US_regression.R

wanchuangzhu

2020-06-23

```r
##################################################prepare the data
us=read.csv('../raw-data/us_result.csv')
# gt=read.csv('../raw-data/gt.csv')
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(tidyr)
models= c("Geneva","YYG") # YYG and IHME predict USA and also states, Geneva only predicts USA
us=filter(us,model_name %in% models,location_long =="United States")

ahead=7 # decide how many days are included in the regression


us_part1=filter(us,model_name %in% models[1])

us_part1=group_by(us_part1,target_end_date,forecast_date,model_name,gt_source)
us_part1=summarise(us_part1,expected_value=sum(expected_value),gt=sum(gt),lookahead=mean(lookahead))

us_part2=filter(us,model_name %in% models[2]) # YYG predict USA and also states
us_part2=group_by(us_part2,target_end_date,forecast_date,model_name,gt_source)
us_part2=summarise(us_part2,expected_value=max(expected_value),gt=max(gt),lookahead=mean(lookahead))

us_group=bind_rows(us_part1,us_part2)

us_wide=pivot_wider(us_group,names_from = c("model_name"), values_from = expected_value) %>% filter(.,a

for(i in 1:nrow(us_wide)){
  temp=filter(us_wide,forecast_date==us_wide$forecast_date[i],target_end_date==us_wide$target_end_date[
  us_wide[i,models[1]]=mean(as.matrix(temp[,models[1]]),na.rm = T)
  us_wide[i,models[2]]=mean(as.matrix(temp[,models[2]]),na.rm = T)

}

us_wide=ungroup(filter(us_wide,gt_source=="JHU")) %>% dplyr::select(.,-forecast_date,-gt_source)
```

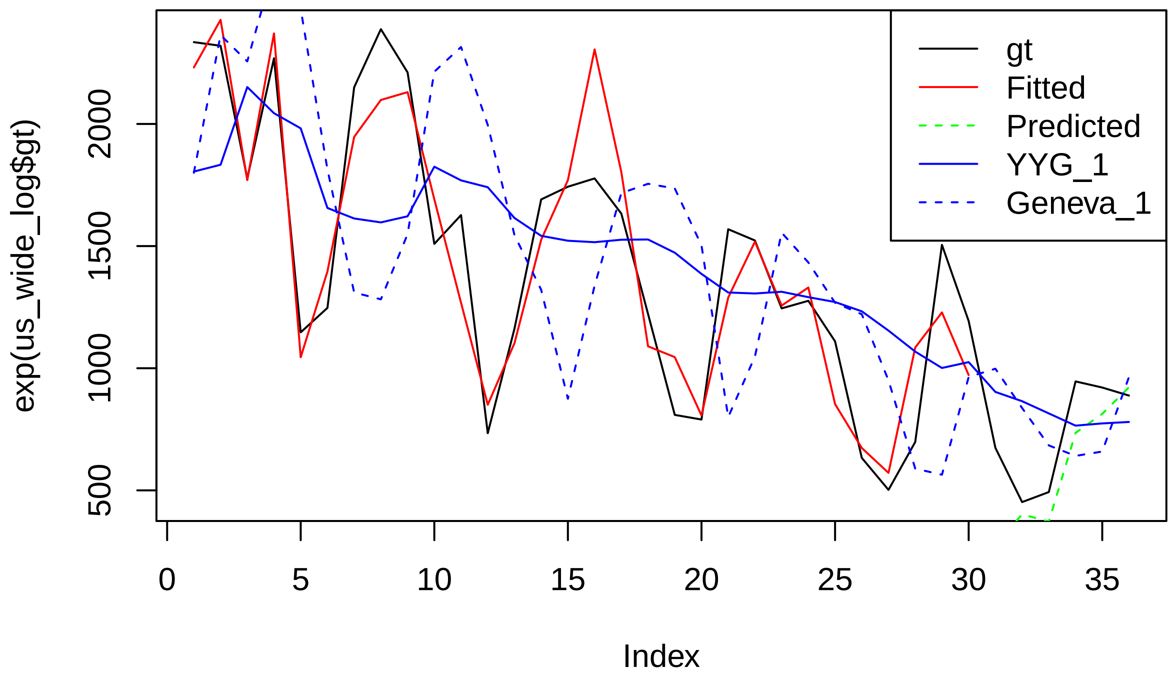
```r
us_wide=pivot_wider(us_wide,names_from = lookahead,values_from = c("Geneva","YYG"))
us_wide=drop_na(us_wide) %>% mutate(.,week=as.numeric(weekdays(as.Date(target_end_date)) %in% c("Saturda
us_wide$target_end_date=as.Date(us_wide$target_end_date)
us_wide=us_wide[order(us_wide$target_end_date),]
## take the data into log-scale
us_wide_log=(us_wide)
us_wide_log[,c(2:16)]=log(us_wide_log[,c(2:16)])
train.num=30 # split training and prediction set

## regression
model=lm(gt~. ,data=us_wide_log[1:train.num,-1])
fitted=model$fitted.values
pred=c()
for(train in train.num:(nrow(us_wide_log)-1)){
  model=lm(gt~. ,data=us_wide_log[1:train,-1])
  pred=c(pred,predict.lm(model,newdata = us_wide_log[(train+1),-1]))}


plot(exp(us_wide_log$gt),type = 'l')
lines(c(1:train.num),exp(fitted),col="red")
lines(c((train.num+1):nrow(us_wide_log)),exp(pred),col="green",lty=2)
lines(us_wide$YYG_1,col='blue')
lines(us_wide$Geneva_1,col='blue',lty=2)
legend("topright",c("gt",'Fitted','Predicted',"YYG_1","Geneva_1"),lty = c(1,1,2,1,2),col=c('black','red

## Compare the combined model with single model prediction
sum(abs(us_wide$gt[(train.num+1):(nrow(us_wide_log))]-exp(pred))) # combined
```

```
## [1] 920.1755
```

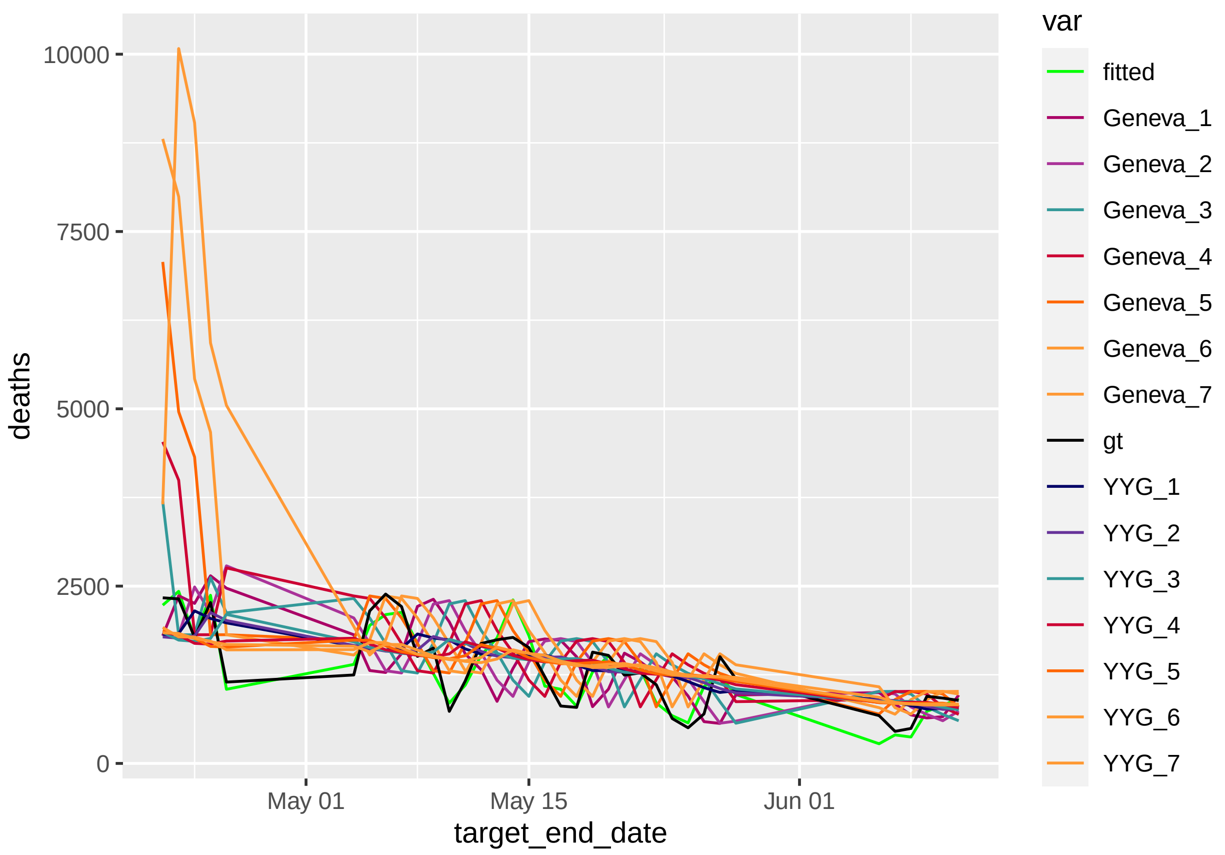
```r
sum(abs((us_wide$gt[(train.num+1):(nrow(us_wide_log))]-us_wide$YYG_1[(train.num+1):(nrow(us_wide_log))]
```

```
## [1] 1400
```

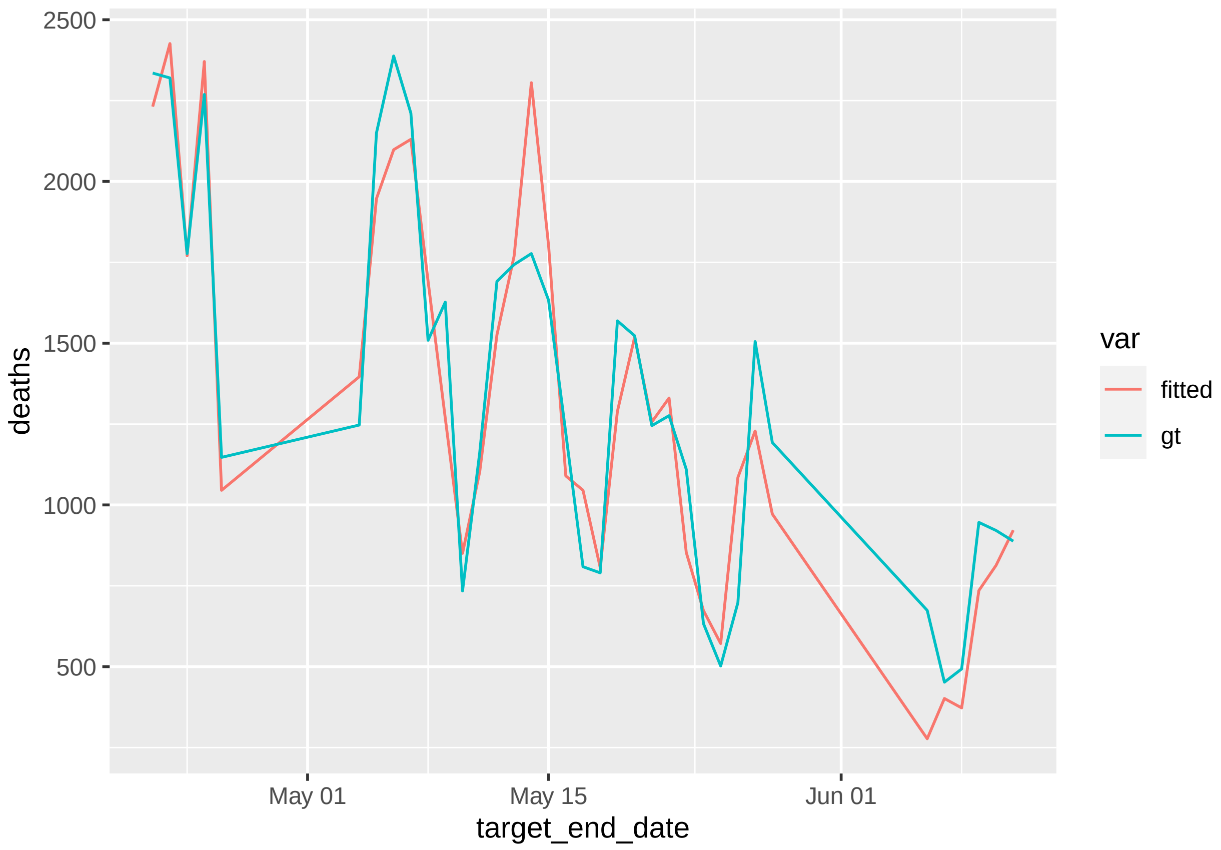
```r
## Results
library(ggplot2)
```



```
us_wide_all=us_wide
us_wide_all$fitted = exp(c(fitted,pred))

us_long=pivot_longer(us_wide_all,cols=-c(target_end_date,week),names_to ="var",values_to = "deaths" )
us_long$target_end_date=as.Date(us_long$target_end_date)

ggplot(us_long, aes(x=target_end_date,y=deaths,colour=var)) + geom_line() + scale_colour_manual(values=
```



```
ggplot(filter(us_long,var %in% c("gt","fitted")), aes(x=target_end_date,y=deaths,colour=var)) + geom_li
```



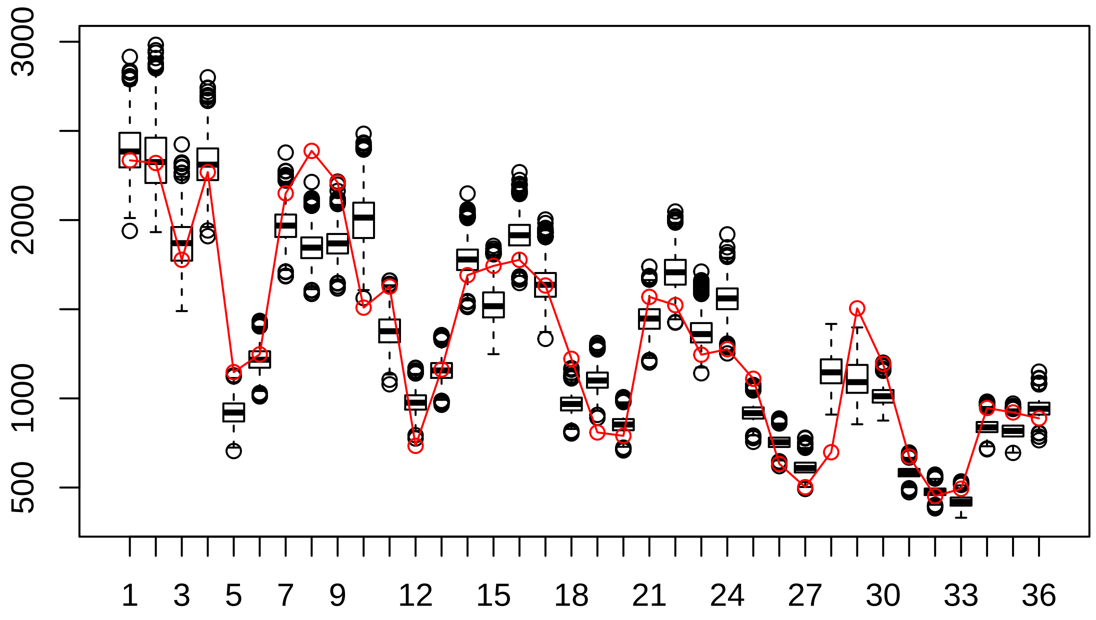
```r
### Bayesian inference without constraints with uniform priors
priors.dist=matrix(c(c(-10^5,10^5),rep(c(-100,100),14),c(-10^5,10^5),c(0,500)),ncol=2,byrow=T)
row.names(priors.dist)=c("inter",rep("coef",14),"week","sig")
log.post.likelihood=function(data,paras){
  #data is the data
  #paras is the parameters. [intercept,coefficients,week,sigmas]
  X=cbind(1,data[,-1])
  l=sum(dnorm(x=as.matrix(X)%*%paras[-length(paras)],mean = as.matrix(data[,1]),sd=paras[length(paras)]
  return(l)
}
# starting points
train.num=36
model=lm(gt~. ,data=us_wide_log[1:train.num,-1])
paras=c(model$coefficients,sqrt(var(model$residuals)/length(model$residuals))) # use MLE of unconstrain

hyper=rep(0.5,length(paras))
counts=rep(0,length(paras))
acc.counts=rep(0,length(paras))
# tune the proposal hyperparameters
for(iter in 1:3000){
  for(i in 1:length(paras)){
    paras.new=paras
    paras.new[i]= rnorm(1,paras[i],hyper[i])
    if(paras.new[i] > priors.dist[i,1] && paras.new[i] < priors.dist[i,2]){
      counts[i]=counts[i]+1
      if(log.post.likelihood(data = us_wide_log[1:train.num,-1],paras =paras.new ) - log.post.likelihoo
        acc.counts[i]=acc.counts[i]+1
        paras=paras.new
      }
    }
  }

  if(iter %%200 ==0){ # update the hyperparameter
    acc=acc.counts/counts
    for(j in 1:length(acc)){
      if(acc[j]<0.44){
        hyper[j]=hyper[j]/(2^(200/iter))
      } else{hyper[j]=hyper[j]*(2^(200/iter))}
    }
    counts=rep(0,length(paras))
    acc.counts=rep(0,length(paras))
    cat("iteration=",iter,'\n')
  }
}
```

```
## iteration= 200
## iteration= 400
## iteration= 600
## iteration= 800
## iteration= 1000
## iteration= 1200
## iteration= 1400
## iteration= 1600
## iteration= 1800
## iteration= 2000
## iteration= 2200
## iteration= 2400
## iteration= 2600
## iteration= 2800
## iteration= 3000
```

```
# MCMC
iter.num=2000
result.mat=matrix(NA,iter.num,length(paras))
y.hat=matrix(NA,iter.num,nrow(us_wide))
for(iter in 1:iter.num){
  for(i in 1:length(paras)){
    paras.new=paras
    paras.new[i]= rnorm(1,paras[i],hyper[i])
    if(paras.new[i] > priors.dist[i,1] && paras.new[i] < priors.dist[i,2]){
      counts[i]=counts[i]+1
      if(log.post.likelihood(data=us_wide_log[1:train.num,-1],paras =paras.new ) - log.post.likelihood(
        acc.counts[i]=acc.counts[i]+1
        paras=paras.new
      }
    }
  }
  result.mat[iter,]=paras
  temp=cbind(1,us_wide_log[1:train.num,-c(1,2)])
  y.hat[iter,]= as.matrix(temp)%*%paras[-length(paras)]
  if(iter%%500 ==0){cat("iteration=",iter)}
}
```

```
## iteration= 500iteration= 1000iteration= 1500iteration= 2000
```

```
boxplot(exp(y.hat))
points(us_wide$gt,col='red')
lines(us_wide$gt,col='red')
```



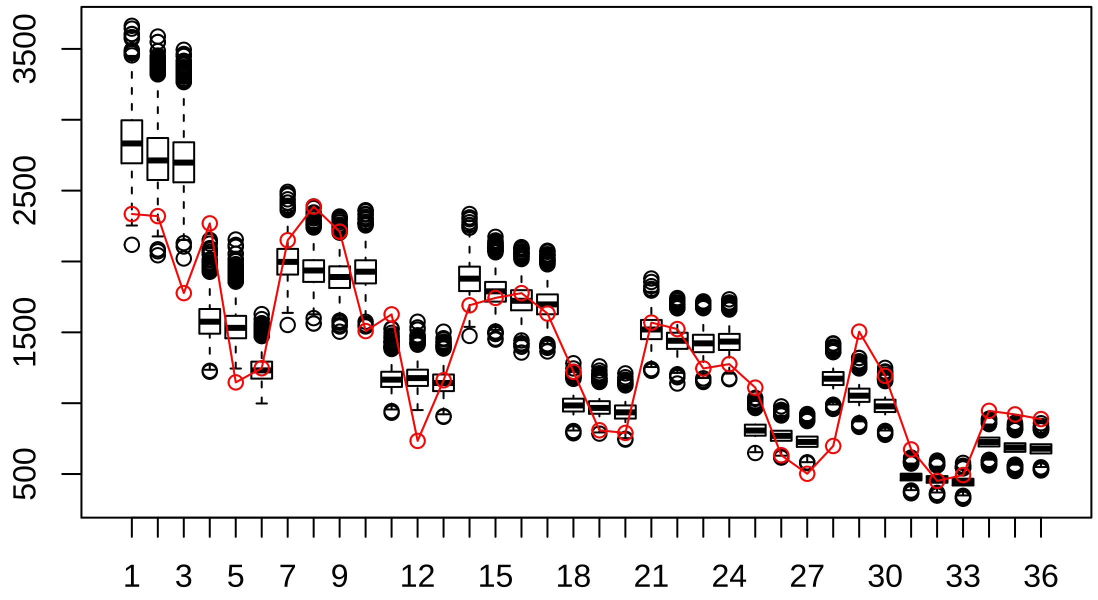
```
### Bayesian inference with constraints with uniform priors
priors.dist=matrix(c(c(-10^5,10^5),rep(c(0,100),14),c(-10^5,10^5),c(0,500)),ncol=2,byrow=T)
row.names(priors.dist)=c("inter",rep("coef",14),"week","sig")

# starting points
train.num=36
model=lm(gt~. ,data=us_wide_log[1:train.num,-1])
paras=c(model$coefficients,sqrt(var(model$residuals)/length(model$residuals))) # use MLE of unconstrain
paras[ 2:(length(paras)-2)] =abs(paras[ 2:(length(paras)-2)])

hyper=rep(0.5,length(paras))
counts=rep(0,length(paras))
acc.counts=rep(0,length(paras))
# tune the proposal hyperparameters
for(iter in 1:4000){
  for(i in 1:length(paras)){
    paras.new=paras
    paras.new[i]= rnorm(1,paras[i],hyper[i])
    if(paras.new[i] > priors.dist[i,1] && paras.new[i] < priors.dist[i,2]){
      counts[i]=counts[i]+1
      if(log.post.likelihood(data = us_wide_log[1:train.num,-1],paras =paras.new ) - log.post.likelihoo
        acc.counts[i]=acc.counts[i]+1
        paras=paras.new
      }
    }
  }
  if(iter %%200 ==0){ # update the hyperparameter
    acc=acc.counts/counts
    for(j in 1:length(acc)){
      if(acc[j]<0.44){
        hyper[j]=hyper[j]/(2^(200/iter))
      } else{hyper[j]=hyper[j]*(2^(200/iter))}
    }
    counts=rep(0,length(paras))
    acc.counts=rep(0,length(paras))
    cat("iteration=",iter,'\n')
  }
}
```

```
## iteration= 200
## iteration= 400
## iteration= 600
## iteration= 800
## iteration= 1000
## iteration= 1200
## iteration= 1400
## iteration= 1600
## iteration= 1800
## iteration= 2000
## iteration= 2200
## iteration= 2400
## iteration= 2600
## iteration= 2800
## iteration= 3000
## iteration= 3200
## iteration= 3400
## iteration= 3600
## iteration= 3800
## iteration= 4000
```

```
# MCMC
iter.num=2000
result.mat=matrix(NA,iter.num,length(paras))
y.hat=matrix(NA,iter.num,nrow(us_wide))
for(iter in 1:iter.num){
  for(i in 1:length(paras)){
    paras.new=paras
    paras.new[i]= rnorm(1,paras[i],hyper[i])
    if(paras.new[i] > priors.dist[i,1] && paras.new[i] < priors.dist[i,2]){
      counts[i]=counts[i]+1
      if(log.post.likelihood(data=us_wide_log[1:train.num,-1],paras =paras.new ) - log.post.likelihood(
        acc.counts[i]=acc.counts[i]+1
        paras=paras.new
      }
    }
  }
  result.mat[iter,]=paras
  temp=cbind(1,us_wide_log[1:train.num,-c(1,2)])
  y.hat[iter,]= as.matrix(temp)%*%paras[-length(paras)]
  if(iter%%500 ==0){cat("iteration=",iter)}
}
```

```
## iteration= 500iteration= 1000iteration= 1500iteration= 2000
```

```
boxplot(exp(y.hat))
points(us_wide$gt,col='red')
lines(us_wide$gt,col='red')
```

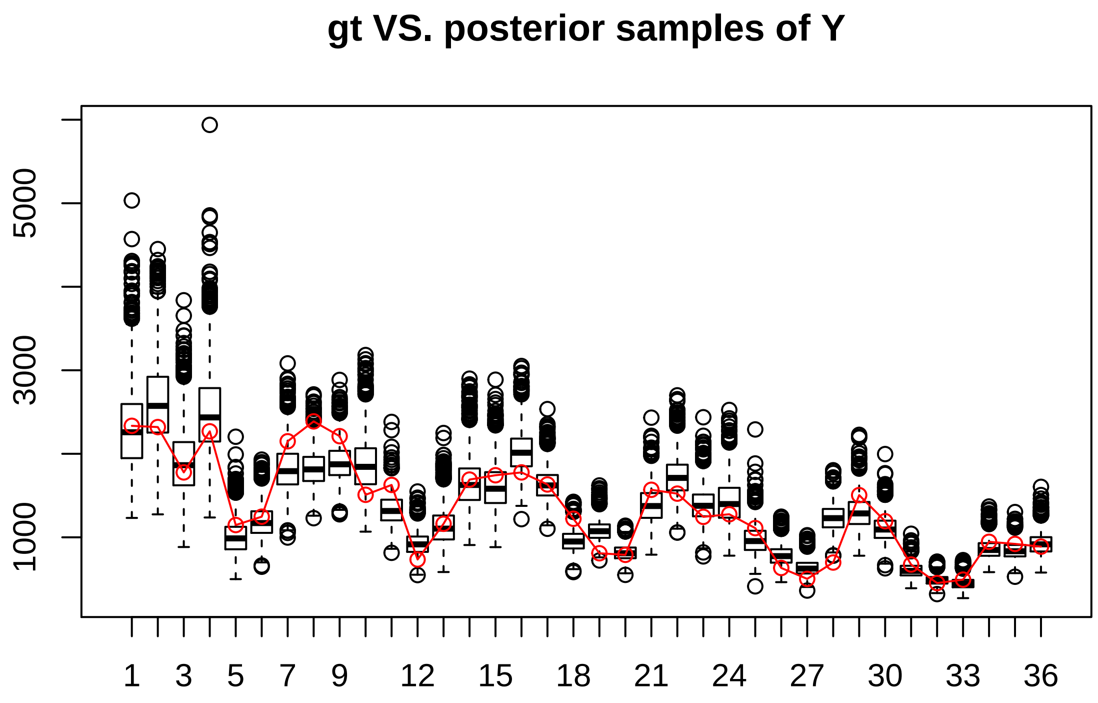


```
### Bayesian inference without constraints with Conjugate priors
# sigma^2 is Inverse-Gamma distributed, beta is conditionally distributed as normal distribution
library(invgamma)
library(MASS)
```

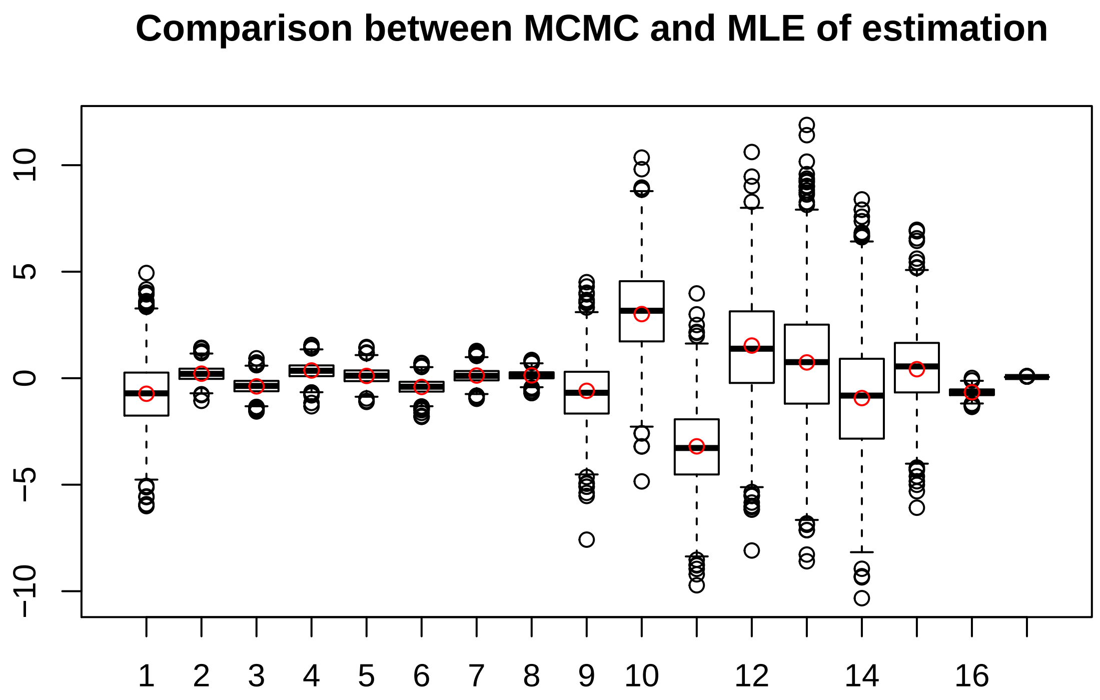
```
##
## Attaching package: 'MASS'
```

```
## The following object is masked from 'package:dplyr':
##
##     select
```

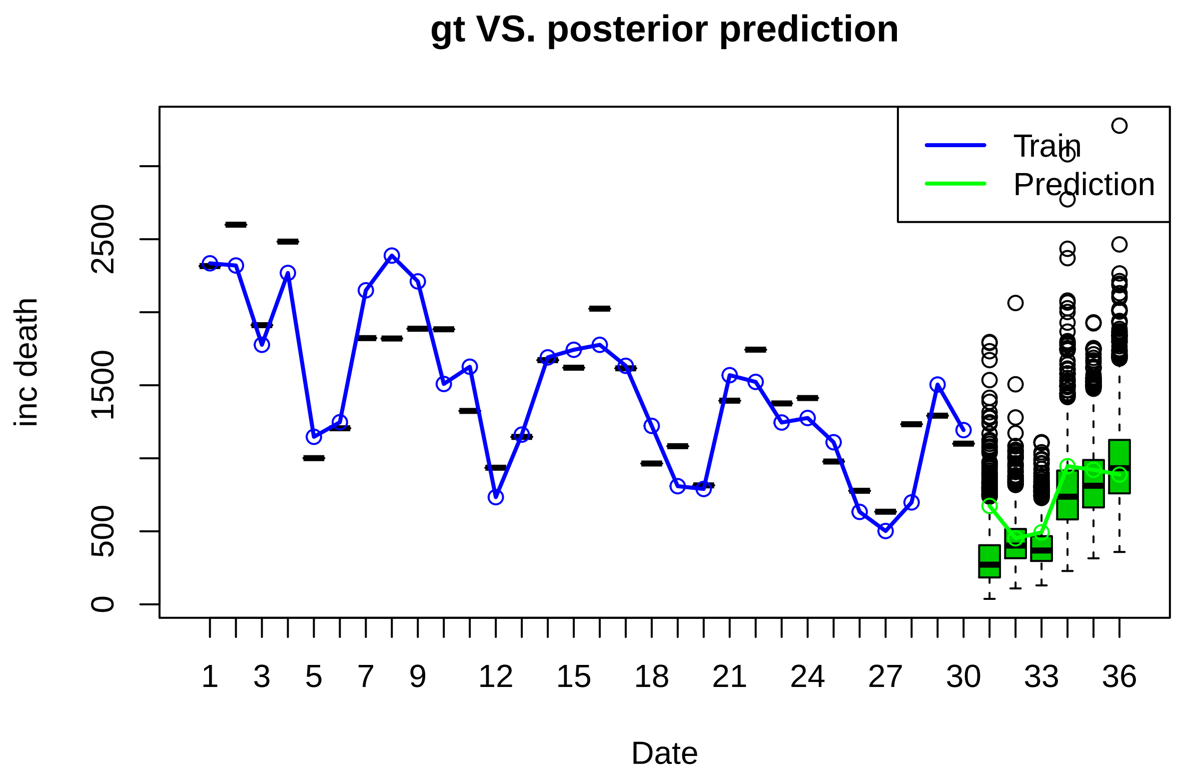
```
train.num=36
num.para=17
X=cbind(1,as.matrix(us_wide_log[1:train.num,-c(1,2)]))
y=as.matrix(us_wide_log[1:train.num,2])

priors.dist=list(beta=list(mu=rep(0,num.para-1),cv=diag(10^5,num.para-1,num.para-1)),sig=c(a0=0.0001,b0=

#hyperparameters
Delta_0 = solve( priors.dist$beta$cv)
Delta_n = Delta_0 + t(X)%*%X
iDelta_n = solve(Delta_n)
mu_0 = priors.dist$beta$mu
mu_n = solve(Delta_n)%*% (Delta_0 %*% mu_0 + t(X)%*% y)
a_0=priors.dist$sig["a0"]
a_n = a_0  + nrow(us_wide_log[1:train.num,]) # posterior parameter of sigma^2
b_0 = priors.dist$sig["b0"]
b_n = b_0 + 0.5* (t(y)%*%y + t(as.matrix(mu_0))%*% Delta_0 %*% as.matrix(mu_0) - t(mu_n) %*% Delta_n%*%
# starting points
model=lm(gt~. ,data=us_wide_log[1:train.num,-1])
paras=c(model$coefficients,(var(model$residuals)/length(model$residuals))) # use MLE of unconstrained l


# MCMC-Gibbs sampler
iter.num=2000
result.mat=matrix(NA,iter.num,length(paras))
y.hat=matrix(NA,iter.num,nrow(us_wide))
for(iter in 1:iter.num){
  # draw sigma^2
  result.mat[iter,length(paras)] = rinvgamma(n=1,shape = a_n, scale = b_n)
  result.mat[iter,-length(paras)] = mvrnorm(n=1,mu=mu_n,Sigma = result.mat[iter,length(paras)]* iDelta_

  temp=cbind(1,us_wide_log[,-c(1,2)])
  y.hat[iter,]= X%*%result.mat[iter,-length(paras)]
}
boxplot(exp(y.hat))
points(us_wide$gt,col='red')
lines(us_wide$gt,col='red')
title(main="gt VS. posterior samples of Y")
```



```
boxplot(result.mat)
points(model$coefficients,col='red')
title(main= " Comparison between MCMC and MLE of estimation")
```



```
train.num=30
pred.Y=c()
num.para=17

priors.dist=list(beta=list(mu=rep(0,num.para-1),cv=diag(10^5,num.para-1,num.para-1)),sig=c(a0=0.0001,b0=

for(train.num in 30:nrow(us_wide_log)){

  X=cbind(1,as.matrix(us_wide_log[1:train.num,-c(1,2)]))
  y=as.matrix(us_wide_log[1:train.num,2])
  #hyperparameters
  Delta_0 = solve( priors.dist$beta$cv)
  Delta_n = Delta_0 + t(X)%*%X
  iDelta_n = solve(Delta_n)
  mu_0 = priors.dist$beta$mu
  mu_n = solve(Delta_n)%*% (Delta_0 %*% mu_0 + t(X)%*% y)
  a_0=priors.dist$sig["a0"]
  a_n = a_0  + nrow(us_wide_log[1:train.num,]) # posterior parameter of sigma^2
  b_0 = priors.dist$sig["b0"]
  b_n = b_0 + 0.5* (t(y)%*%y + t(as.matrix(mu_0))%*% Delta_0 %*% as.matrix(mu_0) - t(mu_n) %*% Delta_n%
  # starting points
  model=lm(gt~. ,data=us_wide_log[1:train.num,-1])
  paras=c(model$coefficients,(var(model$residuals)/length(model$residuals))) # use MLE of unconstrained

  # MCMC-Gibbs sampler
  iter.num=2000
  result.mat=matrix(NA,iter.num,length(paras))
  y.hat=matrix(NA,iter.num,nrow(us_wide_log[1:train.num,]))
  for(iter in 1:iter.num){
    # draw sigma^2
    result.mat[iter,length(paras)] = rinvgamma(n=1,shape = a_n, scale = b_n)
    result.mat[iter,-length(paras)] = mvrnorm(n=1,mu=mu_n,Sigma = result.mat[iter,length(paras)]* iDelta
    temp=cbind(1,us_wide_log[,-c(1,2)])
    y.hat[iter,]= X%*%result.mat[iter,-length(paras)]
  }
  X.new=cbind(1,us_wide_log[(1+train.num),-c(1,2)])
  if(!anyNA(X.new)){
    pred.mean=as.matrix(X.new)%*% t(result.mat[,-length(paras)])
    temp.Y=c()
    for(ii in 1:length(pred.mean)){
      temp.Y=c(temp.Y, rnorm(n=1,mean=pred.mean[ii],sd=sqrt(result.mat[ii,length(paras)])))
    }
    pred.Y= rbind(pred.Y, temp.Y) # incorporating the coef and sigma
  }
}

mixed=cbind(exp(y.hat[,1:30]),exp(t(pred.Y)))
for(i in 1:30){
  mixed[,i]=mean(mixed[,i])
}
colnames(mixed)=c(1:ncol(mixed))

boxplot(mixed,col=c(rep(4,30),rep(3,6)),ylab="inc death",xlab="Date")
points(us_wide$gt[1:30],col='blue')
lines(us_wide$gt[1:30],col='blue',lwd=2)
points(31:36,us_wide$gt[31:36],col='green')
lines(31:36,us_wide$gt[31:36],col='green',lwd=2)

title(main="gt VS. posterior prediction")
legend("topright",lwd=2,col=c("blue","green"),c("Train","Prediction"))
```


gt VS. posterior prediction
Train
Prediction
inc death
Date