Stat 8931 (Aster Models)
Lecture Slides Deck 3

Charles J. Geyer

School of Statistics
University of Minnesota

June 7, 2015
Example Two


There are three examples in the paper. This is the one involving *Echinacea angustifolia*.

In that paper and the accompanying tech report this example was held up as one where the model was too complicated to use the R formula mini-language and model matrices for different submodels had to be constructed “by hand” (using R but without any helpful R functions).

The light dawned somewhat later that the R formula mini-language can describe these models if one is sufficiently “tricky”, so we show that here — at least that was the original intention — but if there is a tricky way to do it without model matrices, we never found it.
Also this example illustrates some very important points about aster models and hypothesis tests.
Example Two (cont.)

Sorry the graph is snakelike. Otherwise, it would be really small.
All of the conditionally Bernoulli variables are survival indicators, but they are not all the same. This comes from an experiment in which the individual plants are crosses with parents from different ancestral populations.

The first three survival indicators \(y_1\) though \(y_3\) are for survival in the growth chamber. Individuals that survived these first three time periods were transplanted into the experimental field. The next five survival indicators \(y_5\) though \(y_8\) are for annual survival in the field (2001 through 2005). The conditionally zero-truncated Poisson components of the response vector \(y_9\) though \(y_{11}\) are for rosette (basal leaf cluster) counts. These plants had not lived long enough when these data were analyzed for the 2008 paper to have flowers yet.
Example Two (cont.)

It is not completely clear (to me) why zero-truncated Poisson. Perhaps the issue is that a plant with rosette count $= 0$ has no leaves and is not long for this world if not dead already.

If it were possible to observe rosette count $= 0$ on surviving plants, then the conditional distribution should have been Poisson rather than zero-truncated Poisson.

We will stick with zero-truncated Poisson, following the paper.
We take the rosette count for 2005 (the last year in the data) for an individual to be the best surrogate of observed fitness in these data (hereafter just called “fitness”).

This is often done — use some measure of size as best surrogate of fitness — when actual fecundity traits are not available. This is justified by size being (in many species) positively correlated with fitness.
The data for this example are in the dataset echin2 in the aster package.

```r
> library(aster)
> data(echin2)
> sapply(echin2, class)
```

```
crosstype yearcross   flat     row     posi
"factor"   "factor"   "factor"   "factor"   "numeric"
  varb       resp      id      root
"factor"   "integer"  "integer"  "numeric"
```

Since we already have varb, resp, and id as variables, it is clear (as the help page for this dataset says) that this is a “long format” data frame that does not need to be reshaped with the reshape function.
All datasets in the aster package except the very first one (the dataset echinacea used for example one in decks 1 and 2) are like this: “long format” so that they do not need to be reshaped with the reshape function.

The idea is we left one example of how to go from what we guess is a more likely format for users to use (one row of the data for each individual) to the format the aster and reaster functions need (go from “wide” to “long”) with the reshape function.

After that one example, we don’t need more.
Set up the graphical model.

```r
> levels(echin2$varb)
```

```
[1] "ld01"  "ld02"  "ld03"  "ld04"
[5] "ld05"  "lds1"  "lds2"  "lds3"
```

```r
> vars <- unique(as.character(echin2$varb))
> vars
```

```
[1] "lds1"  "lds2"  "lds3"  "ld01"
[5] "ld02"  "ld03"  "roct2003" "ld04"
[9] "roct2004" "ld05"  "roct2005"
```

```r
> pred <- c(0, 1, 2, 3, 4, 5, 6, 6, 8, 8, 10)
> fam <- c(1, 1, 1, 1, 1, 1, 3, 1, 3, 1, 3)
```
Example Two (cont.)

```r
> foo <- rbind(vars, c("initial", vars)[pred + 1])
> rownames(foo) <- c("successor", "predecessor")
> t(foo)
```

```
successor  predecessor
[1,]  "lds1"     "initial"
[2,]  "lds2"     "lds1"
[3,]  "lds3"     "lds2"
[4,]  "ld01"     "lds3"
[5,]  "ld02"     "ld01"
[6,]  "ld03"     "ld02"
[7,]  "roct2003" "ld03"
[8,]  "ld04"     "ld03"
[9,]  "roct2004" "ld04"
[10,] "ld05"     "ld04"
[11,] "roct2005" "ld05"
```
Example Two (cont.)

```r
> cbind(vars, fam)

vars       fam
[1,]  "lds1"   "1"
[2,]  "lds2"   "1"
[3,]  "lds3"   "1"
[4,]  "ld01"   "1"
[5,]  "ld02"   "1"
[6,]  "ld03"   "1"
[7,] "roct2003" "3"
[8,]  "ld04"   "1"
[9,] "roct2004" "3"
[10,] "ld05"   "1"
[11,] "roct2005" "3"
```
I have to admit the first time I did this slide deck I fouled this up. I decided another order of the node names in vars would look nicer.

Does not work. The R function `aster` is going to use the order in the data frame it is given (we do not feed it the `vars` object, it only has the `varb` variable in that data frame to work with). vars must be defined as above, if we are using a pre-existing “long format” data frame.
So that takes care of the graphical model, and we know what the variables `varb`, `resp`, `id`, and `root` do.

How about covariates?

```r
> sapply(echin2, class)

crosstype yearcross flat row posi
"factor" "factor" "factor" "factor" "numeric"
varb resp id root
"factor" "integer" "integer" "numeric"
```

Those are `crosstype`, `yearcross`, `flat`, `row`, and `posi`. 
Example Two (cont.)

> levels(echin2$crosstype)

[1]  "Br"  "Wi"  "Wr"

From the legend for Figure 2 in the paper

The experimentally imposed crossing treatments are between remnant populations ("Br"), within remnant populations ("Wr"), and inbred within remnants ("Wi")

This is the treatment of scientific interest. Does amount of inbreeding affect fitness?
Example Two (cont.)

```r
> levels(echin2$yearcross)

[1] "1999" "2000"
```

The year in which crosses were done.

```r
> levels(echin2$flat)

[1] "1" "2" "3"
```

The planting tray the individual was in while in the growth chamber.
Example Two (cont.)

> levels(echin2$\text{row})

[1] "0" "10" "11" "12" "13"

The row of the experimental field in which the individual was planted if the individual survived that long. Individuals that did not survive to be transplanted have level "0" of this factor (the four actual rows are levels "10", "11", "12", and "13")

> range(echin2$\text{posi})

[1] -0.350 0.365

Position along the row. This is meaningless for individuals with row == "0".
The predictor variables `row` and `posi` giving the outdoor spatial location after transplantation cause all the difficulty in specifying models. These variables are meaningless for individuals that did not survive to be transplanted but R forces us to give them values. Somehow our model formulas have to do the Right Thing (TRT) despite this complication.

```r
> unique(echin2$posi[as.character(echin2$row) == "0"])

[1] 0
```

At least we know we can ignore this complication for `posi` (multiplying a beta by zero is the same as leaving that term out of a regression equation).
We need to set up some more “predictor variables” which are indicator variables for parts of the graph.

```r
> indoors <- grepl("lds", as.character(echin2$varb))
> indoors <- as.numeric(indoors)
> outdoors <- 1 - indoors
> fit <- grepl("roct2005", as.character(echin2$varb))
> fit <- as.numeric(fit)
> echin2 <- data.frame(echin2, indoors = indoors,
+ outdoors = outdoors, fit = fit)
> names(echin2)
```

```
[1] "crosstype"  "yearcross"  "flat"       "row"
[5] "posi"      "varb"      "resp"      "id"
[9] "root"      "indoors"   "outdoors"  "fit"
```
And we check that our understanding of row == "0" is correct.

```r
> is.lds3 <- as.character(echin2$varb) == "lds3"
> is.dead <- is.lds3 & echin2$resp == 0
> is.norow <- as.character(echin2$row) == "0"
> with(echin2, identical(id[is.dead],
+   unique(id[is.norow]))))
```

[1] TRUE

An individual has row == "0" if and only if that individual died in the growth chamber (lds3 == 0).
Our formula starts (as always) with \( \text{resp} \sim \text{varb} \).

The next term is \( \text{fit} : \text{crosstype} \). This is the most important term for scientific inference. What is the effect of inbreeding on fitness?
The next term is just yearcross. At least that is what was done in the paper.

But does it make sense that the year of crossing has the same effect (numerically) on different kinds of variables? Our slogan **no naked predictors** says this should be something like layer : yearcross with layer defined as follows:

```r
> layer <- gsub("[0-9]", ",", as.character(echin2$varb))
> layer <- as.factor(layer)
> levels(layer)
[1] "ld" "lds" "roct"
```

```r
> echin2 <- data.frame(echin2, layer = layer)
```

(the two kinds of survival variables are for different lengths of time, weeks or years, and so need to be distinguished).
As we shall see, yearcross isn’t statistically significant anyway, so if we hadn’t fussed about this it wouldn’t have mattered for these data.

An alternative way to deal with yearcross would be to have it affect only the first survival node and no other (the term would be lds01 : yearcross with the definition of lds01 left as an exercise for the reader).

Exactly how yearcross should go in is a scientific question not a statistical one.
The next term is indoors: flat because, presumably, what flat an individual was in is no longer relevant after it has been transplanted and is no longer in that flat.

The next term is outdoors: (row + posi) because, presumably, the location where an individual will be transplanted does not matter before the transplantation occurs.

It might be a problem that the interpreter for the R formula mini-language is going to make a dummy variable named outdoors:row0 that will just be the zero vector (every individual who survives to be transplanted outdoors goes in a real row not in the fake row "0"). Since such a predictor variable can never be in a full rank matrix R will drop it (we hope, let’s see).
Example Two (cont.)

```r
> aout <- aster(resp ~ varb + layer : yearcross + indoors : 
+   outdoors : (row + posi) + fit : crosstype, 
+   pred, fam, varb, id, root, data = echin2)
> try(summary(aout), silent = TRUE)

apparent null eigenvectors of information matrix
directions of recession or constancy of log likelihood
[1]  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
```
Well that’s annoying. What is that? What is a direction of recession or constancy?

Too complicated to explain here (much more on this when we get to the subject).
Example Two (cont.)

```r
> names(aout$coefficients)

[1] "(Intercept)"
[2] "varbld02"
[3] "varbld03"
[4] "varbld04"
[5] "varbld05"
[6] "varbls1"
[7] "varbls2"
[8] "varbls3"
[9] "varbroct2003"
[10] "varbroct2004"
[12] "layerld:yearcross1999"
[14] "layerroct:yearcross1999"
[15] "indoors:flat1"
[16] "indoors:flat2"
[17] "outdoors:row0"
[18] "outdoors:row10"
[19] "outdoors:row11"
[20] "outdoors:row12"
[21] "outdoors:posi"
[22] "fit:crosstypeBr"
[23] "fit:crosstypeWi"
```
So our reasoning about `outdoors:row0` was wrong (isn’t it funny how I say “our” reasoning as if the mistake was not entirely mine).

The individuals that died in the growth chamber are outdoors as far as the `aster` function is concerned. They have values for all the nodes. It's just that those values are zero for node "lds3" and all later nodes and perhaps some earlier nodes.

So now are we stuck? Do we have to construct model matrices by hand?
Dealing with this example would be a lot easier if we could use different graphs for different individuals. Don't have “outdoors” nodes for individuals that weren’t transplanted.

Actually, that would be possible. We would tell aster that there is just one individual but the graph has a lot of nodes.

Nothing in the aster function requires that the graph be connected. In our “big graph” every maximal connected component (all the nodes downstream from one initial node) would correspond to one real individual (which the aster function wouldn’t know about or care). But it would be an enormous pain to specify a model this way. (The aster2 packages makes this easier in some cases, but isn’t ready for use yet.)
But there is an easier idea. We didn’t define outdoors correctly. We should have excluded the “outdoors” nodes for individuals who were not transplanted.

```r
> id.transplant <- with(echin2, 
+     id[as.character(varb) == "lds3" & resp > 0])
> transplant <- with(echin2, id %in% id.transplant)
> transplant <- as.numeric(transplant)
> echin2 <- data.frame(echin2, transplant = transplant)
```

Now does it work?
Example Two (cont.)

```r
> aout <- aster(resp ~ varb + layer : yearcross + indoors : flat +
+ transplant : (row + posi) + fit : crosstype,
+ pred, fam, varb, id, root, data = echin2)
> try(summary(aout), silent = TRUE)

apparent null eigenvectors of information matrix
directions of recession or constancy of log likelihood

[,1]            [,2]
[1,]  0.0675709   0.3718754
[2,]  0.0000000   0.0000000
[3,]  0.0000000   0.0000000
[4,]  0.0000000   0.0000000
[5,]  0.0000000   0.0000000
[6,] -0.0675709  -0.3718754
[7,] -0.0675709  -0.3718754
[8,]  0.9838898  -0.1787757
[9,]  0.0000000   0.0000000
[10,]  0.0000000   0.0000000
```
Example Two (cont.)

```r
> names(aout$coefficients)

[1] "(Intercept)"
[2] "varbld02"
[3] "varbld03"
[4] "varbld04"
[5] "varbld05"
[6] "varblds1"
[7] "varblds2"
[8] "varblds3"
[9] "varbroct2003"
[10] "varbroct2004"
[12] "layerld:yearcross1999"
[14] "layerroct:yearcross1999"
[15] "indoors:flat1"
[16] "indoors:flat2"
[17] "transplant:row10"
[18] "transplant:row11"
[19] "transplant:row12"
[20] "transplant:row13"
[21] "transplant:posi"
[22] "fit:crosstypeBr"
[23] "fit:crosstypeWi"
```
We don’t have row0 any more, but we still have a problem. It is still complaining about directions of recession (DOR) or constancy (DOC).

And I will tell you that when there are a lot of zeros and repeated values in the “apparent” DOR or DOC, they probably really are one or the other.

That is, we appear to have a real problem. But we don’t want to delve into that can of worms yet.
Example Two (cont.)

```r
> aout.no.yearcross <- aster(resp ~ varb + indoors : flat +
+ transplant : (row + posi) + fit : crosstype,
+ pred, fam, varb, id, root, data = echin2)
> try(summary(aout), silent = TRUE)

apparent null eigenvectors of information matrix
directions of recession or constancy of log likelihood

[,1]   [,2]
[1,]  0.0675709  0.3718754
[2,]  0.0000000  0.0000000
[3,]  0.0000000  0.0000000
[4,]  0.0000000  0.0000000
[5,]  0.0000000  0.0000000
[6,] -0.0675709 -0.3718754
[7,] -0.0675709 -0.3718754
[8,]  0.9838898 -0.1787757
[9,]  0.0000000  0.0000000
[10,]  0.0000000  0.0000000
```

Example Two (cont.)

> names(aout.no.yearcross$coefficients)

[1] "(Intercept)"   "varbld02"
[3] "varbld03"      "varbld04"
[5] "varbld05"      "varblds1"
[7] "varblds2"      "varblds3"
[9] "varbroct2003"  "varbroct2004"
[13] "indoors:flat2"  "transplant:row10"
[17] "transplant:row13" "transplant:posi"
[19] "fit:crosstypeBr"  "fit:crosstypeWi"
Looks like we are stuck. We need at least a little bit of info about DOR and DOC to get going.

A **direction of constancy** (DOC) is a direction in the parameter space (where $\beta$ lives) in which the log likelihood is constant (moving along lines in that direction does not change the log likelihood).

A **direction of recession** (DOR) is a direction in the parameter space (where $\beta$ lives) in which the log likelihood is nondecreasing (moving along lines in that direction the log likelihood increases or stays the same).

Whenever there is a DOR that is not a DOC, the MLE does not exist (the log likelihood goes up hill in that direction all the way to infinity).
Example Two (cont.)

Theory of exponential families (which we will eventually get to, but for now leave vague) says the MLE does not exist when the canonical statistic vector $M^T y$ is on the boundary of its range.

In general, that is hard to check.

We can only hope it is easy in the case at hand.
If we look inside the R function `summary.aster`, here is how it makes these “apparent” DOR or DOC.

```r
> info.tol <- sqrt(.Machine$double.eps)
> info.tol

[1] 1.490116e-08

> infomat <- aout$fisher
> fred <- eigen(infomat, symmetric = TRUE)
> sally <- fred$values < max(fred$values) * info.tol
> apparent <- zapsmall(fred$vectors[, sally])
> rownames(apparent) <- names(aout$coefficients)
> apparent

        [,1]       [,2]
(Intercept)  0.0675709  0.3718754
varbld02  0.0000000  0.0000000
varbld03  0.0000000  0.0000000
varbld04  0.0000000  0.0000000
varblds1 -0.0675709 -0.3718754
varblds2 -0.0675709 -0.3718754
varblds3  0.9838898 -0.1787757
varbroct2003  0.0000000  0.0000000
varbroct2004  0.0000000  0.0000000
varbroct2005  0.0000000  0.0000000
layerld:yearcross1999  0.0000000  0.0000000
layerlds:yearcross1999  0.0000000  0.0000000
layerroct:yearcross1999  0.0000000  0.0000000
indoors:flat1  0.0000000  0.0000000
indoors:flat2  0.0000000  0.0000000
transplant:row10 -0.0675709 -0.3718754
transplant:row11 -0.0675709 -0.3718754
transplant:row12 -0.0675709 -0.3718754
transplant:row13 -0.0675709 -0.3718754
transplant:posi  0.0000000  0.0000000
fit:crosstypeBr  0.0000000  0.0000000
fit:crosstypeWi  0.0000000  0.0000000
```
Computationally, these are apparent null eigenvectors of the Fisher information matrix (and we don’t yet know what that is either).

Since lengths of direction vectors do not matter, we can try to make them look nicer by changing the length.

We can also just drop all the zero rows. Those are the components of the submodel canonical parameter and statistic that are not involved in the DOR/DOC.

```r
> zero.rows <- apply(apparent == 0, 1, all)
> apparent <- apparent[! zero.rows, ]
```
Example Two (cont.)

We see a pattern. Lots of the same numbers in each column.

```r
> apparent

<table>
<thead>
<tr>
<th></th>
<th>[,1]</th>
<th>[,2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Intercept)</td>
<td>0.0675709</td>
<td>0.3718754</td>
</tr>
<tr>
<td>varblds1</td>
<td>-0.0675709</td>
<td>-0.3718754</td>
</tr>
<tr>
<td>varblds2</td>
<td>-0.0675709</td>
<td>-0.3718754</td>
</tr>
<tr>
<td>varblds3</td>
<td>0.9838898</td>
<td>-0.1787757</td>
</tr>
<tr>
<td>transplant:row10</td>
<td>-0.0675709</td>
<td>-0.3718754</td>
</tr>
<tr>
<td>transplant:row11</td>
<td>-0.0675709</td>
<td>-0.3718754</td>
</tr>
<tr>
<td>transplant:row12</td>
<td>-0.0675709</td>
<td>-0.3718754</td>
</tr>
<tr>
<td>transplant:row13</td>
<td>-0.0675709</td>
<td>-0.3718754</td>
</tr>
</tbody>
</table>
```
Example Two (cont.)

```r
> apparent <- sweep(apparent, 2, apparent[1, ], "/")
> apparent

[,1]      [,2]
(Intercept) 1.000000 1.0000000
varblds1   -1.000000 -1.0000000
varblds2   -1.000000 -1.0000000
varblds3  14.560854 -0.4807409
transplant:row10 -1.000000 -1.0000000
transplant:row11 -1.000000 -1.0000000
transplant:row12 -1.000000 -1.0000000
transplant:row13 -1.000000 -1.0000000
```
I think I see what’s happening.

The (pseudo) predictor transplant perfectly predicts the lds3 node of the graph. That’s DOR.

I am not clear why there are two DOR/DOC instead of one.

But I am out of ideas. Looks like we do have to do model matrices “by hand” for this one.
Example Two (cont.)

```r
> m <- model.matrix(resp ~ varb + layer : yearcross + indoor : flat + outdoors : (row + posi) + fit : crosstype, data = echin2)
> dim(m)

[1] 6127  29

> m <- m[, colnames(m) != "outdoors:row0"]
> dim(m)

[1] 6127  28
```
> aout <- aster(resp ~ 0 + m,  
+     pred, fam, varb, id, root, data = echin2)  
> try(summary(aout), silent = TRUE)

apparent null eigenvectors of information matrix  
directions of recession or constancy of log likelihood  
  [1]  0.3535534  0.0000000  0.0000000  0.0000000  
  [5]  0.0000000 -0.3535534 -0.3535534 -0.3535534  
  [9]  0.0000000  0.0000000  0.0000000  0.0000000  
 [13]  0.0000000  0.0000000  0.0000000  0.0000000  
 [17] -0.3535534 -0.3535534 -0.3535534 -0.3535534  
 [21]  0.0000000  0.0000000  0.0000000
This is hard to see without extracting the eigenvectors as we did before and putting labels on the components of the DOR or DOC. If we do that (for once not shown) we see that it is a problem with lds nodes and outdoors nodes.

And the answer is obvious if we add up all the outdoor:row1x dummy variables we just get outdoor, which is the same as the sum of the of the varb1dsx dummy variables.

Have to drop something. R would have dropped the "row10" level of the row factor if we hadn’t had the bogus "row0" level to confuse it.
Example Two (cont.)

```r
> dim(m)
[1] 6127  28

> grep("row", colnames(m), value = TRUE)

> m <- m[, !grepl("row10", colnames(m))]
> dim(m)
[1] 6127   27
```
Example Two (cont.)

```r
> aout <- aster(resp ~ 0 + m,
+     pred, fam, varb, id, root, data = echin2)
> try(summary(aout))
```

**Call:**

```r
aster.formula(formula = resp ~ 0 + m, pred = pred, fam = fam,
              varvar = varb, idvar = id, root = root, data = echin2)
```

<table>
<thead>
<tr>
<th>Estimate</th>
<th>Std. Error</th>
<th>z value</th>
</tr>
</thead>
<tbody>
<tr>
<td>m(Intercept)</td>
<td>0.74649</td>
<td>0.27173</td>
</tr>
<tr>
<td>mvarbld02</td>
<td>0.18029</td>
<td>0.44949</td>
</tr>
<tr>
<td>mvarbld03</td>
<td>1.54701</td>
<td>0.46449</td>
</tr>
<tr>
<td>mvarbld04</td>
<td>1.44860</td>
<td>0.47435</td>
</tr>
<tr>
<td>mvarbld05</td>
<td>2.90222</td>
<td>0.38842</td>
</tr>
<tr>
<td>mvarblds1</td>
<td>-1.13189</td>
<td>0.50667</td>
</tr>
<tr>
<td>mvarblds2</td>
<td>0.32434</td>
<td>0.57666</td>
</tr>
<tr>
<td>mvarblds3</td>
<td>1.21221</td>
<td>0.56491</td>
</tr>
<tr>
<td>mvarbroct2003</td>
<td>-3.01924</td>
<td>0.34267</td>
</tr>
</tbody>
</table>
```

Original predictor variables dropped (aliased)

- mlayerld:yearcross2000
- mlayerlds:yearcross2000
- mlayerroct:yearcross2000
- mindoors:flat3
- mfit:crosstypeWr
|                   | Estimate  | Std. Error | z value | Pr(>|z|) |
|-------------------|-----------|------------|---------|----------|
| m(Intercept)      | 0.746488  | 0.271726   | 2.7472  | 0.0060105|
| mvarbld02         | 0.180287  | 0.449488   | 0.4011  | 0.6883513|
| mvarbld03         | 1.547012  | 0.464487   | 3.3306  | 0.0008667|
| mvarbld04         | 1.448602  | 0.474349   | 3.0539  | 0.0022590|
| mvarbld05         | 2.902224  | 0.388415   | 7.4720  | 7.901e-14 |
| mvarblds1         | -1.131885 | 0.506668   | -2.2340 | 0.0254845|
| mvarblds2         | 0.324339  | 0.576658   | 0.5624  | 0.5738117|
| mvarblds3         | 1.212208  | 0.564913   | 2.1458  | 0.0318863|
| mvarbroct2003     | -3.019236 | 0.342669   | -8.8109 | < 2.2e-16 |
| mvarbroct2004     | -1.910658 | 0.299234   | -6.3852 | 1.712e-10 |
| mvarbroct2005     | -1.141527 | 0.309447   | -3.6889 | 0.0002252|
| mlayerld:yearcross1999 | 0.025827 | 0.128321   | 0.2013  | 0.8404856|
| mlayerlds:yearcross1999 | 0.311623 | 0.226641   | 1.3750  | 0.1691433|
| mlayerroct:yearcross1999 | -0.028953 | 0.121738   | -0.2378 | 0.8120099|
| mindoors:flat1    | -0.245502 | 0.186994   | -1.3129 | 0.1892203|
| mindoors:flat2    | -0.491118 | 0.172267   | -2.8509 | 0.0043594|
| moutdoors:row11   | 0.145657  | 0.036319   | 4.0105  | 6.060e-05 |
| moutdoors:row12   | 0.121987  | 0.035296   | 3.4561  | 0.0005481|
| moutdoors:row13   | 0.118659  | 0.034554   | 3.4340  | 0.0005947|
| moutdoors:posi    | -0.322929 | 0.069919   | -4.6186 | 3.863e-06 |
| mfit:crosstypeBr  | 0.101693  | 0.139439   | 0.7293  | 0.4658167|
| mfit:crosstypeWi  | -0.562843 | 0.181571   | -3.0999 | 0.0019361|
Example Two (cont.)

Hooray! It worked!
What a struggle!
But maybe it will go faster now that we have a clue.
But another issue occurred to me while the struggle was going on. We are not really following the dictum about “naked” predictors. The row and posi predictors do not make sense for different kinds of variables. As in example one, where we had nsloc and ewloc “interacted with” layer, we should have the same for row and posi here.

So we start over.

There will be a problem that any of these outside : layer : row dummy variables that involve ldx nodes are bogus, since such nodes aren’t outside. We will have to remove them too.
Example Two (cont.)

```r
> m <- model.matrix(resp ~ varb + layer : yearcross +
+     indoors : flat + outdoors : layer : (row + posi) +
+     fit : crosstype, data = echin2)
> outies <- grepl("row0|layerlds:outdoors|row10",
+     colnames(m))
```
Example Two (cont.)

```r
> colnames(m)[outies]

[1] "layer1d:outdoors:row0"
[2] "layer1ds:outdoors:row0"
[3] "layerroct:outdoors:row0"
[8] "layer1ds:outdoors:row12"
[9] "layer1ds:outdoors:row13"
[10] "layer1ds:outdoors:posi"
```
Example Two (cont.)

```r
> dim(m)
[1] 6127 41

> m <- m[, !outies]
> dim(m)
[1] 6127 31
```
> aout <- aster(resp ~ 0 + m, 
+     pred, fam, varb, id, root, data = echin2)
> try(summary(aout))

Call:
aster.formula(formula = resp ~ 0 + m, pred = pred, fam = fam, 
    varvar = varb, idvar = id, root = root, data = echin2)

                      Estimate Std. Error  z value
m(Intercept)          0.56195   0.27577    2.038
mvarbld02             0.08454   0.45061    0.188
mvarbld03            1.63006   0.46838    3.480
mvarbld04            1.54872   0.47826    3.238
mvarbld05            3.00295   0.39238    7.653
mvarbllds1          -0.94874   0.50882   -1.865
mvarbllds2           0.51148   0.57856    0.884
mvarbllds3           1.40328   0.56666    2.476
mvarbroct2003       -2.67680   0.36469   -7.340

Pr(>|z|)  
m(Intercept) 0.041575 * 
mvarbld02 0.851185 
mvarbld03 0.000501 *** 
mvarbld04 0.001203 ** 
mvarbld05 1.96e-14 *** 
mvarbllds1 0.062242 . 
mvarbllds2 0.376672 
mvarbllds3 0.013272 * 
mvarbroct2003 2.14e-13 *** 

---
Signif. codes:  
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Original predictor variables dropped (aliased)
mlayerld:yearcross2000
mlayerlds:yearcross2000
mlayerroct:yearcross2000
mindoors:flat3
mfit:crosstypeWr
### Example Two (cont.)

|                  | Estimate | Std. Error | z value | Pr(>|z|) |
|------------------|----------|------------|---------|----------|
| m(Intercept)     | 0.561948 | 0.275769   | 2.0378  | 0.0415747 |
| mvarbld02        | 0.084537 | 0.450606   | 0.1876  | 0.8511847 |
| mvarbld03        | 1.630055 | 0.468384   | 3.4802  | 0.0005011 |
| mvarbld04        | 1.548720 | 0.478261   | 3.2382  | 0.0012027 |
| mvarbld05        | 3.002947 | 0.392378   | 7.6532  | 1.960e-14  |
| mvarblds1        | -0.948737| 0.508824   | -1.8646 | 0.0622421  |
| mvarblds2        | 0.511475 | 0.578562   | 0.8840  | 0.3766717  |
| mvarblds3        | 1.403275 | 0.566664   | 2.4764  | 0.0132723  |
| mvarbroct2003    | -2.676800| 0.364688   | -7.3400 | 2.136e-13  |
| mvarbroct2004    | -1.576635| 0.324294   | -4.8617 | 1.164e-06  |
| mvarbroct2005    | -0.800018| 0.334819   | -2.3894 | 0.0168756  |
| mlayerld:yearcross1999 | 0.069508 | 0.128345 | 0.5416 | 0.5881161  |
| mlayerlds:yearcross1999 | 0.287124 | 0.226411 | 1.2682 | 0.2047433  |
| mlayerroct:yearcross1999 | -0.081130 | 0.122418 | -0.6627 | 0.5075034  |
| mindoors:flat1   | -0.258737| 0.187393   | -1.3807 | 0.1673669  |
| mindoors:flat2   | -0.476028| 0.172587   | -2.7582 | 0.0058123  |
| mfit:crosstypeBr | 0.086439 | 0.140414   | 0.6156  | 0.5381578  |
| mfit:crosstypeWi | -0.588787| 0.181631   | -3.2417 | 0.0011883  |
| mlayerld:outdoors:row11 | 0.158092 | 0.128729 | 1.2281 | 0.2194098  |
| mlayerroct:outdoors:row11 | 0.103061 | 0.137350 | 0.7504 | 0.4530415  |
| mlayerld:outdoors:row12 | 0.704913 | 0.148235 | 4.7554 | 1.981e-06  |
| mlayerroct:outdoors:row12 | -0.523367 | 0.160789 | -3.2550 | 0.0011339  |
| mlayerld:outdoors:row13 | 0.374546 | 0.132670 | 2.8231 | 0.0047555  |
| mlayerroct:outdoors:row13 | -0.179594 | 0.144162 | -1.2458 | 0.2128475  |
Example Two (cont.)

Doesn’t all fit on a slide, but it can be looked at outside of the slides.

Anyway, we aren’t primarily or even secondarily interested in these “meaningless” parameter estimates.
Now what about yearcross?

```r
> outies <- grepl("yearcross", colnames(m))
> colnames(m)[outies]


> m2 <- m[, !outies]
> dim(m)

[1] 6127  31

> dim(m2)

[1] 6127  25
Example Two (cont.)

```r
> aout2 <- aster(resp ~ 0 + m2,
+ pred, fam, varb, id, root, data = echin2)
> anova(aout2, aout)

Analysis of Deviance Table

Model 1: resp ~ 0 + m2
Model 2: resp ~ 0 + m

|       | Model Df | Model Dev | Df | Deviance | P(>|Chi|) |
|-------|----------|-----------|----|----------|----------|
| 1     | 23       | -2082.5   |    |          |          |
| 2     | 26       | -2079.0   | 3  | 3.4784   | 0.3236   |
```
yearcross is not statistically significant. Not even close.
Why only 3 degrees of freedom, when we dropped 6 variables?

> ncol(m)

[1] 31

> length(aout$coefficients)

[1] 26

> aout$dropped

[1] "mlayerld:yearcross2000"
[4] "mindoors:flat3"
[5] "mfit:crosstypeWr"
Example Two (cont.)

```r
> ncol(m2)
[1] 25

> length(aout2$coefficients)
[1] 23

> aout2$dropped

[1] "m2indoors:flat3"  "m2fit:crosstypeWr"
```
Now what about crosstype?

```r
> outies <- grepl("crosstype", colnames(m2))
> colnames(m2)[outies]

[1] "fit:crosstypeBr" "fit:crosstypeWi"
[3] "fit:crosstypeWr"

> m3 <- m2[, !outies]
> dim(m2)

[1] 6127  25

> dim(m3)

[1] 6127  22
```
Example Two (cont.)

> aout3 <- aster(resp ~ 0 + m3, 
+      pred, fam, varb, id, root, data = echin2) 
> anova(aout3, aout2)

Analysis of Deviance Table

Model 1: resp ~ 0 + m3
Model 2: resp ~ 0 + m2

Model Df Model Dev Df Deviance P(>|Chi|)
1 21 -2108.7
2 23 -2082.5 2 26.197 2.048e-06 ***
---
Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
crosstype is highly statistically significant.

Summary: \texttt{aout2} is the best model (looked at so far).

Once we got on the right track, it wasn’t too hard.

However, we have not yet addressed another issue of scientific interest. Now that we are on the right track, perhaps it will be straightforward. (Actually, it is far from straightforward because there are interesting theoretical issues, more than one of them.)
One of the authors on the paper asked the question about whether differential survival in the growth chamber or differential survival and growth (fecundity wasn’t really measured) outdoors was more responsible for the statistical significance of crosstype.

To examine that, we have to add a term indoors : crosstype
Example Two (cont.)

```r
> m4 <- model.matrix(resp ~ varb + 
+   indoors : flat + outdoors : layer : (row + posi) + 
+   fit : crosstype + indoors : crosstype, data = echin2)
> outies <- grepl("row0\|layerlds\:outdoors\|row10", colnames(m4))
> colnames(m4)[outies]

[1] "outdoors:layerld:row0"
[2] "outdoors:layerlds:row0"
[3] "outdoors:layerroct:row0"
```
Example Two (cont.)

```r
> dim(m4)
[1] 6127 37

> m4 <- m4[, !outies]
> dim(m4)

[1] 6127 31
```
Example Two (cont.)

```r
> aout4 <- aster(resp ~ 0 + m4,
+     pred, fam, varb, id, root, data = echin2)
> anova(aout3, aout2, aout4)

Analysis of Deviance Table

Model 1: resp ~ 0 + m3
Model 2: resp ~ 0 + m2
Model 3: resp ~ 0 + m4

| Model | Df | Model Dev | Df | Deviance | P(>|Chi|) |
|-------|----|-----------|----|----------|-----------|
| 1     | 21 | -2108.7   |    |          |           |
| 2     | 23 | -2082.5   |  2 | 26.1971  | 2.048e-06 *** |
| 3     | 25 | -2082.1   |  2 | 0.3571   | 0.8365    |

---

Signif. codes:
0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
One of these $P$-values we have seen before (aout3 versus aout2). The other is new.

Since the new one is not statistically significant (not even close), it is clear that model aout4 fits no better than aout2. There is no evidence that we need the indoors : crosstype term in the model.
Example Two (cont.)

But there is another way to ask this question. Might it not be that a model with just indoors : crosstype instead of fit : crosstype would also fit the data?

But we cannot compare those models directly because they are not nested (and nested models are required for validity of the tests done by the anova function).

So we have to compare them indirectly by comparing each directly to aout3 and aout4
Example Two (cont.)

```r
> outies <- grepl("fit:crosstype", colnames(m4))
> colnames(m4)[outies]

[1] "fit:crosstypeBr"  "fit:crosstypeWi"
[3] "fit:crosstypeWr"

> m5 <- m4[, !outies]
> dim(m4)

[1] 6127    31

> dim(m5)

[1] 6127    28
```
Example Two (cont.)

```r
> aout5 <- aster(resp ~ 0 + m5,
+         pred, fam, varb, id, root, data = echin2)
> anova(aout3, aout5, aout4)

Analysis of Deviance Table

Model 1: resp ~ 0 + m3
Model 2: resp ~ 0 + m5
Model 3: resp ~ 0 + m4

| Model  | Df | Model Dev | Df | Deviance  | P(>|Chi|) |
|--------|----|-----------|----|-----------|----------|
| 1      | 21 | -2108.7   | 2  | 5.2469    | 0.07255  |
| 2      | 23 | -2103.4   | 2  | 21.3073   | 2.361e-05 ***|
| 3      | 25 | -2082.1   | 2  | 21.3073   | 2.361e-05 ***|

---

Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
The comparison of $a_{out5}$ with $a_{out2}$ is not statistically significant (although there is some weak indication).

The comparison of $a_{out5}$ with $a_{out4}$ is highly statistically significant.

Thus $a_{out5}$ fits no better than the smallest model under consideration ($a_{out2}$) and much worse than the largest ($a_{out4}$). Thus there is no (or very, very weak) evidence in favor of $a_{out5}$.

This is just the opposite of the situation with $a_{out2}$. Since $a_{out2}$ fits as well as $a_{out4}$ and is the smaller, more parsimonious model, we choose it.
So our scientific conclusion is that differential survival in the growth chamber has no effect on fitness. Right? Wrong!

That forgets the aster transform. Recall that its inverse is given by

$$\theta_k = \varphi_k + \sum_{\substack{j \in J \\mid p(j) = k}} c_j(\theta_j)$$

where we calculate successors before predecessors.

This implies that the components of $\varphi$ for all successors, successors of successors, and so forth influence the component of $\theta$ for that node and consequently the component of $\xi$ for that node.

Said another way, changing the component of $\varphi$ for one node changes the components of $\theta$ and $\xi$ for all predecessors, predecessors of predecessors, and so forth of that node.
So the term \texttt{fit : crosstype} propagates backwards in the graph to influence conditional mean values for survival in the growth chamber.

How much does it influence it? What the maximum entropy principle makes it do.

Which way does it influence it?
The derivative of the aster transform is given by

\[ \Delta \varphi_k = \Delta \theta_k - \sum_{j \in J} c'_j(\theta_j) \Delta \theta_j \]

\[ = \Delta \theta_k - \sum_{j \in J} \xi_j \Delta \theta_j \]

and the derivative of the inverse aster transform is given by

\[ \Delta \theta_k = \Delta \varphi_k + \sum_{j \in J} \xi_j \Delta \theta_j \]

which calculates components of \( \Delta \theta \) from components of \( \Delta \varphi \) in any order that does successors before predecessors.
Using

\[ \Delta \theta_k = \Delta \varphi_k + \sum_{j \in J} \xi_j \Delta \theta_j \]

on our example or on any aster model in which the \( \xi_j \) are always nonnegative (because the corresponding components of the response vector are always nonnegative), we see that increasing one component of \( \Delta \varphi \) also increases the corresponding component of \( \Delta \theta \) and then also increases the the components of \( \Delta \theta \) for predecessor nodes and predecessor of predecessor nodes and so forth.

In short, increasing fitness at terminal nodes (“fitness nodes”) of the graph also increases survival at earlier nodes of the graph, and in particular survival in the growth chamber.
But exactly how much? For that we need numbers.

Again we need to use the `newdata` argument to the `predict.aster` function, but we do not need to use the `amat`, because we are only interested in one node (`lds3`) rather than a sum of nodes.

Since we are using formulas but useless ones we will have to use the function `predict.aster.default` rather than `predict.aster.formula`. 
First we subset m2 to have one individual for each cross type.

```r
> inies <- match(levels(echin2$crosstype),
+ as.character(echin2$crosstype))
> inies <- echin2$id[inies]
> inies

[1]  5 21  1

> newmodmat <- m2[echin2$id %in% inies, ]
> dim(newmodmat)

[1] 33 25
```
And then we “fix it up”

```r
> grep("flat", colnames(newmodmat), value = TRUE)

[1] "indoors:flat1" "indoors:flat2" "indoors:flat3"

> newmodmat[, "indoors:flat1"] <-
+   as.numeric(grepl("lds", rownames(newmodmat)))
> newmodmat[, "indoors:flat2"] <- 0
> newmodmat[, "indoors:flat3"] <- 0
```
Example Two (cont.)

> grep("row", colnames(newmodmat), value = TRUE)


Put everyone in row 10.

> newmodmat[, grep("row", colnames(newmodmat))] <- 0
Example Two (cont.)

```r
> grep("posi", colnames(newmodmat), value = TRUE)


Put everyone at position zero.

> newmodmat[, grep("posi", colnames(newmodmat))] <- 0
```
Now more PBD. We have to make `newmodmat` a 3-way array and supply `root` as a 2-way array.

```r
> nind <- nlevels(echin2$crosstype)
> nparm <- ncol(newmodmat)
> nnode <- length(newmodmat) / (nind * nparm)
> all.equal(nnode, as.integer(nnode))

[1] TRUE
```

```r
> newmodmat <- array(as.vector(newmodmat),
+ c(nind, nnode, nparm))
> newroot <- array(1, c(nind, nnode))
```
Example Two (cont.)

```r
> try(pout <- predict(aout2, root = newroot,
+     modmat = newmodmat, se.fit = TRUE), silent = TRUE)
> dim(newmodmat)

[1]  3 11 25

> dim(aout2$modmat)

[1] 557 11 23

> aout2$dropped

[1] "m2indoors:flat3"  "m2fit:crosstypeWr"

doesn’t work. We have to remove the “dropped columns” from newmodmat.
Example Two (cont.)

```r
> droppies <- gsub("m2", "", aout2$dropped)
> droppies

[1] "indoors:flat3"    "fit:crosstypeWr"

> outies <- match(droppies, colnames(m2))
> outies

[1] 14 17

> newmodmat <- newmodmat[, , - outies]
> dim(newmodmat)

[1] 3 11 23
```
Try 2.

```r
> pout <- predict(aout2, root = newroot,
+     modmat = newmodmat, se.fit = TRUE)
> pout.fit <- matrix(pout$fit, nrow = nind)
> pout.se <- matrix(pout$se.fit, nrow = nind)
> rownames(pout.fit) <- levels(echin2$crosstype)
> rownames(pout.se) <- levels(echin2$crosstype)
> colnames(pout.fit) <- vars
> colnames(pout.se) <- vars
```
Example Two (cont.)

```r
> foo <- cbind(pout.fit[, "lds3"],
+    pout.se[, "lds3"])
> colnames(foo) <- c("fit", "se")
> foo

               fit       se
Br  0.8592819 0.03802385
Wi  0.8694220 0.03340710
Wr  0.7920395 0.04824145
```
At this point we give up (even though the paper had more confidence intervals).

Enough has been done to show that

- aster modeling when the R formula mini-language is inadequate to express the models is very difficult,
- but not impossible.