Bayesian Structural Time Series

Steven L. Scott

sharethis

October 21, 2022

Overview

Summary: Bayesian structural time series models are really useful. Make them accessible to non-experts.

1. Why did I create bsts?
2. What does it do?
3. How does it do it?
4. Why has the package done well?
5. Where should we go from here?

R:

```
install.packages("bsts")
library(bsts)
```

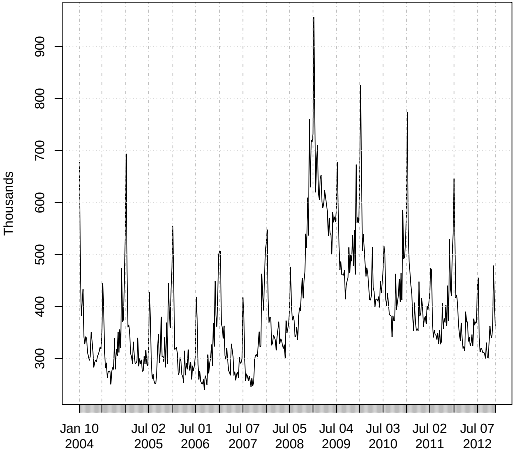
Python:

```
pip install BayesBoom
import BayesBoom.bsts as bsts
```

Outline

Motivation: Nowcasting

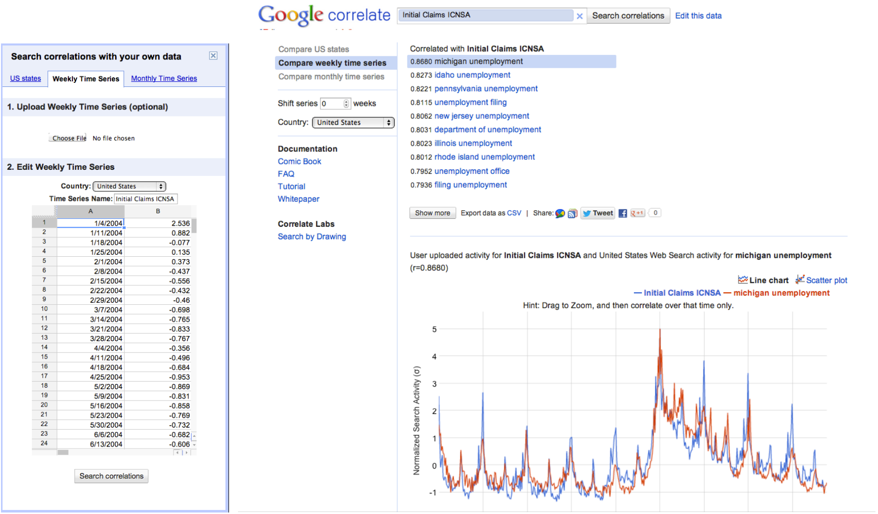
Maintaining "real time" estimates of infrequently observed time series.
(Scott and Varian, 2014, 2015)



- US weekly initial claims for unemployment.
  (Leading indicator of recession.)
- Can we learn this week's number before it is released?
- Google Trends / Correlate provides a real time signal correlated with the outcome.

Google correlate

(Now dead) Could provide the 100 most highly correlated individual queries.



Additive structure

Plays very nicely with MCMC.

If the overall model is

$$y_t = \underbrace{\mu_t + \gamma_t}_{\text{time series}} + \underbrace{\beta^T \mathbf{x}_t}_{\text{regression}} + \epsilon_t$$

Then

- $y_t - \beta^T \mathbf{x}_t$ (conditional on β) is a pure time series problem.
- $y_t - \mu_t - \gamma_t$ (conditional on μ_t and γ_t) is a pure regression problem.

The additive structure comes in handy if you want to do anything "fancy" with either component.

- Fancy models for trend or seasonality.
- Fancy priors on the regression coefficients for handling sparsity.

Outline

Outline

Structural time series models

Observation equation

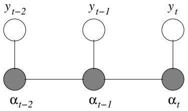
$$y_t = Z_t^T \alpha_t + \epsilon_t \qquad \epsilon_t \sim \mathcal{N}(0, H_t)$$

- y_t is the observed data at time t.
- Z_t and H_t are structural parameters.
- α_t is a vector of latent variables called the "state".

Transition equation

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t \qquad \eta_t \sim \mathcal{N}(0, Q_t)$$

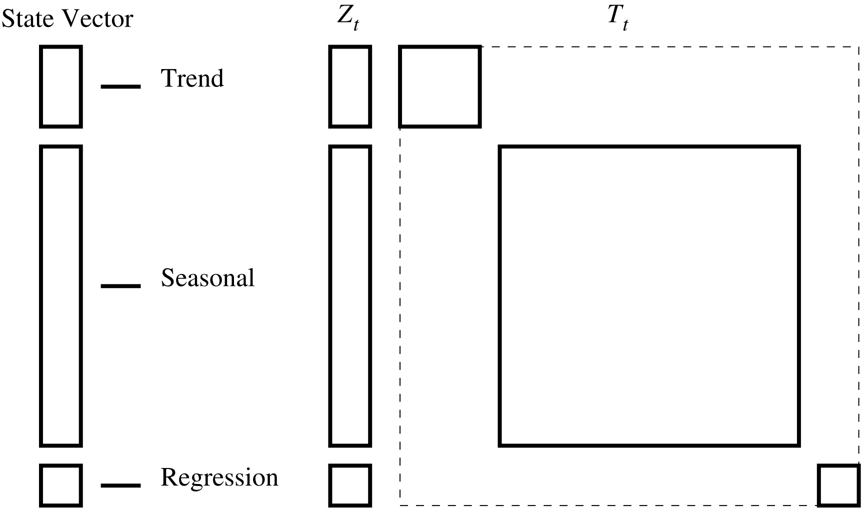
- T_t, R_t, and Q_t are structural parameters (partly known).
- η_t may be of lower dimension that α_t.



Structural parameters T_t, Z_t, R_t are often filled with 0's and 1's.

Structural time series models are modular

Add your favorite trend, seasonal, regression, holiday, etc. models to the mix



An example of how to specify state

This is the "basic structural model" with an added regression effect.

The model with S seasons can be written

$$y_t = \underbrace{\mu_t}_{\text{trend}} + \underbrace{\gamma_t}_{\text{seasonal}} + \underbrace{\beta^T \mathbf{x}_t}_{\text{regression}} + \epsilon_t$$

$$\mu_{t+1} = \mu_t + \delta_t + u_t$$

$$\delta_{t+1} = \delta_t + v_t$$

$$\gamma_{t+1} = -\sum_{s=1}^{S-1} \gamma_{t+1-s} + w_t$$

- Trend: "level" μ_t + "slope" δ_t.
- Seasonal: $S - 1$ dummy variables with time varying coefficients. Sums to zero in expectation.
- Regression: Spike and slab prior to handle sparsity.

Outline

Spike and slab priors

- Reflect the prior belief that most elements of β are zero.
- Let $\gamma_j = 1$ if $\beta_j \neq 0$ and $\gamma_j = 0$ if $\beta_j = 0$.

$$\gamma = [1, 0, 1, 1, 0, 0, \ldots, 1, 0]$$

- Now factor the prior distribution

$$p(\beta, \gamma, \sigma^{-2}) = p(\beta_\gamma | \gamma, \sigma^2) p(\sigma^2 | \gamma) p(\gamma)$$

where...

$$\gamma \sim \prod_j \pi_j^{\gamma_j} (1 - \pi_j)^{1-\gamma_j} \qquad \text{“Spike”}$$

$$\beta_\gamma | \gamma, \sigma^2 \sim \mathcal{N}\left(b_\gamma, \sigma^2 \left(\Omega_\gamma^{-1}\right)^{-1}\right) \qquad \text{“Slab”}$$

$$\frac{1}{\sigma^2} \sim \Gamma\left(\frac{df}{2}, \frac{ss}{2}\right)$$

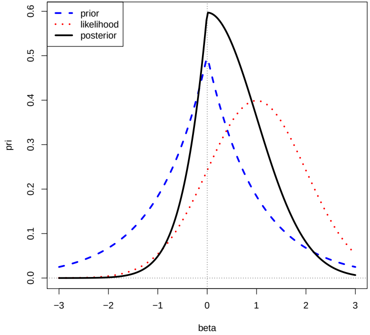
Prior elicitation

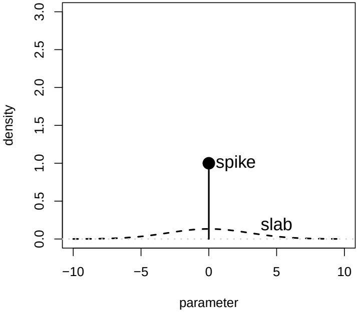
A complicated prior can be reduced to 4 numbers... with sensible defaults.

$$
\begin{aligned}
\pi_j &= \text{“expected model size”} \ / \text{ number of predictors} \\
b &= (\bar{y}, \mathbf{0}) \\
\Omega^{-1} &= \kappa\{\alpha \mathbf{X}^T\mathbf{X} + (1-\alpha)\mathrm{diag}\mathbf{X}^T\mathbf{X}\}/n \\
ss/df &= (1 - R^2_{\text{expected}})s_y^2 \\
df &= 1
\end{aligned}
$$

- The Ω^{-1} expression is κ observations worth of prior information.
- It can help to average Ω^{-1} with its diagonal.
- Prior elicitation is 4 numbers: expected model size, expected R^2, beta weight (κ), and sigma weight (df).

Spike and Slab vs Lasso





- "Lasso" or "L1 regularization" or "double exponential (Laplace) prior" produces point estimates at zero.
- Spike and slab prior produces point masses of probability at zero.

Outline

MCMC for spike and slab regression

Conditional on γ you can integrate out β and σ.

For each variable j, draw $\gamma_j | \gamma_{-j}, \mathbf{y}$.

$$\gamma | \mathbf{y} \sim C(\mathbf{y}) \frac{|\Omega_\gamma^{-1}|^{\frac{1}{2}}}{|V_\gamma^{-1}|^{\frac{1}{2}}} \frac{p(\gamma)}{SS_\gamma^{\frac{DF}{2}-1}}$$

- Each γ_j only assumes the values 0 or 1.
- $\sigma^2 V_\gamma$ is the posterior variance of β_γ in model γ.
- SS_γ is a "sum of squares."
- A $|\gamma| \times |\gamma|$ matrix needs to be inverted to compute $p(\gamma|\mathbf{y})$. This is VERY FAST if $|\gamma|$ is small.

Just Gibbs sample (for example) the discrete distribution $p(\gamma|\mathbf{y})$.
Draw $\beta, \sigma | \gamma, \mathbf{y}$ as needed through conjugacy.

MCMC for bsts

- The model parameters are $\theta = \{\sigma_\epsilon, \sigma_u, \sigma_v, \sigma_w, \beta\}$.
- The state is $\alpha = \{\alpha_1, \ldots, \alpha_n\}$.

MCMC algorithm alternates between:

1. Draw α given **y**, θ
   - Kalman filter "forward filter - backward sampler" draws α directly.
   - Several implementations available.
     I like Durbin and Koopman (2002, Biometrika).
     1.1 Kalman filter forward.
     1.2 Simulate fake data with the wrong mean, but right variance.
     1.3 Adjust the mean.

2. Draw θ given α.
   - Given α, then $[\sigma_u], [\sigma_v], [\sigma_w], [\beta, \sigma_\epsilon]$ are conditionally independent.
   - Independent priors on the time series σ's. Easy peasy.
   - "Spike and slab" prior on β handles sparsity when there are many potential controls.

Outline

Outline

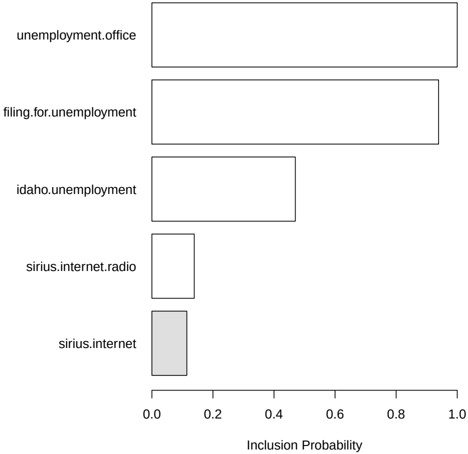
Fitting the model in bsts (R)

```
y <- my.data$ResponseVariable

ss <- AddLocalLinearTrend(
    list(),     ## No previous state specification.
    y)          ## Peek at the data for scaling the prior.

ss <- AddSeasonal(
    ss,             ## Adding state to ss.
    y,              ## Peek at the data for scaling.
    nseasons = 52) ## 52 "seasons" for weekly annual cycle.

model <- bsts(y ~ .,          ## regression formula like 'lm'
              state.specification = ss,  ## time series spec
              niter = 1000,              ## MCMC iterations
              data = my.data,
              expected.model.size = 1)   ## spike-slab
```

Posterior inclusion probabilities

With expected model size = 3, and the top 100 predictors from correlate

```
plot(model, "coef", inc = .1)
```



- Only showing inclusion probabilities $< .1$.
- Shading shows $Pr(\beta_j > 0|\mathbf{y})$.
  - White: positive coefficients
  - Black: negative coefficients

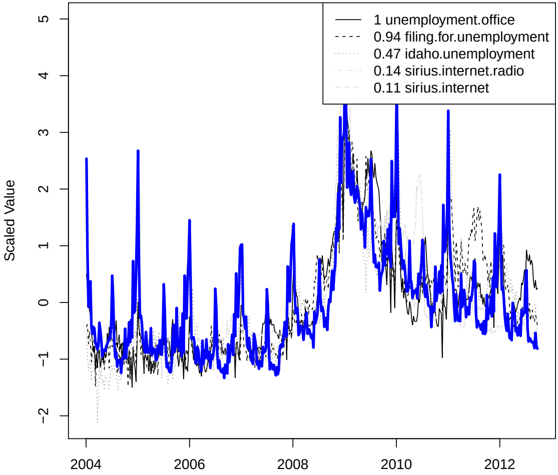
What can you plot?

Calling plot(model, "help") launches a browser with the help entry.

```
plot(x, y = c("state", "components", "residuals",
              "coefficients", "prediction.errors",
              "forecast.distribution",
              "predictors", "size", "dynamic", "seasonal", "help"),
      ...)
  PlotBstsCoefficients(bsts.object, burn = SuggestBurn(.1, bsts.object),
                       inclusion.threshold = 0, number.of.variables = NULL, ...)
  PlotBstsComponents(bsts.object, burn = SuggestBurn(.1, bsts.object),
                       time, same.scale = TRUE,
                       layout = c("square", "horizontal", "vertical"),
                       style = c("dynamic", "boxplot"),
                       ylim = NULL, ...)
  PlotDynamicRegression(bsts.object, burn = SuggestBurn(.1, bsts.object),
                        time = NULL, style = c("dynamic", "boxplot"),
                        layout = c("square", "horizontal", "vertical"),
                        ...)
  PlotBstsState(bsts.object, burn = SuggestBurn(.1, bsts.object),
                       time, show.actuals = TRUE,
                       style = c("dynamic", "boxplot"), ...)
  PlotBstsResiduals(bsts.object, burn = SuggestBurn(.1, bsts.object),
                       time, style = c("dynamic", "boxplot"), ...)
  PlotBstsPredictionErrors(bsts.object, burn = SuggestBurn(.1, bsts.object),
                       time, style = c("dynamic", "boxplot"), ...)
  PlotBstsSize(bsts.object, burn = SuggestBurn(.1, bsts.object), style =
                       c("histogram", "ts"), ...)
  PlotSeasonalEffect(bsts.object, nseasons = 7, season.duration = 1,
                     same.scale = TRUE, ylim = NULL, get.season.name = NULL,
                     burn = SuggestBurn(.1, bsts.object),  ...)
```

What got chosen?

```
plot(model, "predictors", inc = .1)
```

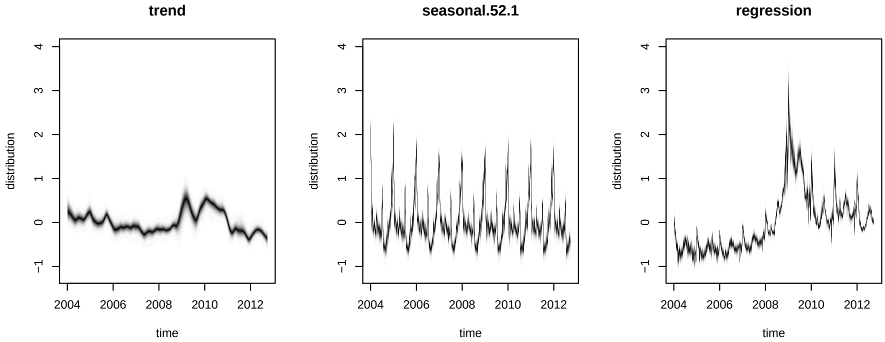


- Solid blue line: actual
- Remaining lines shaded by inclusion probability.

How much explaining got done?

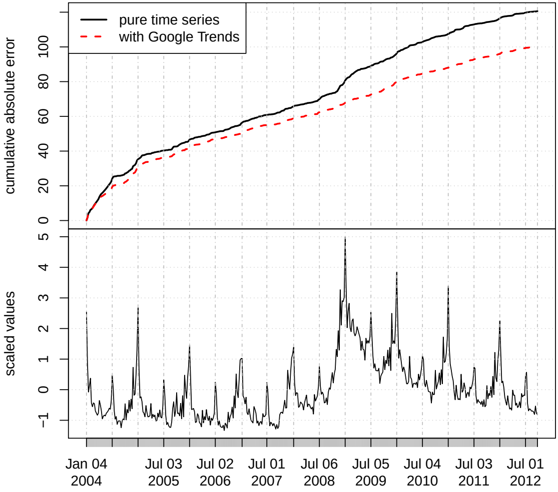
Dynamic distribution plot shows evolving pointwise posterior distribution of state components.

```
plot(model, "components")
```



Did it help?

```
CompareBstsModels(list("pure time series" = model1,
                       "with Google Trends" = model2))
```



- Plot shows cumulative absolute one-step-ahead prediction error
- The regressors are not very helpful during normal times.
- They help the model to quickly adapt to the recession.

Outline

List of state components

Trend

`AddLocalLevel`	Local level model
`AddLocalLinearTrend`	Local linear trend
`AddStudentLocalLinearTrend`	Robust local linear trend
`AddSemiLocalLinearTrend`	LLT with AR(1) slope
`AddAr`	AR(p)
`AddAutoAr`	AR(p) (spike-and-slab for p)

Seasonal

`AddSeasonal`	Seasonal
`AddTrig`	Trigonometric seasonal

Holiday

`AddFixedDateHoliday`	Holiday state models
`AddLastWeekdayInMonthHoliday`	Holiday state models
`AddNamedHolidays`	Holiday state models
`AddNthWeekdayInMonthHoliday`	Holiday state models

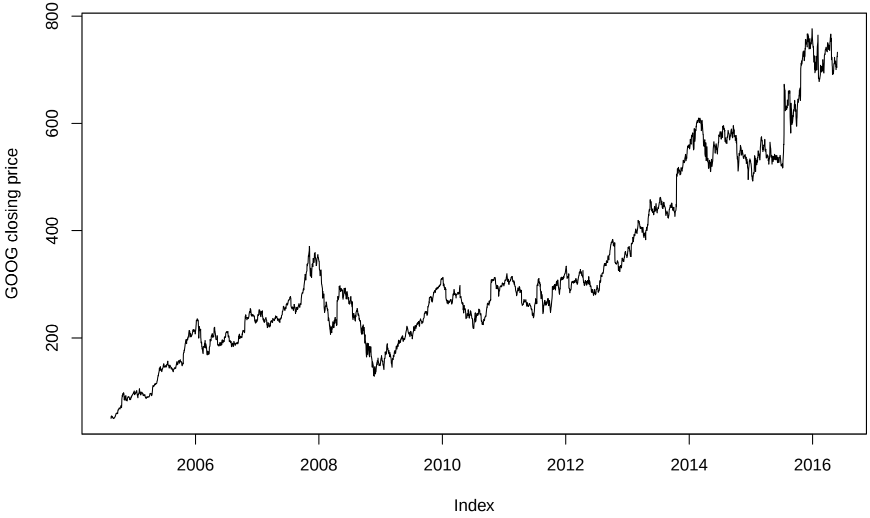
Regression

`AddDynamicRegression`	Dynamic regression

(Static regression is built in... no need for a state component).

Planners need long term forecasts for disk drives, etc.

I can't show you disk drive numbers, so consider GOOG stock price instead (adjusted for stock split).



Local linear trend is too flexible for long term forecasting

The 'slope' in a local linear trend is a random walk.

$$
\begin{aligned}
y_t &= \mu_t + \epsilon_t \\
\mu_{t+1} &= \mu_t + \delta_t + \eta_{0t} \\
\delta_{t+1} &= \delta_t + \eta_{1t}
\end{aligned}
$$

Replacing the slope with an AR(1) process (centered on a global mean) provides additional stability.

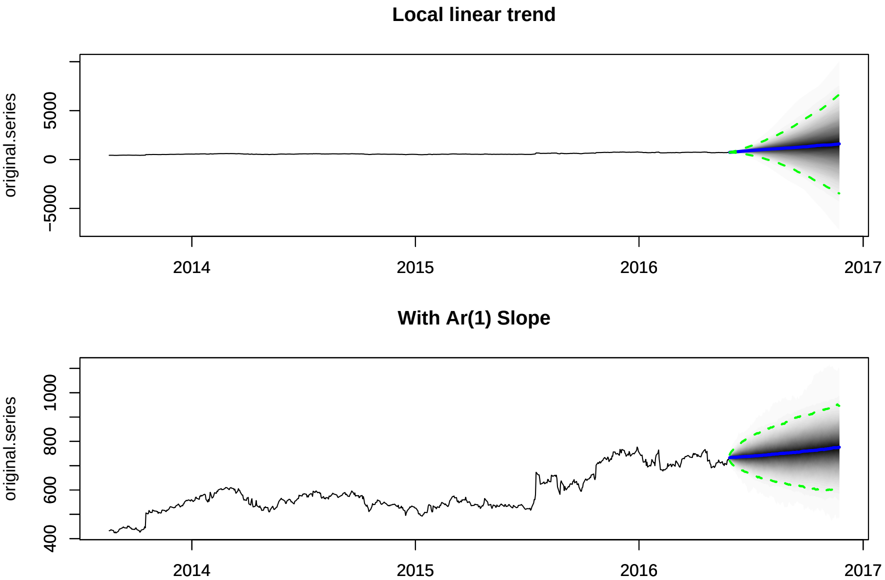
$$
\begin{aligned}
y_t &= \mu_t + \epsilon_t \\
\mu_{t+1} &= \mu_t + \delta_t + \eta_{0t} \\
\delta_{t+1} &= D + \rho(\delta_t - D) + \eta_{1t}
\end{aligned}
$$

Bsts calls this a "semilocal" local linear trend.

Long term forecasts

```
goog.adj <- GetAdjustedStockPriceFromSomewhere()

## Fit the model with local linear trend.
ss <- AddLocalLinearTrend(list(), goog.adj)
model1 <- bsts(goog.adj, ss, niter = 1000)

## Now forecast the next 180 days
pred1 <- predict(model1, horizon = 180)
plot(pred1, plot.original = 700)

## Do the same thing with a different trend model
ss2 <- AddSemiLocalLinearTrend(list(), goog.adj)
model2 <- bsts(goog.adj, ss2, niter = 1000)
pred2 <- predict(model2, horizon = 180)
plot(pred2, plot.original = 700)
```

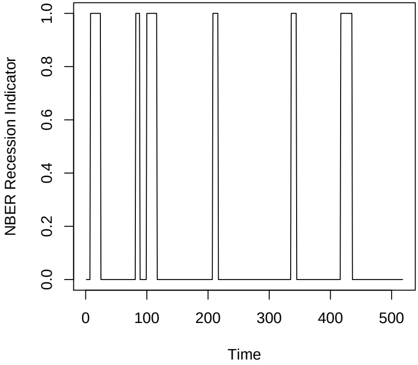
Additional structure leads to plausible uncertainty bounds



Outline

Example

Travis Berge, Nitish Sinha, and Michael Smolyansky (2016) investigate several market indicators to see which ones might be associated with recession.



- They use logistic regression with Bayesian model averaging (in STATA) to identify predictors.
- This can easily be replicated using BoomSpikeSlab (and improved using bsts).
- Response variable is binomial.
- Kalman filter and spike-and-slab **need Gaussian state and Gaussian observations.**

Data augmentation for non-Gaussian outcomes

Bsts supports data augmentation for the following

- Probit / Logit (success / failure, or bounded count data)
- Poisson (unbounded count data)
- Student T (numeric data with outliers)

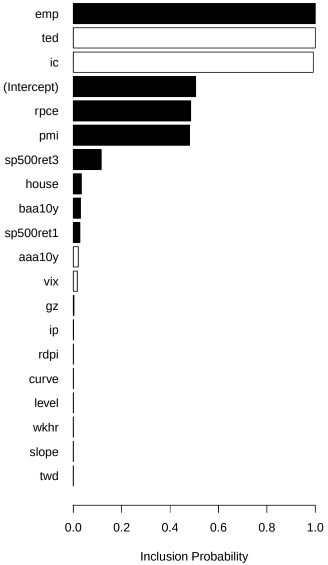
Might one day include

- Multinomial logit
- Support vector machines
- Quantile regression

Identifying relevant variables.

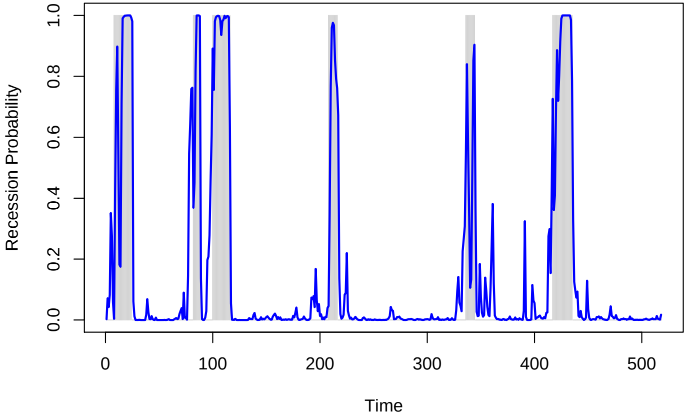
This is just logistic regression with a spike and slab prior. No time series yet...

```
gdp <- ReadFromSomewhere()

library(BoomSpikeSlab)

model <- logit.spike(nber ~ .,
           data = gdp,
           niter = 50000)

## Takes about 30s.

plot(model)
```



Checking predictions

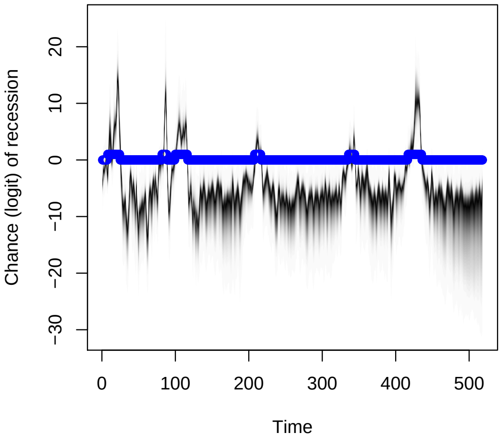
```
pred <- rowMeans(predict(model, newdata = gdp))
```



Missing predictors?
How about whether the previous time period was a recession?

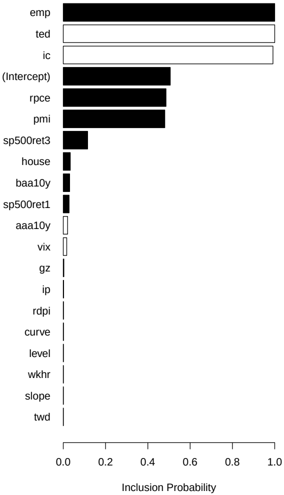
Accounting for time series structure

```
ss <- AddLocalLevel(
   list(),
   initial.y = 0,
   sdy = 1)

ts.model <- bsts(
   nber ~ .,
   ss,
   data = gdp.data,
   niter = 10000,
   family = "logit")
```

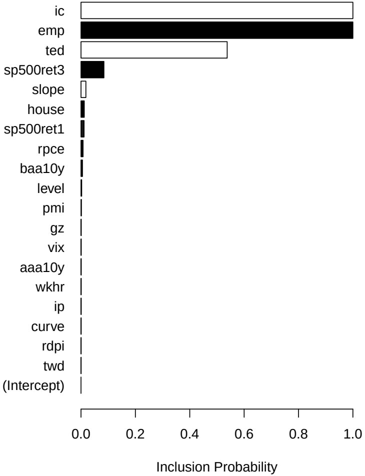


This is logistic regression with a random walk "intercept."

Fewer important predictors in time series model



Regression Alone



Reg + Time Series

Outline

Motivation

Modeling

- Time Series
- Regression
- MCMC

Applications

- Nowcasting (standard models)
- Long term forecasting
- Modeling non-Gaussian Outcomes

Reflections and Future Directions

Why bsts has done well

Some problems with "Bayesian software"

- "Bayesian software" (Stan, PyMC3, WinBUGS) tries to do too much.
  - Solving really hard problems with adaptive rejection, Hamiltonian MC, etc. means you're working too hard to solve easy problems.
  - Some problems have special tricks that can make MCMC go really fast.
- "Bayesian software" assumes you can type in your model.
  - Most users don't know what their model is.
  - Can't tell a Wishart from a Weibull.
  - Have no idea what to use for a prior.
- Most users don't know what to do with MCMC output.
  - If you don't know what the model is, you don't know what the parameters mean.

Problems with non-Bayesian software

It's not Bayesian.

Recent work

- Python implementation.
    - The R package is really just a lot of glue.
    - All the work happens in C++ (BOOM).
    - Python made possible by pybind11.
- Multivariate time series
    - Models with ~1000 series.
    - Not as fleshed out as univariate bsts, but acquiring road miles.
    - Shared vs model specific state.
    - Spike and slab on factor loadings.