: log(mass.b) ~ (Den + Gen)
## Model 3: log(mass.b) ~ (Den + Gen + Den * Gen)
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    213 1161.9
## 2    212 1158.6  1    3.2495 0.5934  0.442
## 3    210 1149.9  2    8.7299 0.7972  0.452
```

Generate Least Square Means (EMeans) for Density treatmetns across High and Low $N_e$

```
emmeans(b.lm2, "Den", type='response', by="Gen")
```

```
## Gen = HG:
##  Den response     SE  df lower.CL upper.CL
##  H     0.6699 0.2131 212   0.3578    1.254
##  L     0.0835 0.0266 212   0.0446    0.156
##  M     0.4262 0.1356 212   0.2277    0.798
##
## Gen = LG:
##  Den response     SE  df lower.CL upper.CL
##  H     0.5242 0.1668 212   0.2800    0.981
##  L     0.0653 0.0208 212   0.0349    0.122
##  M     0.3335 0.1061 212   0.1781    0.624
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```
pairs(emmeans(b.lm2, "Den", type='response', by="Gen"))
```

```
## Gen = HG:
##  contrast ratio     SE  df t.ratio p.value
##  H / L    8.027 3.1275 212  5.346  <.0001
##  H / M    1.572 0.6124 212  1.161  0.4780
##  L / M    0.196 0.0763 212 -4.185  0.0001
##
## Gen = LG:
##  contrast ratio     SE  df t.ratio p.value
##  H / L    8.027 3.1275 212  5.346  <.0001
##  H / M    1.572 0.6124 212  1.161  0.4780
##  L / M    0.196 0.0763 212 -4.185  0.0001
##
## P value adjustment: tukey method for comparing a family of 3 estimates
## Tests are performed on the log scale
```

```
test(emmeans(b.lm2, "Den", type='response', by="Gen"))
```

```
## Gen = HG:
##  Den response     SE  df t.ratio p.value
##  H     0.6699 0.2131 212 -1.259  0.2093
##  L     0.0835 0.0266 212 -7.806  <.0001
##  M     0.4262 0.1356 212 -2.681  0.0079
##
## Gen = LG:
##  Den response     SE  df t.ratio p.value
##  H     0.5242 0.1668 212 -2.030  0.0436
##  L     0.0653 0.0208 212 -8.577  <.0001
##  M     0.3335 0.1061 212 -3.452  0.0007
##
## Tests are performed on the log scale
```

# Aborted Ovules

## Preliminary and Summary of Data

The following code summarizes and performs linear analyses of the number of aborted ovules. The required data is in the same file as the female fitness data.

Reload data and load package `emmeans`

```
fin<- read.csv("C:data/aster.dat.csv")

library(emmeans)
```

Perform linear analysis of aborted ovules (log transformed) with Density, Density + $N_e$, and Density + $N_e$ + Density x $N_e$ in sequential models.

```
lm1<- lm(log(aborted +1) ~ Den , data=fin)

lm2<- lm(log(aborted +1) ~ Den + Gen, data=fin)

lm3<- lm(log(aborted +1) ~ Gen + Den + Den*Gen, data=fin)

summary(lm1)
```

```
##
## Call:
## lm(formula = log(aborted + 1) ~ Den, data = fin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4257 -0.5497  0.2370  1.0091  2.4230
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.4257     0.1599  21.426  < 2e-16 ***
## DenL         -1.3601     0.2261  -6.015 7.74e-09 ***
## DenM         -0.6282     0.2261  -2.778  0.00596 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.357 on 213 degrees of freedom
## Multiple R-squared:  0.1454, Adjusted R-squared:  0.1374
## F-statistic: 18.13 on 2 and 213 DF,  p-value: 5.377e-08
```

```
summary(lm2)
```

```
##
## Call:
## lm(formula = log(aborted + 1) ~ Den + Gen, data = fin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4863 -0.5556  0.1837  0.9946  2.3870
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.3651     0.1849  18.203  < 2e-16 ***
```

```
## DenL         -1.3601     0.2264  -6.007 8.12e-09 ***
## DenM         -0.6282     0.2264  -2.774  0.00603 **
## GenLG         0.1212     0.1849   0.656  0.51280
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.359 on 212 degrees of freedom
## Multiple R-squared:  0.1472, Adjusted R-squared:  0.1351
## F-statistic: 12.19 on 3 and 212 DF,  p-value: 2.154e-07
```

```
summary(lm3)
```

```
##
## Call:
## lm(formula = log(aborted + 1) ~ Gen + Den + Den * Gen, data = fin)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.4981 -0.5671  0.1754  0.9508  2.2743
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.3534     0.2270  14.772  < 2e-16 ***
## GenLG         0.1446     0.3210   0.451    0.653
## DenL         -1.4494     0.3210  -4.515 1.06e-05 ***
## DenM         -0.5037     0.3210  -1.569    0.118
## GenLG:DenL    0.1785     0.4540   0.393    0.695
## GenLG:DenM   -0.2489     0.4540  -0.548    0.584
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 1.362 on 210 degrees of freedom
## Multiple R-squared:  0.1508, Adjusted R-squared:  0.1306
## F-statistic: 7.458 on 5 and 210 DF,  p-value: 1.838e-06
```

Perform liklihood ratio test of above models

```
anova(lm1, lm2, lm3)
```

```
## Analysis of Variance Table
##
## Model 1: log(aborted + 1) ~ Den
## Model 2: log(aborted + 1) ~ Den + Gen
## Model 3: log(aborted + 1) ~ Gen + Den + Den * Gen
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    213 392.06
## 2    212 391.27  1   0.79321 0.4275 0.5139
## 3    210 389.61  2   1.65923 0.4472 0.6400
```

Finally, generate least square means

```
emmeans(lm3, "Den", "Gen", type="response")
```

```
## Gen = HG:
##  Den response   SE  df lower.CL upper.CL
##  H      28.60 6.49 210    18.28     44.7
##  L       6.71 1.52 210     4.29     10.5
```

```
##  M      17.28 3.92 210    11.05     27.0
##
## Gen = LG:
##  Den response   SE  df lower.CL upper.CL
##  H      33.05 7.50 210    21.13     51.7
##  L       9.27 2.11 210     5.93     14.5
##  M      15.57 3.53 210     9.95     24.4
##
## Confidence level used: 0.95
## Intervals are back-transformed from the log scale
```

```
pairs(emmeans(lm3, "Den","Gen", type="response"))
```

```
## Gen = HG:
##  contrast ratio    SE  df t.ratio p.value
##  H / L    4.260 1.368 210  4.515  <.0001
##  H / M    1.655 0.531 210  1.569  0.2614
##  L / M    0.388 0.125 210 -2.946  0.0100
##
## Gen = LG:
##  contrast ratio    SE  df t.ratio p.value
##  H / L    3.564 1.144 210  3.958  0.0003
##  H / M    2.123 0.681 210  2.344  0.0520
##  L / M    0.596 0.191 210 -1.614  0.2419
##
## P value adjustment: tukey method for comparing a family of 3 estimates
## Tests are performed on the log scale
```

```
test(emmeans(lm3, "Den","Gen", type="response"))
```

```
## Gen = HG:
##  Den response   SE  df t.ratio p.value
##  H      28.60 6.49 210 14.772  <.0001
##  L       6.71 1.52 210  8.387  <.0001
##  M      17.28 3.92 210 12.553  <.0001
##
## Gen = LG:
##  Den response   SE  df t.ratio p.value
##  H      33.05 7.50 210 15.409  <.0001
##  L       9.27 2.11 210  9.811  <.0001
##  M      15.57 3.53 210 12.094  <.0001
##
## Tests are performed on the log scale
```