Airlines Fatalities

Giacomo Maretto

28 febbraio 2019

Airlines Fatalities

Introduction

In this project we are analyzing data collected from 1976 to 2001 about airline fatal accidents by the International Civil Aviation Organization in Montreal , Canada (www.icao.int).

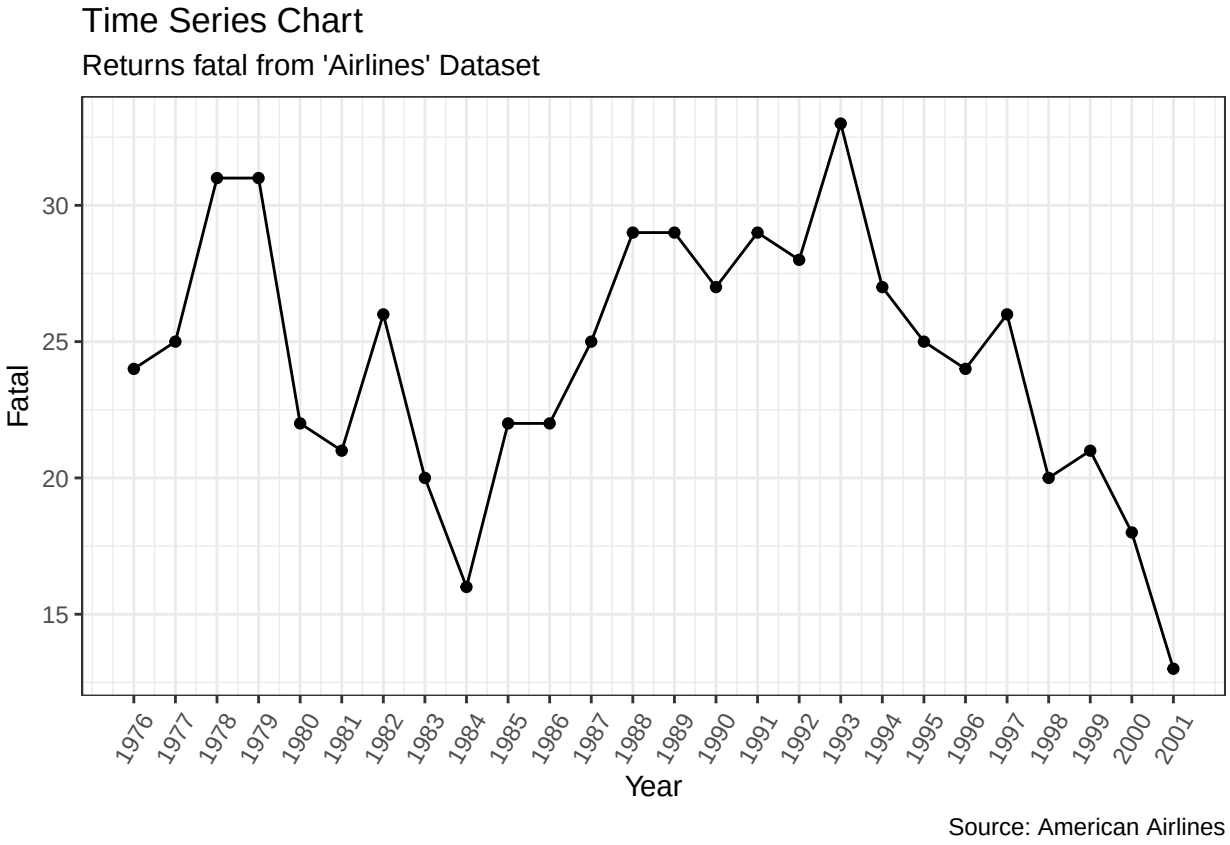
Our goal is to get a good prediction about future fatalitis through a bayesian approach.

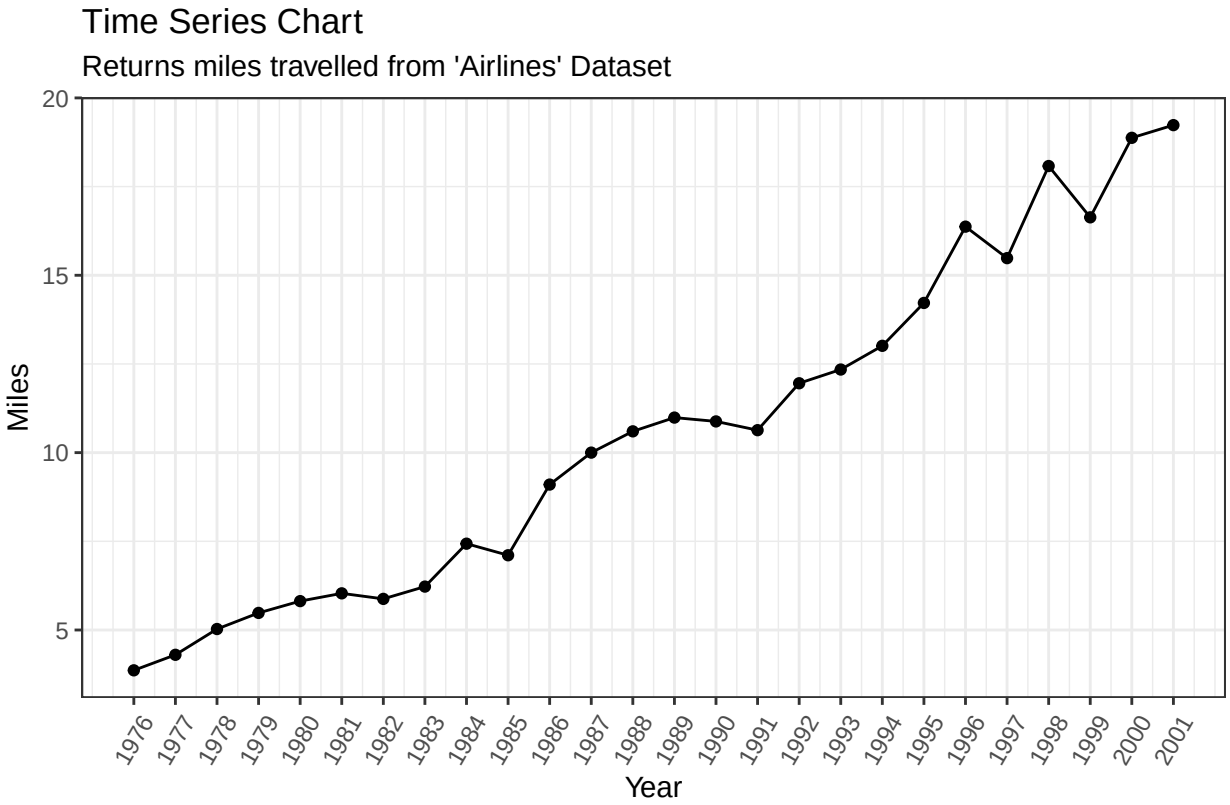
Our data is structured with four columns, year, fatal, miles, rate.

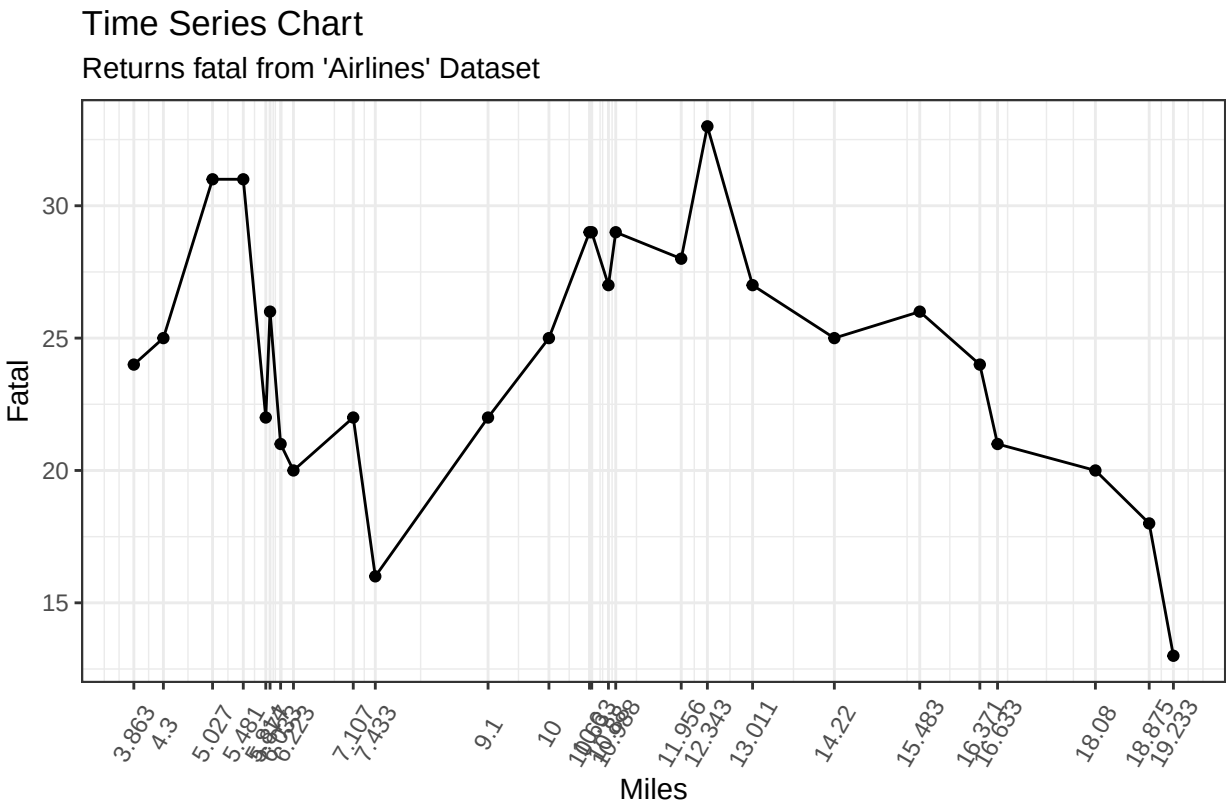
"Passenger miles" are in units of 10^{11} and the "accident rate" is the number of fatal accidents per 10^{11} passenger miles.

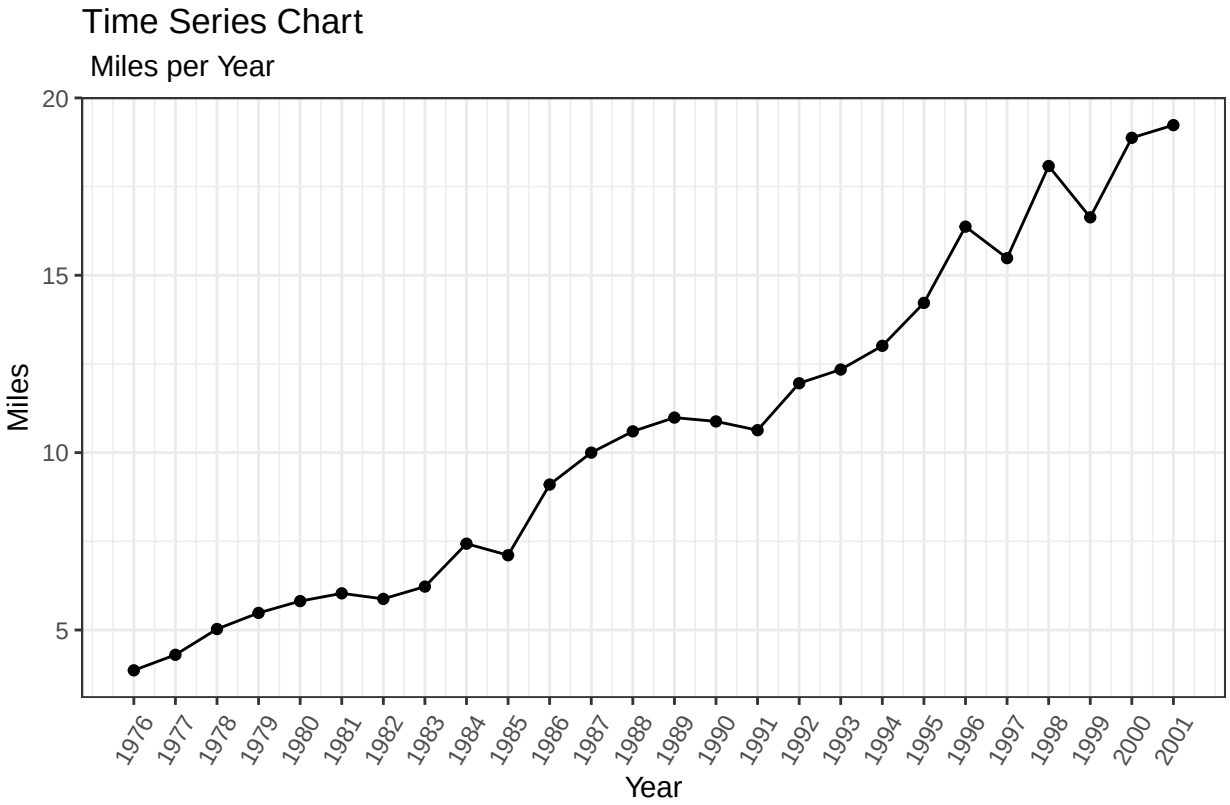
year	fatal	miles	rate
1976	24	3.863	6.213
1977	25	4.3	5.814
1978	31	5.027	6.167
1979	31	5.481	5.656
1980	22	5.814	3.784
1981	21	6.033	3.481

Following three descriptive time series plot respectively about the number of fatal accidents, the miles flown and the accident rate per each year.

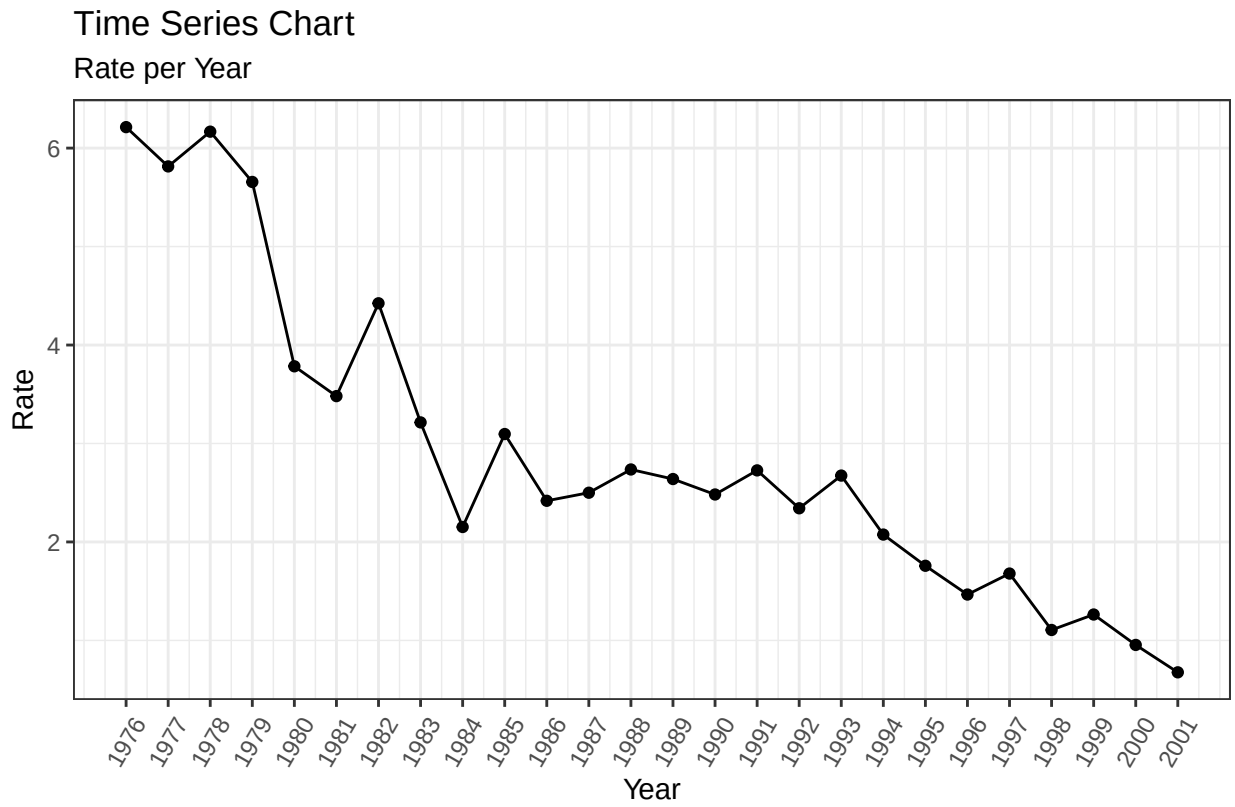



Time Series Chart
Returns miles travelled from 'Airlines' Dataset
Miles
20
15
10
5
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
Year
Source: American Airlines


Time Series Chart
Returns fatal from 'Airlines' Dataset
Fatal
30
25
20
15
3.863
4.3
5.027
7.107
7.433
9.1
10
11.956
12.343
13.011
14.22
15.483
16.371
16.633
18.08
18.875
19.233
Miles
Source: American Airlines



Source: American Airlines



Source: American Airlines

JAGS MODELS:

First model

```
cat( "model {
  for( i in 1:I ) {
    fatal[i] ~ dpois(mu)
    }
    mu ~ dgamma(0.01,0.01)
  }", file="a1.jag" )

a1.par <- c("mu","fatal[27]","fatal[28]","fatal[29]")

a1.ini <- list(list( mu=22 ),
               list( mu=23 ),
               list( mu=24 ) )

a1.dat <- list( fatal = c(airline$fatal,NA,NA,NA), I=29 )

# Model compilation and burn-in
a1.mod <- jags.model( file = "a1.jag",
                      data = a1.dat,
                      inits = a1.ini,
                      n.chains = 3,
                      n.adapt = 1000 )
```

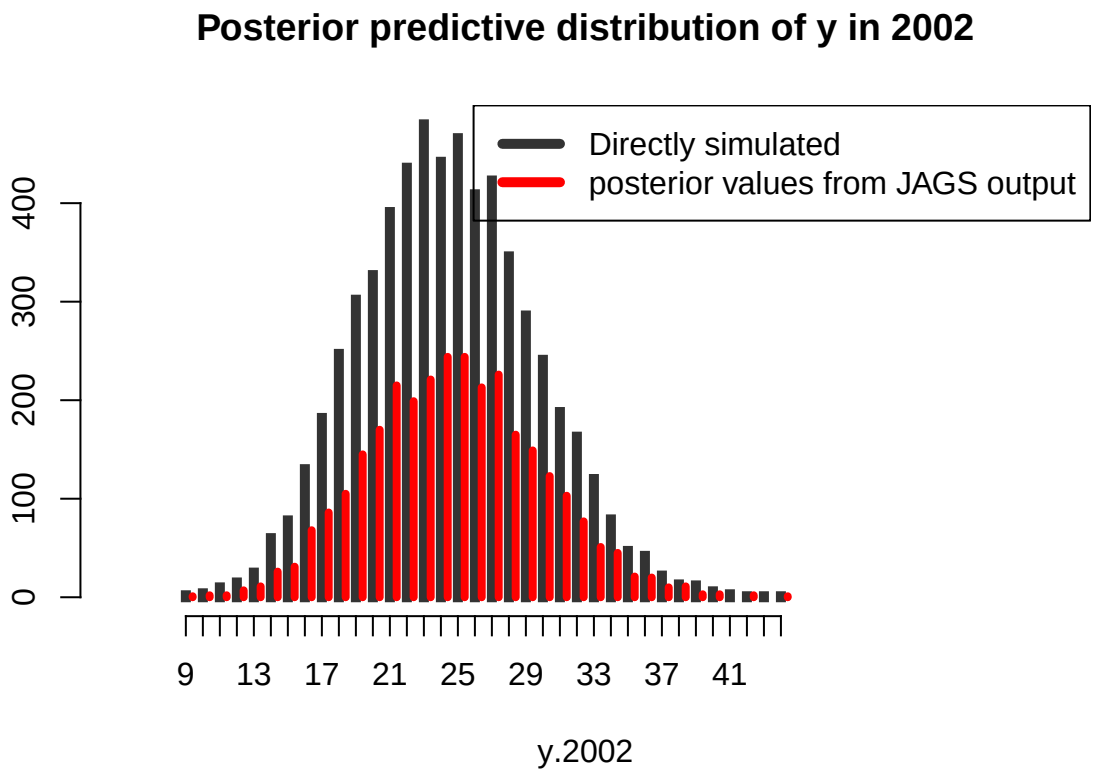
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 26
##    Unobserved stochastic nodes: 4
##    Total graph size: 32
##
## Initializing model
```

```
update(a1.mod,1000)

# Sampling from the posterior
a1.res <- coda.samples( a1.mod,
                        var = a1.par,
                        n.iter = 10000,
                        thin = 10 )
summary( a1.res )
```

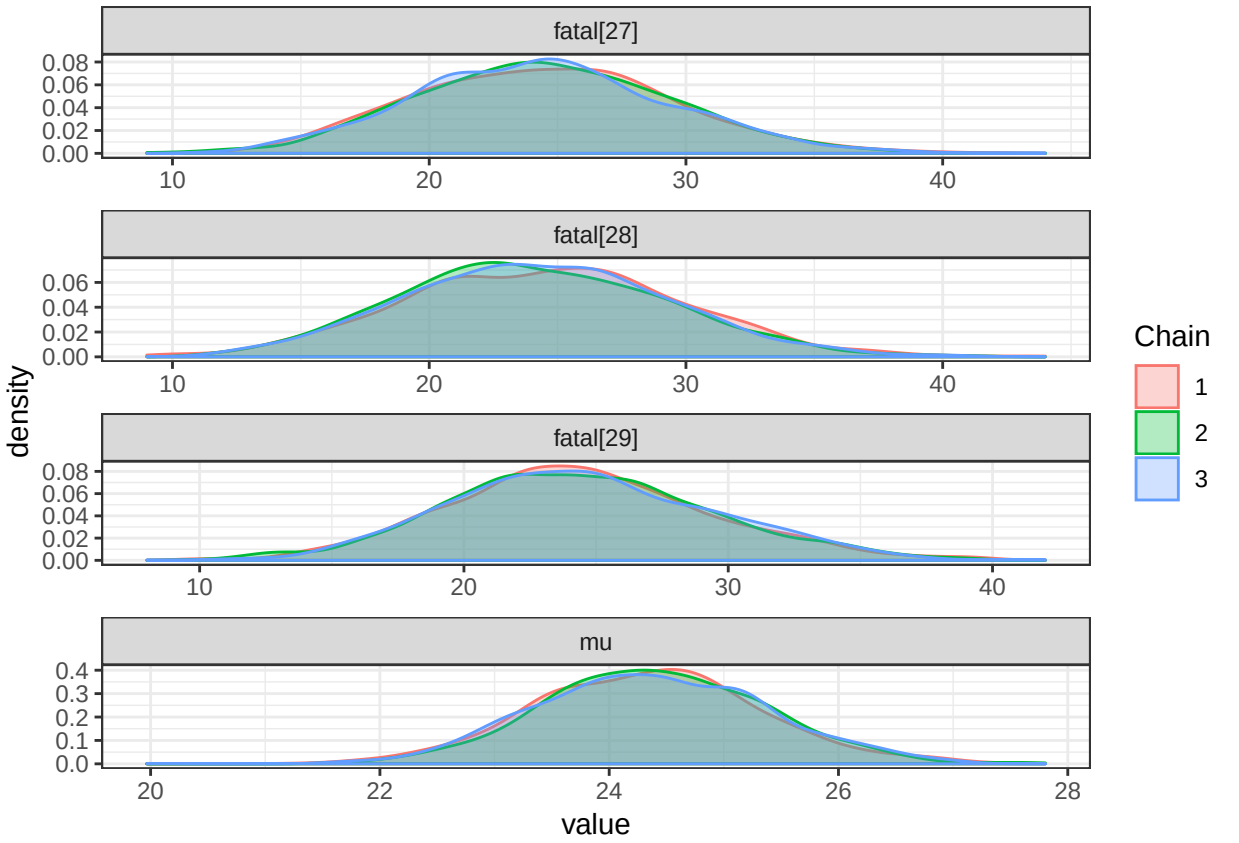
```
##
## Iterations = 1010:11000
## Thinning interval = 10
## Number of chains = 3
## Sample size per chain = 1000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##             Mean     SD Naive SE Time-series SE
## fatal[27] 24.46 4.9572  0.09051        0.09053
## fatal[28] 24.26 5.1762  0.09450        0.09448
## fatal[29] 24.41 4.9616  0.09059        0.08926
## mu        24.37 0.9799  0.01789        0.01749
##
## 2. Quantiles for each variable:
##
##             2.5%   25%   50%   75% 97.5%
## fatal[27] 15.00 21.00 24.00 28.00 34.00
## fatal[28] 15.00 21.00 24.00 28.00 35.00
## fatal[29] 15.00 21.00 24.00 28.00 35.00
## mu        22.48 23.68 24.36 25.04 26.34
```

```
theta <- rgamma(6000, 634, 26 )
y.2002 <- rpois(6000,theta)
plot( main = 'Posterior predictive distribution of y in 2002',table(y.2002), type="h", lwd=5, lend=2, co
tpr <- table( as.matrix( a1.res[,"fatal[27]"] ) )
points( as.numeric(names(tpr))+0.4, tpr, type="h", col="red", lwd=4 )
legend("topright", c('Directly simulated','posterior values from JAGS output'),col=c(gray(0.2),'red'),lw
```

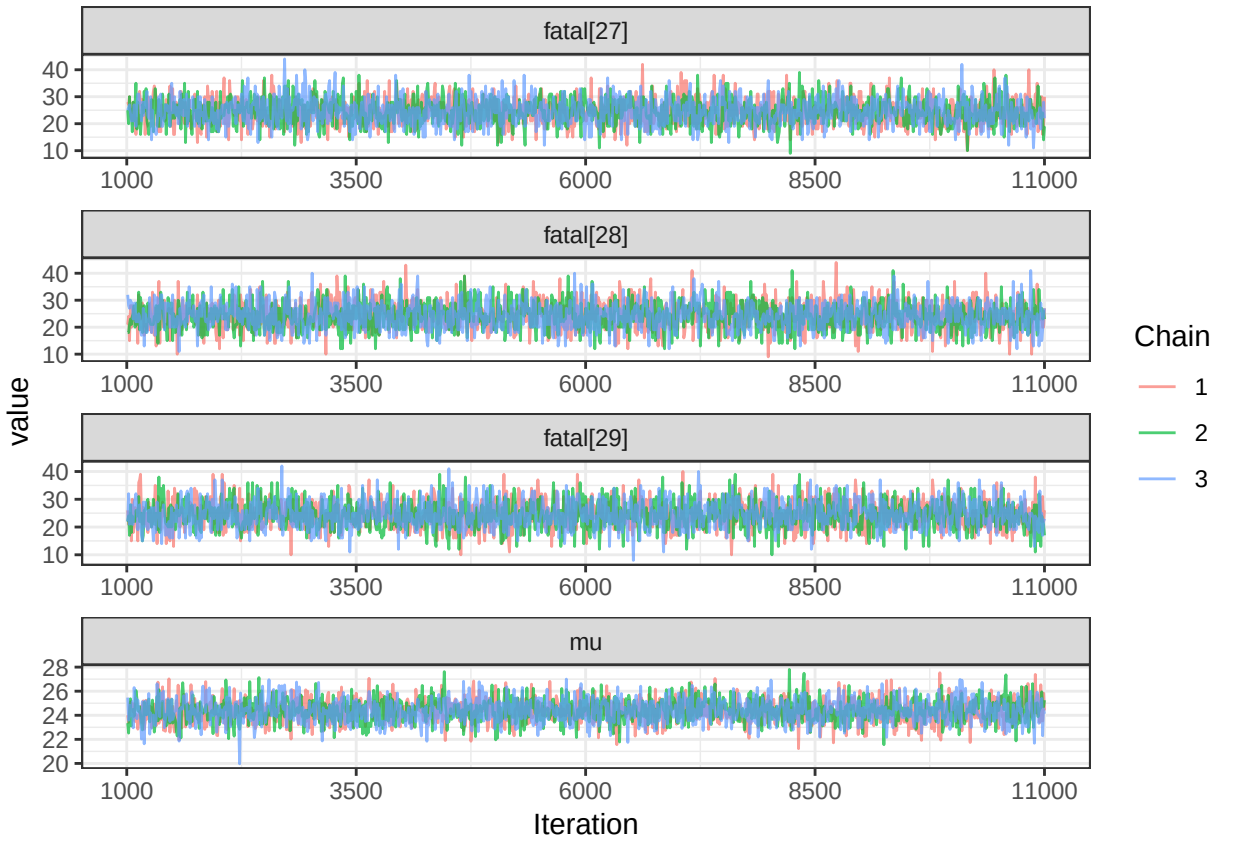


Diagnostics for model 1

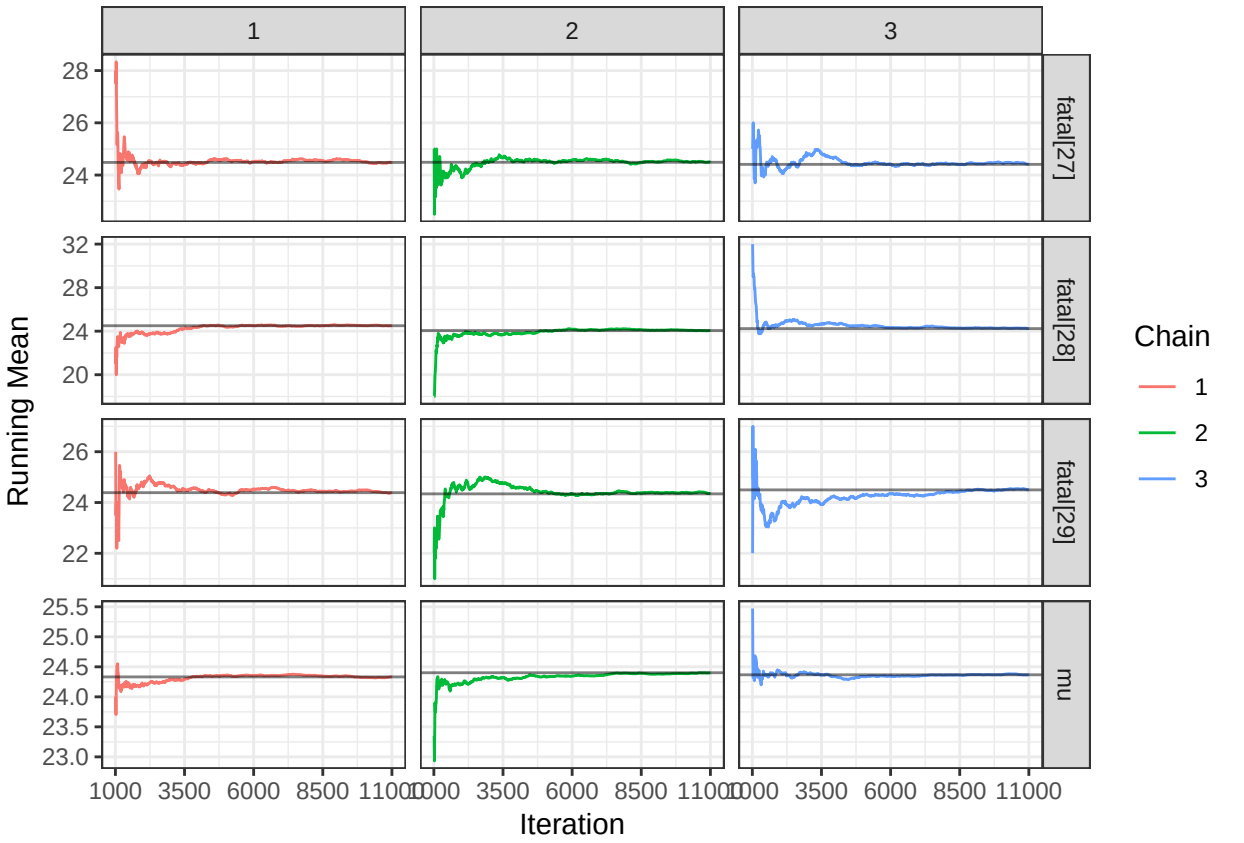
```
result = ggs(a1.res)
ggs_density(result)
```



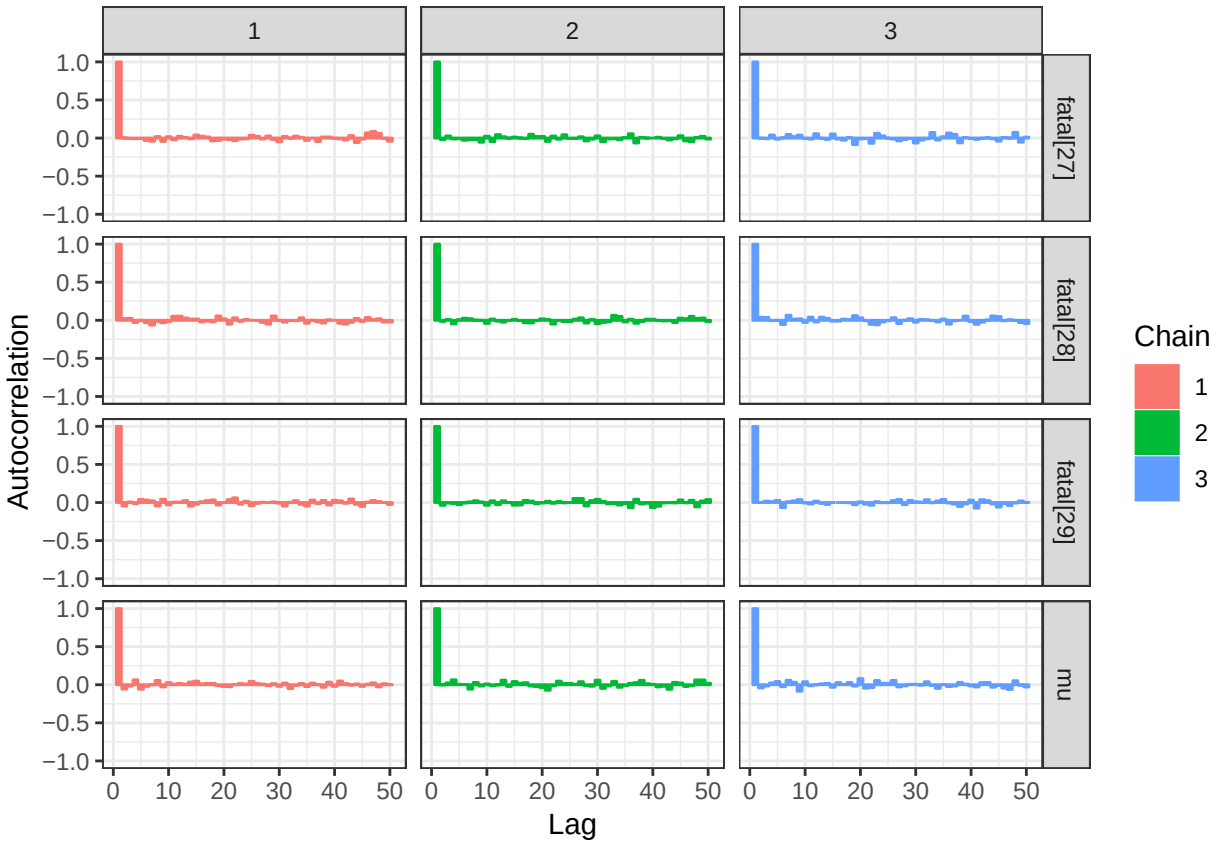

```
ggs_traceplot(result)
```




```
ggs_running(result)
```




```
ggs_autocorrelation(result)
```



Third model

```
library(rjags)
 cat( "model
 {
 for( i in 1:I )
 {
 fatal[i] ~ dpois( mu[i] )
 log(mu[i]) <- alpha + beta*miles[i]+ beta2 * miles[i]*rate[i]
 }
 alpha ~ dnorm(0,0.0001)
 beta ~ dnorm(0.0001,0.00001)
 beta2 ~ dnorm(0.0001,0.00001)
 }",

 file="a3.jag" )

 a3.ini= list(
    list(alpha = 0.01, beta = 0.6, beta2=0.1),
    list(alpha = 0.2, beta = 0.2, beta2=0.2),
    list(alpha = 0.7, beta = 0.7, beta2=0.3))
 a3.dat <- list( rate= c(airline$rate,0.7080,0.3,0.4),fatal=c(airline$fatal,NA,NA,NA), miles=c(airline$
 a3.par <- c("alpha","beta","beta2","fatal[27]","fatal[28]","fatal[29]")
```

```
# Model compilation and burn-in
a3.mod <- jags.model(file = "a3.jag",data = a3.dat,inits = a3.ini,n.chains = 3,n.adapt = 1000)
```

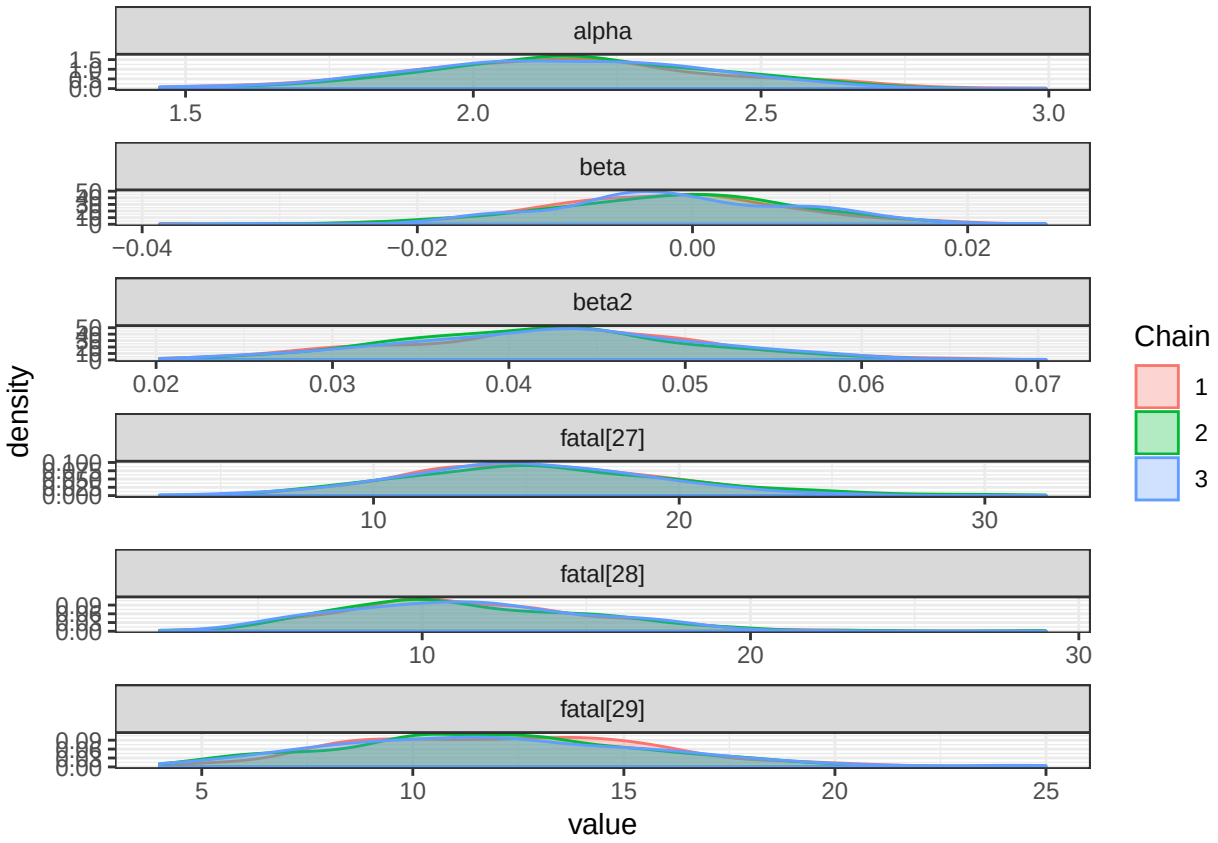
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 26
##    Unobserved stochastic nodes: 6
##    Total graph size: 210
##
## Initializing model
```

```
update(a3.mod,1000)
# Sampling from the posterior
a3.res <- coda.samples( a3.mod,var = a3.par,n.iter = 10000,thin = 50 )
summary( a3.res )
```

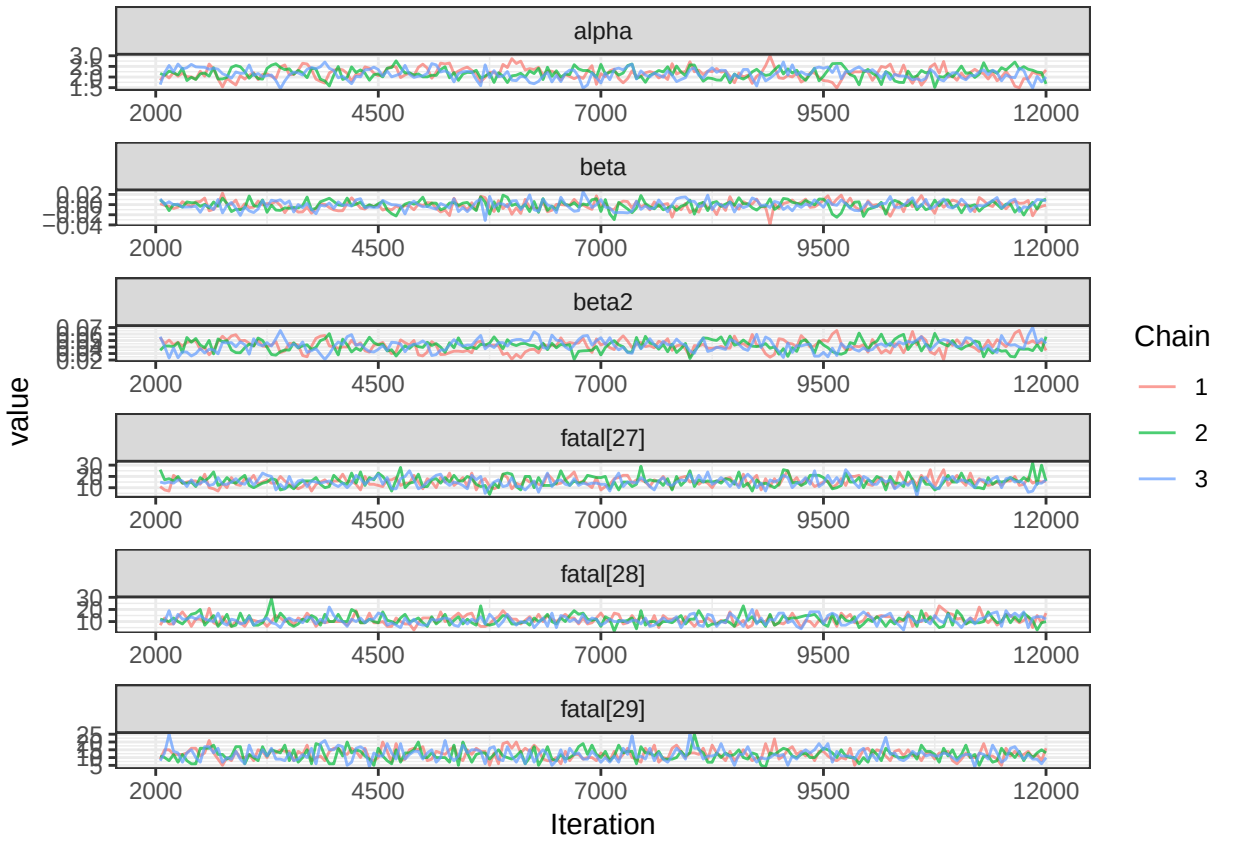
```
##
## Iterations = 2050:12000
## Thinning interval = 50
## Number of chains = 3
## Sample size per chain = 200
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                Mean       SD  Naive SE Time-series SE
## alpha      2.157789 0.255229 0.0104197      0.0156726
## beta      -0.001531 0.008970 0.0003662      0.0003665
## beta2      0.042269 0.008507 0.0003473      0.0005480
## fatal[27] 15.340000 4.278424 0.1746659      0.1625348
## fatal[28] 11.373333 3.783448 0.1544586      0.1423056
## fatal[29] 11.998333 3.683424 0.1503751      0.1366078
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%    97.5%
## alpha      1.64099  1.990990  2.162948  2.326468  2.63506
## beta      -0.01880 -0.006728 -0.001383  0.004598  0.01544
## beta2      0.02564  0.036597  0.042649  0.047512  0.05911
## fatal[27]  8.00000 12.000000 15.000000 18.000000 24.02500
## fatal[28]  5.00000  9.000000 11.000000 14.000000 19.00000
## fatal[29]  5.00000  9.000000 12.000000 14.000000 19.02500
```

Model 3 diagnostics

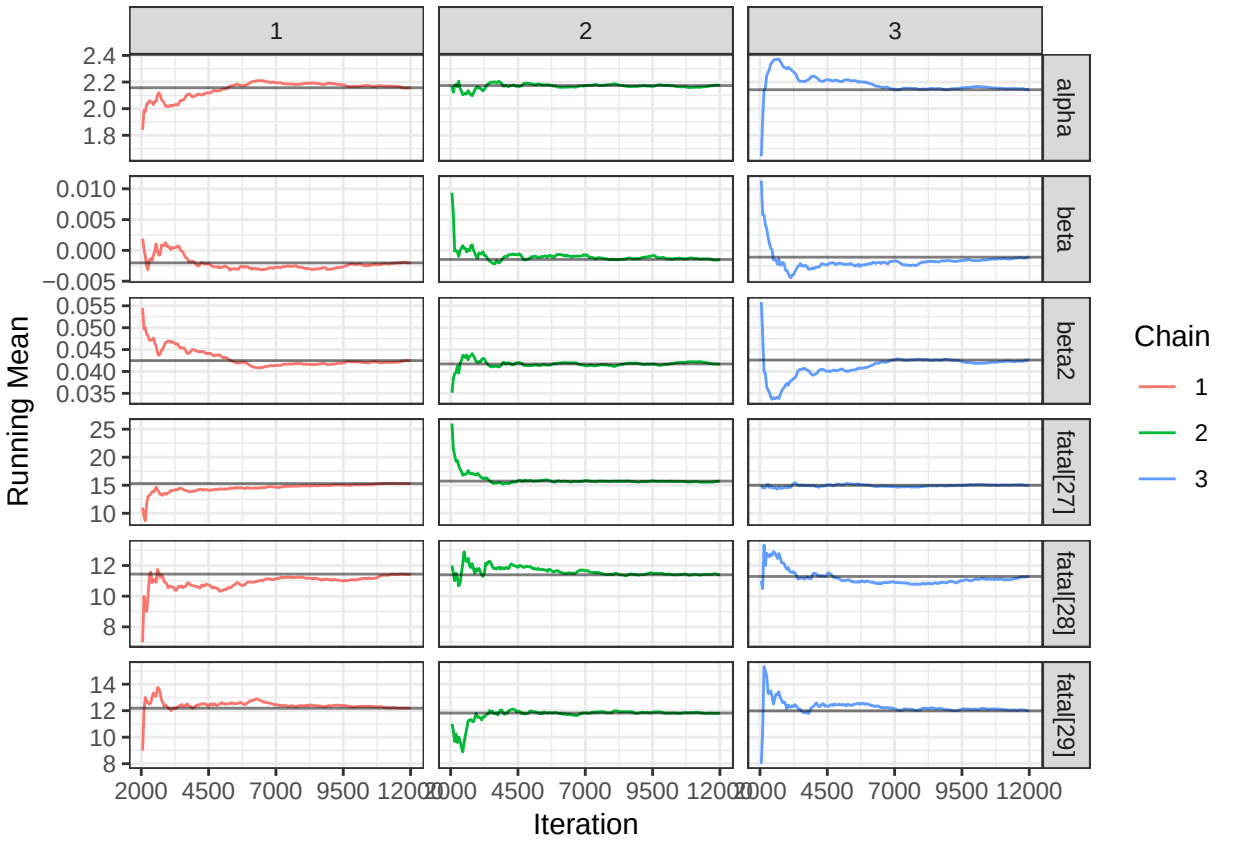
```
result3 = ggs(a3.res)
ggs_density(result3)
```



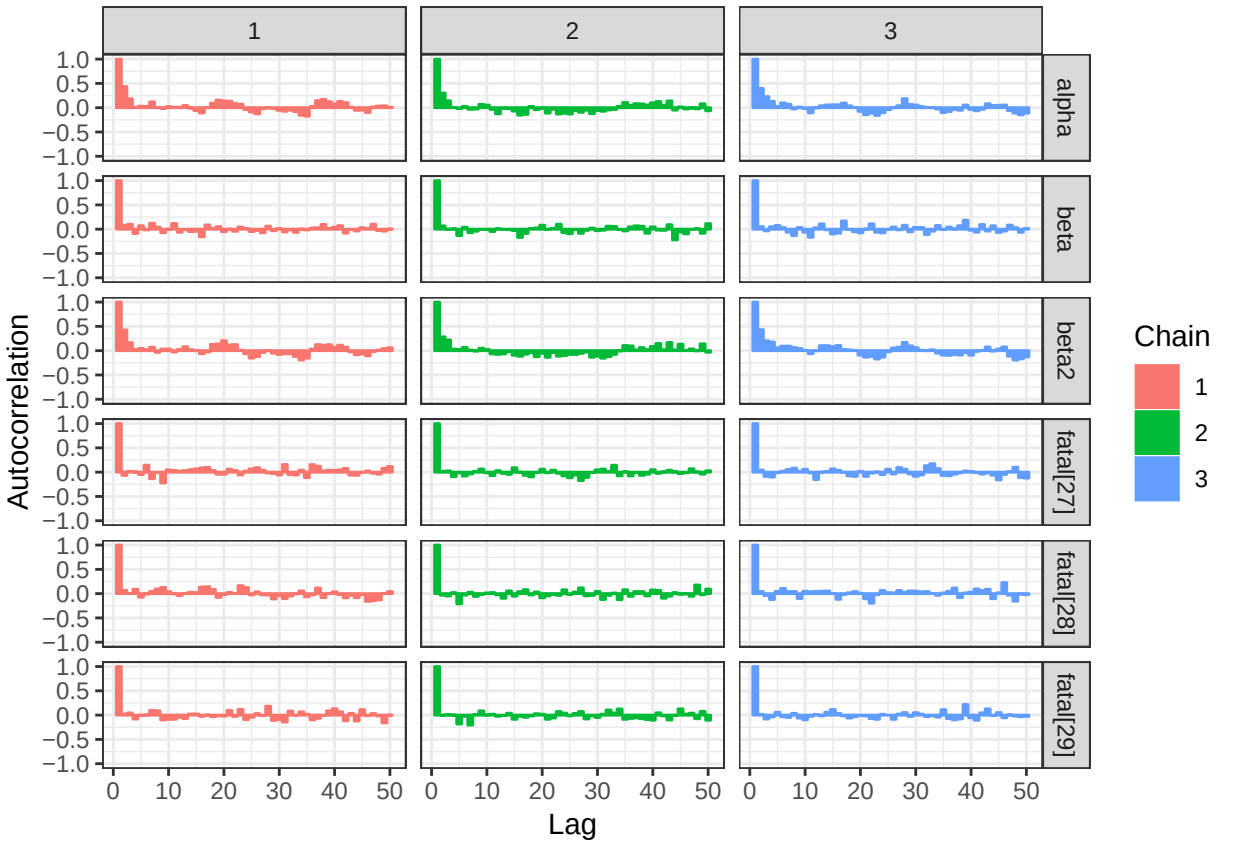

```
ggs_traceplot(result3)
```




```
ggs_running(result3)
```

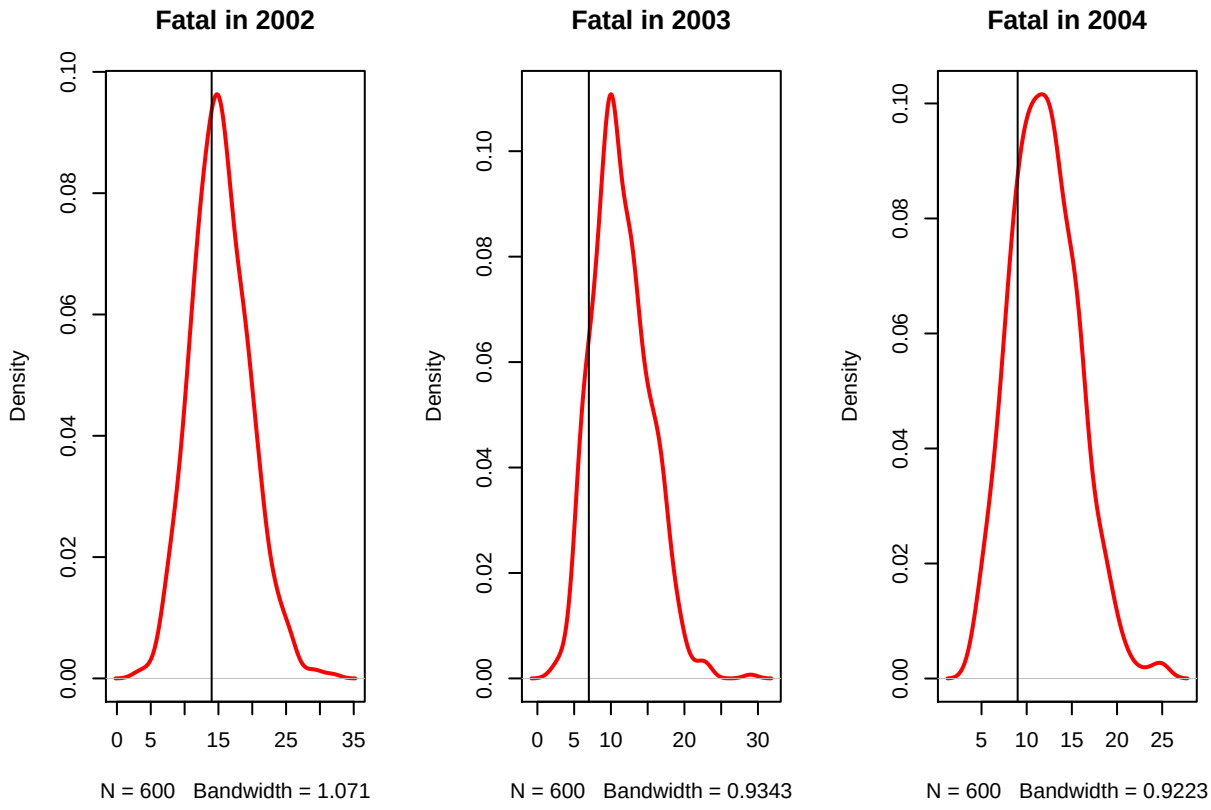



```
ggs_autocorrelation(result3)
```



Predicting values in model 3

```
par(mfrow=c(1,3))
a3.m <- as.matrix(a3.res)
plot(density(a3.m[,"fatal[27]"]),main="Fatal in 2002", col="red", lwd=2)
abline( v=14,col="black")
a3.m <- as.matrix(a3.res)
plot(density(a3.m[,"fatal[28]"]),main="Fatal in 2003", col="red", lwd=2)
abline( v=7,col="black")
a3.m <- as.matrix(a3.res)
plot(density(a3.m[,"fatal[29]"]),main="Fatal in 2004", col="red", lwd=2)
abline( v=9,col="black")
```



Forth model

```
library(rjags)
 cat( "model
 {
 for( i in 1:I )
 {
 fatal[i] ~ dpois( mu[i] )
 log(mu[i]) <- alpha + beta2 * miles[i]*rate[i]
 }
 alpha ~ dnorm(0,0.0001)

 beta2 ~ dnorm(0.0001,0.00001)
 }",

 file="a4.jag" )

 a4.ini= list(
    list(alpha = 0.01, beta2=0.1),
    list(alpha = 0.2, beta2=0.2),
    list(alpha = 0.7,  beta2=0.3))
 a4.dat <- list( rate= c(airline$rate,0.7080,0.3,0.4),fatal=c(airline$fatal,NA,NA,NA), miles=c(airline$
 a4.par <- c("alpha","beta2","fatal[27]","fatal[28]","fatal[29]")

 # Model compilation and burn-in
 a4.mod <- jags.model(file = "a4.jag",data = a4.dat,inits = a4.ini,n.chains = 3,n.adapt = 1000)
```

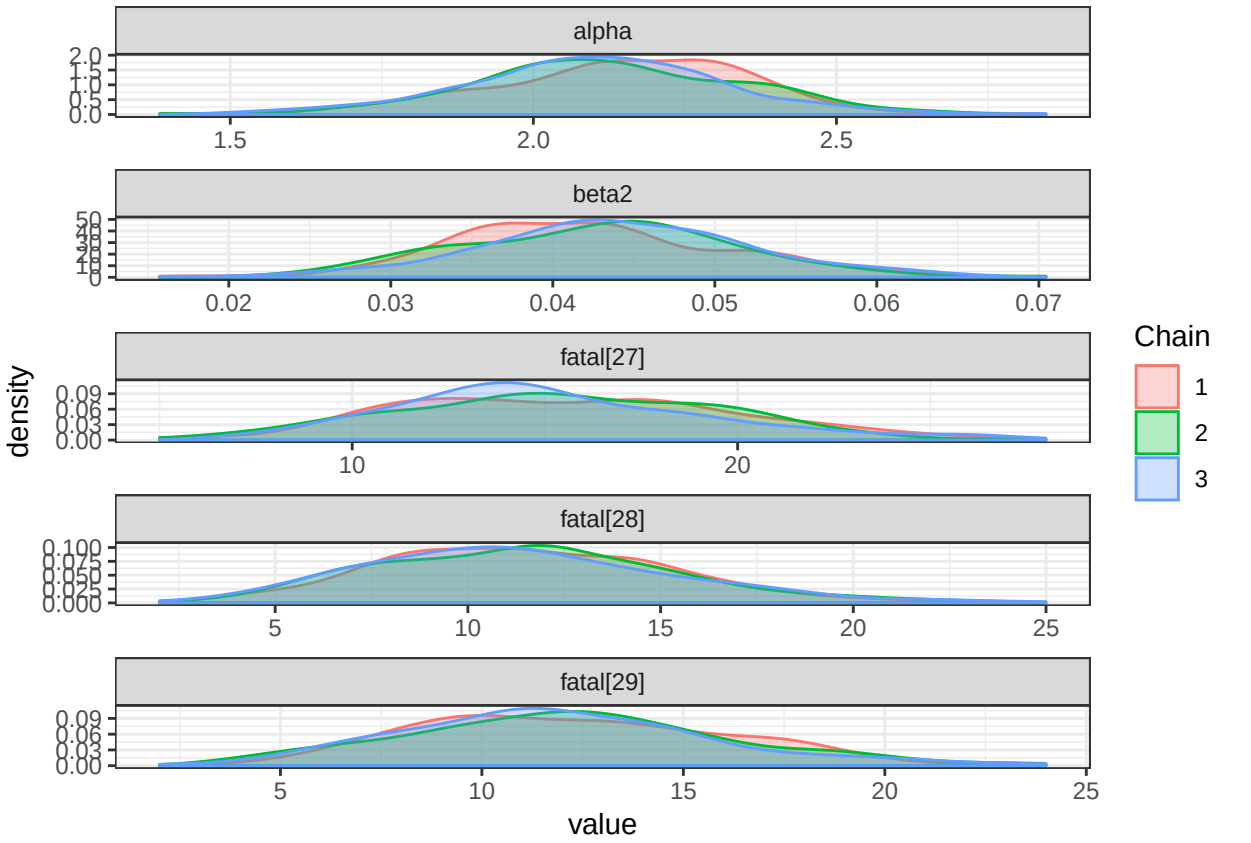
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 26
##    Unobserved stochastic nodes: 5
##    Total graph size: 180
##
## Initializing model
```

```
update(a4.mod,1000)
# Sampling from the posterior
a4.res <- coda.samples( a4.mod,var = a4.par,n.iter = 10000,thin = 50 )
summary( a4.res )
```

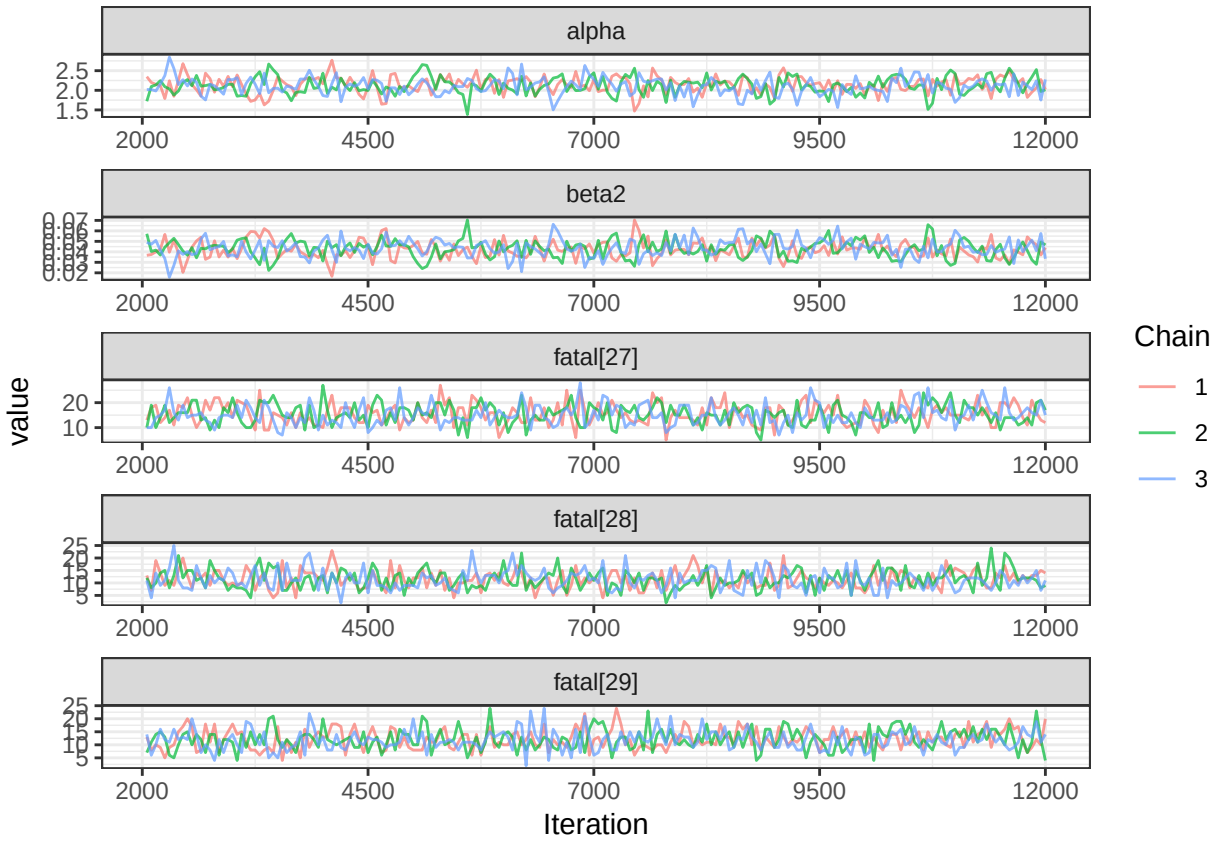
```
##
## Iterations = 2050:12000
## Thinning interval = 50
## Number of chains = 3
## Sample size per chain = 200
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean       SD  Naive SE Time-series SE
## alpha      2.12438 0.216094 0.0088220      0.0106781
## beta2      0.04294 0.008454 0.0003451      0.0004023
## fatal[27] 15.36667 4.179873 0.1706426      0.1845807
## fatal[28] 11.45667 3.872525 0.1580952      0.1583297
## fatal[29] 12.07667 3.890503 0.1588291      0.1589362
##
## 2. Quantiles for each variable:
##
##             2.5%      25%      50%      75%    97.5%
## alpha     1.6888  1.99182  2.12813  2.27721  2.52951
## beta2     0.0272  0.03714  0.04288  0.04846  0.06034
## fatal[27] 8.0000 12.00000 15.00000 18.00000 24.00000
## fatal[28] 5.0000  9.00000 11.00000 14.00000 20.00000
## fatal[29] 5.0000  9.00000 12.00000 15.00000 20.00000
```

Model 4 diagnostics

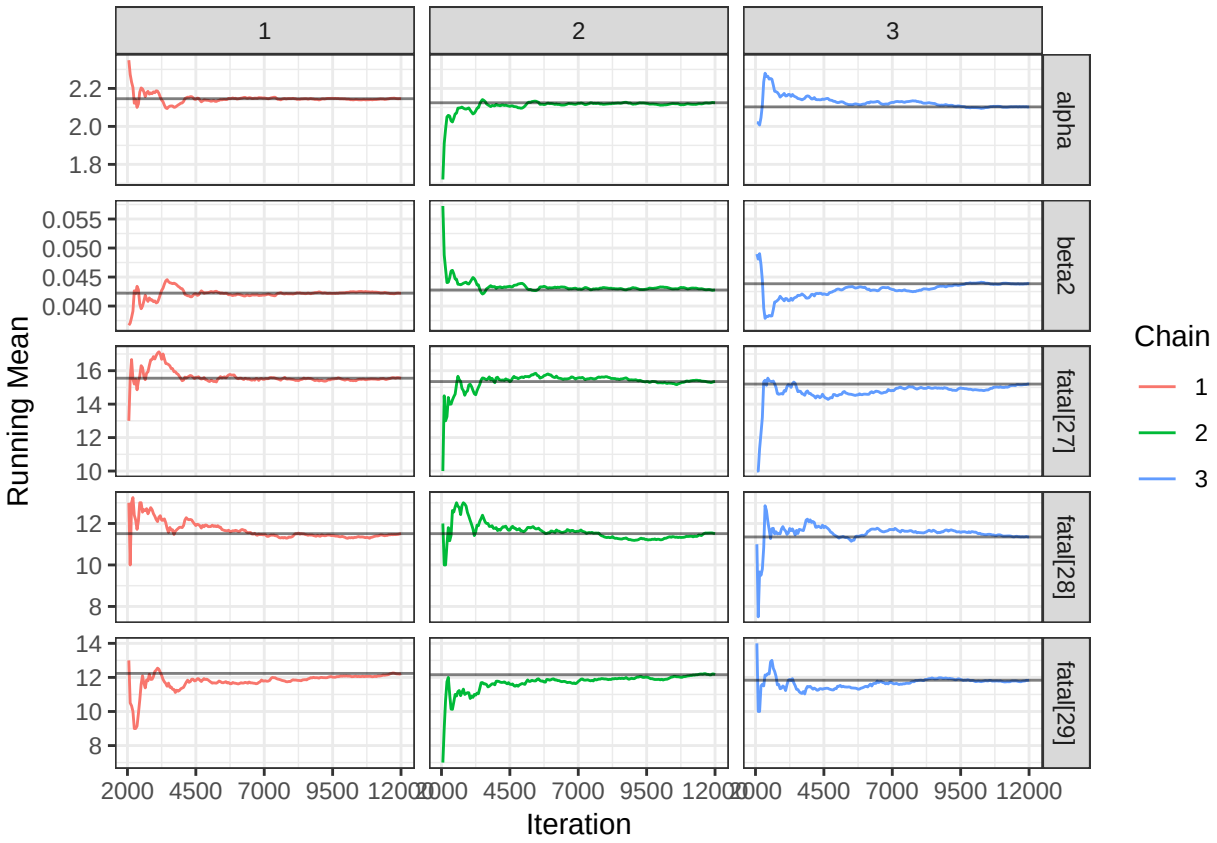
```
result4 = ggs(a4.res)
ggs_density(result4)
```



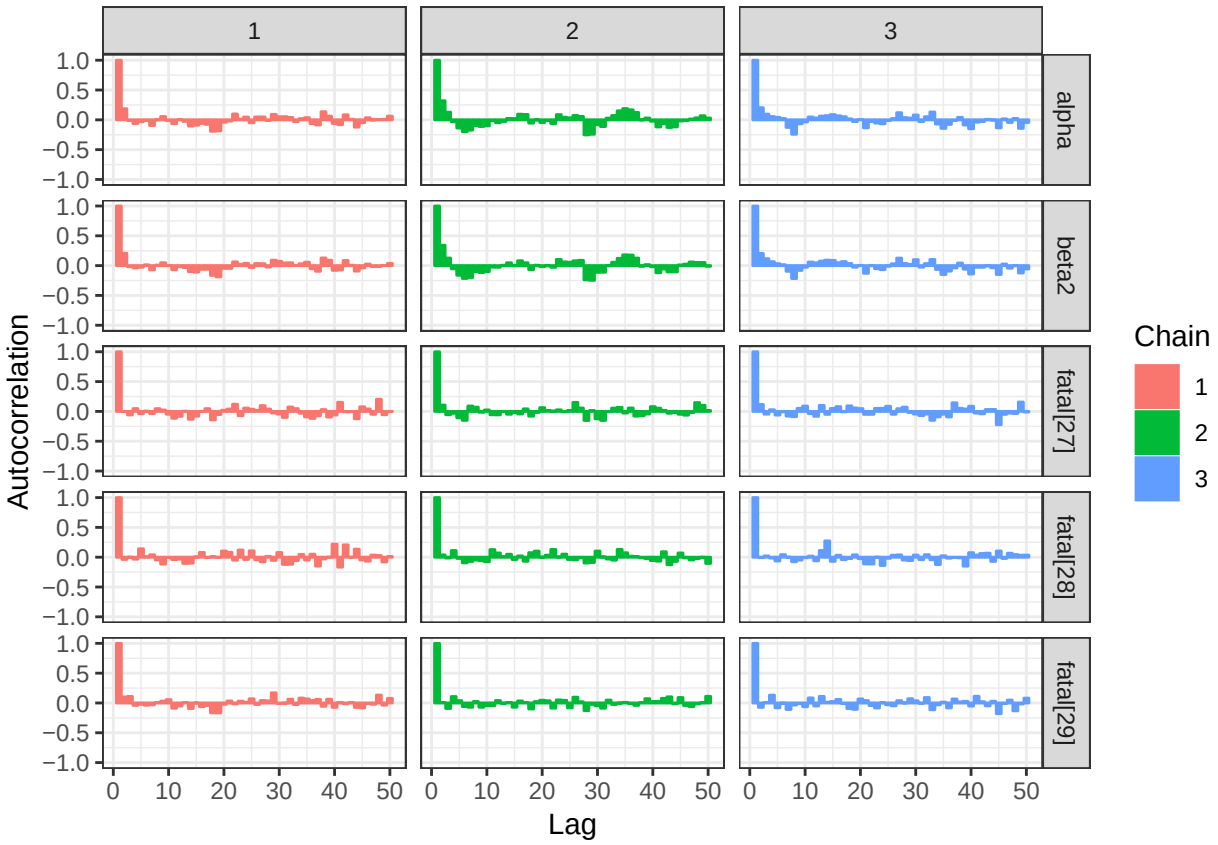

```
ggs_traceplot(result4)
```




```
ggs_running(result4)
```

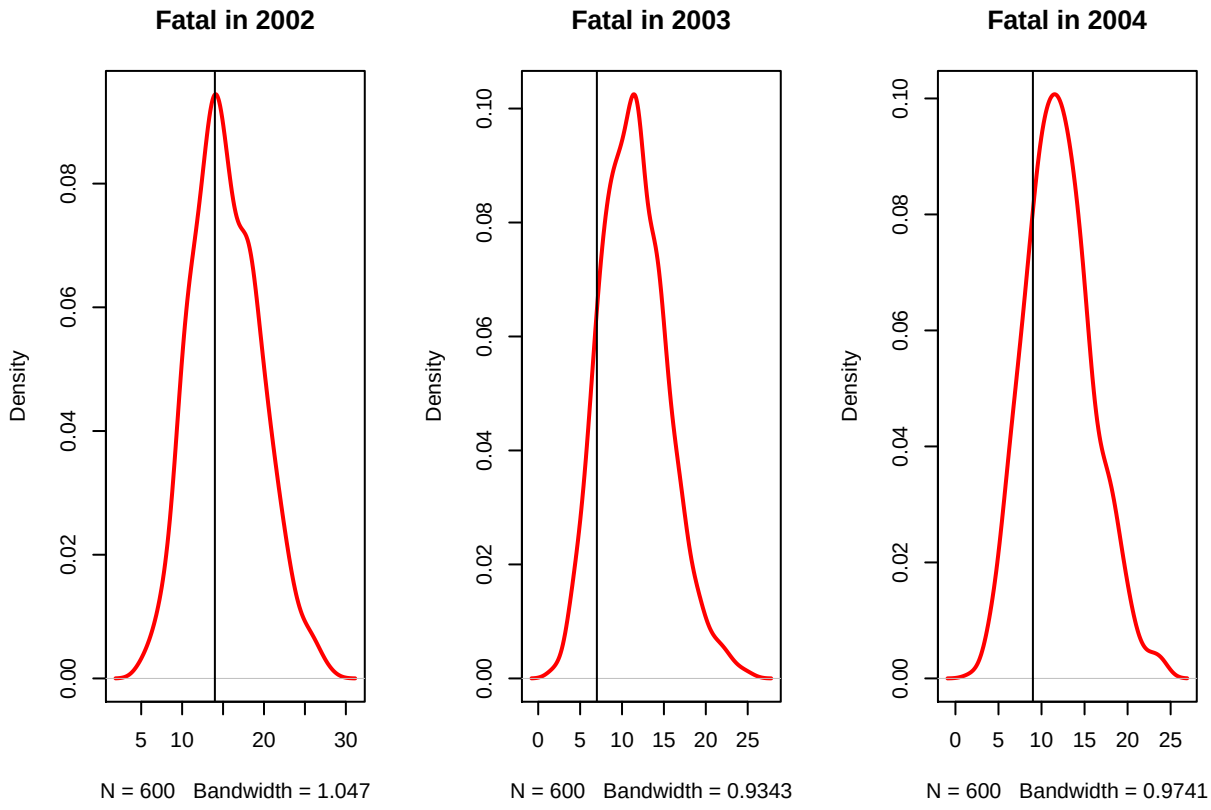



```
ggs_autocorrelation(result4)
```



Predicting values in model 4

```
par(mfrow=c(1,3))
a4.m <- as.matrix(a4.res)
plot(density(a4.m[,"fatal[27]"]),main="Fatal in 2002", col="red", lwd=2)
abline( v=14,col="black")
a4.m <- as.matrix(a4.res)
plot(density(a4.m[,"fatal[28]"]),main="Fatal in 2003", col="red", lwd=2)
abline( v=7,col="black")
a4.m <- as.matrix(a4.res)
plot(density(a4.m[,"fatal[29]"]),main="Fatal in 2004", col="red", lwd=2)
abline( v=9,col="black")
```



Model 5 test with normal approximation

```
library(rjags)
 cat( "model
 {
 for( i in 1:I )
 {
 fatal[i] ~ dnorm( mu[i],1/mu[i] )
 mu[i] <- alpha + beta2 * miles[i]
 }
 alpha ~ dnorm(0,0.0001)
 beta2 ~ dnorm(0,0.0001)
 }",
 file="a5.jag" )

 a5.ini= list(
    list(alpha = 0.01, beta2=3),
    list(alpha = 0.2, beta2=1),
    list(alpha = 0.7, beta2=2))
 a5.dat <- list( rate = c(airline$rate,0.7080,0.3,0.4),fatal=c(airline$fatal,NA,NA,NA), miles=c(airline$
 a5.par <- c("alpha","beta2","fatal[27]","fatal[28]","fatal[29]")

 # Model compilation and burn-in
 a5.mod <- jags.model(file = "a5.jag",data = a5.dat,inits = a5.ini,n.chains = 3,n.adapt = 1000 )
```

```
## Warning in jags.model(file = "a5.jag", data = a5.dat, inits = a5.ini,
## n.chains = 3, : Unused variable "rate" in data
```

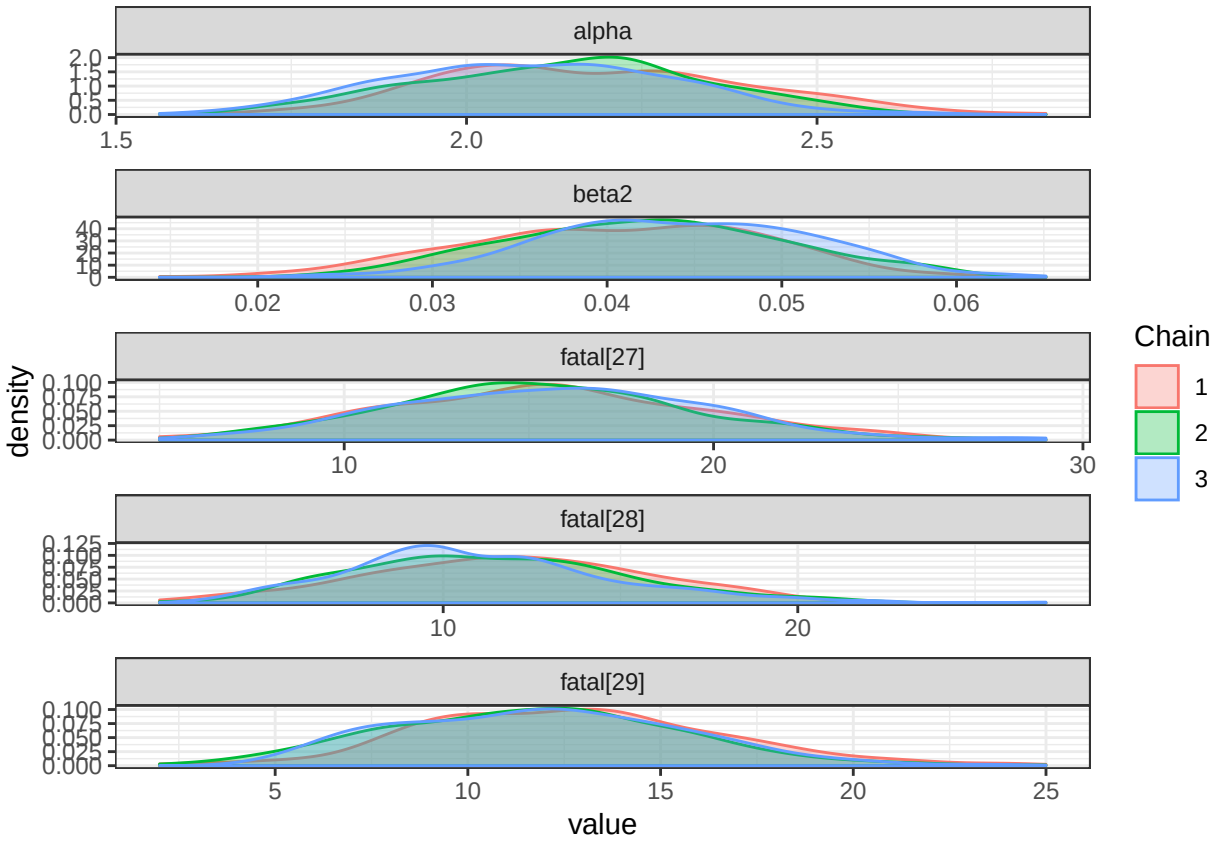
```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 26
##    Unobserved stochastic nodes: 5
##    Total graph size: 151
##
## Initializing model
```

```
# Sampling from the posterior
update(a5.mod,1000)
a5.res <- coda.samples( a4.mod,var = a4.par,n.iter = 3000,thin = 10 )
summary( a5.res )
```

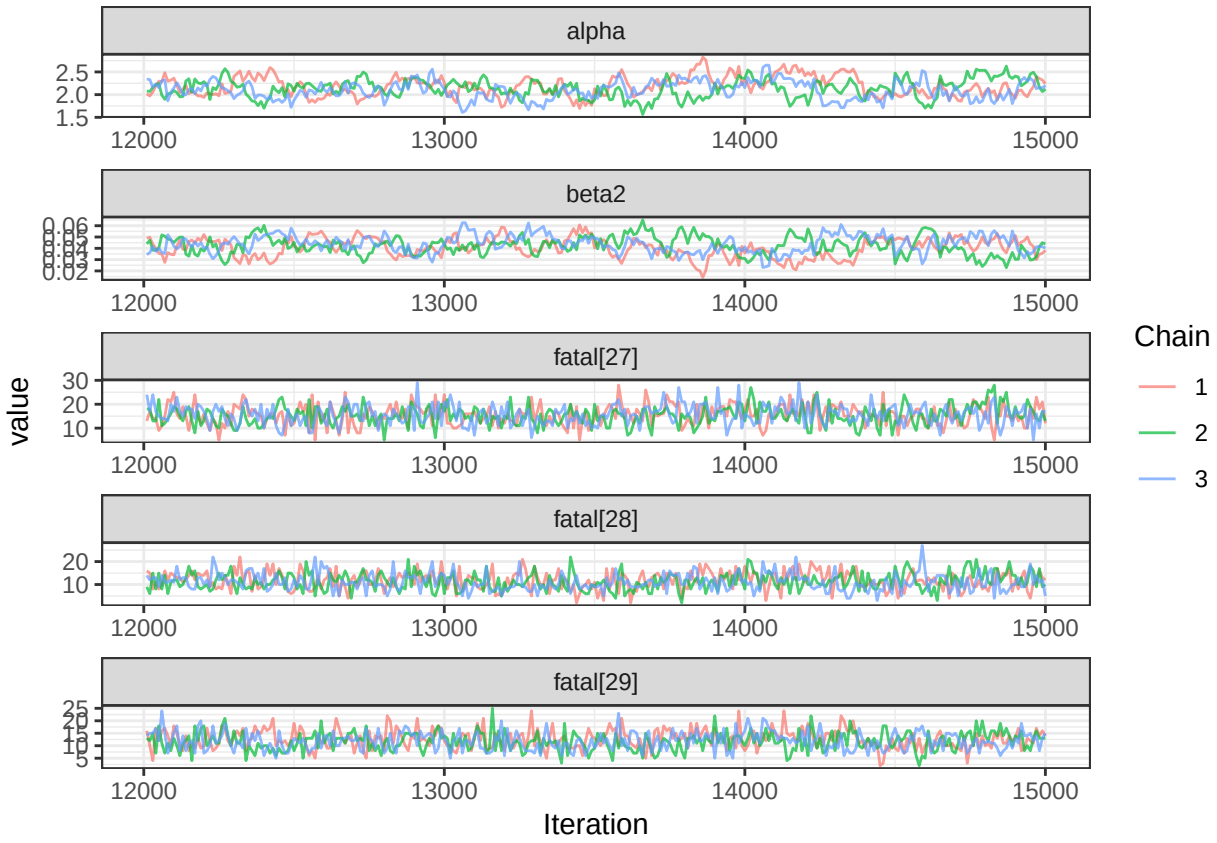
```
##
## Iterations = 12010:15000
## Thinning interval = 10
## Number of chains = 3
## Sample size per chain = 300
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean       SD  Naive SE Time-series SE
## alpha      2.13946 0.207691 0.0069230      0.0211400
## beta2      0.04228 0.008088 0.0002696      0.0008388
## fatal[27] 15.48333 4.200581 0.1400194      0.1954485
## fatal[28] 11.43000 3.832794 0.1277598      0.1548190
## fatal[29] 12.10444 3.779251 0.1259750      0.1680841
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%    97.5%
## alpha     1.75429  1.99413  2.13811  2.28086  2.55063
## beta2     0.02626  0.03673  0.04242  0.04796  0.05746
## fatal[27] 8.00000 13.00000 15.00000 18.00000 24.00000
## fatal[28] 5.00000  9.00000 11.00000 14.00000 19.52500
## fatal[29] 5.00000  9.00000 12.00000 15.00000 20.00000
```

Model5 diagnostics

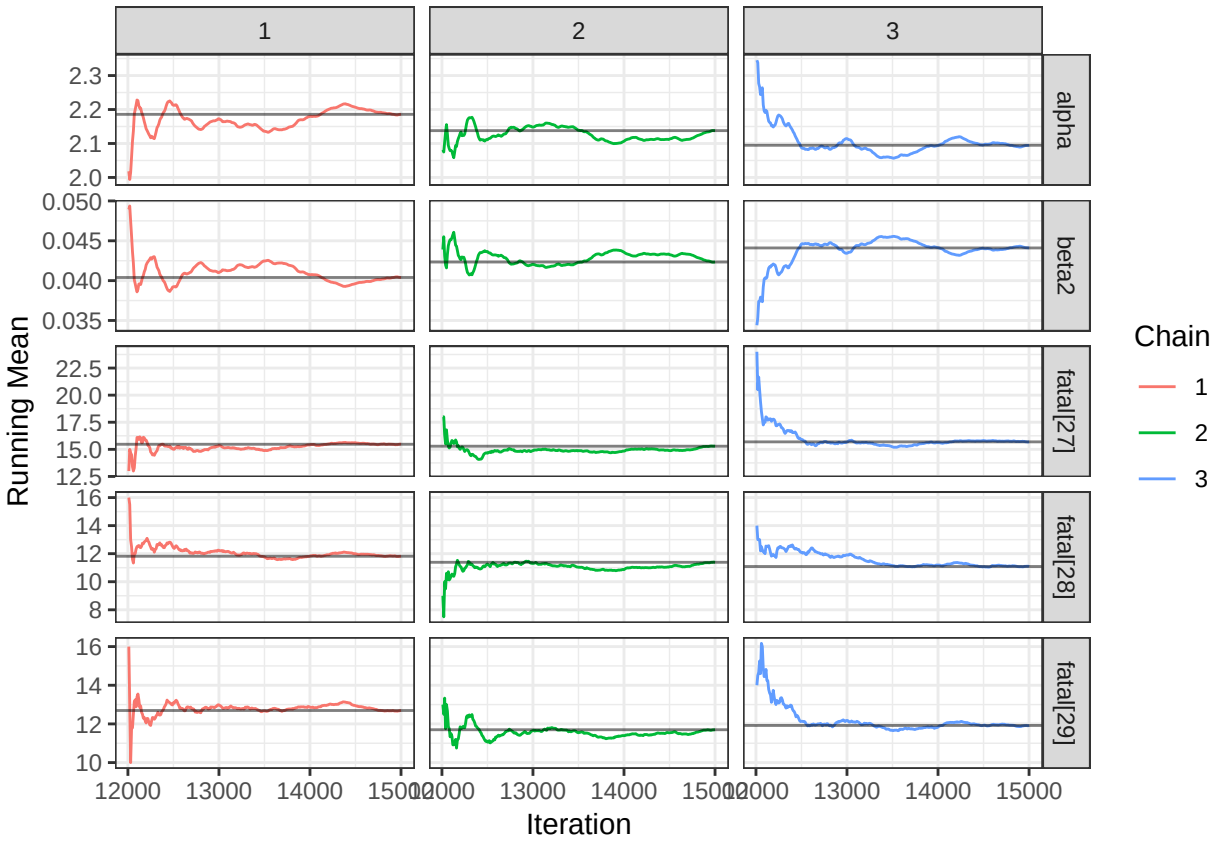
```
result5 = ggs(a5.res)
ggs_density(result5)
```



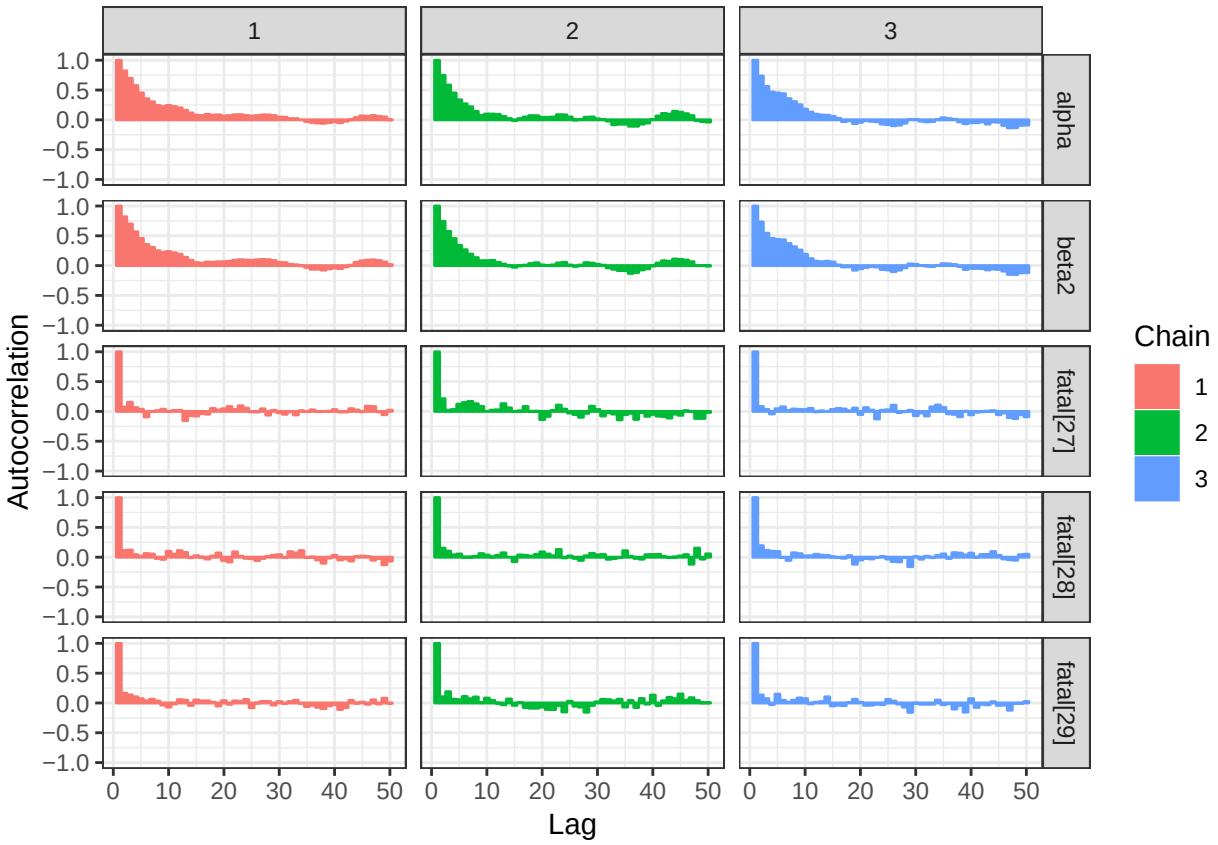

```
ggs_traceplot(result5)
```




```
ggs_running(result5)
```




```
ggs_autocorrelation(result5)
```



Predicting values in model 5

```
par(mfrow=c(1,3))
a5.m <- as.matrix(a5.res)
plot(density(a5.m[,"fatal[27]"]),main="Fatal in 2002", col="red", lwd=2)
abline( v=14,col="black")
a5.m <- as.matrix(a5.res)
plot(density(a5.m[,"fatal[28]"]),main="Fatal in 2003", col="red", lwd=2)
abline( v=7,col="black")
a5.m <- as.matrix(a5.res)
plot(density(a5.m[,"fatal[29]"]),main="Fatal in 2004", col="red", lwd=2)
abline( v=9,col="black")
```

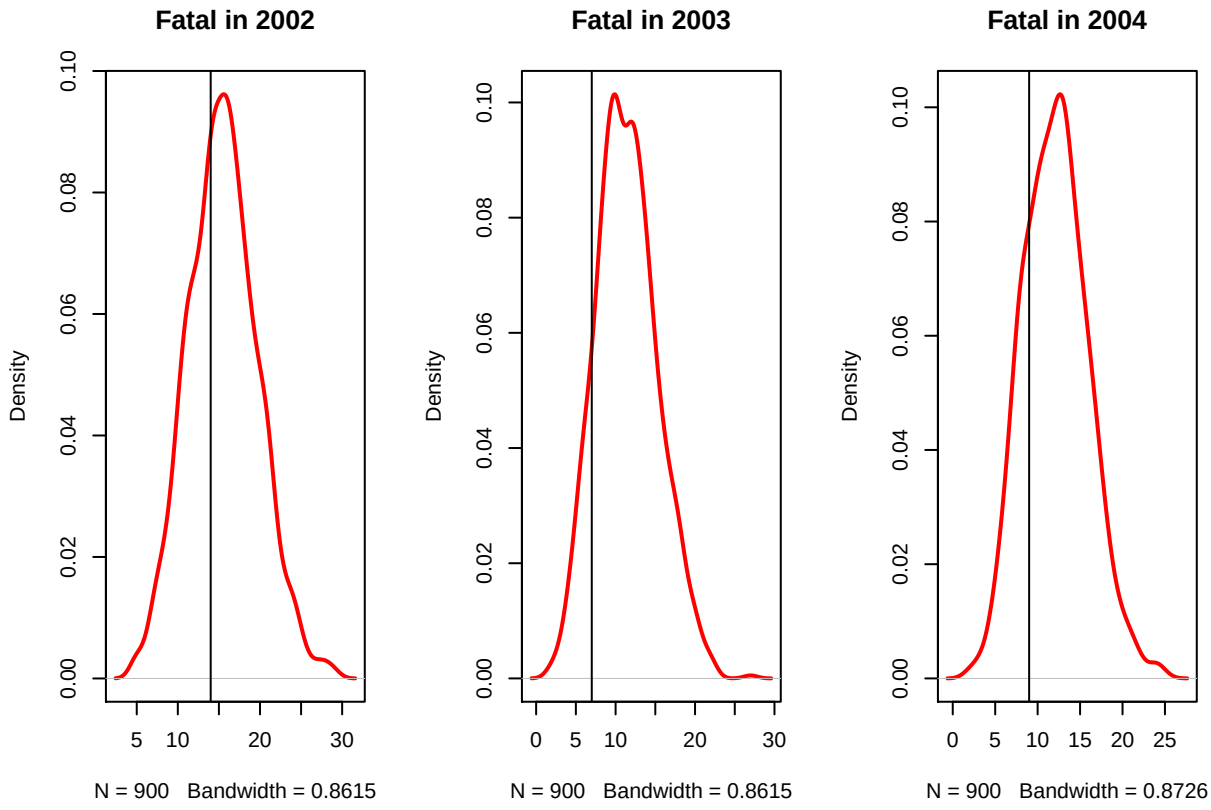


Table of true values

Year	Fatal accidents	Passenger miles	Accident rate
2002	14	19.775	0.7080
2003	7	23.300	0.3004
2004	9	20.300	0.4433

Model comparison throughout DIC index

A good way to select the model that fits best our data, is using the DIC measure. The DIC (deviance information criterion) is a Bayesian criterion for model comparison, defined as:

$$DIC = D(\bar{\theta}) + 2_{p_D}$$

Where: $D(\theta) = -2logL(\theta)$, $p_D = \bar{D} - D(\bar{\theta})$, is the effective number of paramenters of the model, the larger p_D the easier the model fits on the data. $\bar{D} = E[D(\theta)]$, its the mean deviance. is a measure of how well the model fits the data, the larger the worse the fit. $D(\bar{\theta}) = D[E(\theta)]$

the basic idea is that the models with smaller DIC should be preferred. Models are penalized both by the value of $\bar{D}$, which favors a good fit, but also by the effective number of parameters p_D. Since $\bar{D}$ will decrease as the number of parameters in a model increases, the p_D term compensates for this effect by favoring models with a smaller number of parameters.

```
#Model Comparison
DICmodel1 = dic.samples(a1.mod, n.iter=10000, thin=10, type="pD")
DICmodel3 = dic.samples(a3.mod, n.iter=10000, thin =10, type="pD")
DICmodel4 = dic.samples(a4.mod, n.iter=10000, thin=10, type="pD")
DICmodel5 = dic.samples(a5.mod, n.iter=10000, thin =10, type="pD")
DICmodel1
```

```
## Mean deviance:  156.2
## penalty 0.9657
## Penalized deviance: 157.2
```

```
DICmodel3
```

```
## Mean deviance:  133.7
## penalty 2.83
## Penalized deviance: 136.6
```

```
DICmodel4
```

```
## Mean deviance:  132.9
## penalty 1.918
## Penalized deviance: 134.8
```

```
DICmodel5
```

```
## Mean deviance:  154.8
## penalty 2.303
## Penalized deviance: 157.1
```