STA 601 Homework 10c

Lingyun Shao

Dec. 03, 2018

11.4

Hierarchical logistic regression: The Washington Assessment of Student Learning (WASL) is a standardized test given to students in the state of Washington. Letting j index the counties within the state of Washington and i index schools within counties, the file `mathstandard.dat` includes data on the following variables:

$y_{i,j}$ = the indicator that more than half the 10th graders in school i, j passed the WASL math exam;

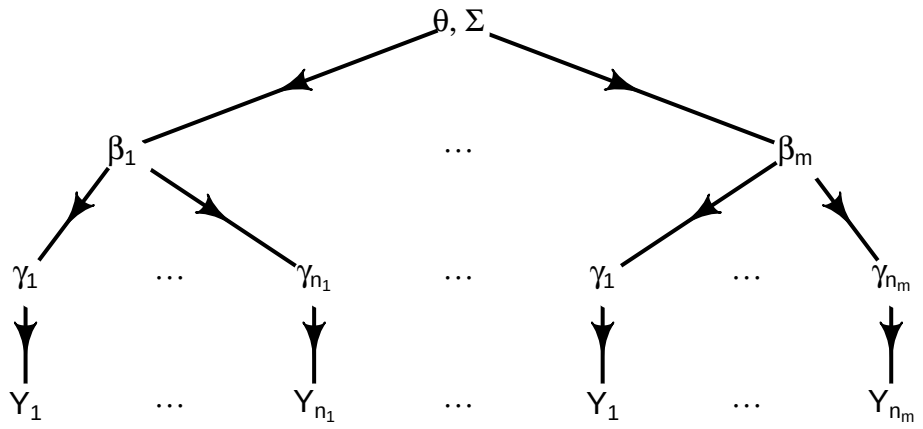
$x_{i,j}$ = the percentage of teachers in school i, j who have a masters degree.

In this exercise we will construct an algorithm to approximate the posterior distribution of the parameters in a generalized linear mixed-effects model for these data. The model is a mixed effects version of logistic regression:

$y_{i,j} \sim binomial(e^{\gamma_{i,j}}/[1 + e^{\gamma_{i,j}}])$, where $\gamma_{i,j} = \beta_{0,j} + \beta_{1,j}x_{i,j}$

$\beta_1, ..., \beta_J \sim i.i.d.$ multivariate normal (θ, Σ), where $\beta_j = (\beta_{0,j}, \beta_{1,j})$

a) The unknown parameters in the model include population-level parameters $\{\theta, \Sigma\}$ and the group-level parameters $\{\beta_1, ..., \beta_m\}$. Draw a diagram that describes the relationships between these parameters, the data $\{y_{i,j}, x_{i,j}, i = 1, ..., n_j, j = 1, ..., m\}$, and prior distributions.



According to the diagram, we can see that our data is connected to the global parameters θ, Σ only through the group parameter β_j and each β_j is connected to our data via the binomial likelihood of logistic regression.

```
d = read.table(file = 'http://www2.stat.duke.edu/~pdh10/FCBS/Exercises/mathstandard.dat',
                    header = TRUE, sep = '\t')
math = data_frame(
  county = d[,1] %>% str_extract('[:alpha:].+[:alpha:]'),
  metstandard = d[,1] %>% str_extract('\\b\\d{1}\\b') %>% as.numeric(),
  percentms = d[,1] %>% str_extract('\\b\\d+\\.\\d+\\b') %>% as.numeric()
)
```

b) Before we do a Bayesian analysis, we will get some ad hoc estimates of these parameters via maximum likelihood: Fit a separate logistic regression model for each group, possibly using the glm command in R via `beta. j <- glm(y.j~X.j,family=binomial)$coef`. Explain any problems you have with obtaining estimates for each county. Plot $\exp\left\{\hat{\beta}_{0,j} + \hat{\beta}_{1,j}x\right\}/(1+\exp\left\{\hat{\beta}_{0,j} + \hat{\beta}_{1,j}x\right\})$ as a function of x for each county and describe what you see. Using maximum likelihood estimates only from those counties with 10 or more schools, obtain ad hoc estimates $\hat{\theta}$ and $\hat{\Sigma}$ of θ and Σ. Note that these estimates may not be representative of patterns from schools with small sample sizes.

```
counties = math$county %>% unique()
res = NULL
for(ct in counties) {
  math_ct = math %>%
    filter(county == ct)
  m = glm(data = math_ct, metstandard~percentms,family=binomial)
  res = rbind(
    res,
    data_frame(
      county = ct,
      schools = nrow(math_ct),
      b0 = m$coef[1],
      b1 = m$coef[2]
    )
  )
}
```

```
## Warning: package 'bindrcpp' was built under R version 3.3.3
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

## Warning: glm.fit: algorithm did not converge

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

From the warnings message, we can know that there are 2 problems with fitting models:

1. There are several (4) counties that have perfectly separable data, thus making the algorithm not converging. So the estimates are not very reliable.
2. There are several (4) counties with only one sample, resulting in NA in coefficient estimates.

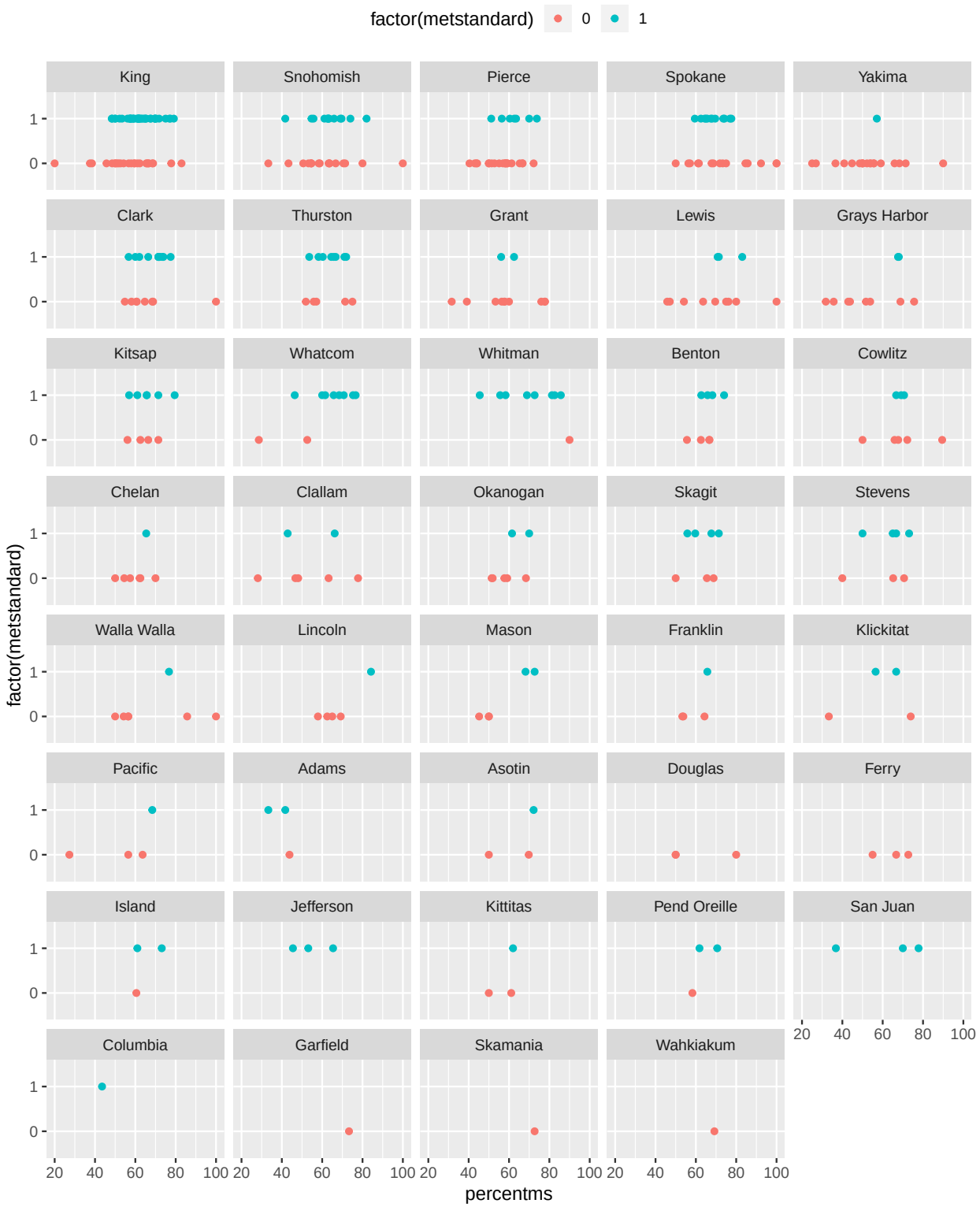
Overall, we are suffering from lack of data. The estimates for groups with small sample sizes may behave weirdly.

```
res = res %>%
  arrange(desc(schools))
res %>%
  kable()
```

county	schools	b0	b1
King	64	-2.8641607	0.0553719
Snohomish	30	-1.2003330	0.0149527
Pierce	27	-7.3545109	0.1064683
Spokane	27	2.3035123	-0.0355588
Yakima	20	-4.0847169	0.0208129
Clark	19	-0.9551942	0.0187739
Thurston	13	-1.5709685	0.0323388
Grant	12	-1.7867982	0.0030048
Lewis	12	-3.5101909	0.0337297
Grays Harbor	10	-8.4978525	0.1174606
Kitsap	10	-3.4675513	0.0592634
Whatcom	10	-8.5641958	0.1818284
Whitman	9	885.2327349	-10.0793869
Benton	8	-17.8895771	0.2735615
Cowlitz	8	-0.3564859	-0.0022385
Chelan	7	-13.8759019	0.1928240
Clallam	7	-1.3131724	0.0073955
Okanogan	7	-14.3877504	0.2177999
Skagit	7	-2.3920152	0.0428273
Stevens	7	-2.2734717	0.0417631
Walla Walla	7	-3.8581685	0.0287833
Lincoln	5	-233.7985949	3.0541587
Mason	5	-153.9108965	2.6008689
Franklin	4	-2094.1646233	32.2185756
Klickitat	4	-2.0628118	0.0356646
Pacific	4	-629.3599478	9.5383416
Adams	3	931.2082644	-21.7845795
Asotin	3	-1376.5980410	19.3891636
Douglas	3	-23.5660685	0.0000000
Ferry	3	-23.5660685	0.0000000
Island	3	-5398.6556245	88.8666444

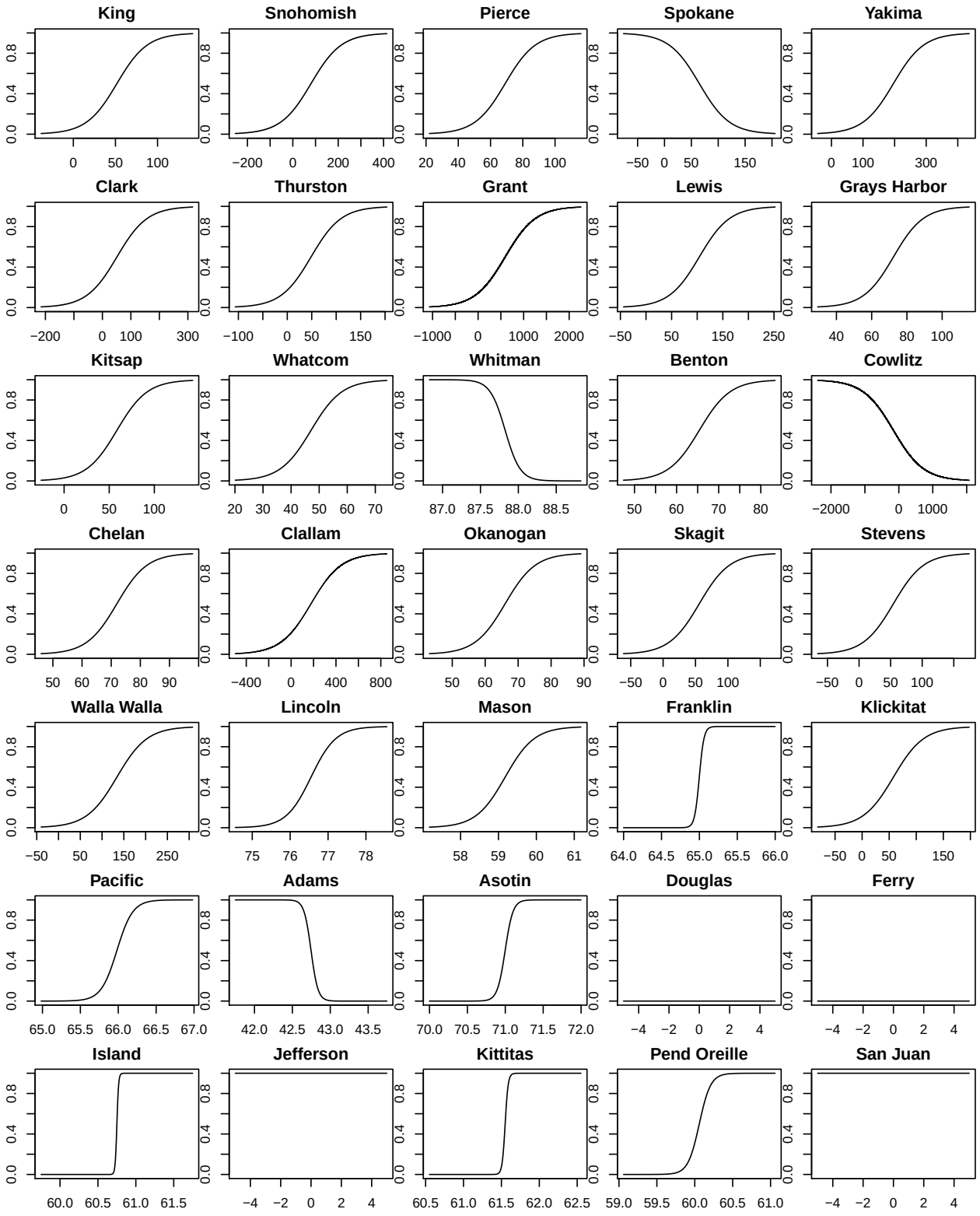
county	schools	b0	b1
Jefferson	3	23.5660689	0.0000000
Kittitas	3	-3110.1242439	50.5304431
Pend Oreille	3	-801.7189011	13.3486687
San Juan	3	23.5660695	0.0000000
Columbia	1	22.5660686	NA
Garfield	1	-22.5660685	NA
Skamania	1	-22.5660685	NA
Wahkiakum	1	-22.5660685	NA

```
lvl = res$county

math %>%
  mutate(county = factor(county, levels = lvl)) %>%
  # filter(county %in% counties[1:10]) %>%
  ggplot(aes(x = percentms, y = factor(metstandard), color = factor(metstandard))) +
  geom_point() +
  facet_wrap(~county, ncol = 5) +
  theme(legend.position = 'top')
```




```
par(mfrow = c(7,5), mar = c(2,1.8,2,0.1))
for(i in 1:(nrow(res)-4)) {
  b0 = res$b0[i]
  b1 = res$b1[i]
```

```
  center = -b0/b1
  if(0.05/abs(b1)>0.01&0.05/abs(b1)<1e5) {
    s = seq(0.1, 100 * round(0.05/abs(b1), 2), 0.1)
  } else {
    s = seq(0.001, 1, 0.001)
  }
  x = c(center-rev(s), center, center+s)
  if(0.05/abs(b1)>1e5) x = seq(-5,5,0.1)
  gamma = b0+b1*x
  pi = exp(gamma)/(1+exp(gamma))
  plot(x, pi, ylim = c(0,1), xlab = '', ylab = '', main = res$county[i], type = 'l')
}
```




```
dt = res %>%
  filter(schools>=10) %>%
  dplyr::select(b0,b1)
(theta.hat = colMeans(dt))
```

```
##          b0          b1
## -3.46274671  0.05070387
```

```
(Sigma.hat = cov(dt))
```

```
##           b0         b1
## b0 10.755821 -0.1793610
## b1 -0.179361  0.0034898
```

c) Formulate a unit information prior distribution for θ and Σ based on the observed data. Specifically, let $\theta \sim$ multivariate normal$(\hat{\theta}, \hat{\Sigma})$ and let Σ^{-1} Wishart$(4, \hat{\Sigma}^{-1})$. Use a Metropolis-Hastings algorithm to approximate the joint posterior distribution of all parameters.

As the diagram in a) shows, the data connect to the global paramters θ, Σ only through β_j, so we have the following form of joint density

$$p(y, \beta_1, ..., \beta_J, \theta, \Sigma | X) = \prod_{j=1}^{J} \{p(y_j|\beta_j, X_j)p(\beta_j|\theta, \Sigma)\}\, p(\theta, \Sigma)$$

We can use different terms in this joint density to derive the full conditional in Gibbs sampler or calculate the proposal acceptance rate in Metropolis Hastings.

Gibbs steps for (θ, Σ)

$$\begin{aligned} p(\theta|\beta, \Sigma) &\propto p(\theta) \prod_{j=1}^{m} p(\beta_j|\theta, \Sigma) \\ &\propto \mathcal{N}((\hat{\Sigma}^{-1} + m\Sigma^{-1})^{-1}(\hat{\Sigma}^{-1}\hat{\theta} + m\Sigma^{-1}\bar{\beta}), (\hat{\Sigma}^{-1} + m\Sigma^{-1})^{-1}) \end{aligned}$$

where $\bar{\beta} = 1/m \sum_{j=1}^{m} \beta_j$

$$\begin{aligned} p(\Sigma|\beta, \theta) &\propto p(\Sigma) \prod_{j=1}^{m} p(\beta_j|\theta, \Sigma) \\ &\propto \mathcal{N}(m + 4, \hat{\Sigma} + S_\beta) \end{aligned}$$

where $S_\beta = \sum_{j=1}^{m} (\beta_j - \theta)(\beta_j - \theta)^T$

Metropolis step for β_j

For β_j, I used Metropolis-Hastings to sample from its posteior.

The Metropolis-Hastings implementation is given as the following

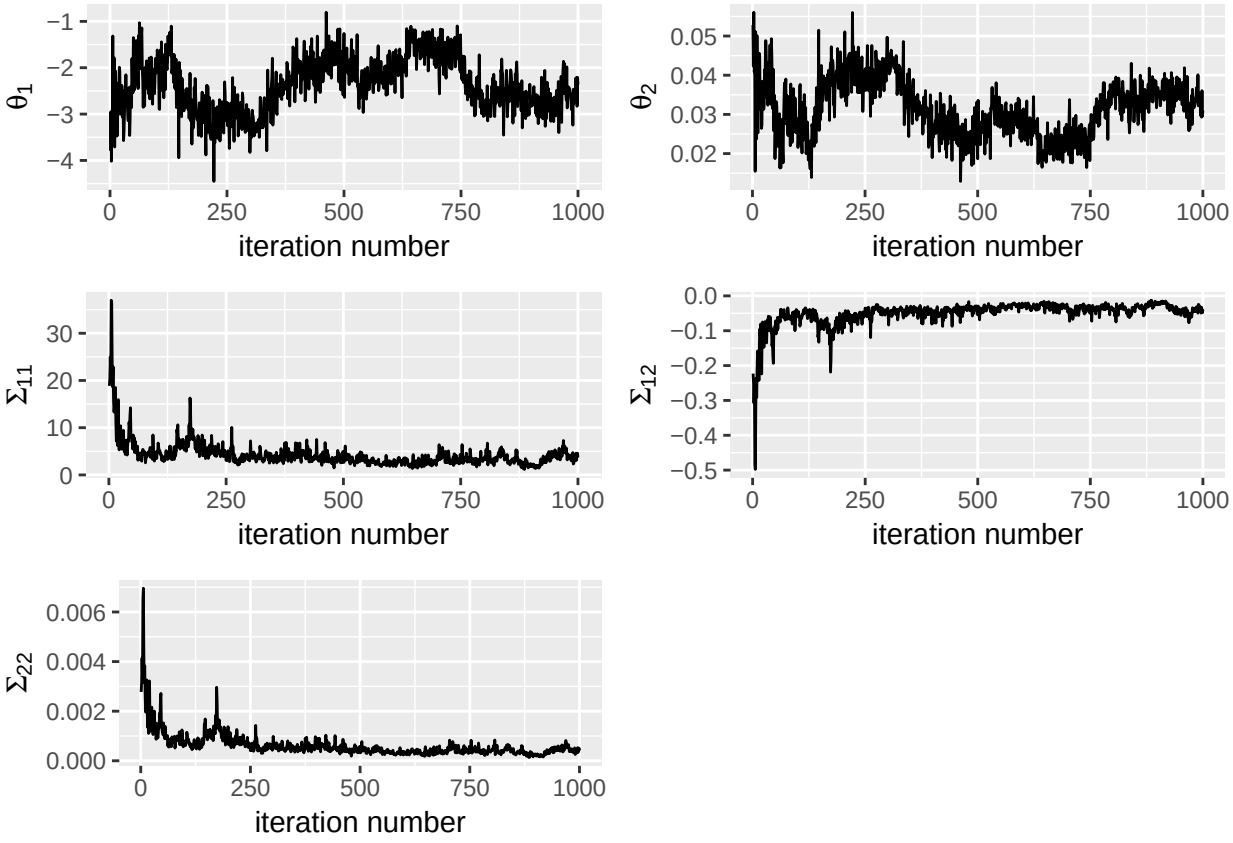
```
m = length(unique(math$county))
theta = theta.hat
Sigma = Sigma.hat
beta = rep(theta.hat, m) %>% matrix(ncol = m) + matrix(rnorm(2 * m), nrow = 2)
lambda = 3
cts = math$county
math_sp = math %>%
  mutate(county = as.factor(county)) %>%
```

```
  split(.$county)


MH = function(theta, Sigma, beta){
  # update theta using Gibbs
  theta_cov = solve(solve(Sigma.hat) + m * solve(Sigma))
  bar_beta = rowMeans(beta)
  theta_mean = theta_cov %*% (solve(Sigma.hat) %*% theta.hat + m * solve(Sigma) %*% bar_beta)
  theta = rmvnorm(1, theta_mean, theta_cov)
  # update Sigma using Gibbs
  S_beta = sweep(beta, 1, theta, "-") %>% {. %*% t(.)}
  Sigma = solve(rWishart(1, m + 4, solve(Sigma + S_beta))[, , 1])
  # update beta using MH
  accep_count = 0
  for (j in 1:m){
    betaj = beta[, j]
    log_prior = - 1 / 2 * (betaj - theta) %*% solve(Sigma) %*% t(betaj - theta)
    gammaj = c((math_sp[[j]] %>% mutate(one = 1) %>% dplyr::select(one, percentms) %>% as.matrix()) %*%
    log_sample = sum(math_sp[[j]]$metstandard * gammaj - log(1 + exp(gammaj)))
    betaj_ = rmvnorm(1, betaj, lambda * Sigma.hat)[1, ]
    log_prior_ = - 1 / 2 * (betaj_ - theta) %*% solve(Sigma) %*% t(betaj_ - theta)
    gammaj_ = c((math_sp[[j]] %>% mutate(one = 1) %>% dplyr::select(one, percentms) %>% as.matrix()) %*
    log_sample_ = sum(math_sp[[j]]$metstandard * gammaj_ - log(1 + exp(gammaj_)))
    log_u = log_prior_ + log_sample_ - log_prior - log_sample
    if (log(runif(1))<log_u) {
      accep_count = accep_count + 1
      beta[, j] = betaj_
    }
  }
  return(list(accep_count = accep_count, theta = theta, Sigma = Sigma, beta = beta))
}

B = 250
N = 1250
# theta_sam = matrix(NA, 2, N %/% 3)
# Sigma_sam = array(NA, c(2, 2, N %/% 3))
# beta_sam = array(NA, c(2, m, N %/% 3))
theta_sam = matrix(NA, 2, N)
Sigma_sam = array(NA, c(2, 2, N))
beta_sam = array(NA, c(2, m, N))
accep_ratio = 0
for (k in 1:N){
  result = MH(theta, Sigma, beta)
  accep_ratio = accep_ratio + result$accep_count
  theta = result$theta
  Sigma = result$Sigma
  beta = result$beta
  theta_sam[, k] = theta
  Sigma_sam[, , k] = Sigma
  beta_sam[, , k] = beta
}
(accep_ratio = accep_ratio / N / m)
```

```
## [1] 0.1106051
```

d) Make plots of the samples of θ and Σ (5 parameters) versus MCMC iteration number. Make sure you run the chain long enough so that your MCMC samples are likely to be a reasonable approximation to the posterior distribution.

```
theta_sam = theta_sam[,(B+1):N]
Sigma_sam = Sigma_sam[,,(B+1):N]
beta_sam = beta_sam[,,(B+1):N]
ggplot_df = data.frame(x = 1:(N-B), p1 = theta_sam[1, ],
                       p2 = theta_sam[2, ], p3 = Sigma_sam[1, 1, ],
                       p4 = Sigma_sam[1, 2, ], p5 = Sigma_sam[2, 2, ])
g1 = ggplot(ggplot_df, aes(x = x, y = p1)) + geom_line() +
  xlab("iteration number") + ylab(expression(theta[1]))
g2 = ggplot(ggplot_df, aes(x = x, y = p2)) + geom_line() +
  xlab("iteration number") + ylab(expression(theta[2]))
g3 = ggplot(ggplot_df, aes(x = x, y = p3)) + geom_line() +
  xlab("iteration number") + ylab(expression(Sigma[11]))
g4 = ggplot(ggplot_df, aes(x = x, y = p4)) + geom_line() +
  xlab("iteration number") + ylab(expression(Sigma[12]))
g5 = ggplot(ggplot_df, aes(x = x, y = p5)) + geom_line() +
  xlab("iteration number") + ylab(expression(Sigma[22]))
grid.arrange(grobs = list(g1, g2, g3, g4, g5), ncol = 2)
```



The traceplot is given as above.

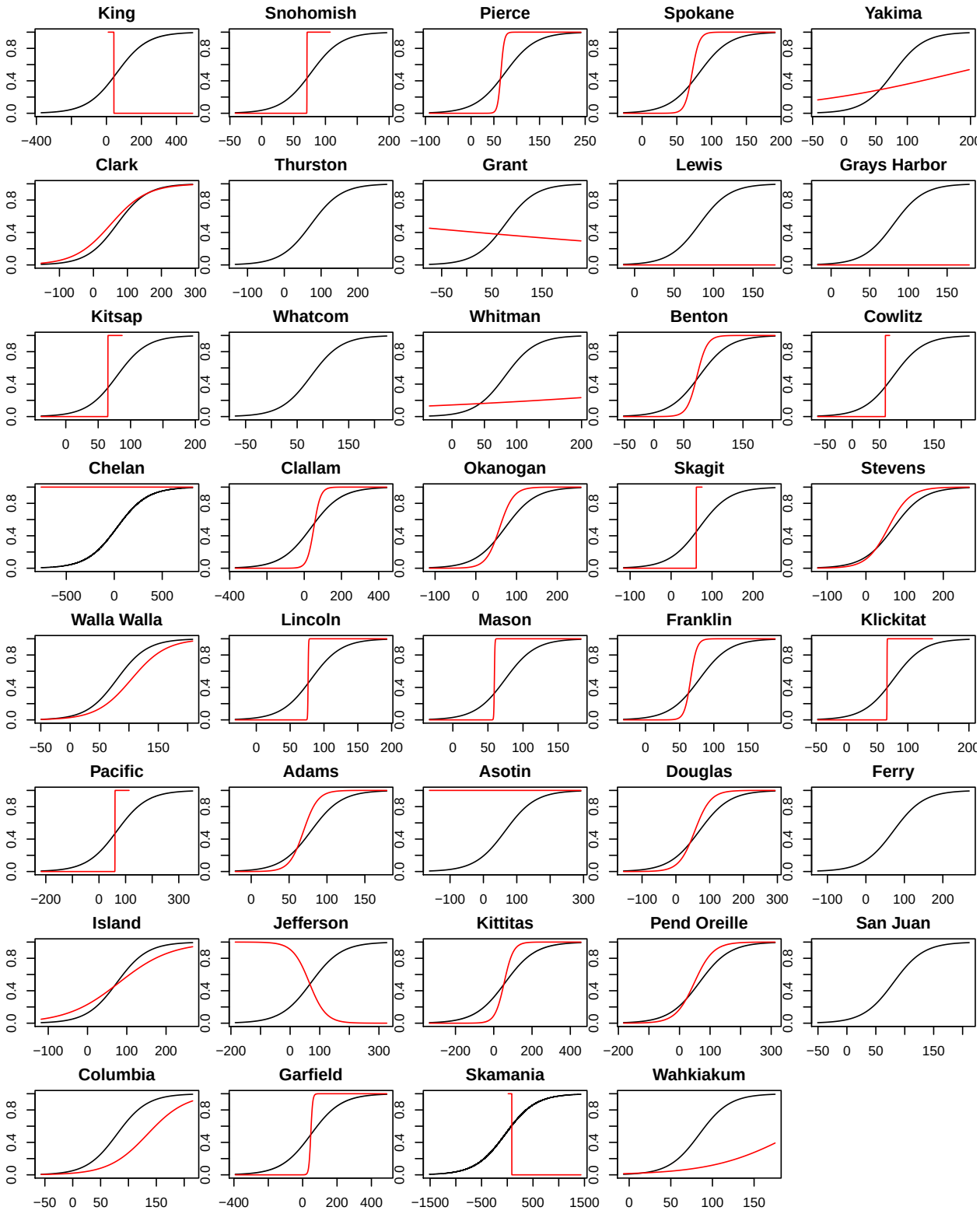
e) Obtain posterior expectations of β_j for each group j, plot $E[\beta_{0,j}|y] + E[\beta_{1,j}|y]x$ as a function of x for each county, compare to the plot in b) and describe why you see any differences between the two sets of regression lines.

```
res2 = sapply(1:m, function(i) {rowMeans(beta_sam[, i, ])}) %>%
t() %>% data.frame()
rownames(res2) = cts%>%unique
colnames(res2) = c("b0", "b1")
kable(res2)
```

	b0	b1
Adams	-0.6922354	0.0115597
Asotin	-3.2632957	0.0421555
Benton	-2.2479257	0.0300899
Chelan	-3.7630228	0.0457320
Clallam	-3.2365882	0.0415064
Clark	-1.5486292	0.0224286
Columbia	-1.7634000	0.0242731
Cowlitz	-2.5315108	0.0330907
Douglas	-4.2862544	0.0522901
Ferry	-4.1286746	0.0512267
Franklin	-3.3776235	0.0428067
Garfield	-2.6103249	0.0339370
Grant	-3.5033599	0.0426981
Grays Harbor	-2.9828707	0.0391448
Island	-2.7261289	0.0360859
Jefferson	-0.1739112	0.0063303
King	-0.4483857	0.0122768
Kitsap	-1.9337574	0.0268654
Kittitas	-1.8595885	0.0269953
Klickitat	-1.8136923	0.0255164
Lewis	-3.0747113	0.0386626
Lincoln	-3.6232411	0.0448008
Mason	-3.4672306	0.0464985
Okanogan	-3.5522585	0.0451056
Pacific	-3.0721336	0.0401671
Pend Oreille	-1.1801771	0.0175309
Pierce	-3.9382901	0.0498789
San Juan	-1.4338830	0.0220037
Skagit	-1.5425288	0.0223015
Skamania	-1.8123271	0.0253253
Snohomish	-1.9356393	0.0258700
Spokane	-1.3782198	0.0195189
Stevens	-0.7529974	0.0126312
Thurston	-1.3043262	0.0201460
Wahkiakum	-3.0775170	0.0382637
Walla Walla	-2.9112260	0.0367360
Whatcom	-0.5519247	0.0113446
Whitman	0.1401427	0.0033955
Yakima	-4.6197420	0.0548535

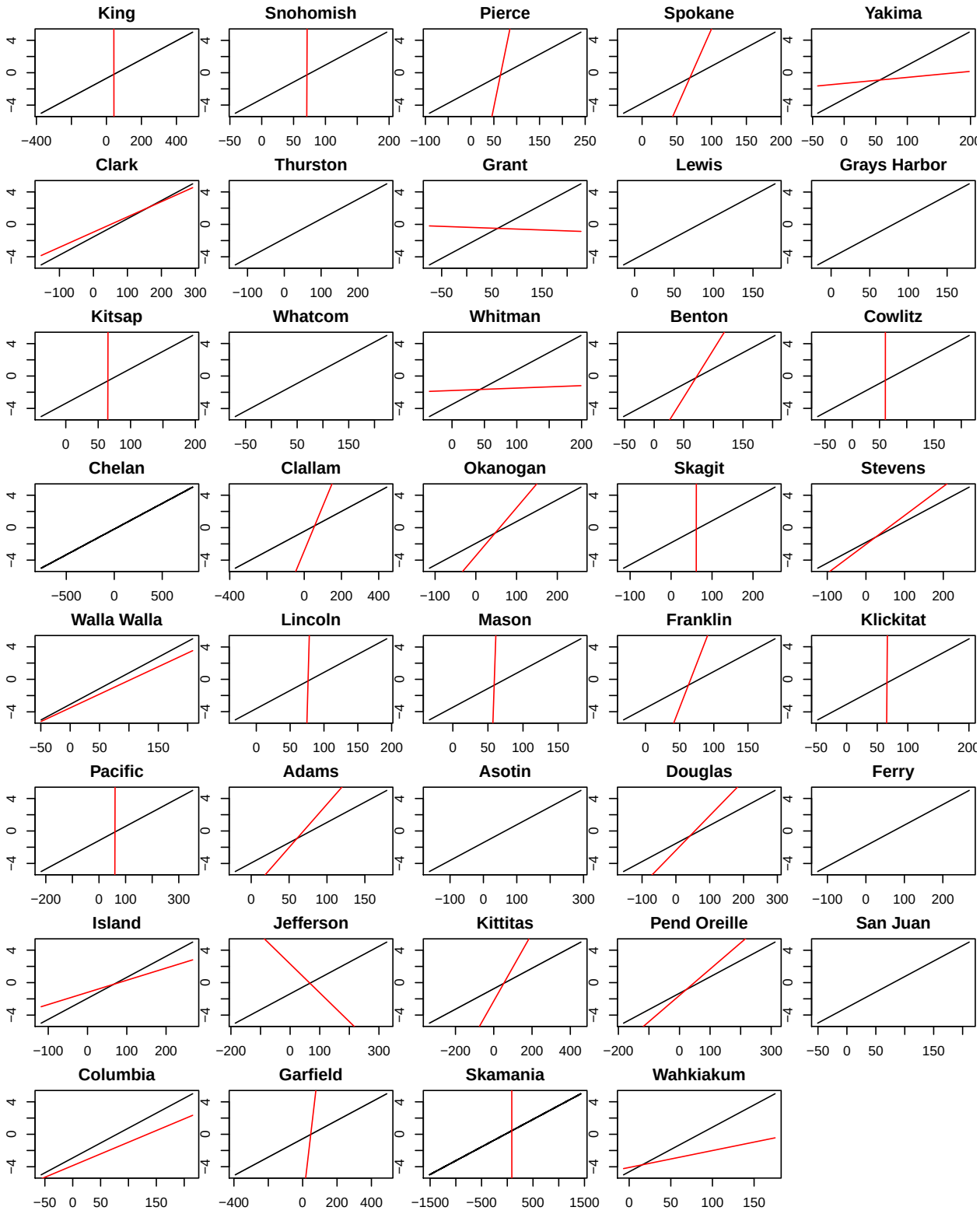
```
par(mfrow = c(8,5), mar = c(2,1.8,2,0.1))
for(i in 1:(nrow(res2))) {
  b0 = res2$b0[i]
  b1 = res2$b1[i]
  b0_ = res[res$county==counties[i],]$b0
```

```
  b1_ = res[res$county==counties[i],]$b1
  center = -b0/b1
  if(0.05/abs(b1)>0.01&0.05/abs(b1)<1e5) {
    s = seq(0.1, 100 * round(0.05/abs(b1), 2), 0.1)
  } else {
    s = seq(0.001, 1, 0.001)
  }
  x = c(center-rev(s), center, center+s)
  if(0.05/abs(b1)>1e5) x = seq(-5,5,0.1)
  gamma = b0+b1*x
  gamma_ = b0_ + b1_ *x
  pi = exp(gamma)/(1+exp(gamma))
  pi_ = exp(gamma_)/(1+exp(gamma_))
  plot(x, pi, ylim = c(0,1), xlab = '', ylab = '', main = res$county[i], type = 'l')
  lines(x, pi_, col =2)
}

par(mfrow = c(8,5), mar = c(2,1.8,2,0.1))
```




```
for(i in 1:(nrow(res2))) {
  b0 = res2$b0[i]
  b1 = res2$b1[i]
  b0_ = res[res$county==counties[i],]$b0
```

```
  b1_ = res[res$county==counties[i],]$b1
  center = -b0/b1
  if(0.05/abs(b1)>0.01&0.05/abs(b1)<1e5) {
    s = seq(0.1, 100 * round(0.05/abs(b1), 2), 0.1)
  } else {
    s = seq(0.001, 1, 0.001)
  }
  x = c(center-rev(s), center, center+s)
  if(0.05/abs(b1)>1e5) x = seq(-5,5,0.1)
  gamma = b0+b1*x
  gamma_ = b0_ + b1_ *x
  plot(x, gamma, xlab = '', ylab = '', main = res$county[i], type = 'l')
  lines(x, gamma_, col =2)
}
```



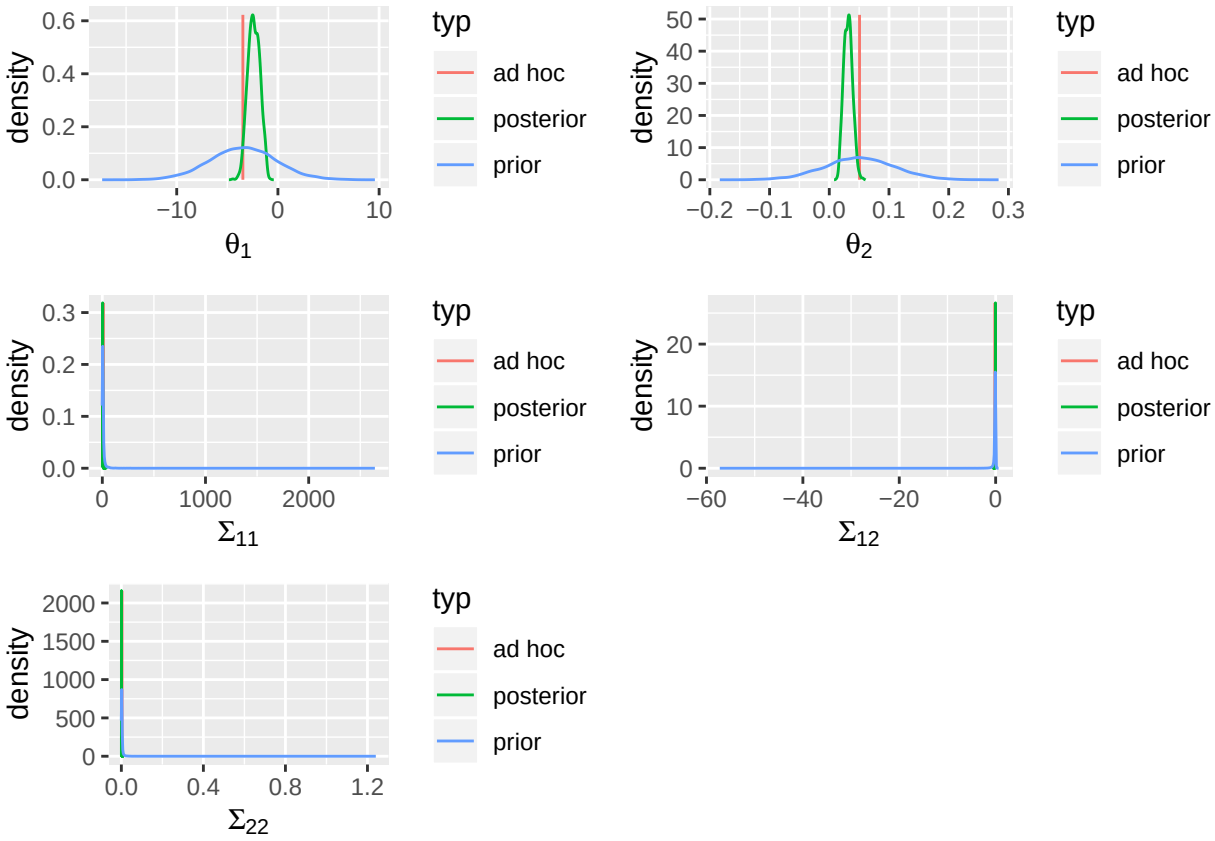
From the plots, we can find that with our proposed model, the regression lines are somewhat pulled towards the global 'regression line' shared by all the counties, accounting for the effect of the presence of global parameter θ, Σ.

And with prior belief, the groups previously suffering from insufficient data now have a decent estimate as

well. The coefficients are shrunk to small values.

f) From your posterior samples, plot marginal posterior and prior densities of θ and the elements of Σ. Include your ad hoc estimates from b) in the plots. Discuss the evidence that the slopes or intercepts vary across groups.

```
plot_all = function(MC_prior, MC_post, ad_hoc, x_lab){
x_seq1 = density(MC_prior)$x
prior_seq = density(MC_prior)$y
x_seq2 = density(MC_post)$x
post_seq = density(MC_post)$y

prior_df = data.frame(x = x_seq1, y = prior_seq, typ = "prior")
post_df = data.frame(x = x_seq2, y = post_seq, typ = "posterior")
adhoc_df = data.frame(x = rep(ad_hoc, 2), y = c(0, max(prior_seq, post_seq)), typ = "ad hoc")
bind_rows(prior_df, post_df, adhoc_df) %>%
mutate(typ = as.factor(typ)) %>%
ggplot(aes(x = x, y = y, col = typ)) +
geom_line() +
xlab(x_lab) +
ylab("density") %>%
return()
}
# use MC for marginal prior
set.seed(0)
MC_theta = rmvnorm(10000, theta.hat, Sigma.hat)
MC_Sigma = rWishart(10000, 4, solve(Sigma.hat)) %>%
{lapply(1:10000, function(i) {solve(.[, , i])})}
MC_theta1 = MC_theta[, 1]
MC_theta2 = MC_theta[, 2]
MC_Sigma11 = MC_Sigma %>% sapply(function(S) {S[1, 1]})
MC_Sigma12 = MC_Sigma %>% sapply(function(S) {S[1, 2]})
MC_Sigma22 = MC_Sigma %>% sapply(function(S) {S[2, 2]})

grid.arrange(
plot_all(MC_theta1, theta_sam[1, ], theta.hat[1], expression(theta[1])),
plot_all(MC_theta2, theta_sam[2, ], theta.hat[2], expression(theta[2])),
plot_all(MC_Sigma11, Sigma_sam[1, 1, ], Sigma.hat[1, 1], expression(Sigma[11])),
plot_all(MC_Sigma12, Sigma_sam[1, 2, ], Sigma.hat[1, 2], expression(Sigma[12])),
plot_all(MC_Sigma22, Sigma_sam[2, 2, ], Sigma.hat[2, 2], expression(Sigma[22])),
ncol = 2
)
```


density
0.6
0.4
0.2
0.0
−10
0
10
θ1
typ
ad hoc
posterior
prior
density
50
40
30
20
10
0
−0.2
−0.1
0.0
0.1
0.2
0.3
θ2
typ
ad hoc
posterior
prior
density
0.3
0.2
0.1
0.0
0
1000
2000
Σ11
typ
ad hoc
posterior
prior
density
20
10
0
−60
−40
−20
0
Σ12
typ
ad hoc
posterior
prior
density
2000
1500
1000
500
0
0.0
0.4
0.8
1.2
Σ22
typ
ad hoc
posterior
prior