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
  for (i in 1:N) {
    log_lik[i] = poisson_log_lpmf(deaths[i] | alpha + beta * x[i] + offset[i]);
  }
}
"
```

```
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 2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.2
```

```
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: 0.018 seconds (Warm-up)
Chain 1:                0.013 seconds (Sampling)
Chain 1:                0.031 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: 0.021 seconds (Warm-up)
Chain 2:                0.015 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: 0.018 seconds (Warm-up)
Chain 3:                0.013 seconds (Sampling)
Chain 3:                0.031 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: 0.019 seconds (Warm-up)
Chain 4:                0.014 seconds (Sampling)
Chain 4:                0.033 seconds (Total)
Chain 4:
```

```
# 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 -0.008931732 0.0004449427 0.01999875 -0.04899366 -0.02257104 -0.008060483
beta   2.428408094 0.0041587096 0.17791653  2.08644319  2.30852598  2.419831781
```

```
              75%      97.5%    n_eff     Rhat
alpha 0.004618753 0.02877339 2020.216 1.000836
beta  2.555079022 2.77368550 1830.271 1.000750
```

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
  for (i in 1:N) {
    log_lik[i] = poisson_log_lpmf(deaths[i] | alpha[i] + beta * x[i] + offset[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 2.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.28
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: 0.076 seconds (Warm-up)
Chain 1:                0.068 seconds (Sampling)
Chain 1:                0.144 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07
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: 0.08 seconds (Warm-up)
Chain 2:                0.067 seconds (Sampling)
Chain 2:                0.147 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: 0.078 seconds (Warm-up)
Chain 3:                0.068 seconds (Sampling)
Chain 3:                0.146 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 6e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.06
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: 0.082 seconds (Warm-up)
Chain 4:                0.069 seconds (Sampling)
Chain 4:                0.151 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
```

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

```r
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]	-0.322977432	0.007975174	0.4241573	-1.21193849	-0.60561244
alpha[2]	0.281221544	0.005636840	0.2768791	-0.28915408	0.09335149
alpha[3]	0.513516088	0.008810156	0.2608349	0.01348884	0.33705220
alpha[4]	-0.328452979	0.008777972	0.4101816	-1.17293809	-0.60329423
alpha[5]	0.535018245	0.007306558	0.3166484	-0.10357347	0.32008391
alpha[6]	-0.724799135	0.007369960	0.2514880	-1.22829986	-0.89166843
alpha[7]	0.504107255	0.009091829	0.2272971	0.02931830	0.35328079
alpha[8]	-0.555327216	0.008751265	0.3192016	-1.21663279	-0.75880623
alpha[9]	0.731163382	0.005263597	0.2606275	0.18573919	0.56336155
alpha[10]	0.788153608	0.003824240	0.2156688	0.36112018	0.64825090
alpha[11]	-0.137675315	0.009020891	0.4569423	-1.08527160	-0.42207601
alpha[12]	0.818404450	0.006539018	0.3115861	0.18074934	0.62095074
alpha[13]	0.002758536	0.005669228	0.3064652	-0.61128213	-0.20160148
alpha[14]	0.656294314	0.005095011	0.2157047	0.19889264	0.52168095
alpha[15]	0.344307828	0.004983427	0.2671957	-0.22397296	0.16952484
alpha[16]	0.899750905	0.005795649	0.2791724	0.33770047	0.72477248
alpha[17]	1.027834309	0.004954550	0.2484019	0.48717418	0.87012342
alpha[18]	0.598052669	0.003676044	0.2151272	0.14904678	0.46949970
alpha[19]	0.057071315	0.008396824	0.2360259	-0.39703754	-0.10089095
alpha[20]	0.455563104	0.008080719	0.4039636	-0.44994316	0.20533252
alpha[21]	0.856833918	0.008725324	0.3120727	0.18546195	0.66898174
alpha[22]	1.065717109	0.006010559	0.2286614	0.59561601	0.91015561
alpha[23]	0.438244874	0.007170078	0.2925918	-0.17562224	0.25666918
alpha[24]	-0.121925903	0.007215983	0.4060839	-0.98685315	-0.37327664
alpha[25]	0.310711771	0.009092820	0.2539898	-0.20270655	0.14212850
alpha[26]	0.838358221	0.004604694	0.2385698	0.34936955	0.67993835
alpha[27]	0.667707534	0.005279605	0.2700652	0.12313790	0.49253113
alpha[28]	0.826379552	0.005632980	0.2532669	0.32892140	0.66030839
alpha[29]	-0.067584211	0.011139022	0.5091674	-1.11480085	-0.38346642
alpha[30]	0.583588924	0.009845603	0.3545814	-0.14270747	0.34740973
alpha[31]	0.182123420	0.008194961	0.3825512	-0.62572550	-0.05798163
alpha[32]	-0.522873598	0.009390743	0.4827998	-1.52156159	-0.83450555
alpha[33]	0.717898924	0.005303951	0.2986833	0.08856555	0.50754055
alpha[34]	0.389092757	0.005020980	0.3094170	-0.25114309	0.19555661
alpha[35]	-0.091980047	0.008209978	0.3802822	-0.86467725	-0.34536191
alpha[36]	-0.447974333	0.009153537	0.2826082	-1.01055552	-0.63358272
alpha[37]	0.590378149	0.005142340	0.2834834	0.03679178	0.40291982
alpha[38]	-0.117130904	0.007540760	0.3822116	-0.89765224	-0.36805338
alpha[39]	-0.172788079	0.006258349	0.3062900	-0.79581162	-0.36948421
alpha[40]	0.432969705	0.007618102	0.3360335	-0.29099638	0.22105002
alpha[41]	-0.457721218	0.009283353	0.4645567	-1.44714054	-0.75387613
alpha[42]	-0.227076573	0.005497907	0.3158655	-0.89027561	-0.43272838
alpha[43]	0.097327707	0.006318357	0.3065003	-0.54126939	-0.10458210
alpha[44]	-0.174707721	0.005167402	0.2788579	-0.76692508	-0.34908399
alpha[45]	0.868282062	0.003933761	0.2059852	0.44939663	0.73093791
alpha[46]	-0.172741937	0.006226038	0.2442696	-0.64562433	-0.33472195
alpha[47]	0.418748506	0.005964600	0.3028610	-0.22728885	0.22819864
alpha[48]	0.752483260	0.007524654	0.3626991	-0.02001710	0.52175326
alpha[49]	0.153755834	0.005437192	0.2762874	-0.42784252	-0.02246503
alpha[50]	0.025757584	0.005359607	0.2753365	-0.58276840	-0.14597589

	50%	75%	97.5%	n_eff	Rhat
alpha[1]	-0.30229158	-0.021665187	0.44539244	2828.6127	0.9989681
alpha[2]	0.28665508	0.481357355	0.79119321	2412.7316	1.0009346

```
alpha[3]   0.51519956  0.691929324  1.02201216  876.5251 1.0012934
alpha[4]  -0.31422901 -0.052313602  0.44129606 2183.5547 1.0004049
alpha[5]   0.55936657  0.751008934  1.10557124 1878.1441 0.9998169
alpha[6]  -0.71369147 -0.544957016 -0.25900060 1164.4059 1.0015340
alpha[7]   0.50628154  0.655919951  0.94331091  625.0078 1.0013354
alpha[8]  -0.55004468 -0.341852409  0.04979803 1330.4188 0.9999661
alpha[9]   0.74442476  0.908784401  1.22016027 2451.7445 0.9991246
alpha[10]  0.79767126  0.938885281  1.19816716 3180.4184 0.9994773
alpha[11] -0.11587848  0.182297451  0.70465727 2565.8065 0.9992320
alpha[12]  0.82529933  1.031378338  1.41387729 2270.5499 1.0000430
alpha[13]  0.01205867  0.220730891  0.55086420 2922.2297 0.9987261
alpha[14]  0.66294635  0.800374068  1.05566318 1792.3752 0.9999299
alpha[15]  0.36278833  0.532856381  0.82712975 2874.7670 0.9988052
alpha[16]  0.90923051  1.095665014  1.40531128 2320.2817 1.0002707
alpha[17]  1.04062311  1.199760444  1.49497271 2513.6295 1.0004662
alpha[18]  0.60204209  0.741553658  1.00297350 3424.7518 0.9986145
alpha[19]  0.04762640  0.216397216  0.52753459  790.1129 1.0036774
alpha[20]  0.47661847  0.729787789  1.17457105 2499.1050 0.9996961
alpha[21]  0.86691088  1.070642666  1.45382298 1279.2297 1.0016216
alpha[22]  1.06855464  1.222055885  1.50074658 1447.2914 0.9993689
alpha[23]  0.45032728  0.645914247  0.96900608 1665.2387 1.0003945
alpha[24] -0.09053686  0.160231435  0.57890660 3166.9451 0.9993246
alpha[25]  0.31520520  0.489734937  0.79528458  780.2528 1.0022005
alpha[26]  0.84409615  1.000918692  1.27870836 2684.2897 0.9995733
alpha[27]  0.68161494  0.861405971  1.16900024 2616.5819 0.9993233
alpha[28]  0.82920809  1.002761431  1.31758101 2021.5316 0.9989642
alpha[29] -0.04216567  0.284787177  0.86943582 2089.4264 0.9992661
alpha[30]  0.59975286  0.825979241  1.24873113 1297.0219 1.0015793
alpha[31]  0.19744009  0.440707818  0.87544011 2179.1407 0.9994746
alpha[32] -0.49981059 -0.195404108  0.37569109 2643.2252 1.0022817
alpha[33]  0.72870372  0.932235225  1.25024980 3171.1937 0.9999068
alpha[34]  0.39946403  0.589886278  0.96349471 3797.6202 0.9985211
alpha[35] -0.06873726  0.177075161  0.58859946 2145.4968 1.0004013
alpha[36] -0.44544164 -0.253068684  0.09179391  953.2165 1.0020673
alpha[37]  0.60302744  0.794418807  1.08849004 3039.0215 0.9992479
alpha[38] -0.10675252  0.160579045  0.59797097 2569.0787 1.0006008
alpha[39] -0.15916918  0.039273722  0.38895072 2395.2246 0.9999254
alpha[40]  0.45115366  0.656905938  1.04591432 1945.6811 1.0018417
alpha[41] -0.42653781 -0.116427566  0.35202638 2504.1931 1.0012713
alpha[42] -0.20635737 -0.008210072  0.35102443 3300.7282 0.9982680
alpha[43]  0.10645573  0.313617878  0.66233860 2353.1711 0.9992026
alpha[44] -0.16053758  0.016011701  0.32416391 2912.2021 0.9982501
alpha[45]  0.87008527  1.010983210  1.26372503 2741.9289 1.0002403
alpha[46] -0.17329501 -0.010039618  0.29311604 1539.2710 1.0015847
alpha[47]  0.43583405  0.629622739  0.95427016 2578.2444 0.9994571
alpha[48]  0.77340477  1.006622480  1.42164458 2323.3780 1.0020469
alpha[49]  0.16112926  0.341728982  0.66403553 2582.0990 0.9989993
alpha[50]  0.04963677  0.207485089  0.52364347 2639.1355 1.0003110
```

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

```
      mean      se_mean           sd         2.5%          25%          50%
 1.47078878   0.03579406   0.59789500   0.28945444   1.07734928   1.47300397
```

```
        75%        97.5%        n_eff         Rhat
 1.87459014   2.62581430 279.01530923   1.01097824
```

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
  for (i in 1:N) {
    log_lik[i] = poisson_log_lpmf(deaths[i] | alpha[i] + beta * x[i] + offset[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 3.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.31
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: 0.108 seconds (Warm-up)
Chain 1:                0.07 seconds (Sampling)
Chain 1:                0.178 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 7e-06 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.07
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: 0.095 seconds (Warm-up)
Chain 2:                0.069 seconds (Sampling)
Chain 2:                0.164 seconds (Total)
Chain 2:

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

```
Chain 3: Gradient evaluation took 8e-06 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.08
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: 0.091 seconds (Warm-up)
Chain 3:                0.07 seconds (Sampling)
Chain 3:                0.161 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'anon_model' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 9e-06 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.09
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: 0.088 seconds (Warm-up)
Chain 4:                0.071 seconds (Sampling)
Chain 4:                0.159 seconds (Total)
Chain 4:
```

```r
# 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    0.08707931 0.0007398024 0.03540539 0.01842742 0.06281712 0.08672262
sigma_alpha 0.38632914 0.0008857934 0.03032605 0.32884427 0.36596130 0.38561395
beta        1.97099040 0.0089397017 0.33389515 1.30813767 1.74793237 1.97241230
                75%     97.5%    n_eff      Rhat
mu_alpha    0.1108278 0.1561686 2290.379 0.9997888
sigma_alpha 0.4062282 0.4464627 1172.106 1.0025580
beta        2.1949735 2.6327143 1395.000 1.0013896
```

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.

```
# Model1

# Extract the posterior samples for alpha and beta
alpha_samples <- rstan::extract(common_intercept_model)$alpha
beta_samples <-  rstan::extract(common_intercept_model)$beta

# Calculate the mean of the posterior samples for alpha and beta
alpha_mean <- mean(alpha_samples)
beta_mean <- mean(beta_samples)

# Calculate the linear predictor for each region using the mean estimates of alpha and
linear_predictor <- alpha_mean + beta_mean * aff_centered + log(expect.i)

# Calculate the estimated theta_i by exponentiating the linear predictor
theta_estimated <- exp(linear_predictor)
```

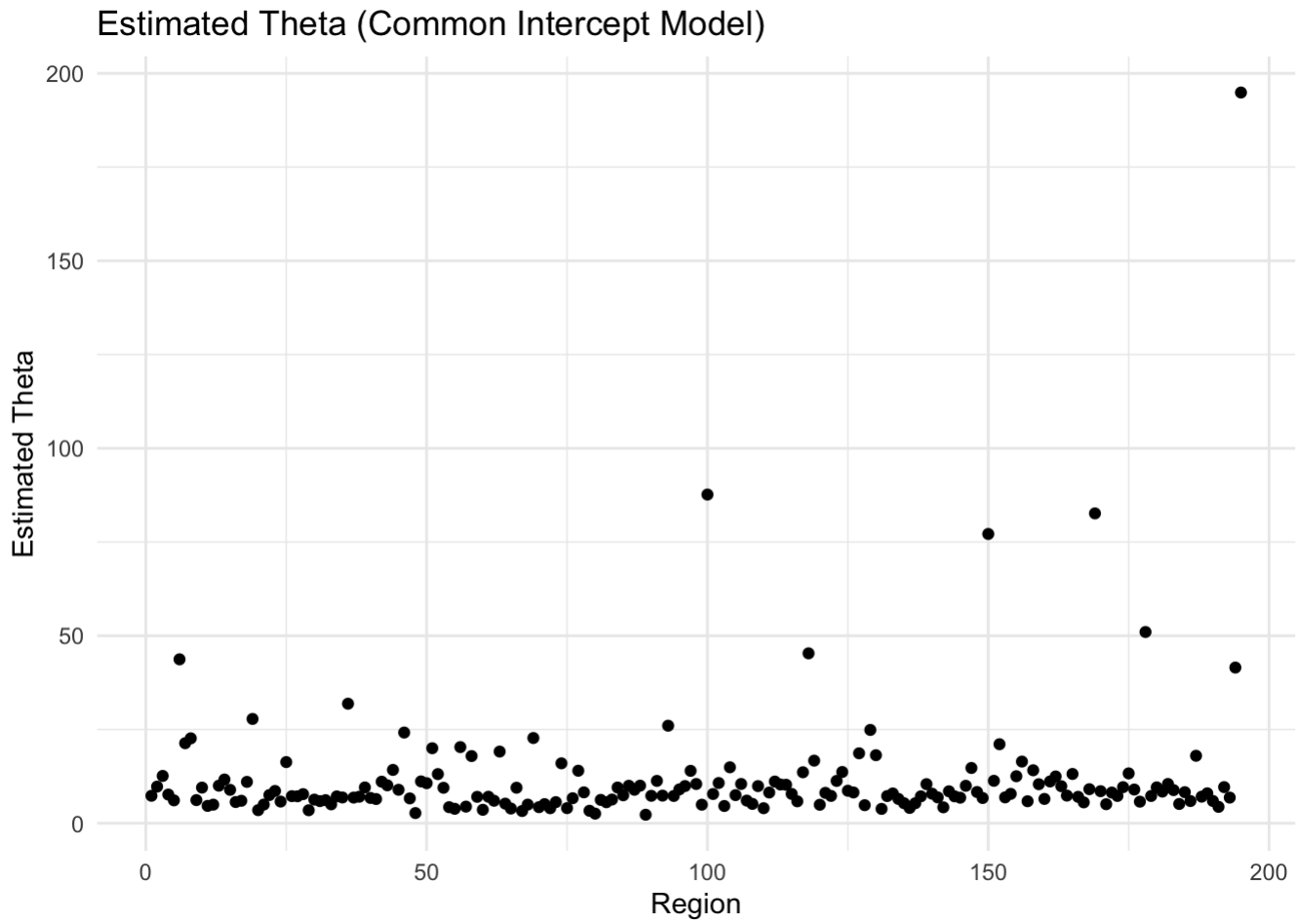
```
# Model2

# Extract the posterior samples for alpha and beta
posterior_alpha <- rstan::extract(different_intercept_model)$alpha  # This will be a m
posterior_beta <- rstan::extract(different_intercept_model)$beta  # This will be a vec

# Calculate the mean of the posterior samples for alpha and beta
alpha_mean2 <- apply(posterior_alpha, 2, mean)  # Take the mean across iterations for
beta_mean2 <- mean(posterior_beta)

# Calculate the linear predictor for each region using the mean estimates of alpha and
linear_predictor2 <- alpha_mean2 + beta_mean2 * aff_centered + log(expect.i)

# Calculate the estimated theta_i by exponentiating the linear predictor
theta_estimated2 <- exp(linear_predictor2)
```

```
# Model3
# Extract the posterior samples for alpha, mu_alpha, sigma_alpha, and beta
```

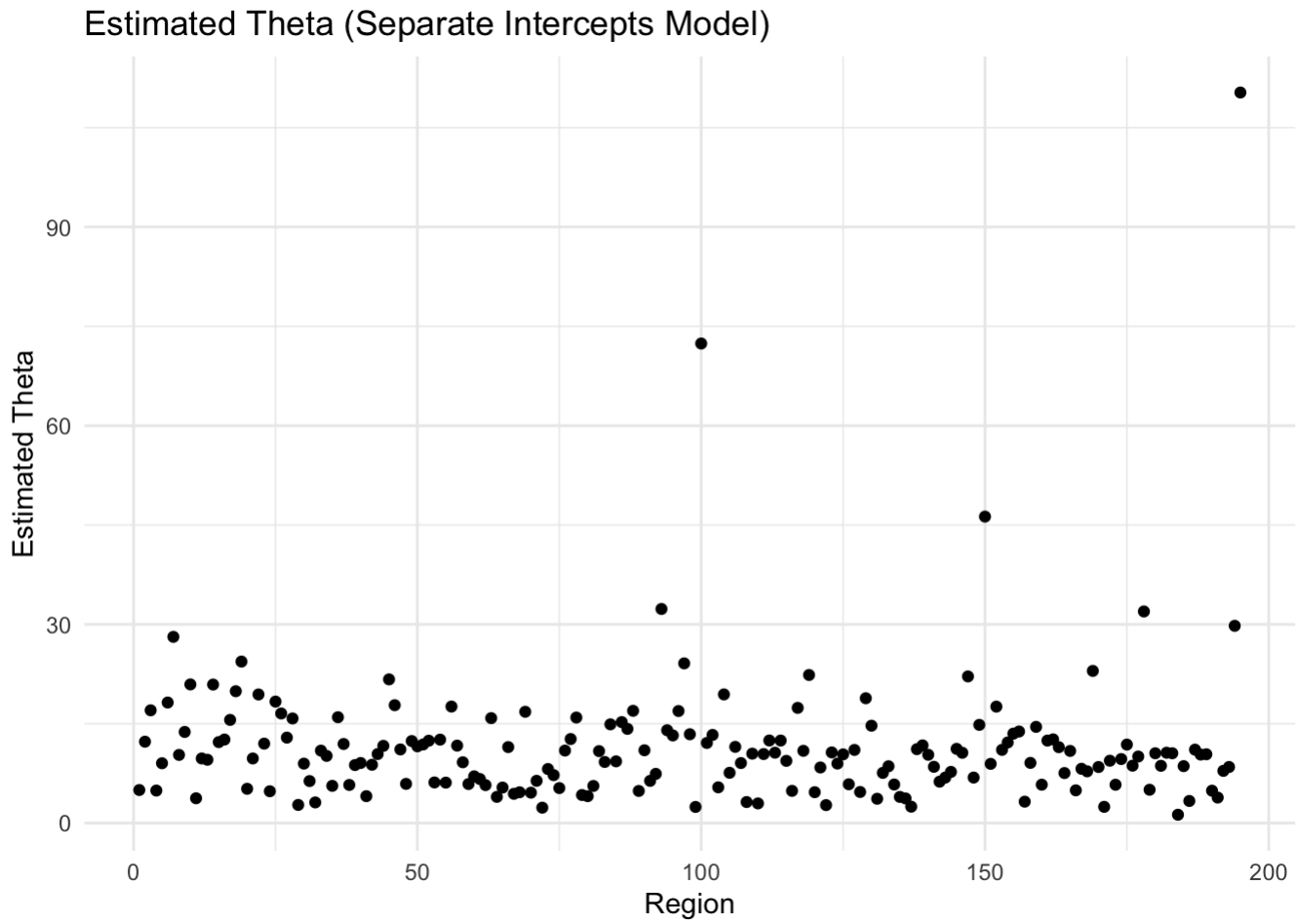
```
posterior_alpha3 <- rstan::extract(hierarchically_intercept_model)$alpha  # This will
posterior_mu_alpha <- rstan::extract(hierarchically_intercept_model)$mu_alpha  # This
posterior_sigma_alpha <- rstan::extract(hierarchically_intercept_model)$sigma_alpha  #
posterior_beta3 <- rstan::extract(hierarchically_intercept_model)$beta  # This will be

# Calculate the mean of the posterior samples for alpha and beta
alpha_mean3 <- apply(posterior_alpha3, 2, mean)  # Take the mean across iterations for
beta_mean3 <- mean(posterior_beta3)

# Calculate the linear predictor for each region using the mean estimates of alpha and
linear_predictor3 <- alpha_mean3 + beta_mean3 * aff_centered + log(expect.i)

# Calculate the estimated theta_i by exponentiating the linear predictor
theta_estimated3 <- exp(linear_predictor3)
```

Differences in estimated θ_i's across regions

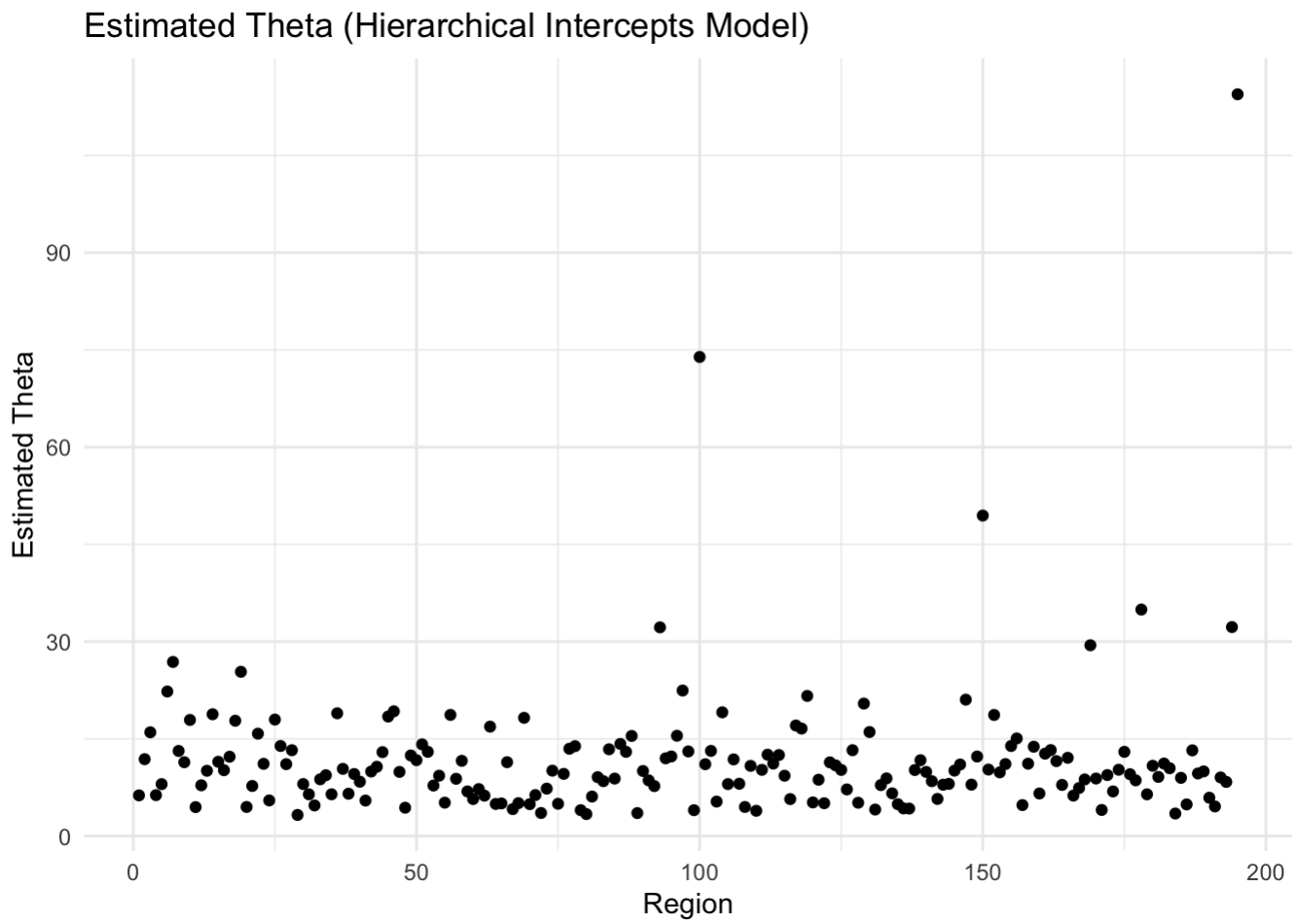
```
library(ggplot2)
# Create data frames for each model
theta_data_common <- data.frame(region = 1:N, theta = theta_estimated, Model = "Common
theta_data_separate <- data.frame(region = 1:N, theta = theta_estimated2, Model = "Sep
theta_data_hierarchical <- data.frame(region = 1:N, theta = theta_estimated3, Model =

# Create a plot for each model
ggplot(theta_data_common, aes(x = region, y = theta)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Estimated Theta (Common Intercept Model)",
       x = "Region",
       y = "Estimated Theta")
```




```
ggplot(theta_data_separate, aes(x = region, y = theta)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Estimated Theta (Separate Intercepts Model)",
       x = "Region",
       y = "Estimated Theta")
```

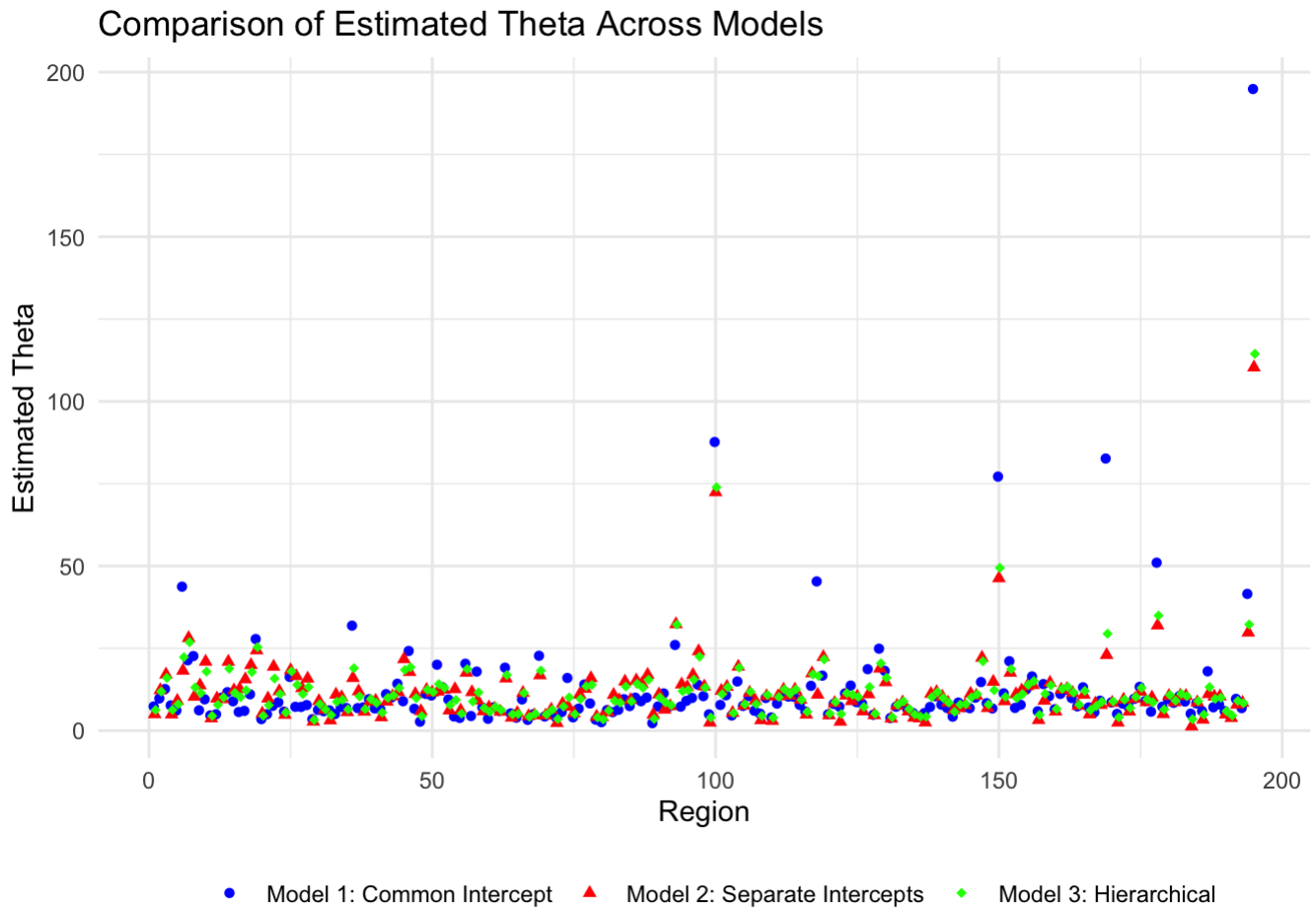



```
ggplot(theta_data_hierarchical, aes(x = region, y = theta)) +
  geom_point() +
  theme_minimal() +
  labs(title = "Estimated Theta (Hierarchical Intercepts Model)",
       x = "Region",
       y = "Estimated Theta")
```



Differences in estimated θ_i's across models

```
# Combine the estimated theta values into a single data frame
comparison_data <- data.frame(
  region = rep(1:N, times = 3),
  theta = c(theta_estimated, theta_estimated2, theta_estimated3),
  model = factor(rep(c("Model 1: Common Intercept", "Model 2: Separate Intercepts", "M
)

# Create the plot
ggplot(comparison_data, aes(x = region, y = theta, color = model)) +
  geom_point(aes(shape = model), position = position_dodge(width = 0.5)) +
  theme_minimal() +
  scale_color_manual(values = c("blue", "red", "green")) +
  scale_shape_manual(values = c(16, 17, 18)) +
  labs(
    title = "Comparison of Estimated Theta Across Models",
    x = "Region",
    y = "Estimated Theta",
    color = "Model",
    shape = "Model"
  ) +
  theme(
    legend.position = "bottom",
    legend.title = element_blank()
  )
```



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  -18.2       7.9
model1 -151.4      45.0
```

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.