Prediction Using Bayesian Structural Time Series

Jeshua Kracht

Technical Problem:

The Federal Reserve has a huge amount of data on the economy of the United States containing information of varying accuracy, relevance, and completeness. Many times series contain vital data about the state of the economy, but do not tell the whole story by themselves. Often, in hindsight, many indicator variables forecasted recessions and depressions before the markets crashed. Detecting symptoms of impending recession across hundreds of time series can be difficult. Much of the available data is noisy and, because of the ever changing nature of the economy, each recession has a slightly different signature. The challenge is to intelligently pair down dimensions and dynamically recognize warning signs based on prior multivariate time series data values without overfitting to only recognize historical recessions. This problem space is applicable to a massive number of problems and, as such, my solution is generic enough to work with essentially any multivariate time series prediction problem.

DSDP Description:

Overview:

1. Fill in missing and sparse data using Cubic Spline Interpolation.

$$S_i(x) = a_i(x - x_i)^3 + b_i(x - x_i)^2 + c_i(x - x_i) + d_i \quad \text{for } x \in [x_i, x_{i+1}].$$

2. Normalize values.

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

3. Pass training data to model.
4. Given the current state, model parameters, and y, calculate Spike and Slab prior (built in variable selection).

http://arxiv.org/pdf/math/0505633.pdf

5. Use Markov Chain Monte Carlo regression to sample at specific probability interval based on Spike and Slab prior.

See Detailed model form description below.

6. Repeat steps 4 and 5 n times.
7. Using Bayesian output, predict value of y for remaining test data.
8. Kernel Smooth output (where applicable).

Daniell Kernel: http://www.stat.berkeley.edu/~aditya/resources/LectureTWENTYSIX.pdf

9. Restrict output value range.
10. Validate actual vs. model prediction (visual inspection).
11. If validation succeeds, choose and build the best AutoRegressive Integrated Moving Average model for each x time series based on Akaike and Bayesian Information Criterion penalties.

http://xyala2.bio.ed.ac.uk/teaching/tutorials/phylogenetics/Bayesian_Workshop/PDFs/Posada%20and%20Buckley%20Syst%20Biol%202004.pdf

12. Forecast n number of future values for each x time series based on selected model.
13. Repeat steps 3-7 using all historical values as training data and forecasted values of x to predict y.

The tool, Causal Impact, is heavily based on Bayesian Structural Time Series and uses three metrics for prediction:

1. The Time-Series behavior response to itself.
2. The predictive behavior of other Time-Series on y.
3. Counter-factual inference based on prior knowledge about model parameters.

The model form is a Markov Chain Monte Carlo Bayesian approach.

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

Where the model parameters are $\theta = \{\sigma\varepsilon, \sigma u, \sigma v, \sigma w, \beta\}$ and the state is $\alpha = \{\alpha 1, \ldots, \alpha n\}$, the Markov Chain Monte Carlo algorithm is:

1. Draw α given y and θ
   a. Use the Kalman filter which is a forward filter - backward sampler
   b. Draw α
2. Draw θ given α
   a. Given α [σu],[σv],[σw],[β,σε] are conditionally independent
   b. Independent priors on the time series σ's
   c. Spike and Slab prior on β

Spike and Slab is vital to the Markov Chain Monte Carlo calculations because it serves as the penalty function. The Markov Chain Monte Carlo calculations use this penalty and Bayesian induction algorithm over n iterations to do variable selection.

DSDP Applied:

Using the Quandl economic data API, I retrieved the 281 time series, available from the Federal Reserve Economic Data set, to use in predicting the probability of recession. The 281

variables were selected based on two complimentary ontologies available on the Federal Reserve Economic Data website. These variables vary greatly in start date, completeness, and frequency. By backfilling 0 values into the data for time series that began later than 1967, which is the first datapoint for y, I was able to pass valid data to the model without affecting the accuracy of the prediction. Additionally, I filled in missing month values from yearly and quarterly series using cubic spline interpolation. This noise created more variance, but produced more accurate results. My dataset contains 571 month data-points from Jun 1967 to Jan 2015. The first 80% of months were used to train the model and the last 20% were used for testing. Before passing the data to the model, I normalized all the data to range from 0 to 1.

CausalImpact, the R function that creates a Bayesian Structural Time Series model which, in turn, runs the Spike Slab and Markov Chain Monte Carlo calculations, accepts a few optional parameters: the number of iterations, the number and length of seasons, the confidence interval, and a custom Bayesian Structural Time Series model. After meticulously experimenting with each parameter, the only parameter I changed from its default value was “alpha”, the confidence interval. By default, the confidence interval is 95%, but by changing it to 90% I was able to significantly improve the predictive accuracy of the model. No significant improvement could be made by increasing the number of iterations or providing a custom Bayesian Structural Time Series model. By setting the number of seasons to two with a length of 6 months I was able to improve the accuracy of the prediction slightly, but at the cost of a large increase in variance. Additionally, to gain a better understanding of the inner workings of the Bayesian Structural Time Series model itself, I built a custom Bayesian Structural Time Series model with many custom parameters set and ran the prediction myself without CausalImpact, achieving almost identical results.

Once the modeling was complete, I used Daniell Kernel Smoothing with a window size of 10 to remove much of the unnecessary noise around zero. However, as analyzing the data and optimizing the model further, I believe in most cases kernel smoothing is not ideal for this problem. Following that, I replaced all the negative values with zero and all the values over one with one, to stay within the range of possibility. The results below show a model that is quite accurate between 1 and 18 months into the future. The model coefficients contain a value representing the usefulness of each of the 281 variables to the final calculation. Unsurprisingly, the variable selection embedded in the model itself showed that around 70 variables were included in the model more than 10% of the time, for a list of selected variables, see below. I am certain there are other variables, within the roughly 140,000 variables available in the Federal Reserve Economic Data set, that have strong predictive correlation to the probability of recession; however, the model already has strong predictive abilities using only a few variables. In fact, my model is able to successfully predict the “Great Recession” of 2008-2009 within one month, as shown in the graphs below. Changes to the one variable can been seen to have a noticeable effect on the prediction showing this model provides extremely intriguing counter-factual inference.

In order to expand my understanding and test the model further, I used the auto.arima R function which uses Akaike and Bayesian Information Criterion penalties to select the best model for each x time series. Once a model was selected, I used each to forecast 12 months of future values for each x. Using all the historical data as training data, I built custom a new Bayesian Structural Time Series model to predict future values of y. Because I only used data though Jan 2015 to train my original validation model I still have 11 months of historical data to compare to my results of y based on my predicted future values of each x. Once again, my results match actual almost perfectly, indicating a minor spike in the probability of recession happening right now, see graph 3.

Code:

```
library(Quandl)
library(forecast)
library(CausalImpact)
library(biganalytics)
Quandl.api_key("ti-yR6gd8be4x4yswvy4")

################## CONSTANTS ##################
priorSampleSize <- 32
expectedModelSize <- 3
expectedR2 <- 0.8
priorDataFrames <- 50
numberIterations <- 1000
variableSelectionMinimum <- 0.1
trainingPercent <- 0.87
normalize <- TRUE
frequency <- "monthly"
start <- "1967-06-01"
end <- "2015-01-01"
futureDataPoints <- 12

# Fill in Null Data and Normalize
standardizeData <- function(data){
	# Linearly Interpolate Missing Values
	data <- na.spline(data)

	# Normalize Variables as Index 0:1
	if(normalize)
	{
		data <- as.zoo(apply(data, MARGIN=2, FUN=function(x)(x-min(x))/diff(range(x))))
	}

	# Remove NAN Columns
	data <- data[, colSums(!is.nan(data)) != 0]

	# Change Name of Response Column
	names(data)[1] = "y"

	return(data)
}

# Build Matrix of Most Significant Variables
getSignificantVariables <- function(model){
   variableRelevance = as.big.matrix(model$coefficients)
   bestVariables = colmax(variableRelevance)
   variables = subset(bestVariables, bestVariables > variableSelectionMinimum)
   return(variables)
}

# Run Causual Impact
causalImpact <- function(completeY, completeData, postY, predictData, startPrediction, length){
	# Null Out Post Data
	impactTrainingData <- completeData
	impactTrainingData[,1][startPrediction : length] <- NA
	impactPreY <- completeY
	impactPreY[startPrediction : length] <- NA

	# Set Prior Sampling Configuration
	sdy <- sd(impactPreY, na.rm = TRUE)
	sd.prior <- SdPrior(sigma.guess = 0.01 * sdy,
```

```
	upper.limit = sdy,
	sample.size = priorSampleSize)

	# Setup 1 Season
	ss <- list()
	ss <- AddLocalLevel(ss, impactPreY, sigma.prior = sd.prior)

	# Define x and y
	formula <- paste0("y", " ~ .")

	# Build Model
	model <- bsts(formula, data = impactTrainingData, state.specification = ss,
	expected.model.size = expectedModelSize,
	expected.r2 = expectedR2,
	prior.df = priorDataFrames,
	save.prediction.errors = TRUE,
	niter = numberIterations, seed = 1, ping = 0)

	# Bayesian Structural Time Series Model Prediction
	result = predict(model, predictData, burn = SuggestBurn(.1, model), oldData)

	# Renormalize Values Outside Range 0:1
	if(normalize)
	{
		result$median[result$median < 0] <- 0
		result$median[result$median > 1] <- 1
	}

	# Causal Impact
	impact <- CausalImpact(bsts.model = model, post.period.response = postY, alpha = .9)

	# Kernel Smoothing
	# kernel <- kernel("daniell", 3)
	# smooth <- kernapply(as.matrix(impact$series$point.pred), kernel)
	# plot.ts(smooth)

	impact$series$point.pred[impact$series$point.pred < 0] <- 0
	impact$series$point.pred[impact$series$point.pred > 1] <- 1
	impact$series$point.pred.lower[impact$series$point.pred.lower < 0] <- 0
	impact$series$point.pred.lower[impact$series$point.pred.lower > 1] <- 1
	impact$series$point.pred.upper[impact$series$point.pred.upper < 0] <- 0
	impact$series$point.pred.upper[impact$series$point.pred.upper > 1] <- 1

	# Plot Model Fit
	plot(impact, "original")
	return(impact)
}

# Run Validation Prediction
validatePrediction <- function(completeY, completeData, postY, predictData, endTraining, length){
	# Remove Post Data
	predictTrainingData <- completeData[c(1 : endTraining),]
	predictPreY <- predictTrainingData[,1]

	# Set Prior Sampling Configuration
	sdy <- sd(predictPreY, na.rm = TRUE)
	sd.prior <- SdPrior(sigma.guess = 0.01 * sdy,
	upper.limit = sdy,
	sample.size = priorSampleSize)

	# Setup 1 Season
	ss <- list()
```

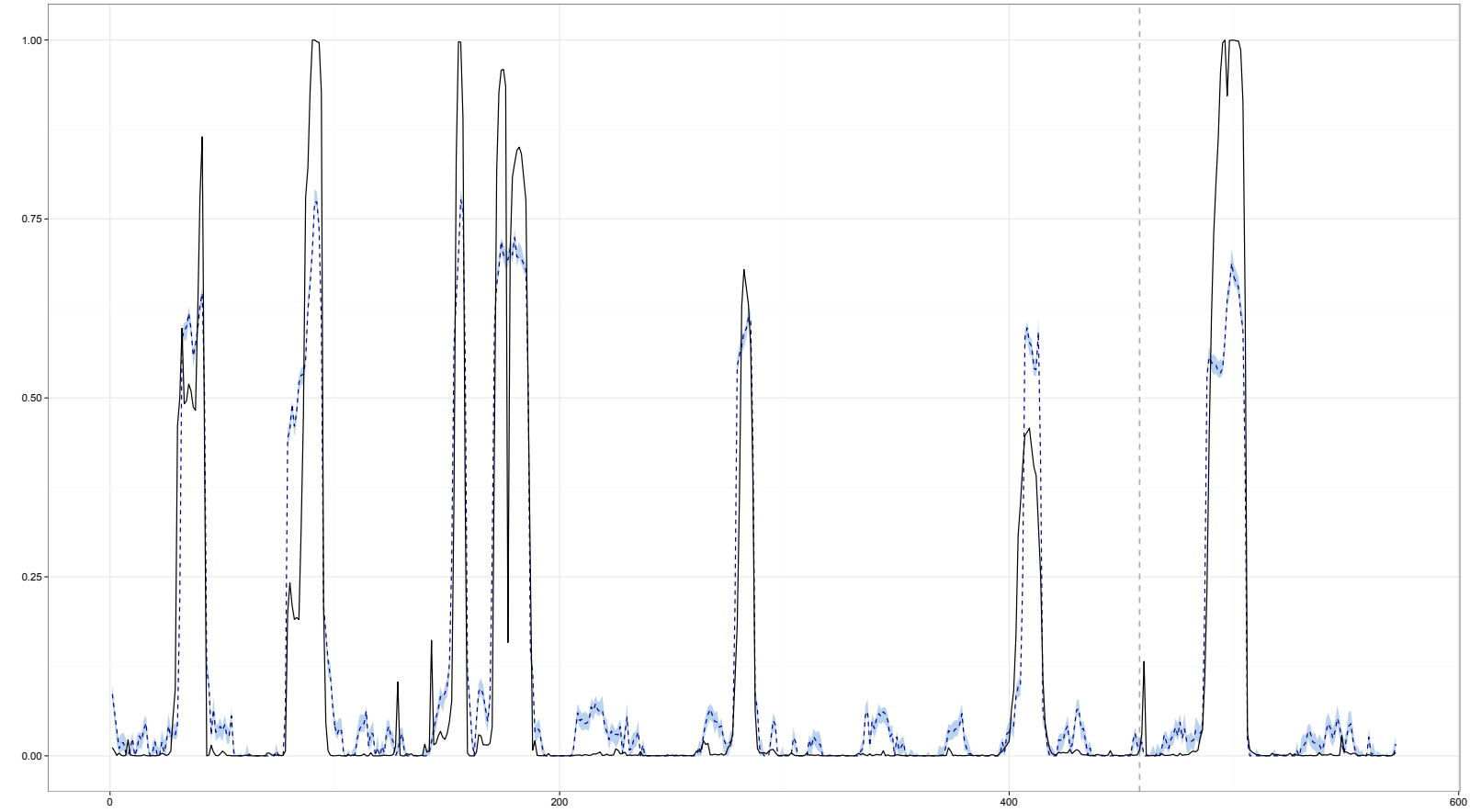
```
	ss <- AddLocalLevel(ss, predictPreY, sigma.prior = sd.prior)

	# Define x and y
	formula <- paste0("y", " ~ .")

	# Build Model
	model <- bsts(formula, data = predictTrainingData, state.specification = ss,
	expected.model.size = expectedModelSize,
	expected.r2 = expectedR2,
	prior.df = priorDataFrames,
	save.prediction.errors = TRUE,
	niter = numberIterations, seed = 1, ping = 0)

	# Bayesian Structural Time Series Model Prediction
	result = predict(model, predictData, burn = SuggestBurn(.1, model), oldData)

	# Renormalize Values Outside Range 0:1
	if(normalize)
	{
		result$median[result$median < 0] <- 0
		result$median[result$median > 1] <- 1
	}

	# Plot Model Fit
	if(normalize)
	{
		if(endTraining != length)
		{
			par(mfrow=c(2,1))
			plot(completeY, ylab = NULL, xlab = "Time", ylim = c(0,1))
		}
		plot(result, interval.width = 0, ylim = c(0,1))
	} else {
		if(endTraining != length)
		{
			par(mfrow=c(2,1))
			plot(completeY, ylab = NULL, xlab = "Time")
		}
		plot(result, interval.width = 0)
	}
	return(result)
}

# Run Future Prediction
futurePrediction <- function(completeY, completeData, length){
	# Extrapolate each Indicator Variable
	ncol <- NCOL(completeData)
	forecast <- matrix(NA, nrow = futureDataPoints, ncol = ncol)
	for(i in 1 : ncol)
	{
		fit <- auto.arima(completeData[,i])
		forecast[,i] <- forecast(fit, h = futureDataPoints)$mean
	}

	# Extract and Name y Variable
	forecastY <- forecast[,1]
	dimnames(forecast) <- list(rownames(forecast, do.NULL = FALSE, prefix = "row"), colnames(forecast, do.NULL = FALSE,
prefix = "col"))
	colnames(forecast)[colnames(forecast) == "col1"] <- "y"

	# Run Model
	result <- validatePrediction(completeY, completeData, forecastY, forecast, length, length)
```

```
	return(result)
}

################## MAIN ##################
responseVariable <- "FRED/RECPROUSM156N"
indicatorVariables <- c("FRED/ACOILBRENTEU","FRED/ACOILWTICO","FRED/AHETPI","FRED/AISRSA","FRED/
ASEANTOT","FRED/AUINSA","FRED/AWHAETP","FRED/BAA10Y","FRED/BUSINV","FRED/CANTOT","FRED/CBI","FRED/
CDSP","FRED/CES0500000003","FRED/CEU0500000002","FRED/CEU0500000003","FRED/CEU0500000008","FRED/
CHNTOT","FRED/CIVPART","FRED/CNP16OV","FRED/COMPHAI","FRED/COMPNFB","FRED/COMPRNFB","FRED/
COMREPUSQ159N","FRED/CPIAUCNS","FRED/CPIAUCSL","FRED/DCOILBRENTEU","FRED/DCOILWTICO","FRED/
DDDM01USA156NWDB","FRED/DED1","FRED/DED3","FRED/DED6","FRED/DEXBZUS","FRED/DEXCAUS","FRED/
DEXCHUS","FRED/DEXJPUS","FRED/DEXKOUS","FRED/DEXMXUS","FRED/DEXNOUS","FRED/DEXSDUS","FRED/
DEXSFUS","FRED/DEXSIUS","FRED/DEXSZUS","FRED/DEXUSAL","FRED/DEXUSEU","FRED/DEXUSNZ","FRED/
DEXUSUK","FRED/DGORDER","FRED/DGS10","FRED/DSPIC96","FRED/DSWP1","FRED/DSWP10","FRED/DSWP2","FRED/
DSWP3","FRED/DSWP30","FRED/DSWP4","FRED/DSWP5","FRED/DSWP7","FRED/DTWEXB","FRED/DTWEXM","FRED/
ECIWAG","FRED/ECOMNSA","FRED/ECOMSA","FRED/EECTOT","FRED/EMRATIO","FRED/ETOTALUSQ176N","FRED/
EVACANTUSQ176N","FRED/FEDFUNDS","FRED/FRNTOT","FRED/FYFSGDA188S","FRED/GASREGCOVM","FRED/
GASREGCOVW","FRED/GASREGM","FRED/GASREGW","FRED/GCT1501US","FRED/GCT1502US","FRED/GCT1503US","FRED/
GERTOT","FRED/GOLDAMGBD228NLBM","FRED/GOLDPMGBD228NLBM","FRED/HCOMPBS","FRED/HDTGPDUSQ163N","FRED/
HOABS","FRED/HOANBS","FRED/HOUST","FRED/HPIPONM226N","FRED/HPIPONM226S","FRED/IC4WSA","FRED/
INDPRO","FRED/INTDSRUSM193N","FRED/IPBUSEQ","FRED/IPDBS","FRED/IPMAN","FRED/IPMAT","FRED/IPMINE","FRED/
IR","FRED/IR10010","FRED/IREXPET","FRED/ISRATIO","FRED/JCXFE","FRED/JPNTOT","FRED/JTS1000HIL","FRED/
JTS1000HIR","FRED/JTSHIL","FRED/JTSHIR","FRED/JTSJOL","FRED/JTSJOR","FRED/JTSLDL","FRED/JTSLDR","FRED/
JTSQUL","FRED/JTSQUR","FRED/JTSTSL","FRED/JTSTSR","FRED/JTU1000HIL","FRED/JTU1000HIR","FRED/JTUHIL","FRED/
JTUHIR","FRED/JTUJOL","FRED/JTUJOR","FRED/JTULDL","FRED/JTULDR","FRED/JTUQUL","FRED/JTUQUR","FRED/
JTUTSL","FRED/JTUTSR","FRED/LNS12032194","FRED/LNS12032196","FRED/LNS14027660","FRED/LNS15000000","FRED/
LNU05026642","FRED/M12MTVUSM227NFWA","FRED/M2V","FRED/MCOILBRENTEU","FRED/MCOILWTICO","FRED/
MCUMFN","FRED/MEHOINUSA646N","FRED/MEHOINUSA672N","FRED/MFGOPH","FRED/MFGPROD","FRED/
MNFCTRIRNSA","FRED/MNFCTRIRSA","FRED/MNFCTRMPCSMNSA","FRED/MNFCTRMPCSMSA","FRED/
MNFCTRSMNSA","FRED/MNFCTRSMSA","FRED/MYAGM2USM052N","FRED/MYAGM2USM052S","FRED/NAPM","FRED/
NAPMBI","FRED/NAPMCI","FRED/NAPMEI","FRED/NAPMEXI","FRED/NAPMII","FRED/NAPMIMP","FRED/NAPMNOI","FRED/
NAPMPI","FRED/NAPMPRI","FRED/NAPMSDI","FRED/NILFWJN","FRED/NILFWJNN","FRED/NMFBAI","FRED/NMFBI","FRED/
NMFCI","FRED/NMFEI","FRED/NMFEXI","FRED/NMFIMI","FRED/NMFINI","FRED/NMFINSI","FRED/NMFNOI","FRED/
NMFPI","FRED/NMFSDI","FRED/NROU","FRED/NROUST","FRED/OPHMFG","FRED/OPHNFB","FRED/OPHPBS","FRED/
OUTBS","FRED/OUTMS","FRED/OUTNFB","FRED/PAYEMS","FRED/PAYNSA","FRED/PCE","FRED/PCECTPICTM","FRED/
PCEPI","FRED/PCEPILFE","FRED/PCETRIM12M159SFRBDAL","FRED/PCETRIM1M158SFRBDAL","FRED/PNRESCON","FRED/
PNRESCONS","FRED/POP","FRED/POPTHM","FRED/PPIACO","FRED/PRRESCON","FRED/PRRESCONS","FRED/PRS30006013",
"FRED/PRS30006023","FRED/PRS84006013","FRED/PRS84006023","FRED/PRS84006163","FRED/PRS84006173","FRED/
PRS85006023","FRED/PRS85006163","FRED/PRS85006173","FRED/RCPHBS","FRED/RETAILIMSA","FRED/RETAILIRSA","FRED/
RETAILMPCSMNSA","FRED/RETAILMPCSMSA","FRED/RETAILSMNSA","FRED/RETAILSMSA","FRED/RHORUSQ156N","FRED/
RIFLPCFANNM","FRED/RPI","FRED/RRSFS","FRED/RSAFS","FRED/RSAFSNA","FRED/RSAHORUSQ156S","FRED/
RSEAS","FRED/RSFSXMV","FRED/RSNSR","FRED/RSXFS","FRED/T10Y2Y","FRED/T10Y3M","FRED/T10YFF","FRED/
T10YIEM","FRED/T5YIEM","FRED/T5YIFR","FRED/TB3SMFFM","FRED/TCU","FRED/TDSP","FRED/TEDRATE","FRED/
TLCOMCON","FRED/TLCOMCONS","FRED/TLNRESCON","FRED/TLNRESCONS","FRED/TLPBLCON","FRED/
TLPBLCONS","FRED/TLPRVCON","FRED/TLPRVCONS","FRED/TLRESCON","FRED/TLRESCONS","FRED/
TOTBUSIMNSA","FRED/TOTBUSIRNSA","FRED/TOTBUSMPCIMNSA","FRED/TOTBUSMPCIMSA","FRED/
TOTBUSMPCSMNSA","FRED/TOTBUSMPCSMSA","FRED/TOTBUSSMNSA","FRED/TOTBUSSMSA","FRED/
TOTDTEUSQ163N","FRED/TRFVOLUSM227NFWA","FRED/TTLCON","FRED/TTLCONS","FRED/U4RATE","FRED/
U4RATENSA","FRED/U6RATE","FRED/U6RATENSA","FRED/UEMPMED","FRED/UKTOT","FRED/ULCBS","FRED/ULCMFG","FRED/
ULCNFB","FRED/UNRATE","FRED/USAGDPDEFAISMEI","FRED/USAGDPDEFQISMEI","FRED/USAGFCFADSMEI","FRED/
USAGFCFQDSMEI","FRED/USAGFCFQDSNAQ","FRED/USARECDM","FRED/USARGDPC","FRED/USASACRAISMEI","FRED/
USASACRMISMEI","FRED/USASACRQISMEI","FRED/USPRIV","FRED/USRECD","FRED/USRECDM","FRED/USSLIND","FRED/
USSTHPI","FRED/WCOILBRENTEU","FRED/WCOILWTICO","FRED/WHLSLRIRNSA","FRED/WHLSLRIRSA")

# Pull Data
rawData <- Quandl(c(toString(responseVariable), indicatorVariables),
	start_date = toString(start), end_date = toString(end), type = "zoo",
	collapse = toString(frequency), transform = "normalize")

completeData <- standardizeData(rawData)

# Get Split Point
```

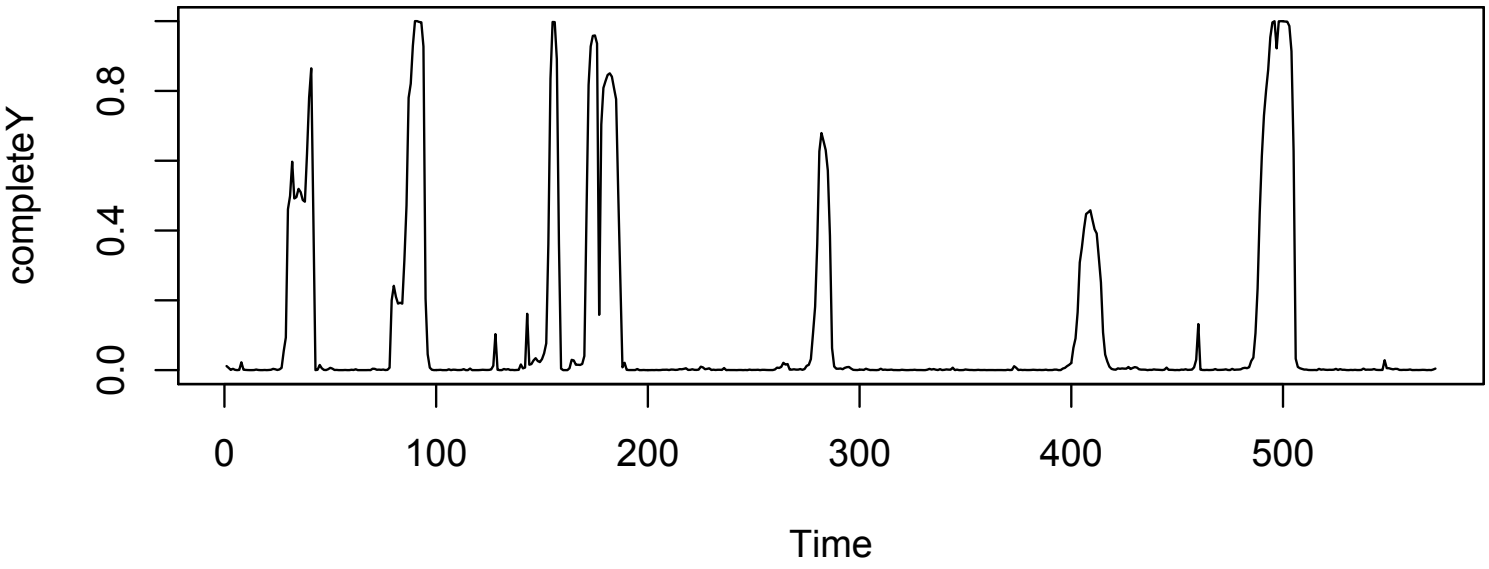
```
length <- NROW(completeData)
endTraining <- round(length * trainingPercent)

# Get Y
completeY <- completeData[, 1]

# Split Off Data Used for Prediction
startPrediction <- endTraining + 1
predictData <- completeData[c(startPrediction : length),]
postY <- as.vector(completeY[startPrediction : length])

impact <- causalImpact(completeY, completeData, postY, predictData, startPrediction, length)
validation <- validatePrediction(completeY, completeData, postY, predictData, endTraining, length)
future <- futurePrediction(completeY, completeData, length)
```

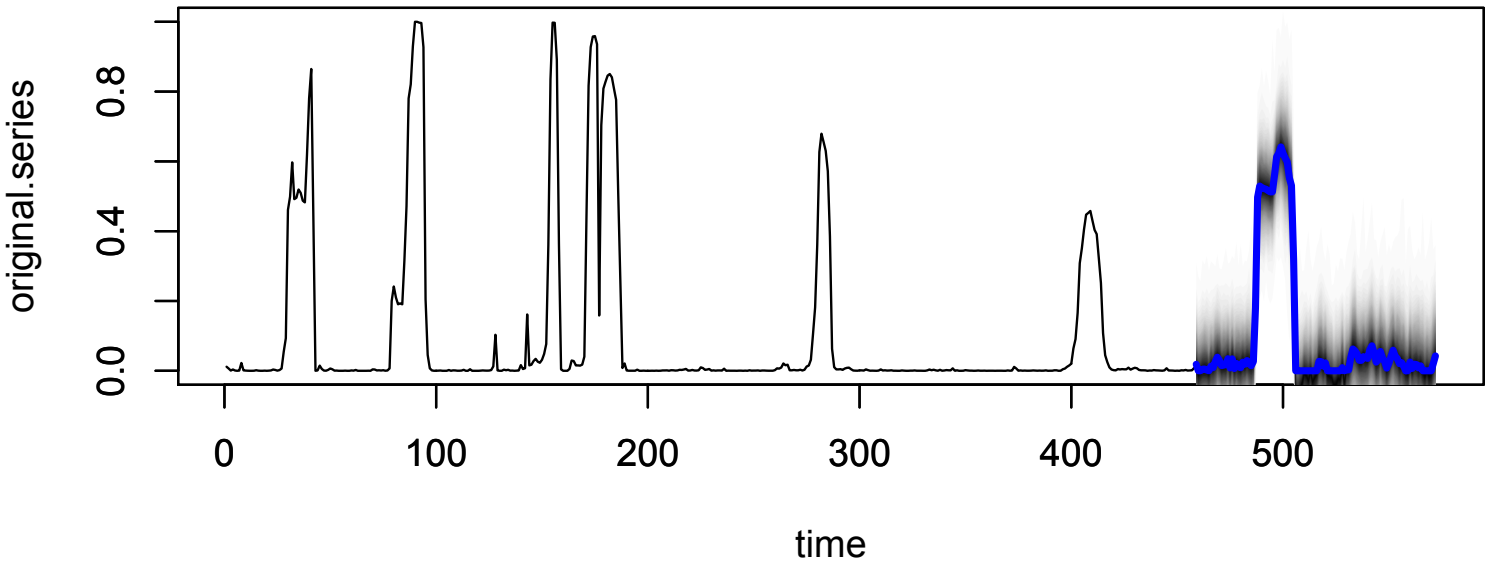
Output:

MODEL FIT
Black Line: Probability of Recession
Dashed Blue: Model Fit
Light Blue: Prediction Variance
Dashed Grey Vertical Line: Separation of Pre/Post Period

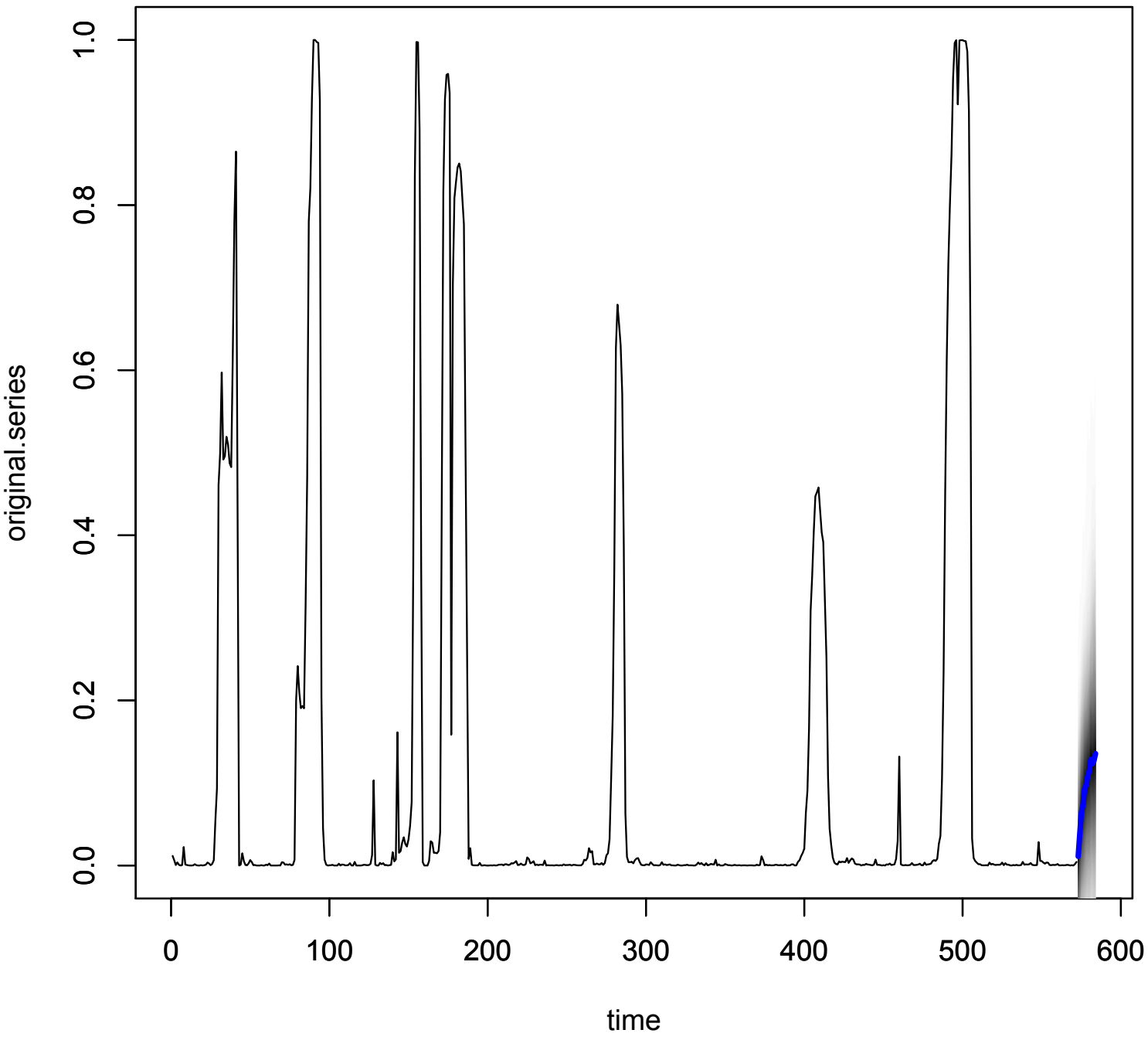


MODEL VALIDATION





PREDICTION BEYOND KNOWN



KEY VARIABLES:
Search Name Here for Variable Description: https://www.quandl.com/data/FRED.
The Higher the Value Below the Name the More Valuable it is to the Model.

ACOILWTICO	AISRSA
1.1138899	0.2664120
ASEANTOT	CBI
0.6603042	0.3109936
CES0500000003	CHNTOT
0.4424198	1.0038771
DSWP1	DSWP10
0.6931829	0.2227537
DSWP30	DSWP7
1.3205573	0.3251478
DTWEXB	FRNTOT
0.6046978	0.5421680
GASREGM	GCT1503US
1.5938929	0.9286644
IC4WSA	INDPRO
0.4048687	0.3264300
JPNTOT	JTS1000HIL
0.7315125	0.2687866
JTS1000HIR	JTSHIL
0.4620399	1.4110502
JTSQUR	JTUJOL
0.5653641	0.7718354
JTUJOR	JTULDR
0.3717774	0.5612181
JTUTSL	JTUTSR
3.5400850	0.7670839
MEHOINUSA672N	MFGOPH
0.3691075	0.4845907
MNFCTRIRSA	NAPMBI
0.7361995	3.4774070
NAPMCI	NAPMIMP
1.4193703	0.4805800
NAPMPRI	NILFWJNN
0.2347718	1.3697178
NMFBAI	NMFEXI
1.3177990	0.8726264
NMFIMI	NMFNOI
4.6654468	1.0529353
NMFPI	OUTMS
3.2064183	0.4349042
PRRESCON	PRRESCONS
10.4042361	1.0517379
RETAILIMSA	RETAILMPCSMNSA
0.8572209	1.2598410
RETAILMPCSMSA	RETAILSMNSA
0.2540421	0.3256416
RETAILSMSA	RSAFS
0.3773479	0.3314937

RSEAS	RSNSR
0.8689523	1.3146534
T10YFF	T10YIEM
0.2205214	1.2436694
T5YIEM	TLCOMCON
1.3124573	1.4526662
TLNRESCONS	TLPBLCON
0.8672961	1.5401124
TLPRVCONS	TLRESCON
1.6656682	1.1183665
TLRESCONS	TOTBUSIMNSA
0.3011249	0.2034188
TOTBUSIRNSA	TOTBUSMPCIMSA
0.5593249	0.7155146
TOTBUSMPCSMNSA	TOTBUSSMNSA
2.6182098	1.7845847
TTLCON	TTLCONS
0.2723859	1.6979103
U6RATE	U6RATENSA
0.9933442	0.4272988
UKTOT	ULCBS
1.1109316	0.5529749
USASACRAISMEI	USRECD
0.4287335	0.4752101
USRECDM	WCOILWTICO
0.2558860	0.3541560
WHLSLRIRNSA	WHLSLRIRSA
0.3994067	1.1229200

References

http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/41854.pdf
http://statmath.wu.ac.at/research/talks/resources/slidesscott.pdf
https://research.stlouisfed.org/fred2/
https://www.quandl.com/tools/r
http://jeremykun.com/2015/04/06/markov-chain-monte-carlo-without-all-the-bullshit/
http://robjhyndman.com/talks/MelbourneRUG.pdf
https://github.com/cran/bsts
https://github.com/google/CausalImpact
https://cran.r-project.org/web/packages/forecast/forecast.pdf
https://cran.r-project.org/web/packages/biganalytics/biganalytics.pdf
http://www.inside-r.org/r-doc/stats/kernel
https://www.khanacademy.org/math/probability/probability-and-combinatorics-topic/probability_combinatorics/v/conditional-probability-and-combinations
https://www.youtube.com/watch?v=12eZWG0Z5gY
http://www-stat.wharton.upenn.edu/~edgeorge/Research_papers/GeorgeMcCulloch97.pdf
http://xyala2.bio.ed.ac.uk/teaching/tutorials/phylogenetics/Bayesian_Workshop/PDFs/Posada%20and%20Buckley%20Syst%20Biol%202004.pdf
http://www.stat.purdue.edu/~mlevins/STAT598K_2012/Box_Pierce_1970.pdf
http://xyala2.bio.ed.ac.uk/teaching/tutorials/phylogenetics/Bayesian_Workshop/PDFs/Alfaro%20et%20al%20Mol%20Biol%20Evol%202003.pdf
http://www.stat.berkeley.edu/~aditya/resources/LectureTWENTYSIX.pdf
http://arxiv.org/pdf/math/0505633.pdf