Lab9

AUTHOR
Chenxi Liu 1010615050

PUBLISHED
March 14, 2024

Lip cancer

Here is the lip cancer data that was used in the lecture.

- `aff.i` is proportion of male population working outside in each region
- `observe.i` is observed deaths in each region
- `expect.i` is expected deaths, based on region-specific age distribution and national-level age-specific mortality rates.

```
observe.i <- c(
  5,13,18,5,10,18,29,10,15,22,4,11,10,22,13,14,17,21,25,6,11,21,13,5,19,18,14,17,3,10,
  7,3,12,11,6,16,13,6,9,10,4,9,11,12,23,18,12,7,13,12,12,13,6,14,7,18,13,9,6,8,7,6,16,
  17,5,7,2,9,7,6,12,13,17,5,5,6,12,10,16,10,16,15,18,6,12,6,8,33,15,14,18,25,14,2,73,1
  12,10,3,11,3,11,13,11,13,10,5,18,10,23,5,9,2,11,9,11,6,11,5,19,15,4,8,9,6,4,4,2,12,1
  8,12,11,23,7,16,46,9,18,12,13,14,14,3,9,15,6,13,13,12,8,11,5,9,8,22,9,2,10,6,10,12,9
  9,11,11,0,9,3,11,11,11,5,4,8,9,30,110)
expect.i <- c(
    6.17,8.44,7.23,5.62,4.18,29.35,11.79,12.35,7.28,9.40,3.77,3.41,8.70,9.57,8.18,4.35
    4.91,10.66,16.99,2.94,3.07,5.50,6.47,4.85,9.85,6.95,5.74,5.70,2.22,3.46,4.40,4.05,
    16.99,6.19,5.56,11.69,4.69,6.25,10.84,8.40,13.19,9.25,16.98,8.39,2.86,9.70,12.12,1
    10.34,5.09,3.29,17.19,5.42,11.39,8.33,4.97,7.14,6.74,17.01,5.80,4.84,12.00,4.50,4.
    6.42,5.26,4.59,11.86,4.05,5.48,13.13,8.72,2.87,2.13,4.48,5.85,6.67,6.11,5.78,12.31
    2.52,6.22,14.29,5.71,37.93,7.81,9.86,11.61,18.52,12.28,5.41,61.96,8.55,12.07,4.29,
    12.90,4.76,5.56,11.11,4.76,10.48,13.13,12.94,14.61,9.26,6.94,16.82,33.49,20.91,5.3
    12.94,16.07,8.87,7.79,14.60,5.10,24.42,17.78,4.04,7.84,9.89,8.45,5.06,4.49,6.25,9.
    9.57,5.83,9.21,9.64,9.09,12.94,17.42,10.29,7.14,92.50,14.29,15.61,6.00,8.55,15.22,
    18.37,13.16,7.69,14.61,15.85,12.77,7.41,14.86,6.94,5.66,9.88,102.16,7.63,5.13,7.58
    18.75,12.33,5.88,64.64,8.62,12.09,11.11,14.10,10.48,7.00,10.23,6.82,15.71,9.65,8.5
    12.31,8.91,50.10,288.00)
aff.i <- c(0.2415,0.2309,0.3999,0.2977,0.3264,0.3346,0.4150,0.4202,0.1023,0.1752,
        0.2548,0.3248,0.2287,0.2520,0.2058,0.2785,0.2528,0.1847,0.3736,0.2411,
        0.3700,0.2997,0.2883,0.2427,0.3782,0.1865,0.2633,0.2978,0.3541,0.4176,
        0.2910,0.3431,0.1168,0.2195,0.2911,0.4297,0.2119,0.2698,0.0874,0.3204,
        0.1839,0.1796,0.2471,0.2016,0.1560,0.3162,0.0732,0.1490,0.2283,0.1187,
        0.3500,0.2915,0.1339,0.0995,0.2355,0.2392,0.0877,0.3571,0.1014,0.0363,
        0.1665,0.1226,0.2186,0.1279,0.0842,0.0733,0.0377,0.2216,0.3062,0.0310,
        0.0755,0.0583,0.2546,0.2933,0.1682,0.2518,0.1971,0.1473,0.2311,0.2471,
        0.3063,0.1526,0.1487,0.3537,0.2753,0.0849,0.1013,0.1622,0.1267,0.2376,
        0.0737,0.2755,0.0152,0.1415,0.1344,0.1058,0.0545,0.1047,0.1335,0.3134,
        0.1326,0.1222,0.1992,0.0620,0.1313,0.0848,0.2687,0.1396,0.1234,0.0997,
        0.0694,0.1022,0.0779,0.0253,0.1012,0.0999,0.0828,0.2950,0.0778,0.1388,
        0.2449,0.0978,0.1144,0.1038,0.1613,0.1921,0.2714,0.1467,0.1783,0.1790,
        0.1482,0.1383,0.0805,0.0619,0.1934,0.1315,0.1050,0.0702,0.1002,0.1445,
        0.0353,0.0400,0.1385,0.0491,0.0520,0.0640,0.1017,0.0837,0.1462,0.0958,
        0.0745,0.2942,0.2278,0.1347,0.0907,0.1238,0.1773,0.0623,0.0742,0.1003,
        0.0590,0.0719,0.0652,0.1687,0.1199,0.1768,0.1638,0.1360,0.0832,0.2174,
```

```
0.1662,0.2023,0.1319,0.0526,0.0287,0.0405,0.1616,0.0730,0.1005,0.0743,
0.0577,0.0481,0.1002,0.0433,0.0838,0.1124,0.2265,0.0436,0.1402,0.0313,
0.0359,0.0696,0.0618,0.0932,0.0097)
```

Question 1

Explain a bit more what the `expect.i` variable is. For example, if a particular area has an expected deaths of 16, what does this mean?

Answer1:

The expect.i variable represents the expected number of deaths due to lip cancer in each region, based on region-specific age distributions and national-level age-specific mortality rates. When a particular area has an expected death count of 16, it means that, given the age distribution of the male population in that area and the national age-specific mortality rates for lip cancer, we would expect 16 deaths to occur in that area due to lip cancer under average conditions.

Question 2

Run four different models in Stan with three different set-ups for estimating θ_i, that is the relative risk of lip cancer in each region:

1. Intercept α_i is same in each region $= \alpha$
2. Intercept α_i is different in each region and modeled separately
3. Intercept α_i is different in each region and the intercept is modeled hierarchically

Note in all three cases, use the proportion of male population working outside in each region as a covariate.

1.Intercept α_i is same in each region $= \alpha$

```
library(rstan)
```

```
Loading required package: StanHeaders
```

```
rstan version 2.32.5 (Stan version 2.32.2)

For execution on a local, multicore CPU with excess RAM we recommend calling
options(mc.cores = parallel::detectCores()).
To avoid recompilation of unchanged Stan programs, we recommend calling
rstan_options(auto_write = TRUE)
For within-chain threading using `reduce_sum()` or `map_rect()` Stan functions,
change `threads_per_chain` option:
rstan_options(threads_per_chain = 1)
```

```
common_intercept_model_code  <- "
data {
  int<lower=0> N;  // number of regions
```

```
  vector[N] x;  // covariate: proportion of male population working outside
  vector[N] offset;  // log of expected deaths
  int<lower=0> deaths[N];  // observed deaths
}
parameters {
  real alpha;  // common intercept
  real beta;  // covariate coefficient
}
model {
  vector[N] log_lambda = alpha + beta * x + offset;  // linear predictor

  // Priors
  alpha ~ normal(0, 1);
  beta ~ normal(0, 1);

  // Likelihood
  deaths ~ poisson_log(log_lambda);
}
generated quantities {
  vector[N] log_lik;  // Log-likelihood for each observation
  vector[N] log_theta;  // Log of estimated theta for each region
  for (i in 1:N) {
    log_lik[i] = poisson_log_lpmf(deaths[i] | alpha + beta * x[i] + offset[i]);
    log_theta[i] = alpha + beta * x[i];  // Store the log of the rate parameter
  }
}
"
```

```
N <- length(observe.i)  # Number of regions

# Centering aff.i by subtracting the mean
aff_centered <- aff.i - mean(aff.i)

# Create a data list for Stan
lip_cancer_data <- list(
  N = N,
  x = aff_centered,
  offset = log(expect.i),
  deaths = observe.i
)

# Compile the model
common_intercept_model <- stan(
  model_code = common_intercept_model_code,
  data = lip_cancer_data,
  chains = 4,
  iter = 1000,
)
```

```
Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
```

```
using SDK: ‘MacOSX13.3.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rcpp/include/"
-I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/unsupported"  -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/BH/include" -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/src/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppParallel/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/rstan/include"
-DEIGEN_NO_DEBUG  -DBOOST_DISABLE_ASSERTS  -DBOOST_PENDING_INTEGER_LOG2_HPP  -
DSTAN_THREADS  -DUSE_STANC3 -DSTRICT_R_HEADERS  -DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION
-D_HAS_AUTO_PTR_ETC=0  -include '/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp'  -
D_REENTRANT -DRCPP_PARALLEL_USE_TBB=1   -I/opt/R/arm64/include    -fPIC  -falign-
functions=64 -Wall -g -O2  -c foo.c -o foo.o
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Dense:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Core:88:
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error:
unknown type name 'namespace'
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error:
expected ';' after top level declarator
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Dense:1:
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex'
file not found
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
```

```
Chain 1: Gradient evaluation took 1.9e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.19
seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.021 seconds (Warm-up)
Chain 1:                0.014 seconds (Sampling)
Chain 1:                0.035 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 5e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.05
seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.02 seconds (Warm-up)
Chain 2:                0.016 seconds (Sampling)
Chain 2:                0.036 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 5e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.05
```

```
seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.021 seconds (Warm-up)
Chain 3:                0.015 seconds (Sampling)
Chain 3:                0.036 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 5e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.05
seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.02 seconds (Warm-up)
Chain 4:                0.018 seconds (Sampling)
Chain 4:                0.038 seconds (Total)
Chain 4:
```

```r
# Checking for convergence using Rhat
common_intercept_summary <- summary(common_intercept_model)
print(common_intercept_summary$summary[c("alpha", "beta"), ])
```

```
              mean      se_mean         sd        2.5%         25%         50%
alpha -0.008926459 0.0004625749 0.02027129 -0.04905644 -0.02245403 -0.00893546
beta   2.430647569 0.0036919220 0.17179817  2.10098061  2.31294173  2.42923424
              75%      97.5%    n_eff     Rhat
alpha 0.004870043 0.03105185 1920.431 1.000210
beta  2.544746418 2.77997902 2165.370 1.000375
```

2. Intercept α_i is different in each region and modeled separately

```
different_intercept_model_code  <- "
data {
  int<lower=0> N;  // number of regions
  vector[N] x;  // covariate
  vector[N] offset;  // log of expected deaths
  int<lower=0> deaths[N];  // observed deaths
}
parameters {
  vector[N] alpha;  // separate intercepts for each region
  real beta;  // covariate coefficient
}
model {
  vector[N] log_lambda = alpha + beta * x + offset;  // linear predictor

  // Priors
  alpha ~ normal(0, 1);
  beta ~ normal(0, 1);

  // Likelihood
  deaths ~ poisson_log(log_lambda);
}
generated quantities {
  vector[N] log_lik;  // log-likelihood for each observation
  vector[N] log_theta;  // log of estimated theta for each region

  for (i in 1:N) {
    log_lik[i] = poisson_log_lpmf(deaths[i] | alpha[i] + beta * x[i] + offset[i]);
    // The log_theta is the linear predictor without the offset
    log_theta[i] = alpha[i] + beta * x[i];
  }
}
"
```

```
# Compile the model
different_intercept_model <- stan(
  model_code = different_intercept_model_code,
  data = lip_cancer_data,
  chains = 4,
  iter = 1000,
)
```

```
Trying to compile a simple C file
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
using SDK: ‘MacOSX13.3.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rcpp/include/"
-I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/unsupported"  -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/BH/include" -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/src/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppParallel/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/rstan/include"
-DEIGEN_NO_DEBUG  -DBOOST_DISABLE_ASSERTS  -DBOOST_PENDING_INTEGER_LOG2_HPP  -
DSTAN_THREADS  -DUSE_STANC3 -DSTRICT_R_HEADERS  -DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION
-D_HAS_AUTO_PTR_ETC=0  -include '/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp'  -
D_REENTRANT -DRCPP_PARALLEL_USE_TBB=1   -I/opt/R/arm64/include    -fPIC  -falign-
functions=64 -Wall -g -O2  -c foo.c -o foo.o
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Dense:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Core:88:
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error:
unknown type name 'namespace'
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error:
expected ';' after top level declarator
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Dense:1:
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex'
file not found
#include <complex>
         ^~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 3.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.32
seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.08 seconds (Warm-up)
Chain 1:                0.069 seconds (Sampling)
Chain 1:                0.149 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 8e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.08
seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.074 seconds (Warm-up)
Chain 2:                0.069 seconds (Sampling)
Chain 2:                0.143 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
```

```
Chain 3: Gradient evaluation took 7e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07
seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.08 seconds (Warm-up)
Chain 3:                0.069 seconds (Sampling)
Chain 3:                0.149 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 7e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.07
seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.08 seconds (Warm-up)
Chain 4:                0.069 seconds (Sampling)
Chain 4:                0.149 seconds (Total)
Chain 4:
Warning: Bulk Effective Samples Size (ESS) is too low, indicating posterior means and
medians may be unreliable.
Running the chains for more iterations may help. See
https://mc-stan.org/misc/warnings.html#bulk-ess
```

```
# Checking for convergence using Rhat
different_intercept_summary <- summary(different_intercept_model)
```

```
alpha_params <- grep("^alpha\\[", rownames(different_intercept_summary$summary), value
print(different_intercept_summary$summary[alpha_params[1:50],])
```

```
                mean     se_mean        sd         2.5%         25%
alpha[1]  -0.3279033623 0.007127669 0.4201719 -1.196914479 -0.60238247
alpha[2]   0.2777955274 0.005605312 0.2821762 -0.304870236  0.09042330
alpha[3]   0.5018051248 0.008652938 0.2820666 -0.054958838  0.32097511
alpha[4]  -0.3255162472 0.007170161 0.4089806 -1.181536551 -0.57724023
alpha[5]   0.5365776697 0.006780572 0.3263397 -0.128583488  0.31749734
alpha[6]  -0.7223092993 0.006135657 0.2361510 -1.184438138 -0.88074226
alpha[7]   0.4874282260 0.008791637 0.2409893 -0.006912346  0.32639376
alpha[8]  -0.5801324870 0.010033586 0.3255560 -1.212286790 -0.78569681
alpha[9]   0.7370127957 0.004287007 0.2537911  0.190211422  0.57745065
alpha[10]  0.7840738191 0.003205310 0.2181108  0.328232607  0.64087190
alpha[11] -0.1381278838 0.007680625 0.4713026 -1.144345411 -0.44107484
alpha[12]  0.8070245608 0.007662846 0.3257399  0.126105862  0.60505555
alpha[13] -0.0005622682 0.005012379 0.3015447 -0.631776101 -0.19279592
alpha[14]  0.6498094009 0.004165064 0.2253630  0.190746766  0.50434643
alpha[15]  0.3377456499 0.004076391 0.2670106 -0.230528230  0.16939091
alpha[16]  0.8982373624 0.005770994 0.2796568  0.349121973  0.70145339
alpha[17]  1.0211889103 0.004949474 0.2603642  0.472802761  0.85307672
alpha[18]  0.6009808210 0.003701576 0.2185282  0.165496979  0.45843959
alpha[19]  0.0523238052 0.007397304 0.2330924 -0.413372694 -0.10300587
alpha[20]  0.4384267581 0.006176919 0.4095593 -0.435684494  0.19274635
alpha[21]  0.8395892878 0.007085055 0.3292331  0.163755002  0.62720996
alpha[22]  1.0587399879 0.005002364 0.2453051  0.557577555  0.90096108
alpha[23]  0.4446418345 0.005816123 0.2873401 -0.146858153  0.25805465
alpha[24] -0.1344009812 0.007296494 0.4121529 -0.996588648 -0.40632248
alpha[25]  0.2958100024 0.007490978 0.2566306 -0.236572936  0.12604627
alpha[26]  0.8521189167 0.003783602 0.2362634  0.356506976  0.69704474
alpha[27]  0.6593011404 0.005339910 0.2657582  0.094566607  0.50188142
alpha[28]  0.8143536825 0.005846687 0.2524026  0.291703707  0.64757091
alpha[29] -0.0896570447 0.009789330 0.5160278 -1.144449649 -0.44141038
alpha[30]  0.5720100196 0.009056289 0.3486586 -0.155214779  0.34986341
alpha[31]  0.1812821027 0.007045342 0.3665650 -0.604471276 -0.05416582
alpha[32] -0.5239371618 0.008418791 0.4878506 -1.524725511 -0.84575086
alpha[33]  0.7083980210 0.004210425 0.3014309  0.068071547  0.51522132
alpha[34]  0.3841237233 0.005385753 0.2972280 -0.269179750  0.18886912
alpha[35] -0.0882202238 0.008136317 0.3907169 -0.866766319 -0.34576308
alpha[36] -0.4631802719 0.009149785 0.2874405 -1.056690705 -0.65285244
alpha[37]  0.5898964417 0.004715852 0.2768708  0.021497898  0.41360099
alpha[38] -0.1151207188 0.006337693 0.3835463 -0.900632795 -0.36354979
alpha[39] -0.1741752078 0.005366016 0.3216315 -0.851056600 -0.36981863
alpha[40]  0.4344034293 0.007357194 0.3301770 -0.220185991  0.21549876
alpha[41] -0.4499187904 0.007250019 0.4416565 -1.382602560 -0.72330038
alpha[42] -0.2238366933 0.005144897 0.3186408 -0.862285993 -0.43069750
alpha[43]  0.1057963928 0.004645822 0.2859970 -0.473595983 -0.08795429
alpha[44] -0.1737750679 0.004944556 0.2819320 -0.745908158 -0.35985938
alpha[45]  0.8722715242 0.002781465 0.2053035  0.439956853  0.74028778
alpha[46] -0.1816211920 0.006507181 0.2410864 -0.672528776 -0.33637517
```

```
alpha[47]  0.4174547187 0.005659778 0.3182969 -0.238670773  0.21344065
alpha[48]  0.7369671064 0.006562409 0.3927558 -0.098646697  0.49000172
alpha[49]  0.1512120457 0.004812352 0.2705926 -0.420371017 -0.01720715
alpha[50]  0.0242684259 0.004913745 0.2804416 -0.559364017 -0.15446711
                 50%          75%       97.5%     n_eff      Rhat
alpha[1]  -0.31054692 -0.038551921  0.46467084 3475.0335 0.9987712
alpha[2]   0.29375344  0.482159657  0.79160617 2534.2021 0.9999744
alpha[3]   0.50453419  0.691650363  1.04617747 1062.6157 1.0026041
alpha[4]  -0.30815563 -0.038386330  0.44091445 3253.4767 0.9992632
alpha[5]   0.54530229  0.764289717  1.14551871 2316.3655 0.9996913
alpha[6]  -0.71408045 -0.560751064 -0.27926828 1481.3487 1.0022447
alpha[7]   0.48909650  0.653100783  0.95306239  751.3736 1.0070922
alpha[8]  -0.57486963 -0.362863172  0.05017226 1052.7832 1.0046788
alpha[9]   0.75080942  0.916548392  1.19830789 3504.6475 0.9989629
alpha[10]  0.79294550  0.932432989  1.19860687 4630.3557 0.9994332
alpha[11] -0.10685502  0.184291813  0.74069531 3765.3602 0.9997242
alpha[12]  0.82026285  1.020592490  1.40493419 1807.0146 1.0012035
alpha[13]  0.01889616  0.205538848  0.54661073 3619.2266 0.9986368
alpha[14]  0.65612046  0.800196130  1.07551675 2927.6691 1.0006635
alpha[15]  0.34972068  0.515399621  0.83167995 4290.4737 1.0001678
alpha[16]  0.91224345  1.099616346  1.40952127 2348.2777 1.0001682
alpha[17]  1.03353544  1.207685740  1.48829658 2767.2243 0.9992161
alpha[18]  0.60554011  0.750998295  1.00751290 3485.3111 0.9994483
alpha[19]  0.05093500  0.209895809  0.50524672  992.9088 1.0021991
alpha[20]  0.45873948  0.716654747  1.19847148 4396.3256 0.9996420
alpha[21]  0.85635344  1.053140786  1.46857150 2159.3373 0.9997610
alpha[22]  1.06610072  1.222122019  1.52545370 2404.7096 1.0002285
alpha[23]  0.45783576  0.640722947  0.99579238 2440.7624 1.0010047
alpha[24] -0.11669437  0.155297075  0.62623561 3190.7166 0.9983671
alpha[25]  0.31400827  0.478497190  0.76638404 1173.6535 1.0042080
alpha[26]  0.86116185  1.012687488  1.28493202 3899.2586 0.9994143
alpha[27]  0.66924938  0.840640330  1.17659159 2476.8834 0.9993096
alpha[28]  0.82104337  0.992955582  1.26952449 1863.6663 1.0008751
alpha[29] -0.05958651  0.277695808  0.80040825 2778.6910 0.9998599
alpha[30]  0.58468669  0.810651851  1.24040968 1482.1777 1.0011830
alpha[31]  0.20169303  0.440078013  0.84340240 2707.0605 1.0001984
alpha[32] -0.49726442 -0.198492901  0.40277710 3357.9505 0.9989854
alpha[33]  0.72328850  0.916027211  1.25670991 5125.3513 0.9986741
alpha[34]  0.40569354  0.590654799  0.95293919 3045.6961 0.9991401
alpha[35] -0.06801694  0.176256725  0.65340610 2306.0492 1.0018174
alpha[36] -0.45345879 -0.267083291  0.09682607  986.9025 1.0034173
alpha[37]  0.59804503  0.778688149  1.10184142 3446.9425 0.9990267
alpha[38] -0.10482306  0.142326860  0.60091969 3662.4637 0.9988689
alpha[39] -0.16464546  0.048185829  0.41563646 3592.6362 0.9993695
alpha[40]  0.44319476  0.665098835  1.03612371 2014.0454 1.0005569
alpha[41] -0.42675783 -0.140648926  0.33407539 3711.0000 0.9993862
alpha[42] -0.21970995  0.009666764  0.33858072 3835.7417 1.0001233
alpha[43]  0.12323925  0.305461663  0.63428983 3789.6383 0.9989518
alpha[44] -0.16460213  0.025109763  0.33932733 3251.1273 0.9990618
alpha[45]  0.88073131  1.013439646  1.25420120 5448.1039 0.9992878
alpha[46] -0.17519131 -0.013975275  0.26931043 1372.6493 1.0051173
alpha[47]  0.43916931  0.633038484  1.02247366 3162.7587 0.9988578
alpha[48]  0.76416399  0.991683571  1.45045716 3581.9431 0.9996139
```

```
alpha[49]  0.16486541  0.330559682  0.64052180 3161.6739 0.9988046
alpha[50]  0.03122626  0.217849823  0.53814389 3257.3153 0.9995139
```

```
print(different_intercept_summary$summary["beta",])
```

```
      mean      se_mean           sd         2.5%          25%          50%
 1.50320798   0.03202149   0.60461583   0.34457848   1.08280494   1.50752378
       75%        97.5%        n_eff         Rhat
 1.92917244   2.65109831 356.51344472   1.01465338
```

3. Intercept α_i is different in each region and the intercept is modeled hierarchically

```
hierarchically_intercept_model_code  <- "
data {
  int<lower=0> N;  // number of regions
  vector[N] x;  // covariate
  vector[N] offset;  // log of expected deaths
  int<lower=0> deaths[N];  // observed deaths
}
parameters {
  vector[N] alpha;  // separate intercepts for each region
  real mu_alpha;  // hyperparameter for the mean of the intercepts
  real<lower=0> sigma_alpha;  // hyperparameter for the standard deviation of the inte
  real beta;  // covariate coefficient
}
model {
  vector[N] log_lambda = alpha + beta * x + offset;  // linear predictor

  // Hyperpriors
  mu_alpha ~ normal(0, 1);
  sigma_alpha ~ normal(0, 1);

  // Priors
  alpha ~ normal(mu_alpha, sigma_alpha);
  beta ~ normal(0, 1);

  // Likelihood
  deaths ~ poisson_log(log_lambda);
}
generated quantities {
  vector[N] log_lik;  // log-likelihood for each observation
  vector[N] log_theta;  // log of estimated theta for each region

  for (i in 1:N) {
    log_lik[i] = poisson_log_lpmf(deaths[i] | alpha[i] + beta * x[i] + offset[i]);
    // The log_theta is the linear predictor without the offset
    log_theta[i] = alpha[i] + beta * x[i];
  }
}
"
```

```
# Compile the model
hierarchically_intercept_model <- stan(
  model_code = hierarchically_intercept_model_code,
  data = lip_cancer_data,
  chains = 4,
  iter = 1000,
)
```

```
Trying to compile a simple C file

Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
using C compiler: ‘Apple clang version 14.0.3 (clang-1403.0.22.14.1)’
using SDK: ‘MacOSX13.3.sdk’
clang -arch arm64 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/Rcpp/include/"
-I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/unsupported"  -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/BH/include" -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/src/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppParallel/include/"  -
I"/Library/Frameworks/R.framework/Versions/4.3-arm64/Resources/library/rstan/include"
-DEIGEN_NO_DEBUG  -DBOOST_DISABLE_ASSERTS  -DBOOST_PENDING_INTEGER_LOG2_HPP  -
DSTAN_THREADS  -DUSE_STANC3 -DSTRICT_R_HEADERS  -DBOOST_PHOENIX_NO_VARIADIC_EXPRESSION
-D_HAS_AUTO_PTR_ETC=0  -include '/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp'  -
D_REENTRANT -DRCPP_PARALLEL_USE_TBB=1   -I/opt/R/arm64/include    -fPIC  -falign-
functions=64 -Wall -g -O2  -c foo.c -o foo.o
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Dense:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Core:88:
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:1: error:
unknown type name 'namespace'
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/src/Core/util/Macros.h:628:16: error:
expected ';' after top level declarator
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
```

```
arm64/Resources/library/StanHeaders/include/stan/math/prim/fun/Eigen.hpp:22:
In file included from /Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Dense:1:
/Library/Frameworks/R.framework/Versions/4.3-
arm64/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal error: 'complex'
file not found
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.42
seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 1: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 1: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 1: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 1: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 1: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 1: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 1: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 1: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 1: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 1: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 1: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.094 seconds (Warm-up)
Chain 1:                0.071 seconds (Sampling)
Chain 1:                0.165 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 1.1e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.11
seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 2: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 2: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 2: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 2: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 2: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 2: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 2: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 2: Iteration: 700 / 1000 [ 70%]  (Sampling)
```

```
Chain 2: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 2: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 2: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.094 seconds (Warm-up)
Chain 2:                0.071 seconds (Sampling)
Chain 2:                0.165 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 7e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.07
seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 3: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 3: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 3: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 3: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 3: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 3: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 3: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 3: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 3: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 3: Iteration: 900 / 1000 [ 90%]  (Sampling)
Chain 3: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.092 seconds (Warm-up)
Chain 3:                0.072 seconds (Sampling)
Chain 3:                0.164 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 8e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.08
seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:   1 / 1000 [  0%]  (Warmup)
Chain 4: Iteration: 100 / 1000 [ 10%]  (Warmup)
Chain 4: Iteration: 200 / 1000 [ 20%]  (Warmup)
Chain 4: Iteration: 300 / 1000 [ 30%]  (Warmup)
Chain 4: Iteration: 400 / 1000 [ 40%]  (Warmup)
Chain 4: Iteration: 500 / 1000 [ 50%]  (Warmup)
Chain 4: Iteration: 501 / 1000 [ 50%]  (Sampling)
Chain 4: Iteration: 600 / 1000 [ 60%]  (Sampling)
Chain 4: Iteration: 700 / 1000 [ 70%]  (Sampling)
Chain 4: Iteration: 800 / 1000 [ 80%]  (Sampling)
Chain 4: Iteration: 900 / 1000 [ 90%]  (Sampling)
```

```
Chain 4: Iteration: 1000 / 1000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.095 seconds (Warm-up)
Chain 4:                0.075 seconds (Sampling)
Chain 4:                0.17 seconds (Total)
Chain 4:
```

```
# Checking for convergence using Rhat
hierarchically_intercept_summary <- summary(hierarchically_intercept_model)
```

```
print(hierarchically_intercept_summary$summary[c("mu_alpha","sigma_alpha", "beta"), ])
```

```
                mean      se_mean         sd       2.5%        25%       50%
mu_alpha    0.08615279 0.0007684378 0.03549652 0.01355827 0.06254441 0.0859434
sigma_alpha 0.38627786 0.0009650845 0.03155956 0.32737648 0.36395739 0.3853644
beta        1.96432410 0.0086682612 0.33211637 1.32371797 1.74659821 1.9537232
                75%     97.5%    n_eff      Rhat
mu_alpha    0.1103232 0.1542645 2133.802 1.0013849
sigma_alpha 0.4068574 0.4524098 1069.378 1.0027186
beta        2.1960509 2.5825357 1467.968 0.9995202
```

Question 3

Make two plots (appropriately labeled and described) that illustrate the differences in estimated θ_i's across regions and the differences in θs across models.

Differences in estimated θ_i's across regions

```
library(tidyverse)
```

```
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.3     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.3     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.0
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ tidyr::extract() masks rstan::extract()
✖ dplyr::filter()  masks stats::filter()
✖ dplyr::lag()     masks stats::lag()
ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

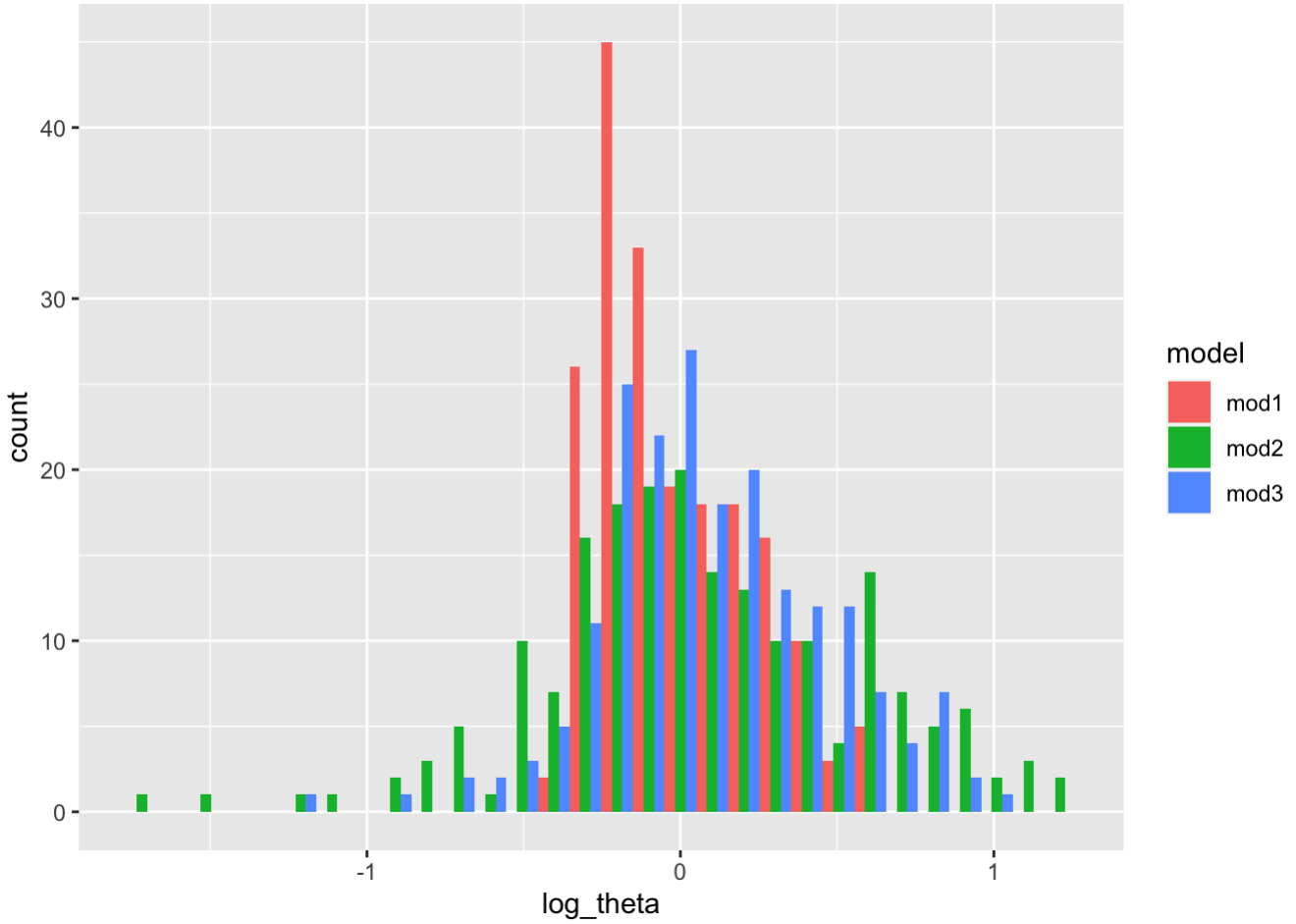
```
library(tidybayes)
library(ggplot2)

res_mod1 <- common_intercept_model %>%
    gather_draws(log_theta[i]) %>%
    median_qi() %>%
    rename(median_mod1 = .value,
```

```
            lower_mod1 = .lower,
            upper_mod1 = .upper) %>%
    select(i,median_mod1:upper_mod1)

res_mod2 <- different_intercept_model %>%
    gather_draws(log_theta[i]) %>%
    median_qi() %>%
    rename(median_mod2 = .value,
           lower_mod2 = .lower,
           upper_mod2 = .upper) %>%
    select(i,median_mod2:upper_mod2)

res_mod3 <- hierarchically_intercept_model %>%
    gather_draws(log_theta[i]) %>%
    median_qi() %>%
    rename(median_mod3 = .value,
           lower_mod3 = .lower,
           upper_mod3 = .upper) %>%
    select(i,median_mod3:upper_mod3)


res <- res_mod1%>%
  left_join(res_mod2) %>%
  left_join(res_mod3)
```

```
Joining with `by = join_by(i)`
Joining with `by = join_by(i)`
```

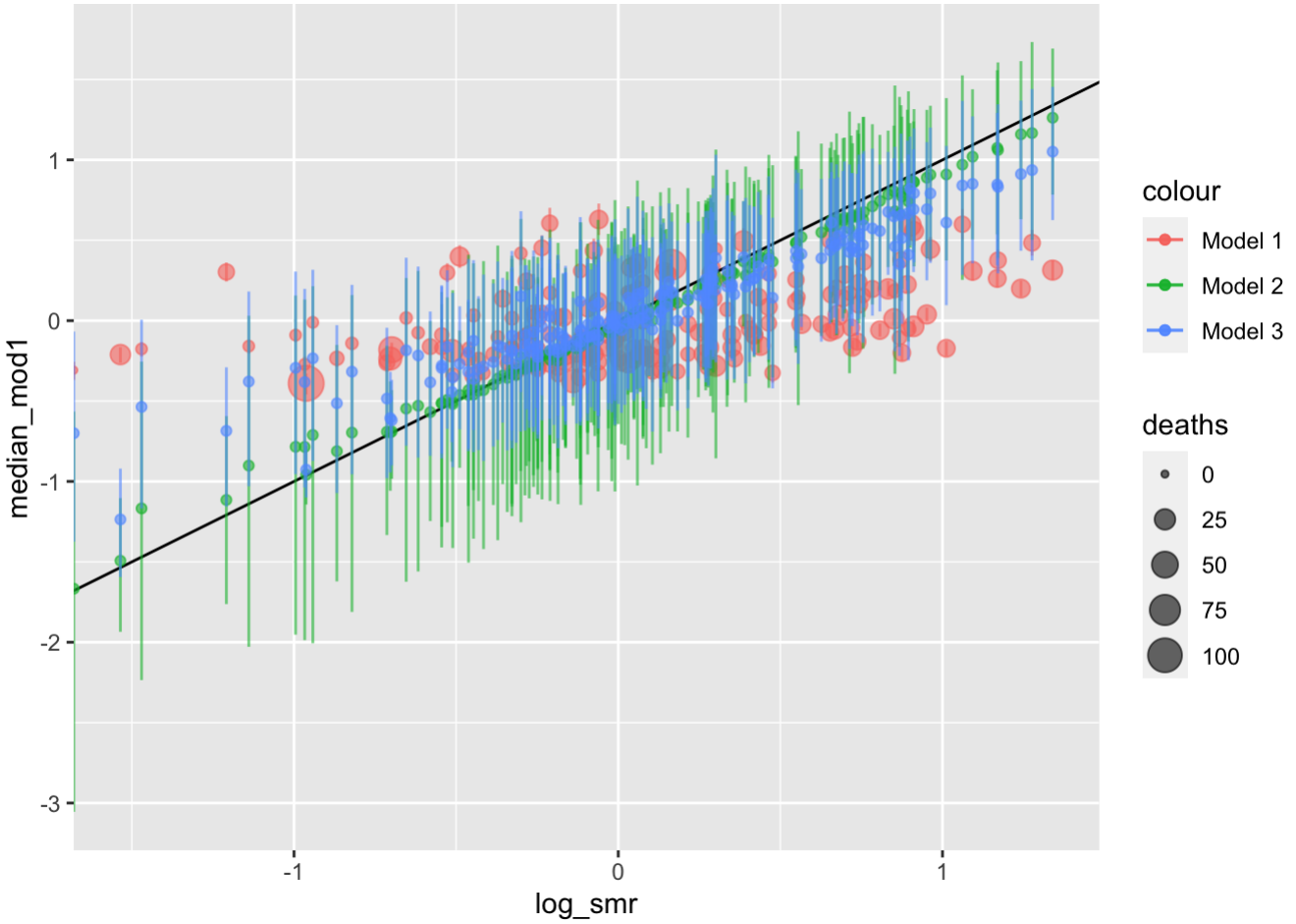
```
res %>%
  select(median_mod1,
         median_mod2,
         median_mod3) %>%
  pivot_longer(median_mod1:median_mod3, names_to = "model", values_to = "log_theta") %
  mutate(model = str_remove(model, "median_")) %>%
  ggplot(aes(log_theta, fill = model)) +
  geom_histogram(position = "dodge")
```

```
`stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Differences in estimated θ_i's across models

```
res  %>%
  mutate(deaths = observe.i)  %>%
  mutate(log_smr = log(observe.i/expect.i))  %>%
  ggplot(aes(log_smr, median_mod1, color = "Model 1"))+
  geom_point(aes(size = deaths), alpha = 0.6)+
  geom_errorbar(aes(ymin = lower_mod1, ymax = upper_mod1, color = "Model 1"), alpha =
  geom_abline(slope = 1, intercept = 0)+
  geom_point(aes(log_smr, median_mod2, color = "Model 2"),alpha = 0.6)+
  geom_errorbar(aes(ymin = lower_mod2, ymax = upper_mod2, color = "Model 2"), alpha =
  geom_point(aes(log_smr, median_mod3, color = "Model 3"), alpha = 0.6)+
  geom_errorbar(aes(ymin = lower_mod3, ymax = upper_mod3, color = "Model 3"), alpha =
```



Question 4

Using tool of your choice, decide which model is the best, and justify your choice.

```
library(loo)
```

```
This is loo version 2.6.0

- Online documentation and vignettes at mc-stan.org/loo

- As of v2.0.0 loo defaults to 1 core but we recommend using as many as possible. Use
the 'cores' argument or set options(mc.cores = NUM_CORES) for an entire session.


Attaching package: 'loo'

The following object is masked from 'package:rstan':

    loo
```

```
# Assuming you have already extracted the log_lik for each model
log_lik1 <- rstan::extract(common_intercept_model)$log_lik
log_lik2 <- rstan::extract(different_intercept_model)$log_lik
log_lik3 <- rstan::extract(hierarchically_intercept_model)$log_lik
```

```
# Calculate LOO-CV for each model
loo1 <- loo(log_lik1)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.

Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic')
for details.
```

```
loo2 <- loo(log_lik2)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.

Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic')
for details.
```

```
loo3 <- loo(log_lik3)
```

```
Warning: Relative effective sample sizes ('r_eff' argument) not specified.
For models fit with MCMC, the reported PSIS effective sample sizes and
MCSE estimates will be over-optimistic.

Warning: Some Pareto k diagnostic values are too high. See help('pareto-k-diagnostic')
for details.
```

```
# Compare models using LOO
loo_compare <- loo_compare(loo1, loo2, loo3)
print(loo_compare)
```

```
       elpd_diff se_diff
model3    0.0       0.0
model2  -15.1       8.2
model1 -147.7      44.0
```

Model3: With an elpd_diff of 0.0, this model has the highest ELPD among the three, making it the best model in terms of predictive performance according to this metric. Based on these results, the hierarchical intercepts model is the best-performing model among the three. It provides the best balance between model complexity and fit to the data.