Journal of Statistical Software

August 2016, Volume 72, Issue 2. doi: 10.18637/jss.v072.i02

Bayesian Nonparametric Mixture Estimation for Time-Indexed Functional Data in R

Terrance D. Savitsky
U.S. Bureau of Labor Statistics

Abstract

We present **growfunctions** for R that offers Bayesian nonparametric estimation models for analysis of dependent, noisy time series data indexed by a collection of domains. This data structure arises from combining periodically published government survey statistics, such as are reported in the Current Population Study (CPS). The CPS publishes monthly, by-state estimates of employment levels, where each state expresses a noisy time series. Published state-level estimates from the CPS are composed from household survey responses in a model-free manner and express high levels of volatility due to insufficient sample sizes. Existing software solutions borrow information over a modeled time-based dependence to extract a de-noised time series for each domain. These solutions, however, ignore the dependence among the domains that may be additionally leveraged to improve estimation efficiency. The **growfunctions** package offers two fully nonparametric mixture models that simultaneously estimate both a time and domain-indexed dependence structure for a collection of time series: (1) A Gaussian process (GP) construction, which is parameterized through the covariance matrix, estimates a latent function for each domain. The covariance parameters of the latent functions are indexed by domain under a Dirichlet process prior that permits estimation of the dependence among functions across the domains: (2) An intrinsic Gaussian Markov random field prior construction provides an alternative to the GP that expresses different computation and estimation properties. In addition to performing denoised estimation of latent functions from published *domain* estimates, **growfunctions** allows estimation of collections of functions for *observation units* (e.g., households), rather than aggregated domains, by accounting for an informative sampling design under which the probabilities for inclusion of observation units are related to the response variable. **growfunctions** includes plot functions that allow visual assessments of the fit performance and dependence structure of the estimated functions. Computational efficiency is achieved by performing the sampling for estimation functions using compiled C++.

Keywords: Gaussian process, Gaussian Markov random field, Dirichlet process, Bayesian hierarchical models, time series, functional data, R, C++.

1. Introduction

1.1. Motivation

Data sets formed from collections of noisy time series for a set of observation units are commonly produced by combining published estimates, across time, drawn from U.S. Federal government surveys. These surveys are administered periodically and report statistics derived from participants (e.g., households or business establishments) aggregated (using sampling weights) to a set of domains (e.g., geographical regions, such as states, or industry categorizations for business establishments). In addition to a time-indexed dependence that one expects among repeated measures for each domain, there is also often an embedded domain-indexed dependence. We are interested to employ a modeling approach to extract a denoised collection of functions for the domains that simultaneously accounts for the both time and domain-based dependencies. Incorporating a domain-based dependence not only improves the efficiency of the extracted functions (by reducing their estimated posterior variances), but also provides inferential context based on discovered similarities across domains.

Our motivating Current Population Survey (CPS) publishes monthly estimates of employment levels for the U.S. states. State policymakers seek estimates that are free of noise-induced volatility that is not reflective of their underlying economic conditions. States additionally desire context around causes for the changes in their employment levels and rates. Subsets of states may share similarities in the structures of their workforces and economies that may produce similar trends in their employment level time series. So our twin goal is to account for the dependencies across both time and state to better detect and remove noise and to identify subsets of states that express similar patterns (e.g., trends and seasonality) in their time series. These patterns provide context, by relative comparison to other states, for state-level policymakers.

Survey administrators assure the quality of reported domain-level estimates by auditing the accuracy of responses provided by observation units nested within the domains. The accuracy is often assessed by comparing responses to the same question asked of the same set of observation units over multiple survey or census instruments. Differences are recorded for each time period and observation unit and produce a data structure very similar to published domain-level survey estimates in the form of a collection of time series, though here the time series are indexed by observation unit, rather than the coarser domain. The Quarterly Census of Employment and Wages (QCEW) and the Current Establishment Survey (CES) are both administered to business establishments and record number of employees. The Bureau of Labor Statistics (BLS) is able to record errors in reported employment levels for those establishments that are included in the CES (since all establishments participate in the QCEW). We are interested to model the dependence structure among the set of by-establishment denoised functions, estimated from the absolute difference in CES and QCEW, so that we may uncover unique patterns expressed among subsets or groupings of the establishments.

1.2. Scope of growfunctions

The **growfunctions** for R (R Core Team 2016) performs Bayesian estimation of dependent denoised functions from a collection of noisy time series (indexed by domain or observation unit). **growfunctions** includes two Bayesian estimation models that provide nonparametric

formulations for estimating denoised, latent functions from the observed, noisy time series. Both models simultaneously estimate a domain-induced dependence structure among the latent functions by employing a Dirichlet process mixture construction over set of covariance or precision parameters, which are indexed by domain. The first model specifies a Gaussian process (GP) formulation for the latent functions that provides a fully nonparametric estimation of the shape, trend and frequency (length scale) of each time-indexed function. The GP is parameterized through the covariance matrix, where these parameters are indexed by domain, so that each domain specifies its own GP. These domain-indexed functions are tied together by placing a prior distribution on the covariance parameters, indexed by domain, where the prior distribution is drawn from a Dirichlet process (DP). This formulation induces a fully nonparametric scale mixture (over the domain-indexed covariance parameters). The second model specifies an intrinsic Gaussian Markov random field (iGMRF) prior for the latent functions under a fixed length scale, in lieu of the GP, that behaves like a probabilistic local smooth. The iGMRF is parameterized through a single precision parameter (rather than a set of covariance parameters, like the GP), and we similarly induce a nonparametric mixture over domains by indexing the precision by domain under a Dirichlet process formulation. The GP and iGMRF models express different estimation and computation properties that we will later explore.

The two estimation functions of **growfunctions** also allow the modeling of unit-level data, which is needed for our analysis of the establishment-level time series of employment level reporting differences between the CES and QCEW. Performing modeling at the establishment (unit) level, instead of the higher domain-level, requires incorporation of the sampling design for unbiased estimation, which **growfunctions** facilitates through input of the establishment sample inclusion probabilities (which are formed into sampling weights).

Further, **growfunctions** also includes a suite of plot functions that input objects returned by the two estimation models to facilitate visual fit assessments and to explore the posterior clustering (of covariance or precision parameters) among the latent functions permitted by the DP mixture prior. Package **growfunctions** is available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=growfunctions`.

GP models are very popular among practitioners for application to time-indexed data because they are highly flexible and are able to estimate the length-scale (that governs the attenuation of the dependence structure) from the data and the estimated functions may be constrained to various degrees of smoothness (e.g., orders of differentiability) based on the form chosen for the covariance formula. There are two well-known software tools that are commonly-used to estimate a latent GP function from a single time series. The first is the **GPstuff** software for MATLAB (The MathWorks, Inc. 2014), published by Vanhatalo, Riihimäki, Hartikainen, Jylänki, Tolvanen, and Vehtari (2013). Estimating the full posterior distribution for a GP function is particularly computationally intensive and **GPstuff** includes lower-dimensional approximations and point estimation algorithms that both reduce computational intensity. For R users, the **tgp** package (Gramacy 2007; Gramacy and Taddy 2010) will estimate the full posterior distributions of a Gaussian process prior for a single time series under a variety of covariance formulas. Employment of regression trees allows for discontinuous change points in the estimated latent function. Finally, the **CARBayes** of Lee (2013) allows estimation of a latent function under a Gaussian Markov random field prior for spatially-indexed data (that may be adapted for time-indexed data). What all of these applications lack is the modeling of dependence among a collection of functions induced by domain (or observation unit). It would

be possible, in theory, to vectorize a collection of functions and use **tgp** for estimation because it will capture non-smooth shifts between domain-indexed functions, but the estimation would be computationally intractable, in practice, and the inference among domains would be far more difficult. Package **growfunctions** simultaneously estimates the latent GP or iGMRF functions and a generalized domain-induced dependence structure between them, which is a more natural and inferentially useful approach.

We outline our motivating data sets in Section 2 to set context for the dependent formulations we next introduce in Section 3 and Section 4, for the GP and iGMRF models, respectively, that estimate a collection of dependent, latent functions. The **growfunctions** estimation models and associated plotting tools are introduced and illustrated in the sections that specify the model formulations. Having enumerated and illustrated each model, Section 5 demonstrates how **growfunctions** may be used to compare the performances between alternate model constructions. This section also highlights a feature of **growfunctions** that allows specification of multiple covariance matrices, in an additive fashion, each under a variety of covariance formulas, to allow estimation of more complex functions. We move from the domain-level to the observation unit (within domain) level in Section 6, where we employ the estimation functions of **growfunctions** and input sample inclusion probabilities for the observation units in order to account for an informative sampling design to produce unbiased parameter estimates. We offer concluding remarks in Section 7.

2. Case study data set

Our data are composed for of $T = 158$ months of direct estimates of employment levels for each of $N = 51$ states (including the District of Columbia) published by the BLS in the Current Population Survey. The employment level statistics are influenced by the underlying economic conditions of the states. We expect a correlation of industrial, service and agricultural mixtures among states, on the one hand, with patterns and trends expressed in their employment level time series, on the other hand. The time series of T months for each state, $i \in (1, \ldots, N)$, is standardized to mean 0 and variance 1 to allow comparability of the trends across states in our model formulations to follow.

We denote the set of $T \times N$ matrix of standardized survey direct estimates for $N = 51$ states at $T = 158$ months with $\{y_{ij}\}_{i=1,\ldots,N;\ j=1,\ldots,T}$. We would like to estimate a collection of N, $T \times 1$ de-noised, smooth functions, $\{\mathbf{f}_i\}$, from the $\{\mathbf{y}_i = (y_{i1}, \ldots, y_{iT})\}$. We estimate the dependence among the collection of state employment levels through a prior construction that permits a grouping or clustering of covariance (under a Gaussian process (GP) prior specification) or precision parameters (under an intrinsic Gaussian Markov random field (iGMRF) prior specification), that we index by state. These state-indexed covariance or precision parameters are used to generate the latent $\{\mathbf{f}_i\}$.

3. Dependent Gaussian processes

We begin by introducing the GP formulation of Savitsky (2015) in the simpler case where all functions are drawn from a GP prior with a single set of covariance parameters for all domains, rather than indexing the covariance parameters by domain. We then demonstrate how we may extend this formulation to allow for subgroups, denoted as clusters, of domains

where each cluster shares its own covariance parameters among members.

A Gaussian process (GP) model is parameterized through the covariance formula of a multivariate Gaussian prior on the $\{\mathbf{f}_i\}$ in the following probability model:

$$\overset{T\times 1}{\mathbf{y}_i}|\mathbf{f}_i, \tau_\epsilon \overset{\text{ind}}{\sim} \mathcal{N}_T\left(\mathbf{f}_i, \tau_\epsilon^{-1}\mathbb{I}_T\right),\ i = 1, \ldots, N \tag{1a}$$

$$\mathbf{f}_i|\boldsymbol{\theta}_i \overset{\text{iid}}{\sim} \mathcal{N}_T\left(\mathbf{0}, \overset{T\times T}{\mathbf{C}}(\boldsymbol{\theta}_i)\right) \tag{1b}$$

$$\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_P|a, b \overset{\text{iid}}{\sim} \prod_{p=1}^{P} \mathcal{G}a(a, b) \tag{1c}$$

$$\tau_\epsilon|c, d \overset{\text{iid}}{\sim} \mathcal{G}a(c, d), \tag{1d}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_P)$. We model a continuous likelihood for each domain, i, even though the CPS application data are in the form of standardized by-domain employment levels. The high magnitude for each y_{ij} makes this approximation reasonable (in lieu of specifying an inhomogenous Poisson process with mean, λ_{ij}, since λ_{ij} would be relatively large). Such approximations are common for modeling domain-based estimates of levels typically published for government surveys, because the estimate for each domain is aggregated over a large number of respondents (Maples, Bell, and Huang 2009; Nugent and Hawala 2012). The GP model is parameterized through the covariance matrix, $\mathbf{C}(\boldsymbol{\theta})$, with covariance formula,

$$\begin{aligned} \mathbf{C}(\boldsymbol{\theta}) &= \left(C_{f_j, f_\ell}\right)_{j,\ell\in(1,\ldots,T)} \\ C_{f_j, f_\ell} &= \frac{1}{\theta_1}\exp\left(\frac{(t_j - t_\ell)^2}{\theta_2}\right), \end{aligned}$$

where $P = 2$. This simple covariance formula is known as the squared exponential and is parameterized by $\boldsymbol{\theta} = (\theta_1, \theta_2)$, where θ_1 controls the vertical variance or magnitude of the $\boldsymbol{f}$ drawn from the GP under $\mathbf{C}(\boldsymbol{\theta})$ and θ_2 controls the length scale (wavelength) or level of roughness expressed by $\boldsymbol{f}$. We see from Equation 1 that $\boldsymbol{\theta}$ are estimated from the data (through updating the prior to the posterior), so that the data are able to influence the properties of the functions, $\{\mathbf{f}_i\}$, through the posterior distribution for $\boldsymbol{\theta}$. The squared exponential covariance matrix produces functions that are constrained to by infinitely smooth or differentiable at all orders. This feature helps prevent overfitting and the differentiation of the smooth signal from rough (non-differentiable) noise. Parameter, τ_ϵ, is the overall model noise precision (inverse of the variance) and receives a Gamma prior that is updated by the data.

We also introduce a slightly more complicated (and more heavily parameterized) covariance formula,

$$\begin{aligned} \mathbf{C}(\boldsymbol{\theta}) &= \left(C_{f_j, f_\ell}\right)_{j,\ell\in(1,\ldots,T)} \\ C_{f_j, f_\ell} &= \frac{1}{\theta_1}\left(1 + \frac{(t_j - t_\ell)^2}{\theta_2\theta_3}\right)^{-\theta_3}, \end{aligned}$$

where $P = 3$. This covariance formula is known as the rational quadratic, which may be formulated as a scale mixture of more commonly-used squared exponential kernels,

$1/\theta_1 \exp\left((t_j - t_\ell)^2/\theta\right)$, with the inverse of the length scale parameter, θ^{-1}, which controls function periodicity, distributed under a Gamma distribution with hyperparameters $\left(\theta_3, \theta_2^{-1}\right)$ (Rasmusen and Williams 2006). The vertical magnitude is directly controlled by θ_1, while θ_2 controls the mean length scale or period, and θ_3 controls smooth deviations from θ_2. As $\theta_3 \uparrow \infty$, this formulation converges to a single squared exponential formula with length-scale, θ_2. While we will see in the sequel that the user may select either the squared exponential or rational quadratic covariance formulations in **growfunctions**, we use the latter as a default because it provides a parsimonious specification for parameterizing the use of a single covariance matrix. It also produces surfaces, $\{\mathbf{f}_i\}$, that are differentiable at all orders, retaining the useful properties of the squared exponential. The structure of Equation 1 is contained in $\mathbf{f}_i$, so that the $\{\mathbf{y}_i\}$ are assumed to conditionally-independent, given $\{\mathbf{f}_i\}$.

3.1. Accounting for dependence among functions

We introduce an extension of Equation 1, which indexes the GP covariance function parameters, $\{\boldsymbol{\theta}_i\} = \{(\theta_{i1}, \ldots, \theta_{iP})\}$, by domain $i \in (1, \ldots, N)$, to permit their probabilistic clustering with,

$$\mathbf{f}_i|\boldsymbol{\theta}_i \overset{\text{ind}}{\sim} \mathcal{N}_T\left(\mathbf{0}, \mathbf{C}\left(\boldsymbol{\theta}_i\right)\right) \tag{2a}$$

$$\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_N|G \overset{\text{iid}}{\sim} G \tag{2b}$$

$$G|G_0 \sim \text{DP}(\alpha, G_0), \tag{2c}$$

where $\{\boldsymbol{\theta}_i\}_{i=1,\ldots,N}$ receive a random distribution prior, G, drawn from a Dirichlet process (DP), specified with a concentration parameter, α, a precision parameter that controls the amount of variation in G around prior mean, G_0. The base or mean distribution, G_0, is usually constructed as parametric; in our case, $G_0 = \mathcal{G}a\left(a, b\right)$, which is typically the same as the prior used for the global GP model of Equation 1 parameterizing a single vector, $\boldsymbol{\theta}$. Equation 2 describes a mixture model of the form, $\mathbf{f}|G \overset{\text{iid}}{\sim} \int \mathcal{N}_T\left(\mathbf{0}, \mathbf{C}\left(\boldsymbol{\theta}\right)\right) G\left(d\boldsymbol{\theta}\right)$, where G is the mixing measure over the $P \times 1$ covariance parameters, $\boldsymbol{\theta}$ (for each domain).

The DP formulation may be described as approximating any unknown distribution by placing spikes at a countably infinite set of "location" values in the support of G with heights equal to density values associated to the locations, such that draws from G are almost surely discrete. The discrete construction for G allows for ties among the $\{\boldsymbol{\theta}_i\}$ that we interpret as probabilistic clusters. We examine this clustering property of the DP by expressing it in a stick breaking form of Sethuraman (1994),

$$G = \sum_{h=1}^{\infty} p_h \delta_{\boldsymbol{\theta}_h^*}, \tag{3}$$

a countably infinite mixture of weighted point masses, where "locations", $\boldsymbol{\theta}_1^*, \ldots, \boldsymbol{\theta}_M^*$, index the *unique* values for the $\{\boldsymbol{\theta}_i\}$ (where M denotes the number of unique location values). The maximum number of clusters would assign each observation to its own cluster. We record domain cluster memberships with $\mathbf{s} = (s_1, \ldots, s_N)$ where $s_i = \ell$ denotes $\boldsymbol{\theta}_i = \boldsymbol{\theta}_\ell^*$ so that $(\mathbf{s}, \{\boldsymbol{\theta}_m^*\})$ provides an equivalent parameterization to $\{\boldsymbol{\theta}_i\}$ and we recover $\boldsymbol{\theta}_i = \boldsymbol{\theta}_{s_i}^*$. We conduct posterior sampling with the cluster locations and assignments, rather than directly sampling $\{\boldsymbol{\theta}_i\}$, as the former produces notably better mixing because it separates re-assignments to clusters from updates to the location values. The weight, $p_h \in (0, 1)$, is composed as

$p_h = v_h \prod_{k=1}^{h-1} (1 - v_k)$ where v_h is drawn from the beta distribution, $\mathcal{B}e\,(1, \alpha)$. This construction provides a prior penalty on the number of mixture components, but we also see that a higher value for α will produce more clusters (unique locations). We place a further prior, $\alpha \sim \mathcal{G}\,(1, 1)$, to allow posterior updating in recognition of the relatively strong influence it conveys on the number of clusters formed (Escobar and West 1995).

We re-state Equation 2 under the parameterization, $(\mathbf{s}, \{\boldsymbol{\theta}^*\}_{m=1,\ldots,M})$, that we will use for producing draws from the joint posterior distribution, after marginalizing over the random measure, G, using the Pólya urn scheme of Blackwell and MacQueen (1973),

$$\mathbf{f}_i | s_i, \boldsymbol{\theta}^*_{s_i} \overset{\text{ind}}{\sim} \mathcal{N}_T \left(\mathbf{0}, \mathbf{C} \left(\boldsymbol{\theta}^*_{s_i} \right) \right) \tag{4a}$$

$$s_i | \mathbf{s}_{-i}, \alpha \sim \frac{1}{N - 1 + c} \sum_{j=1, j \neq i}^{N} \delta_{s_j}(s_i) + \frac{\alpha}{n - 1 + \alpha} \, G_0 \tag{4b}$$

$$\boldsymbol{\theta}^*_1, \ldots, \boldsymbol{\theta}^*_M | G_0 \overset{\text{iid}}{\sim} G_0 \equiv \prod_{p=1}^{p} \mathcal{G}a\,(a, b)\,. \tag{4c}$$

3.2. Computation

The Gaussian process formulation of Equation 2 is very computationally-intensive because computing the cholesky decomposition of the $T \times T$ covariance matrix is $\mathcal{O}(T^3)$. We adapt two algorithms to enhance chain mixing that increase the effective sample size and reduce the required number of posterior sampling iterations, which reduces the computation time. The first algorithm addresses the sampling of cluster locations, $\boldsymbol{\Theta}^* = \{\theta^*_{pm}\}_{p=1,\ldots,P;\ m=1,\ldots,M}$, and the other is used to sample cluster assignments, $\mathbf{s} = (s_1, \ldots, s_N)$. We recall that sampling $(\mathbf{s}, \{\boldsymbol{\theta}^*\})_{m=1,\ldots,M}$ are used to produce draws of $\{\boldsymbol{\theta}_i\}_{i=1,\ldots,N}$, using the relation, $\boldsymbol{\theta}_i = \boldsymbol{\theta}^*_{s_i}$, in a fashion that produces better chain mixing.

We first adapt an algorithm of Wang and Neal (2013) to our more complex, clusters-of-GPs model, that introduces a temporary stochastic space of lower-dimension in which we approximate the evaluations from the posterior kernel for sampling cluster locations, $\boldsymbol{\Theta}^*$. The kernel approximation uses successively smaller subsets of the T time points to build approximations to the $T \times T$ covariance matrix, $\mathbf{C}$. A Metropolis-Hastings proposal is formed by successive moves in this temporary space, but is accepted with respect to the *exact* posterior distribution. If the approximations are reasonably good, this sampling algorithm reduces the computation time per effective sample size as compared to drawing proposals from the full-dimensional space. We then adapt an algorithm of Neal (2000b) to sample cluster locations, $\mathbf{s}$, under our non-conjugate formulation in a fashion that produces chain mixing nearly as good as a conjugate Gibbs sampler used for simpler models. We proceed to provide a sketch of the posterior sampling schemes from their full conditional distributions.

1. Sample cluster locations, $\{\theta^*_{pm}\}_{p=1,\ldots,P;\ m=1,\ldots,M}$, (where p denotes parameter type (e.g., $(\theta_1, \theta_2, \theta_3)$) and $m = 1, \ldots, M$ denotes the cluster): We sample the posterior distribution for locations in by-cluster groups, $\{\theta^*_{pm}\}_{p=1,\ldots,P}$, from the subset of domains (states) assigned to that cluster because $\theta^*_{pm} \perp \theta^*_{pm'}$ for $m' \neq m$, *a posteriori*, in a Metropolis-

Hastings scheme using the following log-posterior kernel,

$$\begin{aligned}&\log \pi\left(\theta^*_{pm}|\boldsymbol{\theta}^*_{-pm}, \mathbf{s}, \tau_\epsilon, \{\mathbf{y}_i : s_i = m\}\right)\\ &\propto -\frac{1}{2} n_m \log\left(|\mathbf{C}^\tau\left(\boldsymbol{\theta}^*_m\right)|\right) - \frac{1}{2}\sum_{i\in s_m} \mathbf{y}_i^\top \mathbf{C}^\tau\left(\boldsymbol{\theta}^*_m\right)\mathbf{y}_i + (a-1)\log(\theta^*_{pm}) - b\theta^*_{pm}, \quad (5)\end{aligned}$$

where $\mathbf{C}^\tau\left(\boldsymbol{\theta}^*_m\right) = \mathbf{C}\left(\boldsymbol{\theta}^*_m\right) + (1/\tau_\epsilon)\,\mathbb{I}_T$, s_m collects the domains assigned to cluster m, n_m denotes the number of domains assigned to cluster m, and (a, b) are shape and rate hyperparameters of a gamma prior, respectively. This posterior representation is a relatively straightforward Gaussian kernel of a non-conjugate probability model.

Starting with the previously sampled value, $\hat{x}_0$, (for each θ_{pm}) from the full-dimensional or exact space, the sampling algorithm builds a sequence of proposed transitions by first "stepping up" in the temporary space using increasingly coarse, "tempered" transitions, $\left(\hat{Z}_1, \hat{Z}_2, \ldots, \hat{Z}_n\right)$, that generate computationally-fast approximations for $\mathbf{C}$. These coarse approximations use fewer observations in each step; for example, we apply $n = 2$ transitions in the lower-dimensional space and use 80 of the $T = 158$ time points to formulate $\hat{Z}_1$ and then 40 time points for $\hat{Z}_2$. This method is motivated by tempering, where each successive distribution is broader (which is used to discover multiple modes). The motivation of Wang and Neal (2013), however, is reduced computation. So, instead of defining each successive distribution in the temporary space to be broader, each step is defined to use fewer time points, such that the computation is faster. This sequence of coarser transitions is followed by transition steps that employ progressively finer distributions (in reverse order), $\left(\check{Z}_2, \check{Z}_1\right)$ that guide the chain back towards the full dimensional space until we conclude the sequence by proposing $\check{x}_0$ from $\check{Z}_1$ to be evaluated in the full-dimensional space. We use univariate slice sampling of Neal (2000a) to accomplish these (reversible) transitions in the lower-dimensional space. The proposal (reversible sequence of steps) is then accepted with (for $n = 2$ tempered transitions),

$$\min\left(\frac{\pi_1(\hat{x}_0)\pi_2(\hat{x}_1)\pi_1(\check{x}_1)}{\pi_1(\hat{x}_1)\pi_2(\check{x}_1)\pi_1(\check{x}_0)}\frac{\pi(\check{x}_0)}{\pi(\hat{x}_0)}\right),$$

where "x" denotes the proposals, and "π" posterior kernel evaluations, for each θ^*_{pm} with subscript 0 pertaining to the exact space and $(1, 2)$ to the successively coarser transition distributions in the temporary space in the sequential order of application. If our lower dimensional approximations are relatively good, this approach will speed chain convergence by producing draws of lower autocorrelation since each proposal includes a sequence of moves generated in the temporary space. The number of time points employed in each, successively decreasing, approximation step is typically between $10\% - 50\%$, where the higher end of the range is required if the true $T \times 1$ function is "rough" or has a small length scale. If the values are too small, the approximation will be poor and the resulting proposals are likely to be rejected at a higher rate, which would not much improve the computation time per effective sample size. We have used trial and error on multiple data sets, including our CPS application, and found that $35 - 50\%$ for the first coarse step and $15 - 25\%$ for the second coarse step generally produces a robust improvement in computation time per effective sample size. More attention to tuning these settings may offer further computational benefits. This algorithm, though employing a fast-computing temporary space, produces samples from

the *exact* posterior distribution. Our adaptation configures Wang and Neal (2013) to apply to clusters of Gaussian processes (rather than to just a single GP) by exploiting the between cluster posterior independence of the $\{\theta^*_{pm}\}$. Wang and Neal (2013) note that maximum improvements in sampling efficiency are achieved with $n = 2$ (rather than employing additional coarse steps) and our simulation studies find the same.

2. Sample cluster assignments, $\mathbf{s} = (s_1, \ldots, s_N)$: We marginalize over G in Equation 2, that results in the Pólya urn representation of Blackwell and MacQueen (1973), to sample $\mathbf{s}$ from its full conditionals,

$$\pi\left(s_i = s|\mathbf{s}_{-i}, \boldsymbol{\Theta}^*, \alpha, \tau_\epsilon, \mathbf{y}_i\right) \propto \begin{cases} \frac{n_{-i,s}}{n-1+\alpha}\mathcal{N}_T\left(\mathbf{y}_i|\mathbf{0}, \mathbf{C}^\tau\left(\boldsymbol{\theta}^*_s\right)\right) & \text{if } 1 \leq s \leq M^- \\ \frac{\alpha/c^*}{n-1+\alpha}\mathcal{N}_T\left(\mathbf{y}_i|\mathbf{0}, \mathbf{C}^\tau\left(\boldsymbol{\theta}^*_s\right)\right) & \text{if } s = M^- + h, \end{cases} \tag{6}$$

where $n_{-i,s} = \sum_{i' \neq i} \mathbf{1}(s(i') = s)$ is the number of sample observations, excluding domain i, assigned to cluster s, so that domains are assigned to an existing cluster with probability proportional to its "popularity". M^- is defined to be the total number of *unique* location values after deleting domain, i (which will be equal to $M - 1$ if i is assigned to a singleton cluster; otherwise, it will be equal to M). The posterior assigns a domain (through s_i) to a new cluster with probability proportional to αd_0 under the mixture prior in the case of a conjugate formulation. The conjugate specification requires the likelihood to be integrable in closed form with respect to the base distribution, G_0, to compute $d_0 = \int \mathcal{N}\left(\mathbf{y}|\boldsymbol{\theta}, \ldots\right) G_0(d\boldsymbol{\theta})$, which is not the case under a (nonconjugate) GP construction. So we employ the auxiliary Gibbs sampler formulation of Neal (2000b) and sample $c^* \in \mathbb{N}$ locations from base distribution, G_0, *ahead* of any assigned observations, to define $h = M^- + c^*$ candidate clusters in an augmented space. We then draw s_i from this augmented space, where any location not assigned domains (over a set of draws for $\mathbf{s}$) is dropped. This procedure effectively performs a Monte Carlo integration of the likelihood with respect to the base distribution. See Savitsky and Vannucci (2010) for a detailed example of a DP implementation on a GP. The larger is the tuning parameter, c^*, the lower is the autocorrelation of the resulting posterior draws (though computation time increases because we are sampling with respect to more cluster locations). Good mixing is typically achieved with c^* set equal to 2 or 3 (and this is set through the `w_star` option in `gpdpgrow()` that we next introduce, where the default is `w_star = 2`). We note that the number of clusters, M, may change in this step as clusters are added and deleted.

The DP construction assumes exchangeability of the $\{\boldsymbol{\theta}_i\}$, *a priori*, given random measure, G, but the almost surely discrete construction of the DP produces estimates which are not exchangeable, *a posteriori*. The prior specification for cluster assignments, $\{s_i|\mathbf{s}_{-i}\}$, (under our re-parameterization to $(\mathbf{s}, \boldsymbol{\theta}^*)$ achieved by marginalizing over G), induces a uniform probability for co-clustering among the states. If the likelihood values for two states, j and k, $\mathcal{N}_T\left(\mathbf{y}_j|\mathbf{0}, \mathbf{C}^\tau\left(\boldsymbol{\theta}^*_s\right)\right)$ and $\mathcal{N}_T\left(\mathbf{y}_k|\mathbf{0}, \mathbf{C}^\tau\left(\boldsymbol{\theta}^*_s\right)\right)$, are both relatively high for assignment to cluster, s, due to underlying similarities in their economies or due to closeness in their geographic locations, then the posterior probability for co-clustering states j and k will be relatively high (versus uniform, *a priori*). This posterior estimation mechanism conducts *unsupervised* (probabilistic) clustering. Sharper (lower posterior variance) estimates may typically be obtained by indexing either the weights or locations in Equation 3 to include predictors in lieu of our unsupervised formulation.

3.3. Illustration using growfunctions on CPS data

We now illustrate the estimation function, `gpdpgrow()` of **growfunctions**, that draws the parameters of Equation 2 from the joint posterior distribution. We also illustrate the associated plot function, `cluster_plot(object)`, where `object <- gpdpgrow()`. This plot function extracts posterior samples from `object` and renders estimated latent functions, $\{\mathbf{f}_i\}$, compared to noisy observations, $\{\mathbf{y}_i\}$. It also produces another plot that collects the $\{\mathbf{f}_i\}$ into the set of estimated clusters that allows us to examine similarities and differences between the clusters.

We first load our `cps` data set, which contains monthly employment levels from $1990 - 2013$. We only want to focus on the more recent years of $2000 - 2013$ to explore the period of the so-called "Great Recession" and create our $(N = 51) \times (T = 158)$ response, `y_short`.

```
R> data("cps")
R> y_short <- cps$y[, (cps$yr_label %in% c(2000:2013))]
```

We invoke the estimation function for a GP model that employs a single, rational quadratic covariance matrix with the following script, where `gp_cov = "rq"` sets the covariance formula to rational quadratic (and `gp_cov = "se"` specifies the squared exponential as another alternative). We employ 10000 iterations, of which 5000 are discarded as burnin and we keep every 5^{th} iteration. Because the transition steps for sampling cluster locations, $\boldsymbol{\Theta}^*$, in the lower dimensional space uses slice sampling, we specify 2600 observations to tune the slice sampling steps, which is conservative (and typically 1000 is enough and the default setting is `n.tune = 2000`). We set `sub_size = c(80, 40)` to specify that we will employ $n = 2$ coarse distributions that approximate the 158×158 covariance matrix in the lower dimensional space. The first distribution employs 80 of the $T = 158$ time points, while the second approximated distribution employs 40. These time points are randomly selected within strata to ensure the whole time span is represented. (The user may dispense with inputs for `gp_cov` and `sub_size`, accepting the defaults of `gp_cov = "rq"` and `n = 2` coarse distributions for `sub_size` that use 35% and 18% of the number of time points, T, respectively.)

```
R> res_gp <- gpdpgrow(y = y_short, gp_cov = "rq", n.iter = 10000,
+    n.burn = 5000, n.thin = 5, n.tune = 2600, sub_size = c(80, 40))

Tuning Interation: 0
Tuning Interation: 90

⋮

Tuning Interation: 2520
Production Interation: 450

⋮

Production Interation: 9900
Finished generating estimated GP functions, bb
Your chosen GP covariance terms includes term = rq
```

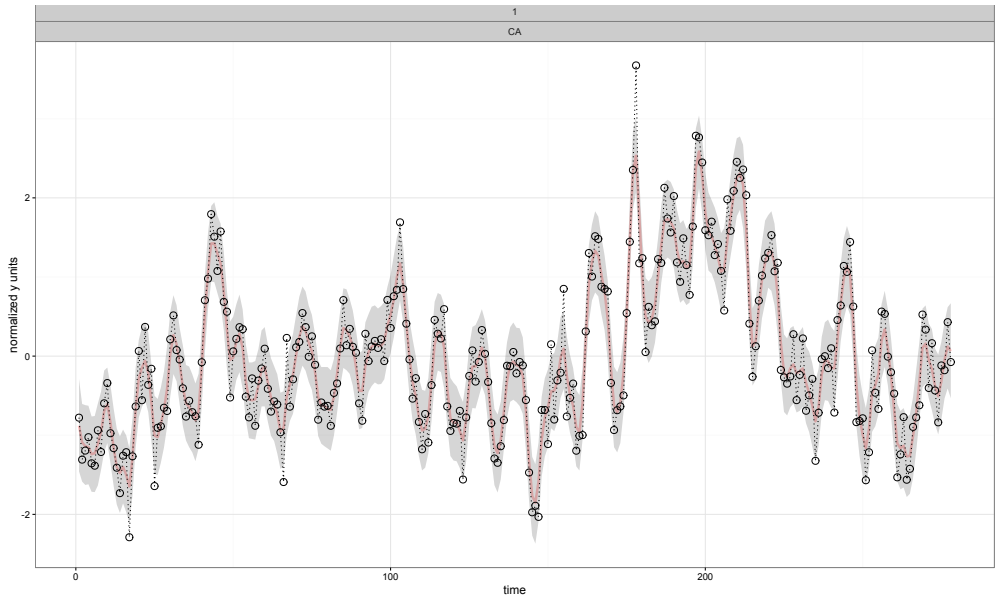

Figure 1: Estimated latent $T \times 1$ function, $\mathbf{f}_i$ (pink line), compared to observed data values, $\mathbf{y}_i$, for a single, randomly-selected domain, i.

The `gpdpgrow()` function prints progress indicators for both the tuning and production runs (which may be suppressed by setting option `progress = FALSE`) and echoes back to the user choice of covariance function when the estimation run concludes.

The plot function, `cluster_plot`, inputs the return object, `res_gp`, from `gpdpgrow()`, and produces 2 plots: the first prints a randomly-selected latent function (with a pink-colored line) against the raw, noisy data (denoted with hollow circles, connected by a dashed line); the second plot aggregates the latent functions into their assigned cluster, so that we may analyze the similarities of domain-indexed functions, within assigned cluster, and differences across the clusters.

```
R> fit_plots_gp <- cluster_plot(object = res_gp, units_name = "state",
+     units_label = cps$st, single_unit = TRUE, credible = TRUE)
R> print(fit_plots_gp$p.fit)
R> print(fit_plots_gp$p.cluster)
```

The optional `units_name` and `units_label` options provides a name (in this case "state") for the collection of domains and a label for each domain (e.g., the states are labeled with 2-digit letters). The setting, `single_unit = TRUE`, (where `FALSE` is the default setting), plots the estimated latent function against data values for a *single* randomly-selected domain. The alternative plots multiple (6) randomly-selected domains (though any integer number of domains may be randomly-selected using optional input, `num_plot`). Selecting `credible = TRUE`, highlights the 95% credible region for the function values in gray. Figure 1 presents an estimated latent function for a randomly-selected state (in this case, Vermont), by running `cluster_plot(res_gp)` with the options as above specified, where we see that the fitted function (pink line) smooths through the observed data values, which is what we expect because the rational quadratic covariance kernel (used to estimate `res_gp`) produces surfaces constrained to be infinitely differentiable.

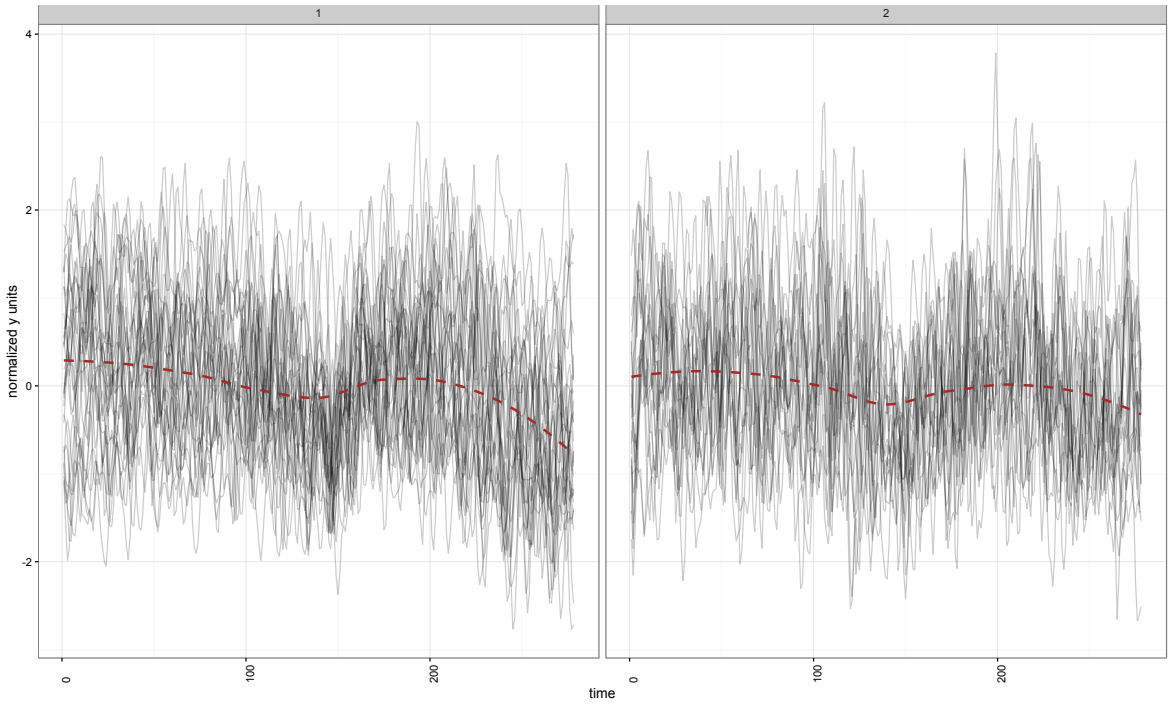

Figure 2: Collections of functions, $\{\mathbf{f}_i\}_{s_i=m}$, in plot panels indexed by cluster, $m = 1, \ldots, (M = 2)$ for the GP model under a rational quadratic covariance formula estimated using `gpdpgrow()`. A local loess smoother is drawn through the set of co-clustered functions to highlight differences between clusters.

Each Markov chain Monte Carlo (MCMC) posterior sample may select a different number of clusters and assignments of domains to those clusters (through their covariance parameters). The set of samples, taken together, represents a distribution over the space of clusterings (or partitions). We select one clustering from among the MCMC draws by using the least-squares algorithm of Dahl (2006) that builds an $N \times N$ matrix of pairwise clustering probabilities to summarize the posterior draws and selects that clustering (posterior draw) which is "closest" to this matrix under a squared Euclidean distance metric. Figure 2 prevents the clustering (including number of clusters, M, and assignment of the domains to the clusters) based on the least squares selection algorithm. The least squares assignment of domains to the clusters may be extracted with `res_gp$bigSmin`, which is a `list` object of length M. Each element contains the labels domains assigned together in each cluster. (Of course, the cluster labels, themselves, have no intrinsic meaning and are not identified).

Figure 2 presents the second plot type rendered from `cluster_plot(res_gp)`, as specified above, which aggregates the line plots of the posterior mean for the latent functions, $\{\mathbf{f}_i\}$, into plot panels indexed by cluster, for the selected least squares clustering. The states are aggregated into 2 clusters, meaning that the functions in each cluster are all generated from a Gaussian distribution with the same covariance parameters (since Equation 2 clusters covariance parameters). Each function in a cluster will not be identical, but will express similarities based on their generation from the same Gaussian distribution. A loess smoother (in red) is drawn through the functions allocated to each cluster, allowing us to see that the clusters are differentiated based on the sensitivities of state economies to the great recession

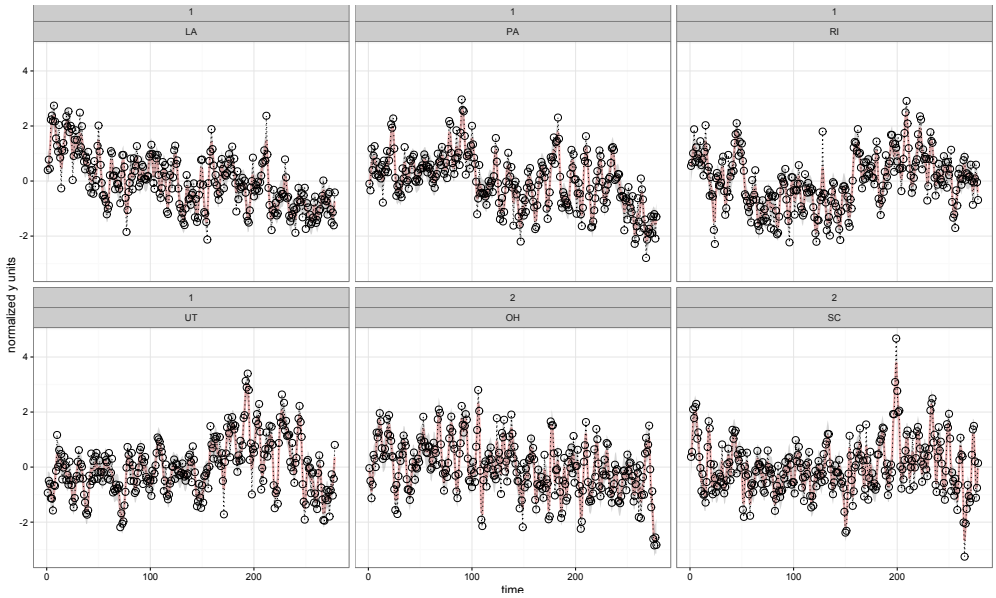

Figure 3: Each panel contains a latent $T \times 1$ function, $\mathbf{f}_i$ (pink line), compared to observed data values, $\mathbf{y}_i$, for 6 randomly-selected domains.

(that began in 2008). States allocated to the left-hand panel demonstrate a more rapid decline in employment through the great recession.

We may re-plot `cluster_plot(res_gp)` using option `single_unit = FALSE`, which will now randomly select 6 states from among the clusters and plots a panel for each with the estimated function and associated data values. (The number of randomly-selected functions may be updated using optional input, `num_plot`, where 6 is the default).

```
R> fit_plots_gp2 <- cluster_plot(object = res_gp, units_name = "state",
+    units_label = cps$st, single_unit = FALSE, credible = TRUE)
R> print(fit_plots_gp2$p.fit)
```

Figure 3 presents a plot panel for each of the 6 randomly-selected states, where each renders the function against the actual data values. Each panel is labeled with the cluster membership and then `units_label`, which is input to be two letter state abbreviations; e.g., "1,UT" denotes the state of Utah from cluster 1. This label for the cluster membership corresponds to the same label on the by-cluster plots shown in Figure 2. The states shown in the top row of panels are assigned to cluster 1, where member states show a more rapid rate of employment decline, on average, than the bottom row of panels, which represent states assigned to cluster 2.

4. Dependent Gaussian Markov random fields

We next introduce a model from Savitsky (2015) that replaces the Gaussian process formulation with an intrinsic Gaussian Markov random field prior. An iGMRF prior may be viewed as a probabilistic local smoother composed from differences in function values, which in our case, are indexed by time. A typical iGMRF specification is placed on the first difference

approximation to the first derivative, $\Delta f_{ij} = f_{i,j+1} - f_{i,j} \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \kappa^{-1}\right)$, where κ is a precision parameter that determines the (vertical) scale of variation among the first differences. This approach uses nearest neighbors defined from first differences, $(f_{i,j-1}, f_{i,j+1})$, to encode the time-indexed dependence structure among the $f_{ij},\ j = 1, \ldots, T$ for each domain, $i \in (1, \ldots, N)$. Using first differences may produce an excessively rough (though continuous) surface that overfits the data, so we prefer a prior construction based on the second difference approximation to the first derivative, $\Delta^2 f_{ij} = \Delta\left(\Delta f_{ij}\right) \overset{\text{iid}}{\sim} \mathcal{N}\left(0, \kappa^{-1}\right)$. This prior on second differences produces the joint distribution, $\mathbf{f}_i \overset{\text{iid}}{\sim} \kappa^{\frac{T-2}{2}} \exp\left(-\frac{\kappa}{2} \sum_{j=1}^{T-2} \left(f_{ij} - 2f_{i,j+1} + f_{i,j+2}^2\right)\right) = \kappa^{\frac{T-2}{2}} \exp\left(-\frac{1}{2}\mathbf{f}_i^\top \mathbf{R} \mathbf{f}_i\right)$, where $\mathbf{R} = \kappa\mathbf{Q}$ specifies a band-diagonal precision matrix with entries with $(Q_{j,j-2}, Q_{j,j-1}, Q_{j,j}, Q_{j,j+1}, Q_{j,j+2}) = (1, -4, 6, -4, 1)$ for $2 < j < T-2$ for non-boundary parameters (Rue and Held 2005). Under this construction, $\mathbf{R}$ is rank-deficient (of rank $T-2$) as the rows sum to 0 since it is composed from second differences, so that the joint distribution is improper; in particular, the prior for the $T \times 1$, $\mathbf{f}_i$, is invariant to the addition of any second order polynomial because the prior supplies no information on such polynomials. We may view the joint distribution as the product of a proper distribution on the space of $T-2$ differences (by employing the Moore-Penrose pseudo inverse, $\mathbf{R}^-$, and $|\mathbf{R}|$ as the product of the $T-2$ non-zero eigenvalues of $\mathbf{Q}$) and an improper, noninformative prior on the order 2 polynomials. These Gaussian Markov random field priors specified through a precision matrix have the property that $f_{ij} \perp f_{ik} | \mathbf{f}_{i,-jk}$ is equivalent to $R_{jk} = 0$, which allows for a parsimonious $\mathbf{Q}$, from which we specify a proper set of full conditionals,

$$f_{ij} | \mathbf{f}_{i,-j}, \kappa_i \sim \mathcal{N}\left(-\frac{1}{Q_{jj}} \sum_{k:k\sim j} Q_{jk} f_{ik}, \left(\kappa_i Q_{jj}\right)^{-1}\right) \tag{7a}$$

$$= \mathcal{N}\left(\frac{4}{6}\left(f_{i,j+1} + f_{i,j-1}\right) - \frac{1}{6}\left(f_{i,j+2} + f_{i,j-2}\right), \left(6\kappa\right)^{-1}\right), \tag{7b}$$

where the prior mean for each f_{ij} is composed as a weighted average of its order 2 nearest neighbors. Rue and Held (2005) refer to this construction as a random walk prior of order 2 or $RW2(\kappa)$.

4.1. Accounting for dependence among functions

An analogous extension to the GP is specified from Equation 7 with,

$$f_{ij} | \mathbf{f}_{i,-j}, \kappa_i \overset{\text{ind}}{\sim} \mathcal{N}\left(-\frac{1}{Q_{jj}} \sum_{k:k\sim j} Q_{jk} f_{ik}, \left(\kappa_i Q_{jj}\right)^{-1}\right) \tag{8a}$$

$$\kappa_1, \ldots, \kappa_N | G \overset{\text{iid}}{\sim} G \tag{8b}$$

$$G | G_0 \sim \text{DP}(\alpha, G_0), \tag{8c}$$

that may be expressed as $f_{ij} | \mathbf{f}_{i,-j}, G \overset{\text{iid}}{\sim} \int \mathcal{N}\left(-\frac{1}{Q_{jj}} \sum_{k:k\sim j} Q_{jk} f_{ik}, \left(\kappa Q_{jj}\right)^{-1}\right) G\left(d\kappa\right)$, marginalizing over $\{\kappa_i\}_{i=1,\ldots,N}$. We specify base distribution, $G_0 = \mathcal{G}a\left(c, d\right)$, the parametric prior used for the global iGMRF model of Equation 7. As with Equation 2, we sample $\boldsymbol{\kappa}$ indirectly through cluster assignments, $\mathbf{s}$, for the $N \times 1$ states, and location values $\kappa_1^*, \ldots, \kappa_M^*$.

We re-state Equation 8 under the parameterization, $(\mathbf{s}, \{\boldsymbol{\kappa}^*\}_{m=1,\ldots,M})$, that we will use for producing draws from the joint posterior distribution, after marginalizing over the random

measure, G, using the Pólya urn scheme of Blackwell and MacQueen (1973),

$$f_{ij}|\mathbf{f}_{i,-j}, s_i, \kappa^*_{s_i} \overset{\text{ind}}{\sim} \mathcal{N}\left(-\frac{1}{Q_{jj}} \sum_{k:k\sim j} Q_{jk} f_{ik}, \left(\kappa^*_{s_i} Q_{jj}\right)^{-1}\right) \tag{9a}$$

$$s_i|\mathbf{s}_{-i}, \alpha \sim \frac{1}{N-1+\alpha} \sum_{j=1, j\neq i}^{N} \delta_{s_j}(s_i) + \frac{\alpha}{n-1+\alpha} G_0 \tag{9b}$$

$$\kappa^*_1, \ldots, \kappa^*_M | G_0 \overset{\text{iid}}{\sim} G_0 \equiv \mathcal{G}a\left(a, b\right). \tag{9c}$$

4.2. Computation

Posterior sampling from the mixtures of iGMRFs employs a Gibbs scan over model parameters from the set of conjugate full conditional distributions. We highlight sampling steps for $(\{\mathbf{f}_i\}, \{\kappa^*_m\}, \mathbf{s}, \tau_\epsilon)$:

1. Sample latent functions, $\{f_{ij}\}_{i=1,\ldots,N;\ j=1,\ldots,T}$ in a Gibbs steps from the full conditionals,

$$\pi\left(f_{ij}|\{f_{ij}\}_{-j}, \tau_\epsilon, y_{ij}\right) = \mathcal{N}\left(\frac{e_{ij}}{\phi_{ij}}, \phi_{ij}^{-1}\right),$$

   where $e_{ij} = \tau_\epsilon y_{ij} + Q_{jj}\kappa^*_{s_i}\bar{f}_{ij}$, with $\bar{f}_{ij} = -\frac{1}{Q_{jj}}\sum_{k:k\sim j} Q_{jk} f_{ik}$ and $\phi_{ij} = \tau_\epsilon + Q_{jj}\kappa^*_{s_i}$.

2. Sample locations, $\{\kappa^*_m\}$, in a Gibbs step from,

$$\pi\left(\kappa^*_m|\{f_{ij}\}\right) = \mathcal{G}a\left(a_2, b_2\right),$$

   with shape parameter, $a_2 = \frac{1}{2} n_m (T - o_Q) + a$, where $o_Q = 2$ is the rank-deficiency of the precision matrix, $\mathbf{Q}$, indicating that $\mathbf{f}_i$ provides the equivalent of $T - o_Q$ degrees of freedom, rather than number of time points, T. The rate parameter, $b_2 = \frac{1}{2}\sum_{i:s_i=m}\left(\sum_{j=1}^{T} Q_{jj}[f_{ij} - \bar{f}_{ij}]^2\right) + b$, where the rate parameter is composed from the subset of latent functions, $\{\mathbf{f}_i\}_{i:s_i=m}$, for those domains assigned to cluster m.

3. Sample cluster assignments under a similar Pólya urn representation shown in Equation 9 as for the GP, only here the mixture posterior is conjugate, so,

$$\pi\left(s_i = s|\mathbf{s}_{-i}, \kappa^*, \alpha, \tau_\epsilon, \mathbf{f}_i\right) \propto \begin{cases} \frac{n_{-i,s}}{n-1+\alpha} \prod_{j=1}^{T} \mathcal{N}\left(f_{ij}|\bar{f}_{ij}, [\kappa^*_{s_i} Q_{jj}]^{-1}\right) & \text{if } 1 \leq s \leq M^- \\ \frac{\alpha}{n-1+\alpha} d_{0,i} & \text{if } s = M^- + 1, \end{cases} \tag{10}$$

   where $d_{0,i} = \int \mathcal{N}\left(y|\kappa, \ldots\right) G_0(d\kappa) = \frac{b^a \Gamma(a_2)\left[\prod_{j=1}^{J} \mathcal{N}\left(0|0, Q_{jj}^{-1}\right)\right]}{\Gamma(a) b_{2,i}^{a_2}}$, where a_2 is as defined above and $b_{2,i}$ is term i in the sum that composes b_2, defined above.

4. Sample global precision, τ_ϵ, in a Gibbs step from,

$$\pi\left(\tau_\epsilon|\{y_{ij}\}\right) = \mathcal{G}a\left(a_1, b_1\right),$$

   with shape parameter, $a_1 = \frac{NT}{2} + c$, and rate parameter, $b_1 = \frac{1}{2}\sum_{i=1}^{N}\sum_{j=1}^{T}\left(y_{ij} - f_{ij}\right)^2 + d$.

4.3. Illustration using growfunctions on CPS data

We perform estimation under the dependent iGMRF estimation of Equation 8 with the function, `gmrfdpgrow()`, as illustated:

```
R> res_gmrf <- gmrfdpgrow(y = y_short, q_type = "tr", q_order = 2,
+    n.iter = 20000, n.burn = 10000, n.thin = 10)

Production Interation: 0
Production Interation: 450

⋮

Production Interation: 19800
Your additive set of iGMRF precision terms includes type = tr and order = 2
```

The `q_type` input allows selection of a precision to capture the trend (using `q_type = "tr"`) or seasonality (by specifying `q_type = "sn"`). The associated order is encoded with an integer value set for `q_order`. The default specification is `q_type = "tr"` and `q_order = 2`, which generally performs well such that the typical user will choose to accept the defaults and will not need to address these inputs. It bears mention that although we prefer an order 2 precision, as noted above, any order may be specified, including order 1. We select more sampling iterations than under the `gpdpgrow()` because the posterior sampling algorithm of the joint distribution under a dependent iGMRF (of Equation 8) admits a conjugate construction that computes much faster than the dependent GP (of Equation 2). So we don't need to employ techniques to improve the sampling efficiency (number of effective samples) as it is relatively inexpensive to perform more sampling iterations to achieve chain convergence. (We compare the computational performance of the GP and iGMRF in the next section in the context of their estimation robustness).

The `cluster_plot(object)` function also inputs objects from `gmrfdpgrow()` and renders the same two plots of fitted functions versus data inputs and functions aggregated to cluster plot panels.

```
R> fit_plots_gmrf <- cluster_plot(object = res_gmrf, units_name = "state",
+    units_label = cps$st, single_unit = TRUE, credible = TRUE)
R> print(fit_plots_gmrf$p.cluster)
R> print(fit_plots_gmrf$p.fit)
```

Figure 4 reveals that the iGMRF models produces a nearly identical result to the GP model with the estimation of 2 clusters differentiated by degree of sensitivity in employment levels to the great recession. Comparing Figure 5 to Figure 1 suggests that the iGMRF tends to fit the data more closely (though the different plotted states are randomly-selected for each).

We may employ a comparison plot function, `fit_compare(objects)`, that will provide a side-by-side comparison of a randomly-selected state for each cluster whose latent function is estimated under two different models.

```
R> objects <- vector("list", 2)
R> objects[[1]] <- res_gmrf
```

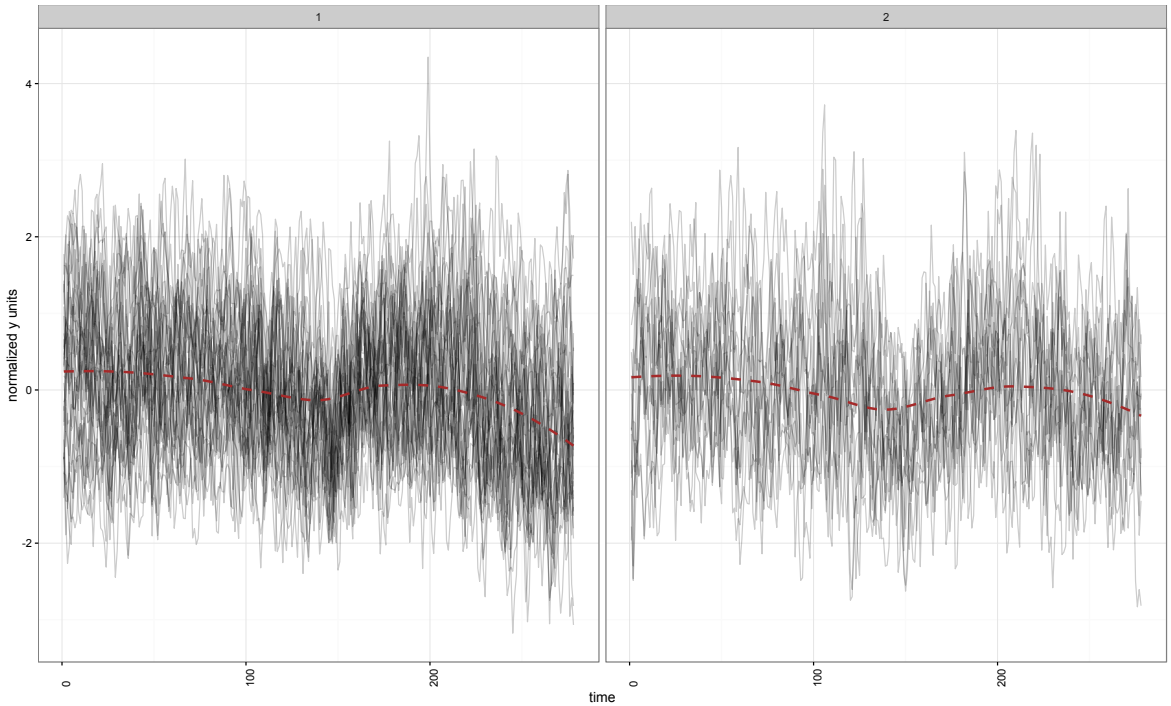

Figure 4: Collections of functions, $\{\mathbf{f}_i\}_{s_i=m}$, in plot panels indexed by cluster, $m = 1, \ldots, (M = 2)$ for the iGMRF model under a $RW2(\kappa)$ (order 2 trend) precision matrix estimated using `gmrfdpgrow()`.

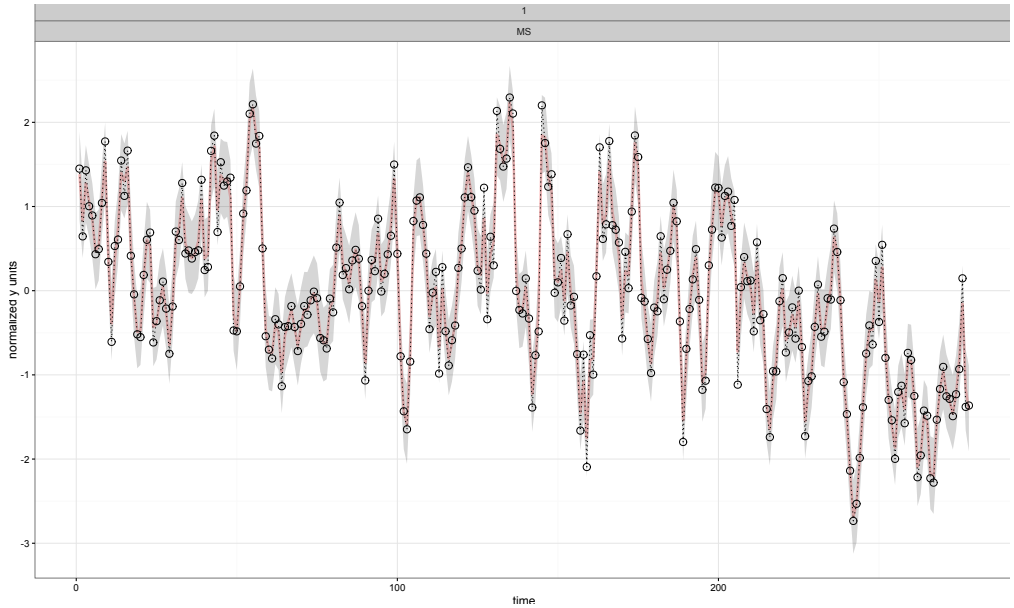

Figure 5: Estimated latent $T \times 1$ function, $\mathbf{f}_i$ (pink line), compared to observed data values, $\mathbf{y}_i$, for a single, randomly-selected domain, i, for the iGMRF $RW2(\kappa)$ model.

```
R> objects[[2]] <- res_gp
R> label.object <- c("gmrf_tr2", "gp_rq")
R> H <- fit_plots_gp$map$cluster
R> fit_plot_compare_facet <- fit_compare(objects = objects, H = H,
+    label.object = label.object, y.axis.label = "normalized y values",
+    units_name = "state", units_label = cps$st)
R> print(fit_plot_compare_facet$p.t)
```

The function, `fit_compare`, requires inputting vector, `H`, of length N (the number of states) that provides an assignment of each state to some cluster, $m \in 1, \ldots, M$, where M denotes the total number of clusters. This input is used to randomly select a state within each cluster for plotting. A `data.frame` object, `map`, returned from `cluster_plot(object)` includes a vector, `cluster`, that provides such information. We choose the clustering of states in `res_gp`, estimated from `gpdpgrow()`, and generate `map` using `cluster_plot(res_gp)`, as earlier shown.

```
R> head(fit_plots_gp$map)
```

```
  units_numeric cluster state
1             1       1    AK
2             2       1    AL
3             3       1    AR
5             5       1    CA
6             6       1    CO
```

The function, `fit_compare()` inputs a `list` of any two `gpdpgrow()` or `gmrfdpgrow()` objects and produces the plot shown in Figure 6. The GP model using the rational quadratic covariance formula produces a notably higher degree of smoothing than does the iGMRF model of order 2 (also referred to as a "random walk 2" or $RW2(\kappa)$). We may assess whether the closer fitting functions estimated under the iGMRF is overfitting the data by comparing fit statistics for `res_gp` with `res_gmrf`. Return objects from both `gpdpgrow()` and `gmrfdpgrow()` include a log pseudo marginal likelihood (`lpml`) statistic that uses the training data to compose the marginal predictive distribution in a leave-one-out fashion, which encodes some penalty for model complexity (Congdon 2005). The `lpml` is extracted with `res_gp$lpml` for the rational quadratic GP and `res_gmrf$lpml` for the $RW2(\kappa)$ iGMRF. If we multiply the `lpml` by -1, lower values indicate better fit and we see that `res_gmrf` provides a substantially better fit than `res_gp`.

```
R> res_gp$lpml
```

```
[1] -11419.37
```

```
R> res_gmrf$lpml
```

```
[1] -9408.844
```

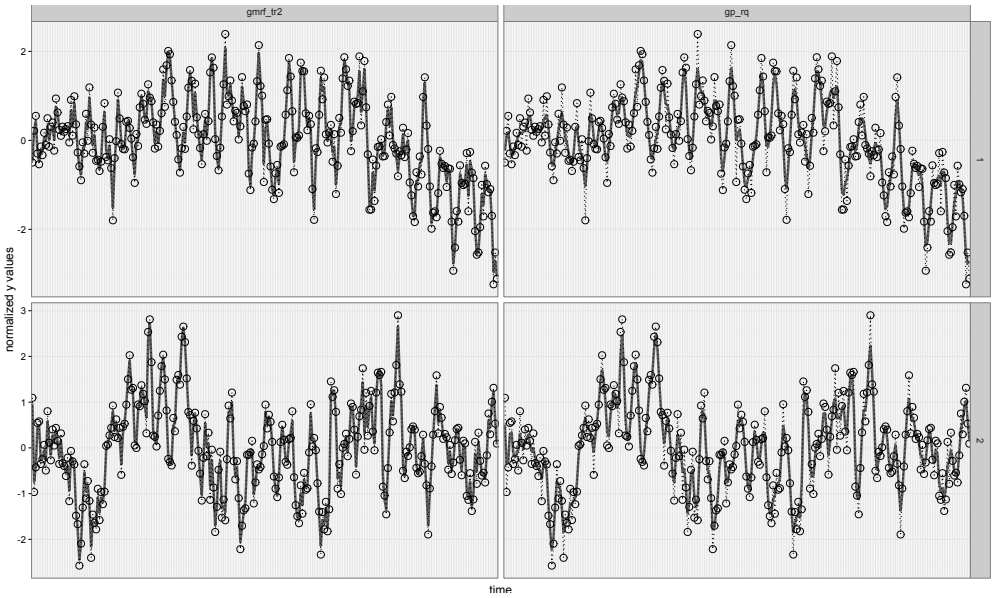

Figure 6: Plots of latent $T \times 1$ functions, $\{\mathbf{f}_i\}$, with each column representing a different model labeled with the option `label.object` in function, `fit_compare()`. The rows represent the clusters and the functions in each model-cluster cell are randomly-selected. This plot compares the GP model under the rational quadratic covariance formula to the iGMRF $RW2(\kappa)$ precision term.

We continue by next illustrating how to extend `gpdpgrow()` and `gmrfdpgrow()` from employment of one function or term to more complex latent functions formed from the sum of multiple functions or terms. We also demonstrate the ability of these estimation functions to handle intermittent missingness-at-random, which we use to offer a more robust comparison of fit performance than the `lpml`.

5. Performance comparison of dependent GPs and iGMRFs

growfunctions allows the user to employ multiple covariance terms, each with its own covariance formula, in `gdpgrow()` and multiple precision terms, each with a different order or length scale, in `gmrfdpgrow()`. We formally specify such a model for the GP with,

$$\overset{T\times 1}{\mathbf{y}_i} |\mathbf{b}_i, \tau_\epsilon \overset{\text{ind}}{\sim} \mathcal{N}_T\left(\mathbf{b}_i, \tau_\epsilon^{-1}\mathbb{I}_T\right),\ i = 1, \ldots, N \tag{11a}$$

$$\mathbf{b}_i = \mathbf{f}_{1,i} + \ldots + \mathbf{f}_{L,i} \tag{11b}$$

$$\mathbf{f}_{\ell,i}|\boldsymbol{\theta}_{\ell,i} \overset{\text{ind}}{\sim} \mathcal{N}_T\left(\mathbf{0}, \overset{T\times T}{\mathbf{C}}(\boldsymbol{\theta}_{\ell,i})\right) \tag{11c}$$

$$\boldsymbol{\Theta}_i = (\boldsymbol{\theta}_{1,i}, \ldots, \boldsymbol{\theta}_{L,i}) \tag{11d}$$

$$\boldsymbol{\Theta}_1, \ldots, \boldsymbol{\Theta}_N | G \overset{\text{iid}}{\sim} G \tag{11e}$$

$$G|G_0 \sim \text{DP}(\alpha, G_0), \tag{11f}$$

where L denotes the number of additive terms, where each specifies its own covariance matrix

with parameters, $\boldsymbol{\theta}_{\ell,i},\ \ell \in (1,\ldots,L)$. The approach of composing $T \times 1$ function, $\mathbf{b}_i$ for each state, $i \in (1,\ldots,N)$, from the addition of multiple functions builds a more complex surface by a layering of simpler surfaces, each drawn from a different Gaussian process. The choice of L is typically motivated by the nature of the features expected in $\mathbf{y}_i$ by the analyst. If the analyst expects both "local" trends for sub-intervals of the total time period, as well as a persistent, "global" trend, then they might specify a covariance matrix formed by the addition of two exponential covariance matrices, which would allow the data to estimate both short and long length-scale trends. Such a case may occur if the time series, $\mathbf{y}_i$, represents an economic variable whose values are characterized by unusual periods of increase or decrease, as well as a long-term trend. Similarly, if one expects both quarterly and yearly seasonality, employment of two seasonal covariance terms could be used to capture both. The choice of L may be evaluated by examining the `lpml` leave-one-out fit statistic, described in Table 3 (that provides a complexity penalty), as well as by using the `MSPE()` function, which we next illustrate, on a test set not used to train the model. The DP prior estimates clusters of states based on all L sets of parameters in $\boldsymbol{\Theta}_i$.

A multiple term dependent iGMRF formulation is specified similarly to Equation 11, with $\boldsymbol{\kappa}_i = (\kappa_{1,i},\ldots,\kappa_{L,i})$ replacing $\boldsymbol{\Theta}_i$ and each $\mathbf{f}_{\ell,i}$ is drawn using a precision matrix, $\kappa_{\ell,i}\mathbf{Q}_\ell$. Each term, $\mathbf{f}_{\ell,i}$, specifies its own (rank deficient) precision matrix.

The estimation functions are designed to handle intermittent missingness-at-random in the $N \times T$ response input matrix, `y`. The missing values in `y` are estimated from their posterior predictive distribution. The plot functions, `cluster_plot()` and `fit_compare()` will leave blanks at any missing data values (but will render the estimated function values). To illustrate, we randomly selected 10% of the CPS observations in `y_short` and set them to missing. Binary matrix, `pos`, indicates randomly-selected positions set to missing from `y_short` with a 1. The resulting data matrix, `y_obs`, sets missing positions to `NA` and is input to our estimation functions.

```
R> m_factor <- 0.1
R> M <- floor(m_factor * N * T)
R> m_vec <- rep(floor(M / N), N)
R> if(sum(m_vec) < M)
+  {
+    m_left <- M - sum(m_vec)
+    pos_i <- sample(1:N, m_left, replace = FALSE)
+    m_vec[pos_i] <- m_vec[pos_i] + 1
+  }
R> pos <- matrix(0, N, T)
R> for(i in 1:N)
+  {
+    sel_ij <- sample(3:(T-3), m_vec[i], replace = FALSE)
+    pos[i,sel_ij] <- 1
+  }
R> y_obs <- y_short
R> y_obs[pos == 1] <- NA
```

We use input, `gp_cov`, to set the number and type of covariance terms in `gpdpgrow()`. `gp_cov` is specified with a vector whose length equals the number of terms. There are three options for

the form of the covariance matrix. Option "se" refers to the squared exponential covariance formula, while option "rq" refers to the rational quadratic formula. The option, "sn", specifies a quasi-periodic or seasonal term that requires another input, sn_order, that specifies the order or length scale of the periodic seasonality. The seasonal term is composed as the product of a seasonal covariance of fixed length scale equal to sn_order and a squared exponential (with length scale estimated by the data) to allow the pattern of seasonality to vary with time. The default input is gp_cov = "rq", which is a single rational quadratic term that provides a parsimonious specification for modeling a multiple length scale covariance matrix. We replace "rq" in the single-term formulation we earlier discussed with a simpler "se" in the presence of a second, quasi-periodic term. We also specify more observations for our coarse covariance matrix approximations because our functions are more complex.

```
R> res_gp_2 <- gpdpgrow(y = y_obs, gp_cov = c("se", "sn"), sn_order = 4,
+    n.iter = 10000, n.burn = 5000, n.thin = 5, n.tune = 2500,
+    sub_size = c(100, 60))

Production Interation: 9000
Production Interation: 9450
Production Interation: 9900
Finished generating estimated GP functions, bb
Your chosen GP covariance terms includes term = se
Your chosen GP covariance terms includes term = sn
```

Input vector, q_type, specifies the form of the precision matrix for each iGMRF term. The options are "tr" for a term to capture trends and "sn" to capture seasonality. The companion input vector, q_order, provides the order associated to each term; for example, the second term is specified as a seasonal term of length 4 months (for our CPS data set).

```
res_gmrf_2 <- gmrfdpgrow(y = y_obs, M_init = 5, q_order = c(2, 4),
+   q_type = c("tr", "sn"), n.iter = 20000, n.burn = 10000,
+   n.thin = 10)

Production Interation: 18450
Production Interation: 18900
Production Interation: 19350
Production Interation: 19800
Your additive set of iGMRF precision terms includes type = tr and order = 2
Your additive set of iGMRF precision terms includes type = sn and order = 4
```

We compare the resulting estimated functions using the 2-term formulations. Figure 7 shows that the higher complexity of the formulations produces functions with less smoothing that more closely fit the data.

```
R> objects <- vector("list", 2)
R> objects[[1]] <- res_gmrf_2
R> objects[[2]] <- res_gp_2
R> label.object <- c("gmrf_tr2sn4", "gp_sesn4")
```

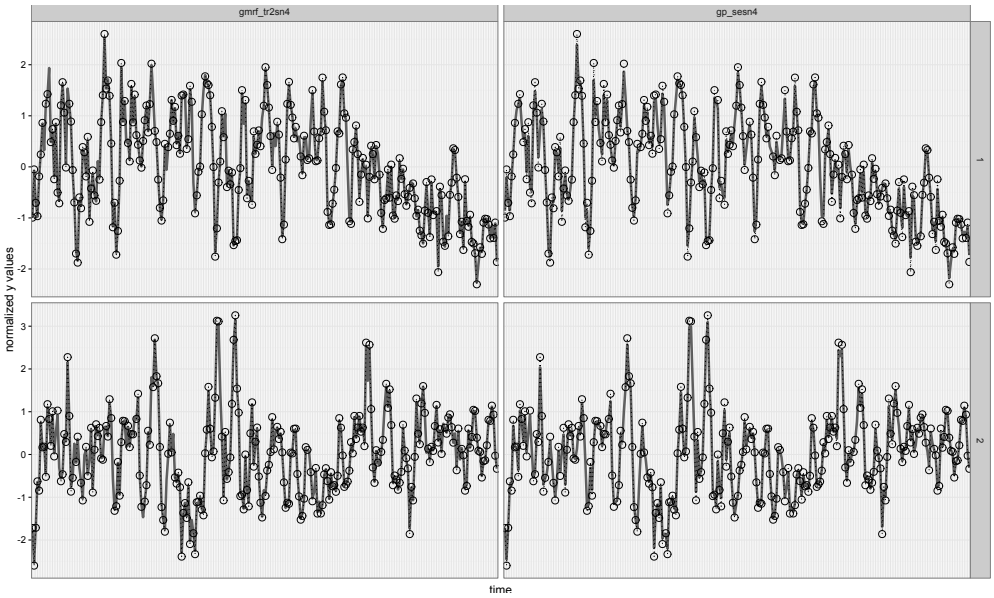

Figure 7: Plots of latent $T \times 1$ functions, $\{\mathbf{f}_i\}$, under employment of 2-terms, each, for the GP and iGMRF models. The left column represents a two-term GP formulation and the right column represents a 2-term iGMRF construction, with column labels based on the input to `label.object` in function, `fit_compare()`. The rows represent the clusters and the functions in each model-cluster cell are randomly selected.

```
R> H <- fit_plots_gp_2$map$cluster
R> fit_plot_compare_facet <- fit_compare(objects = objects, H = H,
+    label.object = label.object, y.axis.label = "normalized y values",
+    units_name = "state", units_label = cps$st)
R> print(fit_plot_compare_facet$p.t)
```

The use of the leave-one-out `lpml` fit statistic we earlier employed is estimated from the training data, rather than a separate test set. So we use the prediction of missing observations (not used to train the model) to compose a mean squared prediction error (MSPE) that compares the prediction of missing values from the posterior predictive distribution to the actual or true values that we held out from the estimation. The MSPE will generally provide a superior indication of the quality of fit than the `lpml` because it assess fit on test data not used to train the model. We normalize the MSPE with division by the variance of the test set (which, in our case is the 10% randomly set to missing) to produce an intuitive percent error measure. **growfunctions** offers function, `MSPE`, that inputs the full data (in this case, `y_short`) and the associated $N \times T$ binary matrix with 1 in each missing position used for model estimation, along with the returned object from `dpgpgrow()` or `gmrfdpgrow()`. We extract `nMSPE`, the normalized mean squared prediction error.

```
R> (nmspe_gp <- MSPE(res_gp_2, y_short, pos)$nMSPE)

[1] 0.3022423

R> (nmspe_gmrf <- MSPE(res_gmrf_2, y_short, pos)$nMSPE)
```

Model	Runtime
`res_gp`	51506
`res_gmrf`	520
`res_gp_2`	71000
`res_gmrf_2`	700

Table 1: Model runtimes in *CPU seconds* on 1 of 2 threads on a single core of a 2.5 GHz Intel Quad-core I-7 CPU on CPS data.

```
[1] 0.5270994
```

We see that the dependent GP formulation, under 2-terms produces a lower `nMSPE` than does the 2-term dependent iGMRF formulation, probably because we chose the simpler squared exponential formula in the GP for the first term when paired with a quasi-periodic function, which produced a formulation less likely to overfit. Although the dependent GP formulation may provide a more robust fit than the iGMRF formulation, it is far more computationally-intensive, even after our implementation of sampling methods that allow us to reduce the number of posterior sampling iterations. Table 1 presents the computer run times in CPU-seconds for the single and multiple-term GP and iGMRF models run in the previous sections using the CPS dataset. The models ending in `2` (e.g., `res_gp_2`) randomly exclude 10% of the CPS observations. The effect of leaving out observations induces sampling the missing values (under a missing at random assumption) from the model posterior predictive distribution. There is a notable increase in the computational intensity for the GP model under missing observations because we are forced to co-sample the functions, $\{\mathbf{f}_i\}$, which we marginalize over in the sampler when there are no missing data. There is little added computation time for the iGMRF model because we co-sample the functions in either event of missing or complete data. We see that the GP models are on the order of 100 times more computationally intensive (for producing an equivalent number of effective sample sizes). The much higher computational intensity of the GP models attributes to computation of the cholesky decompositions of the covariance matrix. While we have employed very recent innovations that reduce the computer run time per effective sample size, while yet producing exact inference from the posterior distribution of the GP model, there remains a large gap in computational performance between the GP and iGMRF formulations. As demonstrated above, the added flexibility of the GP construction, that permits the data to estimate the length scales of the functions, produces more accurate predictions than does the iGMRF construction, which requires the analyst to specify a length scale. So when the focus for inference are the estimated functions we prefer the use of the GP, despite the computational intensity. When the functions are nuisance parameters included in a larger model, however, we prefer the iGMRF for its relatively fast computation. It bears mention that our implementation of the estimation models in C++ makes them tractable.

Although we illustrated `gpdpgrow()` and `gmrfdpgrow()` with 2 terms, the user may specify any number of terms. The incremental increase in computation time for `gpdpgrow()` is of the same order as for `gmrfdpgrow()` (rather than being much larger), because the terms are summed into a single added covariance matrix so that the task to compute a cholesky decomposition (of this dense matrix) remains relatively unchanged.

We conclude this section by highlighting objects returned from our estimation and plot func-

Object	Description
`Theta`	An $O \times NP$ `matrix` of posterior samples for covariance parameters. Specific to GP models estimated with `gpdpgrow()`. O denotes the number of post burn-in posterior sampling iterations. N denotes the number of domains and P, the number of covariance parameters. If multiple covariance matrices are specified, P sums all of these parameters. `Theta` is labeled with column names, `"theta[p,i]"`. `p` indexes each parameter type and `i` indexes the domain.
`Kappa`	An $O \times NK$ `matrix` of posterior samples for precision parameters. Specific to iGMRF models estimated with `gmrfdpgrow()`. L denotes the number of precision matrices. `Kappa` is labeled with column names, `"kappa[l,i]"`. `l` indexes each precision matrix and `i` indexes the domain.
`bb`	An $O \times NT$ `matrix` of posterior samples for latent functions. T denotes the number of time points. `bb` is labeled with column names, `"bb[i,j]"`. `i` indexes the domain and `j`, the time point.
`f`	A `list` object of functions that add to `bb`. `f` is of length `L`, the number of terms specified. An $O \times NT$ `matrix` is returned for each term. Sum of elements of `f` equals `bb`.
`Tau_e`	An $O \times 1$ `matrix` of samples for model noise precision.
`M`	An $O \times 1$ `matrix` of samples for number of clusters.
`Conc`	An $O \times 1$ `matrix` of samples for model DP precision, α.

Table 2: Post-burnin posterior samples for model estimated parameters output from `samples(res)`, where `res` is a `gdpgrow()` or `gmrfdpgrow()` object.

tions that may be useful for further analysis. Table 2 lists objects containing posterior sampled values for each set of model parameters that may be easily extracted from the estimation function using the S3 function, `sample(res)`, where `res` is a return object from either `gpdpgrow()` or `gmrfdpgrow()`. (The S3 class is primarily used to define functions that wrap a return object from an estimation function.)

Table 3 highlights varied objects (and how to access them) returned by estimation and plot functions that are anticipated to be useful for further inference on fit or clustering.

6. Accounting for an informative sampling design

So far we've focused on stabilizing the model-free, survey direct estimates and employed the CPS as an example. Federal government surveys are typically either administered to households (as is the CPS) or to business establishments. The published CPS direct estimates are composed by weighting each household's response inversely proportional to its probability of inclusion. The weighting of survey respondents re-balances the information in the sample to reflect the population to ensure the direct estimate provides an unbiased estimate for the population-level statistic of interest. Our estimation of latent functions from the direct estimates did not require that we employ weighting from the sample to the population because

Object	Description
`bigSmin`	`list` object of M components, where M denotes number of clusters. Component m contains a vector of subjects in cluster m. Accessed with `res$bigSmin`. Where `res` is returned from `gpdpgrow()` or `gmrfdpgrow()`.
`lpml`	A leave-one-out penalized fit statistic described in Congdon (2005). Accessed with `res$lpml`. Where `res` is returned from `gpdpgrow()` or `gmrfdpgrow()`.
`map`	A `matrix` that links each domain to a cluster membership. Accessed with `plot_out$map`. Where `plot_out` is returned from plot routine, `cluster_plot(res)`. Where `res` is returned from `gpdpgrow()` or `gmrfdpgrow()`.

Table 3: Useful objects returned by estimation and plot routines.

such was already handled in the direct estimate.

It is sometimes necessary or more useful to work directly with the household or establishment level responses for modeling and performing inference. The units in the sample data are randomly drawn using a sampling design to ensure adequate coverage or to reduce the cost of conducting the survey. For example, population units (e.g., households) may be disjointly assigned to a set of strata based on some predictors (e.g., gender and age) that may be important for differentiating responses in some variable of interest. Unit inclusion probabilities are set, by stratum, and may differ across the strata if the survey sampler wants to ensure particular groups are over-represented (in order to obtain precise inference). A stratified random sample may be more efficient (in that a lower sample size may be needed to make inference at some desired level of precision) than an *iid* (simple random) sample because it many reduce the variability over all possible samples (Särndal, Swensson, and Wretman 2003). Contrastingly, a block sampling strategy is often used when the units are sampled from a geographic grid in order to reduce the costs of collection; for example, a country may be divided into regions and the regions become primary sampling units, a subset of which are randomly chosen for inclusion. The block sample may be less efficient than the simple random sample to the extent that each block is relatively homogenous (such that within-block variances are small). A well-designed block sampling strategy that composes each block to be heterogenous may be more efficient than a simple random sampling design, however.

Sampling designs where the unit inclusion probabilities are correlated with the response of interest are said to be “informative". An example of such a design is the Current Employment Statistics (CES) survey of business establishments, to estimate employment levels, conducted by the Bureau of Labor Statistics. A stratified sampling design is used where one of the variables selected to compose strata is the employment size category of the establishment. There are 7 defined size categories and establishments in larger-sized categories receive higher probabilities for inclusion in the survey because they provide more information about domain-indexed (e.g., by industry or metropolitan area) employment estimates. The resulting distribution for the observed sample will be different from that for the population under an informative sampling design. We will next employ sampling weights constructed to be inversely proportional to the inclusion probabilities for the observed units in our sample to form a sampling-weighted “pseudo" likelihood that, when convolved with the prior, induces a

pseudo posterior distribution.

We construct the pseudo likelihood,

$$p^{\pi}\left({}^{T\times 1}_{\mathbf{y}_i}|\mathbf{f}_i,\tau_\epsilon\right)=\mathcal{N}_T\left(\mathbf{f}_i,\tau_\epsilon^{-1}\mathbb{I}_T\right)^{\tilde{w}_i},\ i=1,\ldots,N, \tag{12}$$

in place of the usual likelihood of Equation 1a to correct for informative sampling for both the GP and iGMRF models, where $\tilde{w}_i$ is a known sampling weight that is inversely proportional to the probability of inclusion. We denote the pseudo likelihood likelihood by $p^{\pi}(\mathbf{y}_i|-) = p(\mathbf{y}_i|-,w_i)$ from the usual construction for the unweighted likelihood. The weight applied to each likelihood observation assigns the relative importance or representativeness of that observation for describing the population, such that it serves to re-balance the information in the sample to approximate that in the population (on which we wish to make inference). This approach may be viewed as a Bayesian implementation of the pseudo-likelihood approach suggested for MLE by Chambers and Skinner (2003) (which optimizes the logarithm of the pseudo likelihood, which is a sampling weighted score function). By replacing the usual likelihood in our models by the pseudo-likelihood, we estimate an associated pseudo-posterior (from which we draw samples for parameter values in our MCMC), which asymptotically converges to the posterior distribution for the population.

The posterior uncertainty (variance) is regulated by the sum of the sampling weights. We follow Savitsky and Toth (2016) and start with unnormalized weights, $\{w_i = 1/\pi_i\}$, and subsequently normalize them, $\tilde{w}_i = \frac{w_i}{\sum_n^{w_i}}$, $i = 1,\ldots,n$, to sum to the sample size, n, the asymptotic units of information in the sample. Although our method utilizes the weights as a "plug-in", rather a fully Bayesian construction, Pfeffermann and Sverchkov (2009) use Bayes rule to demonstrate one may replace the weights with their conditional expectation given the observed response to correct for informative sampling. Replacing the raw weights with their conditional expectation given the observed response may serve to reduce the total variation attributed to weighting (and the resulting posterior uncertainty) in the case where the actual sampled observations express information in different proportions than intended in the sampling design. Even though the conditional *distribution* of the weights given the response is generally different for the observed sample than for the population, nevertheless their conditional *expectations* are equal.

The pseudo posterior distributions are formed by replacing the usual likelihood by the pseudo likelihood from Equation 12. Taking the logarithm of Equation 12 provides a plug-in correction that convolves the weights with kernel of the likelihood for each observation that serves to weight each likelihood contribution back to the population. The log- pseudo posterior kernel for sampling cluster locations, $\{\theta^*_{pm}\}$, $p = 1,\ldots,P$; $m = 1,\ldots,M$, in step 1 of the sampling algorithm described in Section 3.2 is now updated to include sampling weights, $\{\tilde{w}_i\}$,

$$\begin{aligned}&\log\,\pi^{\pi}\left(\theta^*_{pm}|\boldsymbol{\theta}^*_{-pm},\mathbf{s},\tau_\epsilon,\{\mathbf{y}_i:s_i=m\},\{\tilde{w}_i:s_i=m\}\right)\\&\propto -\frac{1}{2}n_m\log\left(|\mathbf{C}^{\tau}\left(\boldsymbol{\theta}^*_m\right)|\right)-\frac{1}{2}\sum_{i\in s_m}\tilde{w}_i\mathbf{y}_i'\mathbf{C}^{\tau}\left(\boldsymbol{\theta}^*_m\right)\mathbf{y}_i+(a-1)\log(\theta^*_{pm})-b\theta^*_{pm}.\end{aligned}$$

Sampling the cluster assignments, $\mathbf{s} = (s_1,\ldots,s_n)$ (where n denotes the sample size taken from a population of size, N) remains unchanged because we sample each s_i, for observation i, conditionally on the rest, which is only a function of the likelihood for that observation.

So the sampling weights indirectly influence the pseudo posterior distribution for the cluster assignments through conditioning on the cluster locations, $\{\theta^*_{pm}\}$.

By a similar logic, the mean and precision hyperparameters of the Gaussian pseudo posterior,

$$\pi^{\pi}\left(f_{ij}|\{f_{ij}\}_{-j},\tau_\epsilon,y_{ij},\tilde{w}_i\right)=\mathcal{N}\left(\frac{e_{ij}}{\phi_{ij}},\phi_{ij}^{-1}\right),$$

specified in the iGMRF sampling algorithm of Section 4.2 is updated to $e_{ij}=\tau_\epsilon\tilde{w}_i y_{ij}+Q_{jj}\kappa^*_{s_i}\bar{f}_{ij}$ and $\phi_{ij}=\tau_\epsilon\tilde{w}_i+Q_{jj}\kappa^*_{s_i}$. The rate parameter, b_1, of the posterior distribution, $\pi\left(\tau_\epsilon|\{y_{ij},\tilde{w}_i\}\right)=\mathcal{G}a\left(a_1,b_2\right)$ is updated to, $b_1=\frac{1}{2}\sum_{i=1}^{N}\tilde{w}_i\sum_{j=1}^{T}\left(y_{ij}-f_{ij}\right)^2+b$.

Savitsky and Toth (2016) require 3 conditions that, together, define a class of sampling designs under which frequentist consistency of the pseudo posterior distribution at the true generating distribution is guaranteed: 1. The inclusion probabilities, (π_i) are all bounded away from zero; meaning that no portion of the population may be systematically excluded, which would prevent the sample - no matter how large - from ever reflecting the full balance of information in the finite population; 2. The sampling fraction, n/N, where n denotes the sample size and N, the finite population size, must converge to a constant, which indicates that samples drawn from the population express some minimal amount of information about that population; 3. The pairwise inclusion dependencies among units (e.g., establishments) must attenuate to 0 in the limit of N. The first two conditions and readily met by most commonly-used sampling designs. The third condition on asymptotic independence is satisfied by a broad class of sampling designs; for example, nearly all single stage designs, such as the stratified sampling designs we employ. An example of a more complicated sampling design that satisfies the third condition is a two-stage cluster design where the number of second stage units in each cluster grows to infinity in the limit of N.

growfunctions estimation functions are able to model unit level data by inputting a vector (of length N, the number of units) of inclusion probabilities (`ipr`), which are used to perform weighting adjustments of the full conditional posterior distributions. Our approach is based on a recent work of Savitsky and Toth (2016), which we believe offers the first *general* approach that may be used to incorporate sampling weights into Bayesian modeling. Weighting unit level responses is one approach, in the case of an informative sampling design, to re-balance the information in the sample data to more closely reflect the population, which would produce an estimated joint posterior distribution that is consistent at the true population distribution. The implication is that performing estimation on a sample without adjusting for inclusion probabilities would produce parameter estimates that are biased with respect to the distribution for the population if the sampling design is informative. We will next demonstrate how we may use **growfunctions** to assess the degree of informativeness of the sampling design based on differences in estimated parameter distributions with and without inclusion of sampling weights.

Our development of this feature to handle survey weighting was motivated a data set composed of business establishments participating in the Current Establishment Survey (CES), conducted by BLS, who report their number of employees on a monthly basis. These establishments (and all other establishments) are also required to report a set of statistics in the Quarterly Census of Employment and Wages (QCEW), which also includes number of employees. The employment level values reported should be equal between CES and QCEW, but that has not historically been the case for many establishments. So BLS collected 15

months of reported employment levels for those establishments where there were differences in the reported employment levels between CES and QCEW. Taking an absolute value of the monthly differences in the levels for establishment produces the same data structure as for the state-level CPS estimates of a collection of time series; only in this case we are working at the unit (establishment) level, instead of at the domain level.

Because the establishment level data may contain identifying information that would violate privacy, we are unable to include the actual data in **growfunctions**. We have, instead, added a function, `gen_informative_sample()`, that generates a population of N latent $T \times 1$ functions and associated noisy time series and draws an informative sample of size n from this population. The latent set of population functions of length T are drawn from a Gaussian process under a single rational quadratic covariance formula using M clusters of covariance parameters. (Setting `gp_type = "se"` may be alternatively employed to select a squared exponential covariance formula.) All establishments in the population are randomly assigned to the M clusters. Cluster location values are optionally set with a $P \times M$ matrix, `theta_star`, where the columns represent the clusters and the rows represent the parameter types for the selected covariance formula. We use the default for `theta_star`, which sets a 3×3 matrix (under assumption of a rational quadratic covariance formula with $P = 3$ parameters per establishment and $M = 3$ clusters) based on values estimated in the $T = 15$ QCEW-CES data set of employment level errors. The `noise_to_signal` input receives a value in $(0, 1)$ to indicate the percent of the average variances across the latent functions to set as the noise variance for generating the responses, `y`.

An informative sampling of size n is subsequently drawn from this population. By default, the sampling design is a single-stage, stratified random sample with $I = 4$ strata. (A more complex two-stage sample of blocks, followed by stratified random sampling within selected blocks may be invoked with input, `two_stage = TRUE`). The assignment of population units to the 4 strata are based on the variance of their noisy responses and a higher sampling probability is assigned to the larger variance strata. The return object from `gen_informative_sample()` is stored in `dat_sim` that includes the generated observed values for sampled establishments, `y_obs`, and associated latent function values, `bb_obs`.

An *iid* sample is also taken from the same generated population (and recorded in `y_iid` and `bb_iid`) that we will use as a comparison to assess the effectiveness of our use of weighting to control for an informative design since an *iid* sample is non-informative and requires no weighting.

```
R> dat_sim <- gen_informative_sample(N = 10000, n = 900, T = 15,
+    noise_to_signal = 0.1)
R> dat_sim$samp_plot
R> y_obs <- dat_sim$y_obs
R> T <- ncol(y_obs)
R> n.iter <- 10000
R> n.burn <- 5000
R> n.tune <- 2500
R> n.thin <- 5
```

Running `gen_informative_sample()` produces the plot in Figure 8, which renders a randomly-selected set of latent functions in each of the $M = 3$ clusters, drawn from the population of

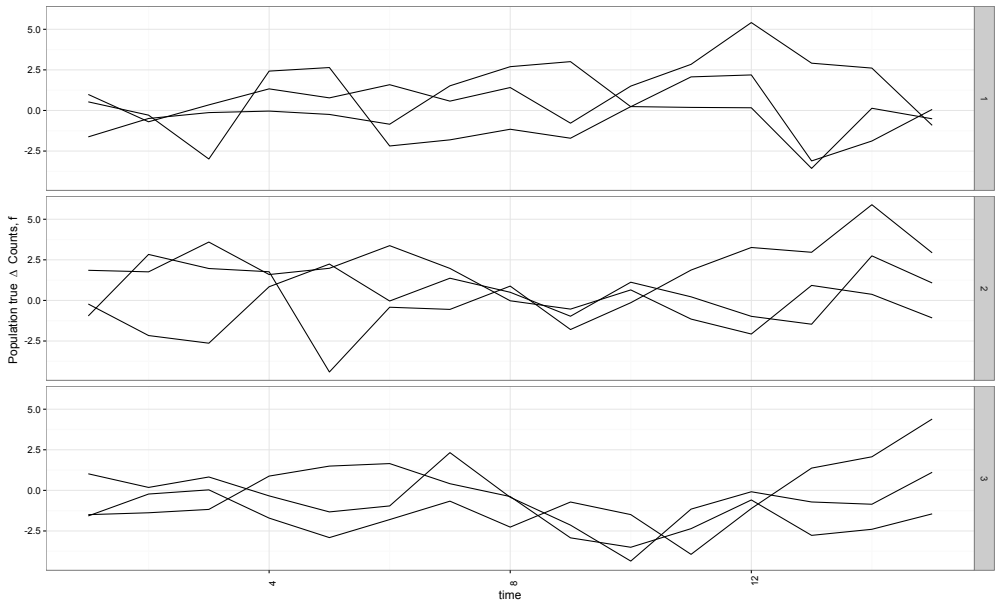

Figure 8: Plots of randomly-selected latent $T \times 1$ functions, $\{\mathbf{f}_i\}$, in each of $M = 3$ clusters, from a population generated with `gen_informative_sample()`.

latent functions, that allows us to see the relative shape and length-scales for the functions assigned to each cluster.

We perform estimation using `gpdpgrow()` and input a vector of $n = 900$ sample inclusion probabilities that we obtain from the `dat_sim` (`list`) object returned by `gen_informative_sample()`. Return object, `dat_sim` contains a `data.frame` object, `map_obs`, which holds a list of included establishments, their assignments to strata and clusters, and their inclusion probabilities in the field, `incl_prob`. The `incl_prob` vector is input to `ipr` in `gpdpgrow()`. The inclusion probabilities from `map_obs` are normalized such that their inverse (the sampling weights) sums to $n = 900$, the sample size, in order to ensure that the weighting conveys the right *amount* of information in the sample (see Savitsky and Toth 2016 for more details). We assign the result for our estimation that includes input of establishment inclusion probabilities to `res_gp_w`.

```
R> res_gp_w <- gpdpgrow(y = y_obs, ipr = dat_sim$map_obs$incl_prob,
+    gp_cov = "rq", n.iter = n.iter, n.thin = n.thin, n.burn = n.burn,
+    n.tune = n.tune)
R> fit_plots_w <- cluster_plot(object = res_gp_w,
+    units_name = "establishment",
+    units_label = dat_sim$map_obs$establishment,
+    single_unit = FALSE, credible = TRUE)
```

We then generate `res_gp_i` from `gpdpgrow()` that ignores the informative sampling design by just inputting the data matrix, `y_obs`, without an input for weighting, as we have previously demonstrated. Finally, we also perform estimation for the *iid* sample using `gpdpgrow(y_iid)`, where no weighting input is required, and assign the results to `res_gp_iid`.

growfunctions includes a plot function, `informative_plot()` that is designed to assess the degree of informativeness in the sampling design by inputting two objects, all either from

`gpdpgrow()` or `gmrfdpgrow()`, where one object is estimated under the inputting of the vector of inclusion probabilities (and labeled with `objects_labels = "weight"`) and the other object is estimated without inclusion probabilities (`"ignore"`). If the objects are estimated under `gpdpgrow()`, `informative_plot()` will present a side-by-side comparison of the estimated posterior distributions for the P covariance parameters in each of the M clusters between the weighted and unweighted estimations. The larger the difference in the locations of the posterior distributions between the two models, the greater the informativeness of the sampling design. If the objects are estimated using `gmrfdpgrow()`, the same plots are presented for the precision parameters. (`informative_plot()` will also render the comparison of posterior distributions for covariance or precision parameters under employment of multiple terms as we earlier modeled).

`informative_plot` inputs the estimation objects and also the `map` objects returned from `cluster_plot()`, as we've earlier seen using `compare_plot()`, in order to convey the clustering information for the sampled units. In addition to inputs for the 2 models that `"ignore"` and `"weight"`, an object labeled `"iid"` (for an *iid* sample drawn from the same population), is optionally allowed for input (but not required) under the assumption that there are some sampled units in common between the informative and *iid* samples. Inclusion of an *iid* sample may be useful because the posterior distribution of the object labeled `"weight"` should be similar that of `"iid"` (though with somewhat higher variance due to the variance in the weights). Lastly, another option input, `true_star`, allows one to input the actual cluster point values (that were input to `gen_informative_sample()` in the case the data are synthetic).

```
R> objects <- map <- vector("list", 3)
R> objects[[1]] <- res_gp_i
R> objects[[2]] <- res_gp_iid
R> objects[[3]] <- res_gp_w
R> map[[1]] <- fit_plots_i$map
R> map[[2]] <- fit_plots_iid$map
R> map[[3]] <- fit_plots_w$map
R> objects_labels <- c("ignore", "iid", "weight")
R> parms_plots_compare <- informative_plot(objects = objects,
+    objects_labels = objects_labels, map = map,
+    units_name = "establishment", model = "gp",
+    true_star = dat_sim$theta_star, map_true = dat_sim$map_obs)
R> parms_plots_compare$p.compare
```

The resulting rendered plot in Figure 9 randomly selects a unit in each cluster and plots its posterior distribution as estimated under the 3 models that ignore the informative design (by not inputting inclusion probabilities), that perform weighted estimation and the comparison *iid* sample (where the selected establishment is included in both the informative and *iid* samples). A dashed line is added in each panel at the true cluster location values input with `true_star`. Not surprisingly, we see that the model ignoring the informative design produces highly biased estimates, while the model that inputs sampling probabilities (converted to weights) produces a much less biased result that is close that for the *iid* sample, but with higher variance due to the added variation in the weights (that incorporates the relative uncertainty about the degree to which the sample reflects information from the population).

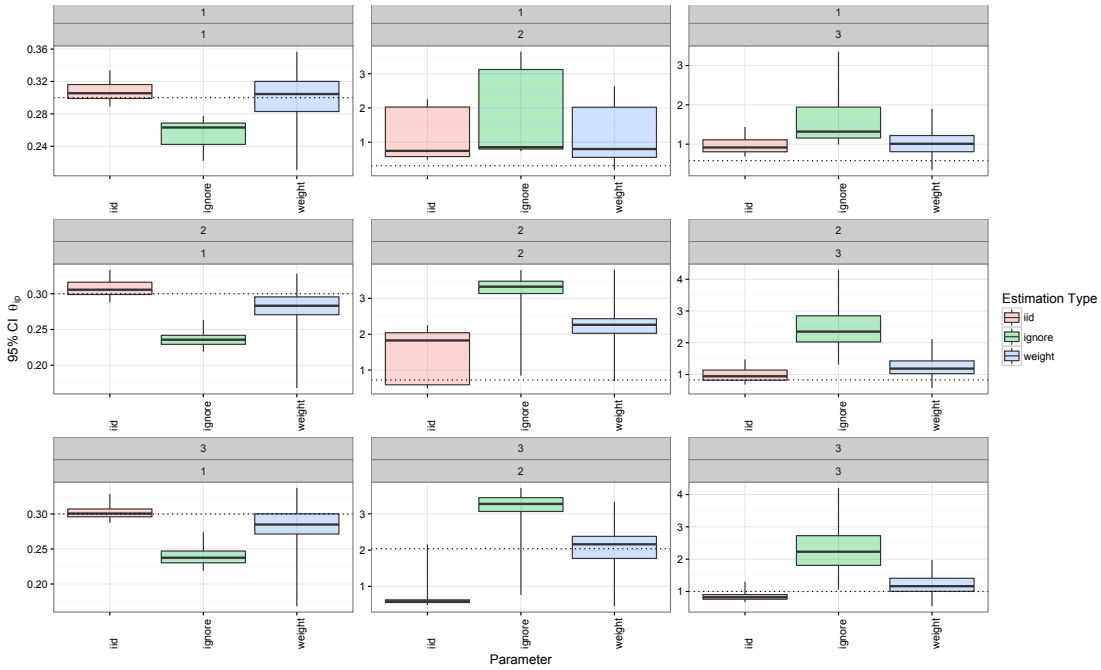

Figure 9: Posterior distributions (within 95% credible intervals) for $\boldsymbol{\theta}^*_m = \left(\theta^*_{m,1}, \theta^*_{m,2}, \theta^*_{m,3}\right)$ estimated under a stratified sampling design. The rows denote clusters $m = 1, \ldots, M = 3$ and the columns denote parameter type, $p = 1, \ldots, 3$. The dashed line in each panel represents the true cluster parameter values input with `true_star`.

7. Concluding remarks

Collections of domain or unit-indexed noisy time series represent a common data type associated with published government survey statistics. The employment of statistical modeling to reduce the volatility present in these time series of published statistics has recently become more common, though popular approaches focus on borrowing information from the time-indexed dependence and assume the time series are independent among the domains. **growfunctions** offers a solution that estimates de-noised functions which incorporates both a time and domain-indexed dependence structure and produces estimates which may be more efficient than those only accounting for the dependence in time. The use of a Dirichlet process mixture of latent functions also permits inference on the clustering structure of the domains that offers context to interpret the estimated function for each domain. The availability of both GP and iGMRF alternatives for modeling the latent functions offers the user an option for robust fit and another for fast computation. Although modeling under the GP is computationally-intensive, we have devised and implemented a posterior sampling algorithm to maximize the effective sample size that permits use of fewer posterior sampling iterations and mitigates the computational burden.

growfunctions further facilitates inference by offering plot functions that anticipate the type of inference users may typically need to conduct on the collection of time series data. These plot functions allow comparisons of the latent functions to the observed time series, provide visual distinctions among the estimated clusters of functions and promote a visual comparison of fit between differently configured estimation models.

There is a growing interest to apply modeling solutions directly to the unit-level responses acquired from surveys (rather than adjusting non-model-based, domain level direct estimates). **growfunctions** accommodates this interest by incorporating sample unit inclusion probabilities to re-balance the information in the sample to reflect that of the population and produces nearly unbiased estimation of parameters with respect to the distribution for the population. This is very new feature available for Bayesian modeling.

growfunctions implements the model formulations developed in Savitsky (2015) and Savitsky and Toth (2016) in a fashion that allows the user to customize their own analyses and extend or create new formulations beyond those envisioned in these enabling methods-focused papers; for example, the user may construct and estimate a GP or iGMRF of any number of terms and render comparison fit plots and generate associated out-of-sample fit statistics - all using the functions of **growfunctions**.

Future developments for **growfunctions** will include expanding the response types using generalized model constructions, as well as incorporating predictors into the determination of clusters. The incorporation of predictors for estimation of the weights or locations of the Dirichlet process prior estimates the conditional distribution, given the predictors, from which various quantiles of interest may be extracted.

References

Blackwell D, MacQueen JB (1973). "Ferguson Distributions via Pólya Urn Schemes." *The Annals of Statistics*, **1**, 353–355. doi:10.1214/aos/1176342372.

Chambers RL, Skinner CJ (2003). *Analysis of Survey Data*. John Wiley & Sons. doi:10.1002/0470867205.

Congdon P (2005). *Bayesian Models for Categorical Data*. John Wiley & Sons. doi:10.1002/0470092394.

Dahl DB (2006). "Model-Based Clustering for Expression Data via a Dirichlet Process Mixture Model." In K Do, P Müller, M Vannucci (eds.), *Bayesian Inference for Gene Expression and Proteomics*, pp. 201–215. Cambridge University Press.

Escobar MD, West M (1995). "Bayesian Density Estimation and Inference Using Mixtures." *Journal of American Statistical Association*, **90**, 577–588. doi:10.1080/01621459.1995.10476550.

Gramacy RB (2007). "**tgp**: An R Package for Bayesian Nonstationary, Semiparametric Nonlinear Regression and Design by Treed Gaussian Process Models." *Journal of Statistical Software*, **19**(9), 1–46. doi:10.18637/jss.v019.i09.

Gramacy RB, Taddy M (2010). "Categorical Inputs, Sensitivity Analysis, Optimization and Importance Tempering with **tgp** Version 2, an R Package for Treed Gaussian Process Models." *Journal of Statistical Software*, **33**(6), 1–48. doi:10.18637/jss.v033.i06. URL http://www.jstatsoft.org/v33/i06/.

Lee D (2013). "**CARBayes**: An R Package for Bayesian Spatial Modeling with Conditional Autoregressive Priors." *Journal of Statistical Software*, **55**(13), 1–24. doi:10.18637/jss.v055.i13.

Maples JJ, Bell WR, Huang ET (2009). "Small Area Variance Modeling with Application to County Povery Estimates from the American Community Survey." In *JSM Proceedings, Section on Survey Research Methods*, pp. 5056–5067. Alexandria. URL https://www.amstat.org/sections/srms/proceedings/y2009/Files/305528.pdf.

Neal R (2000a). "Slice Sampling." *The Annals of Statistics*, **31**, 705–767. doi:10.1214/aos/1056562461.

Neal RM (2000b). "Markov Chain Sampling Methods for Dirichlet Process Mixture Models." *Journal of Computational and Graphical Statistics*, **9**(2), 249–265. doi:10.1080/10618600.2000.10474879.

Nugent C, Hawala S (2012). "Research and Development Methods of Estimating Poverty for School-Age Children." *Publication of U.S. Census Bureau*, pp. 1–13. URL http://www.census.gov/did/www/saipe/publications/files/nugenthawalajsm2012.pdf.

Pfeffermann D, Sverchkov M (2009). "Inference under Informative Sampling." In D Pfeffermann, CR Rao (eds.), *Handbook of Statistics 29B: Sample Surveys: Inference and Analysis*, pp. 455–487. Elsevier.

Rasmusen CE, Williams C (2006). *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge.

R Core Team (2016). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

Rue H, Held L (2005). *Gaussian Markov Random Fields: Theory and Applications*. Chapman & Hall/CRC. doi:10.1201/9780203492024.

Särndal CE, Swensson B, Wretman J (2003). *Model Assisted Survey Sampling*. Springer-Verlag. doi:10.1007/978-1-4612-4378-6.

Savitsky TD (2015). "Bayesian Nonparametric Functional Mixture Estimation for Time-Series Data, With Application to Estimation of State Employment Totals." Unpublished manuscript, URL http://arxiv.org/abs/1508.00615.

Savitsky TD, Toth D (2016). "Bayesian Estimation under Informative Sampling." *Electronic Journal of Statistics*, **10**(1), 1677–1708. doi:10.1214/16-ejs1153.

Savitsky TD, Vannucci M (2010). "Spiked Dirichlet Process Priors for Generalized Gaussian Process Models." *Journal of Probability and Statistics*, **2010**(201489), 1–14. doi:10.1155/2010/201489.

Sethuraman J (1994). "A Contructive Definition Of Dirichlet Priors." *Statistica Sinica*, **4**(2), 639–650.

The MathWorks, Inc (2014). *MATLAB – The Language of Technical Computing, Version 8.3.0 (R2014a)*. The MathWorks, Inc., Natick, Massachusetts. URL http://www.mathworks.com/products/matlab/.

Vanhatalo J, Riihimäki J, Hartikainen J, Jylänki P, Tolvanen V, Vehtari A (2013). "**GPstuff**: Bayesian Modeling with Gaussian Processes." *Journal of Machine Learning Research*, **14**(1), 1175–1179.

Wang C, Neal RM (2013). "MCMC methods for Gaussian Process Models Using Fast Approximations for the Likelihood." arXiv:1305.2235 [stat.CO], URL http://arxiv.org/abs/1305.2235.

Affiliation:

Terrance D. Savitsky
Office of Survey Methods Research
U.S. Bureau of Labor Statistics
2 Massachusetts Ave. N.E.
Washington, D.C. 20212, United States of America
E-mail: tds151@gmail.com
URL: http://www.bls.gov/ore/

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
August 2016, Volume 72, Issue 2 *Submitted:* 2014-11-03
doi:10.18637/jss.v072.i02 *Accepted:* 2015-08-07