Simulations on "Bayesian hierarchical weighting adjustment and survey inference" by Tranguccci 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:

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

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

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

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

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

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

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

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

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

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:

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

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

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

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

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

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

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, $(\delta^\ell)_{\ell=1,\dots,Q}$,$\lambda^l_k$,$\sigma$.
4. compute the $\lambda^{(q_1,\dots,q_\ell)}_{k_1,\dots,k_\ell}$, for possible values of $k_1,\dots,k_\ell{}^{(q_1,\dots,q_\ell)}$.
5. generate the $\alpha^{(q_1,\dots,q_\ell)}_{j,(k_1,\dots,k_\ell)}$.
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	73	0.055
S3	1	3	S3	83	2.687
S4	1	4	S4	86	2.681
S5	2	1	S5	91	-5.146
S6	2	2	S6	82	-10.338
S7	2	3	S7	80	3.576
S8	2	4	S8	78	0.260
S9	3	1	S9	95	6.074
S10	3	2	S10	80	0.900
S11	3	3	S11	91	2.567
S12	3	4	S12	70	-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]*lambda0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*lambda0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]];}
for(k in 1:K_q[1]){lambda0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){lambda0.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]*lambda0.X1[k_qj[1,j]];}
for (j in 1:J){lambda.X2[j]=delta[1]*lambda0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X3[j]=delta[1]*lambda0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X4[j]=delta[1]*lambda0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X5[j]=delta[1]*lambda0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2[j]=delta[2]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]];}
for (j in 1:J){lambda.X1.X3[j]=delta[2]*lambda0.X1[k_qj[1,j]]*lambda0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X1.X4[j]=delta[2]*lambda0.X1[k_qj[1,j]]*lambda0.X4[k_qj[4,j]];}
```

```
for (j in 1:J){lambda.X1.X5[j]=delta[2]*lambda0.X1[k_qj[1,j]]*lambda0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X2.X3[j]=delta[2]*lambda0.X2[k_qj[2,j]]*lambda0.X3[k_qj[3,j]];}
for (j in 1:J){lambda.X2.X4[j]=delta[2]*lambda0.X2[k_qj[2,j]]*lambda0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X2.X5[j]=delta[2]*lambda0.X2[k_qj[2,j]]*lambda0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X3.X4[j]=delta[2]*lambda0.X3[k_qj[3,j]]*lambda0.X4[k_qj[4,j]];}
for (j in 1:J){lambda.X3.X5[j]=delta[2]*lambda0.X3[k_qj[3,j]]*lambda0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X4.X5[j]=delta[2]*lambda0.X4[k_qj[4,j]]*lambda0.X5[k_qj[5,j]];}
for (j in 1:J){lambda.X1.X2.X3[j]=delta[3]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]*lambd
for (j in 1:J){lambda.X1.X2.X4[j]=delta[3]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]*lambd
for (j in 1:J){lambda.X1.X2.X5[j]=delta[3]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]*lambd
for (j in 1:J){lambda.X1.X3.X4[j]=delta[3]*lambda0.X1[k_qj[1,j]]*lambda0.X3[k_qj[3,j]]*lambd
for (j in 1:J){lambda.X1.X3.X5[j]=delta[3]*lambda0.X1[k_qj[1,j]]*lambda0.X3[k_qj[3,j]]*lambd
for (j in 1:J){lambda.X1.X4.X5[j]=delta[3]*lambda0.X1[k_qj[1,j]]*lambda0.X4[k_qj[4,j]]*lambd
for (j in 1:J){lambda.X2.X3.X4[j]=delta[3]*lambda0.X2[k_qj[2,j]]*lambda0.X3[k_qj[3,j]]*lambd
for (j in 1:J){lambda.X2.X3.X5[j]=delta[3]*lambda0.X2[k_qj[2,j]]*lambda0.X3[k_qj[3,j]]*lambd
for (j in 1:J){lambda.X2.X4.X5[j]=delta[3]*lambda0.X2[k_qj[2,j]]*lambda0.X4[k_qj[4,j]]*lambd
for (j in 1:J){lambda.X3.X4.X5[j]=delta[3]*lambda0.X3[k_qj[3,j]]*lambda0.X4[k_qj[4,j]]*lambd
for (j in 1:J){lambda.X1.X2.X3.X4[j]=delta[4]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]*la
for (j in 1:J){lambda.X1.X2.X3.X5[j]=delta[4]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]*la
for (j in 1:J){lambda.X1.X2.X4.X5[j]=delta[4]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]*la
for (j in 1:J){lambda.X1.X3.X4.X5[j]=delta[4]*lambda0.X1[k_qj[1,j]]*lambda0.X3[k_qj[3,j]]*la
for (j in 1:J){lambda.X2.X3.X4.X5[j]=delta[4]*lambda0.X2[k_qj[2,j]]*lambda0.X3[k_qj[3,j]]*la
for (j in 1:J){lambda.X1.X2.X3.X4.X5[j]=delta[5]*lambda0.X1[k_qj[1,j]]*lambda0.X2[k_qj[2,j]]
for(k in 1:K_q[1]){lambda0.X1[k]~dnorm(0,1)}
for(k in 1:K_q[2]){lambda0.X2[k]~dnorm(0,1)}
for(k in 1:K_q[3]){lambda0.X3[k]~dnorm(0,1)}
for(k in 1:K_q[4]){lambda0.X4[k]~dnorm(0,1)}
for(k in 1:K_q[5]){lambda0.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)
}
```

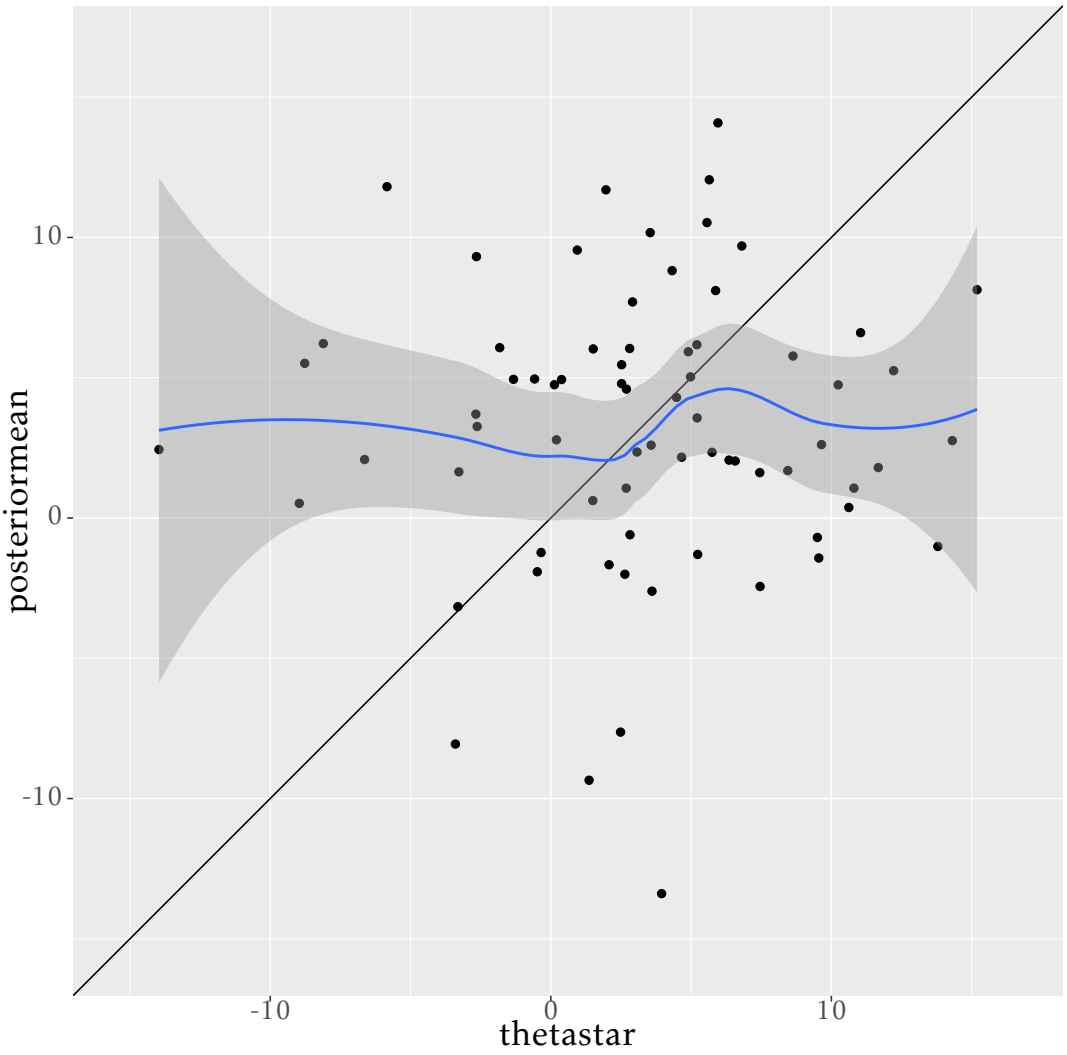
What it does:

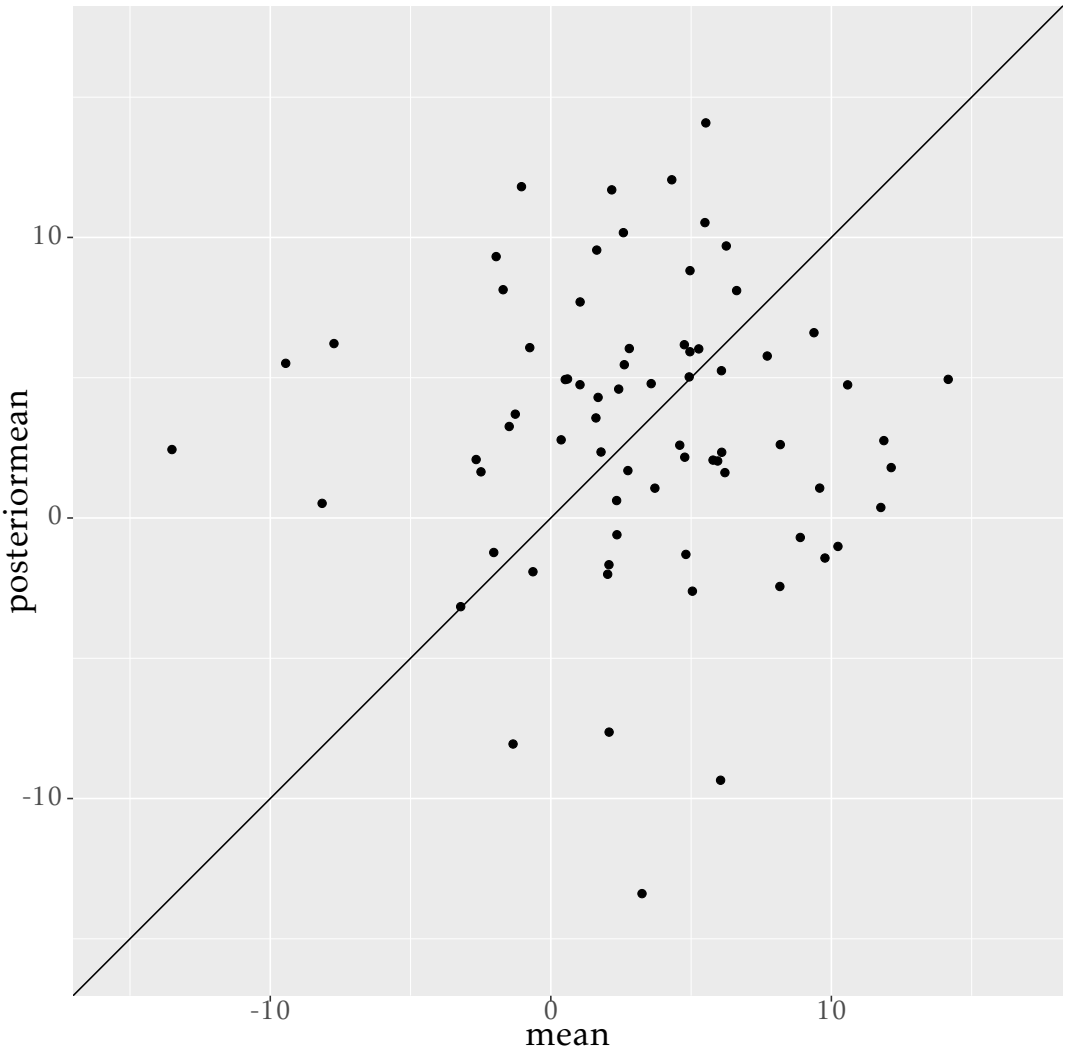
```
set.seed(2)
GG<-Generate_all(N=1000,Q=4,p=5)
GG$XX$K_q
gibbs.samples<-Trangucci.fit(GG)
```

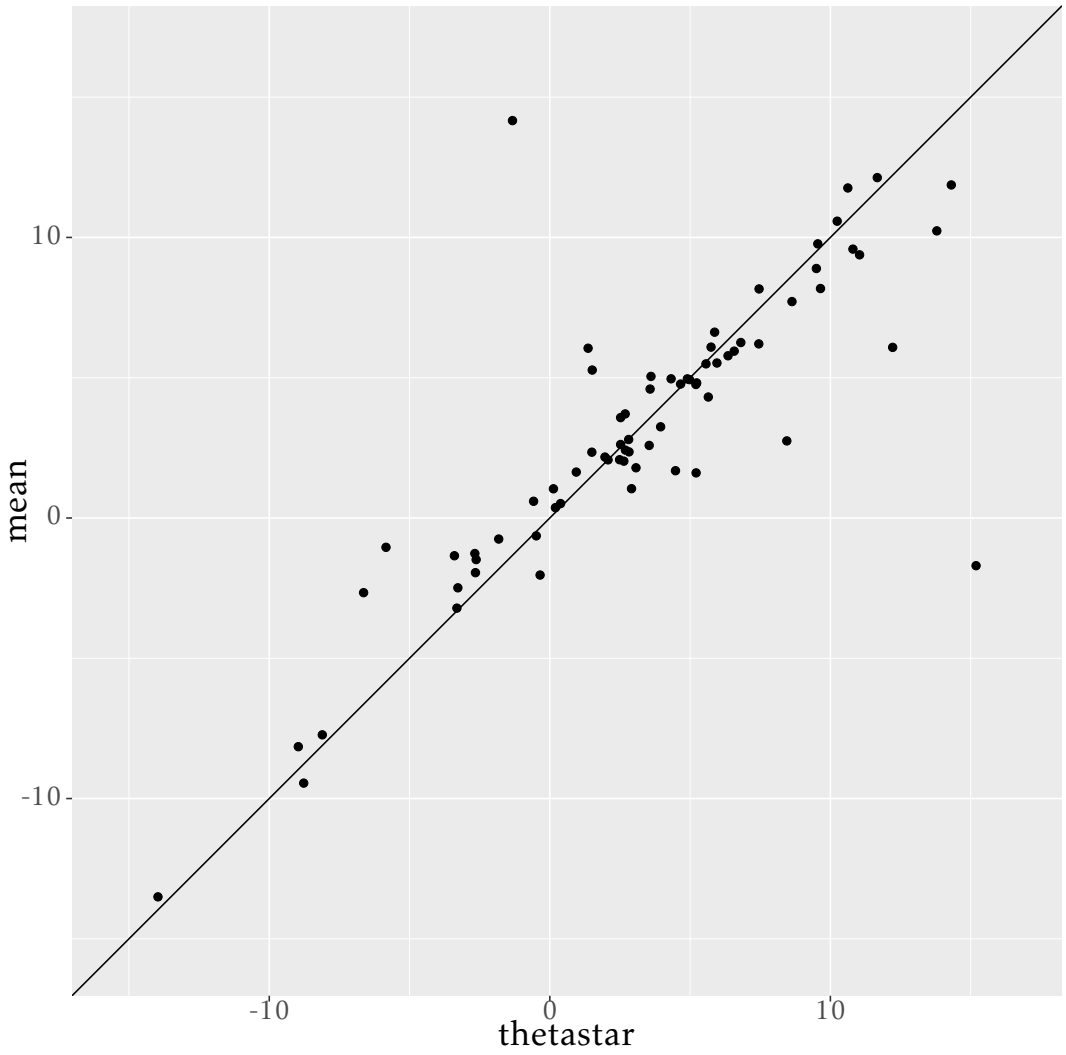
```
[1] 3 2 3 4
```

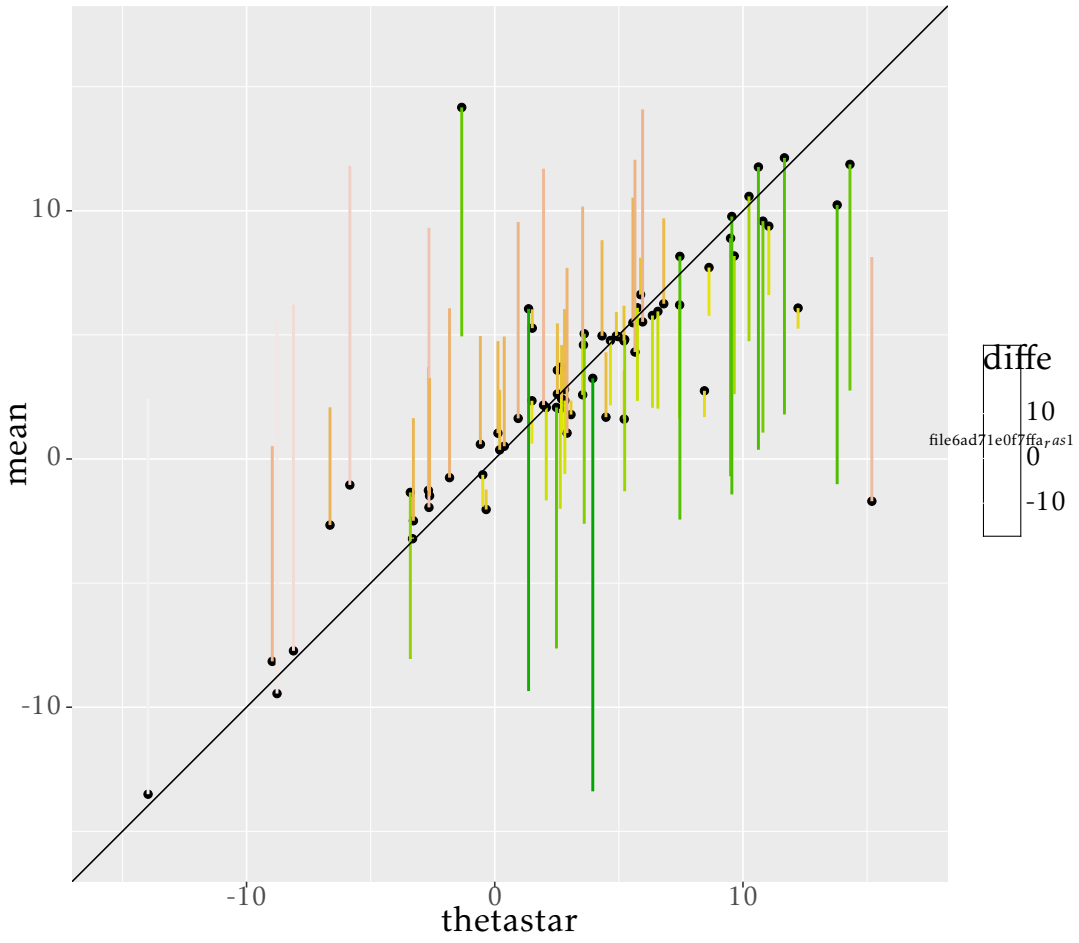
```
  X=data.frame(j=1:GG$XX$J,thetastar=GG$thetastar,t(gibbs.samples[[1]]$BUGSoutput$sims.
      list$thetastar[sample(nrow(gibbs.samples[[1]]$BUGSoutput$sims.list$thetastar),100)
      ,]))
names(X[3:ncol(X)])<-paste0("rep",1:(ncol(X)-2))
XX<-reshape2::melt(X,id.vars=c("j","thetastar"),value.name="sample")
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(X,aes(x,y))+geom_point()+geom_abline(intercept=0,slope=1)
```


10
0
-10
posteriormean
-10
0
10
thetastar


10
0
-10
posteriormean
-10
0
10
mean


10
0
-10
mean
-10
0
10
thetastar



Step 2

same thing with Stan

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 another model to generate the population, a model that does not fit the current one, to fail the product structure.
2. Look at predictions for pop total.

1. Assume the following model:
2.

4 Bayesian computations