Introduction to Hierarchical Continuous Time Dynamic Modelling With ctsem

Charles C. Driver
Max Planck Institute for Human Development

Manuel C. Voelkle
Humboldt University Berlin
Max Planck Institute for Human Development

Abstract

ctsem allows for easy specification and fitting of a range of continuous and discrete time dynamic models. The models may including multiple indicators (dynamic factor analysis), multiple, interrelated, potentially higher order processes, and time dependent (varying within subject) and time independent (not varying within subject) covariates. Classic longitudinal models like latent growth curves and latent change score models are also possible. Version 1 of ctsem provided structural equation model based functionality by linking to the OpenMx software, allowing mixed effects models (random means but fixed regression and variance parameters) for multiple subjects. For version 2 of the R package **ctsem**, we include a Bayesian specification and fitting routine that uses the **Stan** probabilistic programming language, via the **rstan** package in R. This allows for all parameters of the dynamic model to individually vary, using an estimated population mean and variance, and any time independent covariate effects, as a prior. ctsem version 1 is documented in a forthcoming JSS publication (Driver, Voelkle, Oud, in press), and in R vignette form at https://cran.r-project.org/web/packages/ctsem/vignettes/ctsem.pdf, here we provide the basics for getting started with the new Bayesian approach included in version 2.

Keywords: hierarchical time series, Bayesian, longitudinal, panel data, state space, structural equation, continuous time, stochastic differential equation, dynamic models, Kalman filter, R.

1. Overview

1.1. Subject Level Latent Dynamic model

This section describes the fundamental subject level model, and where appropriate, the name of the ctModel argument used to specify specific matrices. The description of the full model, including subject level likelihood and population model, is provided at the end of this document. 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, although the meaning is of course somewhat different.

1.2. Subject level latent dynamic model

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

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

Vector $\boldsymbol{\eta}(t) \in \mathbb{R}^{\upsilon}$ represents the state of the latent processes at time t. The matrix $\mathbf{A} \in \mathbb{R}^{\upsilon \times \upsilon}$ (DRIFT) represents the so-called 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 $\mathbf{b} \in \mathbb{R}^{\upsilon}$ (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 4 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 5, 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 an example see Driver, Oud, and Voelkle (In Press).

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

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 $\mathbf{M} \in \mathbb{R}^{\upsilon \times l}$ (TDPREDEFFECT).

$\mathbf{W}(t) \in \mathbb{R}^{\upsilon}$ (DIFFUSION) represents independent Wiener processes, with a Wiener process being a random-walk in continuous time. $d\mathbf{W}(t)$ 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 $\mathbf{G} \in \mathbb{R}^{\upsilon \times \upsilon}$ 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, $\boldsymbol{\Lambda} \in \mathbb{R}^{c \times \upsilon}$ (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, then a range of matrix transformations are applied to generate the continuous time matrices described, as well as all relevant discrete time instantiations (More variability in measurement time intervals thus means more computations). The higher level distribution has a multivariate normal prior. A more comprehensive description is found at the end of this document.

1.5. Install software and prepare data

Install ctsem software from github repository https://github.com/cdriveraus/ctsem .

```
require('devtools')
install_github("ctsem",username='cdriveraus')
```

Prepare data in long format, each row containing one time point of data for one subject. We need a subject id column containing numbers from 1 to total subjects, rising incrementally with each subject going down the data structure. This is to ensure coherence with the internal structure of the Stan model, the column is named by default "id", though this can be changed in the model specification. We also need a time column "time", containing 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 - generally these would be intervention or event dummy variables), 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 should be the same within a single subject).

```
     id  time    Y1     Y2     TD1    TI1    TI2    TI3
[1,]  1  0.02  3.60 -0.787 -0.5121  0.453  1.239 0.0184
[2,]  1  0.93  3.73 -0.644  2.4852  0.453  1.239 0.0184
[3,]  1  1.98  4.03 -1.768  1.0078  0.453  1.239 0.0184
[4,]  1  2.97  3.46 -0.800  0.2928  0.453  1.239 0.0184
[5,]  1  4.07  4.45 -1.078 -0.2090  0.453  1.239 0.0184
[6,]  2 36.03 -3.86 -1.117  0.0568 -0.668 -0.807 0.2716
[7,]  2 37.12 -3.01 -0.638  0.9502 -0.668 -0.807 0.2716
[8,]  2 37.93 -2.35 -2.119 -1.1268 -0.668 -0.807 0.2716
```

At present, missingness is fine on manifest indicators, but not allowed elsewhere.

1.6. 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 have been removed, 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.

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 × 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 × n.latent matrix of latent process initial covariance, specified with standard deviations on diagonal and correlations on lower triangle.
T0MEANS	η_1	free	n.latent × 1 matrix of latent process means at first time point, T0.
MANIFESTMEANS	τ	free	n.manifest × 1 matrix of manifest means.
MANIFESTVAR	Θ	free diag	lower triangular matrix of var / cov between manifests, specified with standard deviations on diagonal and correlations on lower triangle.
DRIFT	A	free	n.latent × n.latent matrix of continuous auto and cross effects.
CINT	b	0	n.latent × 1 matrix of continuous intercepts.
DIFFUSION	Q	free	lower triangular n.latent × n.latent matrix containing standard deviations of latent process on diagonal, and correlations 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 × 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.
TIPREDEFFECT	β	free	n.latent × n.TIpred effect matrix of time independent predictors on latent processes.

```
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))
```

This generates a first order bivariate latent process model, with each process measured by a single, potentially noisy, manifest variable. Additional complexity or restrictions may be added, the table below shows the basic arguments one may consider and their link to the dynamic model parameters. For more details see the ctsem help files or papers. Note that for the Stan implementation, ctModel requires variance covariance matrices (DIFFUSION, T0VAR, MANIFESTVAR) to be specified with standard deviations on the diagonal, correlations on the lower off diagonal, and zeroes on the upper off diagonal.

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

The parameters subobject of the created model object 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$parameters,8)
```

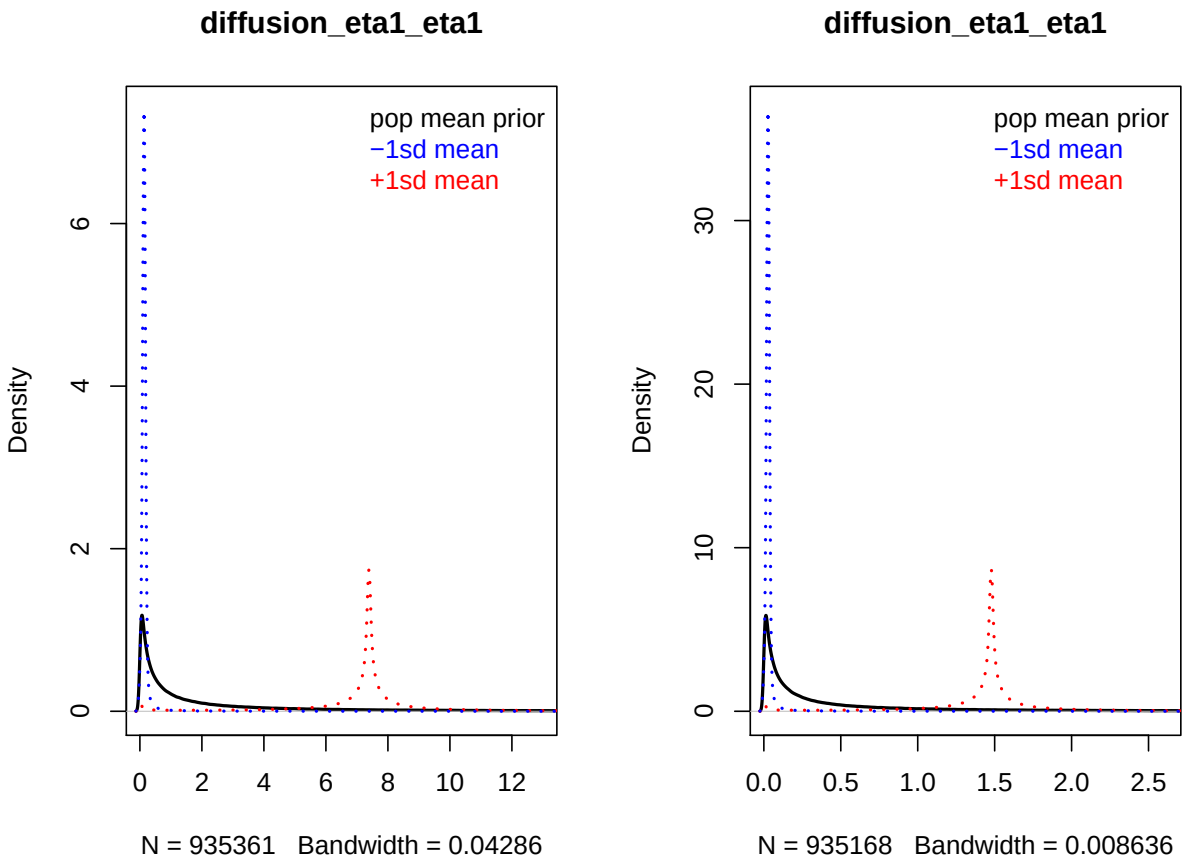
```
   matrix row col           param value                      transform indvarying sdscale
1 TOMEANS   1   1     TOmean_eta1    NA                    (param) * 10       TRUE       1
2 TOMEANS   2   1     TOmean_eta2    NA                    (param) * 10       TRUE       1
3  LAMBDA   1   1            <NA>     1                           <NA>      FALSE       1
4  LAMBDA   1   2            <NA>     0                           <NA>      FALSE       1
5  LAMBDA   2   1            <NA>     0                           <NA>      FALSE       1
6  LAMBDA   2   2            <NA>     1                           <NA>      FALSE       1
7   DRIFT   1   1 drift_eta1_eta1    NA -log(exp(-param*1.5)+1)-.00001       TRUE       1
8   DRIFT   1   2 drift_eta1_eta2    NA                      (param)*.5       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
8       TRUE       TRUE       TRUE
```

One may modify the output model to either restrict between subject differences (set some parameters to fixed over subjects), alter the transformation used to determine the prior / bounds, or restrict which effects of time independent predictors to estimate. Plotting the original prior, making a change, and plotting the resulting prior, are shown here – in this case we believe the latent process innovation for our first latent process, captured by row 1 and column 1 of the DIFFUSION matrix, to be small, so scale our prior accordingly to both speed and improve sampling. Rather than simply scaling by 0.2 as shown here, one could also construct a new form of prior, so long as the resulting distribution was within the bounds required for the specific parameter. Note that the resulting distribution is a result of applying the specified transformation to a standard normal distribution, with mean of 0 and standard deviation of 1. To change the underlying standard normal, one would need to edit the resulting Stan code directly.

```
par(mfrow=c(1,2))
ctStanPlotPriors(model,rows=11)
print(model$parameters$transform[11])
```

```
[1] "exp(param*2) +.00001"
```

```
model$parameters$transform[11]<- "(exp(param*2) +.0001)*.2"
ctStanPlotPriors(model,rows=11)
```



The plots show the prior distribution for the population mean of DIFFUSION[1,1] in black, as well as two possible priors for the subject level parameters. The blue prior results from assuming the population mean is one standard deviation lower than the mean of the prior, and marginalising over the prior for the standard deviation of the population distribution. This latter prior can be scaled using the sdscale column of the parameters subobject, but is by default a normal distribution with mean -3 and SD 2 on the log of the standard deviation, giving essentially a lightly regularised form of Jeffreys, or reference, prior.

Restrict between subject effects as desired. 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. The prior scale for between subject variance may need to be restricted when limited data (in either the number of time points or number of subjects) is available, to ensure adequate regularisation. Here we restrict MANIFESTVAR effects between subjects, and set all prior scales for the standard deviation of the population distribution to 0.2, from the default of 1.0. A rough interpretation of this change in sdscale is simply that we expect lower values for the population standard deviation, but to better interpret the effect of this latter change, see the section on standard deviation transformations.

```
model$parameters[c(15,18),]$indvarying<-FALSE
model$parameters$sdscale[1:28] <- .5
```

Also restrict which parameters to include time independent predictor effects for in a similar way, for similar reasons. In this case, the only adverse effects of restriction are that the

relationship between the predictor and variables will not be estimated, but the subject level parameters themselves should not be very different, as they are still freely estimated. Note that such effects are only estimated for individually varying parameters anyway – so after the above change there is no need to set the tipredeffect to FALSE for the T0VAR variables, it is assumed. Instead, we restrict the tipredeffects on all parameters, and free them only for the manifest intercept parameters.

```
model$parameters[,c('TI1_effect','TI2_effect','TI3_effect')]<-FALSE
model$parameters[c(19,20),c('TI1_effect','TI2_effect','TI3_effect')]<-TRUE
```

1.7. Model fitting

Once model specification is complete, the model is fit to the data using the ctStanFit function as follows – 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. This will of course vary massively depending on model and data. For the sake of speed for this example we generate start values using optimisation, and only sample for 100 iterations with a lowered max_treedepth - this latter parameter controls the maximum number of steps the Hamiltonian sampler is allowed to take per iteration, with each increase of 1 doubling the maximum. With these settings the fit should take only a few minutes (but will not be fit for inference!).

```
fit<-ctStanFit(long, model, iter=100, chains=2,
  initwithoptim=TRUE, plot=FALSE, max_treedepth = 6)
```

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 function to handle smoothly - we have had succcess with up to around 4000 parameters.

In this document we have walked through a multivariate process with cross effects and additional predictors, which will take some time to fit - to more quickly see the software in action, we can also fit the classic sunspot data and

1.8. Output

After fitting, the ctStanSummary function may be used, which runs the rstan summary function but returns only certain output parameters likely to be of interest. However, the standard rstan output functions such as summary and extract are also available, and the shinystan package provides an excellent browser based interface. The parameters which are likely to be of most interest in the output all begin with an "output" prefix, followed by either "hmean" for hyper (population) mean, or "hsd" for hyper standard deviation. Any hmean parameters are returned in the form used for input - so correlations and standard deviations for any of the covariance related parameters. 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

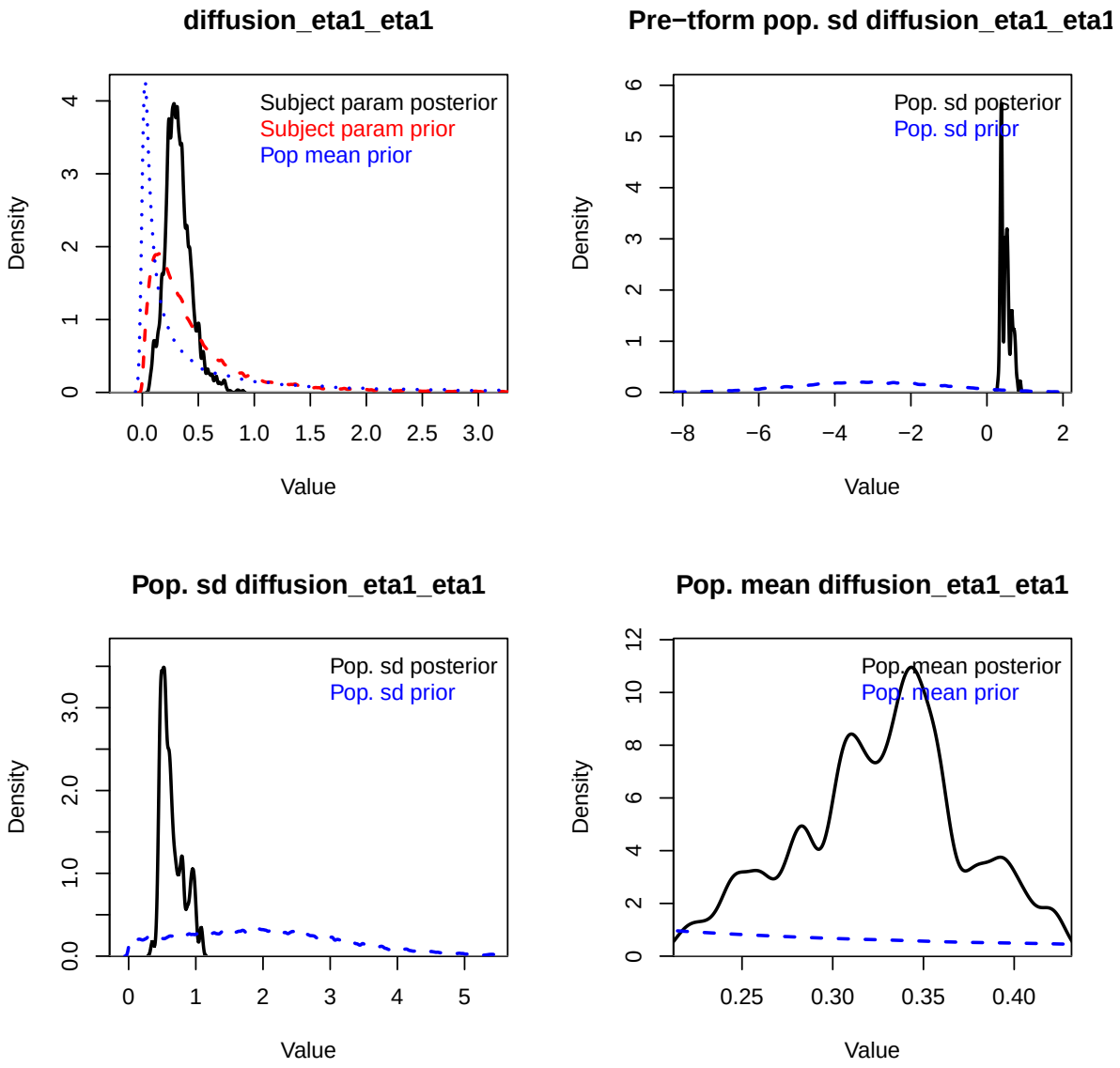
8 is "$DRIFT[8,2,1]$". Parameters in such matrices are returned in the form used for internal calculations – that is, variance covariance matrices are returned as such, rather than the lower-triangular standard deviation and correlation matrices required for input. The exception to this are the time independent predictor effects, prefixed with "*output_tip_*", for which a linear effect of a change of 1 on the predictor is approximated. So although "output_tip_TI1" is only truly linear with respect to internal parameterisations, we approximate the linear effect by averaging the effect of a score of +1 or -1 on the predictor, on the population mean. For any subject that substantially differs from the mean, or simply when precise absolute values of the effects are required (as opposed to general directions), they will need to be calculated manually.

```
ctStanSummary(fit)
library("shinystan")
launch_shinystan(fit)
```

Reported by ctStanSummary are the mean and standard deviation of the distributions, as well as the number of effective samples and the potential scale reduction factor, rhat. More detailed statistics can be obtained using the regular rstan summary function or shinystan.

The relation between posteriors and priors for variables of interest can also be plotted as follows:

```
par(mfrow=c(2,2))
ctStanPlotPost(ctstanmodelobj = model, ctstanfitobj = fit, rows=11)
```



Shown are approximate density plots based on the post-warmup samples drawn. For each parameter four plots are shown – the first displays the posterior distribution of subject level parameters, the subject level prior (generated from repeated sampling of the hyper parameters), and the prior for the population mean.

1.9. Stationarity

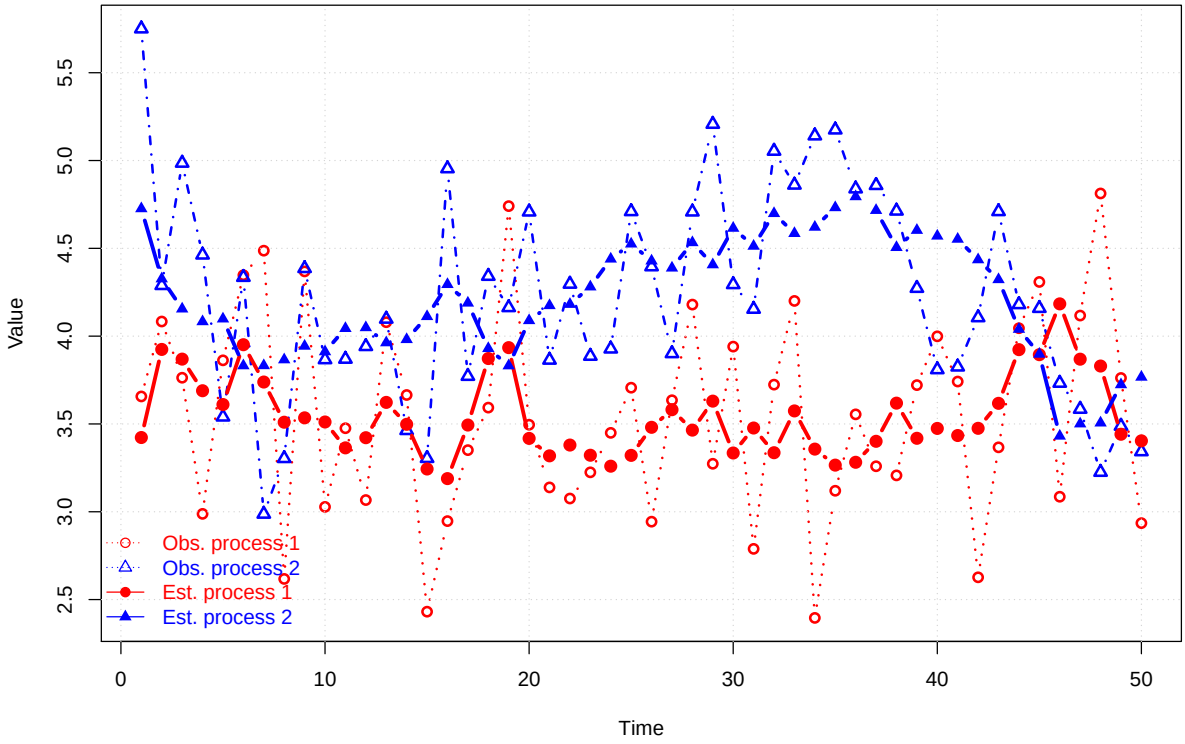
By default, a substantial prior is placed on the stationarity of the dynamic models, calculated as the difference between the T0MEANS and the long run asymptotes of the expected value of the processs, as well as the difference between the diagonals of the T0VAR covariance matrix and the long run asymptotes 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 there are a range of models where one may wish to reduce such a prior – the standard deviation of the normal distribution on the mentioned differences can be observed

and modified using the model*stationarymeanpriorandmodel*stationaryvarprior vectors, where the number of elements in the vector correspond to the number of latent processes.

1.10. Individual level analyses

Individual level results can also be considered, as 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. An example of such is depicted in Figure **??**, where we see observed and estimated scores for a randomly selected subject from our sample. If we wanted to predict unobserved states in the past or future, we would need only to include the relevant time and missing observations in our data structure.

```
subject<-6 #which subject to attain estimates for

#Get kalman estimates
pred<-ctStanPredictions(ctstanmodelobj=model,
  ctstanfitobj=fit,
  datalong=long, subjects=subject)

#Determine y axis limits
ylim<-c(min(c(pred[[subject]]$y,pred[[subject]]$ypred),na.rm=T),
  max(c(pred[[subject]]$y,pred[[subject]]$ypred),na.rm=T))

#plot observed versus smoothed results
plot(pred[[subject]]$y[,1],type='b',lwd=2, ylim=ylim, col='red',
  lty=3,xlab='Time',ylab='Value')
grid()
points(pred[[subject]]$y[,2],type='b',lwd=2,col='blue',pch=2,lty=4)

legend('bottomleft',c('Obs. process 1','Obs. process 2',
  'Est. process 1', 'Est. process 2'),
  pch=c(1,2,19,17),col=c('red','blue'),text.col=c('red','blue'),
  lty=c(3,3,1,1),bty='n')

points(pred[[subject]]$ysmooth[,1],col='red',pch=19,type='b',lty=1,lwd=3)
points(pred[[subject]]$ysmooth[,2],col='blue',pch=17,type='b',lty=1,lwd=3)
```



1.11. Convert from wide data

Data can be converted from the wide format data used for the OpenMx based ctsem approach as follows, though in the event one does not have a ctModel of type omx specified, the necessary values can be filled in by hand.

```
#Where mymodel is a ctsem model of type omx, and mydata is a dataset in the wide for-
mat used by omx type ctsem models.
long<-ctWideToLong(mydata,mymodel$Tpoints,
n.manifest=mymodel$n.manifest, manifestNames = mymodel$manifestNames,
  n.TDpred=mymodel$n.TDpred, TDpredNames = mymodel$TDpredNames,
  n.TIpred=mymodel$n.TIpred, TIpredNames = mymodel$TIpredNames)

long<-ctDeintervalise(long)
```

1.12. 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.

1.13. Oscillating, single subject example - sunspots data

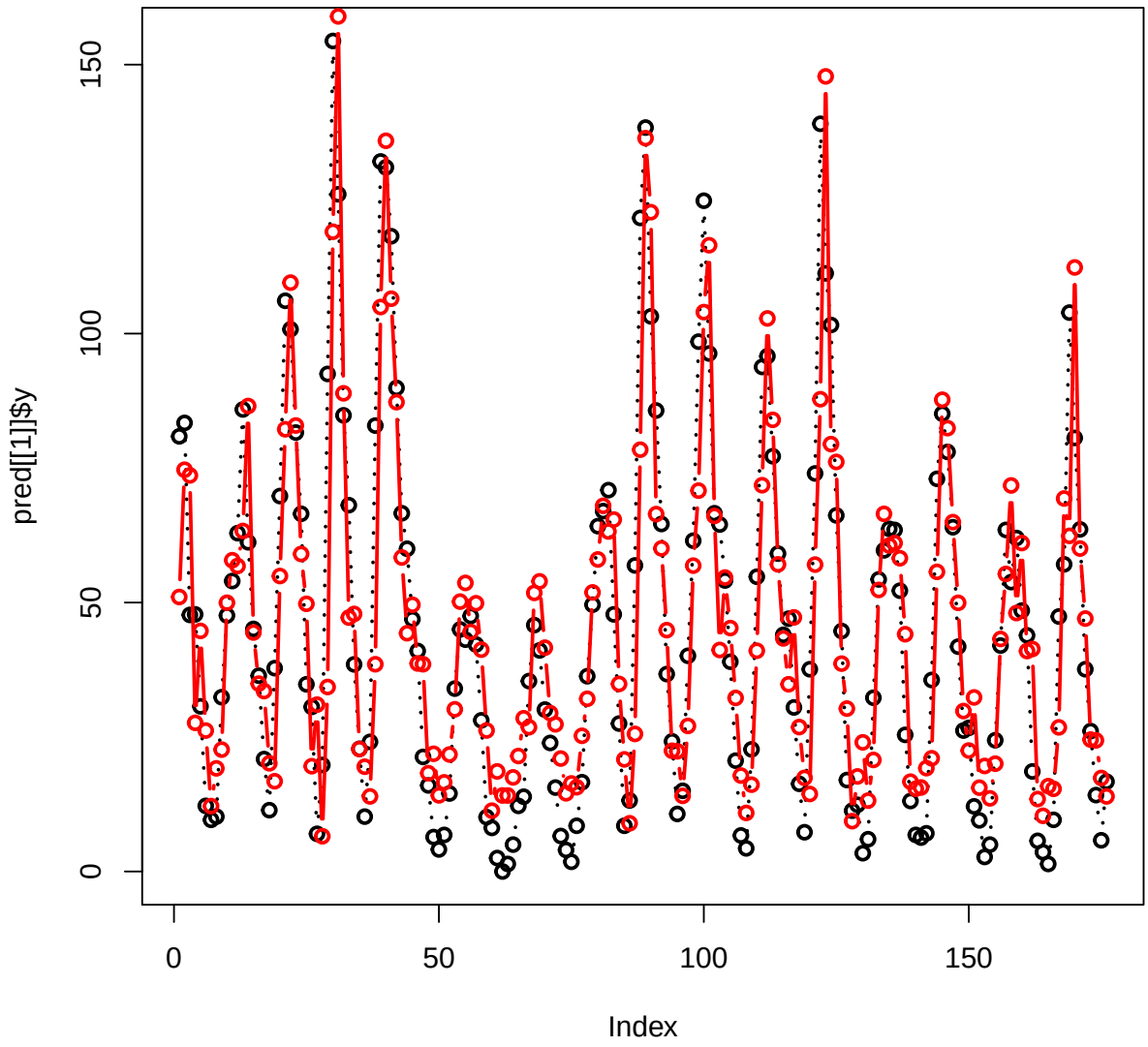
In the following example we fit the sunspots data available within R, which has previously been fit by various authors including Tómasson (2013). We have used the same CARMA(2,1) model and obtained similar estimates – we expect any differences are due to the contrast

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.

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

 model <- ctModel(type='stanct', n.latent=2, n.manifest=1,
  manifestNames='sunspots',
  latentNames=c('sunspots_level','sunspots_velocity'),
   LAMBDA=matrix(c( 1, 'ma1' ), nrow=1, ncol=2),
   DRIFT=matrix(c(
     0,  1,
     'a21',   'a22'
   ), nrow=2, ncol=2, byrow=TRUE),
   MANIFESTMEANS=matrix(c('m1'), nrow=1, ncol=1),
   MANIFESTVAR=matrix(0, nrow=1, ncol=1),
   CINT=matrix(c(0, 0), nrow=2, ncol=1),
   DIFFUSION=matrix(c(
     0, 0,
     0, "diffusion"), ncol=2, nrow=2, byrow=TRUE))

 model$parameters$indvarying<-FALSE
 model$parameters$transform[14]<- '(param)*5+44 '
 model$parameters$transform[7]<-'-log(exp(-param*1.5)+1)'
 model$stationarymeanprior<-5
 model$stationaryvarprior<-5

fit <- ctStanFit(datalong, model, plot=TRUE, iter=200, kalman=T, chains=2,initwithoptim=T)

ctStanSummary(fit)
traceplot(fit,rownames(ctStanSummary(fit)),inc_warmup=TRUE)

pred<-ctStanPredictions(ctstanmodelobj=model,ctstanfitobj=fit,subjects=1,datalong=datalong)

plot(pred[[1]]$y,type='b',lwd=2,lty=3)
points(pred[[1]]$yprior,type='b',col='red',lwd=2)
```

```
iter   10 value 776.339430
iter   20 value 692.196892
iter   30 value 625.808106
iter   40 value 616.155405
iter   50 value 615.901125
iter   60 value 614.050606
iter   70 value 574.428790
iter   80 value 571.462013
iter   90 value 571.398625
iter  100 value 571.323331
iter  110 value 571.320311
iter  120 value 571.297338
```

```
iter  130 value 571.293229
iter  140 value 571.293097
iter  150 value 571.292476
iter  160 value 571.291939
iter  170 value 571.291875
final  value 571.291875
converged
                                                      mean     sd n_eff  Rhat
output_hmean_T0mean_sunspots_level                   3.761 3.4533  7.87 1.639
output_hmean_T0mean_sunspots_velocity                0.267 5.7386  1.87 3.293
output_hmean_ma1                                     0.624 0.1715 21.46 1.072
output_hmean_a21                                    -0.366 0.0428 13.53 1.093
output_hmean_a22                                    -0.412 0.0789  9.09 1.327
output_hmean_diffusion                              16.726 2.5175 15.54 1.101
output_hmean_m1                                     47.106 1.8663  2.81 1.844
output_hmean_T0var_sunspots_level_sunspots_level    32.170 4.8496 25.20 0.991
output_hmean_T0var_sunspots_velocity_sunspots_level  0.117 0.2559  2.77 1.503
output_hmean_T0var_sunspots_velocity_sunspots_velocity 17.876 1.8922 30.00 0.970
lp__                                              -576.188 1.9293  8.64 1.635
```



1.14. Population standard deviations - understanding the transforms

Internally, we sample parameters that we will refer to here as the 'raw' parameters – these parameters have no bounds and are drawn from normal distributions. Both population mean (internally: hypermeans) and subject level (internally: indparamsbase) raw parameters are drawn from a normal(0, 1) distribution. Depending on the specific parameter, various transformations may be applied to set appropriate bounds and priors. The log of the population standard deviation (loghypersd) for these raw parameters is sampled (by default) from a normal(-3, 2) distribution. So, the population standard deviation of the raw parameters can be calculated by a simple exponential of the loghypersd parameter. This resulting distribution can 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 50 subjects. Although we sample the priors here, this is merely to reflect the prior and enable understanding and plotting.

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

```
#population mean and subject level deviations (pre-transformation)

hypermeans_prior <- rnorm(99999, 0, 1)
hypermeans_post <- -2 #hypothetical sample

indparamsbase_prior <- rnorm(99999, 0, 1)
indparamsbase_post <- rnorm(50, 0, 1) #hypothetical sample

#population standard deviation prior

loghypersd_prior <- rnorm(99999, -3, 2)
hypersd_prior <- exp(loghypersd_prior)

#population standard deviation posterior

loghypersd_post <- -1.2 #hypothetical
hypersd_post <- exp(loghypersd_post)

#population cholesky correlation matrix
#lower triangle sampled from uniform(-1, 1),
#upper triangle fixed to 0,
#diagonal calculated according to hypersd.
hypercorrchol_post <- 1 #because only 1 parameter here...

#population cholesky covariance matrix
#here based on mean of hypersd_post, for convenience...
#in reality would have multiple samples.
hypercovchol <- diag(hypercorrchol_post,1) %*%
  diag(hypersd_post,1) %*% diag(hypercorrchol_post,1)

#subject level parameters
#first compute pre transformation parameters
#then transform appropriately (here according to drift auto effect)
indparams <- hypercovchol %*% indparamsbase_post + hypermeans_post
indparams <- -log(exp(-1.5 * indparams) + 1)

#post transformation population standard deviation
hsd_ourparameter <- abs( #via delta approximation
  (-log(exp(-1.5 * (hypermeans_post + hypersd_post)) + 1) -
   -log(exp(-1.5 * (hypermeans_post - hypersd_post)) + 1) ) / 2)

sd(indparams_posttransform) #compute sd directly

Error in is.data.frame(x):  object 'indparams_posttransform' not found
```

2. Complete model specification

3. The model

There are three main elements to our hierarchical continuous time dynamic model. There is a subject level latent dynamic model, a subject level measurement model, and a population level model for the subject level parameters. Note that while various elements in the model depend on time, the fundamental parameters of the model as described here are time-invariant.

3.1. Subject level latent dynamic model

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

$$d\boldsymbol{\eta}(t) = \Big(\mathbf{A}\boldsymbol{\eta}(t) + \mathbf{b} + \mathbf{M}\boldsymbol{\chi}(t)\Big)dt + \mathbf{G}d\mathbf{W}(t) \tag{4}$$

Vector $\boldsymbol{\eta}(t) \in \mathbb{R}^{\upsilon}$ represents the state of the latent processes at time t. The matrix $\mathbf{A} \in \mathbb{R}^{\upsilon \times \upsilon}$ represents the so-called 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 $\mathbf{b} \in \mathbb{R}^{\upsilon}$, 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 4 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 5, 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 an example see Driver *et al.* (In Press).

$$\boldsymbol{\chi}(t) = \sum_{u \in \mathbf{U}} \mathbf{x}_u \delta(t - t_u) \tag{5}$$

Here, time dependent predictors $\mathbf{x}_u \in \mathbb{R}^{l}$ 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 $\mathbf{M} \in \mathbb{R}^{\upsilon \times l}$.

$\mathbf{W}(t) \in \mathbb{R}^{\upsilon}$ represents independent Wiener processes, with a Wiener process being a random-walk in continuous time. $d\mathbf{W}(t)$ 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 $\mathbf{G} \in \mathbb{R}^{\upsilon \times \upsilon}$ 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.

3.2. Subject level dynamic model — discrete time solution

The stochastic differential Equation 4 may be solved and translated to a discrete time repre-

sentation, for any observation $u \in \mathbf{U}$, where $\mathbf{U}$ is the set of measurement occasions from 1 to the number of measurement occasions, with $u = 1$ treated as occurring at $t = 0$:

$$\boldsymbol{\eta}_u = \mathbf{A}^*_u \boldsymbol{\eta}_{u-1} + \mathbf{b}^*_u + \mathbf{M}\mathbf{x}_u + \boldsymbol{\zeta}^*_u \quad \boldsymbol{\zeta}^*_u \sim \mathrm{N}(\mathbf{0}_v, \mathbf{Q^*}_u) \tag{6}$$

The * notation is used to indicate a term that is the discrete time equivalent of the original. $\mathbf{A}^*_u$ then contains the appropriate auto and cross regressions for the effect of latent processes $\boldsymbol{\eta}$ at measurement occasion $u-1$ on $\boldsymbol{\eta}$ at measurement occasion u. $\mathbf{b}^*_u$ represents the discrete time intercept for measurement occasion u. Since $\mathbf{M}$ is conceptualized as the effect of instantaneous impulses $\mathbf{x}$ which only occur at occasions $\mathbf{U}$ (and not continuously present as for the processes $\boldsymbol{\eta}$), the discrete and continuous time forms are equivalent. $\boldsymbol{\zeta}_u$ is the zero mean random error term for the processes at occasion u. Q^*_u represents the multivariate normal disturbance at occasion u. The recursive nature of the solution means that at the first measurement occasion $u = 1$, the system must be initialised in some way, with $\mathbf{A}^*_u \boldsymbol{\eta}_{u-1}$ replaced by $\boldsymbol{\eta}_1$, and $\mathbf{Q^*}_u$ replaced by $\mathbf{Q}^*_1$. These initial states and covariances are later referred to as T0MEANS and T0VAR respectively.

Unlike in a purely discrete time model, where the various effect matrices described above would be unchanging, in a continuous time model the discrete time matrices all depend on some function of the continuous time parameters and the time interval between observations u and $u-1$, these functions look as follows:

$$\mathbf{A}^*_u = e^{\mathbf{A}(t_u - t_{u-1})} \tag{7}$$

$$\mathbf{b}^*_u = \mathbf{A}^{-1}(\mathbf{A}^*_u - \mathbf{I})\mathbf{b} \tag{8}$$

$$\mathbf{Q}^*_u = \mathbf{Q}_\infty - \mathbf{A}^*_u \mathbf{Q}_\infty (\mathbf{A}^*_u)^\top \tag{9}$$

Where $\mathbf{A}_\# = \mathbf{A} \otimes \mathbf{I} + \mathbf{I} \otimes \mathbf{A}$, with $\otimes$ denoting the Kronecker-product, the asymptotic diffusion $\mathbf{Q}_\infty = \mathrm{irow}\left(-\mathbf{A}^{-1}_\# \,\mathrm{row}(\mathbf{Q})\right)$, row is an operation that takes elements of a matrix row wise and puts them in a column vector, and irow is the inverse of the row operation. The covariance update shown in Equation 9 has not been described in the psychological literature so far as we are aware, but is a more computationally efficient form used in Tómasson (2013).

3.3. Subject level measurement model

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

$$\mathbf{y}(t) = \boldsymbol{\Lambda}\boldsymbol{\eta}(t) + \boldsymbol{\tau} + \boldsymbol{\epsilon}(t) \quad \text{where } \boldsymbol{\epsilon}(t) \sim \mathrm{N}(\mathbf{0}_c, \boldsymbol{\Theta}) \tag{10}$$

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

3.4. Subject level likelihood

The subject level likelihood, conditional on time dependent predictors $\mathbf{x}$ and subject level

parameters $\boldsymbol{\theta}$, is as follows:

$$p(\mathbf{y}|\boldsymbol{\theta},\mathbf{x}) = \prod^{\mathbf{U}} p(\mathbf{y}_u|\mathbf{y}_{\left(u-1,\ldots,u-(u-1)\right)},\mathbf{x}_u,\boldsymbol{\theta}) \tag{11}$$

To avoid the large increase in parameters that comes with sampling or optimizing latent states, we use a continuous-discrete (or hybrid) Kalman filter to analytically compute subject level likelihoods, conditional on subject parameters. For more on filtering see Jazwinski (2007) and Särkkä (2013). The filter operates with a prediction step, in which the expectation $\hat{\boldsymbol{\eta}}_{u|u-1}$ and covariance $\hat{\mathbf{P}}_{u|u-1}$ of the latent states are predicted by:

$$\hat{\boldsymbol{\eta}}_{u|u-1} = \mathbf{A}^*_u\hat{\boldsymbol{\eta}}_{u-1|u-1} + \mathbf{b}^*_u + \mathbf{M}\mathbf{x}_u \tag{12}$$

$$\hat{\mathbf{P}}_{u|u-1} = \mathbf{A}^*_u\hat{\mathbf{P}}_{u-1|u-1}(\mathbf{A}^*_u)^\top + \mathbf{Q}^*_u \tag{13}$$

For the first occasion $u = 1$, the priors $\hat{\boldsymbol{\eta}}_{u-1|u-1}$ and $\hat{\mathbf{P}}_{u-1|u-1}$ must be provided to the filter. These parameters may in some cases be freely estimated, but in other cases need to be fixed or constrained, either to specific values or by enforcing a dependency to other parameters in the model, such as an assumption of stationarity.

Prediction steps are followed by an update step, wherein rows and columns of matrices are filtered as necessary depending on missingness of the measurements $\mathbf{y}$:

$$\hat{\mathbf{y}}_{u|u-1} = \boldsymbol{\Lambda}\hat{\boldsymbol{\eta}}_{u|u-1} + \boldsymbol{\tau} \tag{14}$$

$$\hat{\mathbf{V}}_u = \boldsymbol{\Lambda}\hat{\boldsymbol{\eta}}_{u|u-1}\boldsymbol{\Lambda}^\top + \boldsymbol{\Theta} \tag{15}$$

$$\hat{\mathbf{K}}_u = \hat{\mathbf{P}}_{u|u-1}\boldsymbol{\Lambda}^\top\hat{\mathbf{V}}_u^{-1} \tag{16}$$

$$\hat{\boldsymbol{\eta}}_{u|u} = \hat{\boldsymbol{\eta}}_{u|u-1} - \hat{\mathbf{K}}_u(\mathbf{y}_u - \hat{\mathbf{y}}_{u|u-1}) \tag{17}$$

$$\hat{\mathbf{P}}_{u|u} = (\mathbf{I} - \hat{\mathbf{K}}_u\boldsymbol{\Lambda})\hat{\mathbf{P}}_{u|u-1} \tag{18}$$

The log likelihood (ll) for each subject, conditional on subject level parameters, is typically[1]

[1]For computational reasons we use an alternate but equivalent form of the log likelihood. We scale the prediction errors across all variables to a standard normal distribution, drop constant terms, calculate the log likelihood of the transformed prediction error vector, and appropriately update the log likelihood for the change in scale, as follows:

$$ll = \sum^{\mathbf{U}}\left(log\left(tr(\mathbf{V}_u^{-1/2})\right) + \sum 1/2\left(\mathbf{V}_u^{-1/2}(\hat{\mathbf{y}}_{(u|u-1)} - \mathbf{y}_u)\right)\right) \tag{19}$$

Where tr indicates the trace of a matrix, and $\mathbf{V}^{-1/2}$ is the inverse of the Cholesky decomposition of $\mathbf{V}$. The Stan software manual discusses such a *change of variables* (Stan Development Team 2016).

then (Genz and Bretz 2009):

$$ll = \sum^{\mathbf{U}} \Bigg(-1/2(n\ln(2\pi) + \ln\left|\mathbf{V}_u\right| + (\hat{\mathbf{y}}_{(u|u-1)} - \mathbf{y}_u)\mathbf{V}_u^{-1}(\hat{\mathbf{y}}_{u|u-1} - \mathbf{y}_u)^\top) \Bigg) \tag{20}$$

Where n is the number of non-missing observations at u.

3.5. Population level model

Rather than assume complete independence or dependence across subjects, we assume subject level parameters are drawn from a population distribution, for which we also estimate parameters and apply some prior. This results in a joint-posterior distribution of:

$$p(\mathbf{\Phi}, \boldsymbol{\mu}, \mathbf{R}, \boldsymbol{\beta}|\mathbf{Y}, \mathbf{Z}) = \frac{p(\mathbf{Y}|\mathbf{\Phi})p(\mathbf{\Phi}|\boldsymbol{\mu}, \mathbf{R}, \boldsymbol{\beta}, \mathbf{Z})p(\boldsymbol{\mu}, \mathbf{R}, \beta)}{p(\mathbf{Y})} \tag{21}$$

Where subject specific parameters $\mathbf{\Phi}_i$ are determined in the following manner:

$$\mathbf{\Phi}_i = \text{tform}\Big(\boldsymbol{\mu} + \mathbf{R}\mathbf{h}_i + \boldsymbol{\beta}\mathbf{z}_i\Big) \tag{22}$$

$$\mathbf{h}_i \sim \text{N}(0,1) \tag{23}$$

$$\boldsymbol{\mu} \sim \text{N}(0,1) \tag{24}$$

$$\boldsymbol{\beta} \sim \text{N}(0,1) \tag{25}$$

$\mathbf{\Phi}_i \in \mathbb{R}^s$ represents all parameters for the dynamic and measurement models of subject i. $\boldsymbol{\mu} \in \mathbb{R}^s$ parameterizes the population means of the distribution of subject level parameters. $\mathbf{R} \in \mathbb{R}^{s\times s}$ is the Cholesky factor of the population covariance matrix, parameterizing the effect of subject specific deviations $\mathbf{h}_i \in \mathbb{R}^s$ on $\mathbf{\Phi}_i$. $\boldsymbol{\beta} \in \mathbb{R}^{s\times w}$ is the effect of time independent predictors $\mathbf{z}_i \in \mathbb{R}^w$ on $\mathbf{\Phi}_i$. $\mathbf{Y}_i$ contains all the data for subject i used in the dynamic 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 — these transforms and rationale are described below, but are in general necessary because many parameters require some bounded distribution, making a purely linear approach untenable.

Besides the tform operator, Equation 22 looks like a relatively standard hierarchical approach, with subject parameters dependent on a population mean and covariance, and observed covariates. Subject specific parameters $\mathbf{h}_i$ are in standardised deviation form to effect a non-centered parameterization, which appears to improve sampling efficiency in this model. See Bernardo, Bayarri, Berger, Dawid, Heckerman, Smith, and West (2003) and Betancourt and Girolami (2013) for discussion of non-centered parameterizations.

4. Parameter transformations and resulting priors

Because all sampled parameters are drawn from an unconstrained standard normal distribution, we need a variety of transformations that can be applied depending on what the parameter represents. The transformation serves to provide both the upper and lower bounds (if they exist) for the parameter of interest, as well as the prior density. While for many model parameters a simple multiplication of the standard normal is adequate, standard deviations and correlations require a bounded distribution, while we have opted to use a bounded distribution on the drift auto effects for pragmatic reasons – values greater than 0 represent explosive, non-stationary processes that are in most cases not theoretically plausible and occur only due to model misspecification. Further, positive values can result in additional local minima, causing problems for optimization or sampling. While allowing for such values may point to misspecification more readily, the constrained form results in what we believe is a computationally simpler and sensible prior distribution for genuine effects. Estimated values close to 0 when using this constrained form may point to the need to consider the model specification, perhaps by including higher order terms. The transformations we use for the various parameter types are as follows:

$$\text{Standard deviation: } e^{4x} \tag{26}$$

$$\text{Drift auto effect (diagonal): } -log(e^{-1.5x} + 1) \tag{27}$$

$$\text{Correlation: } 2/(1 + e^{-1.5x}) - 1 \tag{28}$$

Covariance matrices are handled in a two step procedure, to ensure that priors on correlation parameters are independent of scale parameters – a long recognised problem with various common parameterisations of covariance matrices, see for instance Huang and Wand (2013) and Gelman (2006). Off-diagonals are first transformed via the inverse logit function and scaled to the range -1 to 1, and used in the lower triangle of a Cholesky decomposed correlation matrix. The diagonal of the Cholesky correlation matrix is determined based on both the standard deviations and correlations. Then, the covariance parameters are calculated by pre-multiplying the Cholesky correlation matrix by the scale matrix, which is simply the diagonal matrix containing the standard deviations. This approach ensures that the off-diagonal parameters are independent of the diagonals, in terms of both prior expectations and during sampling. Discussion of this general approach to handling covariance matrices may be found in the Stan software manual (Stan Development Team 2016), and full details in Lewandowski, Kurowicka, and Joe (2009).

Figure 1 plots the resulting densities when using the described transformations. Note that of course the density for a variance is directly related to the standard deviation, and the density plot for an autoregression assumes that the time interval is 1 with no cross effects involved. For the sake of completeness we include a prior density for all other parameters, such as the drift cross effects, intercepts, and regression type parameters. These use a simple multiplication of the standard normal.

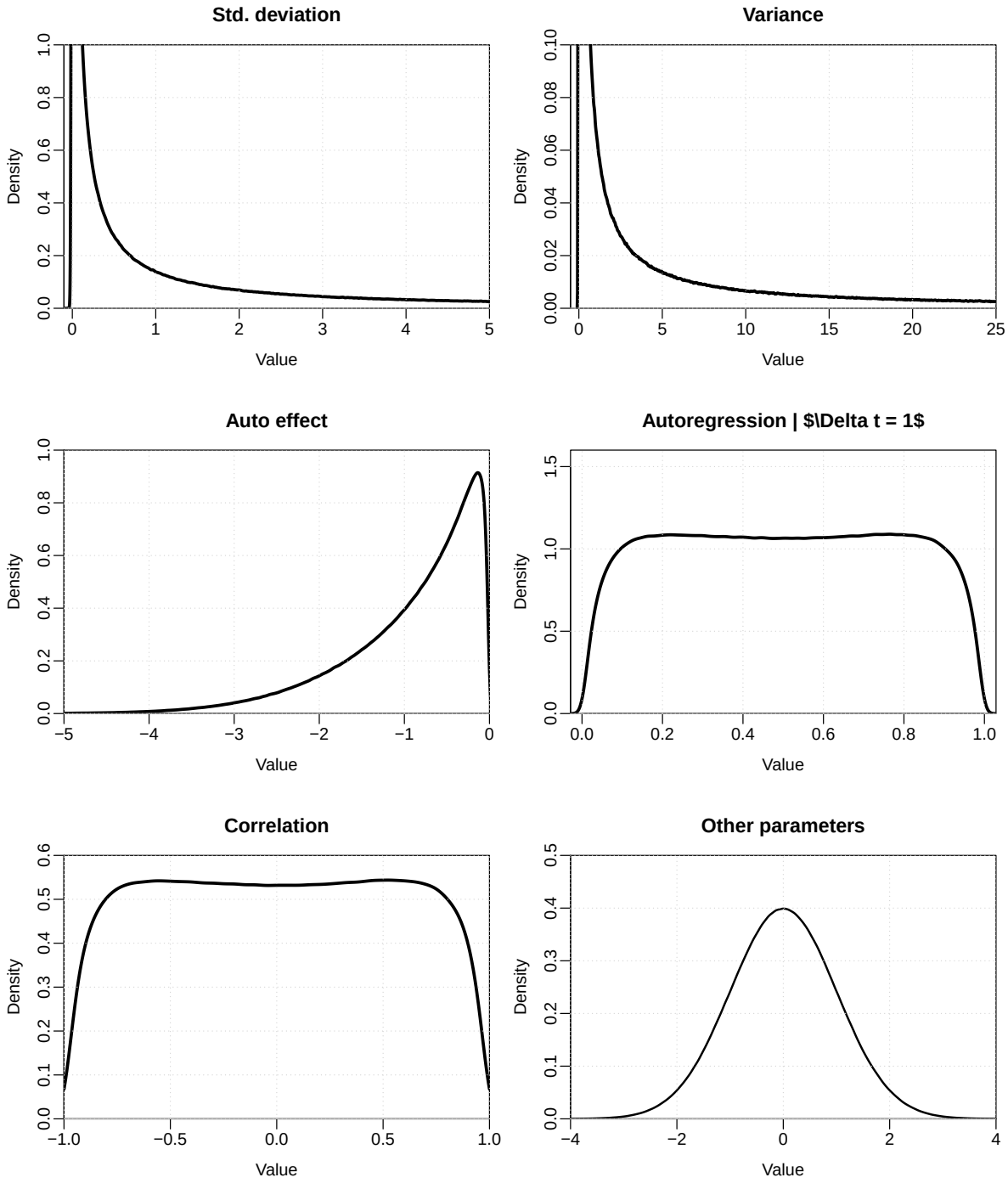


Figure 1: Priors for population mean parameters.

References

Bernardo JM, Bayarri MJ, Berger JO, Dawid AP, Heckerman D, Smith AFM, West M (2003). "Non-Centered Parameterisations for Hierarchical Models and Data Augmentation." *Bayesian Statistics 7: Proceedings of the Seventh Valencia International Meeting*, p. 307.

Betancourt MJ, Girolami M (2013). "Hamiltonian Monte Carlo for Hierarchical Models." *arXiv:1312.0906 [stat]*. 1312.0906.

Driver C, Oud J, Voelkle M (In Press). "Continuous Time Structural Equation Modelling with R Package Ctsem." *Journal of Statistical Software*.

Gelman A (2006). "Prior Distributions for Variance Parameters in Hierarchical Models (Comment on Article by Browne and Draper)." *Bayesian Analysis*, **1**(3), 515–534. ISSN 1936-0975, 1931-6690. doi:10.1214/06-BA117A.

Genz A, Bretz F (2009). *Computation of Multivariate Normal and T Probabilities*, volume 195 of *Lecture Notes in Statistics*. Springer Berlin Heidelberg, Berlin, Heidelberg. ISBN 978-3-642-01688-2 978-3-642-01689-9.

Huang A, Wand MP (2013). "Simple Marginally Noninformative Prior Distributions for Covariance Matrices." *Bayesian Analysis*, **8**(2), 439–452. ISSN 1936-0975, 1931-6690. doi:10.1214/13-BA815.

Jazwinski AH (2007). *Stochastic Processes and Filtering Theory*. Courier Corporation. ISBN 978-0-486-46274-5.

Lewandowski D, Kurowicka D, Joe H (2009). "Generating Random Correlation Matrices Based on Vines and Extended Onion Method." *Journal of Multivariate Analysis*, **100**(9), 1989–2001. ISSN 0047-259X. doi:10.1016/j.jmva.2009.04.008.

Särkkä S (2013). *Bayesian Filtering and Smoothing*. Cambridge University Press. ISBN 978-1-107-03065-7. Google-Books-ID: 5VlsAAAAQBAJ.

Stan Development Team (2016). "Stan Modeling Language Users Guide and Reference Manual, Version 2.9.0."

Tómasson H (2013). "Some Computational Aspects of Gaussian CARMA Modelling." *Statistics and Computing*, **25**(2), 375–387. ISSN 0960-3174, 1573-1375. doi:10.1007/s11222-013-9438-9.

Affiliation:

Charles Driver
Center for Lifespan Psychology
Max Planck Institute for Human Development
Lentzeallee 94, 14195 Berlin
Telephone: +49 30 82406-367 E-mail: driver@mpib-berlin.mpg.de
URL: http://www.mpib-berlin.mpg.de/en/staff/charles-driver