Bayesian estimation for time series models

Feb 14 2017

Eric Ward
Re-cap following models using Bayesian code

• Regression
• ARMA models
• State Space Models
• Dynamic Factor Analysis
• Dynamic Linear Models
• MARSS models (multivariate time series models)
Why Bayesian?

• Complex hierarchical models

• Inference: what’s the probability that the data are less than some threshold?

• No bootstrapping!
  – We get credible intervals for parameters and states simultaneously
Getting started

library(rstan)
library(devtools)
devtools::install_github("eric-ward/safs-timeseries/statss")
library(statss)
Using STAN objects

- Flow volume from Nile River
Fitting linear regression

- We’ll use wrapper function 'fit_stan'

```r
x = model.matrix(lm(Nile~1))

lm_intercept = fit_stan(y = as.numeric(Nile), x = rep(1, length(Nile)), model_name = "regression")
```
Output of fitted object

```r
> lm_intercept
Inference for Stan model: regression.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>se_mean</th>
<th>sd</th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
<th>n_eff</th>
<th>Rhat</th>
</tr>
</thead>
<tbody>
<tr>
<td>beta[1]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>sigma</td>
<td>936.34</td>
<td>1.79</td>
<td>66.16</td>
<td>819.09</td>
<td>891.53</td>
<td>929.75</td>
<td>978.62</td>
<td>1074.10</td>
<td>1365</td>
<td>1</td>
</tr>
<tr>
<td>pred[1]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[2]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[3]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[4]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[5]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[6]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[7]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[8]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[9]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[10]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[11]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
<tr>
<td>pred[12]</td>
<td>0.56</td>
<td>0.05</td>
<td>1.97</td>
<td>-3.25</td>
<td>-0.78</td>
<td>0.51</td>
<td>1.87</td>
<td>4.53</td>
<td>1321</td>
<td>1</td>
</tr>
</tbody>
</table>
```
Output from model

• Can be plotted using base graphics in R

```r
pars = extract(lm_intercept)
hist(pars$beta, 40, col="grey", xlab="Intercept", main="")
quantile(pars$beta, c(0.025,0.5,0.975))
```
Traceplots of parameters

- \texttt{traceplot(lm\_intercept, pars = "beta[1]" )}

- \texttt{traceplot(lm\_intercept, pars = "sigma" )}
Pairs plots between parameters

- pairs(lm_intercept, pars=c("beta[1]","sigma"))
Plots with credible intervals

- `plot(lm_intercept, pars=c("beta[1]","sigma"))`

  - `ci_level`: 0.8 (80% intervals)
  - `outer_level`: 0.95 (95% intervals)
Getting tidy summaries

coeff = broom::tidy(lm_intercept)
Useful in ggplot, for example

- `ggplot(coef[grep("pred", coef$term),], aes(x = 1:100, y=estimate)) + geom_point() + ylab("Estimate +/- SE") + geom_errorbar(aes(ymin=estimate-std.error, ymax=estimate+std.error)) + xlab("")

Not interesting – all values are the same!
Preserving MCMC chains

- Each chain is independent
- Defaults to merging samples from all chains together
  
  `extract(object, pars, permuted = TRUE)`

- But summaries can be generated for each combination of parameters-chains by setting
  
  `extract(object, pars, permuted = FALSE)`
Random walk

• Formula: \( E[Y_t] = Y_{t-1} + e_{t-1} \)

• We’ll fit model to temperature data

data(airquality)
Temp = airquality$Temp # air temperature
rw = fit_stan(y = Temp, est_drift = FALSE, model_name = "rw")
### Model convergence?

> rw

Inference for Stan model: rw.
3 chains, each with iter=1000; warmup=500; thin=1;
post-warmup draws per chain=500, total post-warmup draws=1500.

<table>
<thead>
<tr>
<th></th>
<th>mean</th>
<th>se_mean</th>
<th>sd</th>
<th>2.5%</th>
<th>25%</th>
<th>50%</th>
<th>75%</th>
<th>97.5%</th>
<th>n_eff</th>
<th>Rhat</th>
</tr>
</thead>
<tbody>
<tr>
<td>sigma</td>
<td>5.77</td>
<td>0.01</td>
<td>0.33</td>
<td>5.19</td>
<td>5.53</td>
<td>5.76</td>
<td>5.99</td>
<td>6.44</td>
<td>1125</td>
<td>1</td>
</tr>
<tr>
<td>pred[1]</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[2]</td>
<td>67.00</td>
<td>0.00</td>
<td>0.00</td>
<td>67.00</td>
<td>67.00</td>
<td>67.00</td>
<td>67.00</td>
<td>67.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[3]</td>
<td>72.00</td>
<td>0.00</td>
<td>0.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>72.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[4]</td>
<td>74.00</td>
<td>0.00</td>
<td>0.00</td>
<td>74.00</td>
<td>74.00</td>
<td>74.00</td>
<td>74.00</td>
<td>74.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[5]</td>
<td>62.00</td>
<td>0.00</td>
<td>0.00</td>
<td>62.00</td>
<td>62.00</td>
<td>62.00</td>
<td>62.00</td>
<td>62.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[6]</td>
<td>56.00</td>
<td>0.00</td>
<td>0.00</td>
<td>56.00</td>
<td>56.00</td>
<td>56.00</td>
<td>56.00</td>
<td>56.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[7]</td>
<td>66.00</td>
<td>0.00</td>
<td>0.00</td>
<td>66.00</td>
<td>66.00</td>
<td>66.00</td>
<td>66.00</td>
<td>66.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[8]</td>
<td>65.00</td>
<td>0.00</td>
<td>0.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>65.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[9]</td>
<td>59.00</td>
<td>0.00</td>
<td>0.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
<td>59.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[10]</td>
<td>61.00</td>
<td>0.00</td>
<td>0.00</td>
<td>61.00</td>
<td>61.00</td>
<td>61.00</td>
<td>61.00</td>
<td>61.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
<tr>
<td>pred[11]</td>
<td>69.00</td>
<td>0.00</td>
<td>0.00</td>
<td>69.00</td>
<td>69.00</td>
<td>69.00</td>
<td>69.00</td>
<td>69.00</td>
<td>1500</td>
<td>NaN</td>
</tr>
</tbody>
</table>
Univariate state space models

• State equation

\[ x_t = \phi \cdot x_{t-1} + e_{t-1}; e_{t-1} \sim Normal(0, q) \]

• Observation equation

\[ Y_t \sim Normal(x_t, r) \]

• Let’s compare models with and without the ar parameter phi in the process model
Using our `stan_fit` function

```r
ss_ar = fit_stan(y = Temp, est_drift=FALSE,
model_name = "ss_ar")

ss_rw = fit_stan(y = Temp, est_drift=FALSE,
model_name = "ss_rw")
```
Estimates from AR SS model
Estimates from both models
Posterior probability

- What’s the probability that the temperature exceeds some threshold? 100 degrees?
Probability of $> 100$ degrees

\begin{verbatim}
pars = extract(ss_ar)
length(which(pars$pred > 100))
\end{verbatim}

Low probability: $\sim 20 / 229500$
Dynamic Factor Analysis

- Lake WA plankton example used in manual
  
  ```r
  # load the data (there are 3 datasets contained here)
  data(lakeWAp plankton)
  # we want lakeWAp planktonTrans, which has been transformed
  # so the 0s are replaced with NAs and the data z-scored
  dat = lakeWAp planktonTrans
  # use only the 10 years from 1980-1989
  plankdat = dat[dat[,"Year"]>=1980 & dat[,"Year"]<1990,]
  # create vector of phytoplankton group names
  phytoplankton = c("Cryptomonas", "Diatoms", "Greens",
                     "Unicells", "Other.algae")
  # get only the phytoplankton
  dat.spp.1980 = plankdat[,phytoplankton]
  ```
Plankton data

- Cryptomonas
- Unicells
- Diatoms
- Other.algae
- Greens
Running the model

• 3 trend model to start

```
mod_3 = fit_dfa(y = t(dat.spp.1980),
num_trends=3)
```
Trends need to be rotated (like MARSS)

• Again we’ll use varimax rotation
• Use function we’ve written, `rotate_trends`

```python
rot = rotate_trends(mod_3)
```
names(rot)

- $Z_{\text{rot}}$, rotated $Z$ matrix for each MCMC draw
- trends, rotated trends for each MCMC draw
- $Z_{\text{rot\_mean}}$, mean $Z$ across draws
- trends\_mean, mean trends across draws
- trends\_lower, lower 2.5% on trends
- trends\_upper, upper 97.5% on trends
Predicted values from Bayesian DFA

- Same results as MARSS (trends_mean)
Other variance structures

\[
\text{mod}_3 = \text{fit_dfa}(y = t(\text{dat.spp.1980}), \text{num_trends}=3)
\]

- By default, this is modeling ‘diagonal and equal’

- Diagonal and unequal or shared variances can also be specified using ‘varIndx’ argument

\[
\text{mod}_3 = \text{fit_dfa}(y = t(\text{dat.spp.1980}), \text{num_trends}=3, \text{varIndx} = \text{rep}(1,5))
\]
Model selection: how to select best number of trends?

• First run multiple models with varying trends

  mod_1 = fit_dfa(y = t(dat.spp.1980), num_trends=1)
  mod_2 = fit_dfa(y = t(dat.spp.1980), num_trends=2)
  mod_3 = fit_dfa(y = t(dat.spp.1980), num_trends=3)
  mod_4 = fit_dfa(y = t(dat.spp.1980), num_trends=4)
  mod_5 = fit_dfa(y = t(dat.spp.1980), num_trends=5)

• 3 minutes to fit all models (4000 iterations) – probably could be at least cut in half
Leave One Out Information Criterion (LOOIC)

- Like AIC, lower is better
- Simple function in `library(loo)`

```r
loo(extract_log_lik(mod_1))$looic
```

<table>
<thead>
<tr>
<th>Trends</th>
<th>LOOIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1598.889</td>
</tr>
<tr>
<td>2</td>
<td>1535.885</td>
</tr>
<tr>
<td>3</td>
<td>1469.439</td>
</tr>
<tr>
<td>4</td>
<td>1455.7</td>
</tr>
<tr>
<td>5</td>
<td>1449.304</td>
</tr>
</tbody>
</table>
Predicted values

• Also like MARSS, use $\texttt{pred}$ parameter
Uncertainty intervals on states

• We often just present the trend estimates for DFA – but not uncertainty

• Let’s look at effect of missing data on DFA for the harbor seal dataset

data("harborSealWA")
• Assume single trend for the population
• pars = extract(fit_dfa(y = t(harborSealWA[, -1]),
num_trends = 1))
What happens when we delete last 3 years of data?
DLMs

- Mark’s example of salmon survival
- Logit transformed data
Fitting a DLM with time varying intercept

mod = fit_stan(y = SalmonSurvCUI$logit.s, model_name="dlm-intercept")
Constant intercept, time – varying slope

mod = fit_stan(y = SalmonSurvCUI$logit.s, x = SalmonSurvCUI$CUI.apr, model_name="dlm-slope")

Wider CIs than time varying intercept model
Time varying intercept and slope

• Use `model.matrix()` to specify ‘x’

• `x = model.matrix(lm(SalmonSurvCUI$logit.s ~ SalmonSurvCUI$CUI.apr))`

```
mod = fit_stan(y = SalmonSurvCUI$logit.s, x = model.matrix(lm(SalmonSurvCUI$logit.s ~ SalmonSurvCUI$CUI.apr)), model_name="dlm")
```
Summary

- Additional models available, e.g. 'marss'

- Very flexible

- Easy to add custom features on existing code