Biomedical Statistics Coursework2

Junrui Wang

May 3, 2022

Contents

1 Intro

"I, Junrui Wang, certify that this assessed coursework is my own and unaided work, unless otherwise acknowledged through citations and references. I have not discussed my coursework with anyone else except when seeking clarification with the module lecturer via email or on MS Teams. I understand that all rules regarding academic integrity and plagiarism apply, and that violations of this will be treated as an examination offence. In particular, I have not shared any of my work with anyone else prior to submission."

2 Data Display

3 Reported COVID-19 deaths

The city allocated tome is Belo Horizonte.

3.1 Extract the reported COVID-19 deaths

```r
data.dir <- "/ic math/MSc Stats Data Science/Bio Stats/coursework/Coursework2"
out.dir <- "/ic math/MSc Stats Data Science/Bio Stats/coursework/Coursework2/plots"
dd <- read.csv(file =  file.path(data.dir, "civil_registry_covid_cities_detailed.csv")) %>%
  as.data.table()
mycolumns  <-  c('date','city','place','gender','age_group',
'deaths_covid19','deaths_stroke_covid19','deaths_heart_attack_covid19')
dfcd <- dd[,mycolumns, with = FALSE]
dfcd <- melt(dfcd, id.vars = c('date','city','place','gender','age_group'))
set(dfcd, dfcd[,which(is.na(value))],'value',0L)
dfcd <- dfcd[, list(deaths = sum(value)), by = c('date','city','gender','age_group')]
```

The first 5 data in the extracted dataset is

	date	city	gender	age_group	deaths
1	2019-01-01	Rio Branco	F	80-89	0
2	2019-01-01	Rio Branco	M	30-39	0
3	2019-01-01	Rio Branco	M	50-59	0
4	2019-01-01	Rio Branco	M	80-89	0
5	2019-01-01	Rio Branco	M	9-	0

3.2 Aggregate the reported COVID-19 deaths

```r
dfcd[, month := as.integer(substr(date,6,7 ))]
dfcd[, year := as.integer(substr(date,1,4))]
dfcd[, month_id := (year - 2019) * 12 + month]

# specify age groups as requested
set(dfcd, dfcd[, which(age_group == '9-')], 'age_group', '0-29')
set(dfcd, dfcd[, which(age_group == '10-19')], 'age_group', '0-29')
set(dfcd, dfcd[, which(age_group == '20-29')], 'age_group', '0-29')
set(dfcd, dfcd[, which(age_group == '90-99')], 'age_group', '>=90')
set(dfcd, dfcd[, which(age_group == '100+')], 'age_group', '>=90')
set(dfcd, dfcd[, which(is.na(age_group))], 'age_group', 'unknown')
dfcd <- subset(dfcd, ! age_group %in% c('>=90','unknown'))

# Group
cvid_agg <- dfcd[,.(Deaths = sum(deaths)), by = .(gender,age_group, month_id)]
setkey(cvid_agg, gender, age_group, month_id)
```

Group by gender, the age groups 0-29, 30-39, 40-49, 50-59, 60-69, 70-79, 80-89 (ignoring deaths in older ages), and by continuous month index we get the following table

	gender	age_group	month_id	Deaths
1	F	0-29	1.00	0
2	F	0-29	2.00	0
3	F	0-29	3.00	0
4	F	0-29	4.00	0
5	F	0-29	5.00	0

Table 1: The first 5 elements from *cvid_agg*

3.3 Plot the monthly, reported COVID-19 deaths in the city

```r
mycity <- dfcd[which(city == "Belo Horizonte"),]

mycity_agg <- mycity[,.(Deaths = sum(deaths)), by = .(gender,age_group, month_id)]
p <- ggplot(mycity_agg , aes(x = month_id, y = Deaths, fill = gender)) +
  geom_col() + facet_grid(~age_group) +
  theme_bw()+
    labs( x='Time/Month', y='Monthly Deaths') +
    theme(axis.text.x =element_text(size = 6))

ggsave(file=file.path(out.dir, 'monthly_death.png'), p, w=5, h=5)
include_graphics(file.path(out.dir, 'monthly_death.png') )
```

Apply the code above and have the plot Fig 1.

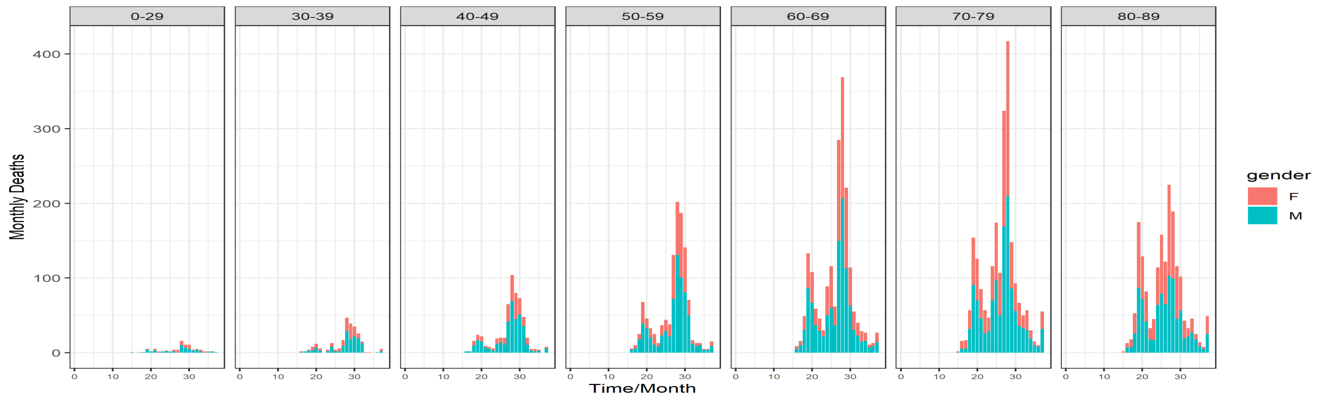

Figure 1: Plot the monthly, reported COVID-19 deaths in the city that you are assigned

3.4 Calculate the cumulating, monthly reported COVID-19 deaths in the city that you are assigned, again by gender and age group.

```
setkey(mycity_agg, 'month_id')
mycity_cumsum <- mycity_agg[,.(Cum_sum_Death = cumsum(Deaths),
                               month.id = month_id), by = .(age_group, gender)]

p <- ggplot(mycity_cumsum , aes(x = month.id, y = Cum_sum_Death, colour = gender)) +
  geom_line() + facet_grid(~age_group) +
  theme_bw()+labs( x='Time/Month', y='Monthly Deaths')+
    theme(axis.text.x =element_text(size = 6))
ggsave(file.path(out.dir, 'cumsum.png'), p, w =5, h = 5)
```

Fig 2 shows the calculation of cumulating, monthly reported COVID-19 deaths in Belo Horizonte and the table below is the data from the dataset *mycity_cumsum*.

	gender	age_group	month_id	Deaths
1	M	80-89	33.00	663
2	M	80-89	34.00	405
3	M	80-89	35.00	164
4	M	80-89	36.00	104
5	M	80-89	37.00	432

Table 2: The last 5 elements in the dataset *mycity_cumsum*

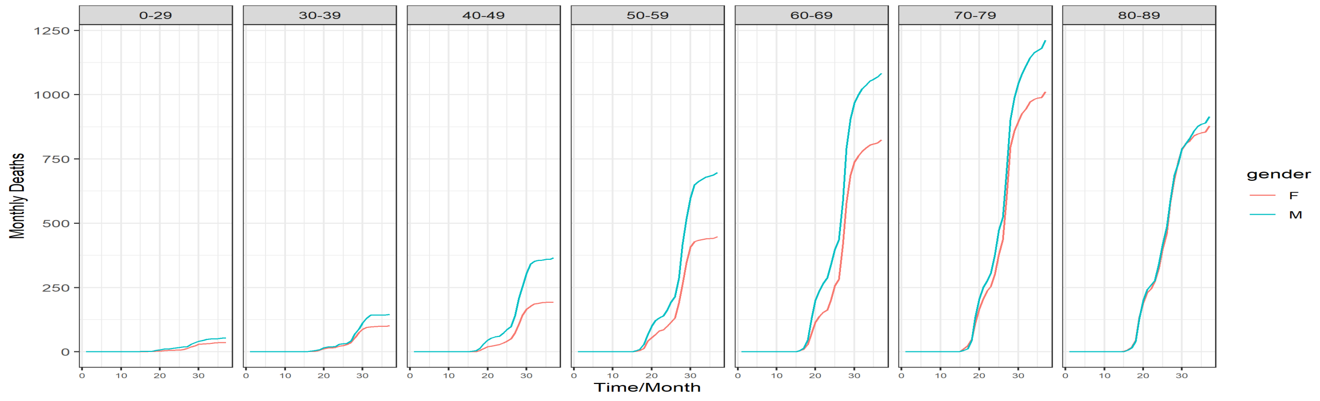

Figure 2: Cumulating monthly reported COVID-19 Deaths in Belo Horizonte

3.5 Plot for the cumulating, monthly reported COVID-19 deaths and the cumulating monthly excess deaths for each gender and age group in the city

```r
dfcd_base <- subset(dfcd, year == 2019)
dfcd_base <- dfcd_base[, list(deaths_2019 = mean(deaths)), by = c('city','month_id','month','gender', 'age_group')]
set(dfcd_base, NULL, 'month_id', NULL)
dfcd2 <- merge(dfcd, dfcd_base, by = c('city','month','gender', 'age_group'))
dfcd2[, exc_deaths := deaths - deaths_2019]
mycity2<- dfcd2[which(city == "Belo Horizonte"),]

mycity2_ttagg <- mycity2[,.(Deaths = sum(deaths),
                            exDeaths = sum(exc_deaths)), by = .(gender,age_group, month_id)]
mycity2_ttcumsum <-  mycity2_ttagg[, .(CumDeath = cumsum(Deaths),
                                       CumExcDeath =cumsum(exDeaths),
                                   month_id = month_id), by =.(gender,age_group)]
mycity2_ttcumsum <- data.table::melt(mycity2_ttcumsum, id.vars = c('gender','age_group','month_id'))

p <- ggplot(mycity2_ttcumsum , aes(x = month_id, y = value, fill =variable)) +
  geom_col() + facet_grid(gender~age_group) +
  theme_bw()+
    labs(
    x='Month Index',
    y='Numer of Death')

ggsave(file.path( out.dir, 'CumDexD.png'),p, w=5, h=5)
```

Fig 3 is as required.

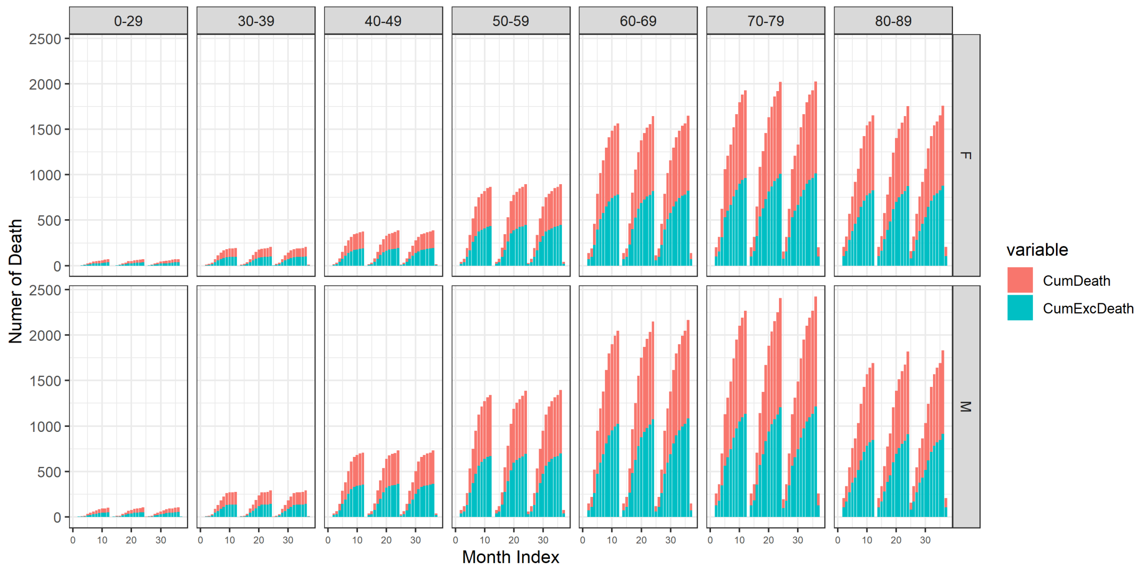

Figure 3: Cumulating and Culmulating Excess Deaths monthly reported in Belo Horizonte

3.6 Calculate the cumulating, monthly reported COVID-19 deaths in the city that you are assigned, per 1,000 population in each gender and age group.

```r
# load population size projections.
data <- as.data.table(read.csv('PNADc_2020.csv',encoding = 'latin1'))
age_group_data <- c('0-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89')
```

```r
cut_breaks <- c(-0.1,29,39,49,59,69,79,89)
data <-subset(data, age <90)
data$age <- cut(x = data$age, breaks = cut_breaks,
                  labels = age_group_data,
                  right = TRUE)

data <- data[!is.na(data$age),]
data <- data[, .(population = sum(population)),by = c('city','sex','age')]
setnames(data,'age', 'age_group')
setnames(data,'sex','gender')
set(data,which(data$gender == 'Women'), 'gender', 'F')
set(data,which(data$gender == 'Men'), 'gender', 'M')
mycity_data <- data[which(city =='Belo Horizonte'),]
mycity_cumsumpp <- merge(mycity_cumsum, mycity_data , by= c('gender','age_group'))
set(mycity_cumsumpp,NULL,'city', NULL)
deathperthousand <- mycity_cumsumpp[, Ppdeath :=1000*Cum_sum_Death/population]
```

deathperthousand stores the wanted data:

	gender	age_group	Cum_sum_Death	month.id	population	Ppdeath
1	M	80-89	687	28.00	24293	28.28
2	M	80-89	733	29.00	24293	30.17
3	M	80-89	790	30.00	24293	32.52
4	M	80-89	809	31.00	24293	33.30
5	M	80-89	831	32.00	24293	34.21

Table 3: The last 10 data in *deathperthousand*

3.7 Make a suitable plot that shows the total reported COVID-19 deaths by the end of 2021 across all cities, per 1,000 population in each city. Show cities on the x-axis and the total per 1,000 population on the y-axis.

```r
population_total <-data[, .(Population = sum(population)), by = .(city)] # 27 rows

dfcd_21 <- dfcd[which(year == 2021),][, .(Death =sum(deaths)), by = .(city)]
dfcd_21pp <- merge(dfcd_21, population_total, by = c('city'))
dfcd_21agg<- dfcd_21pp[, DeathperT := 1000*sum(Death)/Population, by = .(city)]

p <- ggplot(dfcd_21agg, aes(x = city, y = DeathperT)) + geom_col()+
  theme_bw() + theme(axis.text.x = element_text(angle = 60,size = 6)) +
    labs( x='City', y='Death Per one Thousand Population') +
    theme(axis.text.x =element_text(size = 6))

ggsave(file.path(out.dir, 'dth_per_thousand.png'),w = 10, h = 5)
include_graphics(file.path(out.dir, 'dth_per_thousand.png'))
```

Here we use the inner map to map the two dataset according to the name of the city. But since each computer decrypts different for the name of the city (i.e. city name Sao Luis loaded from `civil_registry_covid_cities_detailed.csv` is shown "S*o Lu*s"on my computer but from the dataset `PNADc_2020` is "São Paulo"), there are a few cities not matched.

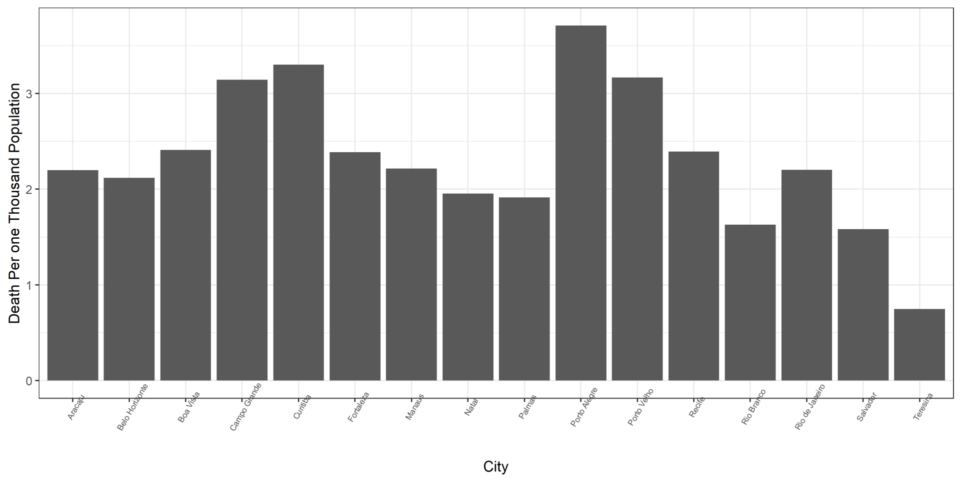

Figure 4: The total reported COVID-19 deaths by the end of 2021 across all cities, per 1,000 population in each city

3.8 Comment if this plot is a fair comparison of total reported COVID-19 deaths across cities.

No. Because the total amount of population for each city is different. For small population, if there are a few people dead, then it would tend to have a bigger proportion compared with the same amount of death in a larger population. So the city with smaller population tends to have a larger proportion of deaths.

4 Assessing the Coursework 1 model (6 points)

First we construct the dataset *stan_data* and generate the posterior estimates and save the data into **m1_fit**.

```r
file <- file.path(data.dir,'civil_registry_covid_cities_detailed.csv')
dd <- as.data.table(read.csv(file))
dd <- subset(dd, select = -c(state,state_ibge_code,city_ibge_code,created_at))
dd <- melt(dd, id.vars = c('date','city','place','gender','age_group'))
set(dd, dd[,which(is.na(value))],'value',0L)

# summarize daily deaths by city gender age
dd <- dd[, list(deaths = sum(value)), by = c('date','city','gender','age_group')]

# define date as Date object
set(dd, NULL, 'date', dd[, as.Date(date)])

# define months deaths, and month index
dw <- data.table(date = dd[, seq(min(date),max(date),1)])
dw[, month := as.integer(strftime(date, format = '%m'))]
dw[, year := as.integer(strftime(date, format = '%Y'))]
dw[, month_id := (year - 2019) * 12 + month]
tmp <- dw[, list(month_start = min(date)), by = 'month_id']
dw <- merge(dw, tmp, by = 'month_id')
dd <- merge(dd, subset(dw, select = -c(month,year)), by = 'date')

# specify age groups as requested
set(dd, dd[, which(age_group == '9-')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '10-19')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '20-29')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '90-99')], 'age_group', '>=90')
set(dd, dd[, which(age_group == '100+')], 'age_group', '>=90')
set(dd, dd[, which(is.na(age_group))], 'age_group', 'unknown')
dd <- subset(dd, ! age_group %in% c('>=90','unknown'))

# define gender as factor
set(dd, NULL, 'gender', dd[, factor(gender, levels = c('M','F'), labels = c('Men','Women'))])

# count deaths by month gender age
dd <- dd[, list(deaths = sum(deaths)), by = c('month_id','month_start','city','gender','age_group')]

# define age as factor
tmp <- dd[,sort(unique(age_group))]
set(dd, NULL, 'age_group', dd[, factor(age_group, levels = tmp)])

mycity <- dd[which(city ==  'Belo Horizonte'),]

data <- as.data.table(read.csv('PNADc_2020.csv',encoding = 'latin1'))
age_group_data <- c('0-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89')
cut_breaks <- c(-0.1,29,39,49,59,69,79,89)
data <-subset(data, age <90)

data$age <- cut(x = data$age, breaks = cut_breaks,
                  labels = age_group_data,
                  right = TRUE,)

data <- data[!is.na(data$age),]
data <- data[, .(population = sum(population)),by = c('city','sex','age')]
setnames(data,'age', 'age_group')
setnames(data,'sex','gender')
mycity_data <- data[which(city =='Belo Horizonte'),]
dpop <- data
#data populaiton & deaths
dds <- merge(mycity,dpop, by = c("city",'age_group', 'gender'))

ddsF <- dds[which(gender == 'Women'),]
tmp <- unique(subset(ddsF, select = age_group))
setkey(tmp, age_group)
tmp[, age_group_id := 1:nrow(tmp)]
ddsF <- merge(ddsF, tmp, by = 'age_group')
ddsF[, id := 1:nrow(ddsF)]
tmp <- subset(ddsF, select = c(id, age_group_id))
tmp <- dcast.data.table(tmp, id ~ age_group_id, value.var = 'age_group_id')
setnames(tmp, 1 + 1:max(ddsF$age_group_id), paste0('AGE',1:max(ddsF$age_group_id)))
for (x in paste0('AGE',1:max(ddsF$age_group_id)))
{
  set(tmp, which(!is.na(tmp[[x]])), x, 1L)
  set(tmp, which(is.na(tmp[[x]])), x, 0L)
}
tmp <- unname(as.matrix(subset(tmp, select = -c(id,AGE1))))

# make binary dummy variables for each month except first
tmp2 <- subset(ddsF, select = c(id, month_id))
tmp2 <- dcast.data.table(tmp2, id ~ month_id, value.var = 'month_id')
setnames(tmp2, 1 + 1:max(ddsF$month_id), paste0('MO',1:max(ddsF$month_id)))
for (x in paste0('MO',1:max(ddsF$month_id)))
{
  set(tmp2, which(!is.na(tmp2[[x]])), x, 1L)
  set(tmp2, which(is.na(tmp2[[x]])), x, 0L)
}
tmp2 <- unname(as.matrix(subset(tmp2, select = -c(id,MO1))))
```

```r
tmp <- cbind(tmp, tmp2)
design_matrix_X <- tmp

stan_data <- list()
stan_data$N <- nrow(ddsF)
stan_data$K <- ncol(design_matrix_X )
stan_data$y <- ddsF$deaths
stan_data$offset <- log(ddsF$population)
stan_data$X <- design_matrix_X

poi_regression_txt <- "
data{
    int<lower=1> N;
    int<lower=1> K;
    int<lower=0> y[N];
    vector[N] offset;
    matrix[N,K] X;
}
parameters{
    real beta0;
    vector[K] beta;
}
transformed parameters{
    vector[N] log_lambda;
    log_lambda = beta0 + X * beta + offset;
}
model{
    beta0 ~ normal( 0 , 10 );
    beta ~ normal( 0 , 1 );
    y ~ poisson_log( log_lambda );
}
"
if (!file.exists(file.path(out.dir, paste0('poi_deaths_fit_',city_of_student_ascii,'.rds'))))
{
  poi_regression_compiled <- rstan::stan_model(
    model_name = 'poi_regression',
    model_code = gsub('\t',' ',poi_regression_txt)
    )
  #m1_init_beta0 <- subset(dds, age_group_id == 1 & month_id == 1)[, log(deaths/population)]
  m1_fit <- rstan::sampling(poi_regression_compiled,
    data = stan_data,
    warmup = 5e2, iter = 1e4, chains = 4,
    init = list(list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K))
                )
  )
  saveRDS(m1_fit, file = file.path(out.dir, paste0('poi_deaths_fit_',city_of_student_ascii,'.rds')))
}
if (file.exists(file.path(out.dir, paste0('poi_deaths_fit_',city_of_student_ascii,'.rds'))))
{
  m1_fit <- readRDS( file.path(out.dir, paste0('poi_deaths_fit_',city_of_student_ascii,'.rds')) )
}
```

4.1 Assess convergence, mixing, and make the trace plot corresponding to the model parameter with smallest effective sample size.

We applied 5000 iterations with 500 iterations as warm-up and built 4 chains. From the code below, we see the effective sample size for $\hat{\beta}$s are no less than 1155.1. There are β_{0-42} with $\hat{R}$ for each parameter close to 1. This implies good convergence for the chain, thereby the estimated values for $\hat{\beta}$s are trustworthy.

```
print(m1_fit, digits = 3, prob = c(0.025,0.5,0.975))
#                       mean se_mean    sd      2.5%       50%     97.5% n_eff  Rhat
# beta0               -9.854   0.002 0.052    -9.959    -9.854    -9.751  1155 1.001
# beta[1]              0.072   0.001 0.048    -0.023     0.072     0.165  8204 1.000
# beta[2]              0.924   0.000 0.039     0.848     0.924     0.999  6204 1.000
# beta[3]              1.840   0.000 0.034     1.774     1.840     1.904  5422 1.000
# beta[4]              2.564   0.000 0.031     2.504     2.564     2.625  4806 1.000
# beta[5]              3.500   0.000 0.030     3.440     3.500     3.559  4775 1.000
# beta[6]              4.114   0.000 0.030     4.056     4.114     4.172  4665 1.000
...

po <- summary(m1_fit)$summary
print(min(po[,'n_eff']))
#1155.1
print(max(po[,'Rhat']))
#1.001915
```

Fig 5 also shows the convergence for the parameter with the smallest effective sample size.

```
po <- rstan:::extract(m1_fit, inc_warmup = TRUE, permuted = FALSE)
bayesplot:::color_scheme_set("mix-blue-pink")
#trace plot
p <- bayesplot:::mcmc_trace(po,  pars = "beta0", n_warmup = 5e2,
        facet_args = list(nrow = 1, labeller = label_parsed))
```

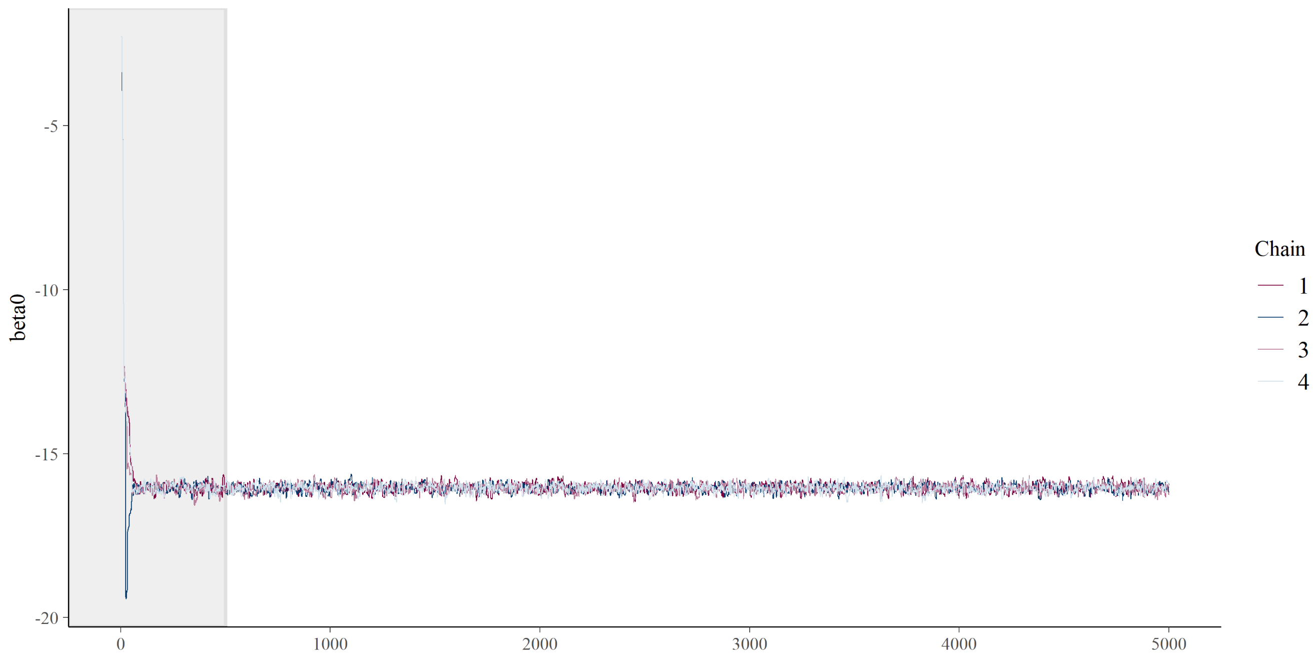

Figure 5: The traceplot for beta0.

There are tota 43 betas but we here only consider the first 6 βs as examples. Fig 9 shows that the β_0 is negatively correlated with β_{1-6} but β_{1-6} are all positively correlated with each other.

```r
po <- rstan:::extract(m1_fit, inc_warmup = FALSE, permuted = FALSE )
p <- bayesplot::mcmc_pairs(po, regex_pars = "beta", diag_fun = "dens", off_diag_fun = "hex")
ggsave(file.path(out.dir,'m1_pairplot.pdf'), p, w=20, h=20)
```

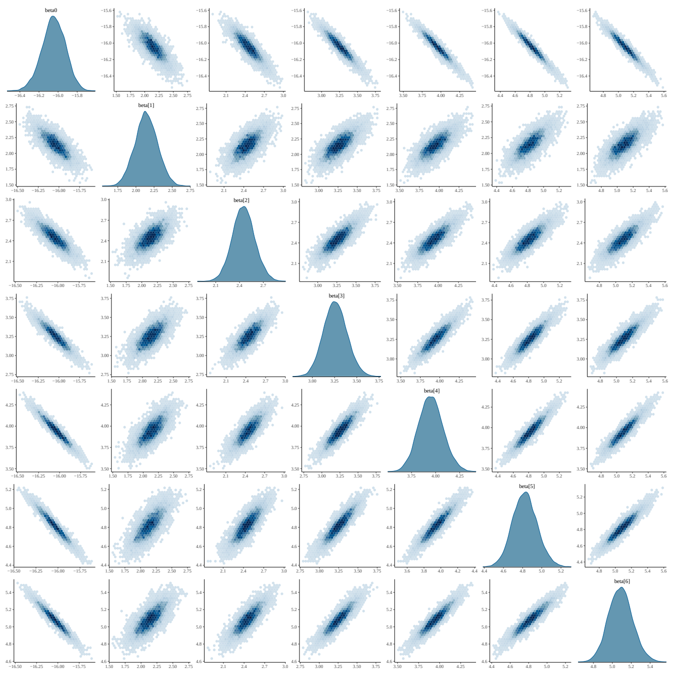

Figure 6: Here we take the pair plot for β_{1-6}. (sub select)

4.2 Make a posterior predictive check and report the proportion of data points inside the 95% posterior predictive credibility intervals associated with each data point, separately for men and women.

```r
setkey(dds, 'gender')
dds[, id := 1:nrow(dds)]
ddsFM <- dds

esti_coef <- rstan::summary(m1_fit)$summary
log_lambdaest <- esti_coef[grepl('log_lambda', rownames(esti_coef)),][,1]
ddsFM <- cbind(ddsFM, log_lambdaest)

log_poi_fixed_pp <- ddsFM[,
  {
    lambda <- exp(log_lambdaest)
    pred_y <- rpois(10000,lambda)
    pred_y_stat <- quantile(pred_y, p=c(0.5,0.025,0.975), type=1)
    list(PP_DEATHS_STAT = pred_y_stat, PP_DEATHS_TYPE=c('PP_M','PP_CL','PP_CU'))
  },
  by=c('gender','month_id','age_group')]

log_poi_fixed_pp2 <- dcast.data.table(log_poi_fixed_pp, month_id+gender+age_group~PP_DEATHS_TYPE, value.var='PP_DEATHS_STAT')

# merge observations
```

```r
tmp2 <- subset(ddsFM, select = c('month_id','gender','deaths', 'age_group', 'id'))
log_poi_fixed_pp3 <- merge(log_poi_fixed_pp2, tmp2, by = c('month_id','gender','age_group'))
set(log_poi_fixed_pp3, which(log_poi_fixed_pp3$gender == 'Women'),
'id', log_poi_fixed_pp3[ which(log_poi_fixed_pp3$gender == 'Women'),]$id - 259 +1)

p <- ggplot(log_poi_fixed_pp3, aes(x=id)) +
        geom_point(aes(y=PP_M)) +
        geom_point(aes(y=deaths), colour='red') +
        geom_errorbar(aes(ymin=PP_CL, ymax=PP_CU)) +
        theme_bw()  +
        labs(x='Individuals', y='posterior predictive vs actual deaths\n')

ggsave(file=file.path(out.dir, 'posterior_pred.png'), p, w=20, h=6)
include_graphics(file.path(out.dir, 'posterior_pred.png'))

ds
# posterior predictive check
tmp3 <- log_poi_fixed_pp3[which(gender =='Women'),][,length(which(deaths <= PP_CU & deaths >= PP_CL)) / length(deaths)]
tmp3
#> 0.9266409

tmp4 <- log_poi_fixed_pp3[which(gender =='Men'),][,length(which(deaths <= PP_CU & deaths >= PP_CL)) / length(deaths)]
tmp4

#0.9111969
```

Since we are considering the model

$$\begin{aligned}
&Y_{a,t} \sim \text{Poisson}\left(\lambda_{a,t}\right)\\
&\log\lambda = \beta_0 + X\beta + \log P\\
&\beta_0 \sim \mathcal{N}\left(0, 10^2\right)\\
&\beta = \left(\beta_1, \ldots, \beta_{A+T-2}\right) \sim \mathcal{N}\left(0, 1^2\right)
\end{aligned}$$

, the first step in a posterior predictive check is to calculate the posterior predictive distribution

$$f\left(y_j^* \mid y\right) = \int l\left(y_j^* \mid \theta\right) f(\theta \mid y) d\theta$$

, where $\theta = \lambda = \exp(\log\lambda)$ and then we find the probability that an observed data point lies in the 95% credible interval of the corresponding posterior predictive distribution,

$$\frac{1}{n}\sum_j 1\left(y_j \in \left[q_{j,\alpha/2}^*, q_{j,1-\alpha/2}^*\right]\right)$$

where $q_{j,\alpha}^*$ denotes the α quantile of the posterior predictive distribution $f\left(y_j^* \mid y\right)$.
By posterior check, the model for the women has about 92.3% observed data points in the 95% credible interval of the corresponding posterior predictive distribution. Then we re-run the code above again but change the dataset from women to men (change the code $ddsF <- dds[which(gender ==' Women'),]$ as $ddsF <- dds[which(gender ==' Men'),]$) and then run the rest of the chunk). For men, it has about 91.1% observed data in the 95% credible interval.

4.3 Comment if the model from Coursework 1 provides a sufficiently good fit to the data for the city that you are assigned.

Referring to Fig 7 and the proportion of data points inside the 95% posterior predictive credibility intervals, we see that the Bayesian model applied to women and men both had high probability coverage for the posterior credible interval so this model fits the data well.

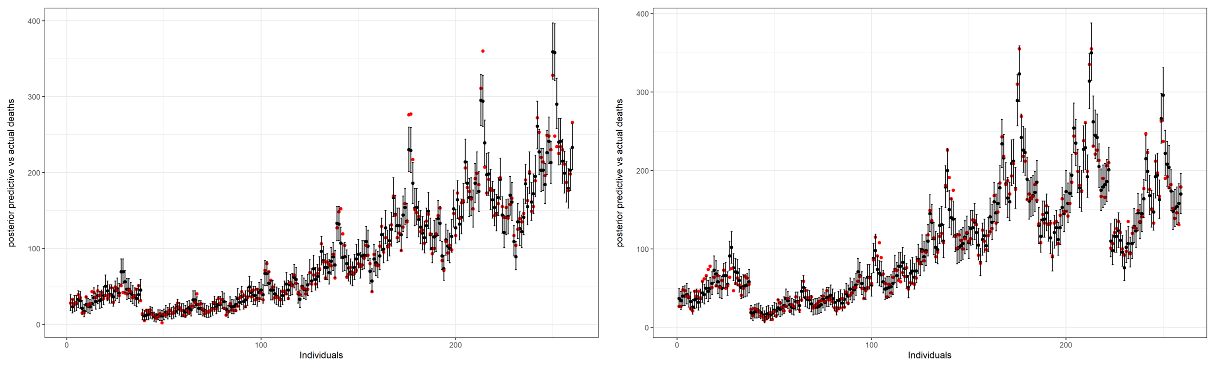

Figure 7: The posterior predictive check for men and women separately in my city (Men right, Women Left). The x-axis is the individual index for men and women separately and y-axis represents the death.

5 Poisson model with random time effects

5.1 Describe Bayesian Poisson model

$$
\begin{aligned}
&Y_{a,g,t} \sim Poisson\left(\lambda_{a,g,t}\right)\\
&\log \lambda_{a,g,t} = \beta_0 + \beta_1 \text{ age } + \beta_2 \text{ gender } + \beta_{3,a,g} \text{ Time } + \log P_{a,g}\\
&\beta_0 \sim N\left(0,1^2\right)\\
&\beta_1 \sim N\left(0,1^2\right)\\
&\beta_2 \sim N\left(0,1^2\right)\\
&\beta_{3,a,g} \sim N\left(\bar{\beta}_{3,a,g}, \sigma_T^2\right)\\
&\sigma_T \sim \text{ Half-Cauchy(0,1)}\\
&\bar{\beta}_{3,a,g} \sim \text{Normal}\left(0,11^2\right)
\end{aligned}
$$

Here we define $Y_{a,g,t}$ to be the number of monthly age- and gender-specific all-cause deaths in my city Belo Horizonte, β_0 is the fixed baseline coefficients for death, β_1 is the fixed effect coefficient for age, β_2 is the fixed effect coefficient for gender, $\beta_{3,a,g}$ is a random effect on Time for the corresponding age and gender, and $P_{a,g}$ is the population for the corresponding age and gender (as we assume the population is independent of the time). I choose the hyperparameter $\bar{\beta}_3$ with standard deviation 11 because there are total 37 months and the **sd** for $1:37$ is 11. In addition, *gender* takes value $\{0,1\}$ where 0 represents men and 1 represents women, and *Time* is monthly-recorded with month_id (which we no longer treat as factor but real continuous values). I created a matrix for *age*: say there are $A=7$ age groups, and we allocate only one fixed effect to the age groups $a=2,\ldots,A$, which is similar to the design matrix X but without the month column.

5.2 Implement the hierarchical Bayesian Poisson model in Stan.

```r
#Data
dpop <- data
dds <- merge(mycity,dpop, by = c("city",'age_group', 'gender'))
tmp <- unique(subset(dds, select = age_group))
setkey(tmp, age_group)
tmp[, age_group_id := 1:nrow(tmp)]
dds <- merge(dds, tmp, by = 'age_group')
tmp <- unique(subset(dds, select = gender))
setkey(tmp, gender)
tmp[, gender_id := 0:(nrow(tmp)-1)]
dds <- merge(dds, tmp, by = 'gender')
dds[, id := 1:nrow(dds)]
ageM <- model.matrix(~ -1 + age_group, data = dds)
ageM <- design_matrix_X[,-1]
#   separate random effects using Stan syntax
log_poi_RD_txt <- "
```

```
data{
  int<lower=1> N; // number of observations
  int<lower=1> K; // number of units
  int<lower=0> y[N];
  vector[N] age; //
  vector[N] gender;
  vector[N] time;
  int<lower=1> Nk_max; // max number of observations for a unit
  int<lower=1, upper=Nk_max> units_to_obs_length[K]; // number of observations per unit
  int<lower=0, upper=N> units_to_obs[K, Nk_max]; //index of observations per unit, with the rest set to 0
}
parameters{

  real beta0;   // baseline
  real beta1;  // age
  real beta2; // gender
  // random effects on treatment effect
  real beta3[K];
  real<lower=0> beta3_hyper_sd;
  real beta3_hyper_mean;
}
transformed parameters{
  vector[N] obs_log_lambda;

  // fixed effects
  obs_log_lambda= beta0 + beta1*age + beta2 * gender + beta3_hyper_mean * time ;
  //
  // add random effects
  for(k in 1:K)
  {
    //
    // add baseline random effects and treatment effects
    obs_log_lambda[ units_to_obs[k, 1:units_to_obs_length[k] ] ] +=
      (
        time[ units_to_obs[k, 1:units_to_obs_length[k] ] ] * beta3[k]
      );
  }
}
model{
  beta3_hyper_mean ~ normal(0,11);
  beta3_hyper_sd ~ cauchy(0,1);
  beta3 ~ normal( 0, beta3_hyper_sd);
  beta0 ~ normal(0,1);
  beta1 ~ normal(0,1);
  beta2 ~ normal(0,1);
  y ~ poisson_log(obs_log_lambda);
}
"


# define data in format needed for model specification

stan_data <- list()
stan_data$N <- nrow(dds)
stan_data$K <- max(dds$month_id)
stan_data$y <- dds$deaths
stan_data$age <- age
stan_data$time <- dds$month_id
stan_data$gender <- dds$gender_id
```

```r
stan_data$Nk_max <- length(unique(dds$age_group_id))*2
# define number of observations per unit
tmp <- dds[, list(LEN=length(age_group_id)), by='month_id']
tmp <- tmp[order(month_id)]
stan_data$units_to_obs_length <- tmp$LEN
# define index of observations per unit, with the rest set to 0
tmp <- matrix(data=0, nrow=stan_data$K, ncol=stan_data$Nk_max)
for(k in 1:stan_data$K)
{
  tmp2 <- dds[,which(month_id==k)]
  tmp[k, 1:length(tmp2) ] <- tmp2
  stopifnot(length(tmp2)==stan_data$units_to_obs_length[k])
}
stan_data$units_to_obs <- tmp

# compile the model
log_poi_RD_compiled <- rstan::stan_model(
  model_name= 'log_poi_RD_model',
  model_code = gsub('\t',' ',log_poi_RD_txt)
  )

# run Stan
log_poi_RD_fit <- rstan::sampling(log_poi_RD_compiled,
  data=stan_data,
  warmup=1e3, iter=1e4, chains=4
  )

# save
saveRDS(log_poi_RD_fit, file=file.path(out.dir, 'log_poi_RD_fit.rds'))

po <- rstan:: summary(log_poi_RD_fit)
print(po)
```

5.3 Check convergence and mixing

We applied 5000 iterations with 1000 iterations as warm-up and built 4 chains. From the code below, we see the smallest effective sample size is 1042. There are β_0,$\beta_1[1:7]$, $beta_2$, $\beta_3[1:37]$ and hyperparameter $\sigma, \bar{\beta}_3$ with $\hat{R}$ for each parameter close to 1. This implies good convergence for the chain, thereby the estimated values for $\hat{\beta}$s are trustworthy.

```r
print(log_poi_RD_fit, digits = 3, prob = c(0.025,0.5,0.975))
# Inference for Stan model: log_poi_Random.
# 4 chains, each with iter=10000; warmup=1000; thin=1;
# post-warmup draws per chain=4000, total post-warmup draws=16000.
#
#                          mean se_mean    sd      2.5%        50%      97.5% n_eff  Rhat
# beta0                   2.840   0.000 0.030     2.783      2.840      2.900  4375 1.001
# beta1[1]                   0.308   0.000 0.002     0.303      0.308      0.313  1482 1.003
#...
# beta2                  -0.146   0.000 0.009    -0.163     -0.146     -0.129  1042 1.004
# beta3[1]                0.010   0.000 0.016    -0.020      0.010      0.042 15135 1.000
# beta3[2]               -0.020   0.000 0.013    -0.048     -0.020      0.005 12853 1.000
# beta3[3]                0.013   0.000 0.010    -0.007      0.013      0.032 13284 1.000
# beta3[4]                0.009   0.000 0.008    -0.007      0.009      0.025 13648 1.000
# beta3[5]                0.025   0.000 0.007     0.012      0.025      0.038 13087 1.000
# beta3[6]                0.007   0.000 0.006    -0.005      0.007      0.018 11602 1.000
# beta3[7]               -0.030   0.000 0.006    -0.041     -0.030     -0.019 11542 1.000
# ...
# beta3_hyper_sd           0.016   0.000 0.002      0.013      0.016      0.022 12172 1.000
# beta3_hyper_mean         0.017   0.000 0.004      0.010      0.017      0.025  3860 1.002
#...
po <- summary(log_poi_RD_fit)$summary
est <-  po[grepl("hyper|beta",rownames(po_Sp)),]
min(est[,'n_eff'])
# 1042
max(est[,'Rhat'])
```

Take $\beta_0, \beta_1[1]$ for example. By Fig 8, the graph shows good convergence

```
po <- rstan:::extract(log_poi_RD_fit, inc_warmup = TRUE, permuted = FALSE)
bayesplot:::color_scheme_set("mix-blue-pink")
#trace plot
p <- bayesplot:::mcmc_trace(po, pars = c("beta0","beta1[1]"), n_warmup = 1e3, facet_args = list(nrow =2, labeller = label_parsed))
ggsave(file.path(out.dir, 'RDtraceplot.png'), p, w = 10, h =5)
include_graphics(file.path(out.dir, 'RDtraceplot.png'))
```

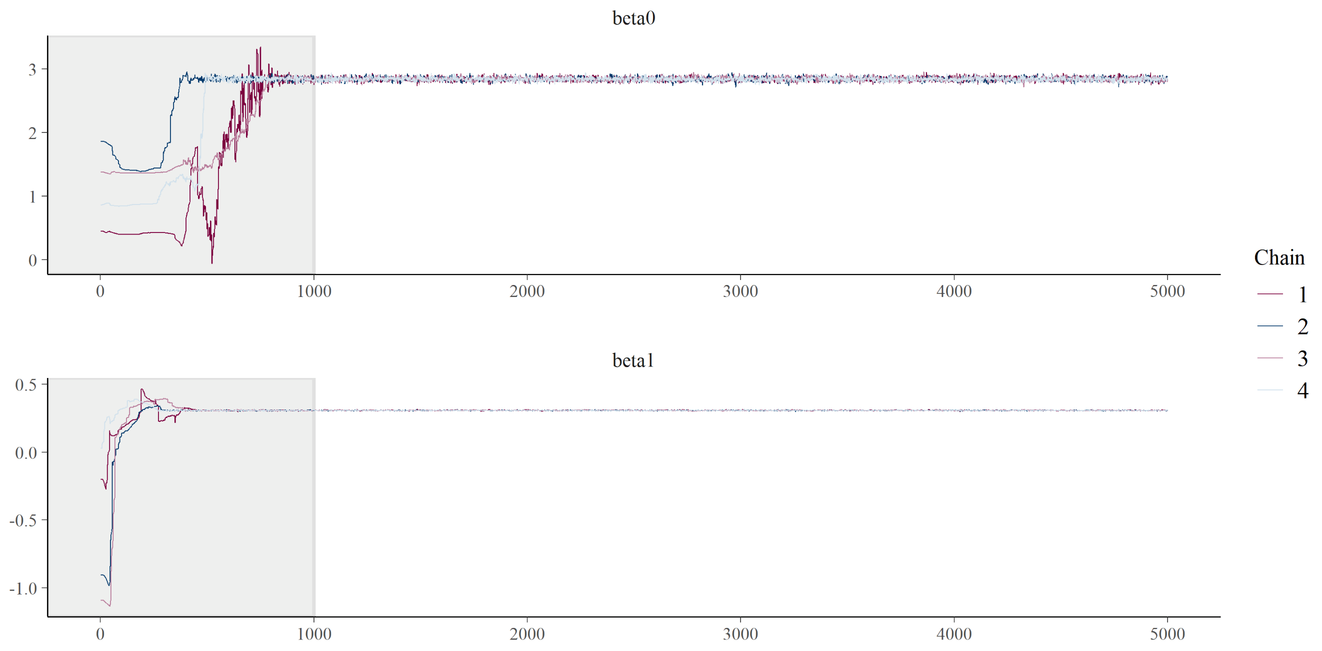

Figure 8: The traceplot for $\beta_0, \beta_1[1]$.

5.4 Make a posterior predictive check and report the proportion of data points inside the 95% posterior predictive credibility intervals associated with each data point, separately for men and women.

```
ddsFM <- dds
esti_coef <- rstan::summary(log_poi_RD_fit)$summary
log_lambdaest <- esti_coef[grepl('log_lambda', rownames(esti_coef)),][,1]
ddsFM <- cbind(ddsFM, log_lambdaest)

log_poi_fixed_pp <- ddsFM[,
  {
    lambda <- exp(log_lambdaest)
    pred_y <- rpois(10000,lambda)
    pred_y_stat <- quantile(pred_y, p=c(0.5,0.025,0.975), type=1)
    list(PP_DEATHS_STAT = pred_y_stat, PP_DEATHS_TYPE=c('PP_M','PP_CL','PP_CU'))
  },
  by=c('gender','month_id','age_group')]

log_poi_fixed_pp2 <- dcast.data.table(log_poi_fixed_pp, month_id+gender+age_group~PP_DEATHS_TYPE, value.var='PP_DEATHS_STAT')

# merge observations
tmp2 <- subset(ddsFM, select = c('month_id','gender','deaths', 'age_group', 'id'))
log_poi_fixed_pp3 <- merge(log_poi_fixed_pp2, tmp2, by = c('month_id','gender','age_group'))
set(log_poi_fixed_pp3, which(log_poi_fixed_pp3$gender == 'Women'),
'id', log_poi_fixed_pp3[ which(log_poi_fixed_pp3$gender == 'Women'),]$id - 259 +1)

# posterior predictive check
tmp3 <- log_poi_fixed_pp3[which(gender =='Women'),][,length(which(deaths <= PP_CU & deaths >= PP_CL)) / length(deaths)]
tmp3
#> 0.7490347

tmp4 <- log_poi_fixed_pp3[which(gender =='Men'),][,length(which(deaths <= PP_CU & deaths >= PP_CL)) / length(deaths)]
tmp4

#0.7837838
```

By posterior predictive check, we see that for women, there is only aboud 74.9% of data within the 95% posterior predictive credibility intervals and 78.4% for men.

5.5 Plot the posterior predictive checks

```
p <- ggplot(log_poi_fixed_pp3, aes(x=month_id)) +
        geom_point(aes(y=PP_M)) +
        geom_point(aes(y=deaths), colour='red') +
        geom_errorbar(aes(ymin=PP_CL, ymax=PP_CU)) +
        theme_bw()  +
        labs(x='Individuals', y='posterior predictive vs actual deaths\n') +
        facet_grid(gender~age_group)

ggsave(file=file.path(out.dir, 'posterior_pred_RD.png'), p, w=20, h=6)
include_graphics(file.path(out.dir, 'posterior_pred_RD.png'))
```

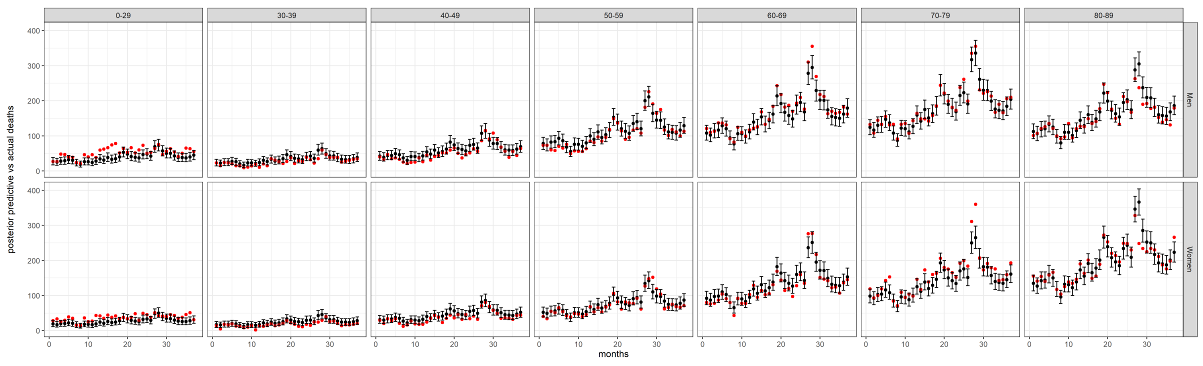

Figure 9: The posterior predictive checks with time on the x-axis, and age and gender in row facets

5.6 Comment

By the posterior check, we see that there are over 25% of observed data outside the 95% posterior predictive credibility intervals and over 21% for men. Even though this model is valid for the city applied, the model is less as good as the model from Coursework1.

6 Your best model

We construct ”simplest plausible Bayesian Poisson model with the fewest possible number of variables”. Let us denote observed all cause deaths among men in the city that you are given, month t and age group a by $Y_{a,t}$, and denote the total number of age groups by A, such that $a = 1, \ldots, A$, and denote the total number of months by T, such that $t = 1, \ldots, T$. In total, we have $A * T * 2$ observations. We then model the count data with the Poisson model

$$\begin{aligned}
&Y_{a,t} \sim \text{Poisson}\left(\lambda_{a,g,t}\right) \\
&\log \lambda = \beta_0 + X\beta + \log P \\
&\beta_0 \sim \mathcal{N}\left(0, 10^2\right) \\
&\beta = (\beta_1, \ldots, \beta_{A+T-2}) \sim \mathcal{N}\left(0, 1^2\right)
\end{aligned}$$

where $\lambda_{a,g,t}$ is the expected number of all cause deaths in this city in age group a, gender g and month t; P is an $A * T * 2$ dimensional column vector of age-specific population sizes in this city, which for deaths in age group a gives the population size in age group a; β_0 is a real-value baseline regression parameter; and β is a $A + T - 1$ dimensional vector of regression parameter, one for each covariate. The first $A + T - 2$ columns of the design matrix X are defined the same as for coursework 1 but add a column 'gender' which has value 0 for men and 1 for women. And the matrix X is therefore has double rows as in coursework 1.

6.1 Implement your hierarchical Bayesian Poisson model in Stan.

Data Construction::

```r
file <- file.path(data.dir,'civil_registry_covid_cities_detailed.csv')
dd <- as.data.table(read.csv(file))
dd <- subset(dd, select = -c(state,state_ibge_code,city_ibge_code,created_at))
dd <- melt(dd, id.vars = c('date','city','place','gender','age_group'))
set(dd, dd[,which(is.na(value))],'value',0L)

# summarize daily deaths by city gender age
dd <- dd[, list(deaths = sum(value)), by = c('date','city','gender','age_group')]

# define date as Date object
set(dd, NULL, 'date', dd[, as.Date(date)])

# define months deaths, and month index
dw <- data.table(date = dd[, seq(min(date),max(date),1)])
dw[, month := as.integer(strftime(date, format = '%m'))]
dw[, year := as.integer(strftime(date, format = '%Y'))]
dw[, month_id := (year - 2019) * 12 + month]
tmp <- dw[, list(month_start = min(date)), by = 'month_id']
dw <- merge(dw, tmp, by = 'month_id')
dd <- merge(dd, subset(dw, select = -c(month,year)), by = 'date')

# specify age groups as requested
set(dd, dd[, which(age_group == '9-')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '10-19')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '20-29')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '90-99')], 'age_group', '>=90')
set(dd, dd[, which(age_group == '100+')], 'age_group', '>=90')
set(dd, dd[, which(is.na(age_group))], 'age_group', 'unknown')
```

```r
dd <- subset(dd, ! age_group %in% c('>=90','unknown'))

# define gender as factor
set(dd, NULL, 'gender', dd[, factor(gender, levels = c('M','F'), labels = c('Men','Women'))])

# count deaths by month gender age
dd <- dd[, list(deaths = sum(deaths)), by = c('month_id','month_start','city','gender','age_group')]

# define age as factor
tmp <- dd[,sort(unique(age_group))]
set(dd, NULL, 'age_group', dd[, factor(age_group, levels = tmp)])

data <- as.data.table(read.csv('PNADc_2020.csv',encoding = 'latin1'))
age_group_data <- c('0-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89')
cut_breaks <- c(-0.1,29,39,49,59,69,79,89)
data <-subset(data, age <90)

data$age <- cut(x = data$age, breaks = cut_breaks,
                  labels = age_group_data,
                  right = TRUE,
)

data <- data[!is.na(data$age),]
data <- data[, .(population = sum(population)),by = c('city','sex','age')]
setnames(data,'age', 'age_group')
setnames(data,'sex','gender')
dpop <- data
#data populaiton & deaths
dds <- merge(dd,dpop, by = c("city",'age_group', 'gender'))
city_of_student_ascii ='Belo Horizonte'
dds <- dds[which(city == 'Belo Horizonte'),]

design_matrix_X <- model.matrix(~ -1 + age_group + as.factor(month_id) +as.factor(gender), data = dds)
design_matrix_X <- design_matrix_X[,-1]


stan_data <- list()
stan_data$N <- nrow(dds)
stan_data$K <- ncol(design_matrix_X )
stan_data$y <- dds$deaths
stan_data$offset <- log(dds$population)
stan_data$X <- design_matrix_X
```

Fit into the Rstan model:

```r
poi_regression2_txt <- "
data{
    int<lower=1> N;
    int<lower=1> K;
    int<lower=0> y[N];
    vector[N] offset;
    matrix[N,K] X;
}
parameters{
    real beta0;
    vector[K] beta;
}
transformed parameters{
    vector[N] log_lambda;
    log_lambda = beta0 + X * beta + offset;
}
model{
    beta0 ~ normal( 0 , 10 );
    beta ~ normal( 0 , 1 );
    y ~ poisson_log( log_lambda );
}
"
if (!file.exists(file.path(out.dir, paste0('poi_deaths_fit2_',city_of_student_ascii,'.rds'))))
{
  poi_regression2_compiled <- rstan::stan_model(
    model_name = 'poi_regression',
    model_code = gsub('\t',' ',poi_regression2_txt)
    )
  #m1_init_beta0 <- subset(dds, age_group_id == 1 & month_id == 1)[, log(deaths/population)]
  m2_fit <- rstan::sampling(poi_regression2_compiled,
    data = stan_data,
    warmup = 5e2, iter = 1e4, chains = 4,
    init = list(list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K))
                )
  )
  saveRDS(m2_fit, file = file.path(out.dir, paste0('poi_deaths_fit2_',city_of_student_ascii,'.rds')))
}
if (file.exists(file.path(out.dir, paste0('poi_deaths_fit2_',city_of_student_ascii,'.rds'))))
{
  m2_fit <- readRDS( file.path(out.dir, paste0('poi_deaths_fit2_',city_of_student_ascii,'.rds')) )
}
```

6.2 Check convergence and mixing in an appropriate manner

We applied 10000 iterations with 500 iterations as warm-up and built 4 chains. The minimum effective sample size is about 2372.063 and the maximum $\hat{R}$ is 1.001915. There are β_{0-43} and $\hat{R}$ for all parameters are close to 1 which indicates good convergence. Also, Fig 10 shows that the chain converges.

```
print(m2_fit)
#                      mean se_mean   sd      2.5%      25%      50%      75%     97.5% n_eff Rhat
# beta0               -9.48    0.00 0.04     -9.55    -9.50    -9.48    -9.45     -9.41  2681    1
# beta[1]              0.15    0.00 0.03      0.10     0.13     0.15     0.18      0.21 14948    1
# beta[2]              0.98    0.00 0.02      0.93     0.96     0.98     1.00      1.03 11968    1
# beta[3]              1.84    0.00 0.02      1.80     1.83     1.84     1.86      1.89  9864    1
#...
# beta[41]             0.37    0.00 0.04      0.28     0.34     0.37     0.39      0.45  3797    1
# beta[42]             0.47    0.00 0.04      0.39     0.44     0.47     0.50      0.55  3329    1
# beta[43]            -0.43    0.00 0.01     -0.44    -0.43    -0.43    -0.42     -0.41 54284    1
po <- summary(m2_fit)$summary
print(min(po[,'n_eff']))
#2372.063
print(max(po[,'Rhat']))
#1.001915

po <- rstan:::extract(m2_fit, inc_warmup = TRUE, permuted = FALSE)
bayesplot:::color_scheme_set("mix-blue-pink")

```

```r
p <- bayesplot:::mcmc_trace(po,  pars = "beta0", n_warmup = 5e2,
        facet_args = list(nrow = 1, labeller = label_parsed))

ggsave(file.path(out.dir,"bayesplot2.png"), p,w=10,h =5)
include_graphics(file.path(out.dir,"bayesplot2.png"))
```

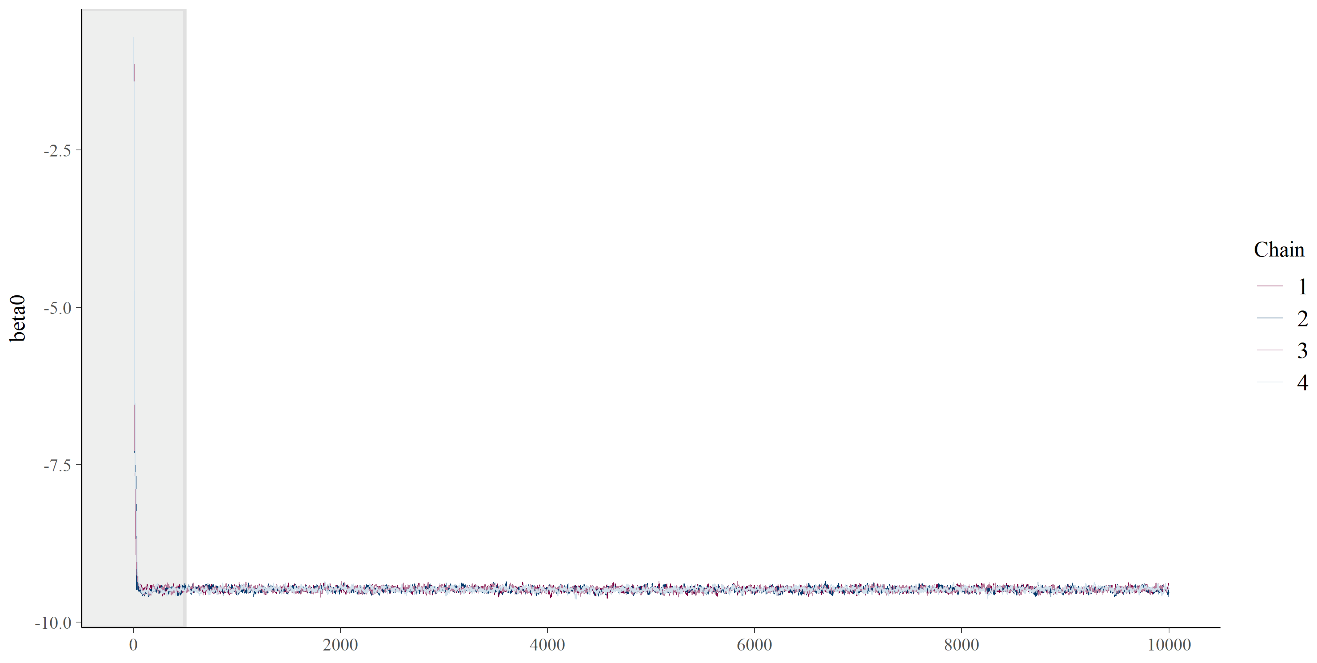

Figure 10: The traceplot for my parsimonious Bayesian Poisson model .

6.3 Quantify model fit using a posterior predictive check.

```r
dds[, id := 1:nrow(dds)]
ddsFM <- dds
esti_coef <- rstan::summary(m2_fit)$summary
log_lambdaest <- esti_coef[grepl('log_lambda', rownames(esti_coef)),][,1]
ddsFM <- merge(ddsFM, log_lambdaest)
log_poi_fixed_pp <- ddsFM[,
  {
    lambda <- exp(log_lambdaest)
    pred_y <- rpois(10000,lambda)
    pred_y_stat <- quantile(pred_y, p=c(0.5,0.025,0.975), type=1)
    list(PP_DEATHS_STAT = pred_y_stat, PP_DEATHS_TYPE=c('PP_M','PP_CL','PP_CU'))
  },
  by=c('gender','month_id','age_group')]

log_poi_fixed_pp2 <- dcast.data.table(log_poi_fixed_pp, month_id+gender+age_group~PP_DEATHS_TYPE, value.var='PP_DEATHS_STAT')

# merge observations
tmp2 <- subset(ddsFM, select = c('month_id','gender','deaths', 'age_group', 'id'))
log_poi_fixed_pp3 <- merge(log_poi_fixed_pp2, tmp2, by = c('month_id','gender','age_group'))
set(log_poi_fixed_pp3, which(log_poi_fixed_pp3$gender == 'Women'),
'id', log_poi_fixed_pp3[ which(log_poi_fixed_pp3$gender == 'Women'),]$id - 259 +1)

p <- ggplot(log_poi_fixed_pp3, aes(x=month_id)) +
        geom_point(aes(y=PP_M)) +
        geom_point(aes(y=deaths), colour='red') +
        geom_errorbar(aes(ymin=PP_CL, ymax=PP_CU)) +
        theme_bw()  +
        labs(x='Individuals', y='posterior predictive vs actual deaths\n') +
        facet_grid(gender~age_group)
ggsave(file=file.path(out.dir, 'posterior_pred2.png'), p, w=8, h=6)
include_graphics(file.path(out.dir, 'posterior_pred2.png'))

# posterior predictive check
tmp3 <- log_poi_fixed_pp3[,length(which(deaths <= PP_CU & deaths >= PP_CL)) / length(deaths)]
tmp3
#> 0.9034749
```

By posterior predictive check, we see that there are about 90% of the data in the 95% credible interval for the entire data. Fig 11 shows how the credible interval covers the real data points.

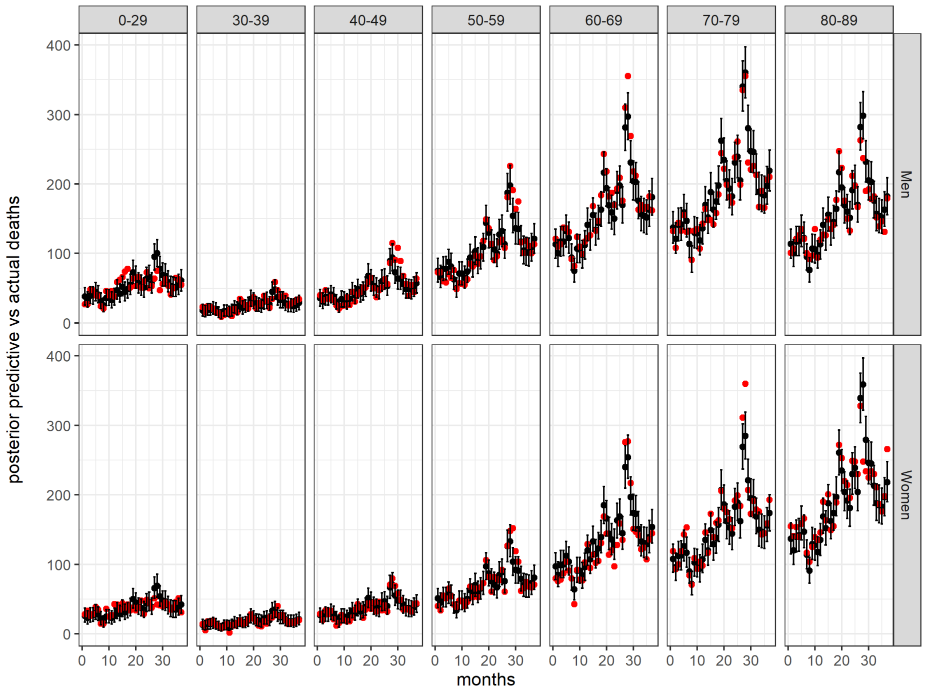

Figure 11: The posterior predictive check plot with month on the x-axis, deaths on the y-axis, facet with gender in row and age group in column.

7 Under-reporting of COVID-19 deaths

For this entire question, we will use the dataset *dfcd* which contains the Covid-19 death, *dd* which contains the death caused by all reasons, and *data* which contains the population for each city, as presented before.
Data For this questions:

~~~
# Total Death
```{r}
file <- file.path(data.dir,'civil_registry_covid_cities_detailed.csv')
dd <- as.data.table(read.csv(file))
dd <- subset(dd, select = -c(state,state_ibge_code,city_ibge_code,created_at))
dd <- melt(dd, id.vars = c('date','city','place','gender','age_group'))
set(dd, dd[,which(is.na(value))],'value',0L)
~~~

```

# summarize daily deaths by city gender age
dd <- dd[, list(deaths = sum(value)), by = c('date','city','gender','age_group')]

# define date as Date object
set(dd, NULL, 'date', dd[, as.Date(date)])

# define months deaths, and month index
dw <- data.table(date = dd[, seq(min(date),max(date),1)])
dw[, month := as.integer(strftime(date, format = '%m'))]
dw[, year := as.integer(strftime(date, format = '%Y'))]
dw[, month_id := (year - 2019) * 12 + month]
tmp <- dw[, list(month_start = min(date)), by = 'month_id']
dw <- merge(dw, tmp, by = 'month_id')
dd <- merge(dd, subset(dw, select = -c(month,year)), by = 'date')

# specify age groups as requested
set(dd, dd[, which(age_group == '9-')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '10-19')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '20-29')], 'age_group', '0-29')
set(dd, dd[, which(age_group == '90-99')], 'age_group', '>=90')
set(dd, dd[, which(age_group == '100+')], 'age_group', '>=90')
set(dd, dd[, which(is.na(age_group))], 'age_group', 'unknown')
dd <- subset(dd, ! age_group %in% c('>=90','unknown'))

# define gender as factor
set(dd, NULL, 'gender', dd[, factor(gender, levels = c('M','F'), labels = c('Men','Women'))])

# count deaths by month gender age
dd <- dd[, list(deaths = sum(deaths)), by = c('month_id','month_start','city','gender','age_group')]

# define age as factor
tmp <- dd[,sort(unique(age_group))]
set(dd, NULL, 'age_group', dd[, factor(age_group, levels = tmp)])
```

#Population Data
```{r}
data <- as.data.table(read.csv('PNADc_2020.csv',encoding = 'latin1'))
age_group_data <- c('0-29', '30-39', '40-49', '50-59', '60-69', '70-79', '80-89')
cut_breaks <- c(-0.1,29,39,49,59,69,79,89)
data <-subset(data, age <90)

data$age <- cut(x = data$age, breaks = cut_breaks,
                  labels = age_group_data,
                  right = TRUE,
)

data <- data[!is.na(data$age),]
data <- data[, .(population = sum(population)),by = c('city','sex','age')]
setnames(data,'age', 'age_group')
setnames(data,'sex','gender')
```
```
```

7.1 Calculate the monthly excess deaths for each gender and age group

```r
dd[, year:= as.integer(substr(month_start, 1,4))]
dd[, month:= as.integer(substr(month_start,6,7))]
dd_base <- subset(dd, year == 2019)
dd_base <- dd_base[, list(deaths_2019 = mean(deaths)), by = c('city','month_id','month','gender', 'age_group')]
set(dd_base, NULL, 'month_id', NULL)
dd <- merge(dd, dd_base, by = c('city','month','gender', 'age_group'))
dd[, exc_deaths := deaths - deaths_2019]
mycity <- dd[which(city == 'Belo Horizonte'),]
dropcol <- c('city')
mycity <- mycity[,!dropcol, with = FALSE]
xtable(head(subset(dd, select = c('month_id','gender','age_group', 'exc_deaths')),10))
```

mycity contains the wanted data. Here we present the first 10 rows of the calculated monthly excess deaths for each gender and age group as follow:

	month_id	gender	age_group	exc_deaths
1	1.00	Men	0-29	0.00
2	13.00	Men	0-29	-6.00
3	25.00	Men	0-29	-2.00
4	37.00	Men	0-29	-4.00
5	1.00	Men	30-39	0.00
6	13.00	Men	30-39	1.00
7	25.00	Men	30-39	1.00
8	37.00	Men	30-39	3.00
9	1.00	Men	40-49	0.00
10	13.00	Men	40-49	-1.00

7.2 Plot

Here we use the posterior estimated data *log_poi_fixed_pp3* from Q6 and make the plot Fig 12:

```r
covidExc <-dfcd
setnames(covidExc, 'deaths','CovidDeaths')
set(covidExc, NULL, 'year',NULL)
ddT <- merge(covidExc,dd, by =c('city', 'gender','age_group','month_id'))
ddT <- merge(ddT, data, by =c('city', 'gender','age_group'))
Mycity <- ddT[which(city == 'Belo Horizonte'),]
Mycity [, ExcDeathPer := 1000 * exc_deaths/population]
# apply the best model
fit_pp <- log_poi_fixed_pp3
Mycity2 <- merge(fit_pp, Mycity, by = c('gender','age_group','month_id','deaths'))
Mycity2[, UpExcDeathPer := 1000 *(PP_CU- deaths_2019)/population]
Mycity2[, LwExcDeathPer := 1000 *(PP_CL- deaths_2019)/population]
Mycity2[, MExcDeathPer := 1000 *(PP_M- deaths_2019)/population]
MycityCum <- Mycity2[, .(CovidCumDeath= cumsum(CovidDeaths),
                     CumExcDeathPer = cumsum(ExcDeathPer),
                     CumUpExcDeathPer = cumsum(UpExcDeathPer),
                     CumLwExcDeathPer = cumsum(LwExcDeathPer),
                     CumMExcDeathPer = cumsum(MExcDeathPer),
                month_id = month_id), by = c('gender','age_group')]
p <- ggplot(MycityCum, aes(x =CovidCumDeath)) +
        geom_point(aes(y=CumExcDeathPer, colour=gender)) +
        geom_point(aes(y=CumMExcDeathPer)) +
        geom_errorbar(aes(ymin=CumLwExcDeathPer, ymax=CumUpExcDeathPer))+ theme_bw()+
        labs(x='CovidCumDeath', y='Monthly CumDeath')
ggsave(file.path(out.dir, "MycityCum.png"),p,w =10,h = 5)
include_graphics(file.path(out.dir, "MycityCum.png"))
```

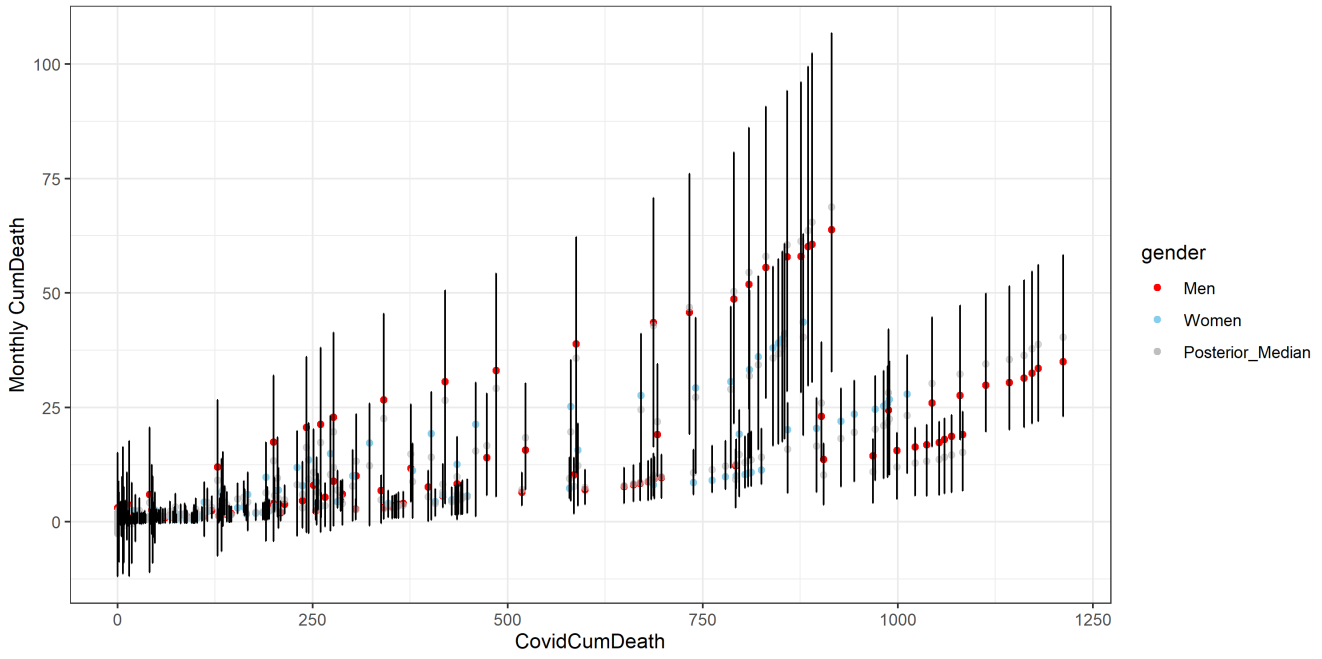

Figure 12: The cumulating monthly excess deaths per 1,000 population in each gender and age group against the corresponding cumulating monthly reported COVID-19 deaths with posterior medians and 95% credible intervals for the estimated cumulating monthly excess deaths per 1,000 population. Red points represent the real cumulating monthly excess deaths per 1,000 population for men and the blue points represent that for women and the half-transparent black points are the posterior medians

7.3 Calculate the difference in total excess deaths and the reported COVID-19 deaths by the end of 2021 in the city that you are assigned. Report posterior median estimates along with 95% posterior credible intervals for both men and women in a table.

```
Mycity3 <- Mycity2[, .(CovidDeaths = sum(CovidDeaths) ,
                        exc_deaths = sum(exc_deaths)), by = c('year')]
Mycity3[,DIF := exc_deaths-CovidDeaths]
sum(Mycity3[which(year <= 2021),'DIF'])
# 7469
```

	year	CovidDeaths	exc_deaths	DIF
1	2019	0	0.00	0.00
2	2020	2324	5706.00	3382.00
3	2021	5485	9572.00	4087.00
4	2022	160	605.00	445.00

Table 4: The Data in *Mycity3*

Thus, by the end of 2021 in the city Belo Horizontewe, the total difference is 4087.

```
Mycity4 <- Mycity2[,.(PP_CL = sum(PP_CL),
                      PP_CU = sum(PP_CU),
                      PP_M = sum(PP_M)), by = c('gender')]
```

The posterior median estimates along with 95% posterior credible intervals for both men and women is shown in the following table 5

	gender	Lower Bound	Upper Bound	Median
1	Men	22489	32294	27223
2	Women	19222	28170	23522

Table 5: The posterior median estimates along with 95% posterior credible intervals for both men and women

8 Evaluating prediction accuracy

8.1

Add prediction and log like to the best model and get the model fit. (Inherit the stan data from Q6)

```
poi_regression3_txt <- "
data{
    int<lower=1> N;
    int<lower=1> K;
    int<lower=0> y[N];
    vector[N] offset;
    matrix[N,K] X;
}
parameters{
    real beta0;
    vector[K] beta;
}
transformed parameters{
    vector[N] log_lambda;
    log_lambda = beta0 + X * beta + offset;
}
model{
    beta0 ~ normal( 0 , 10 );
    beta ~ normal( 0 , 1 );
    y ~ poisson_log( log_lambda );
}

generated quantities {
  int ypred[N];
  real log_lik[N];

  ypred = poisson_log_rng(beta0 + X * beta + offset);
  for(i in 1:N)
  {
    log_lik[i] = poisson_log_lpmf(y[i] |  beta0 + X[i,] * beta + offset);
  }
}
"
if (!file.exists(file.path(out.dir, paste0('poi_deaths_fit3_',city_of_student_ascii,'.rds'))))
{
  poi_regression3_compiled <- rstan::stan_model(
    model_name = 'poi_regression3',
```

```r
    model_code = gsub('\t',' ',poi_regression3_txt)
    )
  #m1_init_beta0 <- subset(dds, age_group_id == 1 & month_id == 1)[, log(deaths/population)]
  m3_fit <- rstan::sampling(poi_regression3_compiled,
    data = stan_data,
    warmup = 5e2, iter = 5e3, chains = 4,
    init = list(list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K))
                )
  )
  saveRDS(m3_fit, file = file.path(out.dir, paste0('poi_deaths_fit3_',city_of_student_ascii,'.rds')))
}
if (file.exists(file.path(out.dir, paste0('poi_deaths_fit3_',city_of_student_ascii,'.rds'))))
{
  m3_fit <- readRDS( file.path(out.dir, paste0('poi_deaths_fit3_',city_of_student_ascii,'.rds')) )
}
```

MEAN ABSOLUTE ERROR and LOG POINTWISE PREDICTIVE DENSITY code::

```r
est_para <- rstan::summary(m3_fit)$summary

log_like_est <- est_para[grepl('log_lik',rownames(est_para)),][,1]
ypred_est <- est_para[grepl('ypred',rownames(est_para)),][,1]
Mycity5 <- cbind(Mycity2,log_like_est,ypred_est)

Mycity5[, Ae:= abs(deaths-ypred_est)]
#Report
Mae <-  Mycity5[,c('Ae'),with = FALSE] %>% as.vector()
Mae_val <- sum(Mae)/518
#63.60698

# Expected log pointwise predictive density of the 518 all-cause death values
exppd <- sum(log_like_est)
#-208236663

# Expected log pointwise predictive density of the all-cause death values that correspond to the last month in 2021.
exppd_36 <- Mycity5[which(month_id ==36), c('log_like_est'), with = FALSE]
exppd_36 <- sum(exppd_36)
# -1075481
```

Denote by z_i a new data point, and by z_i^* the corresponding, predicted pointwise estimate under a fitted model. Then the MEAN ABSOLUTE ERROR is

$$MAE = \frac{1}{n^*} \sum_{i=1}^{n^*} (|z_i - z_i^*|)$$

Then we calculate and report the mean absolute error between the actual 518 all-cause death values and the corresponding posterior median estimates: (Here we take z_i to be all-cause death and z_i^* to be the corresponding posterior median estimates) The mean absolute error between the actual 518 all-cause death values and the corresponding posterior median estimate is 63.60698.
Then we apply the LOG POINTWISE PREDICTIVE DENSITY which is numerically estimated with

$$\text{lppd.est } = \frac{1}{S} \sum_{s=1}^{S} \sum_{i=1}^{n^*} \log p\left(z_i \mid \theta^s\right)$$

The expected log point-wise predictive density of the 518 all-cause death values is -208236663 and the expected log point-wise predictive density of the all-cause death values that correspond to the last month in 2021 is -1075481.

8.2

```r
# calculate approximate LOO cross validation for model without variable selection
m3_fit_loo <- loo::loo(m3_fit)
m3_fit_loo
# Computed from 18000 by 518 log-likelihood matrix
#
# ------
# Monte Carlo SE of elpd_loo is NA.
#
# Pareto k diagnostic values:
#                          Count Pct.    Min. n_eff
# (-Inf, 0.5]   (good)       0     0.0%  <NA>
#  (0.5, 0.7]   (ok)         0     0.0%  <NA>
#    (0.7, 1]   (bad)        0     0.0%  <NA>
#    (1, Inf)   (very bad) 518   100.0%  0
# See help('pareto-k-diagnostic') for details.
m3_fit_loo$elpd_loo
# [1]  -43025176
```

We see that the leave-one-out expected log point-wise predictive density is -43025176.
This numeric approximation is non-stable and high Pareto k diagnostic values. Vehtari (2016) suggest to refit the model to y_{-1} when Pareto k parameter ¿ 0.7 for some y_i. Re-fit your model Command::

```r
dpop <- data
#data populaiton & deaths
dds <- merge(dd,dpop, by = c("city",'age_group', 'gender'))
city_of_student_ascii ='Belo Horizonte'
dds <- dds[which(city == 'Belo Horizonte',month_id =='36' ),]

design_matrix_X <- model.matrix(~ -1 + age_group+ gender, data = dds)
design_matrix_X <- design_matrix_X[,-1]

stan_data <- list()
stan_data$N <- nrow(dds)
stan_data$K <- ncol(design_matrix_X )
stan_data$y <- dds$deaths
stan_data$offset <- log(dds$population)
stan_data$X <- design_matrix_X

m4_fit <- rstan::sampling(poi_regression3_compiled,
    data = stan_data,
    warmup = 5e2, iter = 5e3, chains = 4,
    init = list(list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K))
                )
  )
saveRDS(m4_fit, file = file.path(out.dir, paste0('poi_deaths_fit4_',city_of_student_ascii,'.rds')))
if (file.exists(file.path(out.dir, paste0('poi_deaths_fit4_',city_of_student_ascii,'.rds'))))
{
  m4_fit <- readRDS( file.path(out.dir, paste0('poi_deaths_fit4_',city_of_student_ascii,'.rds')) )
}

est_para <- rstan::summary(m4_fit_loo)
log_like_est <- est_para[grepl('log_lik',rownames(est_para)),][,1]
ypred_est <- est_para[grepl('ypred',rownames(est_para)),][,1]
```

```
MAE <-  sum(abs(dds$deaths - ypred_est))/518
#34.26034

exp_poster_density <- sum(log_like_est)
#[1] -37045684
```

For the last month in 2021, we have the Mean absolute value 34.26034 and expected log pointwise predictive density of the data -37045684.
Denote by z_i all-cause death, and by z_i^* the corresponding posterior median estimates under a fitted model. Then the MEAN ABSOLUTE ERROR is

$$MAE = \frac{1}{n^*}\sum_{i=1}^{n^*}(|z_i - z_i^*|)$$

Then we calculate and report the mean absolute error between the actual 518 all-cause death values and the corresponding posterior median estimates.
The LOG POINTWISE PREDICTIVE DENSITY (within-sample predictive accuracy) is

$$\text{lppd} = \log\prod_{i=1}^{n^*} p\left(y_i \mid y\right) = \log\prod_{i=1}^{n^*}\int p\left(y_i \mid \theta\right) p(\theta \mid y) d\theta$$

and the expected LOG POINTWISE PREDICTIVE DENSITY is

$$\text{lppd.est} \quad = \frac{1}{S}\sum_{s=1}^{S}\sum_{i=1}^{n^*}\log p\left(z_i \mid \theta^s\right)$$

8.3 Rio de Janeiro

Inherit the stan model *poi_regression3_txt* from question 8.1.

```
dpop <- data
#data populaiton & deaths
dds <- merge(dd,dpop, by = c("city",'age_group', 'gender'))
city_of_student_ascii ='Belo Horizonte'
dds <- dds[which(city == 'Rio de Janeiro'),]

design_matrix_X <- model.matrix(~ -1 + age_group+ gender + as.factor(month_id), data = dds)
design_matrix_X <- design_matrix_X[,-1]

stan_data <- list()
stan_data$N <- nrow(dds)
stan_data$K <- ncol(design_matrix_X )
stan_data$y <- dds$deaths
stan_data$offset <- log(dds$population)
stan_data$X <- design_matrix_X
RJ_fit <- rstan::sampling(poi_regression3_compiled,
    data = stan_data,
    warmup = 5e2, iter = 5e3, chains = 4,
    init = list(list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K)),
                list(beta0 = 0 , beta = rep(0, stan_data$K))
                )
  )

saveRDS(RJ_fit, file.path(out.dir, "RJ_fit.rds"))
```

```r
est_para_RJ <- rstan::summary(RJ_fit)$summary
ypred_RJ <- est_para_RJ[which(grepl('ypred', rownames(est_para_RJ))),]
loglike_RJ <- est_para_RJ[which(grepl('log_lik', rownames(est_para_RJ))),]
RJ_fit_loo <- loo::loo(RJ_fit)


yval <- cbind(ypred_RJ, dds) %>% as.data.table()
yval[,id:= 1:nrow(yval)]
# need cumulating
setkey(yval , 'month_id')
yCumu <- yval[, .(Cumdeaths = cumsum(deaths),
                  CumMean = cumsum(mean),
                  CumCL = cumsum(`2.5%`),
                  CumCU = cumsum(`97.5%`),
                  month_id = month_id), by = .(age_group, gender)]

p <-ggplot(yCumu, aes(x =month_id ))+ geom_point(aes(y = Cumdeaths), col = 'red') +
  geom_point(aes(y = CumMean),alpha = 0.3)+
  geom_errorbar(aes(ymin = CumCL, ymax = CumCU),alpha = 0.5)+
  theme_bw() +
  labs(x='month_id', y='posterior predictive vs actual deaths\n') +
  facet_grid(gender~age_group)+
  coord_cartesian(ylim = c(0,30000),xlim = c(0,37))

ggsave(file.path(out.dir, 'RJdeathCum.png'),p,w=10,h =5)
include_graphics(file.path(out.dir, 'RJdeathCum.png'))


Mae_RJ<- sum(abs(yval$mean - yval$deaths))/nrow(dds)
exp_pos_density <- sum(loglike_RJ[,1])
cat('MAE Value: ', Mae_RJ, "\n Expected Posterior Density: ", exp_pos_density)
# MAE Value:  29.92225
# Expected Posterior Density:  -126470044
```

Fig 13 shows the comparison for the cumulating monthly all-cause death and the predicted cumulating monthly all-cause death and Fig 14 shows the comparison for the monthly all-cause death and the predicted monthly all-cause death.

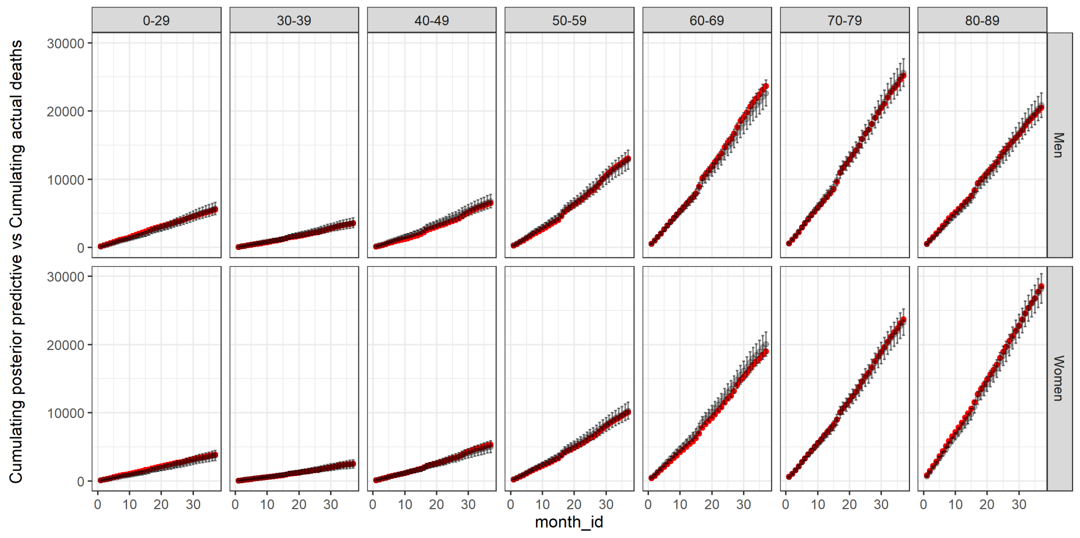

Figure 13: The plot for comparing the predictions to the actual data. The red points represent the actual cumulating monthly all-cause death and the black points represent the predicted cumulating monthly all-cause death. The 95% posterior credible intervals are also included in the graph.

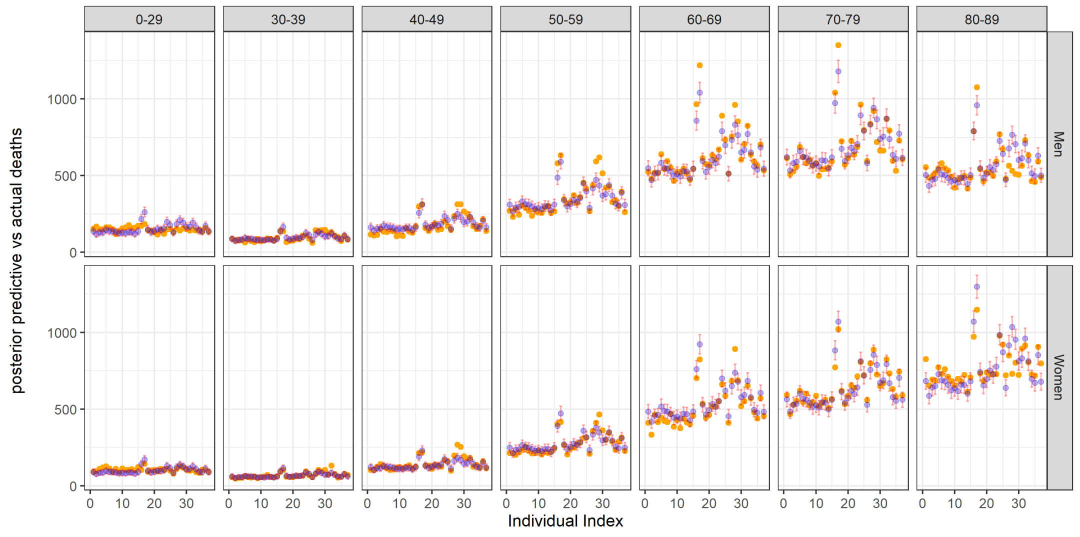

Figure 14: The plot for comparing the predictions to the actual data. The red points represent the actual monthly all-cause death and the blue points represent the predicted monthly all-cause death. The 95% posterior credible intervals are also included in the graph.

The mean absolute error between the actual 518 all-cause death values and the corresponding posterior mean estimates is 29.92225, and the expected log pointwise predictive density is -126470044.

8.4 Comment on the within-sample and out-of-sample prediction accuracies of your best model.

```r
#For My city Belo Horizonte
m4_fit_loo <- loo::loo(m4_fit)
m4_fit_loo$estimates
#expected log pointwise predictive density
m4_fit_loo$estimates[1,1]
# [1] -40406698

```

```
exp_poster_density -m4_fit_loo$estimates[1,1]
#[1] 3361014

#For RJ
pred_density_loo <-RJ_fit_loo$elpd_loo
#[1] -136935950
bay_free_para <- exp_pos_density - pred_density_loo
#[1] 10465906
```

We see that the within-sample accuracy would give smaller prediction accuracy than the out-of-sample accuracy. By having the within-sample and out-of-sample prediction accuracies, we can calculate the BAYESIAN LOO FREE PARAMETERS of a model which is the difference between within-sample accuracy and the out-of-sample accuracy,

$$\text{p.loo.est} \;=\; \text{lppd.est} \;-\; \text{lppd.loo.cv.est}$$

measuring of model complexity, and then make model comparison with other models. For city Belo Horizonte, the BAYESIAN LOO FREE PARAMETERS are 3361014 and for Rio de Janeiro is 10465906. We see the within-sample accuracy for this model highly overestimates the out-of-sample predictive accuracy. The higher *lppd.loo.cv.est*, the higher predictive accuracy the model is.