Time Series Decomposition Based on Markov Random Fields

Bowen Xiao

May 16, 2018

Setup

```
library(rstan)
library(loo)
library(mice)
library(Kendall)
library(trend)
library(forecast)
library(splines)
library(e1071)
```

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, my model will look like this:

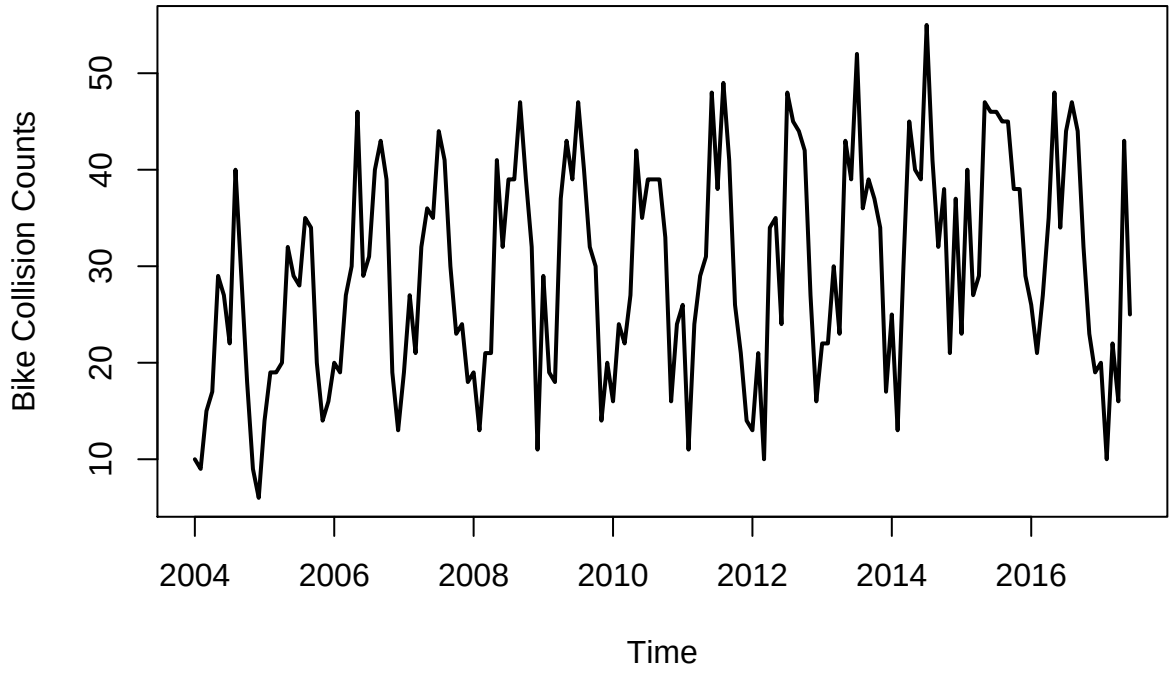
$$
\begin{aligned}
y_i &\sim normal(\theta_i, \sigma); \\
\sigma &\sim exponential(1) \\
\theta_j &= \delta_{j-1} + \theta_{j-1}; \\
\delta_i &\sim normal(0, 1); \\
\theta_1 &= 5 * sd(log(y)) * \theta_0 + log(\bar{y}); \\
\theta_0 &\sim normal(0, 1);
\end{aligned}
$$

For time series decomposition, since it is count variable, in the first fitting I replace $y_i \sim normal(\theta_i, sigma)$ with $y_i \sim Poisson(e^{\theta_i}, \sigma)$. And different priors for δ are compared in both two fitting. As is well known, Bayes methods treat parameters as random variables. consequently, what we get is posterior distribution of θ. To compare with frequentist methods, I will simply use the median of θ.

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. On the contrary, I will use GMRF as a smoothing technique to extract seasonal component and trend component sequentially. Prediction cannot be implemented straightforwardly until the trend line is extracted and given a parametric form by some regression methods, like B-splines or SVM. 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 (150 points) and test set (12 points). I decomposite train set into trend part, seasonal part and random part. I treat it as an additive model and decomposit it by doing Markov random fields fitting twice. Furthermore, I compare the results with some classical parametric techniques, like ARIMA, and machine learning techniques, like RNN/LSTM, with regrard to forcasting the test set. I also compare my result with `Prophet`, a Bayesian time series forecasting package written by Facebook.



Time series decomposition

I apply GMRF fitting into time series decomposition and forcasting. 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.
```

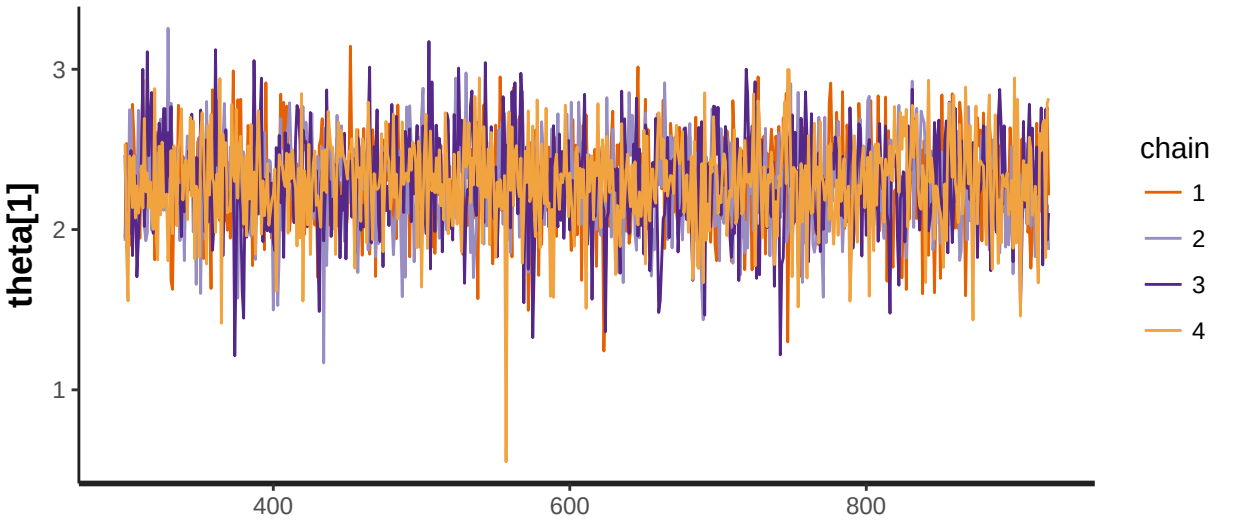
And I compare them based on LOO-PSIS-CV (efficient approximate leave-one-out cross-validation for Bayesian models fit using Markov chain Monte Carlo, which uses Pareto smoothed importance sampling, a new procedure for regularizing importance weights.) and WAIC (a generalized version of AIC)[2].

	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

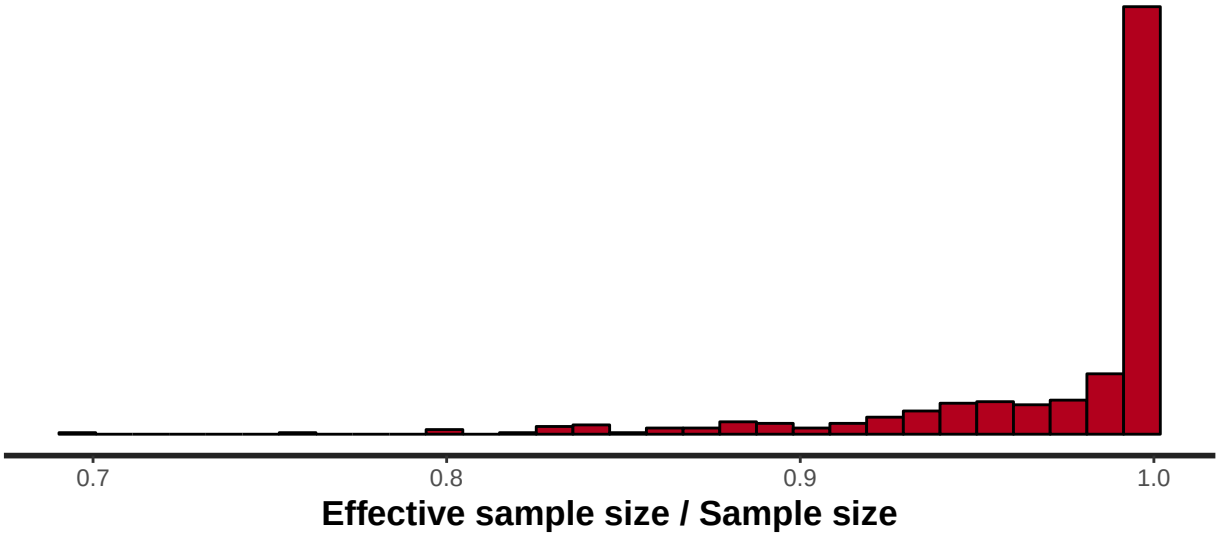
Both LOO-PSIS-CV and WAIC indicate model based on Laplace prior is the best. Thus, I will go on with this model.

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

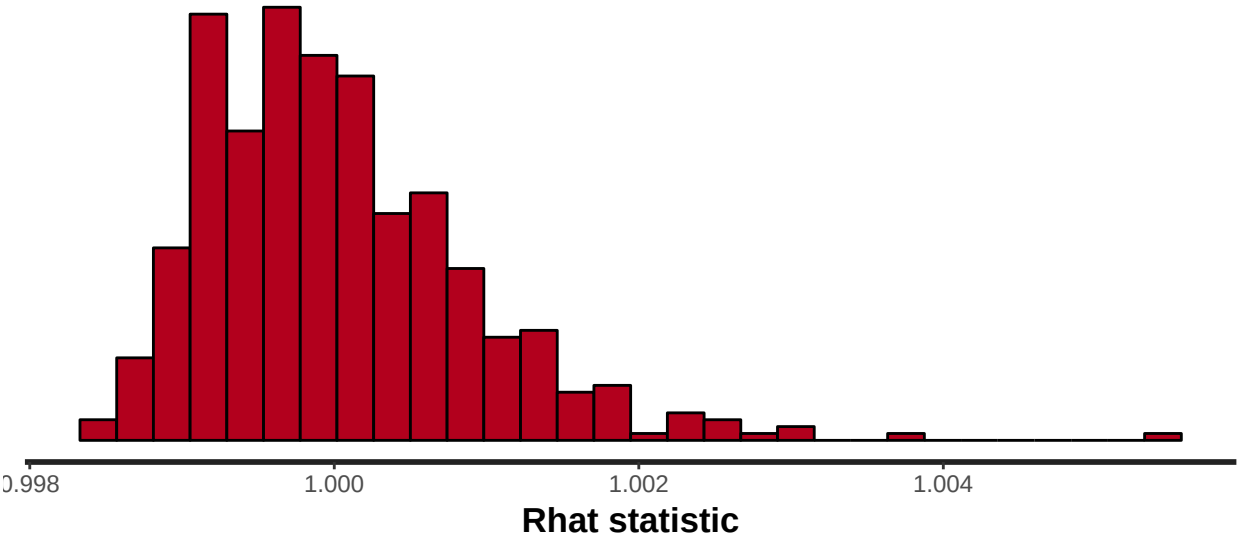


The overlapping lines show that the samples from 4 different chains are comming from one common distribution, or that, there is no violation for $theta_1$. The conclusion is also true for other parameters, because as we can see in the following, all the $\hat{R}s$ are close to 1 (The degree of convergence of a random Markov Chain can be estimated using the Gelman-Rubin convergence statistic, $\hat{R}$, based on the stability of outcomes between and within m chains of the same length, n. Values close to one indicate convergence to the underlying distribution. Values greater than 1.1 indicate inadequate convergence.)[3].

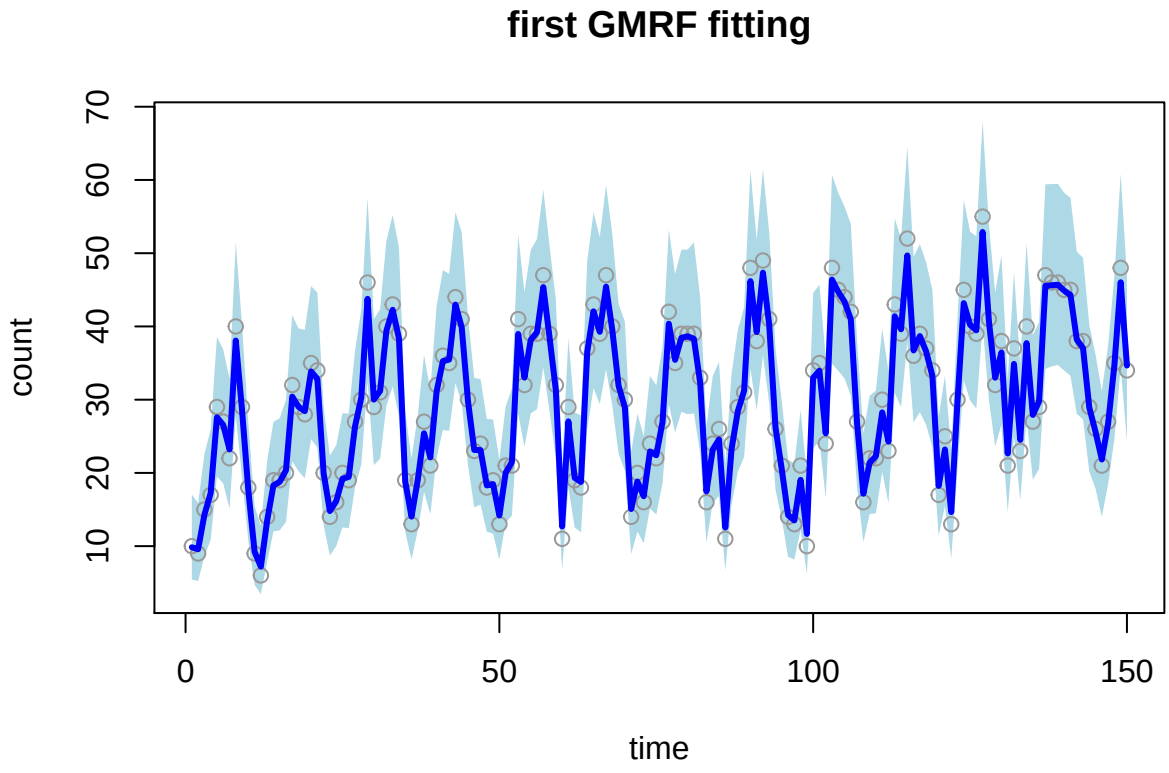
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
## `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 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.
```

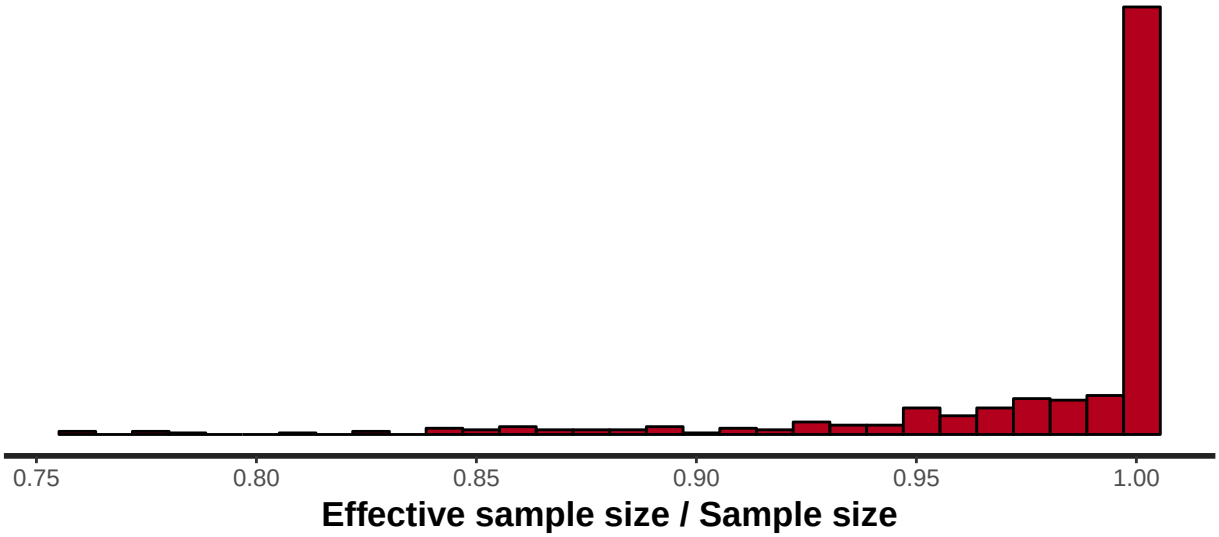
I also consider Gaussian prior, Laplace prior and Horseshoe prior. And both LOO-PSIS-CV and WAIC indicate model based on Gaussian prior is the best. Thus, I will focus on this model.

	elpd_diff	elpd_loo	se_elpd_loo	p_loo	se_p_loo	looic	se_looic
loo4	0.000000	-477.0286	8.241664	14.684622	1.487393	954.0572	16.48333
loo5	-1.240157	-478.2688	8.222502	19.709832	1.970478	956.5375	16.44500
loo6	-102.906839	-579.9354	7.532968	2.848136	0.198552	1159.8709	15.06594

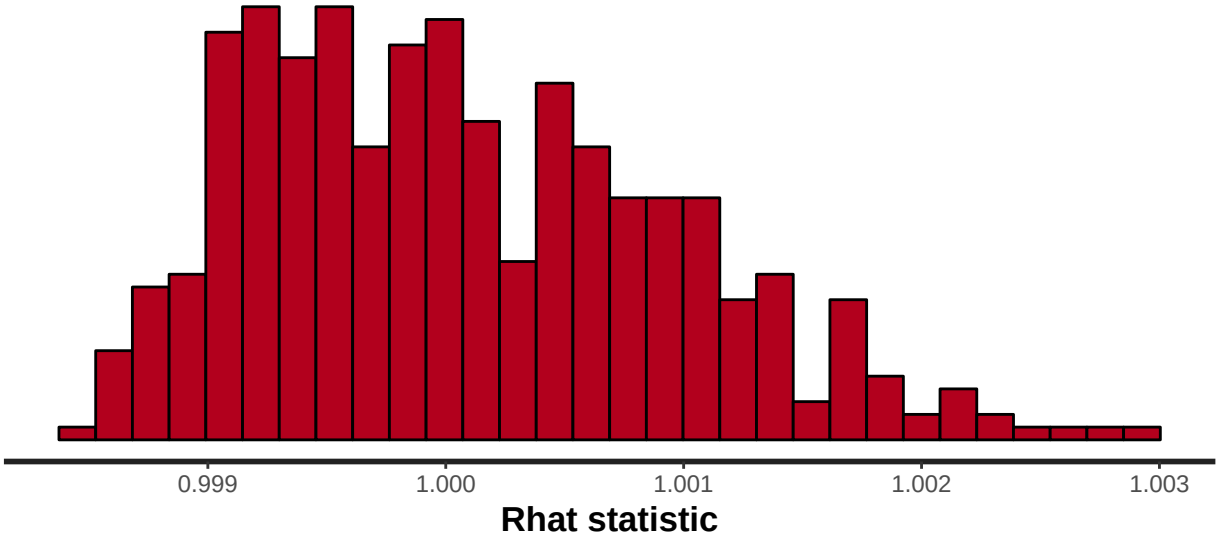
	elpd_diff	elpd_waic	se_elpd_waic	p_waic	se_p_waic	waic	se_waic
waic4	0.0000000	-476.8287	8.220102	14.484681	1.4650838	953.6573	16.44020
waic5	-0.9715345	-477.8002	8.169330	19.241269	1.9161214	955.6004	16.33866
waic6	-103.0990719	-579.9277	7.532083	2.840428	0.1972076	1159.8555	15.06417

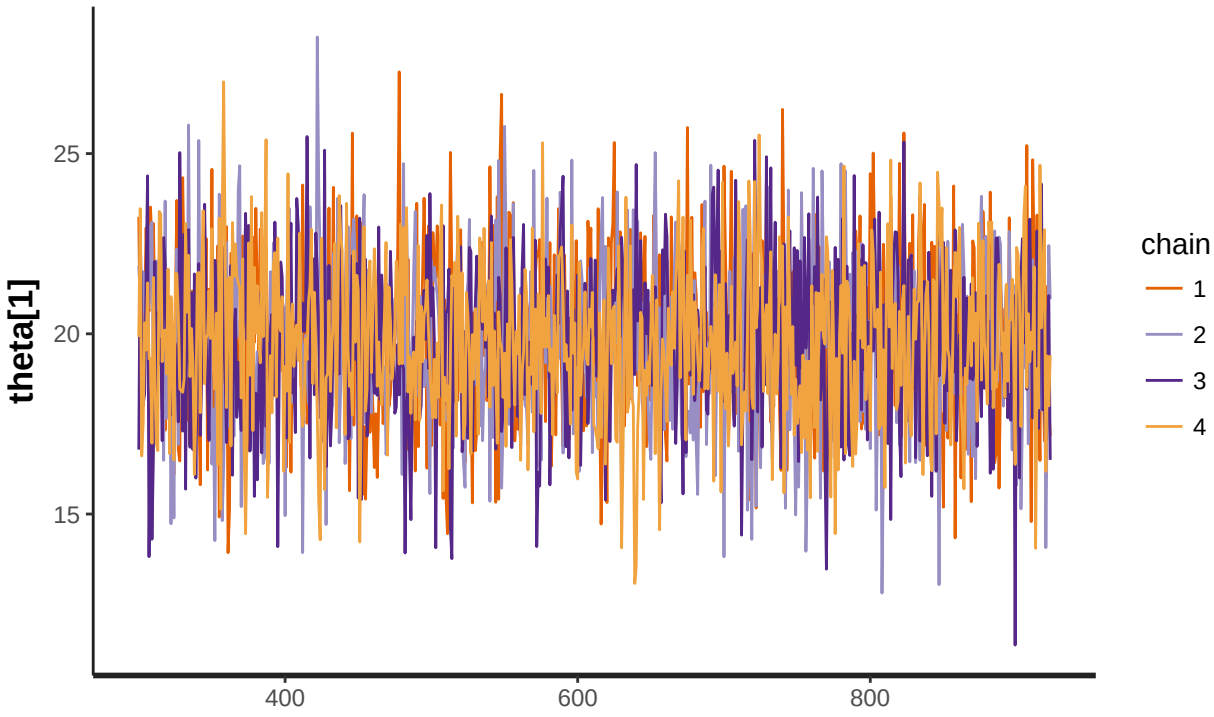
Similarly, here are the results of second fitting.

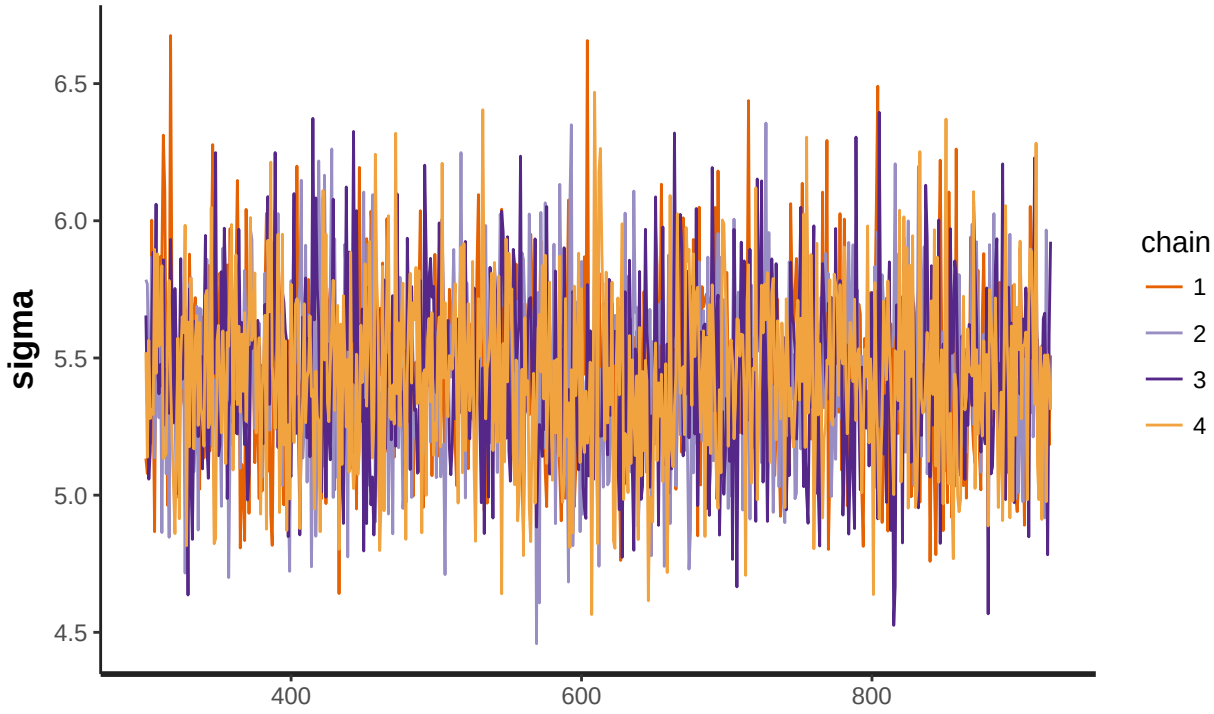
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

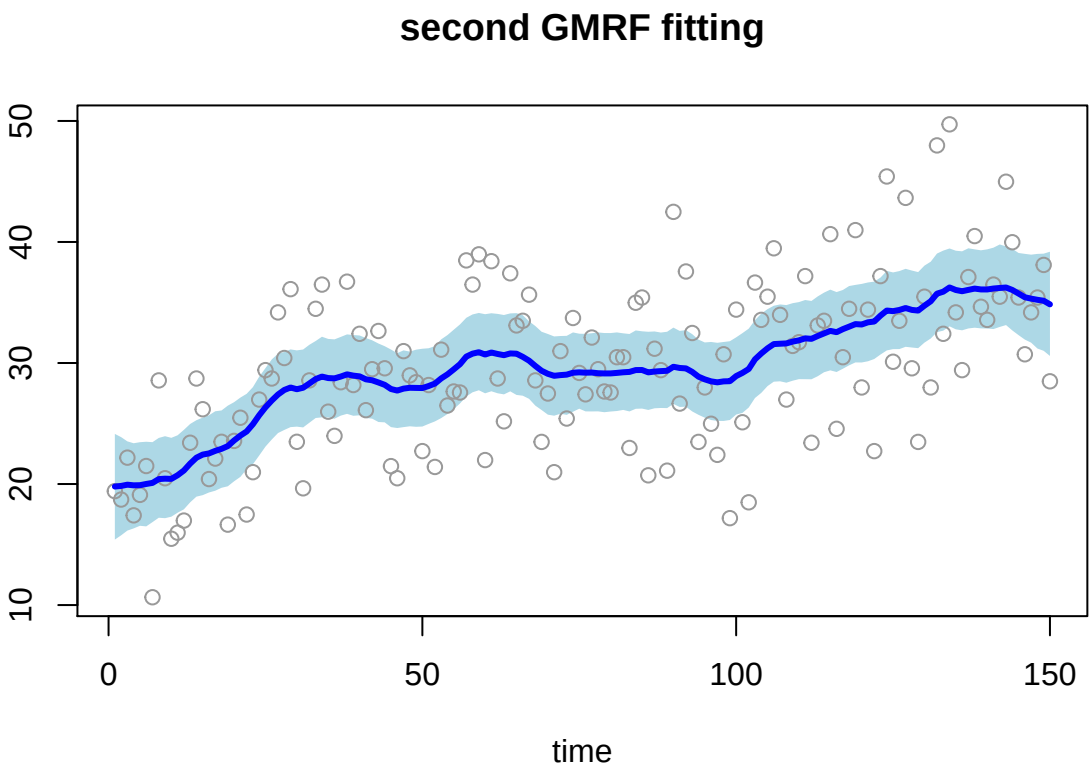



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

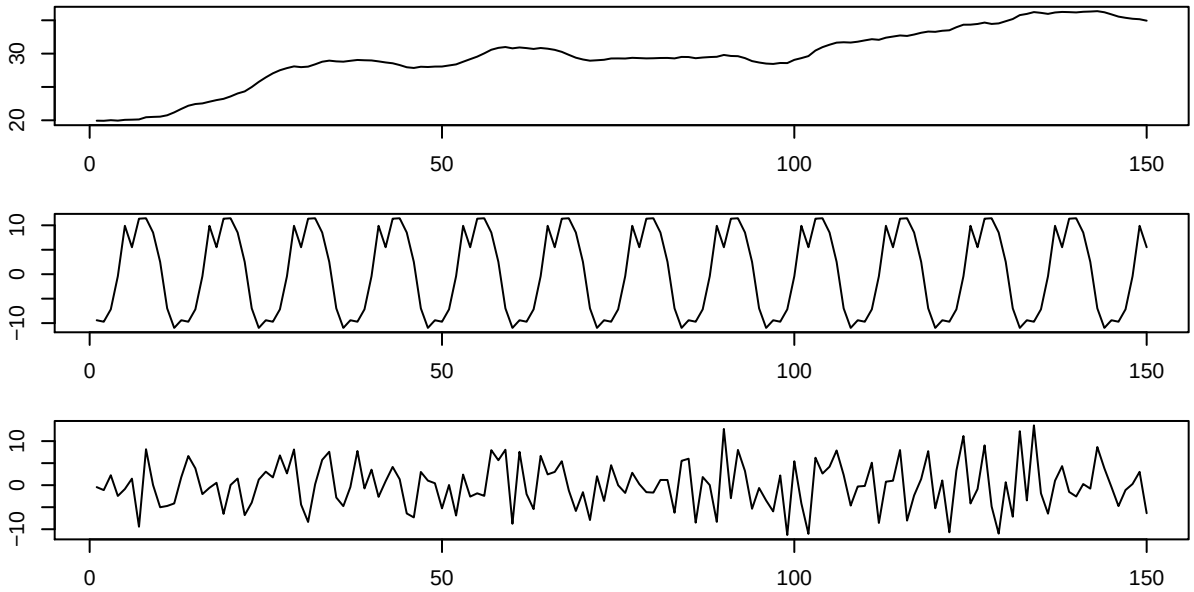



theta[1]
25
20
15
400
600
800
chain
1
2
3
4

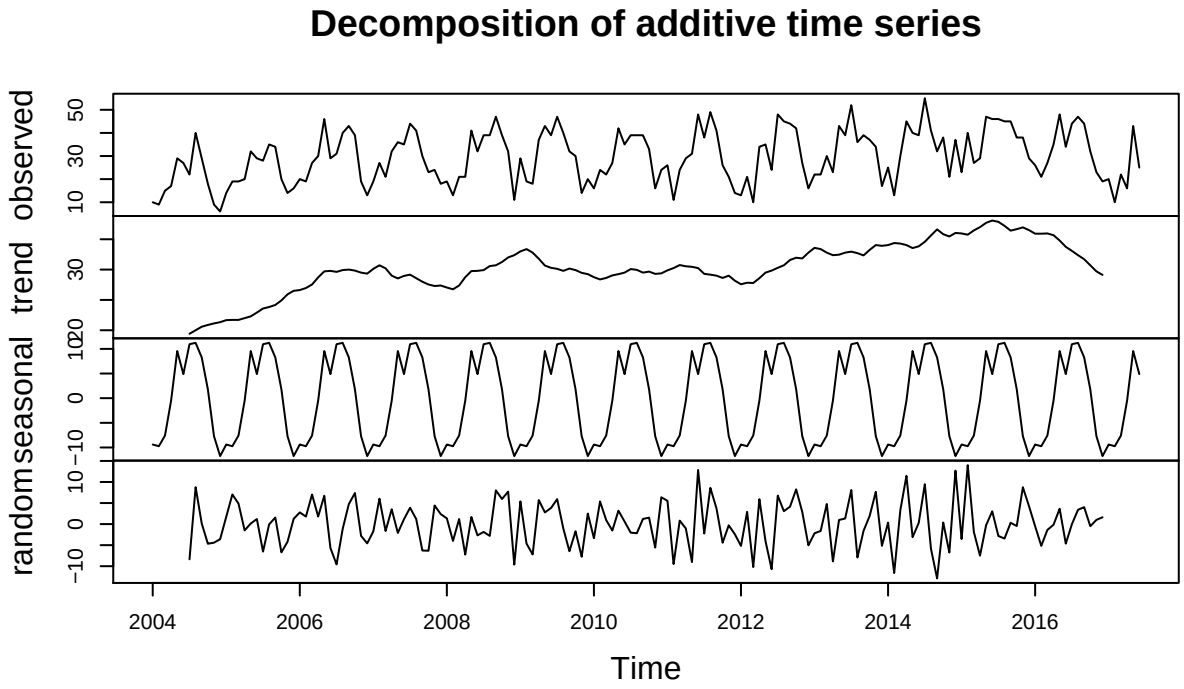

sigma
6.5
6.0
5.5
5.0
4.5
400
600
800
chain
1
2
3
4



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

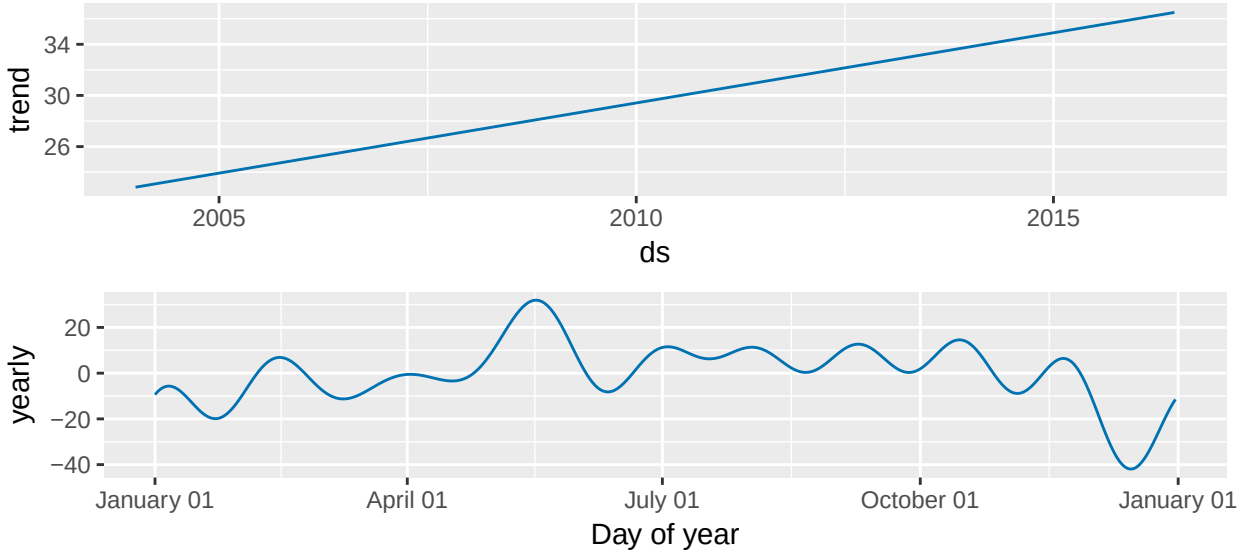


As a comparison, decomposition based on ARIMA looks like the following. Generally speaking, the seasonal part seems to be exactly the same, but my trend line is more smoothing.



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
```

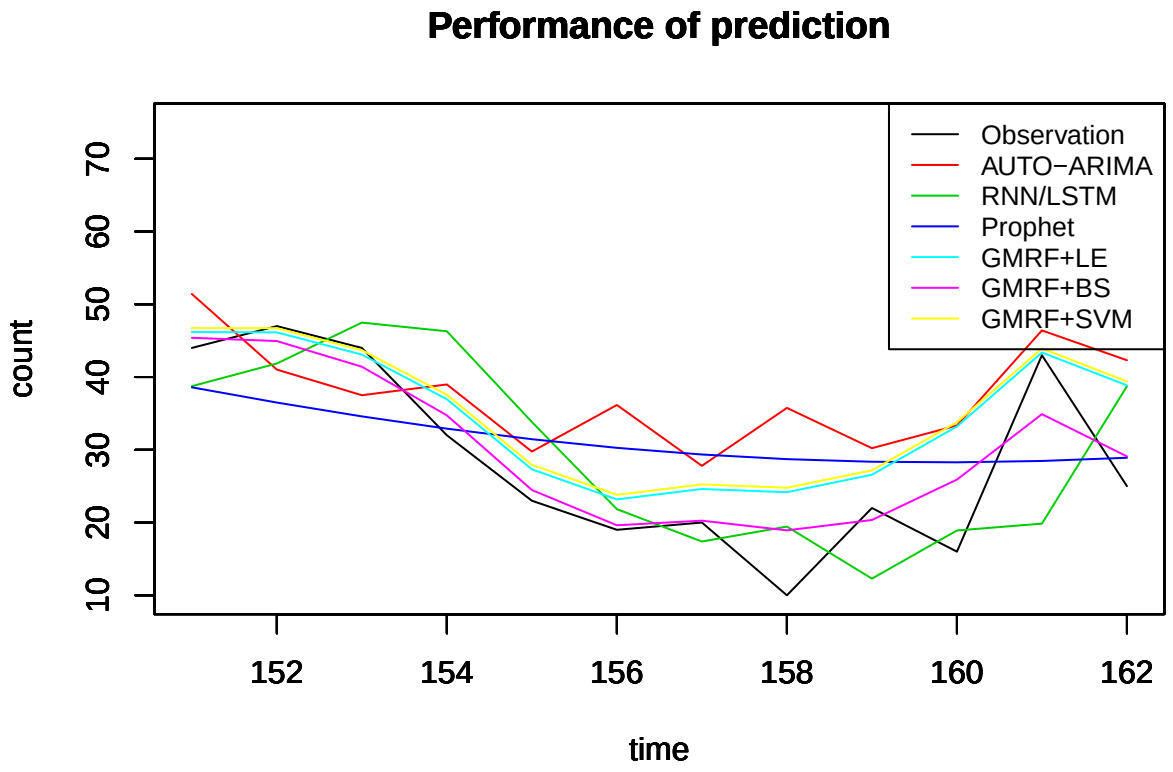


Then, I will go further and compare them with regard to forcasting the last 12 points. Seasonal part seems to be straightforward to use in forcasting. For trend part, firstly I consider a linear extension of trend component. The linear extension is implemented by `lm` function with a step wise chosen by cross validation, that is, I use the last few points to fit a linear regression to predict the future. Secondly, I also try to fit the trend line by B-spline or SVM. And then I sum up the seasonal component with a linear extension of trend

component as my prediction.

The results of the above forcasting 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+LinearExtension	66.5
GMRF+BSpline	32.0
GMRF+SVM	73.6



As is shown above, GMRF obviously outperforms the other methods, and using B-splines to fit the trend line seems to be the best choice. The predictions of GMRF with linear extension and SVM are really close to each other. 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 is 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. Combining GMRF and B-splines achieves really nice results, which is beyond my proposal.

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. In simulation study, for example, it takes totally about an hour to sample. And Horseshoe prior has a more extreme problem of it than Gaussian prior and Laplace prior. Moreover, if we try funtions like $y = sin(100x)$, divergence issues raise up.

My idea is to take advantages of both fitting power of Bayesian frameworks and computational efficience of frequentist methos. firstly, we should never let a Bayesian model to deal with a too complicated function fitting. Alternatively, we may use method like B-splines with knots to do a quick pilot fitting, and then run a divide and conquer algorithm with Bayesian model. Secondly, a Bayesian method could be a correction to a frequentist method, and vice versa. Actually, `GMRF+BSpline` is an example of realization. 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

[2]Vehtari, A., Gelman, A., and Gabry, J. (2017a). Practical Bayesian model evaluation using leaveone-out cross-validation and WAIC. Statistics and Computing. 27(5), 1413â “1432. doi:10.1007/s11222- 016-9696-4. (published version, arXiv preprint)

[3]Gelman, A. and D. B. Rubin (1992) Inference from iterative simulation using multiple sequences (with discussion). Statistical Science, 7:457-511

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] splines   stats     graphics  grDevices utils     datasets  methods
## [8] base
##
## other attached packages:
##  [1] bindrcpp_0.2.2     prophet_0.2.1      Rcpp_0.12.16
##  [4] knitr_1.20         e1071_1.6-8        forecast_8.2
##  [7] trend_1.1.0        Kendall_2.2        mice_2.46.0
## [10] lattice_0.20-35    loo_2.0.0          rstan_2.17.3
## [13] StanHeaders_2.17.2 ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
##  [1] nlme_3.1-131.1     matrixStats_0.53.1 xts_0.10-2
##  [4] threejs_0.3.1      rprojroot_1.3-2    tools_3.4.4
##  [7] backports_1.1.2    R6_2.2.2           DT_0.4
## [10] rpart_4.1-13       lazyeval_0.2.1     colorspace_1.3-2
## [13] nnet_7.3-12        tidyselect_0.2.4   gridExtra_2.3
## [16] curl_3.2           compiler_3.4.4     shinyjs_1.0
## [19] labeling_0.3       colourpicker_1.0   tseries_0.10-43
## [22] scales_0.5.0       dygraphs_1.1.1.4   lmtest_0.9-35
## [25] fracdiff_1.4-2     quadprog_1.5-5     ggridges_0.4.1
## [28] stringr_1.3.1      digest_0.6.15      minqa_1.2.4
## [31] rmarkdown_1.9      rstanarm_2.17.4    extraDistr_1.8.8
## [34] base64enc_0.1-3    pkgconfig_2.0.1    htmltools_0.3.6
## [37] lme4_1.1-16        highr_0.6          htmlwidgets_1.2
## [40] rlang_0.2.0        TTR_0.23-3         quantmod_0.4-12
## [43] shiny_1.1.0        bindr_0.1.1        zoo_1.8-1
## [46] crosstalk_1.0.0    gtools_3.5.0       dplyr_0.7.5
## [49] inline_0.3.14      magrittr_1.5       bayesplot_1.5.0
## [52] Matrix_1.2-14      munsell_0.4.3      stringi_1.1.7
## [55] yaml_2.1.18        MASS_7.3-49        plyr_1.8.4
## [58] grid_3.4.4         parallel_3.4.4     promises_1.0.1
## [61] miniUI_0.1.1       pillar_1.2.1       igraph_1.2.1
## [64] boot_1.3-20        markdown_0.8       shinystan_2.4.0
## [67] reshape2_1.4.3     codetools_0.2-15   stats4_3.4.4
## [70] rstantools_1.4.0   glue_1.2.0         evaluate_0.10.1
## [73] nloptr_1.0.4       httpuv_1.4.3       tidyr_0.8.0
## [76] gtable_0.2.0       purrr_0.2.4        assertthat_0.2.0
## [79] mime_0.5           xtable_1.8-2       later_0.7.2
## [82] rsconnect_0.8.8    class_7.3-14       survival_2.42-3
## [85] timeDate_3043.102  tibble_1.4.2       shinythemes_1.1.1
```