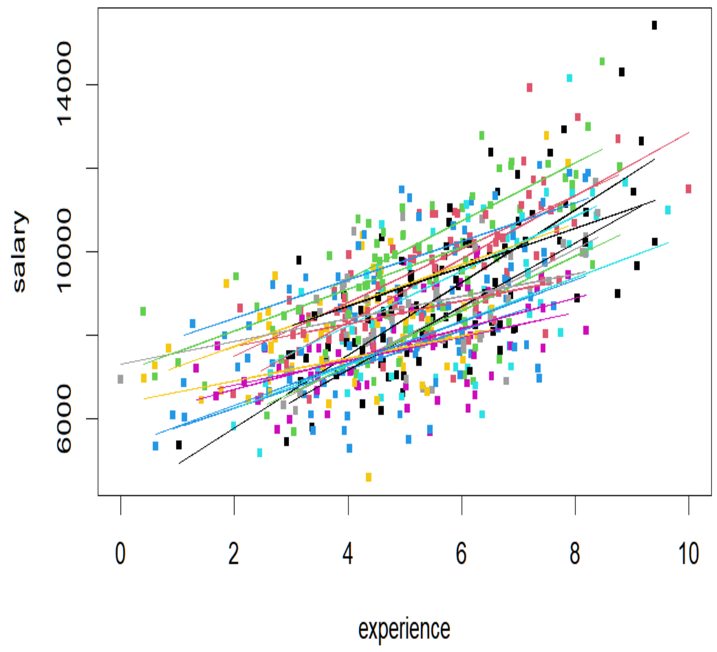


Bayesian hierarchical linear regression.

ISBD@RUC

Jiahui Xin

2022-06-20

Contents

1 Introduction

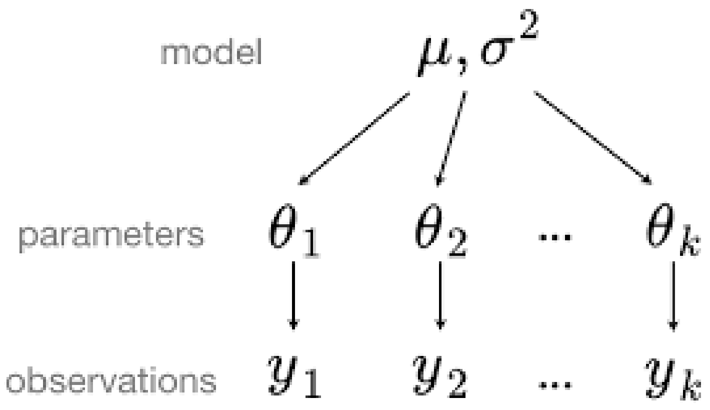


Figura 1: Bayesian hierarchical model (pic from Internet)

Bayesian hierarchical linear models (structure like Figura 1: k clusters with different parameters $\{\theta_i\}$ but generated from the same parameter $\{\mu, \sigma^2\}$) are intensively introduced in Chapter 15 of the book "Bayesian Data Analysis"(Gelman y col., 2021). There are two main advantages of hierarchical models:

- Hierarchical regression models are useful as soon as there are predictors at different levels of variation.
- With cluster sampling, hierarchical modeling is in fact necessary in order to generalize to the unsampled clusters.

I used salary data set (from Methodenlehre@github, Department of Cognitive Psychology, Perception and Methods, University of Bern)with the description as follows.

"These are fictitious salary data from 20 companies with 30 employees each. For each employee information is available about his/her salary (salary), at which company the person is employed (firma), and how long the person has been working there (experience). We also have information about the sector (public or private) in which the company is active (sector)."

I plotted *salary* and *experience* in the graph, using different colors for the points of different firma (companies). The separate fitted lines were plotted with corresponding colors. (Figura 2) There is obvious hierarchical structure, i.e. similarity and variation among clusters.

The goal is to do unified analysis across different firma (companies) with bayesian hierarchical linear regression. Although I only used the first three virables *firma, salary* and *experience*, I

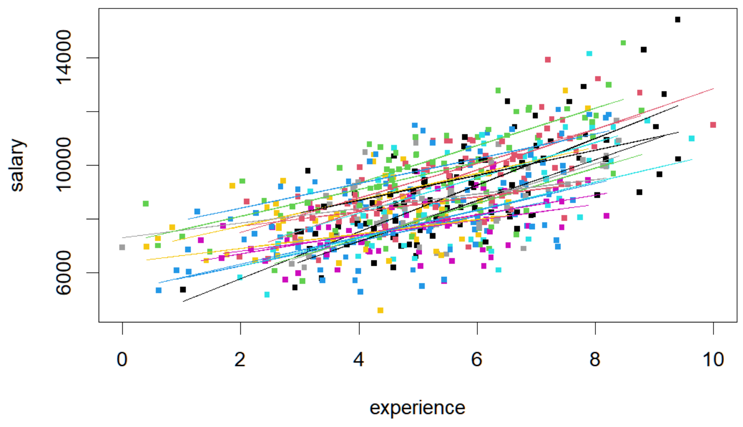


Figura 2: The data set with hierarchical structure

discovered the posterior order of slope parameter β_1 implies the *sector*.

Pooled model and separate model are also considered for comparison.

2 Bayesian sampling with R package *R2jags*

2.1 R2jags

The R package *R2jags* description (Su & Yajima, 2021):

"Providing wrapper functions to implement Bayesian analysis in JAGS. Some major features include monitoring convergence of a MCMC model using Rubin and Gelman Rhat statistics, automatically running a MCMC model till it converges, and implementing parallel processing of a MCMC model for multiple chains."

With *R2jags*, we only need to write our model into the model file and the program will automatically do MCMC sampling. The posterior samples and some test statistics can be easily collected by the written functions in this package.

2.2 Specific bayesian model

$\boldsymbol{Notation}$: y for the salary, x for the experience, i for the sample index, k for the corresponding firma index, β_{0k} and β_{1k} for the regression coefficient in the k-th firma and σ for the standard deviation.

With the above notation, the final bayesian prior model can be written as follows (Ecuación 1). More specifically, I set $(s, r) = (1, 10^8), v_0 = v_1 = 10^8$

$$\begin{aligned} y_i &\sim \mathrm{N}(\beta_{0k} + \beta_{1k}x_i, \sigma^2), 1/\sigma^2 \sim \Gamma(s, r). \\ \beta_{0k} &\sim t_1(\beta_0, \sigma_0^2) \text{ i.i.d. for } k \in [K]; \\ \beta_0 &\sim \mathrm{N}(0, v_0), 1/\sigma_0^2 \sim \Gamma(s_0, r_0). \\ \beta_{1k} &\sim t_1(\beta_1, \sigma_1^2) \text{ i.i.d. for } k \in [K]; \\ \beta_1 &\sim \mathrm{N}(0, v_1), 1/\sigma_1^2 \sim \Gamma(s_1, r_1). \end{aligned} \tag{1}$$

Here we need to set values of shape-rate parameters $\{(s, r), (s_0, r_0), (s_1, r_1)\}$ for $\{1/\sigma^2, 1/\sigma_0^2, 1/\sigma_1^2\}$ and variance parameters $\{v_0, v_1\}$ for $\{\beta_0, \beta_1\}$.

Known that the large rate parameters $\{r_0, r_1\}$ allow much variation among clusters which tends to separate analysis; while the small rate parameter r_0, r_1 allows little variation among clusters which tends to pooled analysis. I set almost flat prior for population parameters $\{\beta_0, \beta_1\}$ and $1/\sigma^2$ and tuned the prior of cluster parameters $\{1/\sigma_0^2, 1/\sigma_1^2\}$. More specifically, I set $(s, r) = (1, 10^8), v_0 = v_1 = 10^8$ and $(s_0, r_0) = (s_1, r_1) \in \{(1, 10^4), (1, 10^5), (1, 10^6), (1, 10^7)\}$.

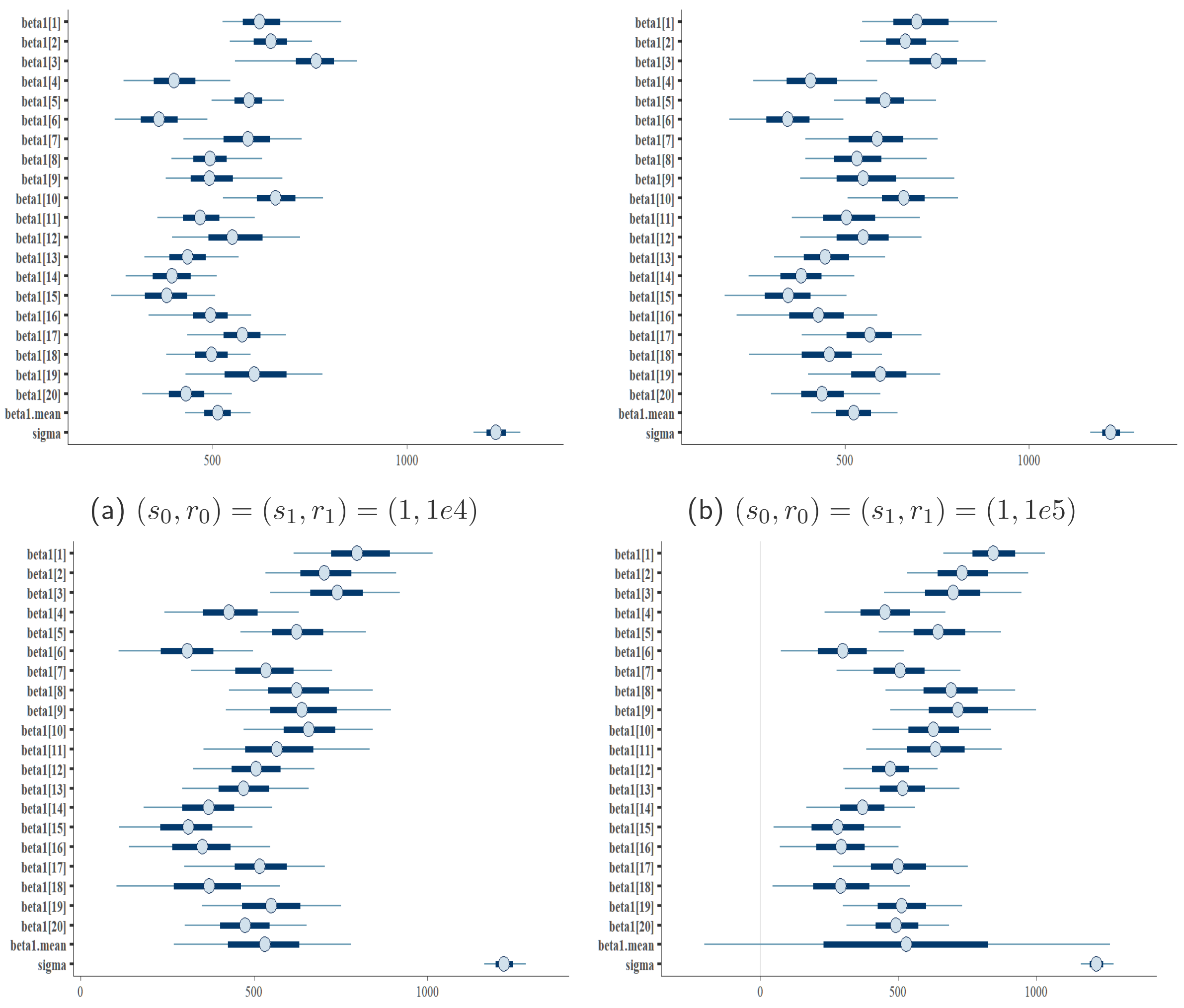
Under each setting, I generated 4 MCMC chains (each chain 4000 samples within 1000 burn-in samples) and got the following posterior distribution (Figura 3). As the dispersion parameter r_0, r_1 increases, the posterior distribution of $\{\beta_{1k}\}$ becomes more and more dispersed.

In the 20 companies, 10 companies (*index*$\in \{1, 2, 3, 5, 7, 10, 12, 16, 17, 19\}$) have *sector* "Private"while others have *sector* "Public". And we notice that the top 10 β_{1k} rough corresponds to the 10 private companies (Tabla 1). It seems that the first two model can almost correctly distinguish companies with different *sector*. Furthermore DIC criterion was used to help model selection (Tabla 2). It suggests the second model has the lowest DIC.

Tabla 1: Top 10 β_{1k} set

	set of top 10 β_{1k}
$(s_0, r_0) = (s_1, r_1) = (1, 1e4)$	$1, 2, 3, 5, 7, 10, 12, 17, 16(8, 9, 18), 19$
$(s_0, r_0) = (s_1, r_1) = (1, 1e5)$	$1, 2, 3, 5, 7, 9, 10, 12, 17, 19$
$(s_0, r_0) = (s_1, r_1) = (1, 1e6)$	$1, 2, 3, 5, 7(17), 8, 9, 10, 11, 19$
$(s_0, r_0) = (s_1, r_1) = (1, 1e7)$	$1, 2, 3, 5, 7(19), 8, 9, 10, 11, 13$
True	$1, 2, 3, 5, 7, 10, 12, 16, 17, 19$

[1]



(a) $(s_0, r_0) = (s_1, r_1) = (1, 1e4)$

(b) $(s_0, r_0) = (s_1, r_1) = (1, 1e5)$

(c) $(s_0, r_0) = (s_1, r_1) = (1, 1e6)$

(d) $(s_0, r_0) = (s_1, r_1) = (1, 1e7)$

Figura 3: Posterior distribution of β_1 under different dispersion parameter

So I selected the model under $(s_0, r_0) = (s_1, r_1) = (1, 1e5)$ as the final bayesian model. It passes posterior model check (Figura 4). More results see the appendix (Subsección 4.1).

2.3 Comparison with pooled analysis and separate analysis (based on residuals)

This part contains classic two OLS regression models: one is to ignore clusters and pool the samples; the other is ignore the similarity among clusters and fit model separately.

Ignoring reduced freedom, the pooled model has RMSE of fitted residuals 1326.87 while the

[1]Red colored number is misspecified. Number in brackets share similar size.

Tabla 2: Model selection with DIC

$(s_0, r_0) = (s_1, r_1)$	$(1, 1e4)$	$(1, 1e5)$	$(1, 1e6)$	$(1, 1e7)$
DIC	10192.0	10184.8	10186.4	10192.1

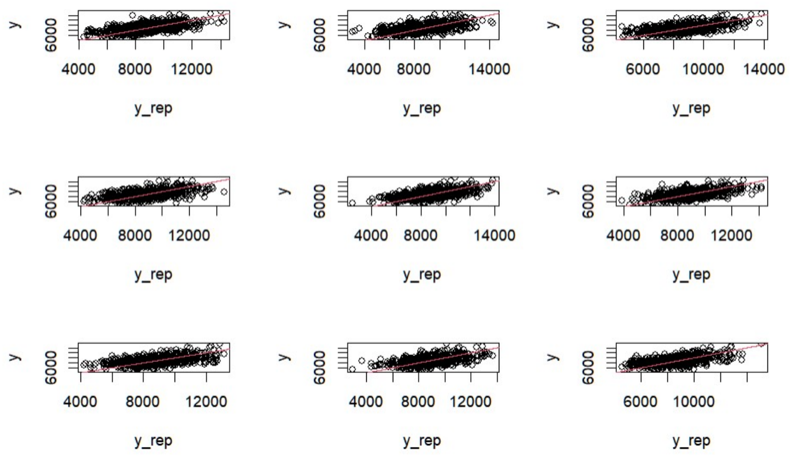


Figura 4: Random selected 9 posterior parameters to generat y_{rep}

bayesian model has RMSE of fitted residuals 1044.68. With shared information among clusters, bayesian model fits better than the pooled model.

Ignoring reduced freedom, I compared the RMSE of fitted residuals of bayesian model and separate model in each cluster (Tabla 3). The bayesian model has slightly larger RMSE because it regularise the parameters among clusters generated from one same mechanism. While seperate model has small RMSE, it tends to overfit in each cluster and cannot generalise to new groups, i.e. old information has to be given up.

3 Conclusion

In this report, I used Bayesian hierarchical model to analyze one data set with 20 clusters. Unlike the classical pooled and separate analysis which whether requires exactly same distribution or shares no information among clusters, the bayesian hierarchical model shares similarity and also allows difference among clusters and hence it can generalise among clusters.

The deeper property *sector* of clusters can be recovered with our bayesian hierarchical model, which shows the ability of generalisation.

Tabla 3: RMSE of fitted residuals (ignoring reduced freedom)

	Bayesian model	Separate model	Difference
1	1496.86	1464.29	32.58
2	1180.38	1170.65	9.73
3	881.27	878.73	2.54
4	971.22	966.30	4.92
5	1162.32	1160.08	2.23
6	983.34	979.54	3.81
7	1072.45	1060.96	11.50
8	903.71	867.86	35.85
9	1101.85	1065.54	36.32
10	815.27	814.50	0.77
11	1000.85	978.53	22.32
12	1057.46	1039.38	18.09
13	1255.14	1248.30	6.83
14	916.37	915.80	0.57
15	971.44	963.98	7.45
16	969.26	935.84	33.43
17	932.49	921.02	11.47
18	905.69	877.27	28.42
19	934.97	919.62	15.36
20	1152.26	1146.40	5.87

Referencias

Gelman, A., Carlin, J. B., Stern, H. S. & Rubin, D. B. (2021). *Bayesian data analysis (3rd edition)*. Chapman; Hall/CRC.

Su, Y.-S. & Yajima, M. (2021). R2jags: Using R to Run 'JAGS'. R package version 0.6-1; 2020. *URL https://CRAN. R-project. org/package= R2jags*.

4 Appendix

4.1 More results of Bayesian model

This part contains posterior distribution of three key parameters $\{\beta_0, \beta_1, \sigma\}$

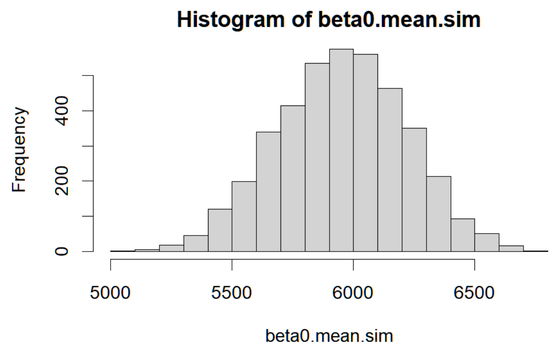


Figura 5: β_0 posterior distribution

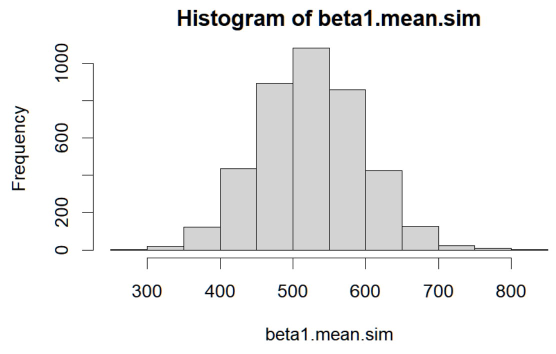


Figura 6: β_1 posterior distribution

4.2 R Code

Copied from my R markdown file. Just create a new R markdown file and copy the following code into it to run.

```
---
title: "Bayesian project"
author: "Jiahui Xin"
date: "`r Sys.Date()`"
output: pdf_document
---
```

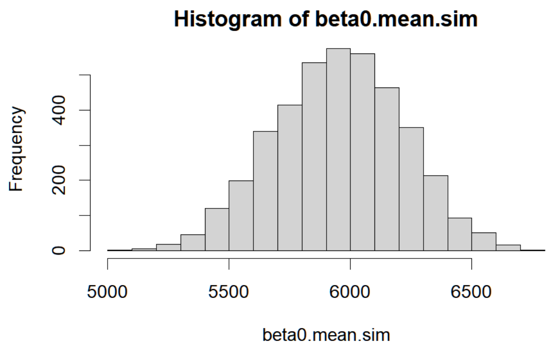


Figura 7: σ posterior distribution

~~~
```{r,warning=FALSE}
library(MASS)
library(R2jags)
library(bayesplot)
```
~~~

```
# new data
```

~~~
```{r}
salary<-read.csv(
"https://raw.githubusercontent.com/methodenlehre/data/master/salary-data.cs
# the last characters are "salary-data.csv".
```
~~~

~~~
```{r}
salary$index=sapply(salary$firma,
                    function(x)as.numeric(strsplit(x," ")[[1]][2]))
```
~~~

~~~
```{r}
with(salary,plot(experience,salary,col=index,cex=4.6,pch=46))
for(i in 1:20){
points(list.fit[[i]]$model$experience,list.fit[[i]]$fitted.values,
                                col=i,type="l")}
~~~
```{r}
fit.pool<-lm(salary~experience,data=salary)
summary(fit.pool)
plot(fit.pool)
plot(fit.pool$fitted.values,salary$salary)
points(1:20000,1:20000,col=2,type="l")
```

```{r}
list.fit<-list()
coeff.seperate=c()
for(i in 1:20){
  list.fit[[i]]=(lm(salary~experience,data=salary[salary$index==i,]))
  coeff.seperate=rbind(coeff.seperate,list.fit[[i]]$coefficients)
}
```

```{r}
sigma.seperate<-unlist(lapply(list.fit,
function(x)sqrt(sum((x$resid)^2)/x$df.residual)))
summary(sigma.seperate)
boxplot(sigma.seperate)

print(lapply(list.fit,summary))

colMeans(coeff.seperate)
coeff.bias=coeff.seperate-t(matrix(rep(colMeans(coeff.seperate),20),2))
```

```
'''



'''{r}
set.seed(114514)
dat.list <- with(salary, list(y=salary,X=experience,
                                    X_firma=index, k=length(unique(index)),
                                    n=length(salary)))
model <- function() {
  #Likelihood
    for (i in 1:n) {
    y[i]~dnorm(mu[i],tau)
    mu[i] <- (beta0[X_firma[i]]) + (beta1[X_firma[i]]) * X[i]
    }
    #Priors
    tau  ~ dgamma(1,1.0E8)
    sigma <- 1/sqrt(tau)

    for(j in 1:k){
      beta0[j]~dt(beta0.mean,tau0[j],1)
      tau0[j]  ~ dgamma(1,1.0E5)#change this line to tune
      sigma0[j] <- 1/sqrt(tau0[j])
    }
    beta0.mean~dnorm(0,1.0E-8)

    for(l in 1:k){
      beta1[l]~dt(beta1.mean,tau1[l],1)
      tau1[l]  ~ dgamma(1,1.0E5)#change this line to tune
      sigma1[l] <- 1/sqrt(tau1[l])
    }
    beta1.mean~dnorm(0,1.0E-8)

    }
system.time(
jags <- jags(model.file=model, data=dat.list, inits=NULL,
             param=c('beta0.mean','beta0','beta1.mean','beta1','sigma'),
             n.chain=4,
             n.iter=3000, n.burnin=1000)
  )
```
```{r}
plot(jags)
par(mar=c(1,1,1,1),mfrow=c(3,3))
#traceplot(jags, ask=FALSE)
```

```{r}
print(jags)
```

```{r}
mcmc_intervals(jags$BUGSoutput$sims.matrix, regex_pars = "beta0|sigma")
mcmc_intervals(jags$BUGSoutput$sims.matrix, regex_pars = "beta1|sigma")
```

```{r}
unique(salary$index[salary$sector=="Privat"])

(1:20)[coeff.bias[,2]>0]
```

```{r}
beta0.sim<-jags$BUGSoutput$sims.matrix[,1:20]
beta1.sim<-jags$BUGSoutput$sims.matrix[,22:41]
sigma.sim<-jags$BUGSoutput$sims.matrix[,"sigma"]
beta0.mean.sim<-jags$BUGSoutput$sims.matrix[,"beta0.mean"]
```

~~~
beta1.mean.sim<-jags$BUGSoutput$sims.matrix[,"beta1.mean"]

par(mfrow=c(3,3))
y<-salary$salary
spl_ind<-sample(1:nrow(jags$BUGSoutput$sims.matrix),9)
for(i in spl_ind){
  y_rep<-rep(beta0.sim[i,],each=30)+
    salary$experience*rep(beta1.sim[i,],each=30)+rnorm(600,0,sigma.sim[i])
  plot(y_rep,y)
  points(0:20000,0:20000,type="l",col=2)
}
```
```{r}
beta0.mean=colMeans(beta0.sim)
beta1.mean=colMeans(beta1.sim)
sigma.mean=mean(sigma.sim)
y<-salary$salary
spl_ind<-sample(1:nrow(jags$BUGSoutput$sims.matrix),9)
y_hat<-rep(beta0.mean,each=30)+salary$experience*rep(beta1.mean,each=30)
plot(y_hat,y)
points(0:20000,0:20000,type="l",col=2)
```
```{r}
hist(beta0.mean.sim)
hist(beta1.mean.sim)

summary(beta0.mean.sim)

summary(beta1.mean.sim)

summary(coeff.seperate[,2])

summary(beta1.mean)
~~~

```
'''



'''{r}
resid.bayes<-y-y_hat
#hist(resid.bayes)

sqrt(mean(resid.bayes^2))
rmse.bayes=numeric(20)
print("calibration")
for(i in 1:20){
  rmse.bayes[i]<-(sqrt(mean(resid.bayes[salary$index==i]^2)))
}

rmse.seperate=numeric(20)
for(i in 1:20){
  rmse.seperate[i]<-(sqrt(mean(list.fit[[i]]$residuals^2)))
}
summary(rmse.bayes)
rmse.bayes-rmse.seperate
'''

'''{r}
round(rmse.bayes,2)
round(rmse.seperate,2)

cbind(round(rmse.bayes,2),round(rmse.seperate,2),
        round(rmse.bayes-rmse.seperate,2))
'''
```