Received: 26 July 2017 | Accepted: 18 October 2017

DOI: 10.1111/2041-210X.12931

REVIEW

Methods in Ecology and Evolution BRITISH ECOLOGICAL SOCIETY

Growing the biphasic framework: Techniques and recommendations for fitting emerging growth models

Kyle L. Wilson[1] | Andrew E. Honsey[2,3] | Brian Moe[4] | Paul Venturelli[5]

[1]Department of Biological Sciences, University of Calgary, Calgary, AB, Canada

[2]Ecology, Evolution, and Behavior Graduate Program, University of Minnesota, St. Paul, MN, USA

[3]Department of Fisheries, Wildlife, and Conservation Biology, University of Minnesota, Saint Paul, MN, USA

[4]Coastal and Marine Laboratory, Florida State University, St. Teresa, FL, USA

[5]Department of Biology, Ball State University, Muncie, IN, USA

Correspondence
Kyle L. Wilson
Email: wilsok@ucalgary.ca

Funding information
Vanier Canada Graduate Scholarship program; Killam Trust

Handling Editor: John Reynolds

Abstract

1. Several new growth models have been proposed to account for the life-history trade-offs that occur when indeterminately growing species allocate energy between somatic growth and reproduction. These models can improve the understanding of lifetime growth and life history, but can be more difficult to fit than conventional growth models. Increased data demands, multiple growth phases and increased parameterization all serve as barriers to the adoption and proper use of these new models.
2. We review and comment on confounding issues during model fitting for several of these models, and provide advice on surmounting such issues. We then simulation-test an example model, the Lester biphasic growth model, using several common fitting approaches. We highlight the biases and precision of each approach and provide guiding documents using R and JAGS code.
3. The Bayesian Markov chain Monte Carlo and likelihood profiling approaches generally provided the best fits. Simpler approaches can be unbiased and precise if sampled data are of relatively high quality (e.g. moderate sample sizes for juvenile and adult phases) and model assumptions are met. Bayesian hierarchical approaches can accommodate more complicated data scenarios (e.g. unbalanced design across multiple populations); we provide an example of such an approach by recovering growth trajectories and inferring growth-associated trait variation and environmental effects across multiple populations.
4. Conventional growth models provide limited inference on life history. Many biphasic growth models can provide direct inference on multiple life-history traits, but can be difficult to fit. The recommended approaches herein provide a path forward for fitting biphasic growth models in a variety of scenarios, allowing for wider application and tests of life history and ecological theory.

KEYWORDS

Bayesian, growth estimation, hierarchical models, life-history theory, likelihood profiling, polyphasic growth, somatic growth

1 | INTRODUCTION

Understanding the rate at which individuals or cohorts grow is a major focus of modern biology. Growth results from a diversity of ecological and evolutionary factors and processes. Growth can therefore be used as an indicator of metabolic rate, nutrition, reproductive investment, behaviour, individual and population health, within- and among-species interactions, habitat and more (Dmitriew, 2011; Dobbertin,

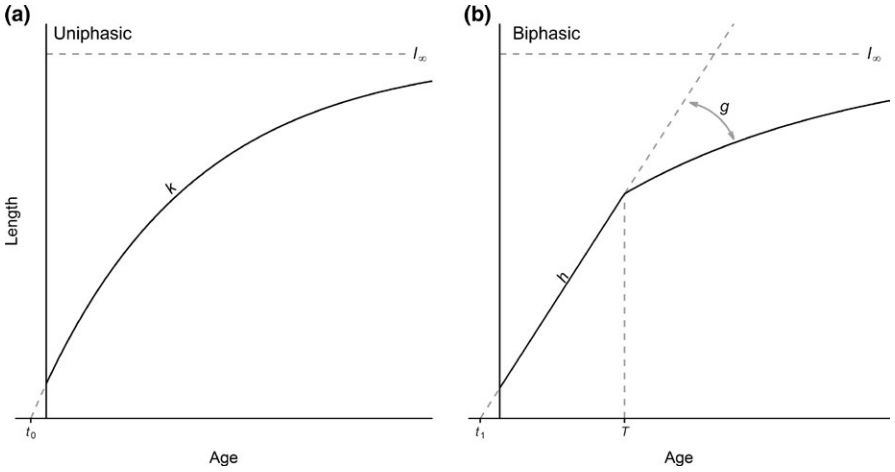


FIGURE 1 Comparison of the (a) uniphasic von Bertalanffy model versus the (b) Lester biphasic model. The Lester model assumes that growth leading up to maturity (*h*) is linear, and that growth shifts when individuals mature (*T*) due to energetic investment in reproduction (*g*). t_0 = von Bertalanffy (adult) hypothetical age at length 0; t_1 = Lester model (immature) hypothetical age at length 0; *k* = von Bertalanffy (Brody) growth coefficient; l_∞ = asymptotic length

2005; Honsey, Staples, & Venturelli, 2017; Rennie & Venturelli, 2015). Growth and size are also under strong selection (Dmitriew, 2011; Okie et al., 2013), correlated with a suite of life-history traits that are important to fitness and population growth (Roff, 1983; Stearns, 1992), and often central to management and conservation (Lorenzen, 2016).

Most vertebrate and plant studies (Paine et al., 2012) describe lifetime growth as a single curve, but this approach has been criticized. Popular uniphasic models include the logistic, Gompertz and von Bertalanffy models (Pardo, Cooper, & Dulvy, 2013). These curves tend to be straightforward to fit and are usually applied to cross-sectional or longitudinal size-at-age data from single populations. A common criticism of these models, particularly in the fish literature, is that lifetime growth comprises two or more stages that result from reproductive investment (Day & Taylor, 1997; Lester, Shuter, & Abrams, 2004) or changes in habitat, food or other stressors (Paloheimo & Dickie, 1965; Parker & Larkin, 1959; Figure 1). Describing these stages with a single curve can obscure important ecological and evolutionary information.

At least 26 biphasic growth models have been proposed for, or are commonly applied to, fishes (Table 1). These models vary in complexity (Minte-Vera, Maunder, Casselman, & Campana, 2016) and tend to describe either an abrupt or smooth transition between growth phases. Most transitions are associated with reproductive investment, although changes in age, diet and habitat are also cited. The von Bertalanffy model is commonly used for at least one phase. Such an approach is consistent with von Bertalanffy's (1957) contention that different parameter values were needed for juveniles and adults (Boukal, Dieckmann, Enberg, Heino, & Jorgensen, 2014), but inconsistent with the modern convention of fitting this model to entire growth histories (Ogle, 2016).

There are advantages to biphasic growth models, but fitting them can be a challenge. Biphasic models allow for the estimation of more life-history traits from growth data compared to uniphasic models (e.g. age-at-maturity; Honsey et al., 2017), can be more statistically or biologically valid than uniphasic models (Lester et al., 2004; Moe, 2015; Quince, Abrams, Shuter, & Lester, 2008b; Roff, Heibo, & Vøllestad, 2006), can relate to behavioural pace-of-life syndromes (Nakayama, Rapp, & Arlinghaus, 2017), can estimate management reference points (Lester, Shuter, Venturelli, & Nadeau, 2014) and can provide evolutionary insight (Roff, 1983; Roff et al., 2006). Biphasic models that explicitly account for energetic costs of reproduction (only implicit with uniphasic models) can also test plastic or evolutionary hypotheses when ecosystem changes alter reproductive schedules (Lorenzen, 2016). However, biphasic models can be difficult to fit due to correlations among parameters, increased model complexity and lack of software. For example, parameter correlations from biphasic models are often similar to the uniphasic von Bertalanffy model ($\rho \approx |0.7|$), except with more parameters (Helser & Lai, 2004; Minte-Vera et al., 2016; Mollet, Ernande, Brunel, & Rijnsdorp, 2010). These correlations can lead to errors during estimation (Minte-Vera et al., 2016; Mollet et al., 2010). Although the life-history trade-offs and correlations of many biphasic models can improve ecological inference (e.g. life-history invariants; Shuter et al., 2005), these challenges preclude their straightforward application.

Herein, we present techniques, discuss recommendations and provide companion code in R (R Core Team, 2016) and JAGS (Plummer, 2003) for fitting biphasic growth models across a range of scenarios. We demonstrate these concepts using the Lester biphasic model (LM; Lester et al., 2004), which is popular in the fish literature and has been adapted for a diversity of ecological and evolutionary applications (Enberg, Jørgensen, Dunlop, Heino, & Dieckmann, 2009; Giacomini & Shuter, 2013; Honsey et al., 2017; Johnston, Arlinghaus, & Dieckmann, 2010; Lester et al., 2014; Rennie, Collins, Shuter, Rajotte, & Couture, 2005). Finally, we provide guidance regarding which approaches to use via simulation tests that evaluate bias and precision for a range of statistical approaches.

2 | THE LESTER BIPHASIC GROWTH MODEL AS AN EXAMPLE

The LM uses a theoretically and empirically derived energy allocation framework to describe the lifetime growth of fishes (Lester et al., 2004). The LM is founded on the common principle (Table 1) that maturation in fishes represents a change in energy allocation such that lifetime growth is in two phases: one before maturity and one after.

TABLE 1 A chronological summary of biphasic growth models that have been proposed for, or are commonly applied to, fishes

Reference	Type	Phase 1	Phase 2	Transition	Biological motivation, justification, or explanation
Brody (1945)	Discrete	Exponential	VB	At maturity	Investment in reproduction
Roff (1983)	Continuous	Linear	Curvilinear	Increase in the proportion mature with age	Investment in reproduction
Condrey, Beckman, and Wilson (1988)[a]	Discrete	VB	VB	k and t_0 change at some age	Growth changes with age
Bayliff, Ishizuka, and Deriso (1991)	Discrete	VB or Gompertz	Linear	At some length	Growth changes with age
Hoese, Beckman, Blanchet, Drullinger, and Nieland (1991)	Continuous	VB	VB	l_∞ increases with age	Growth changes with age
Soriano, Moreau, Hoenig, and Pauly (1992) models 1 and 2	Continuous	VB	VB	l_∞ or k increase with age	Diet shift
Soriano et al. (1992) model 3	Discrete	VB	VB	l_∞ and k change at some age	Diet shift
Rijnsdorp and Storbeck (1995); Baulier and Heino (2008)	Discrete	Linear	Linear	Slope and intercept change at maturity	Investment in reproduction
Craig, Choat, Axe, and Saucerman (1997)[a]	Discrete	VB	VB	l_∞, k and t_0 change at maturity	Investment in reproduction
Day and Taylor (1997)	Discrete	Power	VB	At maturity	Investment in reproduction
Charnov, Turner, and Winemiller (2001)	Discrete	Curvilinear[b]	Curvilinear[b]	Investment in reproduction terms added at maturity	Investment in reproduction
Laslett, Eveson, and Polacheck (2002)	Continuous	VB	VB	k increases with age	Habitat shift
Porch, Wilson, and Nieland (2002)	Continuous	VB	VB	k decreases with age	Ontogeny or habitat
Hearn and Polacheck (2003)	Discrete	VB	VB	l_∞ and k change at some age	Growth changes with age
Lester et al. (2004)	Discrete	Linear	VB	At maturity	Investment in reproduction
Tracey and Lyle (2005)	Discrete	VB	VB	l_∞, k and t_0 change at some age	Habitat shift
Quince et al. (2008a,b); Boukal et al. (2014)	Continuous	Curvilinear	Curvilinear	Investment in reproduction increases with age	Investment in reproduction
Alós et al. (2010)	Discrete	VB	VB	k changes at some age	Growth changes with age (perhaps due to investment in reproduction)
Mollet et al. (2010); Brunel et al. (2013)	Continuous	Curvilinear[b]	Curvilinear[b]	Investment in reproduction term increases with age	Investment in reproduction
Ohnishi, Yamakawa, Okamura, and Akamine (2012)	Discrete	VB	VB	Investment in reproduction term added at maturity	Investment in reproduction
Ohnishi et al. (2012)	Continuous	VB	VB	Investment in reproduction term increases with age	Investment in reproduction
Scott and Heikkonen (2012)	Continuous	Linear	Linear	Slope and intercept change with age	Investment in reproduction
Trip, Clements, Raubenheimer, and Choat (2014)	Discrete	VB	VB	Parameters change at some age	None given
Minte-Vera et al. (2016)	Continuous	VB	VB	Asymptotic size increases with age	Investment in reproduction
Minte-Vera et al. (2016)	Continuous	VB	VB	Investment in reproduction term added at maturity	Investment in reproduction

Discrete models describe two distinct growth phases, and continuous models describe a smooth transition between growth phases. VB refers to some version of the von Bertalanffy growth model, and l_∞, k and t_0 are the typical VB parameters.

[a]An identical model was developed by Fiorentino, Gancitano, Gancitano, Rizzo, and Ragonese (2013).

[b]Based on the metabolic theory of ecology (see West, Brown, & Enquist, 2001).

The key assumptions of the LM are that growth is indeterminate, body mass increases with length cubed, gonad mass is proportional to somatic mass, and metabolism scales with mass to the two-thirds power (Lester et al., 2004). Although this last assumption is debatable (Glazier, 2010; West, Brown, & Enquist, 1997, 1999) and can be relaxed (Boukal et al., 2014; Quince, Abrams, Shuter, & Lester, 2008a), it simplifies model derivation and fitting. The LM performs well for species that mature relatively late and have long reproductive life spans (Lester et al., 2004), and is effective at describing the lifetime growth of numerous fishes (Shuter et al., 2005) and perhaps other ectotherms (Honsey et al., 2017).

The LM describes immature growth in the lead-up to maturity as a straight line and mature growth as a von Bertalanffy curve:

$$l_t = h(t - t_1) \text{ when } t \leq T, \tag{1}$$

$$l_t = l_\infty \left(1 - e^{-k(t-t_0)}\right) \text{ when } t > T, \tag{2}$$

where l_t is length at time t, h is juvenile growth rate (length per unit time), t_1 is the LM (immature) hypothetical age at length 0, T is last immature age (LM parameter for age-at-maturity), l_∞ is asymptotic length, k is the von Bertalanffy (Brody) growth coefficient, and t_0 is the von Bertalanffy (adult) hypothetical age at length 0. Lester et al. (2004) further showed that l_∞, k and t_0 reflect known trade-offs between growth, reproduction and mortality:

$$l_\infty = \frac{3h}{g}, \tag{3}$$

$$k = \ln\left(1 + \frac{g}{3}\right), \tag{4}$$

$$t_0 = T + \frac{\ln\left(1 - \frac{g(T-t_1)}{3}\right)}{\ln\left(1 + \frac{g}{3}\right)}, \tag{5}$$

where g is the cost to somatic growth of maturity (often assumed to be dominated by investment in reproduction; Kozlowski, 1996; Roff, 1983), and that

$$g \approx 1.18(1 - e^{-M}), \tag{6}$$

$$T \approx \frac{1.95}{e^M - 1} + t_1, \tag{7}$$

where M is the instantaneous rate of late-stage juvenile and adult natural mortality. The parameter g captures the proportion of surplus energy allocated into direct (e.g. gonadal development) and indirect (e.g. migration, nesting, displaying, metabolic costs of storing gonads) reproductive investment—for further discussion, see Boukal et al. (2014, section 2.3).

We use the LM to demonstrate biphasic fitting techniques because it is grounded in theory, empirically supported, popular, and exhibits many of the challenges that are encountered when fitting a biphasic model. Unlike uniphasic models, biphasic models are usually selected and applied based on a priori information about how and when growth transitions between phases (Table 1). If this information is available, then it must then be incorporated properly into the model. For example, the LM uses the last immature age (T) to differentiate between growth phases, but empirical data usually describe age-at-first-reproduction (i.e. some age > T). Model parameter estimates can also depend on whether a transition point, such as maturity, is estimated in advance or as part of the model-fitting procedure (Minte-Vera et al., 2016). Such dependencies often stem from confounded and correlated parameters, especially when estimates are uncertain and sensitive to starting values. Other examples include the Mollet et al. (2010) model and modified versions of the LM (Boukal et al., 2014; Quince et al., 2008a,b), all of which can generate variable or biased parameter estimates, or simply fail to converge. Although the correlated parameters in the LM and related biphasic models are statistically inconvenient, they result from important life-history trade-offs and general ecological rules. Many biphasic models incorporate these trade-offs and correlations into the model structure based on empirical relationships between natural mortality and life-history parameters (e.g. Equations 6 and 7 above). Hence, biphasic models can be more realistic than uniphasic models that assume that traits are independent and fixed, and they represent an opportunity to improve ecological and management inference through insight into life-history traits, vital rates and empirical proxies (e.g. the gonadosomatic index as a proxy for reproductive investment).

3 | TECHNIQUES

The first factor to consider when fitting a biphasic model is whether the data meet model assumptions. For example, immature growth can be nonlinear (particularly in early life) due to ontogenetic diet shifts, changes in per capita food availability and other factors (Lester et al., 2004), which would violate the assumptions of the LM. One may therefore need to modify the model (Quince et al., 2008a) or truncate the data accordingly (McDermid, Shuter, & Lester, 2010). One must also consider the structure of the data: Are they cross-sectional (snapshot of a population at one time point) or longitudinal (repeated measures through time)? If the latter, do the data track individuals or cohorts? Do they describe single or multiple populations? The framework used to fit a biphasic model also depends on whether data are available to describe the shift in growth (e.g. maturity data for the LM) and on whether one wishes to include environmental predictors or other variables that might help to explain growth.

In this section, we address many of the considerations mentioned above by introducing and evaluating some techniques for fitting biphasic models across a variety of scenarios. We focus on maturity-based biphasic models and use the LM as an example. We first describe potential discrepancies in the meaning of "maturity" between models and data. We then highlight different approaches for fitting biphasic models. We organize this section according to model complexity, beginning with the simplest fits (fitting the LM with maturity data) and moving to more complex approaches (hierarchical modelling across populations without maturity data). We provide companion code in appendices throughout.

3.1 | Fitting biphasic models with maturity data

When incorporating maturity data in a biphasic fit, one must account for potential differences in the meaning of "maturity" between the model and the data. For instance, the LM assumes that growth slows in response to energetic investment in reproduction rather than a reproductive event (e.g. spawning or breeding). The onset of investment in reproduction often occurs before maturity is assessed. In longer-lived bony fishes, for example, investment in reproduction can occur a year or more before first spawning, and maturity in chondrichthyans usually corresponds to the mean age at which individuals are physically capable of reproducing (Cotton, Dean Grubbs, Dyb, Fossen, & Musick, 2015). Lags between the onset of investment in reproduction and maturity data or metrics (e.g. age-at-50%-maturity or A_{50}) should be corrected for before fitting maturity-based biphasic models. For example, if initial investment in reproduction occurs in the year prior to when age-at-maturity is assessed, and if individual ages are in integer years, then $T = A_{50} - 1$.

Energetic constraints and life-history trade-offs often truncate parameter distributions that are dependent on another estimated parameter. This stochastic truncation can be difficult to account for with typical model-fitting approaches. For example, assuming that g in the LM is constant with age, an individual cannot allocate more surplus energy into reproduction than was available for somatic growth in the juvenile phase. Analytically, g has an intrinsic maximum and is bounded between 0 and $3/(T - t_1)$. Such constraints can lead to errors in optimization routines, particularly if parameter distributions are not appropriately bounded during estimation (e.g. penalized likelihoods or truncated distributions), and may cause other errors (e.g. nonpositive definite Hessian matrix). In a maximum likelihood framework, one can estimate on $\log(g)$ or $\text{logit}(g)$ to constrain estimation and set a penalty to the likelihood if g exceeds $3/(T - t_1)$.

The most intuitive approach to fitting the LM with maturity data is to fit the two growth phases in sequence (Appendix S1). For example, one can subset the data into immature and mature individuals, fit a linear model to the immature subset (Equation 1) and then treat the parameter estimates from the immature fit as fixed when fitting the mature curve (Equation 2). Fitting the model in this manner requires an a priori estimate of age-at-maturity (e.g. A_{50}) to substitute for T. This simplistic approach may be necessary if data are limited for one or both phase(s) of growth. However, this approach restricts error propagation, which can complicate parameter estimation and lead to erroneous conclusions.

Estimates of age-at-maturity can also be incorporated into full model fits as a known breakpoint (e.g. substituting A_{50} for T). Parameter estimation can be carried out in either a maximum likelihood (Appendix S2) or Bayesian framework. This approach allows for better error propagation than the previous method but is still limited given that age-at-maturity is fixed. In addition, this method can be sensitive to starting values and may require some form of restriction on parameter estimates (e.g. penalized likelihoods) to maintain realistic parameter combinations.

The most inclusive method for fitting a biphasic model with maturity data is to estimate maturity using a maturity ogive that complements the growth model. This maturity model can be optimized on a joint likelihood (or posterior) alongside the biphasic model (see Matthias, Ahrens, Allen, Lombardi-Carlson, & Fitzhugh, 2016; Quince et al., 2008b; Ward, Post, Lester, Askey, & Godin, 2017). This method allows for improved estimates of variance around parameters because the errors and correlations among parameters are allowed to trade-off naturally among model subcomponents. The downside of this approach is that it typically requires customized coding, more computational time and assumptions regarding the shape and symmetry of the maturity ogive. Because estimation using this approach relies on the quality of maturity data, errors in the maturity data (e.g. mature but nonreproducing adults may be difficult to distinguish from juveniles) can lead to biased parameter estimates.

3.2 | Fitting biphasic models without maturity data

If maturity data are unavailable, then age-at-maturity can be estimated directly from size-at-age data using maturity-based biphasic models (Table 1). The shift in growth at maturity should lead to detectable changes in model residuals, thus allowing for estimation of age-at-maturity. For example, the sizes-at-age of mature fish are less likely to adhere to predictions from the immature growth phase than the adult phase and vice versa. However, estimating age-at-maturity can be problematic for many biphasic models because it is highly correlated with other parameters and can be confounding (Brunel, Ernande, Mollet, & Rijnsdorp, 2013; Minte-Vera et al., 2016; Mollet et al., 2010).

We simulation-tested the relative performance of three approaches for fitting the LM without maturity data: penalized maximum likelihood (Appendix S3), likelihood profiling (Honsey et al., 2017; Appendix S4) and Bayesian Markov chain Monte Carlo (MCMC; Appendix S5). The simulation tested all combinations of three levels each for adult natural mortality rate (M = {0.1, 0.2, 0.5}, per year) and the coefficient of variation in length-at-age (cv_l = {0.1, 0.15, 0.25}). Other parameters were held constant across simulation scenarios (t_1 = −0.2 year, h = 50 mm/year, slope of age-dependent selectivity = −0.4). The g and T parameters were calculated as functions of M (using Equations 6 and 7, respectively), and the age-at-50%-selectivity was equal to T. We then used a multinomial observation process to generate realistic samples (n = 50) of population age and size structures. The resulting length-at-age data are similar to what might be observed in wild populations. We evaluated the performance of the three methods to estimate the true growth parameters using a bootstrap approach (N_{boot} = 100) and calculated the root-mean-square error (RMSE) per bootstrapped iteration:

$$\text{RMSE} = \sqrt{\frac{1}{n}\Sigma_{i=1}^{n}(\hat{l}_i - l_i)^2}, \quad (8)$$

where n is the total number of fish, l_i is the observed length of fish i, and $\hat{l}_i$ is the maximum likelihood (or posterior mean) predicted length

for fish *i*. Lower RMSE values indicate low bias and high precision. We evaluated the ability of each method to recover true life-history parameters θ using per cent bias:

$$\text{Bias}_{\theta} = \frac{\theta - \hat{\theta}}{\theta} 100\%, \quad (9)$$

where $\hat{\theta}$ is the maximum likelihood (or posterior mean) estimate of the parameter. Although this approach does not capture bias for each parameter's 95% quantile, it does indicate the general performance of each method to estimate specific parameters. Lastly, we tested the sensitivity of each method to a range of starting values. Figure 2 shows results for a single iteration of the simulation, including the true life-history parameters, inducing observation error and evaluating model performance (Appendix S6).

The likelihood profiling and Bayesian MCMC approaches were relatively unbiased and precise (Figure 3; Appendix S7). In contrast, the penalized likelihood approach was sometimes biased and sensitive to starting values. Most of the uncertainty and bias in parameter estimates is in t_1, an intercept parameter with limited scope for ecological and evolutionary inference (but see Lester et al., 2004; Quince et al., 2008a). The relatively high bias in t_1 estimates likely stems from the nonlinear correlations among t_1 and the other parameters (e.g. relatively low bias in *h* can lead to high bias in t_1). All three methods performed worse when mortality rates were low (e.g. *M* = 0.10 per year) than when they were high, with the penalized likelihood approach being the least reliable under low natural mortality rates. This result likely stems from the fact that low adult natural mortality is correlated with low reproductive investment (Equation 6), leading to a subtle (and difficult to detect) change-point between juvenile and adult phases. Bias and precision in estimating specific life-history parameters were relatively equal for the likelihood profiling and Bayesian MCMC approaches (Appendix S7). Estimates of *T* and *h* were the most inconsistent across scenarios, particularly for the penalized likelihood approach. Per cent bias in *T* increased with increasing variance in length-at-age (and, in some cases, increasing mortality), while bias in *h* decreased with increasing mortality.

3.3 | Fitting biphasic models to data describing multiple populations

One may wish to estimate the growth of multiple populations simultaneously for several reasons, including to (1) propagate uncertainty and error trade-offs during estimation, (2) estimate environmental effects on growth trajectories (Helser & Lai, 2004; Shuter, Jones, Korver, & Lester, 1998), (3) gain insight into growth-associated traits and their trade-offs across environments and (4) estimate growth for data-limited populations alongside data-rich populations. Simultaneously estimating biphasic growth model parameters for multiple populations can be challenging. For instance, one must decide whether to assume that growth is related among populations (in which case the parameters should be hierarchical arising as random variables from a common distribution, i.e. a mixed-effects model), or that growth parameters among populations are independent (i.e. fixed effects model). When using a hierarchical approach, one must be aware of the "shrinkage effect" (whereby the population-level parameter estimates are pulled towards the group mean), which can be especially large if samples are unbalanced across populations (Helser & Lai, 2004). Fixed effects approaches should be relatively straightforward adaptations of the hierarchical approaches and code that we discuss below.

Recent methods to estimate hierarchical models have become streamlined (Bolker et al., 2009, 2013), but many biphasic growth models have properties that make standard mixed-effects fitting approaches difficult to adopt (e.g. nonindependent phases, nonlinearity, correlated parameters, stochastic parameter bounds). Hence, the analysis must be coded manually. A relatively robust way to describe growth with a biphasic model is to numerically approximate growth parameter distributions using MCMC algorithms and Bayesian inference. Appendix S8 provides code for fitting the LM using a hierarchical Bayesian approach in the JAGS language (Plummer, 2003) via the RUNJAGS package in R (Denwood, 2016).

The simulation in Appendix S8 has input parameters for the number of populations, the maximum number of age classes and the mean and variance of growth parameters across populations. The latter two features are used to define a global (i.e. cross-population) distribution for a given life-history trait, from which population-specific values arise as random variables. We provide an example in which we simulate growth across 10 populations with an unbalanced design (sample sizes vary between 40 and 200 samples per population), assuming that each of the LM parameters is related among populations (Figure 4). Where appropriate, the parameter values were identical to the single population simulations above, and vague priors were used to fit the model. Our results indicate that the Bayesian hierarchical approach can recover the simulated parameter values despite strong correlations in the posterior parameter space (Figure 4f).

Adapting the LM for different scenarios requires relatively simple tweaks to the approaches described above. For example, a hierarchical framework (Appendix S8) can be used to fit biphasic models to longitudinal data (e.g. repeated measures of growth across individuals, with individual-level parameters treated as random deviances from a population-level mean). In addition, one can estimate the effect of environmental determinants on growth parameters using a Bayesian hierarchical analysis combined with a parameter inclusion probability (modelled as a latent variable; Royle & Dorazio, 2008). In Appendix S9, we demonstrate a simulation in which we use a hierarchical regression and inclusion probability approach to recover both population-specific growth rates and the slope in average growth rate across populations along an environmental gradient. The simulation mirrors those in Appendices S6–S8 to generate realistic, noisy data with an unbalanced sampling design. Using this approach, we recovered both true growth rates and the slope in average growth rate across populations (Figure 5). Modelling frameworks such as these can allow for broader ecological inference regarding trait–environment relationships.

4 | RECOMMENDATIONS

In this section, we make recommendations and highlight potential solutions for barriers that might arise when fitting biphasic models.

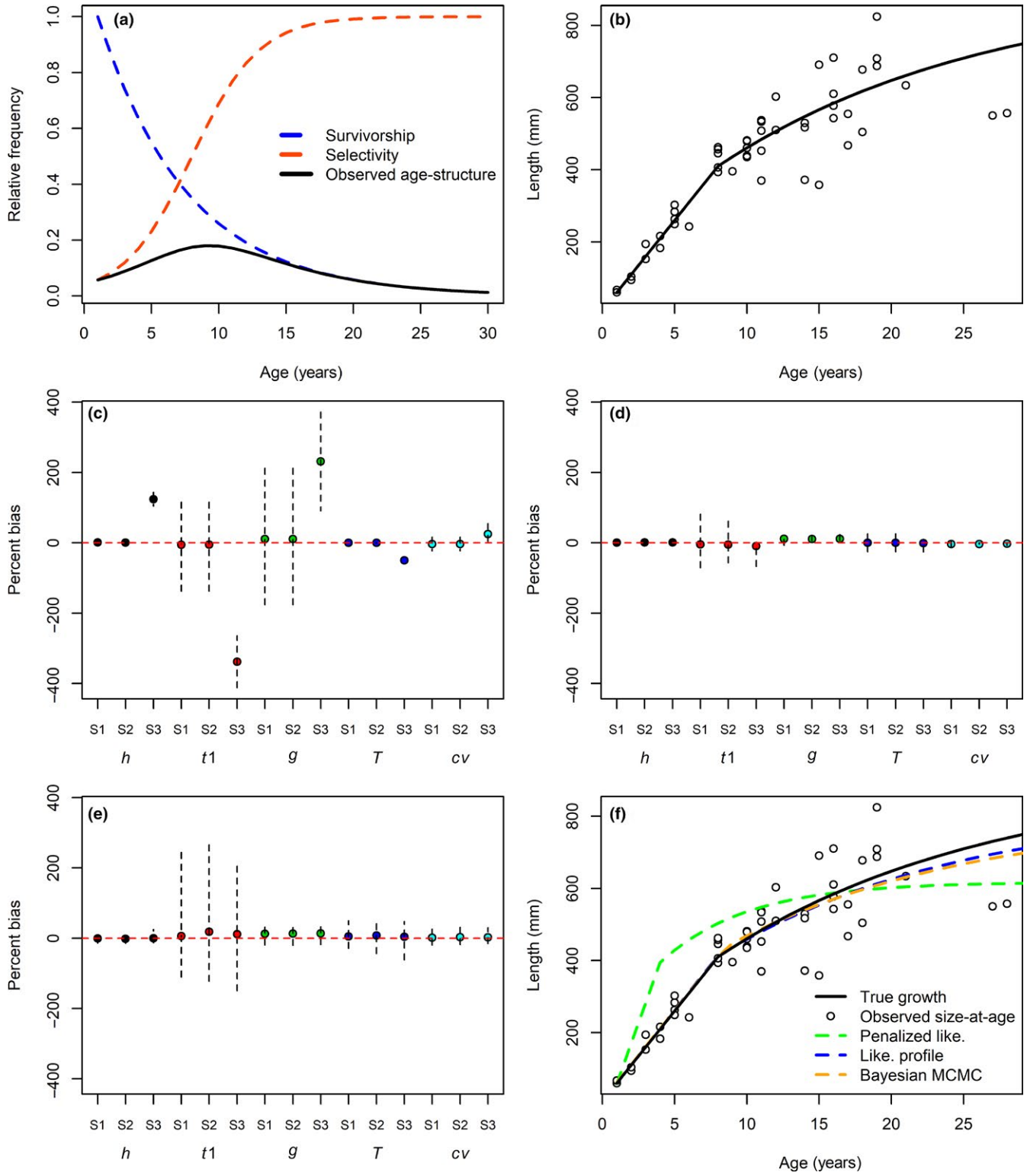


FIGURE 2 Results from a single iteration of the simulation test of three different statistical approaches for fitting the Lester biphasic growth model without maturity data (see Appendix S6). Data were simulated according to the Lester model with stochastic noise. (a) Sample sizes-at-age were based on survivorship (M = 0.15 per year) and gear selectivity to approximate realistic size and age structures. (b) Simulated data (n = 50) and "true" growth trajectory. (c–e) Mean per cent bias (points) and 95% confidence/credible intervals (dashed lines) for four Lester biphasic model parameters and the coefficient of variation in length-at-age (cv) fit using (c) penalized maximum likelihood, (d) likelihood profiling and (e) Bayesian MCMC. Three different vectors of starting values (S1 = below true h, S2 = close to true h, S3 = above true h) were used to evaluate the sensitivity of each approach to starting values. (f) Mean estimated growth trajectory from each approach fitted under S3 starting values compared to the true growth trajectory

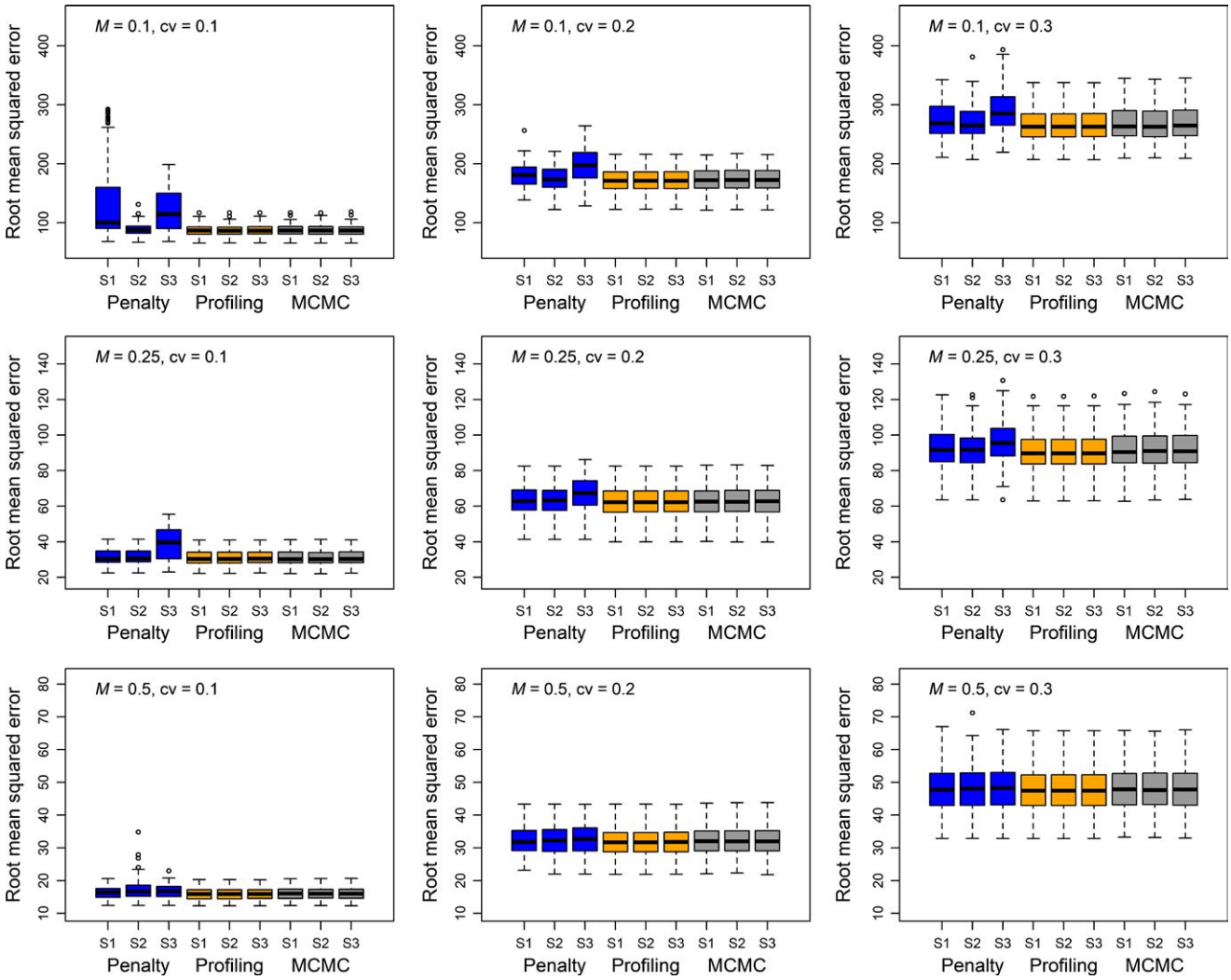


FIGURE 3 Relative performance of three approaches for fitting the Lester biphasic model without maturity data (blue = penalized maximum likelihood, orange = likelihood profiling, grey = Bayesian MCMC) evaluated by root-mean-squared error for 100 bootstrapped datasets across nine scenarios (natural mortality M = {0.1, 0.2, 0.5} per year and coefficient of variation in length-at-age cv_l = {0.1,0.15,0.25}). Other parameters were held constant across scenarios. Box and whiskers show median (black bars) and 95% quantiles. Each approach was initialized using three different vectors of starting values (S1 = below true h, S2 = close to true h, S3 = above true h) to evaluate the sensitivity of each approach to starting values (see Appendix S7 for details and additional results)

Our most general recommendation is to start with simpler approaches (relative to the coding expertise of the practitioner and quality of the data) before adding complexity. This includes starting by penalizing parameter estimates on reasonable bounds (and perhaps iteratively relaxing or tightening these bounds) because the sensitivity and correlations between parameters can lead to errors during estimation (Minte-Vera et al., 2016; Mollet et al., 2010). If parameter correlation continues to be problematic, or accounting for that correlation is of interest, then we recommend drawing parameter estimates from a multivariate normal distribution and estimating a covariance matrix between parameters (Helser & Lai, 2004). This approach can speed up estimation, and most MCMC algorithms can incorporate multivariate distributions. We also recommend varying starting values and re-fitting the model to assess whether parameters converge on a common estimate.

When maturity data are available, we recommend jointly estimating a maturity ogive and biphasic growth model. This approach is generally more robust and statistically appropriate than estimating age-at-maturity a priori, and it allows for better error propagation when estimating parameters. Simulation tests of this technique can be found in Quince et al. (2008a), and example applications can be found in Quince et al. (2008b) and Matthias et al. (2016).

We encountered differences in ease of use, accuracy, precision, speed and robustness among the three approaches that we tested for fitting the LM without maturity data. We recommend choosing a method based almost entirely on accuracy, performance and robustness, even though some methods trade-off one attribute against another (e.g. robustness vs. speed). The penalized likelihood approach is a quick and relatively easy way to fit biphasic models without maturity data. However, we recommend against using this approach for formal

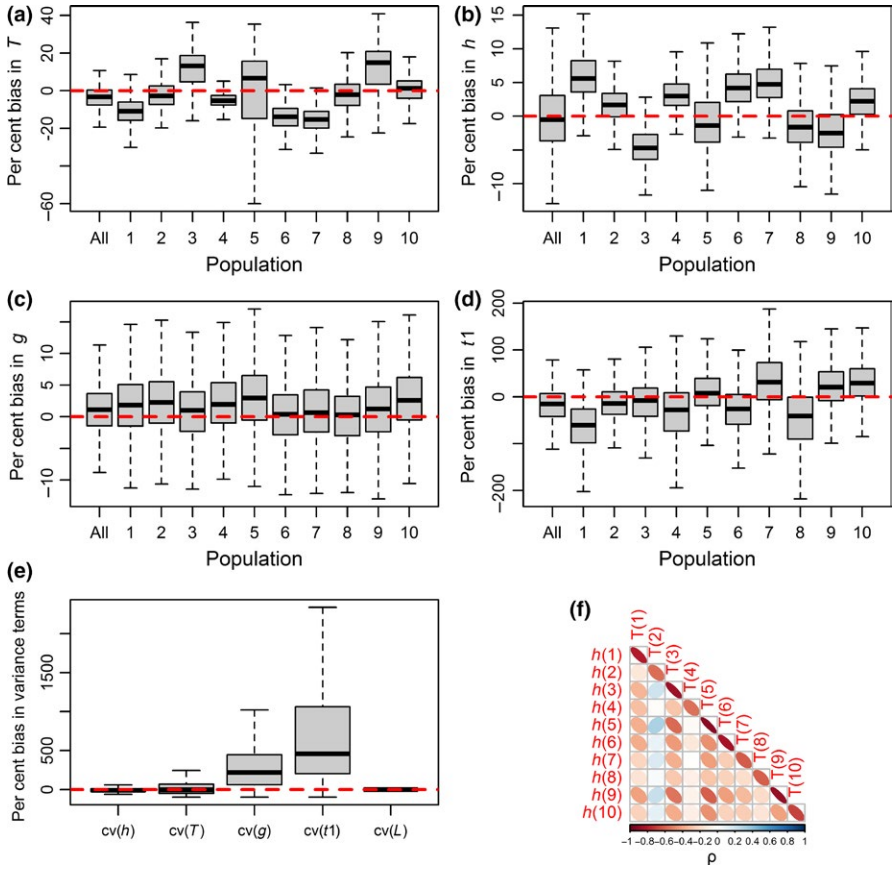


FIGURE 4 Multipopulation Lester model estimation using the Bayesian hierarchical MCMC approach in JAGS. Black bars represent posterior medians, and dashed lines represent 95% credible intervals. The red line indicates the true simulation parameters (zero bias). Ten populations were simulated with varying Lester model parameters and unbalanced sample sizes (n = 40–200 individuals per population). (a–d) Per cent bias in Lester model parameter estimates. (e) Per cent bias in variance terms for Lester model parameters and size (L) across all populations. (f) Correlations between two parameters, T (age-at-maturity) and h (juvenile growth rate), for each of the 10 populations

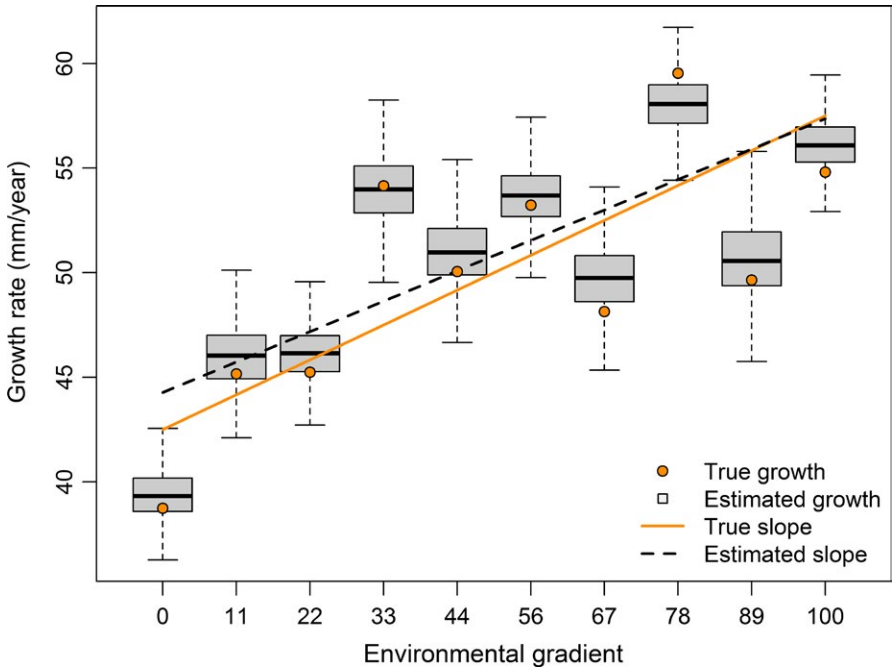


FIGURE 5 Multipopulation growth simulation in which average growth rate increases along some environmental gradient. Lester models were fit using a Bayesian hierarchical method with a parameter inclusion probability approach (Appendix S9). Box and whiskers show the median and 95% credible intervals for the posterior distribution for growth rate for each population; points denote the simulated true growth rates. The estimated environmental trend on average growth rate (posterior mean slope, dashed black line) reflects the true trend (solid orange line)

assessments and ecological inference unless data quality is high (e.g. large sample sizes). We instead recommend either the likelihood profiling or Bayesian MCMC approaches because they more consistently provide accurate parameter estimates. Likelihood profiling was slower and required more manual coding than penalized maximum likelihood, but was faster and arguably easier to code than the Bayesian MCMC method. The MCMC approach readily accommodates more complex applications (e.g. hierarchical models) but is slower and can be more difficult to implement and validate given the added challenge of choosing prior distributions (e.g. informative vs. diffuse, normal vs. uniform, etc.). Aside from philosophical differences between Bayesian and frequentist inference, any further differences are primarily in

implementation and coding. The ease of use for any particular method will depend on one's experience (e.g. familiarity coding in MCMC software). We encourage the use of the R code provided in the appendices as a starting point for implementing all of the approaches discussed herein.

We recommend hierarchical approaches for more complex applications of biphasic growth models, including fits to repeated-measures (e.g. back-calculated) data and estimating environmental trends across multiple populations. Nonlinear mixed-effect models can be coded in both Bayesian and frequentist frameworks in a variety of statistical languages. Bayesian approaches are useful because (1) they can incorporate prior information, (2) MCMC algorithms are typically stable and allow for errors in model components to trade-off naturally, and (3) latent variable and mixture-model approaches can be incorporated in most Bayesian MCMC software (Royle & Dorazio, 2008), which increases the flexibility (and potential complexity) of biphasic model applications and allows one to address interesting eco-evolutionary questions (see Matthias et al., 2016). When using Bayesian MCMC approaches, we strongly recommend assessing whether the model has converged on a stable posterior mode (Gelman et al., 2013) using tools such as posterior convergent diagnostics, residual diagnostics and goodness-of-fit tests (e.g. posterior predictive checks, Bayesian *P*-values).

5 | CONCLUSIONS

The convention in fish science and other disciplines is to fit a single growth curve to size-at-age data across years and cohorts. However, advances in life-history theory, statistical methods and computational power have paved the way for fitting more complex growth models (Lorenzen, 2016), thus improving both our basic understanding of growth across taxa and the efficacy of management. These advances are evident in the many flexible and cutting-edge adaptations of growth models (Maunder, Crone, Punt, Valero, & Semmens, 2016).

We presented techniques, recommendations and code for fitting biphasic growth models across a range of scenarios. These models can accurately describe growth while providing important and direct inference on life-history traits. These advantages result from common life-history principles and trade-offs and should therefore extend to many species that exhibit indeterminate growth (Honsey et al., 2017). We chose the LM as our example model because it is rooted in life-history theory and popular in the literature, particularly for trait-based and eco-evolutionary inference (Alós, Palmer, Balle, Grau, & Morales-Nin, 2010; Nakayama et al., 2017; Uusi-Heikkilä et al. 2016; Ward et al., 2017). However, our recommendations apply to many other biphasic models (e.g. diet and habitat shifts can be modelled much like maturity; Table 1). These models have the potential to improve our understanding of growth and life history across taxa and can be adapted to a variety of interesting applications. Importantly, evaluations of these models (Minte-Vera et al., 2016 and this manuscript) show that certain model-fitting approaches can lead to biased parameter estimates, suggesting that different models and approaches can have consequences for parameter inference. We encourage both the increased application of biphasic models and robust assessments of their ability to describe growth and life history across taxa, conditions and scenarios.

ACKNOWLEDGEMENTS

The authors thank Josep Alós, Shin Nakayama, Dave Staples and Brian Matthias for commenting on their experiences with fitting biphasic growth models. Funding for this work was provided by the Vanier CGS and Killam predoctoral scholarships (K.L.W.), the University of Minnesota (A.E.H.) and Florida State University (B.M.).

AUTHORS' CONTRIBUTIONS

K.L.W. and P.V. conceived of the idea for the manuscript; K.L.W. organized and led the project; K.L.W., A.E.H., B.M. and P.V. wrote the manuscript; K.L.W., A. E. H. and B.M. provided code and Markdown documents; K.L.W., A.E.H. and P.V. designed figures and tables.

DATA ACCESSIBILITY

All simulated data used in this manuscript are available via our R code supplemental files (https://doi.org/10.5281/zenodo.1044474); no empirical data were used in this manuscript.

REFERENCES

Alós, J., Palmer, M., Balle, S., Grau, A. M., & Morales-Nin, B. (2010). Individual growth pattern and variability in *Serranus scriba*: A Bayesian analysis. *ICES Journal of Marine Science*, *67*, 502–512. https://doi.org/10.1093/icesjms/fsp265

Baulier, L., & Heino, M. (2008). Norwegian spring-spawning herring as the test case of piecewise linear regression method for detecting maturation from growth patterns. *Journal of Fish Biology*, *73*, 2452–2467. https://doi.org/10.1111/jfb.2008.73.issue-10

Bayliff, W. H., Ishizuka, Y., & Deriso, R. B. (1991). Growth, movement, and attrition of northern bluefin tuna, *Thunnus thynnus*, in the Pacific Ocean, as determined by tagging. *Inter-American Tropical Tuna Commission Bulletin*, *20*, 94.

von Bertalanffy, L. (1957). Quantitative laws in metabolism and growth. *The Quarterly Review of Biology*, *32*, 217–231. https://doi.org/10.1086/401873

Bolker, B. M., Brooks, M. E., Clark, C. J., Geange, S. W., Poulsen, J. R., Stevens, M. H. H., & White, J. S. S. (2009). Generalized linear mixed models: A practical guide for ecology and evolution. *Trends in Ecology and Evolution*, *24*, 127–135. https://doi.org/10.1016/j.tree.2008.10.008

Bolker, B. M., Gardner, B., Maunder, M., Berg, C. W., Brooks, M., Comita, L., … Zipkin, E. (2013). Strategies for fitting nonlinear ecological models in R, AD model builder, and BUGS. *Methods in Ecology and Evolution*, *4*, 501–512. https://doi.org/10.1111/2041-210X.12044

Boukal, D. S., Dieckmann, U., Enberg, K., Heino, M., & Jorgensen, C. (2014). Life-history implications of the allometric scaling of growth. *Journal of Theoretical Biology*, *359*, 199–207. https://doi.org/10.1016/j.jtbi.2014.05.022

Brody, S. (1945). *Bioenergetics and growth with special reference to the energetic efficiency complex in domestic animals*. New York, NY: Reinhold.

Brunel, T., Ernande, B., Mollet, F. M., & Rijnsdorp, A. D. (2013). Estimating age at maturation and energy-based life-history traits from individual growth trajectories with nonlinear mixed-effects models. *Oecologia*, *172*, 631–643. https://doi.org/10.1007/s00442-012-2527-1

Charnov, E. L., Turner, T. F., & Winemiller, K. O. (2001). Reproductive constraints and the evolution of life histories with indeterminate growth. *Proceedings of the National Academy of Sciences of the United States of America*, *98*, 9460–9464. https://doi.org/10.1073/pnas.161294498

Condrey, R., Beckman, D. W., & Wilson, C. A. (1988). Management implications of a new growth model for red drum. Appendix D. In J. A. Shepard (Ed.), *Louisiana red drum research*, *MARFIN final report* (pp. 26–38). Baton Rouge, LA: Louisiana Department of Wildlife and Fisheries, Seafood Division, Finfish Section.

Cotton, C. F., Dean Grubbs, R., Dyb, J. E., Fossen, I., & Musick, J. A. (2015). Reproduction and embryonic development in two species of squaliform sharks, *Centrophorus granulosus* and *Etmopterus princeps*: Evidence of matrotrophy? *Deep-Sea Research II*, *115*, 41–54. https://doi.org/10.1016/j.dsr2.2014.10.009

Craig, P. C., Choat, J. H., Axe, L. M., & Saucerman, S. (1997). Population biology and harvest of the coral reef surgeonfish *Acanthurus lineatus* in American Samoa. *Fishery Bulletin*, *95*, 680–693.

Day, T., & Taylor, P. D. (1997). von Bertalanffy's growth equation should not be used to model age and size at maturity. *The American Naturalist*, *149*, 381–393. https://doi.org/10.1086/285995

Denwood, M. J. (2016). runjags: An R package providing interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. *Journal of Statistical Software*, *71*, 1–25. https://doi.org/10.18637/jss.v071.i09

Dmitriew, C. M. (2011). The evolution of growth trajectories: What limits growth rate? *Biological Reviews*, *86*, 97–116. https://doi.org/10.1111/j.1469-185X.2010.00136.x

Dobbertin, M. (2005). Tree growth as indicator of tree vitality and of tree reaction to environmental stress: A review. *European Journal of Forest Research*, *124*, 319–333. https://doi.org/10.1007/s10342-005-0085-3

Enberg, K., Jørgensen, C., Dunlop, E. S., Heino, M., & Dieckmann, U. (2009). Implications of fisheries-induced evolution for stock rebuilding and recovery. *Evolutionary Applications*, *2*, 394–414. https://doi.org/10.1111/j.1752-4571.2009.00077.x

Fiorentino, F., Gancitano, V., Gancitano, S., Rizzo, P., & Ragonese, S. (2013). An updated two-phase model for demersal fish with an application to red mullet (*Mullus barbatus* L., 1758) (Perciformes Mullidae) of the Mediterranean. *Naturalista Siciliano*, *32*(Suppl. 4), 529–542.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., & Rubin, D. (2013). *Bayesian data analysis* (3rd ed.). New York, NY: Chapman and Hall/CRC Press.

Giacomini, H. C., & Shuter, B. J. (2013). Adaptive responses of energy storage and fish life histories to climatic gradients. *Journal of Theoretical Biology*, *339*, 100–111. https://doi.org/10.1016/j.jtbi.2013.08.020

Glazier, D. S. (2010). A unifying explanation for diverse metabolic scaling in animals and plants. *Biological Reviews*, *85*, 111–138. https://doi.org/10.1111/brv.2010.85.issue-1

Hearn, W. S., & Polacheck, T. (2003). Estimating long-term growth-rate changes of southern bluefin tuna (*Thunnus maccoyii*) from two periods of tag-return data. *Fishery Bulletin*, *101*, 58–74. Retrieved from http://aquaticcommons.org/id/eprint/15106

Helser, T. E., & Lai, H. L. (2004). A Bayesian hierarchical meta-analysis of fish growth: With an example for North American largemouth bass, *Micropterus salmoides*. *Ecological Modelling*, *178*, 399–416. https://doi.org/10.1016/j.ecolmodel.2004.02.013

Hoese, H. D., Beckman, D. W., Blanchet, R. H., Drullinger, D., & Nieland, D. L. (1991). *A biological and fisheries profile of Louisiana red drum* Sciaenops ocellatus. Fishery mangement plan series, #4, part 1. Baton Rouge, LA: Louisiana Department of Wildlife and Fisheries.

Honsey, A. E., Staples, D. F., & Venturelli, P. A. (2017). Accurate estimates of age-at-maturity from the growth trajectories of fishes and other ectotherms. *Ecological Applications*, *27*, 182–192. https://doi.org/10.1002/eap.2017.27.issue-1

Johnston, F. D., Arlinghaus, R., & Dieckmann, U. (2010). Diversity and complexity of angler behaviour drive socially optimal input and output regulations in a bioeconomic recreational-fisheries model. *Canadian Journal of Fisheries and Aquatic Sciences*, *67*, 1507–1531. https://doi.org/10.1139/F10-046

Kozlowski, J. (1996). Optimal allocation of resources explains interspecific life-history patterns in animals with indeterminate growth. *Proceedings of the Royal Society of London B*, *263*, 559–566. https://doi.org/10.1098/rspb.1996.0084

Laslett, G. M., Eveson, J. P., & Polacheck, T. (2002). A flexible maximum likelihood approach for fitting growth curves to tag-recapture data. *Canadian Journal of Fisheries and Aquatic Sciences*, *59*, 976–986. https://doi.org/10.1139/f02-069

Lester, N. P., Shuter, B. J., & Abrams, P. A. (2004). Interpreting the von Bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society of London B*, *271*, 1625–1631. https://doi.org/10.1098/rspb.2004.2778

Lester, N. P., Shuter, B. J., Venturelli, P. A., & Nadeau, D. (2014). Life-history plasticity and sustainable exploitation: A theory of growth compensation applied to walleye management. *Ecological Applications*, *24*, 38–54. https://doi.org/10.1890/12-2020.1

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, *180*, 4–22. https://doi.org/10.1016/j.fishres.2016.01.006

Matthias, B. G., Ahrens, R. N. M., Allen, M. S., Lombardi-Carlson, L. A., & Fitzhugh, G. R. (2016). Comparison of growth models for sequential hermaphrodites by considering multi-phasic growth. *Fisheries Research*, *179*, 67–75. https://doi.org/10.1016/j.fishres.2016.02.006

Maunder, M. N., Crone, P. R., Punt, A. E., Valero, J. L., & Semmens, B. X. (2016). Growth: Theory, estimation, and application in fishery stock assessment models. *Fisheries Research*, *180*, 1–3. https://doi.org/10.1016/j.fishres.2016.03.005

McDermid, J. L., Shuter, B. J., & Lester, N. P. (2010). Life history differences parallel environmental differences among North American lake trout (*Salvelinus namaycush*) populations. *Canadian Journal of Fisheries and Aquatic Sciences*, *67*, 314–325. https://doi.org/10.1139/F09-183

Minte-Vera, C. V., Maunder, M. N., Casselman, J. M., & Campana, S. E. (2016). Growth functions that incorporate the cost of reproduction. *Fisheries Research*, *180*, 31–44. https://doi.org/10.1016/j.fishres.2015.10.023

Moe, B. J. (2015). *Estimating growth and mortality in elasmobranchs: are we doing it correctly?* Nova Southeastern University. Retrieved from http://nsuworks.nova.edu/occ_stuetd/42

Mollet, F. M., Ernande, B., Brunel, T., & Rijnsdorp, A. D. (2010). Multiple growth-correlated life history traits estimated simultaneously in individuals. *Oikos*, *119*, 10–26. https://doi.org/10.1111/oik.2010.119.issue-1

Nakayama, S., Rapp, T., & Arlinghaus, R. (2017). Fast-slow life history is correlated with individual differences in movements and prey selection in an aquatic predator in the wild. *Journal of Animal Ecology*, *86*, 192–201. https://doi.org/10.1111/1365-2656.12603

Ogle, D. H. (2016). *Introductory fisheries analyses with R*. Boca Raton, FL: CRC Press.

Ohnishi, S., Yamakawa, T., Okamura, H., & Akamine, T. (2012). A note on the von Bertalanffy growth function concerning the allocation of surplus energy to reproduction. *Fishery Bulletin*, *110*, 223–229. Retrieved from http://aquaticcommons.org/id/eprint/8682

Okie, J. G., Boyer, A. G., Brown, J. H., Costa, D. P., Ernest, S. K. M., Evans, A. R., … Sibly, R. M. (2013). Effects of allometry, productivity and lifestyle on rates and limits of body size evolution. *Proceedings of the Royal Society B: Biological Sciences*, *280*, 20131007. https://doi.org/10.1098/rspb.2013.1007

Paine, C. E. T., Marthews, T. R., Vogt, D. R., Purves, D., Rees, M., Hector, A., & Turnbull, L. A. (2012). How to fit nonlinear plant growth models and calculate growth rates: An update for ecologists. *Methods in Ecology and Evolution*, *3*, 245–256. https://doi.org/10.1111/j.2041-210X.2011.00155.x

Paloheimo, J. E., & Dickie, L. M. (1965). Food and growth of fishes. I. A growth curve derived from experimental data. *Journal of the Fisheries Research Board of Canada*, *22*, 521–542. https://doi.org/10.1139/f65-048

Pardo, S. A., Cooper, A. B., & Dulvy, N. K. (2013). Avoiding fishy growth curves. *Methods in Ecology and Evolution*, *4*, 353–360. https://doi.org/10.1111/mee3.2013.4.issue-4

Parker, R. R., & Larkin, P. A. (1959). A concept of growth in fishes. *Journal of the Fisheries Research Board of Canada*, *16*, 721–745. https://doi.org/10.1139/f59-052

Plummer, M. (2003). JAGS: A program for analysis of Bayesian graphical models using Gibbs sampling. *Proceedings of the 3rd International Workshop on Distributed Statistical Computing*, *124*, 125. Retrieved from https://www.r-project.org/conferences/DSC-2003/

Porch, C. E., Wilson, C. A., & Nieland, D. L. (2002). A new growth model for red drum (*Sciaenops ocellatus*) that accomodates seasonal and ontogenetic changes in growth rates. *Fishery Bulletin*, *100*, 149–152. Retrieved from http://aquaticcommons.org/id/eprint/15199

Quince, C., Abrams, P. A., Shuter, B. J., & Lester, N. P. (2008a). Biphasic growth in fish I: Theoretical foundations. *Journal of Theoretical Biology*, *254*, 197–206. https://doi.org/10.1016/j.jtbi.2008.05.029

Quince, C., Abrams, P. A., Shuter, B. J., & Lester, N. P. (2008b). Biphasic growth in fish II: Empirical assessment. *Journal of Theoretical Biology*, *254*, 207–214. https://doi.org/10.1016/j.jtbi.2008.05.030

R Core Team. (2016). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved form https://www.R-project.org

Rennie, M. D., Collins, N. C., Shuter, B. J., Rajotte, J. W., & Couture, P. (2005). A comparison of methods for estimating activity costs of wild fish populations: More active fish observed to grow slower. *Canadian Journal of Fisheries and Aquatic Sciences*, *62*, 767–780. https://doi.org/10.1139/f05-052

Rennie, M. D., & Venturelli, P. A. (2015). The ecology of lifetime growth in percid fishes. In P. Kestemont, K. Dabrowski, & R. Summerfelt (Eds.), *Biology and culture of percid fishes* (pp. 499–536). Dordrecht, The Netherlands: Springer.

Rijnsdorp, A. D., & Storbeck, F. (1995). Determining the onset of sexual maturity from otoliths of individual female North Sea plaice, *Pleuronectes platessa* L. In D. Secor, J. Dean & S. Campana (Eds.), *Recent developments in fish otholith research* (pp. 581–598). Columbia, SC: University of South Carolina Press.

Roff, D. (1983). An allocation model of growth and reproduction in fish. *Canadian Journal of Fisheries and Aquatic Sciences*, *40*, 1395–1404. https://doi.org/10.1139/f83-161

Roff, D. A., Heibo, E., & Vøllestad, L. A. (2006). The importance of growth and mortality costs in the evolution of the optimal life history. *Journal of Evolutionary Biology*, *19*, 1920–1930. https://doi.org/10.1111/jeb.2006.19.issue-6

Royle, J. A., & Dorazio, R. M. (2008). *Hierarchical modeling and inference in ecology*. San Diego, CA: Academic Press.

Scott, R. D., & Heikkonen, J. (2012). Estimating age at first maturity in fish from change-points in growth rate. *Marine Ecology Progress Series*, *450*, 147–157. https://doi.org/10.3354/meps09565

Shuter, B. J., Jones, M. L., Korver, R. M., & Lester, N. P. (1998). A general, life history based model for regional management of fish stocks: The inland lake trout (*Salvelinus namaycush*) fisheries of Ontario. *Canadian Journal of Fisheries and Aquatic Sciences*, *55*, 2161–2177. https://doi.org/10.1139/f98-055

Shuter, B. J., Lester, N. P., LaRose, J., Purchase, C. F., Vascotto, K., Morgan, G., … Abrams, P. A. (2005). Optimal life histories and food web position: Linkages among somatic growth, reproductive investment, and mortality. *Canadian Journal of Fisheries and Aquatic Sciences*, *62*, 738–746. https://doi.org/10.1139/f05-070

Soriano, M., Moreau, J., Hoenig, J. M., & Pauly, D. (1992). New functions for the analysis of two-phase growth of juvenile and adult fishes, with application to Nile perch. *Transactions of the American Fisheries Society*, *121*, 486–493. https://doi.org/10.1577/1548-8659(1992)121<0486:NFFTAO>2.3.CO;2

Stearns, S. C. (1992). *The evolution of life histories*. London, UK: Oxford University Press.

Tracey, S. R., & Lyle, J. M. (2005). Age validation, growth modeling, and mortality estimates for striped trumpeter (*Latris lineata*) from south-eastern Australia: Making the most of patchy data. *Fishery Bulletin*, *103*, 169–182. Retrieved from http://aquaticcommons.org/id/eprint/9650

Trip, E. D. L., Clements, K. D., Raubenheimer, D., & Choat, J. H. (2014). Temperature-related variation in growth rate, size, maturation and life span in a marine herbivorous fish over a latitudinal gradient. *Journal of Animal Ecology*, *83*, 866–875. https://doi.org/10.1111/1365-2656.12183

Uusi-Heikkilä, S., Lindström, K., Parre, N., Arlinghaus, R., Alós, J., & Kuparinen, A. (2016). Altered trait variability in response to size-selective mortality. *Biology Letters*, *12*, 20160584.

Ward, H. G. M., Post, J. R., Lester, N. P., Askey, P. J., & Godin, T. (2017). Empirical evidence of plasticity in life-history characteristics across climatic and fish density gradients. *Canadian Journal of Fisheries and Aquatic Sciences*, *74*, 464–474. https://doi.org/10.1139/cjfas-2016-0023

West, G. B., Brown, J. H., & Enquist, B. J. (1997). General model for the origin of allometric scaling laws in biology. *Science*, *276*, 122–126. https://doi.org/10.1126/science.276.5309.122

West, G. B., Brown, J. H., & Enquist, B. J. (1999). The fourth dimension of life: Fractal geometry and allometric scaling of organisms. *Science*, *284*, 1677–1679. https://doi.org/10.1126/science.284.5420.1677

West, G. B., Brown, J. H., & Enquist, B. J. (2001). A general model for ontogenetic growth. *Nature*, *413*, 628–631. https://doi.org/10.1038/35098076

SUPPORTING INFORMATION

Additional Supporting Information may be found online in the supporting information tab for this article.

How to cite this article: Wilson KL, Honsey AE, Moe B, Venturelli P. Growing the biphasic framework: Techniques and recommendations for fitting emerging growth models. *Methods Ecol Evol*. 2018;9:822–833. https://doi.org/10.1111/2041-210X.12931

Appendix S1:
Biphasic somatic growth fit in two steps

Brian Moe[1] *and Kyle Wilson*[2]
[1] *Florida State University and* [2] *The University of Calgary*

June 21, 2017

This appendix is in support of the main manuscript in Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

Fitting the Lester biphasic growth model in two pieces

The following code and figures highlight fitting the Lester Growth Model to data-limited fishes, particularly elasmobranchs (Lester, Shuter & Abrams 2004). In this example, the two phases of the Lester model (**immature** and **mature** phases) are fit independently based on individual maturity status of fish. This approach assumes that the **age-at-maturity** is known.

Data generation

We come up with true growth parameters for the model. The age range of the fishes were read in as a vector of integers. True somatic growth rate can be any positive number and represents the length (in mm) accumulated per year in the late-stage juvenile phase. The variable t_1 represents the hypothetical age when length is 0, it is the *x*-intercept for the juvenile phase. We use a reasonable value for cv of 15%, which is a very typical observation for the coefficient of variation in length-at-age among most fishes (see review in Lorenzen (2016)). The variable T (or Tmat in the R code below) represents the age-at-maturity for the population of fish. This could be estimated prior to the growth assessment study or taken from the literature.*g* represents the proportion of energy in the adult phase allocated to reproduction per year. Note that *g* must be positive and has an intrinsic maximum such that:

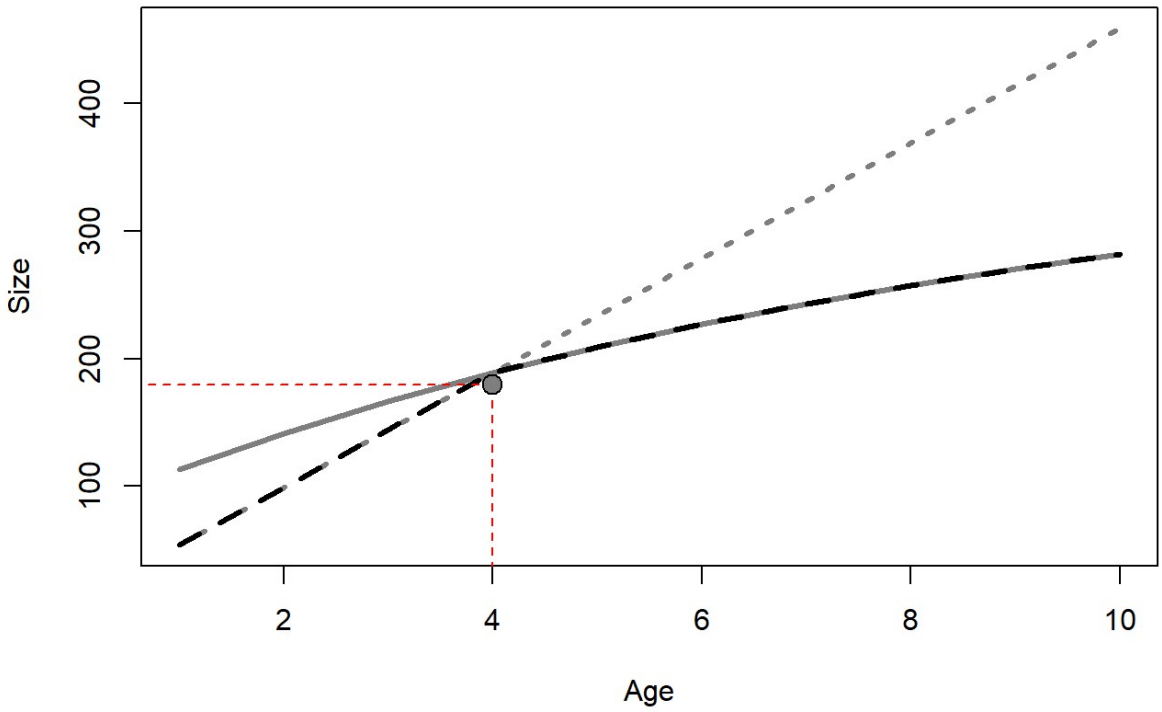
$$g \sim \{0, 3/(T - t_1)\}$$

In this case, we put the true *g* value at the halfway point between 0 and $3/(T - t_1)$.

```
ages <- 1:10
Tmat <- 4  # True age at maturity
h.true <- 45  # somatic growth in millimeters per year
t1.true <- -0.2  #age when size=0 for the juvenile phase
g.true <- 0.5 * (3/(Tmat - t1.true))  # cost of reproduction
cv <- 0.15  # coefficient of variation in size-at-age
```

Next, we convert the Lester parameters to the von Bertalanffy parameters which describe the asymptotic growth of the adult phase.

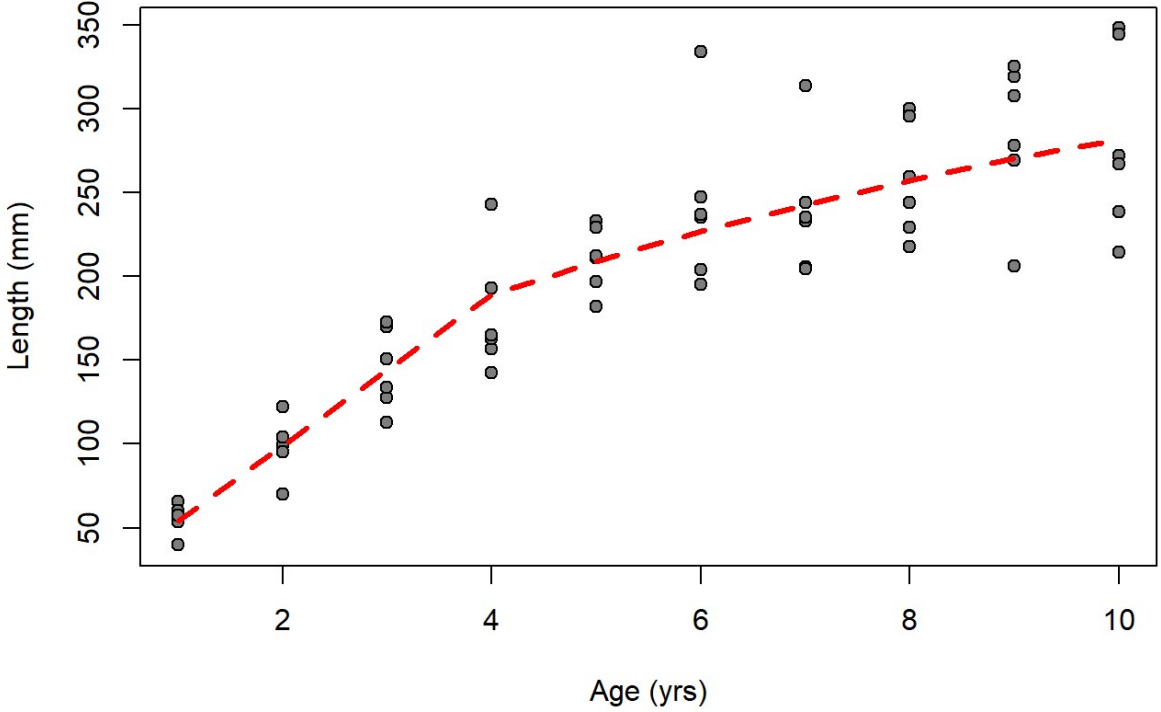
```
linf <- 3 * h.true/g.true  # convert to VBGF L-infinity
vbk <- log(1 + g.true/3)  # convert to VBGF  k
t0 <- Tmat + log(1 - g.true * (Tmat - t1.true)/3)/log(1 + g.true/3)  #convert to VBGF t0
lena_phase1 <- h.true * (ages - t1.true)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages < Tmat, lena_phase1, lena_phase2)
# if-else statement determines whether fish has exceeded age-at-maturity,
# and thus allocates surplus energy into reproduction
plot(ages, lena_phase1, ylab = "Size", xlab = "Age", lty = 3, type = "l", col = "grey50",
    lwd = 3)
lines(ages, lena_phase2, col = "grey50", lwd = 3)
lines(ages, biphasic, lty = 2, lwd = 3)
segments(x0 = Tmat, x1 = Tmat, y0 = 0, y1 = h.true * Tmat + t1.true, col = "red",
    lty = 2)  #plot where maturity occurs
segments(x0 = 0, x1 = Tmat, y0 = h.true * Tmat + t1.true, y1 = h.true * Tmat +
    t1.true, col = "red", lty = 2)  #plot where maturity occurs
points(Tmat, h.true * Tmat + t1.true, pch = 21, bg = "grey50", cex = 1.5)
```



Somatic growth occurs in two phases: juvenile and adult. Red dashed lines indicate the age- and length-at-maturity. Dashed grey line indicates juvenile growth (including if juveniles never matured). Solid grey line indicates adult asymptotic growth (including if individuals were born mature and were investing into reproduction from hatch/birth). Black dash line indicates the composite growth trajectory of the two phases.

We then generate fake data using `rnorm()` or some other random number generator of interest. We specify how many fish will be sampled per age bin, how variable the data is within a given length-at-age, generate fake length-at-age data, then compile that data into a data frame object. We set the seed for the random number generator to allow for all users to come to the same results we get in this document: `set.seed(2017)`, but those that wish to bootstrap this approach (or alter the code for some other purpose) may wish to change this (or not run it).

```
N <- 6  # how many samples per age bin
set.seed(2017)
data <- NULL
for (i in 1:max(ages)) {
    sizes <- abs(rnorm(N, biphasic[i], biphasic[i] * cv))
    mature <- ifelse(i < Tmat, rep(0, N), rep(1, N))
    size_age <- cbind(rep(i, N), sizes, mature)
    data <- rbind(data, size_age)
}
colnames(data) <- c("Age", "Length", "Mature")
data <- as.data.frame(data)  # treat this as a data frame where 'age' is known with certainty
Data <- data
head(Data, 10)
##    Age    Length Mature
## 1    1  65.61703      0
## 2    1  53.37394      0
## 3    1  59.98701      0
## 4    1  39.75530      0
## 5    1  53.43442      0
## 6    1  57.66043      0
## 7    2  69.91826      0
## 8    2  98.97736      0
## 9    2  95.05976      0
## 10   2 122.21386      0

plot(Data$Age, Data$Length, pch = 21, bg = "grey50", xlab = "Age (yrs)", ylab = "Length (mm)")
lines(ages, biphasic, lty = 2, lwd = 3, col = "red")
```



True growth trajectory (line) and randomly generated length-at-age observations (points).

Statistical analysis

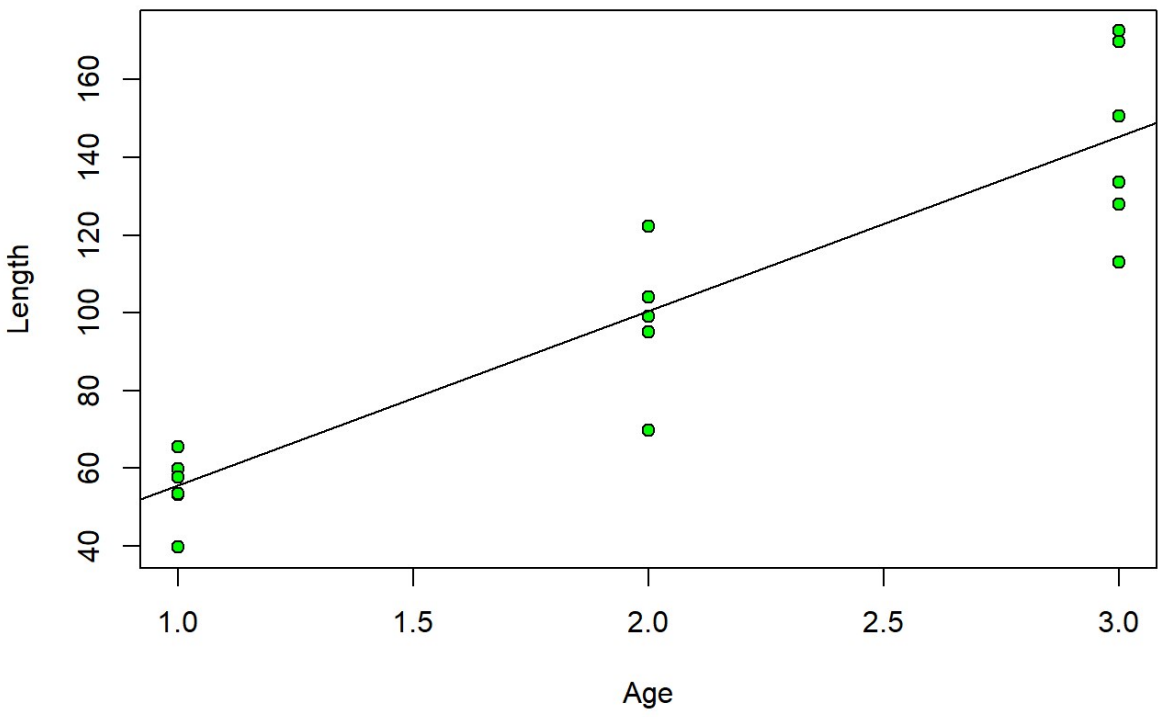
We now subset the data into two datasets: (1) the immature fish and (2) the mature fish. The goal is to evaluate a direct interpretation of the Lester Growth Model which is: (1) linear growth for the late-stage juveniles and (2) nonlinear, asymptotic growth for mature fish that is von Bertalanffy in shape due to investment into reproduction. We fit the linear model using `lm()` (or some variant) and the asymptotic model using `nls()` (nonlinear least squares). The important part for the `nls()` adult growth estimation is that we use the life-history parameters estimated from the juvenile phase (which have information on the shape of the adult phase) and treat them as fixed and known with certainty, then estimate the parameter *g* (the cost of reproduction) as the only remaining free variable.

```
Immature = subset(Data, Data$Mature == 0)   #subset immature
Mature = subset(Data, Data$Mature == 1)   #subset mature

T = Tmat
```

Now call `lm()` to create the linear model for late-stage juveniles individuals

```
lmImmature = lm(Length ~ Age, data = Immature)   #create a linear model for immature individuals
summary(lmImmature)
##
## Call:
## lm(formula = Length ~ Age, data = Immature)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -32.25 -10.20   0.17   8.72  27.11
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   10.946     11.197   0.978    0.343
## Age           44.802      5.183   8.643    2e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 17.96 on 16 degrees of freedom
## Multiple R-squared:  0.8236, Adjusted R-squared:  0.8126
## F-statistic: 74.71 on 1 and 16 DF,  p-value: 2.005e-07
coef(lmImmature)
## (Intercept)         Age
##    10.94586    44.80169
h = lmImmature[[1]][[2]]   #identify h (immature growth rate)
```

```
t1 = lmImmature[[1]][[1]]/-h  #identify t1(x-int from regression)
plot(Immature$Age, Immature$Length, ylab = "Length", xlab = "Age", pch = 21,
    bg = "green")
abline(lmImmature)
```



Linear model fitted to juvenile length-at-age data

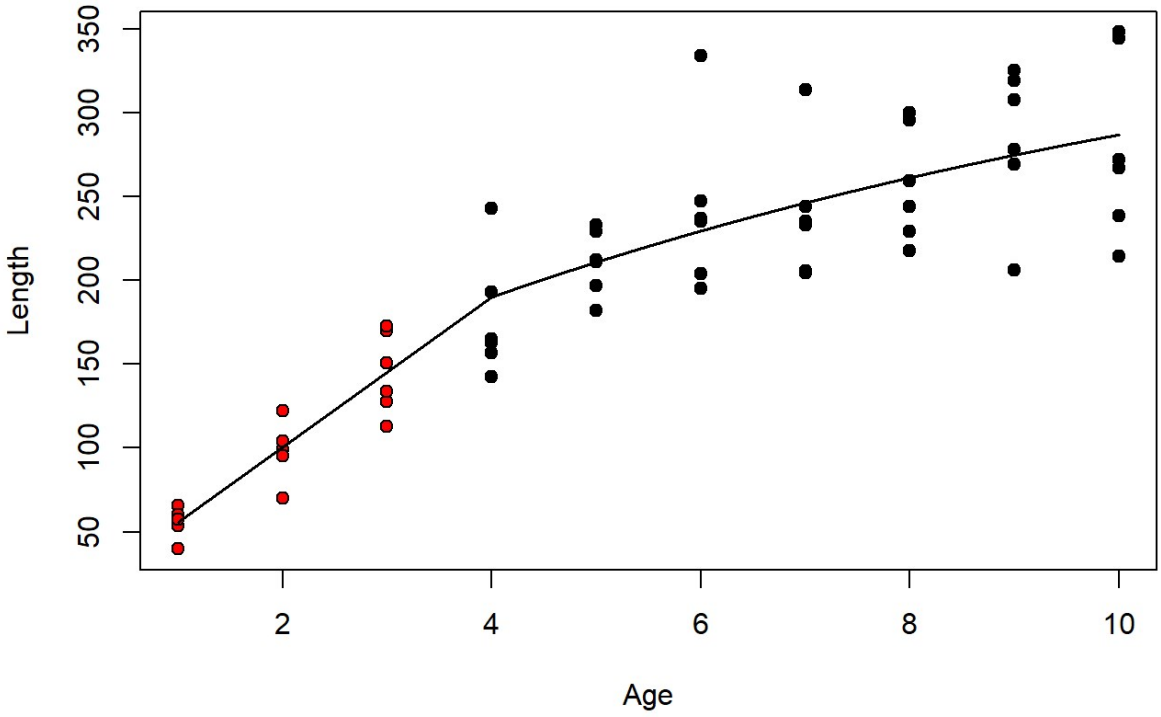
Describe and fit the Lester Growth Model to mature individuals as per Moe (2015)

We use `nls()` to estimate the nonlinear adult growth. We try to use reasonable estimates of g (*warning* you may have to try a few estimates depending on the life history of the fish, as not all values of *g* are reasonable see our text in Wilson *et al.* (2017)).

```
Lester = Length ~ ((3 * h)/g) * (1 - exp(-(log(1 + (g/3))) * (Age - (T + (log(1 -
    ((g * (T - t1))/3))/log(1 + (g/3)))))))
# Because age at maturity and immature growth parameters are know, the only
# parameter estimated is reproductive investment (g)
fitLester = nls(Lester, data = Mature, start = list(g = 0.01))  #fit the biphasic model to the subset
ted mature fish
summary(fitLester)
##
## Formula: Length ~ ((3 * h)/g) * (1 - exp(-(log(1 + (g/3))) * (Age - (T +
##     (log(1 - ((g * (T - t1))/3))/log(1 + (g/3)))))))
##
## Parameters:
##   Estimate Std. Error t value Pr(>|t|)
## g  0.34152    0.02755    12.4 1.87e-15 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 39.01 on 41 degrees of freedom
##
## Number of iterations to convergence: 4
## Achieved convergence tolerance: 7.096e-08
```

Graph the data using `predict()` functions to create a smooth curve. Note that `predict()` only works if the models above identify the data to be used by using the code “data=……” rather than identifying data using $ (ex., `Length$Data~Age$Data`). We add the predictions for the juvenile phase and the adult phase seperately.

```
seqImm = seq(min(Immature$Age), T, by = 0.001)
# create a sequence of lengths ranging from min to max length of immature
# individuals (x-values)
pred1 = predict(lmImmature, newdata = data.frame(Age = seqImm))
# use the 'lmImmature' model to predict y values for the new immature length
# sequence (y-values)
seqMat = seq(T, max(Mature$Age), by = 0.001)
# create a sequence of lengths ranging from min to max length of mature
# individuals (x-values)
pred2 = predict(fitLester, newdata = data.frame(Age = seqMat))
# use the 'fitLester' glm to predict y values for the new mature length
# sequence (y-values)
plot(Length ~ Age, data = Data, xlab = "Age", ylab = "Length", pch = 21, bg = ifelse(Data$Mature ==
    0, "red", "black"))
# plot the data using min and max values of the data sets to set up the axes
# (ylim & xlim)
lines(seqImm, pred1, lwd = 1.75)  #add the regression for immature individuals using the seqImm and p
red1 you created
lines(seqMat, pred2, lwd = 1.75)  #add the regression for mature individuals using the seqMat and pre
d2 you created
```



Observed (points) and predicted (lines) length-at-age for population of interest.

Confidence intervals

We can use diagnostics or interact with the model objects returned from `lm()` and `nls()` to inspect the fit of the models, or to generate confidence intervals in our parameter estimates.

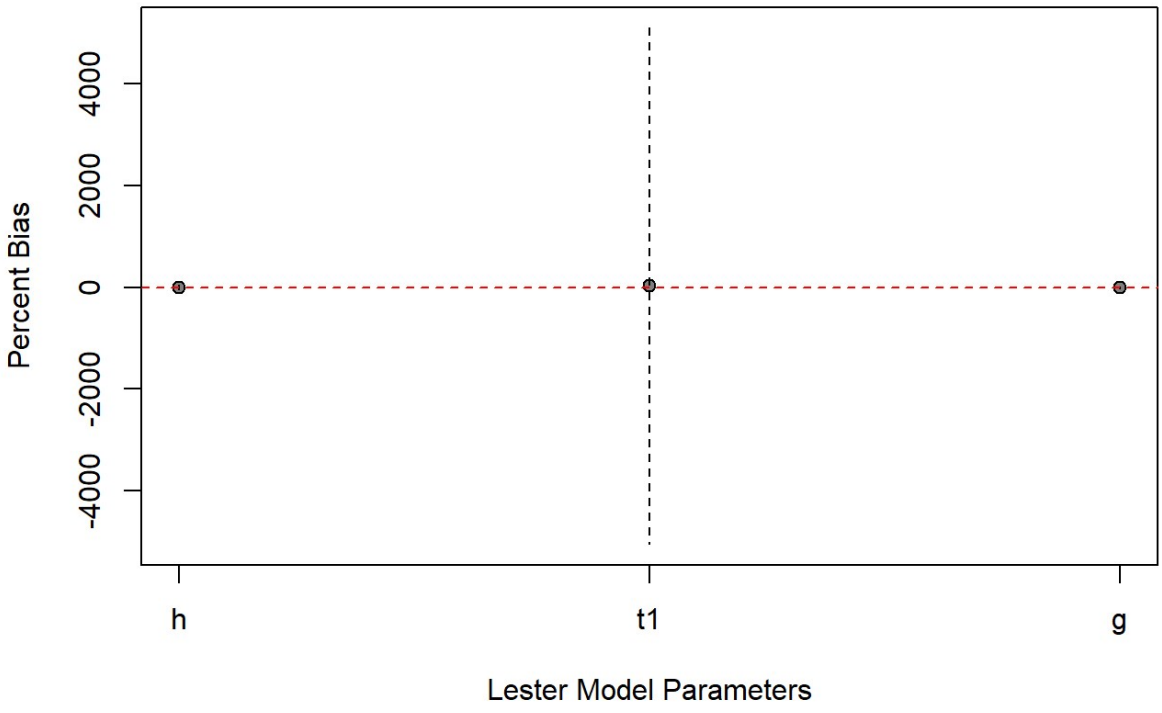
```
UI <- c(h, t1) + 1.96 * as.vector(coef(summary(lmImmature))[, "Std. Error"])
LI <- c(h, t1) - 1.96 * as.vector(coef(summary(lmImmature))[, "Std. Error"])
UIg <- as.vector(coef(fitLester)) + 1.96 * as.vector(coef(summary(fitLester))[,
    "Std. Error"])
LIg <- as.vector(coef(fitLester)) - 1.96 * as.vector(coef(summary(fitLester))[,
    "Std. Error"])
UI <- c(UI, UIg)
LI <- c(LI, LIg)
CI <- data.frame(`Lower 95% CI` = LI, `Upper 95% CI` = UI)
row.names(CI) <- c("h", "t1", "g")
CI
```

```
##    Lower.95..CI Upper.95..CI
## h    22.8548060    66.748579
## t1  -10.4037524     9.915116
## g     0.2875264     0.395505
```

We do some further diagnostics of the model by evaluating how biased the estimates of the growth parameters were in comparison to the true parameters we used to simulate the fake data. To do this we use percent bias:

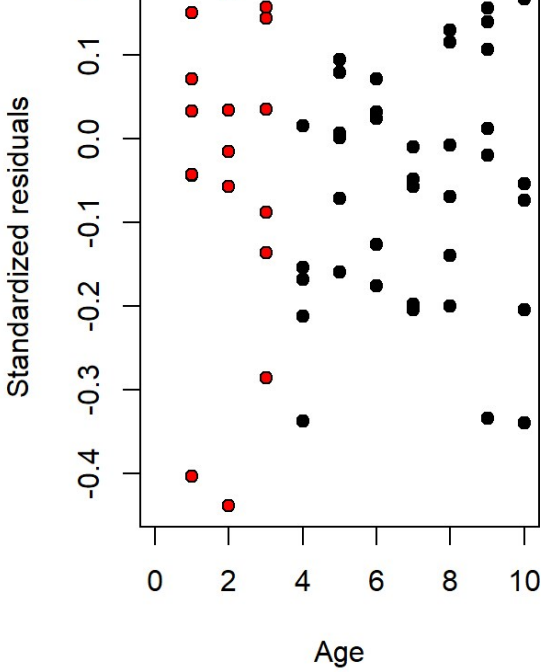
$$Bias = ((L_i - \hat{L}_i)/L_i) * 100$$

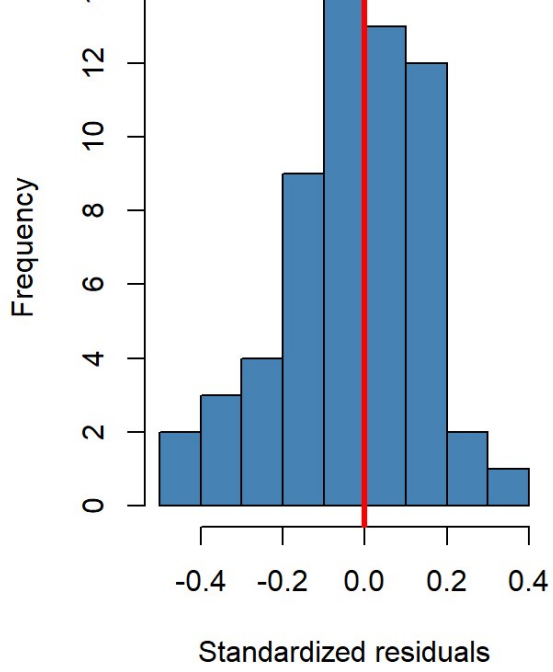
```
par.true <- c(h.true, t1.true, g.true)
plot((c(h, t1, coef(fitLester)[1]) - par.true)/par.true * 100, xlab = "Lester Model Parameters",
    xaxt = "n", ylab = "Percent Bias", ylim = range((c(UI, LI) - par.true)/par.true *
        100), pch = 21, bg = "grey50")

segments(x0 = 1:4, x1 = 1:4, y0 = (LI - par.true)/par.true * 100, y1 = (UI -
    par.true)/par.true * 100, lty = 2, col = "black")

axis(1, at = 1:3, labels = c("h", "t1", "g"))
abline(h = 0, lty = 2, col = "red")
```



Interestingly, most of the bias goes into t_1, while both h and g are estimated reasonably well (see comments in Wilson *et al.* (2017)). Lastly, we visualize standardized residuals for both the juvenile and adult model phases. The residuals appear normally distributed suggesting no systemic bias in the model fitting and the assumptions of normality for the statistical models are met.

```
layout(matrix(1:2, nrow = 1, ncol = 2))
plot(Immature$Age, residuals(lmImmature)/Immature$Length, xlim = c(0, max(ages)),
    pch = 21, bg = "red", ylab = "Standardized residuals", xlab = "Age")
points(Mature$Age, residuals(fitLester)/Mature$Length, pch = 21, bg = "black")

hist(c(residuals(lmImmature)/Immature$Length, residuals(fitLester)/Mature$Length),
    main = "", xlab = "Standardized residuals", col = "steelblue")
abline(v = 0, lwd = 3, col = "red")
```





References

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Moe, B.J. (2015). *Estimating growth and mortality in elasmobranchs: are we doing it correctly?* Master's thesis, Nova Southeastern University.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S2:
Fitting the Lester biphasic growth model with known, fixed age-at-maturity

Kyle Wilson[1] *and Andrew Honsey*[2]
[1]*The University of Calgary*
[2]*University of Minnesota*

June 23, 2017

This appendix is in support of the main manuscript in Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

The following code provides a template for fitting the Lester biphasic growth model to length-at-age data (Lester, Shuter & Abrams 2004). In this example, we assume that age-at-maturity has been estimated a priori, and we treat the estimate as fixed when fitting the model. NB: When using this approach, users should ensure that their estimate for age-at-maturity aligns with the Lester model T parameter (the mean age at which individuals begin to invest energy into reproduction).

Data generation

First, we will generate realistic length-at-age data with a known value for age-at-maturity. The age range of the fishes will be read in as a vector of integers. True somatic growth rate can be any positive number and represents the length (in mm) accumulated per year in the late-stage juvenile phase. The variable t_1 represents the hypothetical age at length 0 (i.e., the *x*-intercept for the juvenile phase). We use a reasonable value for the coefficient of variation in length-at-age (cv) of 15%, a typical observation among most fishes (see review in Lorenzen (2016)). The variable T (or Tmat in the R code below) represents the age-at-maturity for the population of fish. The parameter g represents the proportion of energy in adult phase allocated to reproduction per year. Note that g must be positive and has an intrinsic maximum such that:

$$g \sim \{0, 3/(T - t_1)\}.$$

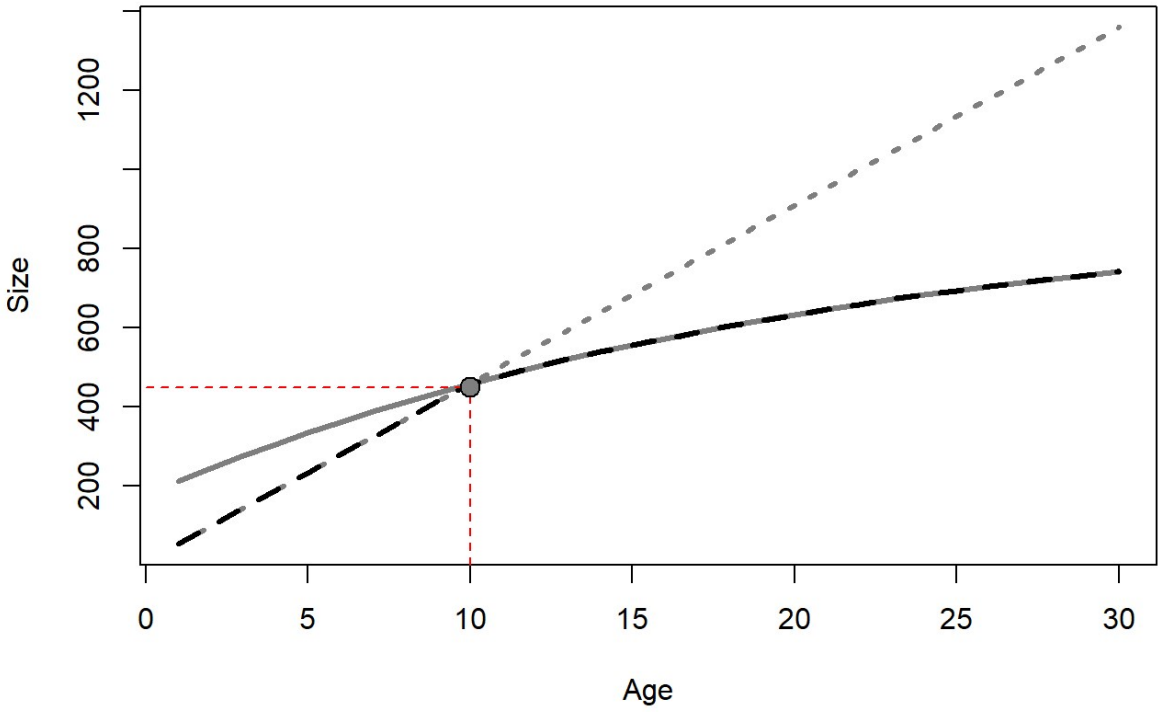
In this case, we put the true g value at the halfway point between 0 and $3/(T - t_1)$.

```
Tmat <- 10  # age-at-maturity
ages <- 1:30
h <- 45  # somatic growth in millimeters per year
t1 <- -0.2  #age when size=0 for the juvenile phase
g <- 0.5 * (3/(Tmat - t1))  # proportion of energy in adult phase allocated to reproduction per year
cv <- 0.15  # coefficient of variation in size-at-age
```

Next, we convert the Lester parameters to von Bertalanffy parameters, which describe the asymptotic growth of the adult phase. We then calculate the 'true' length-at-age for both phases of growth using the parameters specified above. We use an if-else statement to select which 'true' length-at-age value applies for each age, given the *a priori* estimate of age-at-maturity. We then generate a plot of the lifetime growth trajectory, including the entire trajectories for both phases and the point at which growth transitions from the first to the second phase (i.e., age-at-maturity).

```
linf <- 3 * h/g  # conversion for the VBGF L-infinity
vbk <- log(1 + g/3)  # conversion for the VBGF parameter kappa
t0 <- Tmat + log(1 - g * (Tmat - t1)/3)/log(1 + g/3)  #conversion for the VBGF parameter t0

lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages < Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a fish is allocating surplus energy


# generate plot of growth trajectory
```

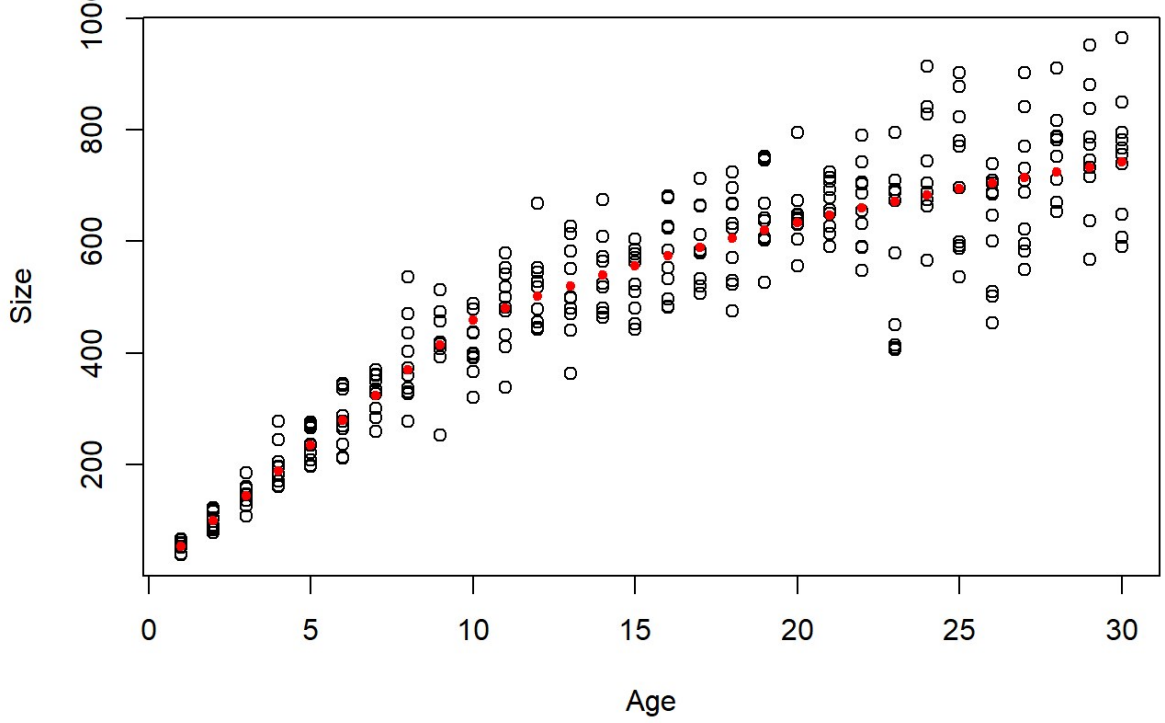
```
layout(matrix(1:1, nrow = 1, ncol = 2))
plot(ages, lena_phase1, ylab = "Size", xlab = "Age", lty = 3, type = "l", col = "grey50",
    lwd = 3)
lines(ages, lena_phase2, col = "grey50", lwd = 3)
lines(ages, biphasic, lty = 2, lwd = 3)
segments(x0 = Tmat, x1 = Tmat, y0 = 0, y1 = h * Tmat + t1, col = "red", lty = 2)  #plot where maturit
y occurs
segments(x0 = 0, x1 = Tmat, y0 = h * Tmat + t1, y1 = h * Tmat + t1, col = "red",
    lty = 2)  #plot where maturity occurs
points(Tmat, h * Tmat + t1, pch = 21, bg = "grey50", cex = 1.5)
```



Somatic growth occurs in two phases: juvenile and adult. Red dashed lines indicate the age- and length-at-maturity. Dashed grey line indicates juvenile growth (including if juveniles never matured). Solid grey line indicates adult asymptotic growth (including if individuals were born mature and were investing into reproduction from hatch/birth). Black dash line indicates the composite growth trajectory of the two phases.

Next, we will generate data using `rnorm()`. We will specify how many individuals will be sampled per age bin, and we will incorporate the coefficient of variation specified above. We compile the length-at-age data into a data frame object. We set the seed for the random number generator to make our results repeatable: `set.seed(2017)`. However, those that wish to bootstrap this approach (or alter the code for some other purpose) may wish to remove this command.

```
N <- 10  # how many samples per age bin
set.seed(2017)
data <- NULL
for (i in 1:max(ages)) {
    sizes <- rnorm(N, biphasic[i], biphasic[i] * cv)

    size_age <- cbind(rep(i, N), sizes)
    data <- rbind(data, size_age)
}
colnames(data) <- c("Age", "Size")  # add column names
data <- as.data.frame(data)  # convert to data frame
plot(data$Age, data$Size, xlab = "Age", ylab = "Size")  # Plot data
points(ages, biphasic, pch = 20, col = "red")  # add 'true' growth trajectory points
```



Likelihood function

Next, we specify the likelihood function. We will estimate four parameters: h, t_1, g, and the cv in length-at-age. We provide conversions for the von Bertalanffy (mature) growth parameters, and we use an if-else statement to differentiate between immature and mature growth. Finally, we used a penalized likelihood to help ensure that estimates of g do not venture outside of analytical bounds (see Lester et al. 2004).

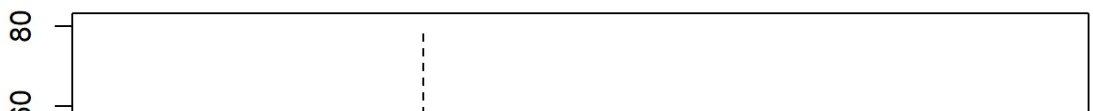
```
## likelihood function

nll <- function(theta) {
    h <- theta[1]
    t1 <- theta[2]
    g <- exp(theta[3])
    size.cv <- theta[4]

    ## Convert to phase 2 VBGF parameters
    linf <- 3 * h/g
    vbk <- log(1 + g/3)
    t0 <- Tmat + log(1 - g * (Tmat - t1)/3)/log(1 + g/3)


    ## Make predictions
    pred1 <- h * (data$Age - t1)  #predicted length for phase 1
    pred2 <- linf * (1 - exp(-vbk * (data$Age - t0)))  #predicted length for phase 2
    pred_all <- ifelse(data$Age < Tmat, pred1, pred2)  #discontinuous maturity breakpoint
    ll <- dnorm(data$Size, mean = pred_all, sd = pred_all * size.cv, log = TRUE)  #normal likelihood
with constant variance
    if (g < 0 | g > (3/(Tmat - t1))) {
        nll <- 1e+06  # penalized likelihood when g goes past bounds given in Lester et al. 2004; g m
ust be > 0 OR < 3/(T-t1)
    } else {
        nll <- -sum(ll)  # return the negative log-likelihood
    }
    return(nll)
}
```

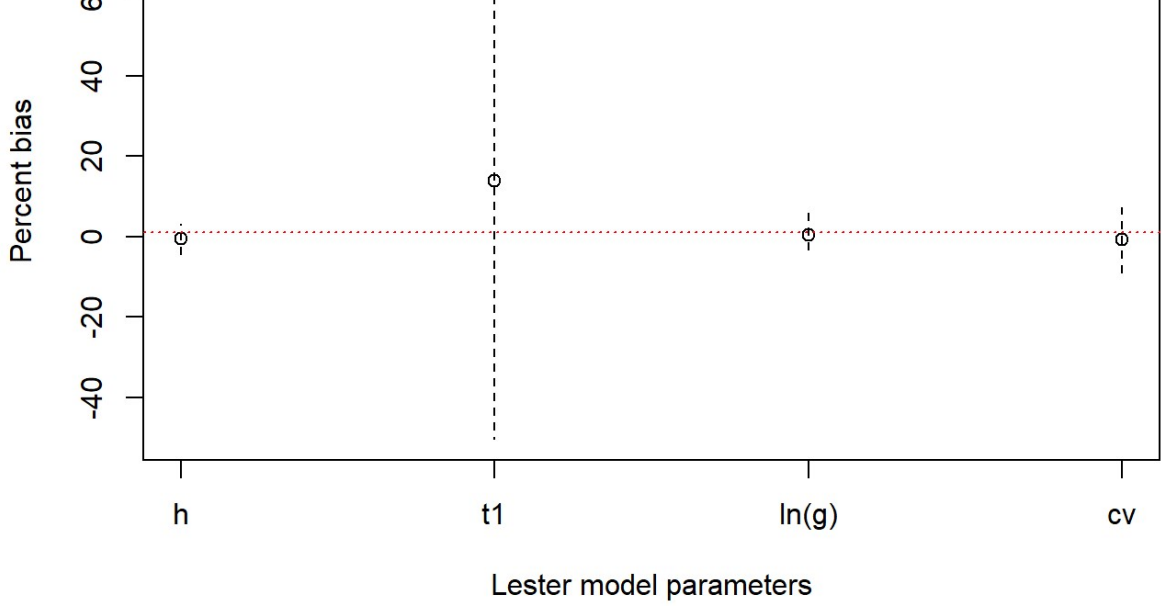
Optimization and visualization of results

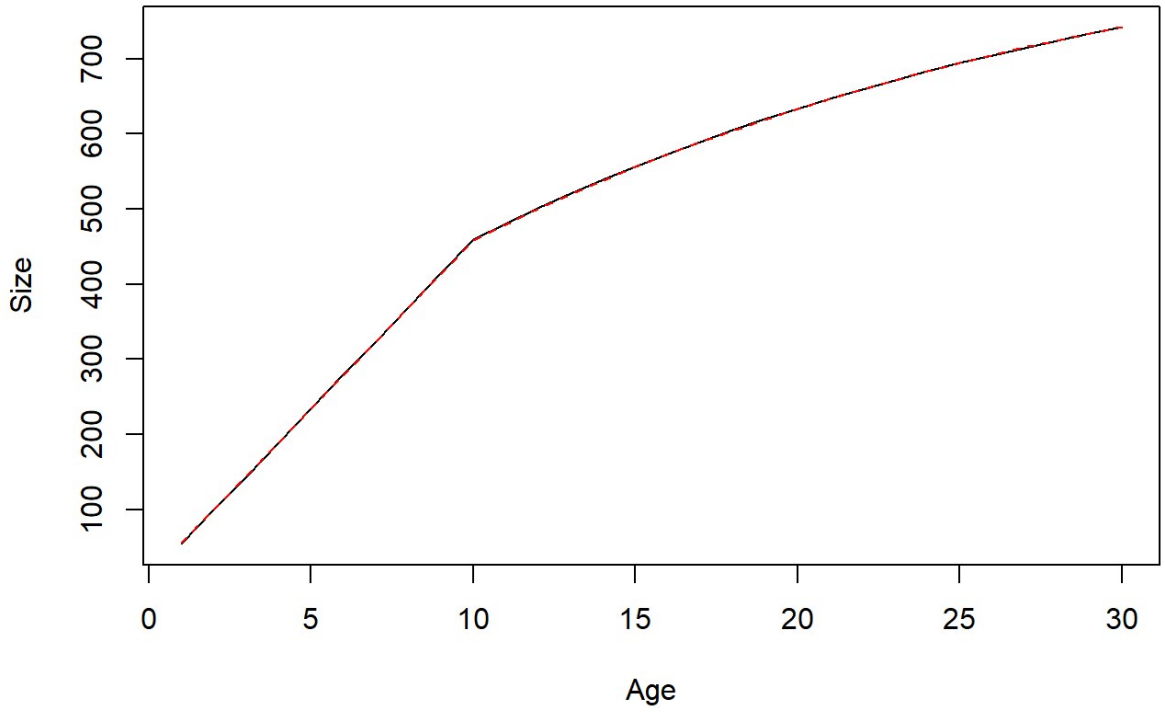
To fit the model, we must first provide and compile starting values for each parameter. We then use the `optim()` function to

optimize the likelihood function for our list of parameters. As mentioned above, we treat T (Tmat) as fixed, and we estimate t_1, h, g, and cv (error term). If desired, one can easily calculate Akaike's information criterion (AIC) for post-hoc model comparisons. The remainder of the code below provides guidelines for extracting and plotting results – see comments for details.

```
## starting values
h.hat <- 35
t1.hat <- -1
g.hat <- log(0.2)
cv.hat <- 0.1

theta <- c(h.hat, t1.hat, g.hat, cv.hat)  # initial parameter estimates

## optimization
fit <- optim(theta, nll, method = "Nelder-Mead", control = list(maxit = 1e+05,
    reltol = 1e-10), hessian = TRUE)

## AIC calculation
AIC <- 2 * length(fit$par) - 2 * (-fit$value)


## plot estimates with 95% asymptotically normal confidence intervals (CI)

h_pred <- fit$par[1]  # extract h estimate
t1_pred <- fit$par[2]  # extract t1 estimate
g_pred <- exp(fit$par[3])  # extract and back-transform g estimate
par.hat <- c(h_pred, t1_pred, log(g_pred), fit$par[4])  # compile parameter estimates
par.true <- c(h, t1, log(g), cv)  # compile 'true' parameter values
fisher_info <- solve(fit$hessian)  #take the inverse of the hessian to get the variance-covariance ma
trix
SE.par <- sqrt(diag(fisher_info))  # square-root the variance-covariance matrix to get standard error
s

UI <- par.hat + 1.96 * SE.par  # upper bounds of 95% CIs
LI <- par.hat - 1.96 * SE.par  # lower bounds of 95% CIs

# generate plot of percent bias in parameter estimates
plot(1:4, (par.hat - par.true)/par.true * 100, ylim = range((c(LI, UI) - par.true)/par.true *
    100), xaxt = "n", ylab = "Percent bias", xlab = "Lester model parameters")
segments(x0 = 1:4, x1 = 1:4, y0 = (LI - par.true)/par.true * 100, y1 = (UI -
    par.true)/par.true * 100, lty = 2, col = "black")
axis(1, at = 1:4, labels = c("h", "t1", "ln(g)", "cv"))
abline(h = 1, lty = 3, col = "red")


## Plot 'true' vs. estimated growth trajectory

# make conversions to phase 2 VBGF parameters
linf.hat <- 3 * h_pred/g_pred
vbk.hat <- log(1 + g_pred/3)
t0.hat <- Tmat + log(1 - g_pred * (Tmat - t1_pred)/3)/log(1 + g_pred/3)

pred.phase1 <- h_pred * (ages - t1_pred)  # predicted growth in phase 1
pred.phase2 <- linf.hat * (1 - exp(-vbk.hat * (ages - t0)))  # predicted growth in phase 2
pred_all <- ifelse(ages < Tmat, pred.phase1, pred.phase2)  # all predictions, similar to likelihood f
unction

# generate plot
plot(ages, biphasic, type = "l", ylab = "Size", xlab = "Age")
lines(ages, pred_all, col = "red", lty = 2)
```







The top panel shows the percent bias for the four estimated parameters, compared to the 'true' values. In this case, the estimates are rather unbiased (although the confidence interval is rather large for t_1). The lower panel displays the estimated growth trajectory (red, dashed line) compared to the 'true' growth trajectory (black line).

References

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S3:
Fitting the Lester biphasic growth model without maturity data using a penalized likelihood approach

Kyle Wilson[1] *and Andrew Honsey*[2]
[1] *The University of Calgary*
[2] *University of Minnesota*

June 26, 2017

This appendix is in support of the main manuscript in Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

The following code is an example application of the Lester biphasic growth model (Lester, Shuter & Abrams 2004) to length-at-age data in the absence of any information on maturity (e.g., maturity data, *a priori* estimates of age-at-maturity, etc.). In this case, the age-at-maturity parameter T is estimated simultaneously with the other model parameters using a penalized likelihood approach.

Data generation

First, we generate realistic length-at-age data with a known value for age-at-maturity. The age range of the fishes were read in as a vector of integers. True somatic growth rate can be any positive number and represents the length (in mm) accumulated per year in the late-stage juvenile phase. The variable t_1 represents the hypothetical age at length 0 (i.e., the *x*-intercept for the juvenile phase). We use a reasonable value for the coefficient of variation in length-at-age (cv) of 15%, a typical observation among most fishes (see review in Lorenzen (2016)). The variable T (or Tmat in the R code below) represents the age-at-maturity for the population. The parameter g represents the proportion of energy in the adult phase allocated to reproduction per unit time (in this case, per year). Note that g must be positive and has an intrinsic maximum such that:

$$g \sim \{0, 3/(T - t_1)\}.$$

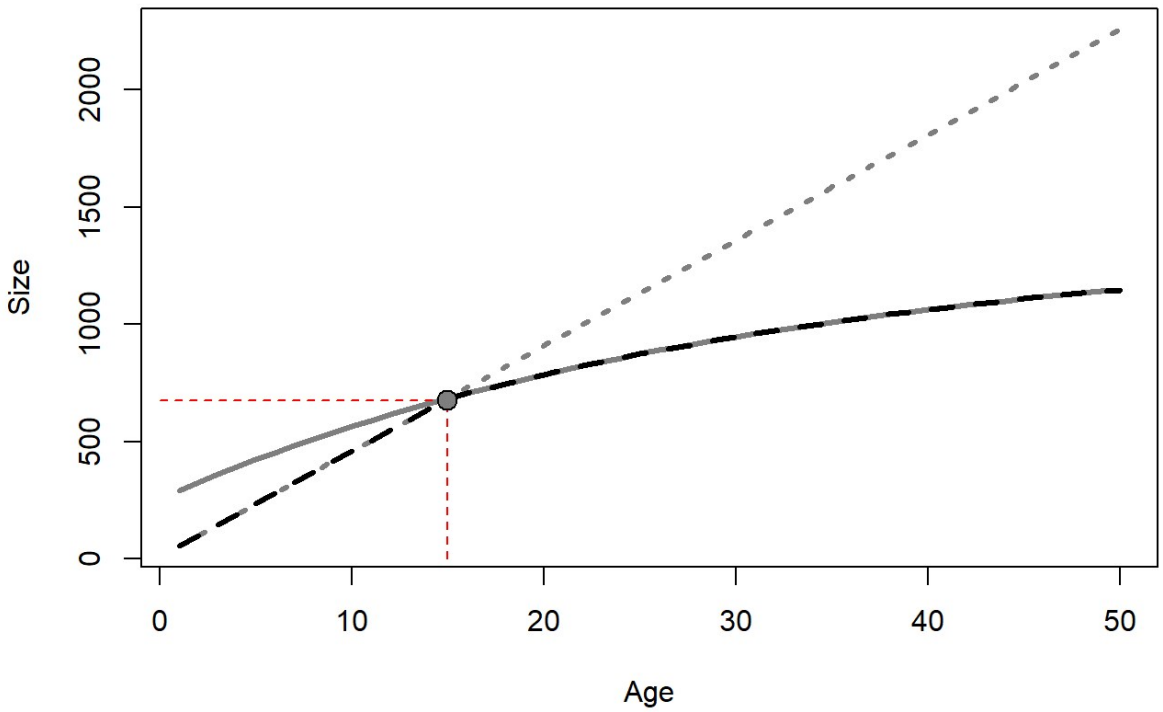
In this case, we put the true g value at the halfway point between 0 and $3/(T - t_1)$.

```
ages <- 1:50
Tmat <- 15  # age at maturity
h <- 45  # somatic growth in millimeters per year
t1 <- -0.2  #age when size=0 for the juvenile phase
g <- 0.5 * (3/(Tmat - t1))  # proportion of energy in adult phase allocated to reproduction per year
cv <- 0.15  # coefficient of variation in size-at-age
```

Next, we convert the Lester parameters to von Bertalanffy parameters, which describe the asymptotic growth of the adult phase. We then calculate the 'true' length-at-age for both phases of growth using the parameters specified above. We use an if-else statement to select which 'true' length-at-age value applies for each age, given the estimate of age-at-maturity. We then generate a plot of the lifetime growth trajectory, including the entire trajectories for both phases and the point at which growth transitions from the first to the second phase (i.e., age-at-maturity).

```
linf <- 3 * h/g  # conversion for the VBGF L-infinity
vbk <- log(1 + g/3)  # conversion for the VBGF parameter kappa
t0 <- Tmat + log(1 - g * (Tmat - t1)/3)/log(1 + g/3)  #conversion for the VBGF parameter t0

lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages < Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a fish is allocating surplus energy
```

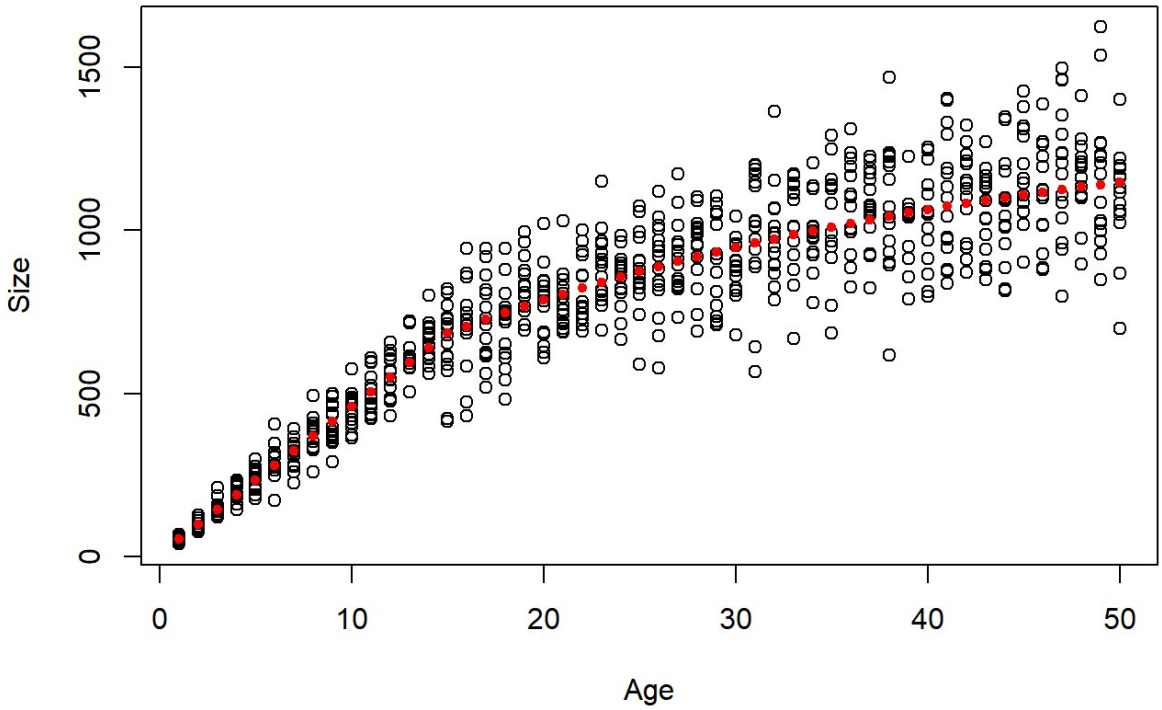
```
# generate plot of growth trajectory
plot(ages, lena_phase1, ylab = "Size", xlab = "Age", lty = 3, type = "l", col = "grey50",
    lwd = 3)
lines(ages, lena_phase2, col = "grey50", lwd = 3)
lines(ages, biphasic, lty = 2, lwd = 3)
segments(x0 = Tmat, x1 = Tmat, y0 = 0, y1 = h * Tmat + t1, col = "red", lty = 2)  #plot where maturit
y occurs
segments(x0 = 0, x1 = Tmat, y0 = h * Tmat + t1, y1 = h * Tmat + t1, col = "red",
    lty = 2)  #plot where maturity occurs
points(Tmat, h * Tmat + t1, pch = 21, bg = "grey50", cex = 1.5)
```



Somatic growth occurs in two phases: juvenile and adult. Red dashed lines indicate the age- and length-at-maturity. Dashed grey line indicates juvenile growth (including if juveniles never matured). Solid grey line indicates adult asymptotic growth (including if individuals were born mature and were investing into reproduction from hatch/birth). Black dash line indicates the composite growth trajectory of the two phases.

Next, we generate data using `rnorm()`. We specify how many individuals will be sampled per age bin, and incorporate the coefficient of variation specified above. We compile the length-at-age data into a data frame object. We set the seed for the random number generator to make our results repeatable: `set.seed(2017)`. However, those that wish to bootstrap this approach (or alter the code for some other purpose) may wish to remove this command.

```
set.seed(2017)
N <- 15  # how many samples per age bin
data <- NULL
for (i in 1:max(ages)) {
    sizes <- rnorm(N, biphasic[i], biphasic[i] * cv)

    size_age <- cbind(rep(i, N), sizes)
    data <- rbind(data, size_age)
}
colnames(data) <- c("Age", "Size")  # add column names
data <- as.data.frame(data)  # convert to data frame
plot(data$Age, data$Size, xlab = "Age", ylab = "Size")  # plot data
points(ages, biphasic, pch = 20, col = "red")  # add 'true' growth trajectory points (in red)
```



Likelihood function

Next, we specify the likelihood function. We estimate five parameters: T, h, t_1, g, and the cv in length-at-age. We provide conversions for the von Bertalanffy (mature) growth parameters, and we use an if-else statement to differentiate between immature and mature growth. Finally, we used a penalized likelihood to help ensure that estimates of g do not venture outside of analytical bounds (see Lester et al. 2004).

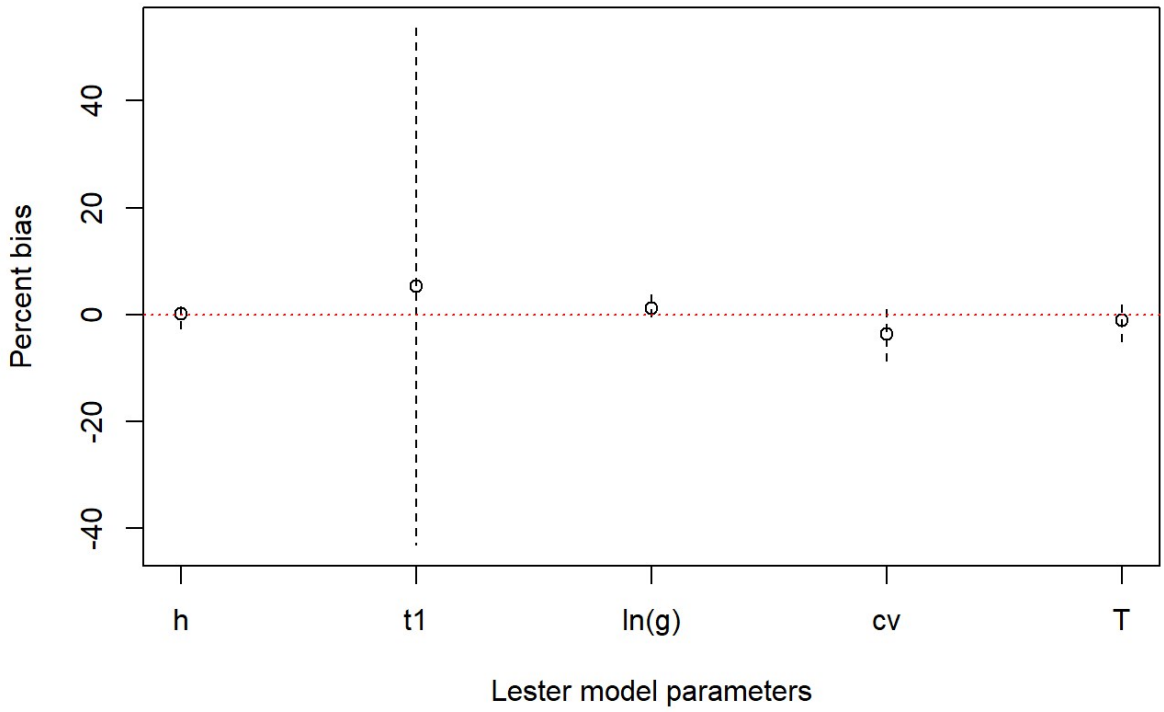
```
## likelihood function

nll <- function(theta) {
    h <- theta[1]
    t1 <- theta[2]
    g <- exp(theta[3])
    size.cv <- theta[4]
    T_pred <- theta[5]

    ## Convert to phase 2 VBGF parameters
    linf <- 3 * h/g
    vbk <- log(1 + g/3)
    t0 <- Tmat + log(1 - g * (T_pred - t1)/3)/log(1 + g/3)


    ## Make predictions
    pred1 <- h * (data$Age - t1)  #predicted length for phase 1
    pred2 <- linf * (1 - exp(-vbk * (data$Age - t0)))  #predicted length for phase 2
    pred_all <- ifelse(data$Age < T_pred, pred1, pred2)  #discontinuous maturity breakpoint
    # points(data$Age,pred_all,col='red')
    ll <- dnorm(data$Size, mean = pred_all, sd = pred_all * size.cv, log = TRUE)  #normal likelihood
with constant variance
    if (g < 0 | g > (3/(T_pred - t1))) {
        nll <- 1e+06  # penalized likelihood when g goes past bounds given in Lester et al. 2004; g m
ust be > 0 OR < 3/(T-t1)
    } else {
        nll <- -sum(ll)  # return the negative log-likelihood
    }
    return(nll)
}
```

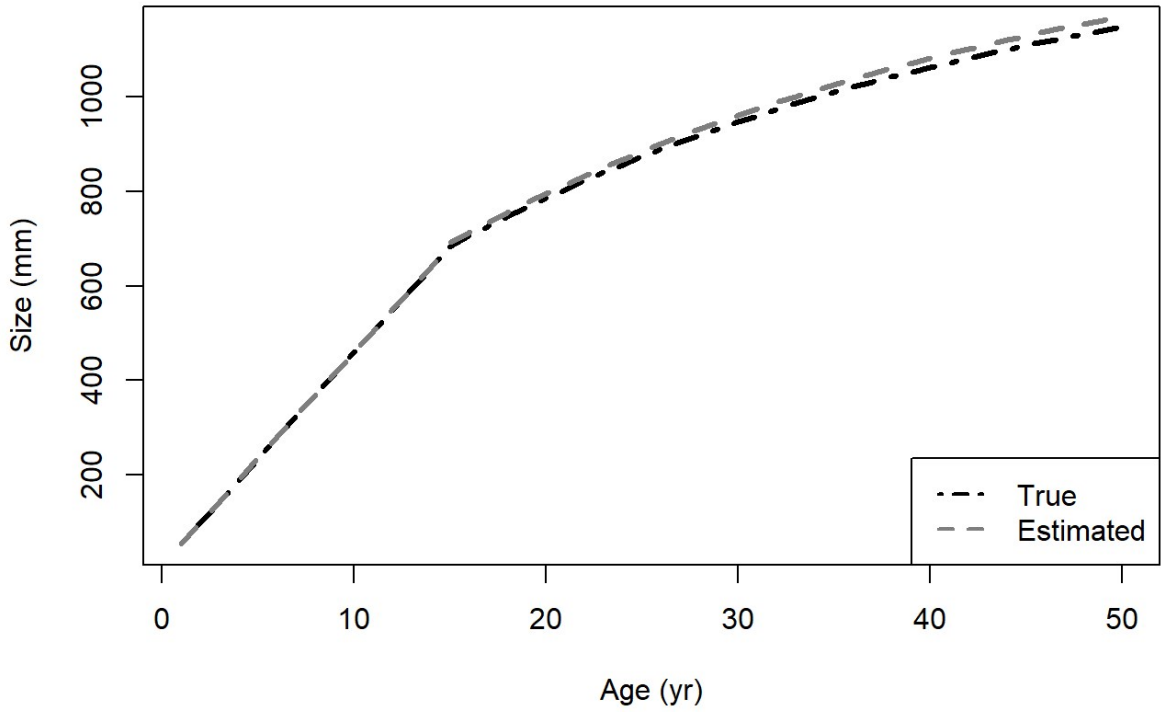
Optimization and visualization of results

To fit the model, we first provide and compile starting values for each parameter. We then use the `optim()` function to optimize the likelihood function for our list of parameters. As mentioned above, we estimate T (Tmat), t_1, h, g, and cv (error term). If desired, one can easily calculate Akaike's information criterion (AIC) for post-hoc model comparisons. The remainder of the code below provides guidelines for extracting and plotting results – see comments for details.

```
## starting values
h.hat <- 35
t1.hat <- -1
g.hat <- log(0.2)
cv.hat <- 0.1
T.hat <- 10
theta <- c(h.hat, t1.hat, g.hat, cv.hat, T.hat)  # initial parameter estimates

## optimization
fit <- optim(theta, nll, method = "Nelder-Mead", control = list(maxit = 1e+05,
    reltol = 1e-10), hessian = TRUE)

## AIC calculation
AIC <- 2 * length(fit$par) - 2 * (-fit$value)


## plot estimates with 95% asymptotically normal confidence intervals (CI)

h_pred <- fit$par[1]  # extract h estimate
t1_pred <- fit$par[2]  # extract t1 estimate
g_pred <- exp(fit$par[3])  # extract and back-transform g estimate
cv_pred <- fit$par[4]  # extract cv estimate
T_pred <- fit$par[5]  # extract T estimate
par.hat <- c(h_pred, t1_pred, log(g_pred), cv_pred, T_pred)  # compile parameter estimates
par.true <- c(h, t1, log(g), cv, Tmat)  # compile 'true' parameter values
fisher_info <- solve(fit$hessian)  # take the inverse of the hessian to get the variance-covariance m
atrix
SE.par <- sqrt(diag(fisher_info))  #square-root the variance-covariance matrix to get standard errors

UI <- par.hat + 1.96 * SE.par  # upper bounds of 95% CIs
LI <- par.hat - 1.96 * SE.par  # lower bounds of 95% CIs

## generate plot of percent bias in parameter estimates
plot(1:5, ((par.hat - par.true)/par.true) * 100, ylim = range((((c(LI, UI) -
    par.true)/par.true) * 100)), xaxt = "n", ylab = "Percent bias", xlab = "Lester model parameters")
segments(x0 = 1:5, x1 = 1:5, y0 = ((LI - par.true)/par.true) * 100, y1 = ((UI -
    par.true)/par.true) * 100, lty = 2, col = "black")
axis(1, at = 1:5, labels = c("h", "t1", "ln(g)", "cv", "T"))
abline(h = 0, lty = 3, col = "red")


## Plot 'true' vs. estimated growth trajectory

# make conversions to phase 2 VBGF parameters

linf.hat <- 3 * h_pred/g_pred
vbk.hat <- log(1 + g_pred/3)
t0.hat <- Tmat + log(1 - g_pred * (T_pred - t1_pred)/3)/log(1 + g_pred/3)

pred.phase1 <- h_pred * (ages - t1_pred)  # predicted growth in phase 1
pred.phase2 <- linf.hat * (1 - exp(-vbk.hat * (ages - t0)))  # predicted growth in phase 2
pred_all <- ifelse(ages < T_pred, pred.phase1, pred.phase2)  # all predictions, similar to likelihood
 function


# generate plot
plot(ages, biphasic, type = "l", xlab = "Age (yr)", ylab = "Size (mm)", lwd = 3,
    lty = 4)
lines(ages, pred_all, col = "grey50", lty = 2, lwd = 3)
legend("bottomright", legend = c("True", "Estimated"), col = c("black", "grey50"),
```

```
    lty = c(4, 2), lwd = 2)
```





The top panel shows the percent bias for the five estimated parameters, compared to the 'true' values. In this case, the estimates are very slightly biased in a couple of cases (likely due to sampling around the 'true' parameter values), and the confidence interval is rather large for t_1. The lower panel displays the estimated growth trajectory (gray, dashed line) compared to the 'true' growth trajectory (black, dot-dashed line).

References

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its

consequences. *Fisheries Research*, **180**, 4–22.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S4:
Fitting the Lester biphasic growth model without maturity data using a profile likelihood approach

Andrew Honsey[1] *and Kyle Wilson*[2]
[1] *University of Minnesota*
[2] *The University of Calgary*

October 5, 2017

This appendix is in support of Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

This file contains example code for estimating age-at-maturity and other life history parameters from length-at-age data using a Lester biphasic growth model (Lester, Shuter & Abrams 2004). The model is fit using a profile likelihood approach. This code is similar to that used in (Honsey, Staples & Venturelli 2016).

First, we load the required `boot` library.

```
# Load required library
library(boot)  # use install.packages('boot') if package isn't already installed
```

Data generation

We generate realistic length-at-age data based on the Lester model. Somatic growth rate h (h.1 in the code below) can be any positive number and represents the length (in mm) accumulated per year in the late-stage juvenile phase. The variable l_0 (`l.0`) represents the y-intercept of the juvenile growth phase. We use a reasonable value for the precision (prec; i.e., the inverse of coefficient of variation) in length-at-age of 12, a typical observation for fisheries data (see review in Lorenzen (2016)). The variable T (`t`) represents the age at which individuals begin to invest energy into reproduction (i.e., age-at-maturity). The parameter g represents the cost to somatic growth of maturity, which is often assumed to be dominated by investment in reproduction. Note that g must be positive and has an intrinsic maximum such that:

$$g \sim \{0, 3/(T - t_1)\}.$$

In this case, we set $g = 0.25\ yr^{-1}$.

```
## Input known parameters for simulating data
l.0 <- 100  ## y-intercept of immature phase (mm)
h.1 <- 50  ## immature somatic growth rate (h; mm/yr)
g.mat <- 0.25  ## cost to somatic growth of maturity (g; equivalent energetic units)
t.1 = -l.0/h.1  ##  Lester hypothetical age at length 0 (yr)
t = 5  ## Lester model age-at-maturity (yr)
prec = 12  # precision in length-at-age (inverse of coefficient of variation) -- adjust as needed. Mo
del estimates will become more inaccurate as precision decreases. Precision = 12 is realistic for fis
heries data.
```

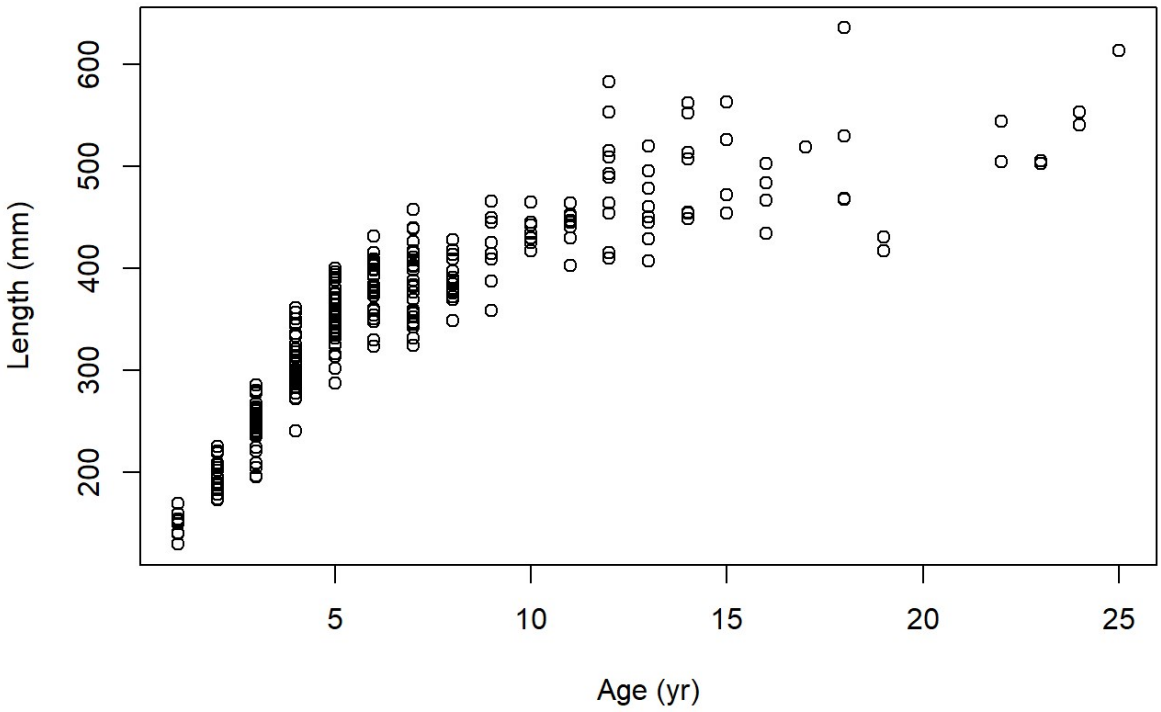
Next, we convert the Lester parameters to von Bertalanffy parameters, which describe the asymptotic growth of the adult phase. We then calculate the mean lengths-at-age for both phases of growth.

```
L.inf = 3 * h.1/g.mat  ## von Bertalanffy asymptotic length (mm)
k.mat = log(1 + g.mat/3)  ## Brody growth coefficient
t.0 = t + log(1 - (g.mat * (t - t.1)/3))/log(1 + g.mat/3)  ## von Bertalanffy hypothetical age at len
gth 0 (yr)
```

```r
# Generate mean lengths-at-age for all fish (immature and mature), based on
# input parameters
imm.list <- c(1:t)  # immature ages
mat.list <- c((t + 1):25)  # Mature ages, with maximum age = 25 years
imm.means <- sapply(imm.list, function(x) l.0 + h.1 * x)  # mean lengths-at-age for immature fish
mat.means <- sapply(mat.list, function(x) L.inf * (1 - exp(-k.mat * (x - t.0))))  # mean lengths-at-a
ge for mature fish
means <- c(imm.means, mat.means)  # concatenate mean lengths-at-age into one vector
```

We adjust sample sizes-at-age to make the data as realistic as possible. In this case, we attempt to account for gear selectivity and natural mortality. We first create a population of 1000 individuals with a sample sizes-at-age that are similar to what is often seen in fisheries data. We then draw a random sample (in this case, 300 data points) from the 1000 individual population. We set the seed for the random number generator to make our results repeatable: `set.seed(2017)`. However, those that wish to bootstrap this approach (or alter the code for some other purpose) may wish to remove this command. Finally, we plot the data.

```r
## Generate data. Sample sizes for each age are realistic for fisheries data,
## based on gear selectivity and natural mortality.
set.seed(2017)
samp.size <- c(30, 70, 130, 140, 150, 130, 65, 45, 38, 32, 28, 24, 22, 18, 15,
    12, 10, 8, 7, 6, 5, 5, 4, 3, 3)  #set sample sizes for each age, based on a population of 1000 in
dividuals
mean.samp <- as.data.frame(cbind(means, samp.size))  #make data frame of mean lengths-at-age and samp
le sizes
mlen <- rep(mean.samp[, 1], mean.samp[, 2])  #repeat each mean 'sample size' number of times
a <- (1:25)  #vector of ages
ages <- rep(a, mean.samp[, 2])  #repeat each age 'sample size' number of times
lengths <- sapply(mlen, function(x) rnorm(1, mean = x, sd = x/prec))  #generate random normal length
data using means & precision
Data <- as.data.frame(cbind(ages, lengths))  #bind vectors into age and length matrix, convert to dat
a frame
Data <- Data[sample(nrow(Data), size = 300, replace = F), ]  #draw a random sample from the populatio
n -- sample size can be adjusted
plot(Data$ages, Data$lengths, ylab = "Length (mm)", xlab = "Age (yr)")  ## plot length-at-age
```



Likelihood function

Next, we specify the likelihood function. We estimate four parameters within the likelihood function itself: h (h1), l_0, g, and the standard deviation in length-at-age (sighat). We provide conversions for the von Bertalanffy (mature) growth parameters, and we define immature and mature ages and lengths based on the age-at-maturity T (mat.age), which we will estimate using a profile likelihood approach (see Optimization -> Profiling for T). In order to improve fit quality, we also include marginal likelihoods for h and l_0. These likelihoods are analogous to prior probability distributions in a Bayesian framework, and they help to ensure that the model converges on realistic parameter estimates. If desired, one can remove these likelihoods from the function.

```
## Store Lester model likelihood function as 'Lester.func' excluding
## age-at-maturity parameter. Optional: include marginal likelihoods for
## immature growth slope and intercept

Lester.func = function(parms) {
    # list parameters
    l0 = parms[1]
    h1 = parms[2]
    g = inv.logit(parms[3])
    sighat = sqrt(parms[4])

    age.i = age[age <= mat.age]  ## define immature ages
    len.i = len[age <= mat.age]  ## define immature lengths
    age.m = age[age > mat.age]  ## define mature ages
    len.m = len[age > mat.age]  ## define mature lengths

    ## Lester model equations
    t1 = -l0/h1
    Linf = 3 * h1/g
    k = log(1 + g/3)
    t0 = mat.age + suppressWarnings(log(1 - (g * (mat.age - t1)/3)))/log(1 +
        g/3)
    mn.i = l0 + h1 * age.i  # immature growth
    mn.m = Linf * (1 - exp(-k * (age.m - t0)))  # mature growth

    ## Likelihoods
    l0.lik = dnorm(l0, mean = l0est, sd = 25, log = T)  #optional (distribution can be adjusted if ne
eded)
    h1.lik = dnorm(h1, mean = h1est, sd = 5, log = T)  #optional (distribution can be adjusted if nee
ded)
    L.i = dnorm(len.i, mean = mn.i, sd = sighat, log = T)  # immature likelihood
    L.m = dnorm(len.m, mean = mn.m, sd = sighat, log = T)  # mature likelihood
    return(sum(c(L.i, L.m, l0.lik, h1.lik)))
}
```

Optimization

To fit the model, we first assign our age and length data vectors to ‘age’ and ‘len’, which are specified in the likelihood function. We then fit a linear model to the first few ages of data, and we use the slope and y-intercept estimates from that linear fit to inform our marginal likelihoods for h and l_0, respectively. This process generally improves full-model convergence without leveraging information outside of the data. That being said, both this linear model fit and the inclusion of the marginal likelihoods for h and l_0 in the likelihood function are optional.

```
## Assign ages as 'age' and lengths as 'len' for likelihood function
age = Data$ages
len = Data$lengths

## OPTIONAL: Use linear model fit to first few years of growth to inform
## marginal likelihoods for immature growth slope & intercept
immdata <- Data[which(age <= (min(age) + 3)), ]  #choose data within first four ages -- number of age
s can be changed
immout <- lm(lengths ~ ages, data = immdata)  #linear regression on 'immature' data
l0est <- immout$coefficients[[1]]  # store intercept estimate, used for prior likelihood
h1est <- immout$coefficients[[2]]  # store slope estimate, used for prior likelihood
```

Next, we provide starting values for each parameter. We use the parameter estimates from the linear model fit as starting values for l_0 and h, and we choose values for g and the error term.

```
## List starting values for each parameter
l0 = l0est  # early growth intercept (if you skipped Step 4, you should put a number here)
h1 = h1est  # early growth slope (if you skipped Step 4, you should put a number here)
g = 0.2  # cost to somatic growth of maturity
sighat = 25  # standard deviation
parms = c(l0, h1, logit(g), sighat^2)  #compile parameters
```

Profiling for T

The profiling procedure uses an iterative approach to find the most likely value for T (and the remaining four parameters). In essence, we (1) fix T at some value, (2) fit the model to the data (estimating the remaining parameters), and (3) store the results, including the full-model likelihoods. We then repeat this procedure for a large number of potential values for T. In this case, our vector of potential T values ranges from ages 2-20 yr in 0.025 yr increments. We loop through this vector, using `optim()` to optimize the likelihood function for each potential value of T. We store our results for each iteration in a matrix, and we include an `if()` statement to ignore results from fits for which the model did not converge.

```
## Create a vector of potential values for age-at-maturity and a matrix for
## storing parameter estimates
Mat.age = seq(2, 20, by = 0.025)  #  range of mat.age values for profile likelihood calculation -- ad
just as needed
lik <- l0 <- h1 <- g <- var <- rep(NA, length(Mat.age))  # create empty vectors for parameters
mat.age.Lik = cbind(Mat.age, lik, l0, h1, g, var)  # create matrix for storing parameter estimates



## Optimize likelihood function for each potential age-at-maturity value and
## store parameter estimates
for (j in 1:length(Mat.age)) {
    mat.age = Mat.age[j]  # fix age-at-maturity at a given value
    L.out = try(optim(par = parms, fn = Lester.func, control = list(fnscale = -1,
        reltol = 1e-08)), silent = T)  # optimize likelihood function
    check <- is.numeric(L.out[[1]])  # check to see if model converged

    ## store values only if model converged
    if (check[[1]] == "TRUE") {

        # Store parameter values (back-transform g)
        mat.age.Lik[j, 2] <- L.out$value
        mat.age.Lik[j, 3] <- L.out$par[[1]]
        mat.age.Lik[j, 4] <- L.out$par[[2]]
        mat.age.Lik[j, 5] <- inv.logit(L.out$par[[3]])
        mat.age.Lik[j, 6] <- L.out$par[[4]]
    }
}
```

Compiling and visualizing results

To view our results, we first convert our output matrix from the profiling procedure to a data frame for easier manipulation, and we remove NAs (failed model fits). We then simply find the maximum full-model likelihood value within our data frame. Our parameter estimates correspond with this maximum likelihood value.

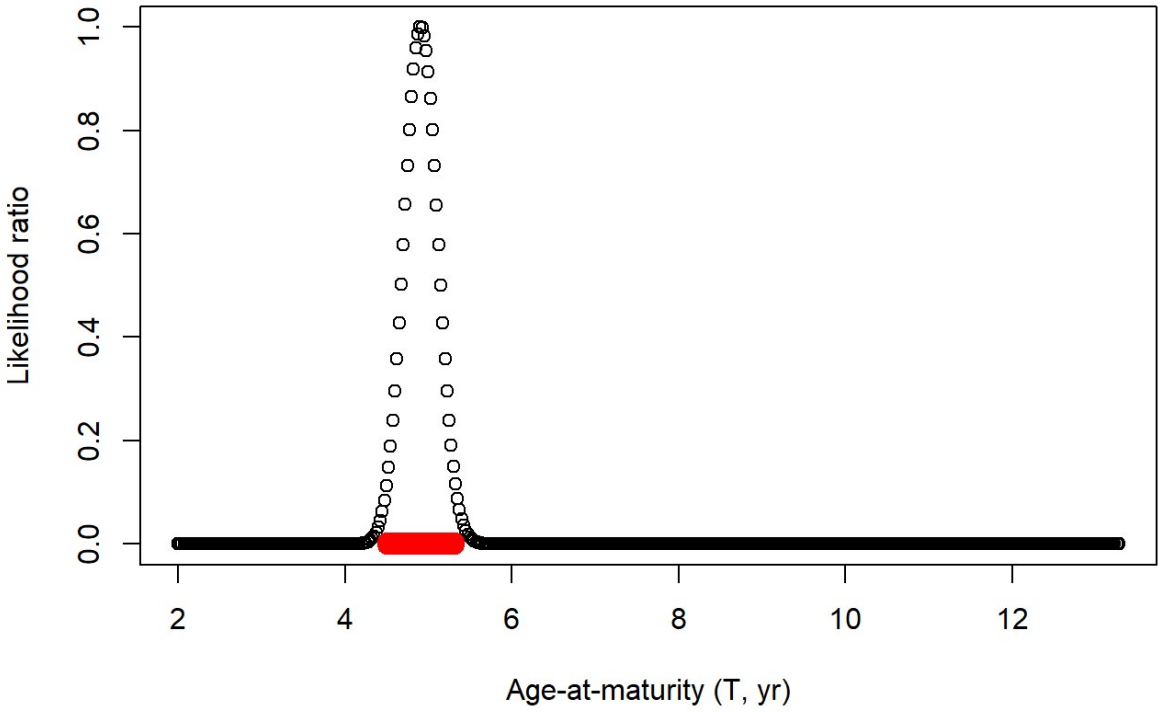
```
## Compile results
mat.age.Lik <- as.data.frame(mat.age.Lik)  # convert to data frame for easier referencing
mat.age.Lik <- mat.age.Lik[which(mat.age.Lik$lik != "NA"), ]  # remove failed runs
mle = max(mat.age.Lik$lik)  # find maximum likelihood
MLE <- mat.age.Lik[which(mat.age.Lik$lik == mle), ]  # maximum likelihood estimates for all parameter
s
MLE  # print maximum likelihood estimates
##     Mat.age      lik       l0      h1         g      var
```

```
## 117      4.9 -1461.68 91.37931 53.1966 0.2745834 955.5614
```

In this case, our estimate for T is 4.9 yr, which is quite close to the simulated 'true' value of 5 yr. The other parameter estimates are also close to the simulated values, and will approach those simulated values as precision and sample size increase. For instance, if one increases the precision in length-at-age to 25, the estimate for l_0 increases to approximately 96 mm (simulated value = 100 mm).

A handy way to examine model fit quality is to plot the likelihood profile for T. In general, the presence of one distinct likelihood peak and a relatively narrow confidence interval indicate a good fit. However, there may be some cases for which the model fits well even when these criteria are not met (e.g., large variability in age-at-maturity in a population, leading to a wide confidence interval around T; multiple ages-at-maturity across cohorts due to plastic or evolutionary life history changes, leading to multiple likelihood peaks; etc.).

```
## Plot likelihood profile
rlike = exp(mat.age.Lik$lik - mle)
plot(mat.age.Lik$Mat.age, rlike, xlab = "Age-at-maturity (T, yr)", ylab = "Likelihood ratio")


## Confidence interval in terms of chi-squared (~ 95% CI)
ndx1 = which(mat.age.Lik$lik > (mle - 1.92))  # change '1.92' to 0.228 for 50% CI, 1.36 for 90% CI
points(mat.age.Lik$Mat.age[ndx1], rep(0, length(ndx1)), col = "red", lwd = 6)
CI = c(min(mat.age.Lik$Mat.age[ndx1]), max(mat.age.Lik$Mat.age[ndx1]))
CI  # print confidence interval
## [1] 4.525 5.300
```

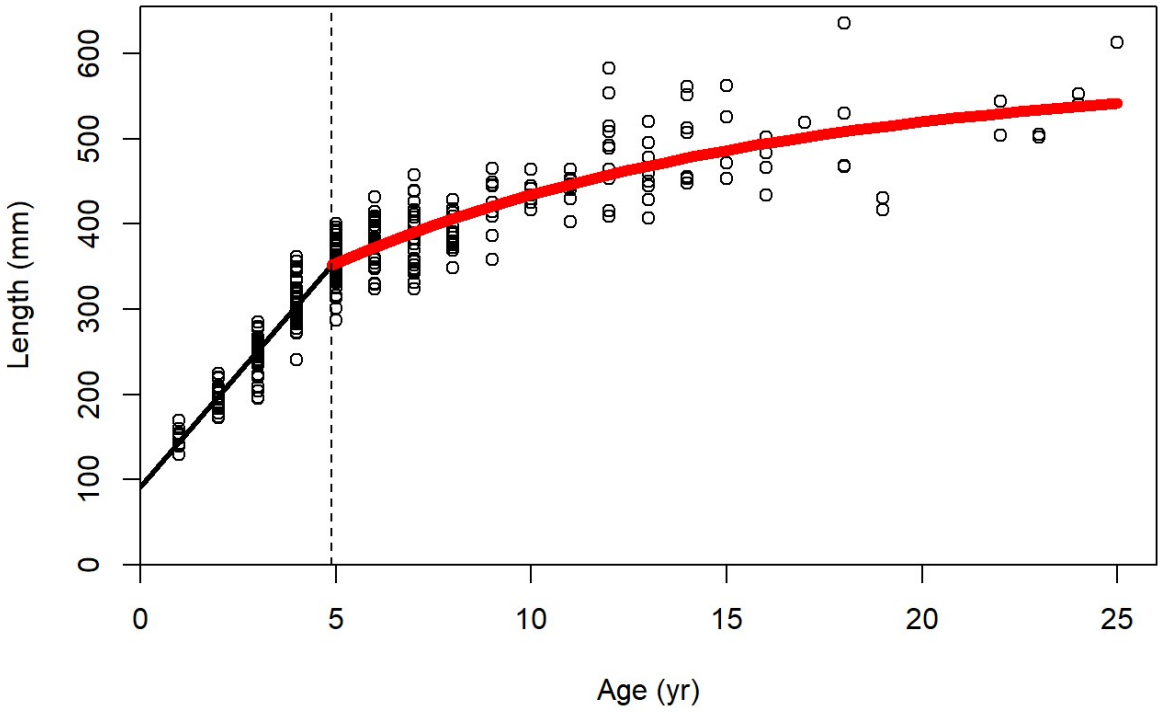


This figure shows the likelihood ratio profile for T across all of the values for which the model converged. The peak of this profile corresponds to the maximum likelihood estimate for T. The red line beneath the peak is the 95% confidence interval.

Finally, we can examine the fit by plotting the curves onto the data:

```
## Plot biphasic growth curves onto data
plot(age, len, xlab = "Age (yr)", ylab = "Length (mm)", xlim = c(0, max(age) +
    1), ylim = c(0, max(len) + 20), xaxs = "i", yaxs = "i")
g. = MLE[[5]]
h1. = MLE[[4]]
```

```
mT = MLE[[1]]
l0. = MLE[[3]]
t1. = -l0./h1.
Linf = 3 * h1./g.
k = log(1 + g./3)
t0 = mT + log(1 - (g. * (mT - t1.)/3))/log(1 + g./3)
segments(0, l0., x1 = mT, y1 = l0. + h1. * mT, lwd = 3)
matX = seq(mT, max(age), length.out = 25)
matY = Linf * (1 - exp(-k * (matX - t0)))
lines(matX, matY, col = "red", type = "l", lwd = 6, lty = 1)
abline(v = mT, lty = 2)
```



In this figure, the black line is the immature growth phase, and the red line is the mature growth phase. Growth shifts from the immature to the mature phase at T, which we've estimated to be 4.9 yr (dashed vertical line; 'true' value = 5 yr).

References

Honsey, A.E., Staples, D.F. & Venturelli, P.A. (2016). Accurate estimates of age at maturity from the growth trajectories of fishes and other ectotherms. *Ecological Applications*, **27**, 182–192.

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S5:
Fitting the Lester biphasic growth model without maturity data using a Bayesian MCMC approach

Kyle Wilson[1] *and Andrew Honsey*[2]
[1] *The University of Calgary*
[2] *University of Minnesota*

October 5, 2017

This appendix is in support of Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

The following code is an example application of the Lester biphasic growth model (Lester, Shuter & Abrams 2004) to length-at-age data in the absence of any information on maturity (e.g., maturity data, *a priori* estimates of age-at-maturity, etc.). In this case, the age-at-maturity parameter T is estimated simultaneously with the other model parameters using Bayesian MCMC via JAGS. First, we load the required libraries. Note that these packages, along with the JAGS program (http://mcmc-jags.sourceforge.net/), must be installed for the code to run.

```
library(runjags)
library(rjags)
## Loading required package: coda
## Linked to JAGS 4.2.0
## Loaded modules: basemod,bugs
library(stats4)
library(coda)
```

Next, we define a few functions that will be used later.

```
Corner_text <- function(text, location="topright") #function to write text to the corner of plots
{
  legend(location,legend=text, bty ="n", pch=NA)
}

get_beta <- function(mean,cv) #function that returns the alpha and beta shape parameters of a beta di
stribution, based on the mean and variation of a given beta distribution
{
  sd <- mean*cv
  alpha <- -((mean*(mean^2+sd^2-mean))/sd^2)
  beta <- alpha/mean-alpha
  return(list(alpha=alpha,beta=beta))
}

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}

rngList <- function(x,y){ #this function creates randomly 'jittered' starting values for each MCMC ch
ain
  lis <- lapply(x, lapply, length) #get the lower order dimensions of the list x
  names(lis) <- lapply(x, length) #get the names of those dimensions of the list x
  l_el <- length(names(lis)) #get the maximum number elements of the highest order dimensions of the
list x
```

```
  for(i in 1:(l_el-2)) #loop through those dimensions which need to be 'jittered'
  {
    x[[i]] <- x[[i]]*(1+runif(1,-0.15,0.15)) #jitter values of the list by +/- 15%
  }
  x[[l_el]] <- round(runif(1,1,100000),0) #have a random RNG seed for the MCMC chain
  return(x)
}
```

Data generation

We simulate length-at-age data for a single population for the Lester growth model (Lester, Shuter & Abrams (2004)). The age range of the fishes is read in as a vector of integers. True somatic growth rate can be any positive number and represents the length (in mm) accumulated per year in the late-stage juvenile phase. The variable t_1 represents the hypothetical age at length 0 (i.e., the *x*-intercept for the juvenile phase). We use a reasonable value for the coefficient of variation in length-at-age (cv) of 15%, a typical observation among most fishes (see review in Lorenzen (2016)). The variable T (or Tmat in the R code below) represents the age-at-maturity for the population. The parameter g represents the proportion of energy in the adult phase allocated to reproduction per unit time (in this case, per year). Note that g must be positive and has an intrinsic maximum such that:

$$g \sim \{0, 3/(T - t_1)\}.$$

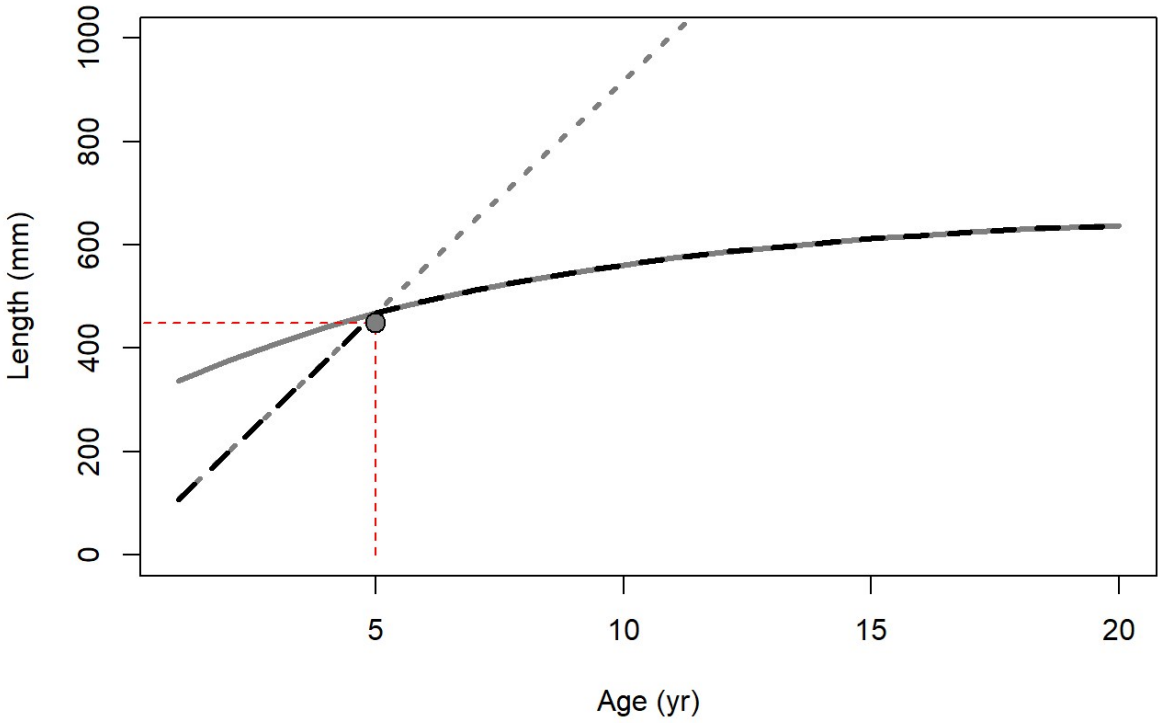
In this case, we set the 'true' g value at 70% of the intrinsic maximum. We also adjust sample sizes-at-age in an attempt to account for factors such as gear selectivity and natural mortality; however, we could simulate this using a multinomial process as well (see Appendix S4).

```
nPop <- 1  # number of populations
N <- c(3, 5, 4, 8, 10, 12, 14, 10, 9, 6, 14, 8, 7, 10, 9, 6, 2, 3, 1, 3)  # maximum number of samples
 for each age group

ages <- 1:20  #create an integer sequence of ages

Tmat <- 5  # age at maturity

h <- 90  # somatic growth in millimeters per year

t1 <- -0.2  #age when length=0 for the juvenile phase

g <- 0.7 * (3/(Tmat - t1))  # proportion of energy in adult phase allocated to reproduction per year

cv <- 0.15  # coefficient of variation in length-at-age
```

Next, we convert the Lester parameters to von Bertalanffy parameters, which describe the asymptotic growth of the adult phase. We then calculate the 'true' length-at-age for both phases of growth using the parameters specified above. We use an if-else statement to select which 'true' length-at-age value applies for each age, given the *a priori* estimate of age-at-maturity. We then generate a plot of the lifetime growth trajectory, including the entire trajectories for both phases and the point at which growth transitions from the first to the second phase (i.e., age-at-maturity).

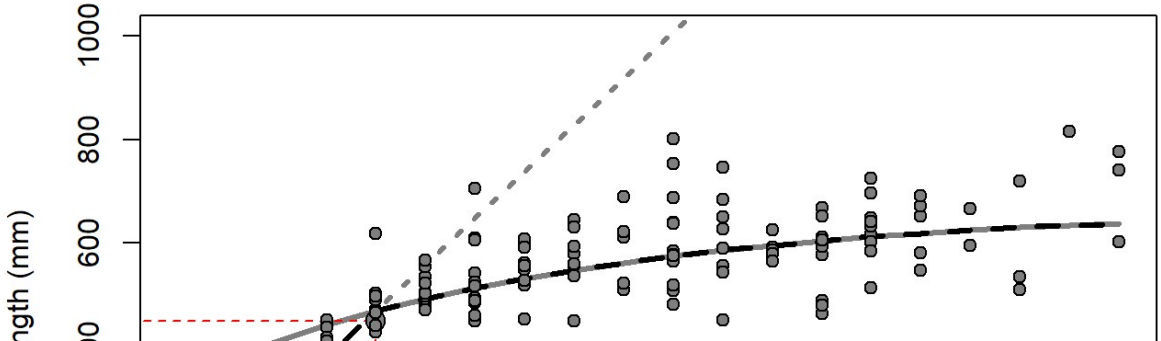
```
linf <- 3 * h/g  # conversion for the VBGF L-infinity
vbk <- log(1 + g/3)  # conversion for the VBGF parameter kappa
t0 <- Tmat + log(1 - g * (Tmat - t1)/3)/log(1 + g/3)  #conversion for the VBGF parameter t0
true.par <- list(h = h, T.mat = Tmat, t1 = t1, g = g, lengthCV = cv)  # save 'true' parameters to a l
ist


lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages <= Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a fish
 is allocating surplus energy

plot(ages, lena_phase1, ylab = "Length (mm)", xlab = "Age (yr)", lty = 3, type = "l",
    col = "grey50", lwd = 3, ylim = c(0, 1000))
lines(ages, lena_phase2, col = "grey50", lwd = 3)
lines(ages, biphasic, lty = 2, lwd = 3)
segments(x0 = Tmat, x1 = Tmat, y0 = 0, y1 = h * Tmat + t1, col = "red", lty = 2)  #plot where maturit
y occurs
segments(x0 = 0, x1 = Tmat, y0 = h * Tmat + t1, y1 = h * Tmat + t1, col = "red",
```

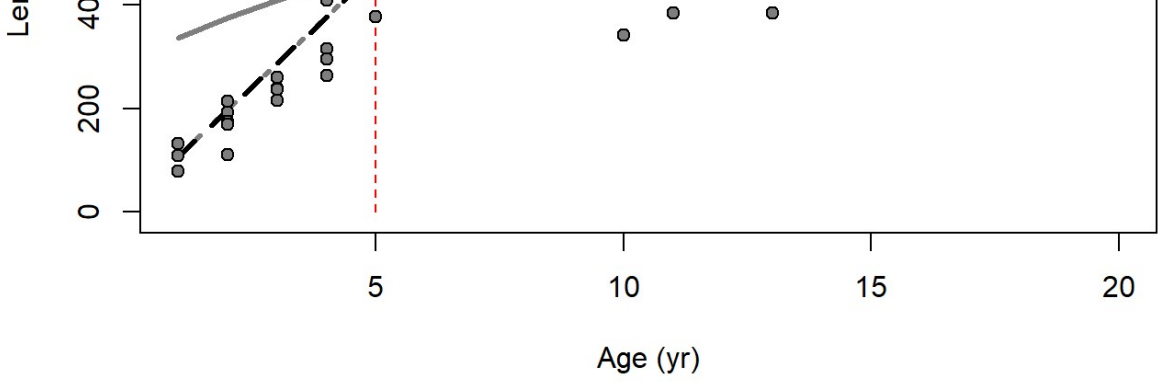
```
    lty = 2)  #plot where maturity occurs
points(Tmat, h * Tmat + t1, pch = 21, bg = "grey50", cex = 1.5)
```



We then simulate the data using `rnorm()` and the coefficient of variation in length-at-age specified above.

```
data <- NULL
for (j in 1:max(ages)) {
    lengths <- rnorm(N[j], biphasic[j], biphasic[j] * cv)  # simulate noisy length-at-age data for ea
ch age-class
    length_age <- cbind(rep(j, N[j]), lengths)
    data <- rbind(data, length_age)
}
colnames(data) <- c("Age", "Length")  # rename columns in data
data <- as.data.frame(data)  # convert to data frame

plot(ages, lena_phase1, ylab = "Length (mm)", xlab = "Age (yr)", lty = 3, type = "l",
    col = "grey50", lwd = 3, ylim = c(0, 1000))
lines(ages, lena_phase2, col = "grey50", lwd = 3)
lines(ages, biphasic, lty = 2, lwd = 3)
segments(x0 = Tmat, x1 = Tmat, y0 = 0, y1 = h * Tmat + t1, col = "red", lty = 2)  #plot where maturit
y occurs
segments(x0 = 0, x1 = Tmat, y0 = h * Tmat + t1, y1 = h * Tmat + t1, col = "red",
    lty = 2)  #plot where maturity occurs
points(Tmat, h * Tmat + t1, pch = 21, bg = "grey50", cex = 1.5)
points(data$Age, data$Length, pch = 21, bg = "grey50")  # plot the simulated data
```





Bayesian estimation

The following model code is in the JAGS language (Plummer 2017). The model loops through each data point and determines the contribution of the predicted length for each fish to the posterior, which is the sum of the log-likelihood and the log-prior. The predicted growth follows the analytical model from the equations in Lester, Shuter & Abrams (2004). There is an if-else statement that determines if the predicted length-at-age of data point *i* comes from the juvenile phase or the adult phase. Modifications to this code require JAGS syntax and not R syntax.

```
model <- "model {

## likelihood

#loop through all data points
for(i in 1:Nfish) {
length[i] ~ dnorm(pred[i],1/(pred[i]*length.cv)^2)T(0,) # observed length-at-age should be normally d
istributed around predictions

juv[i] <- h*(age[i]-t1) # predicted growth for fish i for the juvenile phase

adult[i] <- (3*h/g)*(1-exp(-(log(1+g/3))*(age[i]-(T.mat+log(1-g*(T.mat-t1)/3)/log(1+g/3))))) # predic
ted growth for fish [i] for the adult phase

pred[i] <- ifelse(age[i]<=T.mat,juv[i],adult[i]) # does the age of fish i exceed the maturity predict
ed for its population?

} # end the calculation of the likelihood



## prior distributions

T.mat ~ dunif(0,max(age))
h ~ dnorm(30,1e-3)T(0,)
t1 ~ dnorm(0,1)
g ~ dnorm(0.1,0.001)T(0,3/(T.mat-t1))
length.cv ~ dgamma(0.01,0.01)
}"
```

We then compile the data into a list for JAGS. In addition, we provide starting values for each parameter within each chain. In this case, we use a random number generator to 'jitter' the starting values for each chain. We then compile the starting values for each chain into a list (in effect, creating a 'list of lists').

```
# compile data into a list for JAGS
JAGSdata <- list(Nfish = length(data$Age), age = data$Age, length = data$Length)


inits1 <- list(h = h, T.mat = Tmat, g = g, t1 = t1, length.cv = cv, .RNG.name = "base::Wichmann-Hill"
,
    .RNG.seed = 735)

# initial estimates of each parameter can be provided. If not provided, JAGS
# will automatically sample a random number from the prior distribution. RNG
# is a random number generator for each chain
```

```
inits2 <- inits3 <- inits4 <- inits1

inits2 <- rngList(inits2, inits1)  # jitter chain 2, based on values of chain 1
inits3 <- rngList(inits3, inits1)  # jitter chain 3, based on values of chain 1
inits4 <- rngList(inits4, inits1)  # jitter chain 4, based on values of chain 1

inits <- list(inits1, inits2, inits3, inits4)  # compile all initial values into one list
```

Our last step before fitting the model is to provide some parameters for the JAGS MCMC algorithm. We first create the stochastic nodes to be monitored. We then store values for the thinning rate, the length of the burn-in (or warmup) period, and the length of the adaptation period. In this case, we specify these values based on the total number of posterior draws desired for each chain.

```
mon_names <- c(names(inits3)[-c(length(inits3), length(inits3) - 1)])  # create the stochastic nodes
to be monitored

Nsamp <- 1000  # how many posterior samples does each chain need to get, after thinning and burin-in
and adaptation?
thin_rt <- 20  # thinning rate
burnins <- 0.75 * round(Nsamp * thin_rt, 0)  # length of burn-in, based on the number of total poster
ior draws
adaptin <- round(0.4 * burnins, 0)  # length of adaptation
```

We use the `run.jags()` command (Denwood 2016) to fit the model, and we reference the JAGS parameters as specified above. We use the `rjags` method in this case – for larger datasets or more complicated (e.g., hierarchical) models, the `rjparallel` method may be preferred. We also calculate the computation time using `proc.time()`. Finally, we call the summary of the model fit. Note that results will vary slightly due to MCMC error.

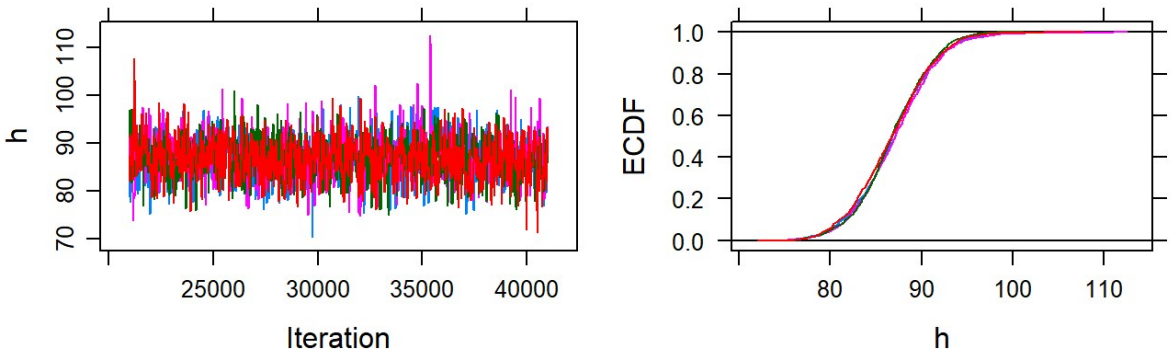
```
a <- proc.time()
results <- run.jags(model = model, monitor = mon_names, data = JAGSdata, n.chains = 4,
    method = "rjags", inits = inits, plots = F, silent.jag = F, modules = c("bugs",
        "glm", "dic"), sample = Nsamp, adapt = adaptin, burnin = burnins, thin = thin_rt,
    summarise = F)
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 6000 iterations...
## Burning in the model for 15000 iterations...
## Running the model for 20000 iterations...
## Simulation complete
## Finished running the simulation
b <- (proc.time() - a)
print(b[3]/60)  # computation time in minutes
## elapsed
##  5.8375

sum_results <- summary(results)
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 5 variables....
sum_results
##               Lower95     Median     Upper95       Mean         SD
## h          78.2704044 86.8639919 94.99308316 86.9212419 4.30658300
## T.mat       4.7854871  5.4669492  6.19101898  5.4745040 0.36118938
## g           0.3368223  0.3854157  0.43763536  0.3861670 0.02593727
## t1         -0.4709363 -0.1916538  0.02721781 -0.2047833 0.12910419
## length.cv   0.1282136  0.1445778  0.16334867  0.1451885 0.00893106
##                  Mode        MCerr MC%ofSD SSeff       AC.200     psrf
## h          86.9204472 0.0989670536     2.3  1894 -0.018079940 1.001956
## T.mat       5.4458714 0.0073556950     2.0  2411  0.002893319 1.000853
## g           0.3847591 0.0005409926     2.1  2299 -0.043920991 1.002376
## t1         -0.1823072 0.0026908676     2.1  2302 -0.015524842 1.001408
## length.cv   0.1435777 0.0001449633     1.6  3796 -0.007107004 1.000261
```

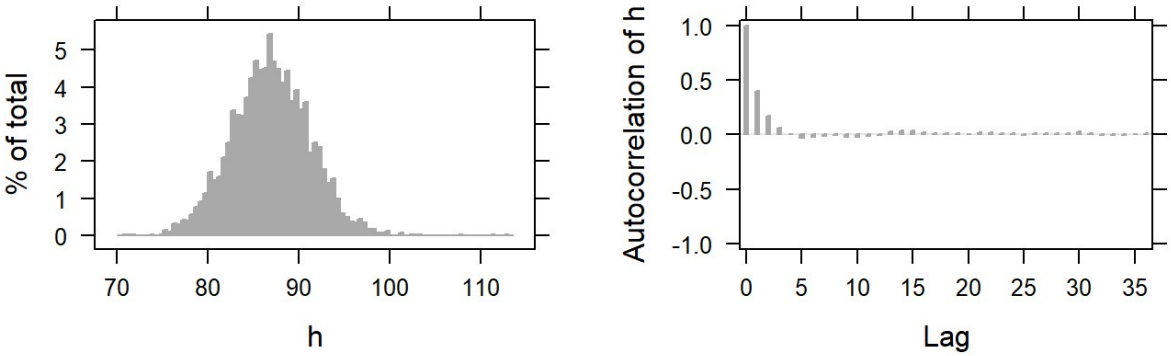
Saving and visualizing results

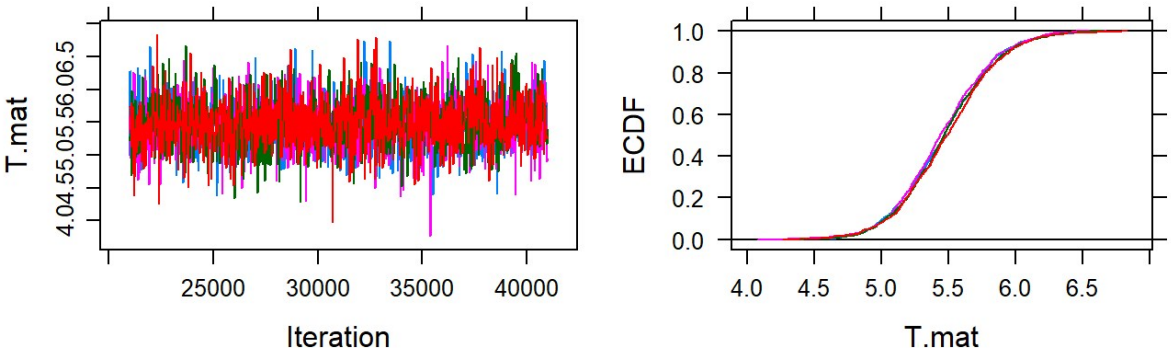
We will now use `plot(results)` to show some diagnostic plots that evaluate whether the posterior has converged on a stable

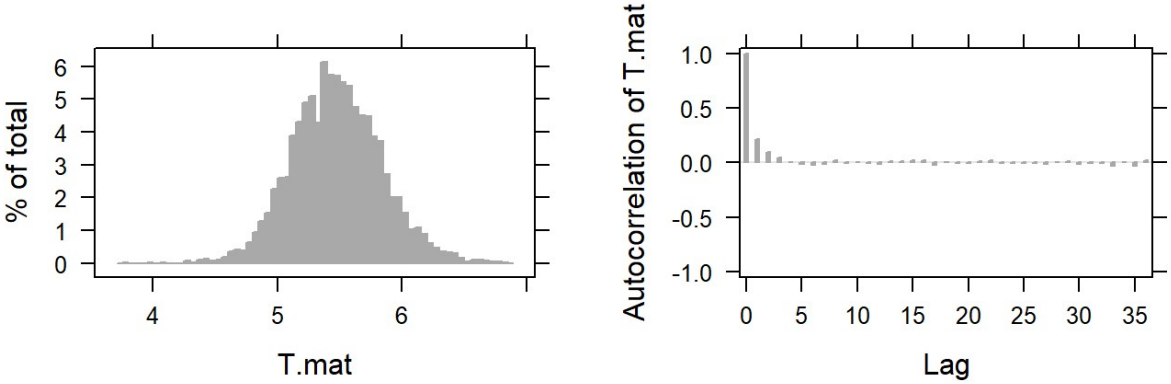
mode using the package `coda()` (Plummer *et al.* 2006).

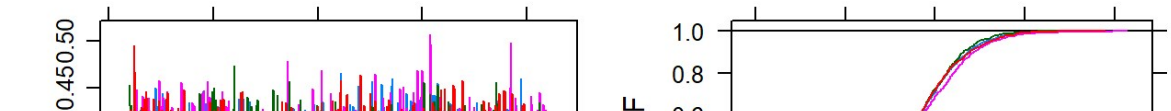
```
plot(results)
## Generating summary statistics and plots (these will NOT be saved
## for reuse)...
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 5 variables....
```

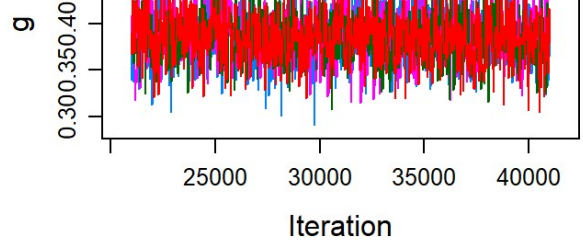


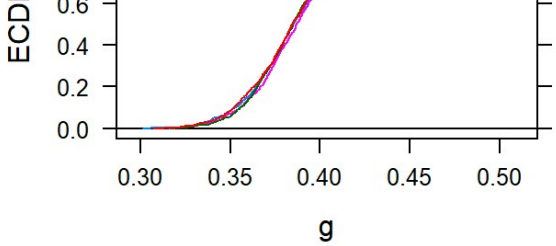


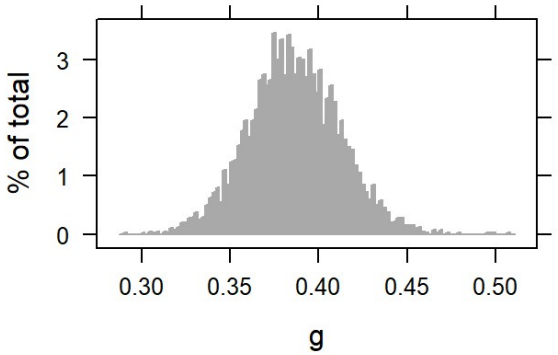


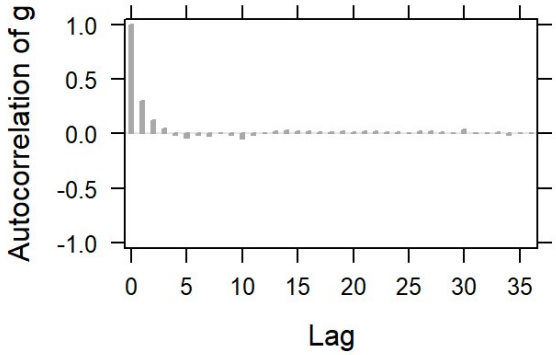


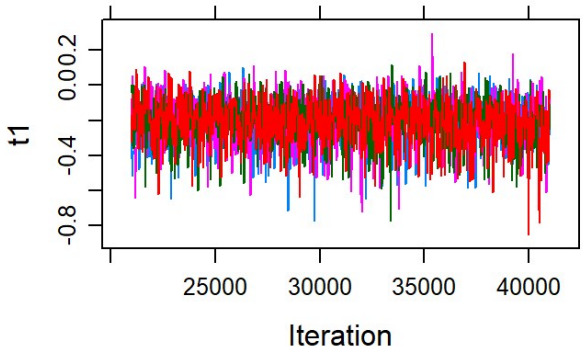


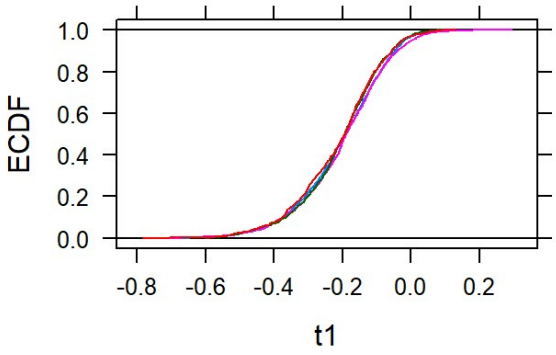

g
0.30 0.35 0.40
25000 30000 35000 40000
Iteration

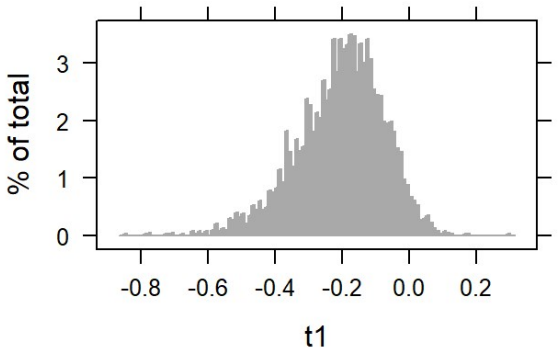

ECDF
0.0 0.2 0.4 0.6
0.30 0.35 0.40 0.45 0.50
g

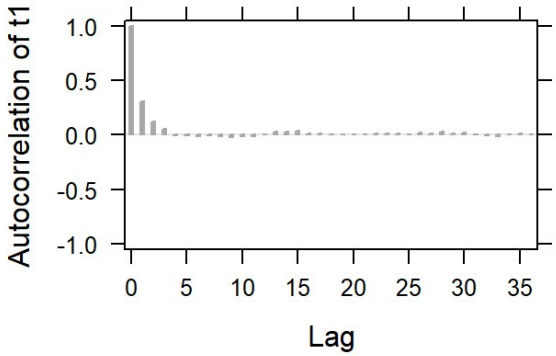

% of total
0 1 2 3
0.30 0.35 0.40 0.45 0.50
g

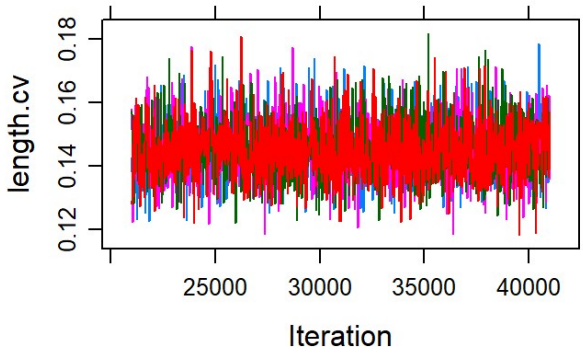

Autocorrelation of g
-1.0 -0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag

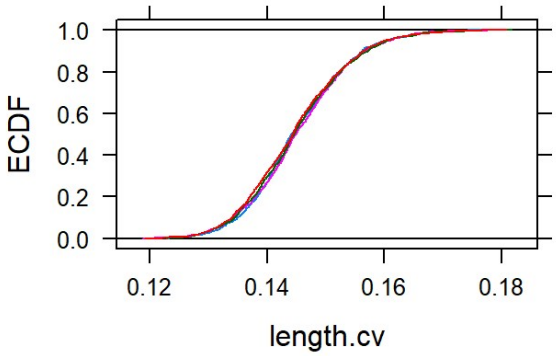

t1
-0.8 -0.4 0.0 0.2
25000 30000 35000 40000
Iteration

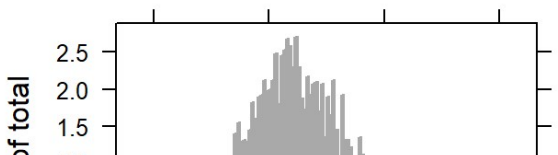

ECDF
0.0 0.2 0.4 0.6 0.8 1.0
-0.8 -0.6 -0.4 -0.2 0.0 0.2
t1

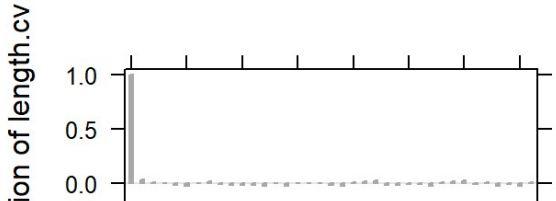

% of total
0 1 2 3
-0.8 -0.6 -0.4 -0.2 0.0 0.2
t1

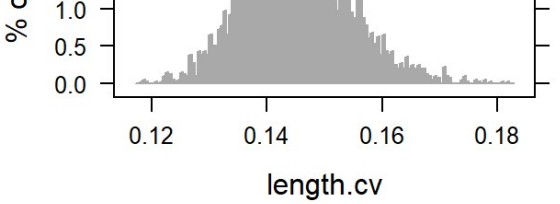

Autocorrelation of t1
-1.0 -0.5 0.0 0.5 1.0
0 5 10 15 20 25 30 35
Lag

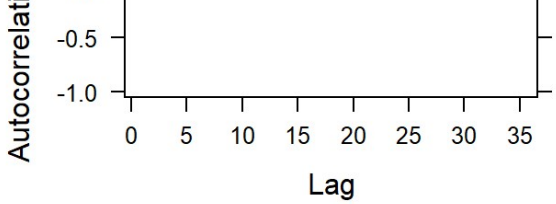

length.cv
0.12 0.14 0.16 0.18
25000 30000 35000 40000
Iteration

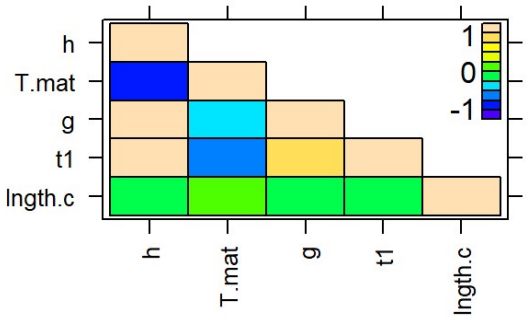

ECDF
0.0 0.2 0.4 0.6 0.8 1.0
0.12 0.14 0.16 0.18
length.cv


of total
0.0 0.5 1.0 1.5 2.0 2.5


ion of length.cv
0.0 0.5 1.0

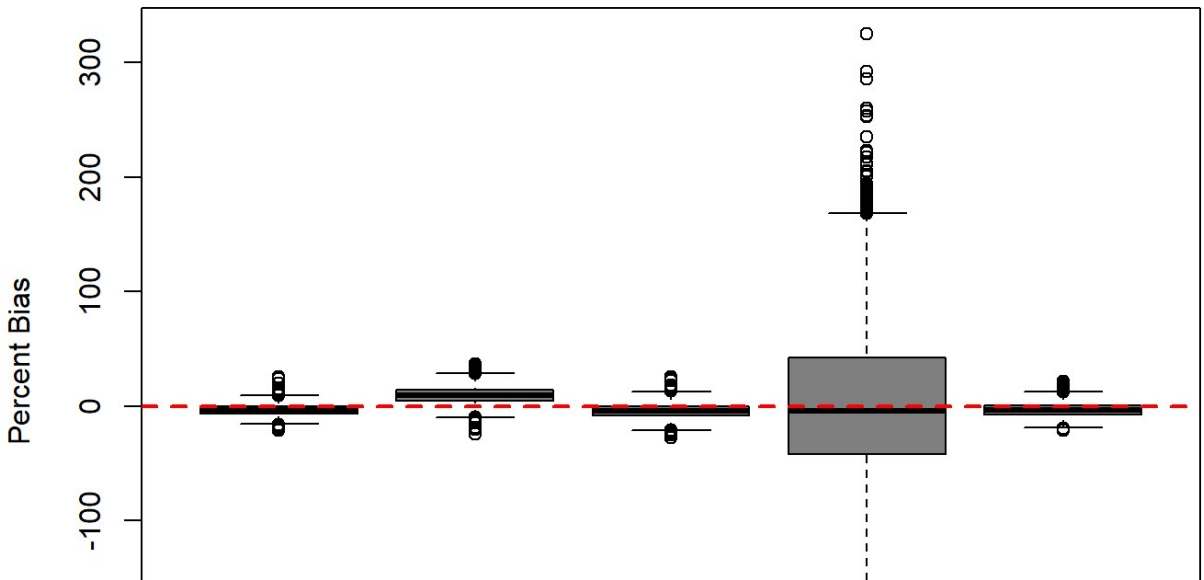


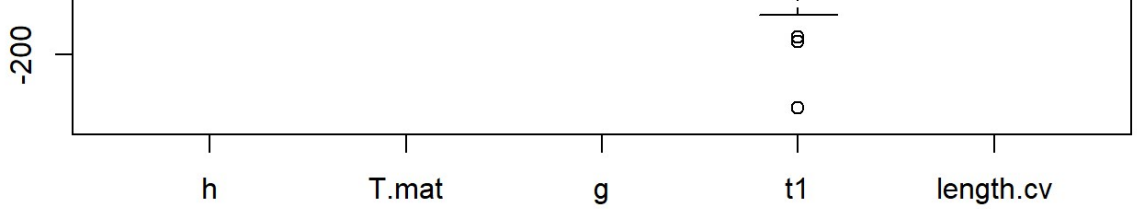




We then show a percent bias plot. Our results suggest that parameters for this one randomized dataset were estimated well, apart from the parameter t_1, which was estimated with noise. We could repeat this trial many times to get a general idea of how well the model recovers true life history parameters of interest (we do this in Appendix S7).

```
par(mfrow = c(1, 1))
par(mar = c(4, 4, 1, 1))
TheRes <- as.matrix(as.mcmc.list(results), chain = F)
# TheRes <- read.table(ResultsFile,header=TRUE) # read into
true.par <- as.vector(c(h, Tmat, g, t1, cv))
dat <- t(apply(TheRes, 1, FUN = function(x) {
    (x - true.par)/true.par * 100
}))
boxplot(dat, ylab = "Percent Bias", xlab = "Lester Model Parameters", col = "grey50")
abline(h = 0, lty = 2, col = "red", lwd = 2)
```

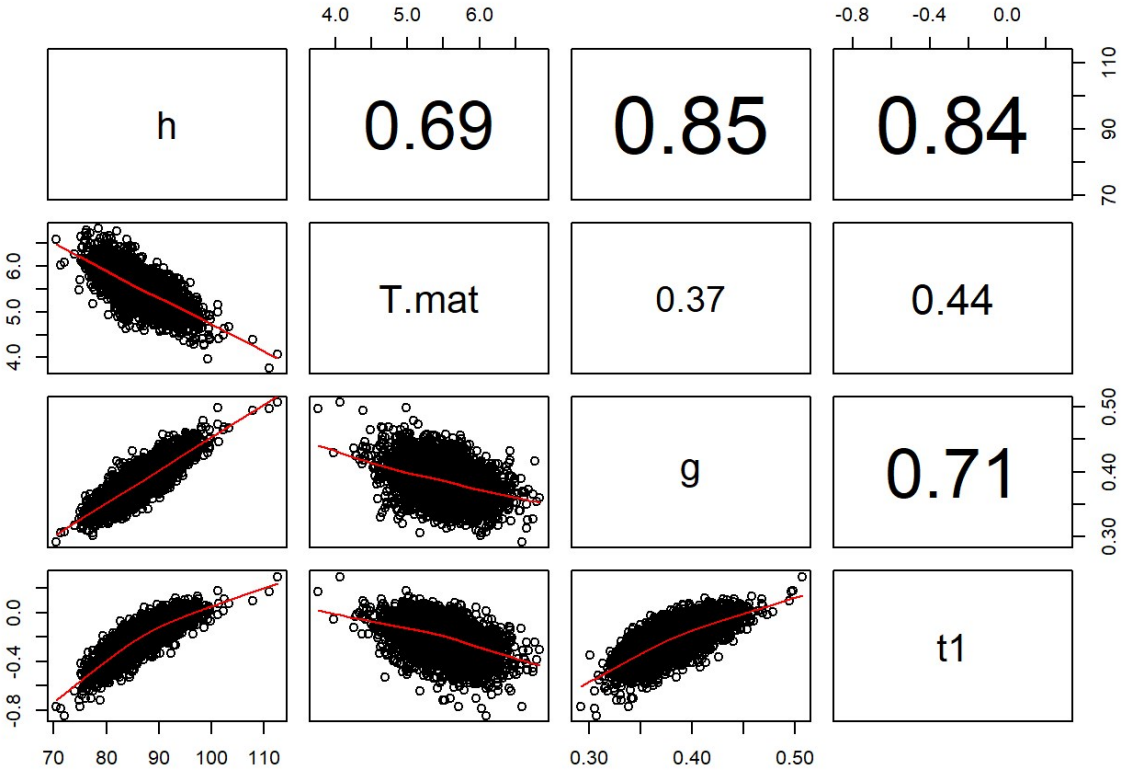




Lester Model Parameters

Next, we will generate a 'pairs plot' to see how correlated the parameters are during the MCMC estimation. One may wish to use a multivariate normal distribution in the JAGS model above to allow for the correlations to be incorporated into the MCMC sampling (Helser & Lai 2004).

```
pairs(TheRes[, -5], lower.panel = panel.smooth, upper.panel = panel.cor)
```

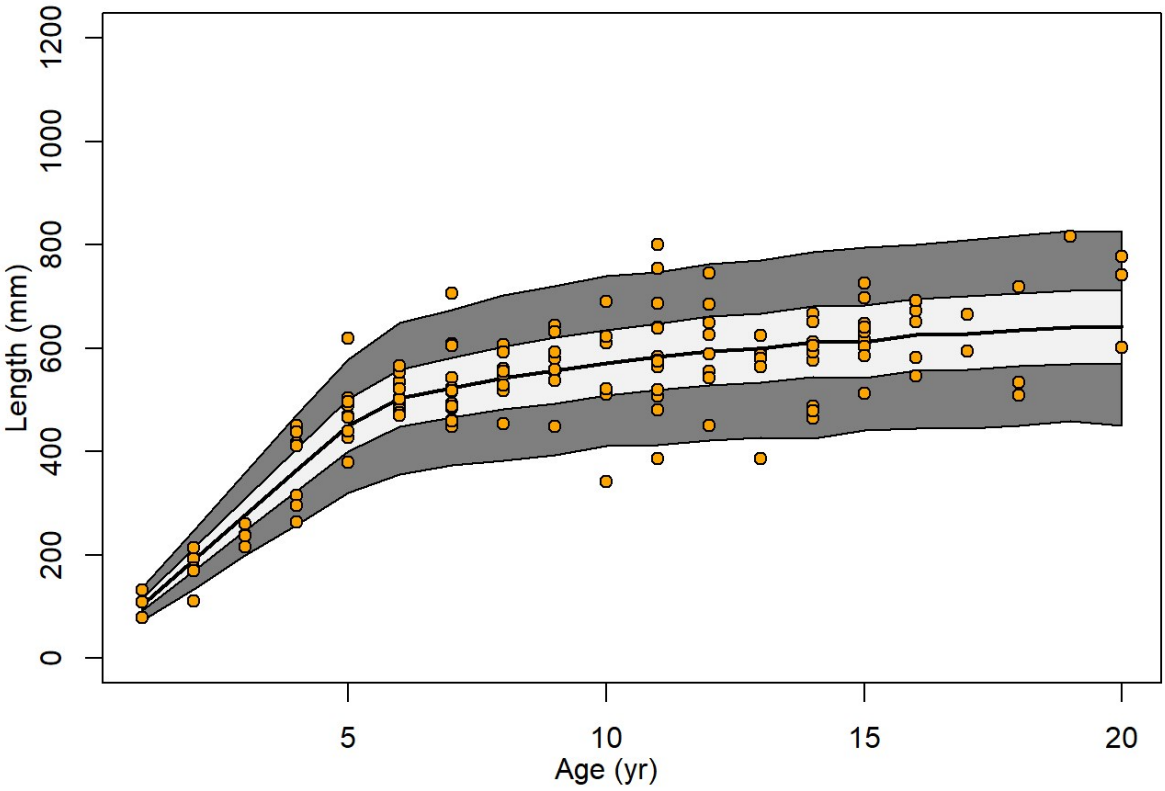


Posterior predictive check

Next, we conduct a posterior predictive check (Gelman *et al.* (2013)). We simulate replicated data from our fitted JAGS model (with associated uncertainty for each estimated parameter from the model) and compare the distribution of our new *simulated* data to the *observed* data. In this case, the *observed* data is our original simulated data used in the model fitting in the `data` object.

```
age_vec <- c(ages, rev(ages))
post_pred <- matrix(NA, nrow = nrow(TheRes), ncol = length(ages))
# Above code creates an empty array to track length-at-age for posterior
# draw j
par(mfrow = c(1, 1))
par(mar = c(5, 4, 1, 1))
layout(matrix(1, nrow = 1, ncol = 1))
for (j in 1:nrow(TheRes)) {
    h_j <- TheRes[j, match("h", colnames(TheRes))]
    g_j <- TheRes[j, match("g", colnames(TheRes))]
    Tm_j <- TheRes[j, match("T.mat", colnames(TheRes))]
    t1_j <- TheRes[j, match("t1", colnames(TheRes))]
    cv_j <- TheRes[j, match("length.cv", colnames(TheRes))]

    linf <- 3 * h_j/g_j  # conversion for the VBGF L-infinity
    vbk <- log(1 + g_j/3)  # conversion for the VBGF parameter kappa
    t0 <- Tm_j + log(1 - g_j * (Tm_j - t1_j)/3)/log(1 + g_j/3)  #conversion for the VBGF parameter t0
```

```
    juv_j <- h_j * (ages - t1_j)
    adult_j <- linf * (1 - exp(-vbk * (ages - t0)))
    pred_j <- ifelse(ages < Tm_j, juv_j, adult_j)
    pred_j[pred_j < 0] <- 0.001
    post_pred[j, ] <- abs(rnorm(length(ages), pred_j, pred_j * cv_j))
}
quants <- t(apply(post_pred[, ], 2, FUN = quantile, probs = c(0.025, 0.225,
    0.5, 0.775, 0.975), na.rm = TRUE))
plot(age_vec, c(quants[, 1], rev(quants[, 5])), type = "l", lwd = 2, col = NA,
    ylab = "", xlab = "", ylim = c(0, 1200))
axis(1, at = median(c(0, max(ages))), "Age (yr)", tick = FALSE, line = 0.9)
axis(2, at = 600, "Length (mm)", tick = FALSE, line = 1)
polygon(age_vec, c(quants[, 1], rev(quants[, 5])), col = "grey50")
polygon(age_vec, c(quants[, 2], rev(quants[, 4])), col = "grey95")
lines(ages, quants[, 3], lwd = 2, col = "black")
points(data$Age, data$Length, pch = 21, bg = "orange")
```



The posterior predictive distribution covers most of the data, and we can see from graphical portrayal that there are no systematic discrepancies between the *observed* and *simulated* data. In general, we might conclude the model is valid and fits well to the data (one could conduct alternative Bayesian goodness-of-fit tests as well).

References

Denwood, M.J. (2016). runjags: An R package providing interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. *Journal of Statistical Software*, **71**, 1–25.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. & Rubin, D. (2013). *Bayesian Data Analysis*, 3rd edn. Chapman and Hall/CRC Press.

Helser, T.E. & Lai, H.-L. (2004). A bayesian hierarchical meta-analysis of fish growth: With an example for north american largemouth bass, micropterus salmoides. *Ecological Modelling*, **178**, 399–416.

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Plummer, M. (2017). JAGS: A program for analysis of bayesian graphical models using gibbs sampling.

Plummer, M., Best, N., Cowles, K. & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for mcmc. *R News*, **6**, 7–11.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S6:
Realistic simulation and comparison of three approaches to fitting biphasic growth models

Kyle L Wilson
The University of Calgary

October 5, 2017

This appendix is in support of Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

Summary description of objectives:

The following script simulates 3 different statistical approaches for fitting the Lester biphasic model where the breakpoint (age-at-maturity) is treated as unknown prior to model estimation (Lester, Shuter & Abrams 2004). We use three different sets of initial parameter values to evaluate the sensitivity of each approach to starting values. We then compare the performance of each approach using percent bias:

$$Bias = ((\theta_i - \hat{\theta}_i)/\theta_i) * 100$$

where $theta_i$ is one of the estimated parameters (e.g., h) for a given approach and set of starting values.

Global functions

We will first define some global functions that will be used below.

```
Corner_text <- function(text, location="topright") #function to write text to the corner of plots
{
  legend(location,legend=text, bty ="n", pch=NA)
}

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) #function to change size of text i
n the pairs plots to match the size of the correlation
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}

biphasicPlot <- function(h=h,g=g,t1=t1,Tmat=Tmat) #function to return the biphasic growth trajectory
for a given set of parameters
{
  LT_ages <- ages
  juv <- h*(LT_ages-t1)
  adult <- (3*h/g)*(1-exp(-(log(1+g/3))*(LT_ages-(Tmat+log(1-g*(Tmat-t1)/3)/log(1+g/3)))))
  pred <- ifelse(LT_ages<=Tmat,juv,adult)
  return(pred)
}
```

First, we must load the required libraries and specify the simulated 'true' life history parameter values. We will set the coefficient of variation in length-at-age at 20%, a little larger than typically observed for fisheries data (see review in Lorenzen (2016)).

```
library(boot)  # use install.packages('boot') if package isn't already installed
library(runjags)  #load, install, or require the 'runjags' library
```

```
library(rjags)
## Loading required package: coda
## Linked to JAGS 4.2.0
## Loaded modules: basemod,bugs
library(stats4)  #load the 'stats4' library
library(coda)

ages <- 1:30  #create an integer sequence of ages

Tmat <- 4  # age of maturity
h <- 50  # somatic growth in millimeters per year
t1 <- -0.2  #age when size=0 for the juvenile phase
M <- 0.25  #Natural mortality for the population
g <- 1.18 * (1 - exp(-M))  # proportion of energy in adult phase allocated to reproduction per year
cv <- 0.2  # coefficient of variation in size-at-age
linf = 3 * h/g  ## von Bertalanffy asymptotic length (mm)
vbk = log(1 + g/3)  ## Brody growth coefficient (per yr)
t0 = Tmat + suppressWarnings(log(1 - (g * (Tmat - t1)/3)))/log(1 + g/3)  ## von Bertalanffy hypotheti
cal age at length 0 (yr)

Nfish <- 100  # how many fish in total do you have?

lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages <= Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a fish
 is allocating surplus energy
```

Next, we will calculate survivorship-at-age l_a, which is the expected discrete annual survival based on constant mortality M calculated from a reference age, for the 30 ages (length of the vector `ages`). In this case $l_1 = 1$ and for every age $a >= 2$:

$$l_a = l_{a-1} * e^{-M}$$

.

We also induce selectivity-at-age s_a in the sampling process with the equation

$$s_a = 1/(1 + e^{slope*(a-a_{50})}$$

.

```
surv <- rep(NA, length(ages))  # create an empty vector
surv[1] <- 1
for (i in 2:max(ages)) {
    surv[i] <- surv[i - 1] * exp(-M)
}  #survivorship from discrte annual survival
gearA50 <- Tmat  # induce a gear selectivity that inflects at A50
gearSlope <- -0.4  # whats the slope of selectivity
select <- 1/(1 + exp(gearSlope * (ages - gearA50)))  # the average selectivity curve
```

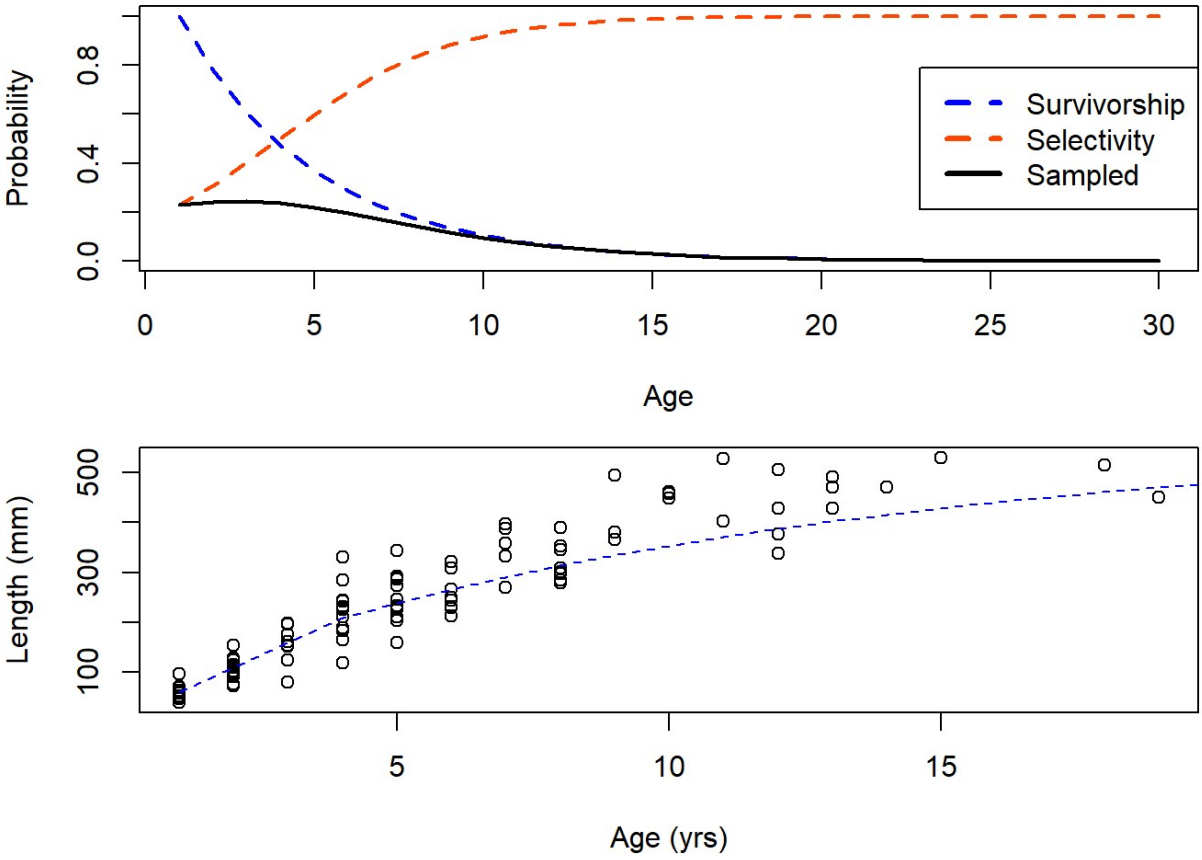
Next, we will generate data using a random normal distribution (realized parameter values and model fit quality will change due to randomness) using the `rnorm()` function. Sample sizes expected for each age are realistic for fisheries data and are determined using a function of gear selectivity and survivorship arising from a multinomial process using the `rmultinom()` function. The expected probability of sampling fish of a particular age class will be $l_a * s_a$. Alternatively, we could have read in our data with, for example, a .csv file replacing the `Data` object.

```
SampSize <- 10000
set.seed(2016)
maxSamp <- as.vector(rmultinom(1, prob = surv * select, size = SampSize))  # whats the maximum number
 of observable samples for an age group in a population?
mean.samp <- as.data.frame(cbind(biphasic, maxSamp))  #make matrix of mean lengths-at-age and sample
sizes
mlen <- rep(mean.samp[, 1], mean.samp[, 2])  #repeat each mean 'sample size' number of times
ageData <- rep(ages, mean.samp[, 2])  #repeat each age 'sample size' number of times
lengths <- sapply(mlen, function(x) rnorm(1, mean = x, sd = x * cv))  #generate random normal length
data using means & cv error
Data <- as.data.frame(cbind(ageData, lengths))  #bind vectors into age and length matrix, covert to d
ata frame
```

```
colnames(Data) <- c("Age", "Size")   # re-name the columns
Data <- Data[sample(nrow(Data), size = Nfish, replace = F), ]  #draw a random sample from the populat
ion -- total sample size can be adjusted with the Nfish parameter
```

Here is how our data look currently with the expected samples-at-age from the multinomial process.

```
layout(matrix(c(1, 2), nrow = 2, ncol = 1))
par(mar = c(4, 4, 1, 1))
plot(ages, surv, ylab = "Probability", xlab = "Age", type = "l", lty = 2, col = "blue",
    lwd = 2)
lines(ages, select, lty = 2, col = "orangered", lwd = 2)
lines(ages, surv * select, lty = 1, lwd = 2)
legend("right", c("Survivorship", "Selectivity", "Sampled"), lty = c(2, 2, 1),
    col = c("blue", "orangered", "black"), lwd = 3)

plot(Data$Age, Data$Size, ylab = "Length (mm)", xlab = "Age (yrs)")   ## plot length-at-age
lines(ages, biphasic, lty = 2, col = "blue")
```



Approach 1 - Penalized likelihood

The next step is to duplicate what we did in Appendix S3, where we specify the likelihood function. We estimate five parameters: T, h, t_1, g, and the cv in length-at-age. We provide conversions for the von Bertalanffy (mature) growth parameters, and we use an if-else statement to differentiate between immature and mature growth. Finally, we use a penalized likelihood to help ensure that estimates of g do not venture outside of analytical bounds (see Lester, Shuter & Abrams (2004)).

```
nll <- function(theta) {
    h <- theta[1]
    t1 <- theta[2]
    g <- inv.logit(theta[3])
    size.cv <- theta[4]
    T_pred <- theta[5]
    ######################################################## make conversions to phase 2 VBGF paramet
ers #########
    linf <- 3 * h/g
    vbk <- log(1 + g/3)
    t0 <- Tmat + log(1 - g * (T_pred - t1)/3)/log(1 + g/3)

    ######################################################## Make predictions to the data ###########
```

```
#############

    pred1 <- h * (Data$Age - t1)  #predicted length for phase 1
    pred2 <- linf * (1 - exp(-vbk * (Data$Age - t0)))  #predicted length for phase 2
    pred_all <- ifelse(Data$Age <= T_pred, pred1, pred2)  #discontinuous maturity breakpoint
    ll <- dnorm(Data$Size, mean = pred_all, sd = pred_all * size.cv, log = TRUE)  #normal likelihood
with constant cv across ages
    if ((g < 0) | (g > (3/(T_pred - t1)))) {
        nll <- 1e+06  # penalized likelihood when g goes past bounds given in Lester et al. 2004; g m
ust be > 0 OR < 3/(T-t1)
    } else {
        nll <- -sum(ll)  # return the negative log-likelihood
    }
    return(nll)
}
```

Optimization and visualization of results

To fit the model, we first provide and compile starting values for each parameter. We then use the `optim()` function to optimize the likelihood function for our list of parameters.

In this case, we use three different starting value vectors to evaluate the sensitivity of each approach to starting values.

```
h.hat <- 35  # the following are initial parameter estimates
t1.hat <- -3
g.hat <- logit(0.07)
cv.hat <- 0.1
T.hat <- 11.5
theta <- c(h.hat, t1.hat, g.hat, cv.hat, T.hat)  # initial parameter estimates vector 1
theta2 <- c(h.hat * 1.5, t1.hat * 0.5, g.hat * 2, cv.hat * 1.5, T.hat * 0.75)  # initial parameter es
timates vector 3
theta3 <- c(h.hat * 2, t1.hat * 0.2, g.hat * 3, cv.hat * 2, T.hat * 0.5)  # initial parameter estimat
es vector 2

par.true <- c(h, t1, g, cv, Tmat)

fit <- optim(theta, nll, method = "BFGS", control = list(fnscale = 1, maxit = 1e+05,
    reltol = 1e-10), hessian = TRUE)
fit2 <- optim(theta2, nll, method = "BFGS", control = list(fnscale = 1, maxit = 1e+05,
    reltol = 1e-10), hessian = TRUE)
fit3 <- optim(theta3, nll, method = "BFGS", control = list(fnscale = 1, maxit = 1e+05,
    reltol = 1e-10), hessian = TRUE)
```

We then create a function that stores the maximum likelihood estimates and 95% asymptotically normal CI from a Hessian matrix.

```
optimFits <- function(x) {
    par.hats <- x$par
    par.hats[3] <- inv.logit(par.hats[3])  # return the g parameter in normal space from logit
    fisher_info <- solve(x$hessian)  #take the inverse of the hessian to get the var-covar matrix
    prop_sigma <- sqrt(diag(fisher_info))  #square-root the var-covar matrix to get sigmas (i.e., sta
ndard errors)
    SE.par <- prop_sigma
    UI <- par.hats + 1.96 * SE.par
    LI <- par.hats - 1.96 * SE.par
    perBias <- ((par.hats - par.true)/par.true) * 100
    LIBias <- ((LI - par.true)/par.true) * 100
    UIBias <- ((UI - par.true)/par.true) * 100
    return(list(mn.95 = rbind(par.hats, UI, LI), bias = rbind(perBias, UIBias,
        LIBias)))
}
m1 <- optimFits(fit)
m2 <- optimFits(fit2)
m3 <- optimFits(fit3)

approach1 <- list(m1, m2, m3)  #Store the results of approach 1 for plotting later
```

Approach 2 - Likelihood profiling

In this case, we will use the profile likelihood approach ((Honsey, Staples & Venturelli 2016)) See Appendix S4 for details.

```
Biphas.Lik.MA = function(parms) {
    # list parameters
    h1 <- parms[1]
    b0 <- parms[2]
    g <- inv.logit(parms[3])
    cv <- parms[4]
    age.i <- Data$Age[Data$Age <= mat.age]  ## define immature ages
    len.i <- Data$Size[Data$Age <= mat.age]  ## define immature lengths
    age.m <- Data$Age[Data$Age > mat.age]  ## define mature ages
    len.m <- Data$Size[Data$Age > mat.age]  ## define mature lengths

    ## Lester model equations
    t1 <- -b0/h1
    Linf <- 3 * h1/g
    k <- log(1 + g/3)
    t0 <- mat.age + suppressWarnings(log(1 - (g * (mat.age - t1)/3)))/log(1 +
        g/3)
    mn.i <- b0 + h1 * age.i
    mn.m <- Linf * (1 - exp(-k * (age.m - t0)))

    ## Likelihoods
    b0.lik <- dnorm(b0, mean = b0est, sd = 25, log = T)  #optional (also, distribution can be adjuste
d if needed)
    h1.lik <- dnorm(h1, mean = h1est, sd = 5, log = T)  #optional (also, distribution can be adjusted
 if needed)
    L.i <- dnorm(len.i, mean = mn.i, sd = mn.i * cv, log = T)
    L.m <- dnorm(len.m, mean = mn.m, sd = mn.m * cv, log = T)
    ll <- sum(c(L.i, L.m))  #without likelihood priors
    jll <- sum(c(L.i, L.m, b0.lik, h1.lik))  # with likelihood priors
    return(ll)
}
```

Below, we specify the three starting value vectors.

```
immdata <- Data[which(Data$Age <= (min(Data$Age) + 3)), ]  #choose data within first four ages -- num
ber of ages can be changed
immout <- lm(Size ~ Age, data = immdata)  #linear regression on 'immature' data -- change formulation
 as needed to match your data column names
b0est <- immout$coefficients[[1]]  # store intercept estimate, used for prior likelihood
h1est <- immout$coefficients[[2]]  # store slope estimate, used for prior likelihood
t1est <- -b0est/h1est
parms1 <- theta[-5]  #compile parameters
parms2 <- theta2[-5]  # initial parameter estimates
parms3 <- theta3[-5]  # initial parameter estimates
parms <- rbind(parms1, parms2, parms3)
parms[, 2] <- -parms[, 1] * parms[, 2]
```

Next, we make empty dataframes for storing the parameter estimates, and we generate a sequence of potential age-at-maturity values for profiling.

```
Mat.age <- seq(2, max(ages), by = 0.025)  #  range of mat.age values for profile likelihood calculati
on -- adjust as needed
lik <- b0.mat <- h.mat <- g.mat <- cv.mat <- rep(NA, length(Mat.age))  # create empty vectors for par
ameters
mat.age.df <- cbind(Mat.age, lik, h.mat, b0.mat, g.mat, cv.mat)  # create matrix for storing paramete
r estimates

mat.age.Lik <- array(mat.age.df, dim = c(dim(mat.age.df), 3))
```

We then optimize the likelihood function for each potential age-at-maturity value and store both parameter estimates and full-model likelihoods.

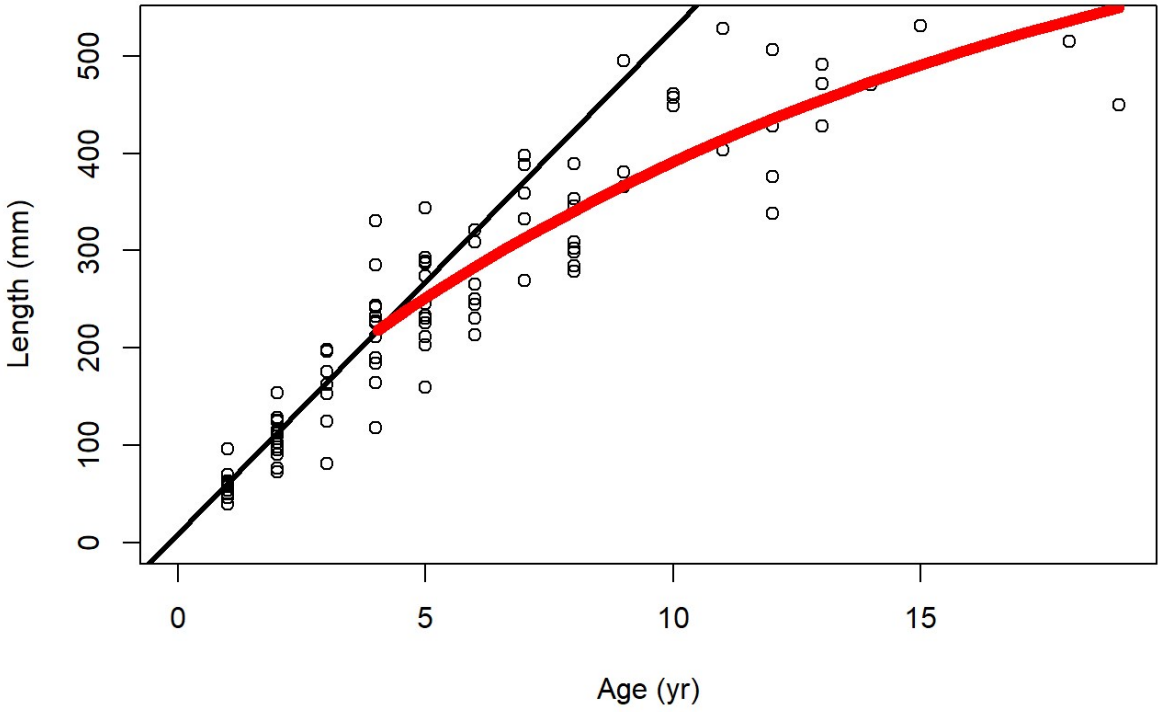
```
for(i in 1:3) #loop over 3 times for different starting vectors
{
  for(j in 1:length(Mat.age))
  {
    mat.age = Mat.age[j] # fix age-at-maturity at a given value
    L.out = try(optim(par=parms[i,],fn=Biphas.Lik.MA,
                      control=list(fnscale=-1,reltol=1e-8)), silent=T) # optimize likelihood function
    check<-is.numeric(L.out[[1]]) # check to see if model converged

    ## store values only if model converged
    if (check[[1]] == "TRUE"){

      #Store parameter values (back-transform g)
      mat.age.Lik[j,2,i] <- L.out$value
      mat.age.Lik[j,3,i] <- L.out$par[[1]]
      mat.age.Lik[j,4,i] <- L.out$par[[2]]
      mat.age.Lik[j,5,i] <- inv.logit(L.out$par[[3]])
      mat.age.Lik[j,6,i] <- L.out$par[[4]]
    }
  }
  XX <- as.data.frame(mat.age.Lik[,,i])
  colnames(XX) <- c("Mat.age","lik","h.mat","b0.mat","g.mat","cv.mat")
  assign(paste("mat.age.df",i,sep=""),XX)
}
```

Next, we find the maximum likelihood estimate for T (and the remaining parameters) from the profiling procedure.

```
findMLE <- function(x) {
    x1 <- x[which(x$lik != "NA"), ]  #remove failed runs
    mle <- max(x1$lik, na.rm = TRUE)  ## find maximum likelihood
    MLE <- x1[which(x1$lik == mle), ]  ## maximum likelihood estimates for all parameters
    ## Confidence interval in terms of chi-squared (~ 95% CI)
    ndx1 = which(x1$lik > (mle - 1.92))  # change '1.92' to 0.228 for 50% CI, 1.36 for 90% CI
    CI = x1[ndx1, -2]
    return(list(data = x1, best.est = MLE, mle = mle, CI95 = CI))  ## print maximum likelihood estima
tes
}

MLE1 <- findMLE(mat.age.df1)
MLE2 <- findMLE(mat.age.df2)
MLE3 <- findMLE(mat.age.df3)

approach2 <- list(MLE1, MLE2, MLE3)  # Store the results of approach 2 for plotting later
```

If desired, one can calculate confidence intervals and plot the biphasic growth curves onto the data.

```
rlike1 <- exp(MLE1$data$lik - MLE1$mle)
rlike2 <- exp(MLE2$data$lik - MLE2$mle)
rlike3 <- exp(MLE3$data$lik - MLE3$mle)

## Plot biphasic growth curves onto data
plot(Size ~ Age, data = Data, xlab = "Age (yr)", ylab = "Length (mm)", xlim = c(0,
    max(Age)), ylim = c(0, max(Size)))
g. = MLE1$best.est[[5]]
h1. = MLE1$best.est[[3]]
mT = MLE1$best.est[[1]]
b0. = MLE1$best.est[[4]]
t1. <- -b0./h1.
Linf. = 3 * h1./g.
k. = log(1 + g./3)
t0. = mT + log(1 - (g. * (mT - t1.)/3))/log(1 + g./3)
abline(b0., h1., lwd = 3)
matX = seq(mT, max(Data$Age), length.out = 25)
matY = Linf. * (1 - exp(-k. * (matX - t0.)))
lines(matX, matY, col = "red", type = "l", lwd = 6, lty = 1)
```



Approach 3 - Bayesican MCMC

Bayesian estimation

The following model code is written in the JAGS language (Plummer 2017). The model loops through each data point and determines the contribution of the predicted length for each fish to the posterior, which is the sum of the log-likelihood and the log-prior. The predicted growth follows the analytical model from the equations in Lester, Shuter & Abrams (2004). There is an if-else statement that determines if the predicted length-at-age of data point i comes from the juvenile phase or the adult phase. Modifications to this code require JAGS syntax and not R syntax.

```
model <- "model {

## likelihood

#loop through all data points
for(i in 1:Nfish) {
size[i] ~ dnorm(pred[i],1/(pred[i]*size.cv)^2)T(0,) # observed length-at-age should be normally distr
ibuted around predictions

juv[i] <- h*(age[i]-t1) # predicted growth for fish i for the juvenile phase

adult[i] <- (3*h/g)*(1-exp(-(log(1+g/3))*(age[i]-(Tmat+log(1-g*(Tmat-t1)/3)/log(1+g/3))))) # predicte
d growth for fish [i] for the adult phase

pred[i] <- ifelse(age[i]<=Tmat,juv[i],adult[i]) # does the age of fish i exceed the maturity predicte
d for its population?

} # end the calculation of the likelihood



## prior distributions

Tmat ~ dunif(0,max(age))
h ~ dnorm(30,1e-3)T(0,)
t1 ~ dnorm(0,1)
g ~ dnorm(0.1,0.001)T(0,3/(Tmat-t1))
size.cv ~ dgamma(0.01,0.01)
}"
```

We then compile the data into a list for JAGS. In addition, we must provide starting values for each parameter within each chain. In this case, we use a random number generator to 'jitter' the starting values for each chain. We then compile the starting values for each chain into a list (in effect, creating a 'list of lists').

```
dataPop <- Data
data <- list(Nfish = length(Data$Age), age = Data$Age, size = Data$Size)

inits1 <- list(h = theta[1], t1 = theta[2], g = inv.logit(theta[3]), Tmat = theta[5],
    size.cv = theta[4])  # initial values mirroring approaches 1 and 2 starting values

inits2 <- list(h = theta2[1], t1 = theta2[2], g = inv.logit(theta2[3]), Tmat = theta2[5],
    size.cv = theta2[4])

inits3 <- list(h = theta3[1], t1 = theta3[2], g = inv.logit(theta3[3]), Tmat = theta3[5],
    size.cv = theta3[4])

inits <- list(inits1, inits2, inits3)  # compile all initial values into one list
```

Our last step before fitting the model is to provide some parameters for the JAGS MCMC algorithm. We first create the stochastic nodes to be monitored. We then store values for the thinning rate, the length of the burn-in (or warm-up) period, and the length of the adaptation period. In this case, we specify these values based on the total number of posterior draws desired for each chain.

We use the `run.jags()` command (Denwood 2016) to fit the model, and we reference the JAGS parameters as specified above. We use the `rjags` method in this case – for larger datasets or more complicated (e.g., hierarchical) models, the `rjparallel` method may be preferred. Users may need to run `install.packages("rjags")` prior to running this portion of code.

```
mon_names <- names(inits1)  # create the stochastic nodes to be monitored
Nsamp <- 1000  # how many posterior samples does each chain need to get, after thinning and burin-in
and adaptation?
thin_rt <- 20  # needs a decent thinning rate
burnins <- 0.75 * round(Nsamp * thin_rt, 0)  # how long is the burnin, this bases it on the number of
 total posterior draws?
adaptin <- round(0.4 * burnins, 0)

a <- proc.time()
results <- run.jags(model = model, monitor = mon_names, data = data, n.chains = 3,
    method = "rjags", inits = inits, plots = F, silent.jag = F, modules = c("bugs",
        "glm", "dic"), sample = Nsamp, adapt = adaptin, burnin = burnins, thin = thin_rt,
    summarise = F)  # Call jags to run the model
## module glm loaded
## module dic loaded
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 6000 iterations...
## Burning in the model for 15000 iterations...
## Running the model for 20000 iterations...
## Simulation complete
## Finished running the simulation
b <- (proc.time() - a)
```

Next, we will call the summary of the model fit. We can also show some diagnostic plots that evaluate whether the posterior has converged on a stable mode using the package `coda()` (Plummer *et al.* 2006).

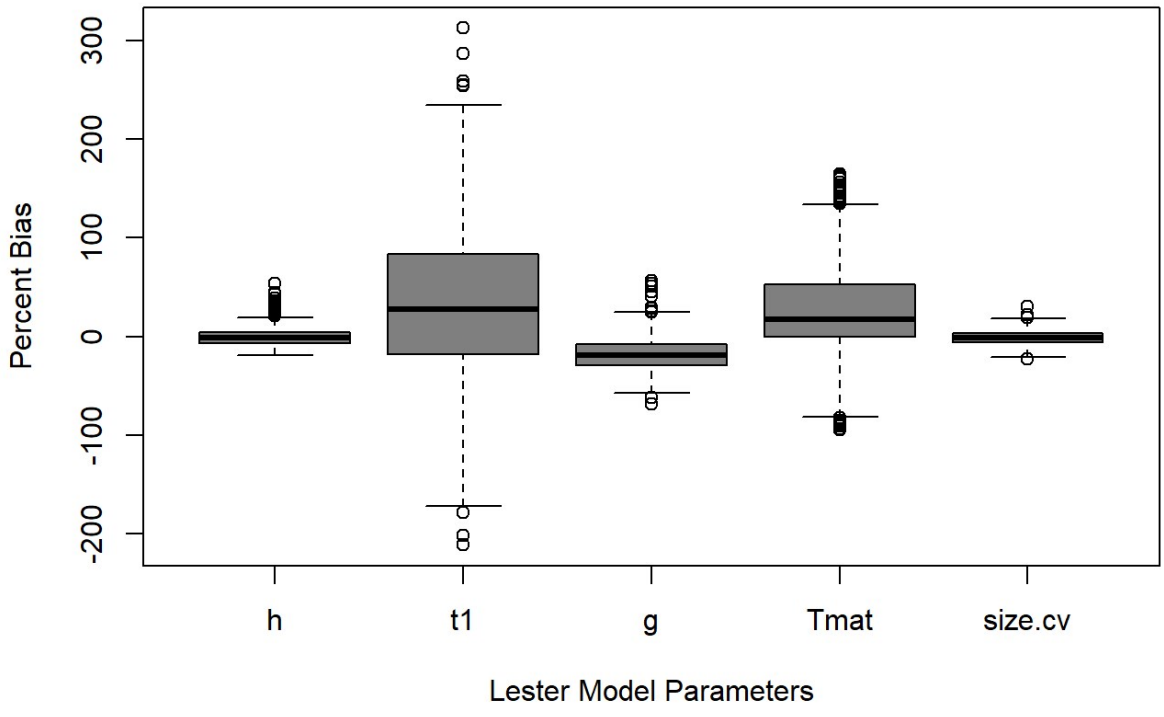
We then show a percent bias plot that suggests that parameters for this one randomized dataset were estimated well, apart from t_1 (which was estimated with noise). We could repeat this trial many times to get a general idea of how well the model recovers true life-history parameters of interest (we do this in Appendix S7).

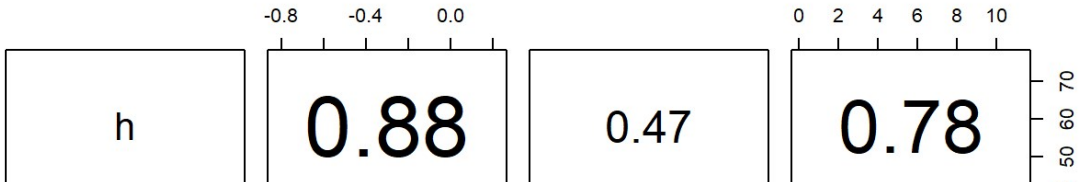
We next show a pairs plot to see how correlated the parameters are during the MCMC estimation. One may wish to use a multivariate normal distribution in the JAGS model above to allow for the correlations to be incorporated into the MCMC sampling (Helser & Lai 2004).

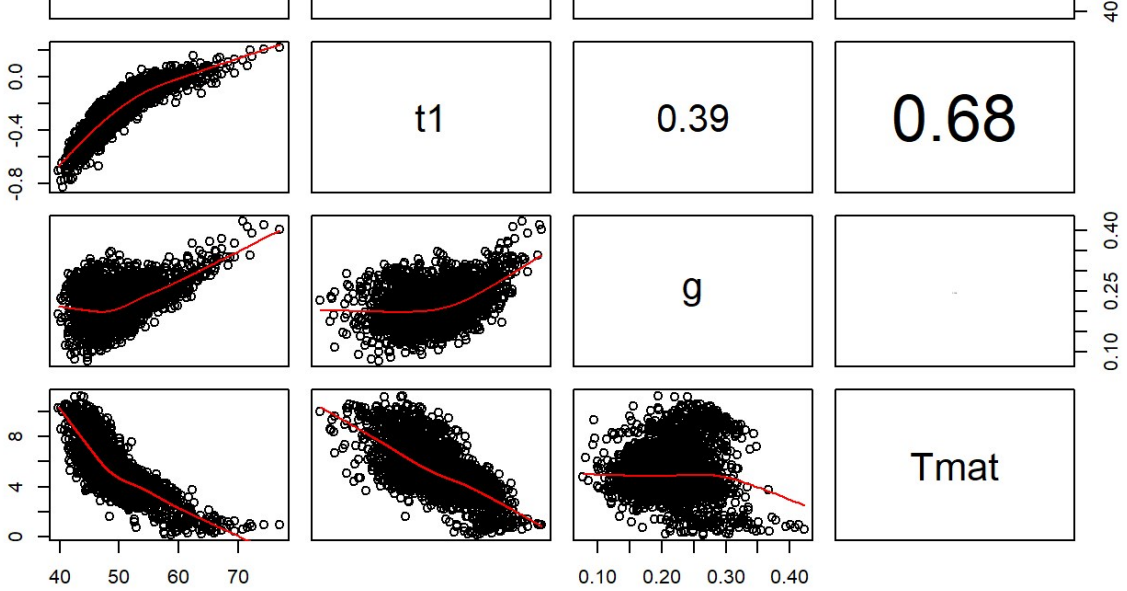
```
sum_results <- summary(results)
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 5 variables....
sum_results
```

```
##             Lower95     Median     Upper95       Mean         SD       Mode
## h       41.4646836 49.3106880 57.81588571 49.7661423 4.36232020 49.2761956
## t1      -0.5535540 -0.2583426  0.01613255 -0.2631185 0.14609852 -0.2420344
## g        0.1335616  0.2134546  0.30103322  0.2145708 0.04352655  0.2074849
## Tmat     2.1697070  4.7192489 10.01351461  5.1120007 1.85645543  4.4441446
## size.cv  0.1691741  0.1966198  0.22626641  0.1974413 0.01510957  0.1954942
##                MCerr MC%ofSD SSeff       AC.200     psrf
## h       0.1411876437     3.2   955  0.002774298 1.003564
## t1      0.0042461454     2.9  1184 -0.011431035 1.001313
## g       0.0010057233     2.3  1873 -0.011275664 1.000195
## Tmat    0.0502570199     2.7  1365  0.017755853 1.000366
## size.cv 0.0002857976     1.9  2795 -0.007222608 1.000591
```

```
TheRes <- as.mcmc.list(results, vars = mon_names)  # this is in the 'coda' package
TheRes1 <- as.matrix(TheRes[[1]])  ## results from starting point 1
TheRes2 <- as.matrix(TheRes[[2]])  ## results from starting point 2
TheRes3 <- as.matrix(TheRes[[3]])  ## results from starting point 3
approach3 <- list(TheRes1, TheRes2, TheRes3)
TheResTot <- as.matrix(TheRes)
BayesTruePar <- as.vector(c(h, t1, g, Tmat, cv))
bias1 <- t(apply(TheRes1, 1, FUN = function(x) {
    (x - BayesTruePar)/BayesTruePar * 100
}))
bias2 <- t(apply(TheRes2, 1, FUN = function(x) {
    (x - BayesTruePar)/BayesTruePar * 100
}))
bias3 <- t(apply(TheRes3, 1, FUN = function(x) {
    (x - BayesTruePar)/BayesTruePar * 100
}))
boxplot(bias1, ylab = "Percent Bias", xlab = "Lester Model Parameters", col = "grey50")

pairs(TheResTot[, -5], lower.panel = panel.smooth, upper.panel = panel.cor)  # look at the correlatio
ns between parameters
```



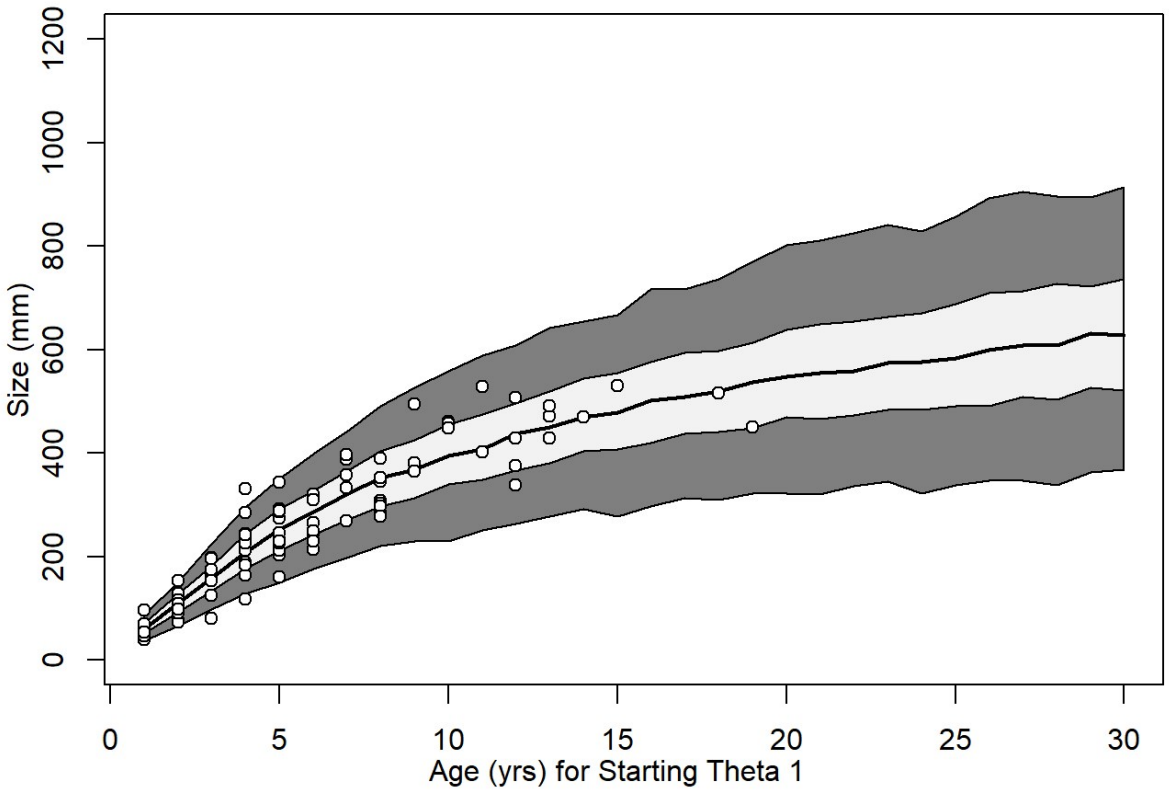


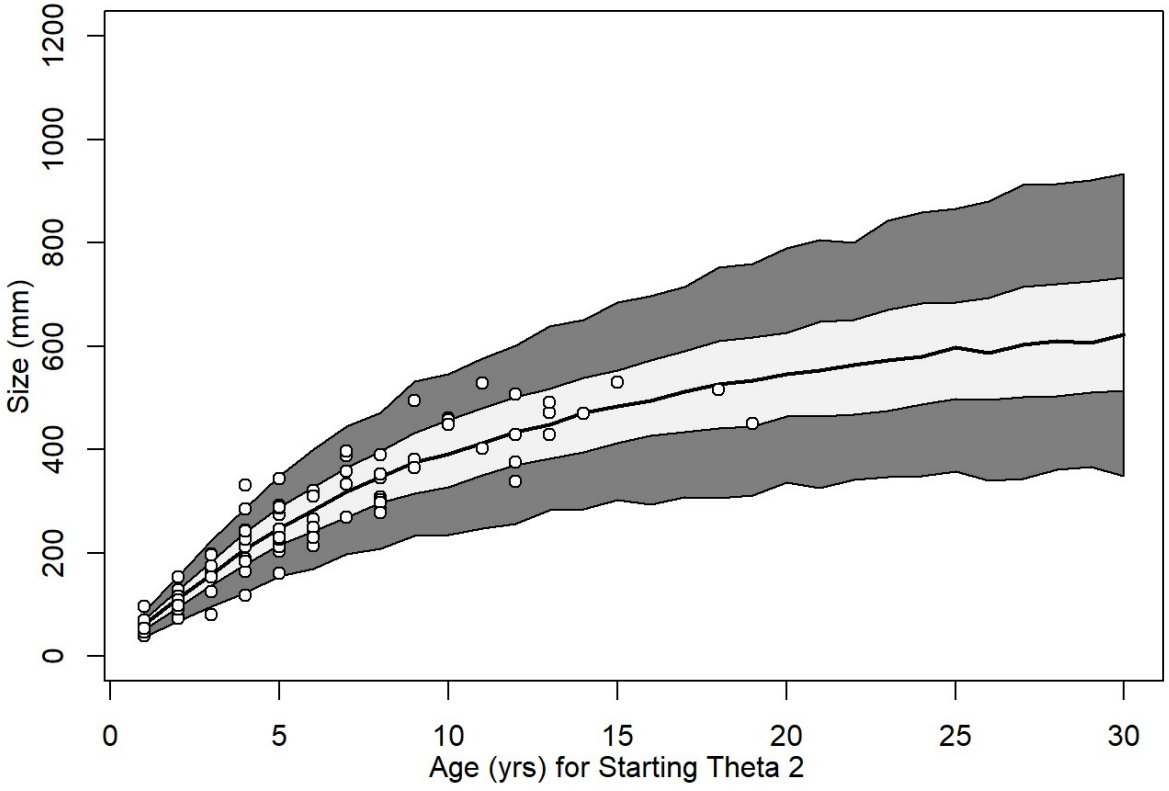


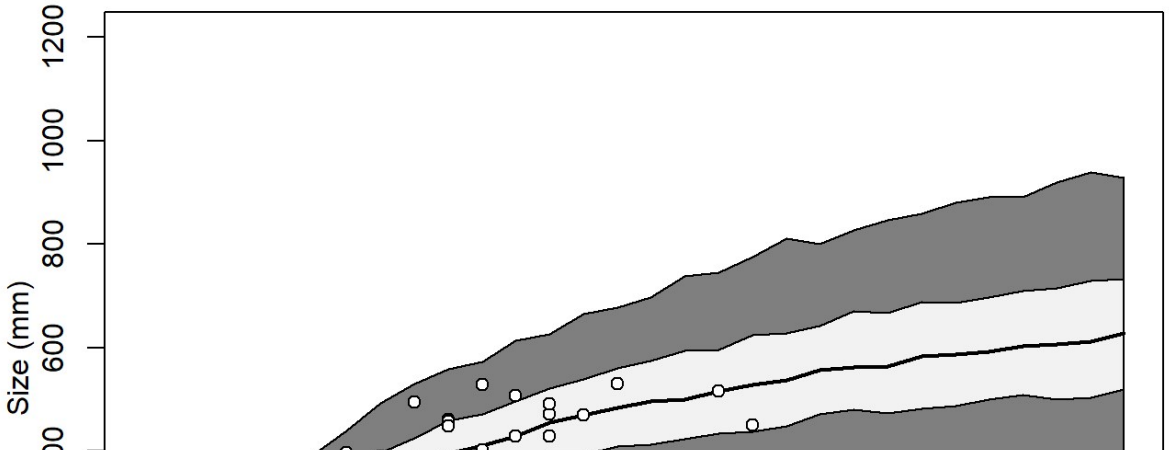
Posterior predictive check

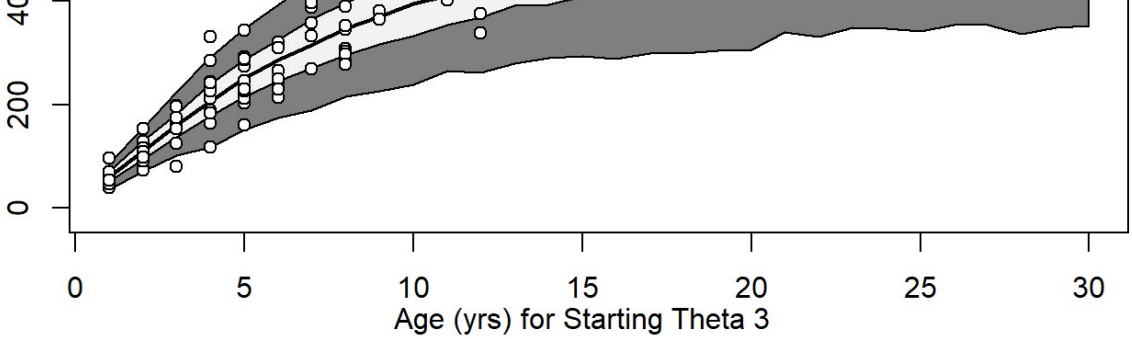
Next, we conduct a posterior predictive check (Gelman *et al.* (2013)). We simulate replicated data from our fitted JAGS model (with associated uncertainty for each estimated parameter from the model) and compare the distribution of our new *simulated* data to the *observed* data. In this case, the *observed* data is our original simulated data used in the model fitting in the `Data` object.

```
age_vec <- c(ages, rev(ages))
# Above code creates an empty array to track size-at-age for population i
# for posterior draw j
par(mfrow = c(1, 1))
par(mar = c(5, 4, 1, 1))
layout(matrix(1, nrow = 1, ncol = 1))
for (i in 1:3) {
    TheRes <- approach3[[i]]
    post_pred <- matrix(NA, nrow = nrow(TheRes1), ncol = length(ages))
    for (j in 1:nrow(TheRes)) {
        h_j <- TheRes[j, match("h", colnames(TheRes))]
        g_j <- TheRes[j, match("g", colnames(TheRes))]
        Tmat_j <- TheRes[j, match("Tmat", colnames(TheRes))]
        t1_j <- TheRes[j, match("t1", colnames(TheRes))]
        cv_j <- TheRes[j, match("size.cv", colnames(TheRes))]

        linf <- 3 * h_j/g_j  # conversion for the VBGF L-infinity
        vbk <- log(1 + g_j/3)  # conversion for the VBGF parameter kappa
        t0 <- Tmat_j + log(1 - g_j * (Tmat_j - t1_j)/3)/log(1 + g_j/3)  #conversion for the VBGF para
meter t0


        juv_j <- h_j * (ages - t1_j)
        adult_j <- linf * (1 - exp(-vbk * (ages - t0)))
        pred_j <- ifelse(ages <= Tmat_j, juv_j, adult_j)
        pred_j[pred_j < 0.001] <- 0.001
        post_pred[j, ] <- rnorm(length(ages), pred_j, pred_j * cv_j)
    }
    quants <- t(apply(post_pred[, ], 2, FUN = quantile, probs = c(0.025, 0.225,
        0.5, 0.775, 0.975)))
    plot(age_vec, c(quants[, 1], rev(quants[, 5])), type = "l", lwd = 2, col = NA,
        ylab = "", xlab = "", ylim = c(0, 1200))
    axis(1, at = median(c(0, max(ages))), paste("Age (yrs) for ", "Starting Theta ",
        i, sep = ""), tick = FALSE, line = 0.9)
    axis(2, at = 600, "Size (mm)", tick = FALSE, line = 1)
    polygon(age_vec, c(quants[, 1], rev(quants[, 5])), col = "grey50")
    polygon(age_vec, c(quants[, 2], rev(quants[, 4])), col = "grey95")
    lines(ages, quants[, 3], lwd = 2, col = "black")
    points(Data$Age, Data$Size, pch = 21, bg = "white")
}
```


Size (mm)
0
200
400
600
800
1000
1200
0
5
10
15
20
25
30
Age (yrs) for Starting Theta 1


Size (mm)
0
200
400
600
800
1000
1200
0
5
10
15
20
25
30
Age (yrs) for Starting Theta 2


Size (mm)
600
800
1000
1200



The posterior predictive distribution covers most of the data, and we can see from graphical portrayal that there are no systematic discrepancies between the *observed* and *simulated* data. In general, we might conclude the model is valid and fits well to the data (one could conduct alternative Bayesian goodness-of-fit tests as well).

Comparing the three different approaches using percent bias

First, we will store the approaches for percent bias calculations.

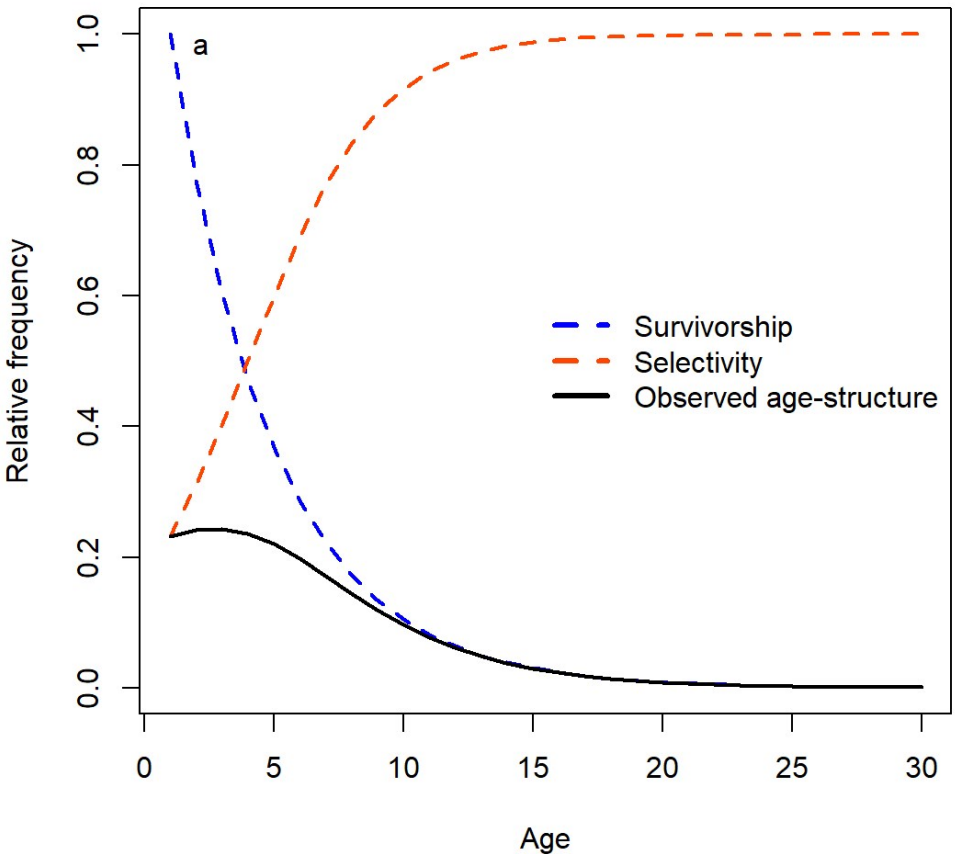
```
m2.1 <- apply(approach2[[1]]$CI95,2,FUN=function(x){return(c(min(x),max(x)))}) ## find the 95% for ap
proach 2 (profiling), start 1
m2.2 <- apply(approach2[[2]]$CI95,2,FUN=function(x){return(c(min(x),max(x)))}) ## find the 95% for ap
proach 2 (profiling), start 2
m2.3 <- apply(approach2[[3]]$CI95,2,FUN=function(x){return(c(min(x),max(x)))}) ## find the 95% for ap
proach 2 (profiling), start 3
CI <- rbind(approach2[[1]]$best.est[-2],m2.1, # storing for best estimates and 95% CI for profiling a
pproach
            approach2[[2]]$best.est[-2],m2.2,
            approach2[[3]]$best.est[-2],m2.3)
bias2 <- apply(CI,1,function(x){(x-c(Tmat,h,-h*t1,g,cv))/c(Tmat,h,-h*t1,g,cv)*100}) # calculate perce
nt bias for profiling approach
m3.1 <- apply(approach3[[1]],2,FUN=quantile,probs=c(0.025,0.5,0.975)) ## find the 95% for approach 3,
 start 1
m3.2 <- apply(approach3[[2]],2,FUN=quantile,probs=c(0.025,0.5,0.975)) ## find the 95% for approach 3,
 start 2
m3.3 <- apply(approach3[[3]],2,FUN=quantile,probs=c(0.025,0.5,0.975)) ## find the 95% for approach 3,
 start 3
bias3.1 <- apply(m3.1,1,function(x){(x-c(h,t1,g,Tmat,cv))/c(h,t1,g,Tmat,cv)*100}) ## calculate percen
t bias for Bayesian approach, starting point 1
bias3.2 <- apply(m3.2,1,function(x){(x-c(h,t1,g,Tmat,cv))/c(h,t1,g,Tmat,cv)*100}) ## calculate percen
t bias for Bayesian approach, starting point 2
bias3.3 <- apply(m3.3,1,function(x){(x-c(h,t1,g,Tmat,cv))/c(h,t1,g,Tmat,cv)*100}) ## calculate percen
t bias for Bayesian approach, starting point 3
```

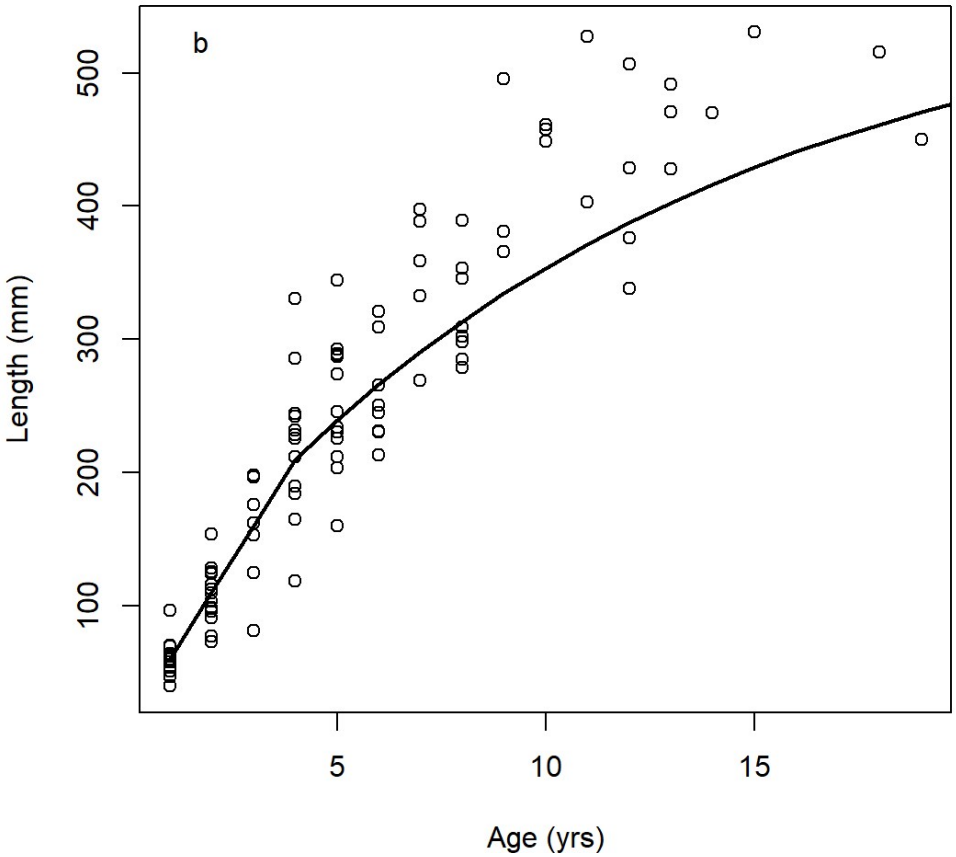
Then we make plots for showing the bias for each parameter, each starting vector, and each approach.

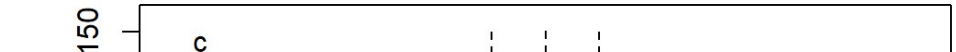
```
# layout(matrix(1:6,nrow=3,ncol=2,byrow=T))
layout(matrix(1, nrow = 1, ncol = 1, byrow = T))
par(mar = c(4, 4, 1, 8))

plot(ages, surv, ylab = "Relative frequency", xlab = "Age", type = "l", lty = 2,
    col = "blue", lwd = 2)
lines(ages, select, lty = 2, col = "orangered", lwd = 2)
lines(ages, surv * select, lty = 1, lwd = 2)
legend("right", c("Survivorship", "Selectivity", "Observed age-structure"),
    lty = c(2, 2, 1), col = c("blue", "orangered", "black"), lwd = 3, bty = "n")
Corner_text("a", "topleft")

plot(Data$Age, Data$Size, ylab = "Length (mm)", xlab = "Age (yrs)")  ## plot length-at-age
lines(ages, biphasic, lty = 1, col = "black", lwd = 2)
Corner_text("b", "topleft")
ylimits <- range(c(approach1[[1]]$bias, approach1[[2]]$bias, approach1[[3]]$bias),
    na.rm = T)
plot(1:15, c(approach1[[1]]$bias[1, 1], approach1[[2]]$bias[1, 1], approach1[[3]]$bias[1,
```

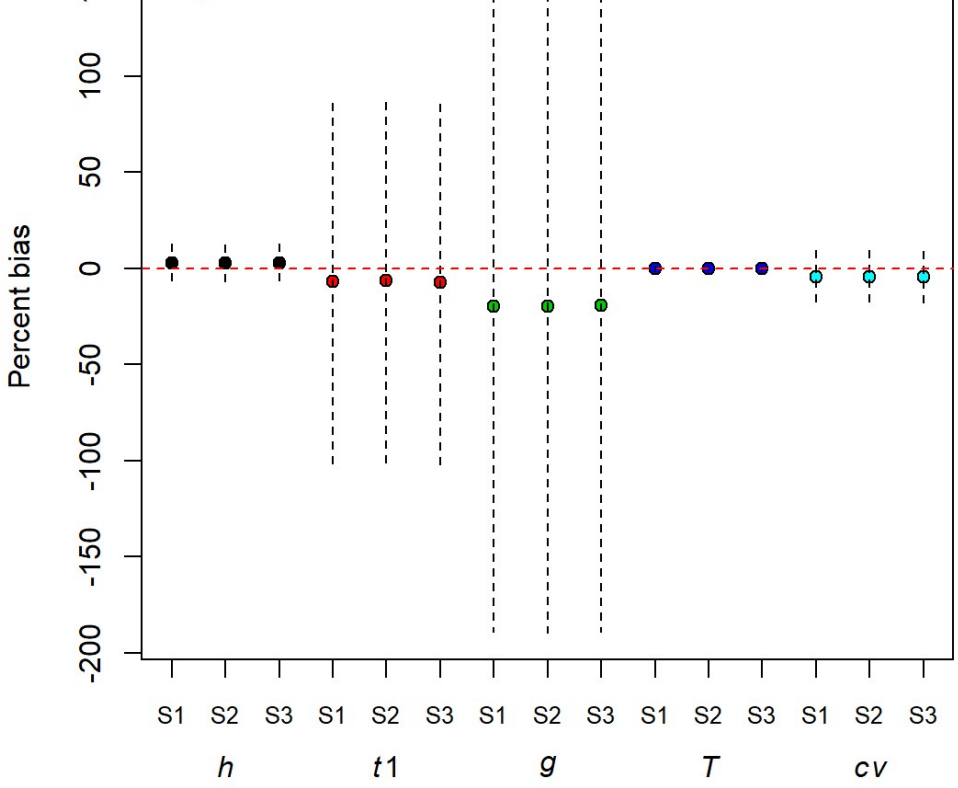
```
    1], approach1[[1]]$bias[1, 2], approach1[[2]]$bias[1, 2], approach1[[3]]$bias[1,
    2], approach1[[1]]$bias[1, 3], approach1[[2]]$bias[1, 3], approach1[[3]]$bias[1,
    3], approach1[[1]]$bias[1, 5], approach1[[2]]$bias[1, 5], approach1[[3]]$bias[1,
    5], approach1[[1]]$bias[1, 4], approach1[[2]]$bias[1, 4], approach1[[3]]$bias[1,
    4]), ylab = "Percent bias", xlab = "", xaxt = "n", ylim = ylimits, pch = 21,
    bg = c(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5))  # this is the bias at the maximum likelihoo
d estimates for approach 1
segments(x0 = 1:15, x1 = 1:15, y0 = c(approach1[[1]]$bias[2, 1], approach1[[2]]$bias[2,
    1], approach1[[3]]$bias[2, 1], approach1[[1]]$bias[2, 2], approach1[[2]]$bias[2,
    2], approach1[[3]]$bias[2, 2], approach1[[1]]$bias[2, 3], approach1[[2]]$bias[2,
    3], approach1[[3]]$bias[2, 3], approach1[[1]]$bias[2, 5], approach1[[2]]$bias[2,
    5], approach1[[3]]$bias[2, 5], approach1[[1]]$bias[2, 4], approach1[[2]]$bias[2,
    4], approach1[[3]]$bias[2, 4]), y1 = c(approach1[[1]]$bias[3, 1], approach1[[2]]$bias[3,
    1], approach1[[3]]$bias[3, 1], approach1[[1]]$bias[3, 2], approach1[[2]]$bias[3,
    2], approach1[[3]]$bias[3, 2], approach1[[1]]$bias[3, 3], approach1[[2]]$bias[3,
    3], approach1[[3]]$bias[3, 3], approach1[[1]]$bias[3, 5], approach1[[2]]$bias[3,
    5], approach1[[3]]$bias[3, 5], approach1[[1]]$bias[3, 4], approach1[[2]]$bias[3,
    4], approach1[[3]]$bias[3, 4]), lty = 2)  # this is the bias at the 95% CI from Hessian matrix fr
om approach 1
abline(h = 0, lty = 2, col = "red")
axis(1, at = c(2, 5, 8, 11, 14), expression(italic(h), italic(t1), italic(g),
    italic(T), italic(cv)), line = 1.5, tick = F)
axis(1, at = 1:15, rep(c("S1", "S2", "S3"), 5), line = 0, cex.axis = 0.8)
Corner_text("c", "topleft")

plot(1:15, c(bias2[2, c(1, 4, 7)], bias2[3, c(1, 4, 7)], bias2[4, c(1, 4, 7)],
    bias2[1, c(1, 4, 7)], bias2[5, c(1, 4, 7)]), pch = 21, bg = c(1, 1, 1, 2,
    2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5), ylab = "Percent bias", xlab = "", xaxt = "n",
    ylim = ylimits)  # this is the bias at the maximum likelihood estimates for profiling approach
segments(x0 = 1:15, x1 = 1:15, y0 = c(bias2[2, c(2, 5, 8)], bias2[3, c(2, 5,
    8)], bias2[4, c(2, 5, 8)], bias2[1, c(2, 5, 8)], bias2[5, c(2, 5, 8)]),
    y1 = c(bias2[2, c(3, 6, 9)], bias2[3, c(3, 6, 9)], bias2[4, c(3, 6, 9)],
        bias2[1, c(3, 6, 9)], bias2[5, c(3, 6, 9)]), lty = 2)  # segments draws the bias for the 95%
CI from profiling approach
abline(h = 0, lty = 2, col = "red")
axis(1, at = c(2, 5, 8, 11, 14), expression(italic(h), italic(t1), italic(g),
    italic(T), italic(cv)), line = 1.5, tick = F)
axis(1, at = 1:15, rep(c("S1", "S2", "S3"), 5), line = 0, cex.axis = 0.8)
Corner_text("d", "topleft")

plot(1:15, c(bias3.1[1, 2], bias3.2[1, 2], bias3.3[1, 2], bias3.1[2, 2], bias3.2[2,
    2], bias3.3[2, 2], bias3.1[3, 2], bias3.2[3, 2], bias3.3[3, 2], bias3.1[4,
    2], bias3.2[4, 2], bias3.3[4, 2], bias3.1[5, 2], bias3.2[5, 2], bias3.3[5,
    2]), pch = 21, bg = c(1, 1, 1, 2, 2, 2, 3, 3, 3, 4, 4, 4, 5, 5, 5), ylab = "Percent bias",
    xlab = "", xaxt = "n", ylim = ylimits)  # this is the bias at the mean posterior for Bayesian app
roach
segments(x0 = 1:15, x1 = 1:15, y0 = c(bias3.1[1, 1], bias3.2[1, 1], bias3.3[1,
    1], bias3.1[2, 1], bias3.2[2, 1], bias3.3[2, 1], bias3.1[3, 1], bias3.2[3,
    1], bias3.3[3, 1], bias3.1[4, 1], bias3.2[4, 1], bias3.3[4, 1], bias3.1[5,
    1], bias3.2[5, 1], bias3.3[5, 1]), y1 = c(bias3.1[1, 3], bias3.2[1, 3],
    bias3.3[1, 3], bias3.1[2, 3], bias3.2[2, 3], bias3.3[2, 3], bias3.1[3, 3],
    bias3.2[3, 3], bias3.3[3, 3], bias3.1[4, 3], bias3.2[4, 3], bias3.3[4, 3],
    bias3.1[5, 3], bias3.2[5, 3], bias3.3[5, 3]), lty = 2)  # this is the bias at the 95% CI posterio
r for Bayesian approach
abline(h = 0, lty = 2, col = "red")
axis(1, at = c(2, 5, 8, 11, 14), expression(italic(h), italic(t1), italic(g),
    italic(T), italic(cv)), line = 1.5, tick = F)

axis(1, at = 1:15, rep(c("S1", "S2", "S3"), 5), line = 0, cex.axis = 0.8)
Corner_text("e", "topleft")
plot(Size ~ Age, data = Data, pch = 21, bg = "white", ylab = "Length (mm)",
    xlab = "Age (yrs)")  # draw the growth trajectories for each of the 3 approaches at starting vect
or 2 (i.e., theta2)
lines(ages, biphasicPlot(h = approach1[[3]]$mn.95[1, 1], g = approach1[[3]]$mn.95[1,
    3], t1 = approach1[[3]]$mn.95[1, 2], Tmat = approach1[[3]]$mn.95[1, 5]),
    lty = 2, lwd = 2, col = "green")
lines(ages, biphasicPlot(h = as.numeric(approach2[[3]]$best.est[3]), g = as.numeric(approach2[[3]]$be
st.est[5]),
```

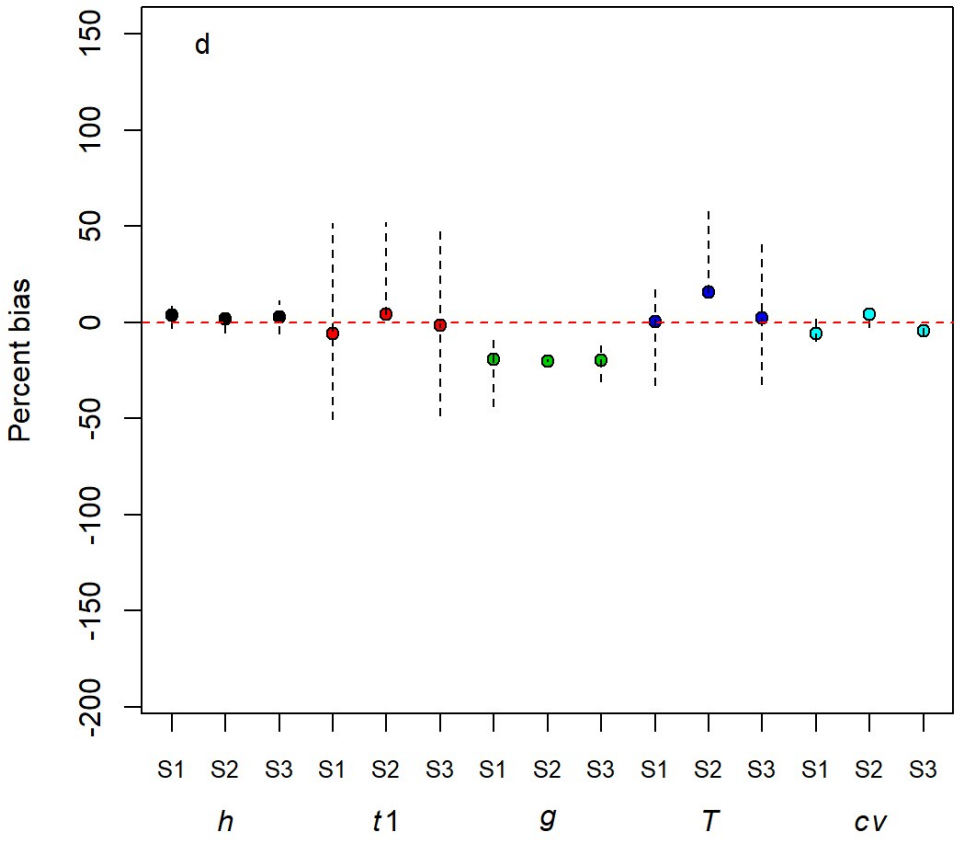
```
    t1 = as.numeric(-approach2[[3]]$best.est[4]/approach2[[3]]$best.est[3]),
    Tmat = as.numeric(approach2[[3]]$best.est[1])), lty = 2, lwd = 2, col = "blue")
lines(ages, biphasicPlot(h = mean(TheResTot[, "h"]), g = mean(TheResTot[, "g"]),
    t1 = mean(TheResTot[, "t1"]), Tmat = mean(TheResTot[, "Tmat"])), lty = 2,
    lwd = 2, col = "orange")
lines(ages, biphasic, lty = 1, lwd = 2, col = "black")
legend("bottomright", c("True growth", "Observed size-at-age", "Penalized like.",
    "Like. profile", "Bayesian MCMC"), lty = c(1, NA, 2, 2, 2), lwd = c(2, 1,
    2, 2, 2), col = c(1, 1, "green", "blue", "orange"), pch = c(NA, 21, NA,
    NA, NA), pt.bg = c(NA, "white", NA, NA, NA), bty = "n", xpd = NA)
Corner_text("f", "topleft")
```

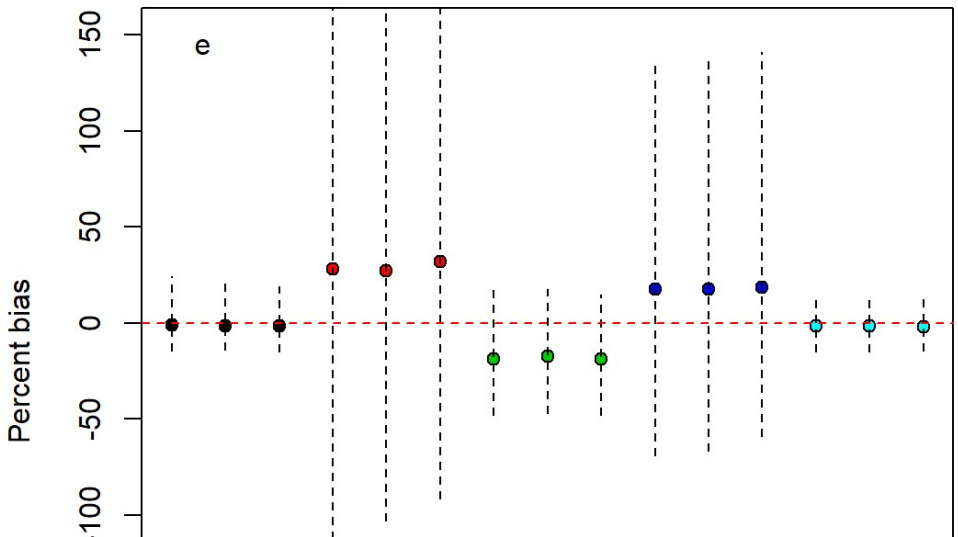


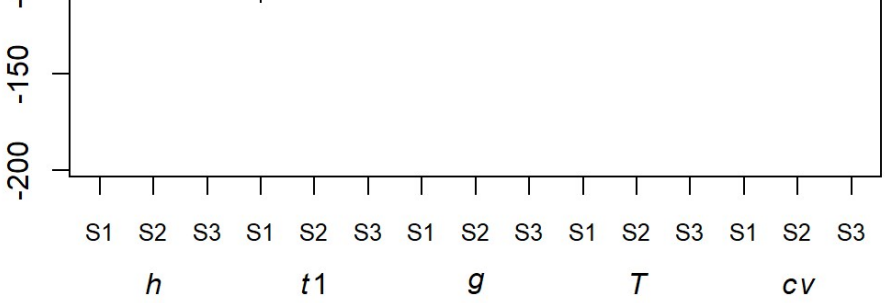


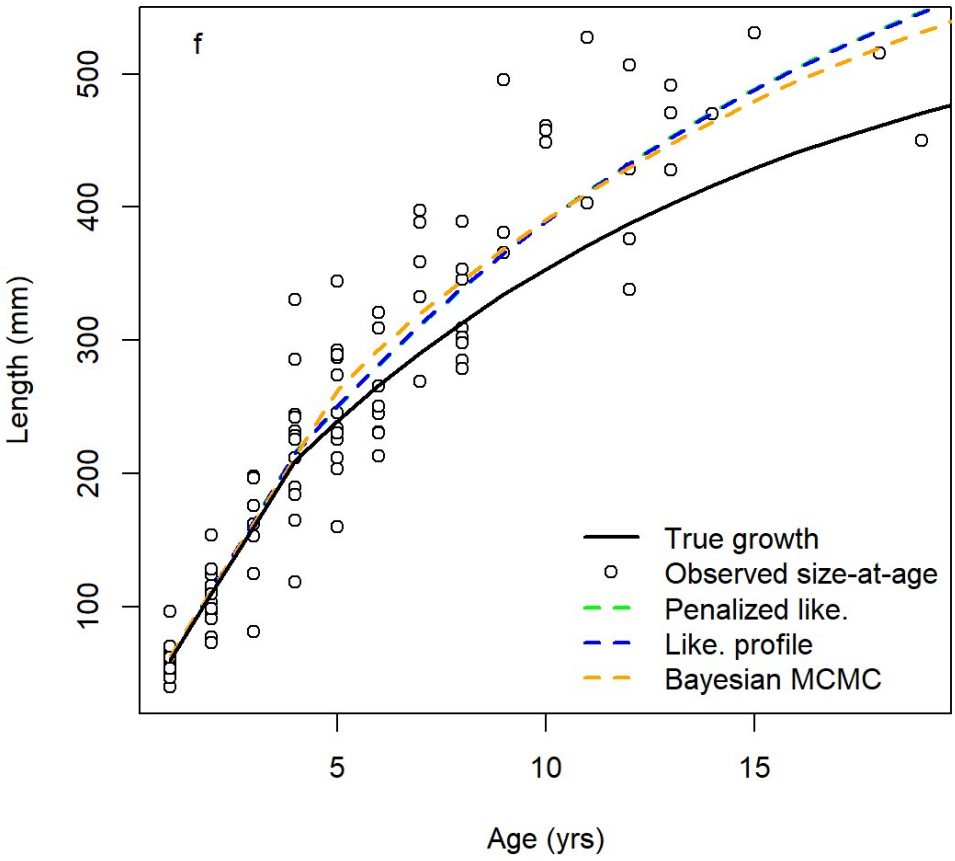



Percent bias
100
50
0
-50
-100
-150
-200
S1 S2 S3 S1 S2 S3 S1 S2 S3 S1 S2 S3 S1 S2 S3
h
t1
g
T
cv


d
Percent bias
150
100
50
0
-50
-100
-150
-200
S1 S2 S3 S1 S2 S3 S1 S2 S3 S1 S2 S3 S1 S2 S3
h
t1
g
T
cv


e
Percent bias
150
100
50
0
-50
-100





Overall, we can see that the parameters are estimated fairly well for all three approaches. For this one simulation run, we can see that MCMC (panel e) and profiling (panel d) did particularly well, while the penalized likelihood (panel c) had less precision on t_1 and g compared to the other two approaches. Different starting values can lead to less accurate or precise estimates of the life-history parameters. A more robust simulation that repeats this simulation several times is available in Appendix S7.

References

Denwood, M.J. (2016). runjags: An R package providing interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. *Journal of Statistical Software*, **71**, 1–25.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. & Rubin, D. (2013). *Bayesian Data Analysis*, 3rd edn. Chapman and Hall/CRC Press.

Helser, T.E. & Lai, H.-L. (2004). A bayesian hierarchical meta-analysis of fish growth: With an example for north american largemouth bass, micropterus salmoides. *Ecological Modelling*, **178**, 399–416.

Honsey, A.E., Staples, D.F. & Venturelli, P.A. (2016). Accurate estimates of age at maturity from the growth trajectories of fishes and other ectotherms. *Ecological Applications*, **27**, 182–192.

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Plummer, M. (2017). JAGS: A program for analysis of bayesian graphical models using gibbs sampling.

Plummer, M., Best, N., Cowles, K. & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for mcmc. *R News*, **6**, 7–11.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for

fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S7:
Bootstrapping three approaches to fitting biphasic growth models

Kyle L. Wilson
The University of Calgary

October 5, 2017

This appendix is in support of Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

Summary description of objectives:

The following script evaluates 3 different statistical approaches for fitting the Lester biphasic model where the breakpoint (age-at-maturity) is treated as unknown prior to model estimation (Lester, Shuter & Abrams 2004). We use three different sets of initial parameter values to evaluate the sensitivity of each approach to starting values. We repeatedly generate random datasets (i.e., bootstrapping) to evaluate model performance. We compare the approaches using root mean squared error

$$rmse = \sqrt{1/n * \sum_{i=1}^{n}(L_i - \hat{L}_i)^2}$$

and percent bias:

$$Bias = ((\theta_i - \hat{\theta}_i)/\theta_i) * 100$$

, which are useful metrics to evaluate model performance (Ono, Punt & Rivot 2012). Users can vary the scenarios and/or the number of bootstraps used (i.e., the `Nbootstraps` object). **Note** 100 iterations took ~24 hours to complete.

The simulation tests all combinations of three levels of late-stage juvenile and adult mortality ($M = \{0.1, 0.2, 0.5\}; yr^{-1}$) and the coefficient of variation in length-at-age ($cv_l = \{0.1, 0.15, 0.25\}$, nine scenarios total). Other leading parameters were held constant across simulation scenarios: age at size-0 ($t_1 = -0.2$), juvenile somatic growth rate ($h = 50mm * yr^{-1}$), and the slope of age-dependent selectivity = -0.3. Reproductive investment (g) and age-at-maturity (T) were calculated as functions of M using equations from Lester, Shuter & Abrams (2004), and the age-at-50% selectivity was equal to T. We then used a multinomial observation process to generate realistic samples (n = 50) of population age- and size-structure. The resulting length-at-age data are similar to what might be observed in a wild population.

The first step is to load the required libraries.

```
# Load required library
library(boot)  # use install.packages('boot') if package isn't already installed
library(runjags)  #load, install, or require the 'runjags' library
library(rjags)
library(stats4)  #load the 'stats4' library
library(coda)
library(parallel)
```

Global functions

The global functions defined below are also used in Appendices S1-S9. For example, the `nll()` function is our penalized negative log-likelihood which we introduced in Appendix S3. The profile likelihood approach is similar to that used in (Honsey, Staples & Venturelli 2016).

```
Corner_text <- function(text, location="topright") #function to write text to the corner of plots
{
  legend(location,legend=text, bty ="n", pch=NA)
}
```

```r
panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...) #function to change size of text i
n the pairs plots to match the size of the correlation
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}

biphasicPlot <- function(h=h,g=g,t1=t1,Tmat=Tmat) #function to return the biphasic growth trajectory
for a given set of parameters
{
  LT_ages <- ages
  juv <- h*(LT_ages-t1)
  adult <- (3*h/g)*(1-exp(-(log(1+g/3))*(LT_ages-(Tmat+log(1-g*(Tmat-t1)/3)/log(1+g/3)))))
  pred <- ifelse(LT_ages<=Tmat,juv,adult)
  return(pred)
}

nll <- function(theta)
{
  h <- theta[1]
  t1 <- theta[2]
  g <- inv.logit(theta[3])
  size.cv <- theta[4]
  T_pred <- theta[5]
  # make conversions to phase 2 VBGF parameters
  linf <- 3*h/g
  vbk <- log(1+g/3)
  t0 <- suppressWarnings(Tmat + log(1-g*(T_pred-t1)/3)/log(1+g/3))

  # Make predictions to the data
  pred1 <- h*(Data$Age-t1) #predicted length for phase 1
  pred2 <- linf*(1-exp(-vbk*(Data$Age-t0))) #predicted length for phase 2
  pred_all <- ifelse(Data$Age<=T_pred,pred1,pred2) #discontinuous maturity breakpoint
  ll <- dnorm(Data$Size,mean=pred_all,sd=pred_all*size.cv,log=TRUE) #normal likelihood with constant
cv across ages
  if((g<0)|(g>(3/(T_pred-t1)))){
    nll <- 1e6 # penalized likelihood when g goes past bounds given in Lester et al. 2004; g must be
> 0 OR < 3/(T-t1)
  }else{
    nll <- -sum(ll) # return the negative log-likelihood
  }
  return(nll)
}


optimFits <- function(x)
{
  par.hats <- x$par
  par.hats[3] <- inv.logit(par.hats[3]) # return the g parameter in normal space from logit
  fisher_info <- solve(x$hessian) #take the inverse of the hessian to get the var-covar matrix
  prop_sigma <- sqrt(diag(fisher_info)) #square-root the var-covar matrix to get sigmas (i.e., standa
rd errors)
  SE.par <- prop_sigma
  UI <- par.hats+1.96*SE.par
  LI <- par.hats-1.96*SE.par
  perBias <- ((par.hats-par.true)/par.true)*100
  LIBias <- ((LI-par.true)/par.true)*100
  UIBias <- ((UI-par.true)/par.true)*100
  return(list(mn.95=rbind(par.hats,UI,LI),bias=rbind(perBias,UIBias,LIBias)))
}
```

Bayesian estimation

The following model code is written in the JAGS language (Plummer 2017). The model loops through each data point and determines the contribution of the predicted length for each fish to the posterior, which is the sum of the log-likelihood and the log-prior. The predicted growth follows the analytical model from the equations in Lester, Shuter & Abrams (2004). There is an if/else statement that determines if the predicted length-at-age of data point i comes from the juvenile phase or the adult phase. Modifications to this code require JAGS syntax and not R syntax.

```
model <- "model {
#run through all the individual fish
for(i in 1:Nfish) {
size[i] ~ dnorm(pred[i],1/(pred[i]*size.cv)^2)T(0,) # cv predicts increasing variance with increasing
 size

#predict growth for each fish
juv[i] <- h*(age[i]-t1) # predicted growth for fish i for the juvenile phase

# below is the predicted growth for fish i for the adult phase
# which follows converting Lester et al. (2004) equations into the von Bertalanffy growth function

adult[i] <- (3*h/g)*(1-exp(-(log(1+g/3))*(age[i]-(Tmat+log(1-g*(Tmat-t1)/3)/log(1+g/3)))))
pred[i] <- ifelse(age[i]<=Tmat,juv[i],adult[i]) # does the age of fish i exceed the maturity predicte
d for its population?

} # end the calculation of the likelihood


# Section below - priors on life history traits
Tmat ~ dunif(min(age),max(age))
h ~ dnorm(40,1e-3)T(0,)
t1 ~ dnorm(0,1)
g ~ dnorm(0.10,0.01)T(0,3/(Tmat-t1))
size.cv ~ dgamma(0.01,0.01)
}"
```

Data generation

The function below generates data for a given life history scenario. We calculate survivorship-at-age l_a for the 30 ages (length of the vector `ages`, see below) which is the expected discrete annual survival based on constant mortality M calculated from a reference age. In this case $l_1 = 1$ and for every age $a >= 2$:

$$l_a = l_{a-1} * e^{-M}$$

.

We induce selectivity-at-age s_a in the sampling process with the equation

$$s_a = 1/(1 + e^{slope*(a-a_{50})}$$

. We then use `rmultinom()` to generate observation error and simulate realistic samples (n = 50) of population age- and size-structure.

```
generateData <- function(Tmat, h, t1, M, cv, gearSlope, ages, Nfish) {
    Tmat <- 1.95/(exp(M) - 1) + t1  # age of maturity
    g <- 1.18 * (1 - exp(-M))  # proportion of energy in adult phase allocated to reproduction per ye
ar
    linf = 3 * h/g  ## von Bertalanffy asymptotic length (mm)
    vbk = log(1 + g/3)  ## Brody growth coefficient (per yr)
    t0 = Tmat + suppressWarnings(log(1 - (g * (Tmat - t1)/3)))/log(1 + g/3)  ## von Bertalanffy hypot
hetical age at length 0 (yr)

    lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
    lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
    biphasic <- ifelse(ages <= Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a
fish is allocating surplus energy
```

```
    ## Step 1b: generate population's survivorship curve This will allow us to
    ## simulate more realistic age-structure
    surv <- rep(NA, length(ages))  # create an empty vector
    surv[1] <- 1
    for (i in 2:max(ages)) {
        surv[i] <- surv[i - 1] * exp(-M)
    }  #survivorship from discrte annual survival
    gearA50 <- Tmat  # induce a gear selectivity that inflects at A50
    select <- 1/(1 + exp(gearSlope * (ages - gearA50)))  # the average selectivity curve

    ## Generate data using a random algorithm (realized parameter values and
    ## model fit quality will change due to randomness) Sample sizes for each age
    ## are realistic for fisheries data, based on gear selectivity and natural
    ## mortality
    SampSize <- 10000
    # set.seed(1016)
    maxSamp <- as.vector(rmultinom(1, prob = surv * select, size = SampSize))  # whats the maximum nu
mber of observable samples for an age group in a population?
    # surv*select is the probability of a fish surviving a certain ageand being
    # sampled
    mean.samp <- as.data.frame(cbind(biphasic, maxSamp))  #make matrix of mean lengths-at-age and sam
ple sizes
    mlen <- rep(mean.samp[, 1], mean.samp[, 2])  #repeat each mean 'sample size' number of times
    ageData <- rep(ages, mean.samp[, 2])  #repeat each age 'sample size' number of times
    lengths <- abs(sapply(mlen, function(x) rnorm(1, mean = x, sd = x * cv)))  #generate random norma
l length data using means & cv error
    Data <- as.data.frame(cbind(ageData, lengths))  #bind vectors into age and length matrix, covert
to data frame
    colnames(Data) <- c("Age", "Size")  # re-name the columns
    Data <- Data[sample(nrow(Data), size = Nfish, replace = F), ]  #draw a random sample from the pop
ulation -- total sample size can be adjusted with the Nfish parameter
    return(Data)
}
```

Function for bootstrapping

The function below, called `bootstrapping()`, can be repeatedly called to fit the Lester biphasic model to various datasets using the three different approaches described above. The function returns the RMSE and percent bias on parameters for each approach within each iteration of the bootstrapping procedure. Note that this function is simply an amalgamation of functions in Appendices S3-S6. This function consists of 3 stages: penalized likelihood, likelihood profiling, and Bayesian MCMC. Each stage has a number of steps that include initializing and then fitting the model. The final lines of code calculate RMSE and percent bias and store those values in list objects named `rmse` and `biasList`. We encourage users to follow the comments closely for more details regarding what each line of code is doing.

```
bootstrapping <- function(start.par)
{
  #First Stage: the penalized likelihood approach
  #Step 1: Initialize and fit the penalized likelihood approach

  theta <- start.par$theta1 # initial parameter estimates vector 1
  theta2 <- start.par$theta2 # initial parameter estimates vector 2
  theta3 <- start.par$theta3 # initial parameter estimates vector 3

  fit <- suppressWarnings(optim(theta,nll,method='BFGS',control=list(fnscale=1,maxit=1e5,reltol=1e-10
),hessian=TRUE)) # fit the model with penalized likelihood for vector 1
  fit2 <- suppressWarnings(optim(theta2,nll,method='BFGS',control=list(fnscale=1,maxit=1e5,reltol=1e-
10),hessian=TRUE)) # fit the model with penalized likelihood for vector 2
  fit3 <- suppressWarnings(optim(theta3,nll,method='BFGS',control=list(fnscale=1,maxit=1e5,reltol=1e-
10),hessian=TRUE)) # fit the model with penalized likelihood for vector 3

  # Step 2: Store estimates with 95% asymptotically normal CI from Hessian matrix by calling the func
tion optimFits above

  m1 <- optimFits(x=fit)
  m2 <- optimFits(x=fit2)
  m3 <- optimFits(x=fit3)
```

```r
  approach1 <- list(m1,m2,m3) #Store the results of approach 1 for plotting later

  # Second stage: The likelihood profiling approach
  # Step 3: List starting values for each parameter
  immdata<- Data[ which(Data$Age <= (min(Data$Age)+3)), ] #choose data within first four ages -- numb
er of ages can be changed
  immout<-lm(Size~Age, data=immdata) #linear regression on "immature" data -- change formulation as n
eeded to match your data column names
  b0est<-immout$coefficients[[1]] # store intercept estimate, used for prior likelihood
  h1est<-immout$coefficients[[2]] # store slope estimate, used for prior likelihood
  t1est <- -b0est/h1est
  parms1 <- theta[-5]  #compile parameters
  parms2 <- theta2[-5] # initial parameter estimates
  parms3 <- theta3[-5] # initial parameter estimates
  parms <- rbind(parms1,parms2,parms3)
  parms[,2] <- -parms[,1]*parms[,2]

  # Create a vector of potential values for age-at-maturity and a matrix for storing parameter estima
tes

  Mat.age <- seq(round(0.25*theta[5],0),round(1.5*theta[5],0),by=0.025) #  range of mat.age values fo
r profile likelihood calculation -- adjust as needed
  lik<-b0.mat<-h.mat<-g.mat<-cv.mat<-rep(NA,length(Mat.age)) # create empty vectors for parameters
  mat.age.df <- cbind(lik,h.mat,b0.mat,g.mat,cv.mat,Mat.age) # create matrix for storing parameter es
timates

  mat.age.Lik <- array(mat.age.df,dim=c(dim(mat.age.df),3))

  # Step 4: Make Lester model likelihood function as 'Biphas.Lik.MA' excluding age-at-maturity parame
ter
  # Optional: include marginal likelihoods for immature growth slope and intercept
  Biphas.Lik.MA <- function(parms) { # likelihood function
    # list parameters
    h1 <- parms[1]
    b0 <- parms[2]
    g <- inv.logit(parms[3])
    cv <- parms[4]
    age.i <- Data$Age[Data$Age<=mat.age] ## define immature ages
    len.i <- Data$Size[Data$Age<=mat.age] ## define immature lengths
    age.m <- Data$Age[Data$Age>mat.age] ## define mature ages
    len.m <- Data$Size[Data$Age>mat.age] ## define mature lengths

    # Lester model equations
    t1 <- -b0/h1
    Linf <- 3*h1/g
    k <- log(1 + g/3)
    t0 <- mat.age +
      suppressWarnings(log(1-(g*(mat.age-t1)/3)))/log(1+g/3)
    mn.i <- b0 + h1*age.i
    mn.m <- Linf*(1-exp(-k*(age.m-t0)))

    # Likelihoods
    b0.lik <- dnorm(b0,mean=b0est,sd=25,log=T) #optional (also, distribution can be adjusted if neede
d)
    h1.lik <- dnorm(h1,mean=h1est,sd=5, log=T) #optional (also, distribution can be adjusted if neede
d)
    L.i <- dnorm(len.i,mean=mn.i,sd=mn.i*cv,log=T)
    L.m <- dnorm(len.m,mean=mn.m,sd=mn.m*cv,log=T)
    ll <- sum(c(L.i,L.m)) #without likelihood priors
    jll <- sum(c(L.i,L.m, b0.lik,h1.lik)) # with likelihood priors
    return(ll)
  }



  # Step 5: Analyze likelihood profiling results
  findMLE <- function(x){
```

```
    x1 <- x[which(x$lik != "NA"),] #remove failed runs
    mle <- max(x1$lik,na.rm=TRUE) ## find maximum likelihood
    MLE <- x1[which(x1$lik == mle),] ## maximum likelihood estimates for all parameters
    ## Confidence interval in terms of chi-squared (~ 95% CI)
    ndx1 = which(x1$lik>(mle-1.92)) # change '1.92' to 0.228 for 50% CI, 1.36 for 90% CI
    CI = x1[ndx1,-1] # exclude the first column, which is the likelihood value
    return(list(data=x1,best.est=MLE,mle=mle,CI95=CI)) ## print maximum likelihood estimates
  }

  # Step 6: Optimize likelihood profiling function for each potential age-at-maturity value and store
 parameter estimates
  for(i in 1:3) #loop over 3 times for three different starting vectors
  {
    for(j in 1:length(Mat.age))
    {
      mat.age = Mat.age[j] # fix age-at-maturity at a given value
      L.out = suppressWarnings(try(optim(par=parms[i,],fn=Biphas.Lik.MA,
                        control=list(fnscale=-1,reltol=1e-8)), silent=T)) # optimize likelihood funct
ion
      check<-is.numeric(L.out[[1]]) # check to see if model converged

      ## store values only if model converged
      if (check[[1]] == "TRUE"){

        #Store parameter values (back-transform g)
        mat.age.Lik[j,1,i] <- L.out$value
        mat.age.Lik[j,2,i] <- L.out$par[[1]]
        mat.age.Lik[j,3,i] <- -L.out$par[[2]]/L.out$par[[1]]
        mat.age.Lik[j,4,i] <- inv.logit(L.out$par[[3]])
        mat.age.Lik[j,5,i] <- L.out$par[[4]]
      }
    }
    XX <- as.data.frame(mat.age.Lik[,,i])
    colnames(XX) <- c("lik","h.mat","t1.mat","g.mat","cv.mat","Mat.age")
    assign(paste("mat.age.df",i,sep=""),XX)
  }

  MLE1 <- findMLE(mat.age.df1)
  MLE2 <- findMLE(mat.age.df2)
  MLE3 <- findMLE(mat.age.df3)

  approach2 <- list(MLE1,MLE2,MLE3) # Store the results of likelihood profiling approach for  later

  # Third stage: Bayesian MCMC approach
  ## Step 7: Declare the data in vectorized form, pass this to a list called 'data'
  dataPop <- Data
  data <- list(Nfish=length(Data$Age),
               age=Data$Age,
               size=Data$Size)
  # the above list compiles the noisy data from the Data dataframe into vectors for age, size

  # Step 8: initialize the parameters at the same starting points as first two approaches

  inits1 <- list(h=theta[1],
                 t1=theta[2],
                 g=inv.logit(theta[3]),
                 size.cv=theta[4],
                 Tmat=theta[5]) # initial values mirroring approaches 1 and 2 starting values

  inits2 <- list(h=theta2[1],
                 t1=theta2[2],
                 g=inv.logit(theta2[3]),
                 size.cv=theta2[4],
                 Tmat=theta2[5])

  inits3 <- list(h=theta3[1],
                 t1=theta3[2],
                 g=inv.logit(theta3[3]),
```

```
                size.cv=theta3[4],
                Tmat=theta3[5])

  inits <- list(inits1,inits2,inits3) # compile all initial values into one list

  mon_names <- names(inits1) # create the stochastic nodes to be monitored

  # Step 9: MCMC phase, call JAGS to run the model for a set amount of posterior draws as determined
by Nsamp and thinning

  Nsamp <- 1000 # how many posterior samples does each chain need to get, after thinning and burin-in
 and adaptation?
  thin_rt <- 15 # needs a decent thinning rate
  burnins <- 0.75*round(Nsamp*thin_rt,0) # how long is the burnin based on the number of total poster
ior draws
  adaptin <- round(0.4*burnins,0)
  cl <- makeCluster(3) # optional: create 3 clusters to do parallel processing on computer
  results <- run.jags(model=model, monitor=mon_names,
                      data=data, n.chains=3, method="rjparallel", inits=inits,
                      plots=F,silent.jag=F, modules=c("bugs","glm","dic"),
                      sample=Nsamp,adapt=adaptin,burnin=burnins,thin=thin_rt,summarise=F,cl=cl) # Cal
l jags to run the model
  stopCluster(cl) # stop the 3 clusters from continuing parallelization

  # Step 10: Store the results from JAGS, summarize, and plot bias and comparisons

  #sum_results <- summary(results)
  #gelman.diag(results)
  TheRes <- as.mcmc.list(results, vars=mon_names) # this is in the 'coda' package
  TheRes1 <- as.matrix(TheRes[[1]]) ## results from starting point 1
  TheRes2 <- as.matrix(TheRes[[2]]) ## results from starting point 2
  TheRes3 <- as.matrix(TheRes[[3]]) ## results from starting point 3
  approach3 <- list(TheRes1,TheRes2,TheRes3)

  # Step 11: Compare the three different approaches using root mean square error
  age_vec <- c(ages,rev(ages))
  rmse1 <- rmse2 <- rmse3 <- rep(NA,3)
  # Above code creates an empty array to track size-at-age for approach i for posterior draw j
  for(i in 1:3)
  {
    TheRes <- approach3[[i]]
    app1 <- approach1[[i]]$mn.95["par.hats",]
    app2 <- as.numeric(approach2[[i]]$best.est[-1])
    post_pred <- matrix(NA, nrow=nrow(TheRes1),ncol=length(ages)) # create empty matrix for posterior
 predictive length-at-age
    for(j in 1:nrow(TheRes))
    {
      h_j <- TheRes[j,match("h",colnames(TheRes))]
      g_j <- TheRes[j,match("g",colnames(TheRes))]
      Tmat_j <- TheRes[j,match("Tmat",colnames(TheRes))]
      t1_j <- TheRes[j,match("t1",colnames(TheRes))]
      cv_j <- TheRes[j,match("size.cv",colnames(TheRes))]

      linf <- 3*h_j/g_j # conversion for the VBGF L-infinity
      vbk <- log(1+g_j/3) # conversion for the VBGF parameter kappa
      t0 <- Tmat_j + log(1-g_j*(Tmat_j-t1_j)/3)/log(1+g_j/3) #conversion for the VBGF parameter t0
      juv_j <- h_j*(ages-t1_j)
      adult_j <- linf*(1-exp(-vbk*(ages-t0)))
      pred_j <- ifelse(ages<=Tmat_j,juv_j,adult_j)
      pred_j[pred_j<0.001] <- 0.001
      post_pred[j,] <- rnorm(length(ages),pred_j,pred_j*cv_j)
    }
    pred1 <- biphasicPlot(h=app1[1],t1=app1[2],g=app1[3],Tmat=app1[5])
    pred2 <- biphasicPlot(h=app2[1],t1=app2[2],g=app2[3],Tmat=app2[5])
    resids1 <- sqrt(mean((pred1[Data$Age]-Data$Size)^2)) # calculate root mean squared error for pena
lized likelihood approach
    resids2 <- sqrt(mean((pred2[Data$Age]-Data$Size)^2)) # calculate root mean squared error for like
```

```
lihood profiling approach
    quants <- t(apply(post_pred,2,FUN=quantile,probs=c(0.025,0.225,0.50,0.775,0.975)))
    pred3 <- t(apply(post_pred,2,FUN=mean))
    resids3 <- sqrt(mean((pred3[Data$Age]-Data$Size)^2)) # calculate root mean squared error for MCMC
 approach
    rmse1[i] <- resids1
    rmse2[i] <- resids2
    rmse3[i] <- resids3
  }

  m2.1 <- apply(approach2[[1]]$CI95,2,FUN=function(x){return(c(min(x),max(x)))}) ## find the 95% for
approach 2 (profiling), start 1
  m2.2 <- apply(approach2[[2]]$CI95,2,FUN=function(x){return(c(min(x),max(x)))}) ## find the 95% for
approach 2 (profiling), start 2
  m2.3 <- apply(approach2[[3]]$CI95,2,FUN=function(x){return(c(min(x),max(x)))}) ## find the 95% for
approach 2 (profiling), start 3
  CI <- rbind(approach2[[1]]$best.est[-1],m2.1, # storing for best estimates and 95% CI for profiling
 approach
              approach2[[2]]$best.est[-1],m2.2,
              approach2[[3]]$best.est[-1],m2.3)
  bias2 <- apply(CI,1,function(x){(x-par.true)/par.true*100}) # calculate percent bias for profiling
approach


  m3.1 <- apply(approach3[[1]],2,FUN=quantile,probs=c(0.025,0.5,0.975)) ## find the 95% for MCMC appr
oach, start 1
  m3.2 <- apply(approach3[[2]],2,FUN=quantile,probs=c(0.025,0.5,0.975)) ## find the 95% for MCMC appr
oach, start 2
  m3.3 <- apply(approach3[[3]],2,FUN=quantile,probs=c(0.025,0.5,0.975)) ## find the 95% for MCMC appr
oach, start 3
  bias3.1 <- apply(m3.1,1,function(x){(x-par.true)/par.true*100}) ## calculate percent bias for MCMC
approach, starting point 1
  bias3.2 <- apply(m3.2,1,function(x){(x-par.true)/par.true*100}) ## calculate percent bias for MCMC
approach, starting point 2
  bias3.3 <- apply(m3.3,1,function(x){(x-par.true)/par.true*100}) ## calculate percent bias for MCMC
approach, starting point 3
  dimnames(bias2) <- NULL

  # Step 12: Store percent bias as a list
  biasList <- list("penalty"=rbind(approach1[[1]]$bias[1,],approach1[[2]]$bias[1,],approach1[[3]]$bia
s[1,]),
                   "profiling"=t(bias2[,c(1,4,7)]),
                   "MCMC"=rbind(bias3.1[,2],bias3.2[,2],bias3.3[,2]))
  # create appropriate names for the list
  colnames(biasList$penalty) <- c("h","t1","g","cv","T")
  colnames(biasList$profiling) <- c("h","t1","g","cv","T")
  colnames(biasList$MCMC) <- c("h","t1","g","cv","T")

  rownames(biasList$penalty) <- c("S1","S2","S3")
  rownames(biasList$profiling) <- c("S1","S2","S3")
  rownames(biasList$MCMC) <- c("S1","S2","S3")

  return(list(approach1=rmse1,approach2=rmse2,approach3=rmse3,bias=biasList)) # return root mean squa
red error and percent bias for each approach
}
```

Life history scenarios

Below, we specify the simulated 'true' life history parameters. Later, we will feed these life history parameters into the data generation function (see above) to generate data describing a variety of life history scenarios. We will then use the bootstrapping function to determine the accuracy and precision of each of the three approaches in recovering the 'true' parameter values across scenarios.

```
ages <- 1:30  #create an integer sequence of ages
h <- 50  # somatic growth in millimeters per year
t1 <- -1  # age when size=0 for the juvenile phase
M <- c(0.1, 0.25, 0.5)  # Natural mortality for the population
```

```
cv <- c(0.1, 0.2, 0.3)  # coefficient of variation in length-at-age
gearSlope <- -0.3  #  the slope of selectivity in observing fish of a certain age
Nfish <- 50  # how many fish will be sampled from the population?
scenario <- expand.grid(h, t1, M, cv, gearSlope, Nfish)  # determine the unique combinations of the l
eading parameters
Nbootstraps <- 100  # how many times will we repeat the simulation?
Nscenario <- length(M) * length(cv)  # how many different scenarios are there?
scenNames <- paste("M=", apply(expand.grid(M, cv), 1, paste, collapse = ", cv="),
    sep = "")  # what are the names of the scenarios?
```

We use the code below to create some empty data objects (in this case, 4-D arrays) for storing percent bias and RMSE results for each approach and bootstrap iteration.

```
res <- array(NA, dim = c(Nbootstraps, Nscenario, 3, 3))  # create array to store rmse for each iterat
ion of the bootstrap, each scenario for the 3 methods and 3 starting points
bias <- array(NA, dim = c(Nbootstraps, Nscenario, 3, 3, 5))  # create array to store percent bias for
 each iteration of the bootstrap, each scenario for the 3 methods and 3 starting points, and 5 parame
ters
# below code creates names for the above arrays
dimnames(res) <- list(Bootstraps = 1:Nbootstraps, Scenario = scenNames, `Start Points` = c("Low",
    "Medium", "High"), Technique = c("Penalty", "Profiling", "MCMC"))
dimnames(bias) <- list(Bootstraps = 1:Nbootstraps, Scenario = scenNames, `Start Points` = c("Low",
    "Medium", "High"), Technique = c("Penalty", "Profiling", "MCMC"), Parameters = c("h",
    "t1", "g", "cv", "T"))
```

Bootstrapping the three approaches

The code below runs the bootstrapping procedure. The three starting values, labeled `startingPars1` `startingPars2` and `startingPars3` are specified. We also store the number of times that a given approach failed to run in the object `y`.

```
y <- rep(0, Nscenario)  # calculate how many times the penalized likelihood fails to estimate the bip
hasic model for a given dataset/scenario due to a non-positive definite Hessian matrix
a <- proc.time()
for (s in 1:Nscenario) {
    trueLH <- as.numeric(scenario[s, ])
    Tmat <- 1.95/(exp(trueLH[3]) - 1) + t1  # age of maturity depends on natural mortality and t1
    g <- 1.18 * (1 - exp(-trueLH[3]))  # proportion of energy in adult phase allocated to reproductio
n per year depends on natural mortality
    par.true <- c(h, t1, g, trueLH[4], Tmat)  # this scenarios true parameters
    startingPars1 <- c(h * 0.65, t1 * 2, logit(g * 0.65), trueLH[4] * 1.5, Tmat *
        1.35)  # starting parameters vector 1 f(h, t1, g, cv, Tmat)
    startingPars2 <- c(h * 1.15, t1 * 0.5, logit(g * 1.15), trueLH[4] * 0.75,
        Tmat * 0.85)  # starting parameters vector 1 f(h, t1, g, cv, Tmat)
    startingPars3 <- c(h * 1.65, t1 * 0.2, logit(g * 1.35), trueLH[4] * 1.25,
        Tmat * 0.65)  # starting parameters vector 1 f(h, t1, g, cv, Tmat)
    startingPars <- list(theta1 = startingPars1, theta2 = startingPars2, theta3 = startingPars3)
    for (i in 1:Nbootstraps) {
        Data <- generateData(Tmat = Tmat, h = trueLH[1], t1 = trueLH[2], M = trueLH[3],
            cv = trueLH[4], gearSlope = trueLH[5], Nfish = trueLH[6], ages = ages)  #generate 1 rando
m length-at-age dataset

        tempRes <- try(bootstrapping(start.par = startingPars), silent = T)  # try to estimate the mo
del parameters for the above fake dataset
        check <- is.numeric(tempRes[[1]])  # check to see if model converged
        if (check == FALSE)
            {
                y[s] <- y[s] + 1
            }  # if the model didn't converge, store how often it failed
        while (check == FALSE) {
            Data <- generateData(Tmat = Tmat, h = trueLH[1], t1 = trueLH[2],
                M = trueLH[3], cv = trueLH[4], gearSlope = trueLH[5], Nfish = trueLH[6],
                ages = ages)  # while the model has continued to fail converge, generate a new random
 length-at-age dataset
            tempRes <- try(bootstrapping(start.par = startingPars), silent = T)  # try to estimate th
e model parameters for the above fake dataset
            check <- is.numeric(tempRes[[1]])  # check to see if model converged
```

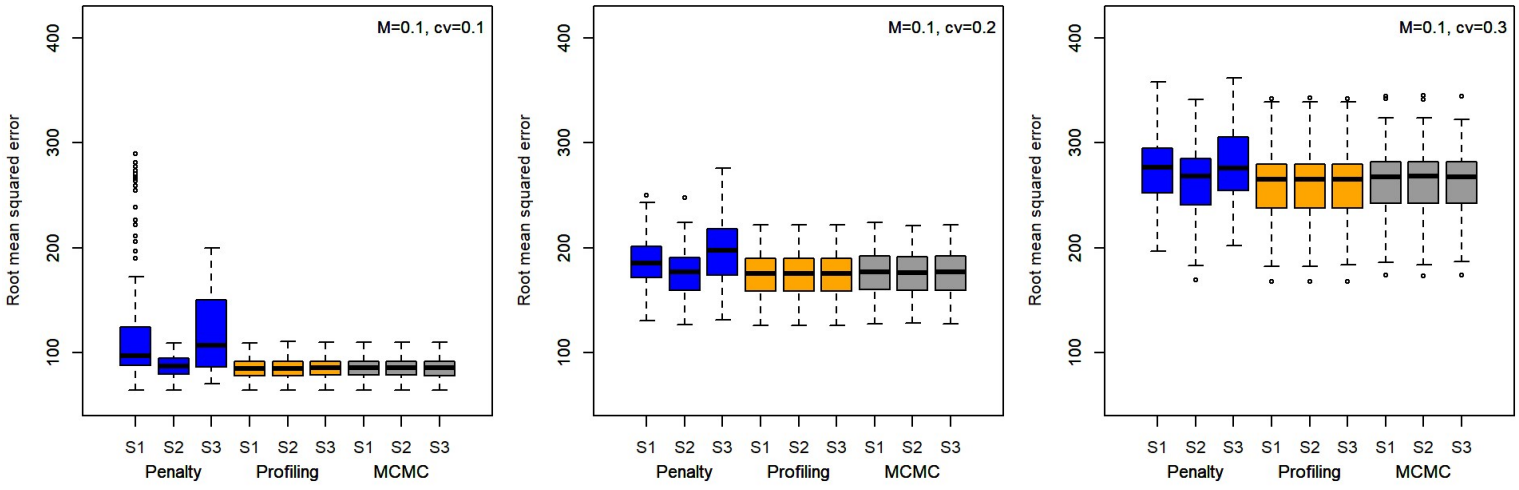
```
            if (check == FALSE)
                {
                  y[s] <- y[s] + 1
                }  # if the model still hasn't converged, add one more failure to the number
        }
        res[i, s, , ] <- cbind(tempRes$approach1, tempRes$approach2, tempRes$approach3)  # if the mod
el converged for all three approaches, store the rmse calculations
        bias[i, s, , 1, ] <- tempRes$bias$penalty  # compile the percent bias on the maximum likeliho
od (or posterior median) parameter estimates
        bias[i, s, , 2, ] <- tempRes$bias$profiling
        bias[i, s, , 3, ] <- tempRes$bias$MCMC
    }
}
b <- proc.time()
time <- (b[3] - a[3])/60/60  # how much time (in hours) has passesd
time/(Nscenario * Nbootstraps)
```

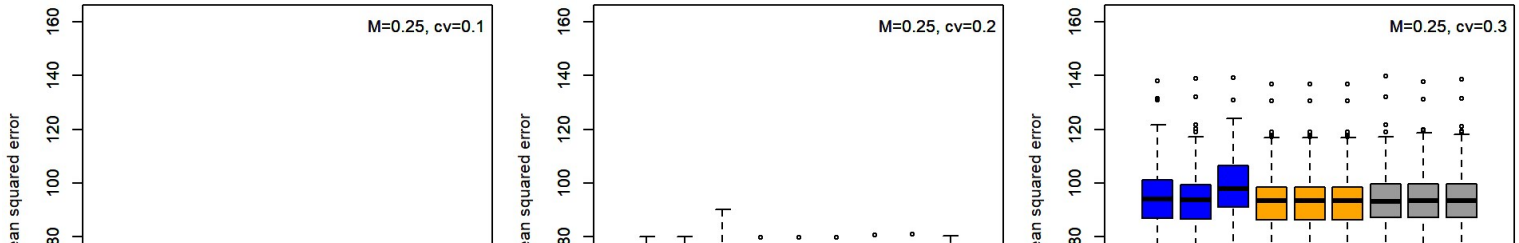
Compiling and visualizing results

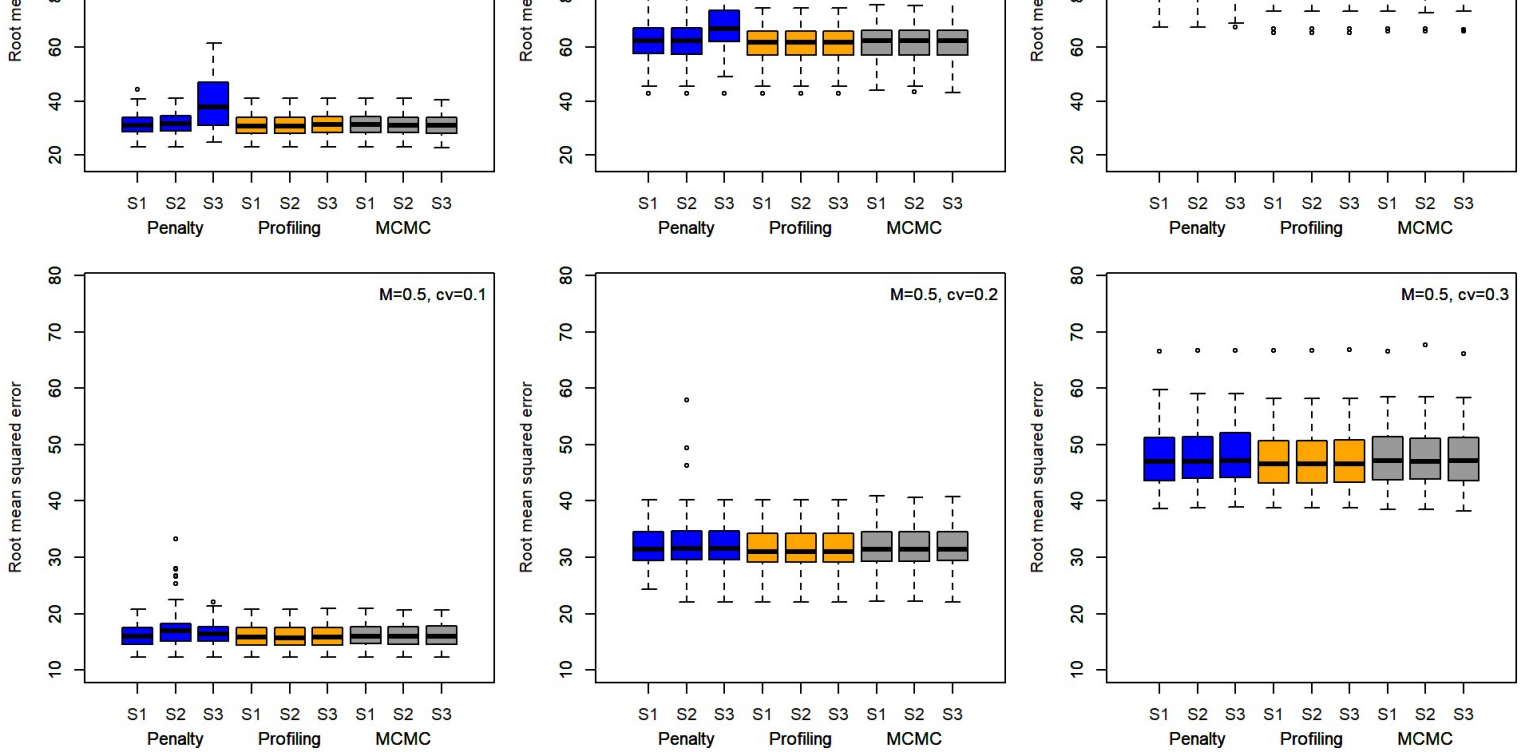
RMSE plots

We plot the RMSE results to compare the different fitting approaches.

```
xlimits <- matrix(1:Nscenario,nrow=3,ncol=3,byrow=TRUE)
naming <- matrix(rep(c("S1","S2","S3"),3),nrow=3,ncol=3,byrow=TRUE)
colours <- c("blue","orange","grey60")
layout(matrix(1:Nscenario,nrow=3,ncol=3))
par(mar=c(4,4,1,1))
for(s in 1:Nscenario)
{
  index <- grep(paste("M=",scenario[s,3],sep=""),dimnames(res)$Scenario)
  for(t in 1:3) # loop across the 3 methodological approaches
  {
    if(t==1){
      boxplot(res[,s,,t],at=xlimits[t,],xlim=c(0,10),names=naming[t,],ylim=range(res[,index,,],na.rm
= T)*c(0.85,1.15),ylab="Root mean squared error",col=colours[t])
    }else{
      boxplot(res[,s,,t],at=xlimits[t,],names=naming[t,],add=TRUE,col=colours[t])
    }
    mtext(side=1,at=c(2,5,8),dimnames(res)$Technique,line=2.25,cex=0.7)
    Corner_text(dimnames(res)$Scenario[s],location="topright")
  }
}
```





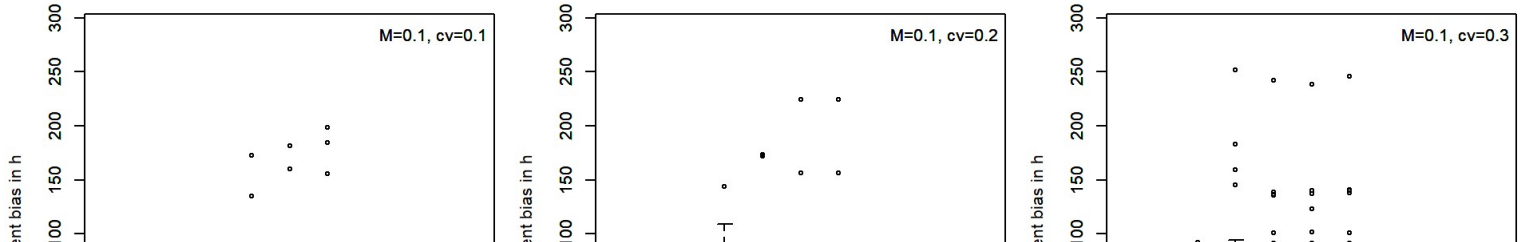


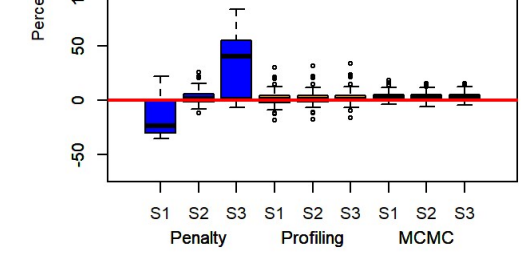
The RMSE results suggest that most of the approaches do well. Note the range of the y-axes decreases as M increases, showing that increased mortality leads to reduced bias and increased precision. This is likely because the breakpoint at maturity is more easily distinguished as mortality increases (due to increased investment in reproduction; Lester, Shuter & Abrams (2004)). At low mortality and low variance, the penalized likelihood approach was sensitive to starting vectors. As expected, RMSE increases as cv_l increases.

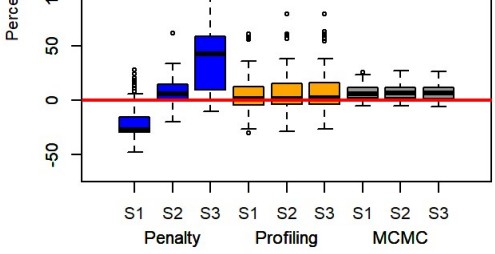
Percent bias plots

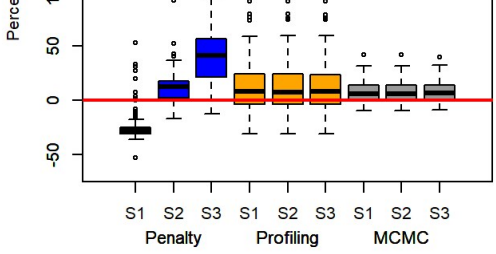
Finally, we plot the bias results for each parameter.

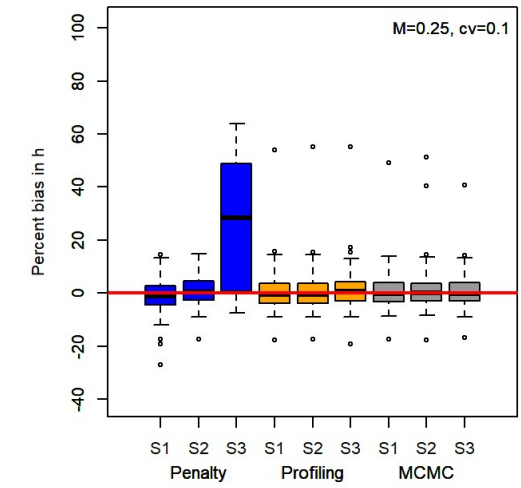
```
xlimits <- matrix(1:Nscenario,nrow=3,ncol=3,byrow=TRUE)
naming <- matrix(rep(c("S1","S2","S3"),3),nrow=3,ncol=3,byrow=TRUE)
layout(matrix(1:Nscenario,nrow=3,ncol=3))
par(mar=c(4,4,1,1))
for(p in 1:5) # loop across the 5 life history parameters
{
  for(s in 1:Nscenario)
  {
    index <- grep(paste("M=",scenario[s,3],sep=""),dimnames(res)$Scenario)
    for(t in 1:3) # loop across the 3 methodological approaches
    {
      if(t==1){
        boxplot(bias[,s,,t,p],at=xlimits[t,],xlim=c(0,10),names=naming[t,],ylim=range(bias[,index,,p
])*1.15,ylab=paste("Percent bias in ",dimnames(bias)$Parameters[p],sep=""),col=colours[t])
        abline(h=0,col="red",lwd=2)
      }else{
        boxplot(bias[,s,,t,p],at=xlimits[t,],names=naming[t,],add=TRUE,col=colours[t])
        abline(h=0,col="red",lwd=2)
      }
      mtext(side=1,at=c(2,5,8),dimnames(bias)$Technique,line=2.25,cex=0.7)
      Corner_text(dimnames(bias)$Scenario[s],location="topright")
    }
  }
}
```

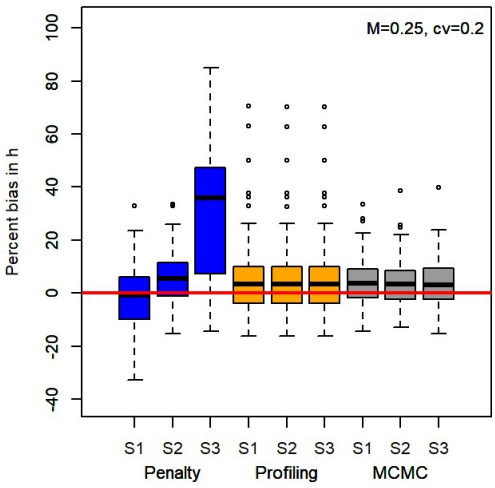


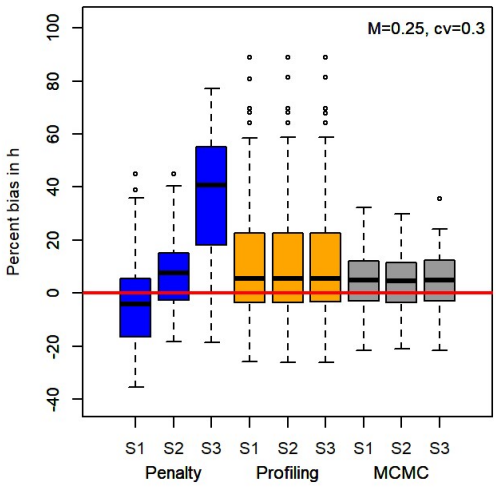

S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

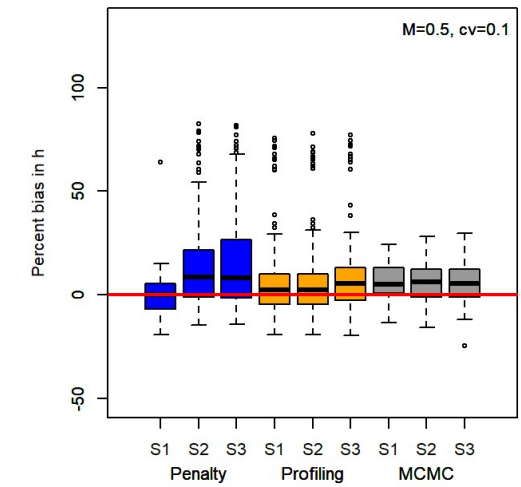

S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

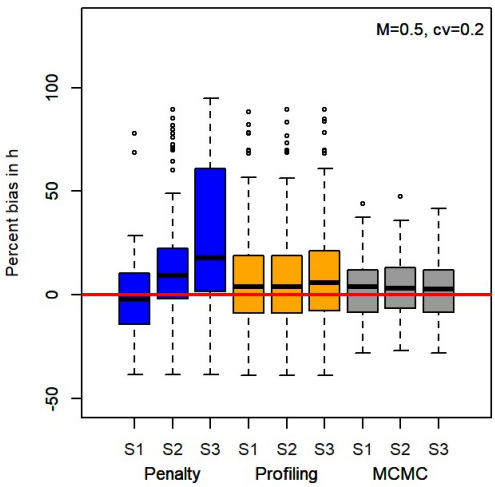

S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

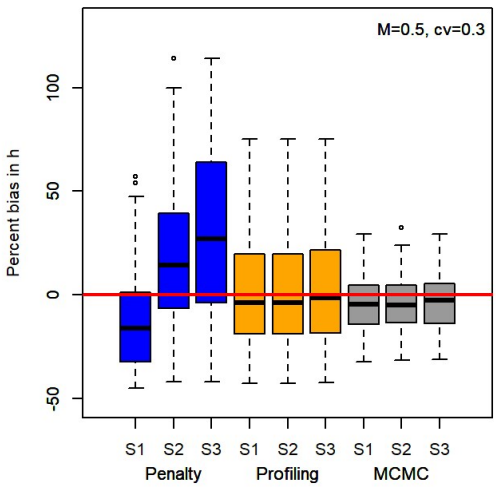

M=0.25, cv=0.1
Percent bias in h
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

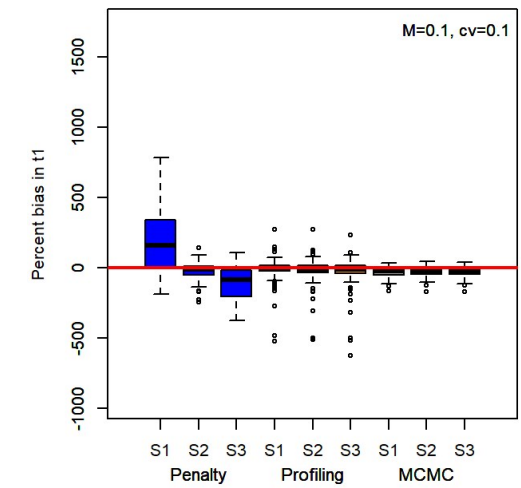

M=0.25, cv=0.2
Percent bias in h
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

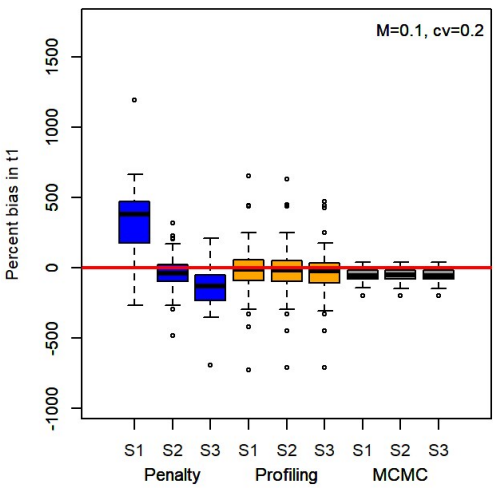

M=0.25, cv=0.3
Percent bias in h
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

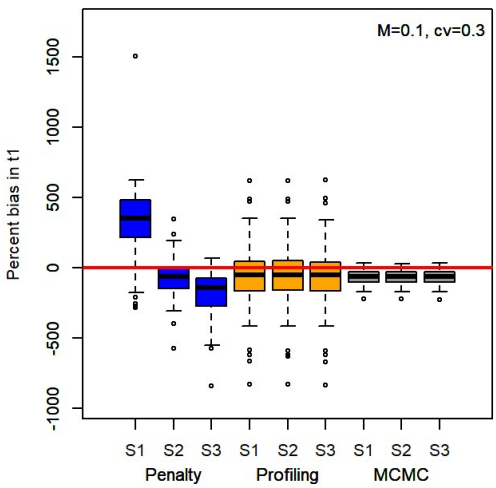

M=0.5, cv=0.1
Percent bias in h
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

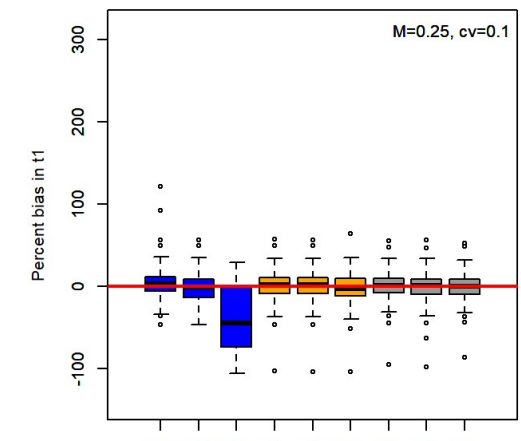

M=0.5, cv=0.2
Percent bias in h
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

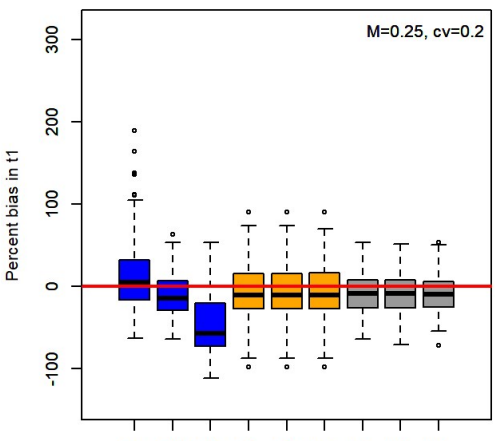

M=0.5, cv=0.3
Percent bias in h
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

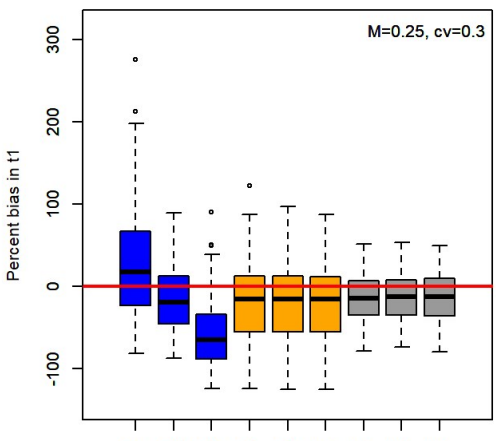

M=0.1, cv=0.1
Percent bias in t1
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

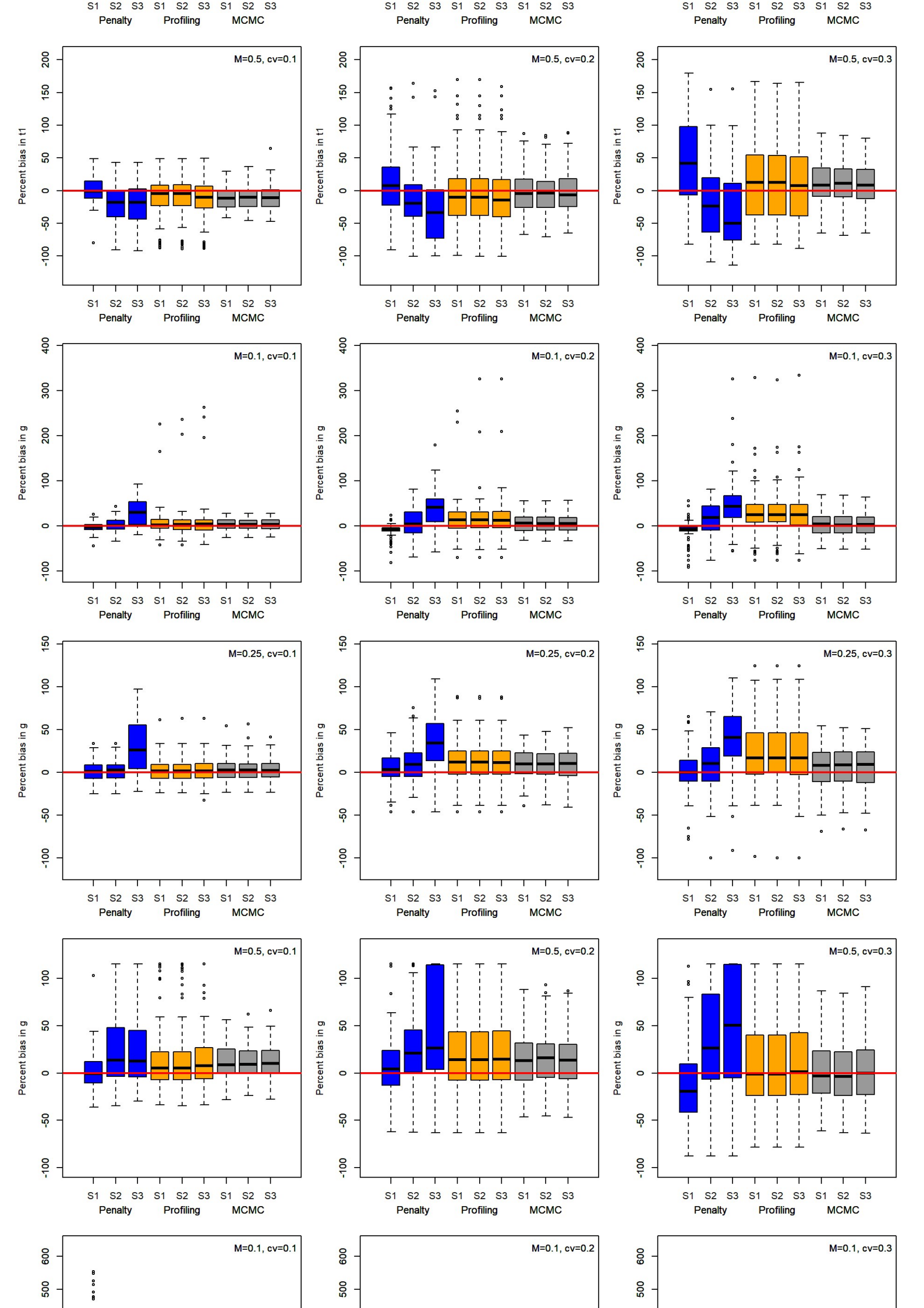

M=0.1, cv=0.2
Percent bias in t1
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

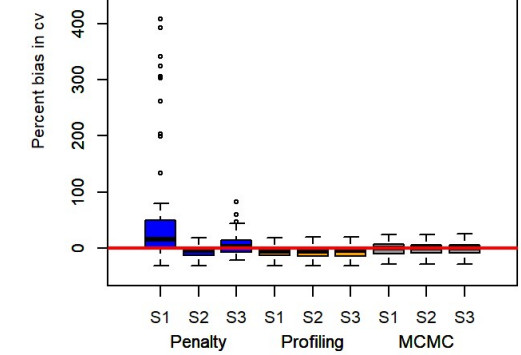

M=0.1, cv=0.3
Percent bias in t1
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

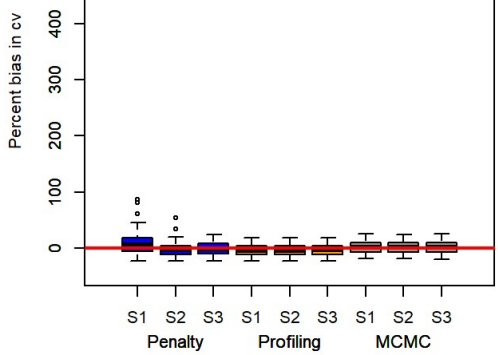

M=0.25, cv=0.1
Percent bias in t1

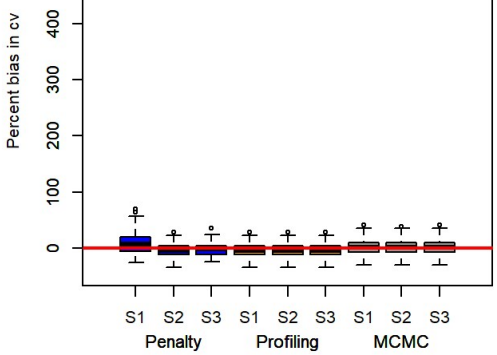

M=0.25, cv=0.2
Percent bias in t1

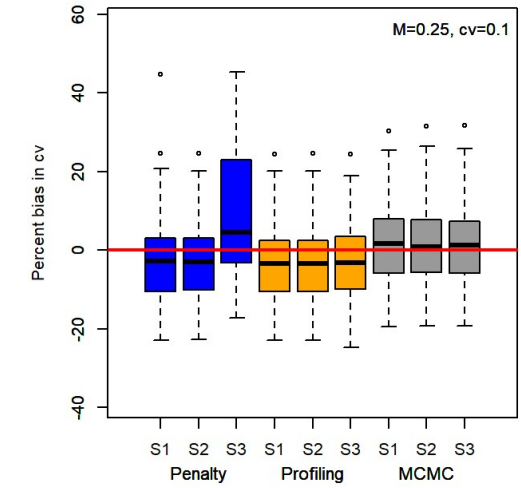

M=0.25, cv=0.3
Percent bias in t1

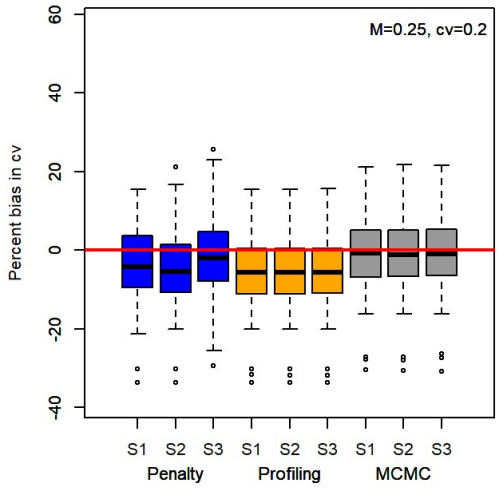

M=0.5, cv=0.1
M=0.5, cv=0.2
M=0.5, cv=0.3
M=0.1, cv=0.1
M=0.1, cv=0.2
M=0.1, cv=0.3
M=0.25, cv=0.1
M=0.25, cv=0.2
M=0.25, cv=0.3
M=0.5, cv=0.1
M=0.5, cv=0.2
M=0.5, cv=0.3
M=0.1, cv=0.1
M=0.1, cv=0.2
M=0.1, cv=0.3
Percent bias in t1
Percent bias in g
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

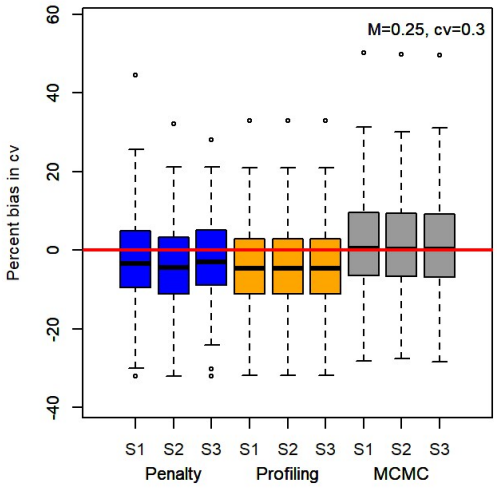

Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

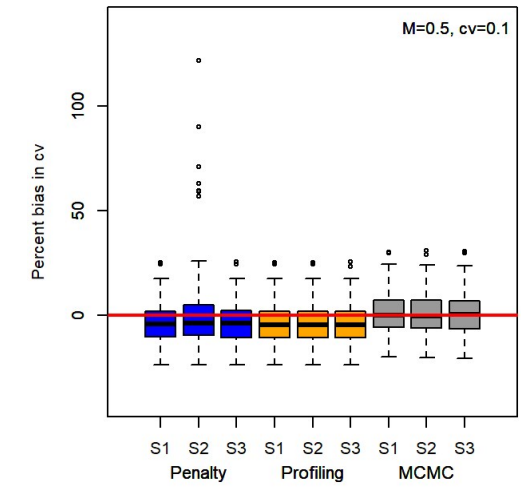

Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

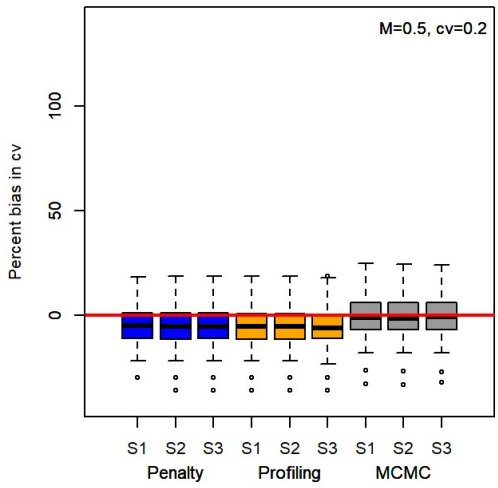

Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

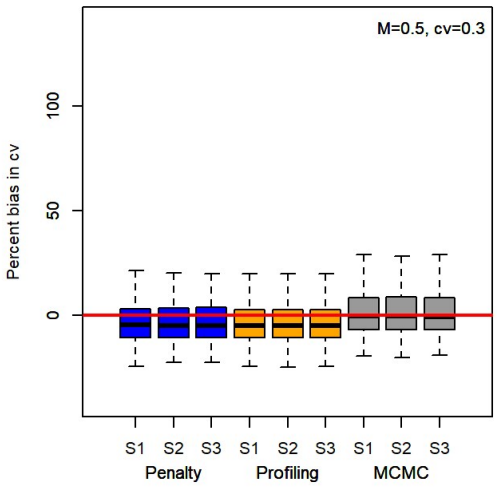

M=0.25, cv=0.1
Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

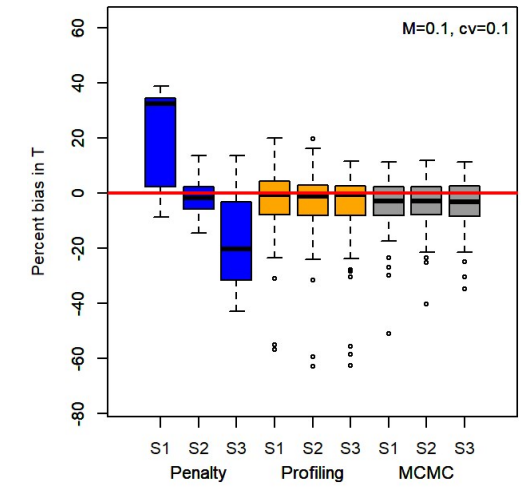

M=0.25, cv=0.2
Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

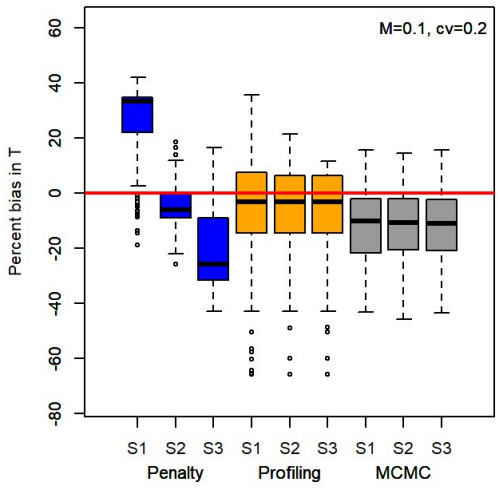

M=0.25, cv=0.3
Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

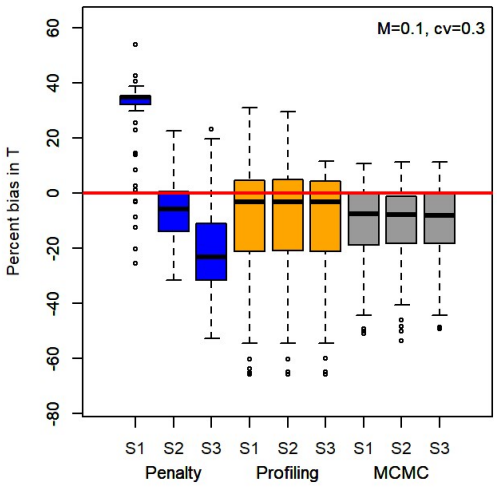

M=0.5, cv=0.1
Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

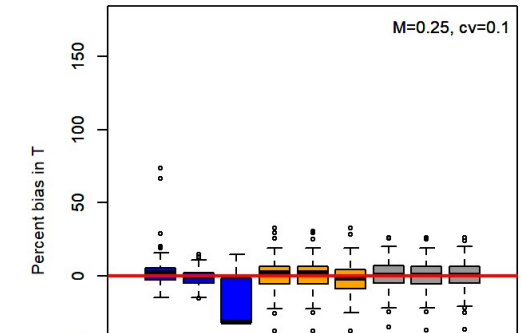

M=0.5, cv=0.2
Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

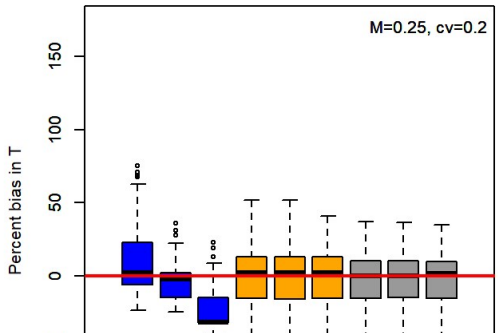

M=0.5, cv=0.3
Percent bias in cv
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

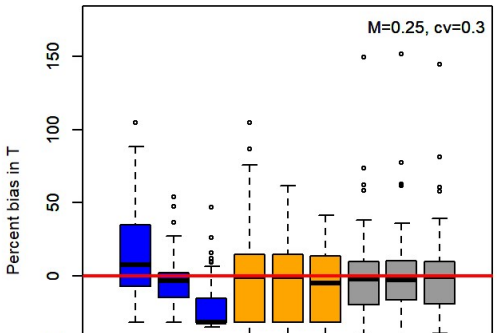

M=0.1, cv=0.1
Percent bias in T
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC

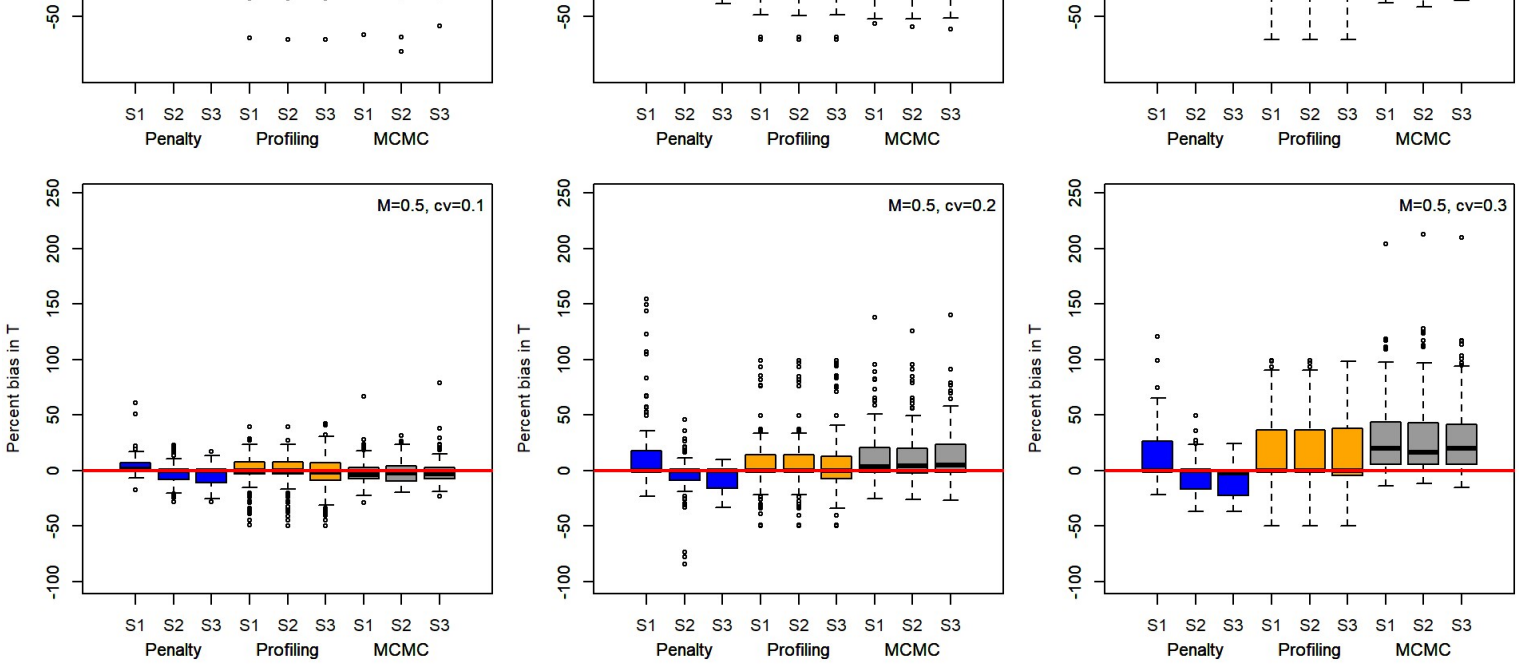

M=0.1, cv=0.2
Percent bias in T
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC


M=0.1, cv=0.3
Percent bias in T
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC


M=0.25, cv=0.1
Percent bias in T
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC


M=0.25, cv=0.2
Percent bias in T
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC


M=0.25, cv=0.3
Percent bias in T
S1 S2 S3 S1 S2 S3 S1 S2 S3
Penalty Profiling MCMC



Summary results

MCMC tended to perform the best in recovering life history parameters. The likelihood profiling also performed relatively well. The penalized likelihood approach performed well in some cases but was generally the most sensitive to starting values.

We note, however, that there were y=1 failures out of 900 successful iterations. The penalized likelihood approach accounted for most of these failures, highlighting that this approach can work but may not be as robust as the other approaches. Lastly, we note that the specification of the prior distributions can alter results and interpretation for the Bayesian MCMC approach. Having a uniform prior on T increased accuracy and precision compared to a normal prior on T in certain scenarios (largely when M and cv_l were both large).

References

Honsey, A.E., Staples, D.F. & Venturelli, P.A. (2016). Accurate estimates of age at maturity from the growth trajectories of fishes and other ectotherms. *Ecological Applications*, **27**, 182–192.

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Ono, K., Punt, A.E. & Rivot, E. (2012). Model performance analysis for bayesian biomass dynamics models using bias, precision and reliability metrics. *Fisheries Research*, **125-126**, 173–183.

Plummer, M. (2017). JAGS: A program for analysis of bayesian graphical models using gibbs sampling.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S8:
Estimating biphasic growth for multiple populations using MCMC

Kyle L. Wilson
The University of Calgary

October 5, 2017

Summary description of objectives:

This appendix is in support of Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

The following is an example application of the Lester biphasic model (Lester, Shuter & Abrams 2004) where the breakpoint T (age-at-maturity) is treated as unknown prior to model estimation. Specifically, we simulate multiple populations, and the life history of each population is related to that of other populations in a hierarchical manner (i.e., each life history parameter is random arising from a global distribution for that life history parameter). Known growth parameters are used to simulate the random, population-specific parameters. We generate size-at-age data given a population-specific, constant coefficient of variation in size-at-age.

We then fit the Lester biphasic model to these simulated data using a hierarchical framework in the Bayesian MCMC software JAGS. This framework allows us to estimate population-specific and 'global' (i.e., average across populations) growth parameters. We build the hierarchical model with vague priors (or hyperpriors), and we run JAGS from the R console using the `runjags` package and `run.jags()` function. JAGS must be installed independently prior to running this code (see 'http://mcmc-jags.sourceforge.net/'). All variables are treated as unknown at both the population and 'global' levels.

Our results summarize variables by their marginal posterior distribution. We then compare the central tendency of each parameter's posterior distribution to determine how well we recover the simulated 'true' parameters at the population and global levels.

Global functions

First, we will define a few global functions that will be used later.

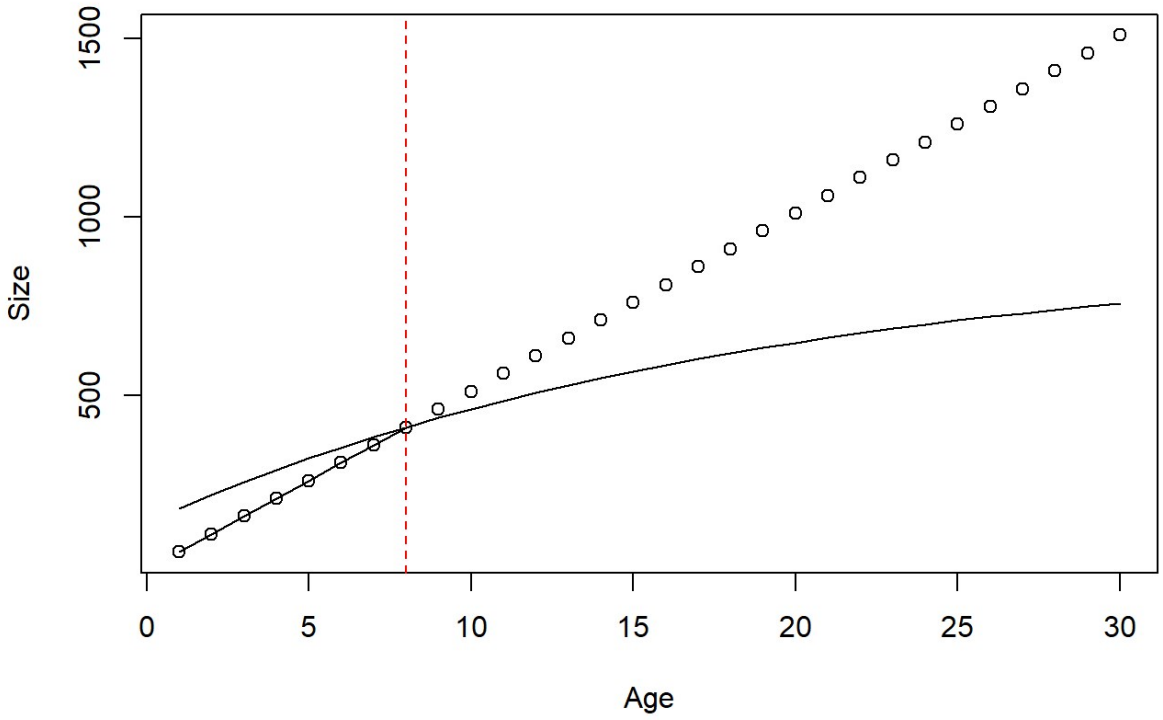
```
Corner_text <- function(text, location="topright") #function to write text to the corner of plots
{
  legend(location,legend=text, bty ="n", pch=NA)
}

get_beta <- function(mean,cv) #function that returns the alpha and beta shape parameters of a beta di
stribution, based on the mean and variation of a given beta distribution
{
  sd <- mean*cv
  alpha <- -((mean*(mean^2+sd^2-mean))/sd^2)
  beta <- alpha/mean-alpha
  return(list(alpha=alpha,beta=beta))
}

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}

rngList <- function(x,y){ #this function creates randomly 'jittered' starting values for each MCMC ch
ain
```

```
  lis <- lapply(x, lapply, length) #get the lower order dimensions of the list x
  names(lis) <- lapply(x, length) #get the names of those dimensions of the list x
  l_el <- length(names(lis)) #get the maximum number elements of the highest order dimensions of the
list x
  for(i in 1:(l_el-2)) #loop through those dimensions which need to be 'jittered'
  {
    x[[i]] <- x[[i]]*(1+runif(1,-0.15,0.15)) #jitter values of the list by +/- 15%
  }
  x[[l_el]] <- round(runif(1,1,100000),0) #have a random RNG seed for the MCMC chain
  return(x)
}
```

Define life history parameters

First, we must load the required libraries. We then start the simulation by specifying how many populations we want to model with `npop`. Then, we specify the leading life history parameters of the growth model: h, t_1, M, g, and cv. Consistent with the hierarchy of the model, we specify the among-population variation of each life history parameter (e.g., `h_cv` describes the coefficient of variation in the parameter h across all populations). The cv parameter for variation in length-at-age is 15%, consistent with typical observations among fishes Lorenzen (2016).

```
library(runjags)  #load, install, or require the 'runjags' library
library(rjags)
## Loading required package: coda
## Linked to JAGS 4.2.0
## Loaded modules: basemod,bugs
library(coda)
library(stats4)  #load the 'stats4' library
library(corrplot)

nPop <- 10  ## how many populations are there?

ages <- 1:30  #create an integer sequence of ages

Tmat <- 8  # age of maturity
Tmat_cv <- 0.1  # what is the variation in maturity across populations?

h <- 50  # somatic growth in millimeters per year
h_cv <- 0.15  # what is the variation in growth rate across populations?

t1 <- -0.2  #age when size=0 for the juvenile phase
t1_cv <- 0.1  # what is the variation in age when size=0 across populations?

M <- 0.15  #Natural mortality for the population
g <- 1.18 * (1 - exp(-M))  # proportion of energy in adult phase allocated to reproduction per year
g_cv <- 0.01  # what is the variation in g across populations?

cv <- 0.15  # coefficient of variation in size-at-age
sizeCV <- 1e-09  # what is the variation in the CV parameter across populations?
# i.e., are some populations more variable in size-at-age than others?

linf <- 3 * h/g  # convert to VBGF L-infinity
vbk <- log(1 + g/3)  # convert to VBGF kappa
t0 <- Tmat + log(1 - g * (Tmat - t1)/3)/log(1 + g/3)  #convert to VBGF t0

lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages < Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a fish
is allocating surplus energy

plot(ages, lena_phase1, ylab = "Size", xlab = "Age")
lines(ages, lena_phase2)
lines(ages, biphasic)
abline(v = Tmat, col = "red", lty = 2)  #plot where maturity occurs
```



Next, we generate the population-specific life history parameters, which arise as a random variable from that parameter's distribution. For instance, for h:

$$h_i \sim N(\mu, \sigma)$$

with mean μ and variance σ of this distribution:

$$\mu = h$$

$$\sigma = h * cv_{\mathrm{h}}$$

We set the random number generator to 100 so that our results are repeatable: `set.seed(100)`.

```
set.seed(100)
h_i <- rnorm(nPop,h,h*h_cv) # growth rate, h, for population i is random arising from a normal distri
bution with mean h and standard deviation of h*h_cv
Tmat_i <- rnorm(nPop,Tmat,Tmat*Tmat_cv)
t1_i <- rnorm(nPop,t1,abs(t1*t1_cv))
g_i <- rbeta(nPop,get_beta(g,g_cv)$alpha,get_beta(g,g_cv)$beta) # grab the shape parameters of a beta
 distribution based on its mean and variance
ifelse(all(g_i < 3/(Tmat_i-t1_i)), # there is a life-hisory constraint on the parameter g
       g_i <- g_i,
       g_i <- rbeta(nPop,get_beta(g,g_cv)$alpha,get_beta(g,g_cv)$beta))
## [1] 0.1641956
cv_i <- rnorm(nPop,cv,cv*sizeCV)

true.par <- list(h=h,T50=Tmat,t1=t1,g=g,sizeCV=cv,
                 h_pop=h_i,T50_pop=Tmat_i,t1_pop=t1_i,g_pop=g_i,
                 h_cv=h_cv,T50_cv=Tmat_cv,t1_cv=t1_cv,g_cv=g_cv)
```

Simulating the data

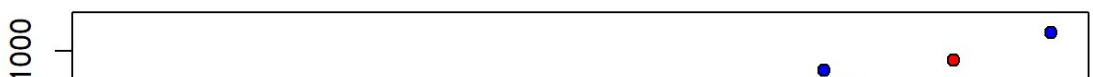
Next, we make an empty data object which we later fill with our population-specific, randomly generated length-at-age data. We use M to generate each population's survivorship curve. This allows us to simulate more realistic age-structures (by multipltying M by selectivity). We generate data using a random normal distribution (realized parameter values and model fit quality will change due to randomness) using the `rnorm()` function. Sample sizes expected for each age should be realistic for fisheries data and are determined from a function of gear selectivity and natural mortality (or survivorship) arising from a multinomial process using the `rmultinom()` function.

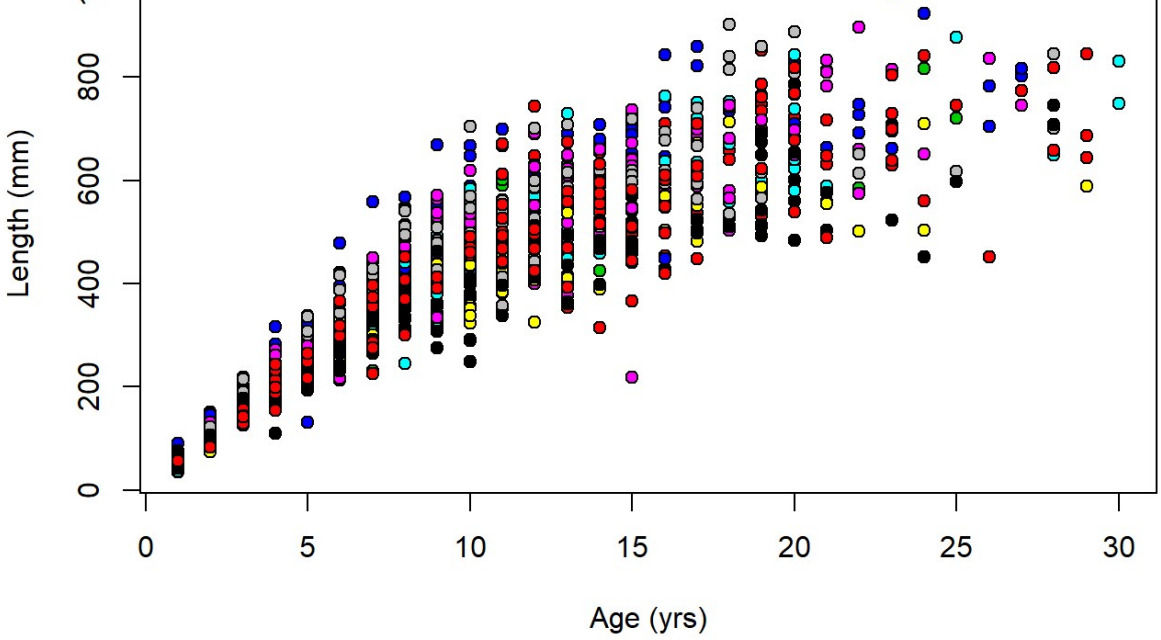
```
dataPop <- NULL  # make an empty object that we will fill in later
for (i in 1:nPop) {
```

```
    surv <- rep(NA, length(ages))  # create an empty vector
    surv[1] <- 1
    for (j in 2:max(ages)) {
        surv[j] <- surv[j - 1] * exp(log(((g_i[i]/1.18) - 1)/-1))
    }  #survivorship from discrete annual survival
    gearA50 <- Tmat_i[i]  # induce a gear selectivity that inflects at T
    gearSlope <- -0.3  # define the slope of selectivity
    select <- 1/(1 + exp(gearSlope * (ages - gearA50)))  # the average selectivity curve


    linf <- 3 * h_i[i]/g_i[i]  # conversion for the VBGF L-infinity
    vbk <- log(1 + g_i[i]/3)  # conversion for the VBGF parameter kappa
    t0 <- Tmat_i[i] + log(1 - g_i[i] * (Tmat_i[i] - t1_i[i])/3)/log(1 + g_i[i]/3)  #conversion for th
e VBGF parameter t0
    lena_juv <- h_i[i] * (ages - t1_i[i])  # length-at-age for phase 1
    lena_adult <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
    mean_size <- ifelse(ages <= Tmat_i[i], lena_juv, lena_adult)  #if-else statement for which phase
a fish is allocating surplus energy

    SampSize <- 10000
    maxSamp <- as.vector(rmultinom(1, prob = surv * select, size = SampSize))  # whats the maximum nu
mber of observable samples for an age group in a population?
    # surv*select is the probability of a fish surviving a certain ageand being
    # sampled

    mean.samp <- as.data.frame(cbind(mean_size, maxSamp))  #make matrix of mean lengths-at-age and sa
mple sizes
    mlen <- rep(mean.samp[, 1], mean.samp[, 2])  #repeat each mean 'sample size' number of times
    ageData <- rep(ages, mean.samp[, 2])  #repeat each age 'sample size' number of times
    lengths <- sapply(mlen, function(x) rnorm(1, mean = x, sd = x * cv))  #generate random normal len
gth data using means & cv error
    Data <- data.frame(cbind(ageData, lengths, rep(i, length(ageData))), row.names = NULL)  #bind vec
tors into age and length matrix, covert to data frame
    colnames(Data) <- c("Age", "Size", "Pop_Num")  # re-name the columns
    Data <- Data[sample(nrow(Data), size = round(runif(1, 40, 200)), replace = F),
        ]  #draw a random sample from the population -- total sample size can be adjusted
    rownames(Data) <- c()
    dataPop <- rbind(dataPop, Data)
}

rownames(dataPop) <- c()
```

Next, we plot the noisy data, with colors assigned to the data coming from each population. We can also take a quick look at some of the data to visualize the data structure.

```
plot(dataPop$Age, dataPop$Size, bg = dataPop$Pop_Num, pch = 21, xlab = "Age (yrs)",
    ylab = "Length (mm)")

dataPop[sample(1:nrow(dataPop), 10), ]
##       Age      Size Pop_Num
## 31     10 479.36465       1
## 1082   11 500.49513      10
## 733     4 174.35544       7
## 860     9 484.72654       8
## 925     6 241.37994       9
## 647    19 580.77590       6
## 423     1  75.94291       4
## 222    10 472.60617       2
## 590    18 741.59461       6
## 954    18 512.28408       9
```





Write the hierarchical model in JAGS

The following code is written in the JAGS language (Plummer 2017). JAGS is fed a list associated with the data. Each data point has an population identification index.

The JAGS model loops through each sample, from *i* to *Nfish*, and determines the contribution of the predicted size of fish *i* to the log-posterior density, which is the sum of the log-likelihood and the log-prior. The predicted growth follows the analytical model from the equations in Lester et al. (2004). There is an if-else statement that determines if the predicted size-at-age of fish *i* comes from the juvenile phase or the adult phase. The priors for each population's life history trait e.g., h for population *j*, comes from a global hyper-prior for h representing the average across all populations. Modifications to this code will need to use JAGS syntax and not R syntax.

```
model <- "model {
#run through all the individual fish
for(i in 1:Nfish) {
  size[i] ~ dnorm(pred[i],1/(pred[i]*size.cv)^2)T(0,) # one variability across all populations

  #predict growth for each fish
  juv[i] <- h_pop[Pop[i]]*(age[i]-t1_pop[Pop[i]]) # predicted growth for fish i for the juvenile phas
e

  # below is the predicted growth for fish i for the adult phase which follows converting Lester et a
l. (2004) equations into the von Bertalanffy growth function

  adult[i] <- (3*h_pop[Pop[i]]/g_pop[Pop[i]])*(1-exp(-(log(1+g_pop[Pop[i]]/3))*(age[i]-(T50_pop[Pop[i
]]+log(1-g_pop[Pop[i]]*(T50_pop[Pop[i]]-t1_pop[Pop[i]])/3)/log(1+g_pop[Pop[i]]/3)))))
  pred[i] <- ifelse(age[i]<=T50_pop[Pop[i]],juv[i],adult[i]) # does the age of fish i exceed the matu
rity predicted for its population?

} # end the calculation of the likelihood


# priors by population
for(j in 1:Npop) {
  # normal priors for population j
  T50_pop[j] ~ dnorm(T50,tau.T50)T(0,)
  h_pop[j] ~ dnorm(h,tau.h)T(0,)
  t1_pop[j] ~ dnorm(t1, tau.t1)

  # Beta distribution for the reproductive investment
  #(bounded between 0 and stochastic upper limit)

  g_pop[j] ~ dbeta(beta_alpha,beta_beta)T(,3/(T50_pop[j]-t1_pop[j]))

} # end the priors for each trait by population
```

```
# hyper priors on life history traits

T50 ~ dnorm(10,1e-7)T(0,)
h ~ dnorm(40,1e-7)T(0,)
t1 ~ dnorm(0,1e-2)
g ~ dgamma(0.001,0.001)T(,3/(T50-t1))

# below are the half-t priors on variance parameters

T50_cv ~ dhalfcauchy(10)
h_cv ~ dhalfcauchy(10)
g_cv ~ dhalfcauchy(10)
t1_cv ~ dhalfcauchy(10)
size.cv ~ dhalfcauchy(10)

# conversion to JAGS precision parameters

tau.T50 <- 1/(T50*T50_cv)^2
tau.h <- 1/(h*h_cv)^2
tau.t1 <- 1/(t1*t1_cv)^2

# re-parameterize the parameters of the beta

g_var <- (g*g_cv)^2
beta_alpha <- -g*(g_var+g^2-g)/g_var
beta_beta <- beta_alpha/g-beta_alpha
}"
```

Let's quickly look at how many samples we have for each of the populations. Then, we will compile the data into a usable list, and pass JAGS some initial starting values for each of the MCMC chains that we will run.

```
print(table(dataPop$Pop_Num))
##
##   1   2   3   4   5   6   7   8   9  10
##  98 147  71 135 118 161  77 104  81  96
data <- list(Nfish = length(dataPop$Age), Npop = length(unique(dataPop$Pop_Num)),
    age = dataPop$Age, size = dataPop$Size, Pop = dataPop$Pop_Num)
# the above list compiles the noisy data from the dataPop dataframe into 3
# vectors for age, size, and Pop (numerically, which population does each
# row correspond to?)

inits1 <- list(h = h, T50 = Tmat, g = g, t1 = t1, h_pop = rep(h, nPop), T50_pop = rep(Tmat,
    nPop), t1_pop = rep(t1, nPop), g_pop = rep(g, nPop), size.cv = cv, h_cv = h_cv,
    t1_cv = t1_cv, g_cv = g_cv, T50_cv = Tmat_cv, .RNG.name = "base::Wichmann-Hill",
    .RNG.seed = 735)

# initial estimates of each parameter must be provided. RNG is random number
# generators for that chain

inits2 <- inits3 <- inits4 <- inits1
inits2 <- rngList(inits2, inits1)  # jitter chain 2, based on values of chain 1
inits3 <- rngList(inits3, inits1)  # jitter chain 3, based on values of chain 1
inits4 <- rngList(inits4, inits1)  # jitter chain 4, based on values of chain 1

inits <- list(inits1, inits2, inits3, inits4)  # compile all initial values into one list

mon_names <- c(names(inits3)[-c(length(inits3), length(inits3) - 1)])  # create the stochastic nodes
to be monitored
```

Run the MCMC chains in JAGS

Our next step is to set how many posterior samples we want, the length of the burn-in period and adaptation period, and the thinning rate. Our thinning rate is a somewhat high value of 10-30 - although this slows us down, the expected large correlations between parameters and the Markovian sampling process can lead to high autocorrelation among the consecutive posterior samples. To gain

more independent samples, we run each chain longer by increasing our thinning rate while still only taking `Nsamp` number of samples.

We then use the `run.jags()` function to call JAGS to run our model (Denwood 2016). We specify the modules we run, and some other parameters internal to `run.jags()`, like the `method` and `modules`. One can use the `rjparallel` method to parallelize the MCMC estimation and speed up JAGS model run times.

The `summary(results)` command reports some quick MCMC diagnostic tests to assess whether the posterior has converged on a stable distribution. The potential scale reduction factor (also called the Gelman-Rubin test) and the effective number of sample sizes are all reported in this summary output. The library `coda` offers more options for MCMC diagnostics.

```
Nsamp <- 1000  # how many posterior samples does each chain need to get, after thinning and burin-in
and adaptation?
thin_rt <- 15  # place some sort of thinning rate?
burnins <- 0.75 * round(Nsamp * thin_rt, 0)  # how long is the burnin, this bases it on the number of
 total posterior draws?
adaptin <- round(0.4 * burnins, 0)

a <- proc.time()
results <- run.jags(model = model, monitor = mon_names, data = data, n.chains = 4,
    method = "rjags", inits = inits, plots = F, silent.jag = F, modules = c("bugs",
        "glm", "dic"), sample = Nsamp, adapt = adaptin, burnin = burnins, thin = thin_rt,
    summarise = F)
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 4500 iterations...
## Burning in the model for 11250 iterations...
## Running the model for 15000 iterations...
## Simulation complete
## Finished running the simulation
b <- (proc.time() - a)

print((b[3]/60)/60)  # how long it took in hours
##   elapsed
## 0.6090278
print((b[3]/(4 * (Nsamp * thin_rt + burnins + adaptin))))  # how long it took in seconds per iteratio
n
##   elapsed
## 0.0178252
sum_results <- summary(results)
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 49 variables....
print(sum_results[, c(1, 2, 3, 8, 9, 10)])
##                  Lower95      Median      Upper95 MC%ofSD SSeff
## h            4.649384e+01 49.94981602 53.455852508     1.9  2887
## T50          7.084469e+00  8.03436030  8.887723589     2.5  1630
## g            1.506883e-01  0.16269884  0.174080574     3.0  1093
## t1          -2.657511e-01 -0.17287370 -0.092649233     3.2   973
## h_pop[1]     4.433244e+01 46.34520920 48.443345887     2.1  2174
## h_pop[2]     4.782944e+01 49.97789393 52.502437503     2.7  1401
## h_pop[3]     4.818523e+01 50.56881488 53.692103522     2.4  1727
## h_pop[4]     5.519325e+01 57.40818322 59.846999538     2.2  1994
## h_pop[5]     4.810679e+01 50.39092665 52.543325733     2.2  2051
## h_pop[6]     5.002176e+01 52.24710809 54.516856597     2.3  1849
## h_pop[7]     4.321259e+01 46.18309012 49.706462504     2.9  1201
## h_pop[8]     5.225525e+01 54.45494270 56.843198182     2.1  2189
## h_pop[9]     4.084219e+01 43.06260933 45.578116794     2.3  1912
## h_pop[10]    4.613312e+01 48.81896176 51.464684902     2.4  1693
## T50_pop[1]   6.927029e+00  8.01874330  8.989006624     2.0  2442
## T50_pop[2]   6.406775e+00  7.85513481  9.088500295     2.3  1862
## T50_pop[3]   6.750745e+00  8.20456066  9.913822463     2.1  2167
## T50_pop[4]   7.313071e+00  8.14484239  9.240209933     2.0  2497
## T50_pop[5]   7.124654e+00  8.20058427  9.577667310     1.8  3059
## T50_pop[6]   7.133746e+00  8.14334636  9.176045140     1.9  2761
## T50_pop[7]   4.720975e+00  7.37203019  8.774680405     3.3   918
## T50_pop[8]   7.173993e+00  8.12860853  9.328926643     1.8  3235
## T50_pop[9]   6.069120e+00  7.76490985  9.016270856     2.3  1866
```

```
## T50_pop[10]  7.411011e+00  8.57837291 10.888082651     2.6  1526
## t1_pop[1]   -2.909656e-01 -0.17010220 -0.040943359     2.3  1961
## t1_pop[2]   -2.691451e-01 -0.15633691 -0.039250862     2.6  1509
## t1_pop[3]   -2.612393e-01 -0.14833339  0.007433762     2.5  1583
## t1_pop[4]   -3.263393e-01 -0.20109210 -0.109670735     2.4  1676
## t1_pop[5]   -2.696089e-01 -0.17009152 -0.055268114     2.4  1807
## t1_pop[6]   -3.311104e-01 -0.20487295 -0.103293270     2.5  1657
## t1_pop[7]   -4.127055e-01 -0.22678106 -0.111059476     2.7  1351
## t1_pop[8]   -2.776487e-01 -0.15835388 -0.032633946     2.3  1947
## t1_pop[9]   -3.103481e-01 -0.17349631 -0.046944556     2.3  1869
## t1_pop[10]  -2.654800e-01 -0.15361313  0.009623031     2.4  1749
## g_pop[1]     1.436476e-01  0.16252290  0.178410689     2.4  1763
## g_pop[2]     1.444304e-01  0.16074914  0.175946243     2.6  1460
## g_pop[3]     1.460852e-01  0.16337181  0.183323811     2.3  1819
## g_pop[4]     1.488560e-01  0.16422766  0.179794586     2.6  1426
## g_pop[5]     1.458651e-01  0.16167129  0.177045018     2.6  1461
## g_pop[6]     1.483595e-01  0.16373907  0.180635449     2.4  1669
## g_pop[7]     1.470679e-01  0.16451806  0.185831512     2.4  1713
## g_pop[8]     1.463945e-01  0.16266911  0.177903577     2.4  1712
## g_pop[9]     1.459918e-01  0.16336353  0.180810626     2.4  1786
## g_pop[10]    1.400414e-01  0.15966047  0.174520626     2.6  1501
## size.cv      1.388802e-01  0.14467780  0.150828968     1.6  4132
## h_cv         5.487548e-02  0.09481096  0.163354092     1.6  3723
## t1_cv        3.648682e-04  0.32138155  1.120533613     3.7   749
## g_cv         2.657749e-05  0.03392223  0.110656255     3.2   988
## T50_cv       4.968083e-04  0.08194528  0.245066608     3.4   856
##                   AC.150
## h            0.0053903955
## T50          0.0199783751
## g           -0.0134837172
## t1           0.0600311714
## h_pop[1]     0.0139598803
## h_pop[2]     0.0320122995
## h_pop[3]     0.0226210350
## h_pop[4]     0.0100645047
## h_pop[5]     0.0223560200
## h_pop[6]     0.0071626210
## h_pop[7]     0.0527786803
## h_pop[8]     0.0221376731
## h_pop[9]     0.0301143755
## h_pop[10]    0.0188682503
## T50_pop[1]   0.0034274450
## T50_pop[2]   0.0527898465
## T50_pop[3]   0.0286888204
## T50_pop[4]   0.0027694684
## T50_pop[5]   0.0363535902
## T50_pop[6]   0.0095321407
## T50_pop[7]   0.0604083388
## T50_pop[8]  -0.0018032233
## T50_pop[9]  -0.0028626945
## T50_pop[10]  0.0261059254
## t1_pop[1]   -0.0020340174
## t1_pop[2]   -0.0059619865
## t1_pop[3]    0.0194030722
## t1_pop[4]    0.0035993777
## t1_pop[5]    0.0066138003
## t1_pop[6]   -0.0018338452
## t1_pop[7]    0.0475078022
## t1_pop[8]    0.0209770936
## t1_pop[9]    0.0010002816
## t1_pop[10]  -0.0097524747
## g_pop[1]     0.0083664084
## g_pop[2]     0.0204205103
## g_pop[3]    -0.0002636327
## g_pop[4]    -0.0048763564
## g_pop[5]     0.0160461782
## g_pop[6]     0.0051757984
## g_pop[7]    -0.0207105312
```

```
## g_pop[8]      0.0013729591
## g_pop[9]      0.0085697605
## g_pop[10]     0.0390299134
## size.cv       0.0131282845
## h_cv          0.0074529744
## t1_cv         0.0480027388
## g_cv          0.0252468096
## T50_cv        0.0646539366
res.corr <- extract.runjags(add.summary(results), "crosscorr")  # extract the cross-correlation matri
x
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 49 variables....
```

Store the model results

Next, we store the results as an `mcmc.list` which could be used in `coda` for MCMC diagnostics (Plummer *et al.* 2006) (see Appendix S5), but here we just quickly coerce this into a `matrix` object for our purposes. We then use some `grep` functions to quickly grab the posterior distributions of monitored parameters associated with specific growth model parameters (e.g., h, h_i, and cv_{h}.

We can do some further diagnostics of the model by evaluating how biased the estimates of the growth parameters were in comparison to the simulated 'true' parameter values. To do this, we will use percent bias:

$$Bias = ((\theta_i - \hat{\theta}_i)/\theta_i) * 100$$

where θ_i is a life history parameter of interest, e.g., juvenile growth rate h_i.

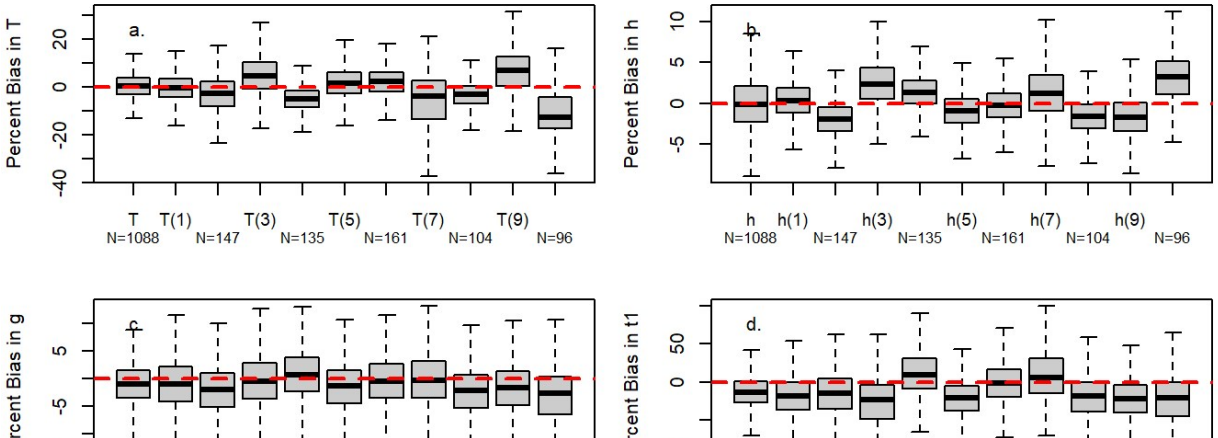
```
TheRes <- as.mcmc.list(results, vars = mon_names)
TheRes <- as.matrix(TheRes)

Ntot <- length(dataPop$Pop_Num)
Npop <- table(dataPop$Pop_Num)

h_plot <- cbind(TheRes[, grep("h", colnames(TheRes))])
Tmat_plot <- cbind(TheRes[, grep("T", colnames(TheRes))])
g_plot <- cbind(TheRes[, grep("g", colnames(TheRes))])
t1_plot <- cbind(TheRes[, grep("t1", colnames(TheRes))])
var_plot <- cbind(TheRes[, grep("cv", colnames(TheRes))])

hBias <- Tbias <- gBias <- t1Bias <- matrix(NA, ncol = ncol(h_plot) - 1, nrow = nrow(h_plot))
for (i in 1:(ncol(h_plot) - 1)) {
    hBias[, i] <- (h_plot[, i] - c(h, h_i)[i])/c(h, h_i)[i] * 100
    Tbias[, i] <- (Tmat_plot[, i] - c(Tmat, Tmat_i)[i])/c(Tmat, Tmat_i)[i] *
        100
    gBias[, i] <- (g_plot[, i] - c(g, g_i)[i])/c(g, g_i)[i] * 100
    t1Bias[, i] <- (t1_plot[, i] - c(t1, t1_i)[i])/c(t1, t1_i)[i] * 100
}

varBias <- matrix(NA, ncol = ncol(var_plot), nrow = nrow(var_plot))
for (i in 1:ncol(var_plot)) {
    varBias[, i] <- (var_plot[, i] - c(h_cv, Tmat_cv, g_cv, t1_cv, cv)[i])/c(h_cv,
        Tmat_cv, g_cv, t1_cv, cv)[i] * 100
}
```

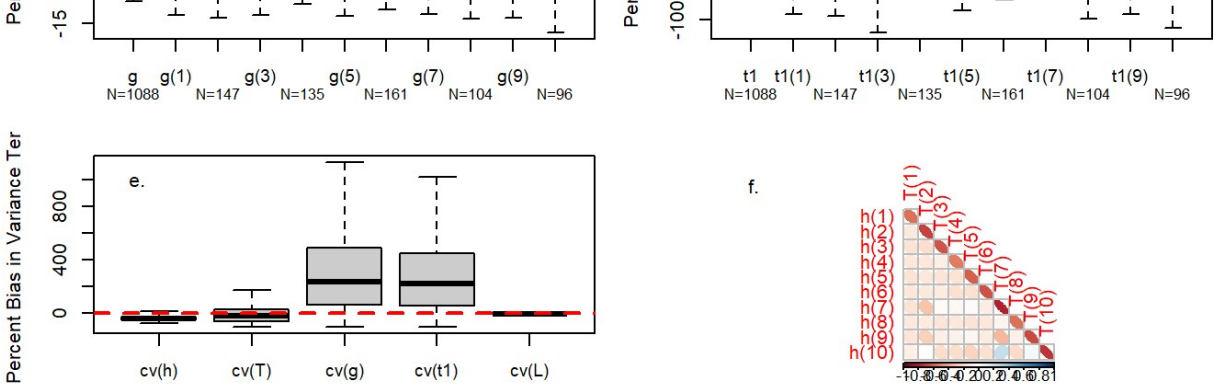
Percent bias and correlation plots

The code below generates a plot showing percent bias for each of the estimated parameters, along with some example correlations between the parameters T_i and h_i.

```
par(mfrow = c(1, 1))
par(mar = c(4, 4, 1, 1))
layout(matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, ncol = 2, byrow = TRUE))

boxplot(Tbias, ylab = "Percent Bias in T", col = "grey80", outline = FALSE,
```

```
    xaxt = "n")
axis(1, 1:(length(Tmat_i) + 1), c("T", paste("T(", 1:nPop, ")", sep = "")),
    line = 0, cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)

abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("a.", "topleft")

boxplot(hBias, ylab = "Percent Bias in h", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:(length(h_i) + 1), c("h", paste("h(", 1:nPop, ")", sep = "")), cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("b.", "topleft")

boxplot(gBias, ylab = "Percent Bias in g", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:(length(g_i) + 1), c("g", paste("g(", 1:nPop, ")", sep = "")), cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("c.", "topleft")

boxplot(t1Bias, ylab = "Percent Bias in t1", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:(length(t1_i) + 1), c("t1", paste("t1(", 1:nPop, ")", sep = "")),
    cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("d.", "topleft")

boxplot(varBias, ylab = "Percent Bias in Variance Terms", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:length(c(h_cv, Tmat_cv, g_cv, t1_cv, cv)), c("cv(h)", "cv(T)", "cv(g)",
    "cv(t1)", "cv(L)"), cex.axis = 0.9)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("e.", "topleft")

corr.mat <- res.corr[5:14, 15:24]
name1 <- name2 <- c()
for (i in 1:nPop) {
    name1[i] <- paste("h(", i, ")", sep = "")
    name2[i] <- paste("T(", i, ")", sep = "")
}
rownames(corr.mat) <- name1
colnames(corr.mat) <- name2
corrplot(corr.mat, is.corr = TRUE, method = "ellipse", mar = c(2, 4, 1, 0),
    type = "lower", diag = T)

Corner_text("f.", "topleft")
```

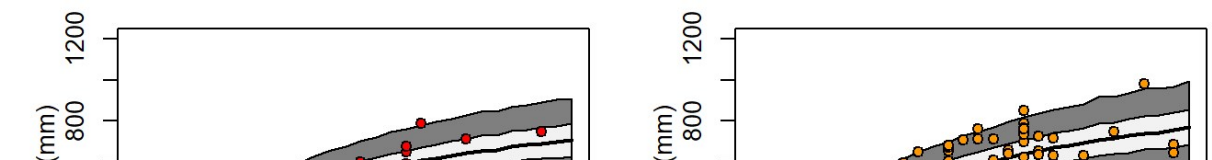


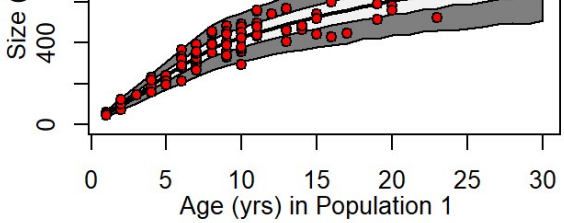


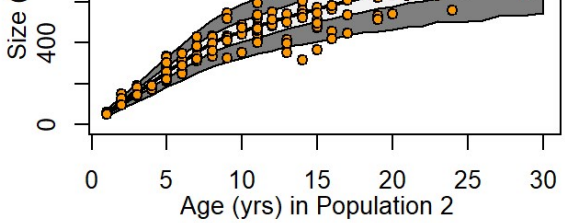
Posterior predictive checks

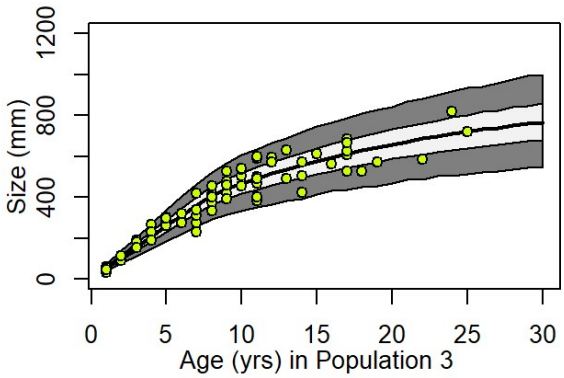
Next, we will run posterior predictive checks, which use the posterior distribution of the parameter estimates to re-generate a randomized posterior distribution of data, assuming the data arose according to the truncated normal distribution (as specified in the model above). We then overlay our observed data atop this distribution to check if there is any systematic bias (Gelman *et al.* (2013)).

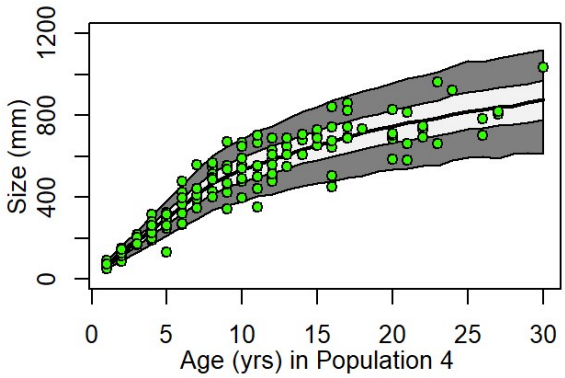
```
age_vec <- c(ages, rev(ages))
post_pred <- array(NA, dim = c(nrow(TheRes), length(ages), nPop))
# Above code creates an empty array to track size-at-age for population i
# for posterior draw j
par(mfrow = c(1, 1))
par(mar = c(5, 4, 1, 1))
layout(matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE))
for (i in 1:nPop) {
    subbed <- subset(dataPop, dataPop$Pop_Num == i)
    for (j in 1:nrow(TheRes)) {
        h_j <- TheRes[j, match(paste("h_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        g_j <- TheRes[j, match(paste("g_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        T50_j <- TheRes[j, match(paste("T50_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        t1_j <- TheRes[j, match(paste("t1_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        cv_j <- TheRes[j, match("size.cv", colnames(TheRes))]

        linf <- 3 * h_j/g_j  # conversion for the VBGF L-infinity
        vbk <- log(1 + g_j/3)  # conversion for the VBGF parameter kappa
        t0 <- T50_j + log(1 - g_j * (T50_j - t1_j)/3)/log(1 + g_j/3)  #conversion for the VBGF parame
ter t0

        juv_j <- h_j * (ages - t1_j)
        adult_j <- linf * (1 - exp(-vbk * (ages - t0)))
        pred_j <- ifelse(ages < T50_j, juv_j, adult_j)
        pred_j[pred_j < 0.001] <- 0.001
        post_pred[j, , i] <- rnorm(length(ages), pred_j, pred_j * cv_j)
    }
    quants <- t(apply(post_pred[, , i], 2, FUN = quantile, probs = c(0.025,
        0.225, 0.5, 0.775, 0.975)))
    plot(age_vec, c(quants[, 1], rev(quants[, 5])), type = "l", lwd = 2, col = NA,
        ylab = "", xlab = "", ylim = c(0, 1200))
    axis(1, at = median(c(0, max(ages))), paste("Age (yrs) in ", "Population ",
        i, sep = ""), tick = FALSE, line = 0.9)
    axis(2, at = 600, "Size (mm)", tick = FALSE, line = 1)
    polygon(age_vec, c(quants[, 1], rev(quants[, 5])), col = "grey50")
    polygon(age_vec, c(quants[, 2], rev(quants[, 4])), col = "grey95")
    lines(ages, quants[, 3], lwd = 2, col = "black")
    points(subbed$Age, subbed$Size, pch = 21, bg = rainbow(nPop)[i])
}
```

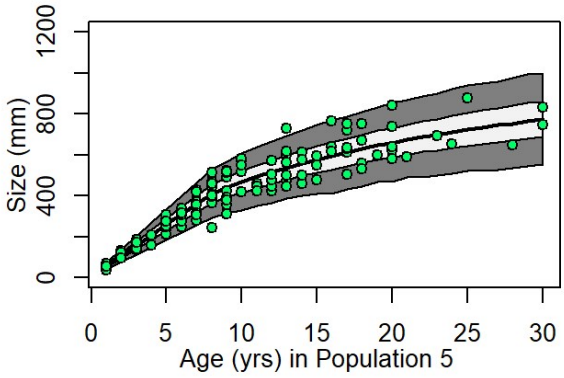


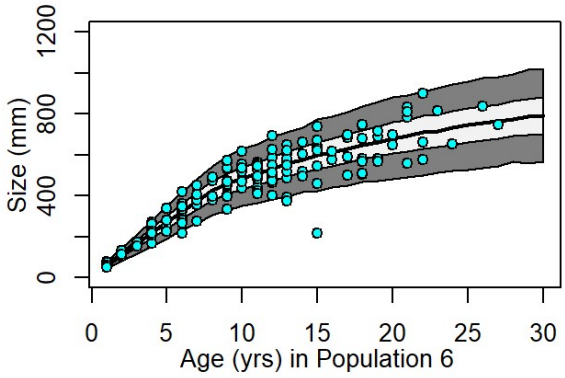

Size (mm)
0
400
Age (yrs) in Population 1

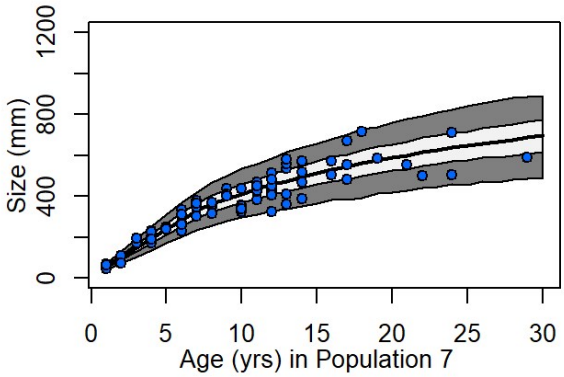

Size (mm)
0
400
Age (yrs) in Population 2

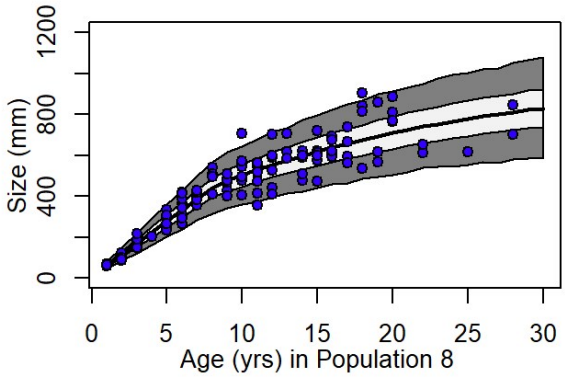

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 3

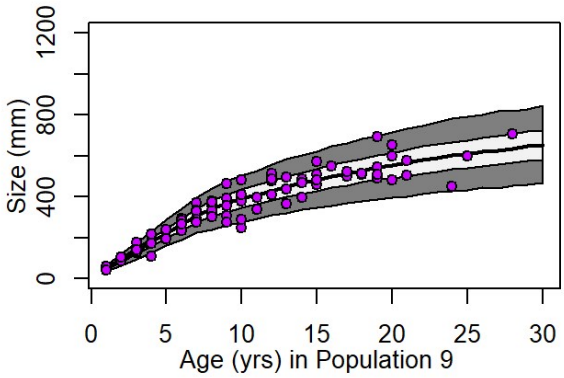

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 4


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 5


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 6


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 7


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 8


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 9


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 10

References

Denwood, M.J. (2016). runjags: An R package providing interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. *Journal of Statistical Software*, **71**, 1–25.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. & Rubin, D. (2013). *Bayesian Data Analysis*, 3rd edn. Chapman and Hall/CRC Press.

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Plummer, M. (2017). JAGS: A program for analysis of bayesian graphical models using gibbs sampling.

Plummer, M., Best, N., Cowles, K. & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for mcmc. *R News*, **6**, 7–11.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.

Appendix S9:
Estimating biphasic growth for multiple populations with an environmental trend using MCMC

Kyle L. Wilson
The University of Calgary

October 5, 2017

Summary description of objectives:

This appendix is in support of Wilson *et al.* (2017). The citation style language (csl) used herein is the methods-in-ecology-and-evolution.csl file which can be downloaded from https://github.com/citation-style-language/styles/blob/master/methods-in-ecology-and-evolution.csl and placed in the same directory as this .rmd file.

The following is an example application of the Lester biphasic model (Lester, Shuter & Abrams 2004) where the breakpoint T (age-at-maturity) is treated as unknown prior to model estimation. There is an environmental trend on growth (e.g., Helser & Lai (2004)). Hence, this serves as an example of how to estimate an environmental determinant on a life history trait of interest, while accounting for uncertainty in both the trait itself and the relationship with its environment.

Specifically, we simulate multiple populations, and the life history of each population is related to that of other populations in a hierarchical manner (i.e., each life history parameter is random arising from a global distribution for that life history parameter). Known growth parameters are used to simulate the random, population-specific parameters. We also simulate an environmental trend on the parameter h (juvenile growth rate) whereby growth increases across a gradient (labeled `enviro` in the code below). We generate size-at-age data given a population-specific, constant coefficient of variation in size-at-age.

We then fit the Lester biphasic model to these simulated data using a hierarchical framework in the Bayesian MCMC software JAGS. This framework allows us to estimate population-specific and 'global' (i.e., average across populations) growth parameters, as well as the slope of the environmental effect on h among populations. We build the hierarchical model with vague priors (or hyperpriors), and we run JAGS from the R console using the `runjags` package and `run.jags()` function. JAGS must be installed independently prior to running this code (see 'http://mcmc-jags.sourceforge.net/'). All variables are treated as unknown at both the population and 'global' levels.

Our results summarize variables by their marginal posterior distribution. We then compare the central tendency of each parameter's posterior distribution to determine how well we recover the simulated 'true' parameters at the population and global levels. Lastly, we evaluate our ability to detect the environmental trend in h.

Global functions

First, we will define a few global functions that will be used later.

```
Corner_text <- function(text, location="topright") #function to write text to the corner of plots
{
  legend(location,legend=text, bty ="n", pch=NA)
}

get_beta <- function(mean,cv) #function that returns the alpha and beta shape parameters of a beta di
stribution, based on the mean and variation of a given beta distribution
{
  sd <- mean*cv
  alpha <- -((mean*(mean^2+sd^2-mean))/sd^2)
  beta <- alpha/mean-alpha
  return(list(alpha=alpha,beta=beta))
}

panel.cor <- function(x, y, digits = 2, prefix = "", cex.cor, ...)
{
  usr <- par("usr"); on.exit(par(usr))
```

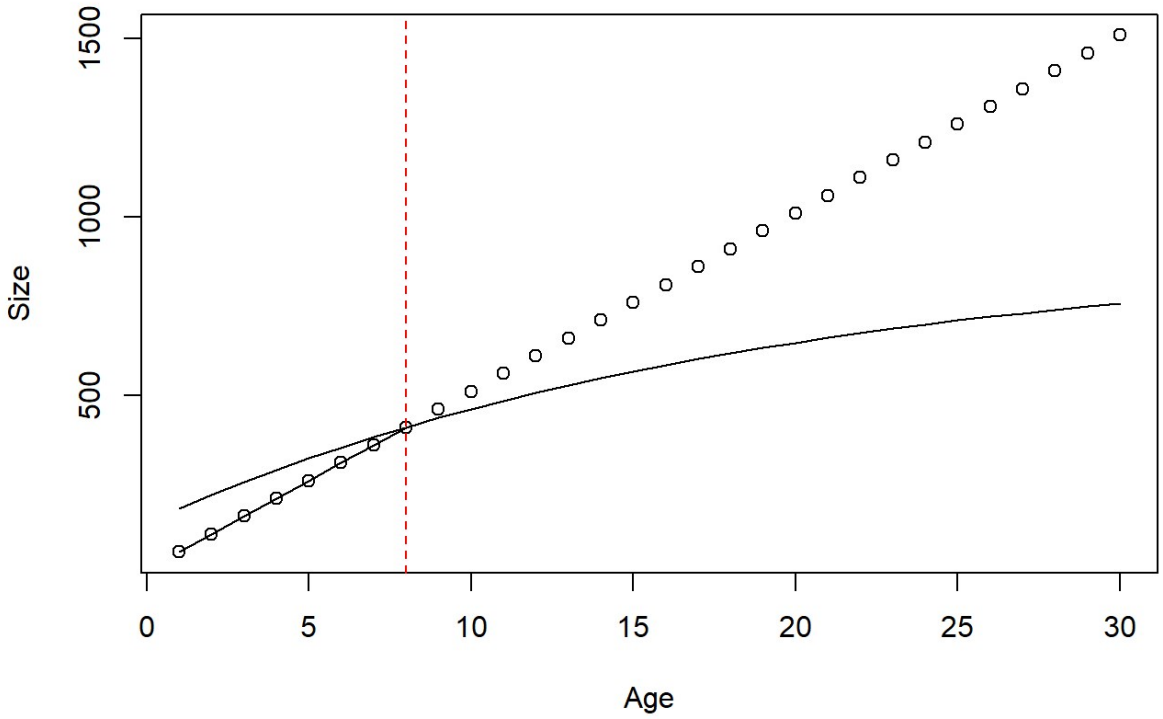
```
  par(usr = c(0, 1, 0, 1))
  r <- abs(cor(x, y))
  txt <- format(c(r, 0.123456789), digits = digits)[1]
  txt <- paste0(prefix, txt)
  if(missing(cex.cor)) cex.cor <- 0.8/strwidth(txt)
  text(0.5, 0.5, txt, cex = cex.cor * r)
}

rngList <- function(x,y){ #this function creates randomly 'jittered' starting values for each MCMC ch
ain
  lis <- lapply(x, lapply, length) #get the lower order dimensions of the list x
  names(lis) <- lapply(x, length) #get the names of those dimensions of the list x
  l_el <- length(names(lis)) #get the maximum number elements of the highest order dimensions of the
list x
  for(i in 1:(l_el-2)) #loop through those dimensions which need to be 'jittered'
  {
    x[[i]] <- x[[i]]*(1+runif(1,-0.15,0.15)) #jitter values of the list by +/- 15%
  }
  x[[l_el]] <- round(runif(1,1,100000),0) #have a random RNG seed for the MCMC chain
  return(x)
}
```

Define life history parameters

First, we must load the required libraries. We then start the simulation by specifying how many populations we want to model with `npop`. Then, we specify the leading life history parameters of the growth model: h, t_1, M, g, and cv. Consistent with the hierarchy of the model, we specify the among-population variation of each life history parameter (e.g., `h_cv` describes the coefficient of variation in the parameter h across all populations). The cv parameter for variation in length-at-age is 15%, consistent with typical observations among fishes Lorenzen (2016).

```
library(runjags)  #load, install, or require the 'runjags' library
library(rjags)
## Loading required package: coda
## Linked to JAGS 4.2.0
## Loaded modules: basemod,bugs
library(coda)
library(stats4)  #load the 'stats4' library
library(corrplot)

nPop <- 10  ## how many populations are there?

ages <- 1:30  #create an integer sequence of ages

Tmat <- 8  # age of maturity
Tmat_cv <- 0.1  # what is the variation in maturity across populations?

h <- 50  # somatic growth in millimeters per year
h_cv <- 0.15  # what is the variation in growth rate across populations?

t1 <- -0.2  #age when size=0 for the juvenile phase
t1_cv <- 0.1  # what is the variation in age when size=0 across populations?

M <- 0.15  #Natural mortality for the population
g <- 1.18 * (1 - exp(-M))  # proportion of energy in adult phase allocated to reproduction per year
g_cv <- 0.01  # what is the variation in g across populations?

cv <- 0.15  # coefficient of variation in size-at-age
sizeCV <- 1e-09  # what is the variation in the CV parameter across populations?
# i.e., are some populations more variable in size-at-age than others?

beta1 <- 0.15  # define the slope between growth rate and environment
enviro <- seq(from = 0, to = 100, length = nPop)  # define the value for the environmental gradient
std.enviro <- (enviro - mean(enviro))  # center the gradient on the mean

linf <- 3 * h/g  # convert to VBGF L-infinity
vbk <- log(1 + g/3)  # convert to VBGF kappa
```

```
t0 <- Tmat + log(1 - g * (Tmat - t1)/3)/log(1 + g/3)  #convert to VBGF t0

lena_phase1 <- h * (ages - t1)  # length-at-age for phase 1
lena_phase2 <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
biphasic <- ifelse(ages < Tmat, lena_phase1, lena_phase2)  #if-else statement for which phase a fish is allocating surplus energy

plot(ages, lena_phase1, ylab = "Size", xlab = "Age")
lines(ages, lena_phase2)
lines(ages, biphasic)
abline(v = Tmat, col = "red", lty = 2)  #plot where maturity occurs
```



Next, we generate the population-specific life history parameters, which arise as a random variable from that parameter's distribution. We set the random number generator to 100 so that our results are repeatable: `set.seed(100)`. Note that the population-specific juvenile growth rate h will increase as the environmental gradient increases with a slope of β, but there is still noise in the relationship:

$$h_i \sim N(\mu, \sigma)$$

with mean μ and variance σ of this distribution defined as

$$\mu = h + \beta * x_i$$

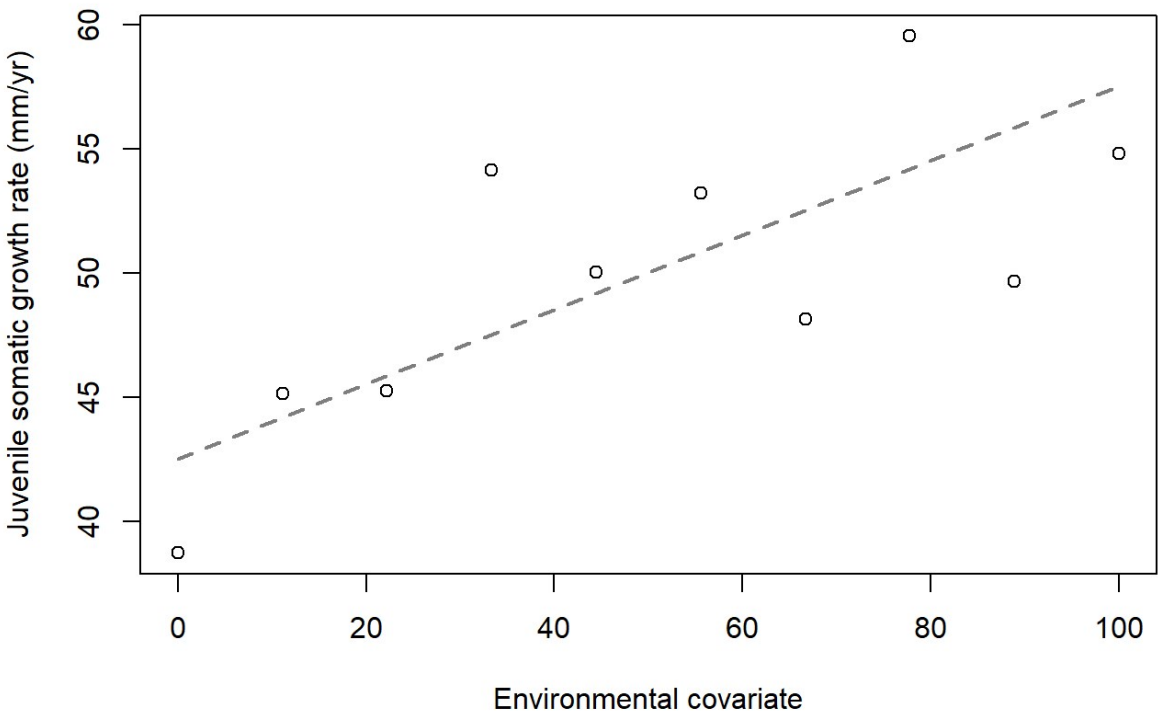
and

$$\sigma = h * cv_{\mathrm{h}}$$

.

```
set.seed(100)
h_i <- rnorm(nPop,(h+beta1*std.enviro),h*h_cv) # growth rate, h, for population i is random arising from a normal distribution with mean h and standard deviation of h*h_cv
plot(enviro, h_i, xlab="Environmental covariate",ylab="Juvenile somatic growth rate (mm/yr)")
lines(enviro,(h+beta1*std.enviro),lty=2,lwd=2,col="grey50")
Tmat_i <- rnorm(nPop,Tmat,Tmat*Tmat_cv)
t1_i <- rnorm(nPop,t1,abs(t1*t1_cv))
g_i <- rbeta(nPop,get_beta(g,g_cv)$alpha,get_beta(g,g_cv)$beta) # grab the shape parameters of a beta distribution based on its mean and variance
ifelse(all(g_i < 3/(Tmat_i-t1_i)), # there is a life-hisory constraint on the parameter g
       g_i <- g_i,
```

```
       g_i <- rbeta(nPop,get_beta(g,g_cv)$alpha,get_beta(g,g_cv)$beta))
## [1] 0.1641956
cv_i <- rnorm(nPop,cv,cv*sizeCV)

true.par <- list(h=h,T50=Tmat,t1=t1,g=g,sizeCV=cv,
                 h_pop=h_i,T50_pop=Tmat_i,t1_pop=t1_i,g_pop=g_i,
                 h_cv=h_cv,T50_cv=Tmat_cv,t1_cv=t1_cv,g_cv=g_cv)
```



Simulating the data

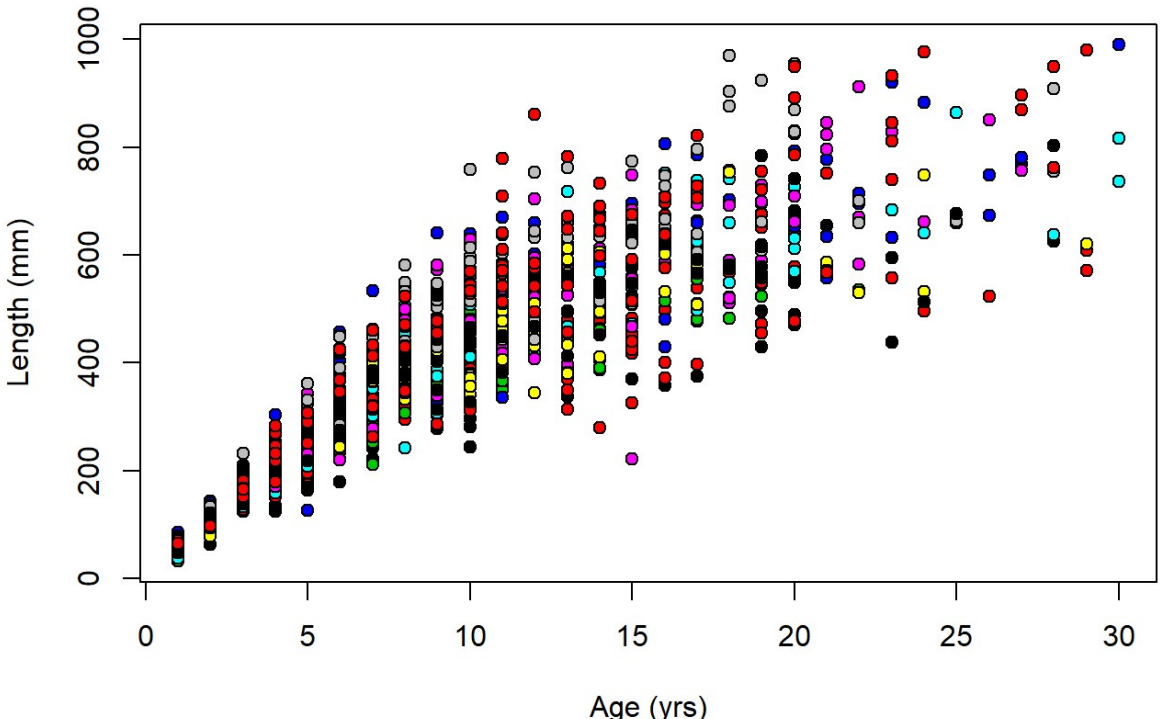
Next, we make an empty data object which we later fill with our population-specific, randomly generated length-at-age data. We use M to generate each population's survivorship curve. This allows us to simulate more realistic age-structures (by multipltying M by selectivity). We generate data using a random normal distribution (realized parameter values and model fit quality will change due to randomness) using the `rnorm()` function. Sample sizes expected for each age should be realistic for fisheries data and are determined from a function of gear selectivity and natural mortality (or survivorship) arising from a multinomial process using the `rmultinom()` function.

```
dataPop <- NULL  # make an empty object that we will fill in later
for (i in 1:nPop) {
    surv <- rep(NA, length(ages))  # create an empty vector
    surv[1] <- 1
    for (j in 2:max(ages)) {
        surv[j] <- surv[j - 1] * exp(log(((g_i[i]/1.18) - 1)/-1))
    }  #survivorship from discrete annual survival
    gearA50 <- Tmat_i[i]  # induce a gear selectivity that inflects at T
    gearSlope <- -0.3  # define the slope of selectivity
    select <- 1/(1 + exp(gearSlope * (ages - gearA50)))  # the average selectivity curve


    linf <- 3 * h_i[i]/g_i[i]  # conversion for the VBGF L-infinity
    vbk <- log(1 + g_i[i]/3)  # conversion for the VBGF parameter kappa
    t0 <- Tmat_i[i] + log(1 - g_i[i] * (Tmat_i[i] - t1_i[i])/3)/log(1 + g_i[i]/3)  #conversion for th
e VBGF parameter t0
    lena_juv <- h_i[i] * (ages - t1_i[i])  # length-at-age for phase 1
    lena_adult <- linf * (1 - exp(-vbk * (ages - t0)))  # length-at-age for phase 2
    mean_size <- ifelse(ages <= Tmat_i[i], lena_juv, lena_adult)  #if-else statement for which phase
a fish is allocating surplus energy
```

```
    SampSize <- 10000
    maxSamp <- as.vector(rmultinom(1, prob = surv * select, size = SampSize))  # whats the maximum nu
mber of observable samples for an age group in a population?
    # surv*select is the probability of a fish surviving a certain ageand being
    # sampled

    mean.samp <- as.data.frame(cbind(mean_size, maxSamp))  #make matrix of mean lengths-at-age and sa
mple sizes
    mlen <- rep(mean.samp[, 1], mean.samp[, 2])  #repeat each mean 'sample size' number of times
    ageData <- rep(ages, mean.samp[, 2])  #repeat each age 'sample size' number of times
    lengths <- sapply(mlen, function(x) rnorm(1, mean = x, sd = x * cv))  #generate random normal len
gth data using means & cv error
    Data <- data.frame(cbind(ageData, lengths, rep(i, length(ageData))), row.names = NULL)  #bind vec
tors into age and length matrix, covert to data frame
    colnames(Data) <- c("Age", "Size", "Pop_Num")  # re-name the columns
    Data <- Data[sample(nrow(Data), size = round(runif(1, 40, 200)), replace = F),
        ]  #draw a random sample from the population -- total sample size can be adjusted
    rownames(Data) <- c()
    dataPop <- rbind(dataPop, Data)
}

rownames(dataPop) <- c()
```

Next, we plot the noisy data, with colors assigned to the data coming from each population. We can also take a quick look at some of the data to visualize the data structure.

```
plot(dataPop$Age, dataPop$Size, bg = dataPop$Pop_Num, pch = 21, xlab = "Age (yrs)",
    ylab = "Length (mm)")

dataPop[sample(1:nrow(dataPop), 10), ]
##       Age      Size Pop_Num
## 31     10 401.60220       1
## 1082   11 579.85309      10
## 733     4 183.90674       7
## 860     9 521.21011       8
## 925     6 273.51945       9
## 647    19 590.01397       6
## 423     1  72.59156       4
## 222    10 418.53558       2
## 590    18 753.39073       6
## 954    18 580.49422       9
```



Write the hierarchical model in JAGS

The following code is written in the JAGS language (Plummer 2017). JAGS is fed a list associated with the data. Each data point has an population identification index.

The JAGS model loops through each sample, from *i* to *Nfish*, and determines the contribution of the predicted size of fish *i* to the log-posterior density, which is the sum of the log-likelihood and the log-prior. The predicted growth follows the analytical model from the equations in Lester et al. (2004). There is an if-else statement that determines if the predicted size-at-age of fish *i* comes from the juvenile phase or the adult phase. The priors for each population's life history trait e.g., h for population *j*, comes from a global hyper-prior for h representing the average across all populations. Modifications to this code will need to use JAGS syntax and not R syntax.

```
model <- "model {
#run through all the individual fish
for(i in 1:Nfish) {
  size[i] ~ dnorm(pred[i],1/(pred[i]*size.cv)^2)T(0,) # one variability across all populations

  #predict growth for each fish
  juv[i] <- h_pop[Pop[i]]*(age[i]-t1_pop[Pop[i]]) # predicted growth for fish i for the juvenile phas
e

  # below is the predicted growth for fish i for the adult phase which follows converting Lester et a
l. (2004) equations into the von Bertalanffy growth function

  adult[i] <- (3*h_pop[Pop[i]]/g_pop[Pop[i]])*(1-exp(-(log(1+g_pop[Pop[i]]/3))*(age[i]-(T50_pop[Pop[i
]]+log(1-g_pop[Pop[i]]*(T50_pop[Pop[i]]-t1_pop[Pop[i]])/3)/log(1+g_pop[Pop[i]]/3)))))
  pred[i] <- ifelse(age[i]<=T50_pop[Pop[i]],juv[i],adult[i]) # does the age of fish i exceed the matu
rity predicted for its population?

} # end the calculation of the likelihood


# priors by population
for(j in 1:Npop) {
  # normal priors for population j
  T50_pop[j] ~ dnorm(T50,tau.T50)T(0,)
  h_pop[j] ~ dnorm(h+wb*beta1*enviro[j],tau.h)T(0,)
  t1_pop[j] ~ dnorm(t1, tau.t1)

  # Beta distribution for the reproductive investment
  #(bounded between 0 and stochastic upper limit)

  g_pop[j] ~ dbeta(beta_alpha,beta_beta)T(,3/(T50_pop[j]-t1_pop[j]))

} # end the priors for each trait by population


# hyper priors on life history traits

T50 ~ dnorm(10,1e-7)T(0,)
h ~ dnorm(40,1e-7)T(0,)
t1 ~ dnorm(0,1e-2)
g ~ dgamma(0.001,0.001)T(,3/(T50-t1))
beta1 ~ dnorm(0,1e-3)
wb ~ dbern(0.5)

# below are the half-t priors on variance parameters

T50_cv ~ dhalfcauchy(10)
h_cv ~ dhalfcauchy(10)
g_cv ~ dhalfcauchy(10)
t1_cv ~ dhalfcauchy(10)
size.cv ~ dhalfcauchy(10)
```

```
# conversion to JAGS precision parameters

tau.T50 <- 1/(T50*T50_cv)^2
tau.h <- 1/(h*h_cv)^2
tau.t1 <- 1/(t1*t1_cv)^2

# re-parameterize the parameters of the beta

g_var <- (g*g_cv)^2
beta_alpha <- -g*(g_var+g^2-g)/g_var
beta_beta <- beta_alpha/g-beta_alpha
}"
```

Let's quickly look at how many samples we have for each of the populations. Then, we will compile the data into a usable list, and pass JAGS some initial starting values for each of the MCMC chains that we will run.

```
print(table(dataPop$Pop_Num))
##
##   1   2   3   4   5   6   7   8   9  10
##  98 147  71 135 118 161  77 104  81  96
data <- list(Nfish = length(dataPop$Age), Npop = length(unique(dataPop$Pop_Num)),
    age = dataPop$Age, size = dataPop$Size, Pop = dataPop$Pop_Num, enviro = std.enviro)
# the above list compiles the noisy data from the dataPop dataframe into 3
# vectors for age, size, and Pop (numerically, which population does each
# row correspond to?)

inits1 <- list(h = h, T50 = Tmat, g = g, t1 = t1, h_pop = rep(h, nPop), T50_pop = rep(Tmat,
    nPop), t1_pop = rep(t1, nPop), g_pop = rep(g, nPop), size.cv = cv, h_cv = h_cv,
    t1_cv = t1_cv, g_cv = g_cv, T50_cv = Tmat_cv, beta1 = 0, wb = 0.5, .RNG.name = "base::Wichmann-Hi
ll",
    .RNG.seed = 735)

# initial estimates of each parameter must be provided. RNG is random number
# generators for that chain

inits2 <- inits3 <- inits4 <- inits1
inits2 <- rngList(inits2, inits1)  # jitter chain 2, based on values of chain 1
inits3 <- rngList(inits3, inits1)  # jitter chain 3, based on values of chain 1
inits4 <- rngList(inits4, inits1)  # jitter chain 4, based on values of chain 1

inits <- list(inits1, inits2, inits3, inits4)  # compile all initial values into one list

mon_names <- c(names(inits3)[-c(length(inits3), length(inits3) - 1)])  # create the stochastic nodes
to be monitored
```

Run the MCMC chains in JAGS

Our next step is to set how many posterior samples we want, the length of the burn-in period and adaptation period, and the thinning rate. Our thinning rate is a somewhat high value of 10-30 - although this slows us down, the expected large correlations between parameters and the Markovian sampling process can lead to high autocorrelation among the consecutive posterior samples. To gain more independent samples, we run each chain longer by increasing our thinning rate while still only taking `Nsamp` number of samples.

We then use the `run.jags()` function to call JAGS to run our model (Denwood 2016). We specify the modules we run, and some other parameters internal to `run.jags()`, like the `method` and `modules`. For example, the `glm` module helps us estimate the linear regression aspect to the relationship between h and the simulated environmental gradient. One can also use the `rjparallel` method to parallelize the MCMC estimation and speed up JAGS model run times.

The `summary(results)` command reports some quick MCMC diagnostic tests to assess whether the posterior has converged on a stable distribution. The potential scale reduction factor (also called the Gelman-Rubin test) and the effective number of sample sizes are all reported in this summary output. The library `coda` offers more options for MCMC diagnostics.

```
Nsamp <- 1000  # how many posterior samples does each chain need to get, after thinning and burin-in
and adaptation?
thin_rt <- 15  # place some sort of thinning rate?
```

```
burnins <- 0.75 * round(Nsamp * thin_rt, 0)  # how long is the burnin, this bases it on the number of
 total posterior draws?
adaptin <- round(0.4 * burnins, 0)

a <- proc.time()
results <- run.jags(model = model, monitor = mon_names, data = data, n.chains = 4,
    method = "rjags", inits = inits, plots = F, silent.jag = F, modules = c("bugs",
        "glm", "dic"), sample = Nsamp, adapt = adaptin, burnin = burnins, thin = thin_rt,
    summarise = F)
## Compiling rjags model...
## Calling the simulation using the rjags method...
## Adapting the model for 4500 iterations...
## Burning in the model for 11250 iterations...
## Running the model for 15000 iterations...
## Simulation complete
## Finished running the simulation
b <- (proc.time() - a)

print((b[3]/60)/60)  # how long it took in hours
##   elapsed
## 0.6061944
print((b[3]/(4 * (Nsamp * thin_rt + burnins + adaptin))))  # how long it took in seconds per iteratio
n
##    elapsed
## 0.01774228
sum_results <- summary(results)
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 51 variables....
print(sum_results[, c(1, 2, 3, 4, 8, 9, 10)])
##                   Lower95      Median      Upper95        Mean MC%ofSD
## h            4.459709e+01 49.73989555 54.305025037 49.73276953     1.6
## T50          7.171984e+00  8.05246034  8.893020071  8.00120022     2.6
## g            1.511950e-01  0.16225976  0.174681524  0.16231723     2.8
## t1          -2.667188e-01 -0.17522595 -0.079346426 -0.17425978     3.3
## h_pop[1]     3.720716e+01 38.82575594 40.580802571 38.84863913     2.0
## h_pop[2]     4.230356e+01 44.33171599 46.702204650 44.37499601     2.5
## h_pop[3]     4.399790e+01 46.42841892 49.336918467 46.55588830     2.2
## h_pop[4]     5.279326e+01 55.03657085 57.286955187 55.06012613     2.1
## h_pop[5]     4.728801e+01 49.55936888 51.680389006 49.57773512     2.0
## h_pop[6]     5.097148e+01 53.04220143 55.371765707 53.04152452     2.2
## h_pop[7]     4.529205e+01 48.42324415 51.713392265 48.51725988     2.9
## h_pop[8]     5.601160e+01 58.48083739 61.217636135 58.50305587     2.2
## h_pop[9]     4.620852e+01 48.41004972 51.136808749 48.46228265     2.2
## h_pop[10]    5.310352e+01 56.28591641 59.440945971 56.25270073     2.6
## T50_pop[1]   6.882894e+00  8.03248598  9.004804818  8.01781285     1.8
## T50_pop[2]   6.274180e+00  7.85110278  8.936561877  7.76005109     2.5
## T50_pop[3]   6.535660e+00  8.15694908  9.750215711  8.13003370     2.0
## T50_pop[4]   7.122263e+00  8.10460195  9.051174935  8.12930264     1.9
## T50_pop[5]   7.123656e+00  8.19906783  9.565861173  8.24895040     1.9
## T50_pop[6]   7.140382e+00  8.11231907  9.134882534  8.13263056     1.9
## T50_pop[7]   5.097564e+00  7.50463728  8.710093183  7.28378094     3.1
## T50_pop[8]   7.167852e+00  8.09836345  9.399720435  8.14886934     1.8
## T50_pop[9]   6.299044e+00  7.89072175  9.014218539  7.82667193     2.1
## T50_pop[10]  7.361425e+00  8.59662967 10.904848435  8.84482647     2.8
## t1_pop[1]   -2.920362e-01 -0.17003090 -0.030831730 -0.16898373     2.1
## t1_pop[2]   -2.692718e-01 -0.15474960 -0.036961337 -0.15173280     2.4
## t1_pop[3]   -2.691294e-01 -0.14337264  0.026122161 -0.13498121     2.5
## t1_pop[4]   -3.210378e-01 -0.20210156 -0.101124312 -0.20657538     2.2
## t1_pop[5]   -2.870487e-01 -0.17238936 -0.063490355 -0.17147511     2.1
## t1_pop[6]   -3.355411e-01 -0.20952555 -0.106254288 -0.21429249     2.2
## t1_pop[7]   -4.157618e-01 -0.23707190 -0.107135932 -0.25140252     2.4
## t1_pop[8]   -2.874664e-01 -0.16084752 -0.032663984 -0.15785912     2.2
## t1_pop[9]   -3.140630e-01 -0.18167051 -0.053471740 -0.18330752     2.0
## t1_pop[10]  -2.873326e-01 -0.15647488 -0.004237005 -0.15161167     2.2
## g_pop[1]     1.460972e-01  0.16202295  0.179833410  0.16224909     2.2
## g_pop[2]     1.431015e-01  0.16031634  0.175309981  0.15992640     2.5
## g_pop[3]     1.461150e-01  0.16328530  0.182359388  0.16383066     2.2
## g_pop[4]     1.481329e-01  0.16426198  0.181122386  0.16466612     2.3
```

```
## g_pop[5]     1.444189e-01  0.16101634  0.176163425  0.16094040     2.3
## g_pop[6]     1.493886e-01  0.16364847  0.180579923  0.16402957     2.4
## g_pop[7]     1.473760e-01  0.16433144  0.185673470  0.16522429     2.2
## g_pop[8]     1.460719e-01  0.16216697  0.178835481  0.16226033     2.3
## g_pop[9]     1.449792e-01  0.16247343  0.179689735  0.16265189     2.3
## g_pop[10]    1.384284e-01  0.15859061  0.174697428  0.15741481     2.5
## size.cv      1.385273e-01  0.14469390  0.151204398  0.14473448     1.6
## h_cv         7.393359e-02  0.13354669  0.234652417  0.14251983     1.6
## t1_cv        2.952710e-03  0.35833715  1.274315264  0.49183174     3.5
## g_cv         2.776564e-05  0.03704686  0.110204438  0.04419136     3.0
## T50_cv       2.290189e-04  0.07565156  0.236526801  0.10434451     3.5
## beta1       -6.318332e+01  0.03537420 58.029987387 -0.20972555     1.5
## wb           0.000000e+00  0.00000000  0.000000000  0.02750000     2.0
##              SSeff        AC.150
## h             3824 -0.0205867081
## T50           1457  0.0913373337
## g             1245  0.0309343576
## t1             943  0.0465231393
## h_pop[1]      2407  0.0015647574
## h_pop[2]      1656  0.0196889482
## h_pop[3]      2154  0.0109568475
## h_pop[4]      2293  0.0153098340
## h_pop[5]      2581 -0.0102818991
## h_pop[6]      2109  0.0108988332
## h_pop[7]      1187  0.0122261944
## h_pop[8]      2148  0.0148342159
## h_pop[9]      2125  0.0041554726
## h_pop[10]     1462  0.0345612849
## T50_pop[1]    3027 -0.0056014931
## T50_pop[2]    1634  0.0536085417
## T50_pop[3]    2398  0.0045593235
## T50_pop[4]    2878 -0.0137065703
## T50_pop[5]    2903 -0.0054253702
## T50_pop[6]    2705  0.0013203524
## T50_pop[7]    1024  0.0180082932
## T50_pop[8]    2981 -0.0102424945
## T50_pop[9]    2204 -0.0037355045
## T50_pop[10]   1311  0.0122221363
## t1_pop[1]     2180  0.0057660223
## t1_pop[2]     1738  0.0295164308
## t1_pop[3]     1613  0.0073198096
## t1_pop[4]     2123  0.0413751730
## t1_pop[5]     2285 -0.0160985689
## t1_pop[6]     1978  0.0115728147
## t1_pop[7]     1797  0.0354908724
## t1_pop[8]     2030  0.0113605123
## t1_pop[9]     2386  0.0188220490
## t1_pop[10]    2092  0.0219827390
## g_pop[1]      2074  0.0077261201
## g_pop[2]      1644  0.0185889411
## g_pop[3]      2070  0.0005512672
## g_pop[4]      1870 -0.0055113086
## g_pop[5]      1862  0.0042657217
## g_pop[6]      1795  0.0343102316
## g_pop[7]      1979 -0.0114286556
## g_pop[8]      1930  0.0062446860
## g_pop[9]      1890  0.0107382224
## g_pop[10]     1625  0.0533494199
## size.cv       3968  0.0048576017
## h_cv          4067 -0.0133208254
## t1_cv          811  0.0941052183
## g_cv          1117  0.0238644584
## T50_cv         810  0.1222339725
## beta1         4166  0.0128988368
## wb            2570 -0.0188111481
res.corr <- extract.runjags(add.summary(results), "crosscorr")  # extract the cross-correlation matri
x
```

```
## Calculating summary statistics...
## Calculating the Gelman-Rubin statistic for 51 variables....
```

Store the model results

Next, we store the results as an `mcmc.list` which could be used in `coda` for MCMC diagnostics (Plummer *et al.* 2006) (see Appendix S5), but here we just quickly coerce this into a `matrix` object for our purposes. We then use some `grep` functions to quickly grab the posterior distributions of monitored parameters associated with specific growth model parameters (e.g., h, h_i, and cv_{h}.

We can do some further diagnostics of the model by evaluating how biased the estimates of the growth parameters were in comparison to the simulated 'true' parameter values. To do this, we will use percent bias:

$$Bias = ((\theta_i - \hat{\theta}_i)/\theta_i) * 100$$

where θ_i is a life history parameter of interest, e.g., juvenile growth rate h_i.

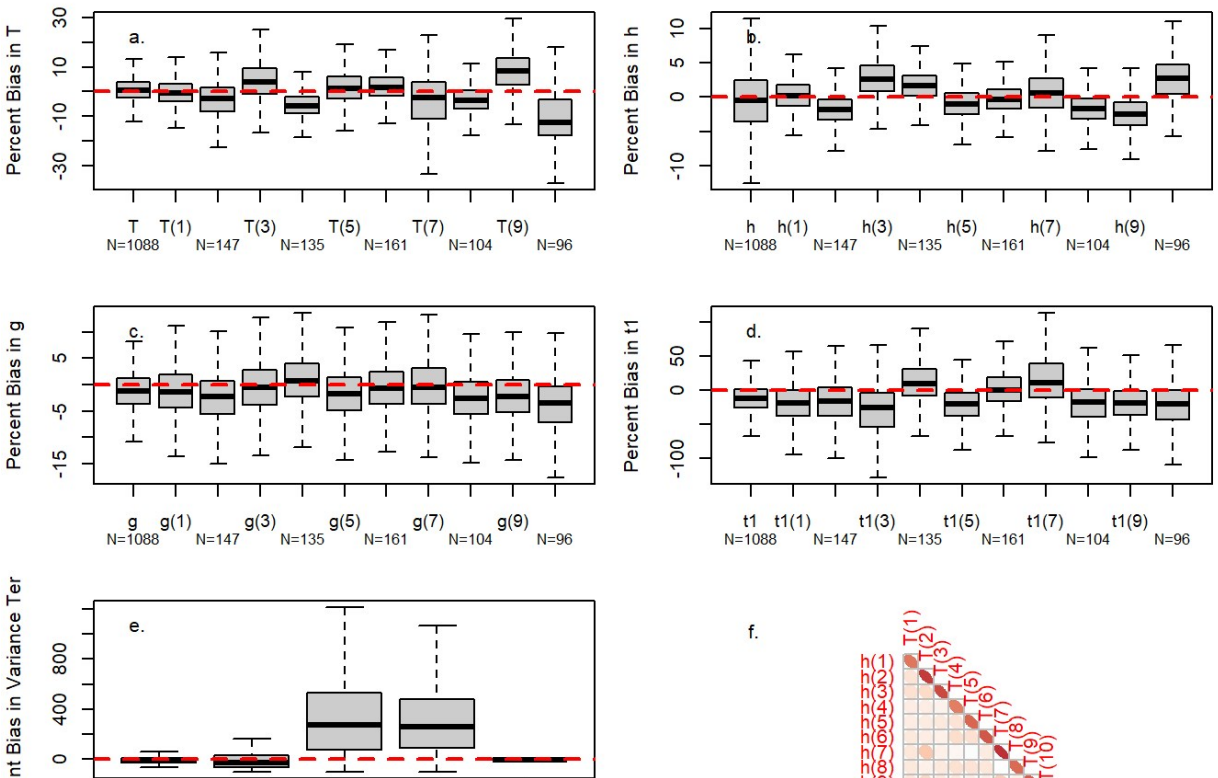
```
TheRes <- as.mcmc.list(results, vars = mon_names)
TheRes <- as.matrix(TheRes)

Ntot <- length(dataPop$Pop_Num)
Npop <- table(dataPop$Pop_Num)

h_plot <- cbind(TheRes[, grep("h", colnames(TheRes))])
Tmat_plot <- cbind(TheRes[, grep("T", colnames(TheRes))])
g_plot <- cbind(TheRes[, grep("g", colnames(TheRes))])
t1_plot <- cbind(TheRes[, grep("t1", colnames(TheRes))])
var_plot <- cbind(TheRes[, grep("cv", colnames(TheRes))])

hBias <- Tbias <- gBias <- t1Bias <- matrix(NA, ncol = ncol(h_plot) - 1, nrow = nrow(h_plot))
for (i in 1:(ncol(h_plot) - 1)) {
    hBias[, i] <- (h_plot[, i] - c(h, h_i)[i])/c(h, h_i)[i] * 100
    Tbias[, i] <- (Tmat_plot[, i] - c(Tmat, Tmat_i)[i])/c(Tmat, Tmat_i)[i] *
        100
    gBias[, i] <- (g_plot[, i] - c(g, g_i)[i])/c(g, g_i)[i] * 100
    t1Bias[, i] <- (t1_plot[, i] - c(t1, t1_i)[i])/c(t1, t1_i)[i] * 100
}

varBias <- matrix(NA, ncol = ncol(var_plot), nrow = nrow(var_plot))
for (i in 1:ncol(var_plot)) {
    varBias[, i] <- (var_plot[, i] - c(h_cv, Tmat_cv, g_cv, t1_cv, cv)[i])/c(h_cv,
        Tmat_cv, g_cv, t1_cv, cv)[i] * 100
}
```

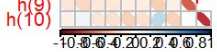
Percent bias and correlation plots

The code below generates a plot showing percent bias for each of the estimated parameters, along with some example correlations between the parameters T_i and h_i

```
par(mfrow = c(1, 1))
par(mar = c(4, 4, 1, 1))
layout(matrix(c(1, 2, 3, 4, 5, 6), nrow = 3, ncol = 2, byrow = TRUE))

boxplot(Tbias, ylab = "Percent Bias in T", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:(length(Tmat_i) + 1), c("T", paste("T(", 1:nPop, ")", sep = "")),
    line = 0, cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)

abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("a.", "topleft")

boxplot(hBias, ylab = "Percent Bias in h", col = "grey80", outline = FALSE,
```

```
    xaxt = "n")
axis(1, 1:(length(h_i) + 1), c("h", paste("h(", 1:nPop, ")", sep = "")), cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("b.", "topleft")

boxplot(gBias, ylab = "Percent Bias in g", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:(length(g_i) + 1), c("g", paste("g(", 1:nPop, ")", sep = "")), cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("c.", "topleft")

boxplot(t1Bias, ylab = "Percent Bias in t1", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:(length(t1_i) + 1), c("t1", paste("t1(", 1:nPop, ")", sep = "")),
    cex.axis = 0.9)
axis(1, 1:(length(Tmat_i) + 1), paste("N=", c(Ntot, Npop), sep = ""), line = 0.75,
    tick = F, cex.axis = 0.8)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("d.", "topleft")

boxplot(varBias, ylab = "Percent Bias in Variance Terms", col = "grey80", outline = FALSE,
    xaxt = "n")
axis(1, 1:length(c(h_cv, Tmat_cv, g_cv, t1_cv, cv)), c("cv(h)", "cv(T)", "cv(g)",
    "cv(t1)", "cv(L)"), cex.axis = 0.9)
abline(h = 0, lty = 2, col = "red", lwd = 2)
Corner_text("e.", "topleft")

corr.mat <- res.corr[5:14, 15:24]
name1 <- name2 <- c()
for (i in 1:nPop) {
    name1[i] <- paste("h(", i, ")", sep = "")
    name2[i] <- paste("T(", i, ")", sep = "")
}
rownames(corr.mat) <- name1
colnames(corr.mat) <- name2
corrplot(corr.mat, is.corr = TRUE, method = "ellipse", mar = c(2, 4, 1, 0),
    type = "lower", diag = T)

Corner_text("f.", "topleft")
```

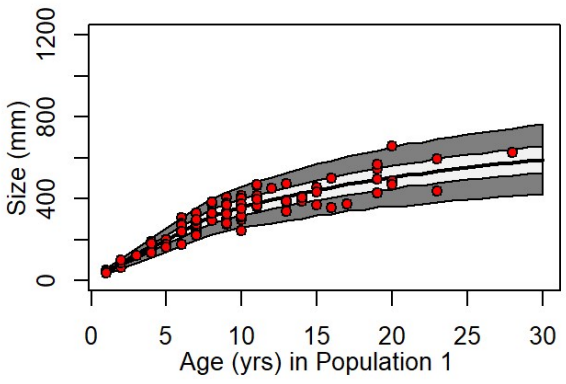


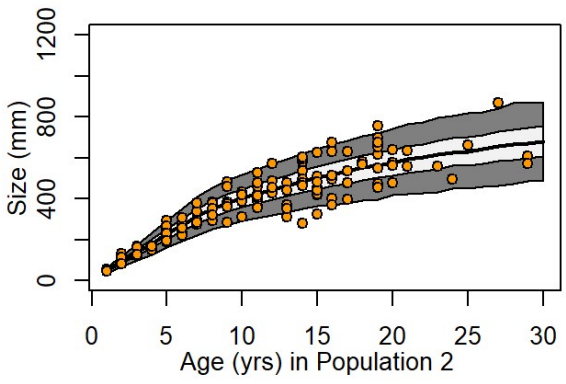


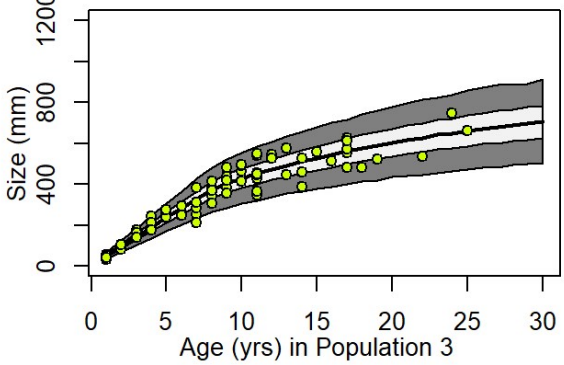
Posterior predictive checks

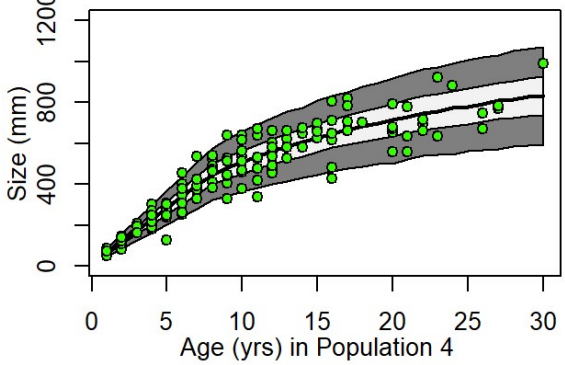
Next, we will run posterior predictive checks, which use the posterior distribution of the parameter estimates to re-generate a randomized posterior distribution of data, assuming the data arose according to the truncated normal distribution (as specified in the model above). We then overlay our observed data atop this distribution to check if there is any systematic bias (Gelman *et al.* (2013)).

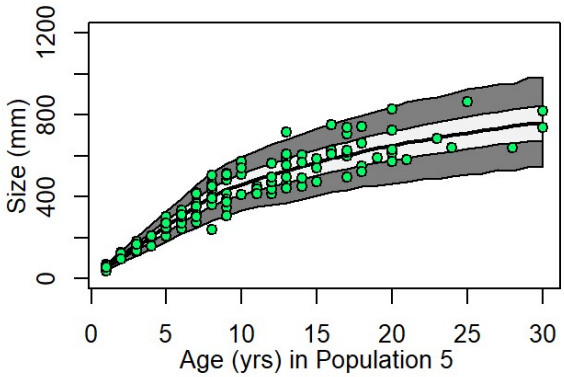
```
age_vec <- c(ages, rev(ages))
post_pred <- array(NA, dim = c(nrow(TheRes), length(ages), nPop))
# Above code creates an empty array to track size-at-age for population i
# for posterior draw j
par(mfrow = c(1, 1))
par(mar = c(5, 4, 1, 1))
layout(matrix(c(1, 2, 3, 4), nrow = 2, ncol = 2, byrow = TRUE))
for (i in 1:nPop) {
    subbed <- subset(dataPop, dataPop$Pop_Num == i)
    for (j in 1:nrow(TheRes)) {
        h_j <- TheRes[j, match(paste("h_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        g_j <- TheRes[j, match(paste("g_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        T50_j <- TheRes[j, match(paste("T50_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        t1_j <- TheRes[j, match(paste("t1_pop", "[", i, "]", sep = ""), colnames(TheRes))]
        cv_j <- TheRes[j, match("size.cv", colnames(TheRes))]

        linf <- 3 * h_j/g_j  # conversion for the VBGF L-infinity
        vbk <- log(1 + g_j/3)  # conversion for the VBGF parameter kappa
        t0 <- T50_j + log(1 - g_j * (T50_j - t1_j)/3)/log(1 + g_j/3)  #conversion for the VBGF parame
ter t0


        juv_j <- h_j * (ages - t1_j)
        adult_j <- linf * (1 - exp(-vbk * (ages - t0)))
        pred_j <- ifelse(ages < T50_j, juv_j, adult_j)
        pred_j[pred_j < 0.001] <- 0.001
        post_pred[j, , i] <- rnorm(length(ages), pred_j, pred_j * cv_j)
    }
    quants <- t(apply(post_pred[, , i], 2, FUN = quantile, probs = c(0.025,
        0.225, 0.5, 0.775, 0.975)))
    plot(age_vec, c(quants[, 1], rev(quants[, 5])), type = "l", lwd = 2, col = NA,
        ylab = "", xlab = "", ylim = c(0, 1200))
    axis(1, at = median(c(0, max(ages))), paste("Age (yrs) in ", "Population ",
        i, sep = ""), tick = FALSE, line = 0.9)
    axis(2, at = 600, "Size (mm)", tick = FALSE, line = 1)
    polygon(age_vec, c(quants[, 1], rev(quants[, 5])), col = "grey50")
    polygon(age_vec, c(quants[, 2], rev(quants[, 4])), col = "grey95")
    lines(ages, quants[, 3], lwd = 2, col = "black")
    points(subbed$Age, subbed$Size, pch = 21, bg = rainbow(nPop)[i])
}
```

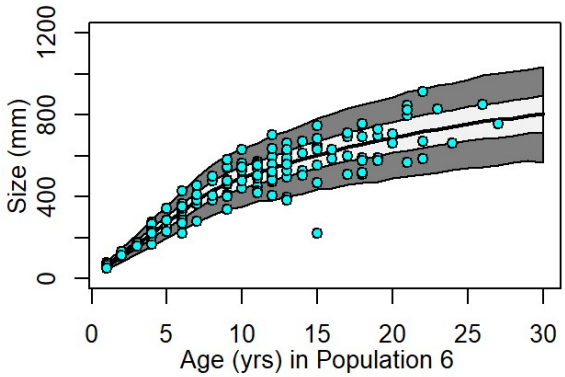


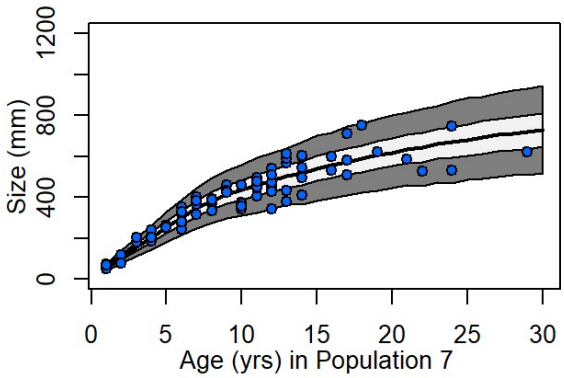


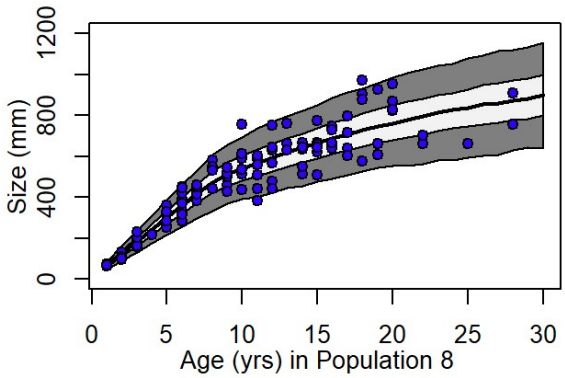

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 3

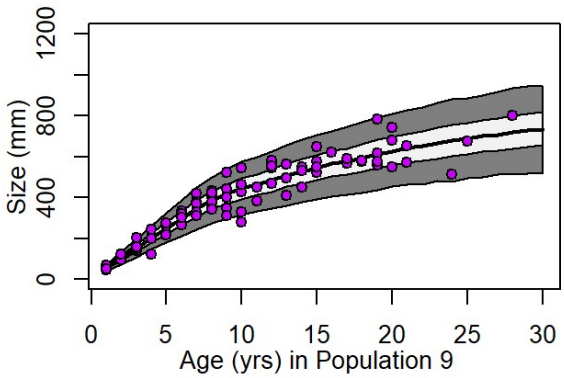

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 4

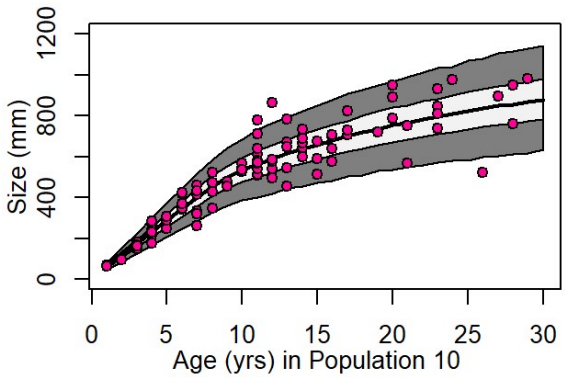

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 5


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 6


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 7


Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 8

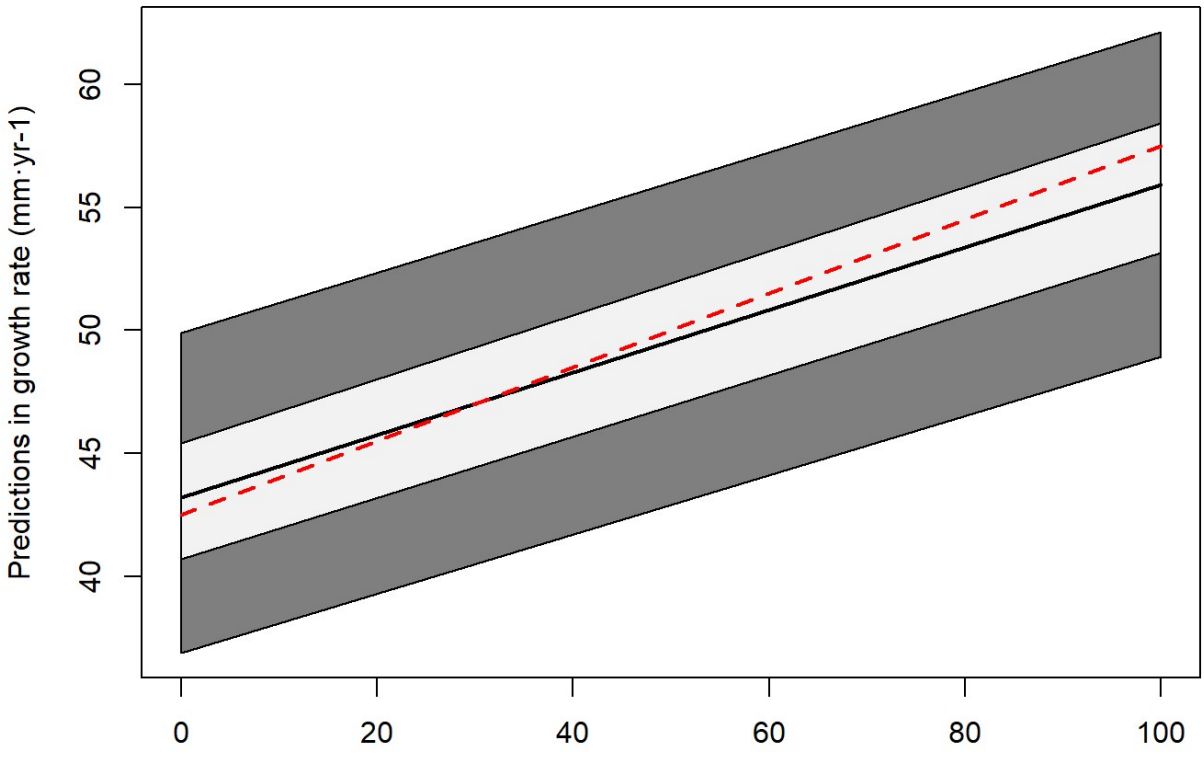

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 9

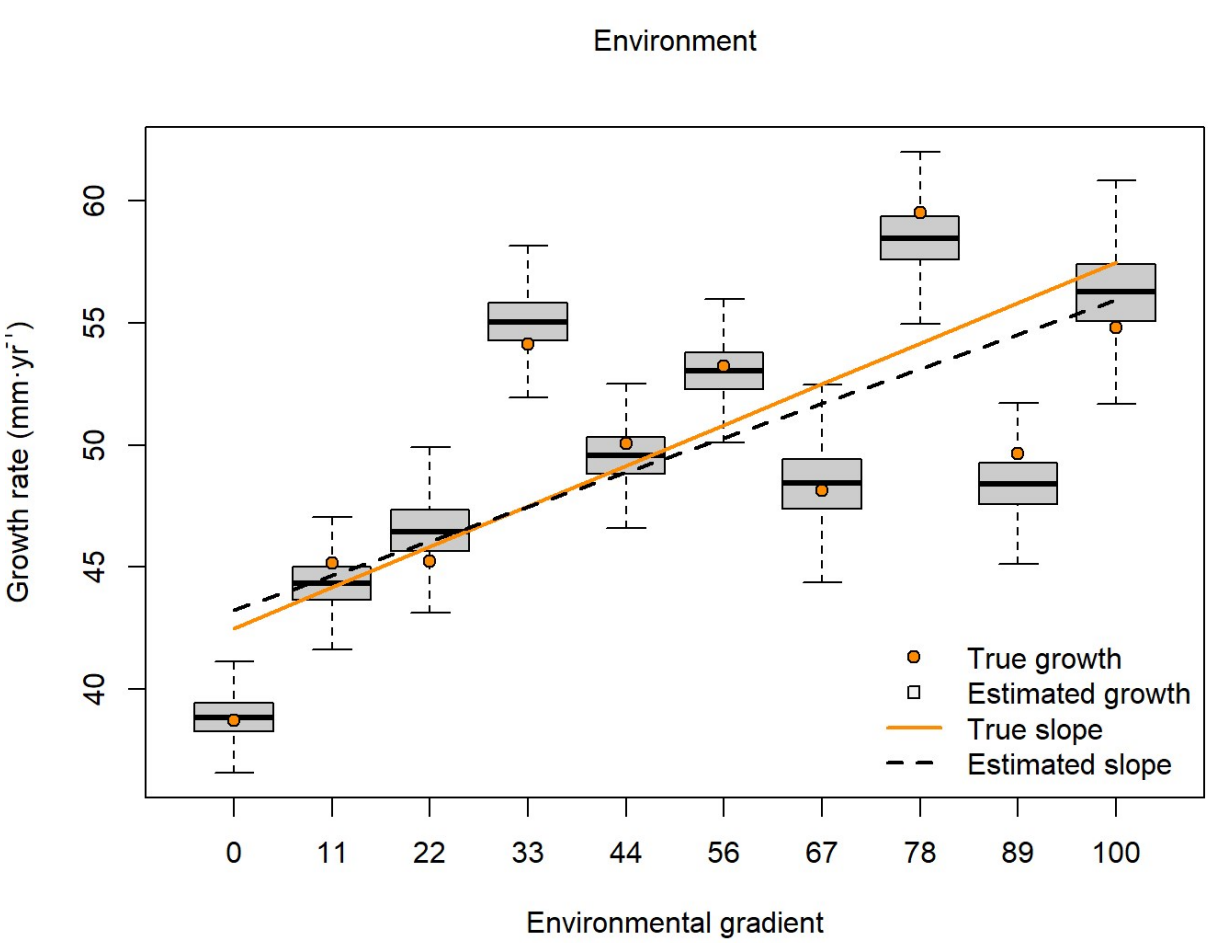

Size (mm)
0
400
800
1200
0
5
10
15
20
25
30
Age (yrs) in Population 10

Trait-environment relationship

Finally, we will plot the environmental trend on h. In our model, we used an inclusion parameter ω (Royle & Dorazio (2008)) which told the MCMC algorithm to sample β only when there was information in the posterior such that the Bernoulli trial on ω was equal to 1. Hence, we must remove all β when ω equals 0 for our inference on the distribution of the slope term.

```
layout(matrix(1, nrow = 1, ncol = 1))
par(mar = c(5, 4, 1, 1))
beta.hat <- sapply(TheRes[, "beta1"], FUN = function(x) {
    x * std.enviro
})
predRes <- beta.hat + TheRes[, "h"]
predRes <- predRes[, TheRes[, "wb"] != 0]
newEnviro <- seq(from = min(enviro), to = max(enviro), length = 2)
newStdEnviro <- newEnviro - mean(newEnviro)
beta.hat <- sapply(TheRes[, "beta1"], FUN = function(x) {
    x * newStdEnviro
})
predRes <- beta.hat + TheRes[, "h"]
predRes <- predRes[, TheRes[, "wb"] != 0]
betaQuants <- t(apply(predRes, 1, FUN = quantile, probs = c(0.025, 0.225, 0.5,
    0.775, 0.975)))
enviroRev <- c(newEnviro, rev(newEnviro))

plot(newEnviro, betaQuants[, 3], type = "l", ylim = range(betaQuants), xlab = "Environment",
    ylab = "Predictions in growth rate (mm·yr-1)")
polygon(enviroRev, c(betaQuants[, 1], rev(betaQuants[, 5])), col = "grey50")
polygon(enviroRev, c(betaQuants[, 2], rev(betaQuants[, 4])), col = "grey95")
lines(newEnviro, betaQuants[, 3], lty = 1, lwd = 2)
lines(newEnviro, beta1 * newStdEnviro + h, lty = 2, col = 2, lwd = 2)

boxplot(t(TheRes[, grep("h_pop", colnames(TheRes))]) ~ enviro, xaxt = "n", xlab = "Environmental grad
ient",
    ylab = expression(paste("Growth rate (mm·yr"^"-1", ")")), outline = F, col = "grey80")
axis(1, at = 1:10, round(enviro))
lines(1:nPop, beta1 * std.enviro + h, lty = 1, col = "darkorange", lwd = 2)
lines(c(1, nPop), betaQuants[, 3], lty = 2, lwd = 2, col = "black")
points(1:nPop, h_i, pch = 21, bg = "darkorange")
legend("bottomright", bty = "n", lty = c(NA, NA, 1, 2), pch = c(21, 22, NA,
    NA), pt.bg = c("darkorange", "grey95", NA, NA), lwd = c(NA, NA, 2, 2), c("True growth",
    "Estimated growth", "True slope", "Estimated slope"), col = c(1, 1, "darkorange",
    "black"))
```





References

Denwood, M.J. (2016). runjags: An R package providing interface utilities, model templates, parallel computing methods and additional distributions for MCMC models in JAGS. *Journal of Statistical Software*, **71**, 1–25.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A. & Rubin, D. (2013). *Bayesian Data Analysis*, 3rd edn. Chapman and Hall/CRC Press.

Helser, T.E. & Lai, H.-L. (2004). A bayesian hierarchical meta-analysis of fish growth: With an example for north american largemouth bass, micropterus salmoides. *Ecological Modelling*, **178**, 399–416.

Lester, N.P., Shuter, B.J. & Abrams, P.A. (2004). Interpreting the von bertalanffy model of somatic growth in fishes: The cost of reproduction. *Proceedings of the Royal Society B: Biological Sciences*, **271**, 1625–1631.

Lorenzen, K. (2016). Toward a new paradigm for growth modeling in fisheries stock assessments: Embracing plasticity and its consequences. *Fisheries Research*, **180**, 4–22.

Plummer, M. (2017). JAGS: A program for analysis of bayesian graphical models using gibbs sampling.

Plummer, M., Best, N., Cowles, K. & Vines, K. (2006). CODA: Convergence diagnosis and output analysis for mcmc. *R News*, **6**, 7–11.

Royle, J.A. & Dorazio, R.M. (2008). *Hierarchical modeling and inference in ecology: the analysis of data from populations, metapopulations and communities*. Academic Press.

Wilson, K.L., Honsey, A., Moe, B. & Venturelli, P. (2017). Growing the biphasic framework: techniques and recommendations for fitting emerging growth models. *Methods in Ecology and Evolution*, **In Review**.