What's for dynr: A Package for Linear and Nonlinear Dynamic Modeling in R

by Lu Ou[+], Michael D. Hunter[+], and Sy-Miin Chow

Abstract Intensive longitudinal data in the behavioral sciences are often noisy, multivariate in nature, and may involve multiple units undergoing regime switches by showing discontinuities interspersed with continuous dynamics. Despite increasing interest in using linear and nonlinear differential/difference equation models with regime switches, there has been a scarcity of software packages that are fast and freely accessible. We have created an R package called **dynr** that can handle a broad class of linear and nonlinear discrete- and continuous-time models, with regime-switching properties and linear Gaussian measurement functions, in C, while maintaining simple and easy-to-learn model specification functions in R. We present the mathematical and computational bases used by the **dynr** R package, and present two illustrative examples to demonstrate the unique features of **dynr**.

Introduction

The past several decades have seen a significant rise in the prevalence of intensive longitudinal data (ILD), particularly in the social and behavioral sciences (Bolger and Laurenceau, 2013; Byrom and Tiplady, 2010; Stone et al., 2008). Differential equation and difference equation models in the form of state-space models have been one of the most dominant tools for representing the dynamics of ILD in disciplines such as the physical sciences, econometrics, engineering, and ecology. In parallel, some computational advances have been proposed in estimating *regime-switching* models — namely, models positing how otherwise continuous dynamic processes may undergo discontinuous changes through categorical but unobserved phases known as "regimes" (Kim and Nelson, 1999; Hamilton, 1989; Muthén and Asparouhov, 2011; Chow et al., 2013, 2015; Dolan, 2009). Throughout, we use the terms *regimes* and *classes* interchangeably to denote unobserved unit- and time-specific indicator variables that serve to group portions of repeated measures into phases with homogeneous dynamics or measurement properties.

Examples of regime-switching phenomena from psychology includes Piaget's (1969) theory of human cognitive development and related extensions (Dolan et al., 2004; van der Maas and Molenaar, 1992; Hosenfeld, 1997); Kohlberg's (Kohlberg and Kramer, 1969) conceptualization of stagewise development in moral reasoning; Van Dijk and Van Geert's (2007) findings on discrete shifts in early language development; as well as Fukuda and Ishihara's (1997) work on the discontinuous changes in infant sleep and wakefulness rhythm during the first six months of life. Related to, but distinct from, hidden Markov models (Elliott et al., 1995; Visser, 2007), regime-switching differential and difference equation models allow researchers to specify targeted differential or difference functions to describe the continuous changes that occur within regimes. Ample work exists on fitting these models (Hamilton, 1989; Dolan, 2009; Yang and Chow, 2010; Chow et al., 2013; Chow and Zhang, 2013; Chow et al., 2015; Muthén and Asparouhov, 2011; Tong and Lim, 1980; Tiao and Tsay, 1994), but readily accessible software suited for handling such models with ILD are lacking.

Several programs and packages exist for fitting differential equation, difference equation, and hidden Markov models. However, each program has certain limitations that **dynr** (Ou et al., 2018) aims to overcome. Speaking broadly, the largest differences between **dynr** and other packages are threefold: (1) **dynr** readily allows for multi-unit models, (2) **dynr** allows for nonlinear discrete-time and continuous-time dynamics, and (3) **dynr** allows for regime switching throughout every part of the model. Many R packages exist for univariate and multivariate time series. CRAN lists hundreds of packages in its task view for *TimeSeries* (Hyndman, 2016), a complete review of which is well-beyond the scope of this work. However, generally these packages lack facilities for fitting time series from multiple units. Likewise there are very few software utilities designed for nonlinear dynamics or regime switching (see Table 1 for an overview). Petris and Petrone (2011) reviewed three packages for linear state-space models: **dlm** (Petris, 2010, 2014), **KFAS** (Helske, 2017a,b), and **dse** (Gilbert, 2006 or later, 2015). These are among the state of the art for state-space modeling in R. Although **KFAS** can accommodate in its measurement model all densities within the exponential family, the corresponding dynamic model is required to be linear. In addition to these R packages, the **OpenMx** 2.0 release (Neale et al., 2016; Boker et al., 2017) has maximum likelihood time-varying linear discrete- and continuous-time state-space modeling (Hunter, 2017). Likewise, the MKFM6 program (Dolan, 2005) implements methods of Harvey (1989) for time-invariant linear state-space models. SsfPack (Koopman et al.,

[+]These two authors contributed equally to the work.

1999) implements the methods of Durbin and Koopman (2001) for linear state-space modeling and Markov chain Monte Carlo methods for nonlinear modeling, but it is primarily restricted to single-unit time series without regime switching. The **ctsem** package (Driver et al., 2017b,a) has utilities for linear state-space modeling of multiple units in continuous time, but lacks functionality for nonlinear models or regime switching. MATLAB (The MathWorks, Inc., 2016) has numerous extensions for time series and state-space modeling (Grewal and Andrews, 2008), but lacks the ability to include regime switching and multiple units. Some R packages that handle regime switching are only designed for hidden Markov models, for example, **depmixS4** (Visser and Speekenbrink, 2016, 2010) and **RHmm** (Taramasco and Bauer, 2012), while the others are only for specific Markov-switching discrete-time time-series models, including **MSwM** (Sanchez-Espigares and Lopez-Moreno, 2014) for univariate autoregressive models, **MSBVAR** (Brandt, 2016) for vector autoregressive models, and **MSGARCH** (Ardia et al., 2017) for generalized autoregressive conditional heteroskedasticity models. The **pomp** package (King et al., 2016, 2018) lists among its features hidden Markov models and state-space models, both of which can be discrete- or continuous-time, non-Gaussian, and nonlinear. However, **pomp** does not currently support regime-switching functionality beyond the regime switching found in hidden Markov modeling. Helske (2017a) included a review of numerous other packages for non-Gaussian time series models which generally do not involve latent variables.

Overall, developments in fitting differential/difference equation models that evidence discontinuities in dynamics are still nascent. Despite some of the above-mentioned advances in computational algorithms, there is currently no readily available software package that allows researchers to fit differential/difference equations with regime-switching properties. As stated previously, currently available computational programs for dynamic modeling are limited in one of several ways: (1) they are restricted to handling only linear models within regimes such as the package **OpenMx**, (2) they can only handle very specific forms of nonlinear relations among latent variables, (3) they are computationally slow, (4) they do not allow for stochastic qualitative shifts in the dynamics over time, or (5) they require that the user write complex compiled code to enhance computational speed at the cost of high user burden. Efficient and user-friendly computer software needs to be developed to overcome these restrictions so the estimation of dynamic models can become more applicable by researchers.

We present an R package, **dynr**, that allows users to fit both linear and nonlinear differential and difference equation models with regime-switching properties. All computations are performed quickly and efficiently in C, but are tied to a user interface in the familiar R language. Specifically, for a very broad class of linear and nonlinear differential/difference equation models with linear Gaussian measurement functions, **dynr** provides R helper functions that write appropriate C code based on user input in R into a local (potentially temporary) C file, which is then compiled on user's end with a call to an R function in **dynr**. The C function pointers are passed to the back-end for computation of a negative log-likelihood function, which is numerically optimized also in C using the optimization routine SLSQP (Kraft, 1988, 1994) for parameter estimation. During the process, the user never has to write or even see the C code that underlies **dynr** and yet, the computations are performed entirely in C, with no interchanges between R and C to reduce memory copying and optimize speed. This removes some of the barriers to dynamic modeling, opening it as a possibility to a broader class of users, while retaining the flexibility of specifying targeted model-specific functions in C for users wishing to pursue models that are not yet supported in the R interface.

In the remaining sections, we will first present the mathematical and computational bases of the **dynr** R package, and then demonstrate the interface of **dynr** for modeling multivariate observations with Gaussian measurement errors using two ILD modeling examples from the social and behavioral sciences. Key features of the **dynr** package we seek to highlight include: (1) **dynr** fits discrete- and continuous-time dynamic models to multivariate longitudinal/time-series data; (2) **dynr** deals with dynamic models with regime-switching properties; (3) for improved speed, **dynr** computes and optimizes negative log-likelihood function values in C; (4) **dynr** handles linear and nonlinear dynamic models with an easy-to-use interface that includes a matrix form (for linear dynamic models only) and formula form (for linear as well as nonlinear models); (5) **dynr** removes the burden on the user to perform analytic differentiation in fitting nonlinear differential/difference equation models by providing the user with R's symbolic differentiation; and (6) **dynr** provides ready-to-present results through LaTeX equations and plots.

General modeling framework

At a basic level, our general modeling framework comprises a dynamic model and a measurement model. The former describes the ways in which the latent variables change over time, whereas the latter portrays the relationships between the observed variables and latent variables at a specific time.

The dynamic model for a particular regime in continuous-time assumes the following form:

$$d\boldsymbol{\eta}_i(t) = \boldsymbol{f}_{S_i(t)}\left(\boldsymbol{\eta}_i(t), t, \boldsymbol{x}_i(t)\right) dt + d\boldsymbol{w}_i(t), \tag{1}$$

where i indexes the smallest independent unit of analysis, t indexes time, $\boldsymbol{\eta}_i(t)$ is the $r \times 1$ vector of latent variables at time t, $\boldsymbol{x}_i(t)$ is the vector of covariates at time t, and $\boldsymbol{f}_{S_i(t)}(.)$ is the vector of (possibly nonlinear) dynamic functions which depend on the latent regime indicator, $S_i(t)$. The left-hand side of Equation 1, $d\boldsymbol{\eta}_i(t)$, gives the differential of the vector of continuous latent variables, $\boldsymbol{\eta}_i(t)$, and $\boldsymbol{f}_{S_i(t)}(.)$ is called the *drift* function. Added to these deterministic changes induced by the drift function is $\boldsymbol{w}_i(t)$, an r-dimensional Wiener process. The differentials of the Wiener processes have zero means and covariance matrix, $\mathbf{Q}_{S_i(t)}$, called the *diffusion* matrix. When the dynamic model consists only of linear functions, Equation 1 reduces to:

$$d\boldsymbol{\eta}_i(t) = \left(\boldsymbol{\alpha}_{S_i(t)} + \boldsymbol{F}_{S_i(t)}\boldsymbol{\eta}_i(t) + \boldsymbol{B}_{S_i(t)}\boldsymbol{x}_i(t)\right) dt + d\boldsymbol{w}_i(t). \tag{2}$$

where the general function $\boldsymbol{f}_{S_i(t)}()$ is replaced with a linear function consisting of (1) an intercept term $\boldsymbol{\alpha}_{S_i(t)}$, (2) linear dynamics in a matrix $\boldsymbol{F}_{S_i(t)}$, and (3) linear covariate regression effects $\boldsymbol{B}_{S_i(t)}$.

For discrete-time processes, we adopt a dynamic model in state-space form (Durbin and Koopman, 2001) as

$$\boldsymbol{\eta}_i(t_{i,j+1}) = \boldsymbol{f}_{S_i(t_{i,j})}\left(\boldsymbol{\eta}_i(t_{i,j}), t_{i,j}, \boldsymbol{x}_i(t_{i,j+1})\right) + \boldsymbol{w}_i(t_{i,j+1}), \tag{3}$$

now postulated to unfold at discrete time points indexed by sequential positive integers, $t_{i,j}$, $j = 1, 2, \cdots$. In this case, $\boldsymbol{w}_i(t_{i,j})$ denotes a vector of Gaussian distributed process noise with covariance matrix, $\mathbf{Q}_{S_i(t_{i,j})}$. We have intentionally kept notation similar between discrete- and continuous-time models to facilitate their linkage. **dynr** allows for an easy transition between these two frameworks with a binary flag. In a similar vein, we refer to $\boldsymbol{f}_{S_i(t)}(.)$ in both Equations 1 and 3 broadly as the *dynamic functions*. The same structure as Equation 2 is possible in discrete time as the linear analog of Equation 3,

$$\boldsymbol{\eta}_i(t_{i,j+1}) = \boldsymbol{\alpha}_{S_i(t_{i,j})} + \boldsymbol{F}_{S_i(t_{i,j})}\boldsymbol{\eta}_i(t_{i,j}) + \boldsymbol{B}_{S_i(t_{i,j})}\boldsymbol{x}_i(t_{i,j+1}) + \boldsymbol{w}_i(t_{i,j+1}). \tag{4}$$

In both the discrete- and continuous-time cases, the initial conditions for the dynamic functions are defined explicitly to be the latent variables at a unit-specific first observed time point, $t_{i,1}$, denoted as $\boldsymbol{\eta}_i(t_{i,1})$, and are specified to be normally distributed with means $\boldsymbol{\mu}_{\boldsymbol{\eta}_1}$ and covariance matrix, $\boldsymbol{\Sigma}_{\boldsymbol{\eta}_1}$:

$$\boldsymbol{\eta}_i(t_{i,1}) \sim N\left(\boldsymbol{\mu}_{\boldsymbol{\eta}_1}, \boldsymbol{\Sigma}_{\boldsymbol{\eta}_1}\right). \tag{5}$$

Likewise for both discrete- and continuous-time models, we assume that observations only occur at selected, discrete time points. Thus, we have a discrete-time measurement model in which $\boldsymbol{\eta}_i(t_{i,j})$ at discrete time point $t_{i,j}$ is indicated by a $p \times 1$ vector of manifest observations, $\boldsymbol{y}_i(t_{i,j})$. Continuous-time processes allow unequal time intervals for these observations. Missing data may be present under either specification. The vector of manifest observations is linked to the latent variables as

$$\boldsymbol{y}_i(t_{i,j}) = \boldsymbol{\tau}_{S_i(t_{i,j})} + \boldsymbol{\Lambda}_{S_i(t_{i,j})}\boldsymbol{\eta}_i(t_{i,j}) + \mathbf{A}_{S_i(t_{i,j})}\boldsymbol{x}_i(t_{i,j}) + \boldsymbol{\epsilon}_i(t_{i,j}), \quad \boldsymbol{\epsilon}_i(t_{i,j}) \sim N\left(\mathbf{0}, \mathbf{R}_{S_i(t_{i,j})}\right), \tag{6}$$

where $\boldsymbol{\tau}_{S_i(t_{i,j})}$ is a $p \times 1$ vector of intercepts, $\mathbf{A}_{S_i(t_{i,j})}$ is a matrix of regression weights for the covariates, $\boldsymbol{\Lambda}_{S_i(t_{i,j})}$ is a $p \times r$ factor loadings matrix that links the observed variables to the latent variables, and $\boldsymbol{\epsilon}_i(t_{i,j})$ is a $p \times 1$ vector of measurement errors assumed to be serially uncorrelated over time and normally distributed with zero means and covariance matrix, $\mathbf{R}_{S_i(t_{i,j})}$. Of course, all parts of the measurement model may be regime-dependent.

The subscript $S_i(t)$ in Equations 1–6 indicates that these functions and matrices may depend on $S_i(t)$, the operating regime. To make inferences on $S_i(t_{i,j})$, we initialize the categorical latent variable $S_i(t_{i,j})$ on the first occasion and then provide a model for how $S_i(t_{i,j})$ changes over time. The initial regime probabilities for $S_i(t_{i,1})$ are represented using a multinomial regression model as

$$\Pr\left(S_i(t_{i,1}) = m | \boldsymbol{x}_i(t_{i,1})\right) \triangleq \pi_{m,i1} = \frac{\exp(a_m + \mathbf{b}_m^T \boldsymbol{x}_i(t_{i,1}))}{\sum_{k=1}^{M} \exp(a_k + \mathbf{b}_k^T \boldsymbol{x}_i(t_{i,1}))}, \tag{7}$$

where M denotes the total number of regimes, a_m is the logit intercept for the mth regime and $\mathbf{b}_m$ is a $n_b \times 1$ vector of regression slopes linked to a vector of covariates that explain between-unit differences in initial log-odds (LO). For identification, a_m and all entries in $\mathbf{b}_m$ are set to zero for some regime, m.

We use a first-order Markov process to define how the classes change over time in a transition probability matrix, which contains all possible transitions from one regime to another. In the matrix, the rows index the previous regime at time $t_{i,j-1}$ and the columns index the current regime at time

$t_{i,j}$. The rows of this matrix sum to 1 because the probability of transitioning from a particular state to any other state must be 1. This transition matrix may also depend on covariates. Thus, a multinomial logistic regression equation is assumed to govern the probabilities of transitions between regimes as:

$$\Pr\left(S_i(t_{i,j}) = m | S_i(t_{i,j-1}) = l, \boldsymbol{x}_i(t_{i,j})\right) \triangleq \pi_{lm,it} = \frac{\exp(c_{lm} + \mathbf{d}_{lm}^T \boldsymbol{x}_i(t_{i,j}))}{\sum_{k=1}^{M} \exp(c_{lk} + \mathbf{d}_{lk}^T \boldsymbol{x}_i(t_{i,j}))}, \tag{8}$$

where $\pi_{lm,it}$ denotes unit i's probability of transitioning from class l at time $t_{i,j-1}$ to class m at time $t_{i,j}$, c_{lm} denotes the logit intercept for the transition probability, and $\mathbf{d}_{lm}$ is a $n_d \times 1$ vector of logit slopes summarizing the effects of the covariates in $\boldsymbol{x}_i(t_{i,j})$ on that transition probability. One regime, again, has to be specified as the reference regime by fixing all LO parameters, including c_{lm} and all elements in $\mathbf{d}_{lm}^T$ for some regime m, to zero for identification purposes.

To summarize, the model depicted in Equations 1 – 8 may take on the form of various linear or nonlinear dynamic models in continuous or discrete time. Moreover, these dynamic models may have regime-switching properties. Systematic between-unit differences stem primarily from changes in the unit- and time-specific regime, $S_i(t_{i,j})$, and the corresponding changes in the dynamic and measurement models over units and occasions.

Estimation procedures

In this section, we outline the procedures implemented in **dynr** for estimating the model shown in Equations 1 – 8. An overview of the estimation procedures involved, the different special cases handled by **dynr**, and the software packages that can handle these special cases are summarized in Table 1.

Discrete-time models

Broadly speaking, the estimation procedures implemented in **dynr** are based on the Kalman filter (KF; Kalman, 1960), its various continuous-time and nonlinear extensions, and the Kim filter (Anderson and Moore, 1979; Bar-Shalom et al., 2001; Kim and Nelson, 1999; Yang and Chow, 2010; Chow and Zhang, 2013; Kulikov and Kulikova, 2014; Kulikova and Kulikov, 2014; Chow et al., 2018). The Kim filter, designed to extend the Kalman filter to handle regime-switching state-space models, was proposed by Kim and Nelson (1999) and extended by Chow and Zhang (2013) to allow for nonlinear dynamic functions. In **dynr**, models are allowed to (1) be in discrete or continuous time, (2) be single regime or regime switching, (3) have linear or nonlinear dynamics, (4) involve stochastic or deterministic dynamics, and (5) have one or more units. All combinations of these variations are possible in **dynr**, creating 32 different kinds of models.

In the case of linear discrete-time dynamics without regime-switching, the model reduces to a linear state-space model, and we apply the Kalman filter to estimate the latent variable values and obtain other by-products for parameter optimization. At each time point, the KF consists of two steps. In the first step, the dynamics are used to make a prediction for the latent state at the next time point conditional on the observed measurements up to time $t_{i,j-1}$, creating a predicted mean $\hat{\boldsymbol{\eta}}_i(t_{i,j}|t_{i,j-1}) = E(\boldsymbol{\eta}_i(t_{i,j})|\mathbf{Y}_i(t_{i,j-1}))$ and covariance matrix for the latent state $\mathbf{P}_i(t_{i,j}|t_{i,j-1}) = Cov[\boldsymbol{\eta}_i(t_{i,j})|\mathbf{Y}_i(t_{i,j-1})]$, where $\mathbf{Y}_i(t_{i,j-1})$ includes manifest observations from time $t_{i,1}$ up to time $t_{i,j-1}$. In the second step, the prediction is updated based on the measurement model (Equation 6) and the new measurements, yielding $\hat{\boldsymbol{\eta}}_i(t_{i,j}|t_{i,j}) = E(\boldsymbol{\eta}_i(t_{i,j})|\mathbf{Y}_i(t_{i,j}))$ and associated covariance matrix, $\mathbf{P}_i(t_{i,j}|t_{i,j}) = Cov[\boldsymbol{\eta}_{it}|\mathbf{Y}_i(t_{i,j})]$. Assuming that the measurement and process noise components are normally distributed and that the measurement equation is linear, as in Equation 6, the prediction errors, $\mathbf{Y}_i(t_{i,j}) - E(\mathbf{Y}_i(t_{i,j})|\mathbf{Y}_i(t_{i,j}))$, are multivariate normally distributed. Thus, these by-products of the KF can be used to construct a log-likelihood function known as the *prediction error decomposition* function (De Jong, 1988; Harvey, 1989; Hamilton, 1994; Chow et al., 2010). This log-likelihood function is optimized to yield maximum-likelihood (ML) estimates of all the time-invariant parameters, as well as to construct information criterion (IC) measures (Chow and Zhang, 2013; Harvey, 1989) such as the Akaike Information Criterion (AIC; Akaike, 1973) and Bayesian Information Criterion (BIC; Schwarz, 1978). Standard errors of the parameter estimates are obtained by taking the square root of the diagonal elements of the inverse of the negative numerical Hessian matrix of the prediction error decomposition function at the point of convergence.

At convergence, other products from the linear KF include updated latent states, $\hat{\boldsymbol{\eta}}_i(t_{i,j}|t_{i,j})$, and the updated latent covariance matrices, $\mathbf{P}_i(t_{i,j}|t_{i,j})$. In the social and behavioral sciences, the entire time series of observations has often been collected prior to model fitting. In such cases, we use the fixed interval smoother (Anderson and Moore, 1979; Ansley and Kohn, 1985) to refine the latent variable estimates, yielding the smoothed latent variable estimates, $\hat{\boldsymbol{\eta}}_i(t_{i,j}|T_i) = E(\boldsymbol{\eta}_i(t_{i,j})|\mathbf{Y}_i(T_i))$, and

associated covariance matrices, $\mathbf{P}_i(t_{i,j}|T_i)$.

When the dynamic model takes on the form of a nonlinear state-space model with differentiable dynamic functions, the linear KF is replaced with the extended Kalman filter (EKF; Anderson and Moore, 1979; Bar-Shalom et al., 2001) so that the nonlinear dynamic functions are "linearized" or approximated by the first-order Taylor series. Then, a log-likelihood function can be constructed in similar form to the linear state-space prediction error decomposition. However, the corresponding parameter estimates are only "approximate" ML estimates due to the truncation errors in the EKF. The feasibility of this approach has been demonstrated by Chow et al. (2007).

When a linear state-space model is used as the dynamic model but it is characterized by regime-switching properties, **dynr** uses an extension of the KF, known as the Kim filter, and the related Kim smoother (Kim and Nelson, 1999; Yang and Chow, 2010). The Kim filter combines the KF, the Hamilton filter (Hamilton, 1989) that yields filtered state probabilities, and a collapsing procedure to avoid the need to store M^2 new values of $\hat{\boldsymbol{\eta}}_i(t_{i,j}|t_{i,j})^{l,m} \triangleq \mathbb{E}[\boldsymbol{\eta}_i(t_{i,j})|S_i(t_{i,j-1}) = l, S_i(t_{i,j}) = m, \mathbf{Y}_i(t_{i,j})]$, as well as $\mathbf{P}_i(t_{i,j}|t_{i,j})^{l,m} \triangleq \mathrm{Cov}[\boldsymbol{\eta}_i(t_{i,j})|S_i(t_{i,j-1}) = l, S_i(t_{i,j}) = m, \mathbf{Y}_i(t_{i,j})]$ with each additional time point. The collapsing procedure averages the estimates over the previous regime l so only the marginal estimates, $\hat{\boldsymbol{\eta}}_i(t_{i,j}|t_{i,j})^m = E[\boldsymbol{\eta}_i(t_{i,j})|S_i(t_{i,j}) = m, \mathbf{Y}_i(t_{i,j})]$), and the associated covariance matrix, $\mathbf{P}_i(t_{i,j}|t_{i,j})^m$, need to be stored at each time step. To handle cases in which nonlinearities are present in Equation 3, a method proposed by Chow and Zhang (2013), called the extended Kim filter, is used for estimation instead. The extended Kim filter replaces the KF portion of the Kim filter with the EKF.

Continuous-time models

Finally, when the dynamics are in continuous time—whether composed of linear or nonlinear dynamic functions—the resultant estimation procedures are the continuous-discrete extended Kalman filter (CDEKF; Bar-Shalom et al., 2001; Kulikov and Kulikova, 2014; Kulikova and Kulikov, 2014). The CDEKF handles a single-regime special case of the general model shown in Equations 1–6.

For continuous processes in the form of Equation 1, let $\hat{\boldsymbol{\eta}}_i(t) = E(\boldsymbol{\eta}_i(t)|\mathbf{Y}_i(t_{i,j-1}))$ and $\mathbf{P}_i(t) = Cov[\boldsymbol{\eta}_i(t)|\mathbf{Y}_i(t_{i,j-1})]$ denote the mean and covariance matrix of the latent variables, respectively, at time t in the interval $[t_{i,j-1}, t_{i,j}]$. In the CDEKF framework, the prediction step of the KF is replaced by solving a set of ordinary differential equations (ODEs) at time $t_{i,j}$, given the initial conditions at time $t_{i,j-1}$: $\hat{\boldsymbol{\eta}}_i(t_{i,j-1}) = \hat{\boldsymbol{\eta}}_i(t_{i,j-1}|t_{i,j-1})$ and $\mathbf{P}_i(t_{i,j-1}) = \mathbf{P}_i(t_{i,j-1}|t_{i,j-1})$. This set of ODEs is obtained by only retaining the first term, $\boldsymbol{f}_{S_i(t)}\left(\hat{\boldsymbol{\eta}}_i(t), t, \boldsymbol{x}_i(t)\right)$, in the Taylor series expansion of $\boldsymbol{f}_{S_i(t)}\left(\boldsymbol{\eta}_i(t), t, \boldsymbol{x}_i(t)\right)$ around the expectation $\hat{\boldsymbol{\eta}}_i(t)$, and is shown below:

$$\frac{d\hat{\boldsymbol{\eta}}_i(t)}{dt} = \boldsymbol{f}_{S_i(t)}\left(\hat{\boldsymbol{\eta}}_i(t), t, \boldsymbol{x}_i(t)\right), \tag{9}$$

$$\frac{d\mathbf{P}_i(t)}{dt} = \frac{\partial \boldsymbol{f}_{S_i(t)}\left(\hat{\boldsymbol{\eta}}_i(t), t, \boldsymbol{x}_i(t)\right)}{\partial \hat{\boldsymbol{\eta}}_i(t)}\mathbf{P}(t) + \mathbf{P}(t)\left(\frac{\partial \boldsymbol{f}_{S_i(t)}\left(\hat{\boldsymbol{\eta}}_i(t), t, \boldsymbol{x}_i(t)\right)}{\partial \hat{\boldsymbol{\eta}}_i(t)}\right)^{\top} + \mathbf{Q}_{S_i(t)}, \tag{10}$$

where $\frac{\partial \boldsymbol{f}_{S_i(t)}(\hat{\boldsymbol{\eta}}_i(t), t, \boldsymbol{x}_i(t))}{\partial \hat{\boldsymbol{\eta}}_i(t)}$ is the Jacobian matrix of $\boldsymbol{f}_{S_i(t)}\left(\hat{\boldsymbol{\eta}}_i(t), t, \boldsymbol{x}_i(t)\right)$ with respect to $\hat{\boldsymbol{\eta}}_i(t)$ at time t. Kulikov and Kulikova (2014, Kulikova and Kulikov 2014) suggested solving for equations 9 and 10 using adaptive ODE solvers. We adopt an approximate numerical solution — the fourth-order Runge-Kutta (Press et al., 2002) method — to solve Equations 9 and 10. In cases where the hypothesized continuous-time dynamics are linear, explicit analytic solutions exist and there is no need to use numerical solvers. However, in our simulation work, estimating known special cases of linear stochastic differential equation models using numerical solvers yielded both comparable estimates and computational time to estimating the same models using their known solutions. Thus, for generality, we utilize numerical solvers in solving both linear and nonlinear differential equations in **dynr**.

As in the case involving nonlinear discrete-time dynamic models, parameter estimates obtained from optimizing the log-likelihood function constructed from by-products of the CDEKF are also approximate ML estimates; however, the approximations now stem both from the truncation errors from the first-order Taylor series in the CDEKF, as well as the numerical solution of Equations 9 and 10.

In cases involving regime-switching ordinary or stochastic differential equations, the algorithms for estimating regime-switching continuous-time models are essentially estimation procedures that combine the CDEKF and part of the Kim filter designed to handle estimation of the regime-switching portion of the model. The resultant procedure, referred to herein as *continuous-discrete extended Kim filter*, is summarized in Chow et al. (2018).

		Discrete-time	Continuous-time
Single-regime	Linear	Linear state-space model KF **dynr**, **OpenMx**, **pomp**, **KFAS**, **dlm**, **dse**, MKFM6, SsfPack, MATLAB	Linear SDE/ODE CDEKF **dynr**, **pomp**, **OpenMx**, **ctsem**, MATLAB
	Nonlinear	Nonlinear state-space model EKF **dynr**, **pomp**, SsfPack, MATLAB	Nonlinear SDE/ODE CDEKF **dynr**, **pomp**, MATLAB
Multiple-regime	Linear	RS state-space model Kim filter **dynr**, GAUSS code, MATLAB	RS SDE/ODE CD Kim filter **dynr** only
	Nonlinear	RS nonlinear state-space model Extended Kim filter **dynr** only	RS nonlinear SDE/ODE CD extended Kim filter **dynr** only

Table 1: Models, algorithms, and software for the framework of regime-switching (non)linear state space models in discrete- and continuous-time. SDE = Stochastic Differential Equation, ODE = Ordinary Differential Equation, CD = Continuous-Discrete, RS = Regime-Switching, KF = Kalman filter (Kalman, 1960), EKF = Extended Kalman filter (Anderson and Moore, 1979; Bar-Shalom et al., 2001), Kim filter = KF + Hamilton filter + Collapsing procedure (Kim and Nelson, 1999). Extended Kim filter was proposed by Chow and Zhang (2013); the CD extended Kim filter is proposed by Chow et al. (2018).

Steps for preparing and "cooking" a model

The theme around the naming convention exploits the pronunciation of the package name: **dynr** is pronounced the same as "dinner". Therefore, the names of functions and methods are specifically designed to relate to things done surrounding dinner, such as gathering ingredients such as the data, preparing recipes, cooking, which involves combining ingredients according to a "modeling" recipe and applies heat, and serving the finished product.

The general procedure for using the **dynr** package can be summarized in five steps. First, data are gathered and identified with the `dynr.data()` function. Second, *recipes* are prepared. To each part of a model there is a corresponding `prep.*()` recipe function. Each of these functions creates an object of class `"dynrRecipe"`. Each `prep.*()` function creates an object of class `"dynr*"` which is in turn a subclass of `"dynrRecipe"`. These recipe functions include:

1. The `prep.measurement()` function defines the measurement part of the model, that is, how latent variables and exogenous covariates map onto the observed variables.
2. The `prep.matrixDynamics()` and `prep.formulaDynamics()` functions define the dynamics of the model with either a strictly linear, matrix interface or with a possibly nonlinear formula interface, respectively.
3. The `prep.initial()` function defines the initial conditions of the model. The initial conditions are used by the recursive algorithms as the starting point for latent variable estimates. As such, the `prep.initial()` function describes the initial mean vector and covariance matrix of the latent variables, assumed to be multivariate normally distributed.
4. The `prep.noise()` function defines the covariance structure for both the measurement (or observation) noise and the dynamic (or latent) noise.
5. The `prep.regimes()` function provides the regime switching structure of the model. Single-regime models do not require a `"dynrRegimes"` object.

Once the data and recipes are prepared, the third step mixes the data and recipes together into a model object of class `"dynrModel"` with the `dynr.model()` function. Fourth, the model is cooked with `dynr.cook()` to estimate the free parameters and standard errors. Fifth and finally, results are served in summary tables using `summary()`, LaTeX equations using `printex()`, and plots of trajectories and equations using `plot()`, `dynr.ggplot()`, `autoplot()`, and `plotFormula()`.

We will demonstrate the interface of **dynr** using two examples: (1) a linear state-space example with regime-switching based on Yang and Chow (2010) and (2) a regime-switching extension of the predator-prey model (Lotka, 1925; Volterra, 1926).

Example 1: Regime-switching linear state-space model

Facial electromyography (EMG) has been used in the behavioral sciences as one possible indicator of human emotions (Schwartz, 1975; Cacioppo and Petty, 1981; Cacioppo et al., 1986; Dimberg et al., 2000). A time series of EMG data contains bursts of electrical activity that are typically magnified when an individual is under emotion induction. Yang and Chow (2010) proposed using a regime-switching linear state-space model in which the individual may transition between regimes with and without facial EMG activation. As such, heterogeneities in the dynamic patterns and variance of EMG data are also accounted for through the incorporation of these latent regimes. Model fitting was previously performed at the individual level. Data from the participant shown in Figure 1(A) are made available as part of the demonstrative examples in **dynr**. A complete modeling script for this example is available as a demo in **dynr** and can be found by calling `file.edit(system.file("demo","RSLinearDiscreteYang.R",package = "dynr"))`, and a full explanation is included as a package vignette called 'LinearDiscreteTimeModels'.

Here we present selected segments of code to showcase how a linear state-space model with regime-switching can be specified in **dynr**. The model of interest is the final model selected for this participant by Yang and Chow (2010):

$$y_i(t_{i,j}) = \mu_{yS_i(t_{i,j})} + \beta_{S_i(t_{i,j})}\text{Self-report}(t_{i,j}) + \eta_i(t_{i,j}), \tag{11}$$

$$\eta_i(t_{i,j+1}) = \phi_{S_i(t_{i,j})}\eta_i(t_{i,j}) + \zeta_i(t_{i,j+1}), \tag{12}$$

in which we allowed the intercept, $\mu_{yS_i(t_{i,j})}$; the regression slope, $\beta_{S_i(t_{i,j})}$; and the autoregression coefficient, $\phi_{S_i(t_{i,j})}$, to be regime-dependent. By allowing $\phi_{S_i(t_{i,j})}$ to be regime-specific, we indirectly allowed the total variance of the latent component, $\eta_i(t_{i,j+1})$, to be heterogeneous across the deactivation and activation stages, in spite of requiring the dynamic noise variance, $E(\zeta_i(t)^2)$, to be constant across regimes.

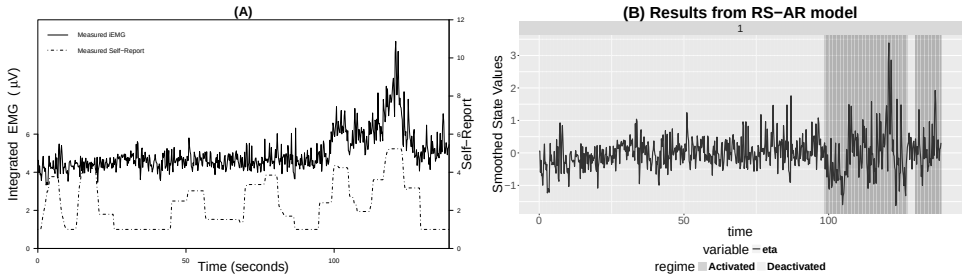

Figure 1: (A) A plot of integrated electromyography (iEMG) and self–report affect ratings for one participant with a time interval of 0.2 seconds between two adjacent observations. Self–report = self–report affect ratings; iEMG = integrated EMG signals. (B) An automatic plot of the smoothed state estimates for the regime-switching linear state-space model.

The first step in **dynr** modeling is to structure the data. This is done with the `dynr.data()` function.

```
require("dynr")
data("EMG")
EMGdata <- dynr.data(EMG, id = 'id', time = 'time',
  observed = 'iEMG', covariates = 'SelfReport')
```

The first argument of this function is either a `"ts"` class object of single-unit time series or a `"data.frame"` object structured in a long relational format with different measurement occasions from the same unit appearing as different rows in the data frame. When a `"ts"` class object is passed to `dynr.data()`, no other inputs are needed. Otherwise, the `id` argument needs the name of the variable that distinguishes units, allowing multiple replicated time series to be analyzed together. The `time` argument needs the name of the variable that indicates unit-specific measurement occasions. If a discrete-time model is desired, the time variable should contain sequential positive integers. If the measurement occasions for a unit are sequential but not consecutive, `NA`s will be inserted automatically

to create equally spaced data. If a continuous-time model is being specified, the time variable can contain unit-specific increasing sequences of irregularly spaced real numbers. In this particular example, a discrete-time model is used. The `observed` and `covariates` arguments are vectors of the names of the observed variables and covariates in the data.

The next step in **dynr** modeling is to build the recipes for the various parts of a model. The recipes are created with `prep.*()` functions.

The dynamic functions in Equations 1 and 3, can be specified using either `prep.formulaDynamics()` or `prep.matrixDynamics()`. In this example, the dynamics as in Equation 12 are linear and discrete-time, so we can describe the dynamics in terms of Equation 4 as

$$\eta_i(t_{i,j+1}) = \underbrace{0}_{\boldsymbol{\alpha}_{S_i(t_{i,j})}} + \underbrace{\phi_{S_i(t_{i,j})}}_{\boldsymbol{F}_{S_i(t_{i,j})}} \eta_i(t_{i,j}) + \underbrace{0}_{\boldsymbol{B}_{S_i(t_{i,j})}} \boldsymbol{x}_i(t_{i,j}) + \underbrace{\zeta_i(t_{i,j+1})}_{\boldsymbol{w}_i(t_{i,j+1})}. \tag{13}$$

The `prep.matrixDynamics()` function allows the user to specify the structures of the intercept vector $\boldsymbol{\alpha}_{S_i(t_{i,j})}$, through `values.int` and `params.int`; the covariate regression matrix $\boldsymbol{B}_{S_i(t_{i,j})}$, through `values.exo` and `params.exo`; and the one-step-ahead transition matrix $\boldsymbol{F}_{S_i(t_{i,j})}$, through `values.dyn` and `params.dyn`. We illustrate this function below. The `values.dyn` argument gives a list of matrices for the starting values of $\boldsymbol{F}_{S_i(t_{i,j})}$. The `params.dyn` argument names the free parameters. These are the ϕ_{S_i} in Equation 12. The `isContinuousTime` argument switches between continuous-time modeling and discrete-time modeling. The arguments corresponding to the intercepts (`values.int` and `params.int`) and the covariate effects (`values.exo` and `params.exo`) are omitted to leave these matrices as zeros.

```
recDyn <- prep.matrixDynamics(values.dyn = list(matrix(0.1, 1, 1), matrix(0.5, 1, 1)),
  params.dyn = list(matrix('phi_1', 1, 1), matrix('phi_2', 1, 1)),
  isContinuousTime = FALSE)
```

The noise recipe is created with `prep.noise()`. The noise recipe is stored in the `recNoise` object, an abbreviation for "recipe noise". The latent noise covariance matrix is a 1×1 matrix with a free parameter called `dynNoise`, short for "dynamic noise". The observed noise covariance matrix is also a 1×1 matrix, but has the measurement noise variance fixed to zero using the special keyword `fixed`.

```
recNoise <- prep.noise(values.latent = matrix(1, 1, 1),
  params.latent = matrix('dynNoise', 1, 1),
  values.observed = matrix(0, 1, 1), params.observed = matrix('fixed', 1, 1))
```

The `prep.regimes()` function specifies the structure of the regime time evolution shown in Equation 8. In this example, we do not have any covariates in the regime-switching (RS) functions. The problem then reduces to the specification of a 2×2 transition log-odds (LO) matrix. We provide starting values that imply persisting in the same regime is more likely than transitioning to another regime, and set the second regime LO to zero for identification, making it the reference regime. The first column of the transition LO matrix, is populated with the starting values of: (1) `c11` $= 0.7$, corresponding to $exp(0.7) = 2.01$ times greater LO of staying within the Deactivated regime as transitioning to the Activated regime; and (2) `c21` $= -1$, corresponding to $exp(-1) = 0.37$ times lower LO of transitioning to the Deactivated regime.

```
recReg <- prep.regimes(values = matrix(c(0.7, -1, 0, 0), 2, 2),
  params = matrix(c('c11', 'c21', 'fixed', 'fixed'), 2, 2))
```

In essence, the above code creates the following transition probability matrix:

$$\begin{array}{c} \\ Deactivated_{t_{i,j}} \\ Activated_{t_{i,j}} \end{array} \begin{array}{c} \begin{array}{cc} Deactivated_{t_{i,j+1}} & Activated_{t_{i,j+1}} \end{array} \\ \left(\begin{array}{cc} \frac{exp(c_{11})}{exp(c_{11})+exp(0)} & \frac{exp(0)}{exp(c_{11})+exp(0)} \\ \frac{exp(c_{21})}{exp(c_{21})+exp(0)} & \frac{exp(0)}{exp(c_{21})+exp(0)} \end{array} \right) \Bigg|_{c_{11}=.7 \;\; c_{12}=-1} \end{array} = \begin{array}{c} \\ D_{t_{i,j}} \\ A_{t_{i,j}} \end{array} \begin{array}{c} \begin{array}{cc} D_{t_{i,j+1}} & A_{t_{i,j+1}} \end{array} \\ \left(\begin{array}{cc} 0.668 & 0.332 \\ 0.269 & 0.731 \end{array} \right). \end{array} \tag{14}$$

In many situations it is useful to specify the structure of the transition LO matrix in deviation form — that is, to express the LO intercepts in all but the reference regime as deviations from the LO intercept in the reference regime. The package vignette illustrates this by invoking the `deviation` argument of `prep.regimes()`.

After the recipes for all parts of the model are defined, the `dynr.model()` function creates the model and stores it in the `"dynrModel"` object. Each recipe object created by `prep.*()` and the data prepared by `dynr.data()` are given to this function. The `dynr.model()` function always requires `dynamics`, `measurement`, `noise`, `initial`, and `data`. When there are multiple regimes, the `regimes` argument should also be provided. When parameters are subject to transformation functions, a `transform`

argument can be added, which will be discussed in the second example. The `dynr.model()` function combines information from the recipes and data to write the text for a C function. This text is written to a file optionally named by the `outfile` argument, so that the user can inspect or modify the generated C code. The default `outfile` is a temporary file returned by `tempfile()`.

```
rsmod <- dynr.model(dynamics = recDyn, measurement = recMeas,
  noise = recNoise, initial = recIni, regimes = recReg,
  data = EMGdata, outfile = "RSLinearDiscreteYang.c")
yum <- dynr.cook(rsmod)
```

In the last line above, the model is "cooked" with the `dynr.cook()` function to estimate the free parameters and their standard errors. When cooking, the C code in the `outfile` is compiled and dynamically linked to the rest of the compiled **dynr** code. If the C functions have previously been compiled then the user can prevent re-compilation by setting `compileLib = FALSE` in the `"dynrModel"` object given to `dynr.cook()`. After compilation the C code is executed to optimize the free parameters while calling the dynamically linked C functions that were created from the user-specified recipes. In this way, **dynr** provides an R interface for dynamical systems modeling while maintaining much of the speed associated with C.

The final step associated with **dynr** modeling is serving results (a `"dynrCook"` object) after the model has been cooked. To this end, several standard, popular S3 methods are defined for the `"dynrCook"` class, including `coef()`, `confint()`, `deviance()`, `logLik()`, `AIC()`, `BIC()`, `names()`, `nobs()`, `summary()`, and `vcov()`. These methods perform the same tasks as their counterparts for regression models in R. Additionally, **dynr** provides a few other model-serving functions illustrated here: `summary()`, `plot()`, `dynr.ggplot()` (or `autoplot()`), `plotFormula()`, and `printex()`. The `summary()` method provides a table of free parameter names, estimates, standard errors, *t*-values, and Wald-type confidence intervals.

```
summary(yum)

Coefficients:
         Estimate Std. Error t value ci.lower ci.upper Pr(>|t|)
phi_1     0.26608    0.04953   5.372  0.16900  0.36315 5.33e-08 ***
phi_2     0.47395    0.04425  10.711  0.38722  0.56068  < 2e-16 ***
beta_2    0.46449    0.04394  10.571  0.37837  0.55061  < 2e-16 ***
mu_1      4.55354    0.02782 163.658  4.49901  4.60807  < 2e-16 ***
mu_2      4.74770    0.14250  33.318  4.46842  5.02699  < 2e-16 ***
dynNoise  0.20896    0.01129  18.504  0.18683  0.23110  < 2e-16 ***
c11       5.50199    0.70939   7.756  4.11160  6.89237  < 2e-16 ***
c21      -5.16170    1.00424  -5.140 -7.12998 -3.19342 1.79e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

-2 log-likelihood value at convergence = 1002.52
AIC = 1018.52
BIC = 1054.87
```

These parameter estimates, standard errors, and likelihood values closely mirror those reported in Yang and Chow (2010, p. 755-756). In the Deactivated regime, the autoregressive parameter (`phi_1`) and the intercept (`mu_1`) are lower than those in the Activated regime. So, neighboring EMG measurements are more closely related in the Activated regime and the overall level is slightly higher. This matches very well with the idea that the Activated regime consists of bursts of facial muscular activities and an elevated emotional state. Similarly, the effect of the self-reported emotional level is positive in the Activated regime and fixed to zero in the Deactivated regime, as the freely estimated value was close to zero with a nonsignificant *t*-value. So, in the Deactivated regime the self-reported emotional level and the facial muscular activity decouple. The dynamic noise parameter gives a sense of the size of the intrinsic unmeasured disturbances that act on the system. These forces perturb the system with a typical magnitude of a little less than half a point on the EMG scale seen in Figure 1(A). Lastly, the log-odds parameters (`c11` and `c21`) can be turned into the transition probability matrix yielding

$$\begin{array}{c} \\ Deactivated_{t_{i,j}} \\ Activated_{t_{i,j}} \end{array} \begin{array}{c} \begin{array}{cc} Deactivated_{t_{i,j+1}} & Activated_{t_{i,j+1}} \end{array} \\ \begin{pmatrix} 0.9959 & 0.0041 \\ 0.0057 & 0.9943 \end{pmatrix}. \end{array} \tag{15}$$

which implies that both the Deactivated and the Activated regimes are strongly persistent with high self-transistion probabilities. Next we consider some of the visualization options for serving a model.

The default `plot()` method is used to visualize the time series in a collection of plots: (1) a plot of time series created by `dynr.ggplot()` (or `autoplot()`), (2) a histogram of predicted regimes, and (3) a plot of equations created by `plotFormula()`.

```
plot(yum, dynrModel = rsmod, style = 1, textsize = 5)
```

The `dynr.ggplot()` (or `autoplot()`) method creates a plot of the smoothed state estimates with the predicted regimes. It needs the result object and model object as inputs, and allows for plotting (1) user-selected smoothed state variables by default or (2) user-selected observed-versus-predicted values by setting `style = 2`. An illustrative plot is created from the code below and shown in Figure 1(B).

```
dynr.ggplot(yum, dynrModel = rsmod, style = 1,
  names.regime = c("Deactivated", "Activated"),
  title = "(B) Results from RS-AR model", numSubjDemo = 1,
  shape.values = 1, text = element_text(size = 24), is.bw = TRUE)
```

This shows that for the first 99 seconds the participant is in the Deactivated regime, with their latent state $\eta_i(t_{i,j+1})$ varying according to the lower autocorrelation model and having no relation to the variation in the self-reported emotional data in Figure 1(A). Then the participant switches to the Activated regime and their latent state becomes more strongly autocorrelated and coupled to the self-report data. There follows a brief period in the Deactivated regime around time=130 seconds with a subsequent return to the Activated regime for the remainder of the observation. Of course, note that Figure 1(A) shows the observed EMG data whereas Figure 1(B) shows the latent state which is related to the observed data by Equation 11.

The `plotFormula()` method can be used to display model equations on R plots. Equations can be viewed in several ways with different inputs to the `ParameterAs` argument: (1) with free parameter names, for example, returned by `names(rsmod)`, as illustrated in Figure 2(A); (2) with parameter starting values; or (3) after estimation with fitted parameter values, for example, returned by `coef(yum)`, as in Figure 2(B). The `plotFormula()` method does not require the user to install LaTeX facilities and compile LaTeX code in a separate step, and hence are convenient to use. To maximize the readability of the equations, it is only shown here using equations for the dynamic and measurement models, which can be obtained by respectively setting the `printDyn` and `printMeas` arguments to true.

```
plotFormula(dynrModel = rsmod, ParameterAs = names(rsmod),
  printDyn = TRUE, printMeas = TRUE) + ggtitle("(A)") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.01, size = 16))

plotFormula(dynrModel = rsmod, ParameterAs = coef(yum),
  printDyn = TRUE, printMeas = TRUE) + ggtitle("(B)") +
  theme(plot.title = element_text(hjust = 0.5, vjust = 0.01, size = 16))
```

We can see that the equations in Figure 2(A) are precisely those from Equations 11 and 12 which we used to define the model except that we have fixed β_1 to zero. If these equations did not match, it may indicate that we made a mistake in our model specification.

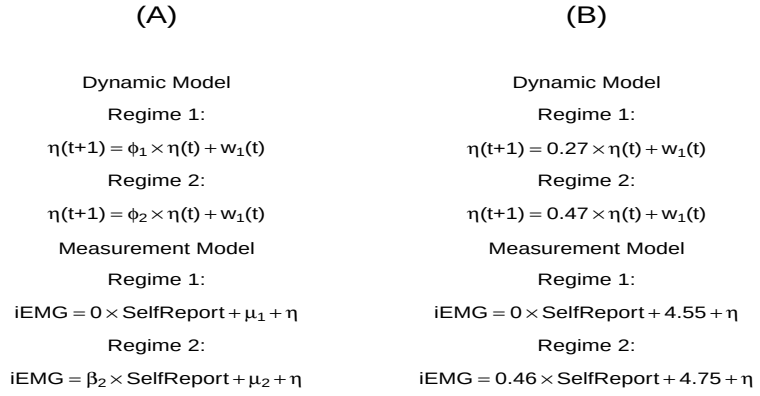

Figure 2: Automatic plots of model equations with (A) parameter names and (B) estimated parameters for the regime-switching linear state-space model.

Finally, for LaTeX users, the `printex()` method helps generate equations for the model in LaTeX form.

```
printex(rsmod, ParameterAs = names(rsmod), printInit = TRUE, printRS = TRUE,
  outFile = "RSLinearDiscreteYang.tex")
```

The `ParameterAs` argument functions the same as that in the `plotFormula()` method. Here we have specified to use the names of the free parameters. In this case, the initial conditions and regime-switching functions are included in the equations, as indicated by the `printInit` and `printRS` arguments being set to `TRUE`. The LaTeX code for the equations is written to 'RSLinearDiscreteYang.tex', which the user can then work with and modify as they wish. Of course, this function is designed more as a convenience feature for users who already use LaTeX and requires all the LaTeX-related facilities on the user's computer.

This example has used real EMG data from a previous study (Yang and Chow, 2010) to illustrate many parts of the user-interface for **dynr**. Of particular note are the various "serving" functions which allow users to both verify their model and examine their results in presentation-ready formats. In the next example, we will use simulated data to further illustrate features of **dynr**, especially the nonlinear formula interface for dynamics.

Example 2: Nonlinear continuous-time models

In the study of human dynamics many processes are characterized by changes that are dependent on interactions with other processes producing dynamics with nonlinearities. Nonlinear ordinary differential equations have been used to model, among other phenomena, ovulatory regulation (Boker et al., 2014), circadian rhythms (Brown and Luithardt, 1999), cerebral development (Thatcher, 1998), substance use (Boker and Graham, 1998), cognitive aging (Chow and Nesselroade, 2004), parent-child interactions (Thomas and Martin, 1976), couple dynamics (Chow et al., 2007; Gottman, 2002), and sudden transitions in attitudes (van der Maas et al., 2003).

Single-regime nonlinear continuous-time model

In addition to the linear/matrix dynamics interface, **dynr** also provides users with a formula interface to accommodate nonlinear as well as linear dynamic functions. To illustrate the use of the formula interface in **dynr**, we use a benchmark nonlinear ordinary differential equation model, the predator-prey model (Lotka, 1925; Volterra, 1926; Hofbauer and Sigmund, 1988). One can find the complete demo scripts in **dynr**, using `file.edit(system.file("demo","NonlinearODE.R",package = "dynr"))` and `file.edit(system.file("demo","RSNonlinearODE.R",package = "dynr"))`, and related explanation in the package vignette 'NonlinearContinuousTimeModels'.

The predator-prey model is a classic model for representing the nonlinear dynamics of interacting populations. The most often cited behavior of the predator-prey system while in a particular parameter range is ongoing nonlinear oscillations in the predator and prey populations with a phase lag between them. The utility of the predator-prey model extends far beyond the area of population dynamics. Direct applications or extensions of this predator-prey system include the epidemic models of the onset of social activities (EMOSA) used to study the spread of smoking, drinking, delinquency, and sexual behaviors among adolescents (Rodgers and Rowe, 1993; Rodgers et al., 1998); the cognitive aging model (Chow and Nesselroade, 2004); and the model of couples' affect dynamics (Chow et al., 2007).

Written as a differential equation, the predator-prey model is expressed as:

$$d(prey(t)) = (a\ prey(t) - b\ prey(t)\ predator(t))\, dt, \tag{16}$$

$$d(predator(t)) = (-c\ predator(t) + d\ prey(t)\ predator(t))\, dt, \tag{17}$$

where the parameters a, b, c, d are all nonnegative. These equations make up the continuous-time dynamics, Equation 1, for this system. Examining the prey equation (Equation 16), the prey population would increase exponentially without bound if there were zero predators. Similarly, examining the predator equation (Equation 17), if the prey population was zero, then the predator population would decrease exponentially to zero. For demonstration purposes, we have included with the **dynr** package a set of simulated data generated with true parameter values: $a = 2, b = 1, c = 4, d = 1, e = .25, f = 5$.

Using the formula interface in **dynr**, which supports all native mathematical functions available in R, the predator-prey model can be specified as:

```
preyFormula <- prey ~ a * prey - b * prey * predator
predFormula <- predator ~ - c * predator + d * prey * predator
ppFormula <- list(preyFormula, predFormula)
ppDynamics <- prep.formulaDynamics(formula = ppFormula,
  startval = c(a = 2.1, c = 0.8, b = 1.9, d = 1.1), isContinuousTime = TRUE)
```

The first argument of the `prep.formulaDynamics()` function is `formula`. More specifically, this is a list of formulas. Each element in the list is a single, univariate, formula that defines a differential (if `isContinuousTime` = TRUE) or difference (if `isContinuousTime` = FALSE) equation. There should be one formula for every latent variable, in the order in which the latent variables are specified by using the `state.names` argument in `prep.measurement()`. The left-hand side of each formula is either the one-step-ahead projection or the differential of the latent variable: namely, the left-hand side of Equations 1 and 3, respectively. In both cases, users only need to specify the names of the latent variables that match the specification in `prep.measurement()` on the left-hand side of the formulas. The right-hand side of each formula gives a linear or nonlinear function that may involve free or fixed parameters, numerical constants, exogenous covariates, and other arithmetic/mathematical functions that define the dynamics of the latent variables. The `startval` argument is a named vector giving the names of the free parameters and their starting values. Just as in the `prep.matrixDynamics()` function, the `isContinuousTime` argument is a binary flag that switches between continuous- and discrete-time modeling. The rest of **dynr** code for fitting the predator-prey model can be specified in similar ways to the code shown in Example 1 and is omitted here for space constraints. A fully functional demo script can be found in **dynr**, using `file.edit(system.file("demo","NonlinearODE.R",package = "dynr"))`, and further comments are included as a package vignette.

With the formula interface, **dynr** uses the `D()` function to symbolically differentiate the formulas provided. Hence, **dynr** uses the analytic Jacobian of the dynamics in its extended Kalman filter, greatly increasing its speed and accuracy. The `D()` function can handle the differentiation of functions involving parentheses, arithmetic operators, for instance, $+$, $-$, $*$, $/$, and `^`, and numerous mathematical functions such as `exp()`, `log()`, `sin()`, `cos()`, `tan()`, `sinh()`, `cosh()`, `sqrt()`, `pnorm()`, `dnorm()`, `asin()`, `acos()`, `atan()`, and `gamma()`. Thus, for a very large class of nonlinear functions, the user is spared from supplying the analytic Jacobian of the dynamic functions. However, symbolic differentiation will not work for all formulas. For instance, formulas involving the absolute value function cannot be symbolically differentiated. For formulas that cannot be differentiated symbolically, the user must provide the analytic first derivatives through the `jacobian` argument. One can use `file.edit(system.file("demo","RSNonlinearDiscrete.R",package = "dynr"))` to find an example. An explanation is also included as a package vignette.

Regime-switching extension

Just as with the `prep.matrixDynamics()`, the formula interface also allows for regime-switching functionality. Consider an extension of the classical predator-prey model that lets the prey and predator interaction follow seasonal patterns. In the Summer regime, we have the predator-prey model as previously described, but in the Winter regime we now have a predator-prey model characterized by within-species competition and limiting growth/decay. In this competitive predator-prey model, the two populations do not grow/decline exponentially without bound in absence of the other, but rather, they grow logistically up to some finite carrying capacity. This logistic growth adds to the between-species interactions with the other population. This model can be specified as:

```
cPreyF <- prey ~ a * prey - e * prey ^ 2 - b * prey * predator
cPredF <- predator ~ f * predator - c * predator ^ 2 + d * prey * predator
cpFormula <- list(cPreyF, cPredF)
```

where the predator and prey equations are combined and supplied as a list.

To specify the regime-switching predator-prey model, we combine the classical predator-prey model and the predator-prey model with within-species competition into a list of lists. Then we provide this list to the usual `prep.formulaDynamics()` function as the `formula` argument.

```
rsFormula <- list(ppFormula, cpFormula)
dynm <- prep.formulaDynamics(formula = rsFormula,
   startval = c(a = 2.1, c = 3, b = 1.2, d = 1.2, e = 1, f = 2),
   isContinuousTime = TRUE)
```

Many dynamic models only lead to permissible values in particular parameter ranges. As such, we often need to add box constraints to model parameters. This is accomplished by setting bounds on the parameters as shown in the next section. An alternative in **dynr** is to apply unconstrained optimization to a transformed set of parameters. This latter strategy uses `prep.tfun()`. For example, the $a - f$ parameters should take on positive values. Thus, we may choose to optimize their log-transformed values and exponentiate the unconstrained parameter values during likelihood evaluations to ensure that their values are always positive. To achieve this, we supply a list of transformation formulas to the `formula.trans` argument in the prep.tfun() function as follows:

```
tformList <- list(a ~ exp(a), b ~ exp(b), c ~ exp(c),
  d ~ exp(d), e ~ exp(e), f ~ exp(f))
```

```
tformInvList <- list(a ~ log(a), b ~ log(b), c ~ log(c),
  d ~ log(d), e ~ log(e), f ~ log(f))
trans <- prep.tfun(formula.trans = tformList, formula.inv = tformInvList)
```

In cases involving transformation functions, the delta method is used to yield standard error estimates for the parameters on the constrained scales. If the starting values of certain parameters are indicated on a constrained scale, the `formula.inv` argument should then give a list of inverse transformation formulas.

In our hypothetical example, we have discussed how the weather condition may govern the regime switching processes. Specifically, we assume a covariate `cond` (with a value of 0 indicating the warmer weather and 1 indicating the colder weather) has an effect on the regime-switching transition probabilities. Then, we can specify the logistic regression model by

```
regimes <- prep.regimes(
  values = matrix(c(0, 0, -1, 1.5, 0, 0, -1, 1.5), nrow = 2, ncol = 4, byrow = TRUE),
  params = matrix(c("fixed", "fixed", "int_1", "slp_1",
  "fixed", "fixed", "int_2", "slp_2"), nrow = 2, ncol = 4, byrow = TRUE),
  covariates = "cond")
```

In essence, the above code creates a matrix in the following form:

$$\begin{pmatrix} c_{11}=0 & d_{11}=0 & c_{12}=int_1=-1 & d_{12}=slp_1=1.5 \\ c_{21}=0 & d_{21}=0 & c_{22}=int_2=-1 & d_{22}=slp_2=1.5 \end{pmatrix}, \tag{18}$$

which in turn creates the following transition probability matrix:

$$\begin{array}{c} \\ Summer_{t_{i,j}} \\ Winter_{t_{i,j}} \end{array} \begin{array}{c} \begin{array}{cc} Summer_{t_{i,j+1}} & Winter_{t_{i,j+1}} \end{array} \\ \begin{pmatrix} \frac{exp(0+0\times cond)}{exp(0+0\times cond)+exp(int_1+slp_1\times cond)} & \frac{exp(int_1+slp_1\times cond)}{exp(0+0\times cond)+exp(int_1+slp_1\times cond)} \\ \frac{exp(0+0\times cond)}{exp(0+0\times cond)+exp(int_2+slp_2\times cond)} & \frac{exp(int_2+slp_2\times cond)}{exp(0)+exp(int_2+slp_2\times cond)} \end{pmatrix} \end{array}. \tag{19}$$

Here we consider the Summer regime as the reference regime, so the first two columns of the transition LO matrix (Equation 18) are fixed at zero. The third and fourth columns of the transition LO matrix respectively correspond to the regression intercepts and slopes associated with the covariate, whose starting values are respectively set at -1 and 1.5. With this set of starting values, the transition probability from any regime to the Summer regime is 0.73 when $\texttt{cond} = 0$, and 0.38 when $\texttt{cond} = 1$. The negative intercept implies that in warmer days ($\texttt{cond} = 0$), there is a greater chance of the process transitioning into the Summer regime, and the regression slope greater than the absolute value of the intercept suggests that in colder days ($\texttt{cond} = 1$), the transition into the Winter regime is more likely.

We fitted the specified model to the simulated data. Figure 3 is created from the `dynr.ggplot()` (or `autoplot()`) method with `style = 2`, and shows that the predicted trajectories match with the observed values and alternate between different regimes.

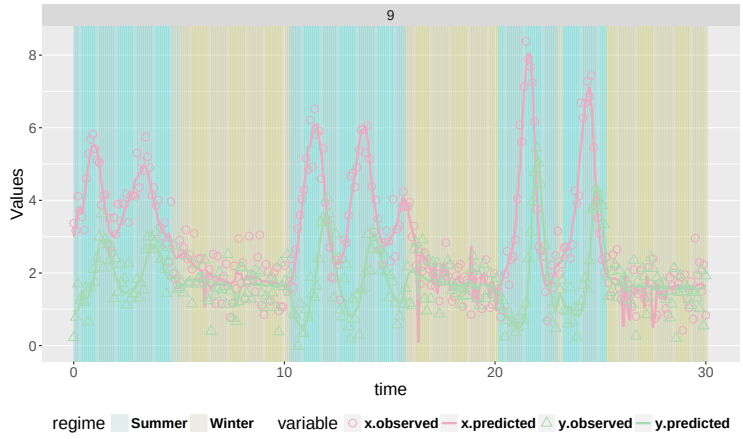

Figure 3: Built-in plotting feature for the predicted trajectories with observed values for the regime-switching nonlinear ODE model.

Other miscellaneous control options

In parameter estimation, **dynr** utilizes a sequential quadratic programming algorithm (Kraft, 1988, 1994) available from an open-source library for nonlinear optimization — NLOPT (Johnson, 2008). By default, we do not set boundaries on the free parameters. However, one can set the upper and lower bounds by respectively modifying the `ub` and `lb` slots of the model object. An example is given below to constrain the `int_1` and `int_2` parameters to be between -10 and 0, while limiting `slp_1` and `slp_2` to be between 0 to 10:

```
model2.2$ub[ c("int_1", "int_2", "slp_1", "slp_2") ] <- c(0, 0, 10, 10)
model2.2$lb[ c("int_1", "int_2", "slp_1", "slp_2") ] <- c(-10, -10, 0, 0)
```

Similarly, the stopping criteria of the optimization can be modified through the `options` slot of the `"dynrModel"` object, which is a list consisting of the relative tolerance on optimization parameters `xtol_rel`; the stopping threshold of the objective value `stopval`; the absolute and relative tolerance on function value, `ftol_abs` and `ftol_rel`; the maximum number of function evaluations `maxeval`; and the maximum optimization time `maxtime`.

If there is no need to re-compile the C functions in a call to `dynr.cook()`, the user can change the `compileLib` slot of the `"dynrModel"` object from default true to false. The output of the estimation function, `dynr.cook()`, is an object of class `"dynrCook"`. It not only includes estimation results displayed with `summary()`, but also contains information on posterior regime probabilities in the `pr_t_given_T` slot, smoothed state estimates $\hat{\boldsymbol{\eta}}_i(t_{i,j}|T_i) = E(\boldsymbol{\eta}_i(t_{i,j})|\mathbf{Y}_i(T_i))$ of the latent variables in the `eta_smooth_final` slot, and smoothed error covariance matrices $\mathbf{P}_i(t_{i,j}|T_i)$ of the latent variables in the `error_cov_smooth_final` slot, at all available time points. They can be retrieved by using the `$` operator.

Discussion and conclusions

This paper has introduced the **dynr** package that attempts to carefully balance intuitive usability with flexibility in the specification to satisfy the need of the broad social and behavioral science community. **dynr** offers linear and nonlinear time series methods for latent variables in both the traditional discrete-time models and in the hybrid continuous-time models that have discrete measurements with continuous underlying processes. Moreover, regime-switching can be layered on top of any aspect of these models.

Even though **dynr** can specify some models that other programs cannot, all of the features of other programs that exist for time series modeling are not subsets of **dynr**. For example, **KFAS** allows for nonlinear measurement (Helske, 2017a) which is not currently possible in **dynr**. Moreover, SsfPack has nonlinear measurement capabilities along with many MCMC methods that **dynr** lacks (Koopman et al., 1999). The **pomp** package has also implemented several algorithms absent in **dynr**, including MCMC methods, Bayesian methods, particle filtering, as well as ensemble filtering and forecasting. However, to our knowledge, no other software allows for regime-switching nonlinear dynamics with latent variables.

The **dynr** package highlighted the use of *recipe* objects to prepare components of the model. The recipes divide the full model into meaningful conceptual chunks for ease of specification and interactive inspection. The recipes seamlessly handle various bookkeeping tasks like the creation and management of the free parameter vector and how free parameters map onto model components. This is in contrast to several other packages offload this management on the user, often writing their own functions in the process. In addition to sparing the user sundry bothersome tasks, the recipes allow for interactive error checking and model verification using standard commands that should already be familiar to users of R. The contents of each recipe can be printed in the R console, letting the user verify that the recipe they intended to specify was actually created. Along this vein, `plotFormula()` allows the user to see nicely formatted equations for their models directly in R, and `printex()` outputs LaTeX equations for their models which can be typeset immediately or modified for inclusion in manuscripts, presentations, and reports.

The **dynr** package critically depends on several data structures and methods from the GNU Scientific Library (GSL; GSL Project Contributors, 2010) for fast and accurate scientific computing, and consequently requires the user to install GSL on their system. We wanted to allow users the flexibility of specifying their own models, while not sacrificing computational speed that would be influenced by frequent interchanges between R and C functions. Thus, **dynr** requires that users generate and compile the C code "on the fly", and pass the C function pointers to the back-end directly. Hence, **dynr** has a nontrivial set-up cost as compared to other R packages. However, to alleviate this burden we have written an installation and configuration guide as a vignette labeled 'InstallationGuideForUsers'. We generally find set-up of **dynr** to be similar to that of other packages that allow "on the fly" compilation

and ready C interfaces like **Rcpp** (Eddelbuettel and Francois, 2011; Eddelbuettel et al., 2018) and **RcppGSL** (Eddelbuettel and Francois, 2018).

Alternative computational strategies tended to worsen performance, increase user burden for model specification, or simply trade one difficult configuration task for another. In **dynr** the user only needs to specify a possibly nonlinear model of interest using standard R syntax. By contrast with **Rcpp**/**RcppGSL** the user would have to write C functions and hand differentiate their nonlinear dynamics functions: an error-prone process with a much steeper learning curve that acts as a deterrent to adoption, particularly to many researchers in the social and behavioral sciences. Additionally, we have found that automatic generation of a model specification file coded in C provides more sophisticated users with the opportunity to define modeling variations directly in C that are not already supported by the R interface functions.

Currently **dynr** only allows nonlinearity in the dynamics but not the measurement model to capitalize on the availability of a Gaussian approximate log-likelihood function for fast parameter estimation. Future extensions will incorporate Markov chain Monte Carlo (MCMC) techniques (Chow et al., 2011; Durbin and Koopman, 2001; Kim and Nelson, 1999; Lu et al., 2015) and pertinent frequentist-based estimation techniques (Fahrmeir and Tutz, 1994) to accommodate a broader class of measurement models consisting of nonlinear functions and non-Gaussian densities. In addition, several other extensions are being pursued and implemented in the **dynr** package. For example, **dynr** currently handles missingness in the dependent variables via full-information maximum likelihood but does not allow for missingness in the covariates. Future plans include interfacing **dynr** with R packages such as **mice** (van Buuren and Groothuis-Oudshoorn, 2011, 2017) to handle missingness in the covariates and/or dependent variables via multiple imputation. Further, models with nonlinearities at the dynamic level currently are not supported by well-established fit indices. Although **dynr** provides AIC (Akaike, 1973) and BIC (Schwarz, 1978) for model comparison purposes, the tenability of using these criteria when nonlinearities at the dynamic level are present and the optimized log-likelihood function involves approximations and truncation errors is yet to be investigated. Finally, even though difference and differential equations have served as and remain one of the most popular modeling tools across myriad scientific disciplines, their use is still nascent in many social and behavioral sciences. Tools to aid model developments and explorations are important extensions to enable and promote modeling efforts utilizing difference/differential equations (Chow et al., 2016; Ramsay et al., 2009). Fortunately, several existing packages in R offer many of the functionalities to support these modeling endeavors and may be used in conjunction or interfaced in the future with **dynr** for these purposes.

Bibliography

H. Akaike. Information theory and an extension of the maximum likelihood principle. In B. N. Petrov and F. Csaki, editors, *Second International Symposium on Information Theory*, pages 267–281. Akademiai Kiado, Budapest, 1973. [p4, 15]

B. D. O. Anderson and J. B. Moore. *Optimal Filtering*. Prentice Hall, Englewood Cliffs, NJ, 1979. [p4, 5, 6]

C. F. Ansley and R. Kohn. Estimation, filtering and smoothing in state space models with incompletely specified initial conditions. *The Annals of Statistics*, 13:1286–1316, 1985. [p4]

D. Ardia, K. Bluteau, K. Boudt, L. Catania, B. Peterson, and D.-A. Trottier. *Markov-Switching GARCH Models in R: The MSGARCH*, 2017. R package version 1.3. [p2]

Y. Bar-Shalom, X. R. Li, and T. Kirubarajan. *Estimation with Applications to Tracking and Navigation: Theory Algorithms and Software*. John Wiley & Sons, New York, NY, 2001. [p4, 5, 6]

S. M. Boker and J. Graham. A dynamical systems analysis of adolescent substance abuse. *Multivariate Behavioral Research*, 33(4):479–507, 1998. URL https://doi.org/10.1207/s15327906mbr3304_3. [p11]

S. M. Boker, M. C. Neale, and K. L. Klump. A differenial equations model for the ovarian hormone cycle. In P. C. M. Molenaar, R. M. Lerner, and K. M. Newell, editors, *Handbook of Developmental Systems Theory and Methodology*, pages 369–391. Guilford Press, New York, NY, 2014. [p11]

S. M. Boker, M. C. Neale, H. H. Maes, M. Spiegel, T. R. Brick, R. Estabrook, T. C. Bates, R. J. Gore, M. D. Hunter, J. N. Pritikin, M. Zahery, and R. M. Kirkpatrick. *OpenMx: Extended Structural Equation Modelling*, 2017. URL https://CRAN.R-project.org/package=OpenMx. R package version 2.8.3. [p1]

N. Bolger and J.-P. Laurenceau. *Intensive Longitudinal Methods: An Introduction to Diary and Experience Sampling Research*. Guilford Press, New York, NY, 2013. [p1]

P. Brandt. *MSBVAR: Markov-Switching, Bayesian, Vector Autoregression Models*, 2016. URL https://CRAN.R-project.org/package=MSBVAR. R package version 0.9-3. [p2]

E. N. Brown and H. Luithardt. Statistical model building and model criticism for human circadian data. *Journal of Biological Rhythms*, 14:609–616, 1999. URL https://doi.org/10.1177/074873099129000975. [p11]

B. Byrom and B. Tiplady. *ePRO: Electronic Solutions for Patient-Reported Data*. Gower, Farnham, England, 2010. [p1]

J. T. Cacioppo and R. E. Petty. Electromyograms as measures of extent and affectivity of information processing. *American Psychologist*, 36:441–456, 1981. [p7]

J. T. Cacioppo, R. E. Petty, M. E. Losch, and H. S. Kim. Electromyographic activity over facial muscle regions can differentiate the valence and intensity of affective reactions. *Journal of Personality and Social Psychology*, 50(2):260–268, 1986. [p7]

S.-M. Chow and J. R. Nesselroade. General slowing or decreased inhibition? Mathematical models of age differences in cognitive functioning. *Journals of Gerontology B*, 59(3):101–109, 2004. [p11]

S.-M. Chow and G. Zhang. Nonlinear regime-switching state-space (RSSS) models. *Psychometrika*, 78 (4):740–768, 2013. URL https://doi.org/10.1007/s11336-013-9330-8. [p1, 4, 5, 6]

S.-M. Chow, E. Ferrer, and J. R. Nesselroade. An unscented Kalman filter approach to the estimation of nonlinear dynamical systems models. *Multivariate Behavioral Research*, 42(2):283–321, 2007. URL https://doi.org/10.1080/00273170701360423. [p5, 11]

S.-M. Chow, M.-H. R. Ho, E. J. Hamaker, and C. V. Dolan. Equivalences and differences between structural equation and state-space modeling frameworks. *Structural Equation Modeling*, 17:303–332, 2010. URL https://doi.org/10.1080/10705511003661553. [p4]

S.-M. Chow, N. Tang, Y. Yuan, X. Song, and H. Zhu. Bayesian estimation of semiparametric nonlinear dynamic factor analysis models using the Dirichlet process prior. *British Journal of Mathematical and Statistical Psychology*, 64(1):69–106, 2011. URL https://doi.org/10.1348/000711010x497262. [p15]

S.-M. Chow, K. J. Grimm, F. Guillaume, C. V. Dolan, and J. J. McArdle. Regime-switching bivariate dual change score model. *Multivariate Behavioral Research*, 48(4):463–502, 2013. URL https://doi.org/10.1080/00273171.2013.787870. [p1]

S.-M. Chow, K. Witkiewitz, R. P. P. P. Grasman, and S. A. Maisto. The cusp catastrophe model as cross-sectional and longitudinal mixture structural equation models. *Psychological Methods*, 20: 142–164, 2015. URL https://doi.org/10.1037/a0038962. [p1]

S.-M. Chow, J. J. Bendezú, P. M. Cole, and N. Ram. A comparison of two-stage approaches for fitting nonlinear ordinary differential equation (ode) models with mixed effects. *Multivariate Behavioral Research*, 51(2–3):154–184, 2016. URL https://doi.org/10.1080/00273171.2015.1123138. [p15]

S.-M. Chow, L. Ou, A. Ciptadi, E. Prince, D. You, M. D. Hunter, J. M. Rehg, A. Rozga, and D. S. Messinger. Representing sudden shifts in intensive dyadic interaction data using differential equation models with regime switching. *Psychometrika*, 83(2):476–510, 2018. URL https://doi.org/10.1007/s11336-018-9605-1. [p4, 5, 6]

P. De Jong. The likelihood for a state space model. *Biometrika*, 75(1):165–169, 1988. URL https://doi.org/10.2307/2336450. [p4]

U. Dimberg, M. Thunberg, and K. Elmehed. Unconscious facial reactions to emotional facial expressions. *Psychological Science*, 11(1):86–89, 2000. URL https://doi.org/10.1111/1467-9280.00221. [p7]

C. V. Dolan. *MKFM6: Multi-Group, Multi-Subject Stationary Time Series Modeling Based on the Kalman Filter*, 2005. URL http://users/fmg.uva.nl/cdolan/. [p1]

C. V. Dolan. Structural equation mixture modeling. In R. E. Millsap and A. Maydeu-Olivares, editors, *The SAGE Handbook of Quantitative Methods in Psychology*, pages 568–592. Sage, Thousand Oaks, CA, 2009. [p1]

C. V. Dolan, B. R. Jansen, and H. L. J. Van der Maas. Constrained and unconstrained multivariate normal finite mixture modeling of piagetian data. *Multivariate Behavioral Research*, 39(1):69–98, 2004. URL https://doi.org/10.1207/s15327906mbr3901_3. [p1]

C. Driver, M. Voelkle, and H. Oud. *Ctsem: Continuous Time Structural Equation Modelling*, 2017a. URL https://CRAN.R-project.org/package=ctsem. R package version 2.5.0. [p2]

C. C. Driver, J. H. L. Oud, and M. C. Voelkle. Continuous time structural equation modelling with R package ctsem. *Journal of Statistical Software*, 2017b. URL https://doi.org/10.18637/jss.v077.i05. [p2]

J. Durbin and S. J. Koopman. *Time Series Analysis by State Space Methods*. Oxford University Press, Oxford, United Kingdom, 2001. [p2, 3, 15]

D. Eddelbuettel and R. Francois. Rcpp: Seamless R and C++ integration. *Journal of Statistical Software*, 40(8):1–18, 2011. URL https://doi.org/10.18637/jss.v040.i08. [p15]

D. Eddelbuettel and R. Francois. *RcppGSL: 'Rcpp' Integration for 'GNU GSL' Vectors and Matrices*, 2018. URL https://CRAN.R-project.org/package=RcppGSL. R package version 0.3.6. [p15]

D. Eddelbuettel, R. Francois, J. Allaire, K. Ushey, Q. Kou, N. Russell, D. Bates, and J. Chambers. *Rcpp: Seamless R and C++ Integration*, 2018. URL https://CRAN.R-project.org/package=Rcpp. R package version 1.0.0. [p15]

R. J. Elliott, L. Aggoun, and J. B. Moore. *Hidden Markov Models: Estimation and Control*. Springer-Verlag, New York, 1995. [p1]

L. Fahrmeir and G. Tutz. *Multivariate Statistical Modelling Based on Generalized Linear Models*. Springer-Verlag, New York, NY, 1994. [p15]

K. Fukuda and K. Ishihara. Development of human sleep and wakefulness rhythm during the first six months of life: Discontinuous changes at the 7th and 12th week after birth. *Biological Rhythm Research*, 28:94–103, 1997. URL https://doi.org/10.1076/brhm.28.3.5.94.13132. [p1]

P. Gilbert. *Dse: Dynamic Systems Estimation (Time Series Package)*, 2015. URL https://CRAN.R-project.org/package=dse. R package version 2015.12-1. [p1]

P. D. Gilbert. *Brief User's Guide: Dynamic Systems Estimation*, 2006 or later. URL http://cran.r-project.org/web/packages/dse/vignettes/Guide.pdf. [p1]

J. M. Gottman. *The Mathematics of Marriage: Dynamic Nonlinear Models*. The MIT Press, Cambridge, MA, 2002. [p11]

M. S. Grewal and A. P. Andrews. *Kalman Filtering: Theory and Practice Using MATLAB*. John Wiley & Sons, Hoboken, NJ, 3rd edition, 2008. [p2]

GSL Project Contributors. *GSL - GNU Scientific Library - GNU Project - Free Software Foundation (FSF)*, 2010. URL http://www.gnu.org/software/gsl/. [p14]

J. D. Hamilton. A new approach to the economic analysis of nonstationary time series and the business cycle. *Econometrica*, 57:357–384, 1989. URL https://doi.org/10.2307/1912559. [p1, 5]

J. D. Hamilton. *Time Series Analysis*. Princeton University Press, Princeton, NJ, 1994. [p4]

A. C. Harvey. *Forecasting, Structural Time Series Models and the Kalman Filter*. Cambridge University Press, Cambridge, United Kingdom, 1989. [p1, 4]

J. Helske. KFAS: Exponential family state space models in R. *Journal of Statistical Software*, 2017a. URL https://doi.org/10.18637/jss.v078.i10. [p1, 2, 14]

J. Helske. *KFAS: Kalman Filter and Smoother for Exponential Family State Space Models*, 2017b. URL https://CRAN.R-project.org/package=KFAS. R package version 1.2.9. [p1]

J. Hofbauer and K. Sigmund. *The Theory of Evolution and Dynamical Systems: Mathematical Aspects of Selection (London Mathematical Society Student Texts)*. Cambridge University Press, 1988. ISBN 0521358388. URL http://www.worldcat.org/isbn/0521358388. [p11]

B. Hosenfeld. Indicators of discontinuous change in the development of analogical reasoning. *Journal of Experimental Child Psychology*, 64:367–395, 1997. URL https://doi.org/10.1006/jecp.1996.2351. [p1]

M. D. Hunter. State space modeling in an open source, modular, structural equation modeling environment. *Structural Equation Modeling: A Multidisciplinary Journal*, pages 1–18, 2017. URL https://doi.org/10.1080/10705511.2017.1369354. [p1]

R. J. Hyndman. CRAN task view: Time series analysis. Online, 2016. URL https://CRAN.R-project.org/view=TimeSeries. Accessed on October 09, 2016. [p1]

S. G. Johnson. *The NLopt Nonlinear-Optimization Package*, 2008. URL http://ab-initio.mit.edu/nlopt. [p14]

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960. URL https://doi.org/10.1115/1.3662552. [p4, 6]

C.-J. Kim and C. R. Nelson. *State-Space Models with Regime Switching: Classical and Gibbs-Sampling Approaches with Applications*. MIT Press, Cambridge, MA, 1999. [p1, 4, 5, 6, 15]

A. A. King, D. Nguyen, and E. L. Ionides. Statistical inference for partially observed Markov processes via the R package pomp. *Journal of Statistical Software*, 69(12):1–43, 2016. URL https://doi.org/10.18637/jss.v069.i12. [p2]

A. A. King, E. L. Ionides, and C. Breto. *Pomp: Statistical Inference for Partially Observed Markov Processes*, 2018. URL https://CRAN.R-project.org/package=pomp. R package version 1.18. [p2]

L. Kohlberg and R. Kramer. Continuities and discontinuities in childhood and adult moral development. *Human development*, 12(2):93–120, 1969. URL https://doi.org/10.1159/000270857. [p1]

S. J. Koopman, N. Shephard, and J. A. Doornik. Statistical algorithms for models in state space using SsfPack 2.2. *Econometrics Journal*, 2(1):113–166, 1999. URL https://doi.org/10.1111/1368-423x.00023. [p1, 14]

D. Kraft. A software package for sequential quadratic programming. Technical Report 88-28, DFVLR-FB, Oberpfaffenhofen, Germany, 1988. [p2, 14]

D. Kraft. Algorithm 733: TOMP — Fortran Modules for Optimal Control Calculations. *ACM Transactions on Mathematical Software*, 20(3):262–281, 1994. URL https://doi.org/10.1145/192115.192124. [p2, 14]

G. Y. Kulikov and M. V. Kulikova. Accurate numerical implementation of the continuous-discrete extended kalman filter. *IEEE Transactions on Automatic Control*, 59(1), 2014. URL https://doi.org/10.1109/tac.2013.2272136. [p4, 5]

M. V. Kulikova and G. Y. Kulikov. Adaptive ODE Solvers in Extended Kalman Filtering Algorithms. *Journal of Computational and Applied Mathematics*, 262:205–216, 2014. URL https://doi.org/10.1016/j.cam.2013.09.064. [p4, 5]

A. J. Lotka. *Elements of Physical Biology*. Williams & Wilkins, Baltimore, MD, 1925. [p7, 11]

Z.-H. Lu, S.-M. Chow, A. Sherwood, and H. Zhu. Bayesian analysis of ambulatory cardiovascular dynamics with application to irregularly spaced sparse data. *Annals of Applied Statistics*, 9:1601–1620, 2015. URL https://doi.org/10.1214/15-aoas846. [p15]

B. O. Muthén and T. Asparouhov. LTA in Mplus: Transition probabilities influenced by covariates. Mplus Web Notes: No. 13., 2011. URL http://www.statmodel.com/examples/{LTA}webnote.pdf. [p1]

M. C. Neale, M. D. Hunter, J. N. Pritikin, M. Zahery, T. R. Brick, R. M. Kirkpatrick, R. Estabrook, T. C. Bates, H. H. Maes, and S. M. Boker. OpenMx 2.0: Extended structural equation and statistical modeling. *Psychometrika*, 80(2):535–549, 2016. URL https://doi.org/10.1007/s11336-014-9435-8. [p1]

L. Ou, M. D. Hunter, and S.-M. Chow. *Dynr: Dynamic Modeling in R*, 2018. R package version 0.1.13-4. [p1]

G. Petris. An R package for dynamic linear models. *Journal of Statistical Software*, 36(12):1–16, 2010. URL https://doi.org/10.18637/jss.v036.i12. [p1]

G. Petris. *Dlm: Bayesian and Likelihood Analysis of Dynamic Linear Models*, 2014. URL https://CRAN.R-project.org/package=dlm. R package version 1.1-4. [p1]

G. Petris and S. Petrone. State space models in R. *Journal of Statistical Software*, 41(4):1–25, 2011. URL https://doi.org/10.18637/jss.v041.i04. [p1]

J. Piaget and B. Inhelder. *The Psychology of the Child*. Basic Books, New York, NY, 1969. [p1]

W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, 2002. [p5]

J. O. Ramsay, G. Hooker, and S. Graves. *Functional Data Analysis with R and MATLAB*. Springer-Verlag, New York, NY, 2009. [p15]

J. L. Rodgers and D. C. Rowe. Social contagion and adolescent sexual behavior: A developmental EMOSA model. *Psychological Review*, 100(3):479–510, 1993. URL https://doi.org/10.1037/0033-295x.100.3.479. [p11]

J. L. Rodgers, D. C. Rowe, and M. Buster. Social contagion, adolescent sexual behavior, and pregnancy: a nonlinear dynamic EMOSA model. *Developmental Psychology*, 34(5):1096–1113, 1998. URL https://doi.org/10.1037/0012-1649.34.5.1096. [p11]

J. A. Sanchez-Espigares and A. Lopez-Moreno. *MSwM: Fitting Markov Switching Models*, 2014. URL https://CRAN.R-project.org/package=MSwM. R package version 1.2. [p2]

G. E. Schwartz. Biofeedback, self-regulation, and the patterning of physiological processes. *American Scientist*, 63:314–324, 1975. [p7]

G. Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978. [p4, 15]

A. Stone, S. Shiffman, A. Atienza, and L. Nebeling. *The Science of Real-Time Data Capture: Self-Reports in Health Research*. Oxford University Press, NY, 2008. [p1]

O. Taramasco and S. Bauer. *Hidden Markov Models Simulations and Estimations*, 2012. R package version 2.0.2. [p2]

R. W. Thatcher. A predator-prey model of human cerebral development. In K. M. Newell and P. C. M. Molenaar, editors, *Applications of Nonlinear Dynamics to Developmental Process Modeling*, pages 87–128. Lawrence Erlbaum, Mahwah, NJ, 1998. [p11]

The MathWorks, Inc. *MATLAB Version 9.1 (R2016b)*. The MathWorks, Inc., Natick, MA, 2016. [p2]

E. A. Thomas and J. A. Martin. Analyses of parent-infant interaction. *Psychological Review*, 83(2): 141–156, 1976. URL https://doi.org/10.1037/0033-295x.83.2.141. [p11]

G. C. Tiao and R. S. Tsay. Some advances in non-linear and adaptive modelling in time series. *Journal of Forecasting*, 13:109–131, 1994. URL https://doi.org/10.1002/for.3980130206. [p1]

H. Tong and K. S. Lim. Threshold autoregression, limit cycles and cyclical data. *Journal of the Royal Statistical Society B*, 42:245–292, 1980. URL https://doi.org/10.1142/9789812836281_0002. [p1]

S. van Buuren and K. Groothuis-Oudshoorn. mice: Multivariate imputation by chained equations in R. *Journal of Statistical Software*, 45(3):1–67, 2011. URL https://doi.org/10.18637/jss.v045.i03. [p15]

S. van Buuren and K. Groothuis-Oudshoorn. *Mice: Multivariate Imputation by Chained Equations*, 2017. URL https://CRAN.R-project.org/package=mice. R package version 2.46.0. [p15]

H. L. J. van der Maas and P. C. M. Molenaar. Stagewise cognitive development: An application of catastrophe theory. *Psychological Review*, 99(3):395–417, 1992. [p1]

H. L. J. van der Maas, R. Kolstein, and J. van der Pligt. Sudden transitions in attitudes. *Sociological Methods & Research*, 32(125–152), 2003. URL https://doi.org/10.1177/0049124103253773. [p11]

M. van Dijk and P. van Geert. Wobbles, humps and sudden jumps: A case study of continuity, discontinuity and variability in early language development. *Infant and Child Development*, 16(1): 7–33, 2007. URL https://doi.org/10.1002/icd.506. [p1]

I. Visser. Depmix: An R-package for fitting mixture models on mixed multivariate data with Markov dependencies. Technical report, University of Amsterdam, 2007. URL http://cran.r-project.org. [p1]

I. Visser and M. Speekenbrink. depmixS4: An R package for hidden Markov models. *Journal of Statistical Software*, 36(7):1–21, 2010. URL https://doi.org/10.18637/jss.v036.i07. [p2]

I. Visser and M. Speekenbrink. *depmixS4: Dependent Mixture Models - Hidden Markov Models of GLMs and Other Distributions in S4*, 2016. URL https://CRAN.R-project.org/package=depmixS4. R package version 1.3-3. [p2]

V. Volterra. Fluctuations in the abundance of a species considered mathematically. *Nature*, 118:558–560, 1926. URL https://doi.org/10.1038/118558a0. [p7, 11]

M. Yang and S.-M. Chow. Using state-space model with regime switching to represent the dynamics of facial electromyography (EMG) data. *Psychometrika: Application and Case Studies*, 74(4):744–771, 2010. URL https://doi.org/10.1007/s11336-010-9176-2. [p1, 4, 5, 7, 9, 11]

Funding for this study was provided by NSF grant SES-1357666, NIH grants R01MH61388, R01HD07699, R01GM105004, Pennsylvania State Quantitative Social Sciences Initiative, and UL TR000127 from the National Center for Advancing Translational Sciences.

Lu Ou
ACTNext
ACT, Inc.
500 Act Drive
Iowa City, IA 52244
E-mail: lu.ou@act.org

Michael D. Hunter
Georgia Institute of Technology
J.S. Coon Bldg, Room 225
648 Cherry St NW
Atlanta, GA 30313
E-mail: mhunter43@gatech.edu

Sy-Miin Chow
Department of Human Development and Family Studies
The Pennsylvania State University
420 Biobehavioral Health Building
University Park, PA 16802
E-mail: emailquc16@psu.edu