Preliminary Analysis on Lilium data

Gan Yao

2022-09-24

1. Some notes

This is just a preliminary analysis of Lilium data. Plenty of model design choices are still left to discuss.

Data inconsistency

There are five individuals in dataset with `flCt == 0` but `flNotConsumed == 1`.

```
data <- read.csv("data/output/remLilium2021Data.csv")
data[data$flCt == 0,c('id','flCt','flNotConsumed')]
```

```
##         id flCt flNotConsumed
## 10  LP1009    0             1
## 69  LP1218    0             1
## 160  LP197    0             0
## 594  LP882    0             1
## 622  LP911    0             1
## 624  LP913    0             1
```

Though the data may be correct as it reflects the scientific truth, it is not allowed in aster graph. So during the data cleaning section below, these individuals' `flNotConsumed` values are manually changed to zero.

Problem with `nn[k]DistNotConsumed`

Columns `nn[k]DistNotConsumed` are missing for individuals not harvested, and it makes no sense to replace these NAs with any numeric values. So these columns are ignored in this analysis since NAs are not allowed to appear in design matrix.

Missing 'fecundity'

The last node of previously proposed aster graph is 'fecundity'. However, it's missing in the dataset and hence is ignored in this analysis.

Additional node accounting for sub-sampling.

In order to exploit the entire data set(i.e. not just the harvested ones), a new node `isHarvested` is added to the aster graph.

`isHarvested` is a Bernoulli node placed between `capsuleCt` and `ovuleCt`, and takes values from column `nCapsulesHarvested`. It indicates whether a capsule is harvested or not.

So, considering all above modifications, the aster graph upon which this analysis is based should be

$$root \to flNotConsumed \to capsuleCt \to isHarvested \to ovuleCt \to SeedSet(embryoCt)$$

2. Load and clean data

Import libraries,

```
library(aster)
library(tidyverse)
```

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v ggplot2 3.3.6     v purrr   0.3.4
## v tibble  3.1.8     v dplyr   1.0.9
## v tidyr   1.2.0     v stringr 1.4.0
## v readr   2.1.2     v forcats 0.5.1
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

Load Data,

```
data <- read.csv("data/output/remLilium2021Data.csv")
names(data)
```

```
##  [1] "id"                  "site"                "year"
##  [4] "Ax"                  "Ly"                  "flCt"
##  [7] "flNotConsumed"       "flUndamaged"         "capsuleCt"
## [10] "nCapsulesHarvested"  "ovuleCt"             "embryoCt"
## [13] "nn1Dist"             "nn2Dist"             "nn3Dist"
## [16] "nn4Dist"             "nn5Dist"             "nn6Dist"
## [19] "nn7Dist"             "nn8Dist"             "nn9Dist"
## [22] "nn10Dist"            "nn1DistNotConsumed"  "nn2DistNotConsumed"
## [25] "nn3DistNotConsumed"  "nn4DistNotConsumed"  "nn5DistNotConsumed"
## [28] "nn6DistNotConsumed"  "nn7DistNotConsumed"  "nn8DistNotConsumed"
## [31] "nn9DistNotConsumed"  "nn10DistNotConsumed"
```

Look at `site` column,

```
data %>% count(site)
```

```
##   site   n
## 1 beng 238
## 2 hegg 103
## 3   lf   2
## 4 sppe  55
## 5 sppw 301
## 6 wrrx   9
```

We should remove rows with site value `lf` and `wrrx`, since they're too rare in this data set.

```
data <- data[data$site != "lf",]
data <- data[data$site != "wrrx",]
data %>% count(site)
```

```
##   site   n
## 1 beng 238
## 2 hegg 103
## 3 sppe  55
## 4 sppw 301
```

Now we handle the inconsistency between `flCt` and `flNotConsumed`,

```
data %>% count(flNotConsumed)
```

```
##   flNotConsumed   n
## 1             0 245
## 2             1 452
```

by manually changing the `flNotConsumed` values of these rows to zero.

```
data <- data %>% mutate(flNotConsumed = replace(flNotConsumed, flCt==0&flNotConsumed==1, 0))
data %>% count(flNotConsumed)
```

```
##   flNotConsumed   n
## 1             0 250
## 2             1 447
```

To add node `isHarvested`, all the NAs in column `nCapsulesHarvested` should be assigned to zeros.

```
sum(as.numeric(is.na(data$nCapsulesHarvested)))
```

```
## [1] 614
```

```
data[is.na(data$nCapsulesHarvested), 'nCapsulesHarvested'] <- 0
sum(as.numeric(is.na(data$nCapsulesHarvested)))
```

```
## [1] 0
```

```
sum(as.numeric(data$nCapsulesHarvested==0))
```

```
## [1] 614
```

Same for columns `ovuleCt` and `embryoCt`.

```
sum(as.numeric(is.na(data$ovuleCt)))
```

```
## [1] 614
```

```
sum(as.numeric(is.na(data$embryoCt)))
```

```
## [1] 614
```

```
data[is.na(data$ovuleCt), 'ovuleCt'] <- 0
data[is.na(data$embryoCt), 'embryoCt'] <- 0
sum(as.numeric(is.na(data$ovuleCt)))
```

```
## [1] 0
```

```
sum(as.numeric(is.na(data$embryoCt)))
```

```
## [1] 0
```

```
sum(as.numeric(data$ovuleCt==0))
```

```
## [1] 614
```

```
sum(as.numeric(data$embryoCt==0))
```

```
## [1] 614
```

3. Wide format to long format

To fit aster model with this dataset, we need to first transform it to so-called 'long format'.

```
data <- data %>% select("id", "site", "flCt", "flNotConsumed",
                                "capsuleCt", "nCapsulesHarvested",
                               "ovuleCt", "embryoCt","nn1Dist","nn2Dist",
                                "nn3Dist", "nn4Dist","nn5Dist",
                                "nn6Dist", "nn7Dist", "nn8Dist",
                                "nn9Dist", "nn10Dist")
names(data)[names(data) == 'nCapsulesHarvested'] <- 'isHarvested'
vars <- c("flCt", "flNotConsumed", "capsuleCt", "isHarvested", "ovuleCt", "embryoCt")
redata <- reshape(data, varying = list(vars), direction="long",
                  timevar="varb", times = as.factor(vars),
                  v.names="resp")
redata <- data.frame(redata, root = 1)
names(redata)
```

```
##  [1] "id"       "site"     "nn1Dist"  "nn2Dist"  "nn3Dist"  "nn4Dist"
##  [7] "nn5Dist"  "nn6Dist"  "nn7Dist"  "nn8Dist"  "nn9Dist"  "nn10Dist"
## [13] "varb"     "resp"     "root"
```

Last step of preparing data, is to add one column that indicates the best surrogate of fitness in data, in this case, the `embryoCt` level of `varb`.

```
redata$fit <- as.numeric(redata$varb == "embryoCt")
names(redata)
```

```
##  [1] "id"       "site"     "nn1Dist"  "nn2Dist"  "nn3Dist"  "nn4Dist"
##  [7] "nn5Dist"  "nn6Dist"  "nn7Dist"  "nn8Dist"  "nn9Dist"  "nn10Dist"
## [13] "varb"     "resp"     "root"     "fit"
```

4. Fixed effects aster model

Let's first fit a big model.

```
pred <- c(0,1,2,3,4,5)
fam <- c(2,1,2,1,2,1)
model1 <- aster(resp ~ varb + fit:(site + nn1Dist + nn2Dist + nn3Dist + nn4Dist + nn5Dist
                                   + nn6Dist + nn7Dist + nn8Dist + nn9Dist + nn10Dist),
                pred, fam,varb,id,root,data=redata)
summary(model1)
```

```
## Error: cannot compute standard errors, apparent directions of recession
```

Unfortunately, the `summary()` function will complain about apparent directions of recession, indicating nearly singular fisher information matrix.

Now let's fit a smaller model and try to locate the problem in our data.

```
model2 <- aster(resp ~ varb + fit:(site), pred, fam,varb,id,root,data=redata)
summary(model2)
```

```
## Error: cannot compute standard errors, apparent directions of recession
```

The error is still there. So we have to add `info.tol` argument and make it print out the summary anyway.

```
summary(model2, info.tol = 1e-9)
```

```
##
## Call:
## aster.formula(formula = resp ~ varb + fit:(site), pred = pred,
```

```
##     fam = fam, varvar = varb, idvar = id, root = root, data = redata)
##
##                      Estimate Std. Error  z value Pr(>|z|)
## (Intercept)       -1.672e+00  1.543e-01  -10.834  < 2e-16 ***
## varbembryoCt       1.091e+00  1.548e-01    7.052 1.77e-12 ***
## varbflCt           1.635e+00  1.637e-01    9.991  < 2e-16 ***
## varbflNotConsumed  2.664e+00  1.848e-01   14.417  < 2e-16 ***
## varbisHarvested   -3.196e+02  1.873e+00 -170.636  < 2e-16 ***
## varbovuleCt        7.699e+00  1.545e-01   49.840  < 2e-16 ***
## fit:sitebeng      -5.894e-03  2.365e-03   -2.492   0.0127 *
## fit:sitehegg      -2.186e-02  8.757e-03   -2.497   0.0125 *
## fit:sitesppe       8.825e-03  1.642e-03    5.375 7.64e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Original predictor variables dropped (aliased)
##      fit:sitesppw
```

Notice that the magnitude of estimated coefficient of `varbisHarvested` is abnormally large. This variable corresponds to the node we added to the graph for taking care of sub-sampling. This result may indicate that the way we are now handling the sub-sampling effect is incorrect.

5. Random effects aster model

Now let's fit the full random effects aster model, and see if the same problem pops out.

```
redata$Nid <- as.numeric(gsub("[^0-9.-]", "", redata$id))
remodel1 <- reaster(resp ~ varb + fit:(nn1Dist + nn2Dist + nn3Dist + nn4Dist + nn5Dist
                                       + nn6Dist + nn7Dist + nn8Dist + nn9Dist + nn10Dist),
                    list(site = ~0 + fit:site),
                   pred, fam,varb,Nid,root,data=redata)
summary(remodel1)
```

```
##
## Call:
## reaster.formula(fixed = resp ~ varb + fit:(nn1Dist + nn2Dist +
##     nn3Dist + nn4Dist + nn5Dist + nn6Dist + nn7Dist + nn8Dist +
##     nn9Dist + nn10Dist), random = list(site = ~0 + fit:site),
##     pred = pred, fam = fam, varvar = varb, idvar = Nid, root = root,
##     data = redata)
##
##
## Fixed Effects:
##                      Estimate Std. Error  z value Pr(>|z|)
## (Intercept)       -1.672e+00  1.543e-01  -10.834  < 2e-16 ***
## varbembryoCt       1.087e+00  1.548e-01    7.021  2.2e-12 ***
## varbflCt           1.635e+00  1.637e-01    9.991  < 2e-16 ***
## varbflNotConsumed  2.665e+00  1.848e-01   14.426  < 2e-16 ***
## varbisHarvested   -3.196e+02  1.873e+00 -170.623  < 2e-16 ***
## varbovuleCt        7.699e+00  1.545e-01   49.839  < 2e-16 ***
## fit:nn1Dist        1.040e-04  1.464e-04    0.711    0.477
## fit:nn2Dist       -1.426e-04  1.679e-04   -0.850    0.395
## fit:nn3Dist       -1.298e-04  1.844e-04   -0.704    0.481
## fit:nn4Dist        8.359e-05  2.247e-04    0.372    0.710
## fit:nn5Dist       -1.766e-04  2.828e-04   -0.624    0.532
```

```
## fit:nn6Dist        4.525e-04  5.033e-04    0.899    0.369
## fit:nn7Dist        4.994e-04  8.084e-04    0.618    0.537
## fit:nn8Dist       -8.712e-04  9.352e-04   -0.932    0.352
## fit:nn9Dist        1.871e-04  5.539e-04    0.338    0.736
## fit:nn10Dist       4.960e-05  6.595e-05    0.752    0.452
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Square Roots of Variance Components (P-values are one-tailed):
##      Estimate Std. Error z value Pr(>|z|)/2
## site 0.008890   0.003829   2.322     0.0101 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Though the `summary()` functions did not complain this time, the problem of `varbisHarvested` remain the same. What's more, none of the `nn[k]Dist` features has significant contribution to this model.

In fact, if we fit a null model,

```
remodel.null <- reaster(resp ~ varb, list(site = ~0 + fit:site),pred, fam,varb,Nid,root,data=redata)
summary(remodel.null)
```

```
##
## Call:
## reaster.formula(fixed = resp ~ varb, random = list(site = ~0 +
##     fit:site), pred = pred, fam = fam, varvar = varb, idvar = Nid,
##     root = root, data = redata)
##
##
## Fixed Effects:
##                     Estimate Std. Error  z value Pr(>|z|)
## (Intercept)          -1.6717     0.1543  -10.834  < 2e-16 ***
## varbembryoCt          1.0887     0.1548    7.031 2.05e-12 ***
## varbflCt              1.6352     0.1637    9.991  < 2e-16 ***
## varbflNotConsumed     2.6625     0.1848   14.409  < 2e-16 ***
## varbisHarvested    -319.6815     1.8734 -170.646  < 2e-16 ***
## varbovuleCt           7.6986     0.1545   49.840  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Square Roots of Variance Components (P-values are one-tailed):
##      Estimate Std. Error z value Pr(>|z|)/2
## site 0.008953   0.003798   2.357     0.0092 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

And compare the deviance of these two random effects model,

```
deviance(remodel1)
```

```
## [1] -250369.1
```

```
deviance(remodel.null)
```

```
## [1] -250356.5
```

```
deviance(remodel.null) - deviance(remodel1)
```

```
## [1] 12.65028
```

we can find that the difference in deviance is only 12.65028, while the difference in number of parameters is 10. Therefore, if we use the AIC as model selection criterion, we will end up picking the null model as our true model.

6. Discussion

To conclude, the analysis result so far is not satisfying. One major issue now seems to be the sub-sampling procedure. We are now treating it as a Bernoulli node. A potential problem with this approach is that by considering sub-sampling as Bernoulli trials, we assume there's a fixed probability or proportion(i.e. parameter θ of Bernoulli distribution) that the sub-sampling follows. However this may be not be the case when the sub-sampling was conducted, which will make the θ correspond to nothing in reality. But we have not come up with other solution for now, so looking forward to further discussion!