1 Overview

In this work, we describe the basic usage of the hierarchical Bayesian formulation of the ctsem (Driver, Oud & Voelkle, 2017) software for continuous-time dynamic modelling in R (R Core Team, 2014). This formulation, described in detail in Driver and Voelkle (in press), offers advantages over what could be considered more typical dynamic modelling approaches, such as vector autoregressive models or latent change score models. The two main advantages relate to the handling of time, and the treatment of individual differences. Time information is explicitly incorporated into the model, such that predictions from one measurement to another relate exactly to the amount of time that has passed, rather than simply the number of measurements, as is typical in discrete-time models. When the time interval between all measurements are equal, there is an exact relationship between the discrete and continuous time form, but when they differ, the continuous-time form is typically more appropriate. For more on continuous-time models, see Oud and Jansen (2000), Singer (1993), Voelkle and Oud (2013). With respect to individual differences, the hierarchical Bayesian approach allows for individual variation across all model parameters, while still making full use of the data from all subjects. This has the result that individual specific parameter estimates may be obtained with far fewer time points than would be required by single-subject time-series type modelling approaches. For more on hierarchical Bayesian models, see Gelman, Carlin, Stern and Rubin (2014).

This document is structured such that we first briefly describe the continuous time dynamic model governing within subject dynamics, and the hierarchical model governing the distribution of subject level parameters. These aspects are covered in detail in Driver and Voelkle (in press). Following, we walk through installing the ctsem software, setting up a data structure, specifying and fitting the model, followed by summary and plotting functions. Some details on additional complexity are then provided, including an example model with a more complex dynamic structure, a discussion of the various options for incorporating stationarity assumptions into the model, and a walk-through of the various transformations involved in the model.

1.1 Subject Level Latent Dynamic model

This section describes the subject level model characterising the system dynamics and measurement properties. Although we do not describe it explicitly, the corresponding discrete time autoregressive / moving average models can be specified and use the same set of parameter matrices we describe.

1.2 Subject level latent dynamic model

The subject level dynamics are described by the following stochastic differential equation:

$$\mathrm{d}\boldsymbol{\eta}(t) = \Big(\mathbf{A}\boldsymbol{\eta}(t) + \mathbf{b} + \mathbf{M}\boldsymbol{\chi}(t)\Big)\mathrm{d}t + \mathbf{G}\mathrm{d}\mathbf{W}(t) \tag{1}$$

Vector $\boldsymbol{\eta}(t) \in \mathbb{R}^v$ represents the state of the latent processes at time t. The matrix $\mathbf{A} \in \mathbb{R}^{v \times v}$ (DRIFT) represents the drift matrix, with auto effects on the diagonal and cross effects on the off-diagonals characterizing the temporal dynamics of the processes.

The continuous time intercept vector $\mathbf{b} \in \mathbb{R}^v$ (CINT), in combination with $\mathbf{A}$, determines the long-term level at which the processes fluctuate around.

Time dependent predictors $\boldsymbol{\chi}(t)$ represent inputs to the system that vary over time and are independent of fluctuations in the system. Equation 1 shows a generalized form for time dependent predictors, that could be treated a variety of ways dependent on the assumed time course (or shape) of time dependent predictors. We use a simple impulse form shown in Equation 2, in which the predictors are treated as impacting the processes only at the instant of an observation occasion u. When necessary, the evolution over time can be modeled by extending the state matrices, for examples and discussion see Driver and Voelkle (2017).

$$\boldsymbol{\chi}(t) = \sum_{u \in \mathbf{U}} \mathbf{x}_u \delta(t - t_u) \tag{2}$$

Here, time dependent predictors $\mathbf{x}_u \in \mathbb{R}^l$ (tdpreds) are observed at measurement occasions $u \in \mathbf{U}$. The Dirac delta function $\delta(t - t_u)$ is a generalized function that is ∞ at 0 and 0 elsewhere, yet has an integral of 1 (when 0 is in the range of integration). It is useful to model an impulse to a system, and here is scaled by the vector of time dependent predictors $\mathbf{x}_u$. The effect of these impulses on processes $\boldsymbol{\eta}(t)$ is then $\mathbf{M} \in \mathbb{R}^{v \times l}$ (TDPREDEFFECT).

$\mathbf{W}(t) \in \mathbb{R}^v$ (DIFFUSION) represents independent Wiener processes, with a Wiener process being a random-walk in continuous time. $\mathrm{d}\mathbf{W}(t)$ is meaningful in the context of stochastic differential equations, and represents the stochastic error term, an infinitesimally small increment of the Wiener process. Lower triangular matrix $\mathbf{G} \in \mathbb{R}^{v \times v}$ represents the effect of this noise on the change in $\boldsymbol{\eta}(t)$. $\mathbf{Q}$, where $\mathbf{Q} = \mathbf{G}\mathbf{G}^\top$, represents the variance-covariance matrix of the diffusion process in continuous time.

1.3 Subject level measurement model

The latent process vector $\boldsymbol{\eta}(t)$ has measurement model:

$$\mathbf{y}(t) = \boldsymbol{\Lambda}\boldsymbol{\eta}(t) + \boldsymbol{\tau} + \boldsymbol{\epsilon}(t) \quad \text{where } \boldsymbol{\epsilon}(t) \sim \mathrm{N}(\mathbf{0}_c, \boldsymbol{\Theta}) \tag{3}$$

$\mathbf{y}(t) \in \mathbb{R}^c$ is the vector of manifest variables, $\boldsymbol{\Lambda} \in \mathbb{R}^{c \times v}$ (LAMBDA) represents the factor loadings, and $\boldsymbol{\tau} \in \mathbb{R}^c$ (MANIFESTMEANS) the manifest intercepts. The residual vector $\boldsymbol{\epsilon} \in \mathbb{R}^c$ has covariance matrix $\boldsymbol{\Theta} \in \mathbb{R}^{c \times c}$ (MANIFESTVAR).

1.4 Overview of hierarchical model

Parameters for each subject are first drawn from a simultaneously estimated higher level distribution over an unconstrained space, then a set of parameter specific transformations are applied so that a) each parameter conforms to necessary bounds and b) is subject to the desired prior. Following this, in some cases matrix transformations are applied to generate the continuous time matrices described. The higher level distribution has a multivariate normal prior. We provide a brief description here, and an R code example later in this work, but for the full details, one should again see Driver and Voelkle (in press).

The joint-posterior distribution of the model parameters given the data is as follows:

$$p(\boldsymbol{\Phi}, \boldsymbol{\mu}, \mathbf{R}, \boldsymbol{\beta} | \mathbf{Y}, \mathbf{z}) \propto p(\mathbf{Y}|\boldsymbol{\Phi}) p(\boldsymbol{\Phi}|\boldsymbol{\mu}, \mathbf{R}, \boldsymbol{\beta}, \mathbf{z}) p(\boldsymbol{\mu}, \mathbf{R}, \boldsymbol{\beta}) \tag{4}$$

Subject specific parameters $\boldsymbol{\Phi}_i$ are determined in the following manner:

$$\boldsymbol{\Phi}_i = \text{tform}\Big(\boldsymbol{\mu} + \mathbf{R}\mathbf{h}_i + \boldsymbol{\beta}\mathbf{z}_i\Big) \tag{5}$$

$$\mathbf{h}_i \sim \text{N}(\mathbf{0}, \mathbf{1}) \tag{6}$$

$$\boldsymbol{\mu} \sim \text{N}(\mathbf{0}, \mathbf{1}) \tag{7}$$

$$\boldsymbol{\beta} \sim \text{N}(\mathbf{0}, \mathbf{1}) \tag{8}$$

$\boldsymbol{\Phi}_i \in \mathbb{R}^s$ is the s length vector of parameters for the dynamic and measurement models of subject i. $\boldsymbol{\mu} \in \mathbb{R}^s$ parameterizes the means of the raw population distributions of subject level parameters. $\mathbf{R} \in \mathbb{R}^{s \times s}$ is the matrix square root of the raw population distribution covariance matrix, parameterizing the effect of subject specific deviations $\mathbf{h}_i \in \mathbb{R}^s$ on $\boldsymbol{\Phi}_i$. $\boldsymbol{\beta} \in \mathbb{R}^{s \times w}$ is the raw effect of time independent predictors $\mathbf{z}_i \in \mathbb{R}^w$ on $\boldsymbol{\Phi}_i$, where w is the number of time independent predictors. $\mathbf{Y}_i$ contains all the data for subject i used in the subject level model – $\mathbf{y}$ (process related measurements) and $\mathbf{x}$ (time dependent predictors). $\mathbf{z}_i$ contains time independent predictors data for subject i. tform is an operator that applies a transform to each value of the vector it is applied to. The specific transform depends on which subject level parameter matrix the value belongs to, and the position in that matrix.

At a number of points, we will refer to the parameters prior to the tform function as 'raw' parameters. So for instance 'raw population standard deviation' would refer to a diagonal entry of $\mathbf{R}$, and 'raw individual parameters for subject i' would refer to $\boldsymbol{\mu} + \mathbf{R}\mathbf{h}_i + \boldsymbol{\beta}\mathbf{z}_i$. In contrast, without the 'raw' prefix, 'population means' would refer to tform($\boldsymbol{\mu}$), and would typically reflect values the user is more likely to be interested in, such as the continuous time drift parameters.

1.5 Install software and prepare data

Install ctsem software within R:

```
install.packages("ctsem")
library("ctsem")
```

Prepare data in long format, each row containing one time point of data for one subject. We need a subject id column, named by default "id", though this can be changed in the model specification. Some of the outputs are simpler to interpret if subject id is a sequence of integers from 1 to the number of subjects, but this is not a requirement. We also need a time column "time", containing positive numeric values for time, columns for manifest variables (the names of which must be given in the next step using `ctModel`), columns for time dependent predictors (these vary over time but have no model estimated and are assumed to impact latent processes instantly), and columns for time independent predictors (which predict the subject level parameters, that are themselves time invariant – thus the values for a particular time independent predictor must be the same across all observations of a particular subject).

```
      id   time     Y1     Y2 TD1    TI1    TI2    TI3
 [1,]  1  1.877 0.2975 -4.728   0 -0.701  1.257 -0.361
 [2,]  1  3.608 6.5180 -2.518   0 -0.701  1.257 -0.361
 [3,]  1  4.765 3.2376  0.755   0 -0.701  1.257 -0.361
 [4,]  1 15.385 0.0471 -7.386   0 -0.701  1.257 -0.361
```

```
 [5,]  1 16.052 0.6934 -8.971   0 -0.701  1.257 -0.361
 [6,]  1 16.934 1.3235 -9.059   0 -0.701  1.257 -0.361
 [7,]  1 18.096 1.7718 -9.941   0 -0.701  1.257 -0.361
 [8,]  2  0.000 0.4283  4.971   0     NA -0.833  0.183
 [9,]  2  0.925 2.6017  6.381   0     NA -0.833  0.183
```

Priors for the model are set up to be 'weakly informative' (in that extreme parameter values are deemed less likely) for typical applications in the social sciences, on data that is centered and scaled. Because of this, we recommend grand mean centering and scaling each variable in the data, with the exception of time dependent predictors, which should be scaled as normal, but centered such that a value of zero implies no effect. Similarly, we expect a time interval of 1.00 to reflect some 'moderate change' in the underlying process. If we wished to model daily hormonal fluctuations, with a number of measurements each day, a time scale of hours, days, or weeks could be sensible – minutes or years would likely be problematic. If the data are not adjusted according to these considerations, the priors themselves should be adjusted, or at least their impact carefully considered.

```
ctstantestdat[,c('Y1','Y2','TI1','TI2','TI3')] <-
 scale(ctstantestdat[,c('Y1','Y2','TI1','TI2','TI3')])
```

Functions to convert between wide and long formats used by ctsem are available, these are `ctWideToLong`, `ctDeintervalise`, `ctLongToWide`, `ctIntervalise`. For details see the relevant help in R.

1.6 Missing values

Missingness in the manifest variables is handled using the typical filtering / full information maximum likelihood approach, missing values on time dependent predictors are replaced with zeros, and missing values on time independent predictors are sampled as part of the model, with a default prior of normal(0,10) (this is specified via a slot in the model object).

1.7 Model specification

Specify model using `ctModel(type="stanct",...)`. `"stanct"` specifies a continuous time model in Stan format, `"standt"` specifies discrete time, while `"omx"` is the classic ctsem behaviour and prepares an OpenMx model. Other arguments to `ctModel` proceed as normal, although some matrices used for type `"omx"` are not relevant for the Stan formats, either because the between subject matrices are handled differently, or because time dependent and independent predictors are now treated as fixed regressors and only require effect (or design) matrices. These differences are documented in the help for `ctModel`, available in R via `?ctModel`.

```
model<-ctModel(type='stanct',
  n.latent=2, latentNames=c('eta1','eta2'),
  n.manifest=2, manifestNames=c('Y1','Y2'),
  n.TDpred=1, TDpredNames='TD1',
  n.TIpred=3, TIpredNames=c('TI1','TI2','TI3'),
  LAMBDA=diag(2))
```

Table 1: ctModel arguments

Argument	Sign	Default	Meaning
n.manifest	c		Number of manifest indicators per individual at each measurement occasion.
n.latent	v		Number of latent processes.
LAMBDA	Λ		n.manifest $\times$ n.latent loading matrix relating latent to manifest variables.
manifestNames		Y1, Y2, etc	n.manifest length character vector of manifest names.
latentNames		eta1, eta2, etc	n.latent length character vector of latent names.
T0VAR	Q_1^*	free	lower tri n.latent $\times$ n.latent matrix of latent process initial covariance, specified with standard deviations on diagonal and covariance related parameters on lower triangle.
T0MEANS	η_1	free	n.latent $\times$ 1 matrix of latent process means at first time point, T0.
MANIFESTMEANS	τ	free	n.manifest $\times$ 1 matrix of manifest means.
MANIFESTVAR	Θ	free	diagonal matrix of var / cov between manifests, specified with standard deviations on diagonal and zeroes elsewhere.
DRIFT	A	free	n.latent $\times$ n.latent matrix of continuous auto and cross effects.
CINT	b	0	n.latent $\times$ 1 matrix of continuous intercepts.
DIFFUSION	Q	free	lower triangular n.latent $\times$ n.latent matrix containing standard deviations of latent process on diagonal, and covariance related parameters on lower off-diagonals.
n.TDpred	l	0	Number of time dependent predictors in the dataset.
TDpredNames		TD1, TD2, etc	n.TDpred length character vector of time dependent predictor names.
TDPREDEFFECT	M	free	n.latent $\times$ n.TDpred matrix of effects from time dependent predictors to latent processes.
n.TIpred	p	0	Number of time independent predictors.
TIpredNames		TI1, TI2, etc	n.TIpred length character vector of time independent predictor names.

This specifies a first order bivariate latent process model, with each process measured by a single, potentially noisy, manifest variable. A single time dependent predictor is included in the model, and three time independent predictors. Additional complexity or restrictions may be added, the Table 1 shows the basic arguments one may consider and their link to the dynamic model parameters. Note that for the Stan implementation, ctModel requires variance covariance matrices (DIFFUSION, T0VAR, MANIFESTVAR) to be specified with standard deviations on the diagonal, covariance related parameters on the lower off diagonal, and zeroes on the upper off diagonal. While it is possible to fix the lower off diagonals to non-zero values, in general this is difficult to interpret because of the necessary matrix transformations, thus we recommend either to fix to zero, or leave free.

These matrices may all be specified using a combination of character strings to name free parameters, or numeric values to represent fixed parameters.

The `pars` subobject of the created model object (in this case, `model$pars`) shows the parameter specification that will go into Stan, including both fixed and free parameters, whether the parameters vary across individuals, how the parameter is transformed from a standard normal distribution (thus setting both priors and bounds), and whether that parameter is regressed on the time independent predictors.

```
head(model$pars,7)
```

```
   matrix row col           param value transform multiplier meanscale offset indvarying sdscale
1 TOMEANS   1   1     T0mean_eta1    NA         0          1        10      0       TRUE       1
2 TOMEANS   2   1     T0mean_eta2    NA         0          1        10      0       TRUE       1
3  LAMBDA   1   1            <NA>     1        NA         NA        NA     NA      FALSE       1
4  LAMBDA   1   2            <NA>     0        NA         NA        NA     NA      FALSE       1
5  LAMBDA   2   1            <NA>     0        NA         NA        NA     NA      FALSE       1
6  LAMBDA   2   2            <NA>     1        NA         NA        NA     NA      FALSE       1
7   DRIFT   1   1 drift_eta1_eta1    NA         1         -2         2      0       TRUE       1
  TI1_effect TI2_effect TI3_effect
1       TRUE       TRUE       TRUE
2       TRUE       TRUE       TRUE
3      FALSE      FALSE      FALSE
4      FALSE      FALSE      FALSE
5      FALSE      FALSE      FALSE
6      FALSE      FALSE      FALSE
7       TRUE       TRUE       TRUE
```

By default, all free model parameters are set to individually varying, except for the T0VAR parameters used to initialise the latent processes. One may modify the output model to either restrict between subject differences (set some parameters to not vary over individuals), alter the transformation used to set the prior / bounds, or restrict which effects of time independent predictors to estimate. Plotting the original prior, making a change to the transform, and plotting the resulting prior, are shown here – in this case we will adjust the prior for the auto effect of our first latent process, captured by row 1 and column 1 of the DRIFT matrix, to also allow positive values, implying an explosive process wherein a change in one direction promotes further change in that direction. To achieve this, we change from built in transform 2, denoting an exponential, to 0, denoting no transformation. built-in transforms can be viewed at any point via inputting `ctsem:::tformshapes`. It is beneficial to use these in general to eliminate compilation time, but one can also specify a character string containing custom transformations of the 'param' – for example `'log(1+exp(param))'` could be used for a positive distribution, though this is also built-in transform number 1. In addition to the transformation field, multiplier, meanscale, and offset can all be used to adjust the transform. In this case we also set a negative offset because we still believe negative values are more likely for the drift auto-effect.One consideration when altering priors in this manner is that the starting values for sampling are taken from around the middle of the distribution normal.

```
par(mfrow=c(1,2))
plot(model,rows=7,rawpopsd=1)
model$pars$transform[7]<- 0
model$pars$meanscale[7] <- 2
model$pars$multiplier[7] <- 1
model$pars$offset[7] <- -1
plot(model, rows=7,rawpopsd=1)
```

Figure 1 shows the prior distribution for the population mean of DRIFT[1,1] in black, as well as two possible priors for the subject level parameters, conditional on our specified raw population standard deviation of 1. The blue prior results from assuming the population mean is one standard deviation lower than the mean of its prior, and the red one standard deviation higher.

In addition to adjusting the prior for the population mean, the prior for the extent of individual variation around that mean can also be adjusted. Amongst other circumstances, this prior may

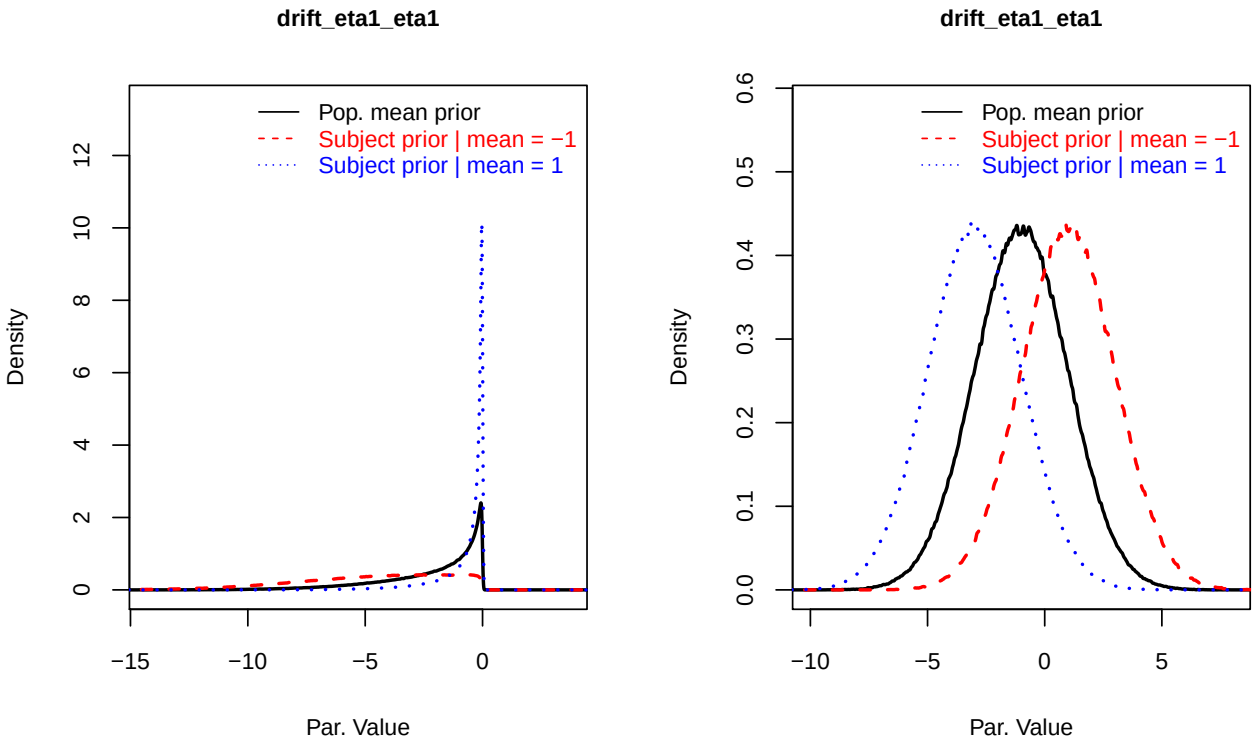

Figure 1: Prior distribution density plots.

need to be reduced when limited time points are available, to ensure adequate regularisation. Here we change the scaling factor of the individual variation for all parameters, from 1.0 to 0.1, and demonstrate the effect of this using the previously adjusted auto effect. In this case, we do not fix the `popsd` when plotting, which now gives the distribution of individual variation over all possible values for the population (pop) sd parameter – the marginal distribution of Figure 2 has a very different shape to the conditional distribution of Figure 1, but the effect of the change in `sdscale` parameter could be seen in both cases.

```
par(mfrow=c(1,2))
plot(model,rows=7)
model$pars$sdscale<- .1
plot(model, rows=7)
```

It can be helpful to completely eliminate individual variation in some parameters, particularly since unnecessary between subject effects will slow sampling and hinder appropriate regularization, but be aware of the many parameter dependencies in these models – restricting one parameter may lead to genuine variation in the restricted parameter expressing itself elsewhere. Here we only allow for individual variation in the DRIFT and MANIFESTMEANS parameters.

```
model$pars$indvarying[!(model$pars$matrix %in% c('DRIFT','MANIFESTMEANS'))] <- FALSE
```

Also similarly restrict which parameters to include time independent predictor effects for. In this case, the only adverse effects of such restrictions are that the relationship between the predictor and variables will not be estimated, but the subject level parameters themselves should not be

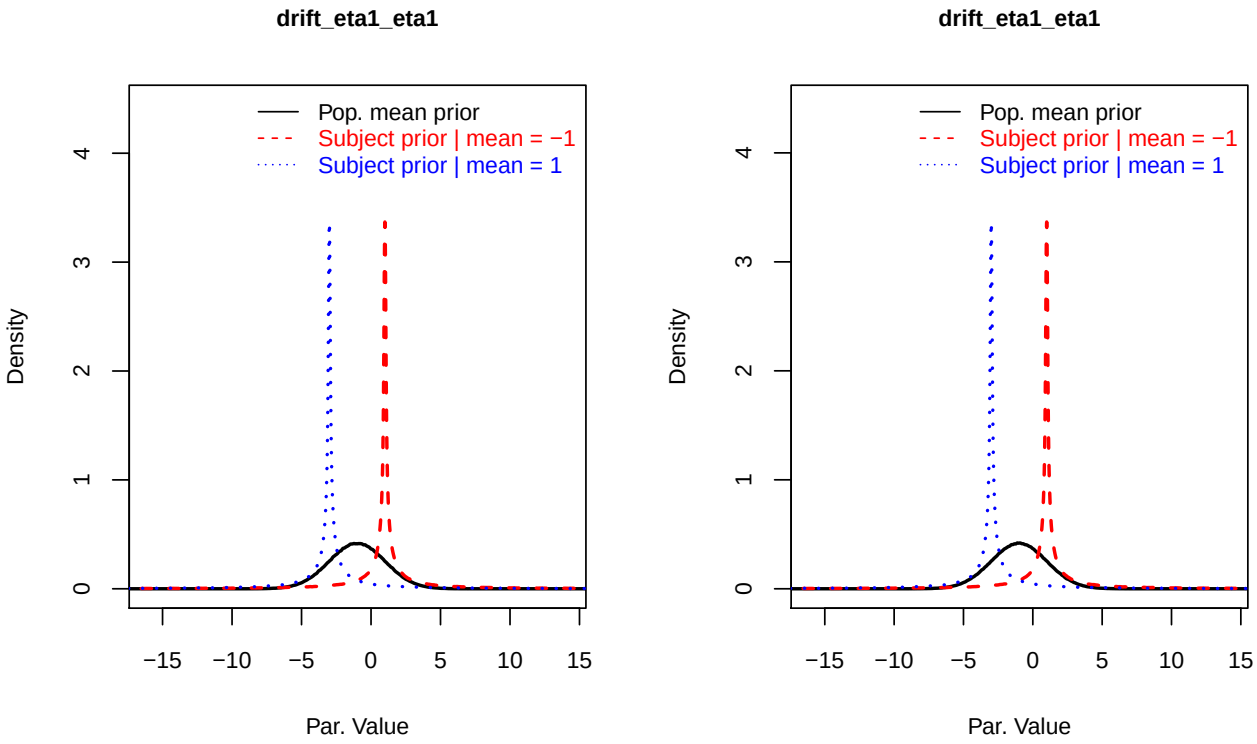

Figure 2: Prior distribution density plots of auto-effects, with default (left) and adjusted (right) scale parameter for population standard deviation.

very different, as they are still freely estimated. Note that such effects can only be estimated for parameters specified as individually varying – in case one particularly wished to model a covariate effect without allowing for residual variation, this could be approximated by setting the parameter to `indvarying`, but putting a very small `sdscale` value for the parameter. Here, we first restrict the tipredeffects on all parameters, and free them only for the drift parameters.

```
model$pars[,c('TI1_effect','TI2_effect','TI3_effect')] <- FALSE
model$pars[model$pars$matrix == 'DRIFT',
  c('TI1_effect','TI2_effect','TI3_effect')] <- TRUE
```

1.8 Model fitting

Once model specification is complete, the model is fit to the data using the `ctStanFit` function as shown in the following example. Depending on the data, model, and number of iterations requested, this can take anywhere from a few minutes to days. Current experience suggests 300 iterations is often enough to get an idea of what is going on, but more may be necessary for robust inference, but this is highly dependent on the specifics. For the sake of speed for this example we only sample for 300 iterations, with a max treedepth of the Hamiltonian sampler reduced from the default of 10 to 6. With these settings the fit should take only a minute or two, but is unlikely adequate for inference!. Those that wish to try out the functions without waiting, can simply use the already existing `ctstantestfit` object with the relevant functions (the code for this is commented out). The dataset specified here is built-in to the ctsem package, and available whenever ctsem is loaded in R.

```
fit<-ctStanFit(datalong = ctstantestdat, ctstanmodel = model, iter=200,
  control=list(max_treedepth=6), chains=2, plot=FALSE)
# fit <- ctstantestfit
```

The plot argument allows for plotting of sampling chains in real time, which is useful for slow models to ensure that sampling is proceeding in a functional manner. Models with many parameters (e.g., many subjects and all parameters varying over subject) may be too taxing for the plotting function to handle smoothly - we have had success with up to around 4000 parameters.

1.9 Summary

After fitting, the summary function may be used on the fit object, which returns details regarding the population mean parameters, population standard deviation parameters, population correlations, and the effect parameters of time independent predictors. Additionally, summary outputs a range of matrices regarding correlations between subject level parameters. `rawpopcorr_means` reports the posterior mean of the correlation between raw (not yet transformed from the standard normal scale) parameters. `rawpopcorr_sd` reports the standard deviation of these parameters.

```
summary(fit,timeinterval = 1)
```

In the summary output, the free population mean parameters under `$popmeans` are likely one of the main points of interest. They are returned in the same form that they are input to `ctModel` - that is, covariance matrix related parameters are in the form of either standard deviations or a transformed correlation parameter. Because the latter is difficult to interpret, various parameter matrices are also returned in the `$parmatrices` section of the summary. The discrete time matrices reported here (prefixed by `dt`) are by default from a time interval of 1, but this can be changed. Asymptotic matrices – those for a time interval of infinity – are also output in some cases, and prefixed by `asym`. Covariance related matrices are reported in covariance form, except where the suffix `cor` is added to indicate correlations.

The function `ctStanContinuousPars` can be used to return the continuous time parameter matrices in actual matrix form, for specific subjects, or groups of subjects. By default the median is calculated, but the function that aggregates over samples can be changed as desired. In the following code, the 97.5% quantile is returned, for subject 3.

```
ctStanContinuousPars(fit,subjects = 3, calcfunc = quantile, calcfuncargs = list(probs=.975))
```

Note that subject 3 refers to the third subject in the data, not any particular identifier specified in the subject id column. The mapping of subject id to the internal sequential integer representation can be found via `fit$setup$idmap` . If a vector of subjects is specified, or the string `'all'` is used, multiple subjects are aggregated over.

1.10 Plotting

The plot function outputs a sequence of plots, all generated by specific functions. The name of the specific function appears in a message in the R console, checking the help for each specific function and running them separately will allow more customization of plots. Some of the plots, such as

the trace, density, and interval, are generated by the relevant rstan function and are hopefully self explanatory. The plots specific to the hierarchical continuous time dynamic model are as follows:

```
ctStanDiscretePars(fit, plot=TRUE, indices = 'CR', subjects = 'all')
```

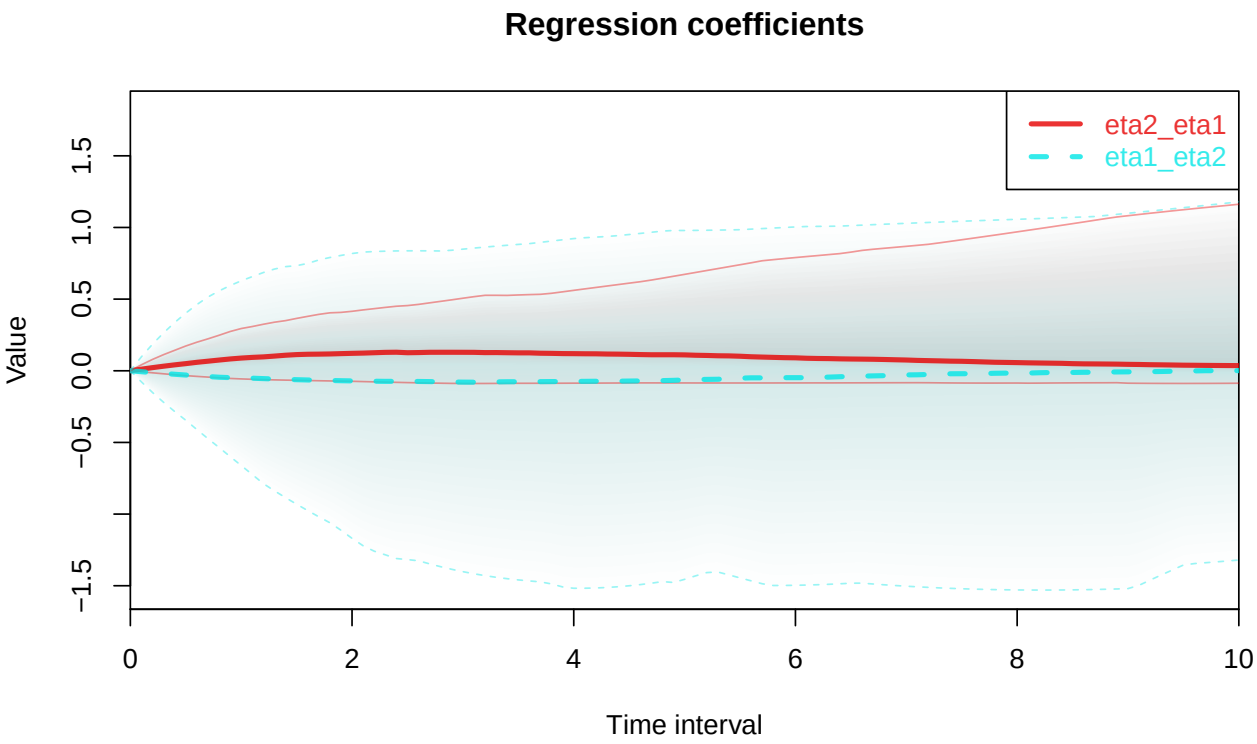

Figure 3: Discrete-time cross-effect dynamics of the estimated system for a range of time intervals, with 95% credible intervals.

Figure 3 shows the dynamic regression coefficients (between latent states at different time points) that are implied by the model for particular time intervals, as well as the uncertainty (default is 95% credible interval) of these coefficients. In this case the estimates are of the cross regression effects, obtained by sampling from all subjects data, but specific subjects, as well as specific indices of the effects (e.g., indices = 'AR' or indices = rbind(c(2,1))' can be specified.

The relation between posteriors and priors for variables of interest can also be plotted as follows – note that the `rows` argument of the shown code is not necessary, but if only particular parameter plots are desired the rows corresponding to specific parameters of `fit$setup$popsetup` can be specified.

```
ctStanPlotPost(obj = fit, rows=3)
```

Shown in Figure 4 are approximate density plots based on the post-warmup samples drawn. For each parameter that has individual variation specified, three plots are shown. These are the population mean posterior compared to the prior, the posterior versus prior distribution of subject level parameters along with the population mean prior, and then the population standard deviation posterior compared to the prior.

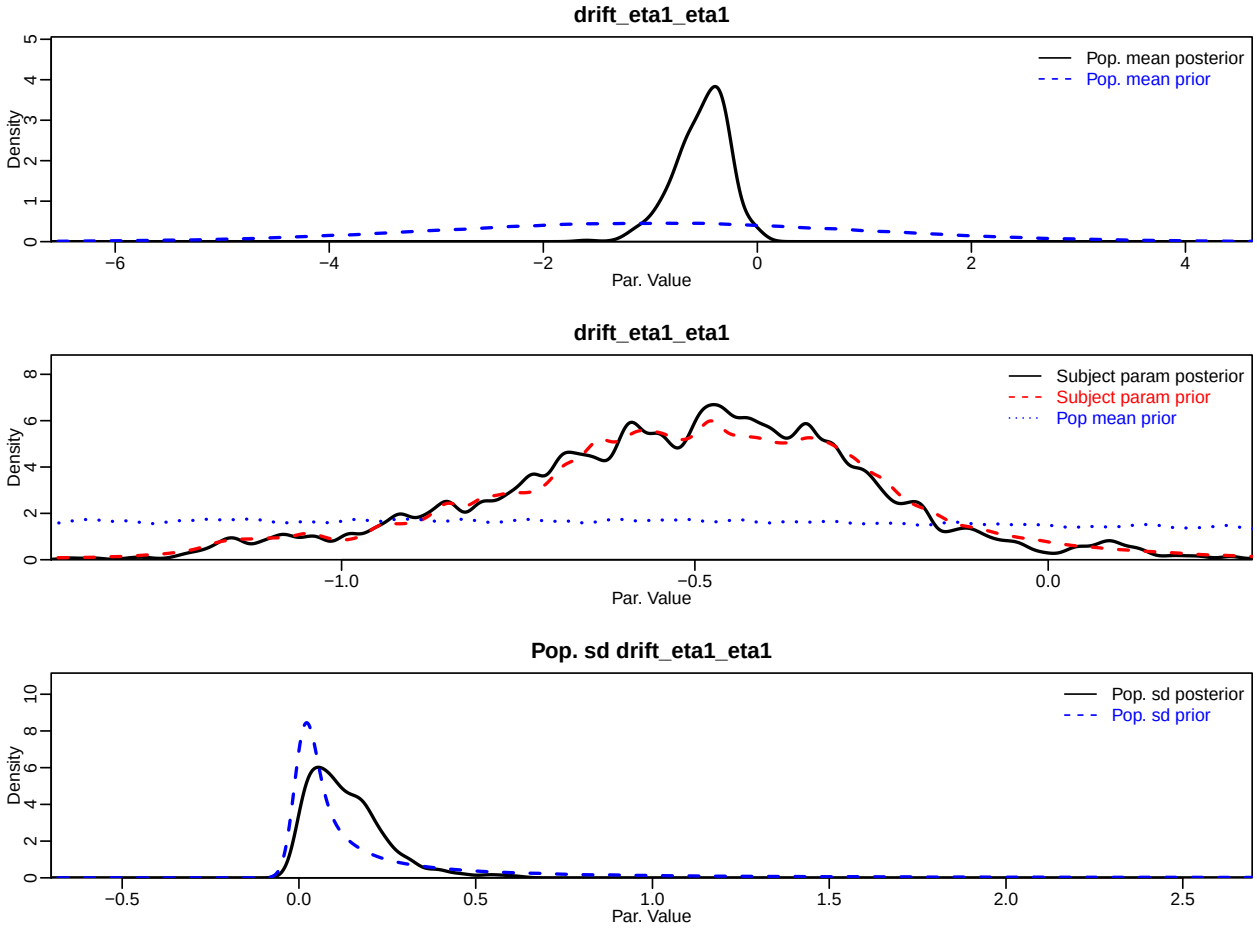

Figure 4: Prior and posterior densities relevant to the second process auto effect.

1.11 Model prediction plots

One means of assessing model performance is to view plots of the observed time series alongside the model predicted time series. ctsem includes functionality to output prior (based on all prior observations), updated (based on all prior and current observations), and smoothed (based on all observations) expectations and covariances from the Kalman filter, based on specific subjects models. For ease of comparison, expected manifest indicator scores conditional on prior, updated and smoothed states are also included. This approach allows for: predictions regarding individuals states at any point in time, given any values on the time dependent predictors (external inputs such as interventions or events); residual analysis to check for unmodeled dependencies in the data; or simply as a means of visualization, for comprehension and model sanity checking purposes. Examples of such are depicted in Figure 5, where we see observed and smoothed scores for a selected subject from our sample. If we wanted to predict unobserved states in the future, we would need only to specify the appropriate timerange (Prediction into earlier times is possible but makes little sense unless the model is restricted to stationarity). For help with these plots, see `?ctKalman` and `?ctKalmanPlot` (arguments for the latter are passed via `ctKalman`, as below).

```
par(mfrow=c(1,2))

ctKalman(fit, subjects=2, timerange=c(0,30), kalmanvec=c('y', 'yprior'), timestep=.01,
plot=TRUE, plotcontrol=list(xaxs='i', main = 'Predicted'))

ctKalman(fit, subjects=2, timerange=c(0,30), kalmanvec=c('y', 'ysmooth'), timestep=.01,
plot=TRUE, plotcontrol=list(xaxs='i',main = 'Smoothed'))
```

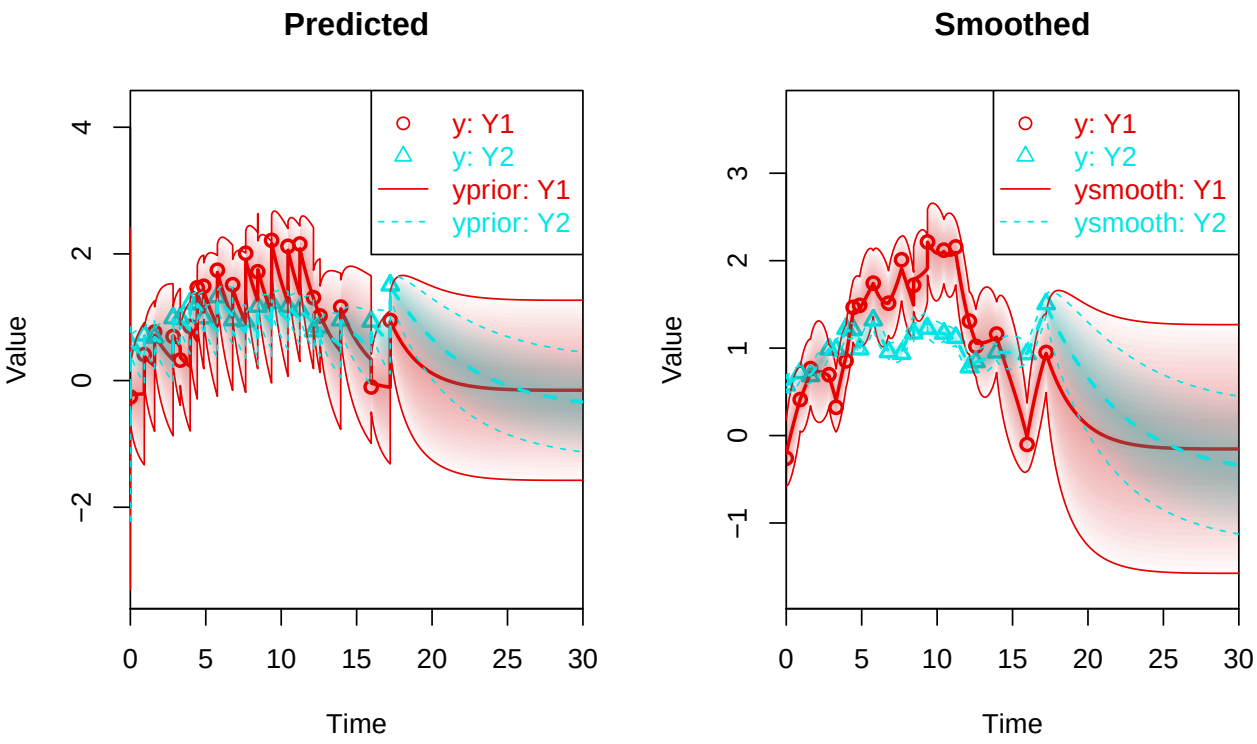

Figure 5: Predicted and smoothed estimates for one subject with two processes. Uncertainty shown is a 95% credible interval comprising both process and measurement error.

1.12 Time independent predictor effect plots

Because time independent predictors give a linear effect *prior* to any necessary transformations, the effects necessarily become non-linear when applied to bounded parameters, which can make them difficult to conceptualise. To aid with this, a visual summary of the full range of effects can be seen using the `ctStanTIpredeffects` function, as follows:

```
ctStanTIpredeffects(fit, plot = TRUE, whichpars=c('dtDRIFT','MANIFESTVAR[2,2]'),
 timeinterval = .5, whichTIpreds = 3, includeMeanUncertainty = FALSE, nsubjects=10,
  nsamples = 50)
```

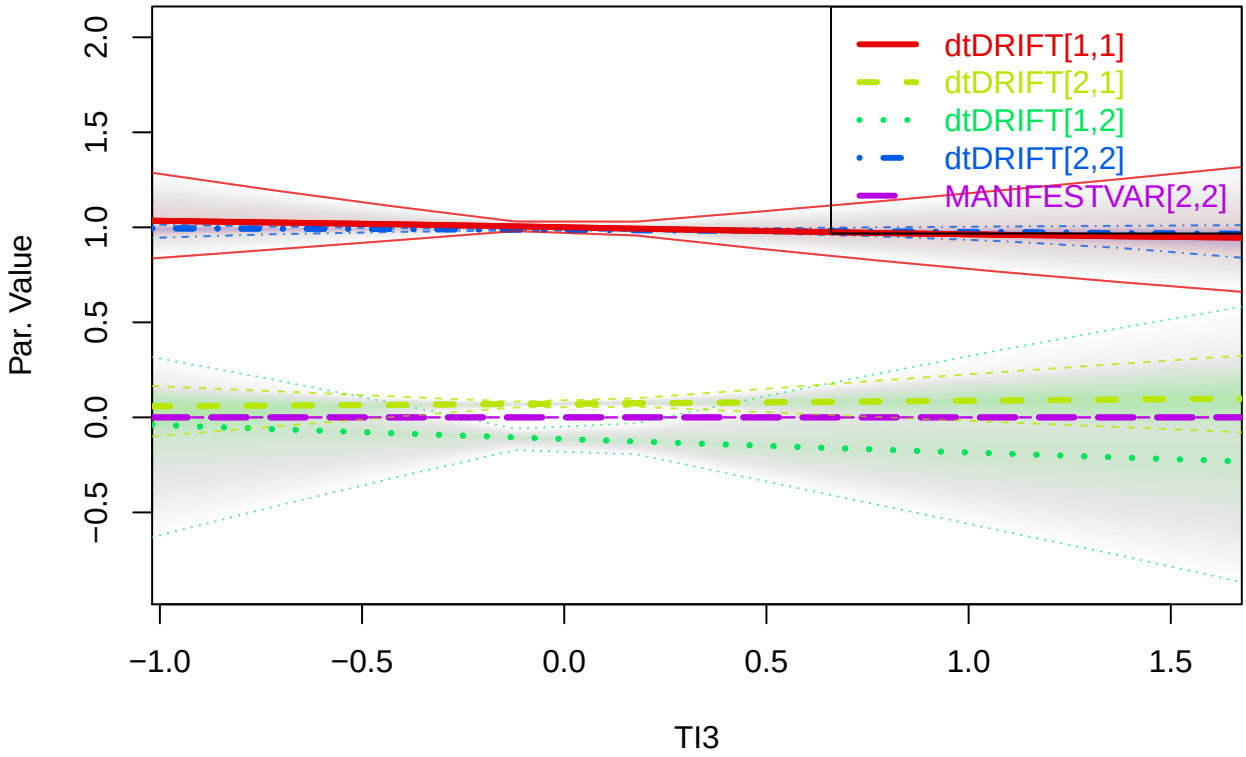

Figure 6: Expectations for individuals parameter values change depending on their score on time independent predictors.

Figure 6 shows how the expectation for an individuals parameter value is likely to change depending on the value they have for the time independent predictor specified. In this example, the discrete time drift effects for a time interval of 0.2, as well as the manifest error variance from row 2 column 2 of the MANIFESTVAR matrix, are shown. Of course, the plot for the error variance parameter is simply a flat line, since we did not allow it to vary across subjects. Multiple predictors can be specified, and the combined effect based on observed predictor combinations will be shown, but this will likely make no sense unless there is a deterministic relation between the two – this can be useful for including linear and quadratic effects, for instance, wherein the first predictor is linear, and the second quadratic. The nsamples and nsubjects parameters specify how many different parameter samples, and predictor values, are used – higher values take longer to compute, but give smoother / more accurate plots.

2 Additional details

2.1 Stationarity

When it is reasonable to assume that the prior for long term expectation and variance of the latent states is the same as (or very similar to) the prior for initial expectations and variances, setting some form of stationarity in advance may be beneficial. Three approaches to this are possible. The first approach is to give free parameters of the T0VAR or T0MEANS matrix the name `'stationary'`, which can be useful if for instance only the initial variance should be stationary, or just one of the initial variances. Another approach is to set the argument stationary=TRUE to ctStanFit. Specifying this argument then ignores any T0VAR and T0MEANS matrices in the input, instead replacing them with asymptotic expectations based on the DRIFT, DIFFUSION, and CINT matrices. Alternatively, a prior can be placed on the stationarity of the dynamic models, calculated as the difference between the T0MEANS and the long run asymptotes of the expected value of the process, as well as the difference between the diagonals of the T0VAR covariance matrix and the long run asymptotes of the covariance of the processes. Such a prior encourages a minimisation of these differences, and can help to ensure that sensible, non-explosive models are estimated, and also help the sampler get past difficult regions of relative flatness in the parameter space due to colinearities between the within and between subject parameters. However if such a prior is too strong it can also induce difficult dependencies in model parameters, and there are a range of models where one may not wish to have such a prior. To place such a prior, the model$stationarymeanprior and model$stationaryvarprior slots can be changed from the default of NA to a numeric vector, representing the normal standard deviation of the deviations from stationarity. The number of elements in the vector correspond to the number of latent processes.

2.2 Accessing Stan model code

For diagnosing problems or modifying the model in ways not achievable via the ctsem model specification, one can use ctsem to generate the Stan code and then work directly with that, simply by specifying the argument `fit=FALSE` to the `ctStanFit` function, and accessing the `$stanmodeltext` subobject. Any altered code can be passed back into `ctStanFit` by using the `stanmodeltext` argument, which can be convenient for setting up the data in particular.

2.3 Using Rstan functions

The standard rstan output functions such as summary and extract are also available, and the shinystan package provides an excellent browser based interface. The stan fit object is stored under the `$stanfit` subobject of the `ctStanFit` output. The parameters which are likely to be of most interest in the output are prefixed by `pop_` for pop (population) mean, and `popsd` for pop standard deviation. Any `pop` parameters are returned in the form of the continuous time matrix equations. Subject specific parameters are denoted by the matrix they are from, then the first index represents the subject id, followed by standard matrix notation. For example, the 2nd row and 1st column of the DRIFT matrix for subject 8 is `DRIFT[8,2,1]`. Parameters in such matrices are returned in the form used for internal calculations – that is, variance covariance matrices are returned as such, rather than the lower-triangular standard deviation and correlation matrices required for input.

2.4 Oscillating, single subject example - sunspots data

In the following example we fit the sunspots data available within R, which has previously been fit by various authors including Tómasson (2013). We have used the same CARMA(2,1) model and obtained similar estimates – some differences are due to the contrast between Bayes and maximum likelihood, though if desired one could adjust the code to fit using maximum likelihood, as here we have only one subject.

```
#get data
 sunspots<-sunspot.year
 sunspots<-sunspots[50: (length(sunspots) - (1988-1924))]
 id <- 1
 time <- 1749:1924
datalong <- cbind(id, time, sunspots)

#setup model
 ssmodel <- ctModel(type='stanct', n.latent=2, n.manifest=1,
  manifestNames='sunspots',
  latentNames=c('ss_level', 'ss_velocity'),
   LAMBDA=matrix(c( 1, 'ma1' ), nrow=1, ncol=2),
   DRIFT=matrix(c(0, 'a21', 1, 'a22'), nrow=2, ncol=2),
   MANIFESTMEANS=matrix(c('m1'), nrow=1, ncol=1),
   CINT=matrix(c(0, 0), nrow=2, ncol=1),
   TOVAR=matrix(c(1,0,0,1), nrow=2, ncol=2), #Because single subject
   DIFFUSION=matrix(c(0, 0, 0, "diffusion"), ncol=2, nrow=2))

 ssmodel$pars$indvarying<-FALSE #Because single subject
 ssmodel$pars$offset[14]<- 44 #Because not mean centered
 ssmodel$pars[4,c('transform','offset')]<- c(1,0) #To avoid multi modality

#fit
ssfit <- ctStanFit(datalong, ssmodel, iter=300, chains=2)

#output
summary(ssfit)$popmeans
```

2.5 Population standard deviations - understanding the transforms

This section is intended as a helper to those trying to work through the various transformations found in the model. Internally, we sample parameters that we refer to as the 'raw' parameters – these parameters have no bounds and are typically drawn from normal distributions. Both raw population mean and subject specific deviation parameters are drawn from normal(0, 1) distributions. Depending on the specific parameter, various transformations may be applied to set appropriate bounds and priors. The raw population standard deviation for these raw parameters is sampled (by default) from a normal(0, 1) distribution called `rawpopsdbase`, which is by default transformed via an exponential function – this ensures the parameters are positive and the prior for the standard deviation is a lognormal distribution. This distribution can be altered via the model subobjects `rawpopsdbase`, `rawpopsdbaselowerbound`, and `rawpopsdtransform`. This distribution can also be scaled on a per parameter basis by the `sdscale` multiplier in the model specification, which defaults to 1. The following script shows a didactic sequence of sampling and transformation for a model with a single parameter, the auto effect of the drift matrix, and 3 subjects. Although we sample the priors themselves here, this is merely to reflect the prior and enable understanding and plotting.

Note also that because we are only displaying the procedure for a single parameter here, we simplify things somewhat by avoiding calculations to determine the square root of the population covariance matrix – with only one individually varying parameter, it is simply the standard deviation.

```
#set plotting parameters
par(mfrow=c(2,2), lwd=3, yaxs='i', mgp=c(1.8,.5,0),
  mar=c(3,3,3,1)+.1)
bw=.03

n <- 999999 #number of samples to draw to from prior for plotting purposes
nsubjects <- 4 #number of subjects

#parameter specific transform
tform <- function(x) -log(exp(-1.5 * x) + 1) #default drift auto effect transform

#raw pop sd transform
sdscale <- 1 #default
rawsdtform <- function(x) exp(x * 2 -2) * sdscale #default

#sd approximation function
sdapprox <- function(means,sds,tform) {
  for(i in 1:length(means)){
    sds[i] <- ((tform(means[i]+sds[i]*3) - tform(means[i]-sds[i]*3))/6 +
      (tform(means[i]+sds[i]) - tform(means[i]-sds[i]))/2) /2
  }
  return(sds)
}

#raw population mean parameters
rawpopmeans_prior <- rnorm(n, 0, 1) #prior distribution for rawpopmeans
rawpopmeans_sample <- -.3 #hypothetical sample
sdscale <- 1 #default

#population mean parameters after parameter specific transform
popmeans_prior <- tform(rawpopmeans_prior)
popmeans_sample <- tform(rawpopmeans_sample)

#plot pop means
plot(density(rawpopmeans_prior), ylim=c(0,1), xlim=c(-5,2),
  xlab='Parameter value', main='Population means')
points(density(popmeans_prior, bw=bw),col=2,type='l')
segments(y0=0,y1=.5,x0=c(rawpopmeans_sample,popmeans_sample),lty=3,col=1:2)
legend('topleft',c('Raw pop. mean prior', 'Pop. mean prior',
  'Raw pop. mean sample', 'Pop. mean sample'),lty=c(1,1,3,3), col=1:2, bty='n')

#population standard deviation parameters
rawpopsd_prior <- rawsdtform(rnorm(n, 0, 1)) #raw population sd prior

popsd_prior <- sdapprox(rawpopmeans_prior,rawpopsd_prior,tform)

#sample population standard deviation posterior
rawpopsd_sample <- rawsdtform(.9) #hypothetical sample
popsd_sample <- sdapprox(means=rawpopmeans_sample,  #transform sample to actual pop sd
  sds=rawpopsd_sample,tform=tform)
```

```
#plot pop sd
plot(density(rawpopsd_prior,from=-.2,to=10,na.rm=TRUE, bw=bw), xlab='Parameter value',
  xlim=c(-.1,3), ylim=c(0,2), main='Population sd')
points(density(popsd_prior,from=-.2,to=10,na.rm=TRUE, bw=bw),type='l', col=2)
segments(y0=0,y1=1,x0=c(rawpopsd_sample, popsd_sample), col=1:2,lty=3)
legend('topright',c('Raw pop. sd prior','Pop. sd prior',
  'Raw pop. sd sample','Pop. sd sample'), col=1:2, lty=c(1,1,3,3),bty='n')

#individual level parameters

#marginal individual level parameters (given all possible values for mean and sd)
rawindparams_margprior <- rawpopmeans_prior + rawpopsd_prior * rnorm(n, 0, 1)
indparams_margprior <- tform(rawindparams_margprior)

plot(density(rawindparams_margprior,from=-10,to=10,bw=bw), xlab='Parameter value',
  xlim=c(-5,2), ylim=c(0,1), main='Marginal dist. individual parameters')
points(density(indparams_margprior,from=-10,to=.2,bw=bw),type='l',col=2)
legend('topleft',c('Raw individual parameters prior','Individual parameters prior'),
  col=1:2,lty=1,bty='n')


#conditional individual level parameters (given sampled values for mean and sd)
rawindparams_condprior<- rawpopmeans_sample + rawpopsd_sample * rnorm(n,0,1)
rawindparams_condsample<- rawpopmeans_sample + rawpopsd_sample * rnorm(nsubjects,0,1)
indparams_condprior<- tform(rawindparams_condprior)
indparams_condsample<- tform(rawindparams_condsample)


plot(density(rawindparams_condprior), xlab='Parameter value', xlim=c(-5,2),
  ylim=c(0,1), main='Conditional dist. individual parameters')
points(density(indparams_condprior),type='l',col=2)
segments(y0=0,y1=.5,x0=c(rawindparams_condsample, indparams_condsample),
  col=rep(1:2,each=nsubjects),lty=3, lwd=2)
legend('topleft',c('Raw ind. pars. prior','Ind. pars. prior',
  'Raw ind. pars. samples','Ind. pars. samples'), col=1:2, lty=c(1,1,3,3),bty='n')
```

In the top left of Figure 7, we can see the prior distribution of population means for, in this case, a diagonal (auto effect) of the drift matrix. The prior for the raw population distribution is a standard normal, while for the actual population distribution it is definitely not normal. We draw a hypothetical sample from the raw distribution, and show the resulting transformed value. To the right, the prior distribution of the raw population standard deviation is shown. This raw distribution is the same for all parameter types, but the resulting prior distribution of population standand deviations is also dependent on the parameter specific transform (although in this particular case the raw and actual population sd priors are almost the same). This dependency is most easily understood if one considers the case where the parameter specific transformation simply multiplied the raw parameter by 2 – if we sampled a raw population sd of 1.5, the actual population sd sample would be 3.0. With nonlinear transformations, the dependency is not so easily calculated, and we use a sigma point approximation (Julier & Uhlmann, 1997), as shown in the code, when it is necessary to plot or summarise the population sd. The lower left plot shows the prior distribution for individual level parameters, marginalising over the priors for population means and standard deviations. This plot is very similar to the means plot directly above it, just somewhat more spread

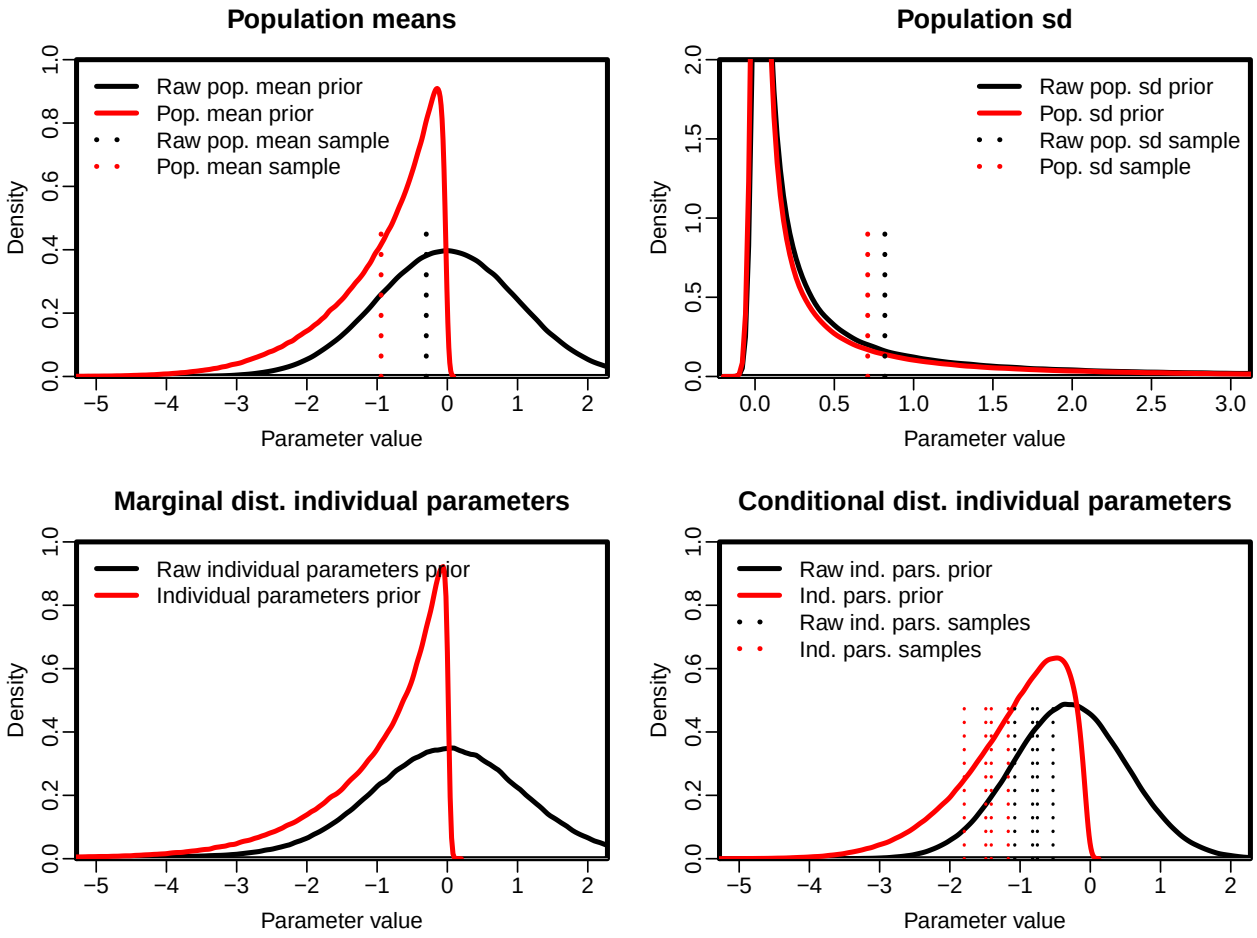

Figure 7: Depiction of the prior distributions and sampling process through which individual specific parameters are determined.

out due to the additional variation included. Things get more interesting when we look at the lower right plot – here, we see the prior distributions for individual level parameters, conditional on the values sampled in the top row of plots. Along with the prior distribution, in this lower right plot we also draw samples for 4 subjects, showing both the raw individual parameter, and the individual parameter after the necessary transforms.

3 Conclusion

With this work, we have described the basics of the hierarchical continuous time dynamic model, and provided detailed discussion on the usage of the R package ctsem (Driver et al., 2017) for fitting such models to data. While the model is necessarily somewhat complex, we believe it offers many interesting possibilities for understanding the dynamics of personality over time, and hope that the overview of the software provided here encourages new and interesting applications of the model.

References

Driver, C. C., Oud, J. H. L. & Voelkle, M. C. (2017). Continuous time structural equation modeling with r package ctsem. *Journal of Statistical Software*, *77*(5). doi:10.18637/jss.v077.i05

Driver, C. C. & Voelkle, M. C. (in press). Hierarchical Bayesian continuous time dynamic modeling. *Psychological Methods*.

Driver, C. C. & Voelkle, M. C. (2017). Understanding the time course of interventions with continuous time dynamic models. *Manuscript submitted for publication.*

Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. (2014). *Bayesian data analysis*. Chapman & Hall/CRC Boca Raton, FL, USA. Retrieved from http://amstat.tandfonline.com/doi/full/10.1080/01621459.2014.963405

Julier, S. J. & Uhlmann, J. K. (1997). New extension of the Kalman filter to nonlinear systems. (Vol. 3068, pp. 182–194). Signal Processing, Sensor Fusion, and Target Recognition VI. International Society for Optics and Photonics. doi:10.1117/12.280797

Oud, J. H. L. & Jansen, R. A. R. G. (2000). Continuous time state space modeling of panel data by means of SEM. *Psychometrika*, *65*(2), 199–215. doi:10.1007/BF02294374

R Core Team. (2014). *R: A language and environment for statistical computing*. Vienna, Austria: R Foundation for Statistical Computing. Retrieved from http://www.R-project.org/

Singer, H. (1993). Continuous-time dynamical systems with sampled data, errors of measurement and unobserved components. *Journal of Time Series Analysis*, *14*(5), 527–545. 00046. doi:10.1111/j.1467-9892.1993.tb00162.x

Tómasson, H. (2013). Some computational aspects of Gaussian CARMA modelling. *Statistics and Computing*, *25*(2), 375–387. doi:10.1007/s11222-013-9438-9

Voelkle, M. C. & Oud, J. H. L. (2013). Continuous time modelling with individually varying time intervals for oscillating and non-oscillating processes. *British Journal of Mathematical and Statistical Psychology*, *66*(1), 103–126. doi:10.1111/j.2044-8317.2012.02043.x