Zip Code Smoothing

Andres Potapczynski (ap3635)

11/9/2018

Summary

The purpose of this `Rmarkdown` is to see whether it makes sense that the zip codes have different default probabilities or else all can be regularized. The first approach will be to regularized them as if they come from the same distribution. The second approach will be to see if they come from a mixture.

The proposed model is the following

$$y_j \sim Binomial(n_j, \theta_j)$$

and we will also try

$$y_j \sim Poisson(n_j \theta_j)$$

since the Poisson distribution is more suitable for rare events.

Moreover, each θ_j is assumed to come from a Beta distribution.

$$\theta_j \sim Beta(\alpha, \beta)$$

where the hyperparameters can be obtained via

$$\alpha \sim Ga(2, 2)$$

and

$$\beta \sim Ga(2, 2)$$

The second approach has the following generative process [...]

Run the analysis

Load the data

```
file <-  './DBs/core.txt'
data <- read_delim(file = file, delim = '|')

# Sample the data
# pct = 1
# pct = 0.1
pct = 0.01
set.seed(seed = 42)
sample_size = round(pct * nrow(data))
sample <- sample(x = nrow(data), size = sample_size, replace = F)
data = data[sample, ]

# Change column format
data$postal_code = factor(data$postal_code)
```

Let's understand the distribution of the zip codes in the DB.

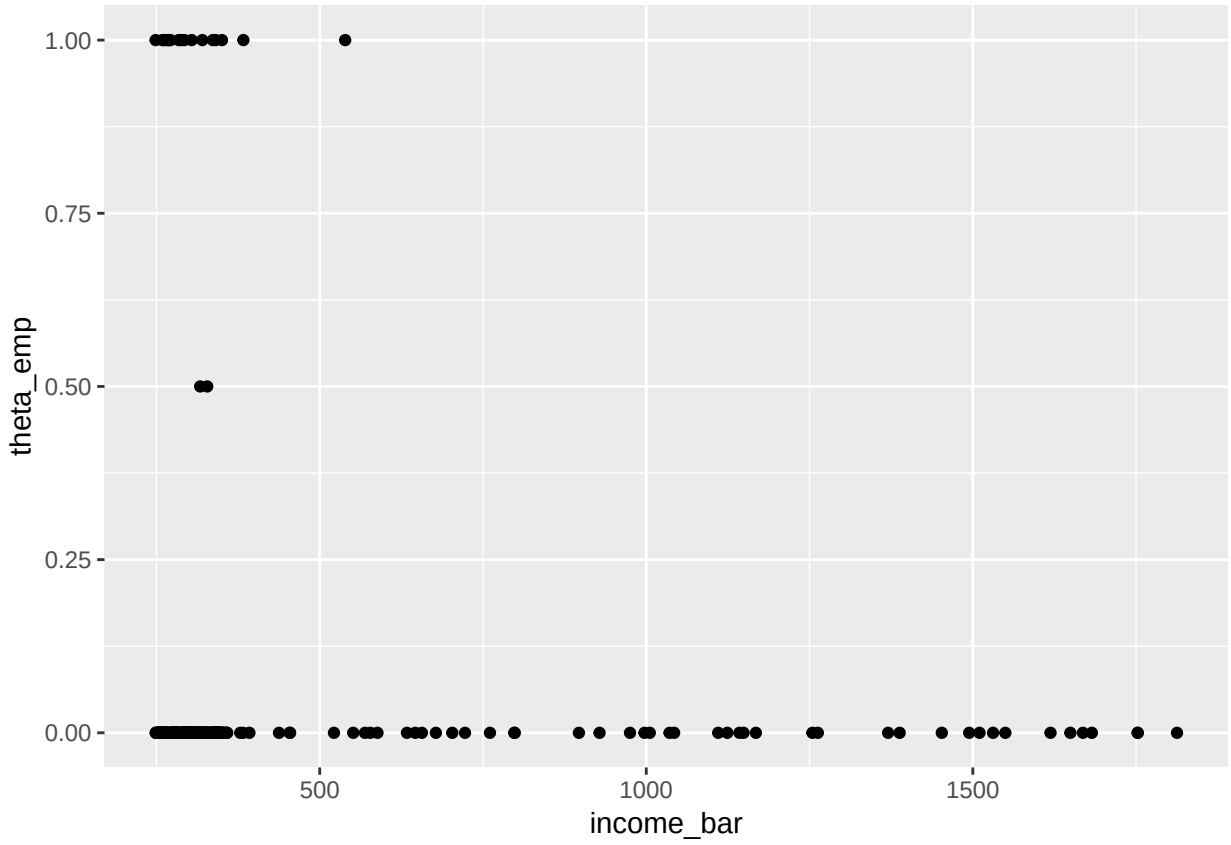
```
zip_summary = data %>%
  # group_by(postal_code, state) %>%
  # group_by(postal_code) %>%
  group_by(postal_code, city, state) %>%
  summarize(mort_no = n(), y_sum = sum(y), income_bar = mean(client_income)) %>%
  mutate(theta_emp = y_sum / mort_no) %>%
  arrange(desc(mort_no))

N = zip_summary$mort_no
y = zip_summary$y_sum
M = nrow(zip_summary)
```

The distribution of the zip codes is

```
ggplot(data = zip_summary, mapping = aes(x=theta_emp)) +
  geom_histogram(fill='lightblue', color='black', binwidth = 0.05)
```

```
zip_df <- zip_summary
# zip_df = zip_summary %>% filter(theta_emp > 0)
ggplot(data = zip_df, mapping = aes(x=income_bar, y=theta_emp)) +
  geom_point()
```



Now, I compile the proposed STAN model.

```
sm <- stan_model('./zip_code_v01.stan')
```

Run first approach

The data for the first model is

```
inputs = list(M=M, N=N, y=y)
model.v01 = sampling(sm, data=inputs)
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

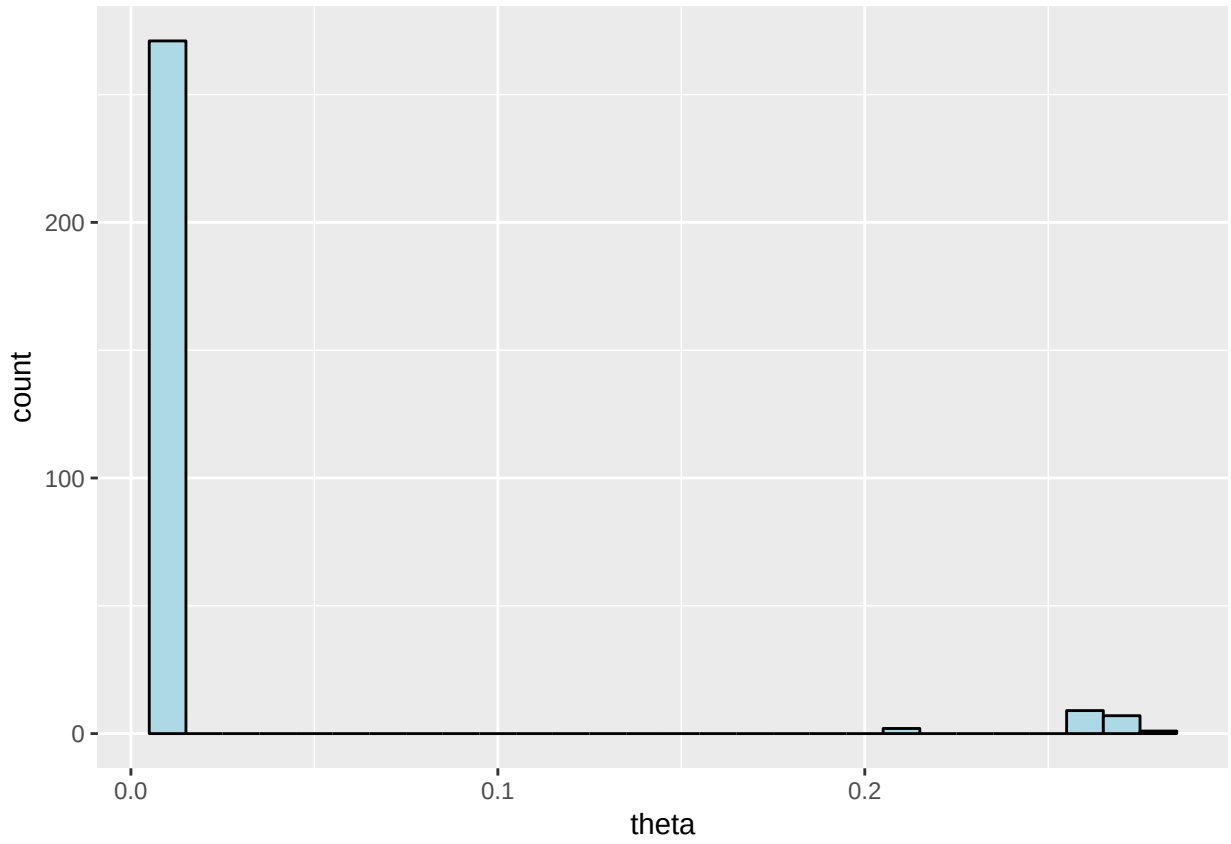
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(model.v01, digits=2, pars = c('alpha', 'beta'))
```

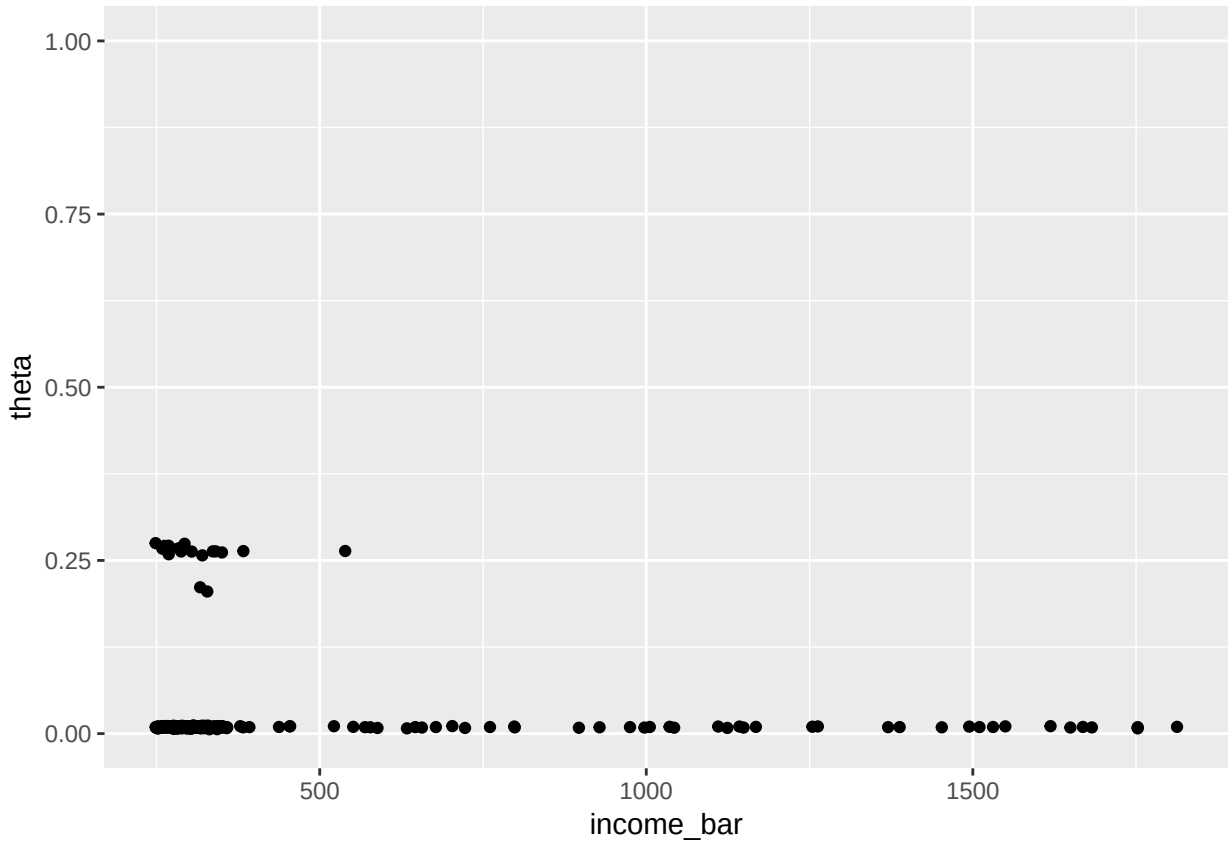
```
## Inference for Stan model: zip_code_v01.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## alpha 0.24    0.01 0.08 0.14 0.18 0.22 0.28  0.43    38 1.11
## beta  2.96    0.12 0.94 1.55 2.29 2.80 3.49  5.16    61 1.06
##
## Samples were drawn using NUTS(diag_e) at Sat Nov 10 07:45:39 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

Look at the shrinkage of the model

```
sims = rstan::extract(model.v01)
theta = apply(X = sims$theta, MARGIN = 2, FUN = median)
df = data.frame(theta=theta)
g_post <- ggplot(data = df, mapping = aes(x = theta)) +
  geom_histogram(fill='lightblue', color='black', binwidth = 0.01)
g_post
```

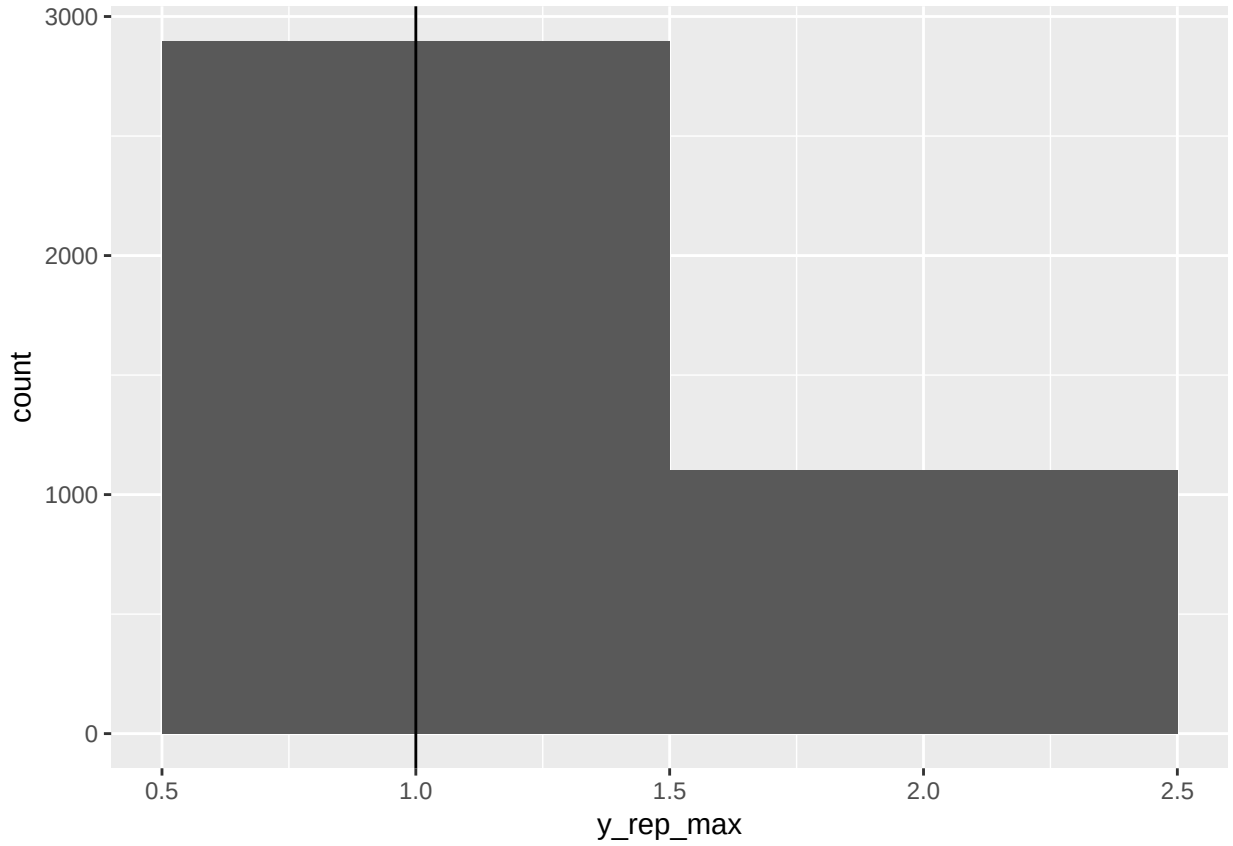



```
sims = rstan::extract(model.v01)
theta = apply(X = sims$theta, MARGIN = 2, FUN = median)
df2 <- data.frame(income_bar=zip_summary$income_bar, theta=theta)
ggplot(data = df2, mapping = aes(x=income_bar, y=theta)) +
  geom_point() +
  ylim(0, 1)
```



Generate data

```
sims = rstan::extract(model.v01)
y_rep_max = apply(X = sims$y_rep, MARGIN = 1, FUN = max)
df_rep = data.frame(y_rep_max = y_rep_max)
ggplot(data = df_rep, mapping = aes(x = y_rep_max)) +
  geom_histogram(binwidth = 1) +
  geom_vline(xintercept = max(zip_summary$y_sum))
```



Run second approach

Now, I compile the proposed STAN model.

```
sm_poi <- stan_model('./zip_code_v02.stan')
```

The data for the second model is

```
inputs = list(M=M, N=N, y=y)
model.v02 = sampling(sm_poi, data=inputs)
```

```
## Warning: There were 4 chains where the estimated Bayesian Fraction of Missing Information was low. S
## http://mc-stan.org/misc/warnings.html#bfmi-low
```

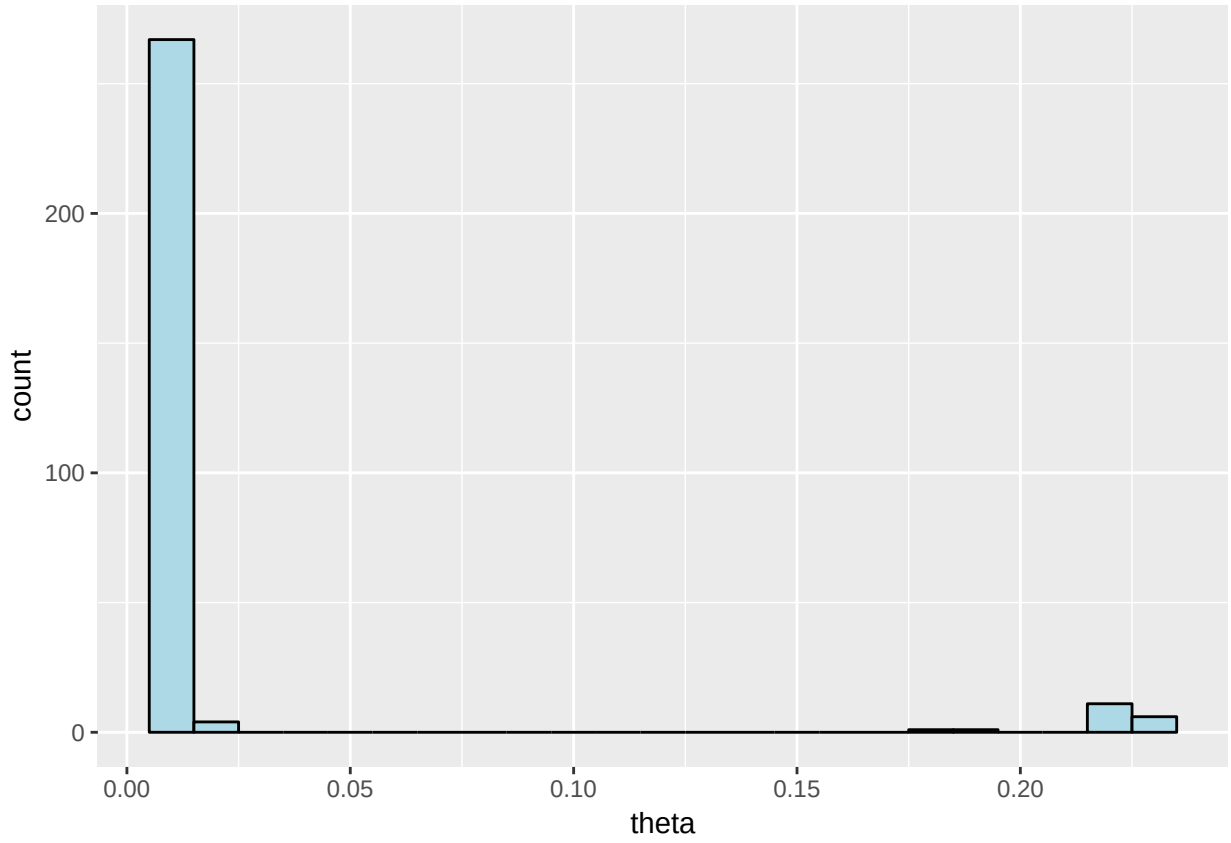
```
## Warning: Examine the pairs() plot to diagnose sampling problems
```

```
print(model.v02, digits=2, pars = c('alpha', 'beta'))
```

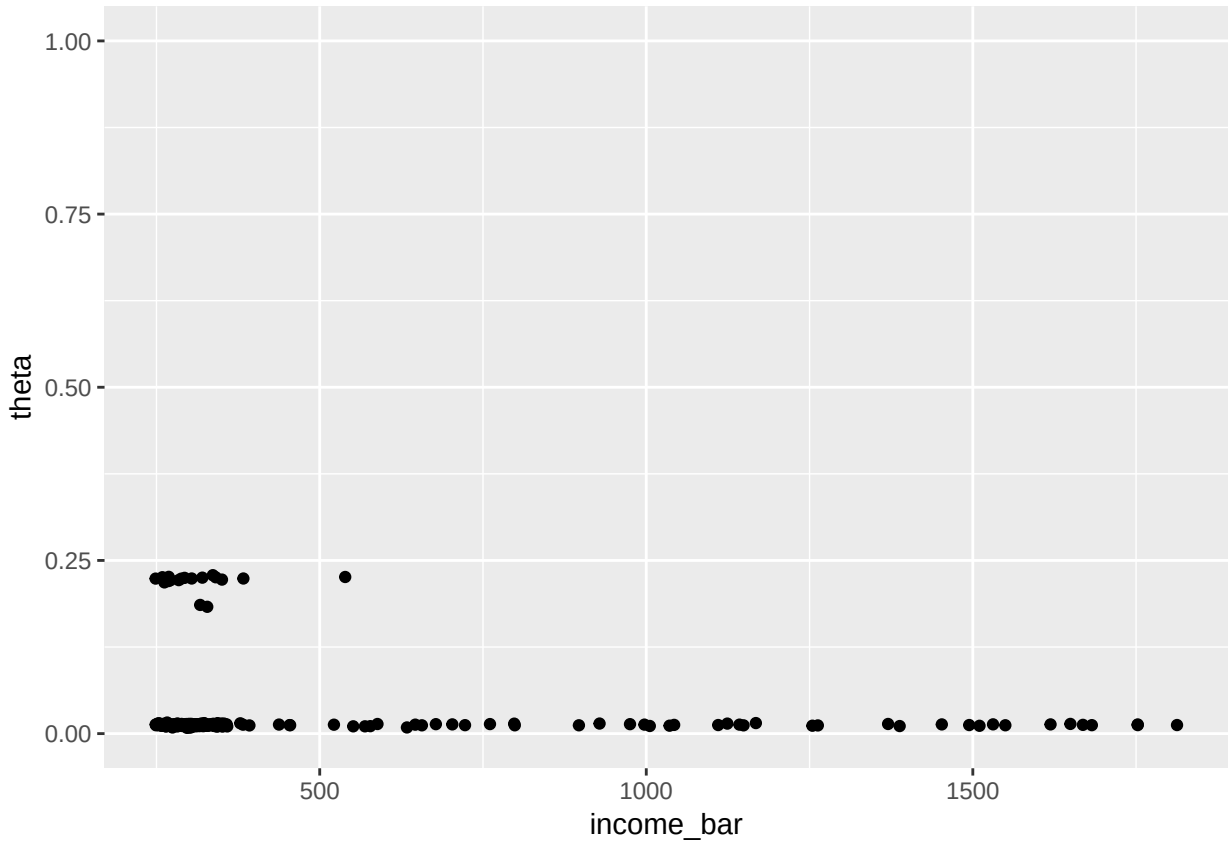
```
## Inference for Stan model: zip_code_v02.
## 4 chains, each with iter=2000; warmup=1000; thin=1;
## post-warmup draws per chain=1000, total post-warmup draws=4000.
##
##       mean se_mean   sd 2.5%  25%  50%  75% 97.5% n_eff Rhat
## alpha 0.26    0.01 0.07 0.15 0.20 0.24 0.30  0.43    67 1.05
## beta  2.99    0.08 0.91 1.56 2.32 2.86 3.54  5.00   129 1.02
##
## Samples were drawn using NUTS(diag_e) at Sat Nov 10 07:45:49 2018.
## For each parameter, n_eff is a crude measure of effective sample size,
```

```
## and Rhat is the potential scale reduction factor on split chains (at
## convergence, Rhat=1).
```

```
sims.poi = rstan::extract(model.v02)
theta = apply(X = sims.poi$theta, MARGIN = 2, FUN = median)
df = data.frame(theta=theta)
g_post.poi <- ggplot(data = df, mapping = aes(x = theta)) +
  geom_histogram(fill='lightblue', color='black', binwidth = 0.01)
g_post.poi
```




```
sims.poi = rstan::extract(model.v02)
theta = apply(X = sims.poi$theta, MARGIN = 2, FUN = median)
df2 <- data.frame(income_bar=zip_summary$income_bar, theta=theta)
ggplot(data = df2, mapping = aes(x=income_bar, y=theta)) +
  geom_point() +
  ylim(0, 1)
```



Run clustering approach

Now, I compile the proposed STAN model.

```
sm_cluster <- stan_model('./zip_code_v03.stan')
```

```
## recompiling to avoid crashing R session
```

The data for the third model is

```
K = 2
z0 = sample(x = K, size = M, replace=TRUE)
inputs = list(K=K, M=M, N=N, y=y, a0=rep(1, 2), z=z0)
model.v03 = sampling(sm_cluster, data=inputs)
```

```
K = 2
z0 = sample(x = K, size = M, replace=TRUE)
inputs = list(K=K, M=M, N=N, y=y, a0=rep(1, 2), z=z0)
model.opt = optimizing(sm_cluster, data=inputs)
```

were the results of the model are

```
print(model.v03, digits=2, pars = c('alpha', 'beta', 'phi', 'theta'))
```

```
sims.cluster = rstan::extract(model.v03)
```

Concluding remarks

[...]