Simulations on "Bayesian hierarchical weighting adjustment and survey inference" by Trangucci et al.

Fall 2018

D.Bonnéry

1 Model description

1.1 Trangucci et al. model0

We describe the model given in (**?**, Sec. 2).

- Population made of H strata $U_1,\ldots,U_J$.
- Stratum j size: N_j, Total size: $N=\sum_j N_j$
- Sample size in stratum j: n_j
- I vector: sample indicator
- y vector: study variable
- θ_j: average of y in stratum j. $\theta_j=\sum_{i\in U_j} y_i$. In the paper there is an ambiguity: First, θ is called the "population estimand of interest[...] the overall or domain mean". Ambiguity comes from the term "mean". Is $\theta=N^{-1}\sum_i y_i$ or Is $\theta=N^{-1}\sum_k E[y_i]$?
- X : Design variables. In the paper $X^1,\ldots,X^J$ are badly defined. The idea is that X variables are Q categorical variables. The strata correspond to the cells obtained from these categorical variables.

- denote by $1,\dots K_q$ the categories for variable X_q.
- denote by $j[i]$ the stratim of unit i
- Denote by $k[q,j]$ the category for variable X_q in stratum j

Consider the following hierarchical model:

$$y_i \sim \mathcal{N}\left(\theta^{\star}_{j[i]}, \sigma_y^2\right) \tag{1.1}$$

$$\theta^{\star}_j = \alpha_0 + \sum_{\ell=1}^{Q}\left(\sum_{q_1<\ldots<q_\ell\in\{1,\ldots,Q\}} \alpha_j^{(q_1,\ldots,q_\ell)}\right) \tag{1.2}$$

$$\forall \ell \in \{1,\ldots,Q\},\ \forall q_1,\ldots,q_\ell \in \{1,\ldots,Q\},\ \forall j \in \{1,\ldots,H\} : \alpha_j^{(q_1,\ldots,q_\ell)} \sim \mathcal{N}(0, (\lambda_j^{(q_1,\ldots,q_\ell)}\sigma)^2) \tag{1.3}$$

$$\forall \ell \in \{1,\ldots,Q\},\ \forall q_1,\ldots,q_\ell \in \{1,\ldots,Q\},\ \forall j \in \{1,\ldots,H\} : \lambda_j^{(q_1,\ldots,q_\ell)} = \delta^{(\ell)}\prod_{l=1}^{\ell}\gamma^{(q_l)}_{k[q_l,j]} \tag{1.4}$$

$$\sigma \sim \text{Cauchy}_+(0,1) \tag{1.5}$$

$$\forall q \in \{1,\ldots,Q\},\ k \in \{1,\ldots,K_q\}\ : \gamma_k^{(q)} \sim \mathcal{N}_+(0,1) \tag{1.6}$$

$$\delta^{(\ell)} \sim \mathcal{N}_+(0,1) \tag{1.7}$$

$$\sigma_y \sim \text{Cauchy}_+(0,5) \tag{1.8}$$

$$\alpha_0 \sim \text{N}(0,10) \tag{1.9}$$

Note I made up the prior on α_0 as I did not find in the paper.

1.2 Competing model

We describe a competing model.

Consider the following hierarchical model:

$$y_i \sim \mathcal{N}\left(\theta^{☾}_{j[i]}, \sigma_y^2\right) \tag{1.10}$$

$$(\theta^{☾}_j)_{j=1}^{Q} \sim \mathcal{N}(0, \Sigma) \tag{1.11}$$

$$\Sigma_{j_1,j_2} = \sigma^2 D_\alpha(j_1,j_2) \tag{1.12}$$

$$D_\alpha(j_1,j_2) = \sigma_y^2\left(\exp\left(-\sum_{q=1}^{Q}\alpha_q(k[q,j_1] \neq k[q,j_2]\right)\right) \tag{1.13}$$

$$\sigma_y \sim \text{Cauchy}_+(0,5) \tag{1.14}$$

$$\alpha_q \sim \text{Cauchy}_+(0,1) \tag{1.15}$$

1.3 Competing model 2 (Joseph)

The MRP should work for any dataset, but it shouldn't be totally unrealistic, relative to the prior. I have written cell mean j in primitive notation for a two-way layout Let $i = 1 \dots I$ denote the rows, $j = 1 \dots J$ the column. Then $\theta_{i,j} = \eta + \alpha_i + \beta_j + \gamma_{ij}$.

For a simple example take the $I = 3$ $J = 2$ and think of constant local effect within level "l" s.e the paper's notation:

$$\begin{aligned}
\theta_{1,1} &= \eta + \alpha + \beta \\
\theta_{1,2} &= \eta + \alpha_i + \beta_j + d_1 \\
\theta_{2,1} &= \eta + \alpha + \beta \\
\theta_{2,2} &= \eta + \alpha + \beta + d_2 \\
\theta_{3,1} &= \eta + \alpha + \beta \\
\theta_{3,2} &= \eta + \alpha + \beta + d_3
\end{aligned}$$

and perhaps $Y_{ij} \equiv \theta_{ij}$

2 Data Generation

Step 1

Create procedures that can

1. generate a population of size N and with Q random categorical variables $X_1, \dots, X_Q$. X_q is generated by drawing with replacement in $\{1, \dots, K_q\}$ where $K_q > 1$, K_q can be generated by drawing from $\mathcal{U}_{1,\dots,p}$.
2. compute the corresponding stratum $j = 1, \dots, J$ and the correspondances $(k[q, j])_{j=1,\dots,J, q=1,\dots,Q}$.
3. generate, for such a population, the hyper parameters σ_y, α_0, $(\delta^\ell)_{\ell=1,\dots,Q}$,$\lambda_k^l$,$\sigma$.
4. compute the $\lambda_{k_1,\dots,k_\ell}^{(q_1,\dots,q_\ell)}$, for possible values of $k_1, \dots, k_\ell{}^{(q_1,\dots,q_\ell)}$.
5. generate the $\alpha_{j,(k_1,\dots,k_\ell)}^{(q_1,\dots,q_\ell)}$.

6. compute $\theta^{\star}$.
7. generate a number r of realisations of y for such hyperparameters and such strata.
8. Set seed.
9. Create a population with $N = 10000, Q = 2$, $p = 5$. Display J, N_j, $\theta^{\star}$.

1. To generate the population and stratification variables we use the function `SimuTrangucci::Gen_design_variables`:

```
?SimuTrangucci::Gen_design_variables
XX=SimuTrangucci::Gen_design_variables()
```

2.
3.
4.
5.
6.
7.
8.
9. 9. This are tables we obtained with different seeds.

Table 2.1:

	X_1	X_2	Stratum (j)	N_j	$\theta^{\star}$
S1	1	1	S1	113	-5.151
S2	1	2	S2	111	-3.938
S3	1	3	S3	119	-2.698
S4	2	1	S4	101	-0.163
S5	2	2	S5	106	-4.607
S6	2	3	S6	105	-1.853
S7	3	1	S7	140	0.434
S8	3	2	S8	104	-1.733
S9	3	3	S9	101	-6.088

Table 2.2:

	X_1	X_2	Stratum (j)	N_j	$\theta^\star$
S1	1	1	S1	91	3.215
S2	1	2	S2	91	0.055
S3	1	3	S3	82	2.687
S4	1	4	S4	80	2.681
S5	2	1	S5	78	-5.146
S6	2	2	S6	95	-10.338
S7	2	3	S7	80	3.576
S8	2	4	S8	91	0.260
S9	3	1	S9	70	6.074
S10	3	2	S10	73	0.900
S11	3	3	S11	83	2.567
S12	3	4	S12	86	-4.289

```
library(SimuTrangucci)
set.seed(11)
GG<-Generate_all(N=1000,Q=2,p=5)
Strata1<-data.frame(GG$XX$Strata,
                    N_j=GG$XX$N_j,
                    thetastar=GG$thetastar)
set.seed(7)
GG<-Generate_all(N=1000,Q=2,p=5)
Strata2<-data.frame(GG$XX$Strata,
                    N_j=GG$XX$N_j,
                    thetastar=GG$thetastar)
```

3 Bayesian computations

Step 1

Using Jags,

1. Draw posterior distribution of $\theta_j^\star$ for the largest value of $\theta_j^\star$, obtained from the observation of

$y^{(1)}$ only. Add real value of θ_1 and prediction to the plot.

2. Draw distribution of predictions of $\theta_j^\star$. Add real value of θ_j to the plot.
3. Draw real values of all $\theta_j^\star$ vs $r = 30$ predictions.

Some functions to generate the jags model file:

```
library(SimuTrangucci)
library(R2jags)
library(ggplot2)
library(plyr)
GG<-Generate_all(N=1000,Q=2,p=5)
x<-model.text(GG);x
```

```
model{
for(i in 1:N){y[i]~dnorm(thetastar[j_i[i]],1/sqrt(sigma_y^2));}
for (j in 1:J){thetastar[j]=alpha0+alpha.X1[j]+alpha.X2[j]+alpha.X1.X2[j];}
for (j in 1:J){alpha.X1[j]~dnorm(0,1/sqrt((lambda.X1[j]*sigma)^2));}
for (j in 1:J){alpha.X2[j]~dnorm(0,1/sqrt((lambda.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2[j]~dnorm(0,1/sqrt((lambda.X1.X2[j]*sigma)^2));}
for (j in 1:J){lambda.X1[j]=delta[1]*gamma0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]];}
for(k in 1:K_q[1]){gamma0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){gamma0.X2[k]~dnorm(0,1)}
for(l in 1:Q){delta[l]~dnorm(0,1)}
sigma=abs(sigmarel)
sigma_y=abs(sigma_yrel)
sigma_yrel~dt(0,1/sqrt(5),1)
sigmarel~dt(0,1,1)
alpha0~dnorm(0,.1)
}
```

```
GG<-Generate_all(N=1000,Q=5,p=5)
x<-model.text(GG);x
```

```
model{
for(i in 1:N){y[i]~dnorm(thetastar[j_i[i]],1/sqrt(sigma_y^2));}
for (j in 1:J){thetastar[j]=alpha0+alpha.X1[j]+alpha.X2[j]+alpha.X3[j]+alpha.X4[j]+alpha.X5[
for (j in 1:J){alpha.X1[j]~dnorm(0,1/sqrt((lambda.X1[j]*sigma)^2));}
for (j in 1:J){alpha.X2[j]~dnorm(0,1/sqrt((lambda.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X3[j]~dnorm(0,1/sqrt((lambda.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X4[j]~dnorm(0,1/sqrt((lambda.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X5[j]~dnorm(0,1/sqrt((lambda.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2[j]~dnorm(0,1/sqrt((lambda.X1.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3[j]~dnorm(0,1/sqrt((lambda.X1.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X4[j]~dnorm(0,1/sqrt((lambda.X1.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X5[j]~dnorm(0,1/sqrt((lambda.X1.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3[j]~dnorm(0,1/sqrt((lambda.X2.X3[j]*sigma)^2));}
```

```
for (j in 1:J){alpha.X2.X4[j]~dnorm(0,1/sqrt((lambda.X2.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X5[j]~dnorm(0,1/sqrt((lambda.X2.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X4[j]~dnorm(0,1/sqrt((lambda.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X5[j]~dnorm(0,1/sqrt((lambda.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X4.X5[j]~dnorm(0,1/sqrt((lambda.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X4[j]~dnorm(0,1/sqrt((lambda.X1.X2.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X4[j]~dnorm(0,1/sqrt((lambda.X1.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X5[j]~dnorm(0,1/sqrt((lambda.X1.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X4[j]~dnorm(0,1/sqrt((lambda.X2.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X5[j]~dnorm(0,1/sqrt((lambda.X2.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X4.X5[j]~dnorm(0,1/sqrt((lambda.X2.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X4[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X2.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){lambda.X1[j]=delta[1]*gamma0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X3[j]=delta[1]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X4[j]=delta[1]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X5[j]=delta[1]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X3[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X1.X4[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X1.X5[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X2.X3[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X2.X4[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X2.X5[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X3.X4[j]=delta[2]*gamma0.X3[k_qj[3,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X3.X5[j]=delta[2]*gamma0.X3[k_qj[3,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X4.X5[j]=delta[2]*gamma0.X4[k_qj[4,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2.X3[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X4[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X3.X4[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X1.X3.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X1.X4.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X2.X3.X4[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X2.X3.X5[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X2.X4.X5[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X3.X4.X5[j]=delta[3]*gamma0.X3[k_qj[3,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X3.X4[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X2.X3.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X2.X4.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
```

```
for (j in 1:J){lambda.X1.X3.X4.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamm
for (j in 1:J){lambda.X2.X3.X4.X5[j]=delta[4]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamm
for (j in 1:J){lambda.X1.X2.X3.X4.X5[j]=delta[5]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*g
for(k in 1:K_q[1]){gamma0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){gamma0.X2[k]~dnorm(0,1)}
for(k in 1:K_q[3]){gamma0.X3[k]~dnorm(0,1)}
for(k in 1:K_q[4]){gamma0.X4[k]~dnorm(0,1)}
for(k in 1:K_q[5]){gamma0.X5[k]~dnorm(0,1)}
for(l in 1:Q){delta[l]~dnorm(0,1)}
sigma=abs(sigmarel)
sigma_y=abs(sigma_yrel)
sigma_yrel~dt(0,1/sqrt(5),1)
sigmarel~dt(0,1,1)
alpha0~dnorm(0,.1)
}
```

What it does:

```
  set.seed(2)
  N=1000; Q=4; p=5; K_q=sample(2:p,Q,replace=T)
  XX=Gen_design_variables(N=N,Q=Q,p=p,K_q = K_q)
  hyper=Gen_hyper_parameters(XX,model="model1")
  #hyper$sigma_y<-0.1
GG<-Generate_all(N=N,Q=Q,p=p,nrep=2,K_q = K_q,XX = XX,hyper = hyper,model = "model1")
gibbs.samples<-Trangucci.fit(GG)
  X=data.frame(j=1:GG$XX$J,thetastar=GG$thetastar,t(gibbs.samples[[1]]$BUGSoutput$sims.
     list$thetastar[sample(nrow(gibbs.samples[[1]]$BUGSoutput$sims.list$thetastar),100)
     ,]))
names(X[3:ncol(X)])<-paste0("rep",1:(ncol(X)-2))
XX<-reshape2::melt(X,id.vars=c("j","thetastar"),value.name="sample")
graph1<-ggplot(XX,aes(x=thetastar,y=sample))+geom_point()+geom_abline(slope=1,intercept
   =0)+
   geom_point(aes(x=thetastar,y=thetastar,colour="red"))
XXX<-plyr::ddply(cbind(GG$XX$Xd,y=GG$y[,1]), ~Strata, function(d){data.frame(mean=mean(d
   $y))})
YYY<-data.frame(thetastar=GG$thetastar,posteriormean=as.vector(gibbs.samples[[2]]$
   BUGSoutput$mean$thetastar),GG$XX$Strata)
ZZZ<-merge(XXX,YYY,by="Strata")
ZZZ<-ZZZ[order(ZZZ$thetastar),]
ZZZ$j<-1:nrow(ZZZ)
ZZZ$diffe<-ZZZ$posteriormean-ZZZ$mean
graph5<-ggplot(ZZZ,aes(x=thetastar,     y=mean))+geom_point()+geom_segment(aes(x =
   thetastar, y =mean , xend = thetastar, yend = posteriormean,colour=diffe),linejoin="
   mitre",size=1)+geom_abline(slope=1,intercept=0) +
   scale_colour_gradientn(colours = terrain.colors(10))
r<-ggplot_build(graph5)$layout$panel_params[[1]]$x.range
s<-ggplot_build(graph5)$layout$panel_params[[1]]$y.range
t<-c(min(r[1],s[1]),max(r[2],s[2]))
graph5<-graph5+coord_equal(xlim=t,ylim=t)
graph4<-ggplot(ZZZ,aes(x=thetastar,     y=mean))+geom_point()+geom_abline(slope=1,
   intercept=0)+coord_equal(xlim=t,ylim=t)
graph2<-ggplot(ZZZ,aes(x=thetastar,y=posteriormean))+
   geom_point()+
   geom_abline(slope=1,intercept=0)+
   coord_equal(xlim=t,ylim=t)+geom_smooth()
graph3<-ggplot(ZZZ,aes(x=mean,     y=posteriormean))+geom_point()+geom_abline(slope=1,
   intercept=0)+coord_equal(xlim=t,ylim=t)
```

```
graph6<-ggplot(reshape2::melt(ZZZ[c("j","thetastar","mean","posteriormean")],id.vars=c("
    j")),aes(x=j,y=value,colour=variable))+geom_line()
graph7<-ggplot(reshape2::melt(ZZZ[c("j","thetastar","posteriormean")],id.vars=c("j")),
    aes(x=j,y=value,colour=variable))+geom_line()
   geom_vline(aes(xintercept=GG$hyper$sigma,color="true"))+
     scale_color_manual("",values=c("estimated"="red","true"="darkblue"),labels = c("
        Real value", "Estimated posterior mean"))+ theme(legend.position="bottom")
   geom_vline(aes(xintercept=GG$hyper$sigma_y,color="true"))+
     scale_color_manual("",values=c("estimated"="red","true"="darkblue"),labels = c("
        Estimated posterior mean","True value"))+ theme(legend.position="bottom")+xlab("
        MCMC Replicates")
   geom_vline(xintercept=GG$lambda$`3`[1],colour="red")
   geom_vline(xintercept=GG$lambda$`3`[1],colour="red")
     geom_vline(xintercept=GG$hyper$delta[2],colour="red")
     geom_vline(xintercept=GG$hyper$delta[3],colour="red")
ggplot(data.frame(y=gibbs.samples[[2]]$BUGSoutput$mean$delta,x=GG$hyper$delta),aes(x,y))
    +
   geom_point()+
   geom_abline(intercept=0,slope=1)
X<-data.frame(y=gibbs.samples[[2]]$BUGSoutput$mean$gamma0.X1,x=GG$hyper$gamma0[[1]])
ggplot(X,aes(x,y))+geom_point()+geom_abline(intercept=0,slope=1)
```

Figure 3.1: Scatter plot of $(\theta_j^\star, E[\theta_j^\star|Y])_{j=1}^J$

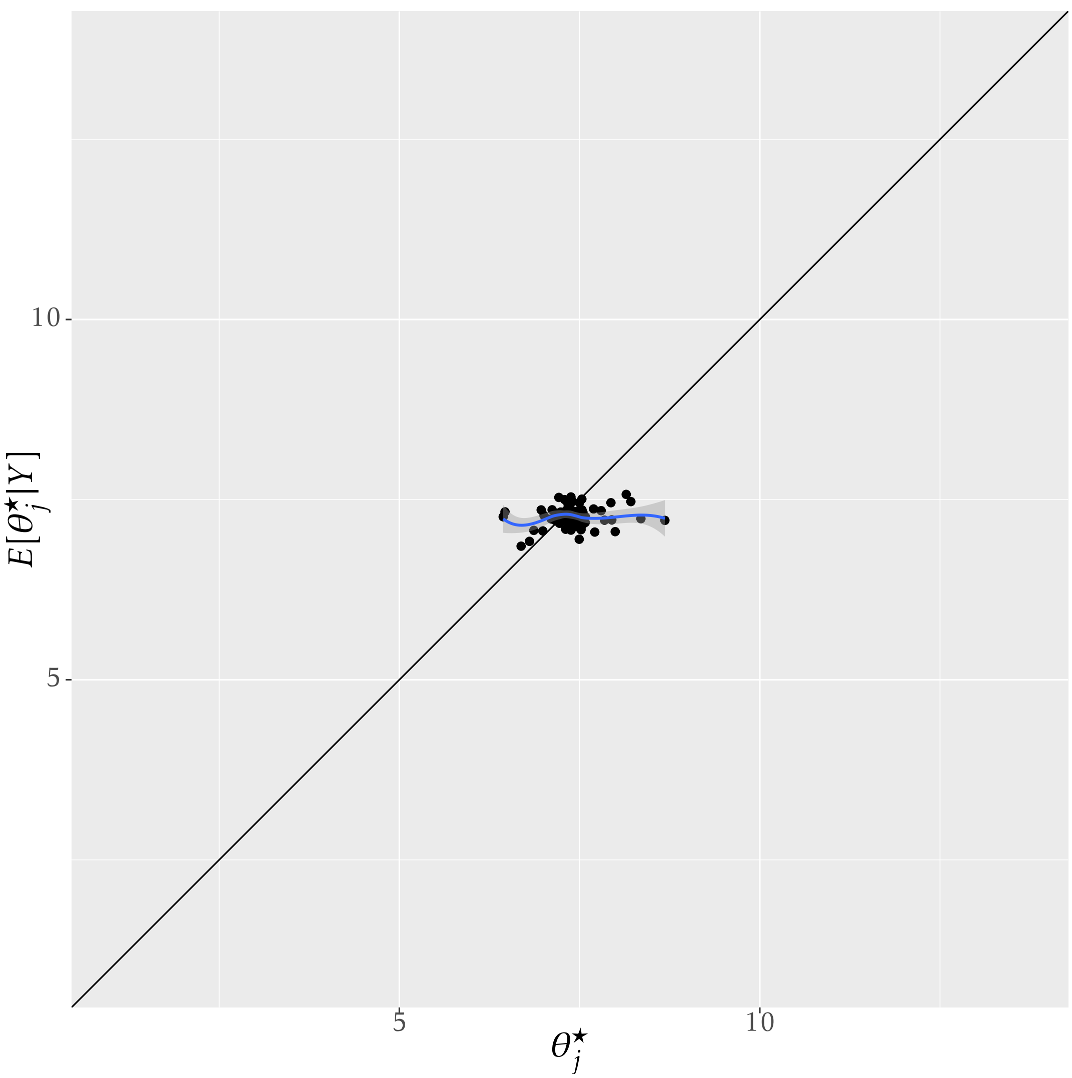


Figure 3.2: Scatter plot of $(\theta_j^\star, 1/N_j \sum_{k \in S_j} y_k$

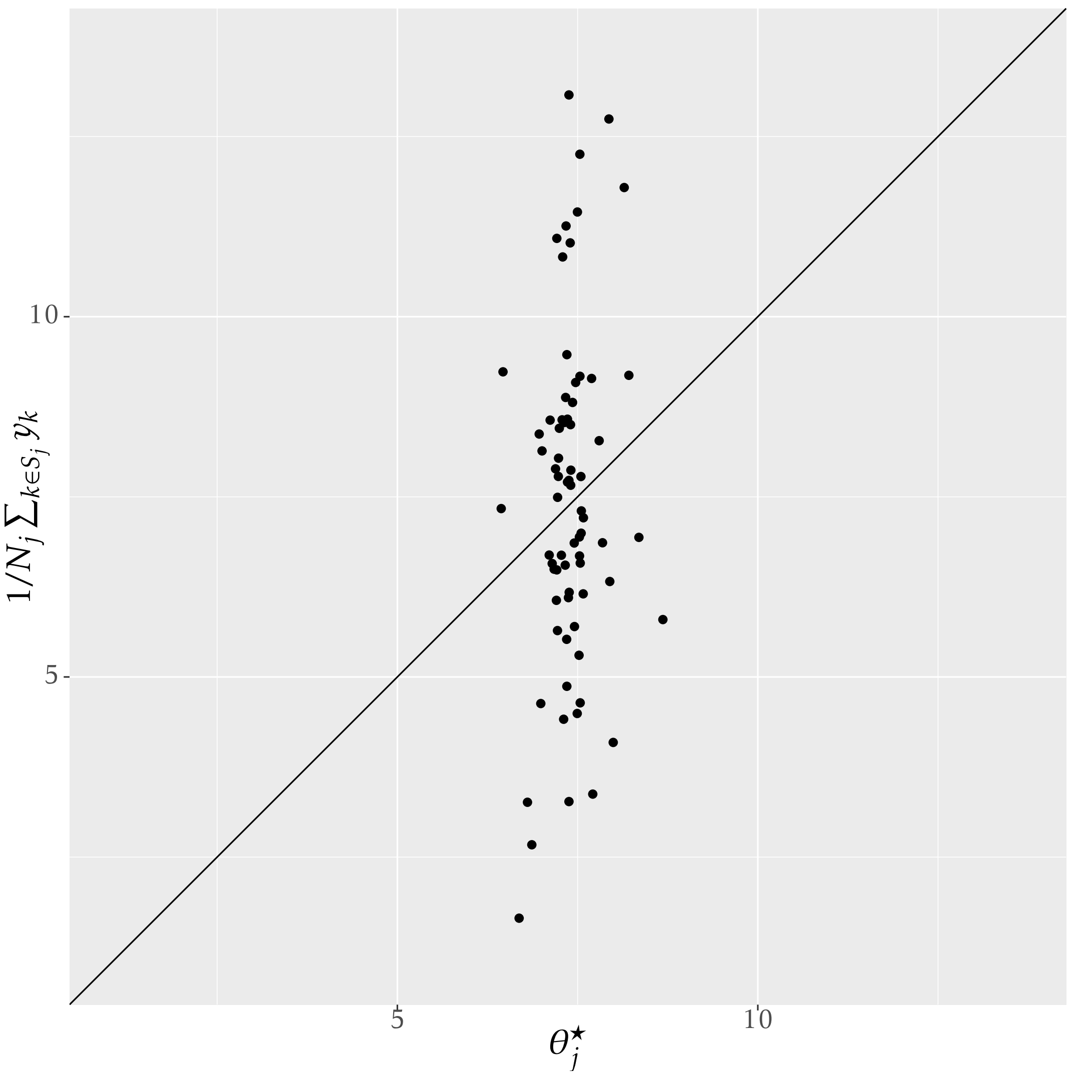
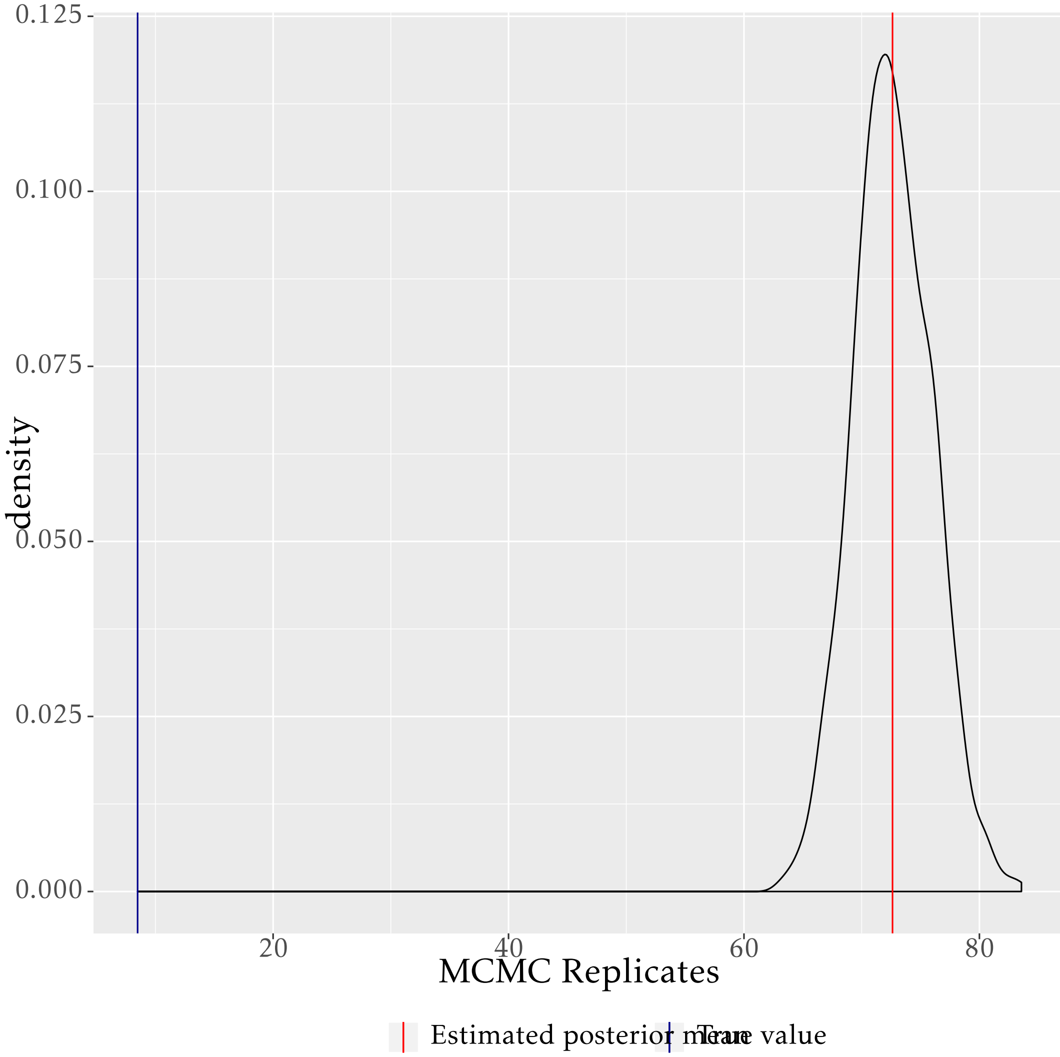


Figure 3.3: Estimated posterior density of (σ_y



Step 2

same thing with Stan

Some functions to generate the jags model file:

```
library(SimuTrangucci)
library(R2jags)
library(ggplot2)
library(plyr)
GG<-Generate_all(N=1000,Q=2,p=5)
x<-model.text(GG);x
```

```
model{
for(i in 1:N){y[i]~dnorm(thetastar[j_i[i]],1/sqrt(sigma_y^2));}
for (j in 1:J){thetastar[j]=alpha0+alpha.X1[j]+alpha.X2[j]+alpha.X1.X2[j];}
for (j in 1:J){alpha.X1[j]~dnorm(0,1/sqrt((lambda.X1[j]*sigma)^2));}
for (j in 1:J){alpha.X2[j]~dnorm(0,1/sqrt((lambda.X2[j]*sigma)^2));}
```

```
for (j in 1:J){alpha.X1.X2[j]~dnorm(0,1/sqrt((lambda.X1.X2[j]*sigma)^2));}
for (j in 1:J){lambda.X1[j]=delta[1]*gamma0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]];}
for(k in 1:K_q[1]){gamma0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){gamma0.X2[k]~dnorm(0,1)}
for(l in 1:Q){delta[l]~dnorm(0,1)}
sigma=abs(sigmarel)
sigma_y=abs(sigma_yrel)
sigma_yrel~dt(0,1/sqrt(5),1)
sigmarel~dt(0,1,1)
alpha0~dnorm(0,.1)
}
```

```
GG<-Generate_all(N=1000,Q=5,p=5)
x<-model.text(GG);x
```

```
model{
for(i in 1:N){y[i]~dnorm(thetastar[j_i[i]],1/sqrt(sigma_y^2));}
for (j in 1:J){thetastar[j]=alpha0+alpha.X1[j]+alpha.X2[j]+alpha.X3[j]+alpha.X4[j]+alpha.X5[
for (j in 1:J){alpha.X1[j]~dnorm(0,1/sqrt((lambda.X1[j]*sigma)^2));}
for (j in 1:J){alpha.X2[j]~dnorm(0,1/sqrt((lambda.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X3[j]~dnorm(0,1/sqrt((lambda.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X4[j]~dnorm(0,1/sqrt((lambda.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X5[j]~dnorm(0,1/sqrt((lambda.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2[j]~dnorm(0,1/sqrt((lambda.X1.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3[j]~dnorm(0,1/sqrt((lambda.X1.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X4[j]~dnorm(0,1/sqrt((lambda.X1.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X5[j]~dnorm(0,1/sqrt((lambda.X1.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3[j]~dnorm(0,1/sqrt((lambda.X2.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X4[j]~dnorm(0,1/sqrt((lambda.X2.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X5[j]~dnorm(0,1/sqrt((lambda.X2.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X4[j]~dnorm(0,1/sqrt((lambda.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X5[j]~dnorm(0,1/sqrt((lambda.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X4.X5[j]~dnorm(0,1/sqrt((lambda.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X4[j]~dnorm(0,1/sqrt((lambda.X1.X2.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X4[j]~dnorm(0,1/sqrt((lambda.X1.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X5[j]~dnorm(0,1/sqrt((lambda.X1.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X4[j]~dnorm(0,1/sqrt((lambda.X2.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X5[j]~dnorm(0,1/sqrt((lambda.X2.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X4.X5[j]~dnorm(0,1/sqrt((lambda.X2.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X4[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X3.X4.X5[j]*sigma)^2));}
```

```
for (j in 1:J){alpha.X2.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X2.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){lambda.X1[j]=delta[1]*gamma0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X3[j]=delta[1]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X4[j]=delta[1]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X5[j]=delta[1]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X3[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X1.X4[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X1.X5[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X2.X3[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X2.X4[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X2.X5[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X3.X4[j]=delta[2]*gamma0.X3[k_qj[3,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X3.X5[j]=delta[2]*gamma0.X3[k_qj[3,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X4.X5[j]=delta[2]*gamma0.X4[k_qj[4,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2.X3[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X4[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X3.X4[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X1.X3.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X1.X4.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X2.X3.X4[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X2.X3.X5[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X2.X4.X5[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X3.X4.X5[j]=delta[3]*gamma0.X3[k_qj[3,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X3.X4[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X2.X3.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X2.X4.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X3.X4.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamm
for (j in 1:J){lambda.X2.X3.X4.X5[j]=delta[4]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamm
for (j in 1:J){lambda.X1.X2.X3.X4.X5[j]=delta[5]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*g
for(k in 1:K_q[1]){gamma0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){gamma0.X2[k]~dnorm(0,1)}
for(k in 1:K_q[3]){gamma0.X3[k]~dnorm(0,1)}
for(k in 1:K_q[4]){gamma0.X4[k]~dnorm(0,1)}
for(k in 1:K_q[5]){gamma0.X5[k]~dnorm(0,1)}
for(l in 1:Q){delta[l]~dnorm(0,1)}
sigma=abs(sigmarel)
sigma_y=abs(sigma_yrel)
sigma_yrel~dt(0,1/sqrt(5),1)
sigmarel~dt(0,1,1)
alpha0~dnorm(0,.1)
}
```

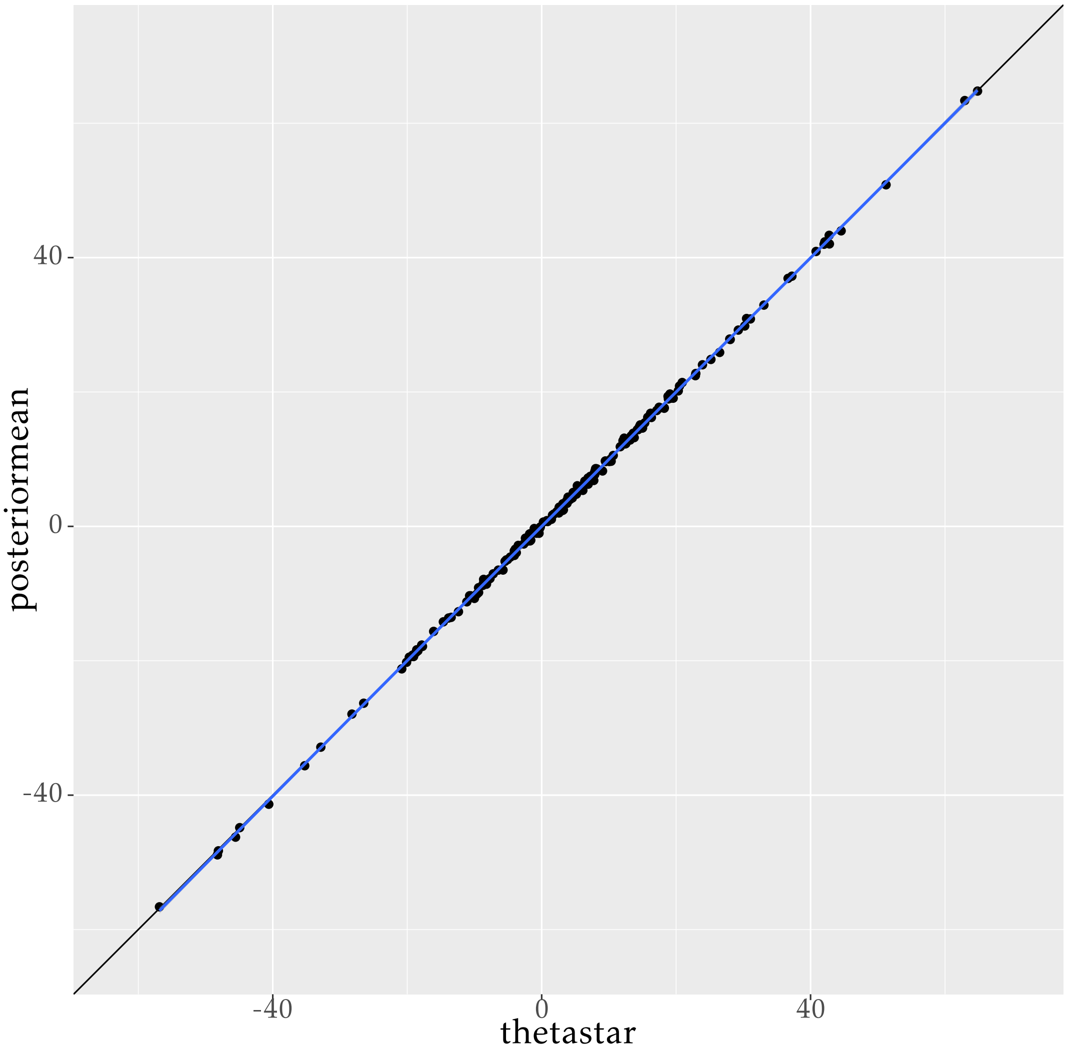
What it does:

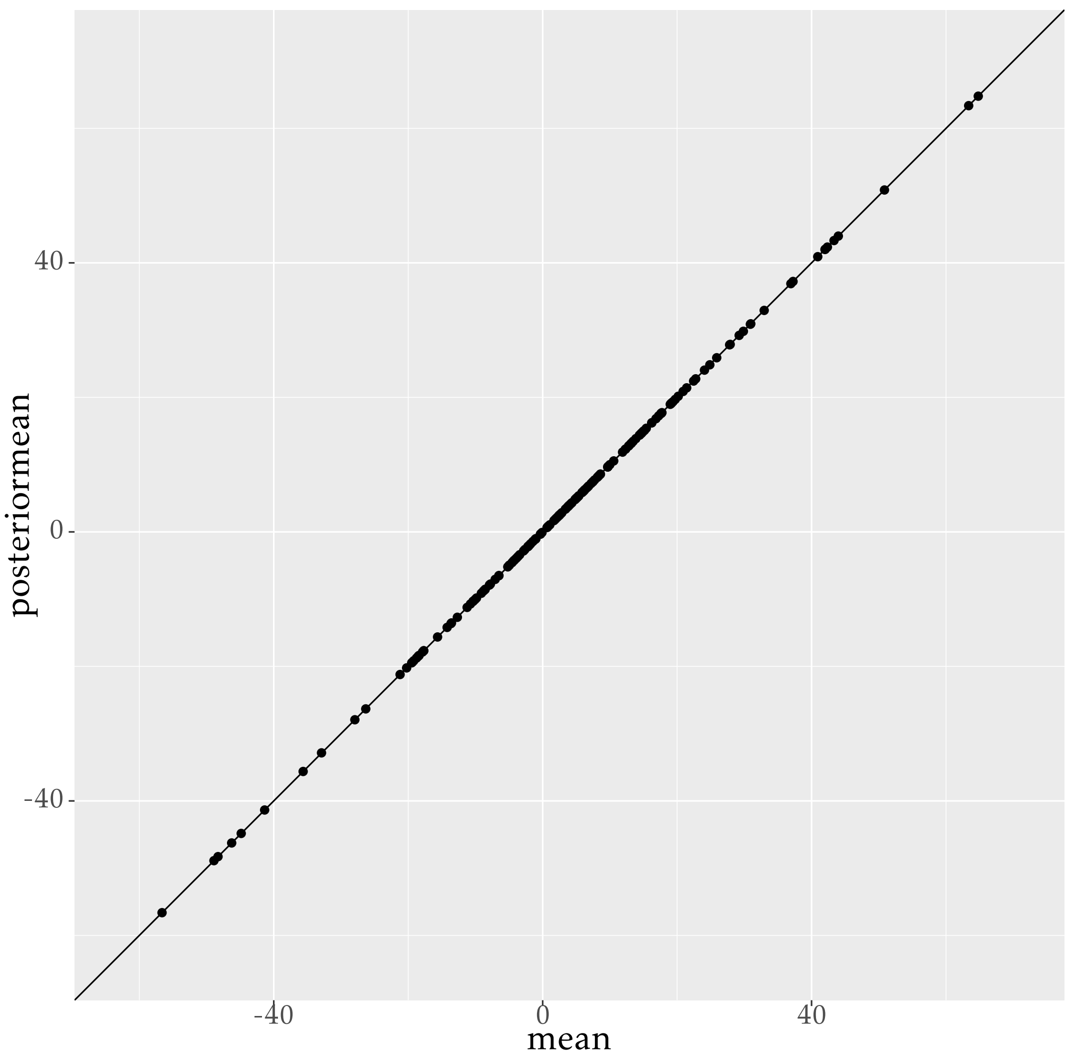
```
set.seed(2)
GG<-Generate_all(N=10000,Q=4,p=5)
```

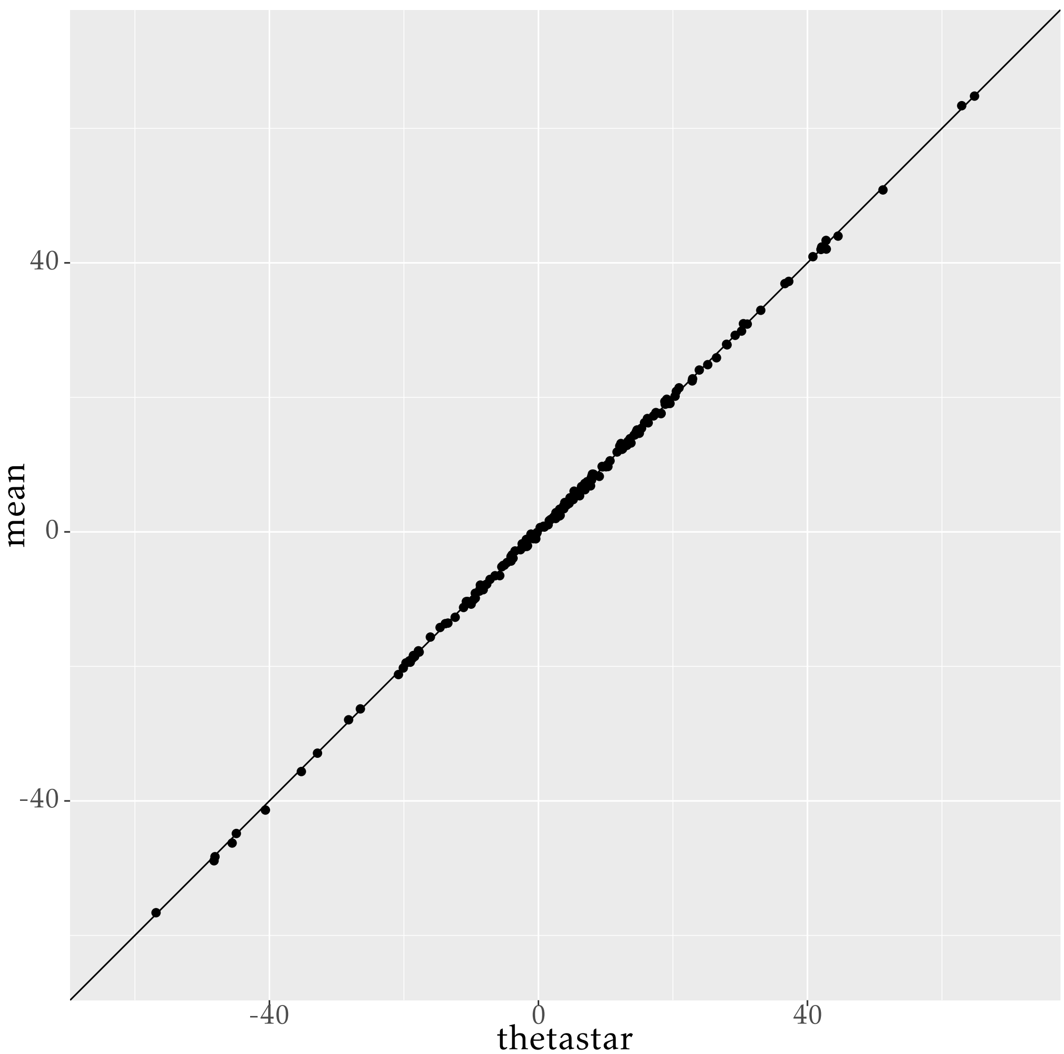
```
GG$XX$K_q
gibbs.samples<-Trangucci.fit(GG)
```

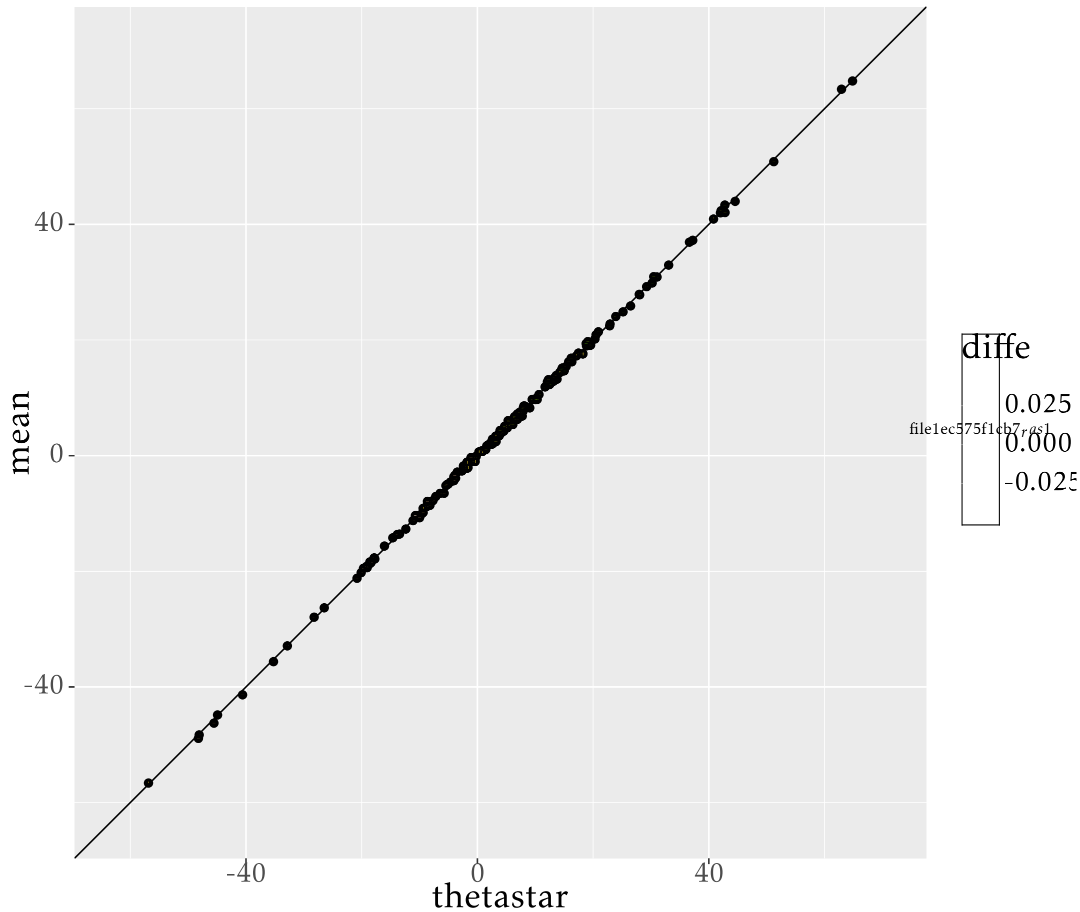
```
[1] 4 5 3 3
```

```
X=data.frame(j=1:GG$XX$J,thetastar=GG$thetastar,t(gibbs.samples[[1]]$BUGSoutput$sims.
    list$thetastar[sample(nrow(gibbs.samples[[1]]$BUGSoutput$sims.list$thetastar),100),])
    )
names(X[3:ncol(X)])<-paste0("rep",1:(ncol(X)-2))
XX<-reshape2::melt(X,id.vars=c("j","thetastar"),value.name="sample")
graph1<-ggplot(XX,aes(x=thetastar,y=sample))+geom_point()+geom_abline(slope=1,intercept
    =0)+
   geom_point(aes(x=thetastar,y=thetastar,colour="red"))
XXX<-plyr::ddply(cbind(GG$XX$Xd,y=GG$y[,1]), ~Strata, function(d){data.frame(mean=mean(d
    $y))})
YYY<-data.frame(thetastar=GG$thetastar,posteriormean=as.vector(gibbs.samples[[2]]$
    BUGSoutput$mean$thetastar),GG$XX$Strata)
ZZZ<-merge(XXX,YYY,by="Strata")
ZZZ<-ZZZ[order(ZZZ$thetastar),]
ZZZ$j<-1:nrow(ZZZ)
ZZZ$diffe<-ZZZ$posteriormean-ZZZ$mean
graph5<-ggplot(ZZZ,aes(x=thetastar,     y=mean))+geom_point()+geom_segment(aes(x =
    thetastar, y =mean , xend = thetastar, yend = posteriormean,colour=diffe),linejoin="
    mitre",size=1)+geom_abline(slope=1,intercept=0) +
   scale_colour_gradientn(colours = terrain.colors(10))
r<-ggplot_build(graph5)$layout$panel_params[[1]]$x.range
s<-ggplot_build(graph5)$layout$panel_params[[1]]$y.range
t<-c(min(r[1],s[1]),max(r[2],s[2]))
graph5<-graph5+coord_equal(xlim=t,ylim=t)
graph4<-ggplot(ZZZ,aes(x=thetastar,     y=mean))+geom_point()+geom_abline(slope=1,
    intercept=0)+coord_equal(xlim=t,ylim=t)
graph2<-ggplot(ZZZ,aes(x=thetastar,y=posteriormean))+geom_point()+geom_abline(slope=1,
    intercept=0)+coord_equal(xlim=t,ylim=t)+geom_smooth()
graph3<-ggplot(ZZZ,aes(x=mean,     y=posteriormean))+geom_point()+geom_abline(slope=1,
    intercept=0)+coord_equal(xlim=t,ylim=t)
graph6<-ggplot(reshape2::melt(ZZZ[c("j","thetastar","mean","posteriormean")],id.vars=c("
    j")),aes(x=j,y=value,colour=variable))+geom_line()
graph7<-ggplot(reshape2::melt(ZZZ[c("j","thetastar","posteriormean")],id.vars=c("j")),
    aes(x=j,y=value,colour=variable))+geom_line()
   geom_vline(xintercept=GG$hyper$sigma,colour="red")
   geom_vline(xintercept=GG$hyper$sigma_y,colour="red")
   geom_vline(xintercept=GG$lambda$`3`[1],colour="red")
   geom_vline(xintercept=GG$lambda$`3`[1],colour="red")
     geom_vline(xintercept=GG$hyper$delta[2],colour="red")
     geom_vline(xintercept=GG$hyper$delta[3],colour="red")
ggplot(data = X,aes(x=x,y=y))+geom_point()+geom_abline(intercept=0,slope=1)
```


posteriormean
40
0
-40
-40
0
40
thetastar


40
0
-40
-40
0
40
posteriormean
mean


40
0
-40
mean
-40
0
40
thetastar



Step 3

1. Design a sampling scheme that favors some cells
2. Compute Trangucci estimator for θ for the $r = 30$ realisations.

Step 4

1. Use the second model to generate the population, and run the analysis on that model.
2. Look at predictions for pop total.

Some functions to generate the jags model file:

```
library(SimuTrangucci)
library(R2jags)
library(ggplot2)
library(plyr)
GG<-Generate_all(N=1000,Q=2,p=5)
x<-model.text(GG);x
```

```
model{
for(i in 1:N){y[i]~dnorm(thetastar[j_i[i]],1/sqrt(sigma_y^2));}
for (j in 1:J){thetastar[j]=alpha0+alpha.X1[j]+alpha.X2[j]+alpha.X1.X2[j];}
for (j in 1:J){alpha.X1[j]~dnorm(0,1/sqrt((lambda.X1[j]*sigma)^2));}
for (j in 1:J){alpha.X2[j]~dnorm(0,1/sqrt((lambda.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2[j]~dnorm(0,1/sqrt((lambda.X1.X2[j]*sigma)^2));}
for (j in 1:J){lambda.X1[j]=delta[1]*gamma0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]];}
for(k in 1:K_q[1]){gamma0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){gamma0.X2[k]~dnorm(0,1)}
for(l in 1:Q){delta[l]~dnorm(0,1)}
sigma=abs(sigmarel)
sigma_y=abs(sigma_yrel)
sigma_yrel~dt(0,1/sqrt(5),1)
sigmarel~dt(0,1,1)
alpha0~dnorm(0,.1)
}
```

```
GG<-Generate_all(N=1000,Q=5,p=5)
x<-model.text(GG);x
```

```
model{
for(i in 1:N){y[i]~dnorm(thetastar[j_i[i]],1/sqrt(sigma_y^2));}
for (j in 1:J){thetastar[j]=alpha0+alpha.X1[j]+alpha.X2[j]+alpha.X3[j]+alpha.X4[j]+alpha.X5[
for (j in 1:J){alpha.X1[j]~dnorm(0,1/sqrt((lambda.X1[j]*sigma)^2));}
for (j in 1:J){alpha.X2[j]~dnorm(0,1/sqrt((lambda.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X3[j]~dnorm(0,1/sqrt((lambda.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X4[j]~dnorm(0,1/sqrt((lambda.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X5[j]~dnorm(0,1/sqrt((lambda.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2[j]~dnorm(0,1/sqrt((lambda.X1.X2[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3[j]~dnorm(0,1/sqrt((lambda.X1.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X4[j]~dnorm(0,1/sqrt((lambda.X1.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X5[j]~dnorm(0,1/sqrt((lambda.X1.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3[j]~dnorm(0,1/sqrt((lambda.X2.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X4[j]~dnorm(0,1/sqrt((lambda.X2.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X5[j]~dnorm(0,1/sqrt((lambda.X2.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X4[j]~dnorm(0,1/sqrt((lambda.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X3.X5[j]~dnorm(0,1/sqrt((lambda.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X4.X5[j]~dnorm(0,1/sqrt((lambda.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X4[j]~dnorm(0,1/sqrt((lambda.X1.X2.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X4[j]~dnorm(0,1/sqrt((lambda.X1.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X5[j]~dnorm(0,1/sqrt((lambda.X1.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X4[j]~dnorm(0,1/sqrt((lambda.X2.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X5[j]~dnorm(0,1/sqrt((lambda.X2.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X4.X5[j]~dnorm(0,1/sqrt((lambda.X2.X4.X5[j]*sigma)^2));}
```

```
for (j in 1:J){alpha.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X4[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X4[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X2.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X2.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){alpha.X1.X2.X3.X4.X5[j]~dnorm(0,1/sqrt((lambda.X1.X2.X3.X4.X5[j]*sigma)^2));}
for (j in 1:J){lambda.X1[j]=delta[1]*gamma0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X3[j]=delta[1]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X4[j]=delta[1]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X5[j]=delta[1]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X3[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X1.X4[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X1.X5[j]=delta[2]*gamma0.X1[k_qj[1,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X2.X3[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X2.X4[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X2.X5[j]=delta[2]*gamma0.X2[k_qj[2,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X3.X4[j]=delta[2]*gamma0.X3[k_qj[3,j]]*gamma0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X3.X5[j]=delta[2]*gamma0.X3[k_qj[3,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X4.X5[j]=delta[2]*gamma0.X4[k_qj[4,j]]*gamma0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2.X3[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X4[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamma0.
for (j in 1:J){lambda.X1.X3.X4[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X1.X3.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X1.X4.X5[j]=delta[3]*gamma0.X1[k_qj[1,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X2.X3.X4[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X2.X3.X5[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamma0.
for (j in 1:J){lambda.X2.X4.X5[j]=delta[3]*gamma0.X2[k_qj[2,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X3.X4.X5[j]=delta[3]*gamma0.X3[k_qj[3,j]]*gamma0.X4[k_qj[4,j]]*gamma0.
for (j in 1:J){lambda.X1.X2.X3.X4[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X2.X3.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X2.X4.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*gamm
for (j in 1:J){lambda.X1.X3.X4.X5[j]=delta[4]*gamma0.X1[k_qj[1,j]]*gamma0.X3[k_qj[3,j]]*gamm
for (j in 1:J){lambda.X2.X3.X4.X5[j]=delta[4]*gamma0.X2[k_qj[2,j]]*gamma0.X3[k_qj[3,j]]*gamm
for (j in 1:J){lambda.X1.X2.X3.X4.X5[j]=delta[5]*gamma0.X1[k_qj[1,j]]*gamma0.X2[k_qj[2,j]]*g
for(k in 1:K_q[1]){gamma0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){gamma0.X2[k]~dnorm(0,1)}
for(k in 1:K_q[3]){gamma0.X3[k]~dnorm(0,1)}
for(k in 1:K_q[4]){gamma0.X4[k]~dnorm(0,1)}
for(k in 1:K_q[5]){gamma0.X5[k]~dnorm(0,1)}
for(l in 1:Q){delta[l]~dnorm(0,1)}
sigma=abs(sigmarel)
sigma_y=abs(sigma_yrel)
sigma_yrel~dt(0,1/sqrt(5),1)
sigmarel~dt(0,1,1)
alpha0~dnorm(0,.1)
```

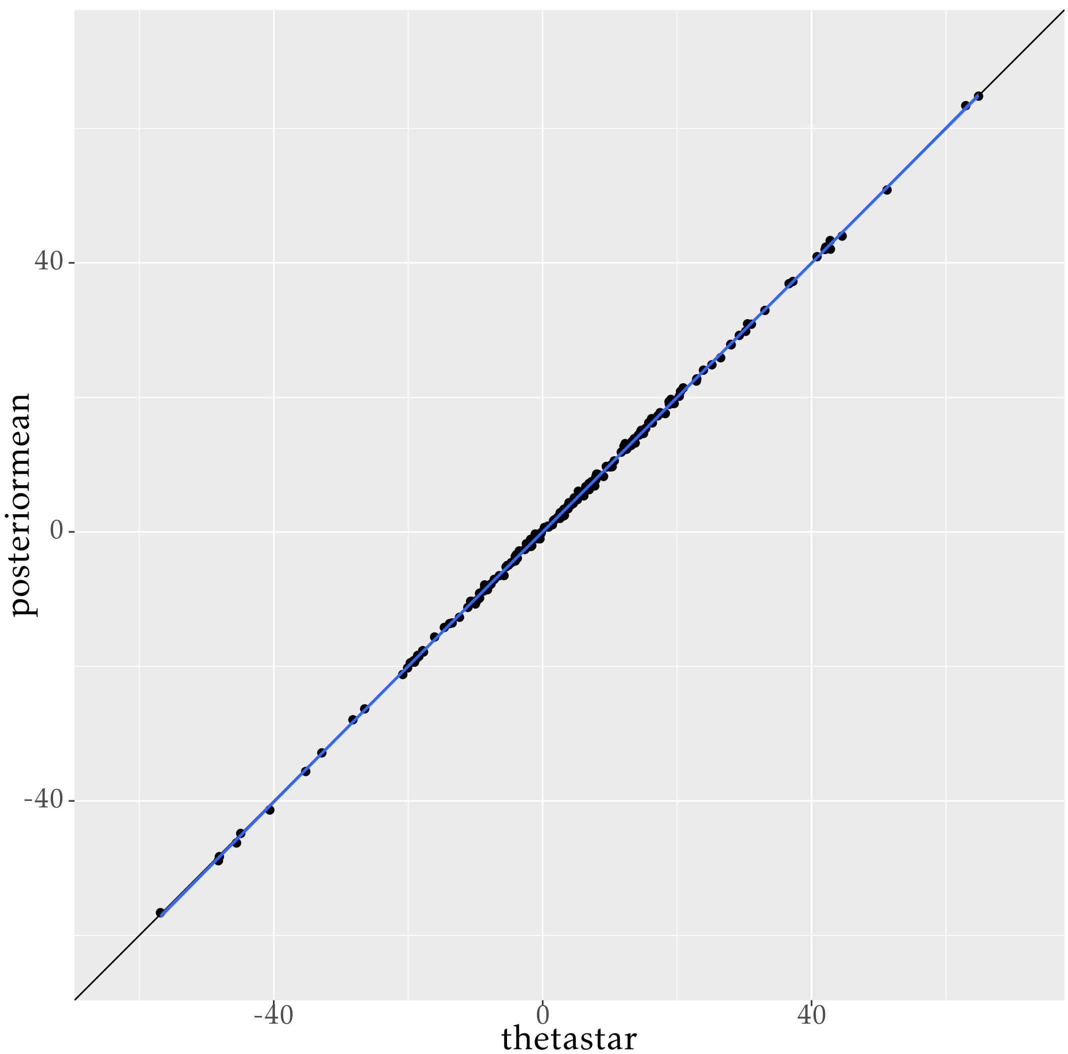
}

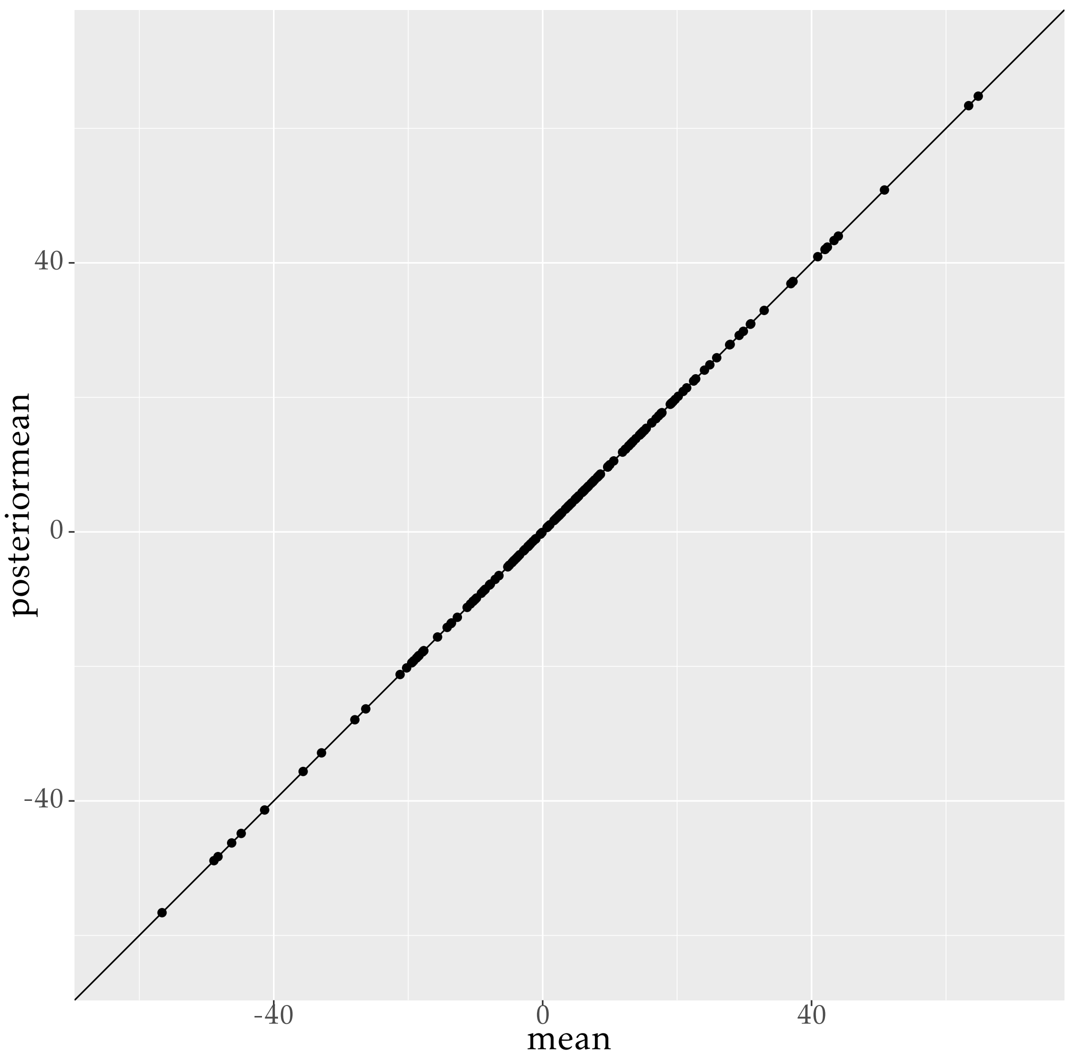
What it does:

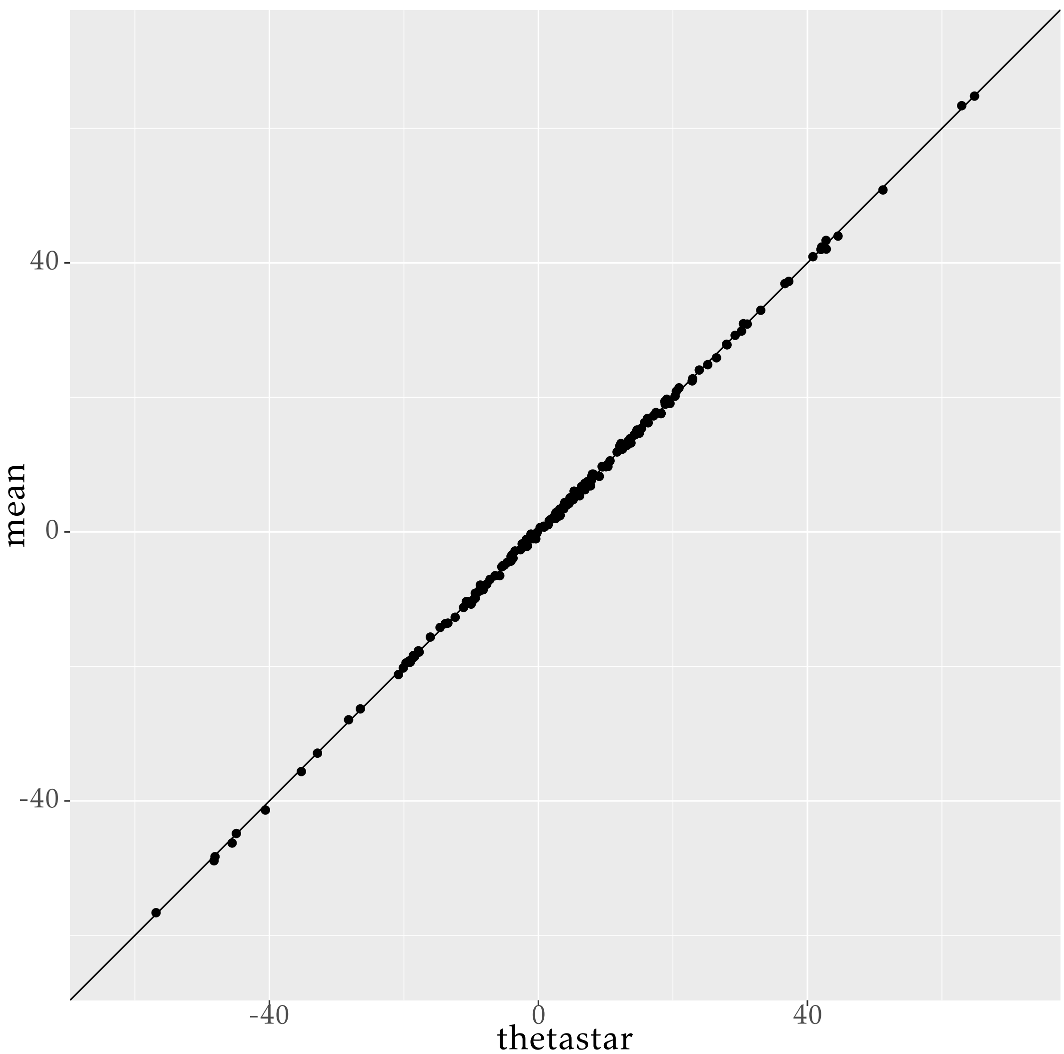
```
set.seed(2)
GG<-Generate_all(N=10000,Q=4,p=5)
GG$XX$K_q
gibbs.samples<-Trangucci.fit(GG)
```

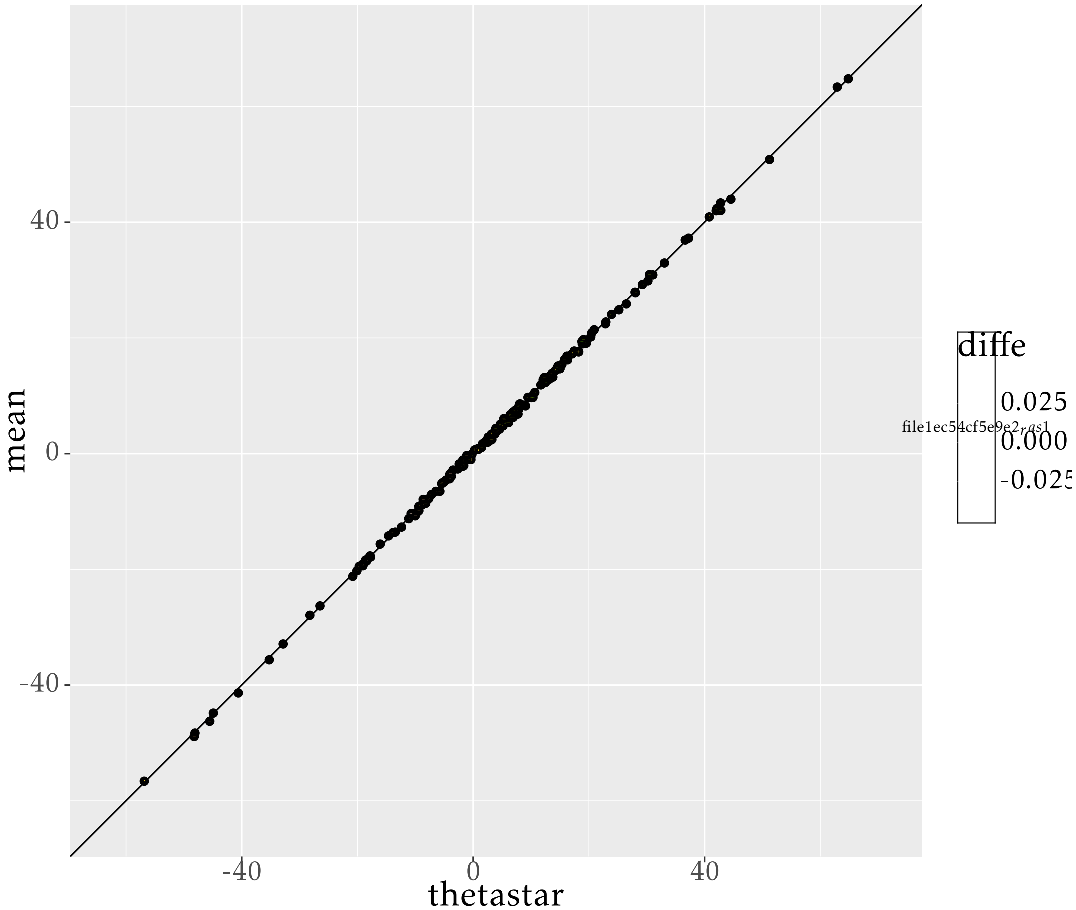
```
[1] 4 5 3 3
```

```
X=data.frame(j=1:GG$XX$J,thetastar=GG$thetastar,t(gibbs.samples[[1]]$BUGSoutput$sims.
    list$thetastar[sample(nrow(gibbs.samples[[1]]$BUGSoutput$sims.list$thetastar),100),])
    )
names(X[3:ncol(X)])<-paste0("rep",1:(ncol(X)-2))
XX<-reshape2::melt(X,id.vars=c("j","thetastar"),value.name="sample")
graph1<-ggplot(XX,aes(x=thetastar,y=sample))+geom_point()+geom_abline(slope=1,intercept
    =0)+
   geom_point(aes(x=thetastar,y=thetastar,colour="red"))
XXX<-plyr::ddply(cbind(GG$XX$Xd,y=GG$y[,1]), ~Strata, function(d){data.frame(mean=mean(d
    $y))})
YYY<-data.frame(thetastar=GG$thetastar,posteriormean=as.vector(gibbs.samples[[2]]$
    BUGSoutput$mean$thetastar),GG$XX$Strata)
ZZZ<-merge(XXX,YYY,by="Strata")
ZZZ<-ZZZ[order(ZZZ$thetastar),]
ZZZ$j<-1:nrow(ZZZ)
ZZZ$diffe<-ZZZ$posteriormean-ZZZ$mean
graph5<-ggplot(ZZZ,aes(x=thetastar,     y=mean))+geom_point()+geom_segment(aes(x =
    thetastar, y =mean , xend = thetastar, yend = posteriormean,colour=diffe),linejoin="
    mitre",size=1)+geom_abline(slope=1,intercept=0) +
   scale_colour_gradientn(colours = terrain.colors(10))
r<-ggplot_build(graph5)$layout$panel_params[[1]]$x.range
s<-ggplot_build(graph5)$layout$panel_params[[1]]$y.range
t<-c(min(r[1],s[1]),max(r[2],s[2]))
graph5<-graph5+coord_equal(xlim=t,ylim=t)
graph4<-ggplot(ZZZ,aes(x=thetastar,     y=mean))+geom_point()+geom_abline(slope=1,
    intercept=0)+coord_equal(xlim=t,ylim=t)
graph2<-ggplot(ZZZ,aes(x=thetastar,y=posteriormean))+geom_point()+geom_abline(slope=1,
    intercept=0)+coord_equal(xlim=t,ylim=t)+geom_smooth()
graph3<-ggplot(ZZZ,aes(x=mean,     y=posteriormean))+geom_point()+geom_abline(slope=1,
    intercept=0)+coord_equal(xlim=t,ylim=t)
graph6<-ggplot(reshape2::melt(ZZZ[c("j","thetastar","mean","posteriormean")],id.vars=c("
    j")),aes(x=j,y=value,colour=variable))+geom_line()
graph7<-ggplot(reshape2::melt(ZZZ[c("j","thetastar","posteriormean")],id.vars=c("j")),
    aes(x=j,y=value,colour=variable))+geom_line()
   geom_vline(xintercept=GG$hyper$sigma,colour="red")
   geom_vline(xintercept=GG$hyper$sigma_y,colour="red")
   geom_vline(xintercept=GG$lambda$`3`[1],colour="red")
   geom_vline(xintercept=GG$lambda$`3`[1],colour="red")
     geom_vline(xintercept=GG$hyper$delta[2],colour="red")
     geom_vline(xintercept=GG$hyper$delta[3],colour="red")
ggplot(data = X,aes(x=x,y=y))+geom_point()+geom_abline(intercept=0,slope=1)
```


40
0
-40
posteriormean
-40
0
40
thetastar


posteriormean
40
0
-40
-40
0
40
mean


40
0
-40
mean
-40
0
40
thetastar



4

Bayesian computations