Density estimate of apparent magnitude according to three different infrared wavelight filter

Davide Manfredini

25/07/2019

Packages

Here there is a list of the packages used:

- **R2Jags**
- **dplyr**
- **gsubfn**
- **gridExtra**
- **grid**
- **gtable**
- **knitr**
- **kableExtra**
- **MASS**
- **coda**
- **lattice**
- **mcmcplots**
- **ggmcmc**
- **e1071**

Data Illustration and download

Data was taken from Kaggle and they were collected during Gaia mission. Gaia is a mission of the European Space Agency (ESA) that aims to accurately measure the position, distance and magnitude of over a billion stars. RAVE is a radial velocity dataset. RAVE also provides spectrophotometric parallax data, as well as cross-identification of stars with a number of other datasets, including Gaia DR2.

The dataset present an huge number of features (36 columns), by my analysis is focused only on three particular features:

- **r_jmag_2mass**
- **r_hmag_2mass**
- **r_kmag_2mass**

These feature regards the *apparent magnitude*, that is the measure of the light flow that we receive from a star, that is different from the quantity of light that the star emits. The bigger this value, the smaller will be the brightness of the star appears.

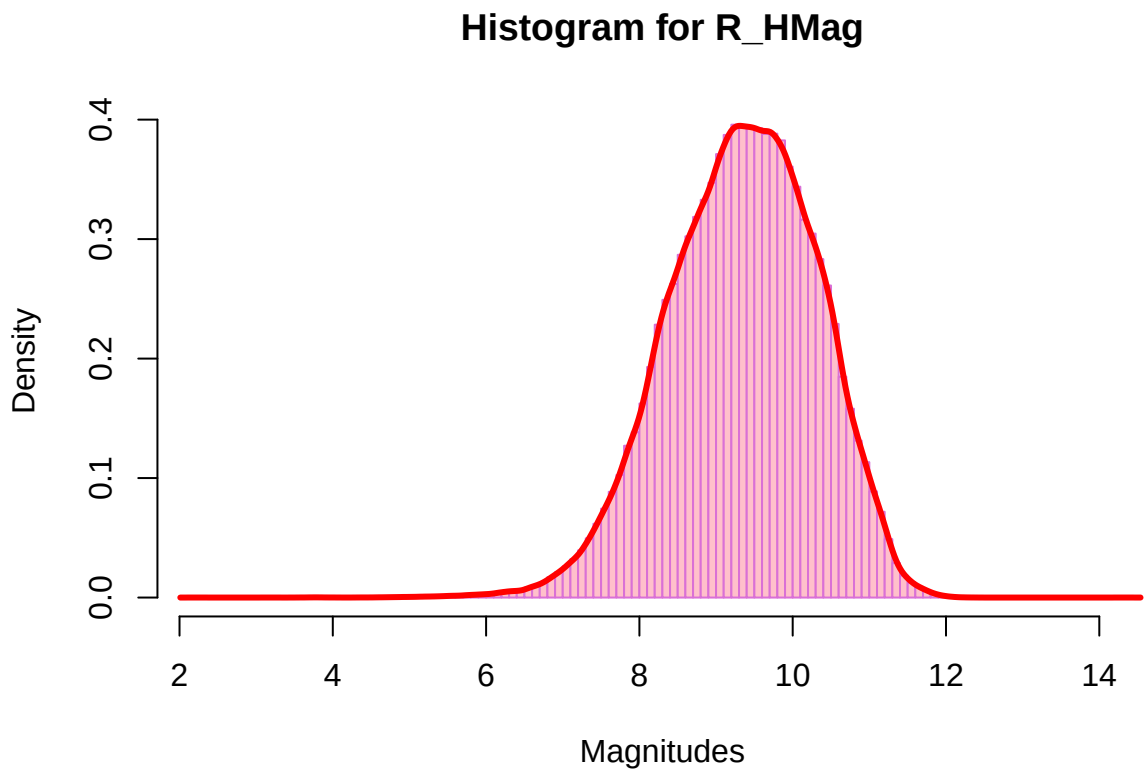
This magnitude were relevated during the mission Gaia, using three different filter, that is the infrared wavelength for which you observe light.

The porpouse of this analysis is to discover if the distribution of data, changes according to the filter used.

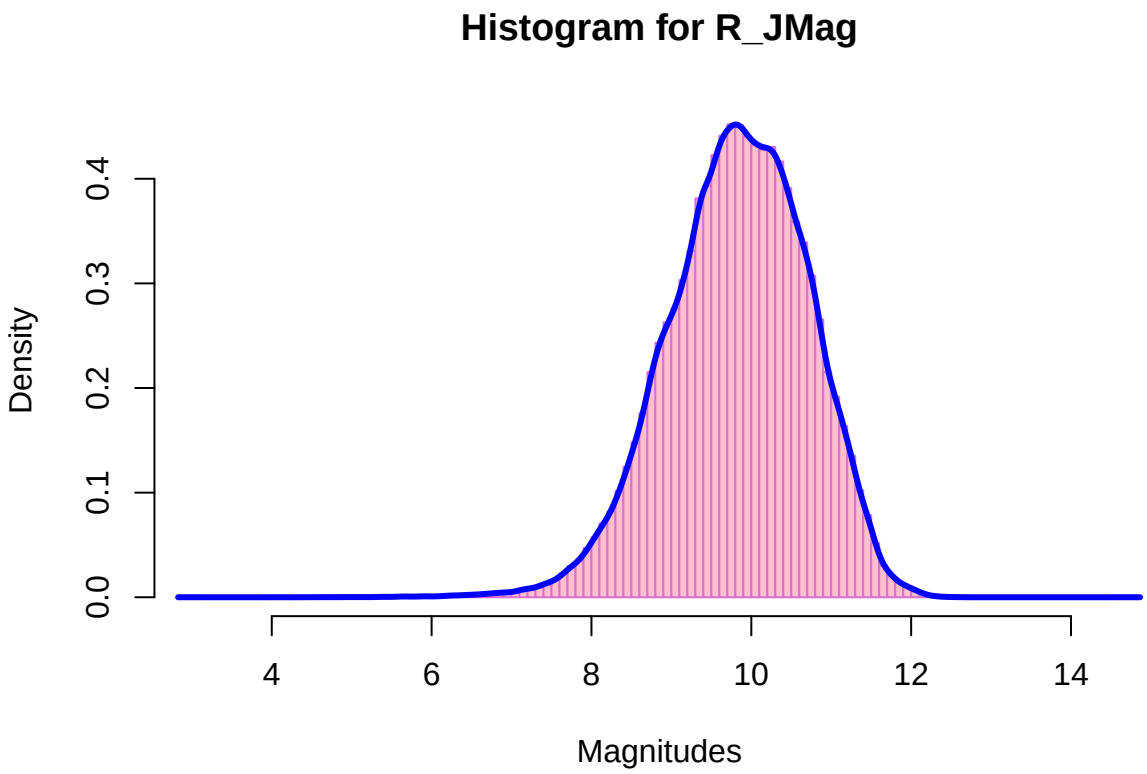
```r
#Illustration of data and upload

data<-read.csv('gaia-dr2-rave-35.csv')


M<-as.data.frame(cbind(data$source_id,data$r_hmag_2mass,data$r_jmag_2mass,data$r_kmag_2mass))
colnames(M)<-c('star_id','R_HMag','R_JMag','R_KMag')
str(M)
```

```
## 'data.frame':    253040 obs. of  4 variables:
##  $ star_id: num  2.42e+18 5.30e+18 2.60e+18 3.18e+18 5.21e+18 ...
##  $ R_HMag : num  9.82 9.43 9.14 9.82 7.08 ...
##  $ R_JMag : num  10.34 10.1 9.45 10.32 7.91 ...
##  $ R_KMag : num  9.71 9.25 9.08 9.73 6.88 ...
```

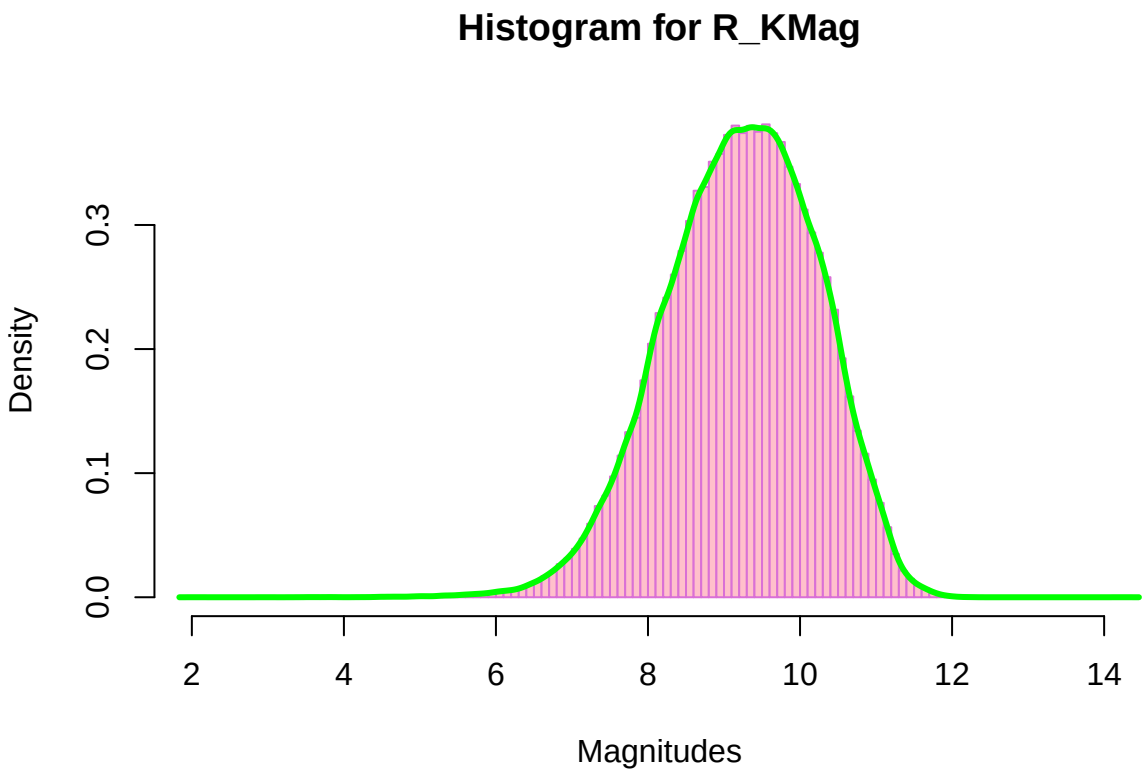
```r
hist(M$R_HMag,prob=TRUE,breaks=100,main = 'Histogram for R_HMag',xlab = 'Magnitudes',col='pink',border =
lines(density(M$R_HMag), lwd=3, col='red')
```




```r
hist(M$R_JMag,prob=TRUE,breaks=100,main = 'Histogram for R_JMag',xlab = 'Magnitudes',col='pink',border =
lines(density(M$R_JMag), lwd=3, col='blue')
```




```
hist(M$R_KMag,prob=TRUE,breaks=100,main = 'Histogram for R_KMag',xlab = 'Magnitudes',col='pink',border =
lines(density(M$R_KMag), lwd=3, col='green')
```



We can see that, more or less, all distributions have the same shape of a **Gaussian distribution** with mean different from 0. We can also observe that the distributions are not perfectly symmetrical, in fact we can see that there is a slight discrepancy between the tails. This suggests a **Gamma** distribution. So the model proposed are two:

Before start our analysis using MCMC simulation, let's do a simple analysis on the data. The first thing we do is check if there is linear correlation among variables.

```
cor(M)
```

```
##             star_id      R_HMag      R_JMag      R_KMag
## star_id  1.00000000 -0.07566649 -0.05028371 -0.08045917
## R_HMag  -0.07566649  1.00000000  0.98632289  0.99884796
## R_JMag  -0.05028371  0.98632289  1.00000000  0.97995491
## R_KMag  -0.08045917  0.99884796  0.97995491  1.00000000
```

They are high correlated variable, because both are data regarding the same star, but the data are collected using a different filter. The next thing is look at *skewness* and *kurtosis*:

```
k_s <- data.frame(skewness=double(),
                  kurtosis=double(),
                  stringsAsFactors = FALSE)
k_s<-rbind(k_s,c(skewness(M$R_HMag),kurtosis(M$R_HMag)))
k_s<-rbind(k_s,c(skewness(M$R_JMag),kurtosis(M$R_JMag)))
k_s<-rbind(k_s,c(skewness(M$R_KMag),kurtosis(M$R_KMag)))
colnames(k_s)<-c('skewness','kurtosis')
```

```
rownames(k_s)<-c('R_HMag','R_JMag','R_KMag')

kable(k_s) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	skewness	kurtosis
R_HMag	-0.3199357	0.1176763
R_JMag	-0.3704677	0.3634866
R_KMag	-0.3227879	0.0833165

These are the value for the skewness and kurtosis for the data taken with a different filter. We can see that the skewness belongs to $[-0.5, 0.5]$, this means that the data are fairly symmetrical. Indeed regarding the kurtosis and rembring that it a measure of the "tailedness" of the probability distribution of a real-valued random variable. The kurtosis decreases as the tails become lighter. It increases as the tails become heavier. In our case for J-Magnitude the kurtosis is slightly higher than the other two cases.

At this point, these are the theoretical models:

Normal Model

$$
\begin{aligned}
&Y|\mu, \sigma^2 \sim \mathcal{N}(\mu, \sigma^2) \\
&\mu \sim \mathcal{N}(0, 1) \\
&\sigma^2 \sim Gamma(0.001, 0.001)
\end{aligned}
$$

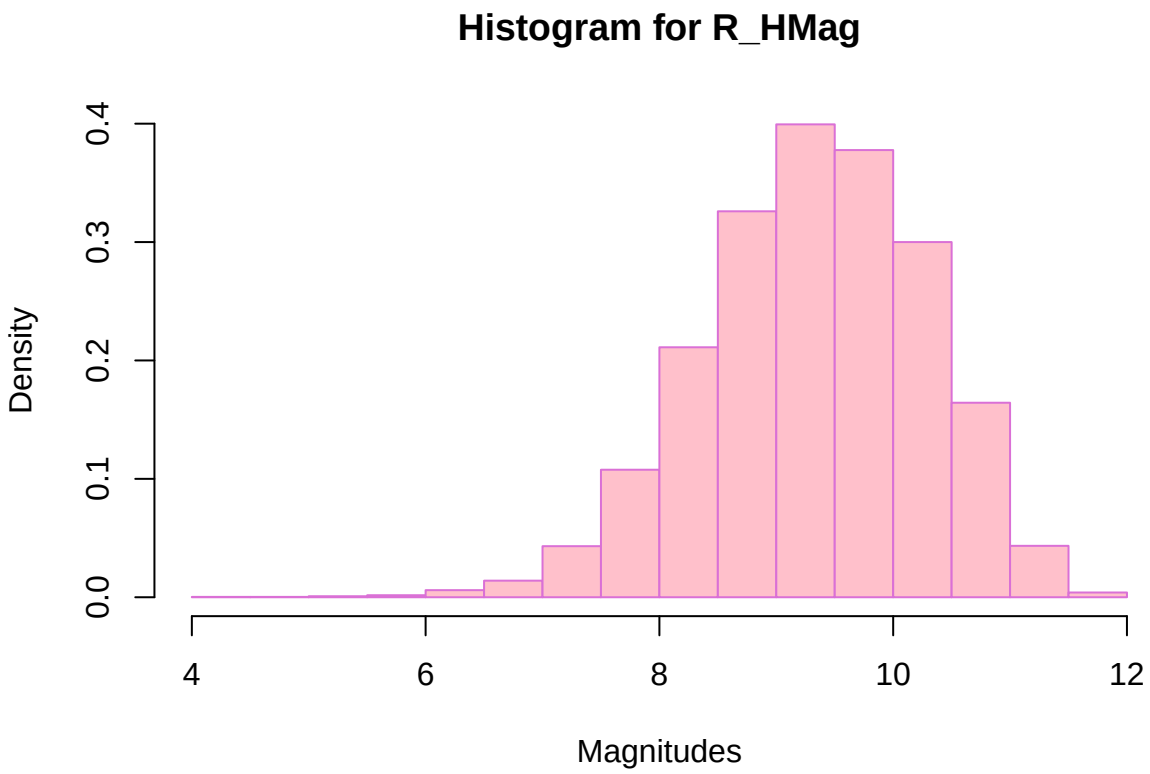
Gamma Model

$$
\begin{aligned}
&Y|k, \theta \sim Gamma(k, \theta) \\
&k \sim Gamma(0.01, 0.01) \\
&\theta \sim Gamma(0.01, 0.01)
\end{aligned}
$$

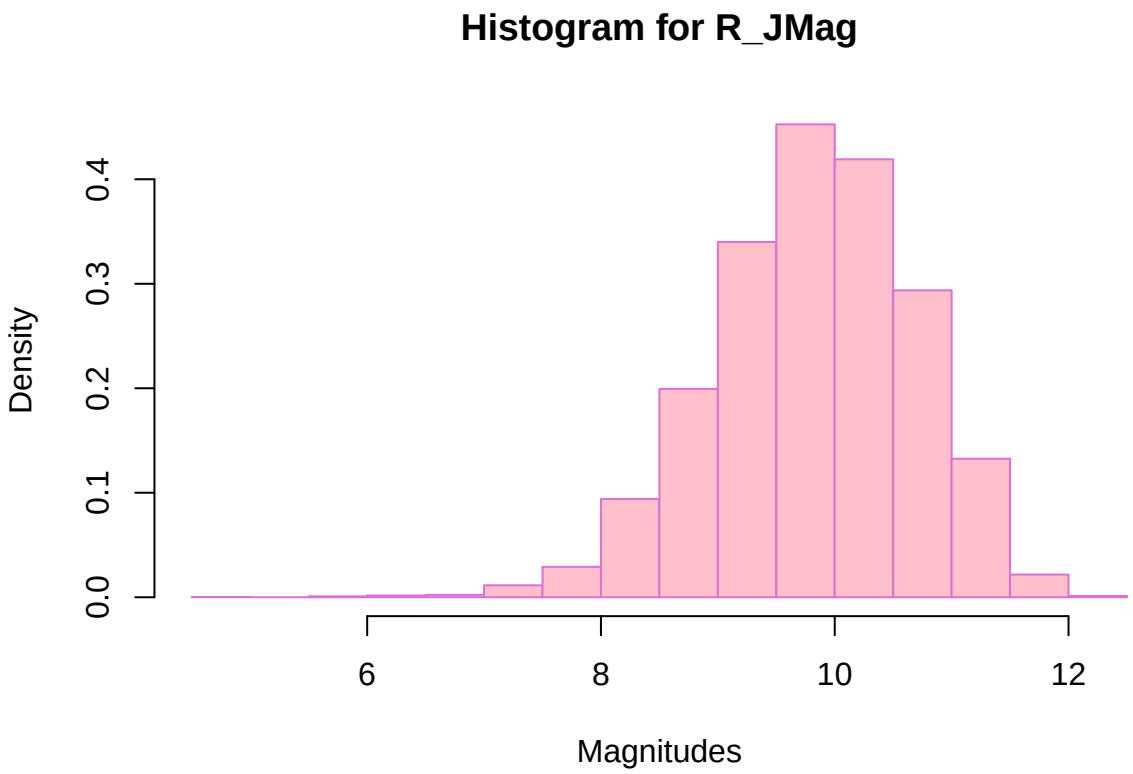
Data reduction

The dataset is huge, and then the estimation of parameter using MCMC it was so slow (it requires more than one hour and half for a single column), so in order to slve this problem, my analysis is made on a smaller sample of the real dataset, that has the same distribution.

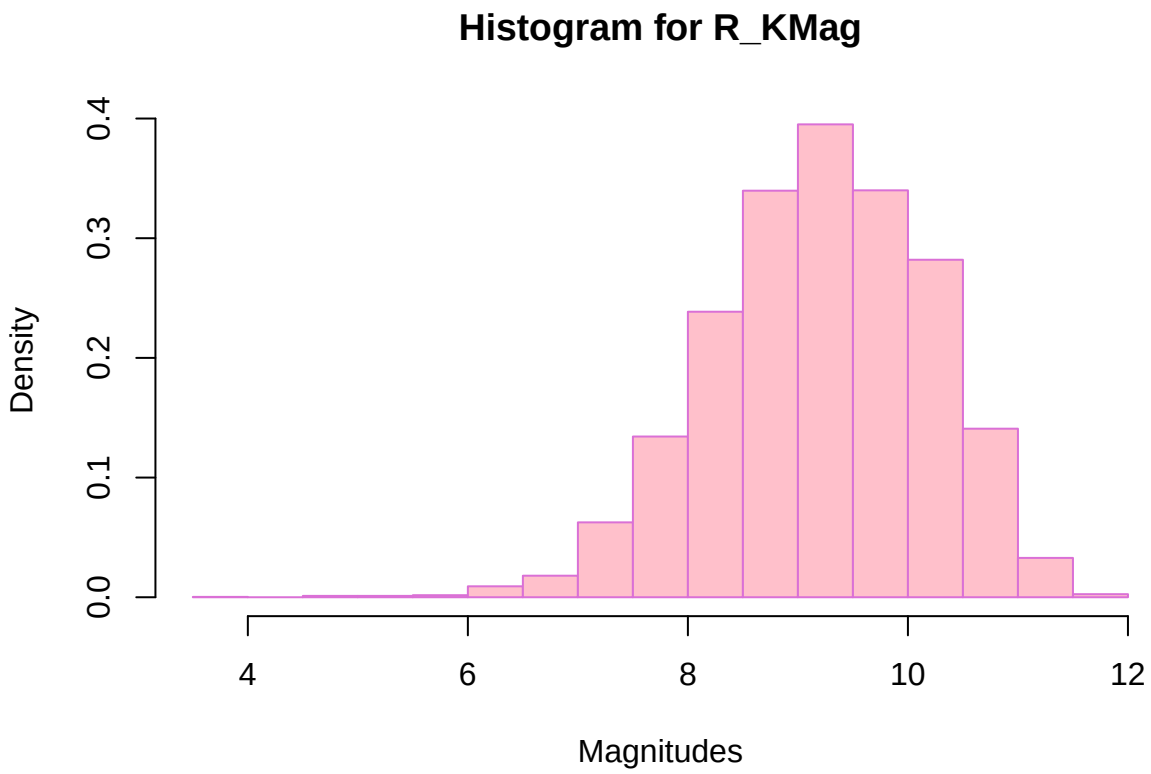
```
Small_M<-sample_n(M, 7000)
hist(Small_M$R_HMag, prob=TRUE,main = 'Histogram for R_HMag',xlab = 'Magnitudes',col='pink',border = 'o
```




```
hist(Small_M$R_JMag, prob=TRUE,main = 'Histogram for R_JMag',xlab = 'Magnitudes',col='pink',border = 'o
```




```
hist(Small_M$R_KMag, prob=TRUE,main = 'Histogram for R_KMag',xlab = 'Magnitudes',col='pink',border = 'o
```




```
k_s_small <- data.frame(skewness=double(),
                  kurtosis=double(),
                  stringsAsFactors = FALSE)
k_s_small<-rbind(k_s_small,c(skewness(Small_M$R_HMag),kurtosis(Small_M$R_HMag)))
k_s_small<-rbind(k_s_small,c(skewness(Small_M$R_JMag),kurtosis(Small_M$R_JMag)))
k_s_small<-rbind(k_s_small,c(skewness(Small_M$R_KMag),kurtosis(Small_M$R_KMag)))

colnames(k_s_small)<-c('skewness','kurtosis')
rownames(k_s_small)<-c('R_HMag','R_JMag','R_KMag')

kable(k_s_small) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	skewness	kurtosis
R_HMag	-0.3586651	0.1660989
R_JMag	-0.4088409	0.3531763
R_KMag	-0.3658836	0.1574082

Bugs Models

These are the models defined before, written using BUGS

```
modelnormal <- function() {
  for (i in 1:N){
    y[i] ~ dnorm(mu, tau)
  }
  mu ~ dnorm(mu0, sigma0)
  sigma ~ dgamma(v1, v2)
  tau <- pow(sigma,  -2)
}
```

```
modelgamma<-function()
   {
    for (i in 1:N) {
      y[i]~dgamma(k,theta)
    }
    k~dgamma(0.01,0.01)
    theta~dgamma(0.01,0.01)
  }
```

Functions

The following functions, make the analysis for three columns using the two different models

```
normal_fit<-function(values)
{
  data_for_fitting <- list(N = length(values),
                           mu0 = 0,
                           sigma0 = 1,
                           v1 = 0.001,
                           v2 = 0.001,
                           y=values)
  #Choose the number of chain
  fit<-jags(model=modelnormal,
            data = data_for_fitting,
            n.iter = 10000,param=c('mu','tau'),
            n.thin = 10,
            DIC=TRUE,
            n.chains = 4)

  DIC<-fit$BUGSoutput$DIC
  pD <-fit$BUGSoutput$pD

  mc.fit<-as.mcmc(fit)
  results<-mc.fit

  print(summary(results))

  mu=summary(results)$statistics[2,1]
  tau=summary(results)$statistics[3,1]

  return(list('mu'=mu,'tau'=tau,'DIC'=DIC,'pD'=pD,'fit'=fit))
```

```
}


gamma_fit<-function(values)
{
  data_for_fitting<-list(N = length(values), y=values)

  fit<-jags(model=modelgamma,
            data = data_for_fitting,
            n.iter = 10000,
            param=c('k','theta'),
            n.thin = 10,
            n.chains = 4,
            DIC=TRUE)

  DIC<-fit$BUGSoutput$DIC
  pD <-fit$BUGSoutput$pD

  mc.fit<-as.mcmc(fit)
  results<-mc.fit

  print(summary(results))

  k=summary(results)$statistics[2,1]
  theta=summary(results)$statistics[3,1]

  return(list('k'=k,'theta'=theta,'DIC'=DIC,'pD'=pD,'fit'=fit))
}
```

Fitting the models

The next part regards the initialization of some parameter in order to have a nice view of the obtained data and then the different analysis will start, showing the different traceplot and autocorrelation plot, in order to show if there is or not covergence to the value to estimate and if there is, how fast it is.

Will be plotted the posterior density for each estimated parameter

Moreover will be shown the *HPD Intervals*. An $1-\alpha$ confidence interval is an interval C_n for which, considered a posterior distribution $\pi(\theta|\underline{x_n})$

$$\int_{C_n} \pi(\theta|\underline{x_n})d\theta = 1-\alpha$$

Any point within the interval has a higher density than any other point outside. Thus, HPD interval is the collection of most likely values of the parameters. The most common value for *alpha* are .90,.95,.99

Initialization of some variables

```
normal_parameters<-data.frame(mu=double(),
                              tau=double(),
                              DIC=double(),
```

```
                              pD=double(),
                              stringsAsFactors=FALSE)

gamma_parameters<-data.frame( k=double(),
                              theta=double(),
                              DIC=double(),
                              pD=double(),
                              stringsAsFactors=FALSE)
likelihood_ratios<-rep(NA,3)
idx=1

set.seed(1234)
```

Analysis for R_HMag

Using Normal Model

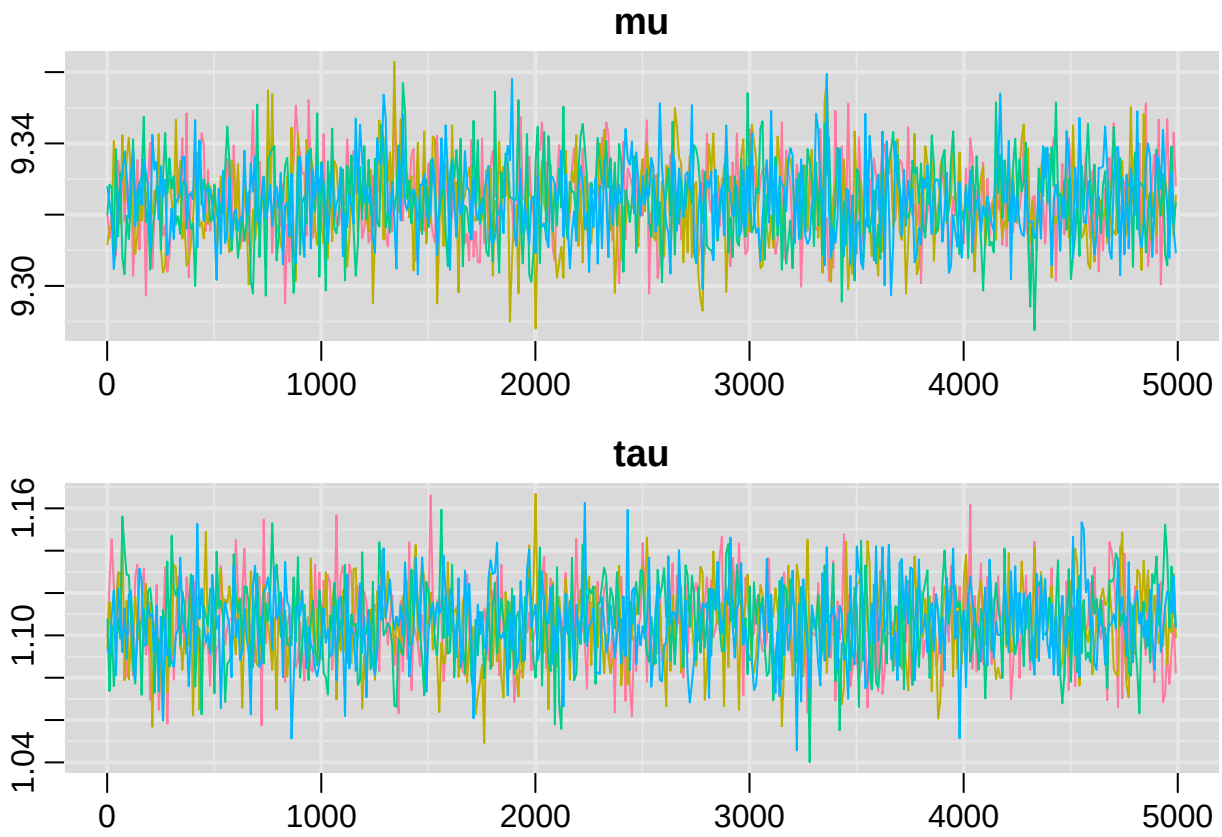
As said before, the first analysis will be taken using the normal model

```
Y=as.vector(Small_M$R_HMag)
normal_temp<-normal_fit(Y)
```

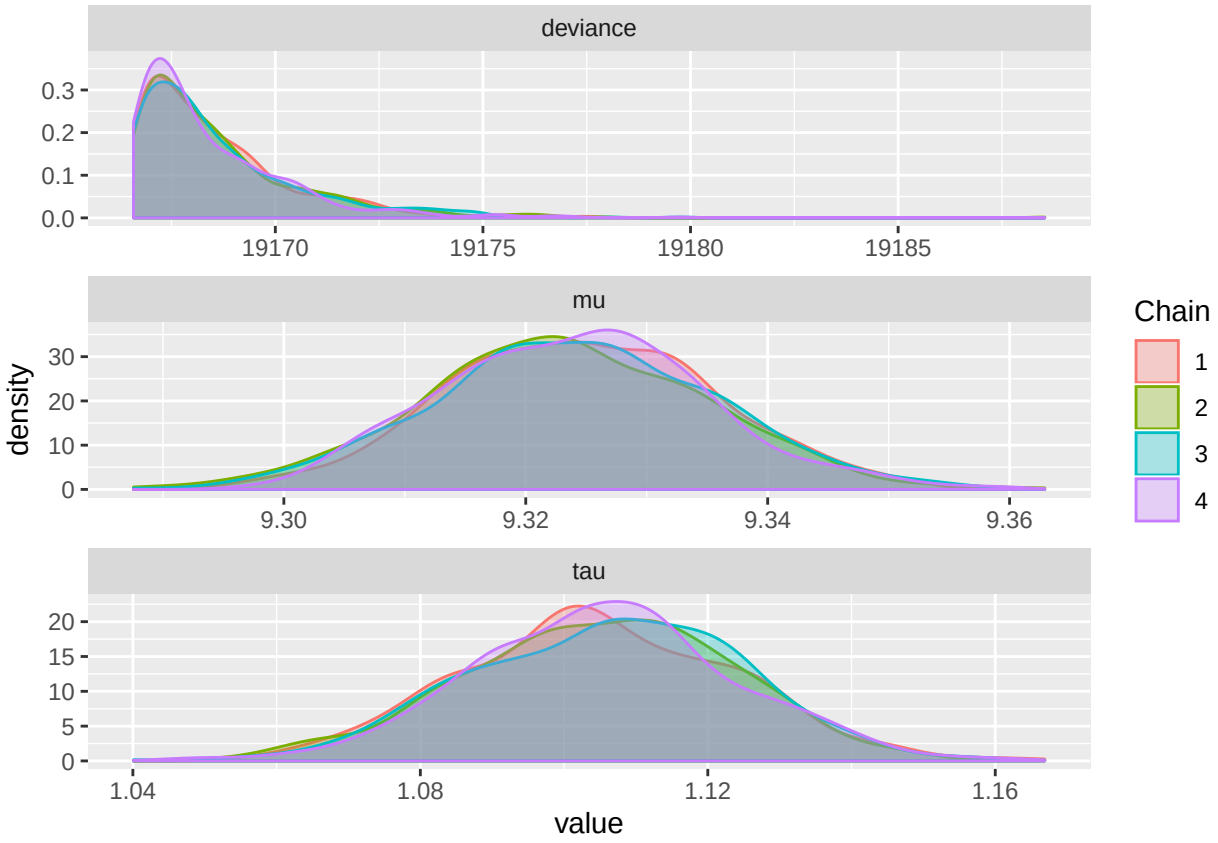
```
## module glm loaded

## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7000
##    Unobserved stochastic nodes: 2
##    Total graph size: 7010
##
## Initializing model
##
##
## Iterations = 1:4991
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 500
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean      SD  Naive SE Time-series SE
## deviance 19168.553 1.94289 0.0434442      0.0434289
## mu           9.324 0.01119 0.0002502      0.0002550
## tau          1.105 0.01855 0.0004148      0.0004424
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance 19166.629 19167.179 19167.963 19169.297 19173.647
```

```
## mu          9.303     9.316     9.324     9.332     9.346
## tau         1.069     1.093     1.105     1.118     1.141
```

```
f<-normal_temp$fit
par(mar = c(2, 0.5, 4, 2), oma = c(4, 4, 0.2, 0.2),mfrow=c(3,1))
traplot(as.mcmc(f), parms = c('mu','tau'))
```

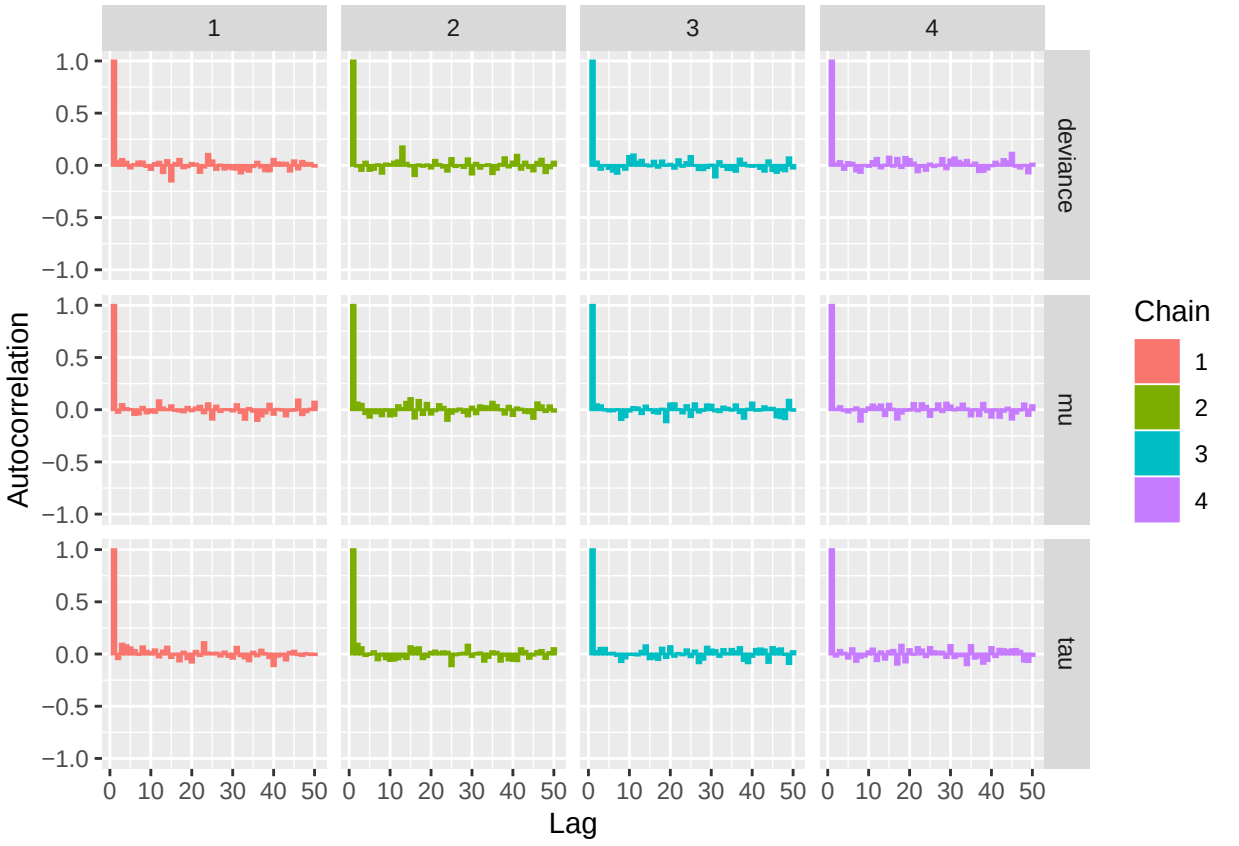



```
f.gg<-ggs(as.mcmc(f))
ggs_density(f.gg)
```



And finally let's look to the autocorrelation

```
ggs_autocorrelation(f.gg)
```



In this case, we can see from the traceplots that we reached the convergence for both paramenters, indeed we can see that the chains for both paramenters, oscillate around a specific value. Then they are stabilized theirself n their stable distribution. We can also see that autocorrelation go to zero for both parameter, and this is another check for the convergence.

The last thing is looking to the High Posterior Density Intervals.

```
cat('----------- 90% -----------')
```

```
## ----------- 90% -----------
```

```
HPDinterval(as.mcmc(f),prob=.90)
```

```
## [[1]]
##                   lower        upper
## deviance 19166.587007 19171.032377
## mu           9.308430     9.342977
## tau          1.074019     1.133472
## attr(,"Probability")
## [1] 0.9
##
## [[2]]
##                   lower        upper
## deviance 19166.599983 19171.437666
## mu           9.304931     9.342829
## tau          1.075710     1.133901
```

```
## attr(,"Probability")
## [1] 0.9
##
## [[3]]
##                  lower        upper
## deviance 19166.588837 19171.405880
## mu           9.304522     9.342259
## tau          1.074436     1.133473
## attr(,"Probability")
## [1] 0.9
##
## [[4]]
##                  lower        upper
## deviance 19166.584302 19170.709589
## mu           9.304743     9.339054
## tau          1.078270     1.136871
## attr(,"Probability")
## [1] 0.9
```

```
cat('----------- 95% -----------')
```

```
## ----------- 95% -----------
```

```
HPDinterval(as.mcmc(f),prob=.95)
```

```
## [[1]]
##                  lower        upper
## deviance 19166.587007 19172.067764
## mu           9.304995     9.346845
## tau          1.068361     1.140989
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##                  lower        upper
## deviance 19166.591617 19172.680711
## mu           9.301146     9.345327
## tau          1.064789     1.136401
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##                  lower        upper
## deviance 19166.588837 19173.029245
## mu           9.301018     9.345513
## tau          1.071695     1.141563
## attr(,"Probability")
## [1] 0.95
##
## [[4]]
##                  lower       upper
## deviance 19166.584302 19172.21508
## mu           9.302951     9.34410
```

```
## tau          1.072871     1.14261
## attr(,"Probability")
## [1] 0.95
```

```
cat('----------- 99% -----------')
```

```
## ----------- 99% -----------
```

```
HPDinterval(as.mcmc(f),prob=.99)
```

```
## [[1]]
##                  lower        upper
## deviance 19166.587007 19174.572873
## mu           9.299781     9.352187
## tau          1.057380     1.147709
## attr(,"Probability")
## [1] 0.99
##
## [[2]]
##                  lower        upper
## deviance 19166.591617 19176.111234
## mu           9.292998     9.354893
## tau          1.059864     1.151638
## attr(,"Probability")
## [1] 0.99
##
## [[3]]
##                  lower        upper
## deviance 19166.588837 19174.895775
## mu           9.297258     9.354565
## tau          1.062704     1.159386
## attr(,"Probability")
## [1] 0.99
##
## [[4]]
##                  lower        upper
## deviance 19166.584302 19175.631756
## mu           9.300177     9.353981
## tau          1.059756     1.159263
## attr(,"Probability")
## [1] 0.99
```

Then we store the approximated parameters

```
mu<-normal_temp$mu
tau<-normal_temp$tau

normal_parameters<-rbind(normal_parameters,normal_temp[-length(normal_temp)])
```

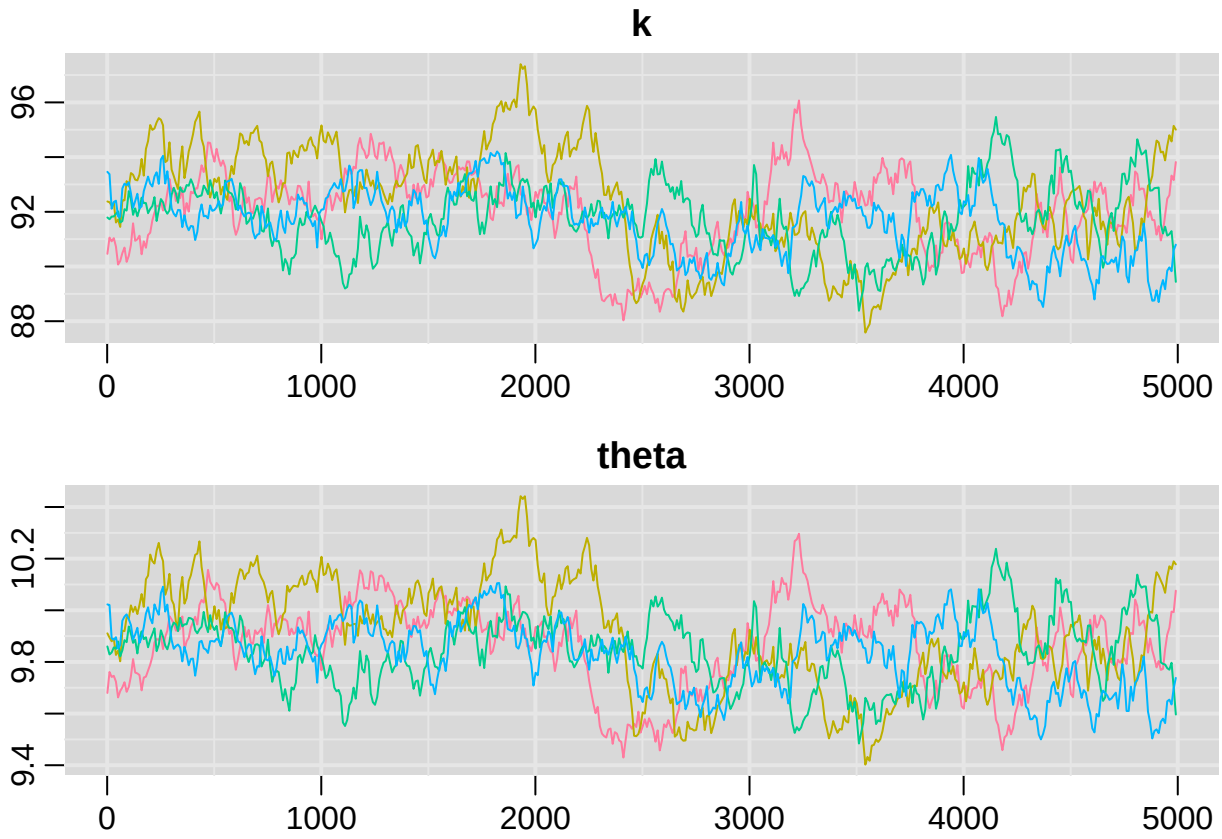
Using Gamma Model

As done before first we take a look at traceplots

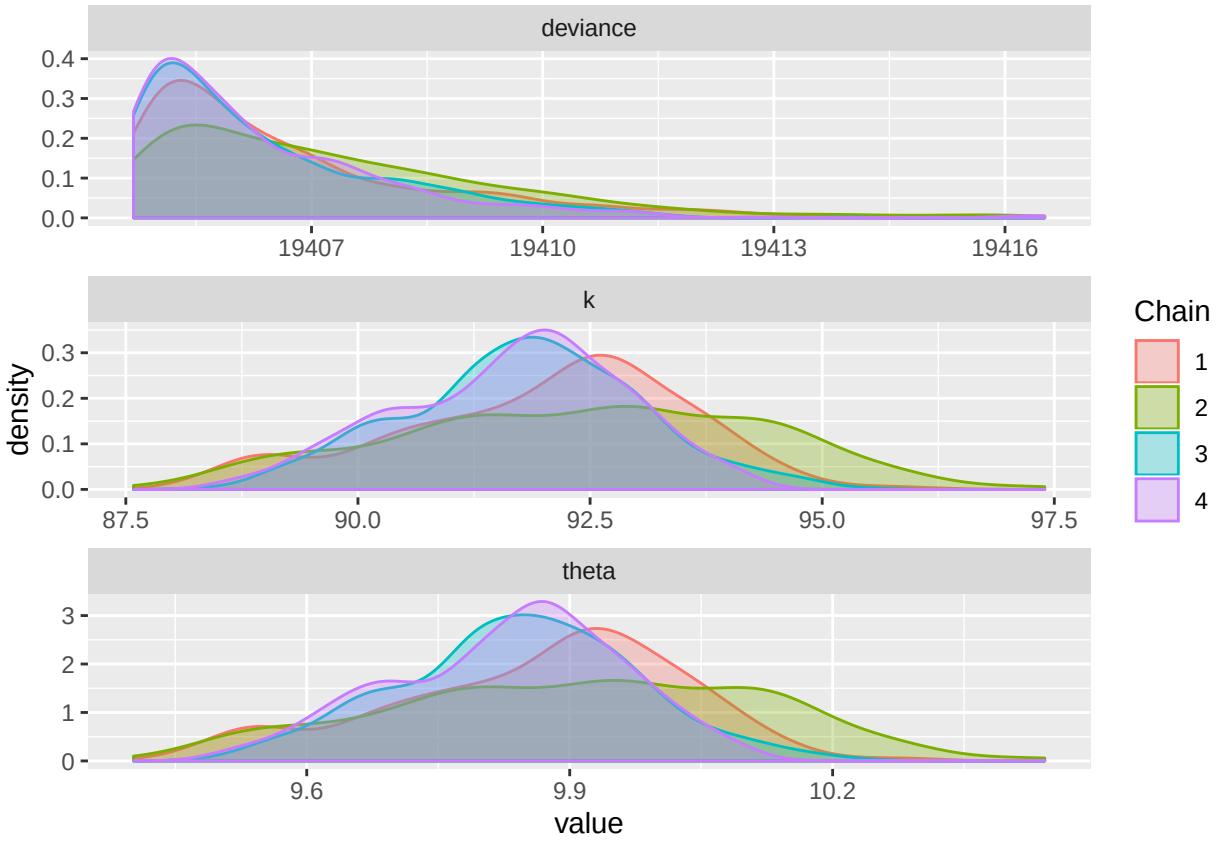
```r
gamma_temp<-gamma_fit(Y)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7000
##    Unobserved stochastic nodes: 2
##    Total graph size: 7004
##
## Initializing model
##
##
## Iterations = 1:4991
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 500
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean     SD Naive SE Time-series SE
## deviance 19406.680 1.9179 0.042886         0.1201
## k           91.956 1.5409 0.034455         0.2338
## theta        9.861 0.1654 0.003699         0.0251
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%    97.5%
## deviance 19404.738 19405.259 19406.061 19407.563 19411.43
## k           88.883    90.949    92.013    92.939    94.93
## theta        9.532     9.752     9.866     9.967    10.18
```

```r
f1<-gamma_temp$fit
par(mar = c(2, 0.5, 4, 2), oma = c(4, 4, 0.2, 0.2),mfrow=c(3,1))
traplot(as.mcmc(f1), parms = c('k','theta'))
```

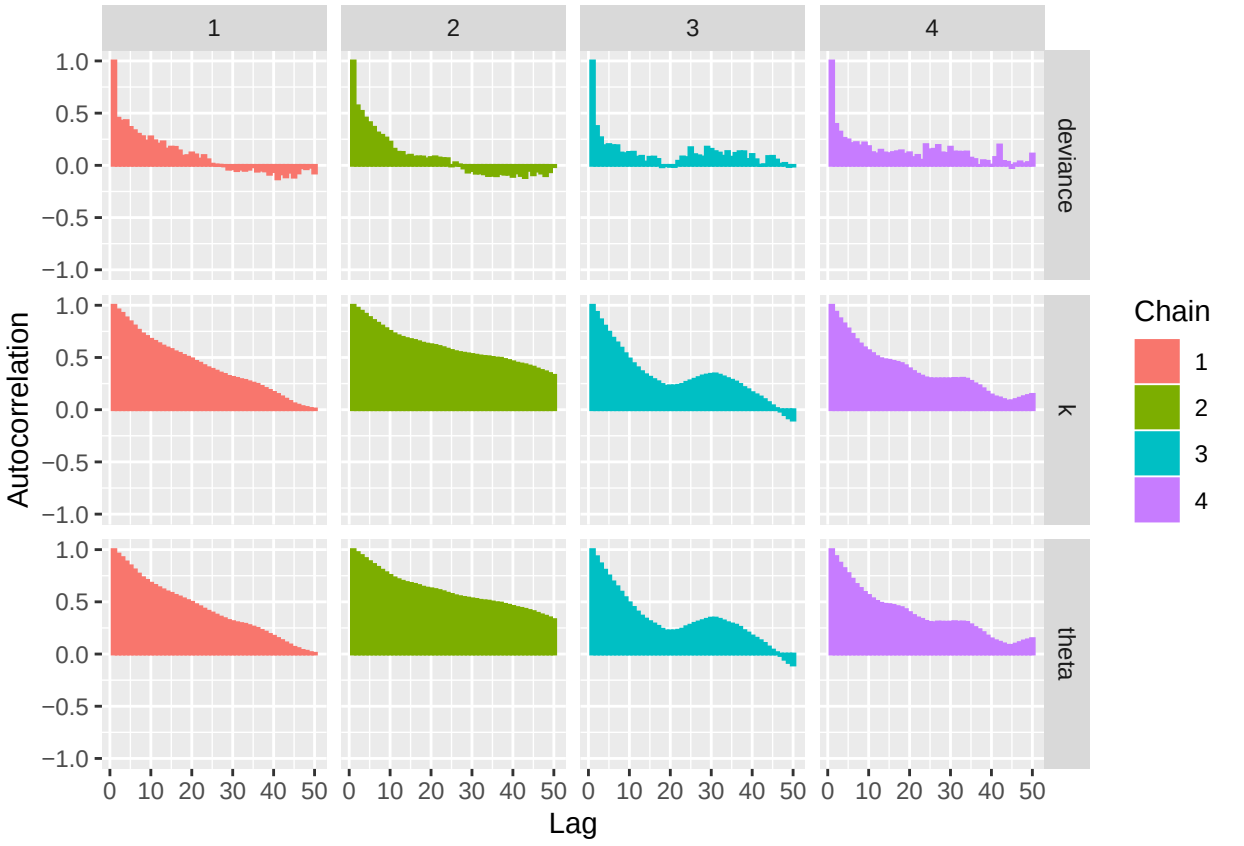



```
f.gg1<-ggs(as.mcmc(f1))
ggs_density(f.gg1)
```



And finally let's look to the autocorrelation

```
ggs_autocorrelation(f.gg1)
```



In this case, we can see from the traceplots that the behaviour of the chains is not the behaviour of a chain that reached the convergence, because we can see that there is not a fluctuation around a value. But this doesn't mean that there is not convergence, maybe this model needs more iteration in order to reach the convergence. We can see this thing from the autocorrelation plot, indeed the plots goes to 0 in a very slowly.

These are the HPD

```
cat('----------- 90% -----------')
```

```
## ----------- 90% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.90)
```

```
## [[1]]
##                   lower       upper
## deviance 19404.689636 19409.56869
## k           88.923956    94.00216
## theta        9.541212    10.08761
## attr(,"Probability")
## [1] 0.9
##
## [[2]]
##                   lower       upper
## deviance 19404.696314 19410.26278
## k           88.868083    95.17176
## theta        9.514094    10.19035
```

```
## attr(,"Probability")
## [1] 0.9
##
## [[3]]
##                lower       upper
## deviance 19404.69266 19408.81645
## k           89.70100    93.81126
## theta        9.61426    10.05387
## attr(,"Probability")
## [1] 0.9
##
## [[4]]
##                 lower       upper
## deviance 19404.690762 19408.43733
## k           89.641351    93.52140
## theta        9.616687    10.02975
## attr(,"Probability")
## [1] 0.9
```

```
cat('----------- 95% -----------')
```

```
## ----------- 95% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.95)
```

```
## [[1]]
##                 lower       upper
## deviance 19404.689636 19410.79524
## k           88.825949    94.54562
## theta        9.490523    10.10599
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##                lower       upper
## deviance 19404.69631 19411.72406
## k           88.74737    95.87362
## theta        9.51835    10.28088
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##                 lower       upper
## deviance 19404.692659 19409.66850
## k           89.105134    93.98915
## theta        9.563396    10.09275
## attr(,"Probability")
## [1] 0.95
##
## [[4]]
##                 lower       upper
## deviance 19404.690762 19409.71166
## k           89.349036    93.91326
```

```
## theta        9.585876    10.07487
## attr(,"Probability")
## [1] 0.95
```

```
cat('----------- 99% -----------')
```

```
## ----------- 99% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.99)
```

```
## [[1]]
##                  lower       upper
## deviance 19404.689636 19412.84486
## k           88.031616    95.01552
## theta        9.429717    10.17811
## attr(,"Probability")
## [1] 0.99
##
## [[2]]
##                  lower       upper
## deviance 19404.696314 19415.46653
## k           87.751621    96.67848
## theta        9.402366    10.36219
## attr(,"Probability")
## [1] 0.99
##
## [[3]]
##                  lower       upper
## deviance 19404.692659 19411.17907
## k           88.921385    94.85310
## theta        9.523866    10.18071
## attr(,"Probability")
## [1] 0.99
##
## [[4]]
##                  lower       upper
## deviance 19404.690762 19411.42684
## k           88.734331    94.11182
## theta        9.518556    10.09498
## attr(,"Probability")
## [1] 0.99
```

Now we store the parameters:

```
gamma_parameters<-rbind(gamma_parameters,gamma_temp[-length(gamma_temp)])

k<-gamma_temp$k
theta<-gamma_temp$theta
likelihood_ratios[1]<-sum(log(dnorm(x=Y,mean=mu,sd=1/tau)))-sum(log(dgamma(x=Y,shape = k,rate = theta)))
```

Analysis for R_JMag

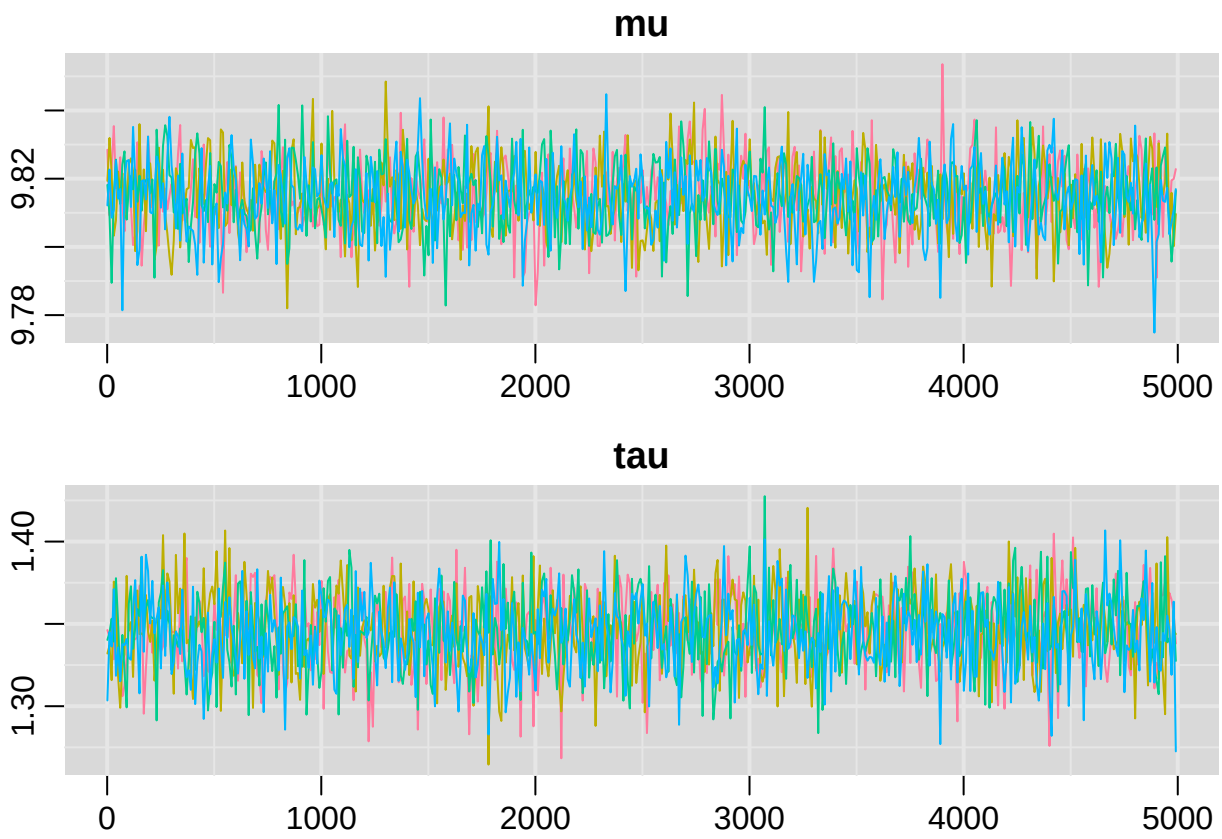
Using Normal Model

As said before, the first analysis will be taken using the normal model

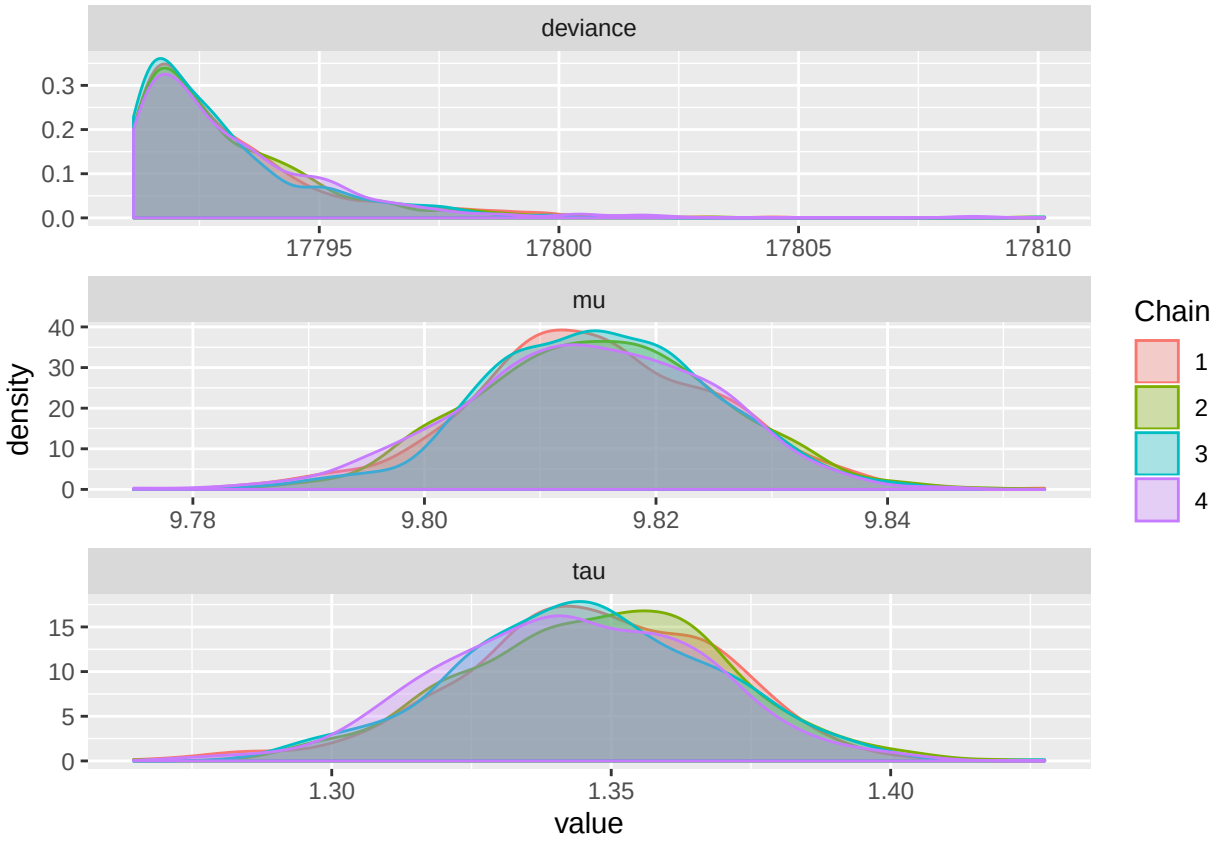
```
Y=as.vector(Small_M$R_JMag)
normal_temp<-normal_fit(Y)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7000
##    Unobserved stochastic nodes: 2
##    Total graph size: 7010
##
## Initializing model
##
##
## Iterations = 1:4991
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 500
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean      SD  Naive SE Time-series SE
## deviance 17793.134 2.06845 0.0462519       0.045632
## mu           9.815 0.01027 0.0002296       0.000232
## tau          1.346 0.02279 0.0005097       0.000507
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance 17791.176 17791.723 17792.506 17793.839 17798.598
## mu           9.794     9.808     9.815     9.822     9.834
## tau          1.299     1.331     1.345     1.362     1.389
```

```
f<-normal_temp$fit
par(mar = c(2, 0.5, 4, 2), oma = c(4, 4, 0.2, 0.2),mfrow=c(3,1))
traplot(as.mcmc(f), parms = c('mu','tau'))
```

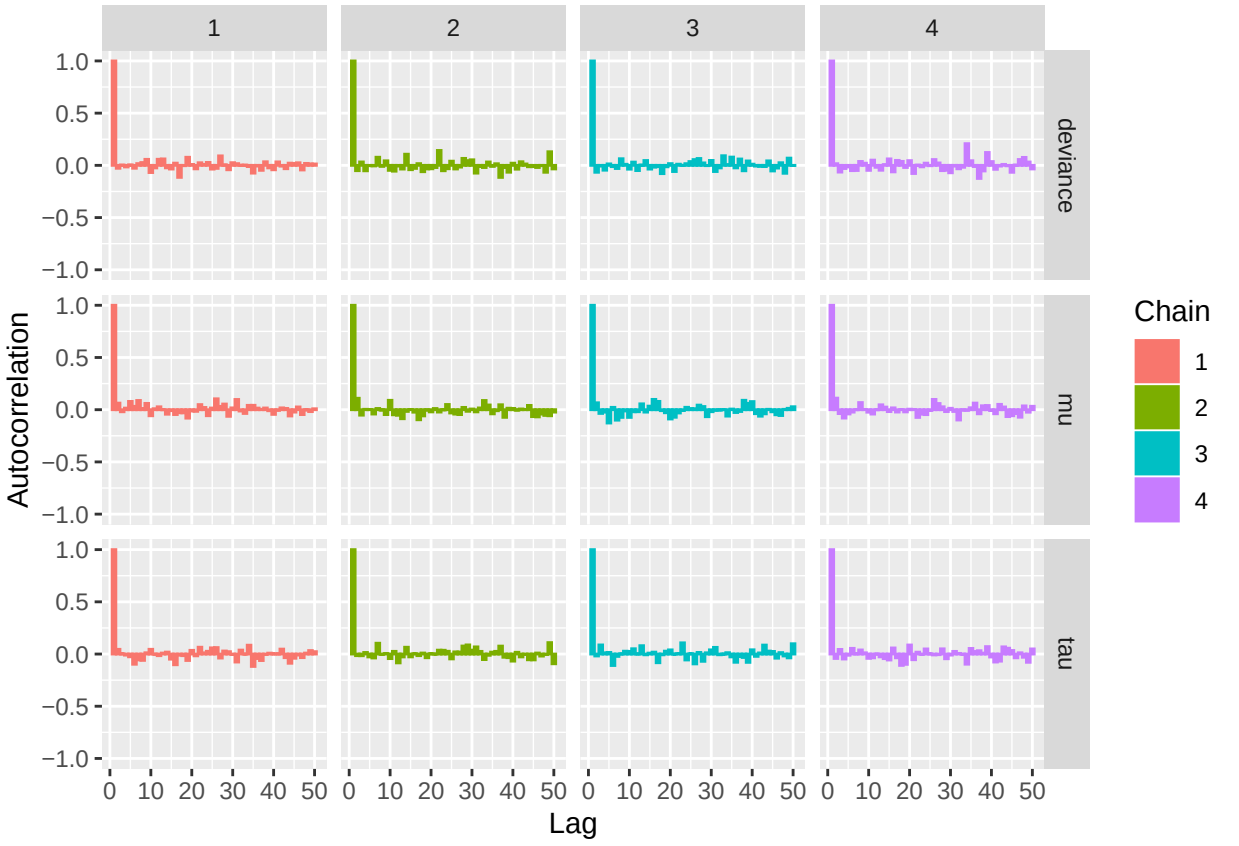



```
f.gg<-ggs(as.mcmc(f))
ggs_density(f.gg)
```



And finally let's look to the autocorrelation

```
ggs_autocorrelation(f.gg)
```



In this case, we can see from the traceplots that we reached the convergence for both paramenters, indeed we can see that the chains for both paramenters, oscillate around a specific value. Then they are stabilized theirself n their stable distribution. We can also see that autocorrelation go to zero for both parameter, and this is another check for the convergence.

Then the HPD:

```
cat('----------- 90% -----------')
```

```
## ----------- 90% -----------
```

```
HPDinterval(as.mcmc(f),prob=.90)
```

```
## [[1]]
##                   lower        upper
## deviance 17791.125025 17796.010542
## mu           9.798041     9.831584
## tau          1.310823     1.381268
## attr(,"Probability")
## [1] 0.9
##
## [[2]]
##                   lower        upper
## deviance 17791.128121 17795.737895
## mu           9.799210     9.832351
## tau          1.312408     1.385763
```

```
## attr(,"Probability")
## [1] 0.9
##
## [[3]]
##                  lower        upper
## deviance 17791.123767 17795.539240
## mu           9.800279     9.830216
## tau          1.307097     1.382200
## attr(,"Probability")
## [1] 0.9
##
## [[4]]
##                  lower        upper
## deviance 17791.124592 17795.624523
## mu           9.798789     9.832281
## tau          1.305694     1.377520
## attr(,"Probability")
## [1] 0.9
```

```
cat('----------- 95% -----------')
```

```
## ----------- 95% -----------
```

```
HPDinterval(as.mcmc(f),prob=.95)
```

```
## [[1]]
##                  lower        upper
## deviance 17791.125025 17797.784697
## mu           9.796959     9.837968
## tau          1.305665     1.391994
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##                  lower        upper
## deviance 17791.128121 17797.356097
## mu           9.796996     9.834453
## tau          1.298951     1.388835
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##                  lower        upper
## deviance 17791.123767 17796.888341
## mu           9.795016     9.832687
## tau          1.299063     1.388742
## attr(,"Probability")
## [1] 0.95
##
## [[4]]
##                  lower        upper
## deviance 17791.124592 17797.053948
## mu           9.793200     9.833053
```

```
## tau          1.299831     1.386219
## attr(,"Probability")
## [1] 0.95
```

```
cat('----------- 99% -----------')
```

```
## ----------- 99% -----------
```

```
HPDinterval(as.mcmc(f),prob=.99)
```

```
## [[1]]
##                  lower        upper
## deviance 17791.125025 17801.404522
## mu           9.786553     9.840475
## tau          1.278656     1.395845
## attr(,"Probability")
## [1] 0.99
##
## [[2]]
##                  lower        upper
## deviance 17791.128121 17801.215478
## mu           9.788242     9.841221
## tau          1.291057     1.404912
## attr(,"Probability")
## [1] 0.99
##
## [[3]]
##                  lower        upper
## deviance 17791.123767 17799.599432
## mu           9.790961     9.841625
## tau          1.291365     1.397081
## attr(,"Probability")
## [1] 0.99
##
## [[4]]
##                  lower        upper
## deviance 17791.124592 17801.664766
## mu           9.785055     9.838144
## tau          1.282831     1.401330
## attr(,"Probability")
## [1] 0.99
```

And we store the parameters

```
mu<-normal_temp$mu
tau<-normal_temp$tau

normal_parameters<-rbind(normal_parameters,normal_temp[-length(normal_temp)])
```

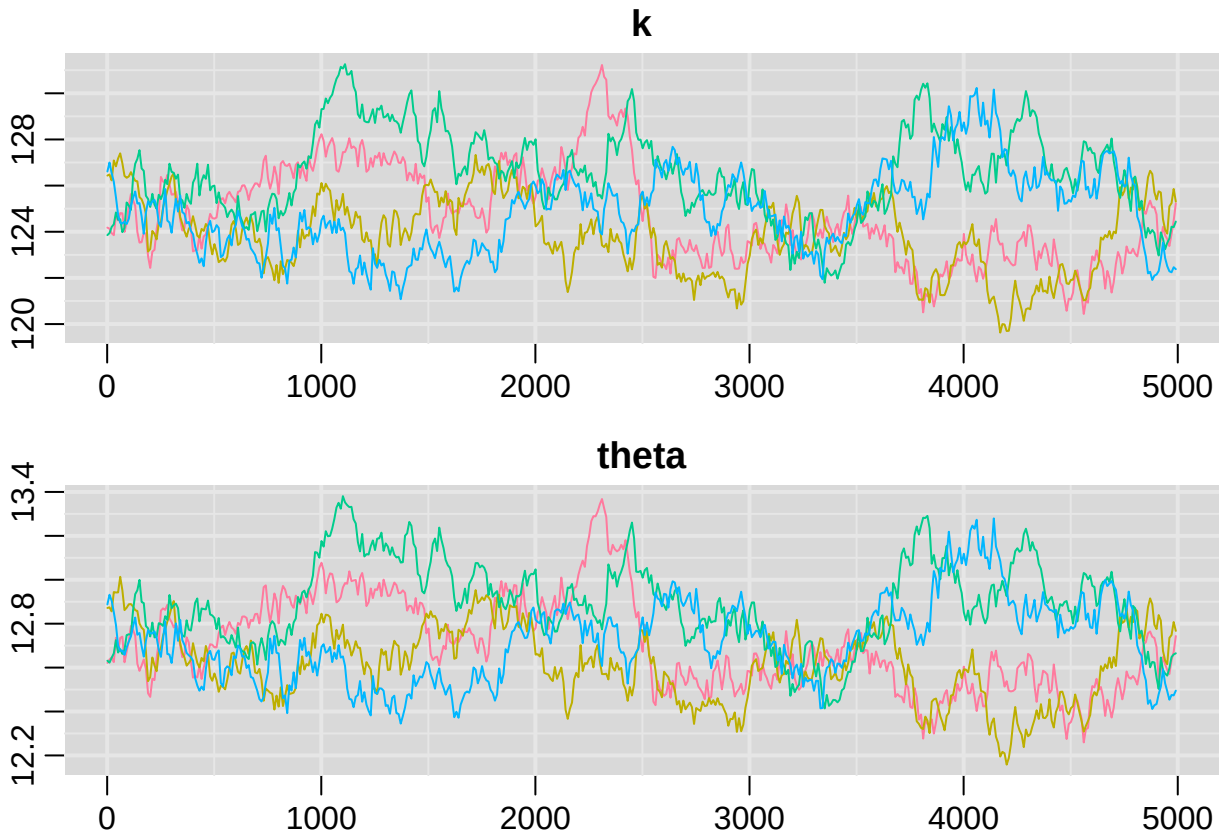
Using Gamma Model

As done before first we take a look at traceplots

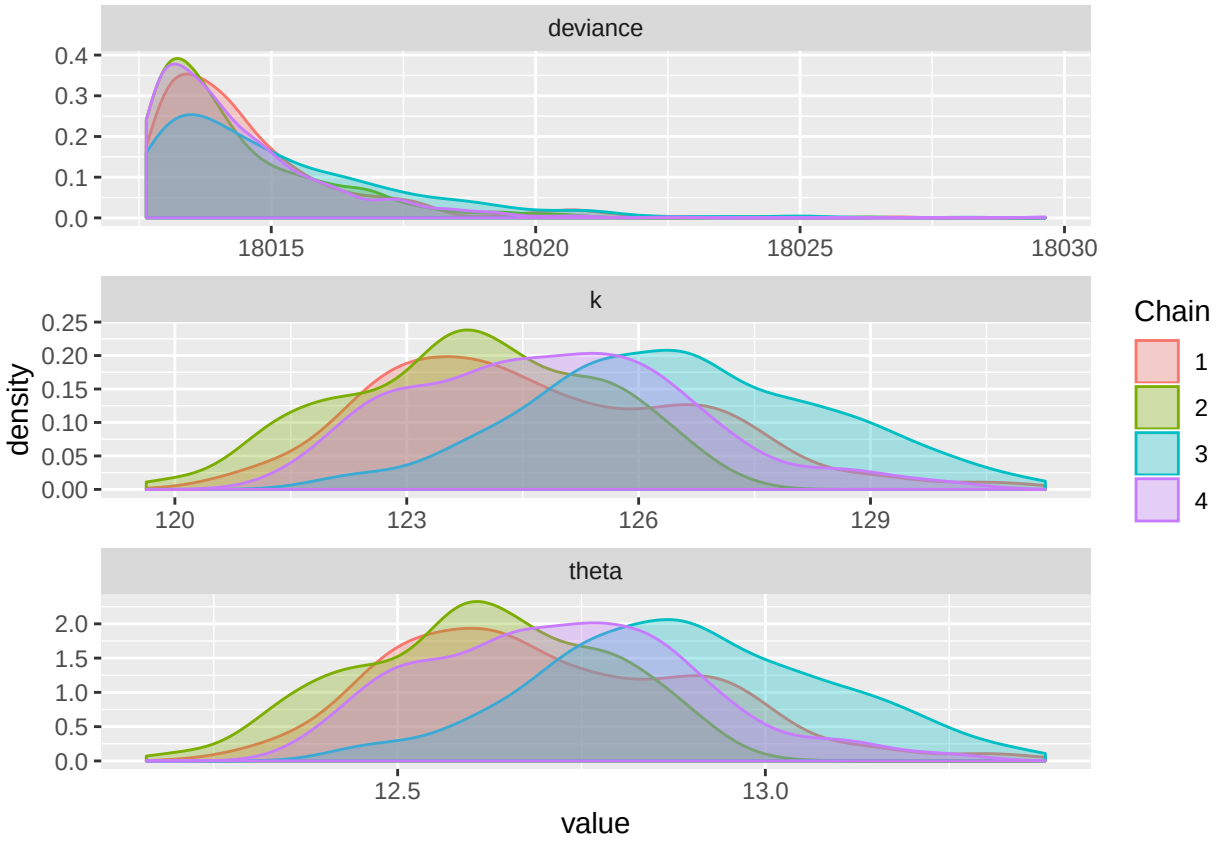
```r
gamma_temp<-gamma_fit(Y)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7000
##    Unobserved stochastic nodes: 2
##    Total graph size: 7004
##
## Initializing model
##
##
## Iterations = 1:4991
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 500
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##              Mean     SD Naive SE Time-series SE
## deviance 18014.62 1.9861 0.044410        0.13700
## k          124.99 2.0958 0.046863        0.30323
## theta       12.73 0.2138 0.004781        0.03047
##
## 2. Quantiles for each variable:
##
##              2.5%      25%      50%      75%    97.5%
## deviance 18012.70 18013.25 18014.01 18015.36 18020.17
## k          121.31   123.46   124.91   126.37   129.47
## theta       12.36    12.58    12.73    12.88    13.19
```

```r
f1<-gamma_temp$fit
par(mar = c(2, 0.5, 4, 2), oma = c(4, 4, 0.2, 0.2),mfrow=c(3,1))
traplot(as.mcmc(f1), parms = c('k','theta'))
```

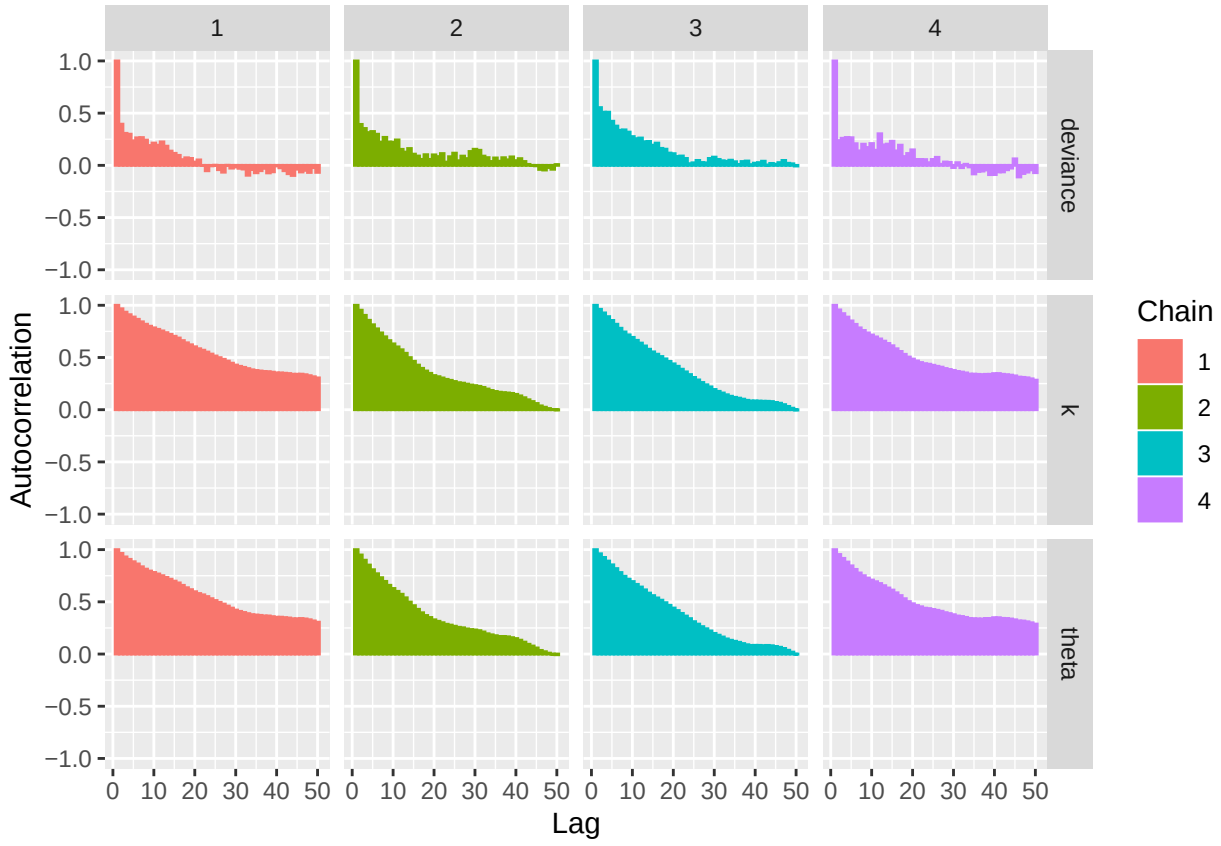



```
f.gg1<-ggs(as.mcmc(f1))
ggs_density(f.gg1)
```



And finally let's look to the autocorrelation

```
ggs_autocorrelation(f.gg1)
```



The same thinghs that we said for H_Magnitude are still true!

Then the HPD:

```
cat('----------- 90% -----------')
```

```
## ----------- 90% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.90)
```

```
## [[1]]
##                lower       upper
## deviance 18012.63649 18016.96963
## k          121.79044   128.06925
## theta       12.41615    13.03826
## attr(,"Probability")
## [1] 0.9
##
## [[2]]
##                lower       upper
## deviance 18012.63680 18016.77176
## k          121.25180   126.51960
## theta       12.34343    12.88014
## attr(,"Probability")
## [1] 0.9
##
```

```
## [[3]]
##                lower       upper
## deviance 18012.64483 18018.41833
## k          123.64548   129.80970
## theta       12.60119    13.23497
## attr(,"Probability")
## [1] 0.9
##
## [[4]]
##                lower       upper
## deviance 18012.63609 18016.46167
## k          122.09780   127.58326
## theta       12.41179    12.97356
## attr(,"Probability")
## [1] 0.9
```

```
cat('----------- 95% -----------')
```

```
## ----------- 95% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.95)
```

```
## [[1]]
##                lower       upper
## deviance 18012.63649 18017.99630
## k          120.76364   128.22693
## theta       12.31963    13.08701
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##                lower       upper
## deviance 18012.63680 18017.70737
## k          120.84050   126.69621
## theta       12.29386    12.90387
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##                lower       upper
## deviance 18012.64483 18019.93719
## k          123.16969   130.43260
## theta       12.52138    13.28253
## attr(,"Probability")
## [1] 0.95
##
## [[4]]
##                lower       upper
## deviance 18012.63609 18017.51411
## k          121.40562   128.24482
## theta       12.37928    13.07385
## attr(,"Probability")
## [1] 0.95
```

```
cat('----------- 99% -----------')
```

```
## ----------- 99% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.99)
```

```
## [[1]]
##                lower       upper
## deviance 18012.6365 18021.42737
## k          120.5066   130.61558
## theta       12.2592    13.28964
## attr(,"Probability")
## [1] 0.99
##
## [[2]]
##                 lower       upper
## deviance 18012.63680 18020.46308
## k          119.98362   127.32977
## theta       12.20953    12.95299
## attr(,"Probability")
## [1] 0.99
##
## [[3]]
##                 lower       upper
## deviance 18012.64483 18024.86877
## k          122.10528   131.07499
## theta       12.44176    13.35255
## attr(,"Probability")
## [1] 0.99
##
## [[4]]
##                 lower       upper
## deviance 18012.63609 18020.18363
## k          121.40562   129.87788
## theta       12.37154    13.23239
## attr(,"Probability")
## [1] 0.99
```

Now we store the parameters:

```
gamma_parameters<-rbind(gamma_parameters,gamma_temp[-length(gamma_temp)])

k<-gamma_temp$k
theta<-gamma_temp$theta
likelihood_ratios[2]<-sum(log(dnorm(x=Y,mean=mu,sd=1/tau)))-sum(log(dgamma(x=Y,shape = k,rate = theta)))
```

Analysis for R_KMag

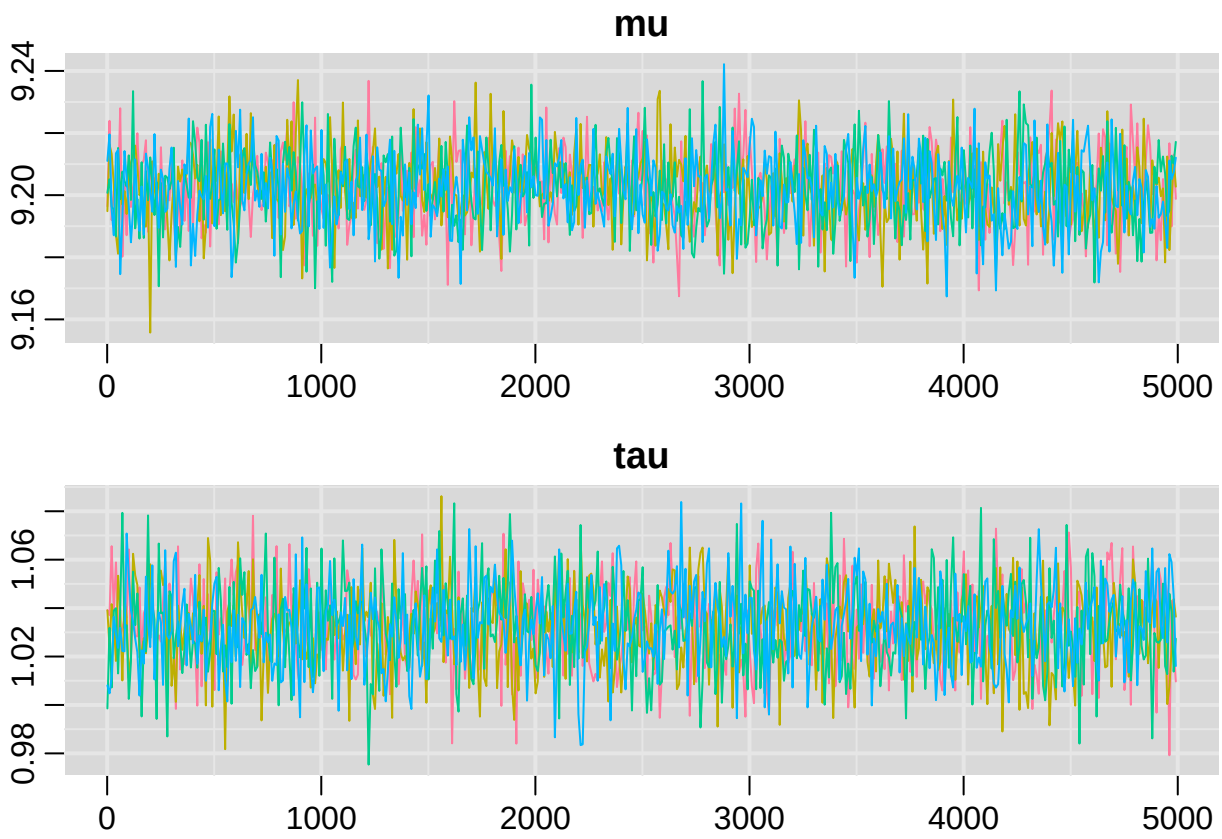
Using Normal Model

As said before, the first analysis will be taken using the normal model

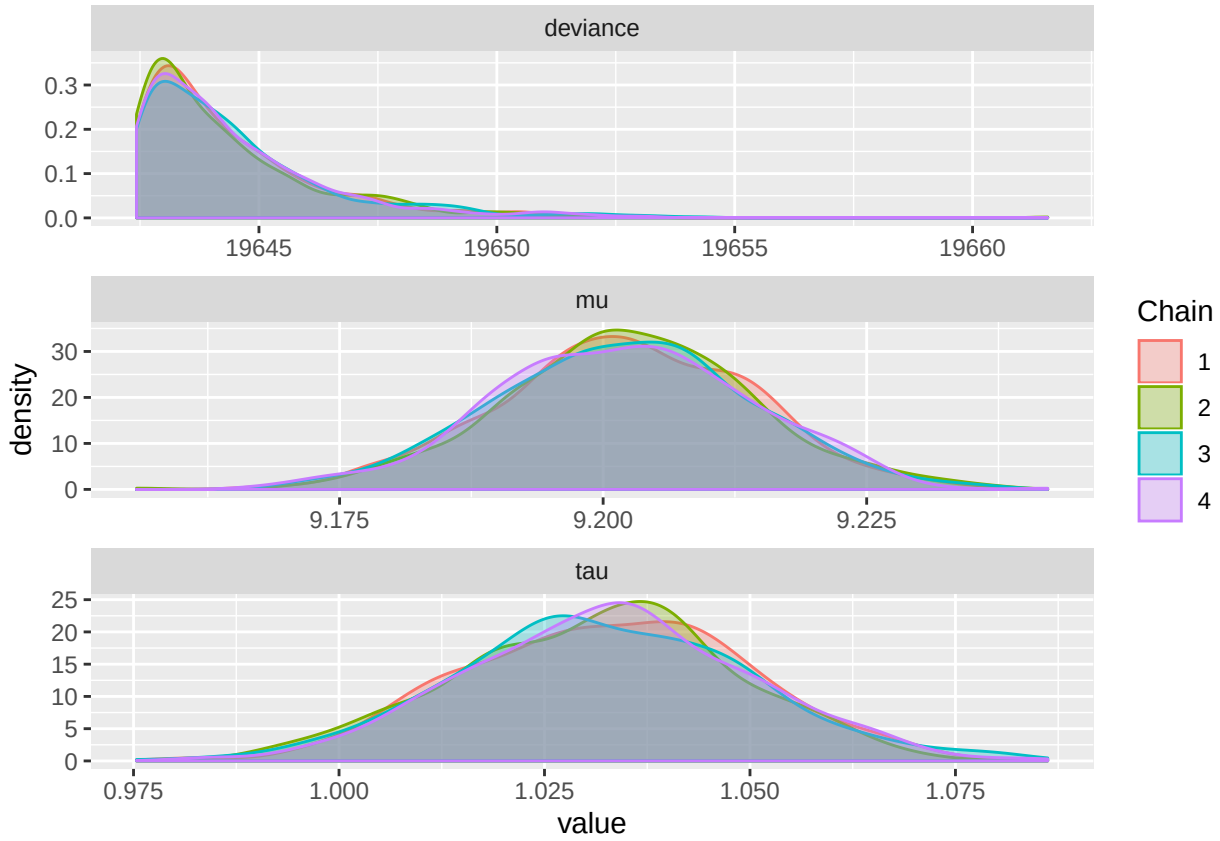
```r
Y=as.vector(Small_M$R_KMag)
normal_temp<-normal_fit(Y)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7000
##    Unobserved stochastic nodes: 2
##    Total graph size: 7010
##
## Initializing model
##
##
## Iterations = 1:4991
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 500
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean      SD  Naive SE Time-series SE
## deviance 19644.429 1.99247 0.0445530      0.0437474
## mu           9.202 0.01192 0.0002665      0.0002666
## tau          1.032 0.01711 0.0003826      0.0003856
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance 1.964e+04 19643.015 19643.861 19645.158 19650.208
## mu       9.178e+00     9.194     9.202     9.210     9.225
## tau      9.989e-01     1.020     1.033     1.044     1.065
```

```r
f<-normal_temp$fit
par(mar = c(2, 0.5, 4, 2), oma = c(4, 4, 0.2, 0.2),mfrow=c(3,1))
traplot(as.mcmc(f), parms = c('mu','tau'))
```

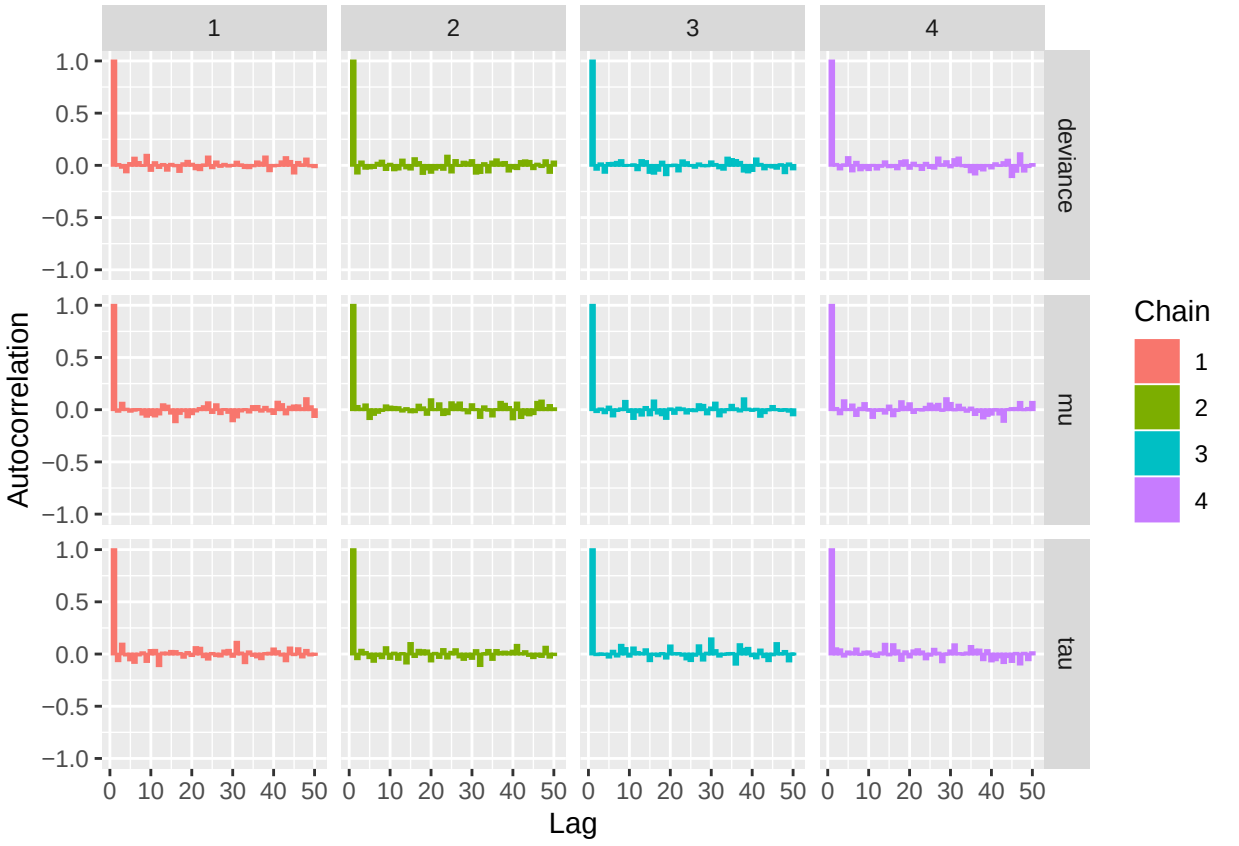



```
f.gg<-ggs(as.mcmc(f))
ggs_density(f.gg)
```



And finally let's look to the autocorrelation

```
ggs_autocorrelation(f.gg)
```



In this case, we can see from the traceplots that we reached the convergence for both paramenters, indeed we can see that the chains for both paramenters, oscillate around a specific value. Then they are stabilized theirself n their stable distribution. We can also see that autocorrelation go to zero for both parameter, and this is another check for the convergence.

Then the HPD

```
cat('----------- 90% -----------')
```

```
## ----------- 90% -----------
```

```
HPDinterval(as.mcmc(f),prob=.90)
```

```
## [[1]]
##                   lower        upper
## deviance 19642.430930 19647.045919
## mu           9.183562     9.221950
## tau          1.005947     1.060081
## attr(,"Probability")
## [1] 0.9
##
## [[2]]
##                   lower        upper
## deviance 19642.429794 19647.195015
## mu           9.182306     9.221491
## tau          1.005119     1.060229
```

```
## attr(,"Probability")
## [1] 0.9
##
## [[3]]
##                  lower        upper
## deviance 19642.428175 19647.340170
## mu           9.180309     9.219376
## tau          1.004409     1.062543
## attr(,"Probability")
## [1] 0.9
##
## [[4]]
##                  lower        upper
## deviance 19642.426104 19646.927908
## mu           9.184624     9.222774
## tau          1.003883     1.059490
## attr(,"Probability")
## [1] 0.9
```

```
cat('----------- 95% -----------')
```

```
## ----------- 95% -----------
```

```
HPDinterval(as.mcmc(f),prob=.95)
```

```
## [[1]]
##                  lower        upper
## deviance 19642.430930 19648.287555
## mu           9.177359     9.223331
## tau          1.004353     1.067195
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##                  lower        upper
## deviance 1.964243e+04 19648.386874
## mu       9.178827e+00     9.226058
## tau      9.979817e-01     1.061231
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##                  lower        upper
## deviance 1.964243e+04 19648.982408
## mu       9.178281e+00     9.224482
## tau      9.942277e-01     1.064378
## attr(,"Probability")
## [1] 0.95
##
## [[4]]
##                  lower        upper
## deviance 19642.426104 19648.521623
## mu           9.180407     9.225101
```

```
## tau          1.001450     1.065760
## attr(,"Probability")
## [1] 0.95
```

```
cat('----------- 99% -----------')
```

```
## ----------- 99% -----------
```

```
HPDinterval(as.mcmc(f),prob=.99)
```

```
## [[1]]
##                 lower        upper
## deviance 1.964243e+04 19651.028887
## mu       9.175267e+00     9.233679
## tau      9.946831e-01     1.072822
## attr(,"Probability")
## [1] 0.99
##
## [[2]]
##                 lower        upper
## deviance 1.964243e+04 19651.649398
## mu       9.171526e+00     9.233599
## tau      9.910683e-01     1.069452
## attr(,"Probability")
## [1] 0.99
##
## [[3]]
##                 lower       upper
## deviance 1.964243e+04 19652.41083
## mu       9.170030e+00     9.23030
## tau      9.906946e-01     1.08328
## attr(,"Probability")
## [1] 0.99
##
## [[4]]
##                 lower        upper
## deviance 19642.426104 19651.855952
## mu           9.171410     9.228078
## tau          0.983738     1.072672
## attr(,"Probability")
## [1] 0.99
```

And we store the parameters:

```
mu<-normal_temp$mu
tau<-normal_temp$tau

normal_parameters<-rbind(normal_parameters,normal_temp[-length(normal_temp)])
```

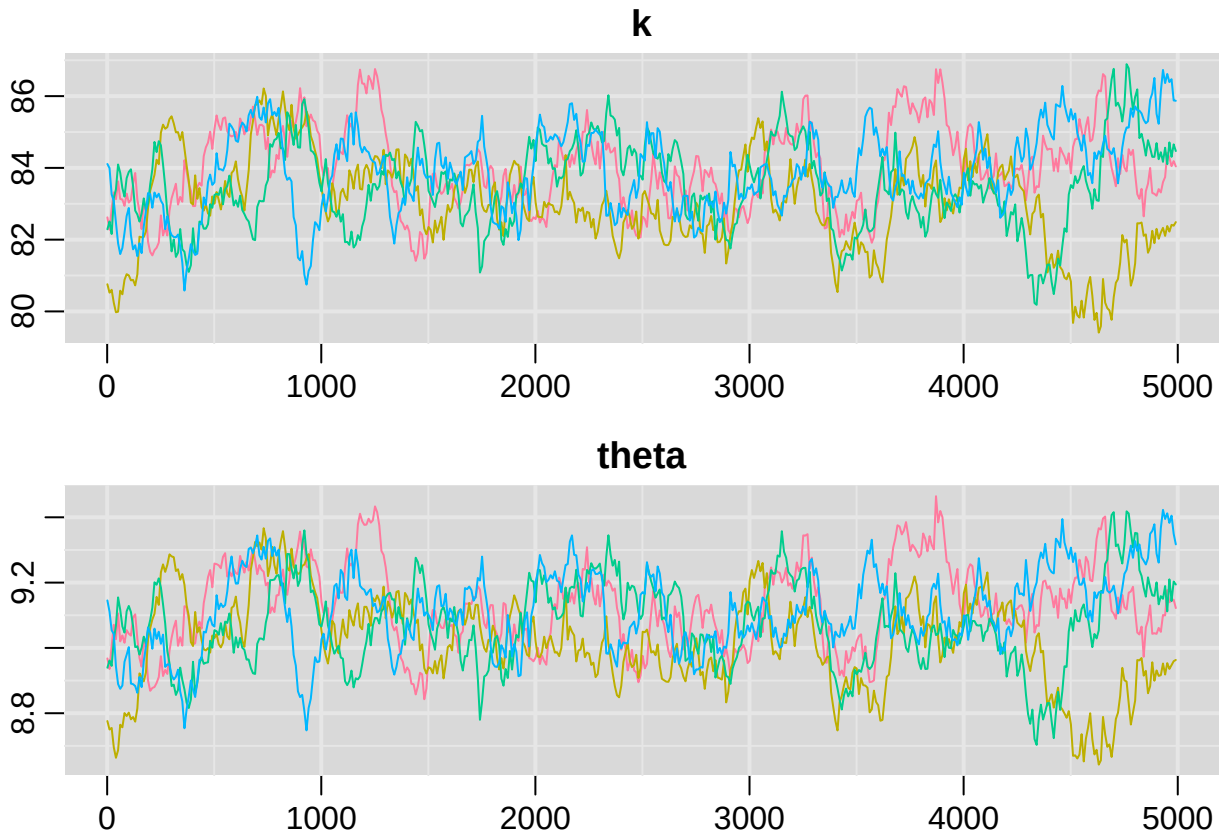
Using Gamma Model

As done before first we take a look at traceplots

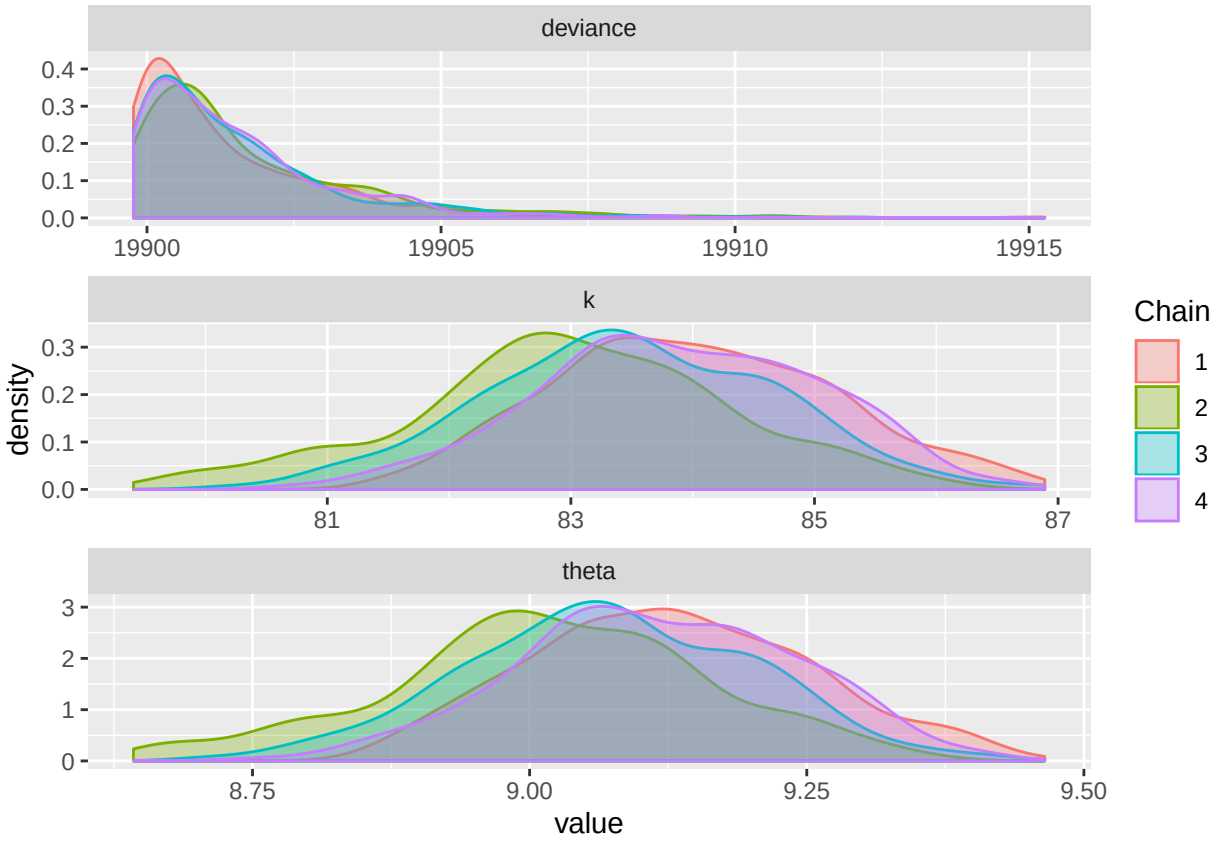
```r
gamma_temp<-gamma_fit(Y)
```

```
## Compiling model graph
##    Resolving undeclared variables
##    Allocating nodes
## Graph information:
##    Observed stochastic nodes: 7000
##    Unobserved stochastic nodes: 2
##    Total graph size: 7004
##
## Initializing model
##
##
## Iterations = 1:4991
## Thinning interval = 10
## Number of chains = 4
## Sample size per chain = 500
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##               Mean    SD Naive SE Time-series SE
## deviance 19901.565 1.847 0.041307        0.10461
## k           83.591 1.263 0.028245        0.14768
## theta        9.083 0.138 0.003086        0.01596
##
## 2. Quantiles for each variable:
##
##               2.5%       25%       50%       75%     97.5%
## deviance 19899.819 19900.298 19900.973 19902.199 19906.733
## k           80.943    82.765    83.559    84.493    85.968
## theta        8.799     8.991     9.081     9.181     9.345
```

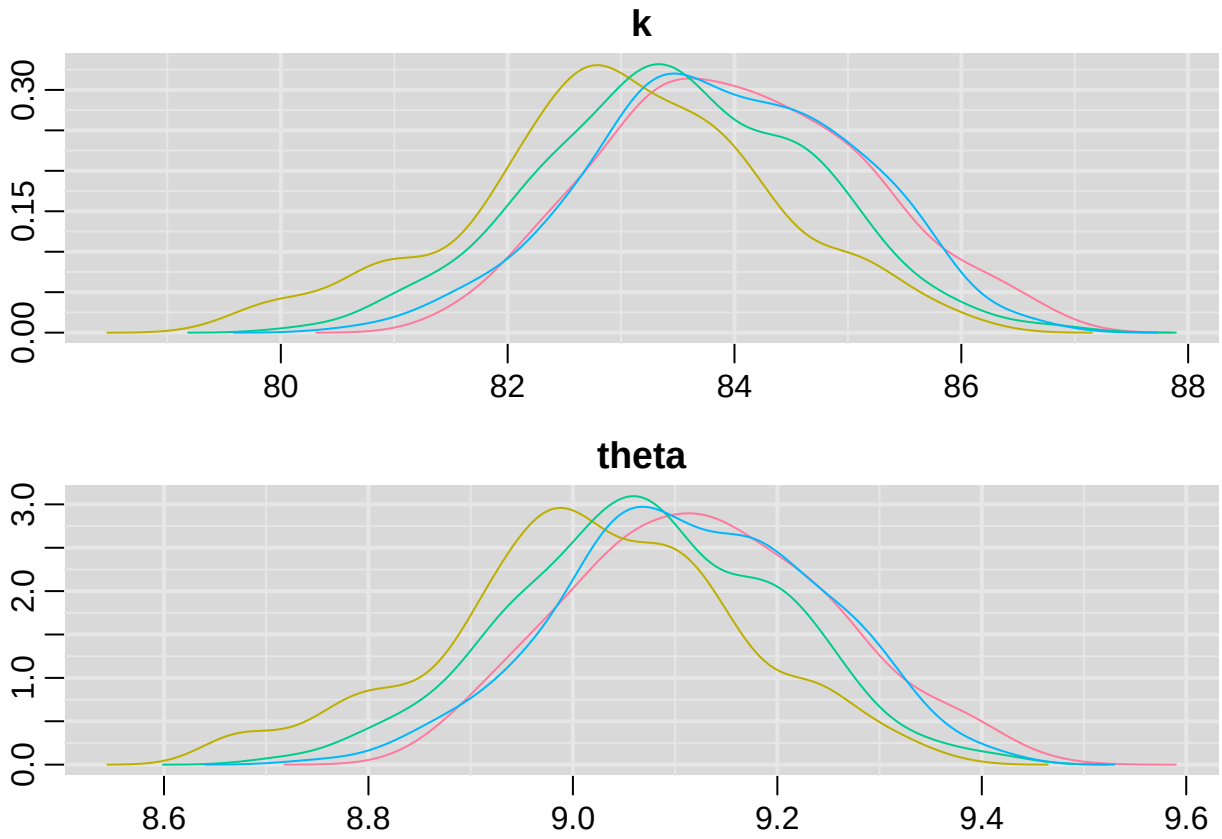
```r
f1<-gamma_temp$fit
par(mar = c(2, 0.5, 4, 2), oma = c(4, 4, 0.2, 0.2),mfrow=c(3,1))
traplot(as.mcmc(f1), parms = c('k','theta'))
```




```
f.gg1<-ggs(as.mcmc(f1))
ggs_density(f.gg1)
```

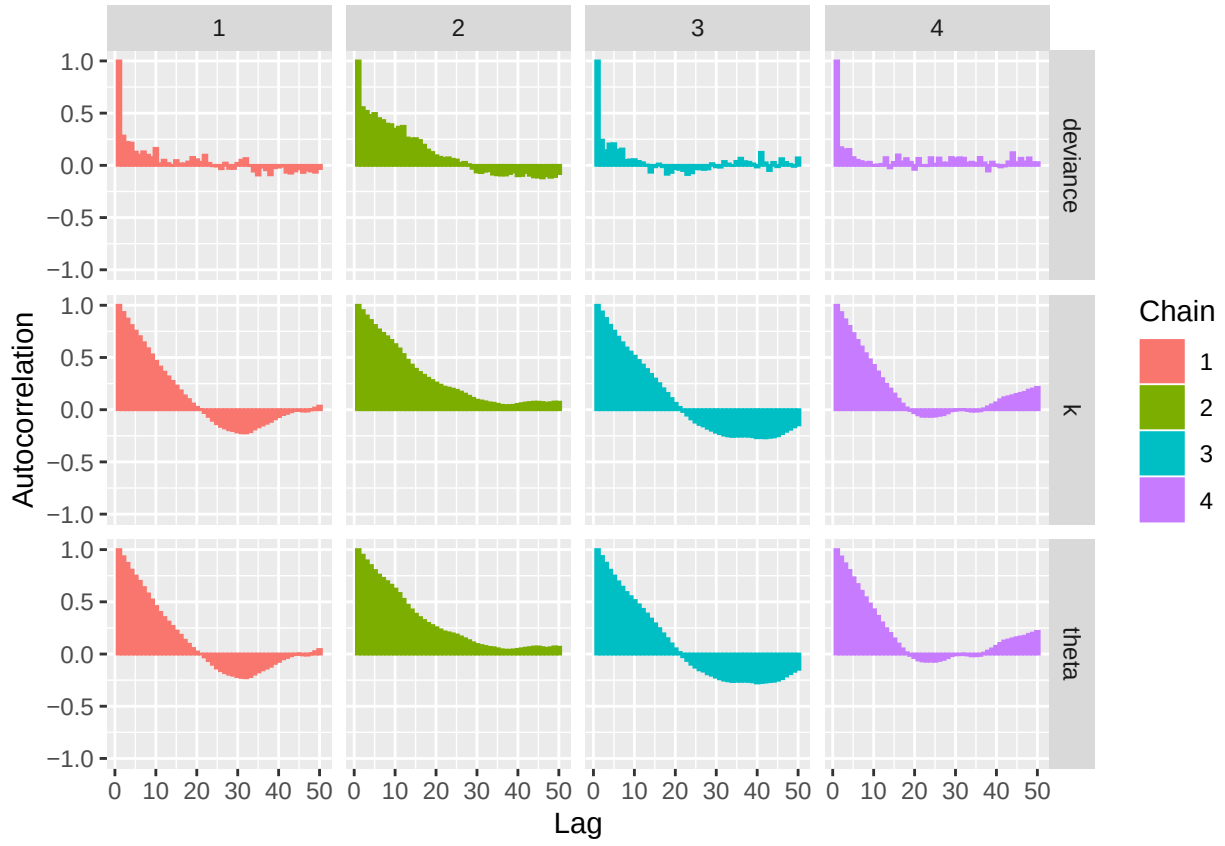



```
denplot(as.mcmc(f1),parms = c('k','theta'))
```



And finally let's look to the autocorrelation

```
ggs_autocorrelation(f.gg1)
```



The thigh we said for H-Magnitude and J-Magnitude are still true!

Then the HPD:

```
cat('----------- 90% -----------')
```

```
## ----------- 90% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.90)
```

```
## [[1]]
##                  lower       upper
## deviance 19899.770911 19903.60648
## k           82.072614    85.80544
## theta        8.894867     9.30854
## attr(,"Probability")
## [1] 0.9
##
## [[2]]
##                  lower        upper
## deviance 19899.774360 19904.151197
## k           80.792991    85.241446
## theta        8.776128     9.255809
## attr(,"Probability")
## [1] 0.9
##
```

```
## [[3]]
##                  lower        upper
## deviance 19899.773802 19904.000547
## k           81.599696    85.456905
## theta        8.872854     9.287685
## attr(,"Probability")
## [1] 0.9
##
## [[4]]
##                  lower        upper
## deviance 19899.770821 19903.688606
## k           82.224837    85.817694
## theta        8.933444     9.327904
## attr(,"Probability")
## [1] 0.9
```

```
cat('----------- 95% -----------')
```

```
## ----------- 95% -----------
```

```
HPDinterval(as.mcmc(f1),prob=.95)
```

```
## [[1]]
##                  lower        upper
## deviance 19899.77091 19904.931208
## k           82.06253    86.352046
## theta        8.91086     9.390936
## attr(,"Probability")
## [1] 0.95
##
## [[2]]
##                  lower        upper
## deviance 19899.774360 19906.075033
## k           80.476758    85.775114
## theta        8.750167     9.325439
## attr(,"Probability")
## [1] 0.95
##
## [[3]]
##                  lower        upper
## deviance 19899.773802 19905.377690
## k           80.929528    85.550506
## theta        8.796846     9.294698
## attr(,"Probability")
## [1] 0.95
##
## [[4]]
##                  lower        upper
## deviance 19899.78134 19904.552994
## k           81.53902    85.817694
## theta        8.86262     9.335493
## attr(,"Probability")
## [1] 0.95
```

```r
cat('----------- 99% -----------')
```

```
## ----------- 99% -----------
```

```r
HPDinterval(as.mcmc(f1),prob=.99)
```

```
## [[1]]
##                  lower        upper
## deviance 19899.770911 19908.113866
## k           81.620479    86.755569
## theta        8.868595     9.419311
## attr(,"Probability")
## [1] 0.99
##
## [[2]]
##                  lower        upper
## deviance 19899.774360 19910.531548
## k           79.588892    85.880975
## theta        8.652364     9.325439
## attr(,"Probability")
## [1] 0.99
##
## [[3]]
##                  lower        upper
## deviance 19899.773802 19909.048242
## k           80.781080    86.774905
## theta        8.775536     9.418264
## attr(,"Probability")
## [1] 0.99
##
## [[4]]
##                  lower        upper
## deviance 19899.770821 19907.678406
## k           81.024353    86.507304
## theta        8.806677     9.405087
## attr(,"Probability")
## [1] 0.99
```

Now we store the parameters:

```r
gamma_parameters<-rbind(gamma_parameters,gamma_temp[-length(gamma_temp)])

k<-gamma_temp$k
theta<-gamma_temp$theta
likelihood_ratios[3]<-sum(log(dnorm(x=Y,mean=mu,sd=1/tau)))-sum(log(dgamma(x=Y,shape = k,rate = theta)))
```

In the following part, will be shown the obtained parameters from JAGS

Normal

```
dd<-normal_parameters[,c('mu','tau')]
rownames(dd)<-colnames(Small_M)[c(2,3,4)]

kable(dd) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	mu	tau
R_HMag	9.323996	1.105299
R_JMag	9.814762	1.345528
R_KMag	9.202165	1.032223

Gamma

```
dd<-gamma_parameters[,c('k','theta')]
rownames(dd)<-colnames(Small_M)[c(2,3,4)]
kable(dd) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

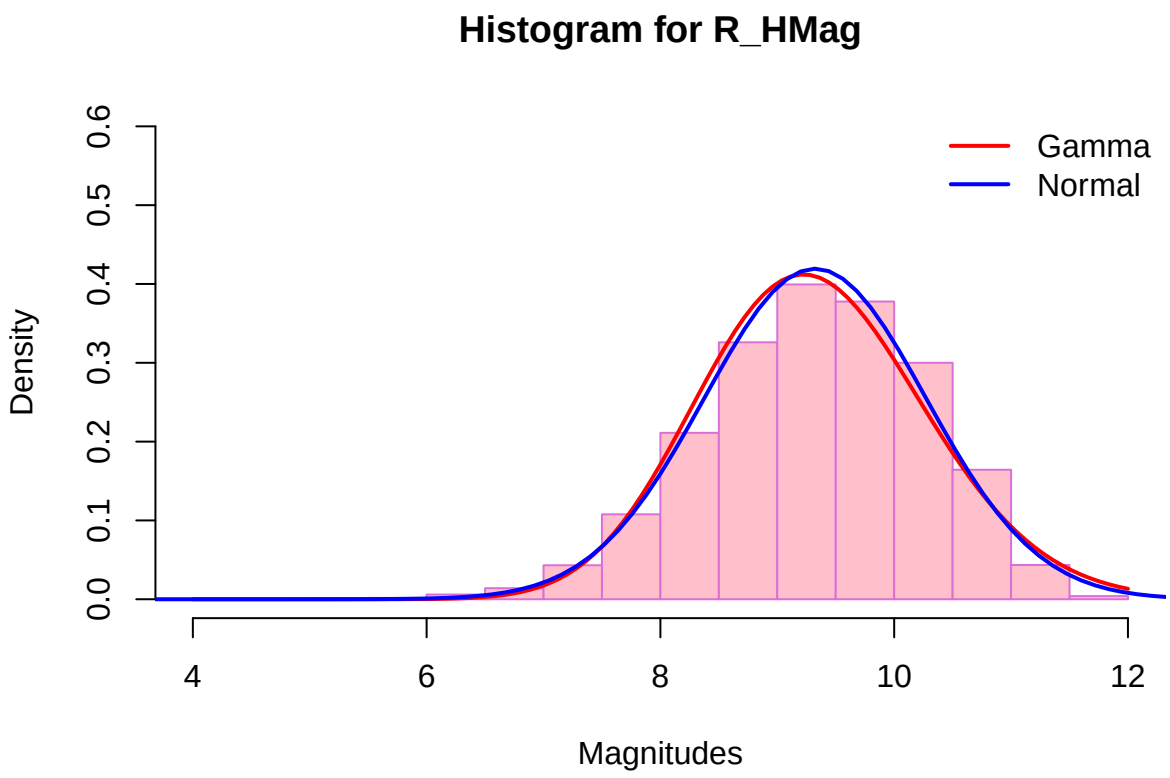
	k	theta
R_HMag	91.95576	9.860713
R_JMag	124.98750	12.733110
R_KMag	83.59126	9.082508

Model comparison

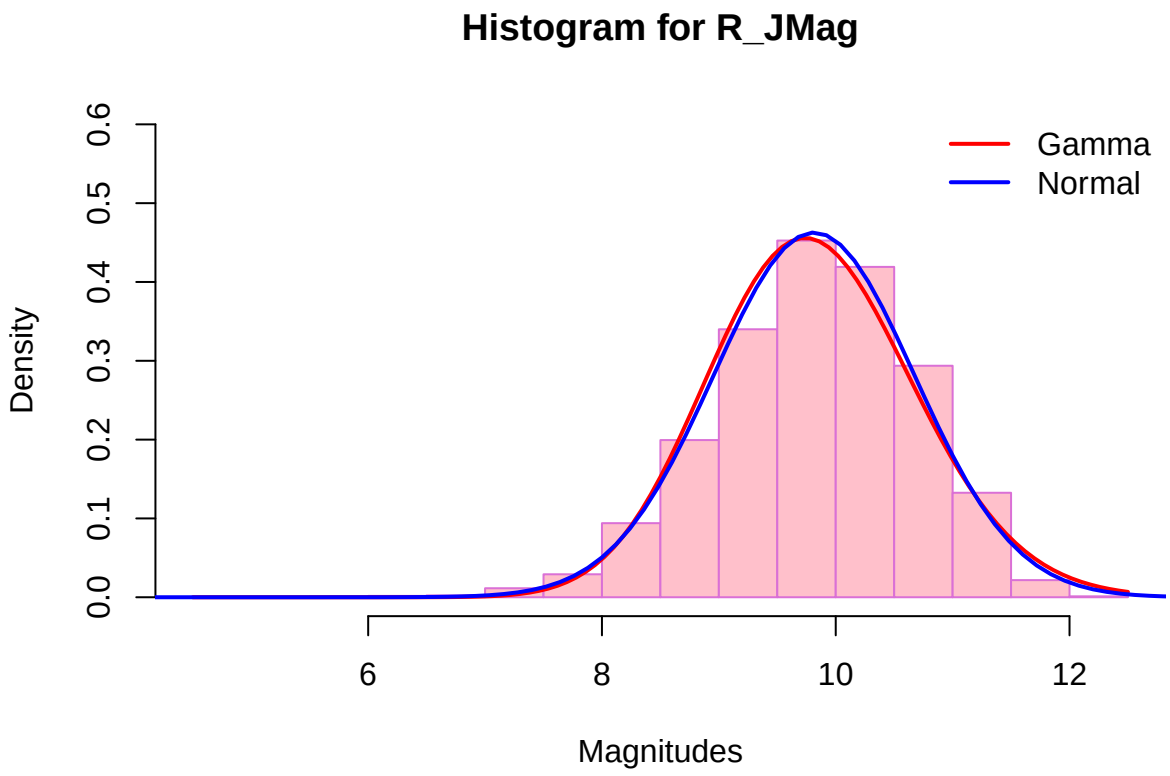
In this part, we re going to see, how the two proposed model fit the data. The first analysis it's only graphical, and what we expect is that since the chain for the parameter of the gamma model, do not reach the convergence, the model that best fits the data is the normal model

Graphical

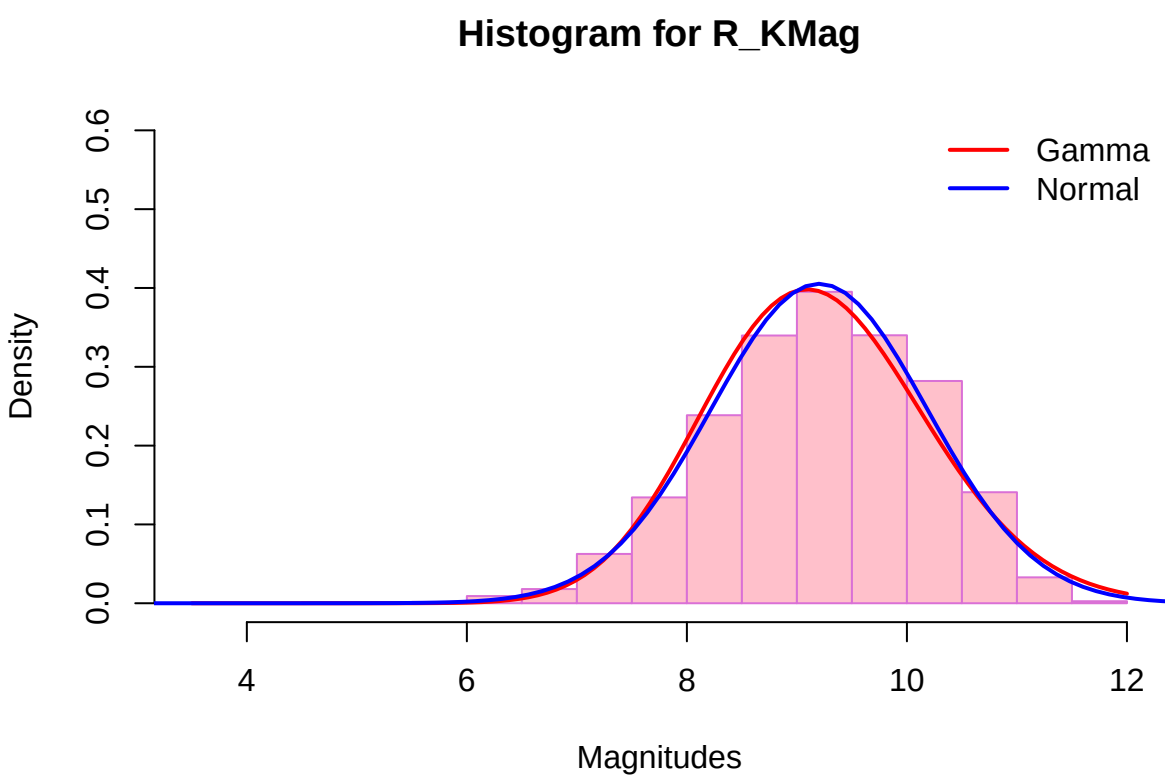
```
hist(Small_M$R_HMag,prob=TRUE,ylim=c(0,.6),
     main='Histogram for R_HMag',xlab = 'Magnitudes',col='pink',border = 'orchid')
curve(dgamma(x,shape = gamma_parameters$k[1],rate = gamma_parameters$theta[1]),
      lwd=2,col='red',add=TRUE,ylim=c(0,1.2))
curve(dnorm(x,mean=normal_parameters$mu[1],
            sd=sqrt(1/normal_parameters$tau[1])),from=2, to=14, add=TRUE,
      lwd=2,col='blue',ylim=c(0,1.2))
legend('topright', legend=c('Gamma','Normal'), col=c('red','blue'),
       lty=1, bty='n', lwd=2, cex=1, text.col='black', horiz=F)
```




```
hist(Small_M$R_JMag,prob=TRUE,ylim=c(0,.6),
     main='Histogram for R_JMag',xlab = 'Magnitudes',col='pink',border = 'orchid')
curve(dgamma(x,shape = gamma_parameters$k[2],rate = gamma_parameters$theta[2]),
      lwd=2,col='red',add=TRUE,ylim=c(0,1.2))
curve(dnorm(x,mean=normal_parameters$mu[2],
            sd=sqrt(1/normal_parameters$tau[2])),from=2, to=14, add=TRUE,lwd=2,col='blue',ylim=c(0,1.2))
legend('topright', legend=c('Gamma','Normal'),
       col=c('red','blue'), lty=1, bty='n', lwd=2, cex=1, text.col='black', horiz=F)
```




```
hist(Small_M$R_KMag,prob=TRUE,ylim=c(0,.6),
     main='Histogram for R_KMag',xlab = 'Magnitudes',col='pink',border = 'orchid')
curve(dgamma(x,shape = gamma_parameters$k[3],rate = gamma_parameters$theta[3]),
      lwd=2,col='red',add=TRUE,ylim=c(0,1.2))
curve(dnorm(x,mean=normal_parameters$mu[3],sd=sqrt(1/normal_parameters$tau[3])),
      from=2, to=14, add=TRUE,lwd=2,col='blue',ylim=c(0,1.2))
legend('topright', legend=c('Gamma','Normal'),
       col=c('red','blue'), lty=1, bty='n', lwd=2, cex=1, text.col='black', horiz=F)
```



Which is better?

In this part, we are going to discuss which of the the two proposed model, better fits the data. In order to do this, two different approach were used:

- Likelihood ratio
- DIC (Deviance Infomation Criterion)

Model comparison is typically done by comparing the likelihoods of some data x_i under two different model:A_1 and A_2. Under a likelihood framework, the interpretation is clear: if the data are more likely under A_1, then A_1 is a better explanation of the data; and conversely for A_2.

Bayes Factor

The first criteria is the likelihood ratio. As said before assesses the goodness of fit of two competing statistical models based on the ratio of their likelihoods. It is possible to do this this check using the **Bayes factor** that is the ratio of marginal likelihood of A_1 to that of A_2. It is defined as:

$$K = \frac{\mathbb{P}(\{x_1, \ldots, x_n\}|A_1)}{\mathbb{P}(\{x_1, \ldots, x_n\}|A_2)} = \frac{\prod_{i=1}^{n} m(x_i|A_1)}{\prod_{i=1}^{n} m(x_i|A_2)}.$$

In this case the model A_1 corresponds to the Normal model, while A_2 corresponds to the Gamma model, and according to K:

- If $K >> 1$ then the model A_1 will be a better fit

- If $K << 1$ then the model A_2 will be a better fit
- If $K \approx 1$ there is not a great difference between the two models

In our case will be use a logarithmic version of this ratio:

$$K = \log\left(\sum_{i=1}^{n} m(x_i|A_1)\right) - \log\left(\sum_{i=1}^{n} m(x_i|A_2)\right)$$

```
likelihood_ratios
```

```
## [1] 100.79790 -60.44916 126.89632
```

According to this, we can see that R_HMag and R_KMag are better approximated using a normal, and R_JMag are better approximated using a Gamma Distribution. The let us think that there is an effect due to the different filter used to retrive data.

DIC

The second test to understand what is the best model is done using the DIC. The Deviance Information Criterion is useful in bayesian model selection problems, where the posterior distributions of the model have been obtained from MCMC simulations. The measure of fit $\bar{D}$ can be combined with the measure of complexity p_D to produce the Deviance Information Criterion:

$$\begin{aligned} DIC &= \bar{D} + p_D \\ &= \hat{D} + 2p_D \end{aligned}$$

where once defined the deviance as $D(\theta) - 2\log(p(y|\theta))$, the pD is computed by JAGS using the formula proposed by **Geleman et al.(2004,p=182)**:

$$p_D = \frac{\mathbb{V}(D(\theta))}{2}$$

The idea is that models with smaller DIC should be preferred to models with larger DIC.

Normal

```
D_gamma<-gamma_parameters[,c('DIC','pD')]
D_normal<-normal_parameters[,c('DIC','pD')]
#dbar=DIC-pD
#Dthetabar=dic-2pD
D_gamma$dBar=D_gamma$DIC-D_gamma$pD
D_gamma$DthetaBar=D_gamma$DIC-2*D_gamma$pD

D_normal$dBar=D_normal$DIC-D_normal$pD
D_normal$DthetaBar=D_normal$DIC-2*D_normal$pD

#Smaller Dic model preferred than high DIC model
rownames(D_normal)<-colnames(Small_M)[c(2,3,4)]
kable(D_normal) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	DIC	pD	dBar	DthetaBar
R_HMag	19170.44	1.886070	19168.55	19166.67
R_JMag	17795.27	2.138873	17793.13	17790.99
R_KMag	19646.42	1.986026	19644.43	19642.44

Gamma

```
rownames(D_gamma)<-colnames(Small_M)[c(2,3,4)]
kable(D_gamma) %>%
  kable_styling(bootstrap_options = "striped", full_width = F)
```

	DIC	pD	dBar	DthetaBar
R_HMag	19408.45	1.767921	19406.68	19404.91
R_JMag	18016.54	1.921531	18014.62	18012.70
R_KMag	19903.26	1.699541	19901.57	19899.87

For what said before, models with smaller DIC should be preferred to models with larger DIC, then in this case, the normal model is the chosen model. This is in contast with what we said before.

Bayesian vs Frequentist

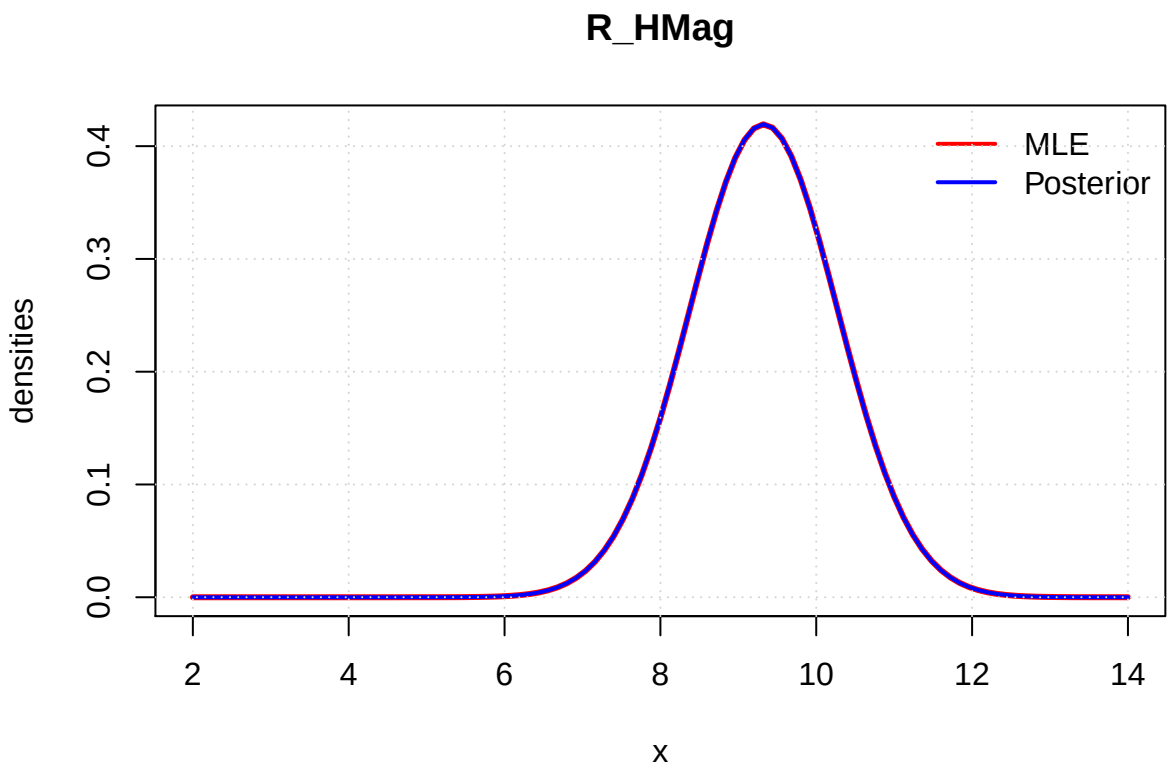
In the previous part, we said that the preferred model is the normal model, now can be interesting to do a comparison between what we find using a Bayesian approch with what will find with a Frequentist approach. Now we are focused only on the normal model.

For the nromal distribution, the *Maximum Likelihood Estimator* for the paramenters are well known, indeed if $\hat{\mu} = \hat{\mu}(x_1, \dots, x_n)$ is the MLE for the mean μ and $\hat{\sigma}^2 = \hat{\sigma}^2(x_1, \dots, x_n)$ is the MLE for the variance σ^2, then

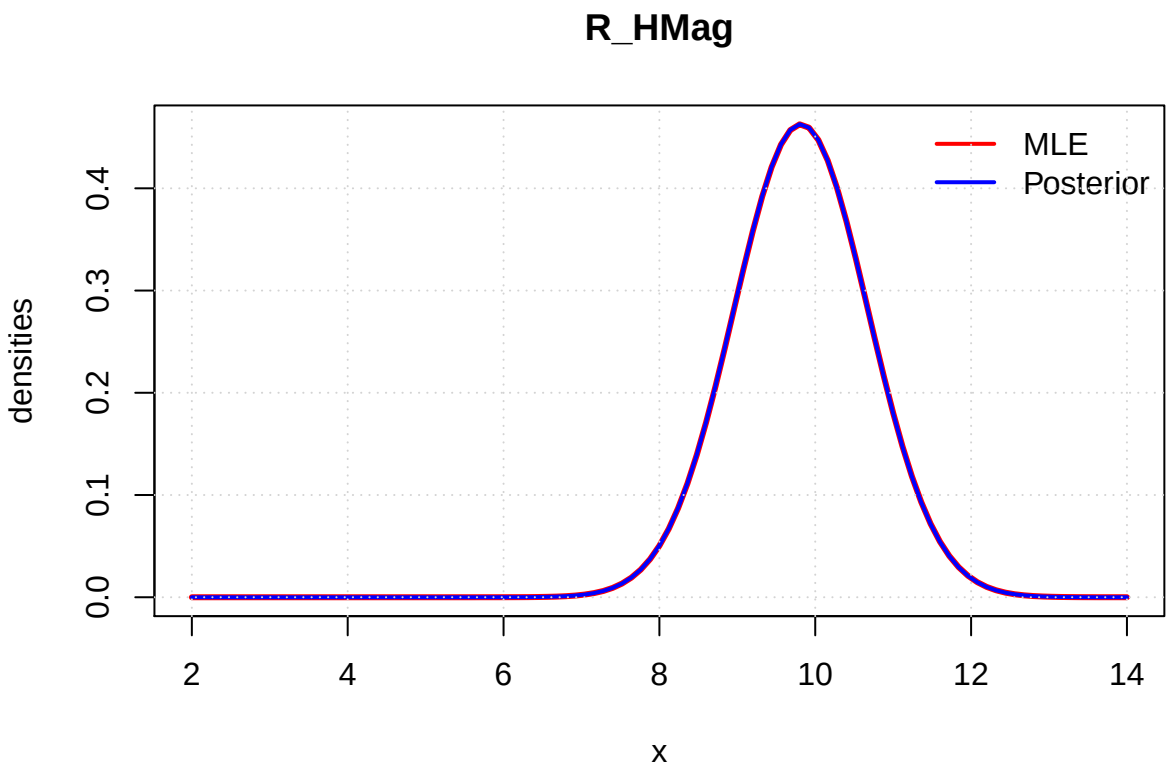
$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x_i$$
$$\hat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{\mu})^2$$

Let's see what happens graphically

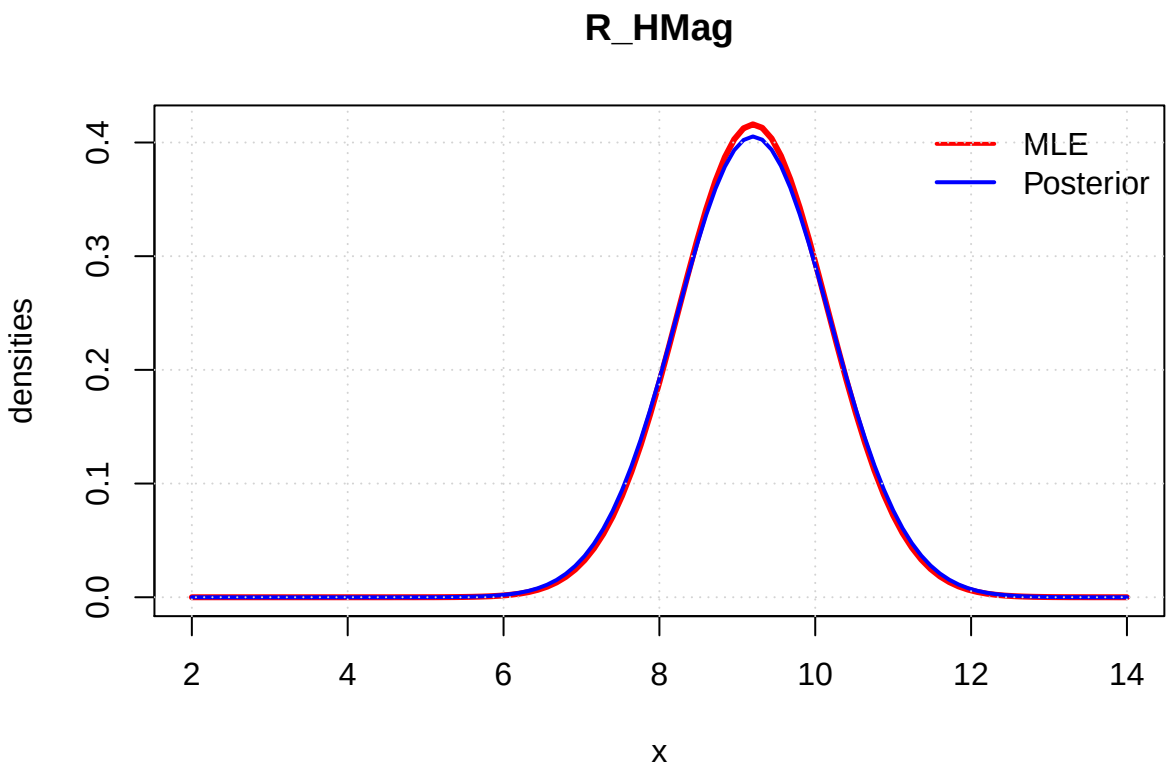
```
mu_MLE_H=sum(Small_M$R_HMag)/length(Small_M$R_HMag)
mu_MLE_J=sum(Small_M$R_JMag)/length(Small_M$R_JMag)
mu_MLE_K=sum(Small_M$R_KMag)/length(Small_M$R_KMag)

sd_MLE_H=sqrt(sum((Small_M$R_HMag-mu_MLE_H)^2)/length(Small_M$R_HMag))
sd_MLE_J=sqrt(sum((Small_M$R_JMag-mu_MLE_J)^2)/length(Small_M$R_JMag))
sd_MLE_K=sqrt(sum((Small_M$R_HMag-mu_MLE_K)^2)/length(Small_M$R_KMag))

curve(dnorm(x,mean=mu_MLE_H,sd=sd_MLE_H),from=2,to=14,
      lwd=3, col='red',main='R_HMag',ylab = 'densities')
curve(dnorm(x,mean=normal_parameters$mu[1],
            sd=sqrt(1/normal_parameters$tau[1])),from=2, to=14, add=TRUE,lwd=2,col='blue',ylim=c(0,1.4))
legend('topright', legend=c('MLE','Posterior'),
       col=c('red','blue'), lty=1, bty='n', lwd=2, cex=1, text.col='black', horiz=F)
grid()
```




```
curve(dnorm(x,mean=mu_MLE_J,sd=sd_MLE_J),from=2,to=14,
      lwd=3, col='red',main='R_HMag',ylab = 'densities')
curve(dnorm(x,mean=normal_parameters$mu[2],sd=sqrt(1/normal_parameters$tau[2])),
      from=2, to=14, add=TRUE,lwd=2,col='blue',ylim=c(0,1.4))
legend('topright', legend=c('MLE','Posterior'),
       col=c('red','blue'), lty=1, bty='n', lwd=2, cex=1,
       text.col='black', horiz=F)
grid()
```




```
curve(dnorm(x,mean=mu_MLE_K,sd=sd_MLE_K),from=2,to=14,
      lwd=3, col='red',main='R_HMag',ylab = 'densities')
curve(dnorm(x,mean=normal_parameters$mu[3],sd=sqrt(1/normal_parameters$tau[3])),
      from=2, to=14, add=TRUE,lwd=2,col='blue',ylim=c(0,1.4))
legend('topright', legend=c('MLE','Posterior'), col=c('red','blue'),
       lty=1, bty='n', lwd=2, cex=1, text.col='black', horiz=F)
grid()
```



Kolmogorov-Smirnov Test

In statistics, the Kolmogorov–Smirnov test (K–S test or KS test) is a nonparametric test of the equality of continuous, one-dimensional probability distributions that can be used to compare a sample with a reference probability distribution (one-sample K–S test), or to compare two samples (two-sample K–S test).

The Kolmogorov–Smirnov statistic quantifies a distance between the empirical distribution function of the sample and the cumulative distribution function of the reference distribution, or between the empirical distribution functions of two samples. The null distribution of this statistic is calculated under the null hypothesis that the sample is drawn from the reference distribution.

Since we discovered that for all the three types of measurements, the model that better fit our data is the normal, we want to do the following test:

H_0 : The sample comes from the normal distribution *vs* H_1 : The sample doesn't come from a normal distribution

This will give two output:

- The statistic D that is the maximum distance between the sample distribution and the fitted distribution. For "small" value of we cannot reject the null hypothesis, otherwise if we get a "large" value for D we can reject the null hypothesis.
- The p-value corresponding to the statistic D. We will reject the null hypothsis if the p-values is less than our significance level.

```
parametersH<-fitdistr(Small_M$R_HMag,"normal")
parametersJ<-fitdistr(Small_M$R_JMag,"normal")
parametersK<-fitdistr(Small_M$R_KMag,"normal")

mH<-parametersH$estimate[1]
sdH<-parametersH$estimate[2]
mJ<-parametersJ$estimate[1]
sdJ<-parametersJ$estimate[2]
mK<-parametersK$estimate[1]
sdK<-parametersK$estimate[2]

ks.test(unique(Small_M$R_HMag),"pnorm",mH,sdH)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  unique(Small_M$R_HMag)
## D = 0.078522, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(unique(Small_M$R_JMag),"pnorm",mJ,sdJ)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  unique(Small_M$R_JMag)
## D = 0.084926, p-value < 2.2e-16
## alternative hypothesis: two-sided
```

```
ks.test(unique(Small_M$R_KMag),"pnorm",mK,sdK)
```

```
##
##  One-sample Kolmogorov-Smirnov test
##
## data:  unique(Small_M$R_KMag)
## D = 0.074823, p-value = 4.441e-16
## alternative hypothesis: two-sided
```

We can see that in all test, we get a low value for D and an high p-value, than in both cases we cannot reject the null hypothesis, so the samples come from a normal distribution.

Conclusions

We analyzed some light flow, coming from observed stars, using a bayesian approach and finding that the best approximating model for the chosen data is the Normal one. We also compare this bayesian approach with the frequentist one, finding that the better approximation is again the normal. Therefore, we can assume that the normal model is the best fit for our porpouse. A possible interesting analysis can be taking into account the different infrared wavelength filters, trying to develop a single more complicated model, modifying the mean and the standard deviation of the normal.

References

- Bayesian Data Analysis,Andrew Gelman and Donald Rubin.
- Bayesian Modeling Using Winbugs, Wiley, Ntzoufras, Ioannis. 2009
- Magnitudine Apparente
- ESA