

Supporting Data Analysis for
“An Application of Envelope Methodology and Aster Models”

By

Daniel J. Eck, Charles J. Geyer, and R. Dennis Cook

Technical Report No. 699

School of Statistics

University of Minnesota

March 23, 2016

In this technical report, we reproduce the examples that appear in Eck, Geyer, and Cook (2015) and show that the combination of general envelope methodology with aster models estimates expected Darwinian fitness with lower variability than estimation conducted with aster models alone. The `envlpaster` package (Eck, 2015) is developed to facilitate these calculations. This package implements the bootstrap algorithms discussed in the paper and contains the two datasets corresponding to the examples discussed in the paper. There are other functions in this package that we do not explain in this technical report. Those functions fall into two categories. They are either necessary for implementing the 1D algorithm or they implement other bootstrap techniques not discussed in Eck, Geyer, and Cook (2015). The other bootstrap techniques are explained in the help pages of the `envlpaster` package.

The bootstraps and model selection calculations in this technical report take substantial time to run. As a result, these calculations are not performed using this `knitr` script; they are performed in five `.RData` files and the code that produced these calculations is given in this technical report. The five `.RData` files are:

1. `ex1-Efronboot.RData`
2. `ex1-secondboot.RData`
3. `ex1-AIC.RData`
4. `techreport_eigenboot_fitboot.RData`
5. `main-all-terms-DavidLowrydata.RData`

Daniel J. Eck (eckxx049@umn.edu) is the current maintainer of these `.RData` files and they are available on request.

1 Replicating Example 1

A population of 3000 organisms was simulated to form the dataset used in this aster analysis. We generated this data to include a true envelope model. The code generating this dataset is provided in Appendix B. The last line of Appendix B shows that this code reproduces the dataset included in the `envlpaster` package. The organisms in this dataset either die or remain living for ten years. We observe three random variables for each year i of the study. These variables are U_i , V_i , and W_i . The variables U_i and V_i

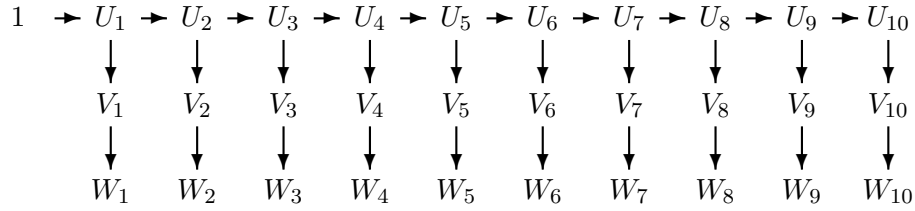


Figure 1: Graphical structure of the aster model for the simulated data in Section 5. The top layer corresponds to survival; these random variables are Bernoulli. The middle layer corresponds to whether or not an individual reproduced; these random variables are also Bernoulli. The bottom layer corresponds to offspring count; these random variables are zero-truncated Poisson.

are Bernoulli where U_i models survival and V_i models whether or not an individual reproduces conditional on survival. The variables W_i count offspring produced in the i -th year conditional on reproduction. Thus, W_i is modeled as zero-truncated Poisson given V_i . The graphical structure for one individual is given by the graph shown in Figure 1. In these data, the variable $\sum_i W_i$ is considered to be Darwinian fitness. There are two covariates (z_1, z_2) thought to be associated with Darwinian fitness and the aster model selected by the likelihood ratio test (LRT) is a full quadratic model with respect to these covariates. A full aster analysis of this data and its construction can be seen in Geyer and Shaw (2009). Note that our data differs from that in Geyer and Shaw (2009) since we have included a true envelope model.

The `envlpaster` and `aster2` packages are needed to carry out all calculations. The `envlpaster` packages requires `aster`, `MASS`, `caTools` and `trust` to be installed.

```

library(aster2)
library(envlpaster)
data(simdata30nodes)
data <- simdata30nodes.asterdata

```

The `simdata30nodes` dataset contains almost all that is necessary to fit the aster model used in the paper. The `root` and `pred` vectors and the model matrix `modmat` necessary for aster model fitting are included directly

in the `asterdata` object included in the `simdata30nodes` dataset within the `envlpaster` package.

```
vars <- variables
pred <- data$pred
families <- data$families
code <- data$code
```

The `fam` vector, which specifies the exponential families appearing in the graphical structure of an aster model, is not included but is constructed using the `code` vector. In this example, the `code` vector has entries that are 1-2 valued where a 1 corresponds to Bernoulli distributions and a 2 corresponds to zero-truncated Poisson distributions. In the `aster` library, a zero-truncated Poisson distribution is coded with the number 3 so we change the 2's that appear in `code` to 3's when we construct the `fam` vector.

```
fam <- code
fam[fam == 2] <- 3
```

The model fitting is not of class `aster.formula` and is fit using the aster data directly. The first `nnode` columns of the `simdata30nodes` dataset contain the responses necessary to fit the aster submodel of interest. Here `nnode = 30`. Recall that the full aster model is a saturated statistical model (one unknown parameter per observed response value) and is of little scientific use; the aster submodel has a reduced parameter count where the parameters of this model correspond to the relationship between Darwinian fitness with both phenotypic traits and covariates of interest. The different parameterizations of the aster model are seen in Figure 2. The relationship between the aster model parameterizations displayed in Figure 2 are further explained in Geyer (2010).

```
nnode <- length(vars)
xnew <- as.matrix(simdata30nodes[,c(1:nnode)])
m1 <- aster(xnew, root, pred, fam, modmat)
m1$formula <- formula
m1$xlevels <- xlevels
m1$terms <- terms
summary(m1)

##
```

```

## Call:
## NULL
##
##           Estimate Std. Error z value Pr(>|z|)
## (Intercept) -0.414198   0.029610 -13.989 < 2e-16 ***
## vtypev      1.240414   0.040284  30.792 < 2e-16 ***
## vtypew      0.464049   0.030958  14.990 < 2e-16 ***
## uyear       0.129734   0.004546  28.539 < 2e-16 ***
## z1          0.013889   0.003181   4.366 1.26e-05 ***
## z2         -0.010015   0.003197  -3.133 0.00173 **
## I(z1^2)     -0.015673   0.001915  -8.185 2.72e-16 ***
## I(z2^2)     -0.010950   0.001884  -5.813 6.14e-09 ***
## I(z1 * z2)  0.015817   0.002966   5.333 9.66e-08 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

```

There are nine components in both the submodel canonical parameter vector β and the submodel mean-value parameter vector τ . We write

$$\tau = \begin{pmatrix} \gamma \\ v \end{pmatrix} = M^T \mu$$

where $\gamma \in \mathbb{R}^{p-k}$ are nuisance parameters and $v \in \mathbb{R}^k$ are the parameters of interest that are relevant to the estimation of expected Darwinian fitness, M is the model matrix, and μ is the mean-value parameter vector of the unconditional aster model, see Figure 2. In our example, $p = 9$ and $k = 5$. The parameter vector v are the five unknown slope coefficients for the full quadratic model in terms of the covariates z_1 and z_2 . Ultimately, there are five parameterizations of interest that are present in the aster analyses we consider. These five parameterizations are given below and the formal relations among the first four parameterizations are shown in Figure 2. These parameterizations are:

- 1) The aster model canonical parameter vector $\beta \in \mathbb{R}^p$.
- 2) The aster model mean-value parameter vector $\tau \in \mathbb{R}^p$.
- 3) The unconditional aster model canonical parameter vector $\varphi \in \mathbb{R}^m$.
- 4) The unconditional aster model mean-value parameter vector $\mu \in \mathbb{R}^m$.

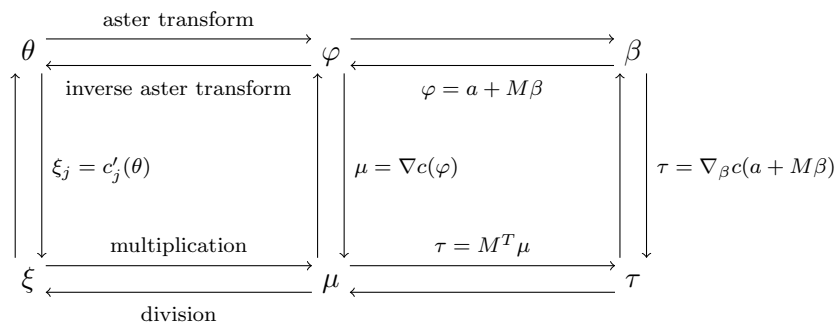


Figure 2: A depiction of the transformations necessary to change aster model parameterizations. Unlabeled arrows represent transformations that need to be solved using optimization software. M is a known model matrix, a is a known offset vector, and c is the cumulant function for the unconditional aster model.

- 5) The best surrogate of Darwinian fitness defined in Eck, Geyer, and Cook (2015) as $h(\mu) \in \mathbb{R}^d$ or $g(\tau) \in \mathbb{R}^d$ where h and g are a differentiable functions and d is the number of hypothetical individuals that expected Darwinian fitness is estimated for.

In this example, true Darwinian fitness is unknown. The best available surrogate of Darwinian fitness is total offspring count. When Darwinian fitness is taken to be total offspring count, the function $h(\mu) = A\mu$ where A is a matrix with entries $a_{i,j} = 1$ when μ_j corresponds to offspring for individual i . The matrix A specifies that expected Darwinian fitness for every individual is the sum of offspring nodes for that individual.

```
beta <- m1$coef; p <- length(beta)
target <- 5:9; m <- length(target)
x <- m1$x
nind <- n <- nrow(x)
```

We obtain the maximum likelihood estimator (MLE) of the submodel mean-value parameter, denoted $\hat{\tau}$ using `aster` software and the relations in Figure 2.

```

mu <- predict(m1, parm.type = "mean.value",
  model.type = "unconditional")
modmat.mat <- matrix(modmat, ncol = p)
tau <- crossprod(modmat.mat, mu)

```

We now transition from aster models to the combination of aster models and envelope methodology. The goal of this combination is to incorporate envelope methodology with respect to parameters of interest in the τ parameterization to yield efficiency gains when estimating expected Darwinian fitness. We first provide some envelope methodology basics relevant to aster modeling scenarios. The particular branch of envelope methodology relevant to aster problems is general envelope methodology discussed in Cook and Zhang (2014). The theory of general envelope methodology requires a \sqrt{n} consistent and asymptotically normal estimator of an unknown parameter vector and a \sqrt{n} consistent estimator of its asymptotic covariance matrix (Cook and Zhang, 2014). Aster models, being an exponential family, satisfy these conditions.

Let $T = \text{span}(v)$. The subspace T is a line in \mathbb{R}^k since the unknown parameter v is a point in \mathbb{R}^k . The envelope subspace $\mathcal{E}_{\Sigma_{v,v}}(T)$ is defined as the intersection of all reducing subspaces of $\Sigma_{v,v}$ that contains T . Reducing subspaces of $\Sigma_{v,v}$ are sums of eigenspaces of $\Sigma_{v,v}$ where the eigenspaces of $\Sigma_{v,v}$ are spans of the eigenvectors of $\Sigma_{v,v}$. An eigenvector x of $\Sigma_{v,v}$ is a solution to the equation $\Sigma_{v,v}x = \lambda x$ where λ is an eigenvalue needed to solve the equation. There are k eigenvectors and eigenvalues in total. Eigenspaces are defined to be the space spanned by all eigenvectors corresponding to one particular eigenvalue. When all of the eigenvalues are unique, all distinct reducing subspaces of $\Sigma_{v,v}$ can be specified by the 2^k unique combinations of spans of eigenvectors. The decomposition of a matrix into eigenvalues and eigenvectors is a general mathematical decomposition that has a statistical context when applied to a covariance matrix. When applied to a covariance matrix, such as $\Sigma_{v,v}$, this decomposition provides insight into the directions of variability (given by the eigenvectors) and the strength of variability in that direction (given by the eigenvalues).

The envelope subspace, as defined above to be the intersection of all reducing subspaces of $\Sigma_{v,v}$ that contain T , satisfies both

1. $T \subset \mathcal{E}_{\Sigma_{v,v}}(T)$
2. $\Sigma_{v,v} = P_{\mathcal{E}}\Sigma_{v,v}P_{\mathcal{E}} + Q_{\mathcal{E}}\Sigma_{v,v}Q_{\mathcal{E}}$

where \mathcal{E} denotes the envelope subspace when used as a subscript, $P_{\mathcal{E}}$ is the projection into $\mathcal{E}_{\Sigma_{v,v}}(\mathbb{T})$, and $Q_{\mathcal{E}}$ is the projection into the orthogonal complement of $\mathcal{E}_{\Sigma_{v,v}}(\mathbb{T})$. The reducing subspace assumption is needed for conditional 2 to hold. Intuitively, the envelope estimator reduces variability in estimation at no cost to consistency by making use of the conditions assumed of $\mathcal{E}_{\Sigma_{v,v}}(\mathbb{T})$. The first condition allows for consistent estimation to take place. The second condition states that only a part of the asymptotic covariance matrix $\Sigma_{v,v}$, seen as $P_{\mathcal{E}}\Sigma_{v,v}P_{\mathcal{E}}$, is variability associated with the estimation of expected Darwinian fitness. The other part, seen as $Q_{\mathcal{E}}\Sigma_{v,v}Q_{\mathcal{E}}$, is immaterial to our estimation problem and can be discarded.

When $P_{\mathcal{E}}$ is known, the envelope estimator of v is then $P_{\mathcal{E}}\hat{v}$, where

$$\sqrt{n}(P_{\mathcal{E}}\hat{v} - v) \xrightarrow{\mathcal{D}} N(0, P_{\mathcal{E}}\Sigma_{v,v}P_{\mathcal{E}}).$$

With condition 2 in mind, we can see that the envelope estimator of v has less asymptotic variability than the MLE when $P_{\mathcal{E}}$ is known. In actual problems, $P_{\mathcal{E}}$ is unknown and requires estimation. The `enlpaster` software can estimate $P_{\mathcal{E}}$ using the 1D algorithm proposed in Cook and Zhang (2014, Algorithm 2) and discussed in Eck, Geyer, and Cook (2015, Sections 3 and 4) or by using reducing subspaces directly as in Eck, Geyer, and Cook (2015, Section 4). The reducing subspace approach to envelope estimation is of particular interest in the current applications and we only motivate this approach in this technical report. See the references in this paragraph for details on the 1D algorithm, specifically the last paragraph of Eck, Geyer, and Cook (2015, Section 4).

In our applications, envelope estimators are constructed using eigenvectors of the estimated Fisher information matrix $\hat{\Sigma}_{v,v}$ which is the estimate of $\Sigma_{v,v}$ obtained from `aster` software. In this particular analysis, $\hat{\Sigma}_{v,v}$ is the lower 5 by 5 block of $\hat{\Sigma}$. All possible envelope estimators constructed from reducing subspaces of $\hat{\Sigma}_{v,v}$ are produced and compared using the `selection` function in the `enlpaster` package. There are 31 possible reducing subspaces in total and the `combs` function in the `caTools` package is used keep track of the indices. Reducing subspaces are constructed one dimension at a time. At a specified dimension, we construct all of the possible envelope estimators using reducing subspaces. There are $\binom{5}{1}$ possible reducing subspaces of dimension 1, $\binom{5}{2}$ possible reducing subspaces of dimension 2, and so on. At a particular iteration of this procedure, denote the reducing subspace of $\hat{\Sigma}_{v,v}$ by \hat{G} . The envelope estimator of v is given by $\hat{v}_{\text{env}} = P_{\hat{G}}\hat{v}$.

The envelope estimator of τ is

$$\hat{\tau}_{\text{env}} = \begin{pmatrix} \hat{\gamma} \\ \hat{v}_{\text{env}} \end{pmatrix}.$$

Estimators of this form correspond to an aster model with a new model matrix that incorporates the estimated projection into the envelope space. This model matrix is given as

$$M_{\text{env}} = M \begin{pmatrix} I & 0 \\ 0 & P_{\hat{G}} \end{pmatrix}.$$

The model matrix M_{env} satisfies $\hat{\tau}_{\text{env}} = M_{\text{env}}^T Y$ which says that the envelope estimator of τ is a maximum likelihood estimator for the aster model with model matrix M_{env} . Once the envelope estimator $\hat{\tau}_{\text{env}}$ is calculated, it is transformed to the β parameterization using the model matrix M_{env} as an argument of `transformUnconditional`. The parameter vector β arises naturally in the log likelihood

$$l(\beta) = \langle M^T Y, \beta \rangle - c(\beta). \quad (1)$$

The transformation from $\hat{\tau}$ to $\hat{\beta}$ provides a maximizer of (1) with model matrix given by M_{env} . However, $\hat{\beta}$ is not unique since M_{env} is not of full column rank. This non identifiability is not problematic because final inference is not conducted with respect to the β parameterization. Final inference is conducted with respect to the parameterization of Darwinian fitness which does not possess such problems. When in the β parameterization, we compute AIC, BIC, and the p-value of the LRT using `aster` software and the likelihood (1). The `selection` function in the `envlpaster` package below computes the AIC, BIC, and the p-value of the LRT for all 31 envelope models including the model corresponding to the MLE. This is done by specifying the `method = "eigen"` and `type = "mean-value"` arguments in the `selection` function.

```
foo2 <- selection(parm = tau, index = target, model = m1,
  data = data, alpha = 0.05, type = "mean-value",
  method = "eigen")
```

The calculation above is not performed in this script. The calculation is performed in `ex1-Efronboot.R` and is sourced into this document from the `ex1-Efronboot.RData` file.

```

load("ex1-Efronboot.RData")
foo2$aic; foo2$bic; foo2$LRT

## [1] 1 2 4 5
## [1] 1 4 5
## [1] 1 4 5

```

We see that BIC and the LRT at size $\alpha = 0.05$ favor the reducing subspace that is the sum of the first, fourth, and fifth eigenspaces of $\widehat{\Sigma}_{v,v}$. This reducing space will be referred to as \widehat{G}_1 . On the other hand, AIC favors the reducing subspace that is the sum of all eigenspaces of $\widehat{\Sigma}_{v,v}$ with the exception of the third eigenspace. This reducing subspace will be referred to as \widehat{G}_2 . Since eigenvalues of $\widehat{\Sigma}_{v,v}$ are unique, the space \widehat{G}_1 can also be defined as the space spanned by the first, fourth, and fifth eigenvector of $\widehat{\Sigma}_{v,v}$ and \widehat{G}_2 can be defined similarly. Both of these reducing subspaces are smaller than the full space which is a sum of all of the eigenspaces of $\widehat{\Sigma}_{v,v}$ and both are larger than the true reducing subspace. The true reducing subspace of $\widehat{\Sigma}_{v,v}$ used to construct this example, G_{true} , is the space spanned by the first and fourth eigenvectors of $\widehat{\Sigma}_{v,v}^1$, where $\widehat{\Sigma}_{v,v}^1$ is the estimated asymptotic covariance matrix of the parameters of interest constructed from a preliminary simulated dataset. The realizations in the dataset used in this example were generated from the aster model distribution with submodel mean-value parameter $\tau_{\text{true}} = (\hat{\gamma}_1^T, P_{G_{\text{true}}} \hat{v}_1^T)^T$. Here, $\hat{\gamma}_1$ and \hat{v}_1 are the MLE of the nuisance parameters and parameters of interest obtained from the preliminary dataset.

The parametric bootstrap procedure given by Algorithm 2 (Eck, Geyer, and Cook, 2015) is used to estimate the envelope estimator selected by BIC and the LRT of size $\alpha = 0.05$. The steps of this procedure are outlined in Algorithm 2 of the current document. Before we run our parametric bootstrap, we see which envelope model is selected when we use the 1D algorithm Cook and Zhang (2014, 2015); Eck, Geyer, and Cook (2015). This is done by changing the `method` argument in the `selection` function to `method = "1D"`. All three selection criteria favor an envelope dimension of $u = 5$. This corresponds to a selection of the full aster model with no envelope structure incorporated. The parametric bootstrap procedure discussed in Algorithm 2 is not very interesting in this case.

We now construct the envelope estimator of τ with respect to \widehat{G}_1 as sug-

gested by BIC and the LRT of size $\alpha = 0.05$. The envelope estimator is given by $\hat{\tau}_{\text{env}} = M_{\text{env}}^T Y$ where

$$M_{\text{env}} = M \begin{pmatrix} I & 0 \\ 0 & P_{\hat{G}_1} \end{pmatrix}.$$

Expected Darwinian fitness is estimated for 144 hypothetical individuals possessing unique values of \mathbf{z}_1 and \mathbf{z}_2 suggested by the original data. These hypothetical individuals will be used to build a fitness landscape for estimated expected Darwinian fitness. The fitness landscape is estimated using both maximum likelihood estimation and envelope estimation. Before we build the dataframe corresponding to these 144 hypothetical individuals, the `vars` vector containing the names of all the nodes in Figure 1 is specified.

```
vars <- as.vector(outer(c("u", "v", "w"), 1:10, paste,
  sep = ""))
```

The dataframe corresponding to the 144 hypothetical individuals is constructed using the code below.

```
nx <- 12; npop <- nx^2
z1.cand <- seq(from = -6, to = 6, length = nx)
z2.cand <- seq(from = -6, to = 6, length = nx)
fred <- data.frame( cbind(rep(z1.cand, each = nx),
  rep(z2.cand, nx), matrix(1, nrow = nx^2, ncol = 30)))
colnames(fred) <- c("z1", "z2", vars)
fred$root <- 1
```

Next, `amat` needs to be specified. `amat` specifies which nodes in the graphical structure for the hypothetical individuals are Darwinian fitness nodes. `amat` can either be a three-dimensional array or a matrix. We construct a three-dimensional array as in Geyer and Shaw (2009).

```
amat <- array(0, c(npop, nnode, npop))
foot <- grepl("w", vars)
for (k in 1:npop) amat[k, foot, k] <- 1
```

The `newdata` argument is specified with a dataframe for the hypothetical individuals created to build the fitness landscape. This dataframe has already been constructed and has been given the name `fred`. The model matrix and

dataframe in long format, corresponding to the provided dataframe specified by `newdata`, are constructed internally. If the construction of either the new model matrix or the new dataframe requires more calculation than a simple `model.matrix` or `renewdata` call respectively, then these quantities can be included using the `modmat.new` or `renewdata` arguments respectively.

```
renewdata <- reshape(fred, varying = list(vars),
  direction = "long", timevar = "varb",
  times = as.factor(vars), v.names = "resp")
layer2 <- gsub("[0-9]", "",
  as.character(renewdata$varb))
renewdata <- data.frame(renewdata, layer = layer2)
renewdata$vttype <- as.factor(substr(as.character(
  renewdata$varb), 1, 1))
renewdata$year <- as.numeric(substr(as.character(
  renewdata$varb), 2))
renewdata$uyear <- renewdata$year *
  as.numeric(as.character(renewdata$vttype) == "u")
```

We also need to specify the `modmat.new` argument. The dataframe in long format is calculated as an intermediate step to the construction of this model matrix.

```
modmat.renew <- model.matrix(m1$formula, data = renewdata)
```

We now have the required quantities necessary to estimate the variability of estimated expected Darwinian fitness using envelope methodology. The envelope estimator of expected Darwinian fitness for these hypothetical individuals has asymptotic distributions given by

$$\sqrt{n}(g(\hat{\tau}_{\text{env}}) - g(\tau)) \xrightarrow{\mathcal{D}} N(0, \nabla g(\tau)\Delta_2\nabla g(\tau)^T) \quad (2)$$

where Δ_2 is the unknown asymptotic covariance matrix of $\hat{\tau}_{\text{env}}$. The asymptotic covariance matrix of estimated expected Darwinian fitness is estimated using the parametric bootstrap adjusting for model selection as seen in Algorithm 2. The dimension of the envelope space is selected using model selection criteria and is not known in advance. Efron (2014, Section 4) provides a double bootstrap procedure which accounts for model selection. This procedure is used to estimate the variance of estimated expected Darwinian

fitness using envelope methodology. At first, data are generated as a realization from the aster model distribution evaluated at the envelope estimator. The envelope dimension is determined for each of these generated datasets. The estimator of expected Darwinian fitness is then taken to be the average of all of the envelope estimators obtained from these datasets. To estimate the variability of this envelope estimator, we generate further datasets from the aster model distribution evaluated at each separate envelope estimator of expected Darwinian fitness used to calculate the aforementioned average. The steps of this procedure are presented in Eck, Geyer, and Cook (2015, Algorithm 2). For completeness, the steps of this algorithm are restated here. To avoid confusion, we label this algorithm as Algorithm 2.

There are two functions in the `envlpaster` package necessary to perform all of the steps in Algorithm 2. The reason for separating this chore into two functions is computational. This separation allows users to parallelize this bootstrap using `mclapply`. The first of these functions is `fit.boot.Efron`. This function implements the first level of the parametric bootstrap procedure given by either Algorithm 1 or Algorithm 2 in Eck, Geyer, and Cook (2015). This is detailed in Steps 1 through 3d outlined in Algorithm 2. This parametric bootstrap generates resamples from the distribution evaluated at an envelope estimator of τ adjusting for model selection volatility.

To use the `fit.boot.Efron` function, the user must specify whether they want to use AIC, BIC, or the LRT of size α as a model selection criterion. The user also specifies which method, either the 1D algorithm or the reducing subspace approach, is to be used to calculate envelope estimators. When one is using a partial envelope model, then this function constructs envelope estimators of v where we write $\tau = (\gamma^T, v^T)^T$ and v corresponds to aster model parameters of interest. In the original sample, the reducing subspaces are indices of eigenvectors of $\widehat{\Sigma}_{v,v}$ where $\widehat{\Sigma}_{v,v}$ is the part of $\widehat{\Sigma}$ corresponding to our parameters of interest. These indices are specified by **vectors**. When all of the components of τ are components of interest, then we write $\widehat{\Sigma}_{v,v} = \widehat{\Sigma}$.

We now call on `fit.boot.Efron` to perform the top level of bootstrapping for our example. The calculation above is not performed in this script. It is performed in `ex1-Efronboot.R` and is sourced into this document from the `ex1-Efronboot.RData` file.

Algorithm 2: Parametric bootstrap reducing subspace envelope estimator of v

1. Fit the aster model to the data and obtain \hat{v} and $\hat{\Sigma}_{v,v}$ from the aster model fit.
2. Compute the envelope estimator of v in the original sample, given as $\hat{v}_{\text{env}} = P_{\hat{G}}\hat{v}$ where $P_{\hat{G}}$ is computed using reducing subspaces and selected via a model selection criterion of choice.
3. Perform a parametric bootstrap by generating resamples from the distribution of the aster submodel evaluated at $\hat{\tau}_{\text{env}} = (\hat{\gamma}^T, \hat{v}_{\text{env}}^T)^T$. For iteration $b = 1, \dots, B$ of the procedure:
 - (3a) Compute $\hat{\tau}^{(b)}$ and $\hat{\Sigma}_{v,v}^{(b)}$ from the aster model fit to the resampled data.
 - (3b) Build $P_{\hat{G}}^{(b)}$ using the indices of $\hat{\Sigma}_{v,v}^{(b)}$ that are selected using the same model selection criterion as Step 2 to build \hat{G} .
 - (3c) Compute $\hat{v}_{\text{env}}^{(b)} = P_{\hat{G}}^{(b)}\hat{v}^{(b)}$ and $\hat{\tau}_{\text{env}}^{(b)} = (\hat{\gamma}^{(b)T}, \hat{v}_{\text{env}}^{(b)T})^T$.
 - (3d) Store $\hat{\tau}_{\text{env}}^{(b)}$ and $g\left(\hat{\tau}_{\text{env}}^{(b)}\right)$ where g maps τ to the parameterization of Darwinian fitness.
4. After B steps, the bootstrap estimator of expected Darwinian fitness is the average of the envelope estimators stored in Step 3d. This completes the first part of the bootstrap procedure.
5. We now proceed with the second level of bootstrapping at the b -th stored envelope estimator $\hat{\tau}_{\text{env}}^{(b)}$. For iteration $k = 1, \dots, K$ of the procedure:
 - (5a) Generate data from the distribution of the aster submodel evaluated at $\hat{\tau}_{\text{env}}^{(b)}$.
 - (5b) Perform Steps 3a through 3d with respect to the dataset obtained in Step 5a.
 - (5c) Store $\hat{\tau}_{\text{env}}^{(b)(k)}$ and $g\left(\hat{\tau}_{\text{env}}^{(b)(k)}\right)$.

```

set.seed(13)
nboot <- 100
blah <- fit.boot.Efron(model = m1, nboot = nboot, index = target,
  dim = foo2$bic, data = data, amat = amat, newdata = fred,
  modmat.new = modmat.renew, renewdata = renewdata, criterion = "BIC",
  method = c("eigen"), quiet = FALSE)

```

The contents stored in `blah`, and previously computed quantities, are all that is needed to perform the second level of bootstrapping necessary for the estimation of the variability of estimated expected Darwinian fitness incorporating envelope methodology. The function `secondboot` in the `envlpaster` package performs this second level of bootstrapping. This function implements the second level of the parametric bootstrap procedure given by either Algorithm 1 or Algorithm 2 in Eck, Geyer, and Cook (2015) with respect to the mean-value parameterization. This is detailed in Steps 4 through 5c in Algorithm 2. At iteration b , this parametric bootstrap generates resamples from the distribution evaluated at the envelope estimator ($\hat{\tau}_{env}^{(b)}$) of τ . In this case, the selected indices producing the reducing subspace which was used to construct the envelope estimator $\hat{\tau}_{env}^{(b)}$ are used to construct envelope estimators for the generated data. These resamples are used to estimate the variability of $\hat{\tau}_{env}^{(b)}$.

We now call on `secondboot` to perform the second level of bootstrapping for our example. The calculations involving `secondboot` are not performed in this script. These calculations are performed in `ex1-secondboot.R` and are sourced into this document from the `ex1-secondboot.RData` file. The bootstrap sample size for the second level of bootstrapping is 500. The script below performs this task. This script also calculates the bootstrapped estimator of the variability of expected Darwinian fitness incorporating envelope methodology.

```
load("ex1-secondboot.RData")
```

```

nboot2 <- 500
set.seed(13)
internal <- function(k){
  set.seed(13)

```

```

bar <- secondboot(k, out = blah, model = m1, data = data,
  nboot2 = nboot2, index = target, newdata = fred,
  amat = amat, method = "eigen")
return(bar$sd.Efron)
}

```

When the top-level of our bootstrap procedure in Algorithm 2 has run for a total of B iterations, we obtain the envelope estimator

$$\hat{\mu}_g = \frac{1}{B} \sum_{b=1}^B g(\hat{\tau}_{\text{env}}^{(b)}). \quad (3)$$

The individual estimates of expected Darwinian fitness in (3) involve model selection which is smoothed out by averaging (Efron, 2014). This envelope estimator obtained from the parametric bootstrap in Algorithm 2 has variability analogous to that in Efron (2014, equation (4.15)). As in Efron (2014), for $b = 1, \dots, B$ we define the matrix $\mathbf{B}^{(b)} \in \mathbb{R}^{K \times p}$ which has rows $\hat{\tau}_{\text{env}}^{(b)(k)} - \sum_{k=1}^K \hat{\tau}_{\text{env}}^{(b)(k)} / K$ and the matrix $C^{(b)} \in \mathbb{R}^{K \times d}$ which has columns $g\left(\tau_{\text{env}}^{(b)(k)}\right) - g\left(\tau_{\text{env}}^{(b)}\right)$. Δ_2 is estimated by

$$\hat{\Delta}_2 = \sum_{b=1}^B \left[\widehat{\text{cov}}^{(b)T} \widehat{V}^{(b)-1} \widehat{\text{cov}}^{(b)} \right] / B \quad (4)$$

where

$$\widehat{\text{cov}}^{(b)} = \mathbf{B}^{(b)T} C^{(b)} / K \quad (5)$$

and

$$\widehat{V}^{(b)} = \mathbf{B}^{(b)T} \mathbf{B}^{(b)} / K. \quad (6)$$

The estimator (4) of Δ_2 takes into account the volatility of model selection when estimating the variability of estimated expected Darwinian fitness using envelope methodology. The method of maximum likelihood estimation does not have the added model selection step that envelope estimation has. The bootstrap procedure outlined in Algorithm 2 efficiently estimates expected Darwinian fitness and accounts for variability associated with model selection volatility.

The code below calculates (4). These calculations are performed in `ex1-secondboot.R` and are sourced into this document from the `ex1-secondboot.RData` file. Note that a machine with five available cores

was used to perform this calculation.

```
y <- nboot / 5
blah.second <- mclapply(1:5, mc.cores = 5, FUN = function(x){
  y <- nboot / 5
  int <- ((y*(x-1) + 1) : (y*x))
  out <- lapply(int, FUN = internal)
  return(out)
})
```

```
sd.Efron.mean <- rep(0,144)
for(k in 1:5){
  for(j in 1:y){
    sd.Efron.mean <- sd.Efron.mean +
      blah.second[[k]][[j]] / nboot
  }
}
```

The file `techreport_eigenboot_fitboot.RData` contains output corresponding to the bootstrapped estimator of the variability of expected Darwinian fitness using the MLE. The bootstrap sample size for this particular bootstrap was 500.

```
nboot1 <- nboot
load("techreport_eigenboot_fitboot.RData")
```

We now compute the ratio of standard errors. Values above 1 indicate that the envelope estimator of expected Darwinian fitness using Algorithm 2 is less variable than the MLE. The code below computes the bootstrapped standard errors of estimated expected Darwinian fitness using the MLE.

```
a <- bar$MLE.boot.out

data.fit <- asterdata(fred, vars, pred,
  group = rep(0,30), code,
  families)
```

```

phi.MLE <- origin + modmat.renew %*% beta
mu.MLE <- transformSaturated(phi.MLE, data.fit, from = "phi",
  to = "mu")
amat.fit <- matrix(amat, nrow = length(mu.MLE))
fit.MLE <- t(amat.fit) %*% mu.MLE
fit.MLE <- as.vector(fit.MLE)
a <- a - fit.MLE

boot.var.MLE <- (a[, 1] %o% a[, 1]) / ncol(a)
for(j in 2:ncol(a)){
  boot.var.MLE <- boot.var.MLE + (a[, j] %o% a[, j]) / ncol(a)
}

sd.MLE <- sqrt(diag(boot.var.MLE))
ratio <- sd.MLE / sd.Efron.mean

```

Output is shown for hypothetical individuals who have estimated expected Darwinian fitness greater than 8 where estimation is calculated using the MLE. This table has 5 columns. The first two columns correspond to estimation of expected Darwinian fitness using envelope methodology and its bootstrapped standard error respectively. The next two columns correspond to estimation of expected Darwinian fitness using the MLE and its bootstrapped standard error respectively. The fifth column displays the ratios of estimated standard errors where ratios greater than 1 indicate that the envelope estimator has lower estimated variability. We can see that efficiency gains are obtained for the displayed output.

```

load("techreport_eigenboot_fitboot.RData")
nboot <- nboot1

```

```

tab <- matrix(0, nrow = 144, ncol = 5)
tab[, 1] <- rowSums(blah$env.boot.out) / nboot
tab[, 2] <- sd.Efron.mean
tab[, 3] <- fit.MLE
tab[, 4] <- sd.MLE
tab[, 5] <- ratio
colnames(tab) <- c("env", "se(env)", "MLE", "se(MLE)", "ratio")
tab[tab[, 3] > 8, ]

```

```
##           env  se(env)      MLE  se(MLE)  ratio
## [1,] 8.556409 0.1742400 8.674101 0.2597591 1.490812
## [2,] 9.013616 0.1105248 8.938662 0.1350701 1.222079
## [3,] 7.817013 0.4135980 8.030673 0.4420406 1.068769
## [4,] 9.174295 0.1627223 9.193001 0.1700020 1.044737
## [5,] 9.018329 0.1126568 9.126989 0.1276560 1.133141
## [6,] 8.611867 0.1624729 8.541299 0.2776820 1.709097
## [7,] 7.761383 0.2153252 8.109067 0.3302726 1.533831
```

We plot the ratio of estimated standard errors in Figures 3 and 4. We can see that efficiency gains are obtained when using envelope methodology for hypothetical individuals that overlap the majority of the original data.

```
z1 <- simdata30nodes$z1
z2 <- simdata30nodes$z2
max.ind <- which(tab[,1] == max(tab[,1]))
x.max = fred[max.ind,1]
y.max = fred[max.ind,2]
plot(z1,z2, pch = 20, col = "grey")
points(x = x.max, y = y.max, col = "red", pch = 20)
max.ind.env <- which(tab[,3] == max(tab[,3]))
x.max.env = fred[max.ind.env,1]
y.max.env = fred[max.ind.env,2]
points(x = x.max.env, y = y.max.env, col = "blue", pch = 20)
contour(z1.cand,z2.cand, t(matrix(tab[,5],nrow = nx)),
  levels = c(0.9,1,1.5,2,3), labcex = 1, add=TRUE)
```

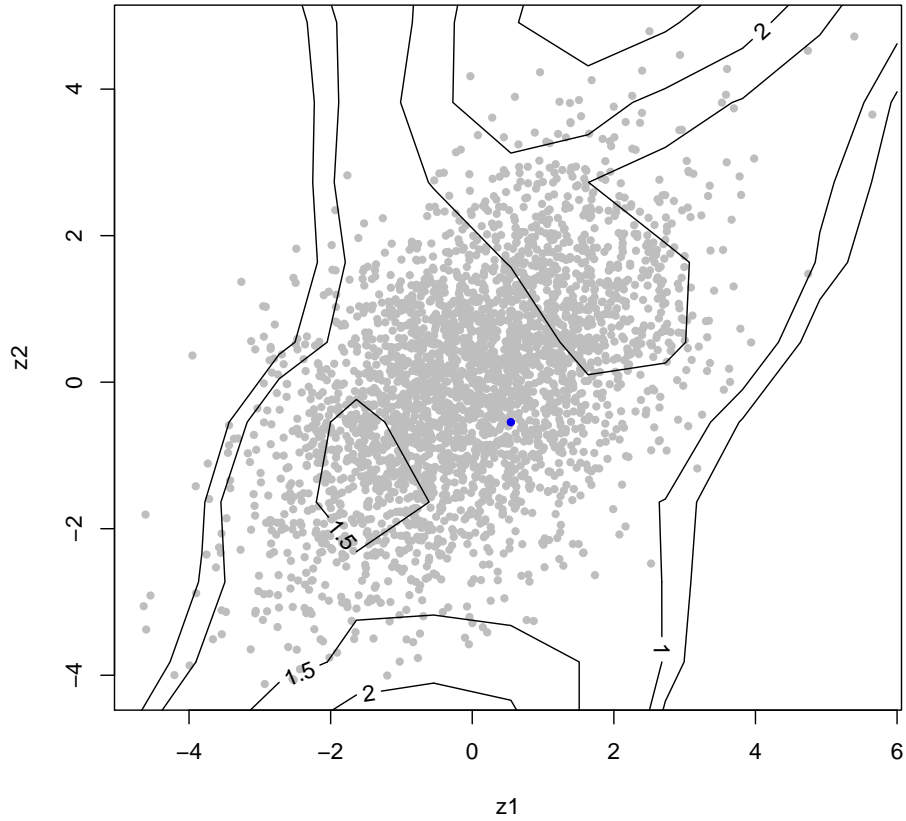


Figure 3: Contour plot for the ratios of $\hat{se}(h(\hat{\tau}))$ to $\hat{se}(h(\hat{\tau}_{env}))$. Ratios greater than 1 indicate efficiency gains using envelope methodology. The point in blue corresponds to the highest estimated expected Darwinian fitness value using envelope methodology and the MLE.

```
filled.contour(z1.cand,z2.cand, t(matrix(tab[,5],nrow = nx)),
  levels = c(0,0.9,1,1.5,2,5))
```

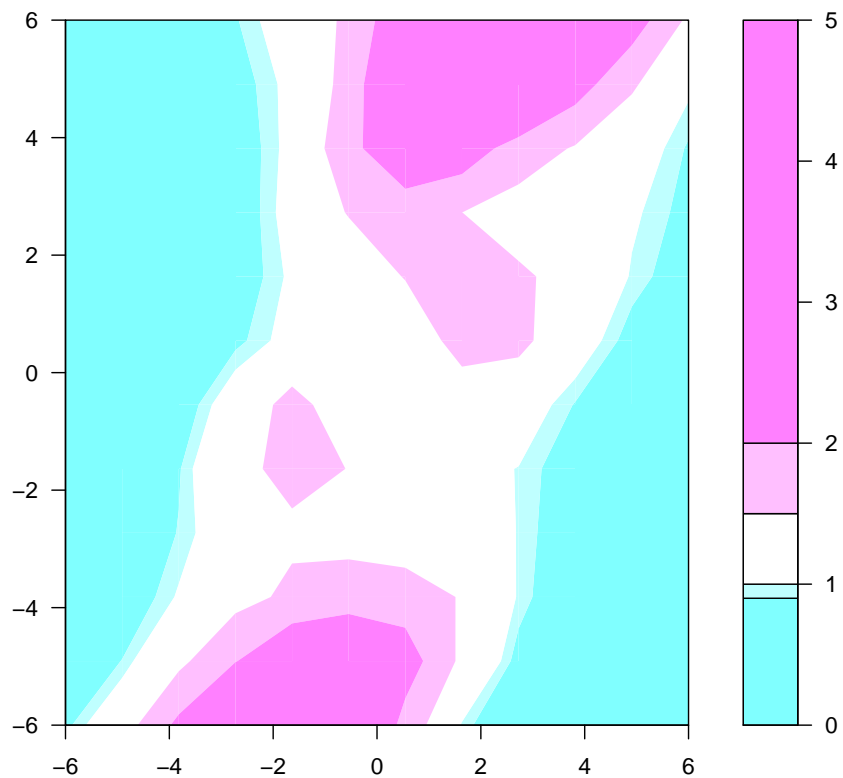


Figure 4: Heat map for the ratios of $\hat{se}(h(\hat{\tau}))$ to $\hat{se}(h(\hat{\tau}_{env}))$. Ratios greater than 1 indicate efficiency gains using envelope methodology.

We now perform our methods with respect to the reducing subspace \hat{G}_2 selected by AIC. Output is shown for hypothetical individuals who have estimated expected Darwinian fitness greater than 8 where estimation is calculated using the MLE.

```
load("ex1-AIC.RData")
```

```

tab2 <- matrix(0, nrow = 144, ncol = 5)
colnames(tab2) <- c("env", "se(env)", "MLE", "se(MLE)", "ratio")
tab2[, 1] <- rowSums(blah$env.boot.out) / nboot
tab2[, 2] <- sd.Efron.mean
tab2[, 3] <- fit.MLE
tab2[, 4] <- sd.MLE
tab2[, 5] <- ratio
tab2[tab2[, 3] > 8, ]

##           env  se(env)      MLE  se(MLE)   ratio
## [1,] 8.749144 0.1964036 8.674101 0.2599230 1.323412
## [2,] 9.025514 0.1116299 8.938662 0.1351961 1.211109
## [3,] 7.940958 0.4204229 8.030673 0.4424621 1.052422
## [4,] 9.159294 0.1638821 9.193001 0.1701628 1.038324
## [5,] 9.047524 0.1138559 9.126989 0.1277835 1.122326
## [6,] 8.393870 0.2005390 8.541299 0.2778990 1.385760
## [7,] 7.833197 0.2152651 8.109067 0.3305165 1.535393

```

The ratios of estimated standard errors are plotted in Figures 5 and 6. We can see in these figures and in the table above that efficiency gains are obtained using envelope methodology. The gains are less substantial than those found when using the indices suggested by BIC and the LRT of size $\alpha = 0.05$.

```

z1 <- simdata30nodes$z1
z2 <- simdata30nodes$z2
max.ind <- which(tab[,1] == max(tab[,1]))
x.max = fred[max.ind,1]
y.max = fred[max.ind,2]
plot(z1,z2, pch = 20, col = "grey")
points(x = x.max, y = y.max, col = "red", pch = 20)
max.ind.env <- which(tab[,3] == max(tab[,3]))
x.max.env = fred[max.ind.env,1]
y.max.env = fred[max.ind.env,2]
points(x = x.max.env, y = y.max.env, col = "blue", pch = 20)
contour(z1.cand,z2.cand, t(matrix(tab[,5],nrow = nx)),
  levels = c(0.9,1,1.5,2,3), labcex = 1, add=TRUE)

```

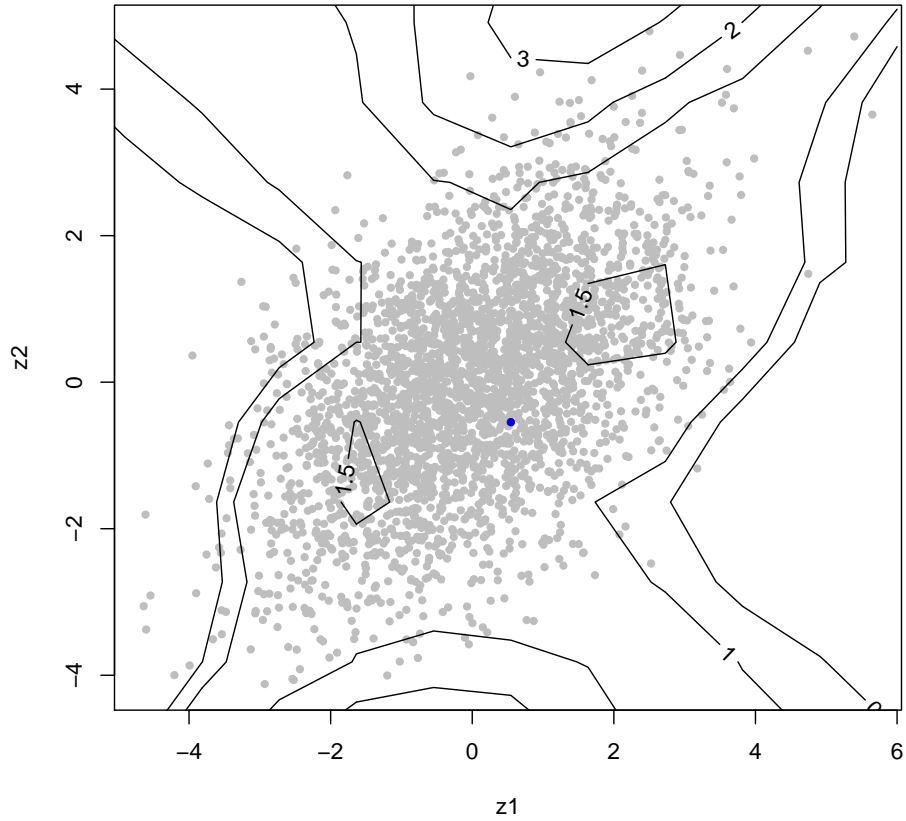


Figure 5: Contour plot for the ratios of $\hat{se}(h(\hat{\tau}))$ to $\hat{se}(h(\hat{\tau}_{\text{env}}))$ using AIC. Ratios greater than 1 indicate efficiency gains using envelope methodology. The point in blue corresponds to the highest estimated expected Darwinian fitness value using envelope methodology and MLE.

```
filled.contour(z1.cand,z2.cand, t(matrix(tab[,5],nrow = nx)),
  levels = c(0,0.9,1,1.1,1.25,1.5,2))
```

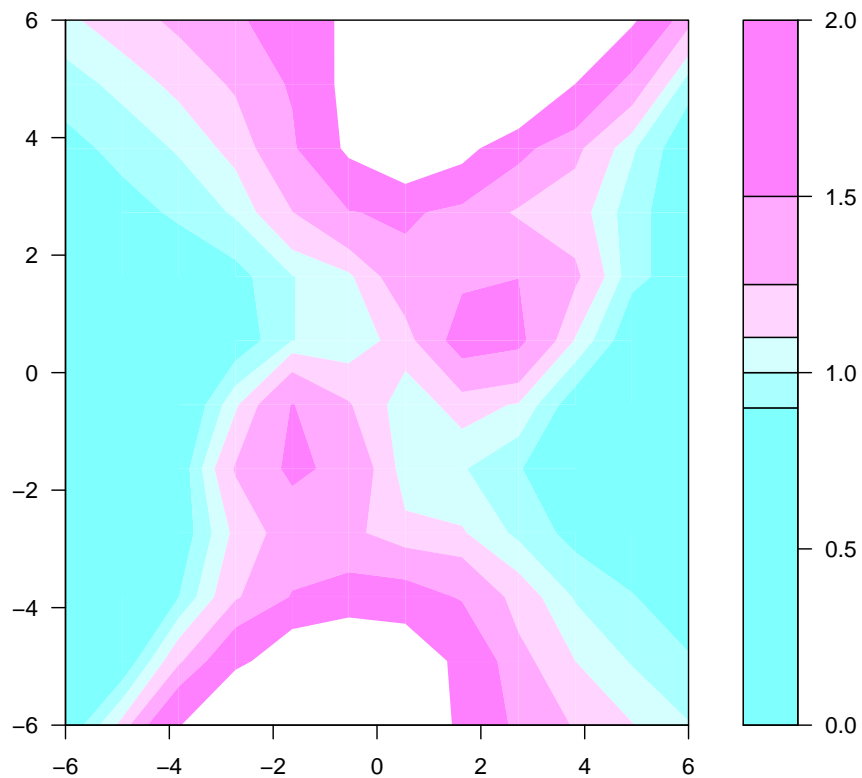


Figure 6: Heat map for the ratios of $\hat{se}(h(\hat{\tau}))$ to $\hat{se}(h(\hat{\tau}_{env}))$ using AIC. Ratios greater than 1 indicate efficiency gains using envelope methodology.

$$1 \rightarrow Y_1 \rightarrow Y_2$$

Figure 7: Graphical structure of the aster model for the data in Example 2. The first arrow corresponds to survival which is a Bernoulli random variable. The second arrow corresponds to reproduction count conditional on survival which is a zero-truncated Poisson random variable.

2 Replicating Example 2

We now reproduce the second example in Eck, Geyer, and Cook (2015). The dataset comes from Lowry and Willis (2010) and the study in which the data is obtained investigates the role of chromosomal inversions in adaptation and speciation. Phenotypic traits and covariates are recorded for 2313 yellow monkeyflowers, *Mimulus guttatus*. The lifecycle of the individual *M. guttatus* flowers is depicted in Figure 7. Much of the details necessary to the reproduction of this example are similar to those in the first example. The data and corresponding `.RData` file are loaded in. The data is then manipulated so that `aster` software can be used and then the quantities needed for the bootstrap functions `fit.Efron` and `secondboot` are computed.

The code below obtains the relevant quantities needed to fit the aster model corresponding to this data and manipulates the dataset so that aster models can be fit.

```
library(aster2)
library(envlpaster)
data(Mguttatus)
```

```
data <- Mguttatus
redata <- Mguttatus.redata
vars <- quantities$vars
pred <- quantities$pred
group <- quantities$group
code <- quantities$code
fam <- quantities$fam
```

```

nnode <- length(vars)
n <- nrow(redata) / nnode
families <- quantities$families
root <- redata$root
fit <- redata$fit
varvar <- quantities$varvar
idvar <- quantities$idvar

```

The .RData file containing the calculations done in this example is loaded in.

```
load("main-all-terms-DavidLowrydata.RData")
```

We fit the main effects model. This model includes genetic background, site, inversion, and ecotype as predictors.

```

m.main <- aster(resp ~ varb + fit:(gen_bac + site +
  inversion + type), pred, fam, varvar = varvar,
  idvar = idvar, data = redata, root = root)

```

There are eight predictors comprising τ and we partition τ into $(\gamma^T, v^T)^T$ where $\gamma \in \mathbb{R}^2$ are nuisance parameters and $v \in \mathbb{R}^6$ are relevant to the estimation of expected Darwinian fitness. In this example $\Sigma_{v,v}$ is the lower 6 by 6 block of the Fisher information matrix Σ . These quantities, as well as β are extracted from the aster model fit.

```

target <- which(grepl("fit", names(m.main$coef)))
avar.targ <- m.main$fisher[target, target]
modmat.mat <- matrix(m.main$modmat, nrow = n*nnode)
tau <- crossprod(modmat.mat, redata$resp)
tau.targ <- tau[target]
beta <- m.main$coef
p <- length(beta)

```

We now call on `selection` to see which reducing subspace is suggested for envelope estimation.

```
foo <- selection(tau, target, m.main, data, type = "mean-value",
  method = "eigen", alpha = 0.01)
```

All of the selection criteria are in alignment, and they suggest using the reducing subspace that is the sum of the first, second, third, and sixth eigenspaces of $\hat{\Sigma}_{v,v}$.

```
foo$bic; foo$aic; foo$LRT
## [1] 1 2 3 6
## [1] 1 2 3 6
## [1] 1 2 3 6
```

The variability of estimated expected Darwinian fitness is now estimated using both `fit.boot.Efron` and `secondboot` to perform the bootstrap procedure outlined in Algorithm 2. First, the hypothetical *M. guttatus* flowers in which expected Darwinian fitness is estimated are created. All variables in our original aster model are factor variables, therefore each hypothetical *M. guttatus* flower possesses a unique factor-level combination.

```
a <- levels(test$gen_bac)
b <- levels(test$site)
c <- levels(test$inversion)
d <- levels(test$type)
fred <- expand.grid(a = a, b = b, c = c, d = d)
colnames(fred) <- c("gen_bac", "site", "inversion", "type")
fred$sur_flw <- 1
fred$flws <- 1
```

The dataset in long form and the model matrix corresponding to the main-effects model with these hypothetical *M. guttatus* flowers is now created.

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

```

renewdata$fit <- fit.renew
renewdata$root <- 1
npop <- nrow(fred)

modmat.renew <- model.matrix(m.main$formula, data = renewdata)
modmat.renew[, 2] <- 1 - modmat.renew[, 2]
index <- !(colnames(modmat.renew) %in% m.main$dropped)
modmat.renew <- modmat.renew[, index]

```

Next, `amat` needs to be specified. `amat` specifies which nodes in the graphical structure for the hypothetical *M. guttatus* flowers correspond to Darwinian fitness nodes. `amat` can either be a three-dimensional array or a matrix. We construct a three-dimensional array as in Geyer and Shaw (2009).

```

amat <- matrix(0, nrow = npop, ncol = nnode * npop)
for(j in 1:npop) amat[j, npop + j] <- fit.renew[npop + j]

```

The `fit.boot.Efron` function performs the top level of bootstrapping for our example. The calculation below is not performed in this script. The calculation is performed in `main-all-terms-DavidLowrydata.R` and is sourced into this document from the `main-all-terms-DavidLowrydata.RData` file.

```

set.seed(13)
nboot <- 100
blah <- fit.boot.Efron(model = m.main, nboot = nboot, index = target,
  vectors = foo$bic, data = data, amat = amat, newdata = fred,
  modmat.new = modmat.renew, renewdata = renewdata,
  criterion = "BIC", method = "eigen", quiet = FALSE)

```

We now call on `secondboot` to perform the second level of bootstrapping for our example. The calculations involving `secondboot` are not performed in this script. These calculations are also performed in `main-all-terms-DavidLowrydata.R` and are sourced into this document from the `main-all-terms-DavidLowrydata.RData` file. The bootstrap sample size for the second level of bootstrapping is 500. The script below performs this task. This script also calculates the bootstrapped estimator of the variability of expected Darwinian fitness incorporating envelope

methodology.

```
nboot2 <- 500
internal <- function(k){
  set.seed(13)
  bar <- secondboot(k, out = blah, model = m.main, data = data,
    nboot2 = nboot2, index = target, newdata = fred,
    amat = amat, method = "eigen")
  return(bar$sd.Efron)
}

## 5 cores are used to perform the second level of bootstrapping
ncores <- 5; nsamp <- nboot / ncores
blah.second <- mclapply(1:ncores, mc.cores = ncores,
  FUN = function(x){
    int <- ((nsamp*(x-1) + 1) : (nsamp*x))
    out <- lapply(int, FUN = internal)
    return(out)
  }
)

## estimated standard error using
sd.Efron.mean <- rep(0, npop)
for(k in 1:ncores){
  for(j in 1:nsamp){
    sd.Efron.mean <- sd.Efron.mean +
      blah.second[[k]][[j]] / nboot
  }
}
```

Expected Darwinian fitness and its variability are now estimated for the hypothetical *M. guttatus* flowers using the MLE. In this setting, the variability is estimated using the output from `fit.boot.Efron` to construct the bootstrap estimator of variability given by

$$\frac{1}{99} \sum_{b=1}^{100} \left(h(\hat{\mu}^{(b)}) - h(\hat{\mu}) \right) \left(h(\hat{\mu}^{(b)}) - h(\hat{\mu}) \right)^T$$

where $h(\hat{\mu})$ is estimated expected Darwinian fitness for the hypothetical *M. guttatus* flowers using the MLE and $h(\hat{\mu}^{(b)})$ are the bootstrapped estimates.

```

data.renew <- asterdata(fred, vars = vars, pred = pred,
  group = rep(0, length(vars)), code = code,
  families = families)
origin.renew <- m.main$origin[1:npop,]
origin.renew <- as.vector(origin.renew)
mu.renew <- transformUnconditional(beta, data.renew,
  modmat = modmat.renew, from = "beta", to = "mu",
  offset = origin.renew)
fit <- amat %*% mu.renew

dev <- blah$MLE.boot.out - as.vector(fit)
x <- (dev[, 1] %*% t(dev[, 1]))
for(j in 2:nboot){
  x <- x + (dev[, j] %*% t(dev[, j]))
}

```

The ratio of the the estimated standard error using envelope estimation obtained from Algorithm 2 to the estimated standard error using the MLE is computed.

```

sd.MLE <- sqrt(diag(x) / (nboot - 1))
ratio <- sd.MLE / sd.Efron.mean

```

The output below is shown for hypothetical *M. guttatus* flowers who have estimated expected Darwinian fitness greater than 7 where estimation is calculated using MLE. The output table has 5 columns. The first two columns correspond to estimation of expected Darwinian fitness using envelope methodology and its bootstrapped standard error respectively. The next two columns correspond to estimation of expected Darwinian fitness using MLE and its bootstrapped standard error respectively. The fifth column displays the ratios of estimated standard errors where ratios greater than 1 indicate that the envelope estimator has lower estimated variability. We can see that efficiency gains are obtained for the displayed output.

```

tab3 <- matrix(0, nrow = npop, ncol = 5)
colnames(tab3) <- c("env", "se(env)", "MLE", "se(MLE)", "ratio")
tab3[, 1] <- rowSums(blah$env.boot.out) / nboot
tab3[, 2] <- sd.Efron.mean
tab3[, 3] <- fit
tab3[, 4] <- sd.MLE
tab3[, 5] <- ratio
tab3[(tab3[, 3] > 7), ]

##           env  se(env)      MLE  se(MLE)   ratio
## [1,]  9.646009 0.3255334  9.170619 0.6421457 1.972595
## [2,]  8.639696 0.2999494  8.887339 0.3688551 1.229724
## [3,]  7.658725 0.3154779  7.602779 0.3608801 1.143916
## [4,]  7.517441 0.5386544  7.009952 0.6493285 1.205464
## [5,] 10.942830 0.6068946 10.474696 0.8958160 1.476065
## [6,]  7.497749 0.5208250  7.521516 0.6579239 1.263234
## [7,]  9.646009 0.3255334  9.170619 0.6421457 1.972595
## [8,]  9.646009 0.3255334  9.170619 0.6421457 1.972595

```

References

- Cook, R. D., Zhang, X. (2014). Foundations for Envelope Models and Methods. *JASA*, **110:510**: 599-611.
- Cook, R. D., Zhang, X. (2015). Algorithms for Envelope Estimation. *Journal of Computational and Graphical Statistics*, Published online. DOI:10.1080/10618600.2015.1029577.
- Eck, D. J. (2015). R package `envlpaster`, version 0.1-1. <http://cran.r-project.org/package=envlpaster>.
- Eck, D. J., Geyer, C. J., and Cook, R. D. (2015). An Application of Envelope Methodology and Aster Models. *in preparation*.
- Efron, B. (2014). Estimation and Accuracy After Model Selection. *JASA*, **109:507**, 991-1007.
- Geyer, C. J. (2010). A Philosophical Look at Aster Models. Technical Report No. 676. School of Statistics, University of Minnesota. <http://purl.umn.edu/57163>.

Geyer, C. J. and Shaw, R. (2009). Model Selection in Estimation of Fitness Landscapes. Technical Report No. 671. School of Statistics, University of Minnesota. <http://conservancy.umn.edu/handle/11299/56219>.

Lowry, D. B. and Willis, J. H. (2010) A Widespread Chromosomal Inversion Polymorphism Contributes to a Major Life-History Transition, Local Adaptation, and Reproductive Isolation. *PLoS Biol*, 8(9): e1000500. doi:10.1371/journal.pbio.1000500.

A. make the *Mguttatus* data

In this Appendix we show how the *Mguttatus* dataset in the `envlpaster` package was initially created starting from the data for the original analysis (Lowry, et al., 2010), which was a comma separated values (CSV) file. This file can be found at the same location at the University of Minnesota Digital Conservancy (<http://conservancy.umn.edu/>) where this technical report is found.

```
test <- read.csv("Aster_across_sites_DIV1.csv",
  sep = ",", header = T)
```

Now we need to specify the rest of the graph shown in Figure 7. One of these variables that specify the graph is obvious. `vars` names the variables that are pasted together to form the response vector (all the variables at all the nodes of the graph). The others are less obvious. `pred` specifies the arrows of the graph, and `group` specifies the lines. In this analysis, there are no groups. If `pred[j]` is not zero, then there is an arrow from node `pred[j]` to node `j`. Otherwise, there is a line from the initial node (marked 1 in the aster graph) to node `j`. `code` is an index vector into the families list; `families[[code[j]]]` is the family for the conditional distribution of the dependence group containing node `j` given its predecessor node.

```
vars <- c("sur_flw", "flws")
pred <- c(0, 1)
group <- rep(0, length(pred))
code <- c(1, 2)
families <- list("bernoulli", fam.zero.truncated.poisson())
```

Now we make the `asterdata` object and verify that this `aster` object is the as the `asterdata` provided by the `envlpaster` package.


```

data <- asterdata(data = test, vars = vars, pred = pred,
  group = group, code = code, families = families)
all.equal(data, Mguittatus)

## [1] TRUE

```

B. make the simulated data

The last line of code shows that the generated dataset analyzed in Example 1 is the same as the dataset included in the `envlpaster` package.

```

set.seed(13)
nind <- 3000
ntime <- 10
psurv <- 0.95
prepr <- 0.7
mpois <- 1
theta.surv <- log(psurv) - log(1 - psurv)
theta.repr <- log(prepr) - log(1 - prepr)
theta.pois <- log(mpois)
tau.pois <- mpois/(1 - exp(-mpois))

vars <- as.vector(outer(c("u", "v", "w"), 1:10, paste,
  sep = ""))
vtype <- as.factor(substr(as.character(vars), 1,1))
fam <- rep(1, length(vars))
fam[vtype == "w"] <- 3
pred <- seq(along = vars) - 1
pred[vtype == "u"] <- seq(along = vars)[vtype == "u"] - 3
pred[1] <- 0

root <- matrix(1, nrow = nind, ncol = length(vars))
theta <- root
theta[, vtype == "u"] <- theta.surv
theta[, vtype == "v"] <- theta.repr
theta[, vtype == "w"] <- theta.pois
x <- raster(theta, pred, fam, root)

```

```

dimnames(x) <- list(NULL, vars)
dat <- as.data.frame(x)
dat <- as.list(dat)
dat[["root"]] <- rep(1, nind)

npheno <- 10
zbase <- rnorm(nind)
for (i in 1:npheno) {
  labz <- paste("z", i, sep = "")
  dat[[labz]] <- zbase + rnorm(nind)
}

dat <- as.data.frame(dat)
redata <- reshape(dat, varying = list(vars), direction = "long",
  timevar = "varb", times = as.factor(vars), v.names = "resp")
wind <- grep("w", as.character(redata$varb))
for (labz in grep("z", names(redata), value = TRUE)) {
  redata[[labz]][-wind] <- 0
}

redata$vtype <- as.factor(substr(as.character(redata$varb),
  1, 1))
out2 <- aster(resp ~ vtype + 0, pred, fam, varb,
  id, root, data = redata, type = "conditional")
out3 <- aster(resp ~ vtype + 0, pred, fam, varb,
  id, root, data = redata)

redata$year <- as.numeric(substr(as.character(redata$varb), 2))
redata$uyear <- redata$year * as.numeric(as.character(redata$vtype) ==
  "u")
out4 <- aster(resp ~ vtype + uyear, pred, fam, varb,
  id, root, data = redata)

#####

```

```

##### Section 4.2 model 1 #####
out5 <- aster(resp ~ vtype + uyear + z1 + z2 + I(z1^2) +
  I(z2^2) + I(z1 * z2), pred, fam, varb, id, root,
  data = redata)

# change the covariate data so that it is mean 0
z1 <- dat$z1
z2 <- dat$z2
ascal <- 0.013
quad <- ascal * ((z1 + z2) - (z1^2 + z2^2) + z1 * z2)
con <- mean(quad)
beta.new <- out5$coefficients
beta.new[1:4] <- out4$coefficients
beta.new[3] <- beta.new[3] - con
beta.new[5:6] <- ascal
beta.new[7:8] <- (-ascal)
beta.new[9] <- ascal
beta.new <- round(beta.new, 3)

theta <- predict(out5, model.type = "conditional",
  parm.type = "canonical", newcoef = beta.new)
theta <- matrix(theta, nrow = nrow(out5$x), ncol = ncol(out5$x))
xnew <- raster(theta, pred, fam, root)
dimnames(xnew) <- list(NULL, vars)
dnew1 <- as.data.frame(xnew)
renew <- reshape(dnew1, varying = list(vars),
  direction = "long", timevar = "varb",
  times = as.factor(vars), v.names = "resp")
redata$resp1 <- renew$resp
redata$taul <- predict(out5, newcoef = beta.new)
redata$phi1 <- predict(out5, parm.type = "canonical",
  newcoef = beta.new)
beta1true <- beta.new

out6 <- aster(resp1 ~ vtype + uyear + z1 + z2 + I(z1^2) +
  I(z2^2) + I(z1 * z2), pred, fam, varb, id, root,
  data = redata)

```

```

#####
# tau envelope model stuff
target <- 5:9
m <- length(target)
p <- length(out6$coef)
avar <- out6$fisher
beta <- out6$coef
avar.targ <- avar[target,target]
beta.targ <- beta[target]

mu <- predict(out6, parm.type = "mean.value",
  model.type = "unconditional")
modmat <- out6$modmat
modmat.mat <- matrix(modmat, ncol = length(beta))
offset <- as.vector(out6$origin)
x <- out6$x
tau <- crossprod(modmat.mat, mu)
tau.targ <- tau[target]

# check whether transformUnconditional and predict.aster
# are returning the same quantity
fam[which(fam == 3)] <- 2
code <- fam
families = list(fam.bernoulli(),fam.zero.truncated.poisson())
data <- asterdata(data = dat, vars = vars, pred = pred,
  group = rep(0, length(vars)), code = code,
  families = families)
tau.foo <- transformUnconditional(parm = beta, modmat.mat, data,
  from = "beta", to = "tau", offset = offset, tol = 1e-10)

# put the envelope truth into the model
G <- eigen(avar.targ)$vec[,c(1,4)]
P <- tcrossprod(G)
tau_true <- tau.foo
M <- t(modmat.mat)
M2 <- M[target,]; M2 <- P %*% M2

```

```

M[target,] <- M2
modmat.mat.env <- t(M)
tau_true[target] <- P %*% tau.targ
beta.foo <- transformUnconditional(parm = tau_true,
  modmat.mat, data, from = "tau", to = "beta",
  offset = offset)

theta <- predict(out6, model.type = "conditional",
  parm.type = "canonical", newcoef = beta.foo)
theta <- matrix(theta, nrow = nrow(out6$x), ncol = ncol(out6$x))
xnew <- raster(theta, out6$pred, out6$fam, out6$root)
dimnames(xnew) <- list(NULL, vars)
xnew <- cbind(xnew, dat[,c(32:41)])
xnew$root <- 1
data <- asterdata(data = xnew, vars = vars, pred = pred,
  group = rep(0, length(vars)), code = code,
  families = families)

all.equal(data, simdata30nodes.asterdata)

## [1] TRUE

```