

Markov Chain Monte Carlo Diagnostics



For ignoring the MCMC diagnostics, he was sentenced to the Markov chain gang.

Additional Topics

Topics

BSTS

- Introduction
- State space models
- Fundamentals of BSTS
- Example (Level and Trend)
- Seasonality
- Example (Seasonality)

Change-point

- Introduction
- Techniques
- Examples

BTST

BSTS

Uses the Bayesian setting (distributions on parameters) and state space (think Exponential Smoothing models) together.

Was developed by Google around 2013

Easy to implement and run in R

R documentation: <https://cran.r-project.org/web/packages/bsts/bsts.pdf>

State Space models:

Recall Holt-Winters Additive ESM

$$\hat{Y}_{t+h} = L_t + hT_t + S_{t-p+h}$$

$$L_t = \theta(Y_t - S_{t-p}) + (1 - \theta)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1 - \gamma)T_{t-1}$$

$$S_t = \delta(Y_t - L_{t-1} - T_{t-1}) + (1 - \delta)S_{t-p}$$

Recall Holt-Winters Additive ESM

$$\hat{Y}_{t+h} = L_t + hT_t + S_{t-p+h}$$

$$L_t = \theta(Y_t - S_{t-p}) + (1 - \theta)(L_{t-1} + T_{t-1})$$

$$T_t = \gamma(L_t - L_{t-1}) + (1 - \gamma)T_{t-1}$$

$$S_t = \delta(Y_t - L_{t-1} - T_{t-1}) + (1 - \delta)S_{t-p}$$

Each of these components evolve over time....

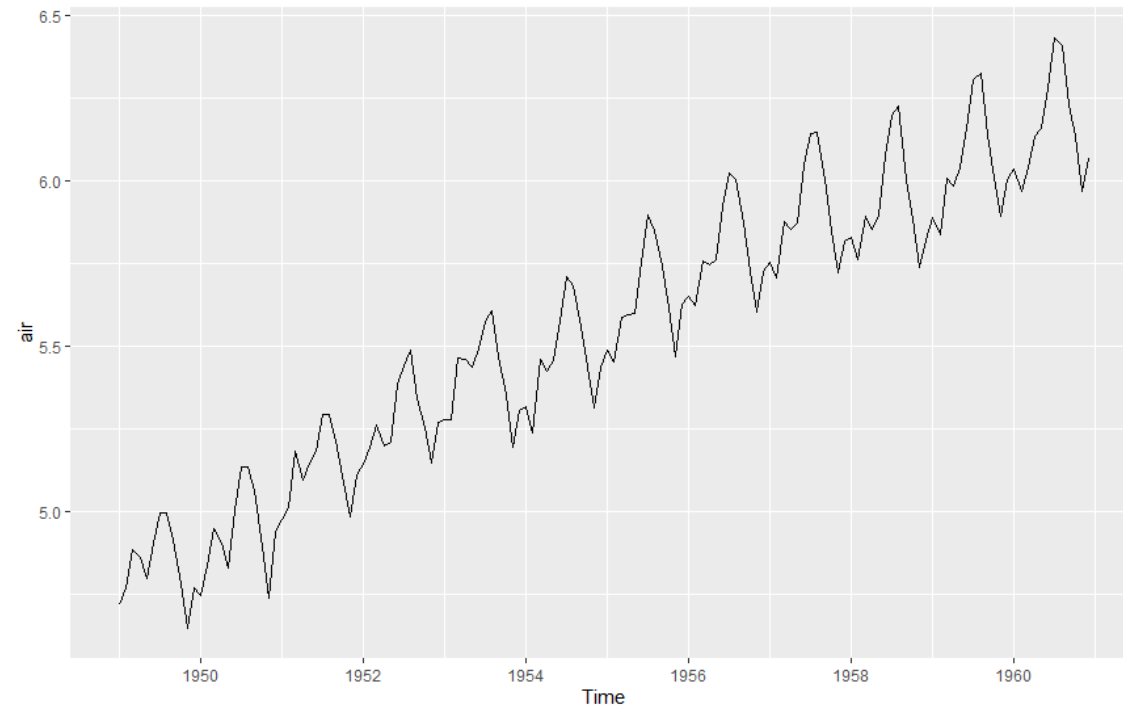
Simple “level” model

$$y_t = \underbrace{\mu_t}_{\text{Level}} + \epsilon_t$$

$$\mu_{t+1} = \mu_t + u_t \quad \text{Level } (u_t \text{ is error....assumes Normal distribution})$$

Example (airline passengers...older data)

Use the older airline passenger data (we will use the $\text{Log}(\text{passenger})$ as our time series)

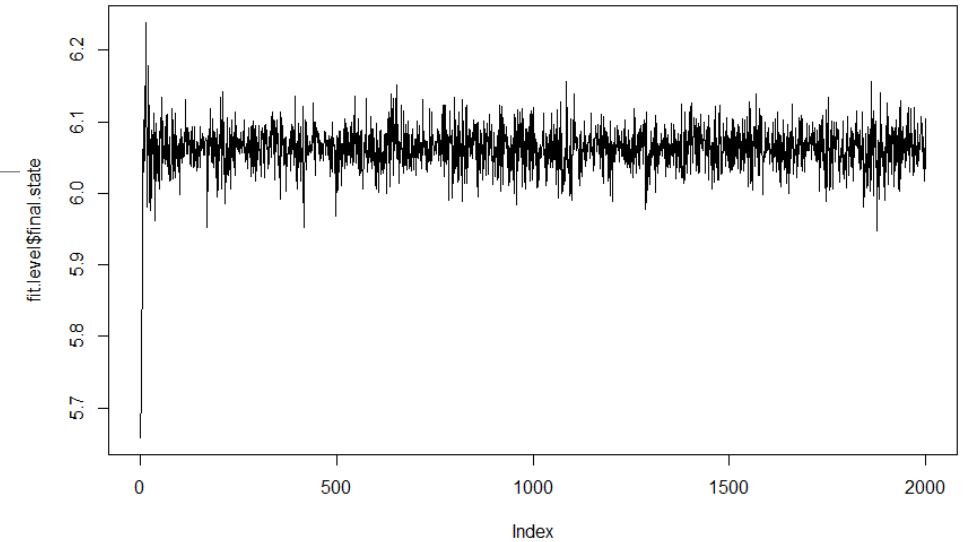


Fitting a level BSTS

```
library(bsts)  
library(tidyverse)  
air.bsts=airline$LogPsngr
```

```
model_components=list()
```

```
model_components <- AddLocalLevel(model_components, y = air.bsts)  
fit.level=bsts(air.bsts, model_components, niter = 2000)  
plot(fit.level$final.state,type='l')
```

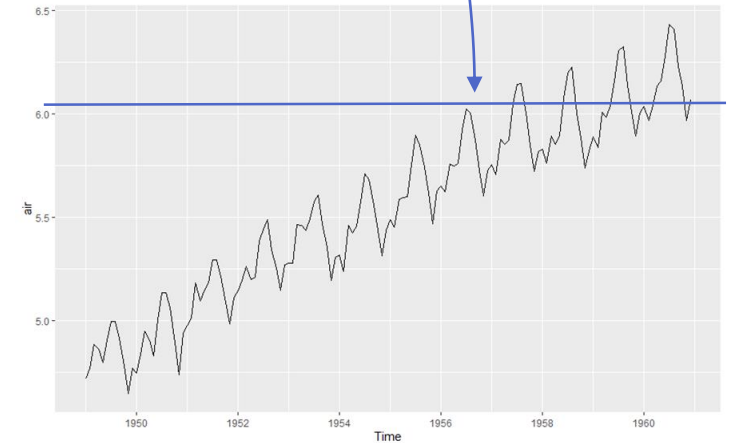
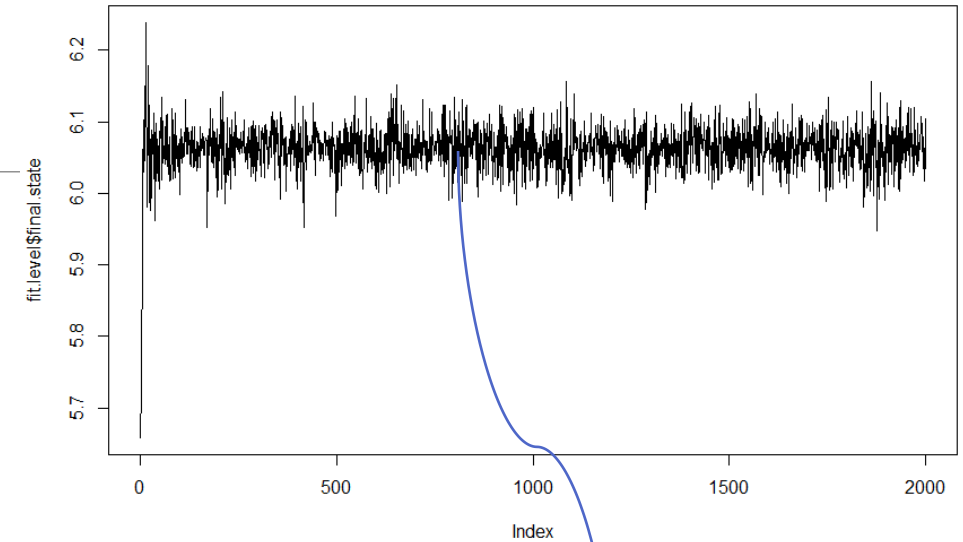


Fitting a level BSTS

```
library(bsts)  
library(tidyverse)  
air.bsts=airline$LogPsngr
```

```
model_components=list()
```

```
model_components <- AddLocalLevel(model_components, y = air.bsts)  
fit.level=bsts(air.bsts, model_components, niter = 2000)  
plot(fit.level$final.state,type='l')
```

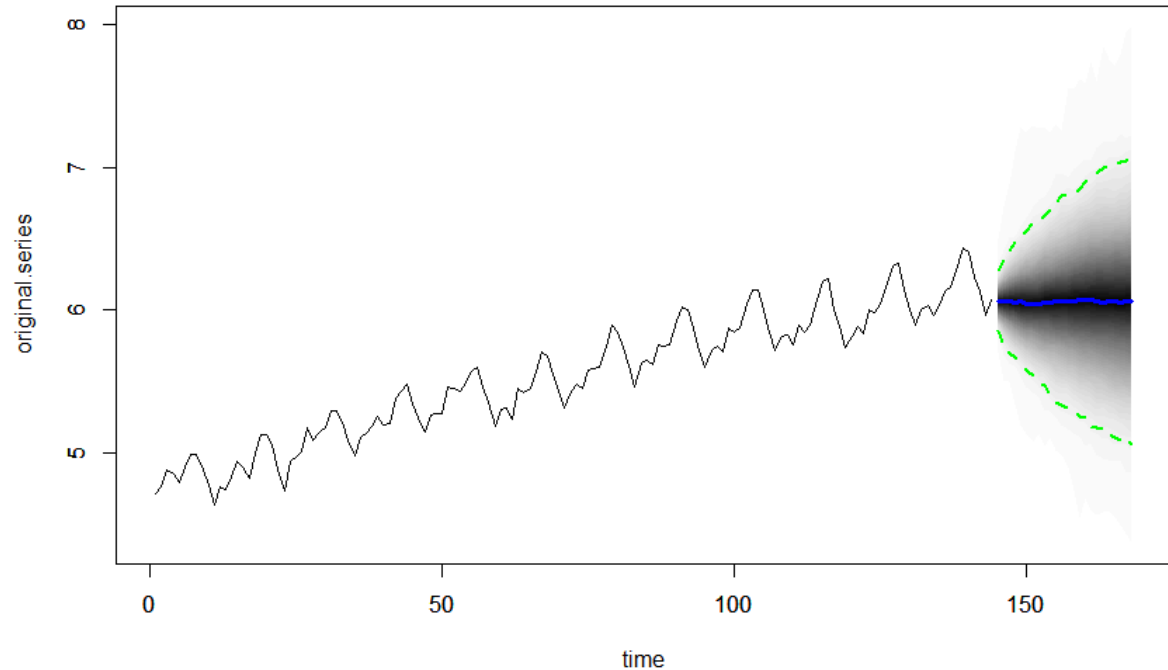


Making forecasts

```
pred.level<-predict(fit.level,burn = 200,horizon = 24)  
plot(pred.level)
```

```
> pred.level$mean  
6.060904 6.058905 6.059396 6.057654  
6.057066 6.052285 6.053708 6.053631  
6.057384 6.059317 6.059745 6.061799  
6.063273 6.064737 6.064800 6.067155  
6.070143 6.066728 6.066637 6.062760  
6.060922 6.056811 6.056758 6.058400
```

Can also ask for median and interval



Trend BSTS

$$y_t = \underbrace{\mu_t}_{\text{Level/Trend}} + \epsilon_t$$

Level/Trend

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

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



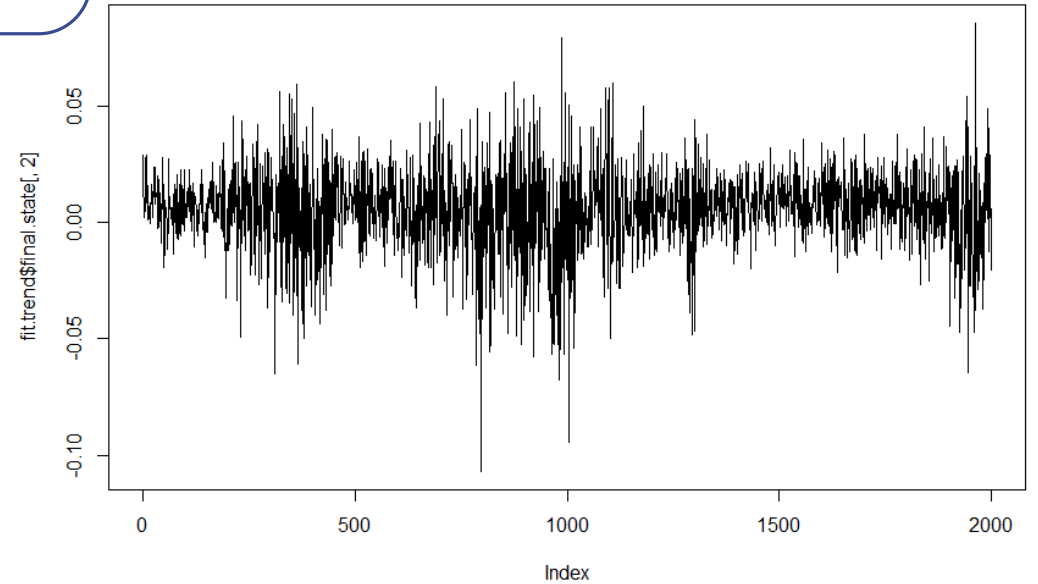
Level and Trend (u_t and v_t are error)

Fit a trend BSTS

```
model_components=list()
model_components=AddLocalLinearTrend(model_components,
                                     y = air.bsts)
fit.trend=bsts(air.bsts, model_components, niter = 2000)
plot(fit.trend$final.state[,2],type='l')
```

```
> head(fit.trend$final.state)
```

	[,1]	[,2]
[1,]	6.270559	0.028903721
[2,]	6.188468	0.015266010
[3,]	6.176237	0.006683758
[4,]	6.144252	0.002293991
[5,]	6.260150	0.005248778
[6,]	6.229388	0.005921211

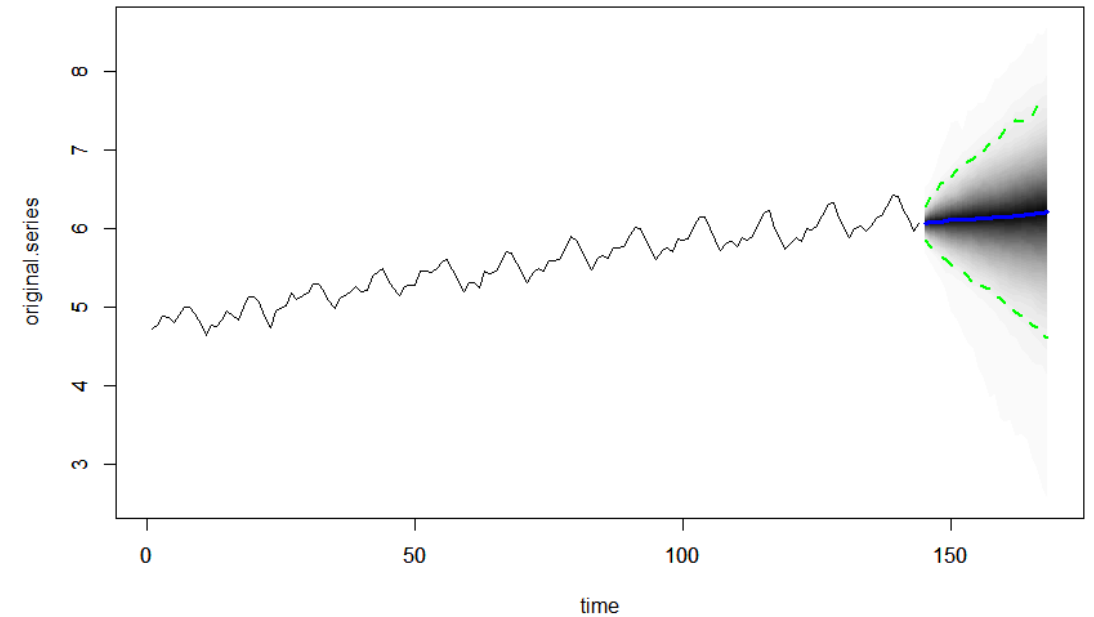


Plot forecasts

```
pred.trend<-predict(fit.trend,burn = 500,horizon = 24)  
plot(pred.trend)
```

```
> pred.trend$median
```

```
6.069794 6.080965 6.084830 6.088169 6.094706  
6.107259 6.112713 6.111001 6.114586 6.120687  
6.129326 6.123488 6.131942 6.145842 6.141558  
6.147235 6.145968 6.165901 6.164742 6.176605  
6.187887 6.196770 6.202536 6.215285
```



Seasonal BSTS

$$y_t = \underbrace{\mu_t}_{\text{Level}} + \underbrace{\tau_t}_{\text{Season}} + \epsilon_t$$

$$\left. \begin{aligned} \mu_{t+1} &= \mu_t + \delta_t + u_t \\ \delta_{t+1} &= \delta_t + v_t \end{aligned} \right\} \text{Level and Trend } (u_t \text{ and } v_t \text{ are error})$$

$$\tau_{t+1} = -\sum \tau_t + w_t \quad \text{Seasonality } (w_t \text{ is the error term})$$

How to model Seasonality

Dummy variables

Trigonometric Functions

Dummy variables

Basically, you are doing a linear regression!!! Let's say we have monthly data (i.e. we need 11 dummy variables)...

$$x_1 = \begin{cases} 1 & \text{if January} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if February} \\ 0 & \text{otherwise} \end{cases}$$

•

•

•

$$x_{11} = \begin{cases} 1 & \text{if November} \\ 0 & \text{otherwise} \end{cases}$$

Dummy variables

Basically, you are doing a linear regression!!! Let's say we have monthly data (i.e. we need 11 dummy variables)...

$$x_1 = \begin{cases} 1 & \text{if January} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if February} \\ 0 & \text{otherwise} \end{cases}$$

•

•

•

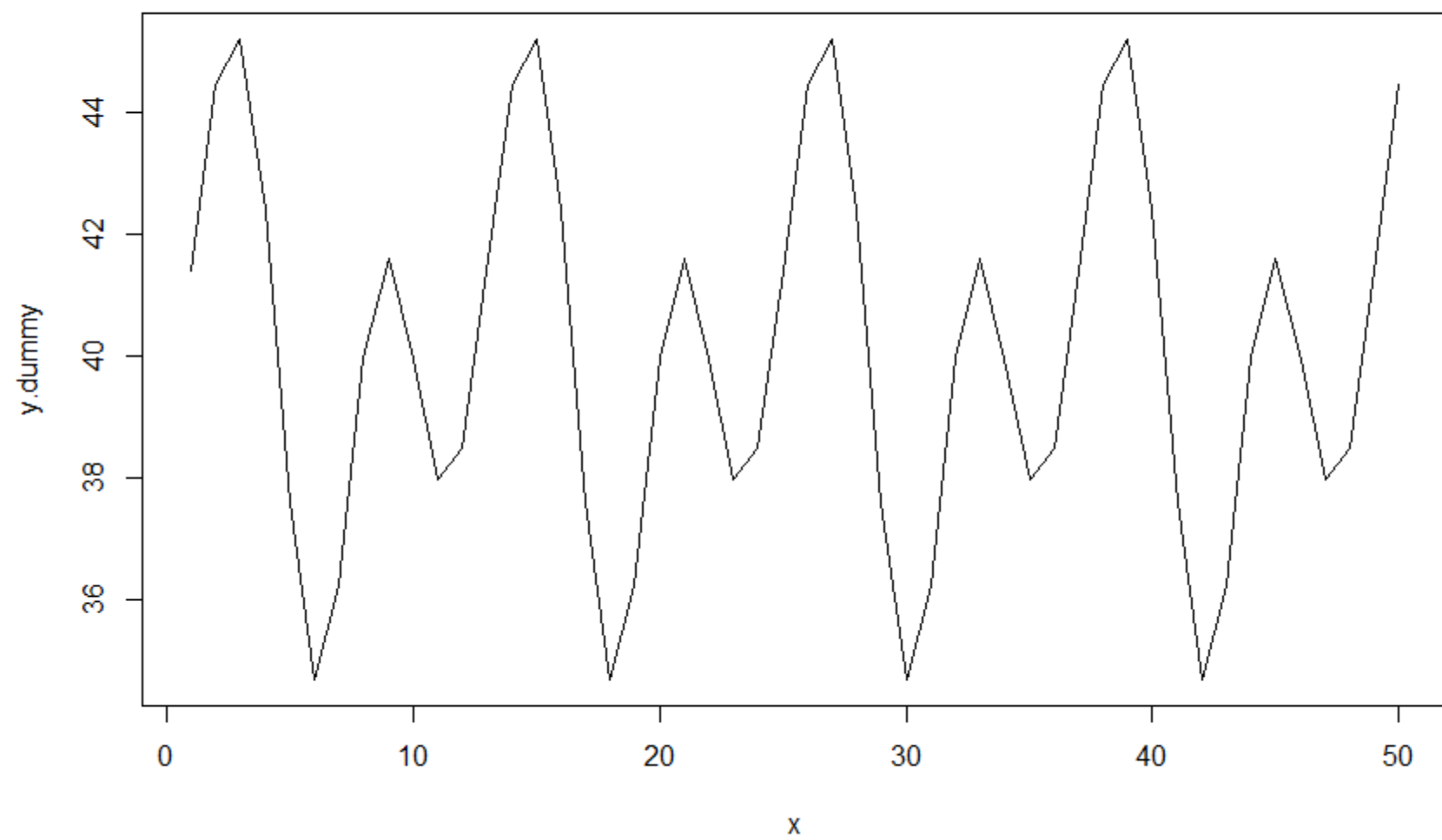
$$x_{11} = \begin{cases} 1 & \text{if November} \\ 0 & \text{otherwise} \end{cases}$$

$$Y_t = 38.5 + 2.9X_1 + 5.95X_2 + 6.7X_3 + 3.9124356X_4 - 0.91X_5 - 3.8X_6 - 2.27X_7 + 1.49X_8 + 3.1X_9 + 1.4X_{10} - 0.53X_{11}$$

x.dummy[1:3,]

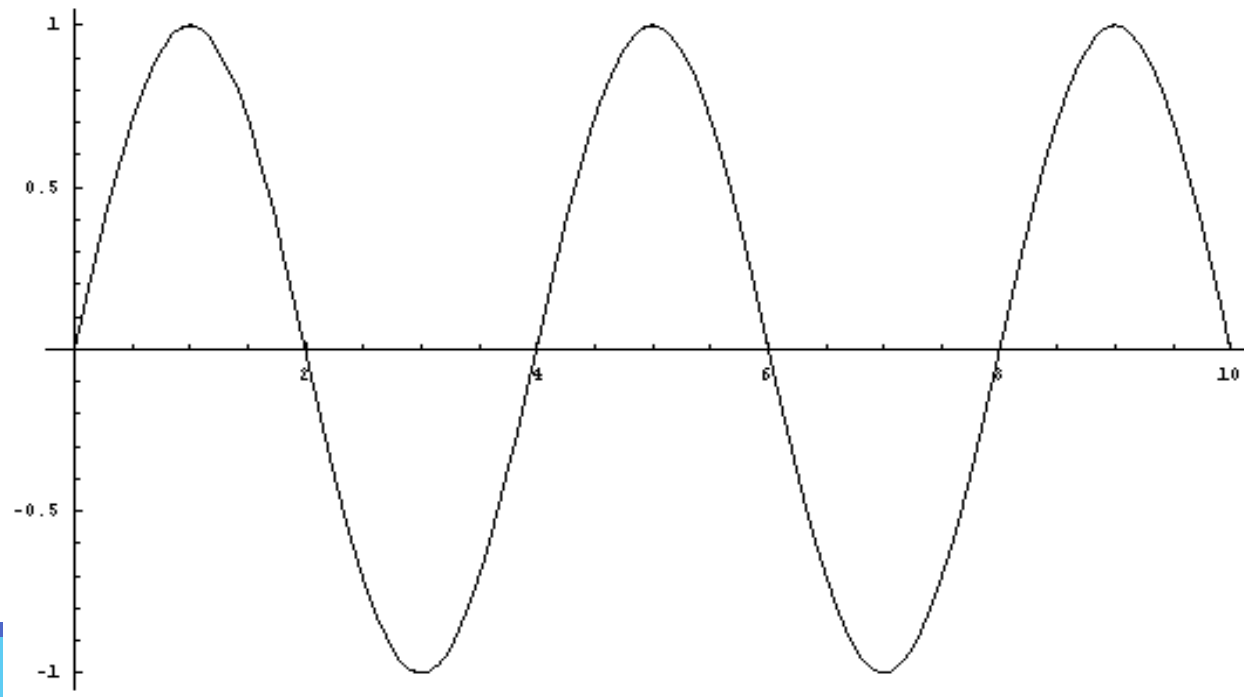
	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]
[1,]	1	0	0	0	0	0	0	0	0	0	0
[2,]	0	1	0	0	0	0	0	0	0	0	0
[3,]	0	0	1	0	0	0	0	0	0	0	0

```
y.dummy=38.5+x.dummy%*%c(2.9052559,5.9516660,6.7000000,3.9124356,-0.9052559,-3.8000000,-2.2660254,1.4875644,3.1000000,1.4483340,-0.5339746)
```



Trigonometric Functions

- Trigonometric functions in mathematics have a cyclical pattern.
- Use trigonometric functions, such as sine and cosine, to model seasonality.



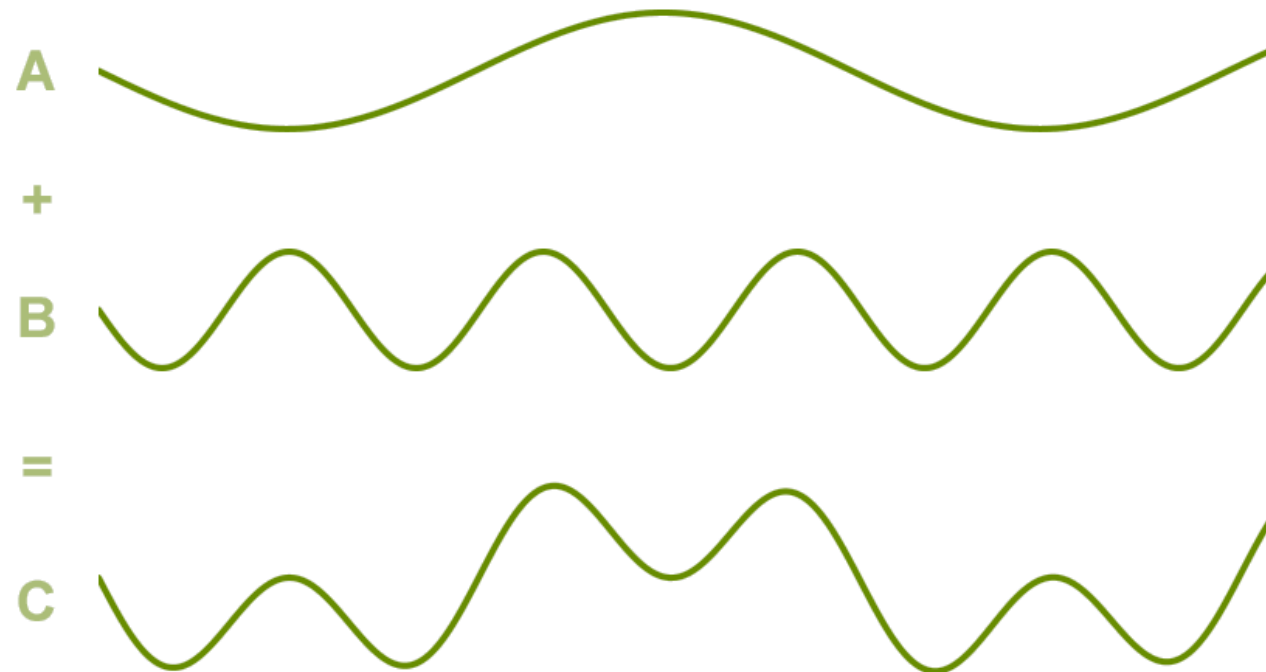
Trigonometric Regression

$$X_t = \sin\left(\frac{2\pi t}{S}\right)$$

$$Y_t = \beta_0 + \beta_1 X_t + \epsilon_t$$

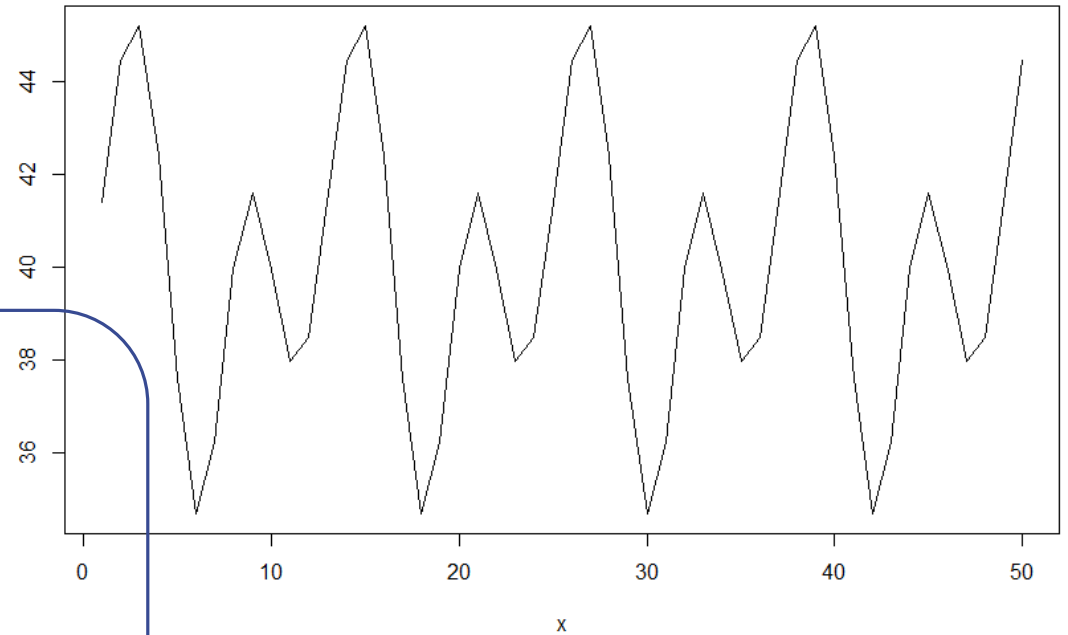
Trigonometric Regression

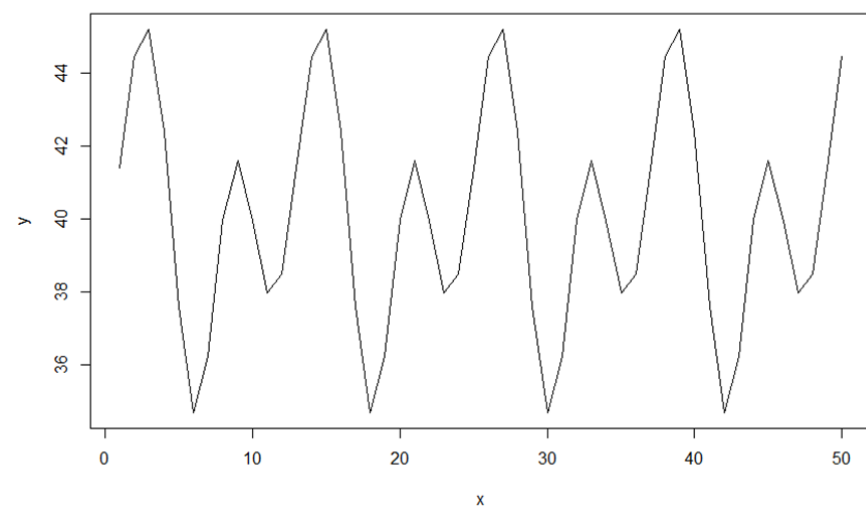
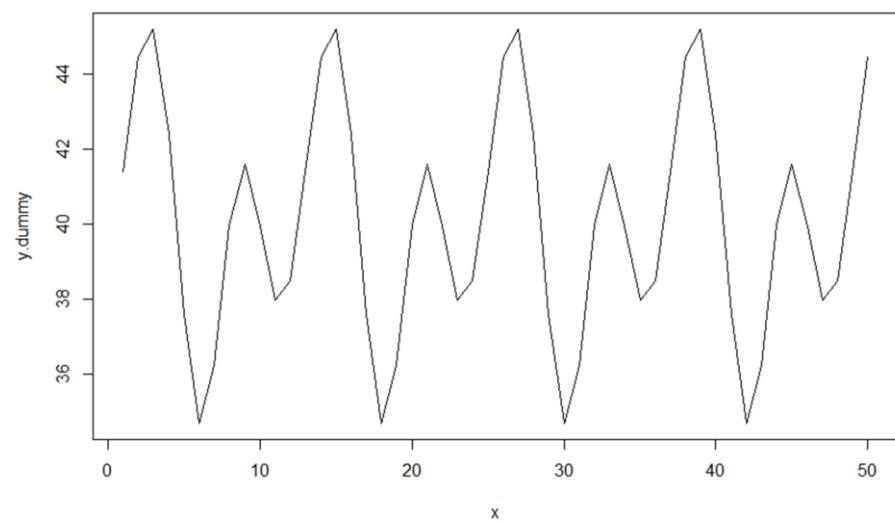
- You don't have to limit yourselves to only one sine or cosine variable.
- Mixing sine and cosine functions might better fit your data (Fourier analysis).



For example:

```
> x=seq(1,50,by=1)
> sin1=sin(1*2*pi*x/12)
> cos1=cos(1*2*pi*x/12)
> sin2=sin(2*2*pi*x/12)
> cos2=cos(2*2*pi*x/12)
> sin3=sin(3*2*pi*x/12)
> cos3=cos(3*2*pi*x/12)
> y=40+2*sin1+1.6*cos1+0.6*sin2-3.4*cos2+0.2*sin3+0.3*cos3
> plot(x,y,type='l')
```





Seasonal (with Dummy and Level)

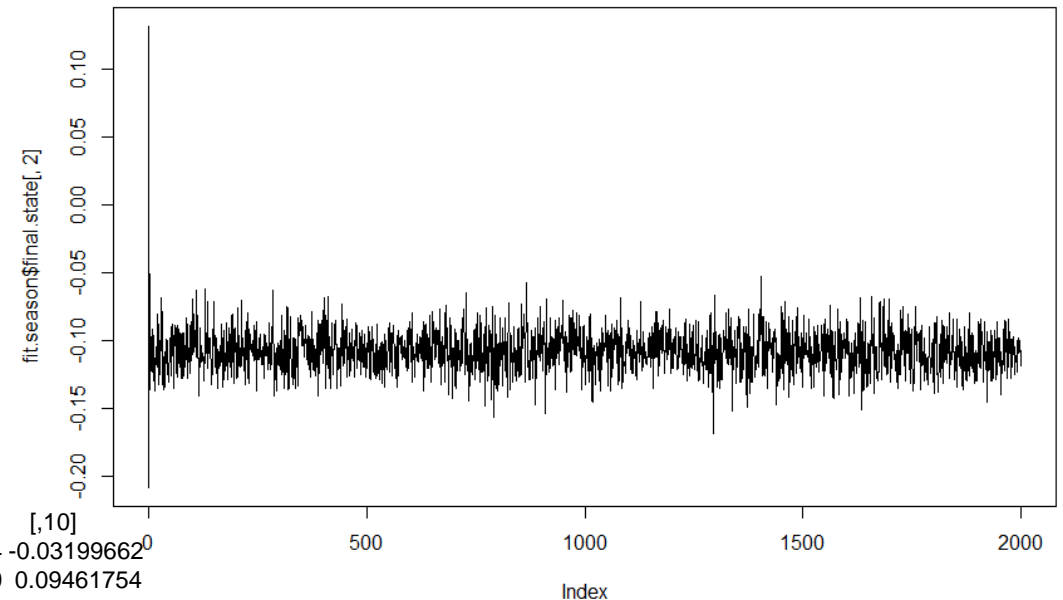
```
model_components=list()
model_components = AddLocalLevel(model_components,
                                y = air.bsts)
model_components=AddSeasonal(model_components, y
                             = air.bsts, nseasons = 12)

fit.season=bsts(air.bsts, model_components, niter = 2000)
plot(fit.season$final.state[,2],type='l')
```

```
> head(fit.season$final.state)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]
[1,]	5.611838	0.13122061	-0.2187225	-0.11985584	-0.009119432	0.18958485	0.11092930	0.20312464	-0.073750644	-0.03199662
[2,]	5.775567	-0.20784084	-0.2606871	-0.01023421	0.039516655	0.08946901	0.28148379	0.21886119	-0.072926289	0.09461754
[3,]	5.839541	-0.05099446	-0.1414026	-0.19565411	0.081198015	0.29500666	0.04275373	0.08099343	0.034689610	0.06828354
[4,]	5.847991	-0.08382377	-0.2623133	-0.20962298	0.230671767	0.20365406	0.18841063	0.12364207	-0.073111699	-0.02250975
[5,]	6.029962	-0.13583900	-0.1897727	-0.04038770	0.146914790	0.20151903	0.35220462	0.01275095	0.001114029	-0.03964137
[6,]	6.095697	-0.10922017	-0.2577005	-0.06908177	0.077977186	0.18769086	0.26883741	0.14330611	-0.023051989	0.04296503

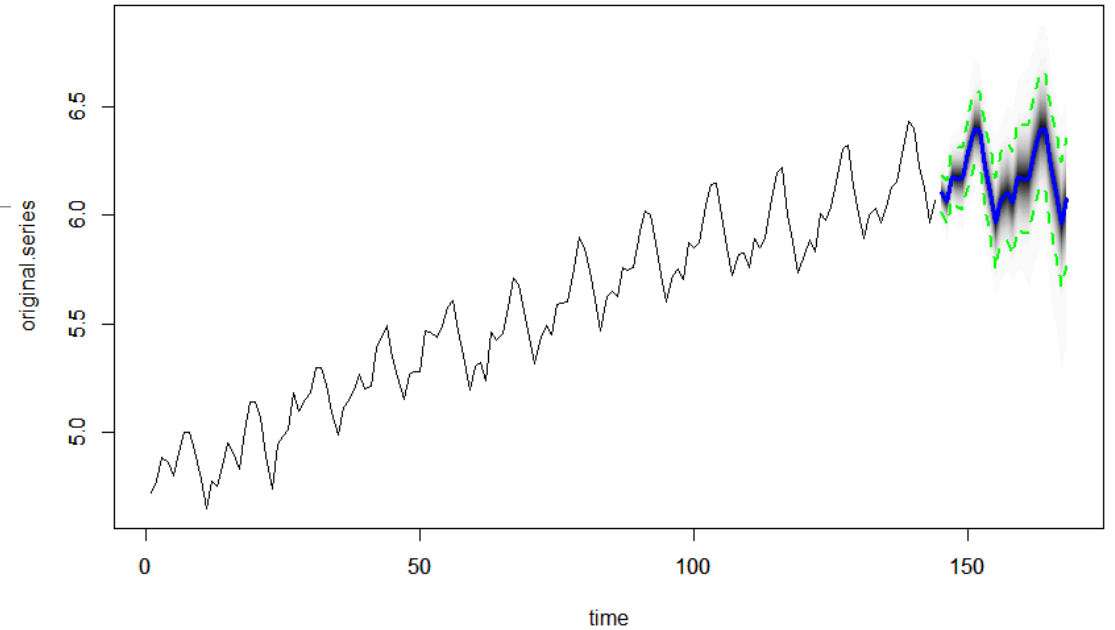
	[,11]	[,12]
[1,]	-0.1377430382	-0.20602724
[2,]	0.0093251931	-0.05982110
[3,]	0.0063418976	-0.05214867
[4,]	0.0804886616	-0.15250747
[5,]	-0.0253000053	-0.05157723
[6,]	-0.0004721743	-0.15021407



1 Level + 11 Dummy variables

Forecast(with dummy)

```
pred.season<-predict(fit.season,burn = 500,horizon = 24)  
plot(pred.season)
```



```
> pred.season$mean
```

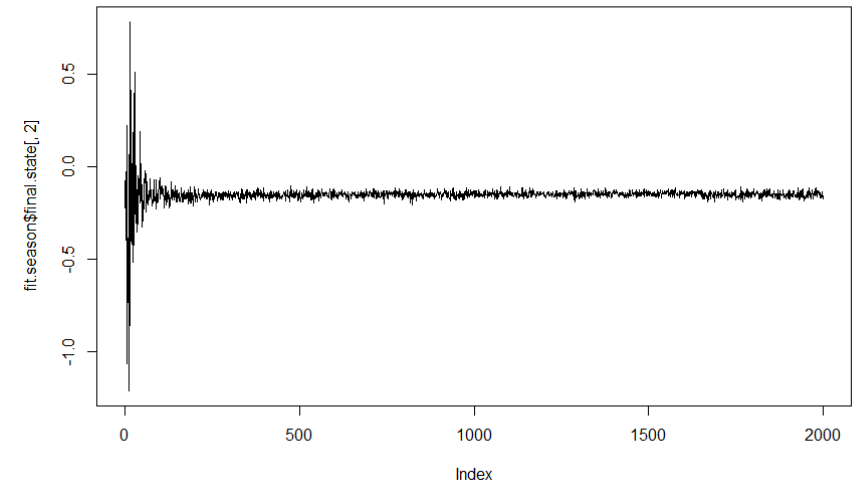
```
[1] 6.104505 6.061527 6.182944 6.172317 6.173189 6.286979 6.401651 6.394572  
6.226572 6.108766 5.964014 6.070397 6.105952 6.060050 6.179451 6.169520 6.167948  
6.283719 6.400189 6.392691 6.224080 6.106716 5.961663 6.069726
```

Seasonal (Trig with level)

```
model_components=list()
model_components = AddLocalLevel(model_components,
                                y = air.bsts)

model_components=AddTrig(model_components, y =
air.bsts,
                        period = 12,frequencies = 1:3)

fit.season=bsts(air.bsts, model_components, niter = 2000)
plot(fit.season$final.state[,2],type='l')
```

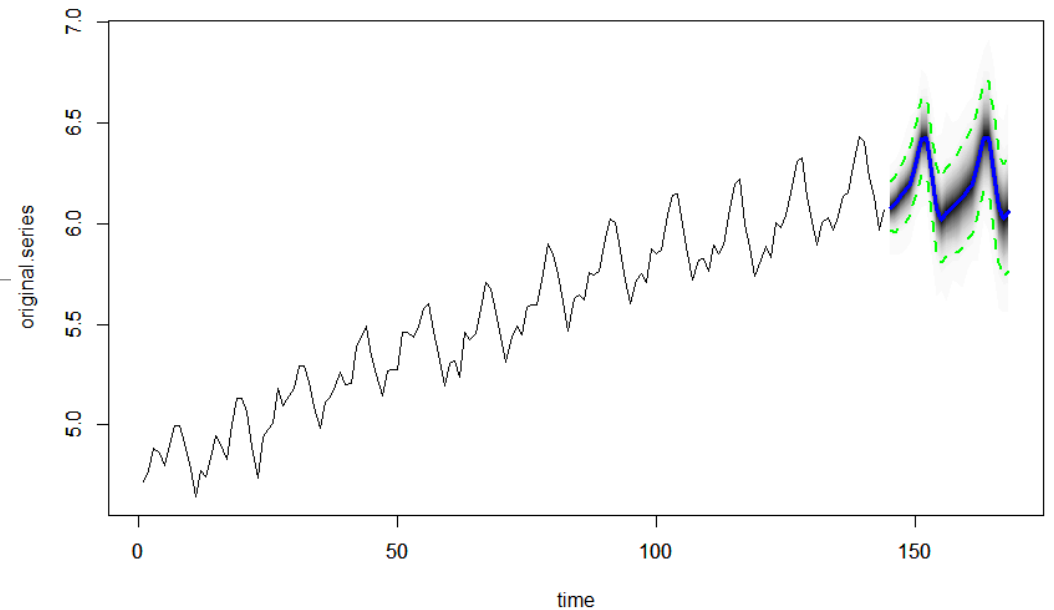


```
> head(fit.season$final.state)
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]
[1,]	5.579081	-0.14760642	-0.5717763	0.01409946	-0.82855883	0.3028314	0.1562200
[2,]	5.601328	-0.07600871	-0.4598823	0.14894884	0.20325190	1.0879606	0.1213216
[3,]	5.574211	-0.37114738	-0.2070764	-0.35487942	-1.35175170	0.5059916	-0.2237004
[4,]	5.614885	-0.39524983	-1.1895666	0.04371342	0.09413661	0.5763212	0.3782019

Forecast (Trig)

```
pred.season<-predict(fit.season,burn = 500,horizon = 24)  
plot(pred.season)
```



```
> pred.season$interval
```

	[,1]	[,2]	[,3]	[,4]	[,5]	[,6]	[,7]	[,8]	[,9]	[,10]	[,11]	[,12]
2.5%	5.965759	5.958060	5.984099	6.002435	6.041751	6.126399	6.241519	6.235748	6.056018	5.864842	5.808555	5.838206
97.5%	6.208353	6.235114	6.278911	6.324366	6.380711	6.484603	6.621303	6.623532	6.464349	6.287322	6.231317	6.276204
	[,13]	[,14]	[,15]	[,16]	[,17]	[,18]	[,19]	[,20]	[,21]	[,22]	[,23]	[,24]
2.5%	5.854577	5.859481	5.883375	5.926380	5.952490	6.034250	6.161035	6.149160	5.988632	5.806161	5.740073	5.762272
97.5%	6.306913	6.345058	6.393916	6.430722	6.483931	6.569421	6.701481	6.709872	6.553636	6.353762	6.293718	6.340721

Can also fit other X variables...

$$y_t = \underbrace{\mu_t}_{\text{Level}} + \underbrace{\tau_t}_{\text{Season}} + \underbrace{\beta^T x_t}_{\text{"X" Variables}} + \epsilon_t$$

$$\left. \begin{aligned} \mu_{t+1} &= \mu_t + \delta_t + u_t \\ \delta_{t+1} &= \delta_t + v_t \end{aligned} \right\} \text{Level and Trend } (u_t \text{ and } v_t \text{ are error})$$

$$\tau_{t+1} = -\sum \tau_t + w_t \quad \text{Seasonality } (w_t \text{ is the error term})$$

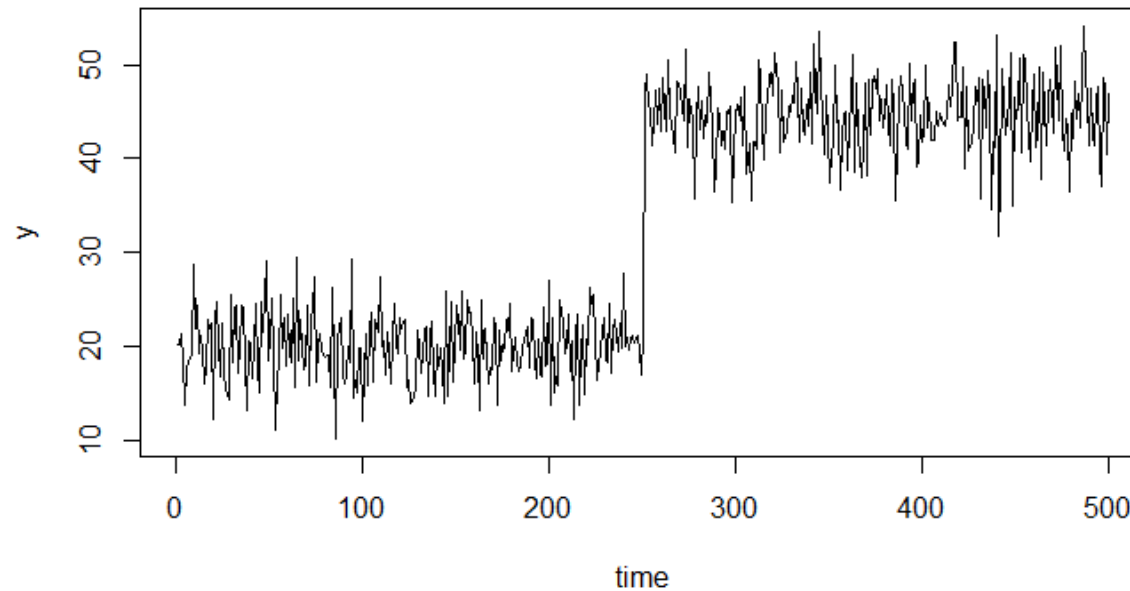
Change-point

A change-point is a point in your data in which the “signal” changes.

CHANGE-POINT

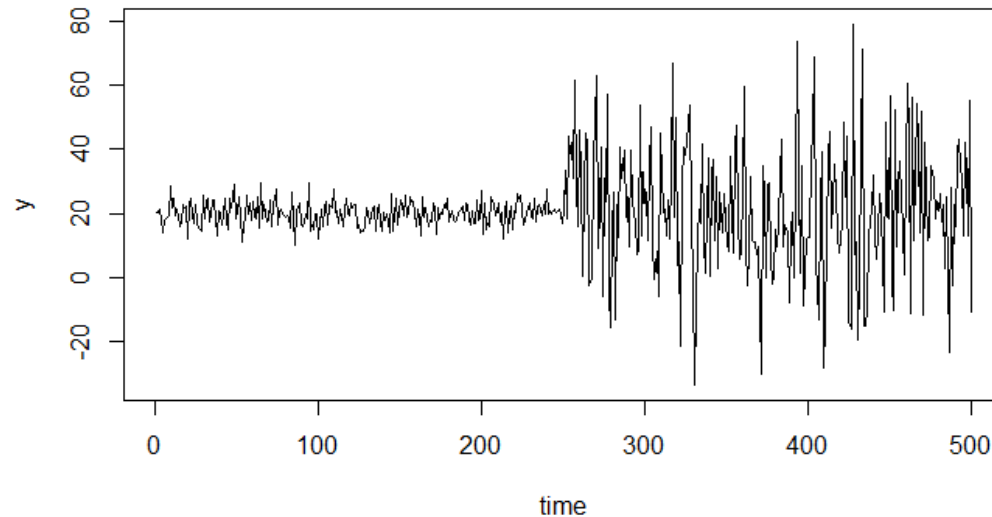
Change-point

A change-point is a point in your data in which the “signal” changes. It can be a shift in means:



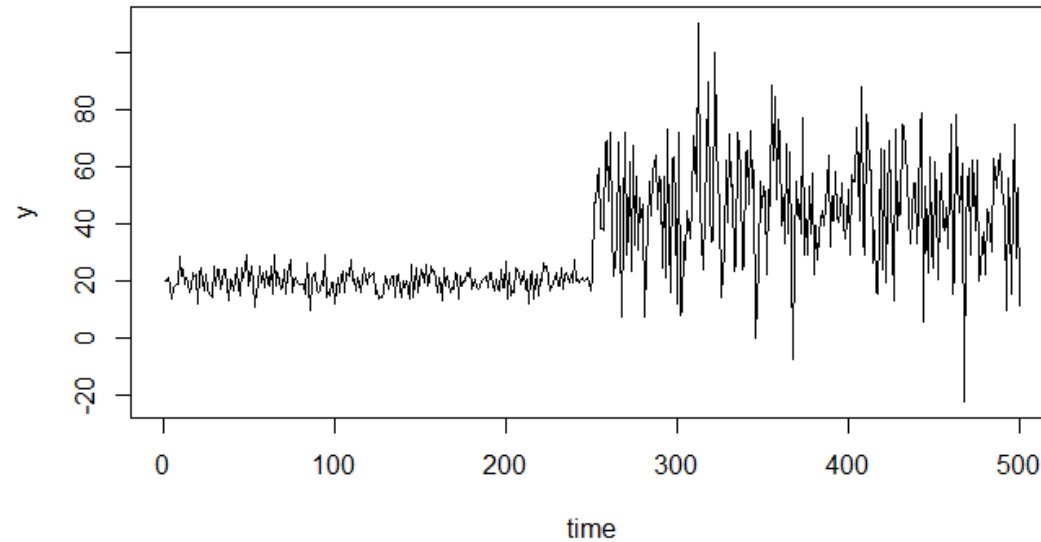
Change-point

A change-point is a point in your data in which the “signal” changes. It can be a change in variance:



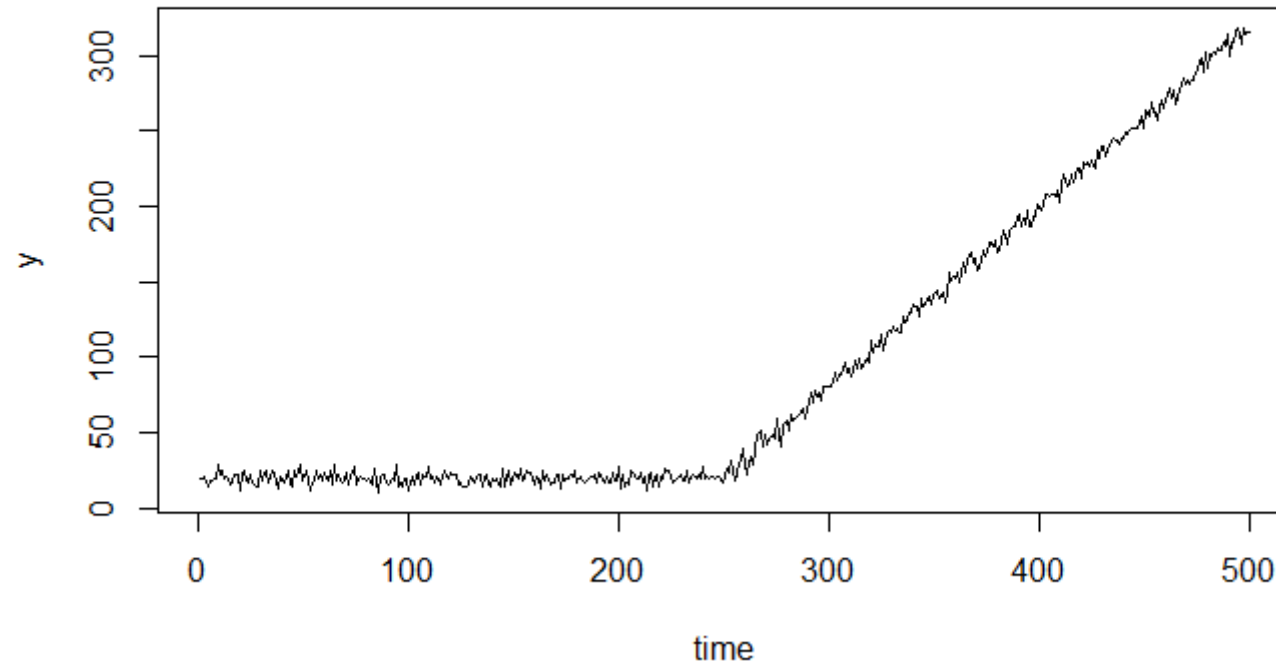
Change-point

A change-point is a point in your data in which the “signal” changes. It can be a change in variance AND a shift in means:



Change-point

A change-point is a point in your data in which the “signal” changes. It can be a change in pattern (going from



Change-point

And sooooo many more!!!

Can be challenging to find change-points (they can happen in many different ways AND there can be more than one!)

An overview of change point packages in R

Source: `vignettes/packages.Rmd`

OBS: I have yet to review these packages: `not`, `breakfast`, `IDetect`, `trendsegmentR`, `mosum`, `ChangepointTesting`, `changepoint.mv`, `changepointsHD`, `changepointsVar`, `InspectChangepoint`, `breakpoint`, `segmentr`, `Segmentor3IsBack`, `trendsegmentR`, `BayesPiecewiseICAR`, `BayesPieceHazSelect`.

There are a lot of change point packages out there already, so why `mcp`? Here are my (probably biased) thoughts about this. I compiled some tables, summarising change point packages ([.xlsx file here](#)). I will demonstrate each of these packages in an applied example below to discuss their merits and shortcomings. I recommend [this nice overview](#) of the methodologies used in many of these packages.

Of the packages reviewed here, I think `segmented`, and `EnvCpt` are good if the data fits what they can model, and `mcp` is a more capable and general-purpose package at the cost of speed. You can see the immediate roadmap for `mcp` at the [GitHub issues tracker](#).

From the author's of `mcp`

Modeling options

How much flexibility does the package allow for modeling the system, you're studying? A clear difference here is between packages that allow you to *specify* the number of change points, and packages which infer the number of change points *automatically* (using some criteria), captured in the `N` column below.

Package		Segment models					Change points					Other
Name	version	Formulas	GLM	Time series	Survival	Multiv.	N	Mean	Random	Variance	AR	Share pars
<code>mcp</code>	0.1.0	specify: all	yes	AR(N)	planned	no	specify	yes	yes	yes	yes	yes
<code>segmented</code>	1.0.0	specify: slope	yes	ARIMA(N)	yes	no	both	yes	no	no	no	no
<code>strucchange::breakpoints</code>	1.5.2	specify: all	no	no	no	no	specify	yes	no	no	no	no
<code>ecp</code>	3.1.2	auto: ?	no	?	no	yes	auto	yes	no	no	no	no
<code>bcp</code>	4.0.3	auto: slope + int	no	no	no	yes	auto	yes	no	no	no	no
<code>changepoint</code>	2.2.2	auto: int	no	no	no	no	auto	yes	no	yes	no	no
<code>changepoint.np</code>	1.0.1	auto: int?	no	no	no	no	auto	yes	no	?	no	no
<code>strucchange::Fstats</code>	1.5.2	specify: all	no	no	no	no	only 1	yes	no	no	no	no
<code>TSMCP</code>	1.0.0	auto: ?	no	yes	no	yes	auto	yes	no	no	no	no
<code>cpm</code>	2.2.0	auto: int	(yes)	no	no	no	auto	yes	no	yes	no	no
<code>EnvCpt</code>	1.1.1	specify: (select)	no	AR(2)	no	no	auto	yes	no	yes	no	no
<code>wbsts</code>	2.0.0	auto: ?	no	yes	no	no	auto	yes	no	no	no	no

Methods

Focus on the change-point and mcp packages

Changepoint package:

- Can detect multiple change-points (do NOT need to specify number of change-points)
- Uses the PELT algorithm (“Pruned Exact Linear Time” by Killick, Fearnhead, and Eckley 2012)
- Combines the binary segmentation algorithm with the segment neighborhood algorithm for an efficient, exact test (minimizes a cost function with a penalty)
- Can handle multiple change-points in mean and/or variances

MCP (Regression with Multiple Change-point...Bayesian)

- Can handle multiple change-points
- Can handle a shift in means and/or variances and regression (need to specify form of each segment)
- Need to identify number of change-points apriori
- Bayesian methodology...need to download JAGS

Data sets

We will be using two different data sets:

Netherlands unemployment data set (use percent unemployed...yearly data): many thanks to Dr. LaBarr for this data set

Neenana Ice Classic data (yearly data....date at which ice breaks): many thanks to Dr. Dickey for this data set

Change-point algorithm

```
library(changepoint)
cp.n <- cpt.mean(netherlands$Percentage.Unemployed, method = "PELT",
Q = 10)
plot(cp.n)
summary(cp.n)
```

Edited R output

Created Using changepoint version 2.2.4

Changepoint type : Change in mean

Method of analysis : PELT

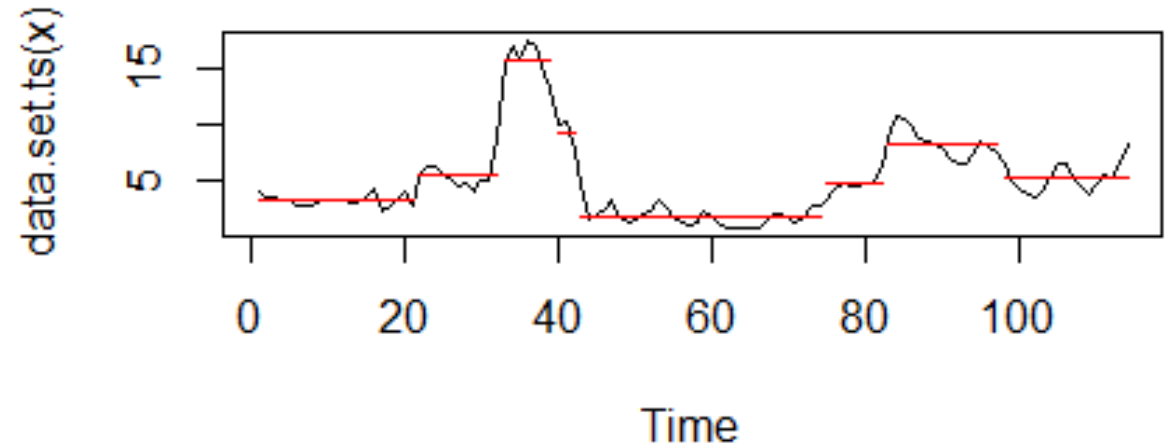
Test Statistic : Normal

Type of penalty : MBIC with value, 14.2086

Minimum Segment Length : 1

Maximum no. of cpts : Inf

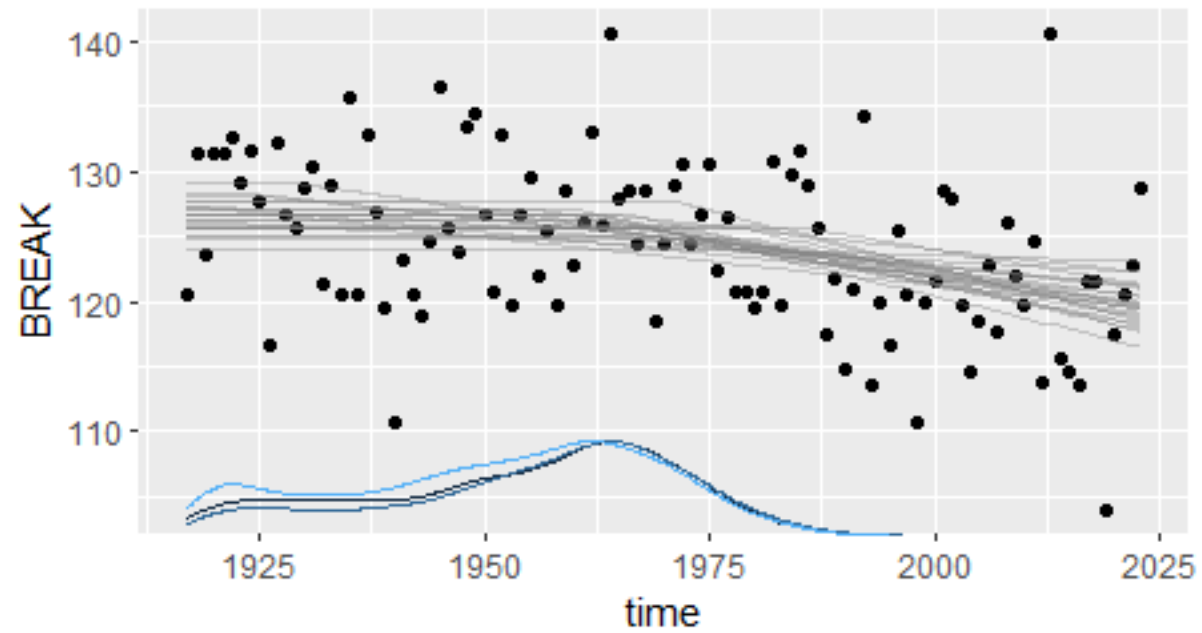
Changepoint Locations : 21 32 39 42 74 82 97

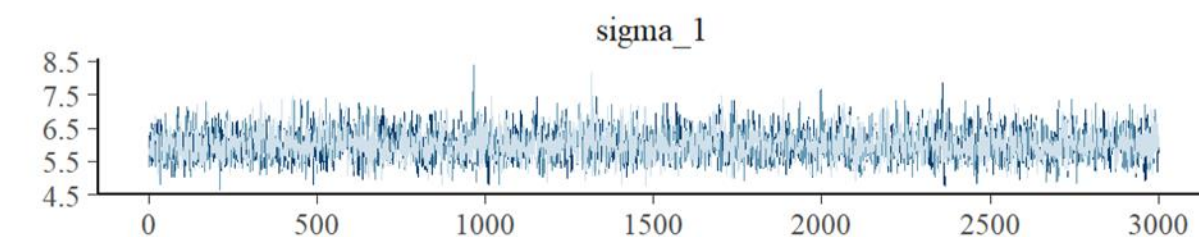
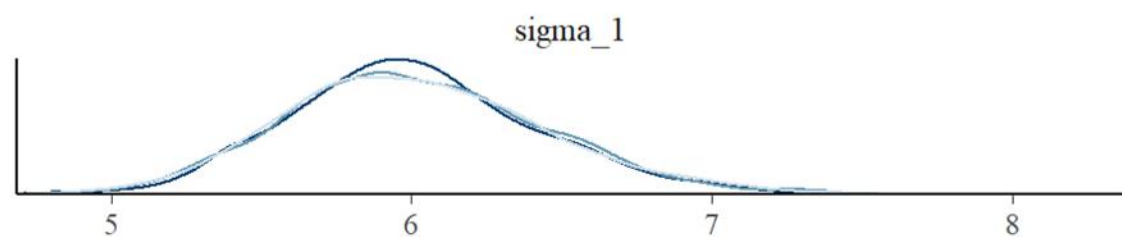
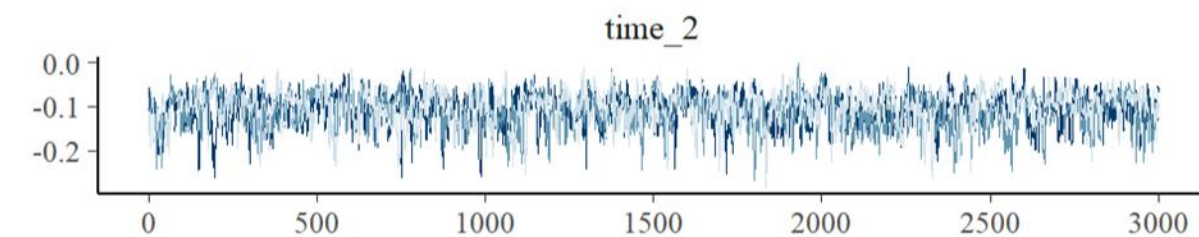
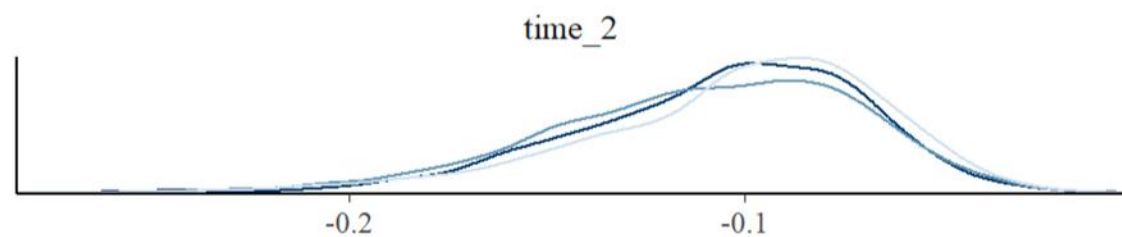
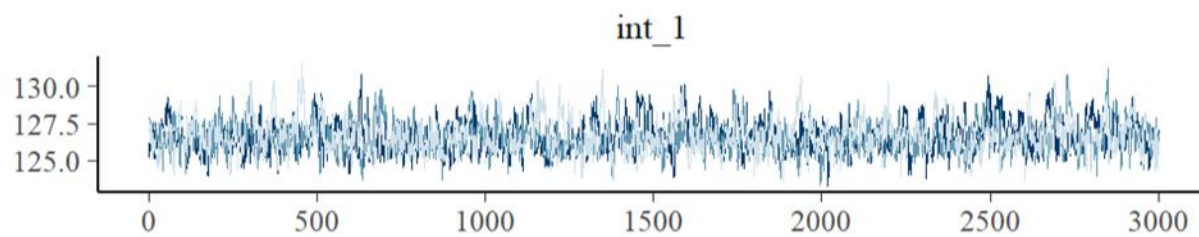
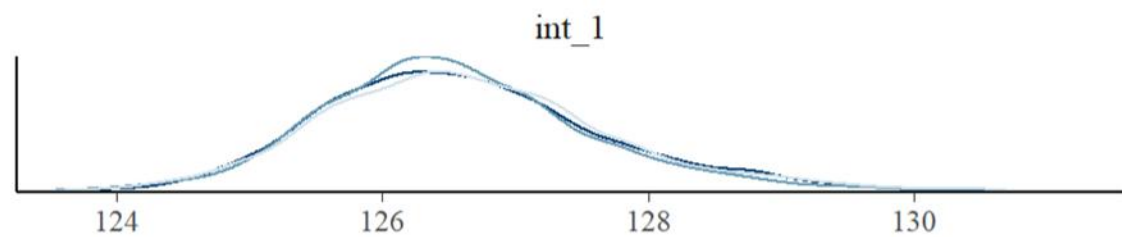
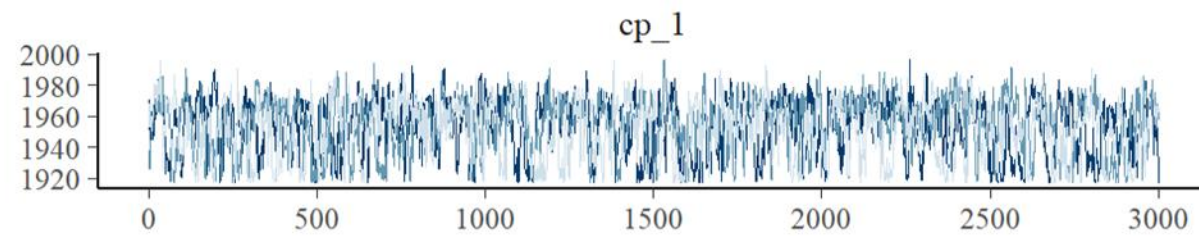
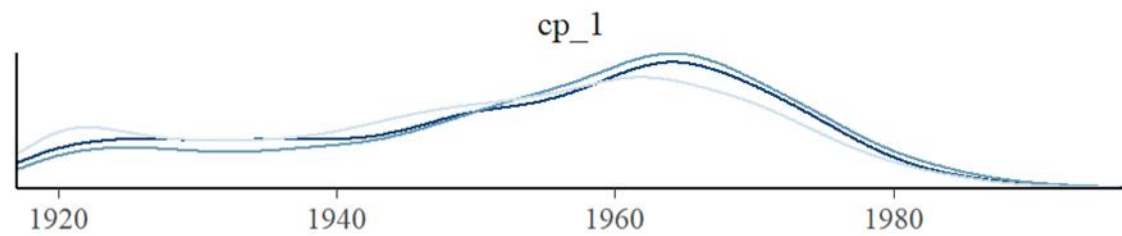


Regression Change-point with MCP

```
library(mcp)
library(rjags)
library(jagsUI)
ind<-seq(1,nrow(Icecap_data2))
ind<-seq(1917,2023)
Icecap_data2$time<-ind
model1<-list(BREAK~1, ~0+time)
mcp.ice=mcp(model1,data=Icecap_data2,sample="post",par_x = "time")
plot(mcp.ice)
plot_pars(mcp.ice)
```

Output





Output

```
> summary(mcp.ice)
```

```
Family: gaussian(link = 'identity')
```

```
Iterations: 9000 from 3 chains.
```

```
Segments:
```

```
1: BREAK ~ 1
```

```
2: BREAK ~ 1 ~ 0 + time
```

```
Population-level parameters:
```

name	mean	lower	upper	Rhat	n.eff
cp_1	1953.51	1917.46	1977.88	1	556
int_1	126.60	124.69	128.93	1	940
sigma_1	6.01	5.20	6.85	1	4605
time_2	-0.11	-0.19	-0.04	1	943

Questions?

