Time Series Decomposition Based on Markov Random Fields

Bowen Xiao

May 16, 2018

Setup

```
library(rstan)
library(loo)
library(forecast)
```

Rest of the paper

Intrduction

The project is about Markov random fields, which is a locally adaptive nonparametric curve fitting method that operates within a fully Bayesian framework. The model assumes that each data point is generated independently with some parametirc models, and the parameters are follow a Markov random fields model, which could, according to the paper, provide a combination of local adaptation and global control. Specificly, after removing seasonal component, my trend model used for forecasting looks like this:

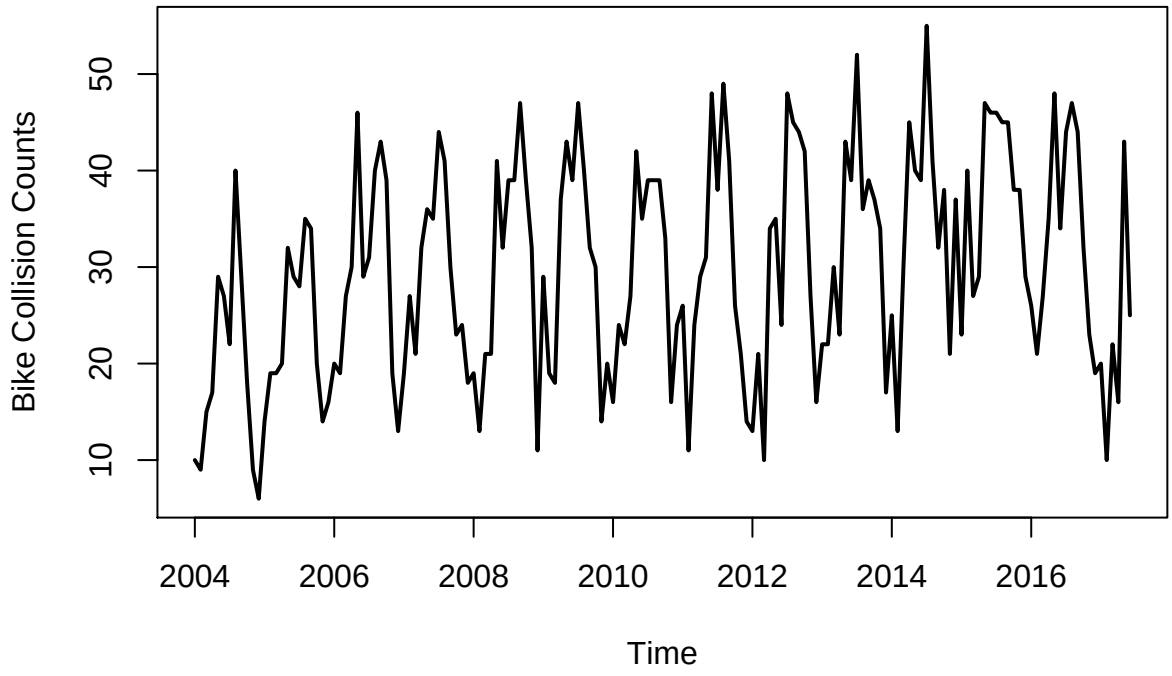
$$
\begin{aligned}
y_j &\sim normal(\theta_j, \sigma); \\
\sigma &\sim exponential(1) \\
\theta_j &= \rho^1 \theta_{j-1} + \delta_{j-1}; \\
\theta_1 &= 5 * sd(y) * \theta_0 + \bar{y}; \\
\theta_0 &\sim normal(0, 1); \\
\rho^1 &\sim beta(2, 2) \\
\delta_j &= \rho^2 \delta_{j-1} + \delta^1_{j-1}; \\
\delta^1_j &\sim normal(0, 1); \\
\delta_1 &\sim normal(0, 1); \\
\rho^2 &\sim beta(2, 2)
\end{aligned}
$$

where the trace of θ is trend component.

What I want to get in time series decomposition is a nonparametric trend component, which is the main difference with classical methods, like ARIMA. In other word, I will not assume the parametric form of the trend component, like linear trend. Furthermore, I compare the results with some classical parametric techniques, like ARIMA, and machine learning techniques, like RNN/LSTM, with regrard to forecasting. Metrics for model evaluation and comparison is MSE.

Data Description

I apply GMRF to my time series data. Specificly, I have data of bike collision records in downtown Seattle from 01/2004 to 06/2017 and I summarize them by month, so that I have 162 bike collision counts, which is shown as following. I split the data into train set (the first 150 points) and forecasting set (the last 12 points).



Time series decomposition

I apply GMRF fitting into time series decomposition and forecasting. Firstly, I use GMRF to fit a smoothing line. I split the line by bandwidth of 12 (since I know the period is 12 month), and avarage the difference with the mean in each piece. So that I get the seasonal component. Secondly, I minus the raw data with seasonal component and do another GMRF fitting on it, which turns out to be the trend component. For the first fitting, y is Poisson distributed because it is count variable, but for the second fitting I use normal distribution.

I try three priors in first fitting: Gaussian prior, Laplace prior and Horseshoe prior. There is no issue of divergence.

```
##
## Divergences:

## 0 of 2500 iterations ended with a divergence.

##
## Tree depth:

## 0 of 2500 iterations saturated the maximum tree depth of 12.
```

```
##
## Energy:

## E-BFMI indicated no pathological behavior.

##
## Divergences:

## 0 of 2500 iterations ended with a divergence.

##
## Tree depth:

## 0 of 2500 iterations saturated the maximum tree depth of 12.

##
## Energy:

## E-BFMI indicated no pathological behavior.

##
## Divergences:

## 0 of 2500 iterations ended with a divergence.

##
## Tree depth:

## 0 of 2500 iterations saturated the maximum tree depth of 12.

##
## Energy:

## E-BFMI indicated no pathological behavior.
```

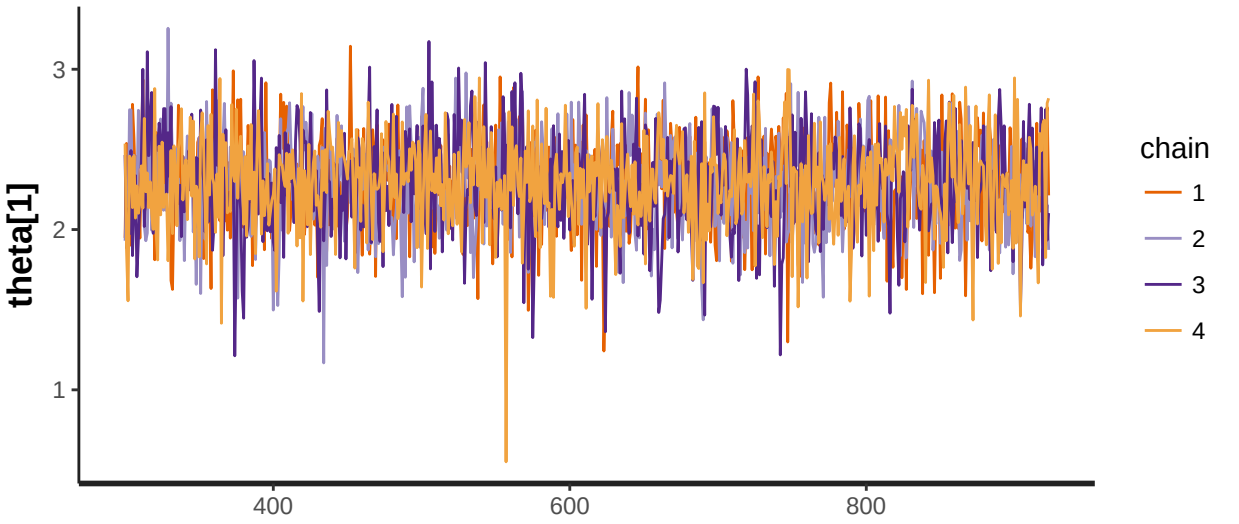
And I compare them based on LOO-PSIS-CV and WAIC.

	elpd_diff	elpd_loo	se_elpd_loo	p_loo	se_p_loo	looic	se_looic
loo2	0.000000	-531.5500	3.666170	96.01435	2.075922	1063.100	7.332340
loo3	-2.745540	-534.2955	12.194588	64.80410	5.673001	1068.591	24.389175
loo1	-9.072317	-540.6223	3.797943	103.95685	1.712851	1081.245	7.595886

	elpd_diff	elpd_waic	se_elpd_waic	p_waic	se_p_waic	waic	se_waic
waic2	0.000000	-498.7724	2.954513	63.23676	0.9022137	997.5448	5.909027
waic1	-5.209681	-503.9821	3.091133	67.31663	0.5741754	1007.9642	6.182267
waic3	-29.672419	-528.4448	11.788323	58.95339	5.2504230	1056.8897	23.576646

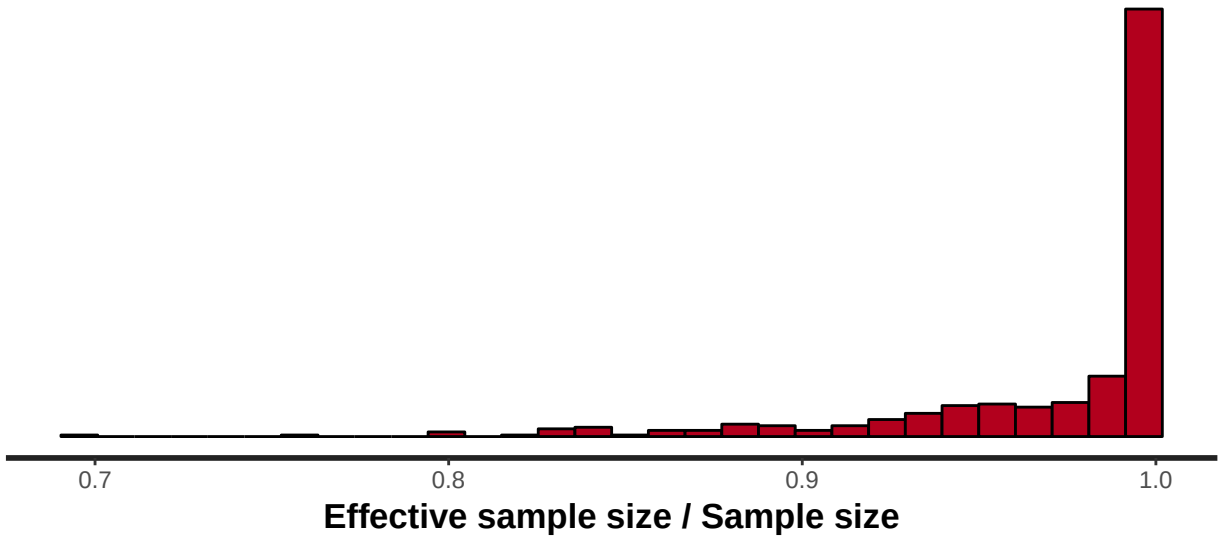
Both LOO-PSIS-CV and WAIC indicate model based on Laplace prior is the best. Thus, I will go on with this model.

The trace of $theta_1$ in the sampling can be shown as following.

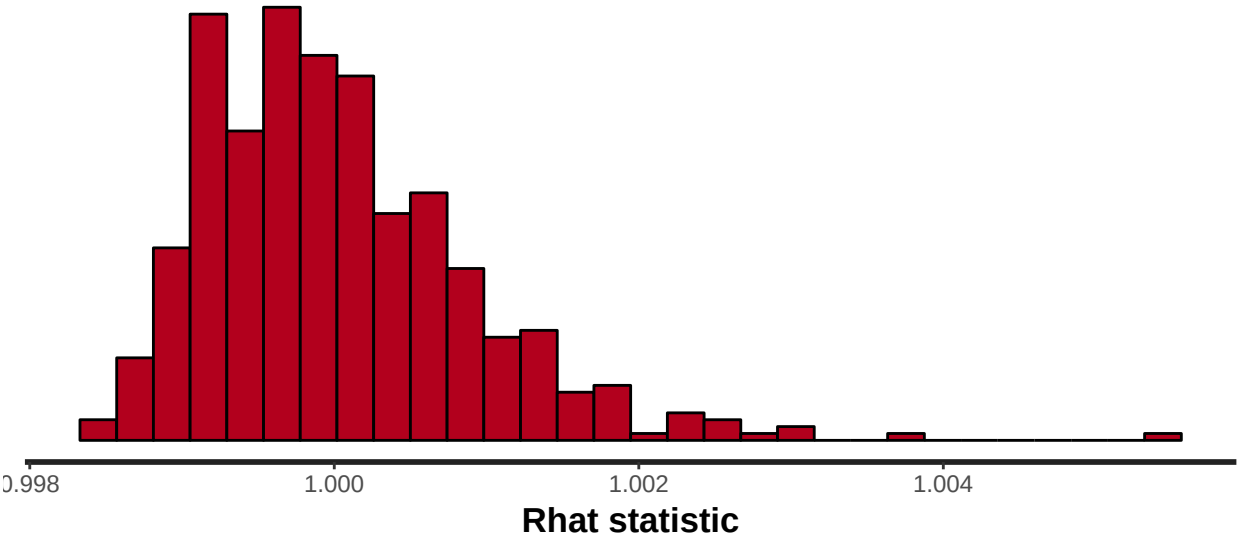


The overlapping lines show that the samples from 4 different chains are comming from one common distribution, or that, there is no violation for $theta_1$. The conclusion is also true for other parameters. And as we can see in the following, all the $\hat{R}s$ are close to 1. Effective sample size is also large enough.

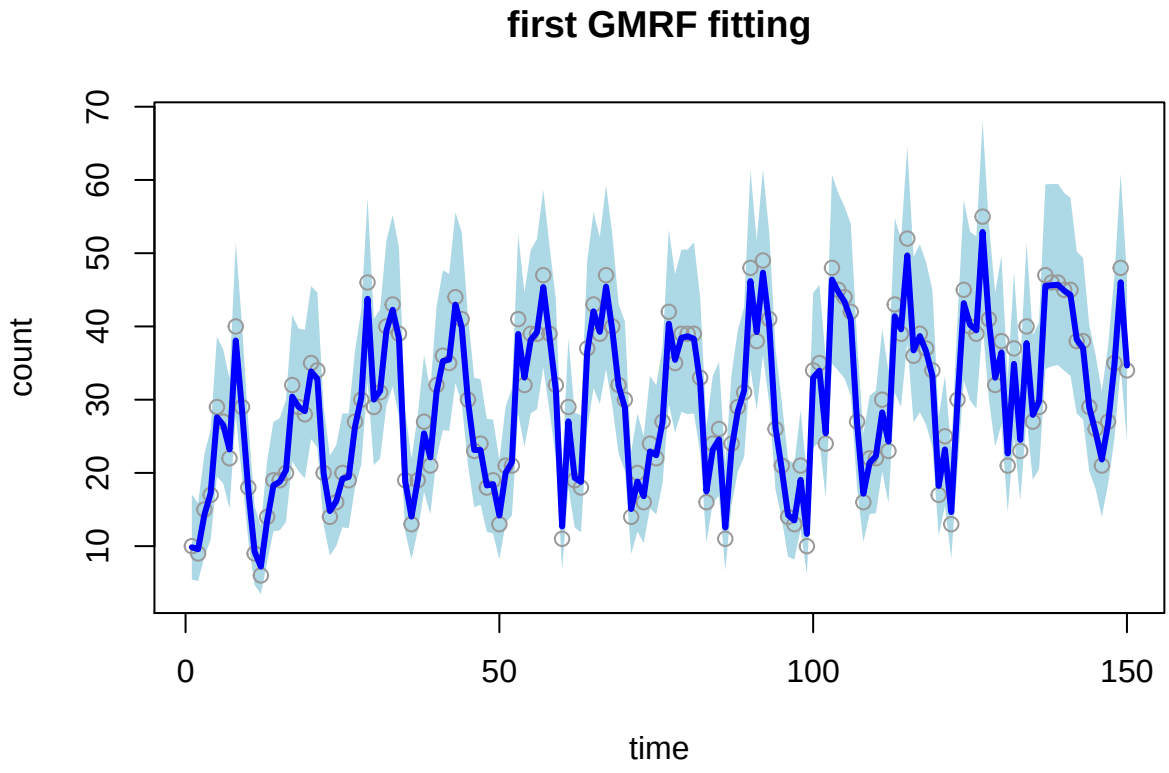
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



95% credible interval of θ can be shown as following.



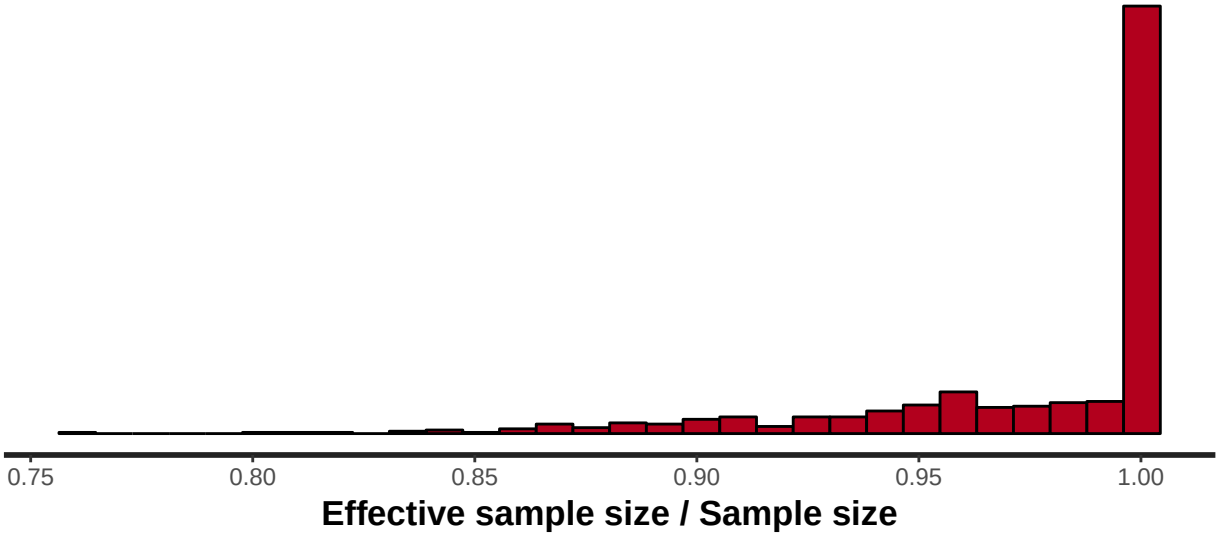
In the second fitting, there is no divergence issue either.

```
##
## Divergences:
## 0 of 2500 iterations ended with a divergence.
```

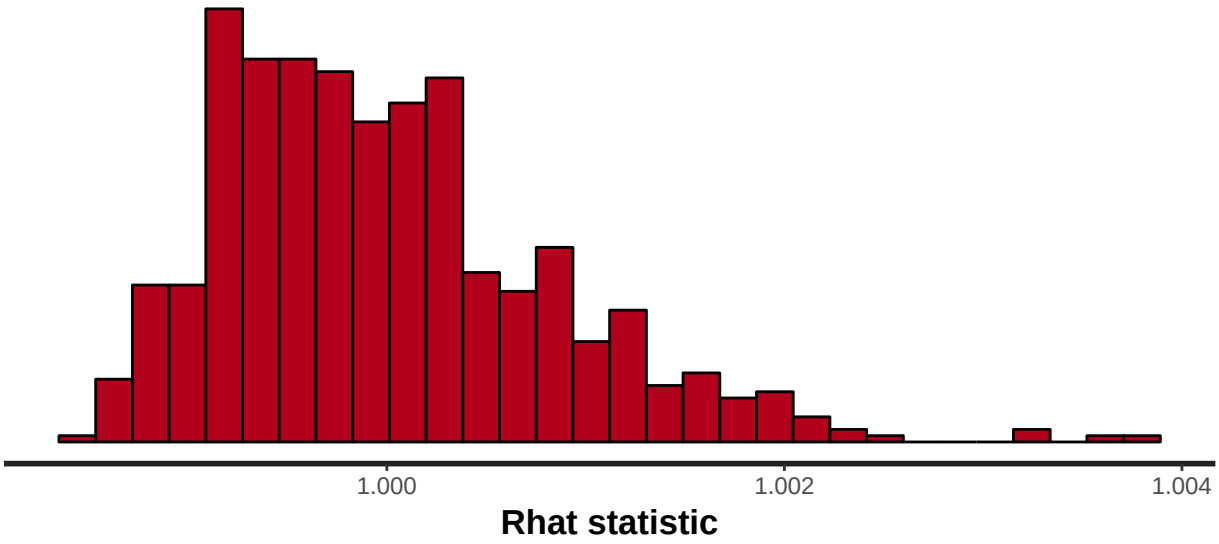
```
##
## Tree depth:
## 0 of 2500 iterations saturated the maximum tree depth of 20.
##
## Energy:
## E-BFMI indicated no pathological behavior.
```

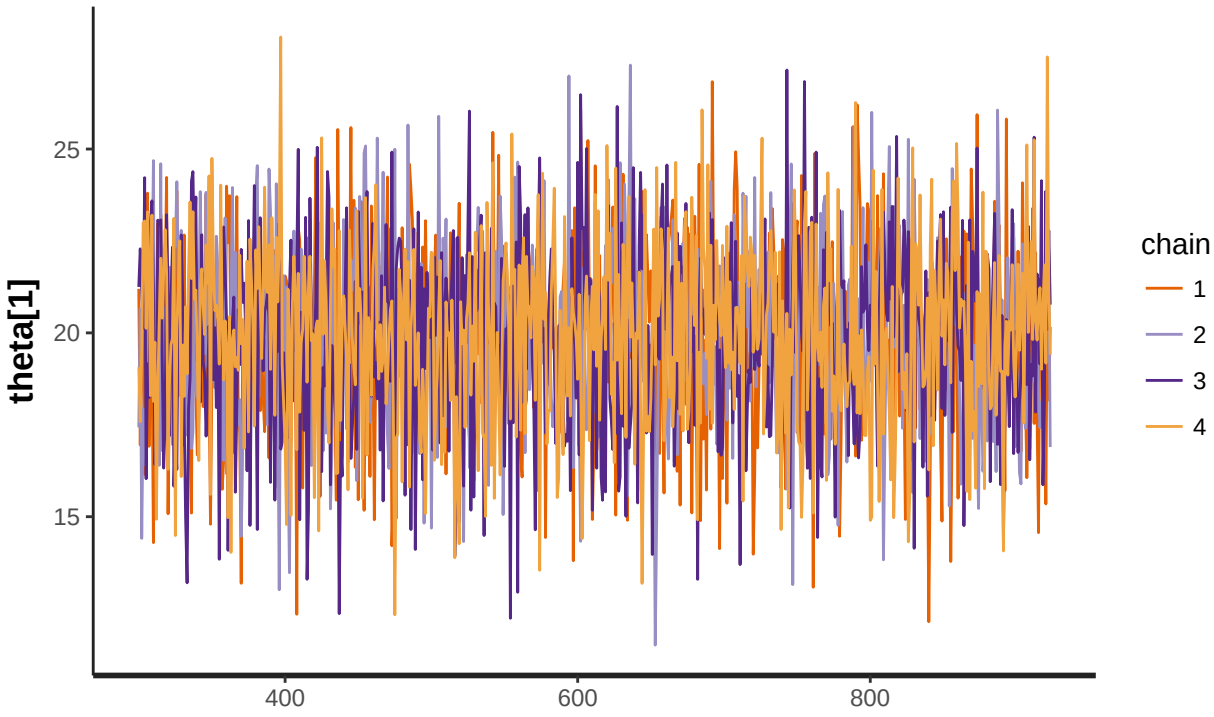
Similarly, here are the results of second fitting.

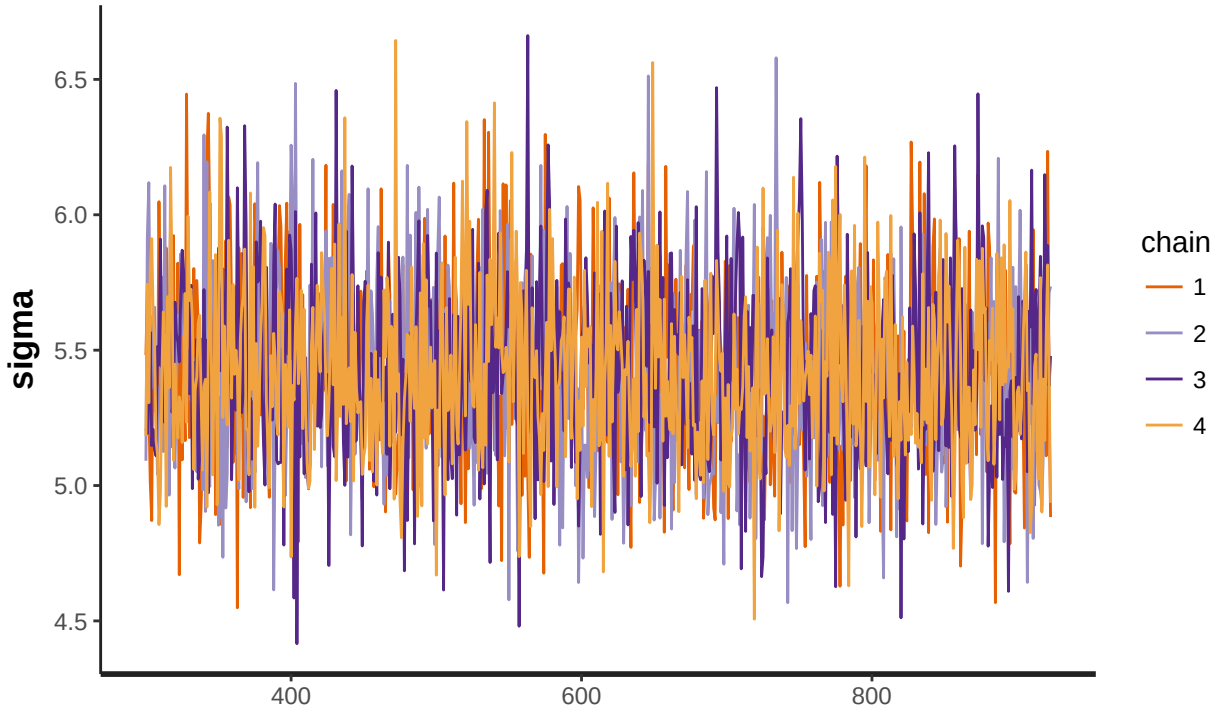
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

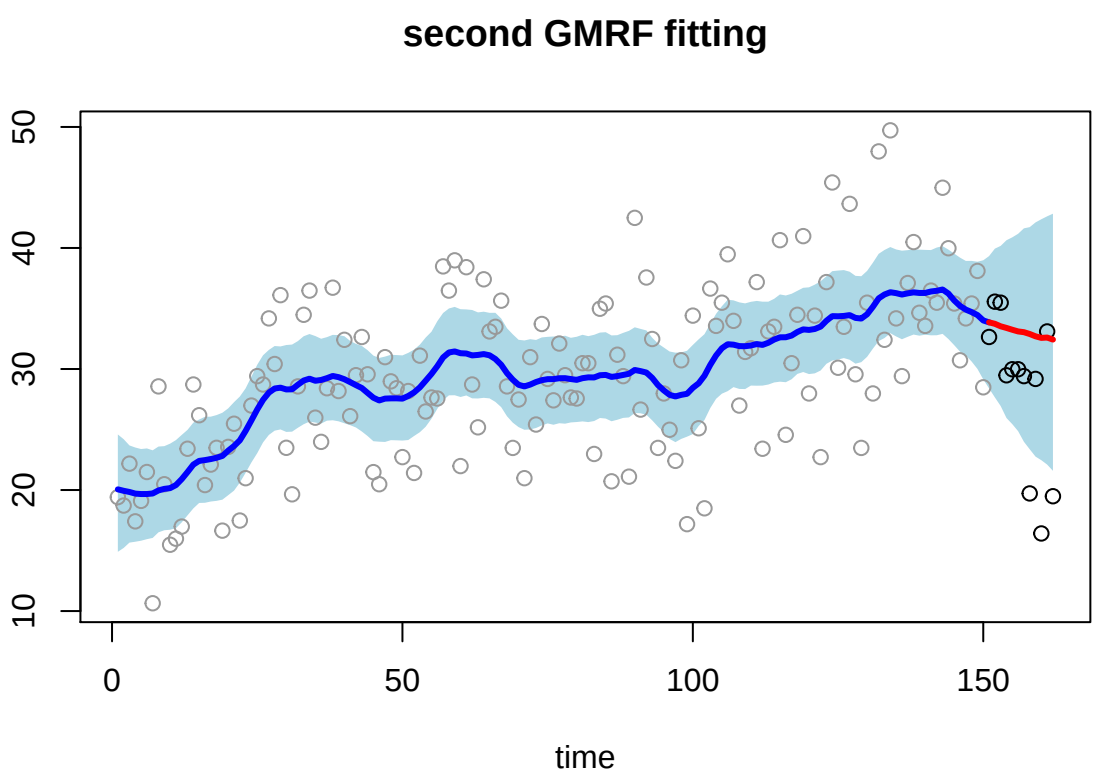



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



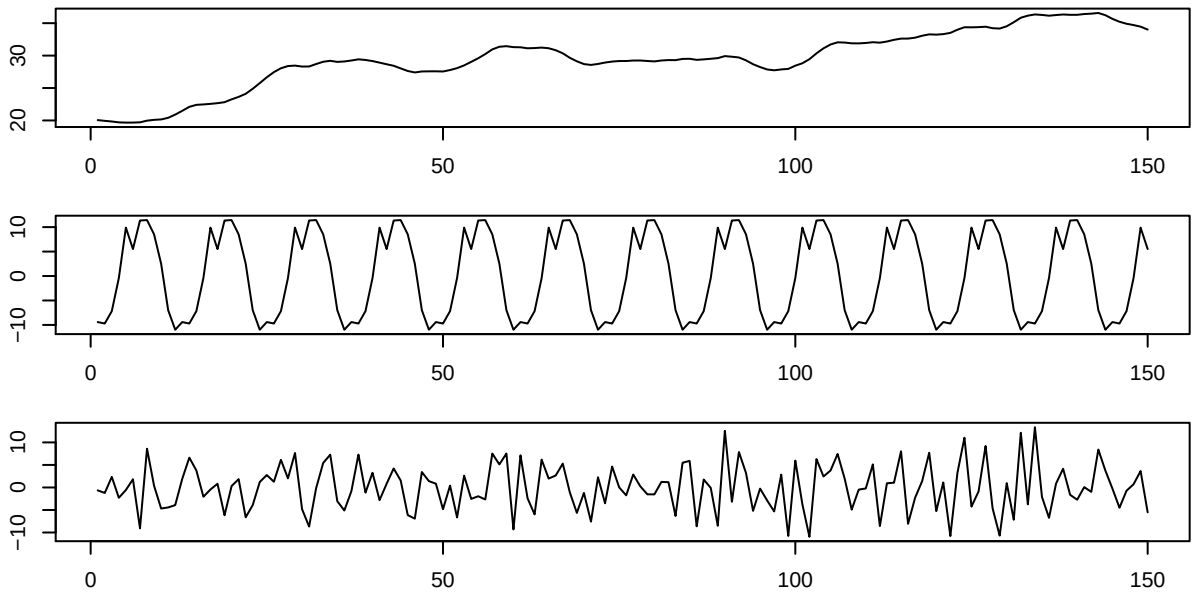

theta[1]
25
20
15
400
600
800
chain
1
2
3
4


sigma
6.5
6.0
5.5
5.0
4.5
400
600
800
chain
1
2
3
4

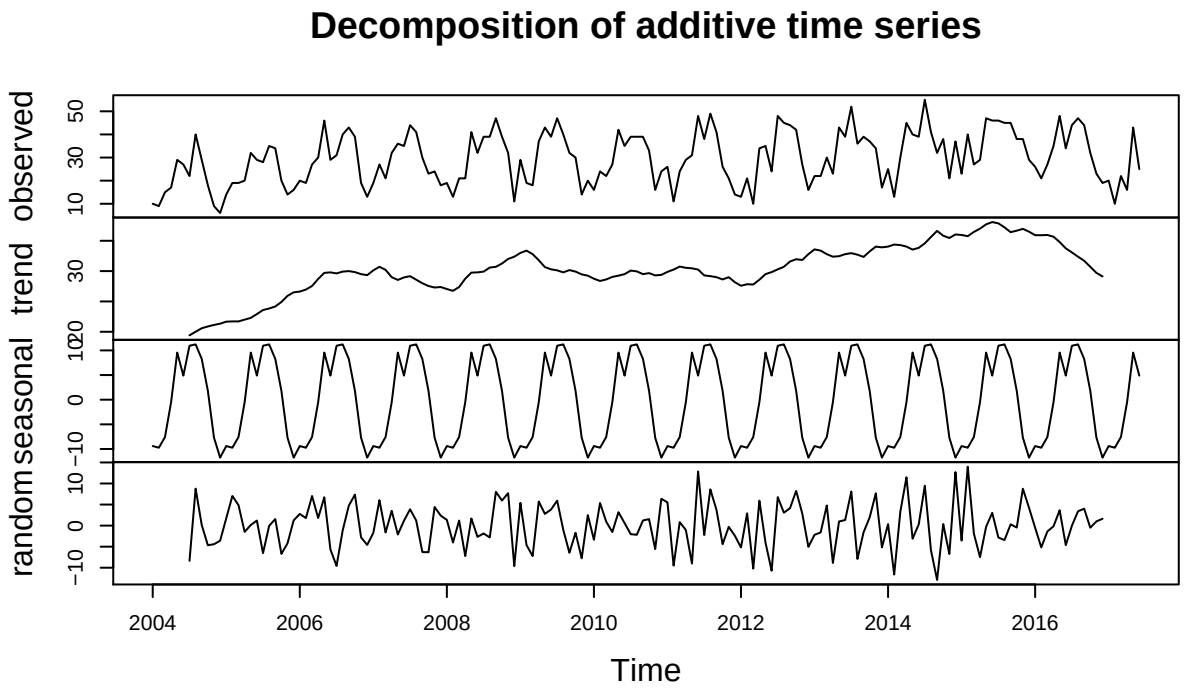


where the last 12 θs are simulated and will be used in forecasting.

The above blue line is my trend component. After second fitting, my decomposition will look like the following.

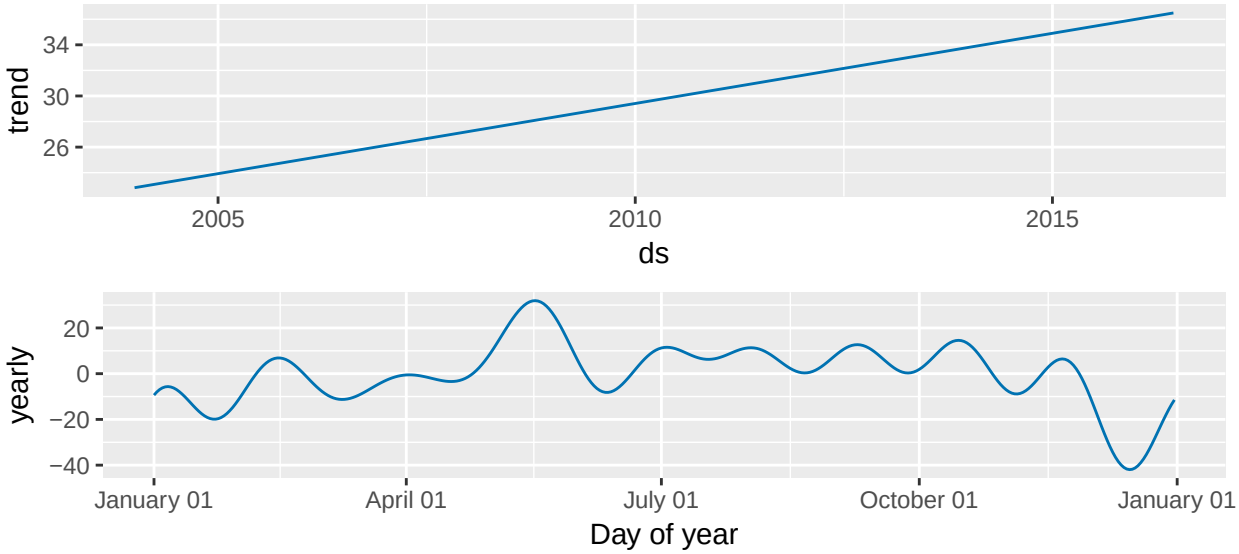


As a comparison, decomposition based on ARIMA looks like the following. Generally speaking, the seasonal part seems to be exactly the same, but my trend line is more smoothing.



And decomposition based on `Prophet` looks like the following.

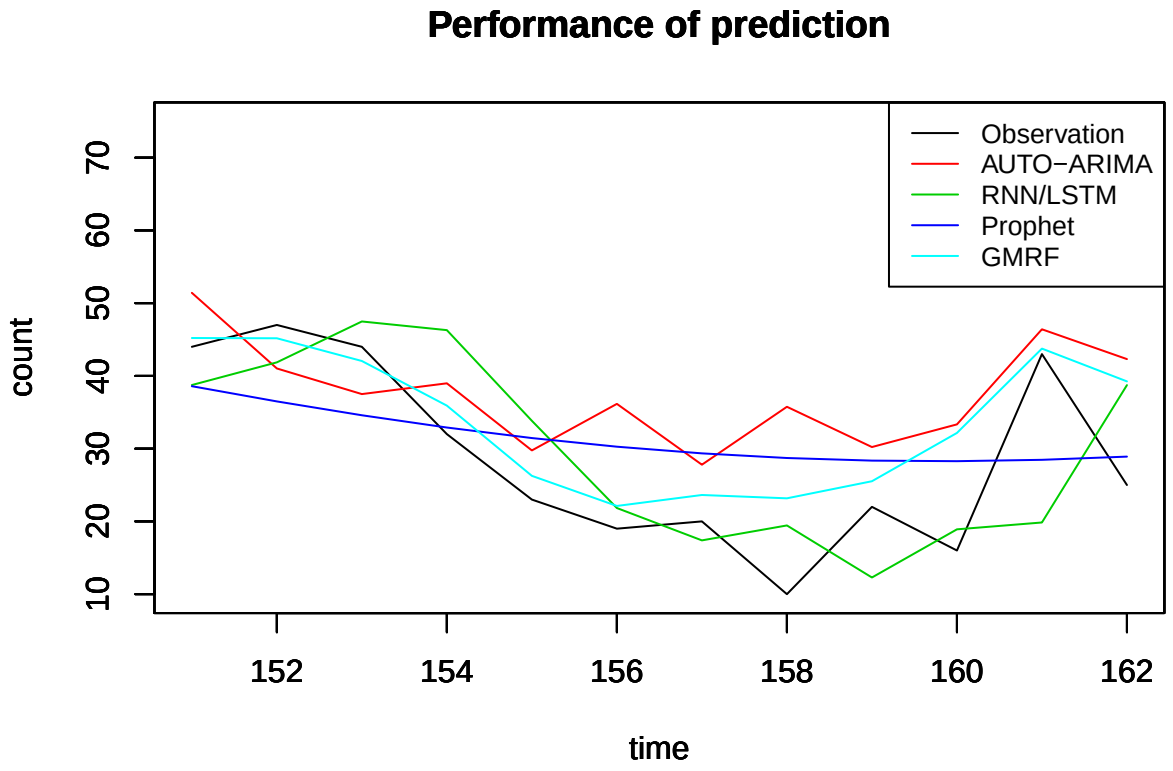
```
## Initial log joint probability = -6.35307
## Optimization terminated normally:
##   Convergence detected: absolute parameter change was below tolerance
```



Then, I will go further and compare them with regard to forecasting the last 12 points. Seasonal part seems to be straightforward to use in forecasting. For trend part, I use posterior median of *theta*. And then I sum up the seasonal component with a linear extension of trend component as my prediction.

The results of the above forecasting strategy along with other model's prediction, like ARIMA and RNN/LSTM are shown as following. I also show the result of `Prophet`.

Method	MSE
AUTO-ARIMS	160.4
RNN/LSTM	109.9
Prophet	106.9
GMRF	59.0



As is shown above, GMRF obviously outperforms the other methods. RNN/LSTM and `Prophet` make similar performances. They are well-used models in parctice, for example, `Prophet` is used in many applications across Facebook for producing reliable forecasts for planning and goal setting. The main reason that they are not competitive here could be the amount of data is really small, or that, the issue of overfitting.

Discussion

Conclusion

As is mentioned above, GMRF is a powerful nonparametric regression method which can also be applied into time series. It has a good chance to catch various characteristic features of interests, like autocorrelation structure and periodic component.

In fact, priors have a shrinkage effect, which can also be seen as a kind of regularization. when it comes to MLE, ridge regression is equivalent to Gaussian prior, and Lasso regression is equivalent to Laplace prior. Under Bayesian framework, we are more flexible to choose from different kinds of priors.

When it comes to model structure, GMRF is a realization of partial-pooling model. The connections between θ_i is limited by distance, which balances local adaptation and global control. A partial-pooling model is always a good idea, bacause it is more flexible than a fully-pooling model and more controllable than a non-pooling model. It is a tradeoff of bias and variance.

Limitation

The main issue here is computational efficience. Bayesian method is time-consuming. And Horseshoe prior has a more extreme problem of it than Gaussian prior and Laplace prior.

The idea is to never let a Bayesian model to deal with a too complicated function fitting. Alternatively, we could firstly remove some component to make the model simpler. Another idea is to leave a mangeable amount of data for Bayesian model and fit a freqentist model with the rest data. And then include the results of frequentist model as priors. Besides, A general idea of online learning and ensemble learning is also appealing to me.

Reference

[1]Faulkner, James R.; Minin, Vladimir N. Locally Adaptive Smoothing with Markov Random Fields and Shrinkage Priors. Bayesian Anal. 13 (2018), no. 1, 225–252. doi:10.1214/17-BA1050. https://projecteuclid.org/euclid.ba/1487905413

Original Computational Environment

```
sessionInfo()
```

```
## R version 3.4.4 (2018-03-15)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 10 x64 (build 17134)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=English_United States.1252
## [2] LC_CTYPE=English_United States.1252
## [3] LC_MONETARY=English_United States.1252
## [4] LC_NUMERIC=C
## [5] LC_TIME=English_United States.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] bindrcpp_0.2.2     prophet_0.2.1      Rcpp_0.12.16
## [4] knitr_1.20         forecast_8.2       loo_2.0.0
## [7] rstan_2.17.3       StanHeaders_2.17.2 ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
##  [1] tseries_0.10-43    tidyr_0.8.0        splines_3.4.4
##  [4] gtools_3.5.0       threejs_0.3.1      shiny_1.1.0
##  [7] assertthat_0.2.0   TTR_0.23-3         highr_0.6
```

```
## [10] stats4_3.4.4       yaml_2.1.18        pillar_1.2.1
## [13] backports_1.1.2    lattice_0.20-35    quadprog_1.5-5
## [16] glue_1.2.0         digest_0.6.15      promises_1.0.1
## [19] minqa_1.2.4        colorspace_1.3-2   htmltools_0.3.6
## [22] httpuv_1.4.3       Matrix_1.2-14      plyr_1.8.4
## [25] timeDate_3043.102  dygraphs_1.1.1.4   pkgconfig_2.0.1
## [28] purrr_0.2.4        xtable_1.8-2       scales_0.5.0
## [31] later_0.7.2        lme4_1.1-16        tibble_1.4.2
## [34] bayesplot_1.5.0    DT_0.4             shinyjs_1.0
## [37] nnet_7.3-12        lazyeval_0.2.1     quantmod_0.4-12
## [40] survival_2.42-3    magrittr_1.5       mime_0.5
## [43] evaluate_0.10.1    nlme_3.1-131.1     MASS_7.3-49
## [46] xts_0.10-2         colourpicker_1.0   rsconnect_0.8.8
## [49] tools_3.4.4        matrixStats_0.53.1 extraDistr_1.8.8
## [52] stringr_1.3.1      munsell_0.4.3      compiler_3.4.4
## [55] rlang_0.2.0        grid_3.4.4         nloptr_1.0.4
## [58] ggridges_0.4.1     rstanarm_2.17.4    htmlwidgets_1.2
## [61] crosstalk_1.0.0    igraph_1.2.1       miniUI_0.1.1
## [64] labeling_0.3       base64enc_0.1-3    rmarkdown_1.9
## [67] fracdiff_1.4-2     gtable_0.2.0       codetools_0.2-15
## [70] curl_3.2           inline_0.3.14      markdown_0.8
## [73] reshape2_1.4.3     R6_2.2.2           gridExtra_2.3
## [76] rstantools_1.4.0   zoo_1.8-1          dplyr_0.7.5
## [79] bindr_0.1.1        shinystan_2.4.0    shinythemes_1.1.1
## [82] rprojroot_1.3-2    stringi_1.1.7      parallel_3.4.4
## [85] lmtest_0.9-35      tidyselect_0.2.4
```