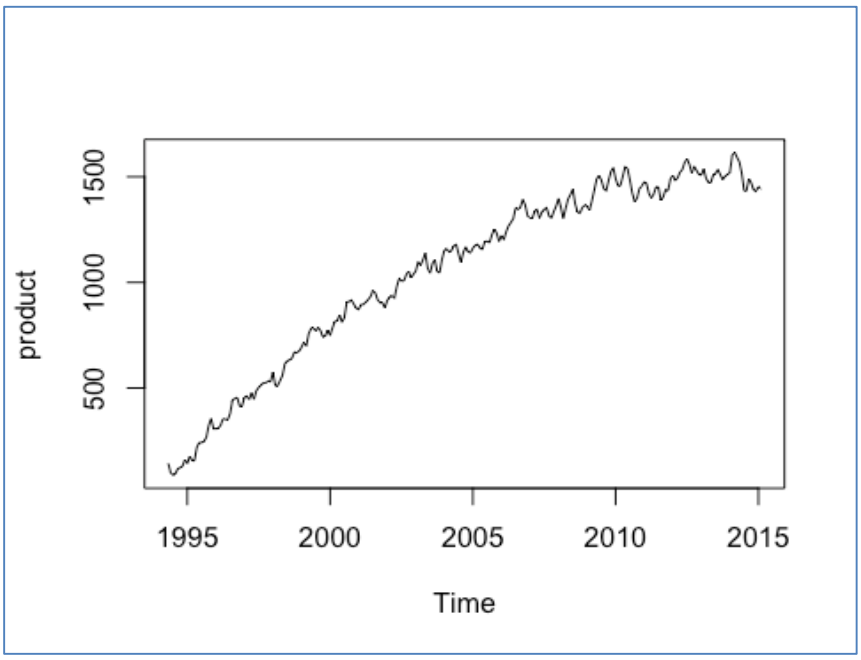
Bayesian Linear Regression in Time Series

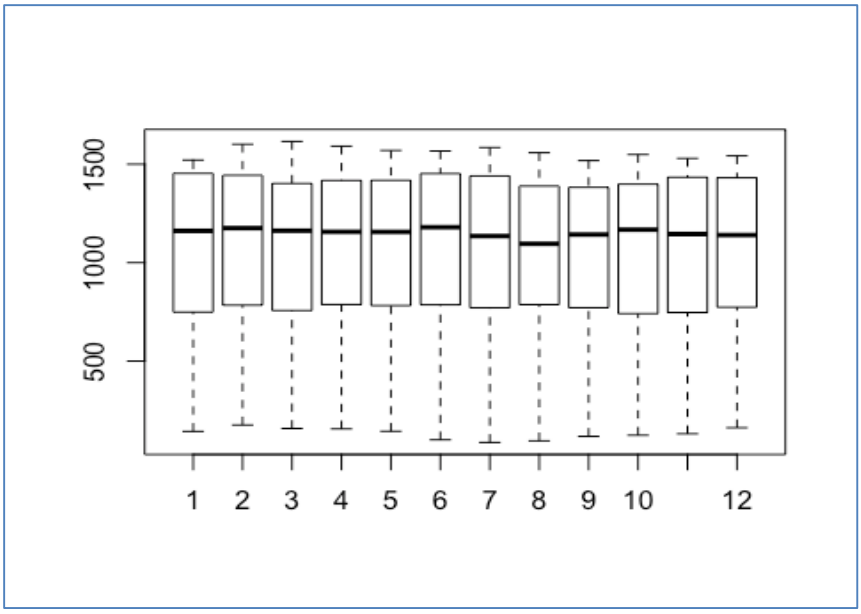
```
library(FinTS)
library(forecast)
library(MCMCpack)
library(broom)
library(lattice)
```

Loading the dataset

```
load("~/Library/Mobile Documents/com~apple~CloudDocs/Bentley/Fall 2017/MA611/
R/product.rdata")
plot(product)
```




```
boxplot(product~cycle(product))
```



We observe that the dataset appears to have trend and randomness. From boxplot we can conclude that the data does not have seasonality.

We now divide our dataset into training data and test data. We will build our model on training data and compare the predictions from this model with the actual data. Training dataset has the last 12 months value from the original dataset.

```
prdtr = window(product,start=c(1994,5),end=c(2014,2)) # training dataset. hen
ceforth called as data
prdtst=window(product,start=c(2014,3),end=c(2015,2))  # test dataset
prod.time.tr=time(prdtr)                              # extracting time compo
nent from data
prod.seas.tr=cycle(prdtr)                             # extracting seasonalit
y
prod.data.tr = coredata(prdtr)                        #extracting core data
```

From the time plot of the data we notice that there is a curve in the data and thus quadratic term may be required. Unlike normal regression (lm) the analysis pack we are using ony supports the linear regression model, so we have computed the quadratic component manually and used it in our model just as a linear one

```
prod.time.tr2=prod.time.tr^2
```

Now we build our model using MCMCregress which is part of MCMCpack. MCMC sampling requires priors on all parameters. However,we will employ weakly informative priors. Specifying 'uninformative' priors is always a bit of a balancing act. If the priors are too vague (wide) the MCMC sampler can wander off into nonscence areas of likelihood rather than concentrate around areas of highest likelihood (desired when wanting the outcomes to be largely driven by the data). On the other hand, if the priors are too strong, they may

have an influence on the parameters. In such a simple model, this balance is very forgiving - it is for more complex models that prior choice becomes more important.

```
prod.reg2.tr= MCMCregress(prod.data.tr~prod.time.tr+prod.time.tr2,burnin=1000
,thin =1)

summary(prod.reg2.tr)

##
## Iterations = 1001:11000
## Thinning interval = 1
## Number of chains = 1
## Sample size per chain = 10000
##
## 1. Empirical mean and standard deviation for each variable,
##    plus standard error of the mean:
##
##                     Mean        SD  Naive SE Time-series SE
## (Intercept)   -1.396e+07 3.481e+05 3.481e+03      3.469e+03
## prod.time.tr   1.386e+04 3.474e+02 3.474e+00      3.462e+00
## prod.time.tr2 -3.441e+00 8.666e-02 8.666e-04      8.636e-04
## sigma2         1.552e+03 1.455e+02 1.455e+00      1.478e+00
##
## 2. Quantiles for each variable:
##
##                     2.5%        25%        50%        75%      97.5%
## (Intercept)   -1.464e+07 -1.420e+07 -1.397e+07 -1.373e+07 -1.328e+07
## prod.time.tr   1.318e+04  1.363e+04  1.386e+04  1.410e+04  1.454e+04
## prod.time.tr2 -3.609e+00 -3.500e+00 -3.441e+00 -3.383e+00 -3.270e+00
## sigma2         1.290e+03  1.450e+03  1.544e+03  1.642e+03  1.866e+03


tidyMCMC(prod.reg2.tr) # point estimates

##            term      estimate    std.error
## 1   (Intercept) -1.396432e+07 3.481172e+05
## 2  prod.time.tr  1.386397e+04 3.473882e+02
## 3 prod.time.tr2 -3.440712e+00 8.666464e-02
## 4        sigma2  1.552244e+03 1.455024e+02

apply(prod.reg2.tr,2,quantile,probs=c(0.025,0.5,0.975)) # credible interval

##       (Intercept) prod.time.tr prod.time.tr2   sigma2
## 2.5%    -14637638     13180.76     -3.608517 1290.204
## 50%     -13965486     13864.93     -3.440908 1543.713
## 97.5%   -13279573     14536.53     -3.270241 1866.418
```

The summary output of MCMCregress looks somewhat similar to the output of lm() but there are some conceptual differences in the output
1. The output from MCMCregress speaks directly of what posteriors of the parameters are and does not refer to any hypothesis tests

2.Output shows the quantiles of exact posterior for each distribution
3.The output contains a $\sigma 2$ parameter which is the variance of distribution of y_i

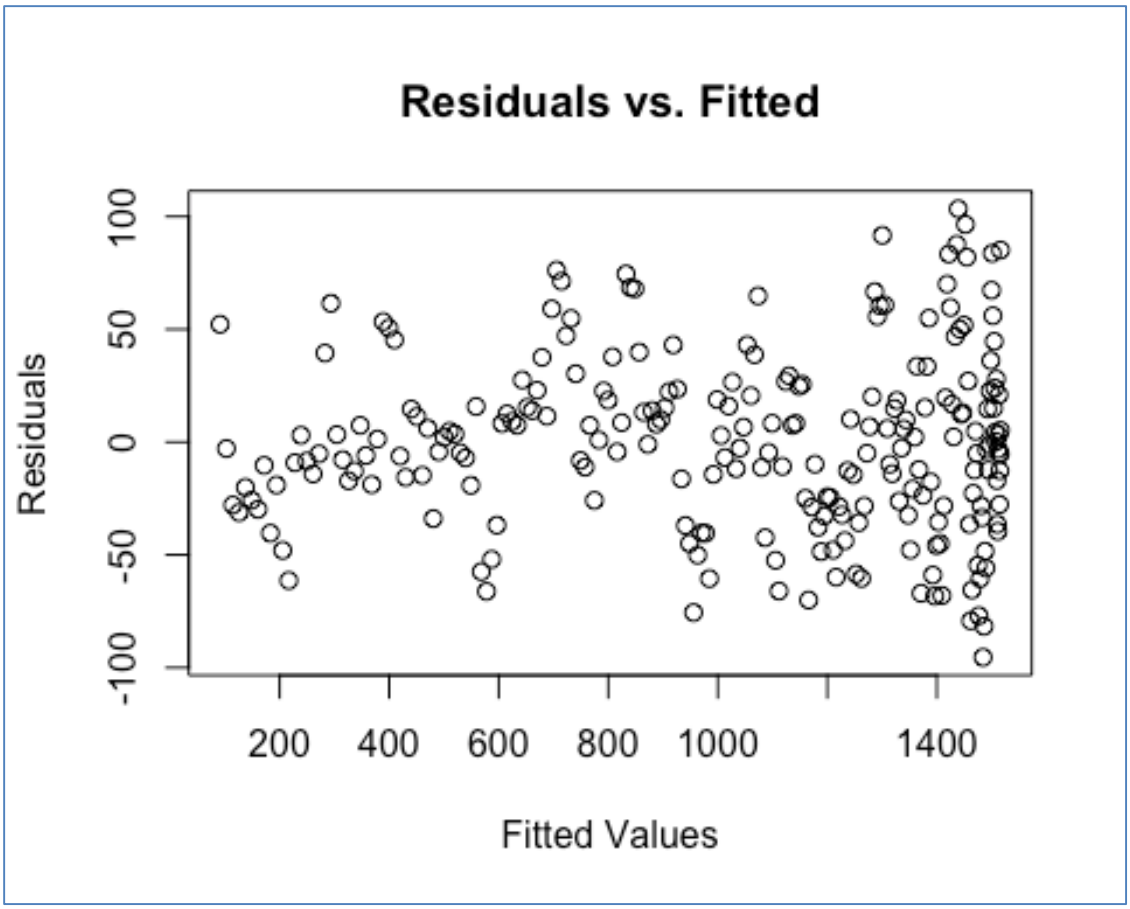
summary() output also provides two separate estimates of the standard error

- a naive estimate (the empirical standard deviation divided by the square root of the number of iterates) which assumes the sampled values to be independent
- a time-series estimate which gives the asymptotic standard error (the square root of the spectral density estimate divided by the sample size);

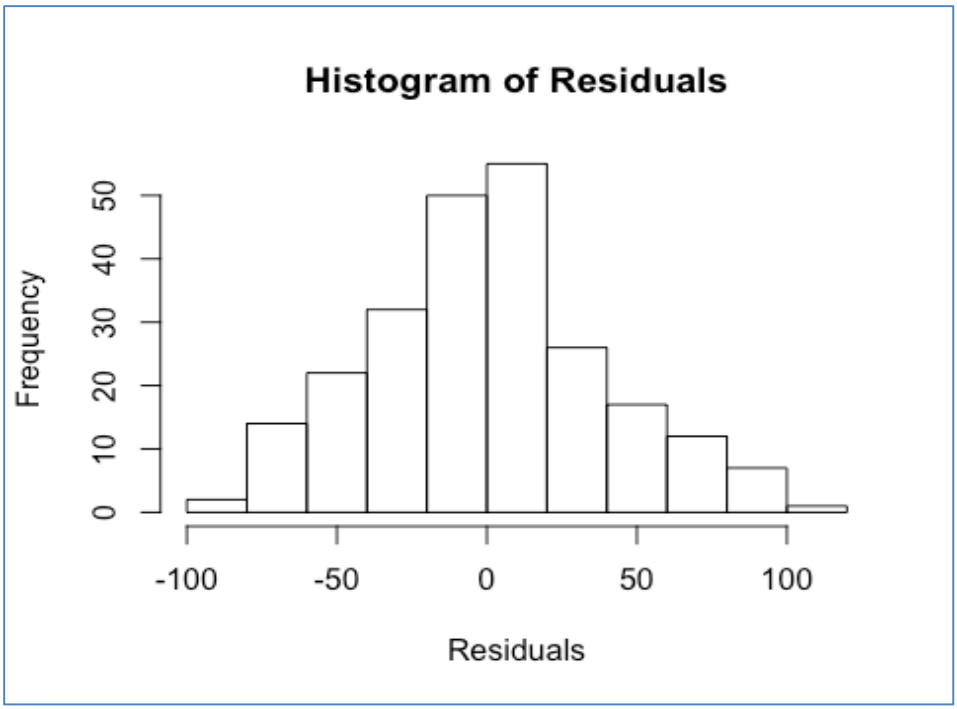
Diagnostics

Diagnostics of Regression

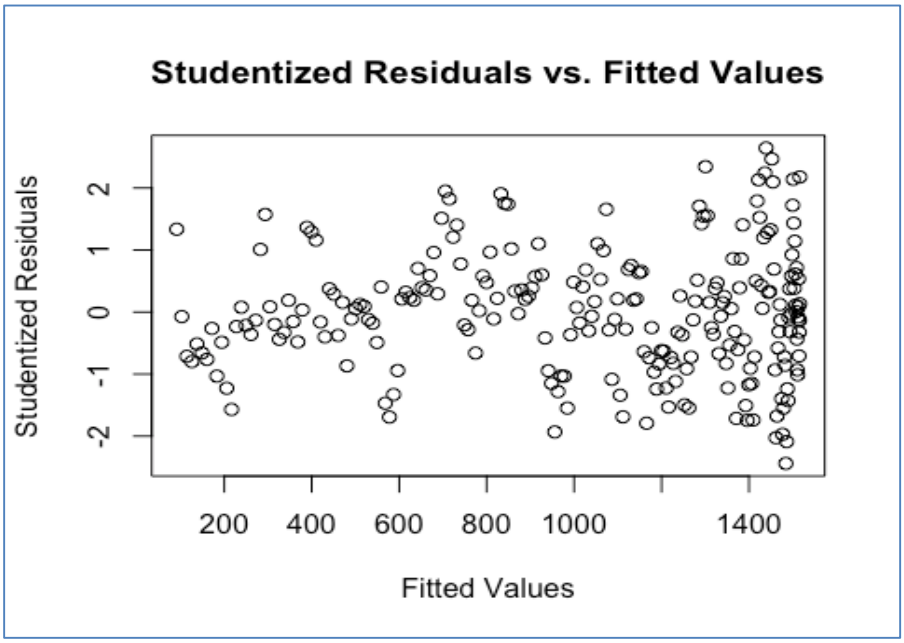
```
mymcmc=as.data.frame(prod.reg2.tr)
newdata=data.frame(x=prod.time.tr, x2=prod.time.tr2)
xmat=model.matrix(~x+x2,newdata)
coefs=apply(mymcmc[,1:3],2,mean)
fit=as.vector(coefs %*% t(xmat))
e=prod.data.tr-fit #residual
ste=e/sd(e)

plot(e~fit,xlab = "Fitted Values",ylab ="Residuals",main ="Residuals vs. Fitted")
```




```
hist(e,xlab="Residuals",ylab = "Frequency",main="Histogram of Residuals")
```



```
plot(ste~fit,xlab="Fitted Values",ylab = "Studentized Residuals",main = "Stud
entized Residuals vs. Fitted Values")
```

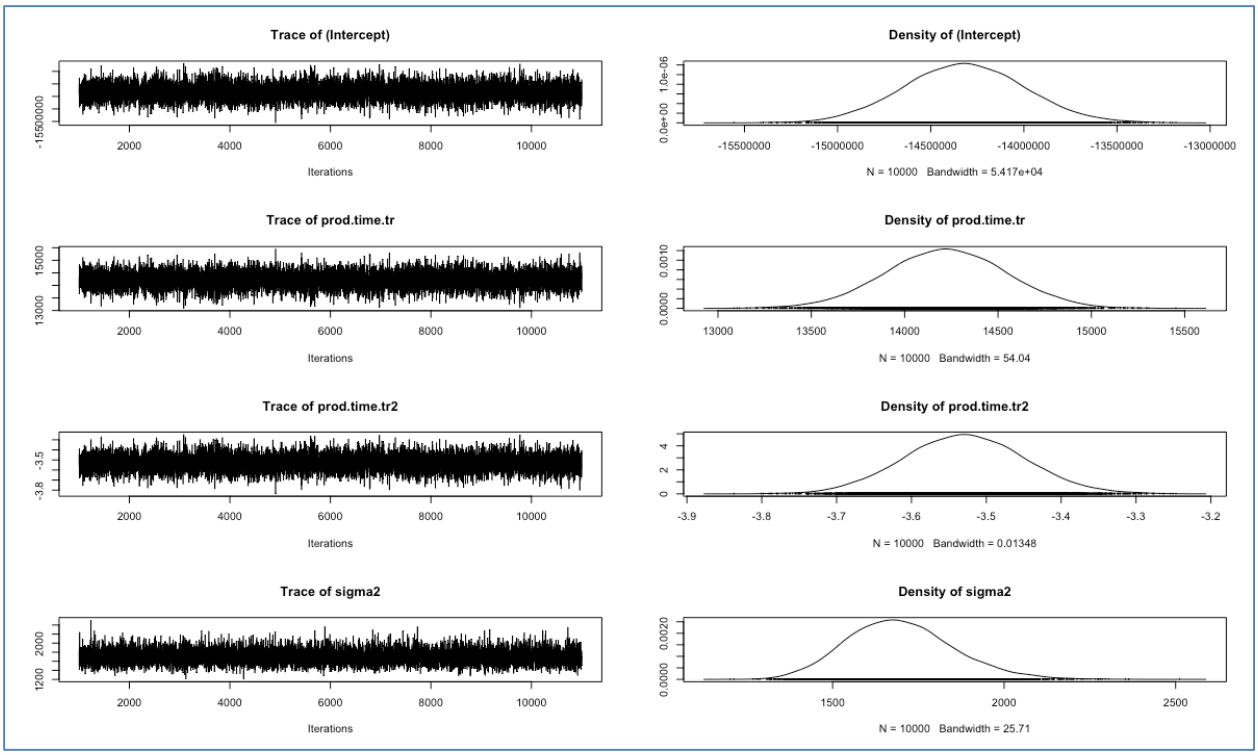


Diagnostics of MCMCregress

For Bayesian analyses, it is necessary to explore the characteristics of the MCMC chains and the sampler in general. Purpose of MCMC sampling is to replicate the posterior distribution of the model likelihood and priors by drawing a known number of samples from this posterior (thereby formulating a probability distribution). This is only reliable if the MCMC samples accurately reflect the posterior.One of the best ways of evaluating whether or not sufficient samples have been collected is with a trace plot.

Trace plots essentially display the iterative history of MCMC sequences. Ideally, the trace plot should show no trends, just consistent random noise around a stable baseline.

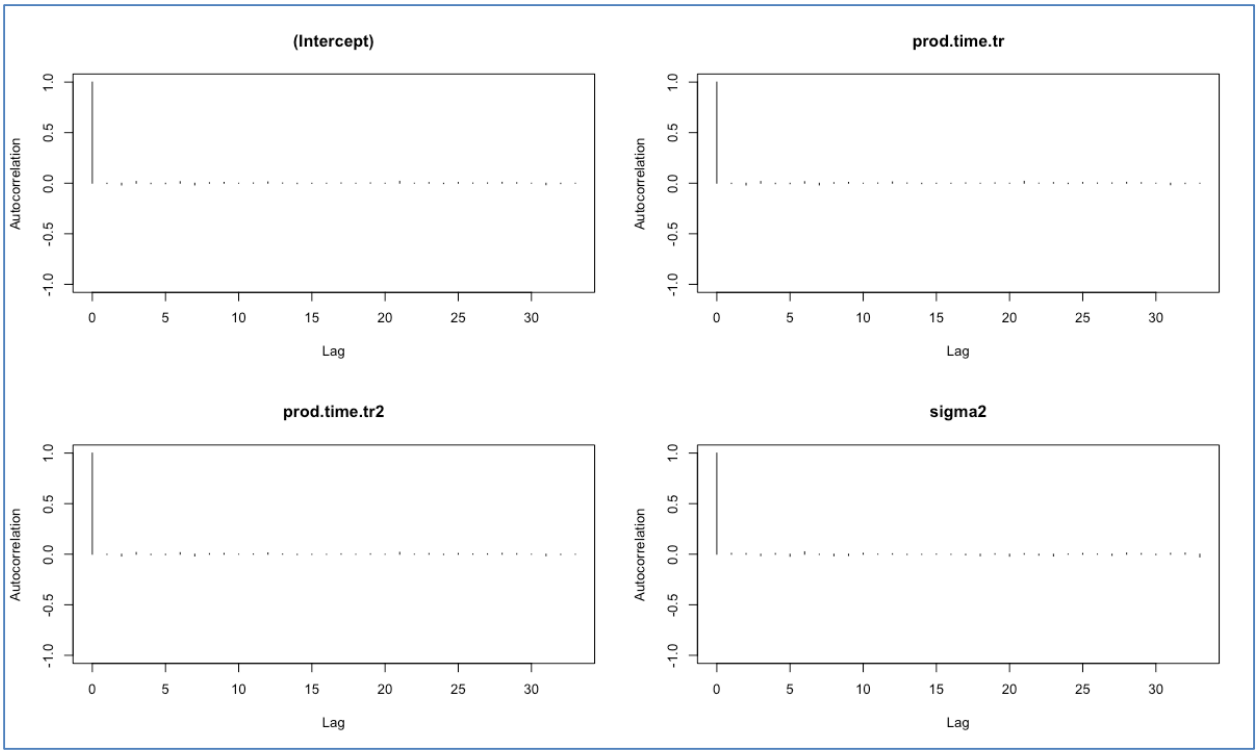
```
par(mar=c(1,1,1,1))
plot(prod.reg2.tr)
```



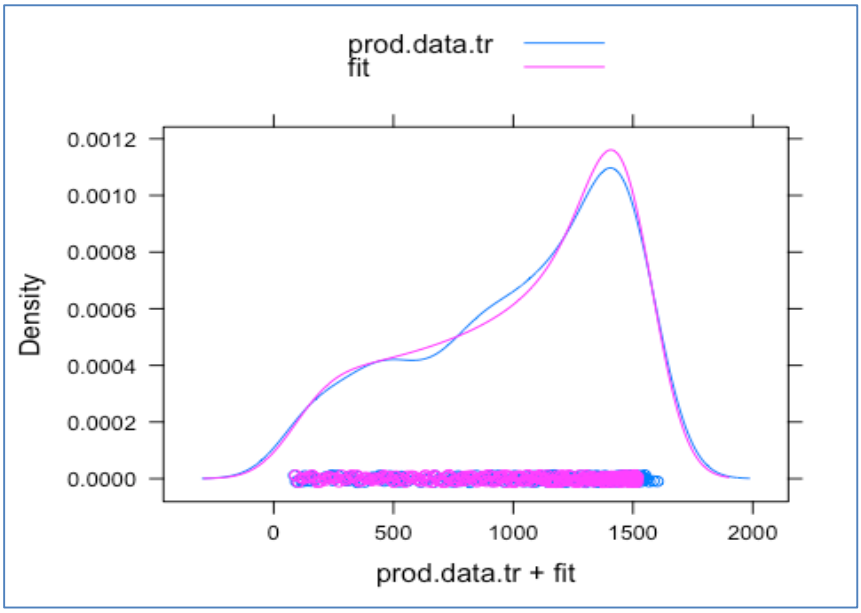

```
autocorr.diag(prod.reg2.tr)
```

```
##          (Intercept)  prod.time.tr prod.time.tr2       sigma2
## Lag 0   1.0000000000  1.0000000000  1.0000000000  1.000000000
## Lag 1  -0.0039416372 -0.0039127517 -0.0038839142  0.012408335
## Lag 5  -0.0037473233 -0.0037271260 -0.0037069236 -0.021101454
## Lag 10 -0.0008829189 -0.0008912236 -0.0008995335  0.011249974
## Lag 50  0.0033467409  0.0033643624  0.0033821145 -0.007218008
```

```
autocorr.plot(prod.reg2.tr,lag.max=25)
```




```
densityplot(~prod.data.tr+fit,auto.key = TRUE)
```



We now create a sequence of time interval for which prediction is to be made. We also manualy create the parameter of time^2 since we have used that as a predictor in our regression.

```
predtime = seq(2014.167,2015.0833,by=0.0833)
predtime2=predtime^2
```

Note that output of MCMCregress is not a model but the posterior values of predictors, thus we have to manually forecast using the coefficients from the output of MCMCregress function

```
forecastbayes=-1.396432e+07+1.386397e+04*predtime-3.440712*predtime2
forecastbayes

##  [1] 1514.031 1514.310 1514.541 1514.724 1514.860 1514.947 1514.987
##  [8] 1514.980 1514.924 1514.821 1514.670 1514.471
```

On the same dataset, we now perform a linear regression and make a prediction

```
lm.tr = lm(prod.data.tr~prod.time.tr+I(prod.time.tr^2))
lm.tr.predict = predict(lm.tr,data.frame(prod.time.tr = predtime))
lm.tr.predict

##        1        2        3        4        5        6        7        8
## 1516.663 1516.941 1517.171 1517.353 1517.488 1517.575 1517.614 1517.605
##        9       10       11       12
## 1517.549 1517.445 1517.293 1517.093
```

Now calculating the RMSE of both Bayesian Linear regression and linear regression

```
ebayes= forecastbayes-prdtst
elm = lm.tr.predict-prdtst
rmsebayes=sqrt(sum(ebayes^2)/length(ebayes))
rmselm=sqrt(sum(elm^2)/length(elm))
```

RMSE of Bayes Regression is: 68.579

RMSE of Linear Regression : 69.5248