Running head: Ornstein-Uhlenbeck state-space model

Ecology: in press

Density dependent state-space model for population abundance data with unequal time intervals

Brian Dennis[1, 3] and José Miguel Ponciano[2]

[1]*Department of Fish and Wildlife Sciences and Department of Statistical Science, University of Idaho, Moscow ID 83844-1136, USA*

[2]*Department of Biology, University of Florida, Gainesville, FL, 32611-8525, USA*

[3]E-mail: brian@uidaho.edu

Abstract. The Gompertz state-space (GSS) model is a stochastic model for analyzing time series observations of population abundances. The GSS model combines density dependence, environmental process noise, and observation error toward estimating quantities of interest in biological monitoring and population viability analysis. However, existing methods for estimating the model parameters apply only to population data with equal time intervals between observations. In the present paper, we extend the GSS model to data with unequal time intervals, by embedding it within a state-space version of the Ornstein-Uhlenbeck process, a continuous-time model of an equilibrating stochastic system. Maximum likelihood and restricted maximum likelihood calculations for the Ornstein-Uhlenbeck state-space model involve only numerical maximization of an explicit multivariate normal likelihood, and so the extension allows for easy bootstrapping, yielding confidence intervals for model parameters, statistical hypothesis testing of density dependence, and selection among submodels using information criteria. Ecologists and managers previously drawn to models lacking density dependence or observation error because such models accommodated unequal time intervals (for example, due to missing data) now have an alternative analysis framework incorporating density dependence, process noise and observation error.

Key words: density dependence, diffusion process, Gompertz model, lognormal distribution, mean-reverting process, Ornstein-Uhlenbeck process, state-space model, stationary distribution, stochastic differential equation, stochastic population model

INTRODUCTION

Gaps in time series observations of population abundances pose challenges for analysis. Various statistical models for ecological time series data, especially models incorporating realistic population dynamics, require observations spaced at equal time intervals. However, ecological sampling involves constraints of time, personnel, and budgets that do not always live up to the designs and requirements of statistical models. As well, various ecological systems such as aquatic or microbial systems (Kirchman 2012) have intrinsically continuous-time dynamics and are sometimes sampled at unequal time intervals. Ordinarily equal time intervals will be stretched and shrunk into unequal intervals in systems in which time is measured in degree days (Metcalf and Luckmann 1994). The data that do exist in studies with missing data or unequal time intervals are potentially informative, and precluding such data from analysis could affect conclusions regarding the biological resources in question. The monitoring and management of biological populations would benefit from having better models for analyzing time series data with unequal time intervals.

Dennis et al. (2006) described a "state-space" population model for use in ecological time series analysis. The model, termed the Gompertz state space (GSS) model, is one of the simplest possible formulations containing density dependence, stochastic process variability, and stochastic observation or measurement error. The simplicity of the model allows for an explicit likelihood function and for parameter estimation through ordinary numerical maximization. Although the GSS model can accomodate missing observations from otherwise equally spaced time series, the GSS model does not allow observations collected at unequal, non-integer time intervals and hence is not suitable for ecological situations in which time is considered to be real-valued.

Various state space population models of greater complexity can handle unequally spaced data (de Valpine and Hastings 2002, de Valpine 2002, 2004, Clark and Bjørnstad 2004, Ionides et al. 2006, Lele et al. 2007, Ponciano et al. 2009) . Such models require custom programming of simulation-intensive computer algorithms for fitting (parameter estimation) and other statistical inferences. For example, Lele et al. (2007) and Ponciano et al. (2009) used data cloning (a Monte Carlo Markov Chain method) to calculate maximum likelihood estimates of parameters in state space models for analyzing the well-known laboratory *Paramecium* (sp.) data of Gause (1934) which have missing observations on day 1. Adapting such models to other data sets with other configurations of missing observations would require reprogramming the calculations for each new case.

A special case of the GSS model is a density independent state space model. The exponential growth state space (EGSS) model was introduced by Holmes (2001), and parameter estimation was studied by Lindley (2003) and Staples et al. (2004). By contrast to the GSS model, the EGSS model has been generalized to apply to unequal, real-valued time intervals (Staudenmayer and Buonaccorsi 2006, Humbert et al. 2009).

This paper extends the full, density dependent GSS model to unequal, real-valued time intervals. The method used employs a stochastic version of the continuous-time, deterministic Gompertz growth model as the underlying model of population growth (process model). The stochastic version is a continuous-time diffusion process representing Gompertz-type density dependent growth perturbed by environmental noise. Transforming the process model to the logarithmic scale produces the Ornstein-Uhlenbeck process, a well-studied diffusion process. Finally, adding an observation error component produces a state-space version of the Ornstein-Uhlenbeck model. The state space model has discrete-time statistical properties

identical to those of a GSS model, but has a likelihood function that can be written for time series observations recorded at arbitrary real-valued time intervals. The likelihood makes accessible a variety of statistical inferences based on parametric bootstrapping. Four example data sets serve to illustrate statistical inferences with the model. Procedures for calculating confidence intervals for model parameters and for conducting bootstrap tests of density dependence are described in the online Appendix to this paper, and R programs for all statistical inferences are provided (Supplementary Material online).

POPULATION MODEL

Deterministic Gompertz model

The Gompertz curve (after Gompertz 1825) originally was an actuarial model of mortality, but since the 1920s biologists have been using the curve as a deterministic model of biological growth, variously to describe the growth of tumors, individuals, or populations (Winsor 1932). The model is frequently presented in the form of an ordinary differential equation:

$$\frac{dn(t)}{dt} = \theta n(t)[\log \kappa - \log n(t)]. \quad (1)$$

Here $n(t)$ is the abundance of the growing entity at time t, and θ and κ are positive constants, with κ being the equilibrium abundance and θ being a measure of the speed of equilibration. The similar logistic growth model has the density dependence term proportional to $n(t)$ instead of $\log n(t)$. The solution trajectory for Eq. 1 is given by

$$n(t) = \kappa \exp\left[e^{-\theta t} \log(n_0/\kappa)\right], \quad (2)$$

with $n_0 = n(0)$ denoting the initial population abundance. The Gompertz trajectory has an inflection point at κ/e if the initial population abundance is below that level (inflection for the logistic model is at $\kappa/2$).

Stochastic Gompertz model

A stochastic version of the Gompertz model is represented by the following stochastic differential equation (SDE):

$$dN(t) = \theta N(t)[\log \kappa - \log N(t)]dt + \beta N(t)dW(t). \quad (3)$$

Here an infinitesimal increment of the Gompertz growth model (Eq. 1 written in differential form) is perturbed by a random noise term in which $dW(t)$ has a normal distribution with a mean of 0 and a variance of dt (where the correlation between $dW(t_i)$ and $dW(t_j)$ is assumed equal to 0 if $t_i \neq t_j$), and in which the intensity of the noise is scaled by the term $\beta N(t)$, with $\beta > 0$. The scaling term proportional to population abundance is a common model of environmental stochasticity (Tier and Hanson 1981, Dennis and Patil 1984). Such stochasticity can produce substantial population variability at high as well as low abundances. The assumption of zero-correlated process noise describes the unpredictability of growth increment fluctuations in response to environmental buffeting has theoretical and empirical utility (Allen 2010, Dennis et al. 1991). Temporal correlation in process noise, if present, is difficult to estimate in the presence of sampling error (Staples et al. 2008).

The SDE as written in the form of Eq. 3 can be understood as an easy recipe for simulating a trajectory of population abundance for specified values of θ, κ, and β (for instance, Higham 2001, Allen 2010): (1) Over a tiny time interval dt, calculate an increment $dN(t)$ of abundance from a normal distribution with a mean of $\theta N(t)[\log \kappa - \log N(t)]dt$ and a variance of $[\beta N(t)]^2 dt$, with $N(t)$ being the current population abundance and t being the current time in the simulation. (2) Update population abundance as $N(t) + dN(t)$ and update time as $t + dt$. (3) Return to step (1) and repeat the process for a new time interval, generating a new normal random growth increment using the updated population abundance. The accuracy of the

simulation (in terms of the statistical properties of $N(t)$) is degraded if the time interval dt is not small. While various alternative simulation algorithms can improve the numerical accuracy of SDEs in general (Allen 2010) and the OU process in particular (Gillespie 1996), the preceding "Euler-Maruyama" method for SDEs serves as an accessible pedagogical entry point to a highly mathematized subject.

Ornstein-Uhlenbeck model

Written as an SDE in the form of Eq. 3, population abundance is a type of stochastic process known as a diffusion process with infinitesimal mean function given by $m(n) = \theta n(\log \kappa - \log n)$ and infinitesimal variance function $v(n) = \beta^2 n^2$ (Karlin and Taylor 1981). A useful property of diffusion processes is that a smooth invertible transformation of a diffusion process is also a diffusion process (Karlin and Taylor 1981). Let $X(t) = g\big(N(t)\big)$ be such a transformation (perhaps $\log N(t)$ or $\sqrt{N(t)}$, etc.). The infinitesimal mean and variance of $X(t)$ are given by the Itô transformation formulas: $m_X(x) = m(n)g'(n) + \frac{1}{2}v(n)g''(n)$, $v_X(x) = v(n)[g'(n)]^2$, with $n = g^{-1}(x)$. The transformation property opens the possibility that a given diffusion process might be transformed to a scale for which results helpful for analysis and inference can be derived. Such a transformation exists for the stochastic Gompertz model.

The logarithmic transformation, $X(t) = \log N(t)$, converts population abundance $N(t)$ under the stochastic Gompertz model into a well-studied diffusion process known as an Ornstein-Uhlenbeck (OU) process (after Uhlenbeck and Ornstein 1930; see Allen 2010). Under the Itô transformation formulas, the infinitesimal mean and variance functions for $X(t)$ become

$$m_X(x) = \theta(\mu - x), \tag{4}$$

$$v_X(x) = \beta^2, \tag{5}$$

where $\mu = \log \kappa - \beta^2/(2\theta)$. The SDE for $X(t)$ takes the form

$$dX(t) = \theta[\mu - X(t)]dt + \beta dW(t), \quad (6)$$

which is a growth increment $\theta[\mu - X(t)]dt$ from a linear deterministic model perturbed by normally-distributed noise with constant variance $\beta^2 dt$. The SDE given by Eq. 6 defines an OU process $X(t)$. Among the well-known results (see Allen 2010) are that $X(t)$ has a normal transition probability distribution with a mean function given by

$$\mathrm{E}[X(t)|X(0) = x_0] = \mu - (\mu - x_0)e^{-\theta t}, \quad (7)$$

where $x_0 = \log n_0$, and a variance function given by

$$\mathrm{V}[X(t)|X(0) = x_0] = \frac{\beta^2}{2\theta}\left(1 - e^{-2\theta t}\right). \quad (8)$$

The covariance of the process at two different times is given by

$$\mathrm{Cov}[X(t), \mathrm{X}(t+s)] = \frac{\beta^2}{2\theta}\left(1 - e^{-2\theta t}\right)e^{-\theta s}, t, s > 0, \quad (9)$$

and values of the process at multiple different times have a joint multivariate normal distribution. As time t becomes large, the transition distribution converges to a normal stationary distribution with mean

$$\mathrm{E}[X(\infty)] = \mu, \quad (10)$$

and variance

$$\mathrm{V}[X(\infty)] = \frac{\beta^2}{2\theta}. \quad (11)$$

The stationary distribution on the original abundance scale is a lognormal distribution and represents the stochastic counterpart to the point equilibrium or "carrying capacity" in the deterministic model (Dennis and Patil 1984).

Ornstein-Uhlenbeck state space model

We define an Ornstein-Uhlenbeck state space (OUSS) model by adding an observation error component. The observations can be taken at arbitrary times which need not be equally spaced. The observed population log-abundance at time t_i is modeled as

$$Y(t_i) = X(t_i) + F_i \ , \tag{12}$$

where F_i has a normal distribution with mean 0 and variance τ^2, with F_i, F_j uncorrelated ($i \neq j$). Observation error on the original abundance scale thereby has a lognormal distribution, which can be regarded as a model of ecological sampling under heterogeneous sampling conditions (Dennis et al. 2006). Eqs. 6 and 12 together constitute the OUSS model. The underlying actual population log-abundance $X(t_i)$ is unobserved.

The observations $Y(0), Y(t_1), Y(t_2), \ldots, Y(t_q)$ (estimates or indexes of log-population abundance) recorded at times $t_0 = 0, t_1, \ldots, t_q$ need not be equally spaced. The observations under the OUSS model have a joint multivariate normal distribution, but the distribution takes two different forms, non-stationary and stationary, depending on the situation being modeled. If the population abundances have commenced far from equilibrium, the observations have the non-stationary distribution in which the mean of each $Y(t_i)$ is

$$\mathrm{E}[Y(t_i)] \ = \ \mu - (\mu - x_0)e^{-\theta t_i} \ , \tag{13}$$

the variance is

$$\mathrm{V}[Y(t_i)] = \frac{\beta^2}{2\theta}\left(1 - e^{-2\theta t_i}\right) + \ \tau^2 \ , \tag{14}$$

and the covariance of the process at two different times t_i and t_j is given by

$$\mathrm{Cov}\left[Y(t_i), \mathrm{Y}\left(t_j\right)\right] = \frac{\beta^2}{2\theta}\left(1 - e^{-2\theta \min(t_i, t_j)}\right)e^{-\theta|t_i - t_j|} \ . \tag{15}$$

If however the population abundances have become stationary (fluctuating stochastically around an equilibrium) before sampling commenced, the observations have the stationary distribution in which the mean of each $Y(t_i)$ is

$$\mathrm{E}[Y(t_i)] = \mu\,, \tag{16}$$

the variance is

$$\mathrm{V}[Y(t_i)] = \frac{\beta^2}{2\theta} + \tau^2\,, \tag{17}$$

and the covariance of the process at two different times t_i and t_j is given by

$$\mathrm{Cov}\left[Y(t_i), \mathrm{Y}\left(t_j\right)\right] = \frac{\beta^2}{2\theta} e^{-\theta|t_i - t_j|}\,. \tag{18}$$

In either the stationary or non-stationary case the joint distribution of the observations is similar to that of the underlying OU process, except that the quantity τ^2 is added to the variance of each $Y(t_i)$. Two different methods for fitting the model to data (Statistical Methods, below) arise from whether or not the population has become stationary.

The OUSS model can be regarded as a continuous time version of the GSS model. The GSS model is defined (in the notation of Dennis et al. 2006) by $X_t = a + cX_{t-1} + E_t, Y_t = X_t + F_t$, where X_t and Y_t are respectively the underlying and observed population log-abundances at times $t = 0, 1, 2, \ldots, q$, and E_t and F_t have independent normal distributions with means of 0 and respective variances of σ^2 and τ^2. The parameters a, σ^2, and τ^2 are positive, and $-1 < c < +1$. If $0 < c < 1$, the transition probability distributions of the OUSS and GSS models evaluated at integer times coincide as identical normal distributions means and variances given by Eqs. 13 and 14, with the relationships between parameters given by the following transformations: $a = \mu\left(1 - e^{-\theta}\right), c = e^{-\theta}, \sigma^2 = \left(1 - e^{-2\theta}\right)\beta^2/(2\theta)$; $\mu = a/(1-c)$, $\theta = -\ln(c), \beta^2 = -[2\sigma^2\ln(c)]/(1-c^2)$, with τ^2 having the same value in both models. If

$-1 < c < 0$ the, GSS model has a deterministic component with damped oscillatory behavior for which there is no straightforward continuous time version. However, oscillatory behavior in population data, damped or otherwise, suggests the presence of nonlinear dynamic forces (May 1994), and one should explore alternative model structures with additional state variables and/or nonlinear interactions.

STATISTICAL METHODS

Maximum likelihood estimation for the OUSS stationary case

Under stationarity, the initial log-population abundance, $X(0)$, is assumed to arise from the stationary normal distribution of the population abundances. Such an assumption of might be plausible if the monitoring program commenced after the population had been existing for a time long enough for stochastic equilibration in the particular environment. The model then has four unknown parameters: θ, μ, β^2, and τ^2.

Fitting the model to data means estimating the unknown model parameters. Using the multivariate normal distribution, the model can be fitted to data with maximum likelihood (ML) estimation. For ML estimation with the OUSS model, the likelihood function is the joint multivariate normal probability density for the observations evaluated at their realized data values. The ML estimates then are the parameter values that jointly maximize the likelihood or log-likelihood function. The multivariate normal log-likelihood for the stationary OUSS model is given by

$$\log L(\mu, \theta, \beta^2, \tau^2) = -\frac{(q+1)}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbf{V}|) - \frac{1}{2}(\mathbf{y}-\mathbf{m})'\mathbf{V}^{-1}(\mathbf{y}-\mathbf{m}) , \quad (19)$$

where the elements of the column vector $\mathbf{y}$ are the data values $y_0, y_1, y_2, \ldots, y_q$ (realized values of the random variables $Y(0), Y(t_1), Y(t_2), \ldots, Y(t_q)$) recorded at times $0, t_1, t_2, \ldots, t_q$. Also, the column vector $\mathbf{m}$ has all $q+1$ elements identically equal to μ (Eq. 16), the variances on the main

diagonal of the variance-covariance matrix $\mathbf{V}$ are all equal to $\tau^2 + [\beta^2/(2\theta)]$ (Eq. 17), and the (i,j)th and (j,i)th covariance elements in $\mathbf{V}$ are both equal to $[\beta^2/(2\theta)]e^{-\theta s_{ij}}$, where $s_{ij} = |t_i - t_j|$ (Eq. 18).

The ML estimates must be calculated with numerical optimization (for instance, with the "optim" function in R; R Core Development Team 2006). An R program for such calculation is provided in the Supplementary Material online.

Restricted ML estimation for the stationary case

Restricted maximum likelihood (REML) estimation for the multivariate normal distribution uses a linear combination of the observations that eliminates the parameters in the mean vector, leaving only parameters in the variance-covariance matrix. The statistical properties of the resulting estimates for the GSS and EGSS models are frequently better than those of ML estimates (Dennis et al. 2006, Humbert et al. 2009). For the OUSS model under the stationary case, a linear combination that produces a zero-mean vector is the simple differencing of the observations. Let $W_i = Y(t_i) - Y(t_{i-1}), i = 1, 2, \ldots, q$. The W_i's have a joint multivariate normal distribution with a mean vector of all zeros and a variance-covariance matrix given by

$$\mathbf{\Phi} = \mathbf{DVD}', \tag{20}$$

where $\mathbf{D}$ is the $q \times (q+1)$ differencing matrix given by

$$\mathbf{D} = \begin{bmatrix} -1 & 1 & 0 & 0 \cdots \\ 0 & -1 & 1 & 0 \cdots \\ 0 & 0 & -1 & 1 \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{21}$$

For REML estimation, the multivariate normal log-likelihood is a function of three of the parameters in the OUSS model:

$$\log L(\theta, \beta^2, \tau^2) = -\frac{q}{2}\log(2\pi) - \frac{1}{2}\log(|\mathbf{\Phi}|) - \frac{1}{2}\mathbf{w}'\mathbf{\Phi}^{-1}\mathbf{w}. \tag{22}$$

Here the column vector of differenced observations is $\mathbf{w} = \mathbf{Dy}$. After numerical maximization of the REML log-likelihood, the mean parameter μ is estimated by

$$\hat{\mu} = \mathbf{j}'\hat{\mathbf{V}}^{-1}\mathbf{y}/\left(\mathbf{j}'\hat{\mathbf{V}}^{-1}\mathbf{j}\right), \quad (23)$$

where $\hat{\mathbf{V}}$ is the variance-covariance matrix $\mathbf{V}$ (for the undifferenced observations, Eq. 19) evaluated with the REML estimates of θ, β^2, and τ^2, and $\mathbf{j}$ is a $(q+1) \times 1$ column vector of ones. The R program for parameter estimation accompanying this paper (Supplementary Material online) calculates the REML estimates in addition to the ML estimates.

ML estimation for the nonstationary case

The nonstationary OUSS model should be considered for use when an appreciable proportion of the observations are recorded during a transient growth phase, as for instance when a population displays a sigmoid trajectory instead of a trajectory solely fluctuating around a steady state. Examples of such situations might occur when monitoring the translocation of a species by release of a few individuals into the wild, the colonization of a new area by an invading species, the increase of an agricultural pest through a growing season starting from low density, the recovery of a species after a catastrophic decline, or the growth of microbial populations in experimental laboratory cultures (Ponciano et al. 2005). The initial population log-abundance x_0 usually becomes an additional unknown parameter in the nonstationary model, unless its value is known without sampling error (such as might be the case for a species translocation or a laboratory inoculation).

ML estimation is available for the nonstationary case, but REML estimation is not available. ML estimation uses the multivariate normal log-likelihood (Eq. 19) for the data values $y_0, y_1, y_2, \ldots, y_q$, but with a different mean vector and variance-covariance matrix. The element in the mean vector $\mathbf{m}$ corresponding to the data value y_i is given by Eq.13 , the variances on the

main diagonal of $\mathbf{V}$ are computed with Eq. 14, and the (i,j)th and (j,i)th covariance elements in $\mathbf{V}$ are obtained with Eq. 15. The log-likelihood must be numerically maximized jointly for the five unknown parameters x_0, θ, μ, β^2, and τ^2.

If the initial log-abundance x_0 is known, the log-likelihood has just the four unknown parameters θ, μ, β^2, and τ^2, with x_0 fixed at its known value. Also, when x_0 is known the initial observation y_0 is equal to x_0 and is omitted from the likelihood (that is, omitted from the data vector $\mathbf{y}$). ML estimation uses the multivariate normal log-likelihood (Eq. 19) modified to apply to just the q observations $y_1, y_2, \ldots, y_q$ recorded at times $t_1, t_2, \ldots, t_q$. The modifications consist of substituting a $q \times 1$ vector $\mathbf{m}$ containing means defined by Eq. 13, a $q \times q$ variance-covariance matrix $\mathbf{V}$ containing elements defined by Eqs. 14 and 15, and a leading term $-\frac{q}{2}\log(2\pi)$ instead of $-\frac{q+1}{2}\log(2\pi)$. The R program for parameter estimation accompanying this paper also optionally calculates estimates for the non-stationary case (Supplementary Material online).

Additional statistical inferences for the OUSS model include confidence intervals for parameters, hypothesis tests for density dependence, and model selection among submodels. The statistical justifications and calculations involved in the additional inferences are described in the Appendix.

EXAMPLES

Bobcat (*Lynx rufus*) abundances in Idaho seemingly held steady, albeit with substantial fluctuations, over a 25 yr period beginning in 1956 (Figure 1, upper left). Bobcat abundances in Maine displayed fluctuations of even greater magnitude (Figure 1, upper right). The bobcat data are furbearer harvest records listed in the Global Population Dynamics Database (NERC Centre for Population Biology 2010), data sets 212 (Idaho) and 216 (Maine). As an index of population

abundance, fur harvest records can be assumed to arise from a variable and heterogeneous sampling process, and any models of such time series should incorporate an appropriate sampling component. In Idaho, an unexplained hiatus in the data occurs during the years 1966-69, while in the Maine data the years 1938-41, 1967 and 1969 are missing. Mark-recapture estimates of the elk (*Cervus elaphus*) population in the central valley of Grand Teton National Park, Wyoming, are missing the year 1983 (Figure 1, lower left; data reported by Boyce 1989 as used by Dennis and Taper 1994). Yearly sampling of adult rangeland grasshopper (Orthoptera: Acrididae, mixed species) abundances in Montana began in 1948, with the years 1949-50, 1977, and 1982 missing (Figure 1, lower right; data reported by Kemp and Dennis 1993). The gaps are the main reason for the selection of the particular data sets here to illustrate the OUSS model. All the data sets in Figure 1 are among those included in the R programs accompanying this paper (Supplementary Material online).

The parameter μ tends to be estimated with precision, according to the REML bootstrap confidence intervals (Figure 1). The parameters θ, β^2 and τ^2 are estimated well for these data, as suggested by the wide confidence intervals (Figure 1). The GSS model, and by implication the OUSS model, sometimes has ridge-like likelihoods, or likelihoods with multiple local maxima corresponding to sub-models with all observation error or all process noise (Dennis et al. 2006, 2011, Knape 2008), especially for short time series. Replicating the sampling process one or more times at selected sampling times can substantially improve parameter estimation (Dennis et al. 2011), as can substantial amounts of transience in the data (nonstationarity) resulting from initiating the time series far from equilibrium (Dennis and Taper 1994).

DISCUSSION

If unequal sampling intervals are due simply to data missing from otherwise equally spaced observations (such as the data in Figure 1), then the ordinary GSS model can in fact be used with modification. The GSS model, like the OUSS model, has observations with a multivariate normal log-likelihood (Eq. 19). To calculate ML estimates for the GSS model, the missing observations in question are deleted from the vector $\mathbf{y}$, the entries corresponding to the missing observations are deleted from $\mathbf{m}$, and the rows and columns corresponding to the missing observations are deleted from $\mathbf{V}$. The resulting expression is the proper log-likelihood function for the multivariate normal distribution of the remaining observations, under a standard property of the multivariate normal distribution (Seber 1984). While deleting observations from the GSS is easy in principal, programming software to automate the deletions in order to analyze many data sets would not be straightforward.

An additional problem with missing observations occurs for restricted maximum likelihood (REML) estimation. For the GSS model, REML estimates are calculated from the likelihood function for the first differences (Dennis et al. 2006). For the EGSS model, REML estimates are calculated from the likelihood function for the second differences (Staples et al. 2004). The problem is that each missing observation removes two observations from REML estimation in the GSS model and three observations from REML estimation in the EGSS model. REML estimation with missing data thus becomes costly, in terms of information about parameters, for the low sample sizes common in ecological time series.

Also, the GSS model cannot be used for systems sampled naturally at unequal time intervals. Plankton sampling schedules in aquatic systems, for instance, might not adhere to rigidly equal intervals. In insect ecology, degree days are routinely used as a time scale, and the application of a degree-day transformation to sampling times with equal intervals invariably

produces unequal intervals on the new scale. In such cases, the unequal intervals cannot be handled merely by dropping observations from the multivariate normal log-likelihood function. A solution instead is to use the OUSS model.

Alternative ways for accommodating unequal intervals in state space models are computationally challenging. The basic approach treats the underlying population abundance as an unobserved "latent variable" in a hierarchical statistical model. Parameters are estimated with intensive MCMC simulation, coupled with either a Bayesian (Clark and Bjornstadt 2004) or frequentist (Lele et al. 2007) inferential method. While the approach allows the use of more complex ecological models of population growth and/or sampling, the problems of parameter estimability are not made any easier. Recent developments in estimation diagnostics (Lele et al. 2010, Ponciano et al. 2012) at least are helping investigators recognize when problems exist. As well, the MCMC calculations require considerable expertise in programming and statistical theory. By contrast, the OUSS model presented here brings density dependence, process noise, and observation error within reach of ordinary numerical maximization.

Acknowledgements

BD was supported by US Department of Defense SERDP project SI-1477 and National Marine Fisheries Service contract AB133F-06-SE-5682. JMP was supported by the National Institute of General Medical Sciences of the National Institutes of Health, under grant number 1R01GM103604-01.

LITERATURE CITED

Allen, L. J. S. 2010. An introduction to stochastic processes with applications to biology, second edition. CRC Press, Boca Raton, Florida,, USA.

1 Boyce, M. S. 1989. The Jackson elk herd: intensive management in North America. Cambridge
2 University Press, Cambridge, UK.
3 Clark, J. S., and O. N. Bjørnstad. 2004. Population time series: process variability, observation
4 errors, missing values, lags, and hidden states. Ecology 85:3140-3150.
5 Dennis, B., P. L. Munholland, and J. M. Scott. 1991. Estimating growth and extinction
6 parameters for endangered species. Ecological Monographs 61:115-143.
7 Dennis, B., and G. P. Patil. 1984. The gamma distribution and weighted multimodal gamma
8 distributions as models of population abundance. Mathematical Biosciences 68:187-212.
9 Dennis, B., J. M. Ponciano, S. R. Lele, M. L. Taper, and D. F. Staples. 2006. Estimating density
10 dependence, process noise, and observation error. Ecological Monographs 76:323-341.
11 Dennis, B., J. M. Ponciano, and M. L. Taper. 2011. Replicated sampling increases efficiency in
12 monitoring biological populations. Ecology 91:610-620.
13 Dennis, B., and M. L. Taper. 1994. Density dependence in time series observations of natural
14 populations: estimation and testing. Ecological Monographs 64:205-224.
15 de Valpine, P. 2002. Review of methods for fitting time-series models with process and
16 observation error and likelihood calculations for nonlinear, non-Gaussian state-space models.
17 Bulletin of Marine Science 70:455-471.
18 de Valpine, P. 2004. Monte Carlo state–space likelihoods by weighted posterior kernel density
19 estimation. Journal of the American Statistical Association 99:523-536.
20 de Valpine, P., and A. Hastings. 2002. Fitting population models with process noise and
21 observation error. Ecological Monographs 72:57-76.
22 Gause, G. F. 1934. The struggle for existence. Williams & Wilkins, Baltimore, Maryland, USA.

Gillespie, D. T. 1996. Exact numerical simulation of the Ornstein-Uhlenbeck process and its integral. Physical Review E 54:2084-2091.

Gompertz, B. 1825. On the nature of the function expressive of the law of human mortality, and on a new mode of determining the value of life contingencies. Philosophical Transactions of the Royal Society of London 115:513-583.

Higham, D. J. 2001. An algorithmic introduction to numerical simulation of stochastic differential equations. SIAM Review 43:525-546.

Holmes, E. E. 2001. Estimating risks in declining populations with poor data. Proceedings of the National Academy of Sciences (USA) 98:5072-5077.

Humbert, J.-Y., L. S. Mills, J. S. Horne, J. S., and B. Dennis. 2009. A better way to estimate population trend. Oikos 118:1940-1946.

Ionides, E. L., C. Breto, and A. A. King. 2006. Inference for nonlinear dynamical systems. Proceedings of the National Academy of Sciences (USA) 103:18438-18443.

Karlin, S., and H. M. Taylor. 1981. A second course in stochastic processes. Academic Press, San Diego, California, USA.

Kemp, W. P., and B. Dennis. 1993. Density dependence in rangeland grasshoppers (Orthoptera: Acrididae). Oecologia 96:1-8.

Kirchman, D. L. 2012. Processes in microbial ecology. Oxford University Press, Oxford, UK.

Knape, J. 2008. Estimability of density dependence in models of time series data. Ecology 89:2994-3000.

Lele, S. R., B. Dennis, and F. Lutscher. 2007. Data cloning: easy maximum likelihood estimation for complex ecological models using Bayesian Markov chain Monte Carlo methods. Ecology Letters 10:551-563.

Lele, S. R., K. Nadeem, and B. Schmuland. 2010. Estimability and likelihood for generalized linear mixed models using data cloning. Journal of the American Statistical Association 105:1617-1625.

Lindley, S. T. 2003. Estimation of population growth and extinction parameters from noisy data. Ecological Applications 13:806–813.

May, R. M. 1974. Stability and complexity in model ecosystems, 2nd edition. Princeton University Press, Princeton, New Jersey, USA.

NERC Centre for Population Biology, Imperial College. 2010. The Global Population Dynamics Database Version 2. http://www3.imperial.ac.uk/cpb/databases/gpdd

Metcalf, R. L., and W. H. Luckmann, eds. 1994. Introduction to insect pest management, third edition. John Wiley, New York, New York, USA.

Ponciano J. M, F. Vandecasteele, T. Hess, L. J. Forney, R. L. Crawford, and P. Joyce. 2005. A stochastic approach to model the effect of environmental factors on bacterial growth. Applied and Environmental Microbiology 71: 2355-2364.

Ponciano, J.M., M. L. Taper, B. Dennis and S. R. Lele. 2009. Hierarchical models in ecology: confidence intervals, hypothesis testing and model selection using data cloning. Ecology 90:356-362.

Ponciano, J. M., G. Burleigh, E, Braun, and M. L. Taper 2012. Assessing parameter identifiability in phylogenetic models using Data Cloning. Systematic Biology 61:955-972.

R Core Development Team. 2006. R: a language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria.

Seber, G. A. F. 1984. Multivariate observations. John Wiley, New York, New York, USA.

Staples, D. F., M. L. Taper, and B. Dennis. 2004. Estimating population trend and process variation for PVA in the presence of sampling error. Ecology 85:923-929.

Staples, D. F., M. L. Taper, B. Dennis, and R. J. Boik. 2008. Effects of sampling error and temporal correlations in population growth on process variance estimators. Environmental and Ecological Statistics (online) DOI 10.1007/s10651-008-0097-5.

Staudenmayer, J. and J. P. Buonaccorsi, J. P. 2006. Measurement error in a random walk model with applications to population dynamics. Biometrics 62:1178-1189.

Tier, C., and F. B. Hanson. 1981. Persistence in density dependent stochastic populations. Mathematical Biosciences 53:89-117.

Uhlenbeck, G. E., and L. S. Ornstein. 1930. On the theory of Brownian Motion. Physical Review 36:823-841.

Winsor, C. P. 1932. The Gompertz curve as a growth curve. Proceedings of the National Academy of Sciences (USA) 18:1-8.

APPENDIX

Confidence intervals, hypothesis tests, and model selection for the Ornstein-Uhlenbeck state-space model (*Ecological Archives* 0000-000-00).

SUPPLEMENT

R scripts for calculating maximum likelihood estimates, restricted maximum likelihood estimates, and parametric bootstrap likelihood ratio tests of density dependence, for the Ornstein Uhlenbeck state space model, using population abundance data having possibly unequal observation time intervals (*Ecological Archives* 0000-000-00).

FIGURE CAPTION

Figure 1. Four population abundance time series (circles), with expected values of true abundances given the other observations (short dashed lines), and bootstrapped 95% confidence intervals for true abundances (long dashed lines), estimated with restricted maximum likelihood under the Ornstein-Uhlenbeck state space model. Upper left: bobcat furbearer harvest records from Idaho, USA (parameter estimates with bootstrapped 95% confidence intervals: $\hat{\mu} = 6.79$ $(6.61, 6.97)$, $\hat{\theta} = 1.26$ $(3.91\times 10^{-7}, 20.2)$, $\hat{\beta}^2 = 0.272$ $(7.43\times 10^{-3}, 3.94)$, $\hat{\tau}^2 = 7.48\times 10^{-4}$ $(9.14\times 10^{-10}, 0.0847)$). Upper right: bobcat furbearer harvest records from Maine, USA ($\hat{\mu} =$ 5.78 $(5.47, 6.08)$, $\hat{\theta} = 0.877$ $(0.0226, 10.3)$, $\hat{\beta}^2 = 0.735$ $(0.0202, 2.40)$, $\hat{\tau}^2 = 0.00475$ $(1.79\times$ $10^{-8}, 0.334)$). Lower left: elk population estimates from Grand Teton National Park, Wyoming, USA ($\hat{\mu} = 7.29$ $(7.14, 7.44)$, $\hat{\theta} = 0.868$ $(0.229, 19.3)$, $\hat{\beta}^2 = 0.0990$ $(0.0296, 1.45)$, $\hat{\tau}^2 = 9.80\times$ 10^{-9} $(2.93\times 10^{-11}, 1.22\times 10^{-3})$). Lower right: grasshopper density estimates from the western mountainous region of Montana, USA ($\hat{\mu} = 1.56$ $(1.31, 1.82)$, $\hat{\theta} = 0.722$ $(0.272, 2.01)$, $\hat{\beta}^2 = 0.347$ $(0.160, 0.752)$, $\hat{\tau}^2 = 2.27\times 10^{-7}$ $(3.81\times 10^{-8}, 2.27\times 10^{-5})$).

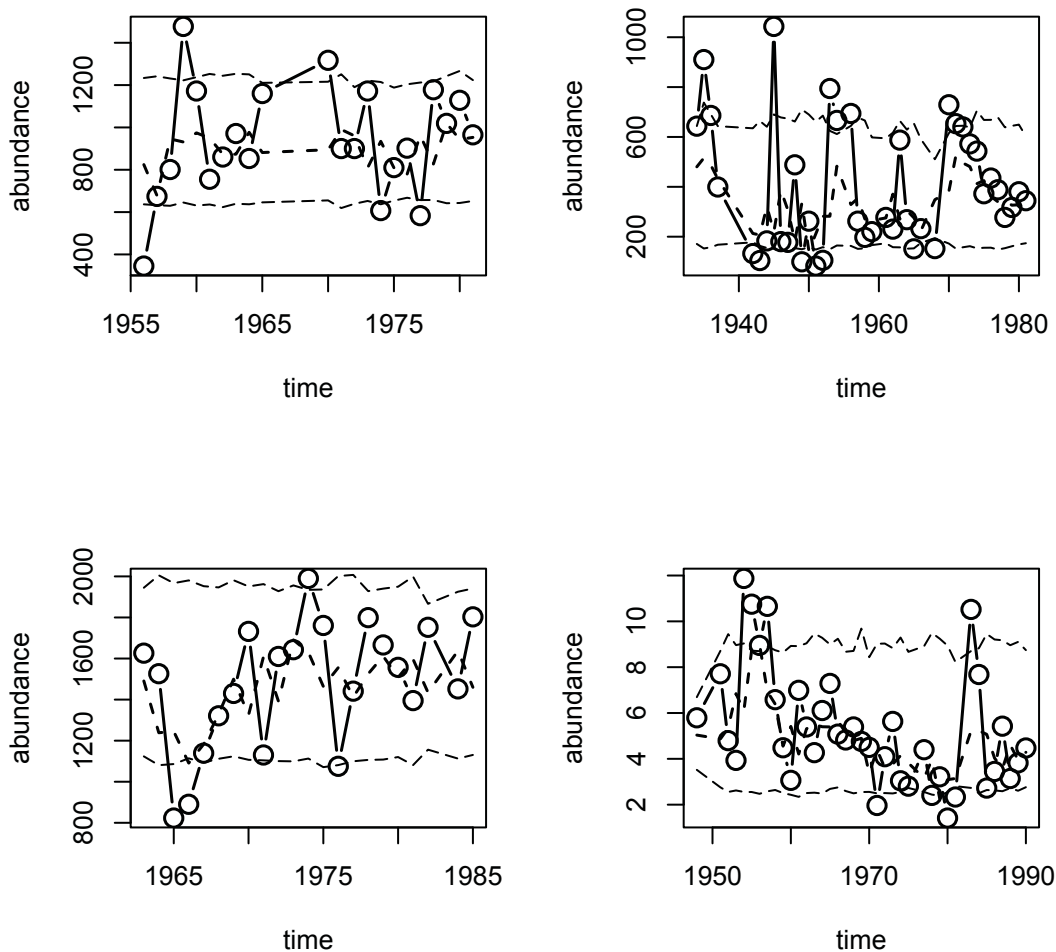
abundance
400
800
1200
1955
1965
1975
time
abundance
200
600
1000
1940
1960
1980
time
abundance
800
1200
1600
2000
1965
1975
1985
time
abundance
2
4
6
8
10
1950
1970
1990
time

Figure 1

Ecological Archives

Brian Dennis, José Miguel Ponciano. 201X. Density dependent state space model for population abundance data with unequal time intervals. Ecology VOL: pp-pp.

Appendix. Confidence intervals, hypothesis tests, and model selection for the Ornstein-Uhlenbeck state-space model.

The Ornstein-Uhlenbeck state-space (OUSS) model is given by $Y(t_i) = X(t_i) + F_i$, $dX(t) = \theta[\mu - X(t)]dt + \beta dW(t)$, where $Y(t_i)$ is the observed or estimated log-abundance of the population at time t_i, $X(t_i)$ is the actual (unobserved) log-abundance, where F_i has a normal distribution with mean 0 and variance τ^2, with F_i, F_j uncorrelated ($i \neq j$), $dW(t)$ has a normal distribution with a mean of 0 and a variance of dt (where the correlation between $dW(t_i)$ and $dW(t_j)$ is equal to 0 if $t_i \neq t_j$), μ is a real-valued parameter, and θ, β^2, and τ^2 are positive real-valued parameters.

Confidence intervals for the model parameters are readily obtained using parametric bootstrapping. For the OUSS model, parametric bootstrapping involves simulating 2000 or so data sets from the estimated OUSS model (the OUSS model equations evaluated at the maximum likelihood (ML) or restricted maximum likelihood (REML) estimates) and reestimating the model parameters for each simulated data set. Percentiles (or alternatively, median-adjusted percentiles) of the 2000 bootstrap parameter values form the ends of the confidence intervals (see Manly 1997).

Confidence intervals obtained with parametric bootstrapping are in general not symmetric and tend to have better coverage properties than intervals based on large sample ML theory (Pawitan 2001). Also, parametric bootstrap intervals are easily obtained for functions of parameters (such as the stationary variance of $X(t)$ given by $\beta^2/(2\theta)$) just by calculating the

value of the particular function for each set of bootstrap parameter values. One of the R programs accompanying this paper (Supplementary Material online) calculates parametric bootstrap confidence intervals.

A statistical hypothesis test of density independence versus density dependence can be performed for the OUSS model with parametric bootstrapping. The exponential growth state-space (EGSS) model serves as the null hypothesis of density independence, and the OUSS model serves as the alternative hypothesis of density dependence. The procedure represents an extension of the method of Dennis and Taper (1994), which used only process noise and equal time intervals. The procedure is to simulate 2000 or more data sets from the EGSS model that has been fitted to the data with ML estimates (as described in Humbert et al. 2009). The EGSS and the OUSS model are then fitted to each simulated data set with ML estimation, and a likelihood ratio statistic ($-2\log\left(\hat{L}_{\mathrm{EGSS}}/\hat{L}_{\mathrm{OUSS}}\right)$) is then calculated for each simulated data set. Here $\hat{L}$ represents a multivariate normal likelihood maximized under the EGSS or OUSS model. The proportion of the 2000 (or more) simulated likelihood ratio values that exceed the value of likelihood ratio statistic calculated for the data constitutes the P-value for the test. One of the R programs accompanying this paper (Supplementary Material online) calculates the parametric bootstrap likelihood ratio test of the EGSS model versus the OUSS model. The procedure basically tests for the presence of a tendency toward equilibration (or stationarity), and the ecological interpretation of such a test as biological density dependence requires caution (Wolda and Dennis 1993) in the absence of other biological information about the population.

Model selection can be performed among submodels, or among alternative model forms, with information criteria such as AIC and its variants (Burnham and Anderson 2002). The model selection indexes typically require the values of the maximized log-likelihoods for the various

models under question (or at least the differences of the maximized log-likelihoods for every pair as in Ponciano et al. 2009). For the OUSS model and its submodels, the maximized log-likelihood values are readily available as byproducts of model-fitting. However, one should not compare models with likelihoods arising from fundamentally different data, such as the raw observations for ML estimates and the differenced observations for REML estimates (or even first differences for OUSS/REML and second differences for EGSS/REML). Rather, the likelihoods for the models being compared should be defined for the same unique observations, which usually means comparing ML with ML.

LITERATURE CITED

Burnham, K. P. and D. R. Anderson, D. R. 2002. Model selection and multimodel inference: a practical information-theoretic approach, 2nd ed. Springer-Verlag, New York, New York, USA.

Dennis, B., and M. L. Taper. 1994. Density dependence in time series observations of natural populations: estimation and testing. Ecological Monographs 64:205-224.

Humbert, J.-Y., L. S. Mills, J. S. Horne, J. S., and B. Dennis. 2009. A better way to estimate population trend. Oikos 118:1940-1946.

Manly, B. F. J. 1997. Randomization, bootstrap and Monte Carlo methods in biology. Second edition. Chapman and Hall, London, UK.

Pawitan, Y. 2001. In all likelihood: statistical modelling and inference using likelihood. Oxford University Press, Oxford, UK.

Ponciano, J.M., M. L. Taper, B. Dennis and S. R. Lele. 2009. Hierarchical models in ecology: confidence intervals, hypothesis testing and model selection using data cloning. Ecology 90:356-362.

Wolda, H., and B. Dennis. 1993. Density dependence tests, are they? Oecologia 95:581-591.

Ecological Archives

Brian Dennis, José Miguel Ponciano. 201X. Density dependent state space model for population abundance data with unequal time intervals. *Ecology* VOL:pp-pp.

Supplement

R scripts for calculating maximum likelihood estimates, restricted maximum likelihood estimates, and parametric bootstrap likelihood ratio tests of density dependence, for the Ornstein Uhlenbeck state space model, using population abundance data having possibly unequal observation time intervals.

Authors

Brian Dennis

Department of Fish and Wildlife Sciences and Department of Statistical Science, University of Idaho, Moscow ID 83844-1136, USA

E-mail: brian@uidaho.edu

José Miguel Ponciano

Department of Biology, University of Florida, Gainesville, FL, 32611-8525, USA

E-mail: josemi@ufl.edu

Description

This supplement contains three R scripts: (1) RUNNING-ROUSS.R calculates maximum likelihood and restricted maximum likelihood estimates for the Ornstein

Uhlenbeck state space model (stationary and non-stationary), using population abundance data having possibly unequal observation time intervals. (2) RUNNING-PBLRT.R performs parametric bootstrap likelihood ratio test of exponential growth state space model vs. Ornstein-Uhlenbeck state space model. (3) ROUSSE-1.0.R contains the functions used in statistical inferences for the Ornstein-Uhlenbeck state space model. ROUSSE-1.0.R is called by both RUNNING-ROUSS.R and RUNNING-PBLRT.R and must be present (as a file by that name) in the working directory of R.

```
#=========================================================================
#  RUNNING-ROUSS.R:  calculates maximum likelihood and restricted
#  maximum likelihood estimates for the Ornstein-Uhlenbeck state space
#  model (stationary and non-stationary), using population abundance
#  data having possibly unequal observation time intervals.  The script
#  ROUSSE-1.0.R must be present in the working directory of R.
#  Be patient;  R is slow.
#=========================================================================

#-------------------------------------------------------------------------
#  Load all the OUSS model functions and needed packages
#-------------------------------------------------------------------------
library("MASS")
source("ROUSSE-1.0.R")

#-------------------------------------------------------------------------
#        USER INPUT SECTION
#-------------------------------------------------------------------------
#  User supplies time series data here into the vector "Observed.t.
#  User can substitute R statements to read population abundance data
#  from a file into the vector "Observed.t". No zeros! Do not change
#  the object name "Observed.t". Times of observation are entered into
#  the vector "Time.t". Do not change the object name "Time.t".

#  First example data set is bobcat (Lynx rufus) in Idaho, data set
#  212 from the Global Population Dynamics Database. Various other data
#  sets are also included.  Pick any of these data sets and associated
#  sampling years by "commenting out" the Idaho bobcat data and
#  "uncommenting" the desired data.

# Lynx rufus, from Idaho, GPPD data set 212.
Observed.t=c(346,675,802,1478,1173,756,861,972,854,1161,1318,901,901,
1173,608,811,903,584,1179,1020,1129,966)  #  No zeros!
Time.t=c(1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1970,1971,
1972,1973,1974,1975,1976,1977,1978,1979,1980,1981)
```

```
#  Linx rufus, from Florida, GPPD data set 211.
# Time.t=c(1946,1947,1948,1949,1950,1954,1955,1956,1957,1958,
# 1959,1960,1961,1963,1964,1965,1966,1967,1968,1975,1976,1977,
# 1978,1979,1980,1981)
# Observed.t=c(672,1028,538,566,300,400,400,400,400,300,250,
# 450,450,13,23,23,2,400,20,389,537,983,1698,1132,1702,1031)

# Lynx rufus, California, GPPD data set 208.
# Time.t=c(1934,1935,1936,1938,1940,1941,1942,1943,1944,1945,
# 1946,1947,1948,1949,1950,1951,1952,1954,1955,1956,1957,1958,
# 1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,
# 1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981)
# Observed.t=c(1994,1436,1290,2292,2776,3239,1923,2898,2063,1730,
# 1072,689,169,375,293,239,336,223,228,276,202,142,175,304,205,
# 295,361,221,221,241,244,381,588,319,588,686,1244,1393,2203,
# 3618,4445,6928,7809,9595,9337)

# Lynx rufus, Michigan, GPPD data set 218.
# Time.t=c(1936,1937,1939,1940,1941,1942,1943,1944,1945,1946,
# 1947,1948,1949,1950,1951,1952,1954,1955,1956,1957,1958,1962,
# 1963,1964,1966, 1969,1976,1977,1978,1979,1980,1981)
# Observed.t=c(1134,811,598,528,529,375,2538,2802,2910,2363,
# 2174,2063,1547,1753,1443,1836,696,847,880,762,200,588,494,265,
# 400,300,341,331,386,597,223,200)

# Lynx rufus, Maine, GPPD data set 216.
# Time.t=c(1934,1935,1936,1937,1942,1943,1944,1945,1946,1947,
# 1948,1949,1950,1951,1952,1953,1954,1956,1957,1958,1959,1961,
# 1962,1963,1964,1965,1966,1968,1970,1971,1972,1973,1974,1975,
# 1976,1977,1978,1979,1980,1981)
# Observed.t=c(644,911,687,400,133,105,184,1044,181,178,489,100,
# 263,83,106,795,667,695,263,198,221,278,231,588,269,152,233,153,
# 730,654,641,573,544,373,436,389,278,318,381,345)

# Lynx rufus, Wisconsin, GPPD data set 239.
# Time.t=c(1934,1935,1936,1937,1938,1940,1941,1942,1943,1944,
# 1945,1946,1947,1948,1949,1950,1951,1952,1954,1956,1959,1960,
# 1969,1970,1971,1974,1975,1976,1977,1978,1979,1980,1981)
# Observed.t=c(302,428,513,461,593,180,283,191,765,384,1048,577,
# 427,437,482,525,724,740,524,321,479,869,148,148,147,205,223,
# 275,163,223,131,81,168)

# Elk, central valley of Grand Teton National Park, cited in
# Dennis and Taper (1994 Ecological Monographs).
# Observed.t = c(1627,1527,824,891,1140,1322,1431,1733,1131,1611,
# 1644,1991,1762,1076,1442,1800,1667,1558,1396,1753,1453,1804)
# Time.t = c(1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,
# 1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1984,1985)

# Rangeland grasshoppers, MT western mountain region, from
# Kemp and Dennis 1993 Oecologia).
# Observed.t=c(5.7981,7.7194,4.8022,3.9397,11.8806,10.7568,8.9586,
# 10.6619,6.5895,4.4905,3.0684,6.9973,5.3986,4.2777,6.1166,7.2989,
# 5.0850,4.8298,5.3997,4.7679,4.5073,1.9714,4.1007,5.6403,3.0492,
# 2.8144,4.4071,2.4121,3.2233,1.4236,2.3404,10.5283,7.6872,2.7305,
# 3.4570,5.4336,3.1487,3.8315,4.4805)
# Time.t=c(1948,1951,1952,1953,1954,1955,1956,1957,1958,1959,1960,
```

```
# 1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,
# 1974,1975,1977,1978,1979,1980,1981,1983,1984,1985,1986,1987,1988,
# 1989,1990)

# American Redstart, record # 02014 3328 08636 from the North
# American Breeding Bird Survey 1966-95 (Table 1 in Dennis et
# al. 2006 Ecological Monographs).
# Observed.t = c(18,10,9,14,17,14,5,10,9,5,11,11,4,5,4,8,2,3,9,2,4,
# 7,4,1,2,4,11,11,9,6)
# Time.t = c(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
# 21,22,23,24,25,26,27,28,29)

# Log-transform the observations to carry out all the calculations
# in this program.
log.obs = log(Observed.t)
#-----------------------------------------------------------------------


#-----------------------------------------------------------------------
#        PARAMETER ESTIMATION, PARAMETRIC BOOTSTRAP AND PREDICTIONS
#-----------------------------------------------------------------------
# Before doing the calculations, the user has to specify ONLY the
# following 4 options:

# 1. Do you want to compute the ML estimates or the REML estimates?

method = "REML" # alternatively, set method = "ML"

# 2. Do you want to plot the predictions?
pred.plot = "TRUE" # Set it to "FALSE" if you do not want to plot the
                   # predictions

# 3. Do you want to plot the parametric bootstrap distribution of the
# estimates?
pboot.plot = "TRUE" # Set it to "FALSE" if you do not want to plot the
                    # bootstrap distribution of the estimates

# 4. How many bootstrap replicates?
NBoot = 1000


#-----------------------------------------------------------------------
#  5. THE FOLLOWING LINES OF CODE COMPUTE THE ESTIMATES, PREDICTIONS,
#  AND PARAMETRIC BOOTSTRAP CONFIDENCE INTERVALS. THE USER DOES NOT
#  NEED TO MODIFY THESE LINES OF CODE.
#
#  THE OUTPUT OF THE FUNCTION 'ROUSS.CALCS' IS A LIST AND THE USER
#  CAN RETRIEVE EACH OF THE LIST ELEMENTS PRINTED AND SAVED IN THE
#  OBJECT "all.results".
#
#  THE 95% PARAMETRIC BOOTSTRAP FOR BOTH, THE PARAMETERS AND THE
#  PREDICTIONS ARE COMPUTED BY THE FUNCTION "ROUSS.CALCS".
#
#-----------------------------------------------------------------------

all.results = ROUSS.CALCS(Yobs=log.obs,Tvec=Time.t,pmethod=method,
              nboot=NBoot,plot.pred=pred.plot,
```

```
                plot.bootdists=pboot.plot)


#=========================================================================
#  RUNNING-PBLRT.R:  performs parametric bootstrap likelihood ratio
#  test of exponential growth state space model vs. Ornstein-Uhlenbeck
#  state space model.  The script ROUSS-1.0.R must be in the working
#  directory of R.
#=========================================================================

#-------------------------------------------------------------------------
#  Load all the OUSS model functions and needed packages
#-------------------------------------------------------------------------
library("MASS")
source("ROUSSE-1.0.R")

#-------------------------------------------------------------------------
#        USER INPUT SECTION
#-------------------------------------------------------------------------
#  User supplies time series data here into the vector "Observed.t.
#  User can substitute R statements to read population abundance data
#  from a file into the vector "Observed.t". Do not change the object
#  name "Observed.t".
#  Times of observation are entered into the vector "Time.t". Do not
#  change the object name "Time.t".

# Lynx rufus, from Idaho, GPPD data set 212.
Observed.t=c(346,675,802,1478,1173,756,861,972,854,1161,1318,901,901,
1173,608,811,903,584,1179,1020,1129,966)  #  No zeros!
Time.t=c(1956,1957,1958,1959,1960,1961,1962,1963,1964,1965,1970,1971,
1972,1973,1974,1975,1976,1977,1978,1979,1980,1981)

# Lynx rufus, from Florida, GPPD data set 211.
# Time.t=c(1946,1947,1948,1949,1950,1954,1955,1956,1957,1958,
# 1959,1960,1961,1963,1964,1965,1966,1967,1968,1975,1976,1977,
# 1978,1979,1980,1981)
# Observed.t=c(672,1028,538,566,300,400,400,400,400,300,250,
# 450,450,13,23,23,2,400,20,389,537,983,1698,1132,1702,1031)

# Lynx rufus, California, GPPD data set 208.
# Time.t=c(1934,1935,1936,1938,1940,1941,1942,1943,1944,1945,
# 1946,1947,1948,1949,1950,1951,1952,1954,1955,1956,1957,1958,
# 1959,1960,1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,
# 1971,1972,1973,1974,1975,1976,1977,1978,1979,1980,1981)
# Observed.t=c(1994,1436,1290,2292,2776,3239,1923,2898,2063,1730,
# 1072,689,169,375,293,239,336,223,228,276,202,142,175,304,205,
# 295,361,221,221,241,244,381,588,319,588,686,1244,1393,2203,
# 3618,4445,6928,7809,9595,9337)

# Lynx rufus, Michigan, GPPD data set 218.
# Time.t=c(1936,1937,1939,1940,1941,1942,1943,1944,1945,1946,
# 1947,1948,1949,1950,1951,1952,1954,1955,1956,1957,1958,1962,
# 1963,1964,1966,1969,1976,1977,1978,1979,1980,1981)
# Observed.t=c(1134,811,598,528,529,375,2538,2802,2910,2363,
# 2174,2063,1547,1753,1443,1836,696,847,880,762,200,588,494,265,
# 400,300,341,331,386,597,223,200)
```

```
# Lynx rufus, Maine, GPPD data set 216.
# Time.t=c(1934,1935,1936,1937,1942,1943,1944,1945,1946,1947,
# 1948,1949,1950,1951,1952,1953,1954,1956,1957,1958,1959,1961,
# 1962,1963,1964,1965,1966,1968,1970,1971,1972,1973,1974,1975,
# 1976,1977,1978,1979,1980,1981)
# Observed.t=c(644,911,687,400,133,105,184,1044,181,178,489,100,
# 263,83,106,795,667,695,263,198,221,278,231,588,269,152,233,153,
# 730,654,641,573,544,373,436,389,278,318,381,345)

# Lynx rufus, Wisconsin, GPPD data set 239.
# Time.t=c(1934,1935,1936,1937,1938,1940,1941,1942,1943,1944,
# 1945,1946,1947,1948,1949,1950,1951,1952,1954,1956,1959,1960,
# 1969,1970,1971,1974,1975,1976,1977,1978,1979,1980,1981)
# Observed.t=c(302,428,513,461,593,180,283,191,765,384,1048,577,
# 427,437,482,525,724,740,524,321,479,869,148,148,147,205,223,
# 275,163,223,131,81,168)

# Elk, central valley of Grand Teton National Park, cited in
# Dennis and Taper (1994 Ecological Monographs).
# Observed.t = c(1627,1527,824,891,1140,1322,1431,1733,1131,1611,
# 1644,1991,1762,1076,1442,1800,1667,1558,1396,1753,1453,1804)
# Time.t = c(1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,
# 1973,1974,1975,1976,1977,1978,1979,1980,1981,1982,1984,1985)

# Rangeland grasshoppers, MT western mountain region, from
# Kemp and Dennis 1993 Oecologia).
# Observed.t=c(5.7981,7.7194,4.8022,3.9397,11.8806,10.7568,8.9586,
# 10.6619,6.5895,4.4905,3.0684,6.9973,5.3986,4.2777,6.1166,7.2989,
# 5.0850,4.8298,5.3997,4.7679,4.5073,1.9714,4.1007,5.6403,3.0492,
# 2.8144,4.4071,2.4121,3.2233,1.4236,2.3404,10.5283,7.6872,2.7305,
# 3.4570,5.4336,3.1487,3.8315,4.4805)
# Time.t=c(1948,1951,1952,1953,1954,1955,1956,1957,1958,1959,1960,
# 1961,1962,1963,1964,1965,1966,1967,1968,1969,1970,1971,1972,1973,
# 1974,1975,1977,1978,1979,1980,1981,1983,1984,1985,1986,1987,1988,
# 1989,1990)

# American Redstart, record # 02014 3328 08636 from the North
# American Breeding Bird Survey 1966-95 (Table 1 in Dennis et
# al. 2006 Ecological Monographs).
# Observed.t = c(18,10,9,14,17,14,5,10,9,5,11,11,4,5,4,8,2,3,9,2,4,
# 7,4,1,2,4,11,11,9,6)
# Time.t = c(0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
# 21,22,23,24,25,26,27,28,29)

# Log-transform the observations to carry out all the calculations
# in this program.
log.obs = log(Observed.t)
#-------------------------------------------------------------------------

#-------------------------------------------------------------------------
#   PARAMETER ESTIMATION AND PARAMETRIC BOOTSTRAP LIKELIHOOD RATIO TEST
#-------------------------------------------------------------------------
#  The output contains 6 objects:
#  Object 1:  "egssml": this is a list that contains the ml estimates
#  of the EGSS model, the maximized log likelihood and the AIC value
#  under that model.
#
```

```
#  Object 2:  "roussml": this is a list that contains the ml estimates
#  of the OUSS model, the maximized log likelihood and the AIC value
#  under that model.
#
#  Object 3:  "Lam.obs":  this is the observed log-likelihood ratio
#  statistic given by -2*ln[L(H0)/L(H1)].
#
#  Object 4:  "Lam.vec":  this is a vector of the bootstrapped values
#  of -2*ln[L(H0)/L(H1)].
#
#  Object 5:  "pvalue":  this is the proportion of the bootstrapped
#  values of -2*ln[L(H0)/L(H1)] that are more extreme than the observed
#  value of -2*ln[L(H0)/L(H1)].
#
#  Object 6:  "Decision.rule": this is a character object that prints
#  out the decision of the test (Fail to reject or reject the null
#  hypothesis of the EGSS density independent model).
#
#  Before doing the calculations, the user has to specify ONLY the
#  following 3 options:

# 1. How many bootstrap replicates do you want to use? (Default is
# one thousand)
NBoot = 1000

# 2. At what significance level alpha do you want to test the null
# hypothesis of density independence vs. the alternative of density
# dependence?
my.alpha = 0.05

# 3. Do you want to plot the sampling distribution of the log-
# likelihood ratio statistic -2*ln[L(H0)/L(H1)]?
plot.it = TRUE

# Performs the bootstrap analysis (be patient):
pblrt.trial = PBLRT.ddpcont(B=NBoot, Yobs=log.obs, Tvec=Time.t,
alpha=my.alpha, plot.PBLR=plot.it)

# Printing results:

#  Object 1:  "egssml": this is a list that contains the ml estimates
#  of the EGSS model, the maximized log likelihood and the AIC value
#  under that model.
pblrt.trial$egssml

#  Object 2:  "roussml": this is a list that contains the ml estimates
#  of the OUSS model, the maximized log likelihood and the AIC value
#  under that model.
pblrt.trial$roussml

#  Object 3:  "Lam.obs":  this is the observed value of the log-
#  likelihood ratio statistic -2*ln[L(H0)/L(H1)]
pblrt.trial$Lam.obs

#  Object 5:  "pvalue":  this is the proportion of the bootstrapped
#  values of -2*ln[L(H0)/L(H1)] that are more extreme than the observed
#  value of -2*ln[L(H0)/L(H1)].
```

```
pblrt.trial$pvalue

#  Object 6:  "Decision.rule": this is a character object that prints
#  out the decision of the test (Fail to reject or reject the null
#  hypothesis of density independence)
pblrt.trial$Decision.rule

#  To plot the sampling distribution of the likelihood ratio test, we
#  recommend producing a trial histogram first, and after a visual
#  inspection, eliminating from the plotting range the numerical
#  extremes, which should be very few.
#
#  The script lines below will display an example.

#  Histogram of the pboot LRT with the observed LRT in red.
hist(pblrt.trial$Lam.vec)
abline(v=pblrt.trial$Lam.obs,lwd=2,col="red")

#  After plotting, note that the American Redstart example has
#  numerical extremes around -600 and around +600.
#  Re-do the plot zooming into the bulk of the bootstrap distribution.
hist(pblrt.trial$Lam.vec[abs(pblrt.trial$Lam.vec)<100],
   xlab="Bootstrap values of -2*ln[L(H0)/L(H1)]",main="")
abline(v=pblrt.trial$Lam.obs,lwd=2,col="red")

#=========================================================================
#  ROUSSE-1.0.R:  Functions used in statistical inferences for the
#  Ornstein-Uhlenbeck state space model.  This script should be present
#  in the working directory of R before running either RUNNING-ROUSS.R
#  or RUNNING-PBLRT.R.
#=========================================================================

##  PART I)  OUSS model R functions
##  PART II) EGSS model R functions
##  PART III) Density dependence vs. Density independence
##     Parametric Bootstrap Likelihood Ratio Test (PBLRT) functions

##  All EGSS Functions originally published in
##  Humbert, J.-Y., Mills, S., Horne, J. S. and Dennis, B. 2009. A
##  better way to estimate population trend. Oikos 118:1940-1946.

# PART I):  The OUSS functions ------------------------------------

# 1. Negative log-likelihood score, for ML estimation or Model
# Selection.
#  ML objective function "negloglike.OU.ml" is negative of log-
#  likelihood.
#  The function uses a multivariate normal log-
#  likelihood (Eq. 19).  The three function arguments are:
#  fguess, vector of parameters (transformed to the real line),
#  yt, vector of time series of log observed abundances (cannot
#  have 0's ), tt, vector of observation times.
negloglike.OU.ml <- function(fguess,yt,tt){
   mu = fguess[1]
   guess = exp(fguess[2:4]) # Constrains parameters > 0
   theta = guess[1]
   betasq = guess[2]
```

```
   tausq = guess[3]
   q = length(yt) - 1
   qp1 = q+1
   Var.inf = betasq/(2*theta) # Stationary variance
   ss = tt[2:qp1] - tt[1:q]
   t.cols = matrix(rep(tt,each=qp1),nrow=qp1,ncol=qp1, byrow=FALSE)
   t.rows = t(t.cols)
   abs.diffs = abs(t.rows-t.cols)
   V = Var.inf*exp(-theta*abs.diffs)
   diag(V) = diag(V) + rep(tausq,qp1)
   mu.vec = rep(mu,qp1)
   neglogl = (qp1/2)*log(2*pi)+(1/2)*log(det(V))+
      (1/2)*(yt-mu.vec)%*%ginv(V)%*%(yt-mu.vec)
   return(neglogl)
}

#  2. Negative log-likelihood for REML estimation or Model
#  Selection (Eq. 22).
#  REML objective function "negloglike.OU.reml" is negative of
#  log-likelihood for first differences of the log-scale observations.
#  The three function arguments are:
#  phi, vector of parameters (transformed to the real line),
#  yt, vector of time series observations (log scale),
#  tt, vector of observation times.
#  The function performs the differencing.
negloglike.OU.reml=function(phi,yt,tt) {
   theta = exp(phi[1])          #  Constrains theta > 0.
   betasq = exp(phi[2])         #  Constrains betasq > 0.
   tausq = exp(phi[3])          #  Constrains tausq > 0.
   Var.inf = betasq/(2*theta)   #  Recurring quantity.
   q = length(yt)-1
   qp1 = q+1
   ss = tt[2:qp1]-tt[1:q]       #  Time intervals.
   t.cols = matrix(rep(tt,each=qp1),nrow=qp1,ncol=qp1, byrow=FALSE)
   t.rows = t(t.cols)
   abs.diffs = abs(t.rows-t.cols)
   Sigma.mat = Var.inf*exp(-theta*abs.diffs)
   Itausq = matrix(0,qp1,qp1)
   diag(Itausq) = rep(tausq,qp1)
   V = Sigma.mat+Itausq
   Dmat = cbind(-diag(1,q),matrix(0,q,1)) +
      cbind(matrix(0,q,1),diag(1,q))        #  Differencing matrix.
   Phi.mat = Dmat%*%V%*%t(Dmat)             #  REML var-cov matrix.
   wt = yt[2:qp1]-yt[1:q]
   ofn = (q/2)*log(2*pi)+0.5*log(det(Phi.mat)) +
      0.5*wt%*%ginv(Phi.mat)%*%wt         #  ginv() is numerically
                                          #  more robust than solve().
   return(ofn)
}

# 3. rand.MVN:  Multivariate Normal random number generator:
#    n = number of random samples of a MVN vector,
#    mu = mean vector of the MVN distribution,
#    cov.mat = Variance-covariance matrix of the MVN distribution.
randmvn = function(n,mu.vec, cov.mat) {
   p = length(mu.vec)
   Tau = chol(cov.mat)
```

```
   Zmat = matrix(rnorm(n=p*n,mean=0,sd=1),nrow=p,ncol=n)
          #  generate normal deviates outside loop.
   out = matrix(0,nrow=p,ncol=n)
   for(i in 1:n) {
      Z = Zmat[,i]
      out[,i] = t(Tau)%*%Z + mu.vec
   }
   return(out)
}

# 4. Simulation function: requires as input parameter values and
#  a vector of observation times. The multivariate Normal model (see
#  eq. 19) is used to simulate the data.
#    nsims = number of bootstrap replicates to simulate.
#    parms = vector of parameter values = c(mu_hat, theta_hat,
#       betasq_hat,tausq_hat), where 'hat' denotes neither the
#       ML or the REML estimates.
#    Tvec = vector of ORIGINAL observation times (t_0, t_1, t_2, ...,
#       t_q)
ROUSS.sim = function(nsims,parms,Tvec) {
   tt = Tvec-Tvec[1]
   mu = parms[1]
   theta = parms[2]
   betasq = parms[3]
   tausq = parms[4]
   q = length(tt)-1
   qp1 = q+1
   Var.inf = betasq/(2*theta)
   ss = tt[2:qp1]-tt[1:q]
   t.cols = matrix(rep(tt,each=qp1),nrow=qp1,ncol=qp1,byrow=FALSE)
   t.rows = t(t.cols)
   abs.diffs = abs(t.rows-t.cols)
   V = Var.inf*exp(-theta*abs.diffs)
   diag(V) = diag(V)+rep(tausq,qp1)
   m.vec = rep(mu,qp1)
   out = randmvn(n=nsims,mu.vec=m.vec,cov.mat=V)
   return(out)
}

# 5. Computing rough initial guesses for ML estimation.
#     Yobs = log(Observed time series of abundances),
#     Tvec = vector of sampling times.
guess.calc = function(Yobs,Tvec) {
   T.t = Tvec-Tvec[1]       #  For calculations, time starts at zero.
   q = length(Yobs)-1       #  Number of time series transitions, q.
   qp1 = q+1                #  q+1 gets used a lot, too.
   S.t = T.t[2:qp1]-T.t[1:q]  #  Time intervals.
   Ybar = mean(Yobs)
   Yvar = sum((Yobs-Ybar)*(Yobs-Ybar))/q
   mu1 = Ybar             # Kludge an initial value for theta based
                          #  on mean of Y(t+s) given Y(t).
   th1 = -mean(log(abs((Yobs[2:qp1]-mu1)/(Yobs[1:q]-mu1)))/S.t)
   bsq1 = 2*th1*Yvar/(1+2*th1)       # Moment estimate using stationary
   tsq1 = bsq1                       # variance, with betasq=tausq.
   out = c(mu1,th1,bsq1,tsq1)
   return(abs(out))
}
```

```
# 6.  Computing the ML estimates of the OUSS model, maximized log-
# likelihood and the AIC score:
#     Yobs = log(Observed time series of abundances),
#     Tvec = vector of sampling times,
#     parms.guess = vector of initial guesses for the parameters
#       = c(mu_hat,theta_hat,betasq_hat,tausq_hat)
ROUSS.ML = function(Yobs,Tvec,parms.guess) {
   tt = Tvec-Tvec[1]
   q = length(tt)-1
   qp1 = q+1
   guess.optim = c(parms.guess[1],log(parms.guess[2:4]))
   optim.out = optim(par=guess.optim,fn=negloglike.OU.ml,
      method="Nelder-Mead",yt=Yobs,tt=tt)
   mles = c(optim.out$par[1],exp(optim.out$par[2:4]))
   lnL.hat = - optim.out$value[1]
   AIC = -2*lnL.hat + 2*4       # where 4 = length(mles)...
   out = list(mles=mles, lnL.hat = lnL.hat, AIC=AIC)
   return(out)
}

# 7.  Computing the REML estimates of the OUSS model, maximized log-
# likelihood and the AIC score:
#     Yobs = log(Observed time series of abundances),
#     Tvec = vector of sampling times,
#     parms.guess = vector of initial guesses for the parameters
#      = c(mu_hat,theta_hat,betasq_hat,tausq_hat).
ROUSS.REML = function(Yobs,Tvec,parms.guess) {
   tt = Tvec-Tvec[1]
   q = length(tt)-1
   qp1 = q+1
   ss = tt[2:qp1]-tt[1:q]
   guess.optim = log(parms.guess[2:4])
   optim.out = optim(par = guess.optim,fn=negloglike.OU.reml,
      method="Nelder-Mead",yt=Yobs,tt=tt)
   remls = exp(optim.out$par)
   lnL.hat = -optim.out$value[1]
   theta.reml = remls[1]
   betasq.reml = remls[2]
   tausq.reml = remls[3]
   Var.inf = betasq.reml/(2*theta.reml)
   vx = matrix(1,qp1,qp1)
   for (ti in 1:q) {
      vx[(ti+1):qp1,ti] = exp(-theta.reml*cumsum(ss[ti:q]))
      vx[ti,(ti+1):qp1] = vx[(ti+1):qp1,ti]
   }
   Sigma.mat = vx*Var.inf
   Itausq = matrix(0,qp1,qp1)
   diag(Itausq) = rep(tausq.reml,qp1)
   V.reml = Sigma.mat+Itausq
   j = matrix(1,qp1,1)
   Vinv = ginv(V.reml)
   mu.reml = (t(j)%*%Vinv%*%Yobs)/(t(j)%*%Vinv%*%j)  #  REML estimate
                                                     # of mu.
   out = list(remls=c(mu.reml,theta.reml,betasq.reml,tausq.reml),
      lnLhat = lnL.hat)
   return(out)
```

```
}

# 8. Parameteric Bootstrap function.
# B = number of bootstrap replicates,
# parms = c(mu.mle,theta.mle,betasq.mle,tausq.mle) = ML estimates of
# the parameters with the original time series,
# tt = vector of observation times (t_0, t_1, t_2, ..., t_q) from the
# original time series.
# If REML="TRUE", then the parametric bootsrap is computed for REML
# estimation.  In that case, 'parms' must contain the REML estimates
# for the original time series.
ROUSS.pboot = function(B=2,parms,Yobs,Tvec,REML="FALSE",
                  plot.it="FALSE") {
   tt = Tvec-Tvec[1]
   nparms = length(parms)
   preds.boot = matrix(0,nrow=B,ncol=length(tt))
   if(REML=="TRUE") {
      boot.remles = matrix(0,nrow=B,ncol=nparms+1)
      all.sims = ROUSS.sim(nsims=B,parms=parms,Tvec=Tvec)
      reml.preds = ROUSS.predict(parms=parms,Yobs=Yobs,Tvec=Tvec,
                     plot.it="FALSE")[,2]
      for(b in 1:B) {
         bth.timeseries = all.sims[,b]
         remles.out = ROUSS.REML(Yobs=bth.timeseries,Tvec=Tvec,
                        parms.guess=parms)
         boot.remles[b,] = c(remles.out$remls,remles.out$lnLhat)
         preds.boot[b,] = ROUSS.predict(parms=remles.out$remls,
                             Yobs=bth.timeseries,Tvec=Tvec,
                              plot.it="FALSE")[,2]
      }
      CIs.mat = apply(boot.remles,2,FUN=function(x){quantile(x,
                  probs=c(0.025,0.975))})
      CIs.mat = rbind(CIs.mat[1,1:4],parms,CIs.mat[2,1:4])
      rownames(CIs.mat) = c("2.5%","REMLE","97.5%")
      colnames(CIs.mat) = c("mu","theta","betasq","tausq")
      preds.CIs = apply(preds.boot,2,FUN=function(x){quantile(x,
                    probs=c(0.025,0.975))})
      preds.CIs = t(rbind(Tvec,preds.CIs[1,],reml.preds,
                    preds.CIs[2,]))
      colnames(preds.CIs) = c("Year","2.5%","REMLE","97.5%")
      boot.list = list(boot.remles=boot.remles,CIs.mat=CIs.mat,
                    preds.CIs=preds.CIs)
      if(plot.it=="TRUE") {
         X11()
         par(mfrow=c(2,2))
         hist(boot.remles[,1],main=expression(hat(mu)),xlab="")
         abline(v=parms[1],lwd=2,col="red")
         hist(boot.remles[,2],main=expression(hat(theta)),
            xlab="")
         abline(v=parms[2],lwd=2,col="red")
         hist(boot.remles[,3],main=expression(hat(beta^2)),
            xlab="")
         abline(v=parms[3],lwd=2,col="red")
         hist(boot.remles[,4],main=expression(hat(tau^2)),
            xlab="")
         abline(v=parms[4],lwd=2,col="red")
      }
```

```
      return(boot.list)
   } else {
      boot.mles = matrix(0,nrow=B,ncol=nparms+2)
      all.sims = ROUSS.sim(nsims=B,parms=parms,Tvec=Tvec)
      ml.preds = ROUSS.predict(parms=parms,Yobs=Yobs,
                    Tvec=Tvec,plot.it="FALSE")[,2]
      for(b in 1:B) {
         bth.timeseries = all.sims[,b]
         mles.out = ROUSS.ML(Yobs=bth.timeseries,Tvec=Tvec,
                       parms.guess=parms)
         boot.mles[b,] = c(mles.out$mles, mles.out$lnL.hat,
                             mles.out$AIC)
         preds.boot[b,] = ROUSS.predict(parms=mles.out$mles,
                              Yobs=bth.timeseries,Tvec=Tvec,
                              plot.it="FALSE")[,2]
      }
      CIs.mat = apply(boot.mles,2,FUN=function(x){quantile(x,
                   probs=c(0.025,0.975))})
      CIs.mat = rbind(CIs.mat[1,1:4],parms,CIs.mat[2,1:4])
      rownames(CIs.mat) = c("2.5%","MLE","97.5%")
      colnames(CIs.mat) = c("mu","theta","betasq","tausq")
      preds.CIs = apply(preds.boot,2,FUN=function(x){quantile(x,
                    probs=c(0.025,0.975))})
      preds.CIs = t(rbind(Tvec,preds.CIs[1,],ml.preds,preds.CIs[2,]))
      colnames(preds.CIs) = c("Year","2.5%","MLE","97.5%")
      boot.list = list(boot.mles=boot.mles,CIs.mat=CIs.mat,
                    preds.CIs=preds.CIs)
      if(plot.it=="TRUE") {
         X11()
         par(mfrow=c(2,2))
         hist(boot.mles[,1],main=expression(hat(mu)),
            xlab="")
         abline(v=parms[1],lwd=2,col="red")
         hist(boot.mles[,2],main=expression(hat(theta)),
            xlab="")
         abline(v=parms[2],lwd=2,col="red")
         hist(boot.mles[,3],main=expression(hat(beta^2)),
            xlab="")
         abline(v=parms[3],lwd=2,col="red")
         hist(boot.mles[,4],main=expression(hat(tau^2)),
            xlab="")
         abline(v=parms[4],lwd=2,col="red")
      }
      return(boot.list)
   }                    # End if/else
}

# ROUSS.pboot(B=20,parms=linx.remles,Yobs=log(Observed.t),
#    Tvec=Time.t,REML="TRUE",plot.it="FALSE")

# 9. Function to compute the predicted trajectory of the unobserved
# process.  The arguments are:
#    parms = ML or REML estimates of c(mu,theta,betasq,tausq),
#       whichever you prefer,
#    Yobs  = Log observations,
#    Tvec  = vector of original observation times (t_0,t_1,...,t_q).
ROUSS.predict = function(parms,Yobs,Tvec,plot.it="TRUE") {
```

```
   qp1 = length(Yobs)
   q = qp1-1
   tt = Tvec - Tvec[1]
   ss = tt[2:qp1] - tt[1:q]
   mu = parms[1]
   theta = parms[2]
   betasq = parms[3]
   tausq = parms[4]
   Var.inf = betasq/(2*theta)
   t.cols = matrix(rep(tt,each=qp1),nrow=qp1,ncol=qp1,byrow=FALSE)
   t.rows = t(t.cols)
   abs.diffs = abs(t.rows-t.cols)
   Sigma.mat = Var.inf*exp(-theta*abs.diffs)
   Itausq = matrix(0,qp1,qp1)
   diag(Itausq) = rep(tausq,qp1)
   V = Sigma.mat+Itausq
   Predict.t = rep(0,qp1)
   Muvec = rep(mu,q)
   for (tj in 1:qp1) {
      Y.omitj = Yobs[-tj]   #  Omit observation at time tj.
      V.omitj = V[-tj,-tj]  #  Omit row tj and col tj from var-cov
                            #    matrix.
      V12 = V[tj,-tj]       #  Submatrix:  row tj without col tj.
      Predict.t[tj] = mu+V12%*%ginv(V.omitj)%*%(Y.omitj-Muvec)
               #  Usual expression for conditional MV normal mean.
   }
   Predict.t = exp(Predict.t)
   if(plot.it=="TRUE") {
      #  Plot the data & model-fitted values
      X11()
      plot(Time.t,exp(Yobs),xlab="time",ylab="abundance",lty=1,
         type="b",cex=1.5,lwd=1.5)
      #  Population data are circles.
      points(Tvec,Predict.t,type="l",lty=2,lwd=1.5)
      #  Predicted abundances are dashed line.
   }
   return(cbind(Tvec,Predict.t))
}

# 10. Function to run the estimation, compute the predictions and
# run a parametric bootstrap.
ROUSS.CALCS = function(Yobs,Tvec,pmethod="ML",nboot,
                  plot.pred="TRUE",plot.bootdists="TRUE") {
   # 10.1 Compute a rough guess of the parameter estimates to
   #  initialize the search:
   guesscalc = guess.calc(Yobs=log.obs,Tvec=Tvec)
   # 10.2 Compute either the ML or the REML estimates, according
   #  to what was specified in point 1 above.
   if (pmethod=="ML") {
      best.guess = ROUSS.ML(Yobs=Yobs,Tvec=Tvec,
                     parms.guess=guesscalc)
      AIC = best.guess[[3]]
      reml.option = "FALSE"
   } else if (pmethod=="REML") {
      best.guess = ROUSS.REML(Yobs=Yobs,Tvec=Tvec,
                     parms.guess=guesscalc)
      reml.option = "TRUE"
```

```
   } else {
      print("Error: ML and REML are the only options allowed for
        'method' ")
   }
   # 10.3 Parameter estimates and maximized log-likelihood
   # (will be printed at the end).
   parms.est = best.guess[[1]]
   lnLhat = best.guess[[2]]

   # 10.4 Computing the predictions.
   # rouss.preds = ROUSS.predict(parms=parms.est,
   #               Yobs=Yobs,Tvec=Tvec,plot.it=plot.pred)
   # print("These are the predictions of what the true, unobserved
   #    population abundances were")
   # print(rouss.preds)

   # 10.5 Parametric bootstrap: computing both, parameters and
   # predictions 95 % CI's.
   pboot.calcs = ROUSS.pboot(B=nboot,parms=parms.est,Yobs=Yobs,
                   Tvec=Tvec,REML=reml.option,plot.it=plot.bootdists)
   print("These are the predictions of what the true, unobserved
      population abundances were, along with 95% CI's")
   print(pboot.calcs$preds.CIs)
   rouss.preds = ROUSS.predict(parms=parms.est,Yobs=Yobs,
                  Tvec=Tvec,plot.it=plot.pred)
   if(plot.pred=="TRUE") {
      points(Tvec,pboot.calcs$preds.CIs[,2],type="l",col="black",lty=5)
      points(Tvec,pboot.calcs$preds.CIs[,4],type="l",col="black",lty=5)
   }
   print("This is the matrix of Parametric Bootstrap 95% CI's
      along with the estimates")
   print(pboot.calcs$CIs.mat)
   print("Maximized log-likelihood score")
   print(lnLhat)
   if(pmethod=="ML") {
      print("AIC score")
      print(AIC)
      out = list(parms.est=parms.est,lnLhat=lnLhat,AIC=AIC,
               pbootmat=pboot.calcs[[1]],pboot.cis=pboot.calcs[[2]],
               pboot.preds=pboot.calcs$preds.CIs)
   } else {
      out = list(parms.est=parms.est,lnLhat=lnLhat,
               pbootmat=pboot.calcs[[1]],pboot.cis=pboot.calcs[[2]],
               pboot.preds=pboot.calcs$preds.CIs)
   }
   return(out)
}

# PART II):  The EGSS functions -------------------------------

# 11.  Negative Log-Likelihood function for the EGSS model.
negloglike.EGSS.ml = function(theta,yt,tt) {
   mu = theta[1]
   sigmasq = exp(theta[2])
   tausq = exp(theta[3])
   xo = theta[4]
   q = length(yt) - 1
```

```
   qp1 = q+1
   yt = matrix(yt,nrow=qp1,ncol=1)  # makes data a matrix object.
   vx = matrix(0,qp1,qp1)
   for(i in 1:q) {
      # Create the matrix with the correct dimensions
      # instead of relying on R's automatic recycling-
      # to-match-dimensions property.
      vx[((i+1):qp1),((i+1):qp1)] = matrix(1,(qp1-i),
                                      (qp1-i))*tt[(i+1)]
   }
   Sigma.mat = sigmasq*vx
   Itausq = matrix(rep(0,(qp1*qp1)),nrow=qp1,ncol=qp1)
   diag(Itausq) = rep(tausq,qp1)
   V = Sigma.mat + Itausq
   mu.vec = matrix((xo+mu*tt),nrow=qp1,ncol=1)
   return((qp1/2)*log(2*pi) + 0.5*log(det(V)) + 0.5*(t(yt-
            mu.vec)%*%ginv(V)%*%(yt-mu.vec)))
}

#  12. Function to propose initial values of the parameters
#  to feed the parameter estimation function under the EGSS model.
init.egss = function(Yobs,Tvec) {
   q = length(Yobs)-1       #  Number of time series transitions, q.
   qp1 = q+1                #  q+1 gets used a lot, too.
   T.t = Tvec-Tvec[1]       #  For calculations, time starts at zero.
   Ybar = mean(Yobs)
   Tbar = mean(T.t)
   mu.egoe = sum((T.t-Tbar)*(Yobs-Ybar))/sum((T.t-Tbar)*(T.t-Tbar))
   x0.egoe = Ybar-mu.egoe*Tbar
   ssq.egoe = 0
   Yhat.egoe = x0.egoe+mu.egoe*T.t
   tsq.egoe = sum((Yobs-Yhat.egoe)*(Yobs-Yhat.egoe))/(q-1)
   S.t = T.t[2:qp1]-T.t[1:q]   #  Time intervals.
   Ttr = sqrt(S.t)
   Ytr = (Yobs[2:qp1]- Yobs[1:q])/Ttr
   mu.egpn = sum(Ttr*Ytr)/sum(Ttr*Ttr)
   Ytrhat = mu.egpn*Ttr
   ssq.egpn = sum((Ytr-Ytrhat)*(Ytr-Ytrhat))/(q-1)
   tsq.egpn = 0
   x0.egpn = Yobs[1]
   mu0 = (mu.egoe+mu.egpn)/2
   ssq0 = ssq.egpn/2
   tsq0 = tsq.egoe/2
   xo.out = x0.egoe
   return(c(mu0,ssq0,tsq0,xo.out))
}

#  13. Function to optimize the negative log-likelihood and return
#  parameter estimates, likelihood score and AIC, BIC value.
EGSS.ML = function(Yobs,Tvec,parms.guess) {
   tt = Tvec-Tvec[1]
   q = length(tt)-1
   qp1 = q+1
   guess.optim = c(parms.guess[1],log(parms.guess[2:3]),
                     parms.guess[4])
   optim.out = optim(par=guess.optim,fn=negloglike.EGSS.ml,
                  method="Nelder-Mead",yt=Yobs,tt=tt)
```

```
   mles = c(optim.out$par[1],exp(optim.out$par[2:3]),
             optim.out$par[4])
   lnL.hat = - optim.out$value[1]
   AIC = -2*lnL.hat + 2*4  # where 4 is the number of parameters.
   out = list(mles=mles,lnL.hat=lnL.hat,AIC=AIC)
   return(out)
}

# guess.egss = init.egss(Yobs=yt,Tvec=Time.t)
# trial.egssml = EGSS.ML(Yobs=yt,Tvec=Time.t,parms.guess=guess.egss)


#  14. Simulation from the EGSS model for parametric bootstrap:
#  simulate data using the observed time intervals and the ML
#  estimates.  The simulation is greatly eased by the fact that
#  the log observations are multivariate normally distributed.

EGSS.sim = function(nsims,parms,Tvec) {
   tt = Tvec-Tvec[1]
   mu = parms[1]
   sigmasq = parms[2]
   tausq = parms[3]
   xo = parms[4]
   q = length(tt)-1
   qp1 = q+1
   vx = matrix(0,qp1,qp1)
   for(i in 1:q) {
      vx[((i+1):qp1),((i+1):qp1)] = matrix(1,(qp1-i),(qp1-
                                          i))*tt[(i+1)]
   }
   Sigma.mat = sigmasq*vx
   Itausq = matrix(rep(0,(qp1*qp1)),nrow=qp1,ncol=qp1)
   diag(Itausq) = rep(tausq,qp1)
   V = Sigma.mat + Itausq
   mu.vec = matrix((xo+mu*tt),nrow=qp1,ncol=1)
   out = randmvn(n=nsims,mu.vec=mu.vec,cov.mat=V)
   return(out)
}

# PART III):  PBLR function -------------------------------

# 15. Function to do a PBLRT between the EGSS model (null)
# and the OUSS model (alternative).
PBLRT.ddpcont = function(B=10,Yobs,Tvec,alpha=0.05,plot.PBLR=TRUE) {
   #  Estimation under the null Ho: the EGSS model.
   guess.egss = init.egss(Yobs=Yobs,Tvec=Tvec)
   egssml = EGSS.ML(Yobs=Yobs,Tvec=Tvec,parms.guess=guess.egss)
   lnL.Ho = egssml$lnL.hat
   #  Estimation under the alternative H1:  the OUSS model
   guess.rouss  = guess.calc(Yobs=Yobs,Tvec=Tvec)
   roussml = ROUSS.ML(Yobs=Yobs,Tvec=Tvec,parms.guess=guess.rouss)
   lnL.H1 = roussml$lnL.hat
   #  Computing the observed Lam.obs = -2*ln[L(H0)/L(H1)].
      Lam.obs = -2*(lnL.Ho-lnL.H1)
   #  Simulating under H0 using the ML estimates
   mlsforboot = egssml$mles
   sims.mat = EGSS.sim(nsims=B,parms=mlsforboot,Tvec=Tvec)
```

```
   #  Looping over the simulations and maximizing the likelihood
   #  for both models each time and computing the bootstrapped
   #  values of Lam.boot = -2*ln[L(H0)/L(H1)].
   Lam.vec = rep(0,B)
   for(b in 1:B) {
      # Estimating parameters and maximizing under the null.
      Yobs.b = sims.mat[,b]
      egssml.b = EGSS.ML(Yobs=Yobs.b,Tvec=Tvec,
                    parms.guess=mlsforboot)
      lnL.Ho.b = egssml.b$lnL.hat
      # Estimating parameters and maximizing under the alternative.
      guess.rouss.b = guess.calc(Yobs=Yobs.b,Tvec=Tvec)
      roussml.b = ROUSS.ML(Yobs=Yobs.b,Tvec=Tvec,
                      parms.guess=guess.rouss.b)
      lnL.H1.b = roussml.b$lnL.hat
      # Computing the boostrapped likelihood ratio.
      Lam.boot = -2*(lnL.Ho.b-lnL.H1.b)
      Lam.vec[b] = Lam.boot
   }
   #  Computing the proportion of the simulations that have a
   #  more extreme Lamb.boot value than the observed Lamb.obs.
   pval = sum(Lam.vec>Lam.obs)/B
   Decision.rule = "Fail to Reject Density Independence"
   if(pval<alpha) {
      Decision.rule = "Reject Density Independence"
   }
   #  Return the vector of Lam.boot values and
   #  a plot if plot.PBLR=TRUE
   out = list(egssml=egssml,roussml=roussml,Lam.obs=Lam.obs,
            Lam.vec=Lam.vec,pvalue=pval,Decision.rule=Decision.rule)
#   if(plot.PBLR==TRUE) {
#      hist(Lam.vec,main="P-boot distribution of -2*ln[L(H0)/L(H1)]")
#      abline(v=Lam.obs,col="red")
#   }
   return(out)
}
```