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

Mason W. Kulbaba & Ruth G. Shaw

April 5, 2019

Introduction

The following code performs fixed-effects aster analyses on data examining the effects of density and effective genetic population size (Ne) 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 (Ne 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(var
```

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"
```

Aster Analyses

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
```

Make Up Data

Simulate regression data, and do the regression.

```
n <- 50
x <- seq(1, n)
a.true <- 3
b.true <- 1.5
y.true <- a.true + b.true * x
s.true <- 17.3
y <- y.true + s.true * rnorm(n)
out1 <- lm(y ~ x)
summary(out1)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -49.295 -13.076  -0.139  18.363  43.578
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   3.4926     5.6624   0.617     0.54
## x             1.5946     0.1933   8.252  9.2e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 19.72 on 48 degrees of freedom
## Multiple R-squared:  0.5865, Adjusted R-squared:  0.5779
## F-statistic: 68.09 on 1 and 48 DF,  p-value: 9.202e-11
```

Figure with Code to Make It Shown

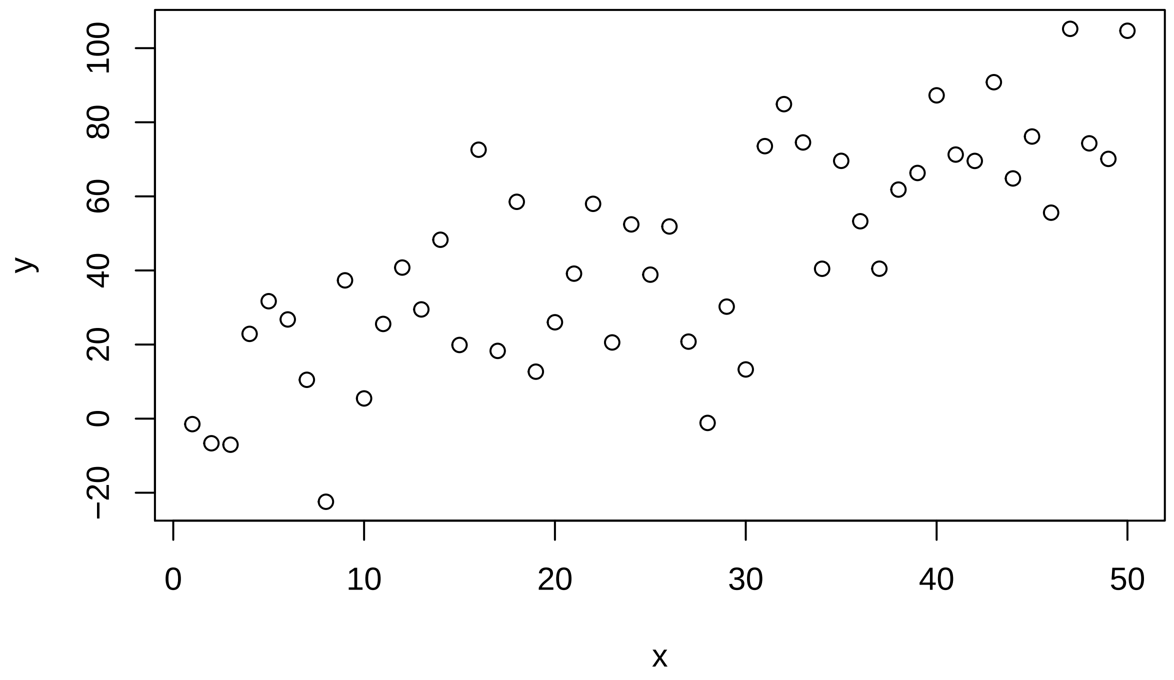

Figure 1: Simple Linear Regression

The following figure is produced by the following code

```
mydata <- data.frame(x, y)
plot(mydata)
```