Appendices

A Data-dependent parameters for Student-T reweighting function

Following (4), the reweighting function G is a product of p Student-T distributions whose location, scale and degrees of freedom are data-dependent [24]:

$$
\begin{aligned}
&G(\mathbf{x}_{t-p:t-1}; D_{tk}, \lambda_G) \\
&= \prod_{i=1}^{p} G_i(x_{t-i}; D_{tki}, \lambda_{Gi}) \\
&= \prod_{i=1}^{p} \mathrm{T}_{2a_{tki}}\left(x_{t-i}; m_{tki}, b_{tki}\frac{V_{tki}+1}{a_{tki}}\right) \qquad (9)\\
\lambda_{Gi} &= (m_{i0}, V_{i0}, a_{i0}, b_{i0}) \\
D_{tki} &= \{x_{t'-i} : z_{t'} = k, 1 \le t' < t\} \\
n_{tki} &= |D_{tki}| \\
\bar{x}_{tki} &= \frac{1}{n_{tki}} \textstyle\sum_{t' \in D_{tki}} x_{t'-i} \qquad (10)\\
V_{tki} &= 1/(V_{i0}^{-1} + n_{tki}) \\
m_{tki} &= V_{tki}(V_{i0}^{-1} m_{i0} + n_{tki}\bar{x}_{tki}) \\
a_{tki} &= a_{i0} + n_{tki}/2 \\
b_{tki} &= b_{k0} + \frac{1}{2}\left(m_{i0}^2 V_{i0}^{-1} + \textstyle\sum_{t'} x_{t'-i}^2 - m_{itk}^2 V_{tki}^{-1}\right).
\end{aligned}
$$

B Markov chain Monte Carlo methods for posterior inference

Here, we provide the details of the MCMC method for posterior simulation from the nonparametric mixture model developed in Section 3. As discussed in the main text, conjugacy of F and π_Θ in (3) means we can analytically marginalize parameters $\{\theta_k^n\}$ when defining the generative process of the TRCRP mixture. The model in Figure 2a therefore becomes:

$$
\begin{aligned}
&\alpha \sim \mathrm{Gamma}(1,1) && (11)\\
&\lambda_G^n \sim H_G^n && n = 1, 2, \ldots, N\\
&\lambda_F^n \sim H_F^n && n = 1, 2, \ldots, N\\
&\mathbf{x}_{-p+1:0}^n := (x_{-p+1}^n, \ldots, x_0^n) && n = 1, 2, \ldots, N\\
&\Pr\left[z_t = k \mid \mathbf{z}_{1:t-1}, \mathbf{x}_{t-p:t-1}^{1:N}, \alpha, \lambda_G^{1:N}\right] && t = 1, 2, \ldots, T\\
&\quad \propto \mathrm{CRP}(k|\alpha, \mathbf{z}_{1:t-1}) \textstyle\prod_{n=1}^N G(\mathbf{x}_{t-p:t-1}^n; D_{tk}^n, \lambda_G^n)\\
&\quad \text{where } D_{tk}^n := \left\{\mathbf{x}_{t'-p:t'-1}^n \mid z_{t'} = k, 1 \le t' < t\right\}\\
&\quad \text{and } k = 1, \ldots, \max(\mathbf{z}_{1:t-1}) + 1\\
&x_t^n \mid \left\{z_t = k, \mathbf{x}_{1:t-1}^{1:N}\right\} \sim \textstyle\int_\theta F(\cdot|\theta)\pi_\Theta(\theta|D_{tk}'^n, \lambda_F^n)\mathrm{d}\theta\\
&\quad \text{where } D_{tk}'^n := \{\mathbf{x}_{t'}^n \mid z_{t'} = k, 1 \le t' < t\}. \\
& && n = 1, 2, \ldots, N
\end{aligned}
$$

The integration of F against $\pi_\Theta(\theta|D_{z_t}'^n)$ in the right hand-side of the final line evaluates to a Student-T distribution as in (9), whose updates given $D_{tz_t}'^n$ and λ_F^n are identical to those in (10) with $i = 0$.

Inference on temporal regime assignments $(z_t|\mathbf{z}_{1:T\setminus t}, \ldots)$. We first describe how to transition $\mathbf{z}_{1:T}$, assuming the collapsed version of the TRCRP (11) with N time series. Note that since the hierarchical prior (5) for structure learning results in $M = \max(\mathbf{c}^{1:N})$ independent TRCRP mixtures (conditioned on the assignment vector), it suffices to describe inference on $\mathbf{z}_{1:T}$ in one of the mixtures (which keeps notation significantly simpler). Given observations $\mathbf{x}_{-p+1:T}^{1:N}$, the joint likelihood of model (11) is:

$$
\begin{aligned}
&P\left(\alpha, \lambda_G^{1:N}, \lambda_F^{1:N}, \mathbf{z}_{1:T}, \mathbf{x}_{1:T}^{1:N} \;;\; \mathbf{x}_{-p+1:0}^{1:N}, p\right)\\
&= \Gamma(\alpha; 1, 1)\left(\prod_{n=1}^N H_G^n(\lambda_G^n)\right)\left(\prod_{n=1}^N H_F^n(\lambda_F^n)\right)\\
&\quad \prod_{t=1}^T \Big[b_t \mathrm{CRP}(z_t \mid \mathbf{z}_{1:t-1}, \alpha)\\
&\quad \prod_{n=1}^N G(\mathbf{x}_{t-p:t-1}^n; D_{tz_t}^n, \lambda_G^n) F(x_t^n \mid D_{tz_t}'^n, \lambda_F^n)\Big] \qquad (12)
\end{aligned}
$$

The normalizer at time t is given by:

$$
\begin{aligned}
&b_t(\mathbf{x}_{1:t-1}^{1:N}, \mathbf{z}_{1:t-1}) \qquad (13)\\
&= \left(\sum_{k=1}^{K_t} \mathrm{CRP}(k|\alpha, \mathbf{z}_{1:t-1}) \prod_{n=1}^N G(\mathbf{x}_{t-p:t-1}^n; D_{tk}^n, \lambda_G^n)\right)^{-1},
\end{aligned}
$$

where $K_t = \max(\mathbf{z}_{1:t-1}) + 1$. Note that the normalizer $b_t(\mathbf{x}_{1:t-1}^{1:N}, \mathbf{z}_{1:t-1})$ ensures the reweighted cluster probabilities sum to one. It will also be convenient to define the predictive density q_t at time t of data $\mathbf{x}_t^{1:N}$, which sums out all possible values of z_t:

$$
\begin{aligned}
&q_t(\mathbf{x}_{1:t}^{1:N}, \mathbf{z}_{1:t-1}) \qquad (14)\\
&= b_t(\mathbf{x}_{1:t-1}^{1:N}, \mathbf{z}_{1:t-1})\left(\sum_{k=1}^{K_t} \mathrm{CRP}(k|\alpha, \mathbf{z}_{1:t-1})\right.\\
&\quad \left.\prod_{n=1}^N G(\mathbf{x}_{t-p:t-1}^n; D_{tk}^n, \lambda_G^n) F(x_t^n \mid D_{tk}'^n, \lambda_F^n)\right).
\end{aligned}
$$

Let the current state of the Markov chain be $(\alpha, \lambda_G^{1:N}, \lambda_F^{1:N}, \mathbf{z}_{1:T})$. We present two algorithms for sampling the latent regimes assignments. Algorithm 1 is a single-site Metropolis-Hastings procedure that targets $(z_t|\mathbf{z}_{1:T\setminus t}, \ldots)$ at each step, where we assume that all data in $\mathbf{x}_{1:T}^{1:N}$ are fully observed. Algorithm 2 is an SMC scheme to block sample $(\mathbf{z}_{1:T}|\ldots)$ using particle learning [9]. Arbitrary observations may be missing, as they are imputed over the course of inference.

Algorithm 1: single-site Metropolis-Hastings. This algorithm proposes $(z_t|\mathbf{z}_{1:T\setminus t}, \dots)$ at each step, assuming fully observed data $\mathbf{x}_{1:T}^{1:N}$. Repeat for $t = 1, 2, \dots, T$:

1. Propose z_t' from the multinomial distribution:

$$\Pr[z_t' = k \mid \mathbf{z}_{1:T\setminus t}, \mathbf{x}^{1:N}, \alpha] \propto \mathrm{CRP}(k|\alpha, \mathbf{z}_{1:T\setminus t}) \prod_{n=1}^{N} G(\mathbf{x}_{t-p:t-1}^{n}; D_{Tk}^{n} \setminus \{x_t^n\}, \lambda_G^n) F(x_t^n \mid D_{Tk}'^{n} \setminus \{x_t^n\}, \lambda_F^n), \tag{15}$$

   for $k \in \mathrm{unique}(\mathbf{z}_{1:T\setminus t}) \cup \{\max(\mathbf{z}_{1:T\setminus t}) + 1\}$.

2. Compute the MH acceptance ratio $r(z_t \to z_t')$, using b_t defined in (13):

$$r(z_t \to z_t') = \frac{\prod_{t'>t} b_{t'}(\mathbf{z}_{1:t'-1\setminus t} \cup z_t', \mathbf{x}_{1:t'-1}^{1:N})}{\prod_{t'>t} b_{t'}(\mathbf{z}_{1:t'}, \mathbf{x}_{1:t'-1}^{1:N})}. \tag{16}$$

3. Set $z_t \leftarrow z_t'$ with probability $\min(1, r)$, otherwise leave z_t unchanged.

Algorithm 2: block sampling with particle-learning. This algorithm block samples $\mathbf{z}_{1:T}$ without any assumptions on missingness of observations. Let o_t^n be the "observation indicator" so that $o_t^n = 1$ if x_t^n is observed, and 0 if it missing ($n = 1, 2, \dots, N$ and $t = 1, 2, \dots, T$). Let $J > 0$ be the number of particles. Since we will be simulating missing values over the course of inference, we superscript all data with j to indicate the inclusion of any imputed values by particle j.

1. Set $w^j \leftarrow 1$ for $j = 1, 2, \dots, J$
2. Repeat for $t = 1, 2, \dots, T$
   2.1. Repeat for $j = 1, 2, \dots, J$
      2.1.1. Sample z_t^j from the multinomial distribution:

$$\Pr[z_t^j = k \mid \mathbf{z}_{1:t-1}^j, \mathbf{x}^{1:N,j}, \alpha] \propto \mathrm{CRP}(k|\alpha, \mathbf{z}_{1:t-1}^j) \prod_{n=1}^{N} G(\mathbf{x}_{t-p:t-1}^{n,j}; D_{tk}^{n,j}, \lambda_G^n) \prod_{n=1}^{N} \left(F(x_t^n \mid D_{tk}'^{n,j}, \lambda_F^n)\right)^{o_t^n}, \tag{17}$$

      for $k = 1, 2, \dots, \max(\mathbf{z}_{1:t-1}^j) + 1$.

      2.1.2. Update particle weight using predictive density q_t defined in (14):

$$w^j \leftarrow w^j q_t \left(\mathbf{x}_{1:t-1}^{1:N,j} \cup \{x_t^n \mid o_t^n = 1\}, \mathbf{z}_{1:t-1}^j\right). \tag{18}$$

      2.1.3. For each n such that $o_t^n = 0$, simulate a value $x_t^{n,j} \sim F(\cdot \mid D_{tz_t^j}'^{n}, \lambda_F^n)$.
   2.2. If resampling criterion met, then:
      2.2.1. Resample $(\mathbf{z}_{1:t}^j, \mathbf{x}_{1:t}^{1:N,j})$ proportionally to w^j, $j = 1, 2, \dots, J$.
      2.2.2. Renormalize weights $w^j \leftarrow w^j / \sum_{j'} w^{j'}$, $j = 1, 2, \dots, J$.
3. Resample $j \sim \mathrm{Categorical}(w^1, \dots, w^J)$ and return $(\mathbf{z}_{1:T}^j, \mathbf{x}_{1:T}^{1:N,j})$.

It is worth discussing the computational trade-offs between MH Algorithm 1 and SMC Algorithm 2. In step 1 of Algorithm 1, (15) is recomputed $K = O(\max(\mathbf{z}_{1:T}))$ times. Each assessment requires $O(Np)$ computations, where the factor of N is the product over the time series, and the factor of p is the cost of assessing G per (4). In step 2, computing the terms $b_{t'}$ in the acceptance ratio (16) requires revisiting $O(T)$ data points. Therefore a single iteration requires $O(TKNp)$ computations, so that the cost of a full sweep over all T time points is $O(T^2KNp)$. Note that it is not necessary to sum over K_t in (13) when computing the $b_{t'}$ terms in (16), since the data in at most two clusters will change when proposing z_t to $z_{t'}$. The sufficient statistics can be updated in constant time using a simple dynamic programming approach.

In practice, we consider several computational approximations that simplify the scaling properties of the single-site MH Algorithm 1. For missing data, rather than evaluate the full model likelihood (12) on imputed data for each $t = 1, \ldots, T$, we instead adopt a "data-dependent" prior, similar to the strategy described by [10] in the context of Bayesian density regression. Namely, letting o_t^n be the indicator for having observed x_t^n, we let the reweighting function G consider only those data points that have actually been observed. Therefore, (4) becomes:

$$G(\mathbf{x}_{t-p:t-1}; D_{tk}, \lambda_G) = \prod_{i=1}^{p} (G_i(x_{t-i}; D_{tki}, \lambda_{Gi}))^{o_{t-i}^n}. \tag{19}$$

Second, note that the MH proposal (15) is very similar to the Gibbs proposal from Algorithm 3 of [25], except we must account for the temporal coupling so that the transition is guaranteed to leave (12) invariant. Empirical evidence suggest that, when using the proposal (15), acceptance ratios center around one. This observation suggests a good initialization strategy for the Markov chain (prior to running the full MH algorithm): run several rounds of step 1 always accepting the proposal $z_t \to z_t'$ without computing (16), which eliminates the additional $O(T)$ factor.

Unlike the MH Algorithm 1, the SMC algorithm (2) with requires $O(KNp)$ to assess (17) in step 2.1.1; the total cost of a complete pass through all T data points (step 2) and all J particles (step 2.1) is therefore $O(JTKNp)$. Note that in SMC, the normalizers b_t need not to be retroactively computed, which is the key overhead of MH. In addition to its linear scaling in T, SMC is able to (i) more tractably handle missing data, and (ii) use a posterior particle filter by sampling from the conditionally optimal proposal distribution in step 2.1.1, resulting in significantly lower variance of the weights [9].

Inference on time series cluster assignments $(c^n | \mathbf{c}^{1:N\setminus n}, \ldots)$. This section describes an MCMC algorithm for sampling the time series cluster assignments when using the hierarchical CRP structure prior (5). For notational simplicity, let $B \subseteq [N]$ and define:

$$L^m(\mathbf{z}_{1:T}, \mathbf{x}_{1:T}^B) = \prod_{t=1}^{T} \Bigg[b_t \mathrm{CRP}(z_t \mid \mathbf{z}_{1:t-1}, \alpha^m) \prod_{n=1}^{N} G(\mathbf{x}_{t-p:t-1}^n; D_{tz_t}^n, \lambda^n) F(x_t^n \mid D_{tz_t}'^n, \lambda_F^n) \Bigg]. \tag{20}$$

The term L^m is a short-hand for the product from $t = 1$ to T in the full model likelihood (12) for a single TRCRP mixture, with latent sequence $\mathbf{z}_{1:T}$, data $\mathbf{x}_{1:T}^B$, and CRP concentration α^m. Second, let $A^m = \{n \mid c^n = m\}$ be the indices of the time series currently assigned to cluster m.

Algorithm 3: Sampling time series cluster assignments. Let the current state of the Markov chain be $(\alpha_0, \mathbf{c}^{1:N}, \alpha^{1:M}, \lambda_G^{1:N}, \lambda_F^{1:N}, \mathbf{z}_{1:T}^{1:M})$ with observations $\mathbf{x}_{1:T}^{1:N}$. This algorithm resamples $(c^n | \mathbf{c}^{1:N\setminus n}, \ldots)$. Repeat for $n = 1, 2, \ldots, N$:

1. If c^n is not a singleton cluster, i.e. $|A^{c^n}| > 1$, then generate a proposal sequence by forward sampling $\mathbf{z}_{1:T}^{M+1}$ from model prior (11), holding the data $\mathbf{x}_{1:T}^n$ fixed at the observed values.
2. If c^n is a singleton, i.e. $|A^{c^n}| = 1$, then re-use the current latent regime sequence by setting $\mathbf{z}_{1:T}^{M+1} = \mathbf{z}_{1:T}^{c^n}$
3. For $m \in \mathrm{unique}(\mathbf{c}^{1:N\setminus n})$, compute
$$p^m = \begin{cases} |A^m| L^m(\mathbf{z}_{1:T}^m, \mathbf{x}_{1:T}^n) & \text{if } c^n \neq m, \\ (|A^m| - 1) L^m(\mathbf{z}_{1:T}^m, \mathbf{x}_{1:T}^n) & \text{if } c^n = m. \end{cases}$$
4. Compute the singleton proposal probability:
$$p^{M+1} = \alpha_0 L_{m+1}(\mathbf{z}_{1:T}^{M+1}, \mathbf{x}_{1:T}^n)$$
5. Sample $c' \sim \mathrm{Categorical}(\{p^m\})$.
6. Compute the MH acceptance ratio
$$r(c^n \to c') = \left(\frac{L^{c'}(\mathbf{z}_{1:T}^{c'}, \mathbf{x}_{1:T}^{A^{c'}} \cup \mathbf{x}_{1:T}^n) L^{c^n}(\mathbf{z}_{1:T}^{c^n}, \mathbf{x}_{1:T}^{A^{c^n}} \setminus \mathbf{x}_{1:T}^n)}{L^{c'}(\mathbf{z}_{1:T}^{c'}, \mathbf{x}_{1:T}^{A^{c'}}) L^{c^n}(\mathbf{z}_{1:T}^{c^n}, \mathbf{x}_{1:T}^{A^{c^n}})} \right) \left(\frac{L^{c^n}(\mathbf{z}_{1:T}^{c^n}, \mathbf{x}_{1:T}^n)}{L^{c'}(\mathbf{z}_{1:T}^{c'}, \mathbf{x}_{1:T}^n)} \right). \tag{21}$$
7. Set $c^n \leftarrow c'$ with probability $\min(1, r)$, else leave c^n unchanged.

By proposing the latent regime singleton from the (conditional) prior in Step 2 of Algorithm 3, transdimensional adjustments such as reversible jump MCMC [14] need not be considered. Second, when computing the MH acceptance ratio (21) in step 6, it is not necessary to recompute all the L^m terms at each iteration. First, writing out the full products (20) results in cancellation of several terms in the numerator and denominator of (21). Second the b_t^m terms that do not cancel contain several duplicated components, which can be reused from one transition to the other.

In practice, we find that a similar heuristic to the one described for Algorithm 1 provides good transitions in the state space, given the similarities between Algorithm 3 and the Gibbs Algorithm 8 from [25].

Inference on model hyperparameters $(\alpha_0, \{\alpha^m\}, \{\lambda_G^n\}, \{\lambda_F^n\} \mid \dots)$. This section describes the empirical Bayes approach [30] for transitioning model hyperparameters, using the "griddy Gibbs" approach from [29]. For each hyperparameter, we construct a grid of 30 data-dependent logarithmically-spaced bins as follows:

Outer CRP concentration

$$\text{grid}(\alpha_0) \quad = \text{logspace}(1/N, N)$$

TRCRP concentration

$$\text{grid}(\alpha^m) \quad = \text{logspace}(1/T, T)$$

Normal-InverseGamma hyperparameters

$$\begin{aligned}
\text{grid}(m_0^n) &= \text{logspace}(\min(\mathbf{x}_{1:T}^n) - 5, \max(\mathbf{x}_{1:T}^n) + 5) \\
\text{grid}(V_0^n) &= \text{logspace}(1/T, T) \\
\text{grid}(a_0^n) &= \text{logspace}(\text{ssqdev}(\mathbf{x}_{1:T}^n)/100, \text{ssqdev}(\mathbf{x}_{1:T}^n)) \\
\text{grid}(b_0^n) &= \text{logspace}(1, T).
\end{aligned}$$

Grids for the Normal-InverseGamma hyperparameters apply to both λ_F $(n = 1, 2, \dots, N)$ and λ_G (windows $i = 1, 2 \dots, p$). We cycle through the grid points of each hyperparameter, and assess the conditional likelihood at each bin using (6). We find that this method is both computationally reasonable and finds good hyperparameter settings. However, alternative approaches based on slice sampling offer a promising alternative to achieve fully Bayesian inference over hyperparameters.

C Experimental Methods

This section describes the quantitative experimental methods used for forecasting, clustering, and imputation pipelines in Section 5. Access to experimental pipeline code is available upon request.

C.1 Flu forecasting

The full CDC flu datasets used in this paper are available at https://github.com/GaloisInc/ppaml-cp7/tree/master/data. Flu populations were constructed from the following csv files: USA-flu.csv, USA-tweets.csv, and USA-weather.csv. In each of US Regions 1 through 10, we held out data from weeks 2014.40 through 2015.20, and produced forecasts with a 10 week horizon on a rolling basis. Tweet and minimum temperature covariates were used. More precisely, for a region r (such as US Region 10) a forecaster F for week t extending h weeks into the future is a function:

$$F_{r,t,h} : \left\{ \mathbf{x}_{1:t-2}^{\text{flu},r}, \mathbf{x}_{1:t}^{\text{cov},r} \right\} \mapsto \left\{ \mathbf{x}_{t:t+h}^{\text{flu},r} \right\}. \tag{22}$$

The forecastors iterated over regions $r = 1, 2, \dots, 10$, weeks $t = 2014.40, 2014.41, \dots, 2015.20$, and horizons $h = 1, 2, \dots, 10$. Note that the two week delay in the latest flu data is expressed by only having data up to $t-2$ when forecasting at week t. Second, $\mathbf{x}^{\text{cov}}$ contains arbitrary missing values (see for example the tweets time series from Figure 2b). When forecasting, covariate values are only available up to the current week t, not the entire course of the forecast horizon. Nine forecasting methods were used in the paper, shown in Figure 6. Below are further details on each forecaster:

Constant. This method returns a constant prediction based on the most recently observed flu value x_{t-2}^{flu} over the entire course of the horizon.

Linear extrapolation. This method fits a straight line through the three most recently observed flu values, $\mathbf{x}_{t-4:t-2}^{\text{flu}}$, and returns predictions by extrapolating the line for h weeks.

GP (SE+PER+WN). This method is a Gaussian process whose covariance kernel is a sum of squared exponential, periodic, and white noise components. Hyperparameter inference was conducted using the open source implementation from the Venture platform [34; https://github.com/probcomp/Venturecxx]. MH sampling on data-dependent hyperparameter grids were run for a burn-in period of 10000 iterations. Predictions were obtained by drawing 500 independent curves from the posterior predictive distribution, evaluated jointly at the forecast weeks.

GP (SE×PER+WN). Identical to above, except to using a covariance kernel with a product of squared exponential and periodic components, plus white noise. The change in covariance kernel resulted in little quantitative and qualitative differences.

Facebook Prophet. We used the open-source python implementation of Facebook Prophet [36; https://facebook.github.io/prophet]. We specified the data sampling rate as weekly. The method requires

no additional specification or tuning. The predictor returns point estimates, as well as upper and lower confidence intervals, at the held-out weeks.

Seasonal ARIMA. We used the R implementation of seasonal ARIMA from the `forecast` package [17; https://cran.r-project.org/web/packages/forecast]. The model is parameterized as ARIMA$(p,d,q)(P,D,Q)_m$, where p is the non-seasonal AR order, d is the non-seasonal differencing, q is the non-seasonal MA order, P is the seasonal AR order, D is the seasonal differencing, Q is the seasonal MA order, and m is the sampling frequency per period. For each of the 10 flu seasons, we used `auto.arima` to perform model selection. We manually specified the weekly sampling rate by setting $m = 52$, and set $D = 1$ to specify 1 flu season per year. The program optimize all other parameters using non-stepwise grid search, which is significantly slower to fit than stepwise search, but is both more extensive and more appropriate for data with seasonal behavior (according to the package documentation). While `auto.arima` can in principle support covariate data using the `xreg` parameter, we were unable to successfully use `xreg` due to missing data in the matrix of external regressors (tweets and weather) at the held-out weeks. The predictor returns point estimates, as well as upper and lower confidence intervals, at the held-out weeks.

Multi-output GP This method is a single-input (time) multiple-output (flu, tweets, and weather data) Gaussian process. We used the the open source MATLAB implementation of sparse convolved Gaussian process for multi-output regression from the `multigp` package [3; https://github.com/SheffieldML/multigp]. We used the following configuration options:

i `multigpOptions('ftc');`
ii `options.kernType='ggwhite';`
iii `options.optimizer='scg';`
iv `options.nlf=1,`

to specify (i) full estimation without running likelihood approximations; (ii) a Gaussian-Gaussian kernel with white noise; (iii) scaled conjugate gradient optimization; and (iv) one latent function. Moreover, the `options.bias` and `options.scale` parameters were initialized to their empirical values from the training set. Optimization was run until convergence for all forecastors. This method is the only baseline which can handle arbitrary patterns of missing data, thereby making use of the weather and tweet signals when forecasting predictions at time t. However, the absence of a periodic kernel in the convolved GP implementation made it difficult to capture the seasonal dynamics. Predictions were obtained by sampling 500 independent normal random variables from the posterior predictive distribution evaluated at the forecast weeks.

HDP-HSMM. This method is the hierarchical Dirichlet process semi-Markov model; experiments were run using the open-source python package `pyhsmm` [19; https://github.com/mattjj/pyhsmm]. While the HDP-HSMM cannot handle missing values in the training data, it can handle missing data over the course of the prediction horizon. Therefore, flu and weather time series were modeled jointly, leaving out the tweets. We used the `WeakLimitHDPHSMM` model, with a Poisson duration distribution and Gaussian observation distribution. Default configurations of all hyperparameters of these distributions and the HDP-HSMM concentration were taken from examples made available by the authors. MCMC inference with 1000 steps of burn-in was used. Predictions were obtained by drawing 100 independent curves from the posterior predictive evaluated at the forecast weeks.

Univariate TRCRP mixture. This method only considered the flu time series using model (2). We used a window size of $p = 10$ weeks, and $S = 64$ parallel MCMC runs with a burn-in period of 5000 iterations. Predictions were obtained by drawing 500 independent curves from the posterior predictive distribution evaluated at the forecast weeks.

Multivariate TRCRP mixture. This method considered flu, weather and tweet time series using the model in Figure 2a. We used a window size of $p = 10$ weeks, and $S = 64$ parallel MCMC runs with a burn-in period of 5000 iterations. Missing covariate data was handled using the approximation given in (19). Using the hierarchical structure prior (5) resulted in little to no quantitative difference. The three time series are dependent, which was reflected in their posterior dependence probability (7) being 1 across all 64 independent chains. Predictions were obtained by sampling 500 independent curves from the posterior predictive distribution evaluated at the forecast weeks. An open-source implementation of the method used in this paper is at https://github.com:probcomp/trcrpm.

C.2 Flu imputation

We constructed a single population of 10 flu time series for US Regions 1 through 10. Missing data was dropped independently in each time series by removing consecutive windows of length 10 at a rate of 5%. The full and dropped datasets used for benchmarking are shown in Figure 8. Below are further on details on each of the five imputation methods:

Mean imputation. This method returns the per-series mean as the imputed value for each data point.

Linear interpolation. This method constructs a straight line between every pair of time points $t_1 < t_2$ which have at least one missing observation between them. The interpolation method used was `pandas.Series.interpolate` from the python `pandas` package at https://pandas.pydata.org.

Cubic interpolation. The cubic interpolation routine used was `scipy.interpolate.interp1d` from the python `scipy` package at https://scipy.org.

Amelia II. This method uses the R package `amelia` [16; http://cran.r-project.org/web/packages/Amelia] for multiple imputation. We used 100 samples per missing data point. Imputation errors were averaged over the multiple imputations.

Multivariate TRCRP mixture. A window of $p = 10$ weeks was used, with $S = 64$ parallel MCMC runs and a burn-in period of 5000 iterations. 100 predictive samples from each of the chains were obtained using (8), and imputation errors were averaged over the multiple imputations. Joint imputations of Regions 1 through 10 are are shown Figure 8.

C.3 Sensitivity of imputation performance to the TRCRP mixture window size

We further studied how imputation performance of the TRCRP mixture varied as we changed the window size p. Figure 7 shows the outcome of this sensitivity analysis. In all cases, the sampler was run for a burn-in of 5000 iterations with $S = 16$ chains. While imputation is generally not highly sensitive to p, median imputation values degrades slightly with increasing p and the variance of imputation errors increases. (At higher p, the MCMC chains need a significantly higher number of iterations to mix well than at lower p.)

The reason that small p works well for jointly imputing the 10 time series in Figure 8 is that the multivariate TRCRP mixture shares statistical strength across time series. Namely, when imputing a missing value $x_t^{n_0}$ at time t for time series n_0, the relevant variables for predicting the hidden state z_t are (i) the history $\mathbf{x}_{t-p:t-1}^{n}$ of the current time series; and (ii) values $\{x_t^n \mid n_0 \neq n\}$ of other time series at time t. The latter effect is the dominant one in this imputation problem, leading to less sensitivity to p than might be expected.

C.4 Clustering GDP time series

The clustering results from Figure 4 were obtained by using a TRCRP with a window of $p = 5$ years. The nine clusters that are shown were obtained by averaging dependence probabilities over $S = 60$ posterior samples (using a burn-in of 5000 iterations), and extracting groups of variables whose dependence probabilities (7) exceeded 80%. All time series in Figure 4 are linearly rescaled to $[0, 1]$ for plotting purposes only.

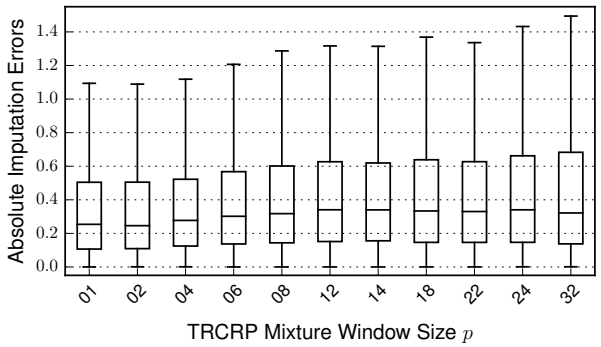
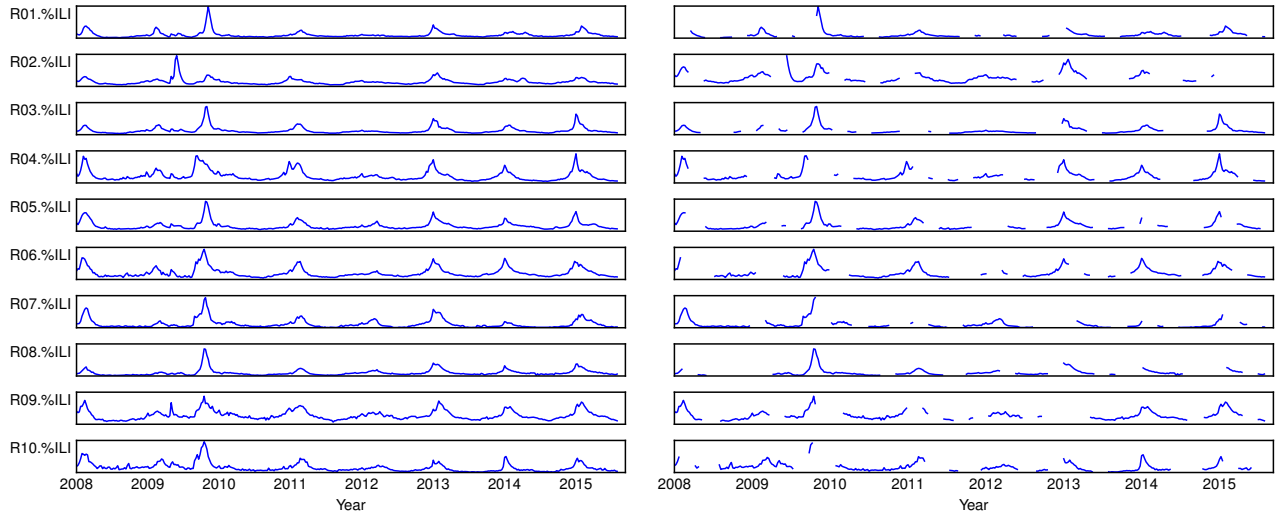


Figure 7: Sensitivity of imputation performance to TRCRP window size p.

While clustering is an unsupervised task that is challenging to evaluate quantitatively (especially for real-world data, where there is no "ground-truth"), qualitative comparisons to k-medoids clustering with the dynamic time warping metric on the same GDP time series are shown and discussed in Figure 9.

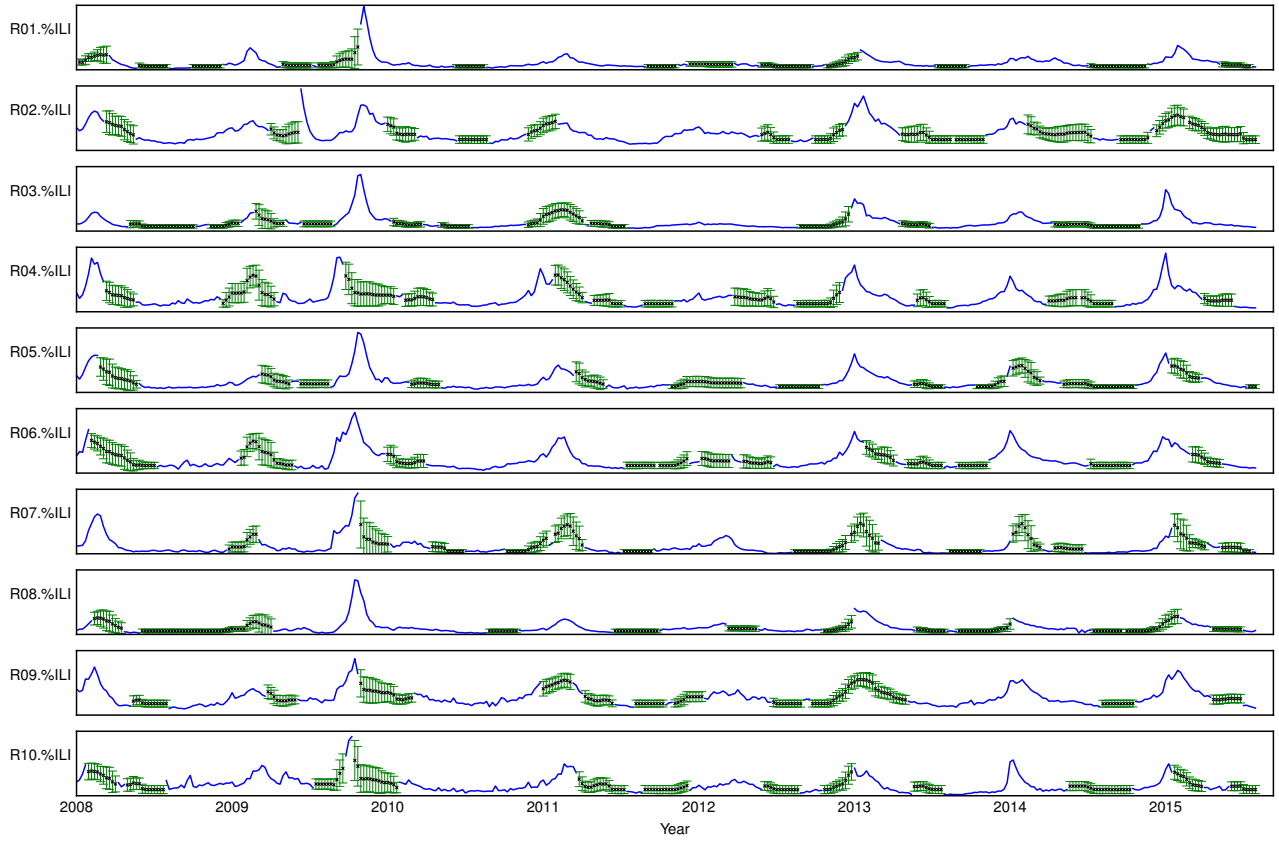
C.5 Expanded results on clustering cell phone subscription time series

In addition to clustering GDP series from Figure 4, we applied the TRCRP prior with hierarchical extension (5) to cluster historical cell phone subscription data. The outcome of the clustering is shown in Figure 10, where we show all 170 time series in the left most figure, along with three representative clusters from one posterior sample. Each cluster corresponds to countries whose change point in cell phone subscribers from zero to non-zero fell in a distinct window: 1985-1995 in cluster 1, 1995-2000 in cluster 2, and 2000-2005 in cluster 3. We also compare renderings of the the pairwise dependence probability matrix with the pairwise cross-correlation matrix. Refer to the caption of Figure 10 for additional details.



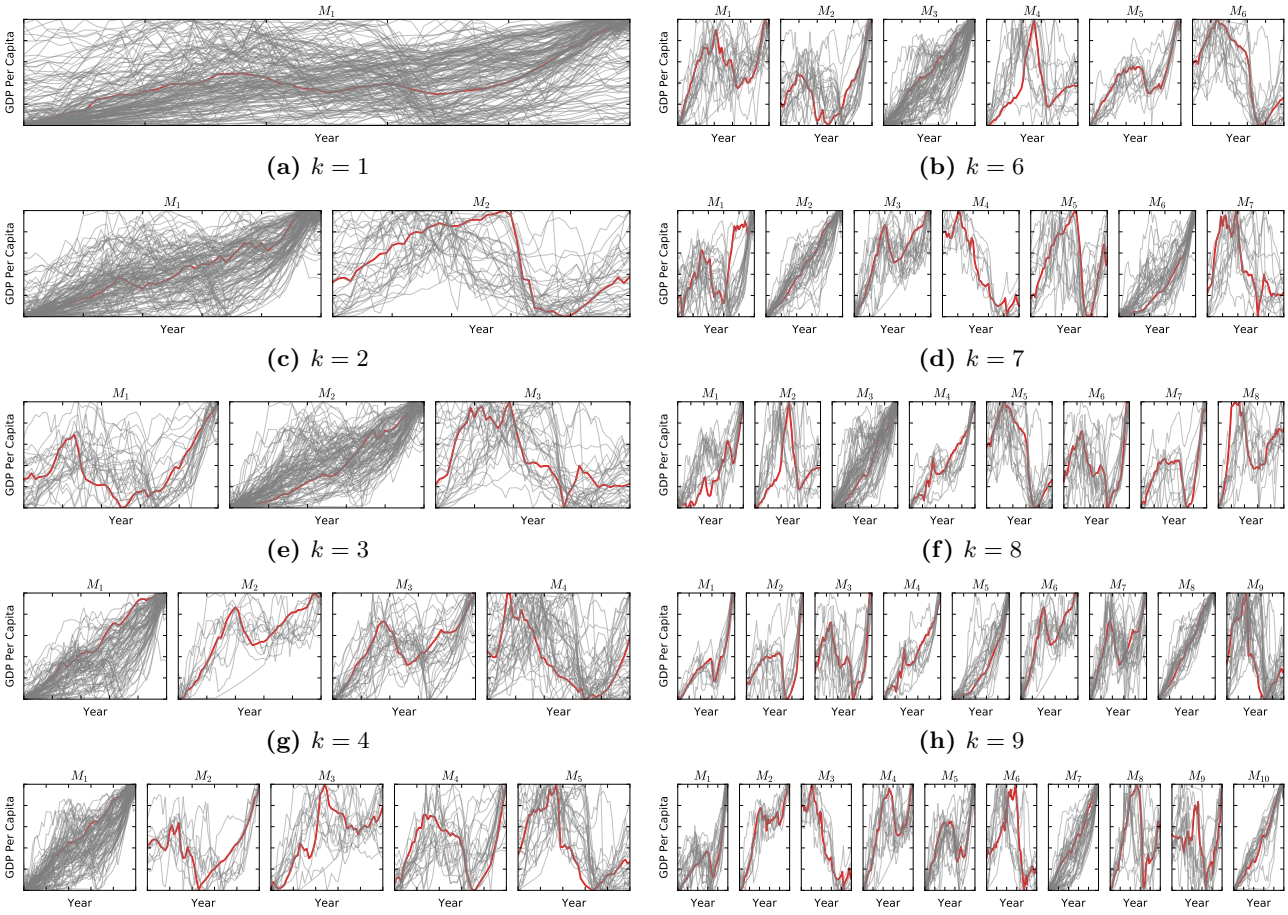
(a) Original flu time series

(b) Time series after dropping data



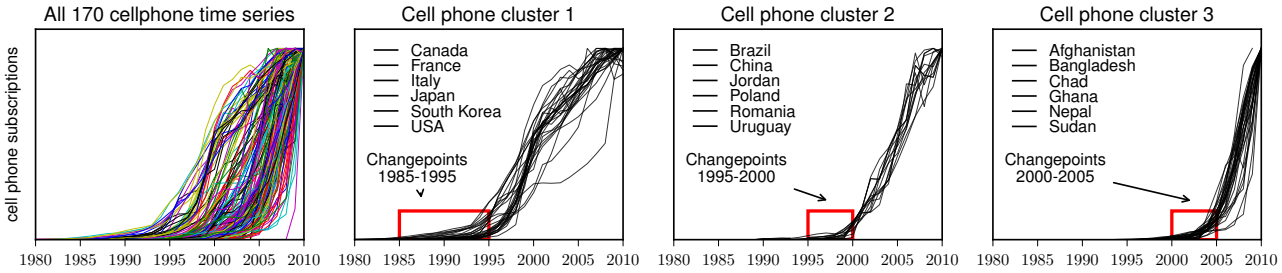
(c) Jointly imputed time series using TRCRP mixture ($p = 10$)

Figure 8: Full, missing, and imputed flu time series over eight years in US Regions 1 through 10.

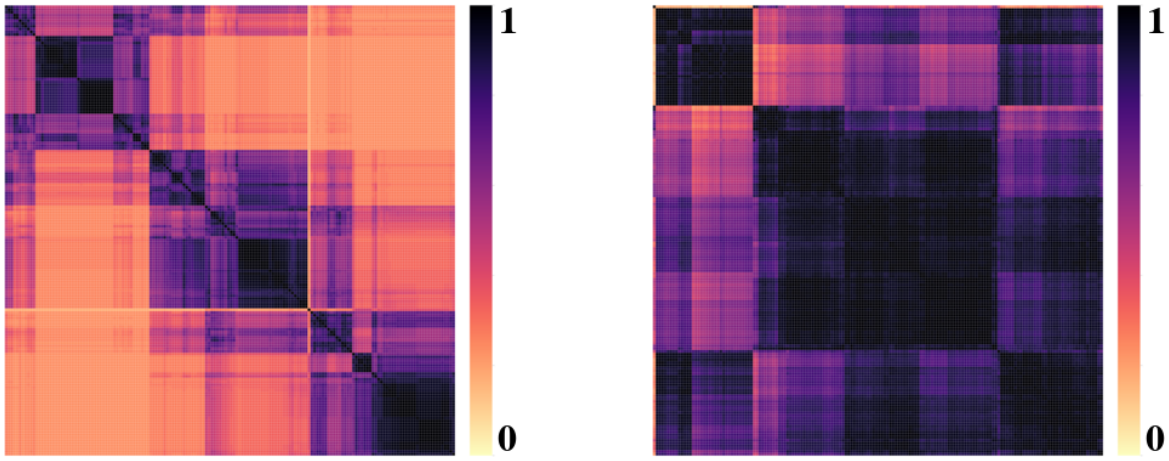


(a) $k = 1$ **(b)** $k = 6$ **(c)** $k = 2$ **(d)** $k = 7$ **(e)** $k = 3$ **(f)** $k = 8$ **(g)** $k = 4$ **(h)** $k = 9$ **(i)** $k = 5$ **(j)** $k = 10$

Figure 9: Outputs of k-medoids clustering on the GDP per capita time series for all 170 countries in the Gapminder dataset, with $k = 1, 2, \ldots, 10$. Distances are computed using the dynamic time warping (DTW) metric, a common similarity measure between a pair of time series [6]. For each k, we randomly initialized the medoids and ran the algorithm to convergence (medoids are shown in red, and time series assigned to that medoid in gray). Using k-medoids requires hand-tuning the number of latent clusters k, whereas the proposed method (whose posterior clustering is shown in Figure 4 of the main text), places a non-parametric Bayesian prior over this parameter. Moreover, when compared to the clusters detected by the proposed method, those detected by k-medoids with DTW appear qualitatively less distinct, and have more repetitive and duplicated temporal patterns (especially apparent at higher k). Finally, k-medoids outputs a fixed cluster assignment for each time series in the population; these assignments are sensitive to the random initialization and cannot be aggregated in a principled way. In contrast, inference in the proposed method assigns probabilistic cluster assignments that can be averaged coherently using (7) to express posterior uncertainty.



(a) Three posterior clusters in the TRCRP mixture correspond to three non-overlapping change point windows.



(b) Pairwise dependence probability heatmap

(c) Pairwise cross-correlation heatmap

Figure 10: Discovering changepoint patterns in cell phone subscriptions for 170 countries in the Gapminder dataset. (a) The three clusters (extracted from one posterior sample) correspond to three regimes each with non-overlapping change point windows, annotated by red boxes. The representative countries in each cluster have similar adoption times of cell phone technology, a feature which differs across the clusters. (b) and (c) The matrix of dependence probabilities (averaged over 60 posterior samples using (7)) and the matrix of pairwise cross-correlations (bottom) between all pairs 170 time series. Each row and column is a time series, and the color of a cell (a value between 0,1) indicates the posterior dependence probability, resp. cross-correlation coefficient (significant at the 0.05 level with Bonferroni correction). The TRCRP mixture detects more refined dependence structures than those captured by linear statistics.