SAGA tutorial

Heath Blackmon and Jeffery Demuth

last updated: 2015-11-14

Contents

Introduction

Line cross analysis (LCA) or partitioning the contribution of composite genetic effects (CGEs) to the mean phenotype of cohorts is widely used to investigate the genetic architecture of traits. This approach uses two parental strains which have diverged in a phenotype of interest. These parents are crossed, producing an F1, and subsequent crosses (e.g. F2, backcross, reciprocals) are made to generate groups that have different combinations of parental genes. We refer to each of these groups as cohorts. Using a weighted least squares regression with weights inversely proportional to the variance of the cohort means, the degree to which a phenotype is determined by different CGEs (e.g. additive, dominance, and epistatic gene action) may be estimated [1, 2]. Traditionally LCA has been accomplished by a process referred to as the joint-scaling test, essentially forward variable selection weighted least squares regression. However this approach has a number of documented problems [3]. A full information-theoretic (I-T) approach to model selection and parameter estimation alleviates difficulties associated with previous approaches and provides additional understanding that is not possible under older approaches such as the joint-scaling test [4]. SAGA provides a full I-T approach to LCA that leverages the finite sample size corrected version of the Akaike information criterion

($AICc$) to explore all possible models and make unbiased and, when appropriate, model averaged estimates of the contribution of CGEs to cohort means. SAGA includes eight functions and seven empirical datasets.

Functions:

- **AnalyzeCrossesMM**: Primary function that generates and tests all models of genetic architecture possible for a given set of cohorts.
- **EvaluateModel**: Returns parameter estimates conditional on a single model.
- **VisModelSpace**: Plots the distribution of Akaike weights across model space.
- **DisplayCmatrix**: Loads the default c-matrix as a dataframe.
- **plotObserved**: provides a traditional plot of observed line cross analysis data as well as the expectation under an additive model.
- **plot.genarch**: An S3 method for plotting genarch objects. Can be used to produce publication quality plots with control over most aspects of the plot including the CGEs included.
- **cohortID**: Loads a dataframe with cohort IDs and descriptions. Allowing users to look up their own cohorts and prepare their data for analysis.
- **AICtoMW**: Converts a vector of ΔAIC or $\Delta AICc$ values to model weights.

Data:

- **ban.osa**: Number of offspring from crosses involving *Tribolium castaneum* from Ecuador and Japan [5]
- **dar.bho**: Number of offspring from crosses involving *Tribolium castaneum* from Tanzania and India [5]
- **per.inf**: Number of offspring from crosses involving *Tribolium castaneum* from Peru and Portugal [5]
- **sin.cro**: Number of offspring from crosses involving *Tribolium castaneum* from Malaysia and Croatia [5]
- **PH**: Height data for crosses involving strains of *Nicotiana rustica* [2].
- **SL**: Sperm length data for crosses between disjunct populations of *Drosophila mojavensis* [6].
- **SR**: Sperm receptacle length data for crosses between disjunct populations of *Drosophila mojavensis* [6].

Mathematical Approach

We use the function `GLM` from the base R package to perform weighted least square regression [7]. `GLM` returns the parameter and standard error estimates conditional on the model as well as the AIC value for the model. We convert AIC to $AICc$ using equation 1. Where n is the number of cohorts and K is the number of parameters being estimated.

Equation 1:

$$AICc = AIC + \frac{2K(K+1)}{n-K-1}$$

We then calculate $AICc$ differences ($\Delta AICc$) using equation 2.

Equation 2:

$$\Delta AICc_i = AICc_i - AICc_{min}$$

Where $\Delta AICc_min$ is the minimum $AICc$ score calculated across all possible models and $AICc_i$ is the $AICc$ calculated for a specific model. $\Delta AICc$ is used in generating Akaike weights (w_i) using equation 3. The denominator in this equation is the summation of the numerator across all possible models being evaluated (R).

Equation 3:

$$w_i = \frac{e^{-0.5 \times \Delta AICc_i}}{\sum_{r=1}^{R} e^{-0.5 \times \Delta AICc_r}}$$

Under the default settings, if w_i of the best model is 0.9 or greater then SAGA will perform parameter estimation under a single model. If no model reaches this threshold then we construct a 95% confidence set of models that contains the minimum number of models whose w_i sum to 0.95. To calculate model averaged parameter estimates and unconditional standard errors we recalculate w_i for each model performing the summation in the denominator of equation 3 across all models in the confidence set. The model weighted parameter estimates are then calculated using equation 4 where w_i is the recalculated model weight and $\hat{\theta}_i$ is the parameter estimate from the model; the product of these values is summed across all models R in the confidence set.

Equation 4:

$$\hat{\bar{\theta}} = \sum_{i=1}^{R} w_i \times \hat{\theta}_i$$

Standard error estimates that are unconditional on any one model are calculated using equation 5.

Equation 5:

$$\hat{se}\left(\hat{\bar{\theta}}\right) = \sum_{i=1}^{R} w_i \sqrt{\hat{var}\left(\hat{\theta}_i | g_i\right) + \left(\hat{\theta}_i - \hat{\bar{\theta}}\right)^2}$$

Finally variable importance v_i is calculated by summing w_i of all models in the confidence set R in which a CGE occurs (Eq. 6).

Equation 6:

$$v_i = \sum_{i=1}^{R} w_i$$

Installation

A stable tested version of SAGA is available from the CRAN repository or the most recent version may be installed from github using the devtools package:

Installing from CRAN

```
install.packages("SAGA")
```

Installing from github

```
library(devtools)
install_github("coleoguy/SAGA", build_vignettes = TRUE)
```

Fitting LCA models

Matrix of composite genetic effects

The supplied c-matrix of composite genetic effects

The first step in analysis of line cross data is choice of a c-matrix that describes the expected contribution of different types of gene action to mean cohort phenotypes. By default SAGA will use a c-matrix that is designed for a species with XY sex determination and scaled to the midparent mean (equivalent to F_∞), and includes 23 potential CGEs. For each CGE it includes coefficients for 24 potential crosses; each of which is divided into male, female, or mixed sex cohorts. The c-matrix has 72 rows and the row numbers are used to identify the cohorts being used in an experiment. The function DisplayCmatrix is available so you can decide if the supplied version has all of the CGEs and cohorts you are interested in.

```
# print the c-matrix to the terminal
DisplayCmatrix(table = "XY")
```

Table 1. The first 15 rows and 13 columns of the c-matrix supplied for species with XY sex determination.

	X.sire.x.dam.	ID	M	Aa	Ad	Xa	Xd	Ya	Ca	Mea	Med	AaAa	AaAd
1	P1:daughters	1	1	1	0.0	1.00	0.00	0.0	1	1	0	1	0
2	P1:sons	2	1	1	0.0	1.00	0.00	1.0	1	1	0	1	0
3	P1:mixed	3	1	1	0.0	1.00	0.00	0.5	1	1	0	1	0
4	P2:daughters	4	1	-1	0.0	-1.00	0.00	0.0	-1	-1	0	1	0
5	P2:sons	5	1	-1	0.0	-1.00	0.00	-1.0	-1	-1	0	1	0
6	P2:mixed	6	1	-1	0.0	-1.00	0.00	-0.5	-1	-1	0	1	0
7	F1 (P2xP1):daughters	7	1	0	1.0	0.00	1.00	0.0	1	1	0	0	0
8	F1 (P2xP1):sons	8	1	0	1.0	1.00	0.00	-1.0	1	1	0	0	0
9	F1 (P2xP1):mixed	9	1	0	1.0	0.50	0.50	-0.5	1	1	0	0	0
10	rF1 (P1xP2):daughters	10	1	0	1.0	0.00	1.00	0.0	-1	-1	0	0	0
11	rF1 (P1xP2):sons	11	1	0	1.0	-1.00	0.00	1.0	-1	-1	0	0	0
12	rF1 (P1xP2):mixed	12	1	0	1.0	-0.50	0.50	0.5	-1	-1	0	0	0
13	F2a (F1xF1):daughters	13	1	0	0.5	0.50	0.50	0.0	1	0	1	0	0
14	F2a (F1xF1):sons	14	1	0	0.5	0.00	0.00	-1.0	1	0	1	0	0
15	F2a (F1xF1):mixed	15	1	0	0.5	0.25	0.25	-0.5	1	0	1	0	0

Editing the c-matrix

Some studies might need a different c-matrix than what is supplied with SAGA. One easy way to handle this is to use one of the supplied c-matrices and edit it. First we will load the XY c-matrix:

```
NEW.cmat <- SAGA::DisplayCmatrix(table = "XY")
```

Now lets print the CGEs so we can determine which ones we need:

```
colnames(NEW.cmat)
```

```
##  [1] "M"    "Aa"   "Ad"   "Xa"   "Xd"   "Ya"   "Ca"   "Mea"  "Med"  "AaAa"
## [11] "AaAd" "AdAd" "XaAa" "XaAd" "XdAa" "XdAd" "YaAa" "YaAd" "YaXa" "CaAa"
## [21] "CaAd" "CaXa" "CaXd" "CaYa"
```

We will reduce the c-matrix to only CGEs that do not have sex chromosomes in them:

```
NEW.cmat <- NEW.cmat[, c(1:3,6:12,20:21)]
```

Now we have a c-matrix appropriate for an ESD species that can be used as normal:

```
AnalyzeCrossesMM(data, Cmatrix=NEW.cmat)
```

Prepare LCA Data for input

Data that will be analyzed with SAGA should be in a dataframe with three columns:

- id of the cohort which should match the appropriate row number of the c-matrix you are using
- mean phenotype measure of the cohort
- standard error of the cohort's mean phenotype.

SAGA comes with several empirical datasets already appropriately formatted. To illustrate the format we can load data on the number offspring produced by crosses involving *Tribolium castaneum* from Tanzania and India [6].

```
data(per.inf, package="SAGA")
```

Table 2. per.inf data illustrating the format required for analysis with SAGA.

Cohort ID	Mean	SE
3	33.62500	2.31407
6	42.50000	4.31774
9	44.80000	6.93830
12	37.25000	8.22977
15	23.85714	4.52205
21	25.85714	2.88203
18	33.25000	5.46008
24	24.12500	3.28110
27	35.12500	6.18303
30	54.66667	6.55574
33	43.50000	7.23303
36	43.12500	5.65824
45	19.20000	4.16413
48	13.00000	2.94958
39	47.66667	11.06245
42	47.66667	11.20020

To look up your cohorts and determine what id numbers to use you can use the function `cohortID` this function will return a data frame with the ID number for each type of cohort. Parents are listed in parentheses and are in the order sire x dam. Remember that these numbers only apply to the supplied c-matrix. If you alter it or supply your own then your cohort ID numbers should match the appropriate rows of your own c-matrix.

```
cohortID()
```

Table 3. cohort ID lookup table.

ID	cohorts	ID	cohorts
1	P1:daughters	37	BC2a (P2xF1):daughters
2	P1:sons	38	BC2a (P2xF1):sons
3	P1:mixed	39	BC2a (P2xF1):mixed
4	P2:daughters	40	BC2b (P2xrF1):daughters
5	P2:sons	41	BC2b (P2xrF1):sons
6	P2:mixed	42	BC2b (P2xrF1):mixed
7	F1 (P2xP1):daughters	43	rBC2a(F1xP2):daughters
8	F1 (P2xP1):sons	44	rBC2a(F1xP2):sons
9	F1 (P2xP1):mixed	45	rBC2a(F1xP2):mixed
10	rF1 (P1xP2):daughters	46	rBC2b(rF1xP2):daughters
11	rF1 (P1xP2):sons	47	rBC2b(rF1xP2):sons
12	rF1 (P1xP2):mixed	48	rBC2b(rF1xP2):mixed
13	F2a (F1xF1):daughters	49	2BC1b(BC1bxP1):daughters
14	F2a (F1xF1):sons	50	2BC1b(BC1bxP1):sons
15	F2a (F1xF1):mixed	51	2BC1b(BC1bxP1):mixed
16	F2b (rF1xF1):daughters	52	2BC2a(BC2axP2):daughters
17	F2b (rF1xF1):sons	53	2BC2a(BC2axP2):sons
18	F2b (rF1xF1):mixed	54	2BC2a(BC2axP2):mixed
19	rF2b (F1xrF1):daughters	55	F1p(pooled)rF1&F1:daughters
20	rF2b (F1xrF1):sons	56	F1p(pooled)rF1&F1:sons
21	rF2b (F1xrF1):mixed	57	F1p(pooled)rF1&F1:mixed
22	F2c (rF1xrF1):daughters	58	rBC1ap(P1xF1p):daughters
23	F2c (rF1xrF1):sons	59	rBC1ap(P1xF1p):sons
24	F2c (rF1xrF1):mixed	60	rBC1ap(P1xF1p):mixed
25	BC1a (F1xP1):daughters	61	BC1ap (F1pxP1):daughters
26	BC1a (F1xP1):sons	62	BC1ap (F1pxP1):sons
27	BC1a (F1xP1):mixed	63	BC1ap (F1pxP1):mixed
28	BC1b (rF1xP1):daughters	64	rBC2ap(P2xF1p):daughters
29	BC1b (rF1xP1):sons	65	rBC2ap(P2xF1p):sons
30	BC1b (rF1xP1):mixed	66	rBC2ap(P2xF1p):mixed
31	rBC1a(P1xF1):daughters	67	BC2ap (F1pxP2):daughters
32	rBC1a(P1xF1):sons	68	BC2ap (F1pxP2):sons
33	rBC1a(P1xF1):mixed	69	BC2ap (F1pxP2):mixed
34	rBC1b(P1xrF1):daughters	70	2rBC2a(rBC2axP2):daughters
35	rBC1b(P1xrF1):sons	71	2rBC2a(rBC2axP2):sons
36	rBC1b(P1xrF1):mixed	72	2rBC2a(rBC2axP2):mixed

Analyze Models

Once data is prepared as above you perform the analysis with the function `AnalyzeCrossesMM`. This function has a variety of arguments to allow fine control of the analysis:

General Arguments

- **data:** a data frame with the first three columns:

  1) id of the cohort this must mach the coefficient row of the c-matrix
  2) mean phenotype measure of the cohort
  3) Standard error of the cohort's mean phenotype

- **Cmatrix:** A text string used to select the c-matrix to be used in the analysis. Included options are "XY", "XO", "ZW", "ZO", or "esd". A user matrix can also be supplied.
- **model.sum:** This is the sum of the probability of the models to be included in the confidence set.
- **even.sex:** A logical by default it is false. It should be set as true if either sexed cohorts are included or if mixed sex cohorts are included but have equal numbers of males and females.

Arguments to help with large datasets

- **max.pars:** Optional parameter limiting the size of the equations evaluated.
- **max.models:** The maximum number of fitted models to return from the function. This is included as an option to allow analysis of large model space on computers with limited RAM. This argument only impacts the number of models stored in the returned genarch object. Internally all models are still fit and the results are based on all models and not the subset of models returned.

Arguments to control plotting

- **graph:** Logical indicating whether a plot of results should be produced.
- **cex.axis:** expansion factor for numeric axis labels.
- **cex.names:** expansion factor for name labels.
- **cex.main:** expansion factor for main title.

This will return a list of the class "genarch". The list has four elements:

- **models:** a list containing the weighted least squares solution for all models tested.
- **estimates:** a data frame containing Model Weighted Average for each parameter and its unconditional standard error.
- **daicc:** a vector of the $\Delta AICc$ scores for all models tested.
- **varimp:** a data frame containing the v_i scores for composite effects

As SAGA is analyzing the data it will print the composite effects being tested as well as progress in analyzing models to the terminal, and by default a plot of the primary results of the analysis.

Mixed sex cohorts

The argument `even.sex` in `AnalyzeCrossesMM` is important to consider. This is set as `FALSE` by default. The builds in the assumption that neither sex specific cohorts or cohorts with an equal number of males and females measured. Lacking either single sex cohorts or at least even sexed cohorts will greatly reduce the number of CGEs that can be investigated. This argument is also only used when the supplied c-matrix is being used. If you are using a custom c-matrix you will need to insure that you have calculated coefficients that match your cohorts with regard to sex or ratio of sexes.

The size of model space

Using the default settings SAGA will evaluate every possible model that has fewer parameters than the number of cohorts included. In almost all studies we can evaluate every possible model given the cohorts available. However, in cases where many different types of sexed cohorts (i.e. more than 15) are available and the species have sex chromosomes the number of models can become prohibitive. One solution to this is to use the `max.pars` argument. This will limit the size of the equations evaluated. Mixed model analyses are relatively robust to the exclusion of models with low probability. Because of this we recommend that you begin with an intermediate value like 6. This will allow SAGA to analyze every possible equation with 6 or fewer parameters. Next begin increasing the `max.pars` argument, if the inferred architecture is stable as you increase it from 7, 8, 9, 10 and the number of models included in the confidence set is not increasing then it means that none of these additional models are better than simpler models already analyzed. In this situation we believe that the answer based on all equations below with less than 10 parameters is likely reliable. We have found no empirical datasets that place high support on equations with greater than 6 CGEs.

Example 1

In this example none of the models tested has a w_i greater than 95%. Figure 1 displays the model averaged parameter estimation from equation 4, and unconditional standard errors calculated in equation 5 are indicated with whiskers on each bar. The colors of the bars reflects the v_i calculated in equation 6.:

```
# we will need the plotrix package for plotting
library(plotrix)
results <- SAGA::AnalyzeCrossesMM(per.inf, graph=T, cex.names=.8)
```

```
## The composite genetic effects that will be tested are:
##  Aa, Ad, Ca, Mea, Med, AaAa, AaAd, AdAd, CaAa, CaAd
##
## Generating Models..........
##  500
##  1000
## AICc weights were used to select the minimum number of models whose weights sum
## to greater than 95% this model set includes 219 model(s)
```

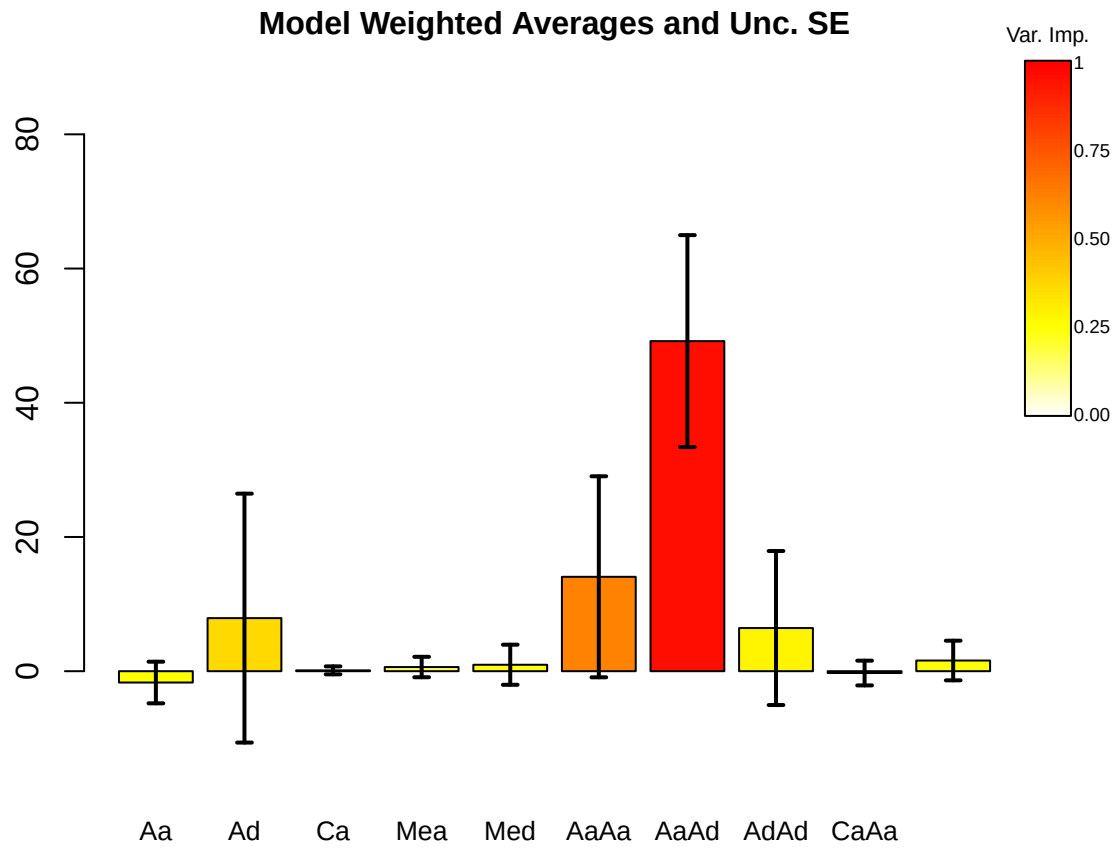

Figure 1: Model averaged estimate of genetic architecture.

Example 2

Now we can load a different dataset to demonstrate what happens when there is less model selection uncertainty. This dataset is from a study of sperm receptacle length measured in crosses between disjunct populations of *Drosophila mojavensis* [6].

```
#Sperm receptacle length in Drosophila mojavensis
data(SR)
#Because we are using cohorts where we know the distribution of sexes we set sexed=T.
AnalyzeCrossesMM(SR, sexed=T, graph=T)
```

```
library(plotrix)
data(SR, package="SAGA")
results2 <- SAGA::AnalyzeCrossesMM(SR, even.sex=T, graph=T, cex.names=.7)
```

```
## The following composite effects cannot be estimated with the line
## means available because they estimate identical quantities to
## lower order effects:
## Xd, XaAa, XdAa, XdAd, CaXd
##
## The composite genetic effects that will be tested are:
##  Aa, Ad, Xa, Ca, Mea, Med, AaAa, AaAd, AdAd, XaAd, CaAa, CaAd, CaXa
##
## Since there are 7098 possible models this may take a bit:
## Generating Models........
##  500
##  1000
##  1500
##  2000
##  2500
##  3000
##  3500
##  4000
##  4500
##  5000
##  5500
##  6000
##  6500
##  7000
## 1519 models were removed due to high covariances
## or linear relationships between predictor variables.
## The remaining 5579 models have been evaluated.
##
##
## AICc weights were used to select the minimum number of models whose weights sum
## to greater than 95% this model set includes 82 model(s)
```

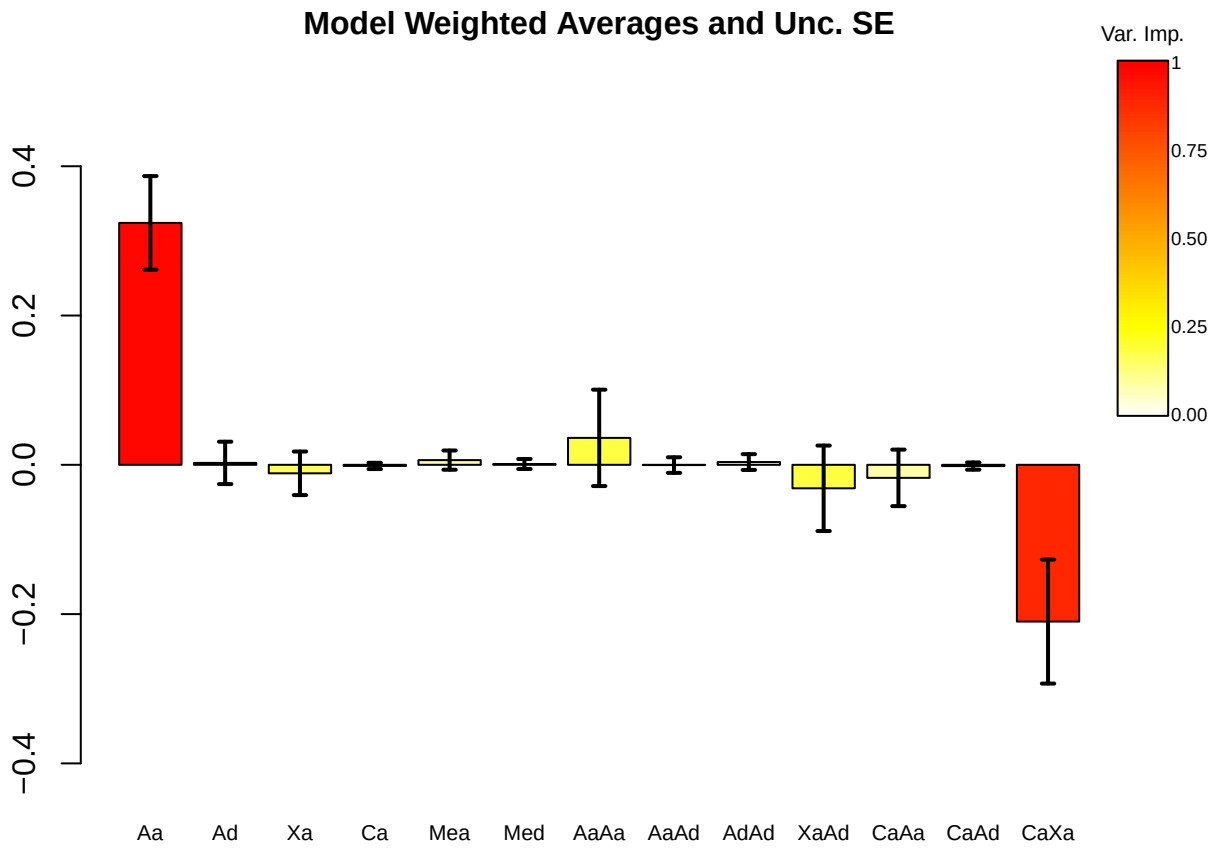

Figure 2: Unconditional estimate of genetic architecture.

Figure two illustrates that though no single model has a w_i sufficient to ignore model selection uncertainty the evidence ratio (w_i of the best model divided by the next best w_i) is over 2 for the best model and two CGEs exhibit v_i scores of much higher than any others. In this case autosomal additive and cytotype by X chromosome additive epistasis have v_i of .98 and .89 respectively. Additionaly as figure 2 shows no other CGEs have estimates that exclude zero, and if we look at the 10 best models we find that it is Aa and CaXa that are present in all of them. The other included CGEs explain very little of the variation and we would interpret only Aa and CaXa as being important in this trait.

Table 3 The ten best models the included CGEs and model weights.

model	CGEs	w_i
25	**Aa CaXa**	.23
154	**Aa CaXa** XaAd	.11
112	**Aa CaXa** Xa	.09
142	**Aa CaXa** AaAa	.07
129	**Aa CaXa** Mea	.03
102	**Aa CaXa** Ad	.03
147	**Aa CaXa** AaAd	.02
570	**Aa CaXa** XaAd AaAa	.02
121	**Aa CaXa** Ca	.02
156	**Aa CaXa** CaAa	.02

Plotting Results

To plot something differently than the default plot returned from the analysis; access the results stored in the genarch object. First we can simply look at what is stored by looking at the names of the list.

```
names(results)
```

```
## [1] "models"    "estimates" "daicc"     "varimp"
```

This shows us that we can access the parameter estimates in element two.

```
results[[2]]
```

```
##                              M                  Aa
## Model Weighted Average       "21.4607792647418" "-1.68700480520697"
## Unconditional Standard Error "12.9468262550248" "3.10140626583427"
##                              Ad                 Ca
## Model Weighted Average       "7.90842136260565" "0.123295299060826"
## Unconditional Standard Error "18.5483476254936" "0.595472269628727"
##                              Mea                Med
## Model Weighted Average       "0.61712250746008" "0.959584524461747"
## Unconditional Standard Error "1.52173246723402" "2.98767829428868"
##                              AaAa               AaAd
## Model Weighted Average       "14.0649194843012" "49.1857365590455"
## Unconditional Standard Error "14.9765251223035" "15.7915583830595"
```

```
##                                AdAd                CaAa
## Model Weighted Average       "6.43130864321397" "-0.276142119261707"
## Unconditional Standard Error "11.4777691366579" "1.84510559036607"
##                                CaAd
## Model Weighted Average       "1.59045906428731"
## Unconditional Standard Error "2.95532939628649"
```

We can also access the variable importances in the fourth element of the genarch object

```
results[[4]]
```

```
##       [,1]   [,2]
##  [1,] "Aa"   "0.258743993553868"
##  [2,] "Ad"   "0.371220175749559"
##  [3,] "Ca"   "0.150976324361842"
##  [4,] "Mea"  "0.171646401865819"
##  [5,] "Med"  "0.239313659551563"
##  [6,] "AaAa" "0.615806053631536"
##  [7,] "AaAd" "0.958558314566902"
##  [8,] "AdAd" "0.293109981553278"
##  [9,] "CaAa" "0.168600431505884"
## [10,] "CaAd" "0.252521439479819"
```

Lets look at how we might plot these using base R functions:

```
    # here we extract the 4 largest composite effects found in the first analysis
    estimates <- as.numeric(results[[2]][1, c(3, 7, 8, 9)])
    names(estimates) <- colnames(results[[2]])[c(3, 7, 8, 9)]
    barplot(estimates, main = "Estimate for composite effects",
            names.arg = names(estimates))
```

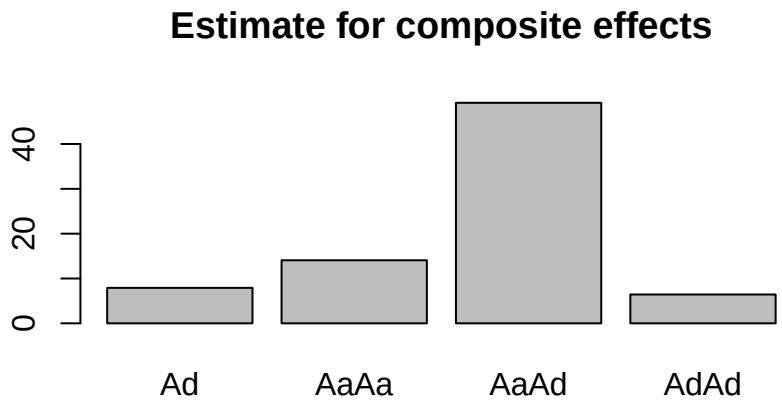

Figure 3: Subset of model averaged estimate of genetic architecture.

To alleviate the need to manually build a plot from scratch like above we have provided an S3 plot method for `genarch` objects. It has many useful built-in options such as restricting the plot to only include those CGEs that have a v_i score over a cutoff that you set. It also allows you to adjust the axis labels and color pallet to be used. For instance, here are 3 versions of the same results.

```
plot(results)
```

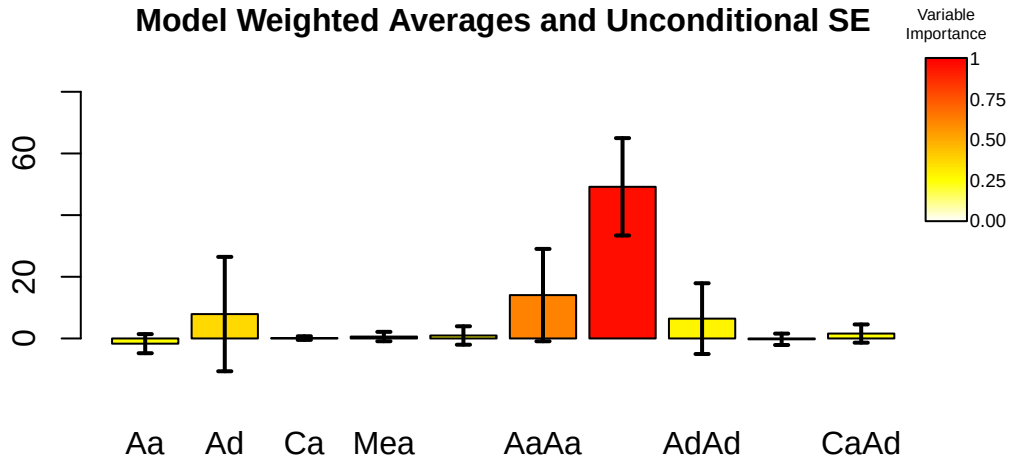

Figure 4: The standard full plot returned from the analysis

```
plot(results, min.vi=.25)
```

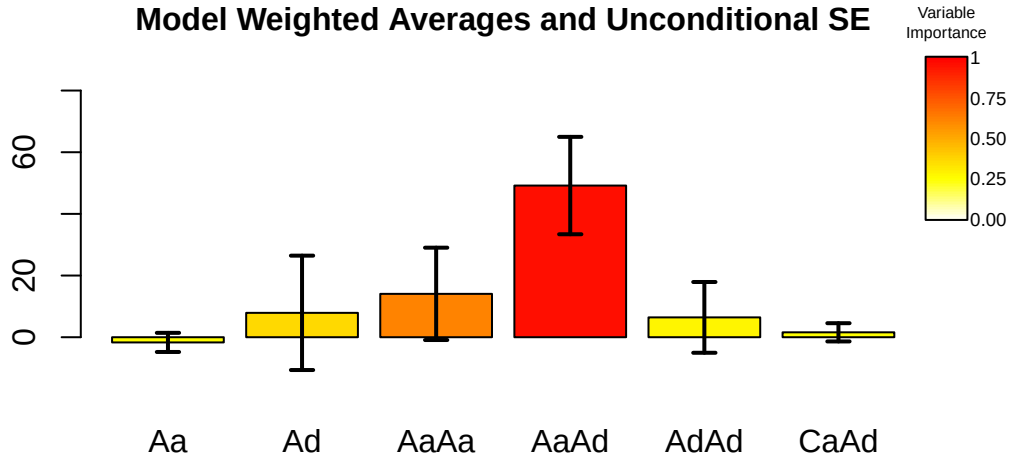

Figure 5: Now we have reduced the plot to include just those CGEs with a *vi* of at least .25

```
plot(results, maxval = 85, min.vi = .253, main = "A nicer plot", viridis = T)
```

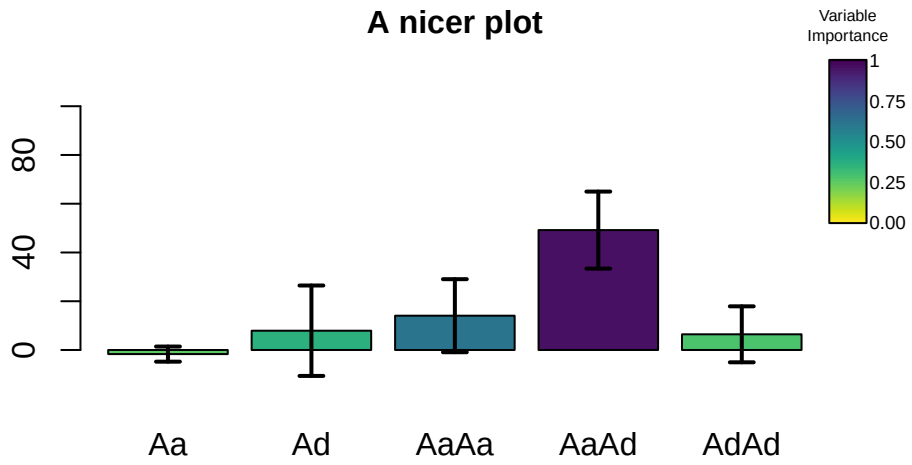

Figure 6: Now have switched to the viridis color pallete to make it color blind friendly, adjusted the Y axis, and changed the main title.

Finally, although few datasets support inference from any one model SAGA does provide the ability to investigate the results of individual models. For instance by accessing the $\Delta AICc$ scores saved in the third element of the genarch object we could find the best two models and then plot these using the function `EvaluateModel`. To illustrate this lets find the best two models from the *Tribolium* dataset in example 1 and plot them just to see how they differ.

```
# first lets find the best two models
order(results[[3]])[1:2]
```

```
## [1] 166 110
```

We plot the inference conditional on the best fit model below:

```
SAGA::EvaluateModel(results, 166, cex.names=.7, cex.main=.7)
```

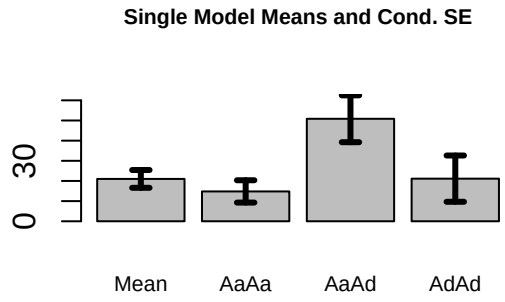

Figure 7: Estimates conditional on model 166

Next the evaluation of the second best model:

```
SAGA::EvaluateModel(results, 110, cex.names=.7, cex.main=.7)
```

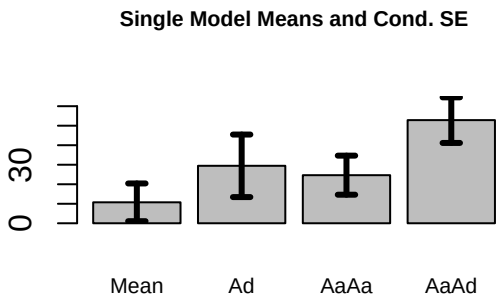

Figure 8: Estimates conditional on model 110

Here we can see that the top two models both include 3 composite genetic effects, and in both cases the strongest effect is assigned to autosomal additive by autosomal dominance epistasis. We can also see that the first model includes autosomal dominance by dominance epistasis while in the second this is replaced by simple autosomal dominance.

In LCA papers it is customary to plot the observed phenotype measure of each cohort as a function of percent genome of one parental line. These graphs are then overlayed with the expectation if the phenotype was purely additive. SAGA includes the plotting function `plotObserved` to produce these graphs. These plots have an X axis representing the amount of parent 1 genome and a Y axis representing the phenotype measure. They also include the expectation for the phenotype assuming a purely additive model. Currently this function has only been tested with the supplied c-matrices.

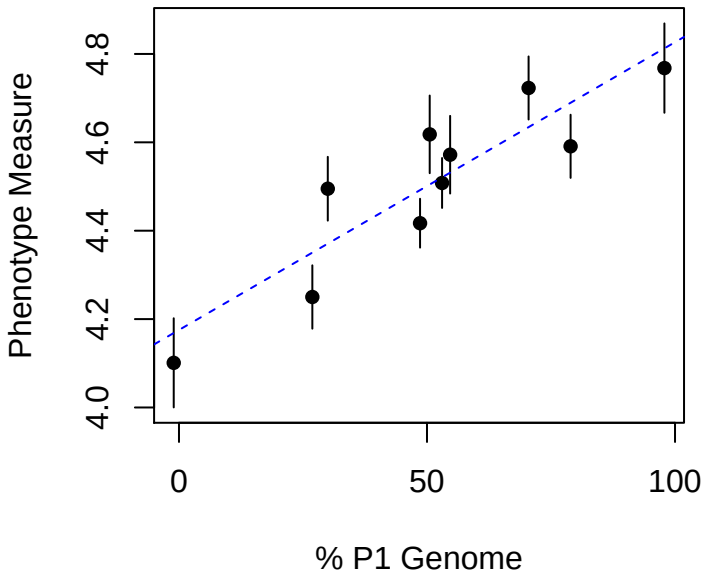

Figure 9: Observed line means.

Assesing model uncertainty

The relative fit of models to the data can be explored using the function 'VisModelSpace'. This function will plot a box for each model tested and will color it based on its w_i. The models are organized from the simplest model (autosomal additive) in the bottom left hand corner and increase in complexity from left to right (first all one parameter models then all two parameter models etc.) once the right hand side of the plot is reached a second row is added above the first. Only models evaluated and stored in the genarch object are plotted - for instance a model removed due to a colinearity will not be represented in the plot. To illustrate the differences in model space we can plot the results of the two analyses stored in `results` and `results2`. First lets look at the *Tribolium* analysis which indicated a nontrivial level of model selection uncertainty. The results from this analysis are stored in `results`

```
VisModelSpace(results, cex.u=1.6)
```

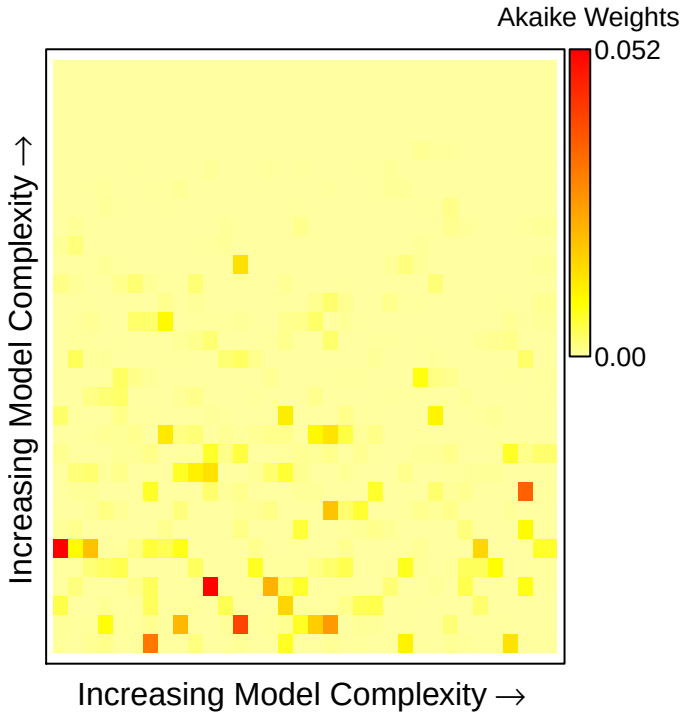

Figure 10: Distribution of akaike weights across model space for *Tribolium* dataset.

This plot shows us that their are a number of models of varying complexity that have very similar akaike weights, and this dataset highlights why our understanding of the genetic architecture should not be based on any single model.

Next lets create the same plot but this time for the *Drosophila* sperm receptacle length dataset which indicated very little model selection uncertainty. The results from our analysis of this dataset are stored in `results2`.

```
VisModelSpace(results2, cex.u=.4)
```

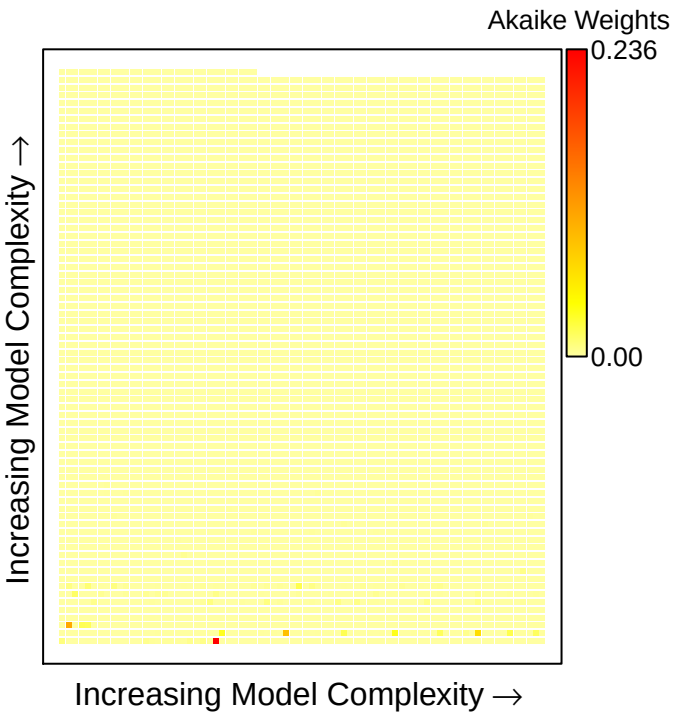

Figure 11: Distribution of akaike weights across model space for *Drosophila* sperm receptacle length.

Conclusion

Users should consider the presence of linear dependencies in c-matrices. Depending on your choice of cohorts some CGEs will likely be linearly dependent. This means that they can not be distinguished from each other and can not be estimated simultaneously. SAGA deals with this first by dropping any higher order CGEs that are perfectly correlated with a lower order effect from the c-matrix being used. However some combinations of CGEs may still be highly correlated (colinear) with each other. SAGA deals with this by dropping any model that includes the CGEs that are highly correlated. The importance and magnitude of each particular CGE involved in the colinearity is estimated only by the subset of models where it still appears. In our experience a strong signal of the importance of CGEs remains, but not surprisingly the parameter estimates become less accurate and the variable importance scores will be lower. When variables with high v_i scores do not appear jointly in the equations included in the confidence set, it is a strong indication that there is a colinearity, and warrants additional investigation. Often the only solution to this problem will be a careful examination of the c-matrix to determine what type of additional cohort(s) could be measured to most effectively demonstrate the difference in the contributions of the CGEs that are confounded.

Despite the improvements provided by implementing an I-T approach in SAGA, we would caution users that in our analysis of simulated datasets we have found that spurious variables are often included in models that are part of the confidence set, (i.e. table3). These spurious variables are likely included because they are

able to explain some stochastic noise in the dataset. However, these are usually easily identified by small parameter estimates with standard errors overlapping zero.

In reporting the results of line cross analysis experiments we recommend reporting estimates and standard errors from model averaged results unless a single model has greater than 95% w_i. It is also important to report v_i scores since these give an indication of our certainty that a particular composite genetic effect is important in the genetic architecture of the trait in question. Finally, although one of the benefits of our approach is a move away from strict arbitrarily defined p-values we have found that as a rule of thumb across all simulated datasets that we have analyzed v_i scores of greater than 50% have only been associated with CGEs that were included in the generating model.

Citations

[1] Mather, K., and J. L. Jinks, 1982 Biometrical genetics: The study of continuous variation. Chapman and Hall, London.

[2] Lynch, M., and B. Walsh, 1998 Lynch, M., & Walsh, B. (1998). Genetics and analysis of quantitative traits. Sinauer Associates, Inc., Sunderland, Massachusetts.

[3] Whittingham, M. J., P. A. Stephens, R. B. Bradbury and R. P. Freckleton, 2006 Why do we still use stepwise modelling in ecology and behaviour? J Anim Ecol 75: 1182-1189.

[4] Burnham, K. P., and D. R. Anderson, 2002 Model selection and multimodel inference: a practical information-theoretic approach. Springer, New York.

[5] Demuth, J. P., 2004 Evolution of Hybrid Incompatibility in the beetle Tribolium Castaneum, pp. 152 in Biology. Indiana University, Bloomington.

[6] Miller, G. T., Starmer, W. T., & S. Pitnick 2003. Quantitative genetic analysis of among-population variation in sperm and female sperm-storage organ length in Drosophila mojavensis. Genetical research, 81(03), 213-220.

[7] R Development Core Team, 2013 R: A Language and Environment for Statistical Computing, pp., Vienna, Austria.