ISyE6420 Bayesian Statistics
Project: Movie Recommendation System

Swati Gupta
sgupta612
GT ID : 903409986

April 22, 2019

Introduction

The aim of the project is to build a movie recommendation system. Based on a given set of ratings for movies by users, we want to predict a user's rating for a movie and provide recommendations based on that. To do so, we will use Bayesian techniques and Gibbs sampling.

Data

To build this model, we used a subset of the movie ratings data provided by GroupLens. The data was obtained from Kaggle. It contains close to 97000 ratings for over 6000 movies by more than 600 users. The data can be found here: `https://www.kaggle.com/rounakbanik/the-movies-dataset#ratings_small.csv`. The data was cleaned to include movies which had at least 2 ratings and users who had rated at least 2 movies.

Approach

The model we build will use collaborative filtering to predict the rating by a user for a movie. The principle behind collaborative filtering is that similar people will rate similar movies in a similar way. If A and B both like a movie X, and A also likes movie Y, B's likeliness towards Y would be closer to A's than any random user.

To evaluate similarity between users or movies, we introduce latent variables in our model. The latent variables will capture properties of a movie that makes a user like or dislike it. Some examples of such properties could be popularity of the cast, duration of the movie, amount of humor in the movie, etc.

Defining the model

The following variables are used in the model

- L: The number of latent factors
- U: The number of users
- M: The number of movies
- N: The number of observed ratings
- Y_{um}: The rating given to movie m by user u
- Y: The full vector of observed ratings
- γ_m: A vector of length $L+1$ for movie m. The first element $\gamma_m[0]$ is the bias for the movie. The remaining L elements, $\gamma_m[1:]$, are the latent factors associated with the movie
- Γ: $M \times (L+1)$ matrix where row m is γ_m
- θ_u: A vector of length $L+1$ for user u. The first element $\theta_u[0]$ is the bias for the user. The remaining L elements, $\theta_u[1:]$, are user's preferences for the latent factors
- Θ: $U \times (L+1)$ matrix where row u is θ_u
- μ: The overall mean of ratings
- σ^2: The residual variance of ratings

We define the following model to predict the ratings

$$\begin{aligned}
Y_{um} &= \mu + \theta_u[0] + \gamma_m[0] + \theta_u[1:]^T \gamma_m[1:] + \epsilon_{um} \\
\epsilon_{um} &\sim N(0, \sigma^2) \\
p(Y_{um}|\mu, \sigma^2, \theta_u, \gamma_m) &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(- \frac{(y_{um} - (\mu + \theta_u[0] + \gamma_m[0] + \theta_u[1:]^T \gamma_m[1:]))^2}{2\sigma^2} \right)
\end{aligned}$$

Priors

The following priors are assumed for the parameters

$$\begin{aligned}
p(\mu) &\propto 1 \\
p(\sigma^2) &\propto \frac{1}{\sigma^2} \\
p(\gamma_m) &= N(0, \Sigma_\gamma) \\
p(\theta_u) &= N(0, \Sigma_\theta)
\end{aligned}$$

where Σ_γ, Σ_θ are $L+1 \times L+1$ covariance matrices. They are assumed to be diagonal matrices with all diagonal elements equal to λ_γ and λ_θ respectively both assumed to be 1.

Conditionals

$$p(Y,\mu,\sigma^2,\Theta,\Gamma) = \prod_{i=1}^{N} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\Bigg(- \frac{(y_i - (\mu + \theta_{u_i}[0] + \gamma_{m_i}[0] + \theta_{u_i}[1:]^T \gamma_{m_i}[1:]))^2}{2\sigma^2} \Bigg)$$
$$\times \frac{1}{\sqrt{(2\pi)^n |\Sigma_\gamma|}} \exp\Bigg(- \frac{1}{2}\gamma_m^T \Sigma_\gamma^{-1} \gamma_m \Bigg) \frac{1}{\sqrt{(2\pi)^n |\Sigma_\theta|}} \exp\Bigg(- \frac{1}{2}\theta_u^T \Sigma_\theta^{-1} \theta_u \Bigg) \frac{1}{\sigma^2}$$

Conditional for μ

$$p(\mu|Y,\sigma^2,\Theta,\Gamma) = N(\mu_n, \sigma_n^2)$$
$$\text{where } \mu_n = \frac{1}{N}\sum_{i=1}^{N}(y_i - (\theta_{u_i}[0] + \gamma_{m_i}[0] + \theta_{u_i}[1:]^T \gamma_{m_i}[1:]))$$
$$\sigma_n^2 = \frac{\sigma^2}{N}$$

Conditional for σ^2

$$p(\sigma^2|Y,\mu,\Theta,\Gamma) \propto (\sigma^2)^{-\frac{n}{2}-1} \exp\Bigg(- \frac{1}{2\sigma^2}\sum_{i=1}^{N}(y_i - (\mu + \theta_{u_i}[0] + \gamma_{m_i}[0] + \theta_{u_i}[1:]^T \gamma_{m_i}[1:]))^2 \Bigg)$$
$$p(\sigma^2|Y,\mu,\Theta,\Gamma) = IG(\alpha_n, \beta_n)$$
$$\text{where } \alpha_n = \frac{N}{2}$$
$$\beta_n = \frac{\sum_{i=1}^{N}(y_i - (\mu + \theta_{u_i}[0] + \gamma_{m_i}[0] + \theta_{u_i}[1:]^T \gamma_{m_i}[1:]))^2}{2}$$

Conditional for $\gamma_{m'}$

The term $\gamma_{m'}$ appears only in the terms where a user rated the movie m' or in the prior itself.

$$Y_{um'} = \mu + \theta_u[0] + \gamma_{m'}[0] + \theta_u[1:]^T \gamma_{m'}[1:] + \epsilon_{um'}$$

We can rewrite the model as

$$Y_{um'} - \mu - \theta_u[0] = \gamma_{m'}[0] + \theta_u[1:]^T \gamma_{m'}[1:] + \epsilon_{um'}$$

Assuming we have g users who rated the movie m', we can treat this as a linear regression

$$Y_{m'} = \gamma_{m'} X_{m'}$$

where

$$Y_{m'} = \begin{bmatrix} Y_{u_1 m'} - \mu - \theta_{u_1}[0] \\ Y_{u_2 m'} - \mu - \theta_{u_2}[0] \\ \vdots \\ Y_{u_g m'} - \mu - \theta_{u_g}[0] \end{bmatrix}, X_{m'} = \begin{bmatrix} 1, \theta_{u_1}[1:]^T \\ 1, \theta_{u_2}[1:]^T \\ \vdots \\ 1, \theta_{u_g}[1:]^T \end{bmatrix}$$

Taking into account the prior on $\gamma_{m'}$, we get

$$
\begin{aligned}
p(\gamma_{m'}|Y,\mu,\sigma^2,\Theta,\Gamma_{-m'}) &= N(\mu_{m'},\Sigma_{m'}) \\
\text{where } \mu_{m'} &= \Sigma_{m'}\frac{1}{\sigma^2}X_{m'}^T Y_{m'} \\
\Sigma_{m'}^{-1} &= \frac{1}{\sigma^2}X_{m'}^T X_{m'} + \Sigma_{\gamma}^{-1}
\end{aligned}
$$

Conditional for $\theta_{u'}$

The term $\theta_{u'}$ appears only in the terms where a movie is rated by user u' or in the prior itself.

$$Y_{u'm} = \mu + \theta_{u'}[0] + \gamma_m[0] + \theta_{u'}[1:]^T\gamma_m[1:] + \epsilon_{u'm}$$

We can rewrite the model as

$$Y_{u'm} - \mu - \gamma_m[0] = \theta_{u'}[0] + \theta_{u'}[1:]^T\gamma_m[1:] + \epsilon_{u'm}$$

Assuming we have g movies that were rated by the user u', we can treat this as a linear regression

$$Y_{u'} = \theta_{u'}X_{u'}$$

where

$$Y_{u'} = \begin{bmatrix} Y_{u'm_1} - \mu - \gamma_m[0] \\ Y_{u'm_2} - \mu - \gamma_m[0] \\ \vdots \\ Y_{u'm_g} - \mu - \gamma_m[0] \end{bmatrix}, X_{m'} = \begin{bmatrix} 1, \gamma_{m_1}[1:]^T \\ 1, \gamma_{m_2}[1:]^T \\ \vdots \\ 1, \gamma_{m_g}[1:]^T \end{bmatrix}$$

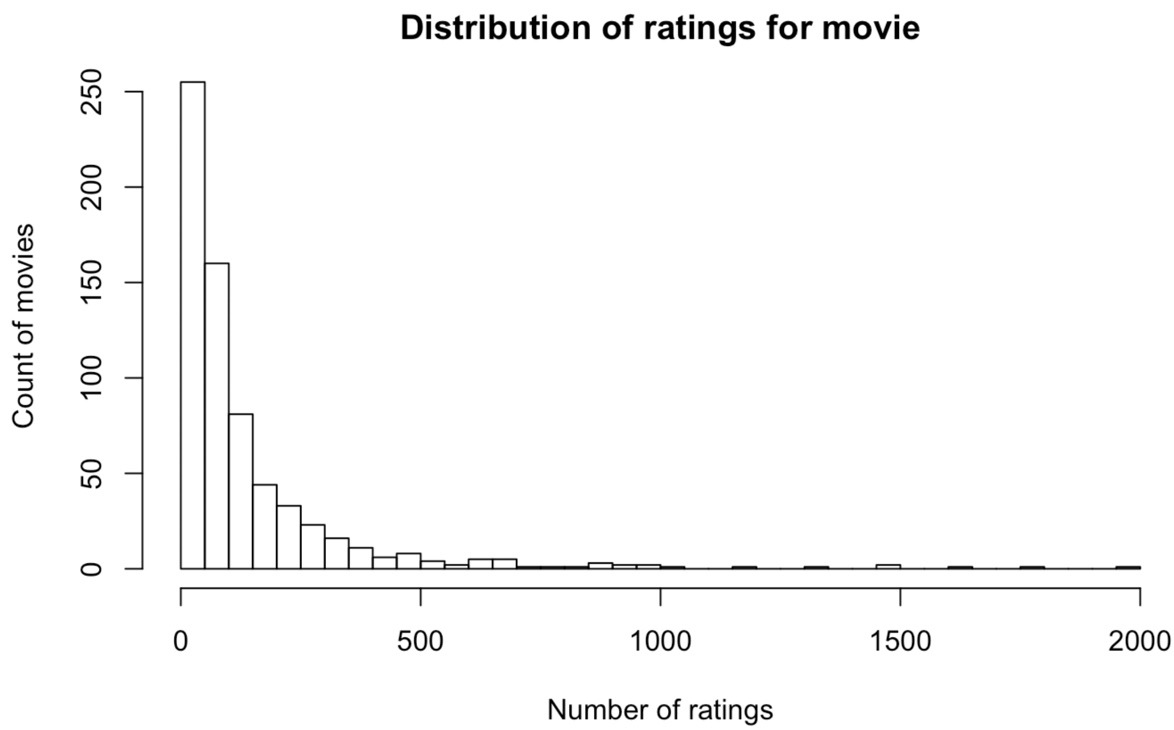
Taking into account the prior on $\theta_{u'}$, we get

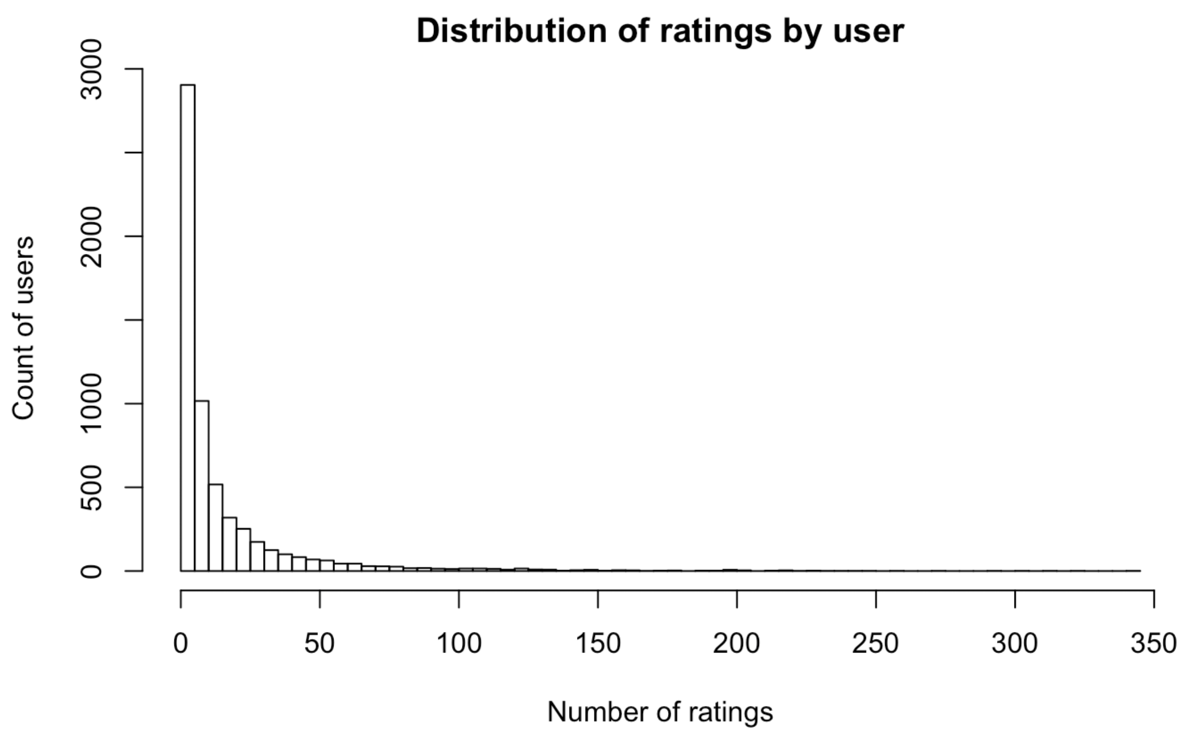
$$
\begin{aligned}
p(\theta_{u'}|Y,\mu,\sigma^2,\Theta_{-u'},\Gamma) &= N(\mu_{u'},\Sigma_{u'}) \\
\text{where } \mu_{u'} &= \Sigma_{u'}\frac{1}{\sigma^2}X_{u'}^T Y_{u'} \\
\Sigma_{u'}^{-1} &= \frac{1}{\sigma^2}X_{u'}^T X_{u'} + \Sigma_{\theta}^{-1}
\end{aligned}
$$

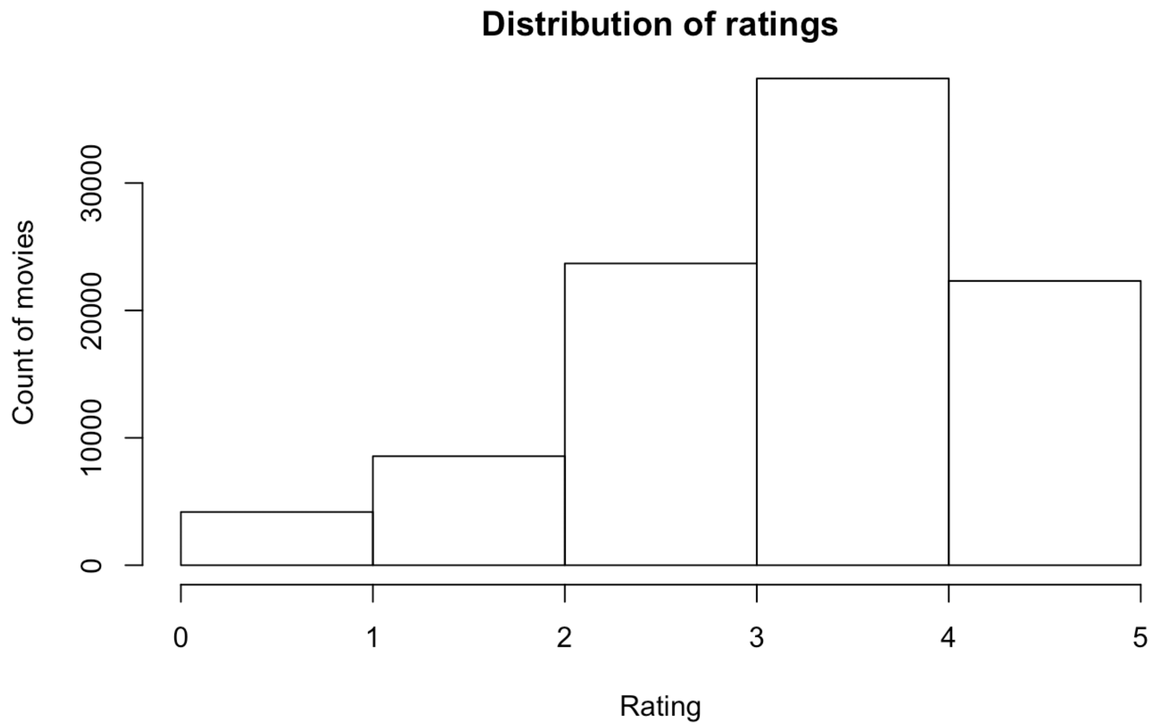
We have $(U+M)\times(L+1)+2$ parameters to estimate. We build a Gibbs sampler to estimate these parameters.

Data Exploration

We do a basic data exploration and see the distribution of number of ratings by a user, number of ratings for a movie and the rating distribution itself.


Distribution of ratings for movie
Count of movies
0
50
100
150
200
250
0
500
1000
1500
2000
Number of ratings


Distribution of ratings by user
Count of users
0
500
1000
2000
3000
0
50
100
150
200
250
300
350
Number of ratings



To evaluate the results of our model, we divide our data into training, validation and testing data. About 70% is kept as training while the rest is equally divided in validation and testing. While sampling data for training, we made sure that all movies and users that appear in test or validation set have at least two entries in the training data.

Gibbs Sampling

We now set up Gibbs sampling to evaluate parameters based on our training data. The following values were chosen as the initial values:

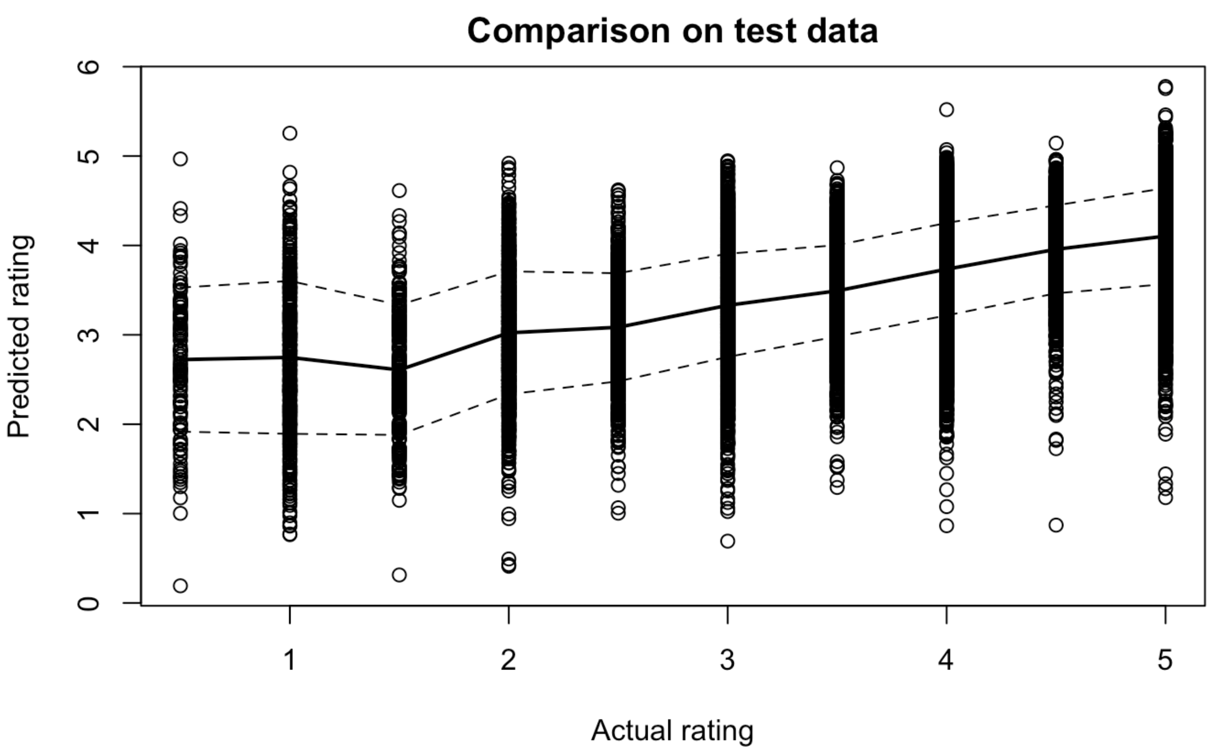
- μ: mean of all ratings in training data
- σ^2: variance of all ratings in training data
- γ_m: $\gamma_m[0] = \mu_m - \mu$ where μ_m is the mean ratings of movie m. $\gamma_m[1:] = 0$
- θ_u: $\theta_u[0] = \mu_u - \mu$ where μ_u is the mean ratings by user u. $\theta_u[1:] = 0$

After removing the burn in samples, the average across all remaining samples was taken to get an estimate of the parameters. These parameters were then used to predict the ratings in validation set. The graphs and details of different trials and their results are included in the Appendix

Results

Case	L	Samples	Burn in	Training error (RMSE)	Validation error (RMSE)
1	4	500	200	0.66	0.87
2	10	1000	200	0.58	0.88
3	20	1500	1000	0.49	0.89
4	2	5000	4000	0.71	0.88

Based on the results we go with model 4. While the error on validation set is not the least, the value of μ had converged for model 4. Running it on the test set we get the following results:



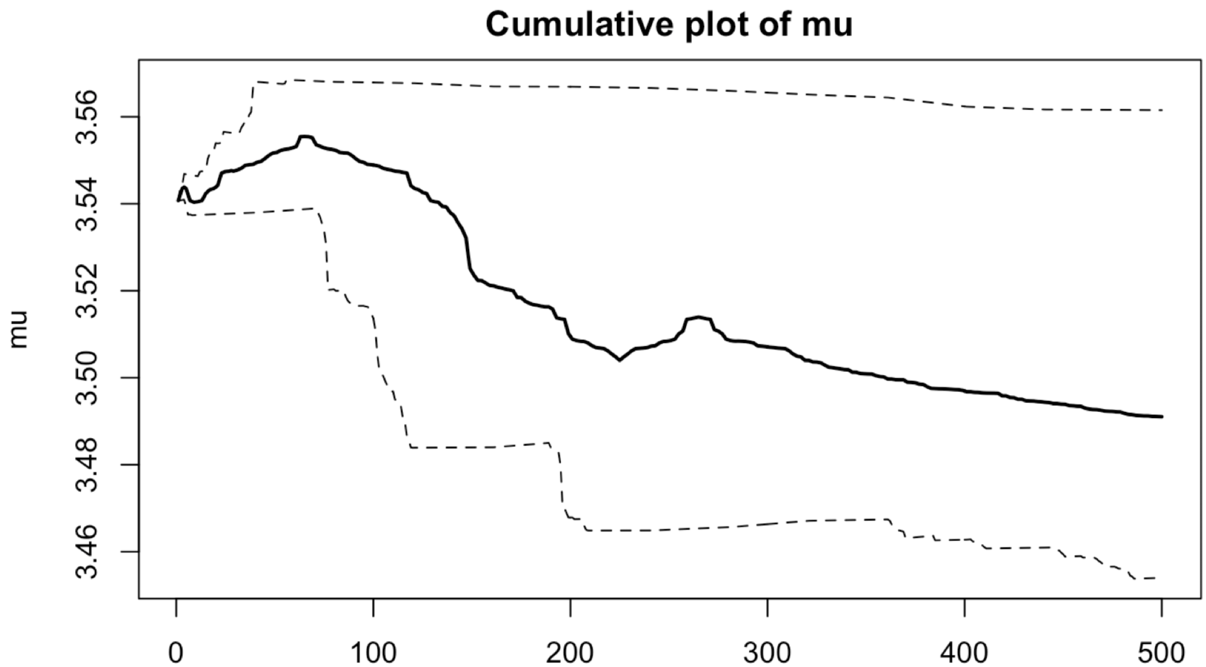
The RMSE on test data was 0.86. The values of μ and σ^2 were 3.46 and 0.6 respectively.

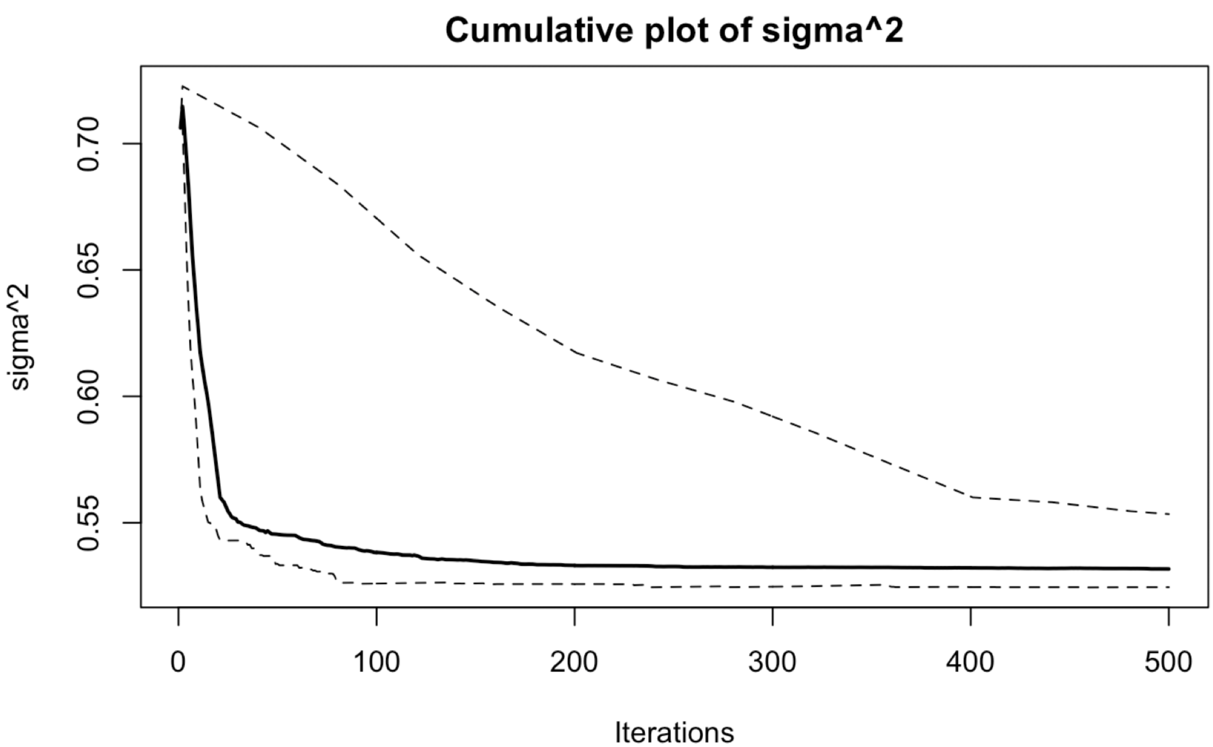
We see that μ does not converge for lower sample sizes. Plotting the predicted data of validation and test set suggests presence of a bias. Running the model for more samples with a higher value of L could result in better accuracy.

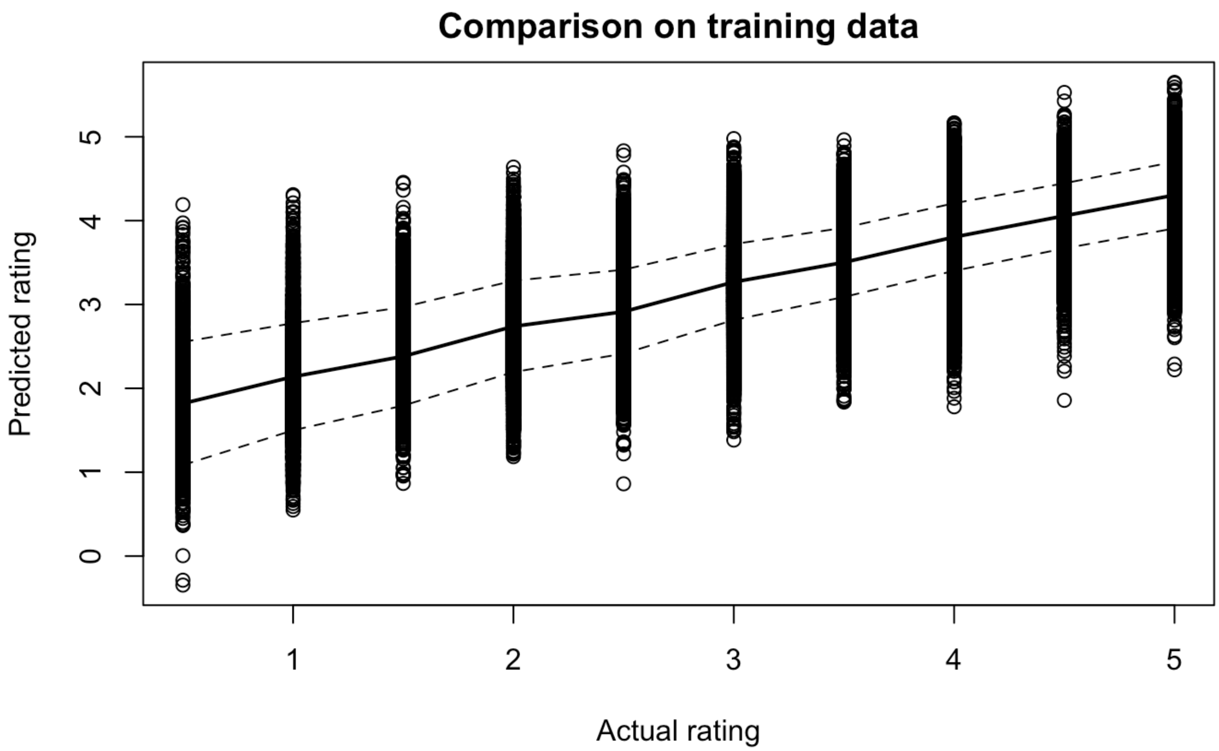
Appendix 1

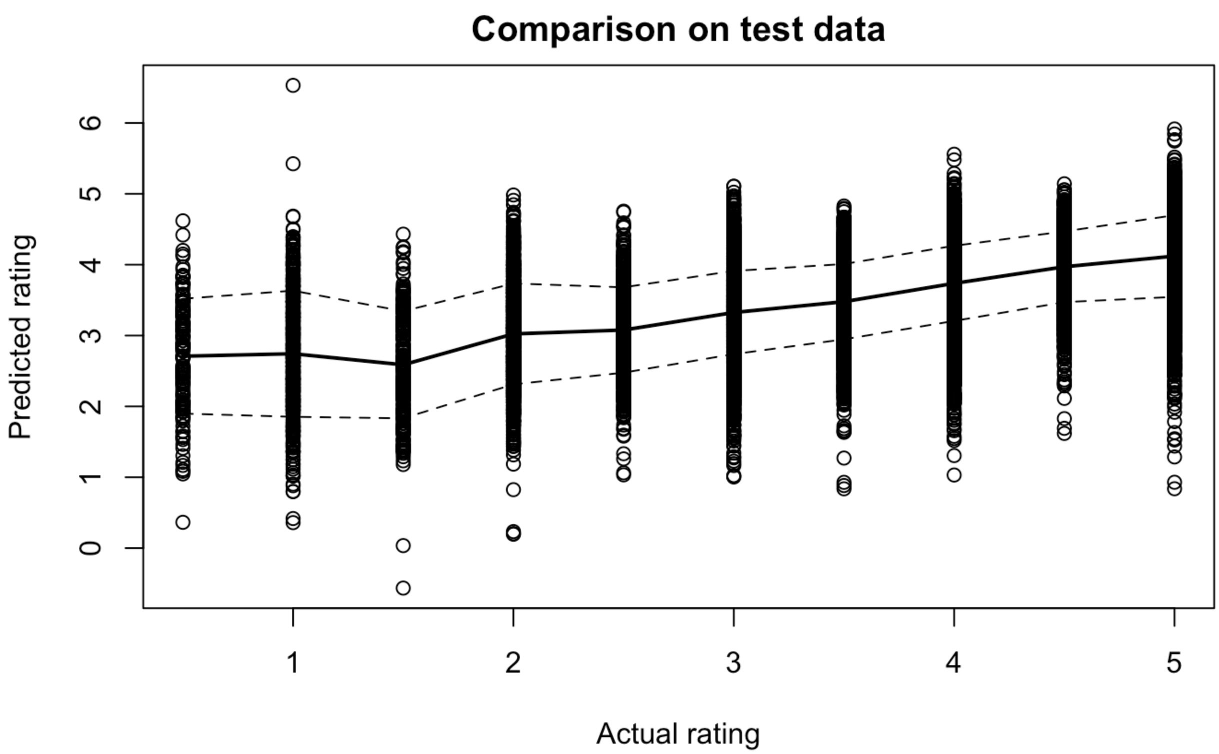
Case 1

$L = 4$, number of samples = 500, burn in = 200





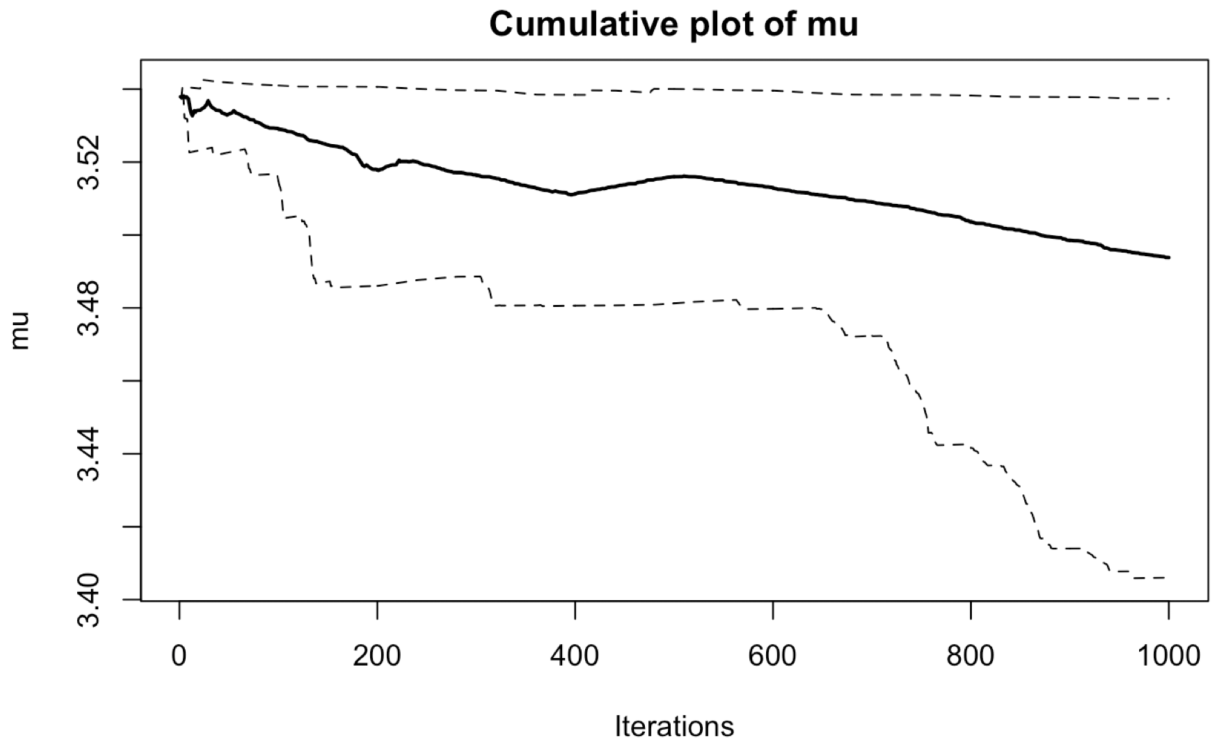


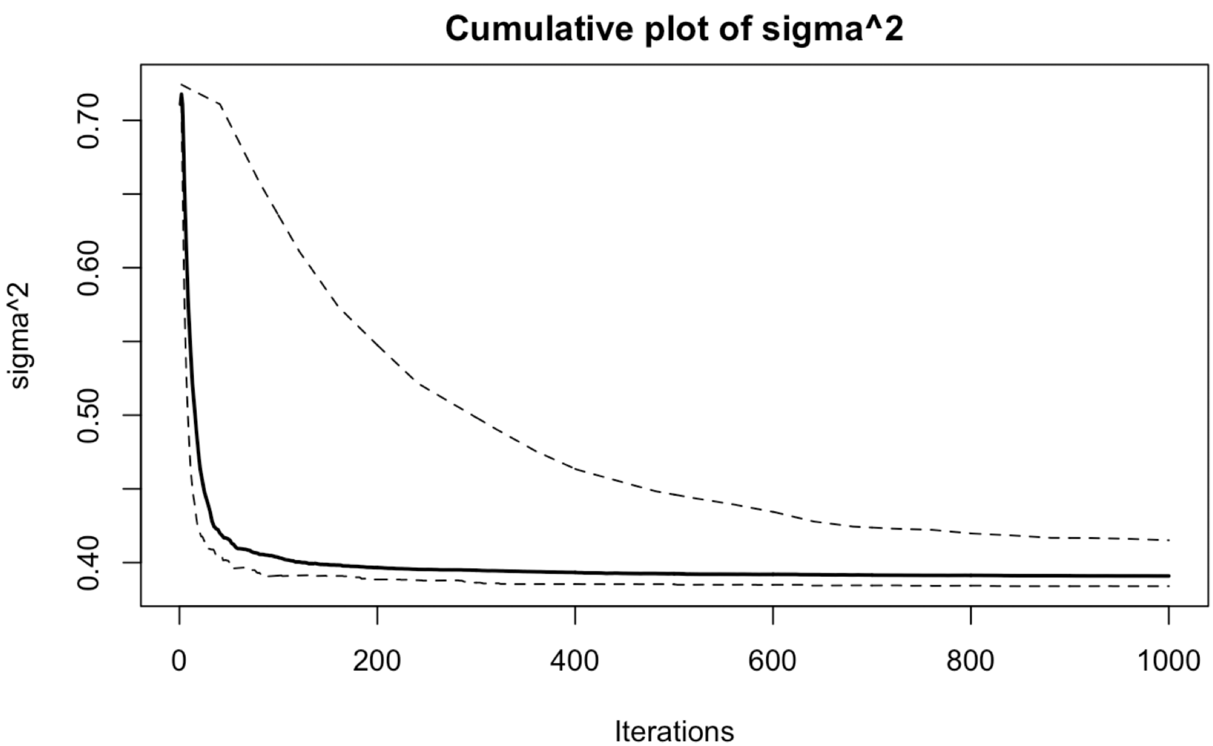


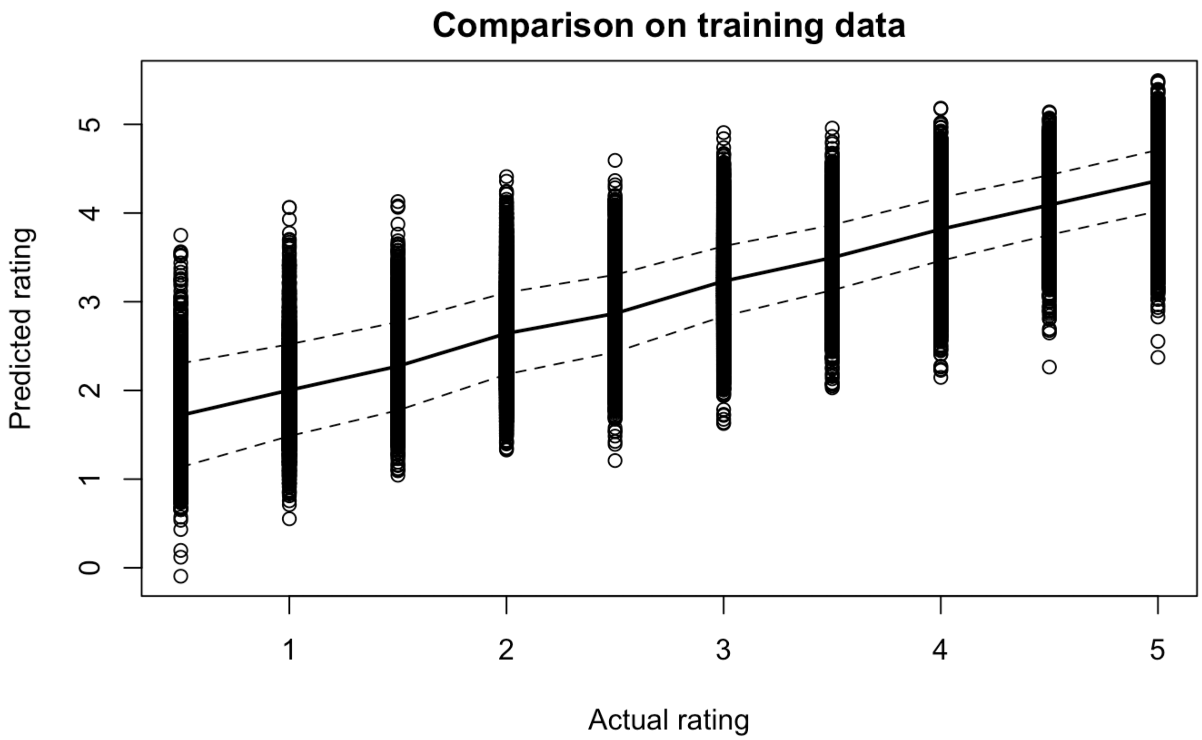
The solid line is the man of predicted ratings and the dotted lines show the range between one standard deviation from the mean

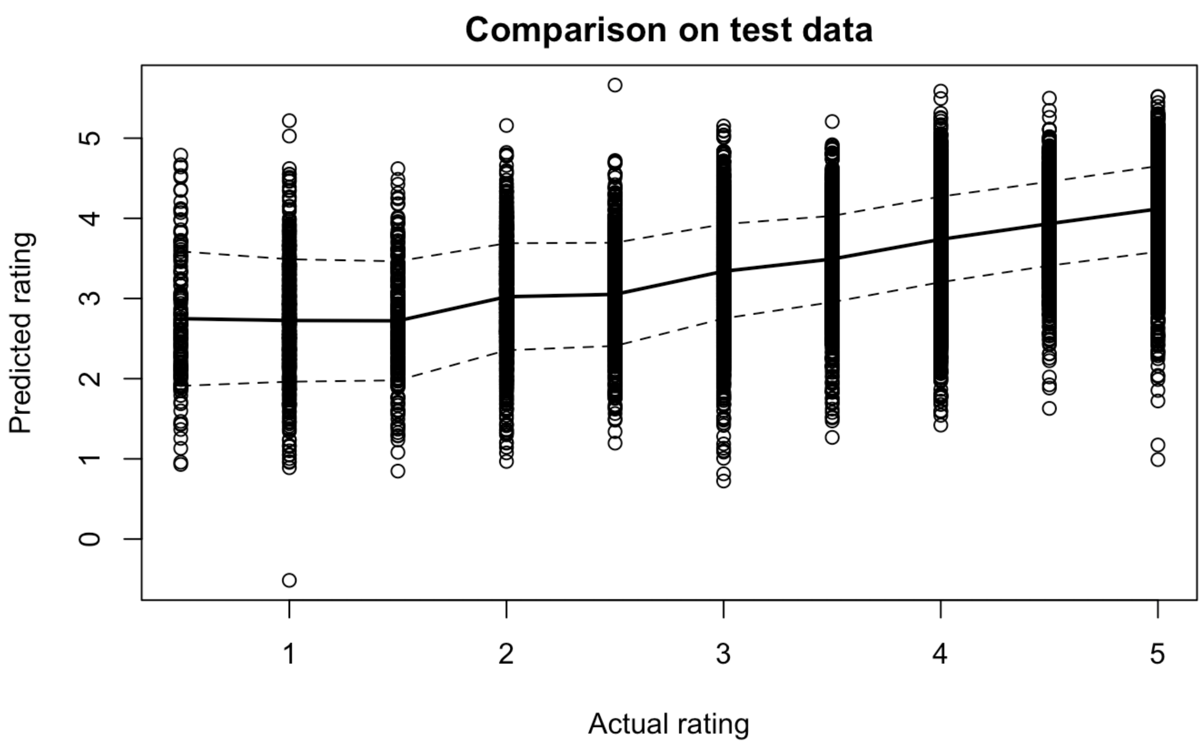
Case 2

$L = 10$, number of samples = 1000, burn in = 200



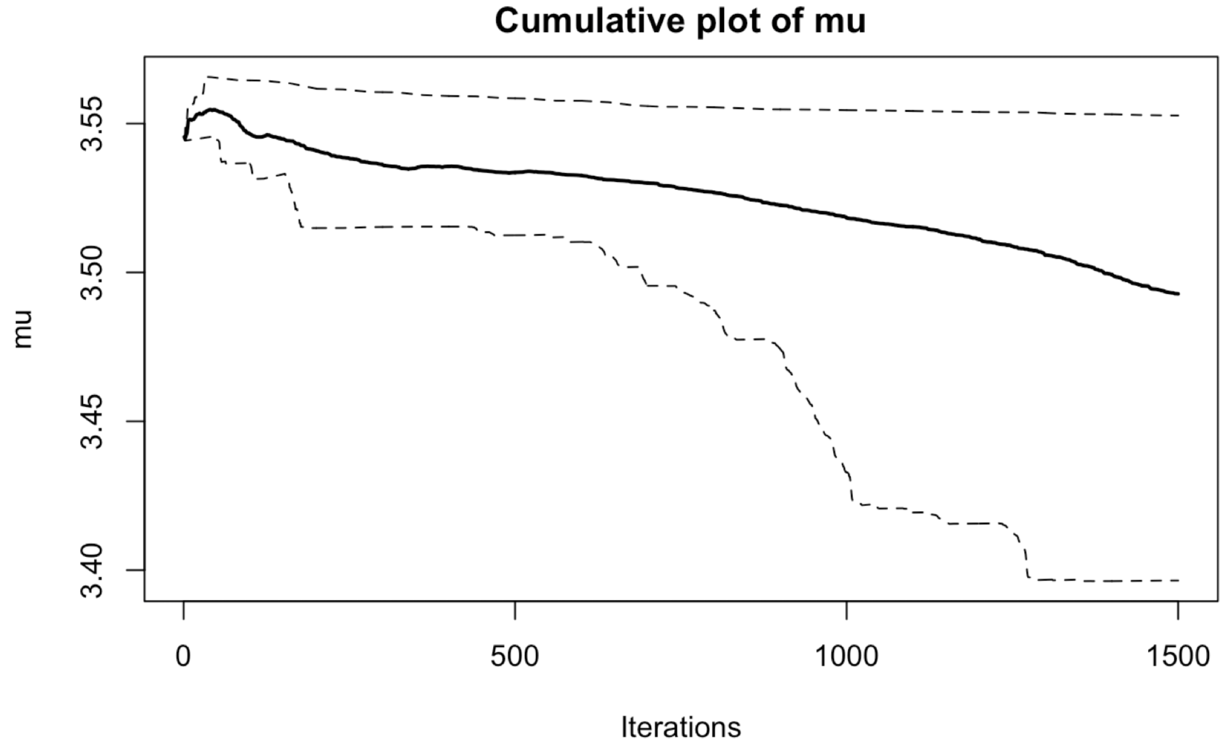


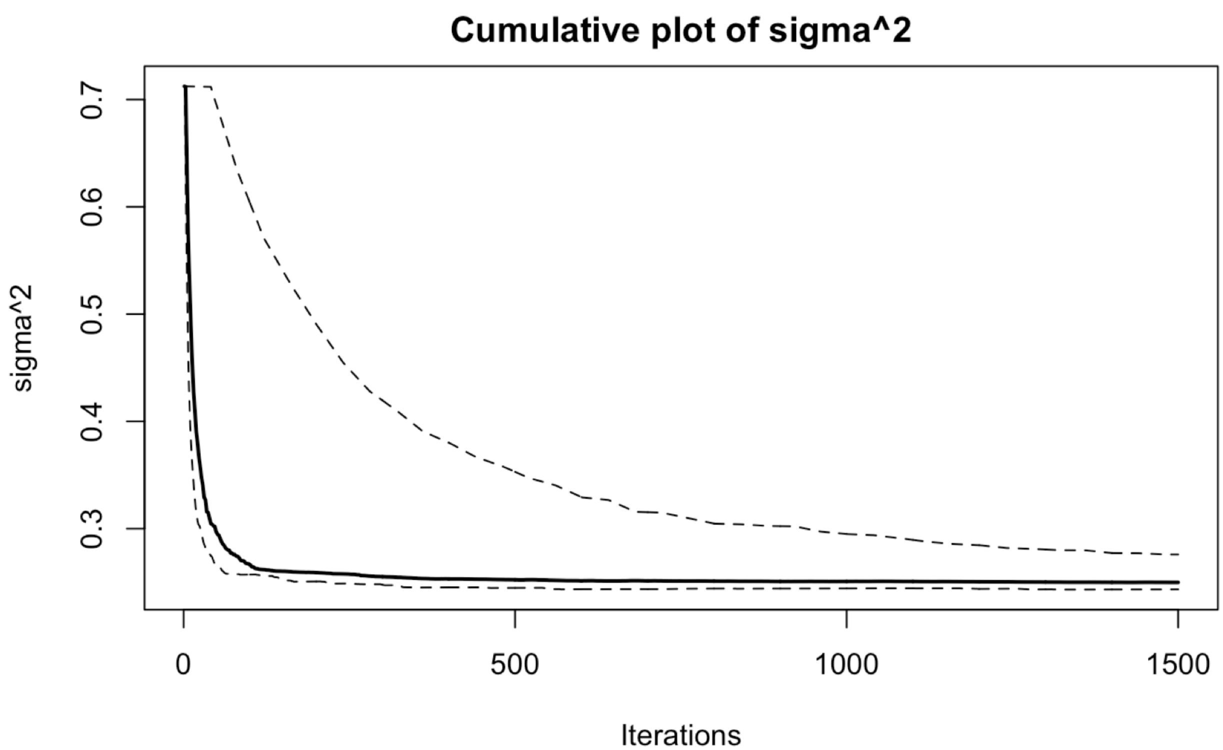

Comparison on training data
Predicted rating
Actual rating
0
1
2
3
4
5
1
2
3
4
5

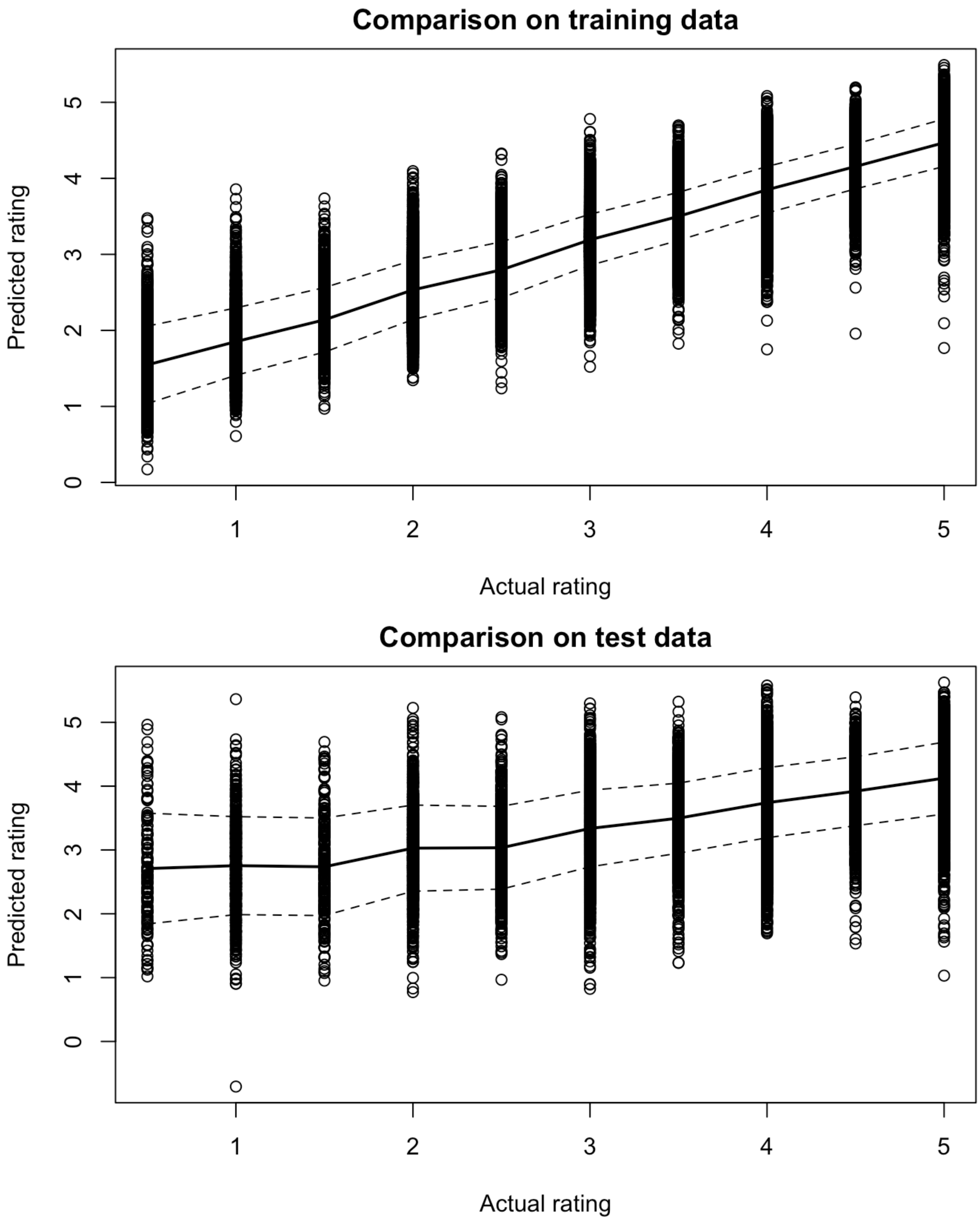

Comparison on test data
Predicted rating
Actual rating
0
1
2
3
4
5
1
2
3
4
5

Case 3

$L = 20$, number of samples = 1500, burn in = 1000

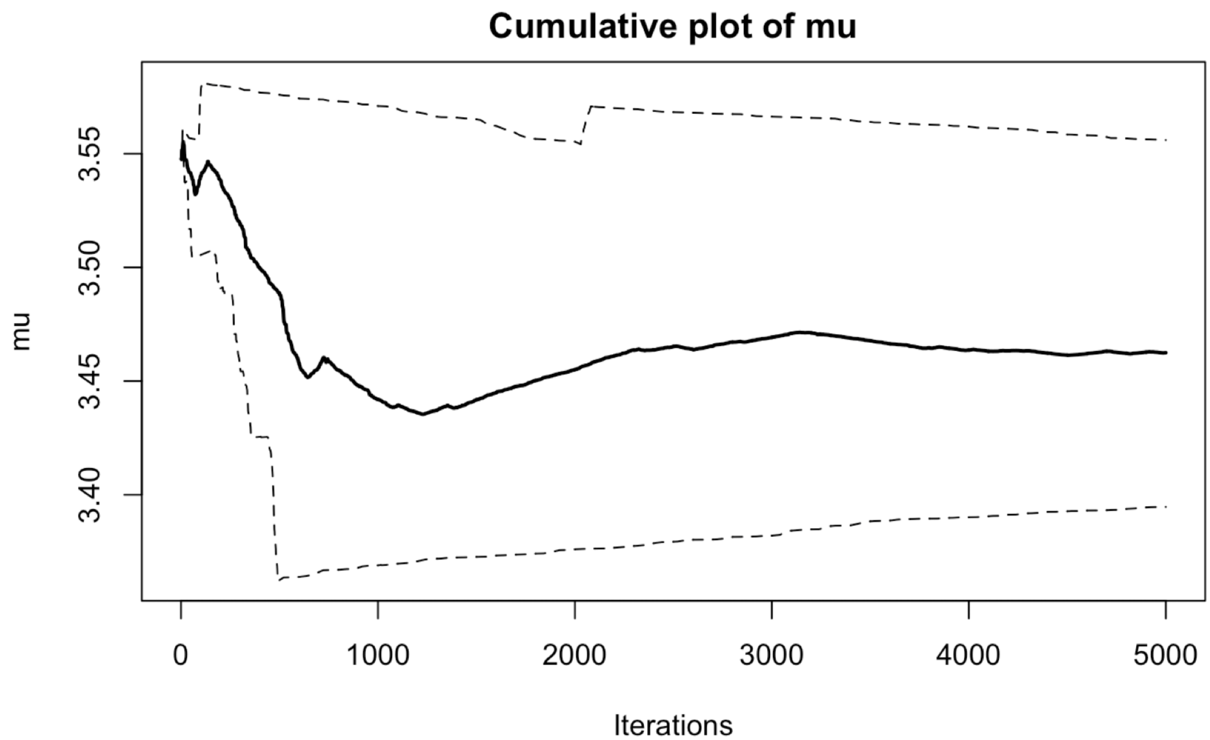


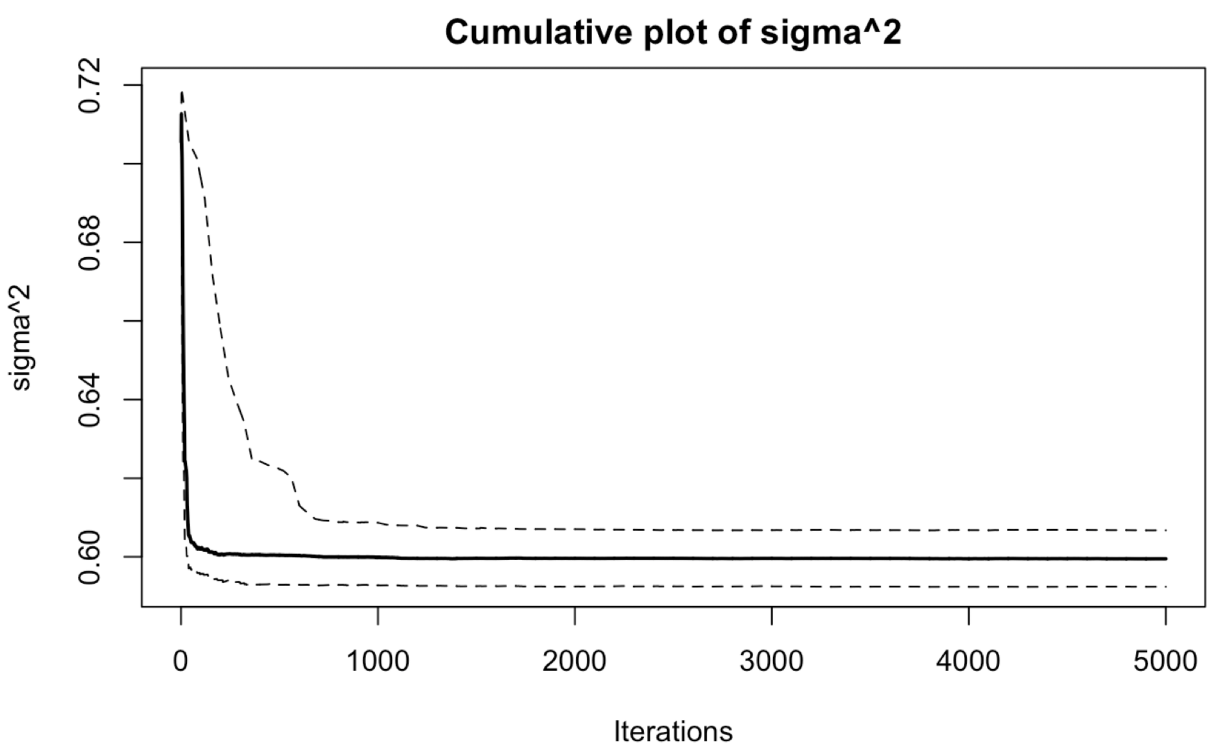


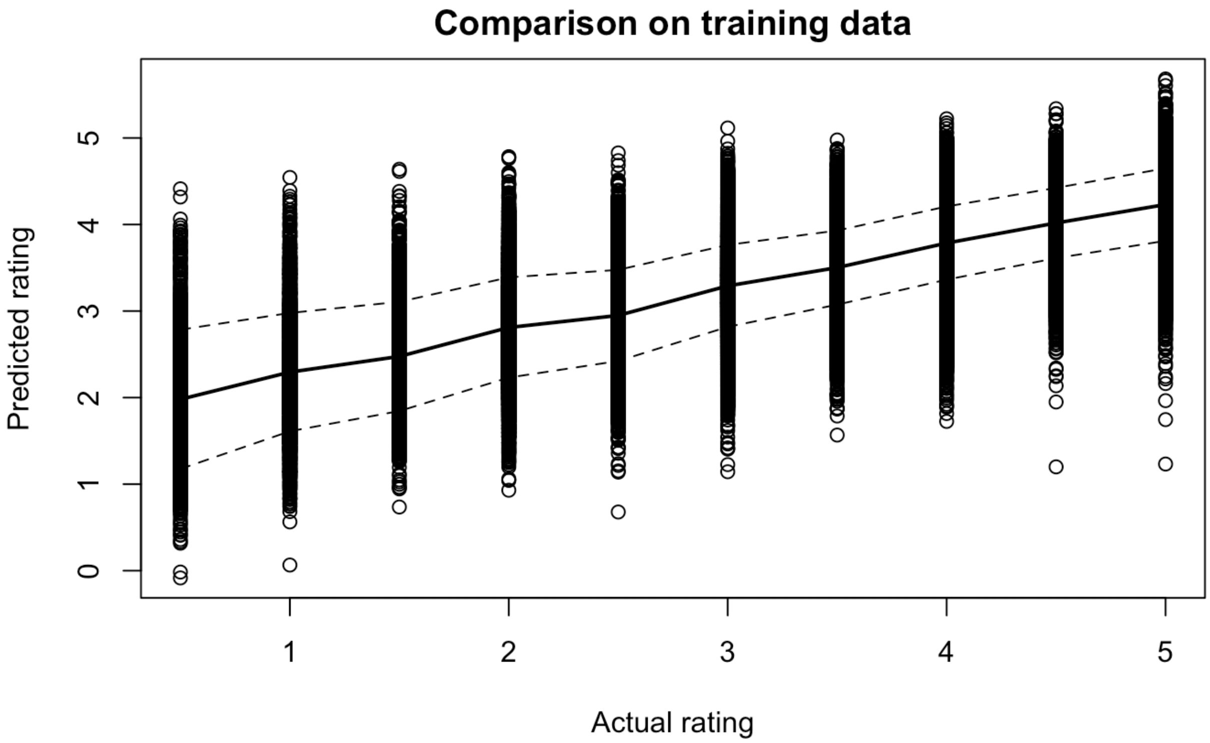

Comparison on training data
Predicted rating
Actual rating
0
1
2
3
4
5
1
2
3
4
5
Comparison on test data
Predicted rating
Actual rating
0
1
2
3
4
5
1
2
3
4
5

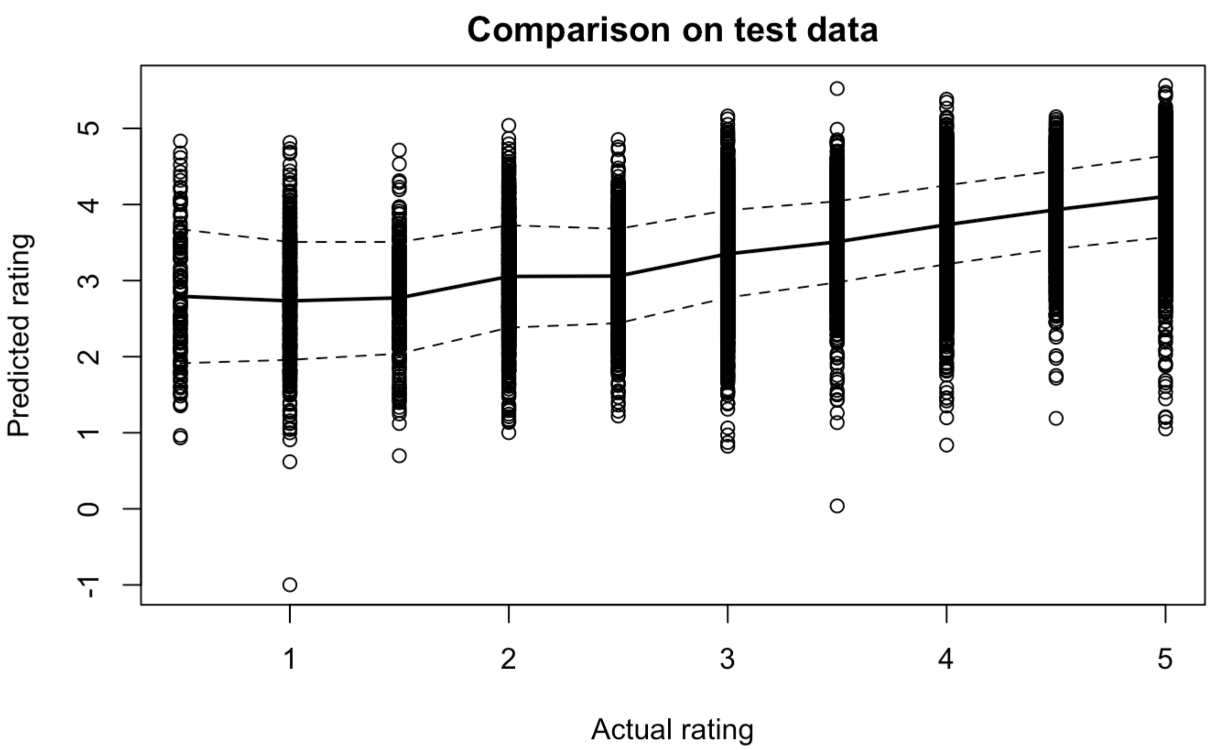
Case 4

$L = 2$, number of samples = 5000, burn in = 4000






Comparison on training data
Predicted rating
Actual rating
0
1
2
3
4
5
1
2
3
4
5


Comparison on test data
Predicted rating
Actual rating
-1
0
1
2
3
4
5
1
2
3
4
5

Appendix 2

```
rm(list = ls())
library(invgamma)
library(data.table)
library(mnormt)
library(coda)
library(ggplot2)

gamma_m_sample=function(xm,ym,sigma2,sig_gamma)
{
  inverse=solve(sig_gamma)
  variance=solve((t(xm)%*%xm)/sigma2+inverse)
  mean=variance%*%t(xm)%*%ym/sigma2
  sample=rmnorm(1,mean,variance)
  return (sample)
}

theta_u_sample=function(xu,yu,sigma2,sig_theta)
{
  inverse=solve(sig_theta)
  variance=solve((t(xu)%*%xu)/sigma2+inverse)
  mean=variance%*%t(xu)%*%yu/sigma2
  sample=rmnorm(1,mean,variance)
  return (sample)
}

gibbs=function (mdata,L,m,gamma_var,theta_var)
{
  data=copy(mdata)
  n=length(data$rating)

  total_avg=mean(data$rating)
  user_avg=aggregate(rating~userId,data,mean)
  movie_avg=aggregate(rating~movieId,data,mean)

  theta=matrix(0,nrow=length(user_avg[,1]),ncol=L+1)
  theta[,1]<-user_avg[,2]-total_avg
  thetasamples=array(0,c(m,length(user_avg[,1]),L+1))

  gamma=matrix(0,nrow=length(movie_avg[,1]),ncol=L+1)
  gamma[,1]<-movie_avg[,2]-total_avg
  gammasamples=array(0,c(m,length(movie_avg[,1]),L+1))

  mu=total_avg
  musamples=array(0,m)

  sigma2=var(data$rating)
  sigma2samples=array(0,m)

  I=diag(L+1)
  sig_gamma=I*gamma_var
  sig_theta=I*theta_var
```

```

  for (i in 1:m)
  {
    print(i)
    y_mu=rowSums(theta[data$uindex,-1]*gamma[data$mindex,-1])+theta[data$
   uindex,1]+gamma[data$mindex,1]
    mu=rnorm(1,mean(data$rating-y_mu),sqrt(sigma2/n))
    sigma2=rinvgamma(1,n/2,sum((data$rating-y_mu-mu)^2)/2)
    for (movie in movieids)
    {
      index=match(movie,movieids)
      g_index=which(data$mindex==index)
      ym=data$rating[g_index]-mu-theta[data$uindex[g_index],1]
      xm=cbind(1,theta[data$uindex[g_index],-1])
      gamma[index,]<-gamma_m_sample(xm,ym,sigma2,sig_gamma)
    }
    for (user in userids)
    {
      index=match(user,userids)
      g_index=which(data$uindex==index)
      ym=data$rating[g_index]-mu-gamma[data$mindex[g_index],1]
      xm=cbind(1,gamma[data$mindex[g_index],-1])
      theta[index,]<-theta_u_sample(xm,ym,sigma2,sig_theta)
    }
    musamples[i]<-mu
    sigma2samples[i]<-sigma2
    thetasamples[i,,]<-theta
    gammasamples[i,,]<-gamma
  }
  r=list(codepurple"codepurplemucodepurple"=musamples,codepurple"codepurplesigma2
   codepurple"=sigma2samples,codepurple"codepurplethetacodepurple"=thetasamples,
   codepurple"codepurplegammacodepurple"=gammasamples)
  return(r)
}

mcmcplots=function(gibbsresult)
{
  cumuplot(mcmc(gibbsresult$mu),main=codepurple"codepurpleCumulativecodepurple
   codepurpleplotcodepurple codepurpleofcodepurple codepurplemucodepurple",ylab=codepurple"
   codepurplemucodepurple")
  cumuplot(mcmc(gibbsresult$sigma2),main=codepurple"codepurpleCumulativecodepurple
   codepurpleplotcodepurple codepurpleofcodepurple codepurplesigmacodepurple^2codepurple",
   ylab=codepurple"codepurplesigmacodepurple^2codepurple")
  cumuplot(mcmc(gibbsresult$gamma[,1,]))
  cumuplot(mcmc(gibbsresult$theta[,1,]))
  plot(mcmc(gibbsresult$mu),trace=F)
  plot(mcmc(gibbsresult$sigma2),trace=F)
  plot(mcmc(gibbsresult$gamma[,1,]),trace=F)
  plot(mcmc(gibbsresult$theta[,1,]),trace=F)
  print(effectiveSize(gibbsresult$mu))
  print(effectiveSize(gibbsresult$sigma2))
  print(effectiveSize(gibbsresult$gamma[,1,]))
  print(effectiveSize(gibbsresult$theta[,1,]))
}
```

```
plot_results=function(data,gibbsresult,burnin)
{
  m=length(gibbsresult$mu)
  mu_final=mean(gibbsresult$mu[c(burnin:m)])
  sigma2final=mean(gibbsresult$sigma2[c(burnin:m)])
  gamma_final=colMeans(gibbsresult$gamma[c(burnin:m),,], dim=1)
  theta_final=colMeans(gibbsresult$theta[c(burnin:m),,], dim=1)

  predicted=mu_final+gamma_final[data$mindex,1]+theta_final[data$uindex,1] + rowSums(theta_final[data$uindex,-1]*gamma_final[data$mindex,-1])

  compare=as.data.frame(predicted)
  compare$actual=data$rating

  hist(compare$actual-compare$predicted, main=codepurple"codepurpleDistribution codepurple codepurpleofcodepurple codepurpleerrorscodepurple codepurpleincodepurple codepurpletrainingcodepurple codepurpledatacodepurple",xlab=codepurple"codepurpleError codepurple",ylab=codepurple"codepurpleFrequencycodepurple")
  print(sd(compare$actual-compare$predicted))

  meantable=aggregate(predicted~actual,compare,mean)
  sdtable=aggregate(predicted~actual,compare,sd)

  plot(compare$actual,compare$predicted,main=codepurple"codepurpleComparison codepurple codepurpleoncodepurple codepurpletrainingcodepurple codepurpledatacodepurple", xlab=codepurple"codepurpleActualcodepurple codepurpleratingcodepurple",ylab=codepurple" codepurplePredictedcodepurple codepurpleratingcodepurple")
  lines(meantable$actual,meantable$predicted,lwd=2)
  lines(sdtable$actual,meantable$predicted+sdtable$predicted,lty=2)
  lines(sdtable$actual,meantable$predicted-sdtable$predicted,lty=2)
}

validation_results=function(gibbsresult,testdata,burnin)
{
  m=length(gibbsresult$mu)
  mu_final=mean(gibbsresult$mu[c(burnin:m)])
  sigma2final=mean(gibbsresult$sigma2[c(burnin:m)])
  gamma_final=colMeans(gibbsresult$gamma[c(burnin:m),,], dim=1)
  theta_final=colMeans(gibbsresult$theta[c(burnin:m),,], dim=1)

  predicted=mu_final+gamma_final[testdata$mindex,1]+theta_final[testdata$uindex,1] + rowSums(theta_final[testdata$uindex,-1]*gamma_final[testdata$mindex,-1])

  compare=as.data.frame(predicted)
  compare$actual=testdata$rating

  hist(compare$actual-compare$predicted, main=codepurple"codepurpleDistribution codepurple codepurpleofcodepurple codepurpleerrorscodepurple codepurpleincodepurple codepurpletestcodepurple codepurpledatacodepurple",xlab=codepurple"codepurpleError codepurple",ylab=codepurple"codepurpleFrequencycodepurple")
  print(sd(compare$actual-compare$predicted))

```

```
  meantable=aggregate(predicted~actual,compare,mean)
  sdtable=aggregate(predicted~actual,compare,sd)

  plot(compare$actual,compare$predicted,main=codepurple"codepurpleComparison codepurple codepurpleoncodepurple codepurpletestcodepurple codepurpledatacodepurple",xlab=codepurple"codepurpleActualcodepurple codepurpleratingcodepurple",ylab=codepurple"codepurplePredictedcodepurple codepurpleratingcodepurple")
  lines(meantable$actual,meantable$predicted,lwd=2)
  lines(sdtable$actual,meantable$predicted+sdtable$predicted,lty=2)
  lines(sdtable$actual,meantable$predicted-sdtable$predicted,lty=2)
}

movie=read.csv(codepurple"codepurpleratings_codepurplesmallcodepurple.codepurplecsv codepurple")
perusercount=aggregate(rating~userId,movie,length)
hist(perusercount$rating,breaks=50,main=codepurple"codepurpleDistributioncodepurple codepurpleofcodepurple codepurpleratingscodepurple codepurpleforcodepurple codepurplemoviecodepurple", xlab=codepurple"codepurpleNumbercodepurple codepurpleof codepurple codepurpleratingscodepurple",ylab=codepurple"codepurpleCountcodepurple codepurpleofcodepurple codepurplemoviescodepurple")
permoviecount=aggregate(rating~movieId,movie,length)
hist(permoviecount$rating,breaks=50,main=codepurple"codepurpleDistribution codepurple codepurpleofcodepurple codepurpleratingscodepurple codepurplebycodepurple codepurpleusercodepurple", xlab=codepurple"codepurpleNumbercodepurple codepurpleof codepurple codepurpleratingscodepurple",ylab=codepurple"codepurpleCountcodepurple codepurpleofcodepurple codepurpleuserscodepurple")
hist(movie$rating,breaks=5,main=codepurple"codepurpleDistributioncodepurple codepurpleofcodepurple codepurpleratingscodepurple", xlab=codepurple"codepurpleRating codepurple",ylab=codepurple"codepurpleCountcodepurple codepurpleofcodepurple codepurplemoviescodepurple")

n=length(movie$rating)

total_avg=mean(movie$rating)
user_avg=aggregate(rating~userId,movie,mean)
movie_avg=aggregate(rating~movieId,movie,mean)

userids=user_avg[,1]
movieids=movie_avg[,1]

userindex=match(movie$userId,userids)
movieindex=match(movie$movieId,movieids)

movie$uindex<-userindex
movie$mindex<-movieindex

test_ind <- sample(seq_len(n), size = floor(0.32*n))

temp=as.data.frame(table(movie$movieId[-test_ind]))
allowed=as.data.frame(temp[temp$Freq>2,]$Var1)
names(allowed)<-(codepurple"codepurplemoviecodepurple")
test_ind=test_ind[movie$movieId[test_ind]%in%allowed$movie]

temp=as.data.frame(table(movie$userId[-test_ind]))
```

```
allowed=as.data.frame(temp[temp$Freq>2,]$Var1)
names(allowed)<-(codepurple"codepurpleusercodepurple")
test_ind=test_ind[movie$userId[test_ind]%in%allowed$user]

movie_train=movie[-test_ind,]

validation_ind=sample(test_ind, size = floor(length(test_ind)/2))
test_ind=test_ind[!test_ind%in%validation_ind]

movie_validate=movie[validation_ind,]
movie_test=movie[test_ind,]

test2=gibbs(movie_train,4,500,1,1)
mcmcplots(test2)
plot_results(movie_train,test2,200)
validation_results(test2,movie_test,200)

test3=gibbs(movie_train,10,1000,1,1)
mcmcplots(test3)
plot_results(movie_train,test3,200)
validation_results(test3,movie_validate,200)

test4=gibbs(movie_train,20,1500,1,1)
mcmcplots(test4)
plot_results(movie_train,test4,1000)
validation_results(test4,movie_validate,1000)

test5=gibbs(movie_train,2,5000,1,1)
mcmcplots(test5)
plot_results(movie_train,test5,4000)
validation_results(test5,movie_validate,4000)
validation_results(test5,movie_test,4000)
print(mean(test5$mu[c(4000:5000)]))
print(mean(test5$sigma2[c(4000:5000)]))
```