A Hierarchical extension to Ornstein-Uhlenbeck-type Student's t-processes

Ville Laitinen & Leo Lahti / University of Turku / velait@utu.fi

20 3 2018

Abstract

Introduction

The aim of this work is to provide a hierarchical extension of the Ornstein-Uhlenbeck type processes, in order to aggregate information across multiple (short) time series. This extends the recent Stan implementation [@Goodman2018], where parameter estimates of a Student-t type OU process are obtained based on a single (long) time series. We have added a level of hierarchy, which allows inference of the model parameters based on multiple time series. We validate the implementation and investigate its robustness to variations in sample size and time series lengths based on simulated data.

Motivation for this work comes from the analysis of human gut microbiome dynamics. It has been reported that on average the abundance of many gut bacteria remains relatively stable over long time periods [@david_host_2014]. On a shorter (daily) time scale these abundances do however have considerable fluctuations. A number of cross-sectional studies of the human microbiome have characterized the diversity and variation of the gut microbiome between individuals [e.g. @hmp_huttenhower, @Qin2009]. The temporal variation of different taxonomic units within individuals is, however, less well understood [@faust_metagenomics_2015]. One of the current challenges is that the currently available time series are often short, sparse and noisy, and robust inference of dynamical models can be challenging in such case. Some of these limitations can be addressed by adding a level of hierarchy to the model in order to aggregate information across multiple time series. In addition, such extensions can potentially help to analyze the individuality of the time series from different study subjects.

Moreover, given the complex and highly individual nature of the gut ecosystem, exact dynamical models of the underlying system are missing. The OU-type processes provide means to characterize key properties of system dynamics, such as the location and resilience of the potential wells, in a non-parametric manner, when knowledge on the data-generating mechanisms is missing. Therefore, variants of the OU process appear to provide rigorous and justified methods for modeling these dynamics. However, apart from [@Goodman2018] we are not aware of applications of these models, in particular its hierarchical extensions that we develop here, in the context of human microbiome studies. The methodology itself is very generic, and its potential applications naturally reach beyond population dynamics.

Background

Ornstein-Uhlenbeck process

The Ornstein-Uhlenbeck process (OUP), also known as the Langevin equation in physics and Vasicek model in finance, is a stochastic process with a wide range of applications [@Iacus_SDE]. It can be used to model systems with a steady state that recover from perturbations by returning to the long term mean. The OUP is defined by the stochastic differential equation

$$dX_t = \lambda(\mu - X_t)dt + \sqrt{2\kappa\lambda}dZ_t,$$

where X_t is the state of the system at time t and Z a stochastic process. Unlike with an ordinary differential equation, the solutions of the stochastic counterpart are nowhere differentiable and non-unique as they are

different for different realizations of the noise term. Averaging over these solutions recovers the deterministic solution.

The first term on the right hand side ("drift") describes the deterministic behavior and the second term ("dispersion") characterises the stochasticity of the system. The parameters have natural interpretations as mean-reversion rate (λ), mean (μ) and size of stochastic fluctuations (κ).

Usually the stochastic process is modeled as white noise but for practical purposes requiring Z_t to be Brownian motion with Gaussian transition density is often too a limiting assumption as it does not allow large enough fluctuations and thus is not robust against outliers [@solin_sarkka]. A more general choice is to use the Student-t process that allows greater variance between consecutive points. The process f is a Student-t process, $f \sim \mathcal{ST}(\nu, \mu, K)$, with ν degrees of freedom, mean paramter μ and covariance kernel K,if any finite set of values is multivariate Student-t distributed. A vector $\bar{y} \in \mathbb{R}^n$ is multivariate Student-t distributed, $\bar{y} \sim \mathcal{ST}_n(\nu, \mu, K)$ if it has density

$$p(\bar{y}) = \frac{\Gamma(\frac{\nu+n}{2})}{((\nu-2)\pi)^{\frac{n}{2}}\Gamma(\frac{\nu}{2})}|K|^{-\frac{1}{2}} \times \Big(1 + \frac{(\bar{y}-\bar{\mu})^T K^{-1}(\bar{y}-\bar{\mu})}{\nu-2}\Big)^{-\frac{\nu+n}{2}}$$

In general this model assumes that the process density is unimodal and likelihood of a point decreases as the distance to the mode increases. This assumption ensures that the model satisfies the relatively simple dynamical nature of a single potential well. Elliptically symmetric processes have such properties and it is known [@shah_student-t] that the Student-t processes are the largest subset of elliptically symmetric process that have an analytical solution. It is a convenient choice also in the sense that the Gaussian process can be obtained as a special case [@solin_sarkka].

Transition density $X_t|X_0$ of a Gaussian OUP is normally distributed, with mean $\mu - (\mu - X_0)e^{-\lambda t}$ and variance $\kappa(1 - e^{-2\lambda t})$. From these expressions it's easy to obtain the long term mean, μ, and variance, κ, as $t \to \infty$. Covariance between two time points is given by

$$\mathrm{Cov}[X_t, X_{t+\Delta t}] = \kappa e^{-\lambda \Delta t}.$$

Now let us recall that if $X \sim \mathcal{N}(\mu, \sigma^2)$, then the random variable $X + \epsilon\sigma$, where $\epsilon \sim \mathcal{N}(0, 1)$ is Student-t distributed with $\nu = 1$ degrees of freedom. Thus we get an expression relating error terms ϵ_i and process values X_i at times t_i and t_{i-1}, $\Delta t = t_i - t_{i-1}$:

$$X_1 = \mu + \epsilon_1\sqrt{\kappa}$$

and

$$X_i = \mu - (\mu - X_{i-1})e^{-\lambda\Delta t} + \epsilon_i\sqrt{\kappa(1 - e^{-2\lambda\Delta t})},$$

for $i = 2, \ldots, n$.

Conditional expression for the density of error terms can be derived from Lemma 3 in [@shah_student-t]

$$\epsilon_i|\epsilon_1, \ldots, \epsilon_{i-1} \sim \mathrm{MVT}_1\Big(\nu + i - 1, 0, \frac{\nu - 2 + \sum_{k=1}^{i-1}\epsilon_k^2}{\nu - 3 + i}\Big),$$

which reduces to

$$p(\epsilon_i|\epsilon_1, \ldots, \epsilon_{i-1}) \propto \Gamma(\tfrac{\nu+i}{2})\Gamma(\tfrac{\nu+i-1}{2})^{-\frac{1}{2}}\big(\nu - 2 + \sum_{k=1}^{i-1}\epsilon_k^2\big)^{-\frac{1}{2}}\Big(1 + \frac{\epsilon_i^2}{\nu - 2 + \sum_{k=1}^{i-1}\epsilon_k^2}\Big)^{\frac{\nu+i}{2}}.$$

We will use this expression in the Stan code model block to increment the log density.

Hierarchical extension

The model outlined above essentially described the Ornstein-Uhlenbeck driven t-process as implemented in [@Goodman]. The novel contribution to this work that we present now is in equipping the model with hierarchical structure and testing the robustness of the extended implementation. Let $\mathcal{X} = \{\bar{X}_j, j \in \{1, \ldots, N\}\}$ be a set of latent values, with n_j observations in each, each j respresenting e.g. a different measurement site. We assume a hierarchical structure for the parameters λ, μ and κ,

$$dX_{i,t} = \lambda_i(\mu_i - X_{i,t})dt + \sqrt{2\kappa_i\lambda_i}dZ_t,$$

for all $i \in \{i, \ldots, N\}$.

The model

The stochastic Gompertz model is one of the simplest available tools for modeling ecological time-series data and is transformed to the OUP via a log-transformation [@dennis_densitydependent_2014]. We model the observations of the latent OU driven t-process as a Poisson process. Although the observations times do not have to be equally spaced, this model requires them to be identical for each series.

The Stan code uses a non-centered parameterization that is modelled using the error terms ϵ_i. We also experimented with the centered parameterization but with less accurate results and more divergent transitions. This is in agreement with [@stan_manual, p.145] where it was mentioned that hierarchical models tend to do better with non-centered parameterizations, especially when the sample size is limited.

Shortly the idea of the Stan code is as follows. After declaring the data and parameters in the corresponding blocks, error terms ϵ_i and latent values X_i are related in the transformed data block. In the model block parameters and conditional densities for the error terms are incremented to the log density and the observations Y_{ji} are sampled from a Poisson distribution with X_i as the rate.

```
data{
int <lower=0> N;          //Number of time series
int <lower=0> T;          //Number of time points, same in each
int<lower=0> Y[N,T];      //observations matrix as 2D array
vector[T] time;           //observation times
}
transformed data{
vector[T] time_vec = to_vector(time);
}
parameters{
vector[N] lambda_log;
vector[N] kappa_log;
vector[N] mu;
real <lower=2> student_df;
matrix[N,T] error_terms;   //Error terms
}
transformed parameters{
vector[N] sigma_log = 0.5*(kappa_log + lambda_log + log2());
vector<lower=0>[N] sigma = exp(sigma_log);
vector<lower=0>[N] lambda = exp(lambda_log);
vector<lower=0>[N] kappa = exp(kappa_log);
vector<lower=0>[N] kappa_inv = exp(-kappa_log);
vector<lower=0>[N] kappa_sqrt = exp(0.5*kappa_log);
matrix[N, T] X_latent;

// Relate error terms to latent values
```

```
for(i in 1:N){

vector [T-1] delta_t = segment(time_vec, 2, T - 1) - segment(time_vec,1,T-1);
real cum_squares = 0;
real X;

for(k in 1:T){
real epsilon = error_terms[i,k];
if(k == 1){
//For the first latent value use the stationary distribution.
X = mu[i] + epsilon * kappa_sqrt[i];
}else{
real t = delta_t[k-1];
real exp_neg_lambda_t = exp(-t*lambda[i]);
real sd_scale = kappa_sqrt[i] .* sqrt(1-square(exp_neg_lambda_t));
X = mu[i] - (mu[i] - X) .* exp_neg_lambda_t + epsilon .* sd_scale;
}
X_latent[i, k] = X;
}
}

}
model{
target += lambda_log;
target += kappa_log;
student_df ~ gamma(2,.1);

// Increment the log probability according to the conditional expression for
// the error terms
for(i in 1:N){
// row vector to regular
vector[T] error_terms2 = to_vector(square(segment(error_terms[i],1,T)));
vector[T] cum_error_terms2 = cumulative_sum(append_row(0,error_terms2[1:T-1]));

for(k in 1:T){
target +=  (lgamma((student_df + k) * 0.5) - lgamma((student_df+ k - 1 )* 0.5));
target += -0.5 * (student_df + k) * log1p(error_terms2[k] / (student_df + cum_error_terms2[k] - 2));
target += -0.5 * log(student_df + cum_error_terms2[k] - 2);
}
}
// Log probability for the observations given the latent values
for(i in 1:N) {
Y[i] ~ poisson_log(X_latent[i]);
}

// Prior probabilities.
lambda ~ gamma(2,2);
kappa ~ gamma(2,2);
mu ~ normal(0,5);

}
```

Testing

The model is tested on simulated data generated by the sampling scheme in Lemma 2.2 [@solin_sarkka]: if $\bar{y}|\gamma \sim \mathcal{N}(\mu, \gamma K)$, where γ is inverse gamma distributed $\gamma \sim \text{IG}(\nu/2, (\nu-2)/2)$, then marginally $\bar{y} \sim \text{MTV}_n(\mu, K, \nu)$. The function `generate_data_set` creates a list of data in the form that the Stan model can read.

Robustness of parameter estimates is tested with several different data sets. First we'll let the number of observations run from 2 to 10 and number of time series from 5 to 25 in increments of 5. Then we'll fix the number of observations (5) and let the number of series run from 5 to 100 in increments of 5.

```
data_set <- list()

# observation 2 -> 10, increments of 1
for(i in 5*(1:5)){
for(j in 2:10) {
data_set[[paste(i,"series",j,"observations")]] <- generate_data_set(kappa=0.1,
lambda=1, mu=5, intervals=1:j, n_series=i, t.df=7, seed=1)
}
}

# 5 observations, series 30 -> 100, increments of 5
for(i in 5*(6:20)) {
data_set[[paste(i,"series",5,"observations")]] <- generate_data_set(kappa=0.1,
lambda=1, mu=5, intervals=1:5, n_series=i, t.df=7, seed=1)
}

n_series <- 5*1:20
n_obs <- 2:10
```

Next input simulated data to `hier_concentered.stan`. Sampling for the entire data set took a considerable amount of time on a basic laptop (1,3GHz, 4 cores), close to 11 hours, so the entire data set is not sampled in this notebook. Default settings were used, but in order to decrease the amount of divergent transitions `adapt_delta` was set to 0.9. Divergencies were not eliminated altogether and the amount ranged from 0 to 50, averaging at about 7 per sampling.

```
hier_model <- stan_model("hierarchical_noncentered.stan")
fit_list <- list()
for(i in 1:length(data_set)) {
fit_list[[names(data_set)[[i]]]] <-
  sampling(noncentered_hiearchical, data_set[[i]], control=list(adapt_delta=0.9))
}
```

We'll measure the success of parameter estimates in two ways. An estimate is considered successful if the real value used in simulation lies within the 95% highest posterior density interval or between the 2.5% and 97.5% percentiles of the posterior. Then we'll calculate the proportion of succesful estimates per data set.

```
results <- list()
simulation_value <- c(kappa=0.1, lambda=1, mu=5, student_df=7)
for(p in c("kappa", "lambda", "mu")) {

quan <- list()
hpdi <- list()

for(i in 1:length(fit_list)) {
quan[[names(fit_list)[[i]]]] <- success_rate_quantiles(
```

```
fit_list[[i]], parameter=p, real_value = simulation_value[[p]])

hpdi[[names(fit_list)[[i]]]] <- success_rate_HPDI(
fit_list[[i]], parameter=p, real_value = simulation_value[[p]])
}

for(v in c("hpdi_table", "quan_table")) {
m <- matrix(nrow=length(n_series), ncol=length(n_obs))
colnames(m) <- as.character(n_obs)
rownames(m) <- as.character(n_series)
results[[p]][[v]] <- m
}

for(i in 1:length(quan)) {
# dissect no. of observation and series from the name
obs <- sub(".*\\b(\\d+).*", "\\1", names(quan)[[i]])
series <- sub("\\D*(\\d+).*", "\\1", names(quan)[[i]])
results[[p]][["hpdi_table"]][series, obs] <- hpdi[[i]]
results[[p]][["quan_table"]][series, obs] <- quan[[i]]
}

}

source("OU.plots.R")
```

Precision

Results for the precicion estimates κ and μ are shown in Figures 1 - 2 below. In terms of these metrics the model estimated λ with perfect precision. The mean parameter μ was recovered on a satisfactory level as well, but interestingly the precision of κ estimates decreased as more data was available. This may possible be due to a less than ideal prior choice of `kappa  Gamma(2,2)` that does not have very much density in the vicinity of the actual simulation value $\kappa = 0.1$. The posterior density may also have a relatively wide shape with small sample sizes which would also make the 95% HPDI wide. On the other hand the initial decline in the precision estimate seems to level in Fig. 2 so it seems that the precision converges to some stable level.

```
grid.arrange(success_plots[["mu"]][["hpdi_table"]][[1]],
             success_plots[["mu"]][["quan_table"]][[1]],ncol=2)
```

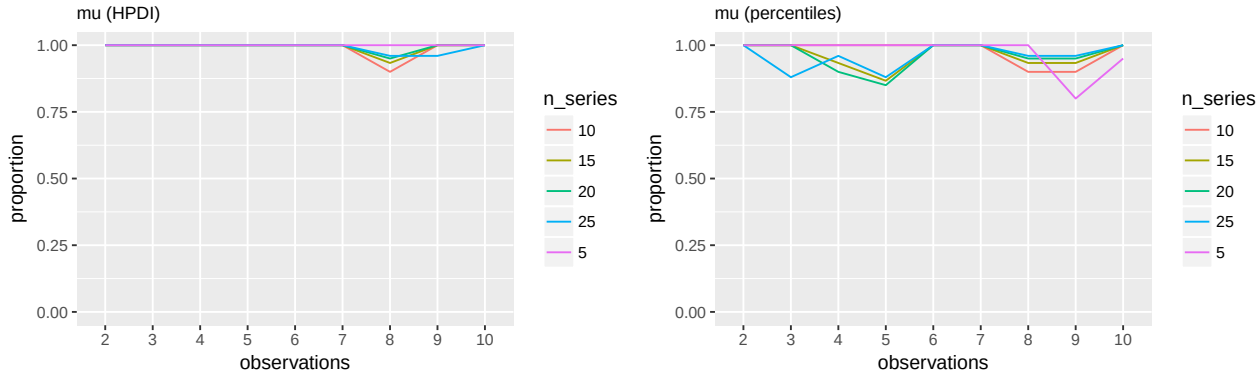

Fig. 3: Proportion of successful mu estimates against number of time points. Different colors denote number of series. HPDI based rates on the left.

```
grid.arrange(success_plots[["mu"]][["hpdi_table"]][[2]],
             success_plots[["mu"]][["quan_table"]][[2]],ncol=2)
```

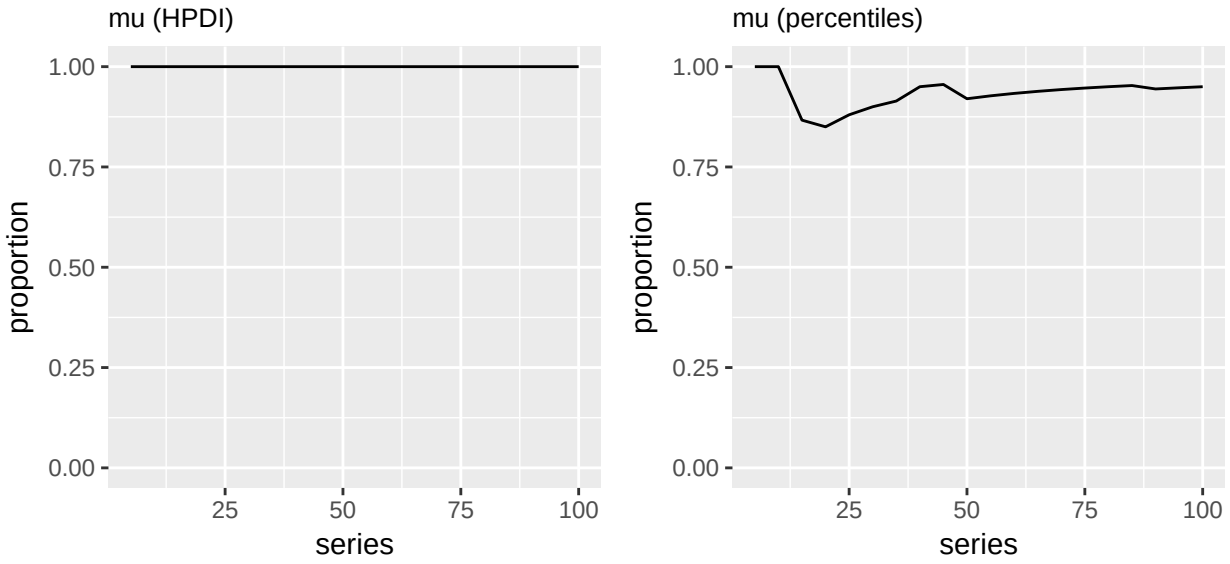

Fig. 4: Proportion of successful mu estimates plotted against number of series. Number of observations is fixed at 5. HPDI based rates on the left.

```
grid.arrange(success_plots[["kappa"]][["hpdi_table"]][[1]],
             success_plots[["kappa"]][["quan_table"]][[1]],ncol=2)
```

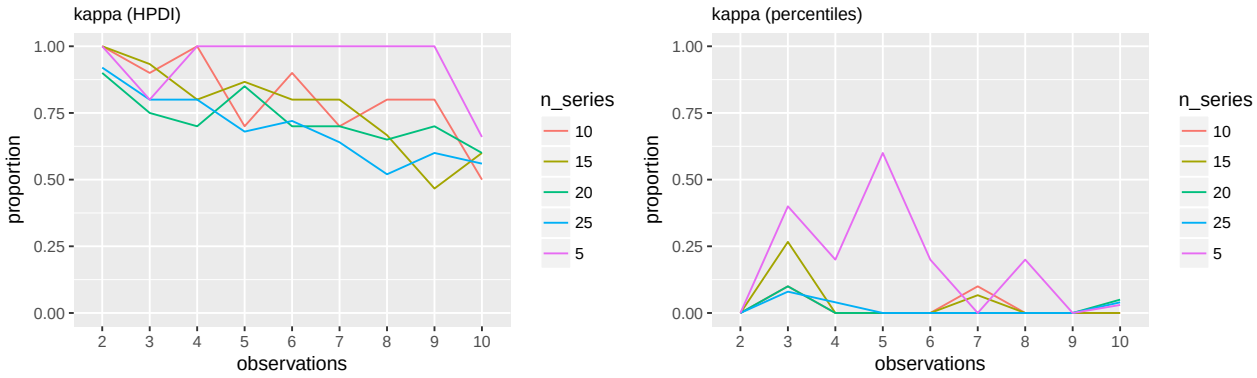

Fig. 5: Proportion of successful kappa estimates for kappa. HPDI based rates on the left.Different colors denote number of series. HPDI based rates on the left.

```
grid.arrange(success_plots[["kappa"]][["hpdi_table"]][[2]],
             success_plots[["kappa"]][["quan_table"]][[2]],ncol=2)
```

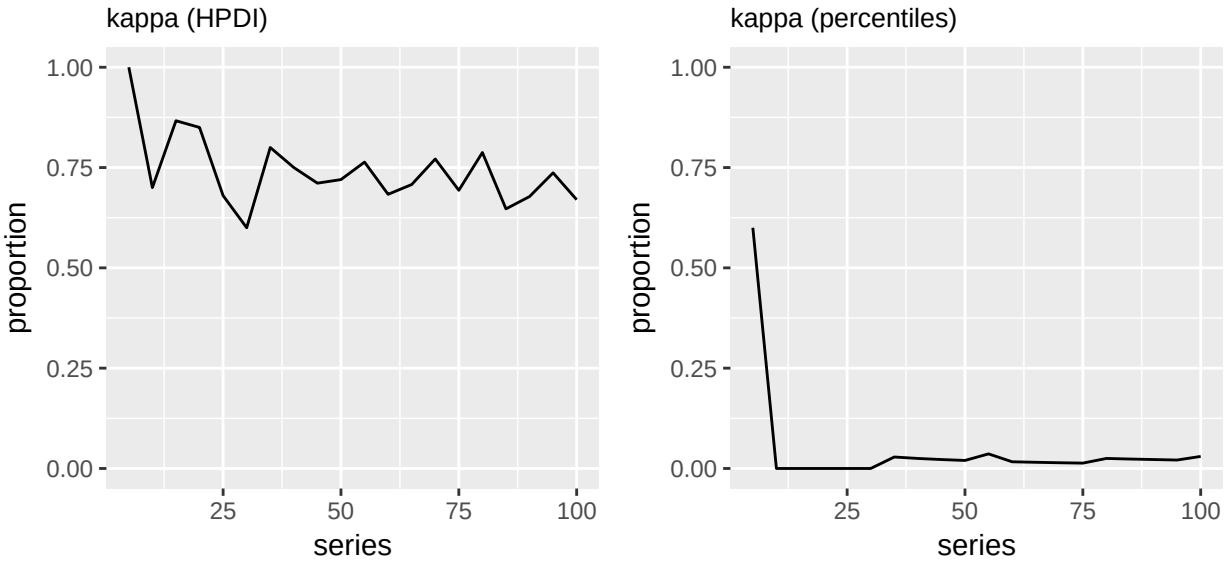

Fig. 2: Proportion of successful kappa estimates plotted against number of series. Number of observations is fixed at 5. HPDI based rates on the left.

Running time

The sampling times presented below, seem to grow in $O(n)$ time.

```
running_time_long
```

	minutes	n_series
5	1	5
10	2	10
15	3	15
20	4	20
25	6	25
30	8	30
35	9	35
40	10	40
45	13	45
50	15	50
55	18	55
60	20	60
65	20	65
70	22	70
75	24	75
80	27	80
85	30	85
90	32	90
95	35	95
100	37	100

Table 1: Sampling times in minutes for time series with sample number of observation 5 and n_series denoting the number of series per data set.

```
# total sampling time in hours for data long data set.
round(sum(running_time[,"5"])/(60*60), 1)
```

```
## [1] 5.6
```

```
# sampling times for 5 -> 25 series, 2 -> 10 observations
running_time_short <- data.frame(running_time[1:5,])/60 %>% round()
colnames(running_time_short) <- c(2:10)
running_time_short %>% kable(digits=2)
```

	2	3	4	5	6	7	8	9	10
5	1.12	1.27	1.68	1.30	2.10	3.07	2.57	1.48	39.32
10	2.08	3.00	2.38	2.18	3.65	5.72	4.57	4.05	10.42
15	3.03	3.87	3.63	3.35	6.05	7.63	7.40	5.88	15.67
20	3.63	4.95	5.55	4.27	8.80	11.32	10.57	8.35	20.17
25	4.18	6.58	8.33	6.12	11.77	14.28	15.50	12.25	25.92

Table 2: Sampling times in seconds for data sets of different length (columns) and number of time series (rows). Total running time for the entire set was ~ 3h 10min

```
# Total running time in hours for 5 -> 25 series, 2 -> 10 observations
round(sum(running_time[1:5,])/(60*60), 1)
```

```
## [1] 5.5
```

```
#total running time for all data in hours
round(sum(running_time, na.rm = TRUE)/(60*60), 1)
```

```
## [1] 10.8
```

Discussion

The main objective of this work was to extend a previously propsed implementation of the Ornstein-Uhlenbeck driven Student-t process in Stan by adding a new level of hierarchy to the model. By partial pooling, the parameter inference could utilize information across multiple time series. In terms of the defined metrics the model performance was satisfactory. The parameter characterizing the stochastic fluctuations, κ was an exception, as the precision of estimates decreased as more data was available.

The tests of robustness executed here provide only preliminary results of the model capabilities. For a complete picture an extensive probing of different parameter ranges and priors should be undertaken. Alternative parameterizations should be tested to see if some perform better with higher sample sizes. As the OU process can be considered an extensions of the autoregressive AR(1) model with arbitrary time intervals, we are planning to develop this work further by enabling the time series to have unequal amounts of observations and unequal measurement times. This would allow the analysis of time series with missing values and eliminate the need for interpolation and forcing even observation times.

The HIT Chip Atlas data set [@TippingElements], consists of stool samples from 1006 Western individuals with multiple (2-5) time points for 78 subjects, and provides an interesting opportunity to test the model on real data. This has motivated the work presented here, and we are next planning apply our validated implementation on this real case study of the human gut microbiome analysis.

Licences

Code and text © 2018, Ville Laitinen & Leo Lahti, licensed under CC BY 4.0.

Bibliography