BDA Final Project Report- Tin Nguyen

EFFECTS OF THE COMPONENTS - OF THE COMPUTERS ON THEIR PRICES

12/3/2021

Contents

LOAD PACKAGES

```
library(aaltobda)
library(loo)
library(bayesplot)

library(brms)
library(rstan)
library(shinystan)
library(tidyverse)

install.packages("remotes")
remotes::install_github("rmcelreath/rethinking",upgrade="never")

theme_set(bayesplot::theme_default (base_family="sans" ,base_size =16))
```

INTRODUCTION

Motivation

In this day and age, personal computers (PCs), laptops are ubiquitous. They become a must-have item for studying, for professional jobs. It is very likely that once in a while, you would have a need to make a purchase. Given that this is such a rather huge expenditure, you may have to put aside some savings for a while before being able to afford it.

Conceive of the case that you are in need of the latest version of the laptop which would be released three months from now. How would you start the saving plan? How much should this everyday lump be? You could not just allocate all budget you have just for saving to buy that computer right? You still have other needs to be fulfilled right?

Given the fixed time (3 months), it would be great if you are able to have a rather accurate estimate of the price of the computers so that you could set the just-right amount to save from now. That is the motivation for us: to be able to estimate the price of the computers. Is that possible?

Modelling idea

+It is very often that before purchasing the computers, we often have an idea about the specs of computer we want (how many RAM is needed? How fast the clock speed should be, etc). It is possible that we could base on the information about the components of the computers to provide an estimate for the price.

+Also, we could grasp a better understanding that which component has a stronger effect on the price (i.e: maybe you need a PC with larger screen which would not have much influence on the price as much as more RAM would).

+Is the effect of the components on the price fixed? Is that true that the computers which have just been released would be set at premium price and very soon, the price tapers off?

Would you like to know how we could learn these?

Let's dive further into it!

Overview

So, here is an overview of the structure of this report:

We firstly provided you with a description of the dataset we collected and defined clearly what was our analysis problem. We determined the likelihood for two models.

Given that we have two distinct models, we did all the necessary analysis for the first model, and then we underwent the same procedure for the other model. These analyses incorporate these steps:

+set prior and justify for prior choices

+analyze the convergence matters

+run a posterior predictive check

+compute elpd

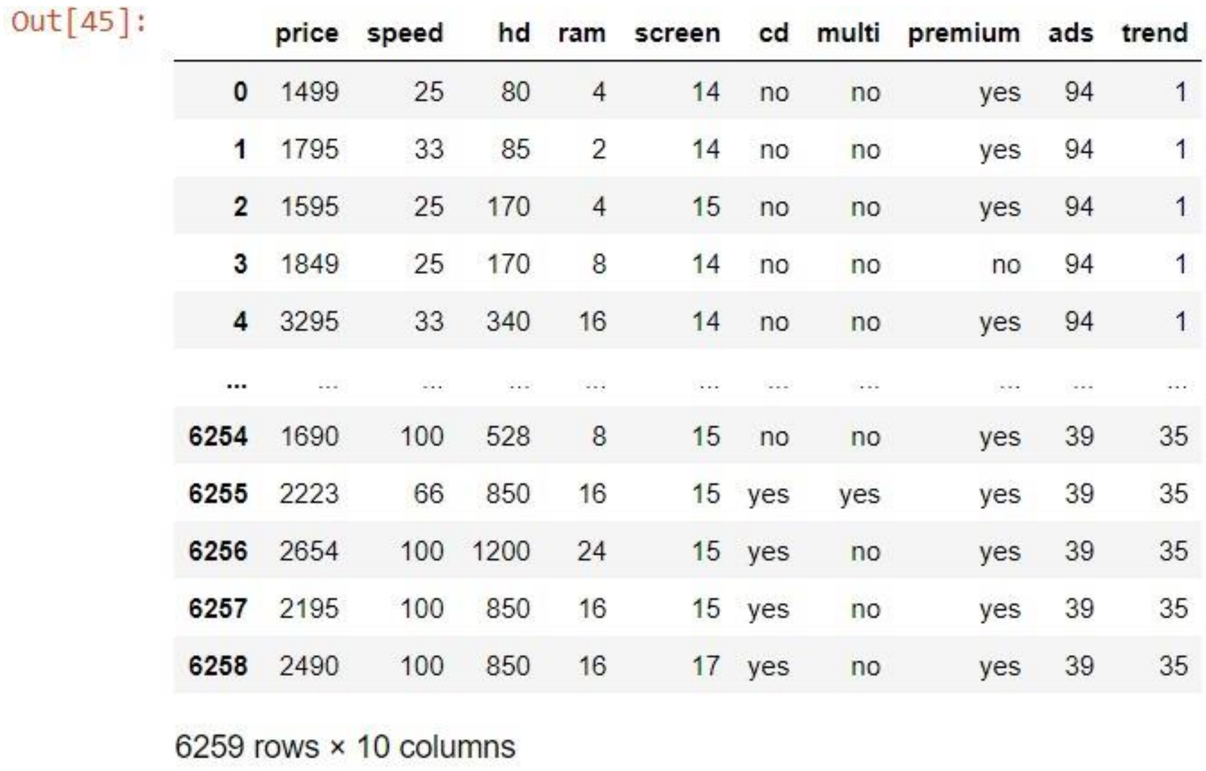
After these separate analyses for each model, we did model comparison as well as predictive performance assessment for two models. After having an idea which model surpassed the other, we ran a sensitivity analysis on the winner to check its robustness.

Finally, we came to the discussion part.

DATA DESCRIPTION AND ANALYSIS PROBLEM

Data description

The original data is called “computers” dataset which is an at hand option in R offering. It is about the “486 processor”



Out[45]:

	price	speed	hd	ram	screen	cd	multi	premium	ads	trend
0	1499	25	80	4	14	no	no	yes	94	1
1	1795	33	85	2	14	no	no	yes	94	1
2	1595	25	170	4	15	no	no	yes	94	1
3	1849	25	170	8	14	no	no	no	94	1
4	3295	33	340	16	14	no	no	yes	94	1
...	...	...	...	...	...	...	...	...	...	...
6254	1690	100	528	8	15	no	no	yes	39	35
6255	2223	66	850	16	15	yes	yes	yes	39	35
6256	2654	100	1200	24	15	yes	no	yes	39	35
6257	2195	100	850	16	15	yes	no	yes	39	35
6258	2490	100	850	16	17	yes	no	yes	39	35

6259 rows × 10 columns

The structure of data is a dataframe whose shape is 6259x10

10 columns including:

+7 non-binary variables:

the price,

the clock speed (in MHz) of the 486 processor, the size of the hard drive (in MB),

the size of the RAM (in MB),

the size of the screen (in inches),

the total number of ads.

the trend (i.e: The data is collected on the first issue of PC magazine of every month from January of 1993 to November of 1995. That is shown in the trend column whose values are from 1 to 35)

+3 binary variables:

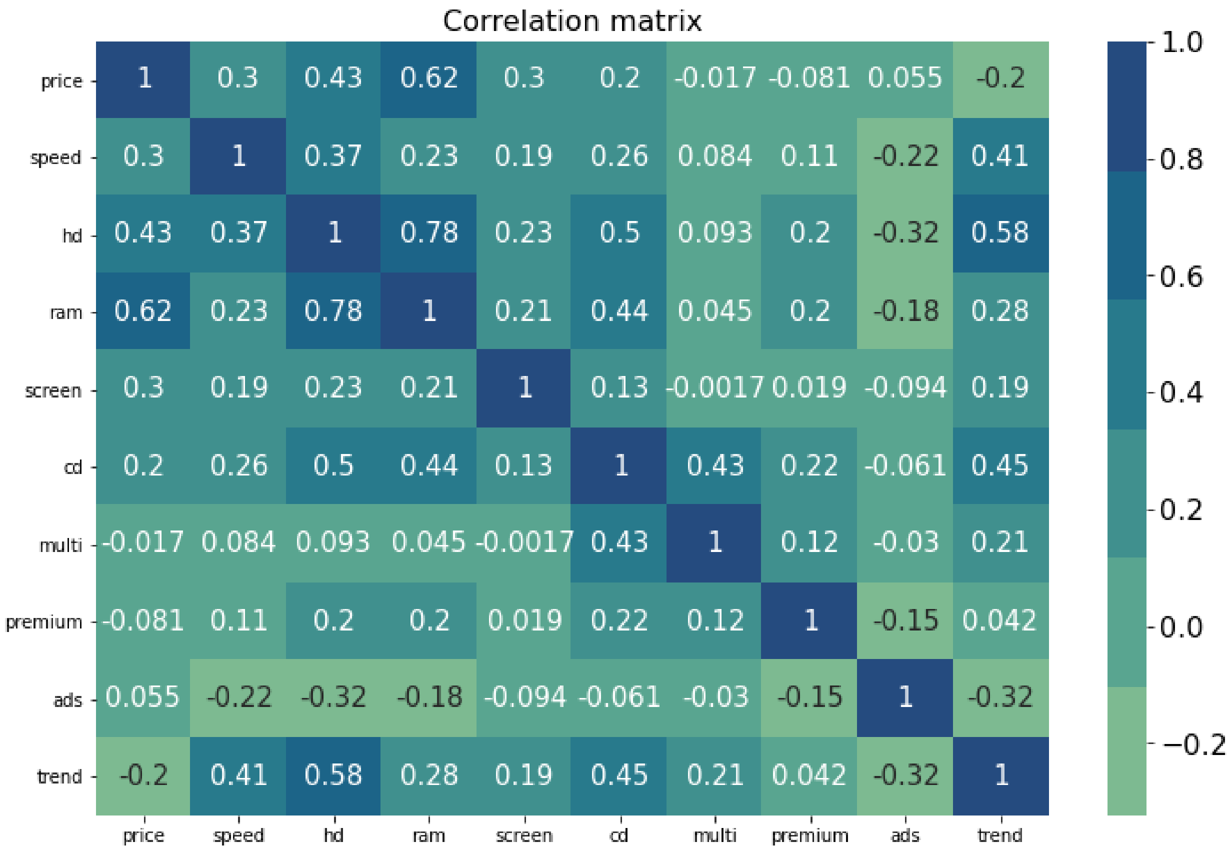
the presence of a CD-ROM,

the presence of a multi-media kit that includes speakers and a sound card,

whether the manufacturer was a “premium” firm or not.

Analysis problem

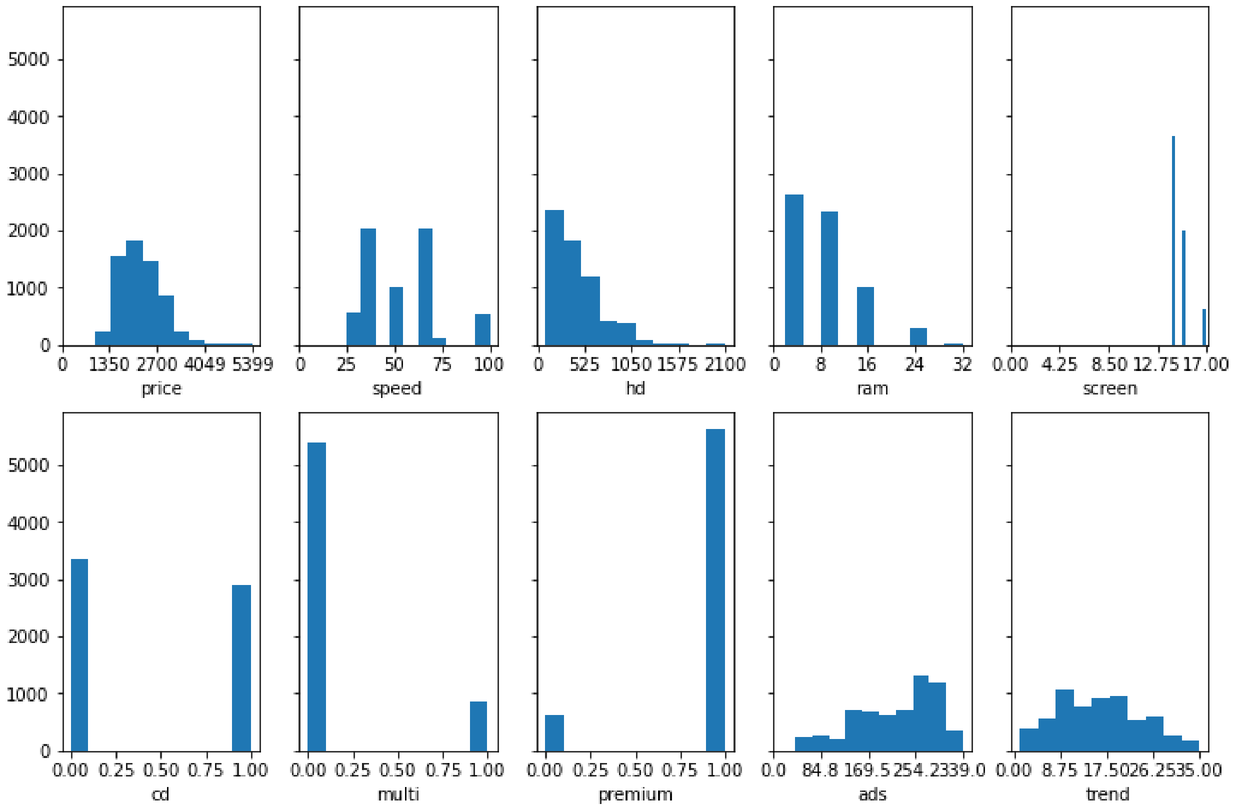
In the project, we had a genuine interest in estimating the price of the computers (Y) based on some other factors. Given that the price could be considered to be a continuous variable, linear regression could be a choice.



	price	speed	hd	ram	screen	cd	multi	premium	ads	trend
price	1	0.3	0.43	0.62	0.3	0.2	-0.017	-0.081	0.055	-0.2
speed	0.3	1	0.37	0.23	0.19	0.26	0.084	0.11	-0.22	0.41
hd	0.43	0.37	1	0.78	0.23	0.5	0.093	0.2	-0.32	0.58
ram	0.62	0.23	0.78	1	0.21	0.44	0.045	0.2	-0.18	0.28
screen	0.3	0.19	0.23	0.21	1	0.13	-0.0017	0.019	-0.094	0.19
cd	0.2	0.26	0.5	0.44	0.13	1	0.43	0.22	-0.061	0.45
multi	-0.017	0.084	0.093	0.045	-0.0017	0.43	1	0.12	-0.03	0.21
premium	-0.081	0.11	0.2	0.2	0.019	0.22	0.12	1	-0.15	0.042
ads	0.055	-0.22	-0.32	-0.18	-0.094	-0.061	-0.03	-0.15	1	-0.32
trend	-0.2	0.41	0.58	0.28	0.19	0.45	0.21	0.042	-0.32	1

<u>Correlation matrix of all data</u>

The correlation matrix displayed that there exist some linear relationships between the price and covariates. It may do a justice to conducting multiple linear regression.



Histograms of data

Running a scout on the histograms, we could tell that: Roughly, Y has a bell shape. It is safe to say that Y could come from the normal distribution.

So, the **likelihood model** would have this form:

$$\boldsymbol{Y \sim normal(X\beta, \sigma)}$$

Two models that we constructed to estimate the price are pooled model (non-hierarchical model) and the partially pooled model (hierarchical model). Here is the justification for why it was what it was:

It is shown in the previous study that the effect of the components of the computers on its price is not fixed but changes over time. This has something to do with the characteristics of high-tech and durable goods. To be specific, for these goods, when they are just launched for the first time, manufacturers have the tendency to set the premium price so that only those who have so much interest in these items and really want to become the first-buyers, would be willing to pay. After this stage, really soon, the price would drop (i.e: the special features of the items become popular that all manufactures could imitate. Also, it is because of the never ending release of new version).

That was the justification for us to regard observations in the same month as belonging to one group, which means that we had 35 groups (partially pooled model).

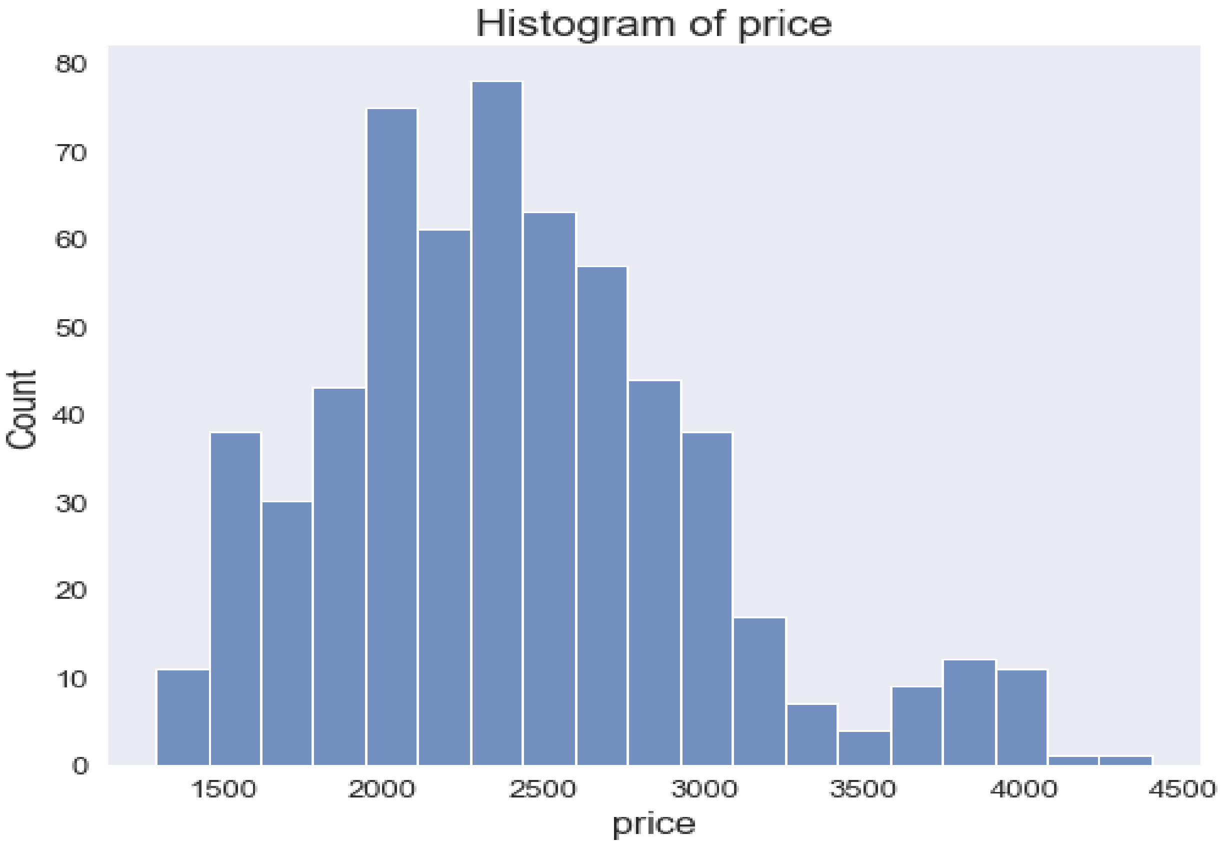
Then, we would like to do compare between the hierarchical model which takes into account this kind of inter-temporal pricing nature against the pooled model which ignored this idea (i.e: ignore any difference among observations coming from different groups; treat all data points as coming from one large group) and see which one is better.

Turning point

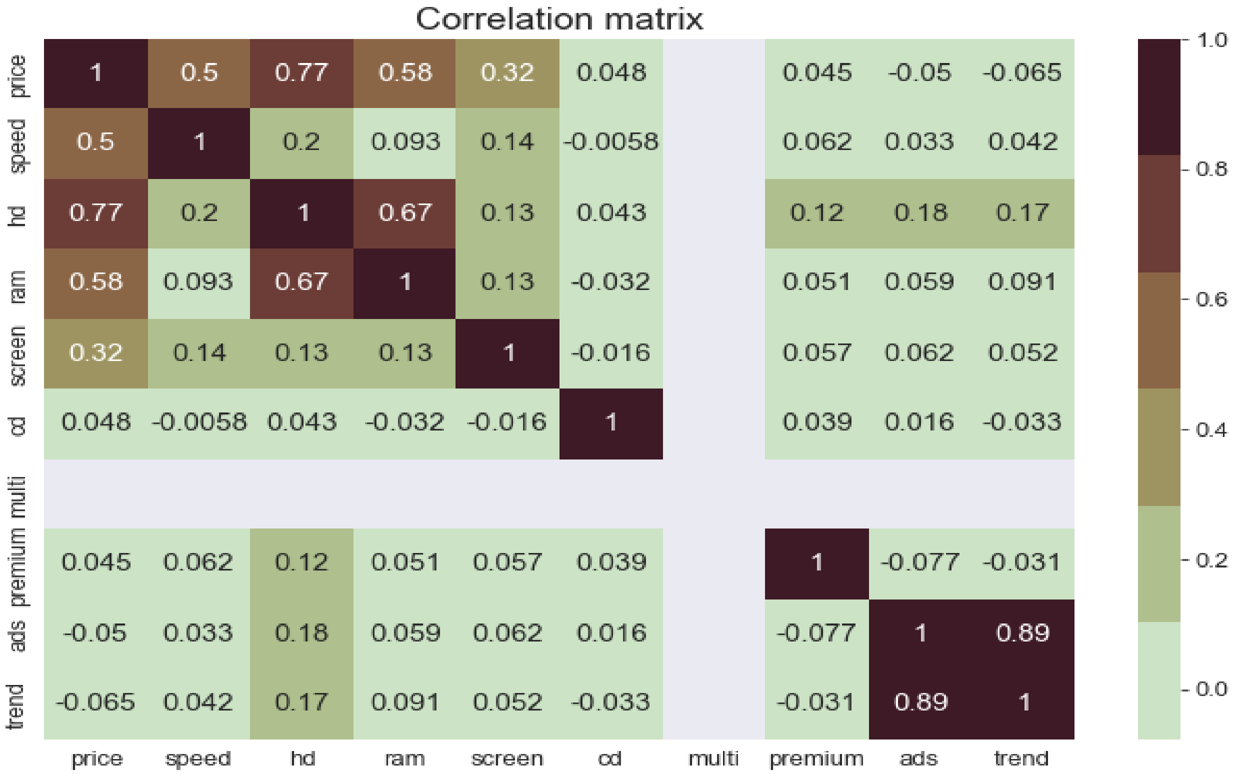
We later found out the problems regarding computational feasibility. To be specific, the models took really long time to proceed(15-30min) and even worse, R session in jupyter kept crashing, which made for us being so exhausted. This made totally sense since the data frame X has the shape of 6259x10 (including the intercept) which is too large for STAN to handle log pointwise predictive density efficiently. We held a belief that the objective of the project is to learn how to apply Bayesian work flow into analyzing data rather than to stress too much about the large number of data points and covariates. That brought us to making a pivotal decision which is to reduce the data.

Data reduction

Instead of calling on all the observations and all covariates as what we did above, firstly, we managed to only use a subset around 10% of the whole dataset which is 600 observations.



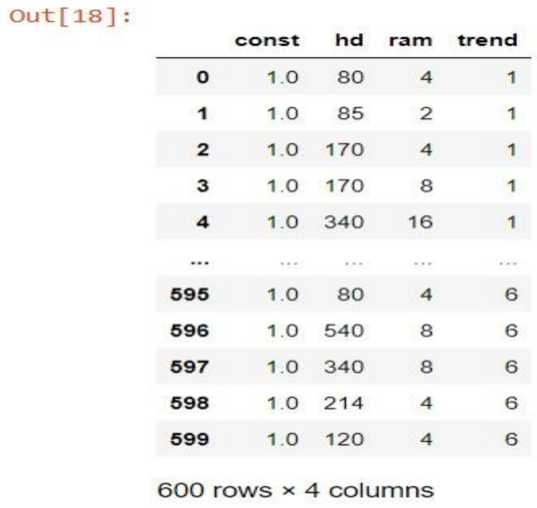
We used normal distribution for likelihood model which is not perfect as the histogram of price showed. We would discuss this in the discussion.



	price	speed	hd	ram	screen	cd	multi	premium	ads	trend
price	1	0.5	0.77	0.58	0.32	0.048		0.045	-0.05	-0.065
speed	0.5	1	0.2	0.093	0.14	-0.0058		0.062	0.033	0.042
hd	0.77	0.2	1	0.67	0.13	0.043		0.12	0.18	0.17
ram	0.58	0.093	0.67	1	0.13	-0.032		0.051	0.059	0.091
screen	0.32	0.14	0.13	0.13	1	-0.016		0.057	0.062	0.052
cd	0.048	-0.0058	0.043	-0.032	-0.016	1		0.039	0.016	-0.033
multi										
premium	0.045	0.062	0.12	0.051	0.057	0.039		1	-0.077	-0.031
ads	-0.05	0.033	0.18	0.059	0.062	0.016		-0.077	1	0.89
trend	-0.065	0.042	0.17	0.091	0.052	-0.033		-0.031	0.89	1

Correlation matrix of 600 datapoints

We had a look again at the correlation matrix and made up our mind to select only two covariates having the highest values of correlation which are hd (.77) and RAM (.58).



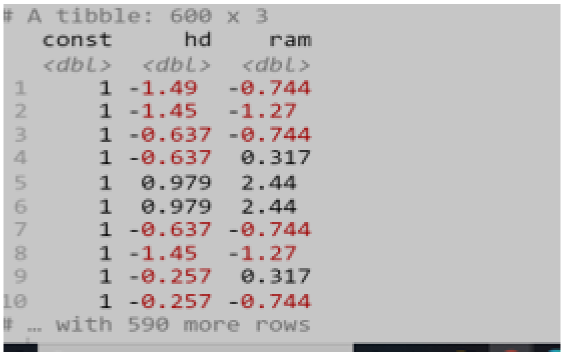
```
Out[18]:
      const   hd  ram  trend
  0     1.0   80    4      1
  1     1.0   85    2      1
  2     1.0  170    4      1
  3     1.0  170    8      1
  4     1.0  340   16      1
...     ...  ...  ...    ...
595     1.0   80    4      6
596     1.0  540    8      6
597     1.0  340    8      6
598     1.0  214    4      6
599     1.0  120    4      6

600 rows × 4 columns
```

These two covariates together with the intercept form the X matrix having the shape of 600x3(without taking into account the trend column). The vector of coefficients β has shape 3x1. Y is a column vector 600x1 displaying the price of 600 observations. And, also bear in mind the number of groups now is marked down to only 6(trend column).

SEPARATE ANALYSIS

Hierarchical model analysis

Dataframe used in the hierarchical model set-up



```
# A tibble: 600 x 3
   const     hd    ram
   <dbl>  <dbl>  <dbl>
 1     1 -1.49  -0.744
 2     1 -1.45  -1.27 
 3     1 -0.637 -0.744
 4     1 -0.637  0.317
 5     1  0.979  2.44 
 6     1  0.979  2.44 
 7     1 -0.637 -0.744
 8     1 -1.45  -1.27 
 9     1 -0.257  0.317
10     1 -0.257 -0.744
# … with 590 more rows
```

Since (hd, ram, price) have different scales, we find it useful to standardize the dataset so that setting prior as well as interpretation any changes in X and changes in Y could be done with ease.(i.e: the trend column is not standardized. Its values are still from 1 to 6)

Jargons and notations (IMPORTANT)

It is worth noting that in the hierarchical model, hyper parameters are also called group-level parameters and parameters are also called individual-level parameters.

When we say “prior” of parameters, it could be understood to be the prior distribution of the individual-level parameters (i.e: priors) or the prior distribution of the group-level parameters (i.e: hyper priors)

For scale parameters such as σ or τ, bear in mind that these are non-negative. Even though we say in general: e.g: draw some parameters from normal (0,10), it means normal (0,10) for unconstrained parameters and halfnormal(0,1) for constrained parameters.

Mathematical notation of the hierarchical model

Likelihood:

$$\boldsymbol{Y \sim normal(X\beta, \sigma)}$$

Prior:

$$\boldsymbol{\beta \sim normal(\mu, \Sigma)}$$

$\boldsymbol{\sigma \sim halfnormal(.,.)}$

Hyperprior:

$\boldsymbol{\mu \sim normal(.,.)}$

Σ = diag_matrix$(\boldsymbol{\tau}) * \boldsymbol{\Omega} *$ ***diag_matrix***$(\boldsymbol{\tau})$ ***where Ω is correlation matrix***

Ω ~ lkj_corr(ν)

$\boldsymbol{\tau \sim halfnormal(.,.)}$

Notations to interpret the results returned by Rstan

Beta[j,1] represents the intercept of group jth

Beta[j,2] represents coefficient estimate of covariate hd of group jth

Beta[j,3] represents coefficient estimate of covariate RAM of group jth

Priors choices

We conducted the prior predictive check as a way to define and justify the prior choices for parameters. Specifically, we tried some different values for the prior distributions, and see if the replicated Y makes sense compared against the true Y we had.

Vague and wide priors

At first, we set the distribution to disperse really widely and to be vague. Since our data has been standardized, we set the vague distribution which is normal(0,10) and $\nu = 1$

Observation distribution:

$\boldsymbol{Y \sim normal(X\beta, \sigma)}$

Prior:

$\boldsymbol{\beta \sim normal(\mu, \Sigma)}$

$\boldsymbol{\sigma \sim halfnormal(0, 10)}$

Hyperprior:

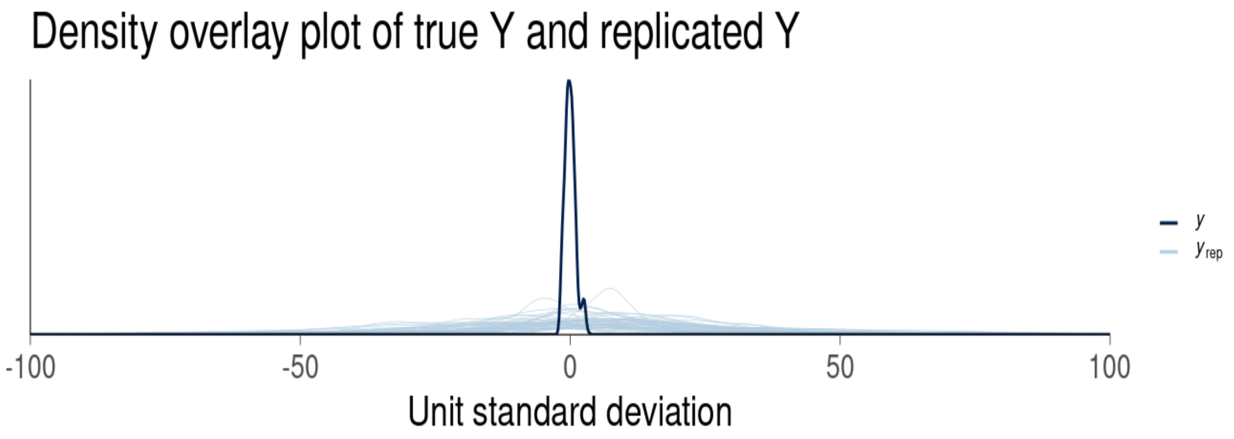
$\boldsymbol{\mu \sim normal(0, 10)}$

Σ = diag_matrix(τ) * Ω * diag_matrix(τ) where Ω is correlation matrix

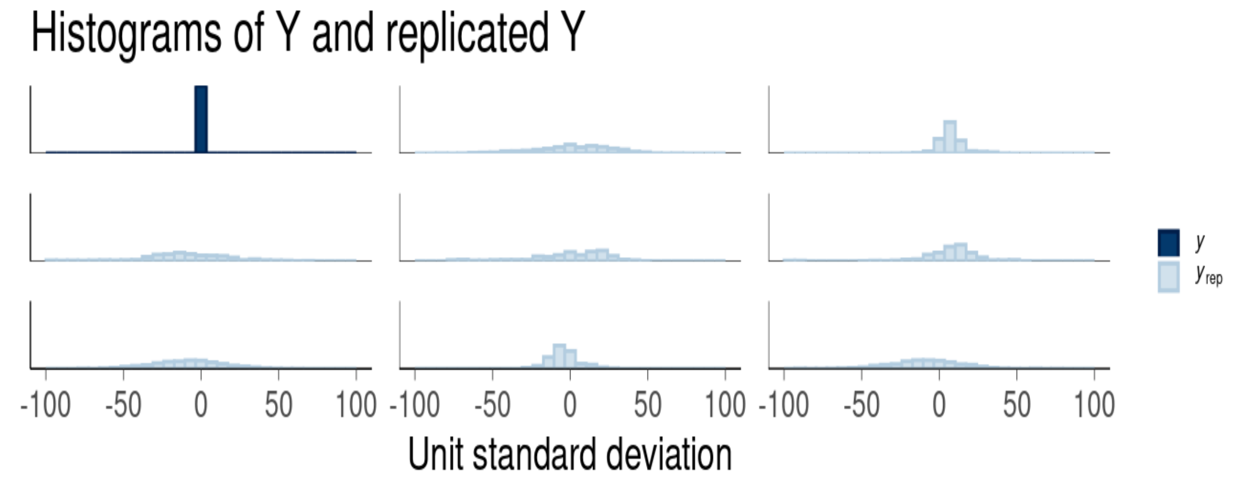
Ω ~ lkj_corr(1)

$\boldsymbol{\tau \sim halfnormal(0, 10)}$

(full STAN code is in the appendix)



Density overlay plot of true Y and replicated Y



Histograms of Y and replicated Y

As you may tell from two plots, the replicated Y is way far off from the range of true Y. To be specific, our replicated data even loomed over the area of around +-100 standard deviation which is, of course, too much in the case of standard normal distribution.

Narrower priors

In an effort to go about the matter we had above, we managed to set a smaller value of the scale. Specifically, instead of drawing values from normal(0,10). We reduce the scale so that the samples would be drawn from normal(0,1) and $nu = 1$

Observation distribution:

$$\boldsymbol{Y \sim normal(X * \beta, \sigma)}$$

Prior:

$$\boldsymbol{\beta \sim normal(\mu, \Sigma)}$$

$$\boldsymbol{\sigma \sim halfnormal(0, 1)}$$

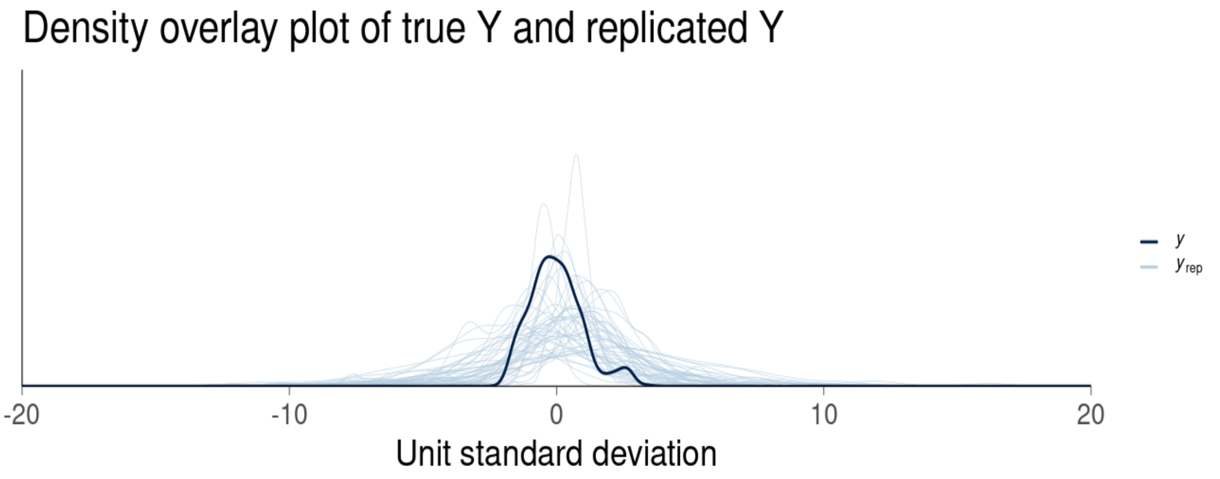
Hyperprior:

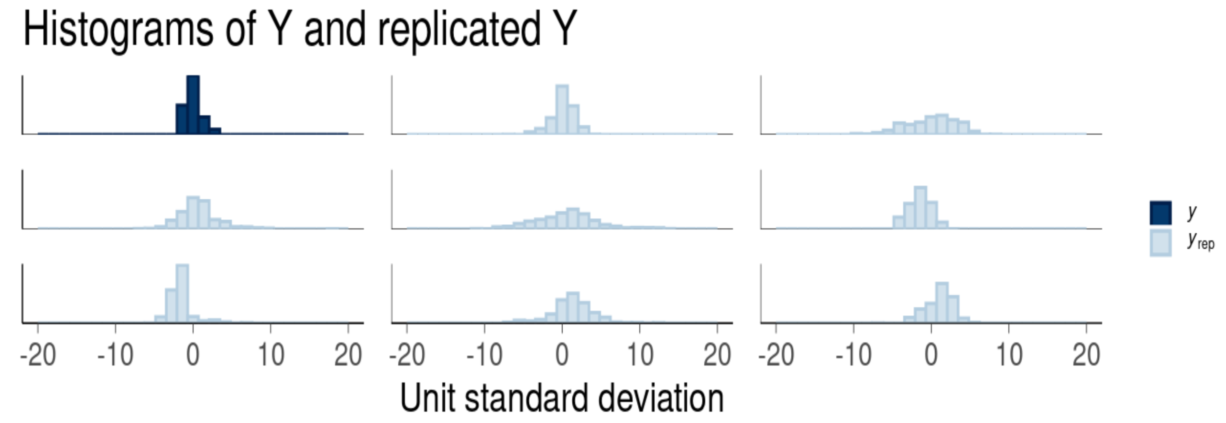
$\boldsymbol{\mu \sim normal(0, 1)}$

$\Sigma = diag\_matrix(\tau) * \Omega * diag\_matrix(\tau)$ ***where Ω is correlation matrix***

$\Omega \sim lkj\_corr(1)$

$\boldsymbol{\tau \sim halfnormal(0, 1)}$





On the basis of the plots, we could tell that replicated Y are much closer to the true Y we observed. So, we are content to what we have and proceed further using normal(0,1) and $\nu = 1$ as the prior distributions for the parameters.

Stan code

(to avoid too complicated and protracted STAN code, we separated the prior predictive check and the other part into two separate files. The STAN code written for prior predictive check is shown in the appendix)

```
#Hierarchical model stored in file= ‘stan code hier.stan’
data {
  #training data
  int<lower=0> N; // number of observations
  int<lower=0> J; // number of groups
  int<lower=0> P; // number of covariates
  int<lower=1,upper=J>group_idx[N]; // group indicator
  matrix[N,P] X;
  vector[N] Y;

  #testing data
  int<lower=0> Z; // number of observationsTest
  matrix[Z,P] XTest;


  #hyper priors
      # for group-level parameter mu
  real mum;
  real mus;

      # for group-level parameter covariance matrix
  real taum;
  real taus;
  real nu;

  #prior of sigma
  real sigmam;
  real sigmas;
}

parameters {

  # group-level parameter
  corr_matrix[P] Omega; //correlation matrix
  vector<lower=0>[P] tau;
  vector[P] mu;


  # individual-level parameter;
  vector[P] Beta[J];  // array size J
  real<lower=0> sigma;
}

model {
  #log_hyperpriors
  tau~ normal(taum,taus);
  Omega~lkj_corr(nu);
  mu~normal(mum,mus);
  #log-prior
  Beta~multi_normal(mu,quad_form_diag(Omega,tau));
  sigma~ normal(sigmam,sigmas);

  #log-likelihood
  {
    vector[N]X_Beta_group_idx;
    for (n in 1:N){
      X_Beta_group_idx[n]=X[n,]*Beta[group_idx[n]];
    }

    Y~normal(X_Beta_group_idx, sigma);
  }
}

generated quantities{
  vector[N] Y_rep;
  vector[Z] Ypred;
  vector[N] log_lik;

  for (n in 1:N){
    Y_rep[n]=normal_rng(X[n,]*Beta[group_idx[n]],sigma);
    log_lik[n]=normal_lpdf(Y[n]| X[n,]*Beta[group_idx[n]],sigma);
  }
  for (z in 1:Z){
    Ypred[z]=normal_rng(XTest[z,]*Beta[6],sigma);
  }
}
```

A bit of clarification: In order to optimize the computation, quad_form_diag() method is used. (i.e: quad_form_diag(Ω, τ)=diag_matrix(τ) $* \; Omega \; *$ diag_matrix(tau))

Stan model run

For the first try, we set: iterations= 1000, chains=4, adapt_delta= 0.8, max_treedepth=10 as default values

```
options(mc.cores=4)
model_stanhier<- rstan::stan_model('stan code hier.stan')

set.seed(1)
fit_stanhier<- sampling(

model_stanhier,

list(N=Nstanhier,P=Pstanhier,J=Jstanhier,X=Xstanhierused,Y=Ystanhier,group_id
x=group_idx, Z=
ZstanhierTest,XTest=XstanhierusedTest,mum=0,mus=1,taum=0,taus=1,nu=1,sigmam=0
,sigmas=1),

chains=4,iter=1000)
```

Convergence analysis

We got the warnings from STAN: There were 137 divergent transitions after warmup. The largest R-hat is NA, indicating chains have not mixed. Bulk Effective Samples Size (ESS) is too low, indicating posterior means and medians may be unreliable.

As you may tell from the warnings:

For divergences:

there were a large number of divergent transitions (137) after warmup

For Effective sample size:

it is too low

For Rhat:

NA values indicate that the chains have not mixed

In an effort to see about the situations, we raised the number of iterations to 3000 and set adapt_deta to be .9999(i.e: taking smaller steps. Though it would take more time to complete the posterior distribution, it could help drop the number of divergent transitions)

```
set.seed(1)
fit_stanhier<- sampling(

model_stanhier,

list(N=Nstanhier,P=Pstanhier,J=Jstanhier,X=Xstanhierused,Y=Ystanhier,group_id
x=group_idx, Z=
ZstanhierTest,XTest=XstanhierusedTest,mum=0,mus=1,taum=0,taus=1,nu=1,sigmam=0
,sigmas=1),

chains=4,iter=3000, control=list(adapt_delta=.9999))
```

This time, we got warnings: There were 11 divergent transitions after warmup

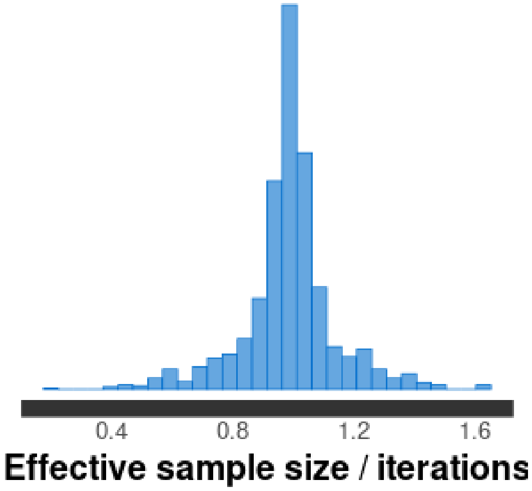
For divergence:

We were able to reduce the total number of divergent transitions from 137 to 11, which was a remarkable improvement. Though this is not completely eliminated, it was the best in our case. We would discuss more about this issue in the conclusion part. But here we could conclude that: the validity of the estimates is not totally guaranteed due to the divergence after warmup

For treedepth issue which relates to the problematic efficiency of the sampling:

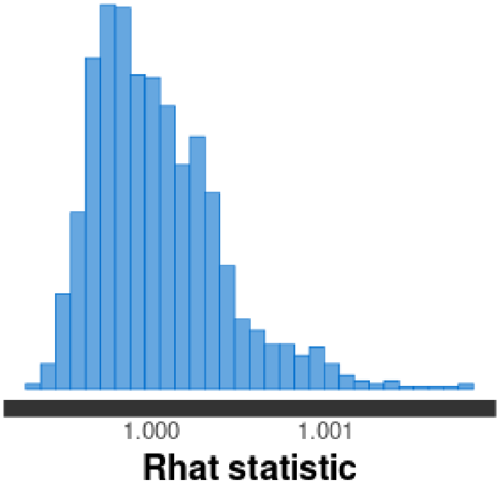
Our sampling did not run into any problem when we set max_treedepth as the default values which is max_treedepth=10. We could conclude that the sampling process does not have any problems relating to the efficiency

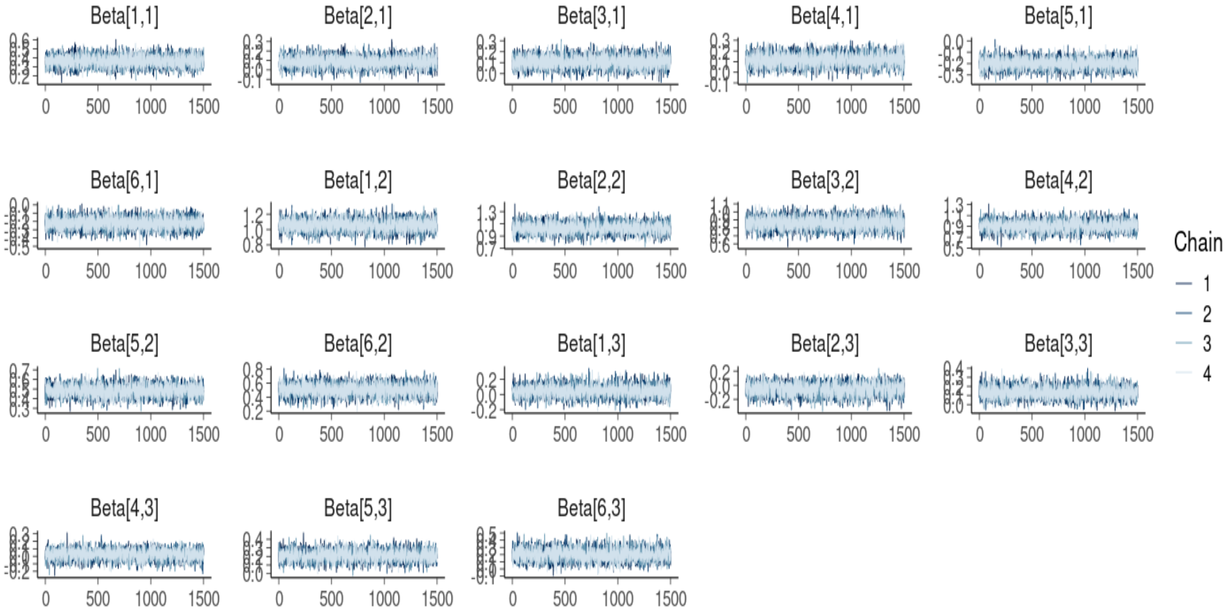
For Effective sample size:



The warnings did not raise any problem regarding the low number of ESS anymore and all N_eff/N are all above .1 which is a good sign. We could conclude that the number of independent draws from our posterior distributions are not too low and are acceptable.

For Rhat:



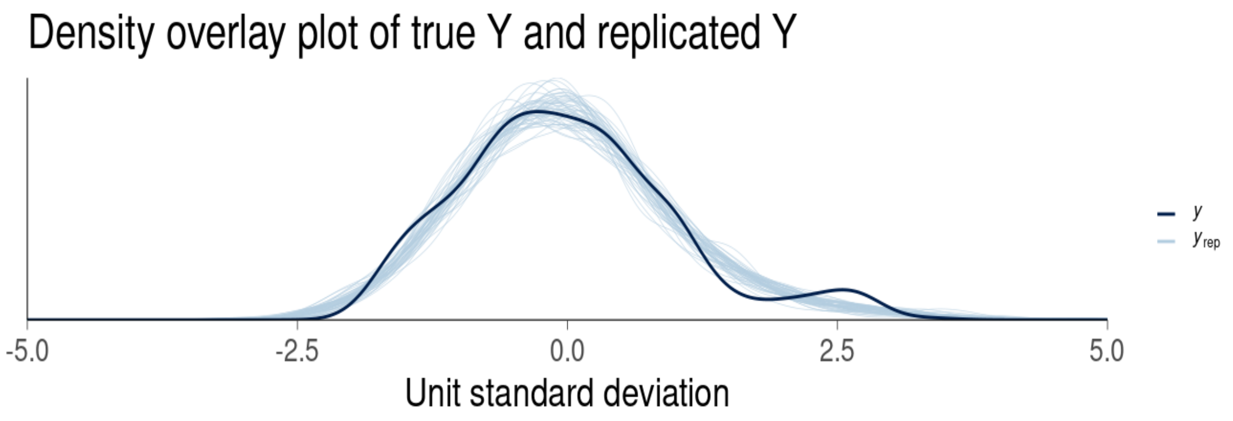


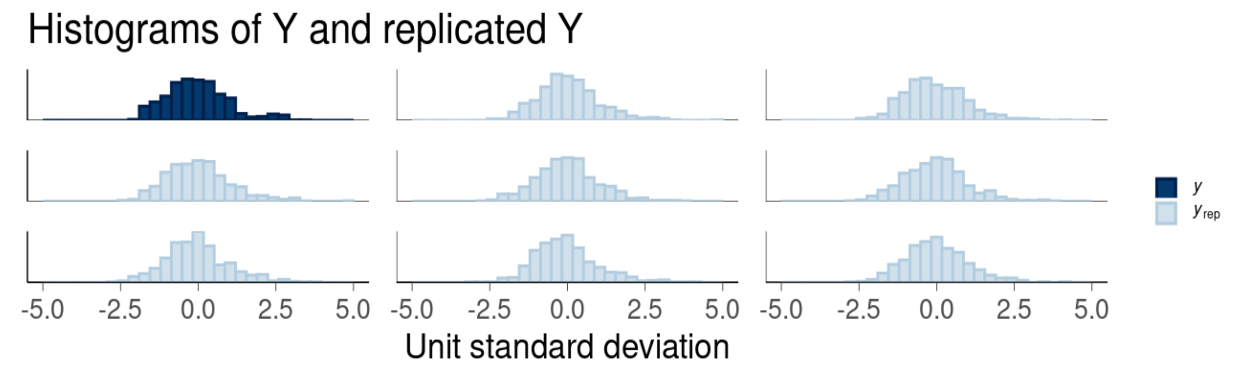
Trace plot of 4 chains of MCMC s draws

All values now were below 1.01. We could conclude that all chains have been mixed

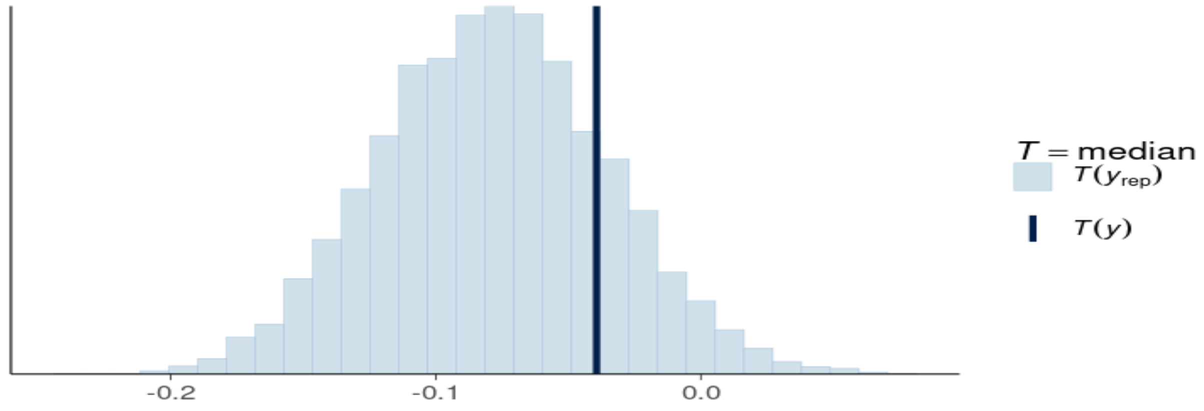
Posterior predictive check

```
Ystan_rephier<-as.matrix(fit_stanhier,pars='Y_rep')

ppc_hist(Ystanhier,Ystan_rephier[1:8,],binwidth=1) +xlim(-5,5)+labs(x='Unit
standard deviation', cex.axis=200, title='Histograms of Y and replicated Y')

ppc_dens_overlay(Ystanhier,Ystan_rephier[1:50,]) +xlim(-5,5)+labs(x='Unit
standard deviation',cex.axis=200, title='Density overlay plot of true Y and
replicated Y')

ppc_stat(Ystanhier,Ystan_rephier,stat='median')
```





As you may tell from the first two plots: the true data distribution is not perfectly captured. There are still some unexpected results (slump and peak suddenly) at the area around +2std to the right=> max values are not captured very well (i.e: our model expected more observations but the number of observations for one sub area is less; and vice versa, the other sub area has higher number of observations than expected). The improvements for this would be discussed in more details in the discussion part.



Histogram of medians

As for the median test statistic histogram, we see that the distribution of median of our replicated Y was able to capture the 50th quantile of true Y.

Overall speaking, the replicated Y are quite in line with Y. The hierarchical model seems to work

Pre model comparison

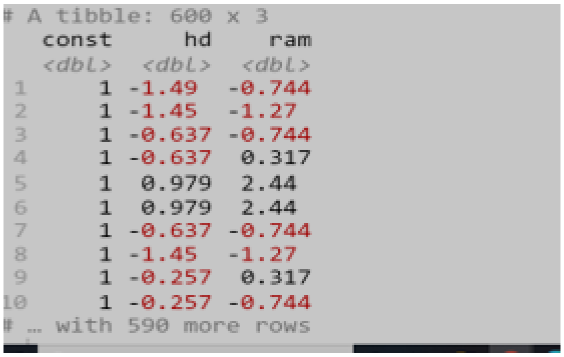
In order to be capable of comparing models, we should first estimate the elpd for each model. We used PSIS-LOO for the hierarchical model

```
log_lik_stanhier <- extract_log_lik(fit_stanhier,merge_chains=FALSE)

r_eff <- relative_eff(exp(log_lik_stanhier))

loo_stanhier <- loo(log_lik_stanhier,r_eff=r_eff)

print(loo_stanhier)

Computed from 6000 by 600 log-likelihood matrix

         Estimate   SE
elpd_loo   -523.0 19.0
p_loo        19.2  2.3
looic      1045.9 37.9
------
Monte Carlo SE of elpd_loo is 0.1.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     599   99.8%   1086
 (0.5, 0.7]   (ok)         1    0.2%   245
   (0.7, 1]   (bad)        0    0.0%   <NA>
   (1, Inf)   (very bad)   0    0.0%   <NA>

plot(loo_stanhier)
```

Bear in mind this table, it would be needed when doing model comparison later

Pooled model analysis

Dataframe used in the pooled model set-up



```
# A tibble: 600 x 3
   const     hd    ram
   <dbl>  <dbl>  <dbl>
 1     1 -1.49  -0.744
 2     1 -1.45  -1.27
 3     1 -0.637 -0.744
 4     1 -0.637  0.317
 5     1  0.979  2.44
 6     1  0.979  2.44
 7     1 -0.637 -0.744
 8     1 -1.45  -1.27
 9     1 -0.257  0.317
10     1 -0.257 -0.744
# … with 590 more rows
```

Jargons and notations (IMPORTANT)

For scale parameters such as σ, bear in mind that it is non-negative. Even though we say in general: e.g: draw some parameters from normal (0,10), it means normal(0,10) for unconstrained parameters and halfnormal(0,1) for constrained parameters.

Mathematical notation of the pooled model

Likelihood:

$$\boldsymbol{Y \sim normal(X\beta, \sigma)}$$

Prior:

$$\boldsymbol{\beta \sim normal(.,.)}$$

$$\boldsymbol{\sigma \sim halfnormal(.,.)}$$

Notations to interpret the results returned by STAN

Beta[1] represents the intercept

Beta[2] represents coefficient estimate of covariate hd

Beta[3] represents coefficient estimate of covariate RAM

Priors choices

The procedure we took to justify our prior choices is similar to what we have done in the hierarchical model. We did prior predictive check: firstly, starting with some really vague and wide distributions of the priors and then scale them down.

Observation distribution:

$$\boldsymbol{Y \sim normal(X\beta, \sigma)}$$

Prior:

$\beta \sim \mathrm{normal}(\mu, \Sigma)$

where μ <- c(0,0,0)

Σ = diag_matrix(τ) * Ω * diag_matrix(τ) where Ω is correlation matrix

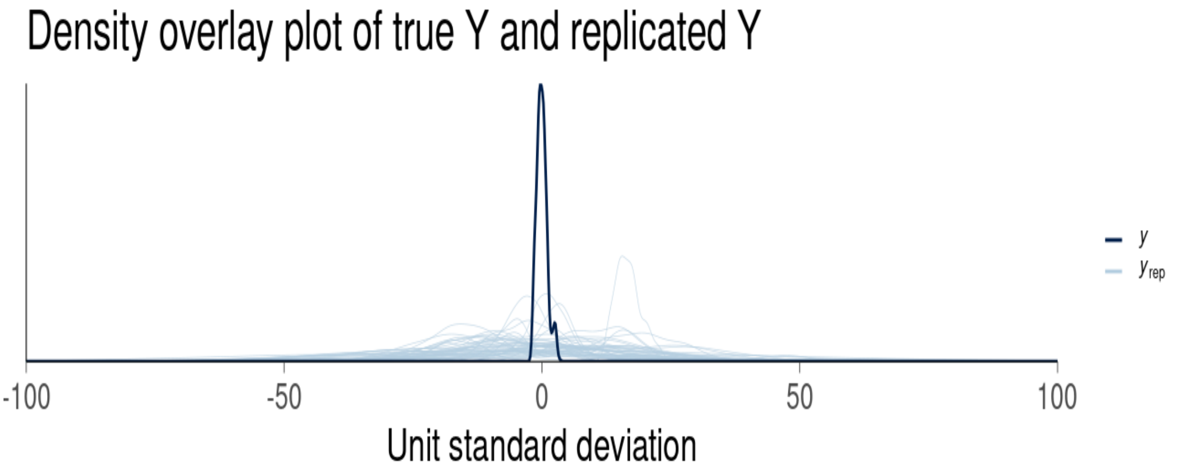
set.seed(1) Ω <- rlkj_corr(1)

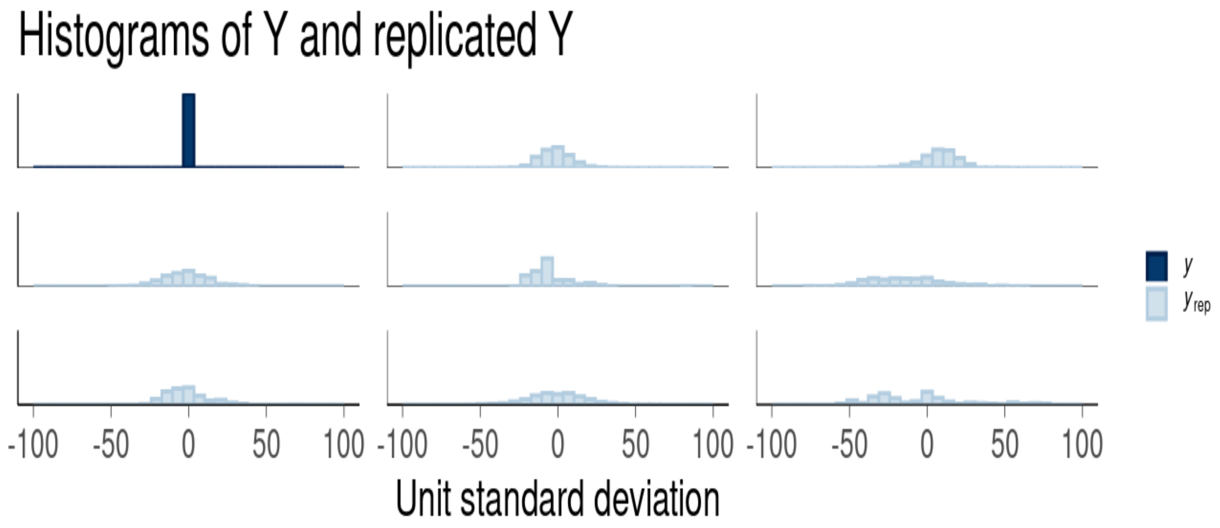
τ <- c(10,10,10)

$\sigma \sim halfnormal(0, 10)$

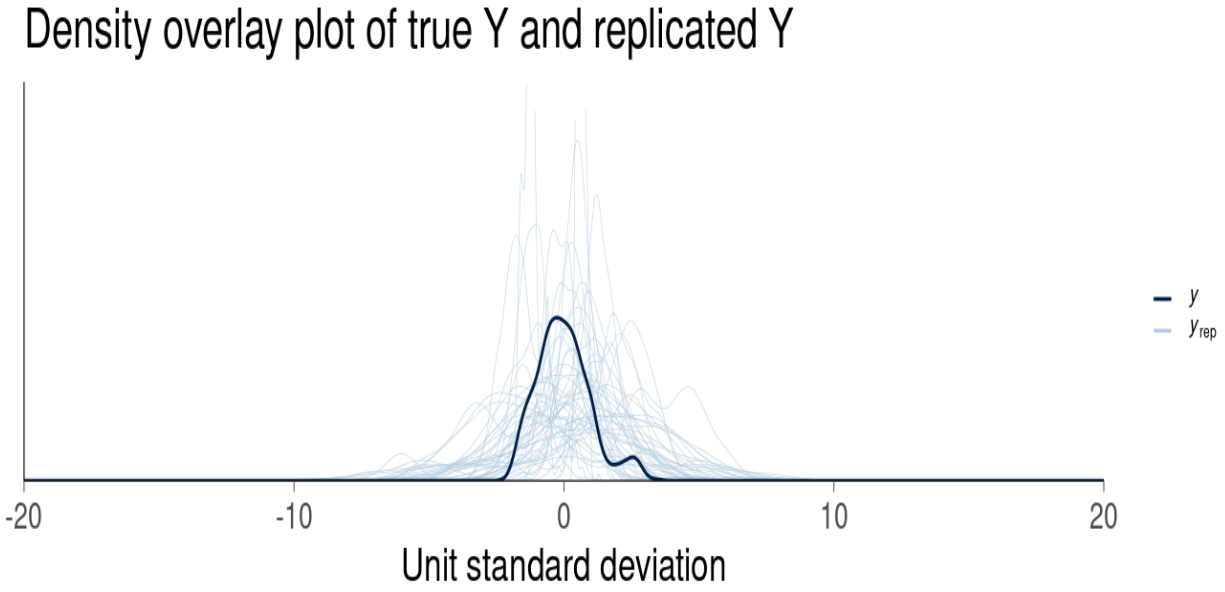
(full STAN code is in the appendix)

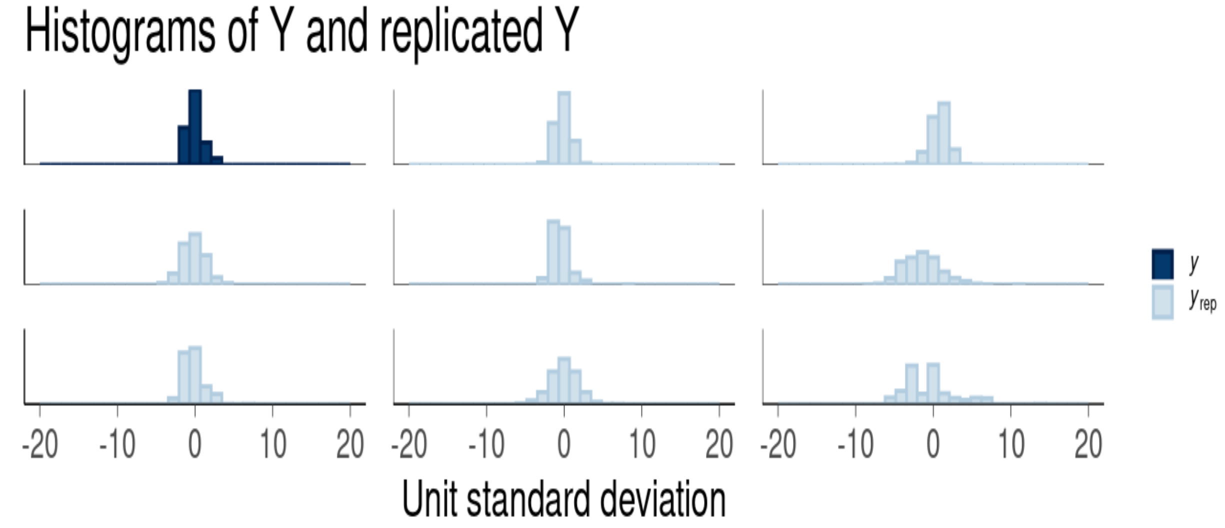
Here are two plots we got when opting for vague prior distribution normal(0,10) and $\nu = 1$.





And below are the plots when setting the prior distribution as normal(0,1), $\nu = 1$





The latter created a much more sensible range of replicated Y compared against the true Y. Therefore, we made use of normal(0,1), $\nu = 1$ as the prior distribution that parameters are drawn from

STAN code

(to avoid too complicated and protracted STAN code, we separated the prior predictive check and the other part into two separate files. The STAN code written for prior predictive check is shown in the appendix)

```
#stored in file named 'stan code OLS.stan'
data {
  # training data
  int<lower=0> N; // number of observations
  int<lower=0> P; // number of covariates
  matrix[N,P] X;
  vector[N] Y;

  #testing data
  int<lower=0> Z; // number of observationsTest
  matrix[Z,P] XTest;

  #prior of Beta
      #center parameter
  vector[P]muBe;

      #scale parameter
  corr_matrix [P] OmegaBe;
  vector<lower=0>[P] tauBe;

  #prior of sigma
  real sigmau;
  real sigmas;
}

parameters {
  vector[P] Beta;
  real<lower=0> sigma;
}

transformed parameters{
  vector[N] Ymu;
  Ymu= X*Beta;
}

model{
# priors
  Beta ~ multi_normal(muBe,quad_form_diag(OmegaBe,tauBe));
  sigma ~ normal(sigmau,sigmas);

# likelihood
  Y~normal(Ymu,sigma);
}
```

```
generated quantities{
  vector[N] Y_rep;
  vector[Z] Ypred;

  vector[N] log_lik;

  for (n in 1:N){
    Y_rep[n]=normal_rng(Ymu[n],sigma);
    log_lik[n]= normal_lpdf(Y[n]|Ymu[n],sigma);
  }
  for (z in 1:Z){
    Ypred[z]= normal_rng(XTest[z,]*Beta,sigma);
  }
}
```

How STAN code is run

In order to ensure the consistency in sampling between partially pooled model and pooled model, we used the same configuration: iterations =3000, chains=4, adapt_delta=.9999, max_treedepth as default value which is 10

```
#Compile STAN code
model_stanOLS <-stan_model('stan code OLS.stan')
options(mc.cores=4)

---
# Setting up value for priors

muBestanOLSused<-c(0,0,0)

set.seed(1)
OmegaBestanOLS<-rethinking::rlkjcorr( 1 , PstanOLS , eta=1 )
tauBe<-c(1,1,1)



#Sampling from the model compiled
set.seed(1)
fit_stanOLS<-sampling(

model_stanOLS,

list(N=NstanOLS,P=PstanOLS,X=XstanOLSused[1:600,],
Y=YstanOLS[1:600],Z=ZstanOLSTest, XTest=XstanOLSusedTest,sigmau=0, sigmas=1,
muBe=muBestanOLSused, OmegaBe=OmegaBestanOLS, tauBe=tauBe),iter=3000,

chains=4, control=list(adapt_delta=.9999))
```

Convergence analysis

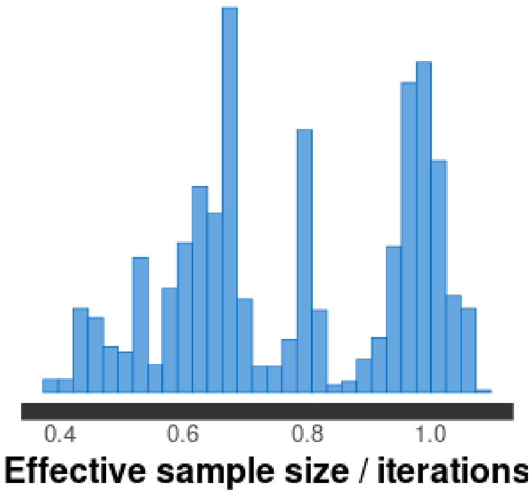
For divergence:

We did not face any problems related to the divergent transitions given the adapt_delta is .9999. We could conclude that the validity of estimates is guaranteed.

For tree depth issue which relates to the problematic efficiency of the sampling:

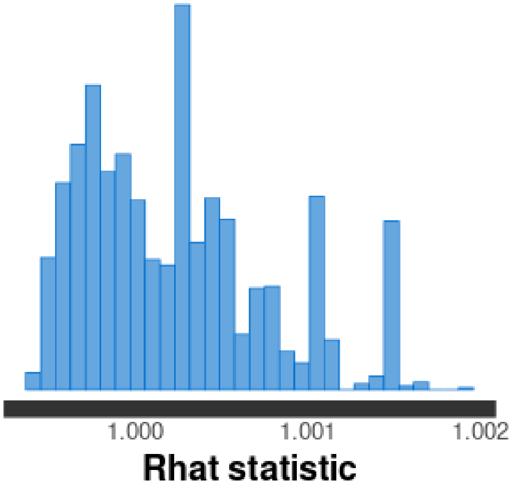
Our sampling did not run into any problem when we set max_treedepth as the default values which is max_treedepth=10. We could conclude that there is no efficiency matter our sampling has to go about.

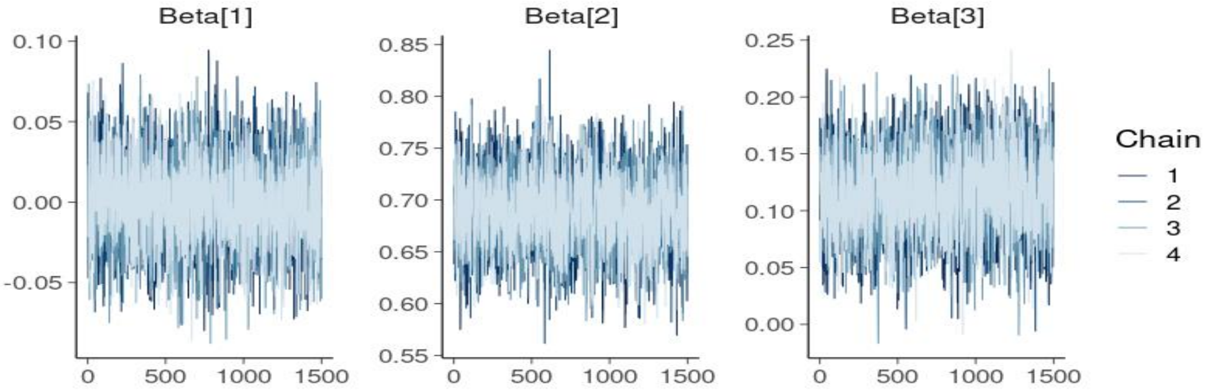
For Effective sample size:



All N_eff/N are above .1 which is a good sign. We could conclude that the number of independent draws from our posterior distributions are not too low and are acceptable.

For Rhat:



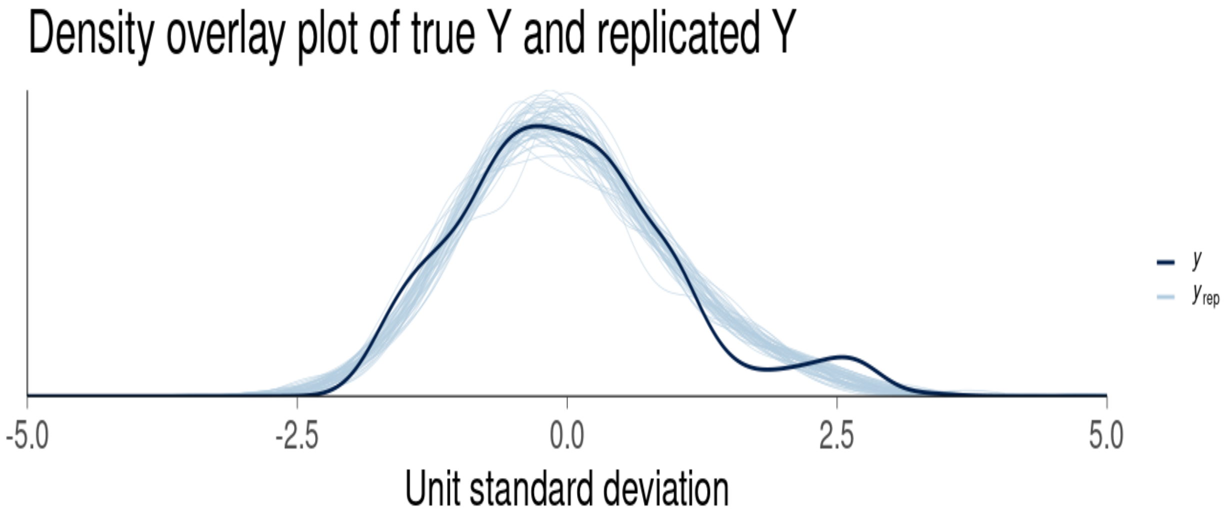


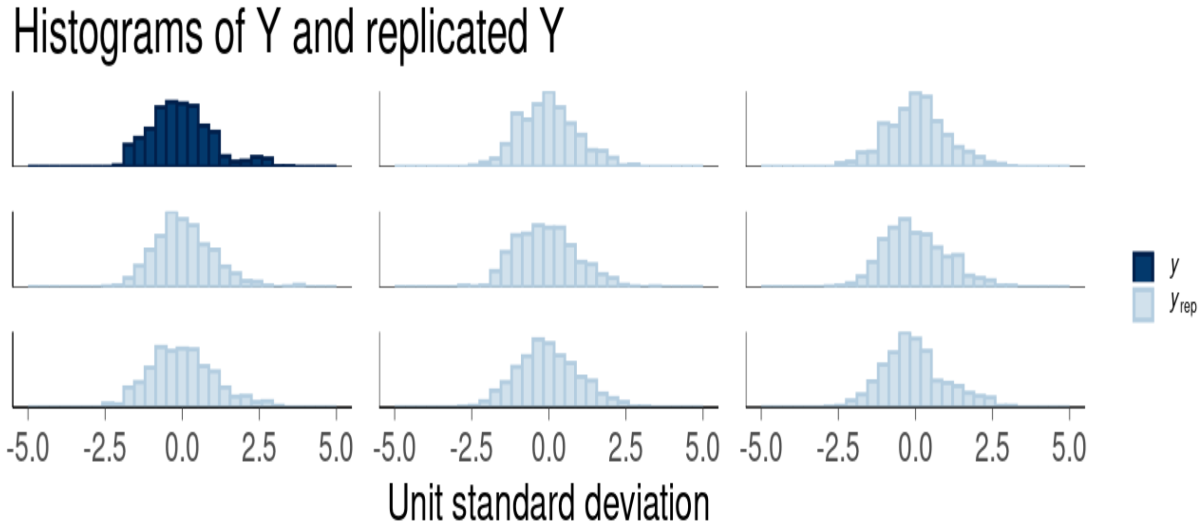
Trace plot of 4 chains of MCMC s draws

All values now were below 1.01. We could conclude that all chains have been mixed

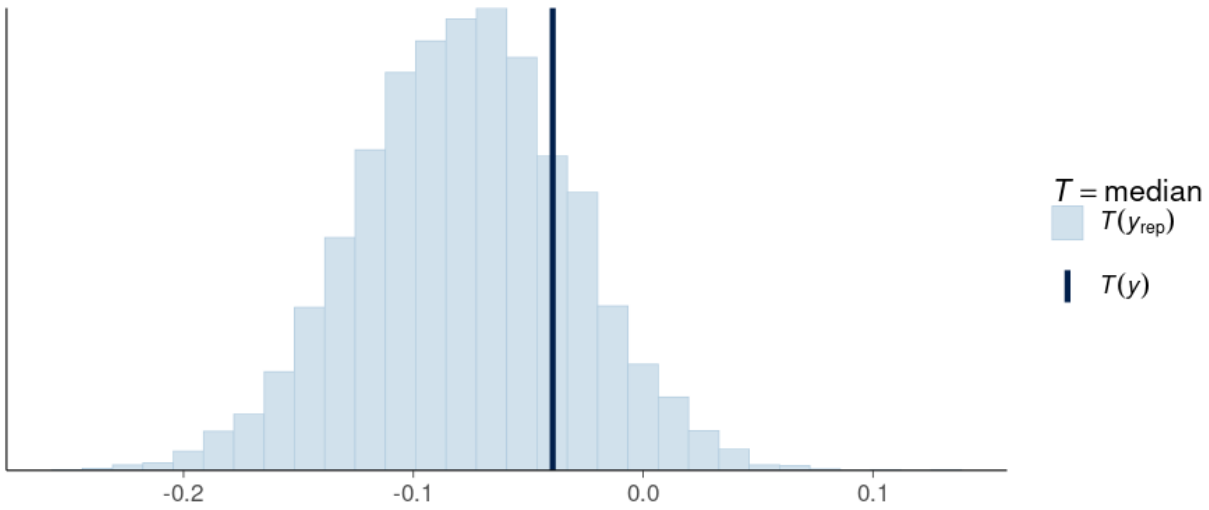
Posterior predictive check

```
Ystan_repOLS<-as.matrix(fit_stanOLS,pars='Y_rep')

ppc_hist(YstanOLS,Ystan_repOLS[1:8,],binwidth=1) +xlim(-5,5)+labs(x='Unit
standard deviation', cex.axis=200, title='Histograms of Y and replicated
Y')+geom_histogram(size=1)

ppc_dens_overlay(YstanOLS,Ystan_repOLS[1:50,]) +xlim(-5,5)+labs(x='Unit
standard deviation',cex.axis=200, title='Density overlay plot of true Y and
replicated Y')

ppc_stat(YstanOLS,Ystan_repOLS,stat='median')
```





As you may tell from the first two plots: the true data distribution is not perfectly captured. There are still some unexpected results (slump and peak suddenly) at the area from around 2std to the right=> max values are not captured very well (i.e: our model expected more observations but the number of observations for one sub area is less; and vice versa, the other sub area has higher number of observations than expected).



As for the median statistic test histogram, it looks like that our model was able to generate the replicated Y so that the distribution of median value of it could capture median of true Y which is great.

Overall speaking, visually, the replicated Y are quite in line with Y. The pooled model seems to work.

Pre model comparison

In order to be capable of comparing between models, we should estimate the elpd for each model. We used PSIS-LOO for the pooled model

```
log_lik_stanOLS <- extract_log_lik(fit_stanOLS,merge_chains=FALSE)

r_eff <- relative_eff(exp(log_lik_stanOLS))

loo_stanOLS <- loo(log_lik_stanOLS,r_eff=r_eff)

print(loo_stanOLS)

Computed from 6000 by 600 log-likelihood matrix

         Estimate   SE
elpd_loo   -585.4 19.6
p_loo         6.8  2.0
looic      1170.7 39.3
------
Monte Carlo SE of elpd_loo is 0.1.

Pareto k diagnostic values:
                         Count Pct.    Min. n_eff
(-Inf, 0.5]   (good)     599   99.8%   1973
 (0.5, 0.7]   (ok)         1    0.2%   176
   (0.7, 1]   (bad)        0     0.0%   <NA>
   (1, Inf)   (very bad)   0     0.0%   <NA>
```

Bear in mind this table, it would be needed when doing model comparison

COMPARISON ANALYSIS

Model comparison

```
print(loo_compare(loo_stanhier,loo_stanOLS))

       elpd_diff se_diff
model1   0.0       0.0  
model2 -62.4      13.7  
```

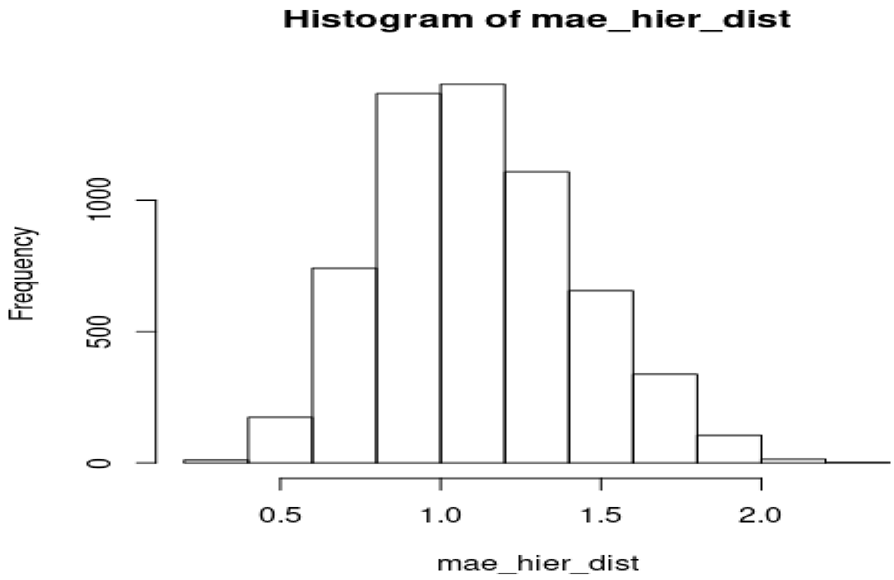
Having a look at the table, we could say that the elpd_diff for the pooled model (model2) is -62.4 which means that it is worse than hierarchical model (model1). Is the difference significant? It could be concluded based on the se_diff. If the |elpd_diff| is larger than 4* se_diff (62.4 > 4*13.7), the difference really matters. Therefore, in our case, the hierarchical model is better compared against the pooled model.

Predictive performance assessment

In this part, we called on the hierarchical model and pooled model to predict some unseen values. In our case, due to computational limitations, we could not have quite a large number of values in test set. Actually, since at first we just used the subset of the whole dataset, we still have some values untouched in group 6. We used these values (around 6 values) as the true and unseen values Y. Each model predicted some values Ypred. We then used the MAE as the metric of assessing predictive performance of two models. (Bear in mind that the data is standardized)

```
# Mean absolute error(MAE)

mae<-function(Y,Ypred){

abs_err<-abs(Y- Ypred)

mae<-(rowSums(abs_err)/6)

mae_dist<-as.matrix(mae)

return (mae_dist)
}
```

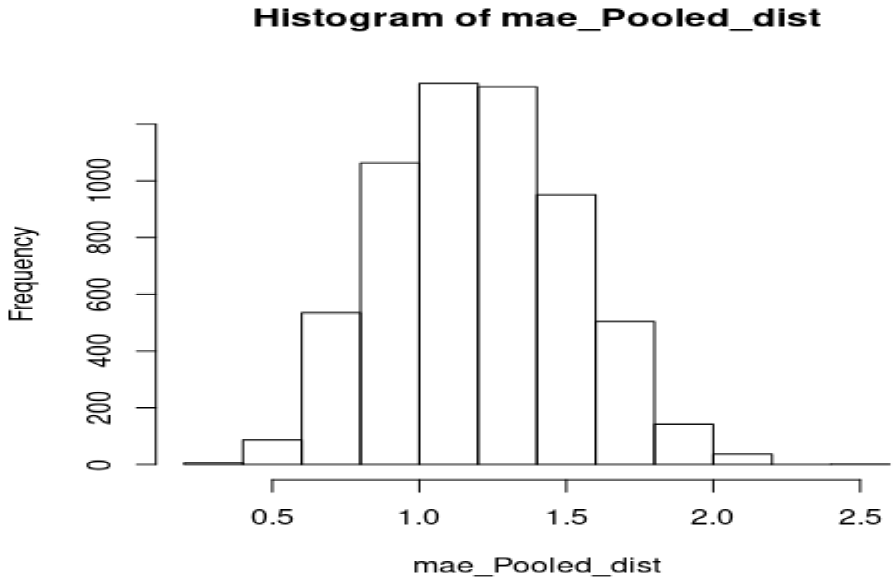
For hierarchical model



Histogram of mean absolute errors Hierarchical model

Median error value is 1.09

For pooled model



Histogram of mean absolute error Pooled model

Median of error value is 1.19

Having a look at both histograms as well as perceiving the median of error values (1.09 <1.19), we could spot straight away that the hierarchical model performs better against the pooled model. Having said that, the accuracy of the hierarchical model itself is not really impressive since the predicted Y of the model is not really close to the true Y (median error being equal to 1.09 means replicated Y around 1.09 standard deviation from the true Y)

EXCLUSIVE ANALYSIS

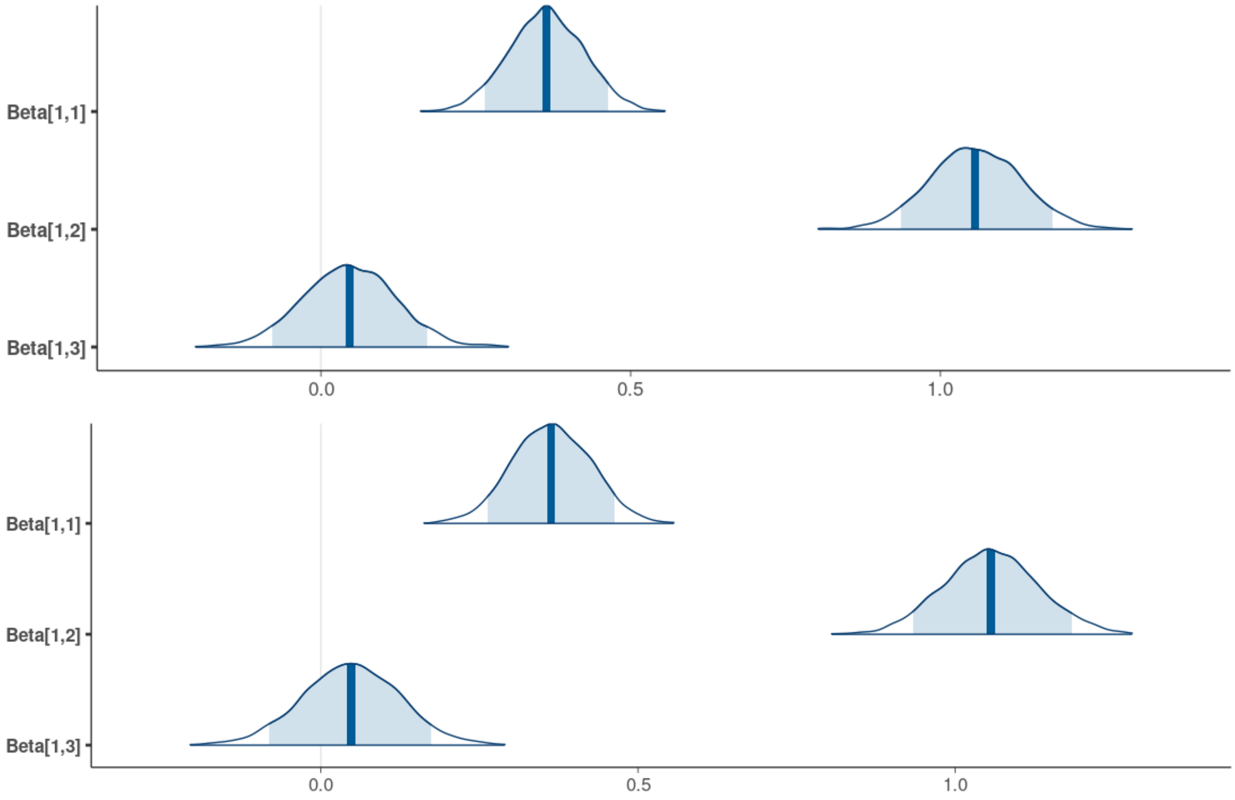
Sensitivity analysis

Having come up with the idea that the hierarchical model surpassed the pooled model, we ran some sensitivity analysis to see if the partially pooled model is sensitive to changes in prior

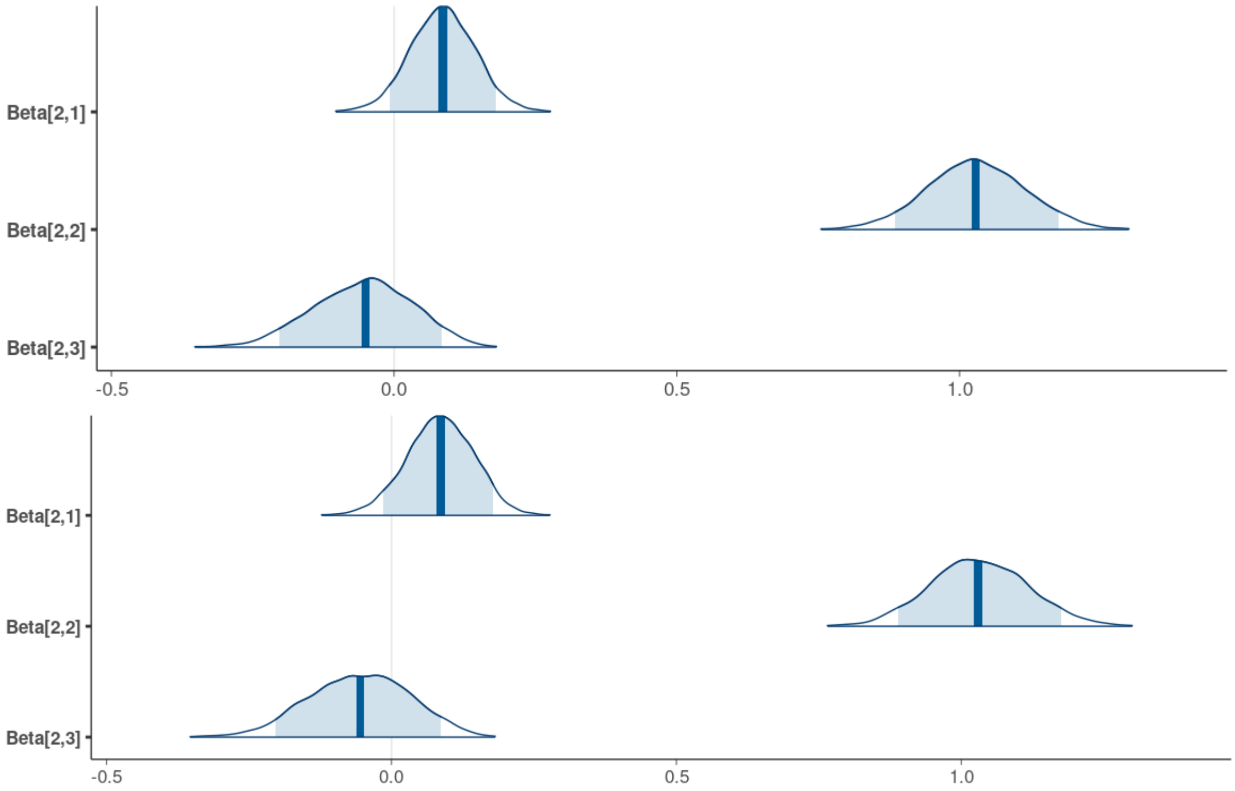
This time, we set really wide priors distribution from the normal(0,10), $\nu = 1$ and observe how things play out.

```
#Sampling
set.seed(1)
fit_stanhierSen<- sampling(

model_stanhier,

list(N=Nstanhier,P=Pstanhier,J=Jstanhier,X=Xstanhierused,Y=Ystanhier,group_id
x=group_idx, Z= ZstanhierTest,
XTest=XstanhierusedTest,mum=0,mus=10,taum=0,taus=10,nu=1,sigmam=0,sigmas=10)

chains=4,iter=3000,control=list('adapt_delta'=.9999))
```

Marginal distributions of each coefficient



Marginal posterior distribution of coef group 1 (weak prior and vague prior respectively)



Marginal posterior distribution of coef group 2 (weak prior and vague prior respectively)

Beta[j,1] represents the intercept of group jth

Beta[j,2] represents coefficient estimate of covariate hd of group jth

Beta[j,3] represents coefficient estimate of covariate RAM of group jth

The plots of the marginal posterior distribution estimates (i.e: the dark blue line represents the median and the shaded areas are .9 hpd interval) looked roughly the same and they covered the same intervals in both case (i.e normal(0,10), and normal(0,1)). We demonstrated our points by showing only the distributions of two groups only.

For other groups and the marginal posterior distribution estimate table, they are included in the appendix

So, it is rather safe to say that the hierarchical model is robust since the estimates are not sensitive to changes in priors.

DISCUSSION

Conclusion

From the analysis, we could conclude that:

+Based on the marginal posterior distribution of the coefficients (plots of groups 3, 4, 5, 6 are in the appendix): hard drive (hd) and RAM specs could provide us with a sense of the price of the computers (i.e: while hard drive invariably has positive effect on the price, the effect of RAM on price is not that straightforward. For the better part of the time, the probability that RAM has positive effect is larger than the probability that RAM has negative effect on price).

+Hard drive has stronger effect on price than RAM has.

+The effects on the price of these components are not fixed but rather change over time (i.e: different groups have different 95 HPDs of the coefficients)

+Be cautious: The estimates for the effect could be somewhat biased (i.e: having divergent transitions in the MCMC draws)

+The estimate of the price is not perfect (i.e: the differences distribution has median around 1.09)

Weaknesses and potential improvements

+As for the distribution of price:

What you may perceive in the graphical posterior predictive check part is that our model did not seem to work well for all areas of price. To be specific, there are more errors in replicating the values in the right tail of the standard normal distribution which is the part of premium price computers. This could boil down to the fact that the subset of data points is not representative of the distribution of price in the whole dataset (we would discuss why we encountered this issue in the reflection). Therefore, cleverer sampling method should be taken into account to improve the results.

In case we conceive of these 600 data points being the whole dataset, then, more complex models (i.e: either by incorporating more covariates or by working out a better distribution for price (such as bimodal distribution)) could be constructed and compared against the hierarchical model for further improvements.

+As for the STAN code:

It is not well optimized. At this point our STAN code still has 11 divergent transition though the adapt_delta has already been set to be .9999. The reason could lie at the fact that we used centered parameterisation. One suggestion for improvement is to call on non-centered reparameterisation instead. When we ran a check with brms code, there was not any divergence at all for the same model, which made sense since the brms code was rather optimal and it parameterized the parameters in a non-centered manner. Then, naturally, comes a question: why did not we use brms code but wrote STAN manually? This would be clarified in the reflection as well.

+As for the predictive performance assessment:

Currently, we just made use of unseen data points of price in group 6. Actually, there are still data points from some unseen groups (i.e: group 7, 8, 9, etc). Performing predictive performance assessment on these groups could also be conducted as well in the future as an improved version of the predictive model assessment.

Reflection

+For the choice of distribution of price: Here comes the true story from our side:

We at first made use of the whole data and it looked roughly normal. After spending a considerable amount of time trying to find ways to fit the whole dataset, turning out to be so exhausted with the computational time, and ultimately making up our mind to truncate the data, we carried on the work and still assumed the normal distribution. We just failed to call to mind to rerun the histogram again on the truncated data. Only at the nearly last minute, we found out that we should have run a check on the trimmed price distribution, it is just too late. We acknowledged this. This is one of our lesson learnt. In our case, after being so emotionally stressed, we somewhat lost our cool and carried on the work mostly by following the invisible voices in our head "hurry up, you are going to miss the deadlines". We lacked some important steps.

Solution: Writing explicitly a procedure guideline and place it at the prominent place so that we could reference it any time is what we should do in the future to avoid this.

+Why did not we use brms code?

Actually, we did make rather a large number of attempts to use brms from the outset. Having said that, for multi-level model, we had a real hard time to extract the coefficients as we want for plotting if using brms(i.e: brms returned the coefficients separately for population-level effect and the group-level effect while what we learnt so far is only about group-level parameters and individual-level parameters). In addition to that, the structure as well as the style of coding STAN model in brms is hard to interpret especially compared to what we have learnt from assignments. Since we are supposed to show how the STAN model is run, not being able to process how the STAN codes written in brms package means that we would have some issues. We reckoned that it is more important to actually have a solid grasp of the steps we were taking rather than just opting for the convenience but a black box to us. Therefore, we tried to construct the STAN code manually on the basis of the instructions from STAN user guide.

Solution: As we are more exposed to R and brms in the future, we reckon that we could learn how to handle the problems we faced.

+We don’t know what we don’t know. Being engaged in the project rather much, we were quite subjective and biased towards our work, not able to notice fatal errors in our reasoning, and in our codes. We thought all were right and only when attending in the t.a sessions, we found how fragile our thinking was. T.a(s), with their fresh perspectives, could help raise the issues and we have handled quite some. But of course, there is not always enough time and chances for us to go about all problems (e.g: using the first 600 datapoints in the dataset rather than clever sampling is one of issues that we failed to handle).

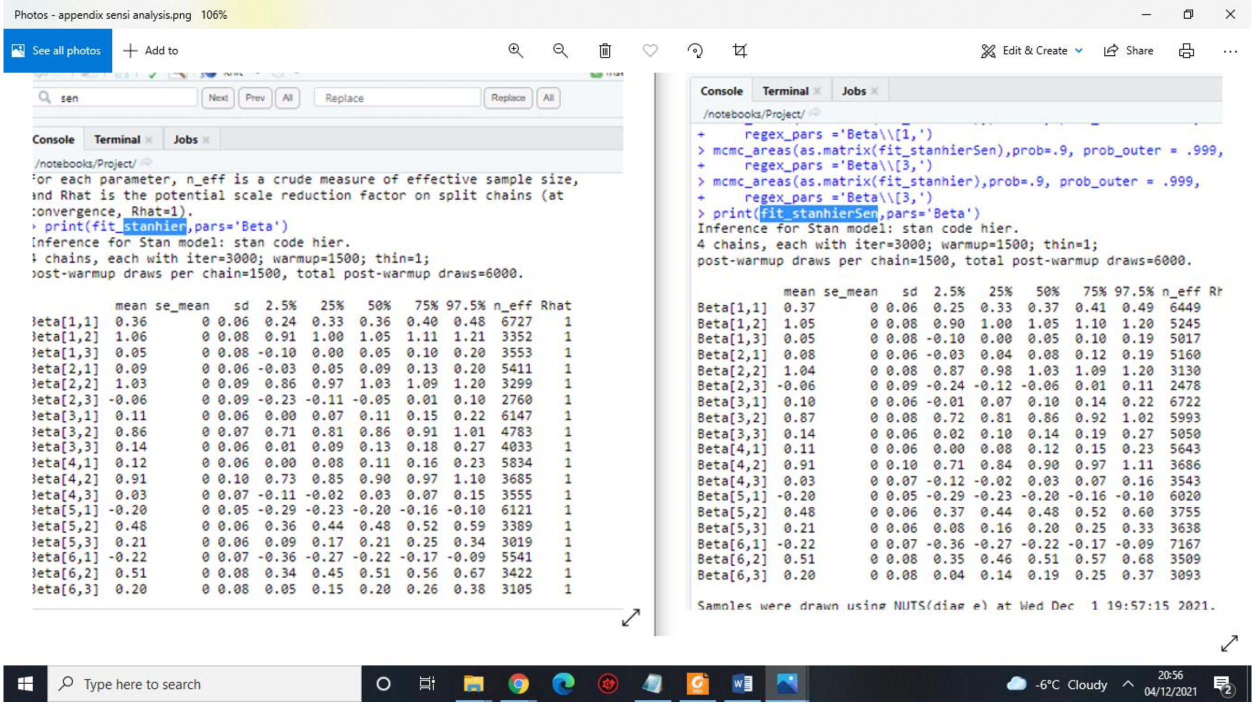
Solution: For future work, we should be more critical of what we did, consult t.a more often as well as allocate some amount of time to have discussions with other groups helping each other improving the work.

APPENDIX

1/ Prior predictive check code

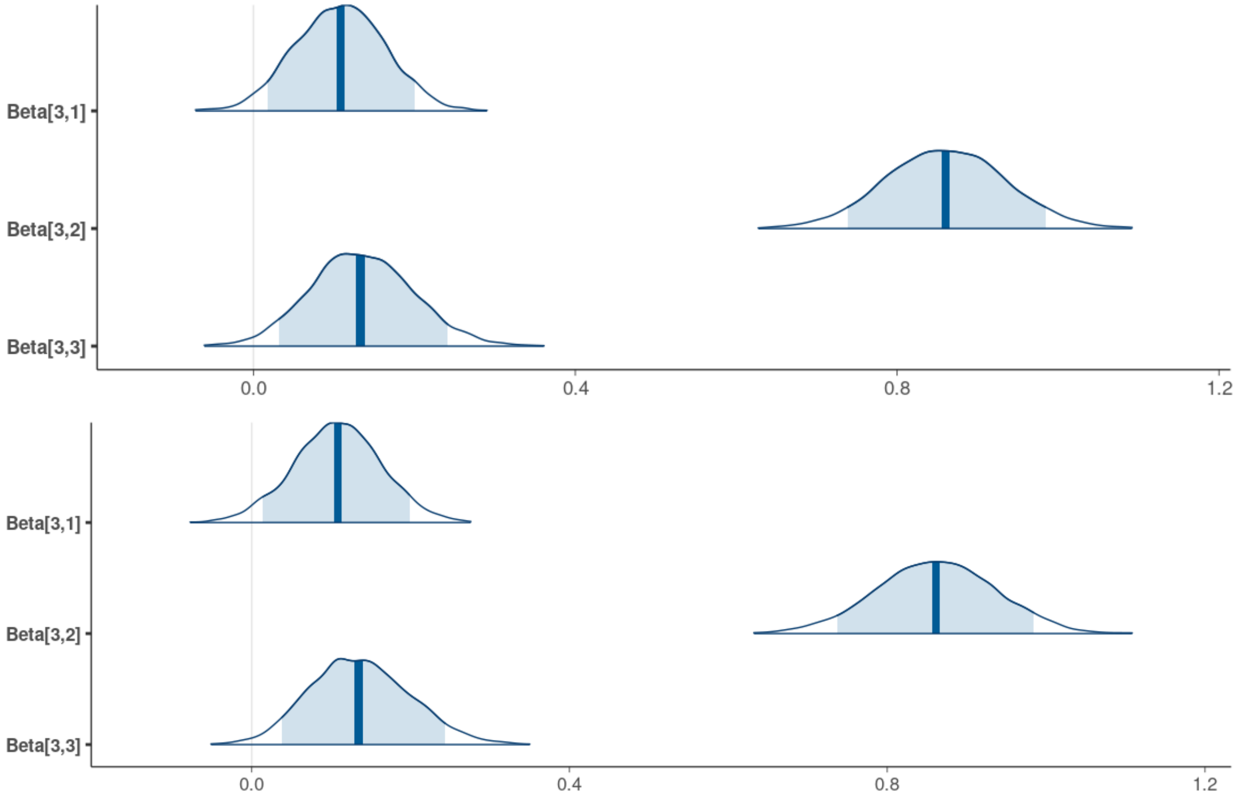
```
#Prior predictive check STAN CODE for hierarchical model

data {
  int<lower=0> N; // etamber of observations
  int<lower=0> J; // etamber of groups
  int<lower=0> P; // etamber of covariates
  int<lower=1,upper=J>group_idx[N]; // group indicator
  matrix[N,P] X;
  real mus;
  real taus;
  real sigmas;
}

generated quantities{
  vector[N] Y_rep;
  corr_matrix[P] Omega;
  vector<lower=0>[P] tau;
  vector[P] mu;
  vector[P] Beta[J]; // array size J where each element in array is vector
size P
  real<lower=0> sigma;

# HYPER PRIORS
  Omega=lkj_corr_rng(P,1);

  for (p in 1:P){
    mu[p]=normal_rng(0,mus);
    tau[p]= normal_rng(0,taus);
    while(tau[p]<0){
      tau[p]= normal_rng(0,taus);
    }
  }

# PRIORS
  sigma = normal_rng(0,sigmas);
  while(sigma<0){
      sigma= normal_rng(0,sigmas);
  }

  for (j in 1:J){
        Beta[j]=multi_normal_rng(mu,quad_form_diag(Omega,tau));
  }

# REPLICATED y
  for (n in 1:N){
    Y_rep[n]=normal_rng(X[n,]*Beta[group_idx[n]], sigma); // Array Beta size
J, matrix X has shape NxP
  }
}

-----------------------------------
# PRIOR PREDICTIVE CHECK STAN CODE for pooled model
data {
  int<lower=0> N; // etamber of observations
  int<lower=0> P; // etamber of covariates
  matrix[N,P] X;

  vector[P] muBe;
  vector[P] tauBe;
  corr_matrix[P] OmegaBe;
  real sigmas;

}

generated quantities{
  real<lower=0> sigma;
  vector[P] Beta;
  vector[N] Y_rep;

# PRIORS
  sigma = normal_rng(0,sigmas);
  while(sigma<0){
      sigma= normal_rng(0,sigmas);
    }

  Beta=multi_normal_rng(muBe,quad_form_diag(OmegaBe,tauBe));

  for (n in 1:N){
    Y_rep[n]=normal_rng(X[n,]*Beta, sigma); // Array Beta size J, matrix X
has shape NxP
    }
}
```

2/ Sensitivity analysis

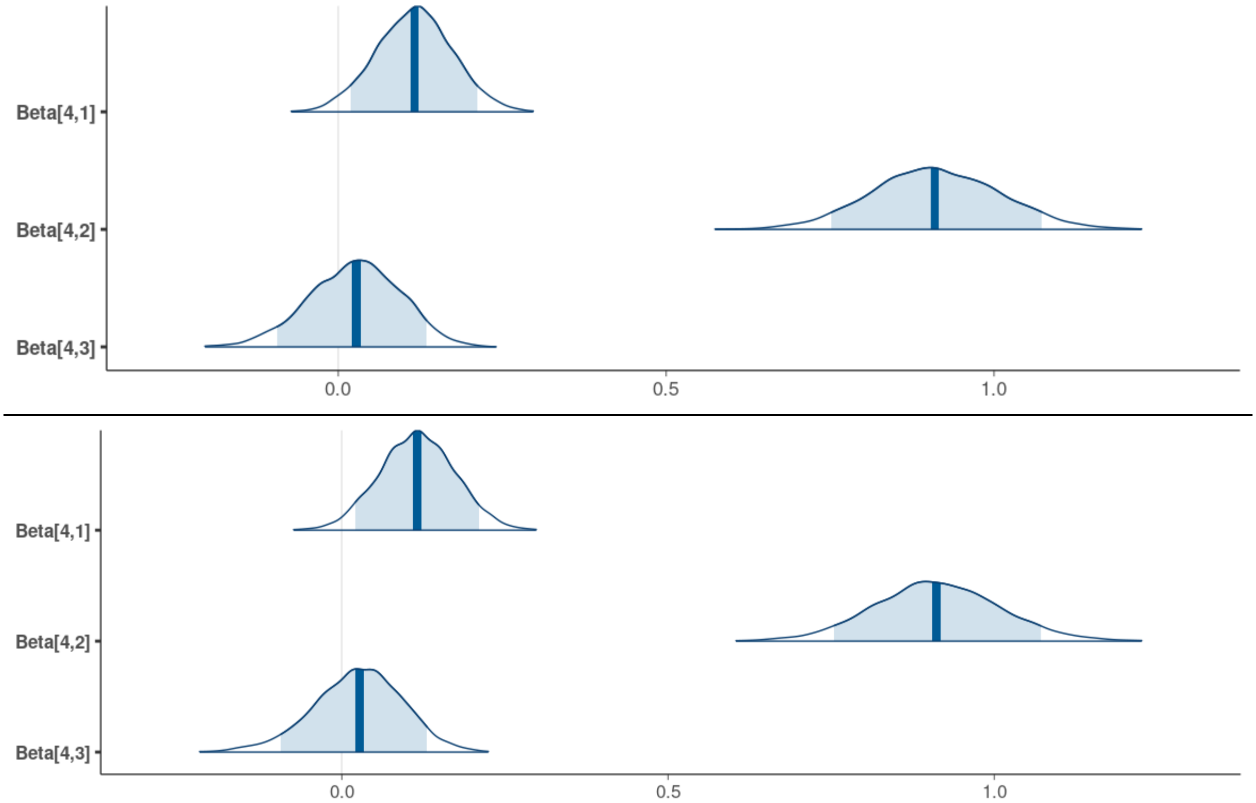


```
or each parameter, n_eff is a crude measure of effective sample size,
ind Rhat is the potential scale reduction factor on split chains (at
:onvergence, Rhat=1).
› print(fit_stanhier,pars='Beta')
Inference for Stan model: stan code hier.
ł chains, each with iter=3000; warmup=1500; thin=1;
›ost-warmup draws per chain=1500, total post-warmup draws=6000.

           mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rhat
3eta[1,1]  0.36       0 0.06  0.24  0.33  0.36  0.40  0.48  6727    1
3eta[1,2]  1.06       0 0.08  0.91  1.00  1.05  1.11  1.21  3352    1
3eta[1,3]  0.05       0 0.08 -0.10  0.00  0.05  0.10  0.20  3553    1
3eta[2,1]  0.09       0 0.06 -0.03  0.05  0.09  0.13  0.20  5411    1
3eta[2,2]  1.03       0 0.09  0.86  0.97  1.03  1.09  1.20  3299    1
3eta[2,3] -0.06       0 0.09 -0.23 -0.11 -0.05  0.01  0.10  2760    1
3eta[3,1]  0.11       0 0.06  0.00  0.07  0.11  0.15  0.22  6147    1
3eta[3,2]  0.86       0 0.07  0.71  0.81  0.86  0.91  1.01  4783    1
3eta[3,3]  0.14       0 0.06  0.01  0.09  0.13  0.18  0.27  4033    1
3eta[4,1]  0.12       0 0.06  0.00  0.08  0.11  0.16  0.23  5834    1
3eta[4,2]  0.91       0 0.10  0.73  0.85  0.90  0.97  1.10  3685    1
3eta[4,3]  0.03       0 0.07 -0.11 -0.02  0.03  0.07  0.15  3555    1
3eta[5,1] -0.20       0 0.05 -0.29 -0.23 -0.20 -0.16 -0.10  6121    1
3eta[5,2]  0.48       0 0.06  0.36  0.44  0.48  0.52  0.59  3389    1
3eta[5,3]  0.21       0 0.06  0.09  0.17  0.21  0.25  0.34  3019    1
3eta[6,1] -0.22       0 0.07 -0.36 -0.27 -0.22 -0.17 -0.09  5541    1
3eta[6,2]  0.51       0 0.08  0.34  0.45  0.51  0.56  0.67  3422    1
3eta[6,3]  0.20       0 0.08  0.05  0.15  0.20  0.26  0.38  3105    1
```

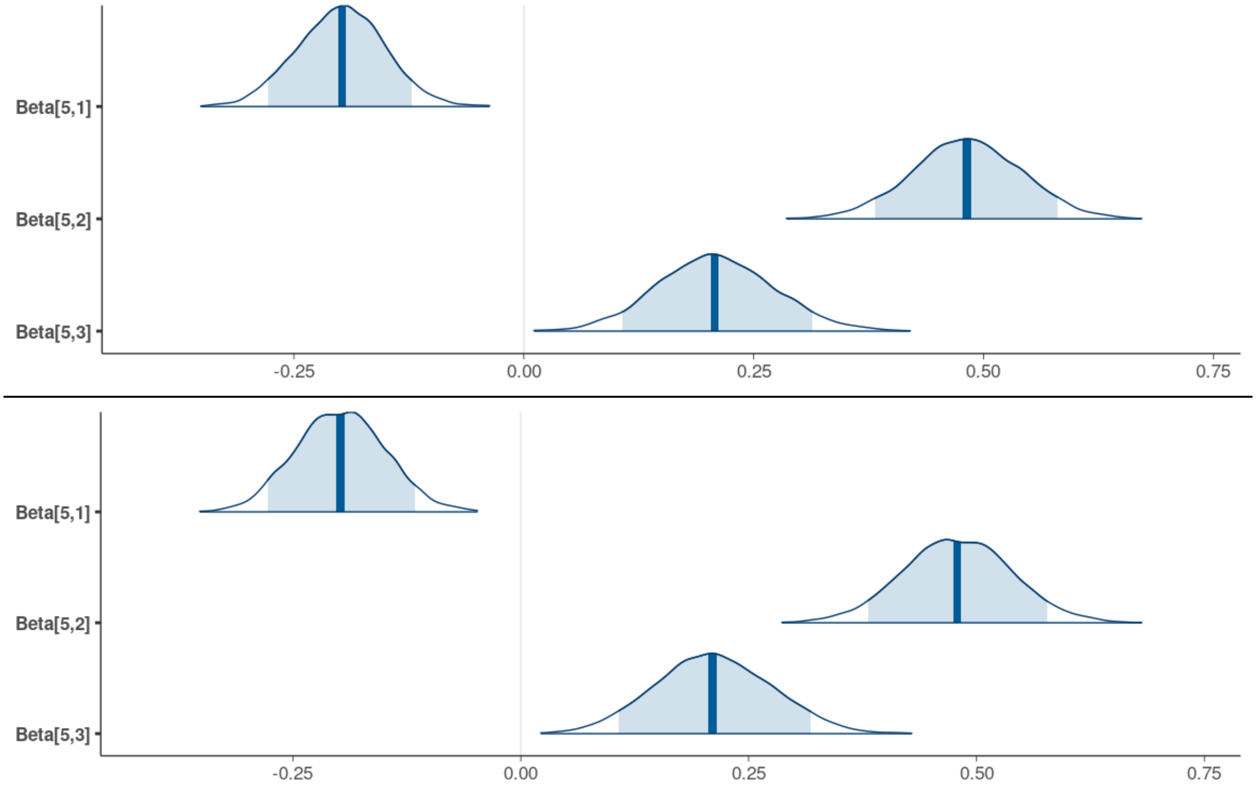
```
+     regex_pars ='Beta\\[1,')
> mcmc_areas(as.matrix(fit_stanhierSen),prob=.9, prob_outer = .999,
+     regex_pars ='Beta\\[3,')
> mcmc_areas(as.matrix(fit_stanhier),prob=.9, prob_outer = .999,
+     regex_pars ='Beta\\[3,')
> print(fit_stanhierSen,pars='Beta')
Inference for Stan model: stan code hier.
4 chains, each with iter=3000; warmup=1500; thin=1;
post-warmup draws per chain=1500, total post-warmup draws=6000.

           mean se_mean   sd  2.5%   25%   50%   75% 97.5% n_eff Rh
Beta[1,1]  0.37       0 0.06  0.25  0.33  0.37  0.41  0.49  6449
Beta[1,2]  1.05       0 0.08  0.90  1.00  1.05  1.10  1.20  5245
Beta[1,3]  0.05       0 0.08 -0.10  0.00  0.05  0.10  0.19  5017
Beta[2,1]  0.08       0 0.06 -0.03  0.04  0.08  0.12  0.19  5160
Beta[2,2]  1.04       0 0.08  0.87  0.98  1.03  1.09  1.20  3130
Beta[2,3] -0.06       0 0.09 -0.24 -0.12 -0.06  0.01  0.11  2478
Beta[3,1]  0.10       0 0.06 -0.01  0.07  0.10  0.14  0.22  6722
Beta[3,2]  0.87       0 0.08  0.72  0.81  0.86  0.92  1.02  5993
Beta[3,3]  0.14       0 0.06  0.02  0.10  0.14  0.19  0.27  5050
Beta[4,1]  0.11       0 0.06  0.00  0.08  0.12  0.15  0.23  5643
Beta[4,2]  0.91       0 0.10  0.71  0.84  0.90  0.97  1.11  3686
Beta[4,3]  0.03       0 0.07 -0.12 -0.02  0.03  0.07  0.16  3543
Beta[5,1] -0.20       0 0.05 -0.29 -0.23 -0.20 -0.16 -0.10  6020
Beta[5,2]  0.48       0 0.06  0.37  0.44  0.48  0.52  0.60  3755
Beta[5,3]  0.21       0 0.06  0.08  0.16  0.20  0.25  0.33  3638
Beta[6,1] -0.22       0 0.07 -0.36 -0.27 -0.22 -0.17 -0.09  7167
Beta[6,2]  0.51       0 0.08  0.35  0.46  0.51  0.57  0.68  3509
Beta[6,3]  0.20       0 0.08  0.04  0.14  0.19  0.25  0.37  3093

Samples were drawn using NUTS(diag_e) at Wed Dec  1 19:57:15 2021.
```



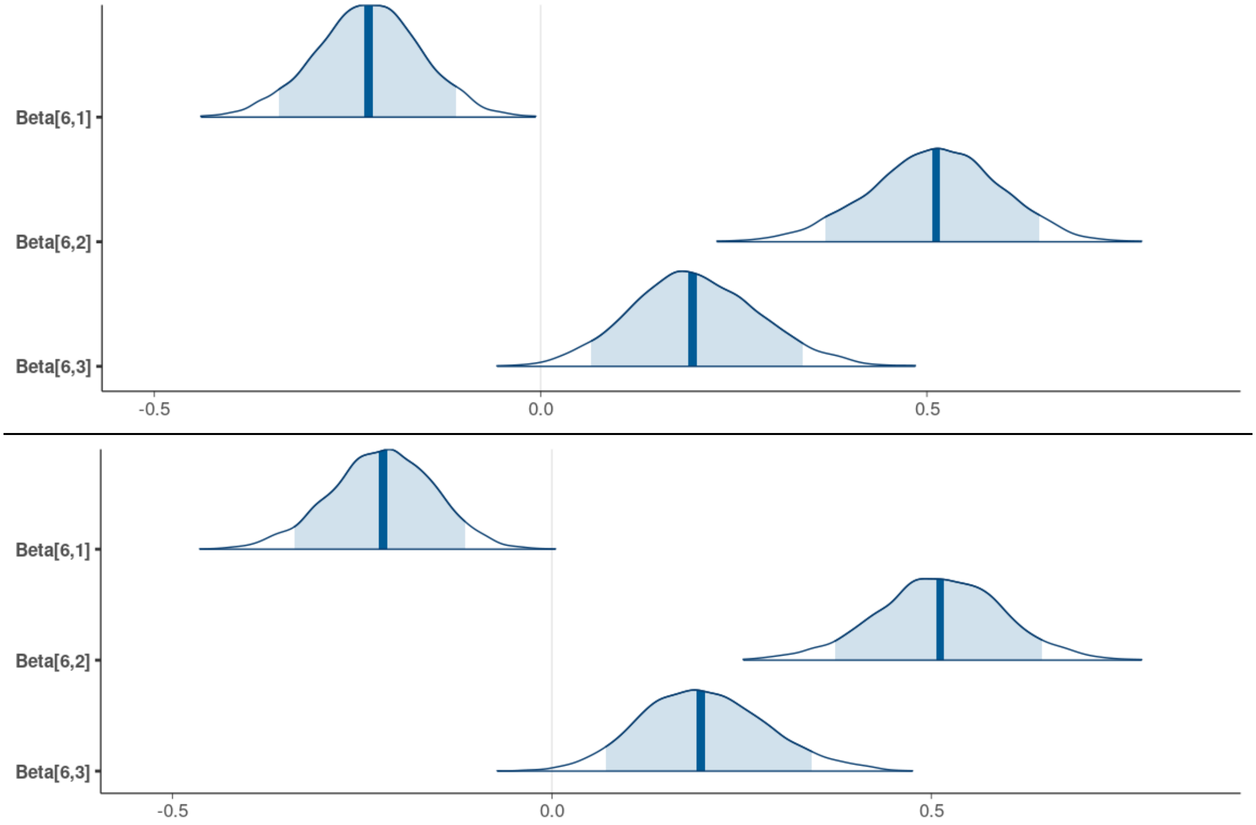
Marginal posterior distributions of coef group 3 (weak prior and vague prior respectively)



Marginal posterior distributions of coef group 4 (weak prior and vague prior respectively)



Marginal posterior distributions of coef group 5 (weak prior and vague prior respectively)



Marginal posterior distributions of coef group 6 (weak prior and vague prior respectively)