Stat 479 Group Project with Matrix and Multiple Logistic Regression

Basketball Playoff Prediction Team

Dec 14 2021

Merging Data and Data Cleaning

```
Regular <- read.csv("cleaned_data_v1_12_18.csv") # import the data
Regular$Success <- as.integer(Regular$Y) # True = 1, False = 0
y <- Regular$Success
deltaOEFF <- Regular[, "deltaOEFF"]
deltaDEFF <- Regular[, "deltaDEFF"]
summary(deltaOEFF) # let the grid for be [-9, 9]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-8.9000 -3.4000  0.4000  0.1876  3.8750  8.9000
```

```
summary(deltaDEFF) # let the grid for be [-9, 9]
```

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-8.8000 -2.7250 -0.3000 -0.1014  2.3000  8.8000
```

After looking at the summary for the 2 predictors, we can see that both range from approximately -9 to 9. Here we decides to create posterior fitting grids also ranges [-9, 9] with 0.1 increment each time.

Model

```
y <- Regular$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for posterior prediction
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # length = 181
x <- cbind(deltaOEFF, deltaDEFF) # dim(502, 2)
K <- 1
N <- length(y) # N = 502
D <- dim(x)[2] # D = 2
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(K = K, # create a list to fill rstan model
                  N = N,
                  D = D,
                  y = y,
                  x = x,
                  n_grid = n_grid,
                  deltaOEFF_grid = deltaOEFF_grid,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "multi_logistic.stan")
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/L
```

```
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal
#include <complex>
         ^~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000329 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 3.29 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 3.26654 seconds (Warm-up)
Chain 1:                3.16912 seconds (Sampling)
Chain 1:                6.43567 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000278 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.78 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 3.19548 seconds (Warm-up)
Chain 2:                3.0831 seconds (Sampling)
Chain 2:                6.27858 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000279 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.79 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 3.24916 seconds (Warm-up)
Chain 3:                3.45328 seconds (Sampling)
Chain 3:                6.70243 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000276 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.76 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

```
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 3.2677 seconds (Warm-up)
Chain 4:                3.35178 seconds (Sampling)
Chain 4:                6.61947 seconds (Total)
Chain 4:
```

```
summary(fit)[[1]][,"Rhat"]
```

```
      beta[1,1]       beta[2,1]  prob_meanD[1]
      1.0002151       0.9996921      1.0004415
 prob_meanD[2]  prob_meanD[3]  prob_meanD[4]
      1.0004384       1.0004352      1.0004321
 prob_meanD[5]  prob_meanD[6]  prob_meanD[7]
      1.0004289       1.0004258      1.0004226
 prob_meanD[8]  prob_meanD[9] prob_meanD[10]
      1.0004194       1.0004163      1.0004131
prob_meanD[11] prob_meanD[12] prob_meanD[13]
      1.0004099       1.0004067      1.0004035
prob_meanD[14] prob_meanD[15] prob_meanD[16]
      1.0004003       1.0003971      1.0003939
prob_meanD[17] prob_meanD[18] prob_meanD[19]
      1.0003907       1.0003874      1.0003842
prob_meanD[20] prob_meanD[21] prob_meanD[22]
      1.0003810       1.0003778      1.0003745
prob_meanD[23] prob_meanD[24] prob_meanD[25]
      1.0003713       1.0003681      1.0003649
prob_meanD[26] prob_meanD[27] prob_meanD[28]
      1.0003617       1.0003584      1.0003552
prob_meanD[29] prob_meanD[30] prob_meanD[31]
      1.0003520       1.0003488      1.0003457
prob_meanD[32] prob_meanD[33] prob_meanD[34]
      1.0003425       1.0003393      1.0003361
prob_meanD[35] prob_meanD[36] prob_meanD[37]
      1.0003330       1.0003299      1.0003268
prob_meanD[38] prob_meanD[39] prob_meanD[40]
      1.0003237       1.0003206      1.0003175
prob_meanD[41] prob_meanD[42] prob_meanD[43]
      1.0003145       1.0003115      1.0003085
prob_meanD[44] prob_meanD[45] prob_meanD[46]
      1.0003055       1.0003026      1.0002997
prob_meanD[47] prob_meanD[48] prob_meanD[49]
      1.0002969       1.0002940      1.0002912
prob_meanD[50] prob_meanD[51] prob_meanD[52]
      1.0002885       1.0002858      1.0002831
prob_meanD[53] prob_meanD[54] prob_meanD[55]
      1.0002805       1.0002779      1.0002754
prob_meanD[56] prob_meanD[57] prob_meanD[58]
      1.0002729       1.0002705      1.0002682
```

prob_meanD[59]	prob_meanD[60]	prob_meanD[61]
1.0002659	1.0002637	1.0002615
prob_meanD[62]	prob_meanD[63]	prob_meanD[64]
1.0002595	1.0002575	1.0002556
prob_meanD[65]	prob_meanD[66]	prob_meanD[67]
1.0002537	1.0002520	1.0002504
prob_meanD[68]	prob_meanD[69]	prob_meanD[70]
1.0002488	1.0002474	1.0002461
prob_meanD[71]	prob_meanD[72]	prob_meanD[73]
1.0002449	1.0002438	1.0002429
prob_meanD[74]	prob_meanD[75]	prob_meanD[76]
1.0002421	1.0002414	1.0002410
prob_meanD[77]	prob_meanD[78]	prob_meanD[79]
1.0002407	1.0002407	1.0002408
prob_meanD[80]	prob_meanD[81]	prob_meanD[82]
1.0002413	1.0002420	1.0002431
prob_meanD[83]	prob_meanD[84]	prob_meanD[85]
1.0002446	1.0002464	1.0002485
prob_meanD[86]	prob_meanD[87]	prob_meanD[88]
1.0002505	1.0002508	1.0002434
prob_meanD[89]	prob_meanD[90]	prob_meanD[91]
1.0001985	0.9999629	0.9996921
prob_meanD[92]	prob_meanD[93]	prob_meanD[94]
0.9999239	1.0000483	1.0001042
prob_meanD[95]	prob_meanD[96]	prob_meanD[97]
1.0001340	1.0001522	1.0001644
prob_meanD[98]	prob_meanD[99]	prob_meanD[100]
1.0001733	1.0001801	1.0001856
prob_meanD[101]	prob_meanD[102]	prob_meanD[103]
1.0001902	1.0001942	1.0001977
prob_meanD[104]	prob_meanD[105]	prob_meanD[106]
1.0002010	1.0002040	1.0002068
prob_meanD[107]	prob_meanD[108]	prob_meanD[109]
1.0002096	1.0002123	1.0002149
prob_meanD[110]	prob_meanD[111]	prob_meanD[112]
1.0002175	1.0002200	1.0002226
prob_meanD[113]	prob_meanD[114]	prob_meanD[115]
1.0002251	1.0002277	1.0002303
prob_meanD[116]	prob_meanD[117]	prob_meanD[118]
1.0002329	1.0002355	1.0002382
prob_meanD[119]	prob_meanD[120]	prob_meanD[121]
1.0002409	1.0002436	1.0002464
prob_meanD[122]	prob_meanD[123]	prob_meanD[124]
1.0002491	1.0002520	1.0002548
prob_meanD[125]	prob_meanD[126]	prob_meanD[127]
1.0002577	1.0002606	1.0002635
prob_meanD[128]	prob_meanD[129]	prob_meanD[130]
1.0002665	1.0002695	1.0002725
prob_meanD[131]	prob_meanD[132]	prob_meanD[133]
1.0002755	1.0002786	1.0002817
prob_meanD[134]	prob_meanD[135]	prob_meanD[136]
1.0002848	1.0002880	1.0002911
prob_meanD[137]	prob_meanD[138]	prob_meanD[139]
1.0002943	1.0002975	1.0003007

```
prob_meanD[140] prob_meanD[141] prob_meanD[142]
      1.0003039       1.0003072       1.0003104
prob_meanD[143] prob_meanD[144] prob_meanD[145]
      1.0003137       1.0003170       1.0003203
prob_meanD[146] prob_meanD[147] prob_meanD[148]
      1.0003236       1.0003269       1.0003302
prob_meanD[149] prob_meanD[150] prob_meanD[151]
      1.0003335       1.0003368       1.0003401
prob_meanD[152] prob_meanD[153] prob_meanD[154]
      1.0003435       1.0003468       1.0003501
prob_meanD[155] prob_meanD[156] prob_meanD[157]
      1.0003535       1.0003568       1.0003601
prob_meanD[158] prob_meanD[159] prob_meanD[160]
      1.0003635       1.0003668       1.0003701
prob_meanD[161] prob_meanD[162] prob_meanD[163]
      1.0003734       1.0003768       1.0003801
prob_meanD[164] prob_meanD[165] prob_meanD[166]
      1.0003834       1.0003867       1.0003900
prob_meanD[167] prob_meanD[168] prob_meanD[169]
      1.0003933       1.0003965       1.0003998
prob_meanD[170] prob_meanD[171] prob_meanD[172]
      1.0004031       1.0004064       1.0004096
prob_meanD[173] prob_meanD[174] prob_meanD[175]
      1.0004129       1.0004161       1.0004193
prob_meanD[176] prob_meanD[177] prob_meanD[178]
      1.0004225       1.0004257       1.0004289
prob_meanD[179] prob_meanD[180] prob_meanD[181]
      1.0004321       1.0004353       1.0004385
 prob_meanD_minus_sd[1]  prob_meanD_minus_sd[2]  prob_meanD_minus_sd[3]
              1.0003884               1.0003836               1.0003787
 prob_meanD_minus_sd[4]  prob_meanD_minus_sd[5]  prob_meanD_minus_sd[6]
              1.0003738               1.0003689               1.0003638
 prob_meanD_minus_sd[7]  prob_meanD_minus_sd[8]  prob_meanD_minus_sd[9]
              1.0003587               1.0003536               1.0003484
prob_meanD_minus_sd[10] prob_meanD_minus_sd[11] prob_meanD_minus_sd[12]
              1.0003431               1.0003377               1.0003322
prob_meanD_minus_sd[13] prob_meanD_minus_sd[14] prob_meanD_minus_sd[15]
              1.0003267               1.0003211               1.0003154
prob_meanD_minus_sd[16] prob_meanD_minus_sd[17] prob_meanD_minus_sd[18]
              1.0003096               1.0003037               1.0002978
prob_meanD_minus_sd[19] prob_meanD_minus_sd[20] prob_meanD_minus_sd[21]
              1.0002917               1.0002855               1.0002793
prob_meanD_minus_sd[22] prob_meanD_minus_sd[23] prob_meanD_minus_sd[24]
              1.0002729               1.0002664               1.0002598
prob_meanD_minus_sd[25] prob_meanD_minus_sd[26] prob_meanD_minus_sd[27]
              1.0002530               1.0002461               1.0002391
prob_meanD_minus_sd[28] prob_meanD_minus_sd[29] prob_meanD_minus_sd[30]
              1.0002319               1.0002246               1.0002172
prob_meanD_minus_sd[31] prob_meanD_minus_sd[32] prob_meanD_minus_sd[33]
              1.0002096               1.0002018               1.0001938
prob_meanD_minus_sd[34] prob_meanD_minus_sd[35] prob_meanD_minus_sd[36]
              1.0001856               1.0001773               1.0001688
prob_meanD_minus_sd[37] prob_meanD_minus_sd[38] prob_meanD_minus_sd[39]
              1.0001600               1.0001511               1.0001419
```

```
 prob_meanD_minus_sd[40]  prob_meanD_minus_sd[41]  prob_meanD_minus_sd[42]
               1.0001325                1.0001228                1.0001129
 prob_meanD_minus_sd[43]  prob_meanD_minus_sd[44]  prob_meanD_minus_sd[45]
               1.0001028                1.0000924                1.0000818
 prob_meanD_minus_sd[46]  prob_meanD_minus_sd[47]  prob_meanD_minus_sd[48]
               1.0000709                1.0000597                1.0000483
 prob_meanD_minus_sd[49]  prob_meanD_minus_sd[50]  prob_meanD_minus_sd[51]
               1.0000366                1.0000247                1.0000125
 prob_meanD_minus_sd[52]  prob_meanD_minus_sd[53]  prob_meanD_minus_sd[54]
               1.0000000                0.9999873                0.9999744
 prob_meanD_minus_sd[55]  prob_meanD_minus_sd[56]  prob_meanD_minus_sd[57]
               0.9999613                0.9999480                0.9999346
 prob_meanD_minus_sd[58]  prob_meanD_minus_sd[59]  prob_meanD_minus_sd[60]
               0.9999210                0.9999073                0.9998935
 prob_meanD_minus_sd[61]  prob_meanD_minus_sd[62]  prob_meanD_minus_sd[63]
               0.9998798                0.9998660                0.9998524
 prob_meanD_minus_sd[64]  prob_meanD_minus_sd[65]  prob_meanD_minus_sd[66]
               0.9998388                0.9998255                0.9998124
 prob_meanD_minus_sd[67]  prob_meanD_minus_sd[68]  prob_meanD_minus_sd[69]
               0.9997996                0.9997872                0.9997752
 prob_meanD_minus_sd[70]  prob_meanD_minus_sd[71]  prob_meanD_minus_sd[72]
               0.9997637                0.9997528                0.9997425
 prob_meanD_minus_sd[73]  prob_meanD_minus_sd[74]  prob_meanD_minus_sd[75]
               0.9997329                0.9997241                0.9997160
 prob_meanD_minus_sd[76]  prob_meanD_minus_sd[77]  prob_meanD_minus_sd[78]
               0.9997087                0.9997024                0.9996969
 prob_meanD_minus_sd[79]  prob_meanD_minus_sd[80]  prob_meanD_minus_sd[81]
               0.9996923                0.9996887                0.9996860
 prob_meanD_minus_sd[82]  prob_meanD_minus_sd[83]  prob_meanD_minus_sd[84]
               0.9996842                0.9996833                0.9996833
 prob_meanD_minus_sd[85]  prob_meanD_minus_sd[86]  prob_meanD_minus_sd[87]
               0.9996843                0.9996860                0.9996886
 prob_meanD_minus_sd[88]  prob_meanD_minus_sd[89]  prob_meanD_minus_sd[90]
               0.9996920                0.9996960                0.9997008
 prob_meanD_minus_sd[91]  prob_meanD_minus_sd[92]  prob_meanD_minus_sd[93]
               0.9997062                0.9997122                0.9997188
 prob_meanD_minus_sd[94]  prob_meanD_minus_sd[95]  prob_meanD_minus_sd[96]
               0.9997259                0.9997334                0.9997413
 prob_meanD_minus_sd[97]  prob_meanD_minus_sd[98]  prob_meanD_minus_sd[99]
               0.9997495                0.9997581                0.9997670
prob_meanD_minus_sd[100] prob_meanD_minus_sd[101] prob_meanD_minus_sd[102]
               0.9997761                0.9997854                0.9997949
prob_meanD_minus_sd[103] prob_meanD_minus_sd[104] prob_meanD_minus_sd[105]
               0.9998045                0.9998142                0.9998240
prob_meanD_minus_sd[106] prob_meanD_minus_sd[107] prob_meanD_minus_sd[108]
               0.9998338                0.9998437                0.9998536
prob_meanD_minus_sd[109] prob_meanD_minus_sd[110] prob_meanD_minus_sd[111]
               0.9998635                0.9998734                0.9998832
prob_meanD_minus_sd[112] prob_meanD_minus_sd[113] prob_meanD_minus_sd[114]
               0.9998930                0.9999028                0.9999124
prob_meanD_minus_sd[115] prob_meanD_minus_sd[116] prob_meanD_minus_sd[117]
               0.9999220                0.9999315                0.9999410
prob_meanD_minus_sd[118] prob_meanD_minus_sd[119] prob_meanD_minus_sd[120]
               0.9999503                0.9999595                0.9999687
```

```
prob_meanD_minus_sd[121] prob_meanD_minus_sd[122] prob_meanD_minus_sd[123]
               0.9999777                0.9999866                0.9999954
prob_meanD_minus_sd[124] prob_meanD_minus_sd[125] prob_meanD_minus_sd[126]
               1.0000041                1.0000127                1.0000211
prob_meanD_minus_sd[127] prob_meanD_minus_sd[128] prob_meanD_minus_sd[129]
               1.0000295                1.0000378                1.0000459
prob_meanD_minus_sd[130] prob_meanD_minus_sd[131] prob_meanD_minus_sd[132]
               1.0000539                1.0000618                1.0000696
prob_meanD_minus_sd[133] prob_meanD_minus_sd[134] prob_meanD_minus_sd[135]
               1.0000773                1.0000849                1.0000924
prob_meanD_minus_sd[136] prob_meanD_minus_sd[137] prob_meanD_minus_sd[138]
               1.0000998                1.0001071                1.0001143
prob_meanD_minus_sd[139] prob_meanD_minus_sd[140] prob_meanD_minus_sd[141]
               1.0001214                1.0001284                1.0001353
prob_meanD_minus_sd[142] prob_meanD_minus_sd[143] prob_meanD_minus_sd[144]
               1.0001421                1.0001488                1.0001554
prob_meanD_minus_sd[145] prob_meanD_minus_sd[146] prob_meanD_minus_sd[147]
               1.0001619                1.0001684                1.0001747
prob_meanD_minus_sd[148] prob_meanD_minus_sd[149] prob_meanD_minus_sd[150]
               1.0001810                1.0001872                1.0001933
prob_meanD_minus_sd[151] prob_meanD_minus_sd[152] prob_meanD_minus_sd[153]
               1.0001994                1.0002054                1.0002113
prob_meanD_minus_sd[154] prob_meanD_minus_sd[155] prob_meanD_minus_sd[156]
               1.0002171                1.0002228                1.0002285
prob_meanD_minus_sd[157] prob_meanD_minus_sd[158] prob_meanD_minus_sd[159]
               1.0002341                1.0002397                1.0002452
prob_meanD_minus_sd[160] prob_meanD_minus_sd[161] prob_meanD_minus_sd[162]
               1.0002506                1.0002560                1.0002613
prob_meanD_minus_sd[163] prob_meanD_minus_sd[164] prob_meanD_minus_sd[165]
               1.0002665                1.0002717                1.0002768
prob_meanD_minus_sd[166] prob_meanD_minus_sd[167] prob_meanD_minus_sd[168]
               1.0002819                1.0002869                1.0002919
prob_meanD_minus_sd[169] prob_meanD_minus_sd[170] prob_meanD_minus_sd[171]
               1.0002968                1.0003016                1.0003065
prob_meanD_minus_sd[172] prob_meanD_minus_sd[173] prob_meanD_minus_sd[174]
               1.0003112                1.0003159                1.0003206
prob_meanD_minus_sd[175] prob_meanD_minus_sd[176] prob_meanD_minus_sd[177]
               1.0003252                1.0003298                1.0003343
prob_meanD_minus_sd[178] prob_meanD_minus_sd[179] prob_meanD_minus_sd[180]
               1.0003388                1.0003433                1.0003477
prob_meanD_minus_sd[181]    prob_meanD_plus_sd[1]    prob_meanD_plus_sd[2]
               1.0003521                1.0003572                1.0003529
   prob_meanD_plus_sd[3]    prob_meanD_plus_sd[4]    prob_meanD_plus_sd[5]
               1.0003485                1.0003441                1.0003397
   prob_meanD_plus_sd[6]    prob_meanD_plus_sd[7]    prob_meanD_plus_sd[8]
               1.0003352                1.0003307                1.0003261
   prob_meanD_plus_sd[9]   prob_meanD_plus_sd[10]   prob_meanD_plus_sd[11]
               1.0003215                1.0003169                1.0003121
  prob_meanD_plus_sd[12]   prob_meanD_plus_sd[13]   prob_meanD_plus_sd[14]
               1.0003074                1.0003026                1.0002977
  prob_meanD_plus_sd[15]   prob_meanD_plus_sd[16]   prob_meanD_plus_sd[17]
               1.0002928                1.0002879                1.0002828
  prob_meanD_plus_sd[18]   prob_meanD_plus_sd[19]   prob_meanD_plus_sd[20]
               1.0002778                1.0002726                1.0002675
```

```
prob_meanD_plus_sd[21]   prob_meanD_plus_sd[22]   prob_meanD_plus_sd[23]
             1.0002622                1.0002569                1.0002516
prob_meanD_plus_sd[24]   prob_meanD_plus_sd[25]   prob_meanD_plus_sd[26]
             1.0002461                1.0002406                1.0002351
prob_meanD_plus_sd[27]   prob_meanD_plus_sd[28]   prob_meanD_plus_sd[29]
             1.0002295                1.0002238                1.0002180
prob_meanD_plus_sd[30]   prob_meanD_plus_sd[31]   prob_meanD_plus_sd[32]
             1.0002122                1.0002062                1.0002003
prob_meanD_plus_sd[33]   prob_meanD_plus_sd[34]   prob_meanD_plus_sd[35]
             1.0001942                1.0001880                1.0001818
prob_meanD_plus_sd[36]   prob_meanD_plus_sd[37]   prob_meanD_plus_sd[38]
             1.0001755                1.0001691                1.0001626
prob_meanD_plus_sd[39]   prob_meanD_plus_sd[40]   prob_meanD_plus_sd[41]
             1.0001561                1.0001494                1.0001426
prob_meanD_plus_sd[42]   prob_meanD_plus_sd[43]   prob_meanD_plus_sd[44]
             1.0001358                1.0001288                1.0001218
prob_meanD_plus_sd[45]   prob_meanD_plus_sd[46]   prob_meanD_plus_sd[47]
             1.0001146                1.0001074                1.0001000
prob_meanD_plus_sd[48]   prob_meanD_plus_sd[49]   prob_meanD_plus_sd[50]
             1.0000925                1.0000850                1.0000773
prob_meanD_plus_sd[51]   prob_meanD_plus_sd[52]   prob_meanD_plus_sd[53]
             1.0000695                1.0000615                1.0000535
prob_meanD_plus_sd[54]   prob_meanD_plus_sd[55]   prob_meanD_plus_sd[56]
             1.0000454                1.0000371                1.0000287
prob_meanD_plus_sd[57]   prob_meanD_plus_sd[58]   prob_meanD_plus_sd[59]
             1.0000202                1.0000115                1.0000028
prob_meanD_plus_sd[60]   prob_meanD_plus_sd[61]   prob_meanD_plus_sd[62]
             0.9999939                0.9999849                0.9999757
prob_meanD_plus_sd[63]   prob_meanD_plus_sd[64]   prob_meanD_plus_sd[65]
             0.9999665                0.9999571                0.9999476
prob_meanD_plus_sd[66]   prob_meanD_plus_sd[67]   prob_meanD_plus_sd[68]
             0.9999381                0.9999284                0.9999186
prob_meanD_plus_sd[69]   prob_meanD_plus_sd[70]   prob_meanD_plus_sd[71]
             0.9999087                0.9998987                0.9998886
prob_meanD_plus_sd[72]   prob_meanD_plus_sd[73]   prob_meanD_plus_sd[74]
             0.9998785                0.9998684                0.9998581
prob_meanD_plus_sd[75]   prob_meanD_plus_sd[76]   prob_meanD_plus_sd[77]
             0.9998479                0.9998377                0.9998275
prob_meanD_plus_sd[78]   prob_meanD_plus_sd[79]   prob_meanD_plus_sd[80]
             0.9998173                0.9998072                0.9997972
prob_meanD_plus_sd[81]   prob_meanD_plus_sd[82]   prob_meanD_plus_sd[83]
             0.9997873                0.9997776                0.9997681
prob_meanD_plus_sd[84]   prob_meanD_plus_sd[85]   prob_meanD_plus_sd[86]
             0.9997588                0.9997498                0.9997412
prob_meanD_plus_sd[87]   prob_meanD_plus_sd[88]   prob_meanD_plus_sd[89]
             0.9997329                0.9997251                0.9997178
prob_meanD_plus_sd[90]   prob_meanD_plus_sd[91]   prob_meanD_plus_sd[92]
             0.9997110                0.9997048                0.9996992
prob_meanD_plus_sd[93]   prob_meanD_plus_sd[94]   prob_meanD_plus_sd[95]
             0.9996944                0.9996904                0.9996871
prob_meanD_plus_sd[96]   prob_meanD_plus_sd[97]   prob_meanD_plus_sd[98]
             0.9996848                0.9996833                0.9996828
prob_meanD_plus_sd[99]  prob_meanD_plus_sd[100]  prob_meanD_plus_sd[101]
             0.9996833                0.9996848                0.9996873
```

```
prob_meanD_plus_sd[102]  prob_meanD_plus_sd[103]  prob_meanD_plus_sd[104]
              0.9996908                0.9996954                0.9997010
prob_meanD_plus_sd[105]  prob_meanD_plus_sd[106]  prob_meanD_plus_sd[107]
              0.9997076                0.9997151                0.9997236
prob_meanD_plus_sd[108]  prob_meanD_plus_sd[109]  prob_meanD_plus_sd[110]
              0.9997330                0.9997431                0.9997540
prob_meanD_plus_sd[111]  prob_meanD_plus_sd[112]  prob_meanD_plus_sd[113]
              0.9997657                0.9997779                0.9997906
prob_meanD_plus_sd[114]  prob_meanD_plus_sd[115]  prob_meanD_plus_sd[116]
              0.9998039                0.9998175                0.9998314
prob_meanD_plus_sd[117]  prob_meanD_plus_sd[118]  prob_meanD_plus_sd[119]
              0.9998455                0.9998599                0.9998743
prob_meanD_plus_sd[120]  prob_meanD_plus_sd[121]  prob_meanD_plus_sd[122]
              0.9998888                0.9999032                0.9999176
prob_meanD_plus_sd[123]  prob_meanD_plus_sd[124]  prob_meanD_plus_sd[125]
              0.9999319                0.9999460                0.9999599
prob_meanD_plus_sd[126]  prob_meanD_plus_sd[127]  prob_meanD_plus_sd[128]
              0.9999737                0.9999872                1.0000004
prob_meanD_plus_sd[129]  prob_meanD_plus_sd[130]  prob_meanD_plus_sd[131]
              1.0000134                1.0000260                1.0000384
prob_meanD_plus_sd[132]  prob_meanD_plus_sd[133]  prob_meanD_plus_sd[134]
              1.0000505                1.0000622                1.0000737
prob_meanD_plus_sd[135]  prob_meanD_plus_sd[136]  prob_meanD_plus_sd[137]
              1.0000849                1.0000958                1.0001063
prob_meanD_plus_sd[138]  prob_meanD_plus_sd[139]  prob_meanD_plus_sd[140]
              1.0001166                1.0001267                1.0001364
prob_meanD_plus_sd[141]  prob_meanD_plus_sd[142]  prob_meanD_plus_sd[143]
              1.0001459                1.0001551                1.0001641
prob_meanD_plus_sd[144]  prob_meanD_plus_sd[145]  prob_meanD_plus_sd[146]
              1.0001729                1.0001814                1.0001897
prob_meanD_plus_sd[147]  prob_meanD_plus_sd[148]  prob_meanD_plus_sd[149]
              1.0001979                1.0002058                1.0002135
prob_meanD_plus_sd[150]  prob_meanD_plus_sd[151]  prob_meanD_plus_sd[152]
              1.0002210                1.0002284                1.0002356
prob_meanD_plus_sd[153]  prob_meanD_plus_sd[154]  prob_meanD_plus_sd[155]
              1.0002427                1.0002496                1.0002564
prob_meanD_plus_sd[156]  prob_meanD_plus_sd[157]  prob_meanD_plus_sd[158]
              1.0002630                1.0002695                1.0002759
prob_meanD_plus_sd[159]  prob_meanD_plus_sd[160]  prob_meanD_plus_sd[161]
              1.0002821                1.0002882                1.0002943
prob_meanD_plus_sd[162]  prob_meanD_plus_sd[163]  prob_meanD_plus_sd[164]
              1.0003002                1.0003060                1.0003118
prob_meanD_plus_sd[165]  prob_meanD_plus_sd[166]  prob_meanD_plus_sd[167]
              1.0003174                1.0003230                1.0003285
prob_meanD_plus_sd[168]  prob_meanD_plus_sd[169]  prob_meanD_plus_sd[170]
              1.0003338                1.0003392                1.0003444
prob_meanD_plus_sd[171]  prob_meanD_plus_sd[172]  prob_meanD_plus_sd[173]
              1.0003496                1.0003547                1.0003597
prob_meanD_plus_sd[174]  prob_meanD_plus_sd[175]  prob_meanD_plus_sd[176]
              1.0003647                1.0003696                1.0003744
prob_meanD_plus_sd[177]  prob_meanD_plus_sd[178]  prob_meanD_plus_sd[179]
              1.0003792                1.0003839                1.0003886
prob_meanD_plus_sd[180]  prob_meanD_plus_sd[181]            prob_meanO[1]
              1.0003932                1.0003978                0.9997662
```

prob_mean0[2]	prob_mean0[3]	prob_mean0[4]
0.9997652	0.9997642	0.9997633
prob_mean0[5]	prob_mean0[6]	prob_mean0[7]
0.9997623	0.9997613	0.9997603
prob_mean0[8]	prob_mean0[9]	prob_mean0[10]
0.9997593	0.9997583	0.9997572
prob_mean0[11]	prob_mean0[12]	prob_mean0[13]
0.9997562	0.9997552	0.9997542
prob_mean0[14]	prob_mean0[15]	prob_mean0[16]
0.9997532	0.9997521	0.9997511
prob_mean0[17]	prob_mean0[18]	prob_mean0[19]
0.9997501	0.9997491	0.9997481
prob_mean0[20]	prob_mean0[21]	prob_mean0[22]
0.9997470	0.9997460	0.9997450
prob_mean0[23]	prob_mean0[24]	prob_mean0[25]
0.9997440	0.9997430	0.9997420
prob_mean0[26]	prob_mean0[27]	prob_mean0[28]
0.9997410	0.9997400	0.9997390
prob_mean0[29]	prob_mean0[30]	prob_mean0[31]
0.9997380	0.9997370	0.9997361
prob_mean0[32]	prob_mean0[33]	prob_mean0[34]
0.9997351	0.9997342	0.9997332
prob_mean0[35]	prob_mean0[36]	prob_mean0[37]
0.9997323	0.9997314	0.9997305
prob_mean0[38]	prob_mean0[39]	prob_mean0[40]
0.9997296	0.9997287	0.9997279
prob_mean0[41]	prob_mean0[42]	prob_mean0[43]
0.9997270	0.9997262	0.9997254
prob_mean0[44]	prob_mean0[45]	prob_mean0[46]
0.9997246	0.9997238	0.9997230
prob_mean0[47]	prob_mean0[48]	prob_mean0[49]
0.9997223	0.9997216	0.9997209
prob_mean0[50]	prob_mean0[51]	prob_mean0[52]
0.9997202	0.9997196	0.9997190
prob_mean0[53]	prob_mean0[54]	prob_mean0[55]
0.9997184	0.9997178	0.9997173
prob_mean0[56]	prob_mean0[57]	prob_mean0[58]
0.9997168	0.9997163	0.9997159
prob_mean0[59]	prob_mean0[60]	prob_mean0[61]
0.9997155	0.9997151	0.9997148
prob_mean0[62]	prob_mean0[63]	prob_mean0[64]
0.9997145	0.9997143	0.9997142
prob_mean0[65]	prob_mean0[66]	prob_mean0[67]
0.9997141	0.9997140	0.9997141
prob_mean0[68]	prob_mean0[69]	prob_mean0[70]
0.9997142	0.9997144	0.9997147
prob_mean0[71]	prob_mean0[72]	prob_mean0[73]
0.9997151	0.9997156	0.9997163
prob_mean0[74]	prob_mean0[75]	prob_mean0[76]
0.9997172	0.9997182	0.9997196
prob_mean0[77]	prob_mean0[78]	prob_mean0[79]
0.9997212	0.9997231	0.9997256
prob_mean0[80]	prob_mean0[81]	prob_mean0[82]
0.9997286	0.9997324	0.9997372

prob_mean0[83]	prob_mean0[84]	prob_mean0[85]
0.9997435	0.9997519	0.9997635
prob_mean0[86]	prob_mean0[87]	prob_mean0[88]
0.9997800	0.9998050	0.9998453
prob_mean0[89]	prob_mean0[90]	prob_mean0[91]
0.9999149	1.0000403	1.0002153
prob_mean0[92]	prob_mean0[93]	prob_mean0[94]
1.0001886	0.9999415	0.9997932
prob_mean0[95]	prob_mean0[96]	prob_mean0[97]
0.9997306	0.9997035	0.9996908
prob_mean0[98]	prob_mean0[99]	prob_mean0[100]
0.9996845	0.9996814	0.9996799
prob_mean0[101]	prob_mean0[102]	prob_mean0[103]
0.9996793	0.9996792	0.9996794
prob_mean0[104]	prob_mean0[105]	prob_mean0[106]
0.9996798	0.9996802	0.9996808
prob_mean0[107]	prob_mean0[108]	prob_mean0[109]
0.9996814	0.9996820	0.9996827
prob_mean0[110]	prob_mean0[111]	prob_mean0[112]
0.9996833	0.9996840	0.9996846
prob_mean0[113]	prob_mean0[114]	prob_mean0[115]
0.9996853	0.9996860	0.9996867
prob_mean0[116]	prob_mean0[117]	prob_mean0[118]
0.9996874	0.9996881	0.9996888
prob_mean0[119]	prob_mean0[120]	prob_mean0[121]
0.9996895	0.9996902	0.9996910
prob_mean0[122]	prob_mean0[123]	prob_mean0[124]
0.9996917	0.9996925	0.9996932
prob_mean0[125]	prob_mean0[126]	prob_mean0[127]
0.9996940	0.9996948	0.9996956
prob_mean0[128]	prob_mean0[129]	prob_mean0[130]
0.9996964	0.9996972	0.9996980
prob_mean0[131]	prob_mean0[132]	prob_mean0[133]
0.9996988	0.9996997	0.9997005
prob_mean0[134]	prob_mean0[135]	prob_mean0[136]
0.9997014	0.9997023	0.9997032
prob_mean0[137]	prob_mean0[138]	prob_mean0[139]
0.9997041	0.9997050	0.9997059
prob_mean0[140]	prob_mean0[141]	prob_mean0[142]
0.9997069	0.9997078	0.9997088
prob_mean0[143]	prob_mean0[144]	prob_mean0[145]
0.9997097	0.9997107	0.9997117
prob_mean0[146]	prob_mean0[147]	prob_mean0[148]
0.9997127	0.9997137	0.9997147
prob_mean0[149]	prob_mean0[150]	prob_mean0[151]
0.9997157	0.9997167	0.9997177
prob_mean0[152]	prob_mean0[153]	prob_mean0[154]
0.9997188	0.9997198	0.9997209
prob_mean0[155]	prob_mean0[156]	prob_mean0[157]
0.9997219	0.9997230	0.9997240
prob_mean0[158]	prob_mean0[159]	prob_mean0[160]
0.9997251	0.9997262	0.9997272
prob_mean0[161]	prob_mean0[162]	prob_mean0[163]
0.9997283	0.9997294	0.9997305

```
        prob_mean0[164]         prob_mean0[165]         prob_mean0[166]
              0.9997315               0.9997326               0.9997337
        prob_mean0[167]         prob_mean0[168]         prob_mean0[169]
              0.9997348               0.9997358               0.9997369
        prob_mean0[170]         prob_mean0[171]         prob_mean0[172]
              0.9997380               0.9997390               0.9997401
        prob_mean0[173]         prob_mean0[174]         prob_mean0[175]
              0.9997412               0.9997422               0.9997433
        prob_mean0[176]         prob_mean0[177]         prob_mean0[178]
              0.9997443               0.9997454               0.9997464
        prob_mean0[179]         prob_mean0[180]         prob_mean0[181]
              0.9997475               0.9997485               0.9997495
 prob_mean0_minus_sd[1]  prob_mean0_minus_sd[2]  prob_mean0_minus_sd[3]
              0.9997145               0.9997161               0.9997177
 prob_mean0_minus_sd[4]  prob_mean0_minus_sd[5]  prob_mean0_minus_sd[6]
              0.9997195               0.9997213               0.9997232
 prob_mean0_minus_sd[7]  prob_mean0_minus_sd[8]  prob_mean0_minus_sd[9]
              0.9997252               0.9997274               0.9997296
prob_mean0_minus_sd[10] prob_mean0_minus_sd[11] prob_mean0_minus_sd[12]
              0.9997319               0.9997343               0.9997369
prob_mean0_minus_sd[13] prob_mean0_minus_sd[14] prob_mean0_minus_sd[15]
              0.9997396               0.9997424               0.9997453
prob_mean0_minus_sd[16] prob_mean0_minus_sd[17] prob_mean0_minus_sd[18]
              0.9997483               0.9997515               0.9997549
prob_mean0_minus_sd[19] prob_mean0_minus_sd[20] prob_mean0_minus_sd[21]
              0.9997583               0.9997620               0.9997658
prob_mean0_minus_sd[22] prob_mean0_minus_sd[23] prob_mean0_minus_sd[24]
              0.9997698               0.9997739               0.9997782
prob_mean0_minus_sd[25] prob_mean0_minus_sd[26] prob_mean0_minus_sd[27]
              0.9997827               0.9997874               0.9997923
prob_mean0_minus_sd[28] prob_mean0_minus_sd[29] prob_mean0_minus_sd[30]
              0.9997974               0.9998027               0.9998083
prob_mean0_minus_sd[31] prob_mean0_minus_sd[32] prob_mean0_minus_sd[33]
              0.9998140               0.9998200               0.9998263
prob_mean0_minus_sd[34] prob_mean0_minus_sd[35] prob_mean0_minus_sd[36]
              0.9998328               0.9998395               0.9998465
prob_mean0_minus_sd[37] prob_mean0_minus_sd[38] prob_mean0_minus_sd[39]
              0.9998538               0.9998614               0.9998693
prob_mean0_minus_sd[40] prob_mean0_minus_sd[41] prob_mean0_minus_sd[42]
              0.9998775               0.9998859               0.9998947
prob_mean0_minus_sd[43] prob_mean0_minus_sd[44] prob_mean0_minus_sd[45]
              0.9999038               0.9999132               0.9999230
prob_mean0_minus_sd[46] prob_mean0_minus_sd[47] prob_mean0_minus_sd[48]
              0.9999330               0.9999434               0.9999541
prob_mean0_minus_sd[49] prob_mean0_minus_sd[50] prob_mean0_minus_sd[51]
              0.9999651               0.9999764               0.9999881
prob_mean0_minus_sd[52] prob_mean0_minus_sd[53] prob_mean0_minus_sd[54]
              1.0000000               1.0000122               1.0000246
prob_mean0_minus_sd[55] prob_mean0_minus_sd[56] prob_mean0_minus_sd[57]
              1.0000373               1.0000502               1.0000633
prob_mean0_minus_sd[58] prob_mean0_minus_sd[59] prob_mean0_minus_sd[60]
              1.0000766               1.0000899               1.0001033
prob_mean0_minus_sd[61] prob_mean0_minus_sd[62] prob_mean0_minus_sd[63]
              1.0001168               1.0001302               1.0001435
```

```
 prob_mean0_minus_sd[64]  prob_mean0_minus_sd[65]  prob_mean0_minus_sd[66]
               1.0001567                1.0001697                1.0001824
 prob_mean0_minus_sd[67]  prob_mean0_minus_sd[68]  prob_mean0_minus_sd[69]
               1.0001948                1.0002068                1.0002184
 prob_mean0_minus_sd[70]  prob_mean0_minus_sd[71]  prob_mean0_minus_sd[72]
               1.0002294                1.0002398                1.0002496
 prob_mean0_minus_sd[73]  prob_mean0_minus_sd[74]  prob_mean0_minus_sd[75]
               1.0002587                1.0002670                1.0002745
 prob_mean0_minus_sd[76]  prob_mean0_minus_sd[77]  prob_mean0_minus_sd[78]
               1.0002813                1.0002871                1.0002921
 prob_mean0_minus_sd[79]  prob_mean0_minus_sd[80]  prob_mean0_minus_sd[81]
               1.0002962                1.0002994                1.0003018
 prob_mean0_minus_sd[82]  prob_mean0_minus_sd[83]  prob_mean0_minus_sd[84]
               1.0003032                1.0003038                1.0003036
 prob_mean0_minus_sd[85]  prob_mean0_minus_sd[86]  prob_mean0_minus_sd[87]
               1.0003027                1.0003010                1.0002985
 prob_mean0_minus_sd[88]  prob_mean0_minus_sd[89]  prob_mean0_minus_sd[90]
               1.0002955                1.0002918                1.0002876
 prob_mean0_minus_sd[91]  prob_mean0_minus_sd[92]  prob_mean0_minus_sd[93]
               1.0002829                1.0002778                1.0002722
 prob_mean0_minus_sd[94]  prob_mean0_minus_sd[95]  prob_mean0_minus_sd[96]
               1.0002664                1.0002602                1.0002538
 prob_mean0_minus_sd[97]  prob_mean0_minus_sd[98]  prob_mean0_minus_sd[99]
               1.0002472                1.0002405                1.0002336
prob_mean0_minus_sd[100] prob_mean0_minus_sd[101] prob_mean0_minus_sd[102]
               1.0002266                1.0002196                1.0002126
prob_mean0_minus_sd[103] prob_mean0_minus_sd[104] prob_mean0_minus_sd[105]
               1.0002056                1.0001986                1.0001917
prob_mean0_minus_sd[106] prob_mean0_minus_sd[107] prob_mean0_minus_sd[108]
               1.0001848                1.0001780                1.0001714
prob_mean0_minus_sd[109] prob_mean0_minus_sd[110] prob_mean0_minus_sd[111]
               1.0001648                1.0001584                1.0001521
prob_mean0_minus_sd[112] prob_mean0_minus_sd[113] prob_mean0_minus_sd[114]
               1.0001459                1.0001399                1.0001341
prob_mean0_minus_sd[115] prob_mean0_minus_sd[116] prob_mean0_minus_sd[117]
               1.0001284                1.0001228                1.0001175
prob_mean0_minus_sd[118] prob_mean0_minus_sd[119] prob_mean0_minus_sd[120]
               1.0001122                1.0001072                1.0001022
prob_mean0_minus_sd[121] prob_mean0_minus_sd[122] prob_mean0_minus_sd[123]
               1.0000975                1.0000929                1.0000884
prob_mean0_minus_sd[124] prob_mean0_minus_sd[125] prob_mean0_minus_sd[126]
               1.0000841                1.0000800                1.0000760
prob_mean0_minus_sd[127] prob_mean0_minus_sd[128] prob_mean0_minus_sd[129]
               1.0000721                1.0000684                1.0000648
prob_mean0_minus_sd[130] prob_mean0_minus_sd[131] prob_mean0_minus_sd[132]
               1.0000613                1.0000580                1.0000547
prob_mean0_minus_sd[133] prob_mean0_minus_sd[134] prob_mean0_minus_sd[135]
               1.0000516                1.0000487                1.0000458
prob_mean0_minus_sd[136] prob_mean0_minus_sd[137] prob_mean0_minus_sd[138]
               1.0000430                1.0000404                1.0000378
prob_mean0_minus_sd[139] prob_mean0_minus_sd[140] prob_mean0_minus_sd[141]
               1.0000354                1.0000330                1.0000307
prob_mean0_minus_sd[142] prob_mean0_minus_sd[143] prob_mean0_minus_sd[144]
               1.0000286                1.0000265                1.0000244
```

```
prob_mean0_minus_sd[145] prob_mean0_minus_sd[146] prob_mean0_minus_sd[147]
               1.0000225                1.0000207                1.0000189
prob_mean0_minus_sd[148] prob_mean0_minus_sd[149] prob_mean0_minus_sd[150]
               1.0000172                1.0000155                1.0000139
prob_mean0_minus_sd[151] prob_mean0_minus_sd[152] prob_mean0_minus_sd[153]
               1.0000124                1.0000110                1.0000096
prob_mean0_minus_sd[154] prob_mean0_minus_sd[155] prob_mean0_minus_sd[156]
               1.0000082                1.0000069                1.0000057
prob_mean0_minus_sd[157] prob_mean0_minus_sd[158] prob_mean0_minus_sd[159]
               1.0000045                1.0000033                1.0000022
prob_mean0_minus_sd[160] prob_mean0_minus_sd[161] prob_mean0_minus_sd[162]
               1.0000012                1.0000002                0.9999992
prob_mean0_minus_sd[163] prob_mean0_minus_sd[164] prob_mean0_minus_sd[165]
               0.9999983                0.9999973                0.9999965
prob_mean0_minus_sd[166] prob_mean0_minus_sd[167] prob_mean0_minus_sd[168]
               0.9999956                0.9999948                0.9999941
prob_mean0_minus_sd[169] prob_mean0_minus_sd[170] prob_mean0_minus_sd[171]
               0.9999933                0.9999926                0.9999919
prob_mean0_minus_sd[172] prob_mean0_minus_sd[173] prob_mean0_minus_sd[174]
               0.9999912                0.9999906                0.9999900
prob_mean0_minus_sd[175] prob_mean0_minus_sd[176] prob_mean0_minus_sd[177]
               0.9999894                0.9999888                0.9999882
prob_mean0_minus_sd[178] prob_mean0_minus_sd[179] prob_mean0_minus_sd[180]
               0.9999877                0.9999871                0.9999866
prob_mean0_minus_sd[181]    prob_mean0_plus_sd[1]    prob_mean0_plus_sd[2]
               0.9999861                1.0000082                1.0000088
   prob_mean0_plus_sd[3]    prob_mean0_plus_sd[4]    prob_mean0_plus_sd[5]
               1.0000094                1.0000101                1.0000108
   prob_mean0_plus_sd[6]    prob_mean0_plus_sd[7]    prob_mean0_plus_sd[8]
               1.0000115                1.0000122                1.0000130
   prob_mean0_plus_sd[9]   prob_mean0_plus_sd[10]   prob_mean0_plus_sd[11]
               1.0000137                1.0000145                1.0000154
  prob_mean0_plus_sd[12]   prob_mean0_plus_sd[13]   prob_mean0_plus_sd[14]
               1.0000162                1.0000171                1.0000180
  prob_mean0_plus_sd[15]   prob_mean0_plus_sd[16]   prob_mean0_plus_sd[17]
               1.0000189                1.0000198                1.0000208
  prob_mean0_plus_sd[18]   prob_mean0_plus_sd[19]   prob_mean0_plus_sd[20]
               1.0000218                1.0000229                1.0000240
  prob_mean0_plus_sd[21]   prob_mean0_plus_sd[22]   prob_mean0_plus_sd[23]
               1.0000251                1.0000263                1.0000275
  prob_mean0_plus_sd[24]   prob_mean0_plus_sd[25]   prob_mean0_plus_sd[26]
               1.0000287                1.0000300                1.0000314
  prob_mean0_plus_sd[27]   prob_mean0_plus_sd[28]   prob_mean0_plus_sd[29]
               1.0000327                1.0000342                1.0000357
  prob_mean0_plus_sd[30]   prob_mean0_plus_sd[31]   prob_mean0_plus_sd[32]
               1.0000372                1.0000388                1.0000405
  prob_mean0_plus_sd[33]   prob_mean0_plus_sd[34]   prob_mean0_plus_sd[35]
               1.0000422                1.0000440                1.0000458
  prob_mean0_plus_sd[36]   prob_mean0_plus_sd[37]   prob_mean0_plus_sd[38]
               1.0000478                1.0000497                1.0000518
  prob_mean0_plus_sd[39]   prob_mean0_plus_sd[40]   prob_mean0_plus_sd[41]
               1.0000539                1.0000562                1.0000584
  prob_mean0_plus_sd[42]   prob_mean0_plus_sd[43]   prob_mean0_plus_sd[44]
               1.0000608                1.0000633                1.0000658
```

```
prob_mean0_plus_sd[45]   prob_mean0_plus_sd[46]   prob_mean0_plus_sd[47]
             1.0000685                1.0000712                1.0000741
prob_mean0_plus_sd[48]   prob_mean0_plus_sd[49]   prob_mean0_plus_sd[50]
             1.0000770                1.0000800                1.0000832
prob_mean0_plus_sd[51]   prob_mean0_plus_sd[52]   prob_mean0_plus_sd[53]
             1.0000864                1.0000898                1.0000932
prob_mean0_plus_sd[54]   prob_mean0_plus_sd[55]   prob_mean0_plus_sd[56]
             1.0000968                1.0001005                1.0001044
prob_mean0_plus_sd[57]   prob_mean0_plus_sd[58]   prob_mean0_plus_sd[59]
             1.0001083                1.0001124                1.0001166
prob_mean0_plus_sd[60]   prob_mean0_plus_sd[61]   prob_mean0_plus_sd[62]
             1.0001209                1.0001254                1.0001300
prob_mean0_plus_sd[63]   prob_mean0_plus_sd[64]   prob_mean0_plus_sd[65]
             1.0001347                1.0001396                1.0001446
prob_mean0_plus_sd[66]   prob_mean0_plus_sd[67]   prob_mean0_plus_sd[68]
             1.0001497                1.0001549                1.0001603
prob_mean0_plus_sd[69]   prob_mean0_plus_sd[70]   prob_mean0_plus_sd[71]
             1.0001658                1.0001714                1.0001771
prob_mean0_plus_sd[72]   prob_mean0_plus_sd[73]   prob_mean0_plus_sd[74]
             1.0001829                1.0001888                1.0001948
prob_mean0_plus_sd[75]   prob_mean0_plus_sd[76]   prob_mean0_plus_sd[77]
             1.0002009                1.0002071                1.0002133
prob_mean0_plus_sd[78]   prob_mean0_plus_sd[79]   prob_mean0_plus_sd[80]
             1.0002195                1.0002258                1.0002321
prob_mean0_plus_sd[81]   prob_mean0_plus_sd[82]   prob_mean0_plus_sd[83]
             1.0002383                1.0002445                1.0002507
prob_mean0_plus_sd[84]   prob_mean0_plus_sd[85]   prob_mean0_plus_sd[86]
             1.0002568                1.0002627                1.0002685
prob_mean0_plus_sd[87]   prob_mean0_plus_sd[88]   prob_mean0_plus_sd[89]
             1.0002741                1.0002795                1.0002846
prob_mean0_plus_sd[90]   prob_mean0_plus_sd[91]   prob_mean0_plus_sd[92]
             1.0002895                1.0002940                1.0002981
prob_mean0_plus_sd[93]   prob_mean0_plus_sd[94]   prob_mean0_plus_sd[95]
             1.0003018                1.0003051                1.0003079
prob_mean0_plus_sd[96]   prob_mean0_plus_sd[97]   prob_mean0_plus_sd[98]
             1.0003101                1.0003118                1.0003129
prob_mean0_plus_sd[99]  prob_mean0_plus_sd[100]  prob_mean0_plus_sd[101]
             1.0003133                1.0003130                1.0003121
prob_mean0_plus_sd[102]  prob_mean0_plus_sd[103]  prob_mean0_plus_sd[104]
             1.0003105                1.0003081                1.0003050
prob_mean0_plus_sd[105]  prob_mean0_plus_sd[106]  prob_mean0_plus_sd[107]
             1.0003011                1.0002965                1.0002912
prob_mean0_plus_sd[108]  prob_mean0_plus_sd[109]  prob_mean0_plus_sd[110]
             1.0002851                1.0002784                1.0002709
prob_mean0_plus_sd[111]  prob_mean0_plus_sd[112]  prob_mean0_plus_sd[113]
             1.0002628                1.0002541                1.0002448
prob_mean0_plus_sd[114]  prob_mean0_plus_sd[115]  prob_mean0_plus_sd[116]
             1.0002350                1.0002247                1.0002139
prob_mean0_plus_sd[117]  prob_mean0_plus_sd[118]  prob_mean0_plus_sd[119]
             1.0002028                1.0001913                1.0001795
prob_mean0_plus_sd[120]  prob_mean0_plus_sd[121]  prob_mean0_plus_sd[122]
             1.0001675                1.0001553                1.0001430
prob_mean0_plus_sd[123]  prob_mean0_plus_sd[124]  prob_mean0_plus_sd[125]
             1.0001305                1.0001181                1.0001055
```

```
prob_mean0_plus_sd[126]  prob_mean0_plus_sd[127]  prob_mean0_plus_sd[128]
              1.0000931                1.0000807                1.0000683
prob_mean0_plus_sd[129]  prob_mean0_plus_sd[130]  prob_mean0_plus_sd[131]
              1.0000561                1.0000441                1.0000322
prob_mean0_plus_sd[132]  prob_mean0_plus_sd[133]  prob_mean0_plus_sd[134]
              1.0000206                1.0000091                0.9999979
prob_mean0_plus_sd[135]  prob_mean0_plus_sd[136]  prob_mean0_plus_sd[137]
              0.9999869                0.9999761                0.9999656
prob_mean0_plus_sd[138]  prob_mean0_plus_sd[139]  prob_mean0_plus_sd[140]
              0.9999554                0.9999455                0.9999358
prob_mean0_plus_sd[141]  prob_mean0_plus_sd[142]  prob_mean0_plus_sd[143]
              0.9999264                0.9999172                0.9999084
prob_mean0_plus_sd[144]  prob_mean0_plus_sd[145]  prob_mean0_plus_sd[146]
              0.9998998                0.9998915                0.9998834
prob_mean0_plus_sd[147]  prob_mean0_plus_sd[148]  prob_mean0_plus_sd[149]
              0.9998756                0.9998681                0.9998608
prob_mean0_plus_sd[150]  prob_mean0_plus_sd[151]  prob_mean0_plus_sd[152]
              0.9998537                0.9998469                0.9998404
prob_mean0_plus_sd[153]  prob_mean0_plus_sd[154]  prob_mean0_plus_sd[155]
              0.9998340                0.9998279                0.9998220
prob_mean0_plus_sd[156]  prob_mean0_plus_sd[157]  prob_mean0_plus_sd[158]
              0.9998163                0.9998108                0.9998055
prob_mean0_plus_sd[159]  prob_mean0_plus_sd[160]  prob_mean0_plus_sd[161]
              0.9998005                0.9997956                0.9997908
prob_mean0_plus_sd[162]  prob_mean0_plus_sd[163]  prob_mean0_plus_sd[164]
              0.9997863                0.9997819                0.9997777
prob_mean0_plus_sd[165]  prob_mean0_plus_sd[166]  prob_mean0_plus_sd[167]
              0.9997737                0.9997698                0.9997660
prob_mean0_plus_sd[168]  prob_mean0_plus_sd[169]  prob_mean0_plus_sd[170]
              0.9997624                0.9997590                0.9997557
prob_mean0_plus_sd[171]  prob_mean0_plus_sd[172]  prob_mean0_plus_sd[173]
              0.9997525                0.9997494                0.9997465
prob_mean0_plus_sd[174]  prob_mean0_plus_sd[175]  prob_mean0_plus_sd[176]
              0.9997436                0.9997409                0.9997383
prob_mean0_plus_sd[177]  prob_mean0_plus_sd[178]  prob_mean0_plus_sd[179]
              0.9997359                0.9997335                0.9997312
prob_mean0_plus_sd[180]  prob_mean0_plus_sd[181]                      lp__
              0.9997290                0.9997269                1.0021591
```

Preliminary Visual Plots

```
beta1_samples <- rstan::extract(fit, pars = "beta[1,1]")[["beta[1,1]"]] # extract 4000 coefficients
beta2_samples <- rstan::extract(fit, pars = "beta[2,1]")[["beta[2,1]"]]
mean(beta1_samples) # posterior mean
```

```
[1] 0.305669
```

```
mean(beta2_samples)
```

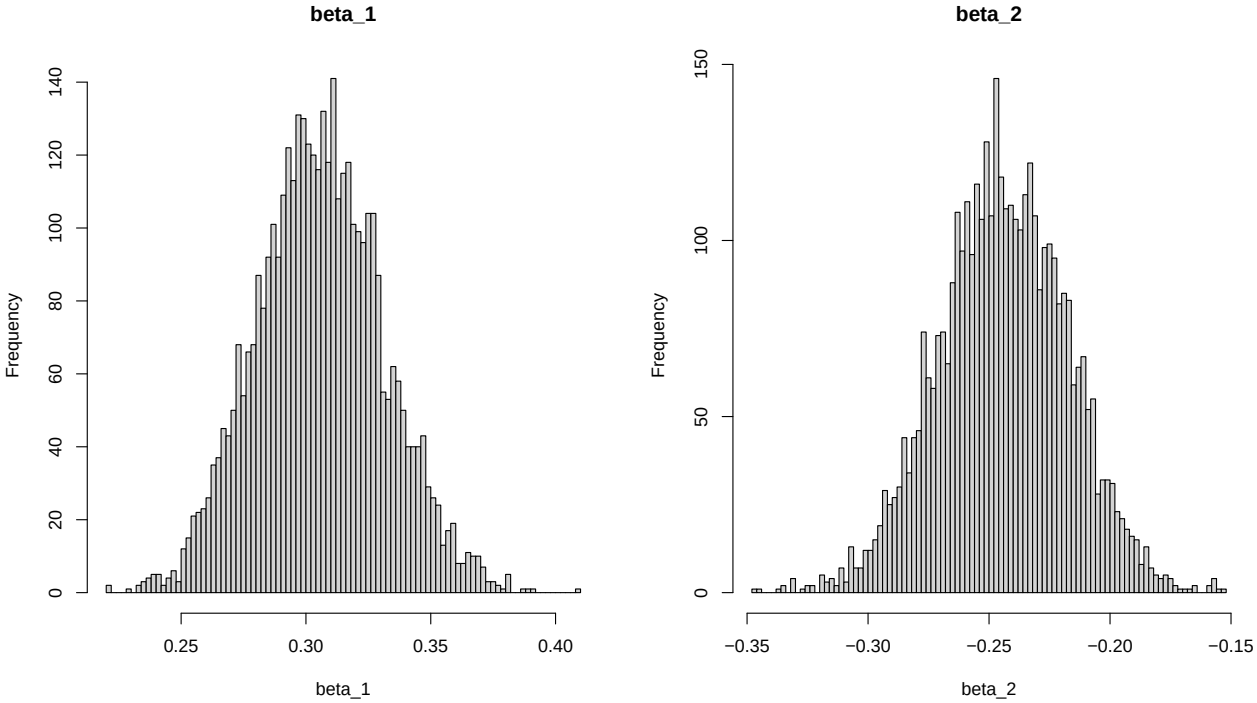
```
[1] -0.2442365
```

As we tune the prior of beta_1 and beta_2, the posterior betas do not change very much. As long as the prior stays within a reasonable range like keeping beta_1 positive and beta_2 negative, the posterior coefficients will stay within the range where beta_1 is around 0.25 and beta_2 around -0.2. This is coherent with the

Bayesian characteristic that the posterior probability will be close to the likelihood / observed data if given sufficient data. In our case, we have 502 training data, which can be considered significantly large.

Plots for coefficients:

```
par(mfrow = c(1,2))
hist(beta1_samples, main = "beta_1", breaks = 100, xlab = "beta_1")
hist(beta2_samples, main = "beta_2", breaks = 100, xlab = "beta_2")
```



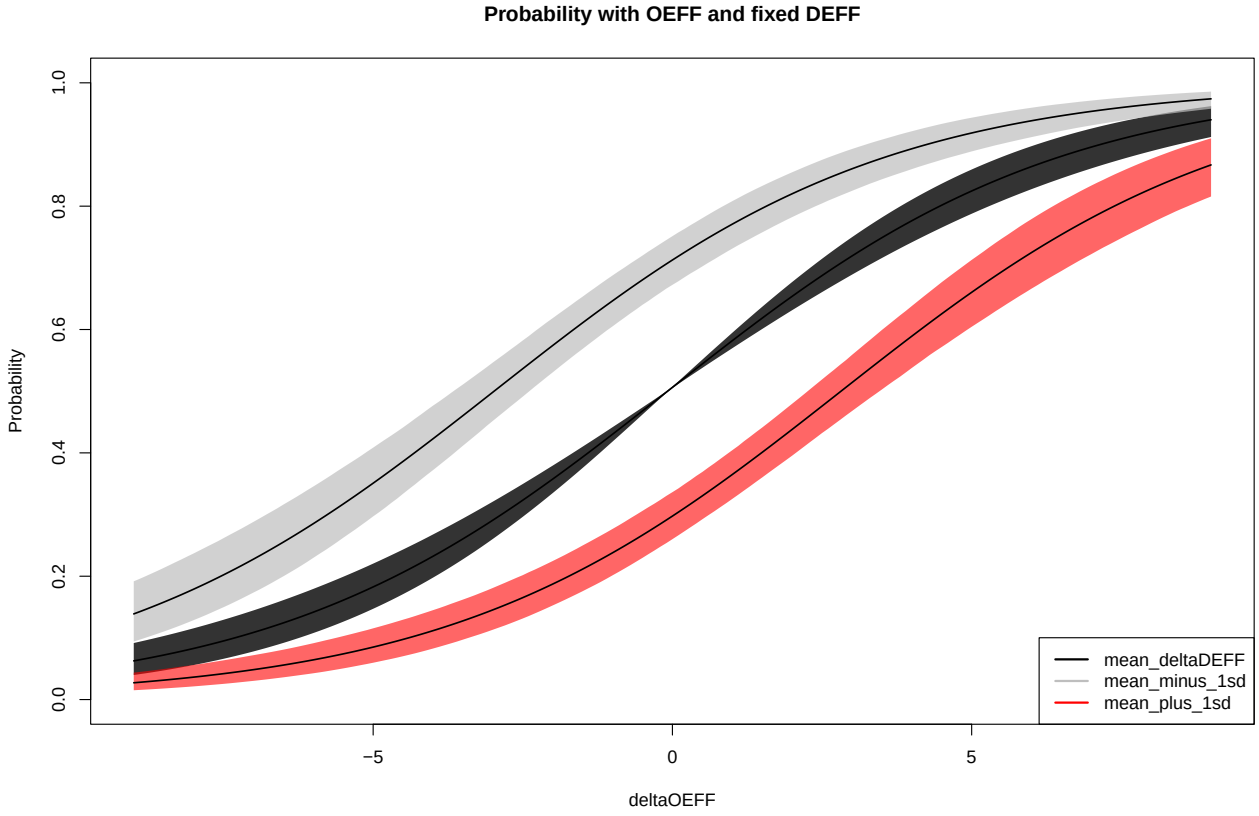
Posterior Model Plots

Since we have 2 predictors, the ideal plot shall be 3D. In order to present the plot in 2D, we fix one predictor on certain values and put the other predictor on the x-axis. In order to present the data properly, we decide to fix one predictor on 3 values: its mean, mean + 1sd, mean - 1sd.

Plot the predictive probabilities

```
post_grid_meanD <- rstan::extract(fit, pars = "prob_meanD")[["prob_meanD"]]
prob_meanD_minus_sd <- rstan::extract(fit, pars = "prob_meanD_minus_sd")[["prob_meanD_minus_sd"]]
prob_meanD_plus_sd <- rstan::extract(fit, pars = "prob_meanD_plus_sd")[["prob_meanD_plus_sd"]]
# at mean deltaDEFF
post_grid_meanD_mean <- apply(post_grid_meanD, MARGIN = 2, FUN = mean)
post_grid_meanD_l95 <- apply(post_grid_meanD, MARGIN = 2, FUN = quantile, probs = 0.025)
post_grid_meanD_u95 <- apply(post_grid_meanD, MARGIN = 2, FUN = quantile, probs = 0.975)
# at mean deltaDEFF - sd
prob_meanD_minus_sd_mean <- apply(prob_meanD_minus_sd, MARGIN = 2, FUN = mean)
prob_meanD_minus_sd_l95 <- apply(prob_meanD_minus_sd, MARGIN = 2, FUN = quantile, probs = 0.025)
prob_meanD_minus_sd_u95 <- apply(prob_meanD_minus_sd, MARGIN = 2, FUN = quantile, probs = 0.975)
# at mean deltaDEFF + sd
prob_meanD_plus_sd_mean <- apply(prob_meanD_plus_sd, MARGIN = 2, FUN = mean)
```

```
prob_meanD_plus_sd_l95 <- apply(prob_meanD_plus_sd, MARGIN = 2, FUN = quantile, probs = 0.025)
prob_meanD_plus_sd_u95 <- apply(prob_meanD_plus_sd, MARGIN = 2, FUN = quantile, probs = 0.975)
# plot
plot(1, type= "n", xlim = c(-9, 9), ylim = c(0,1),
     main = "Probability with OEFF and fixed DEFF", xlab = "deltaOEFF", ylab = "Probability")
# mean
polygon(x = c(deltaOEFF_grid, rev(deltaOEFF_grid)),
        y = c(post_grid_meanD_l95, rev(post_grid_meanD_u95)),
        col = alpha("black", 0.8),
        border = NA)
lines(deltaOEFF_grid, post_grid_meanD_mean, lwd = 1.5)
# mean - sd
polygon(x = c(deltaOEFF_grid, rev(deltaOEFF_grid)),
        y = c(prob_meanD_minus_sd_l95, rev(prob_meanD_minus_sd_u95)),
        col = alpha("grey", 0.7),
        border = NA)
lines(deltaOEFF_grid, prob_meanD_minus_sd_mean, lwd = 1.5)
# mean + sd
polygon(x = c(deltaOEFF_grid, rev(deltaOEFF_grid)),
        y = c(prob_meanD_plus_sd_l95, rev(prob_meanD_plus_sd_u95)),
        col = alpha("red", 0.6),
        border = NA)
lines(deltaOEFF_grid, prob_meanD_plus_sd_mean, lwd = 1.5)
legend("bottomright", legend = c("mean_deltaDEFF", "mean_minus_1sd", "mean_plus_1sd"),
       lty = c(1, 1, 1), lwd = c(2, 2, 2), col = c('black', 'grey', 'red'))
```

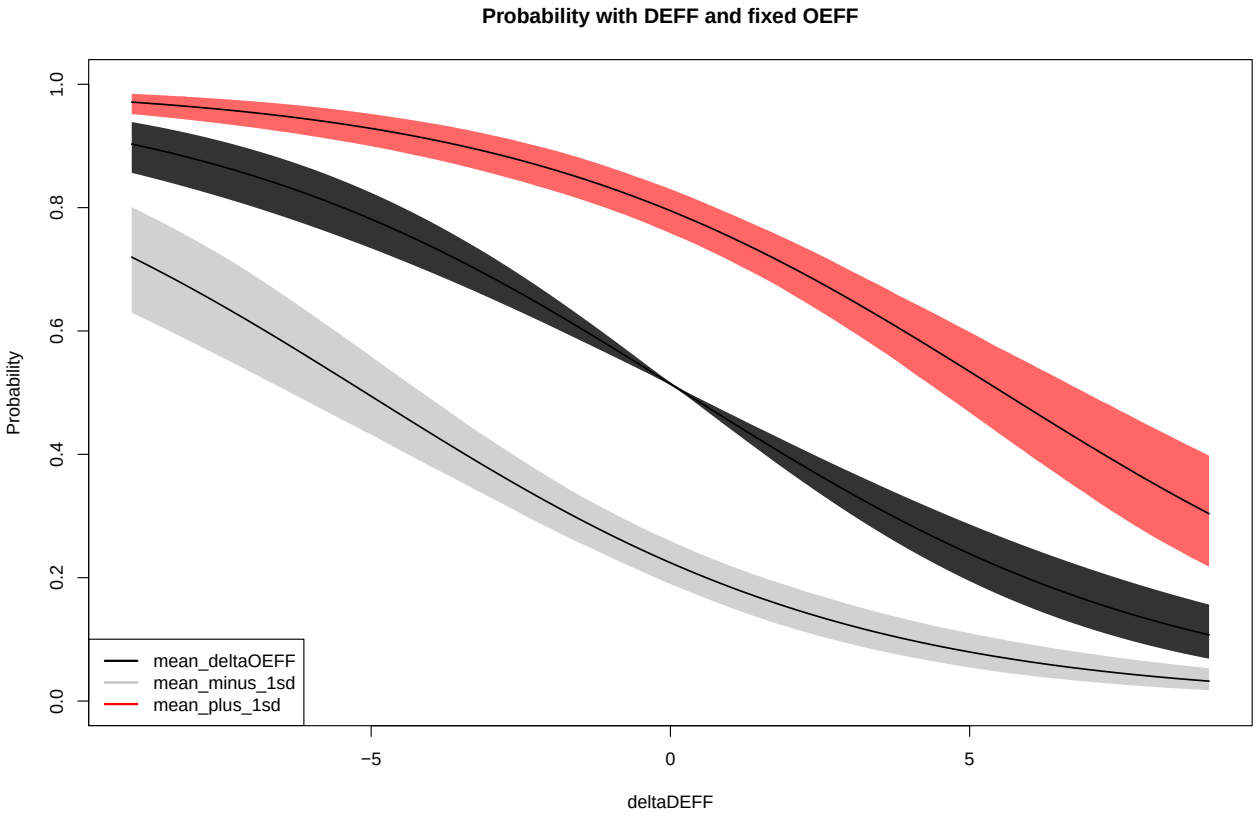


The first plot uses all 4000 beta_1s and beta_2s as coefficients of the logistic regression. Here we can see

that the x-axis deltaOEFF is positively associated with the posterior probability, where the black line is predictor deltaOEFF when deltaDEFF is at its mean, the grey line is deltaDEFF at mean minus one standard deviation, and the red line is deltaDEFF at mean plus one standard deviation. We can see the grey line generally has a higher probability (the position is mean - 1sd). This may be because the two predictors tend to be oppositely plotted, which makes sense in real life as the offensive rate is calculated as the opposite of the other team's defensive rate.

Now try to fix deltaOEFF and put deltaDEFF on x-axis

```
post_grid_mean0 <- rstan::extract(fit, pars = "prob_mean0")[["prob_mean0"]]
prob_mean0_minus_sd <- rstan::extract(fit, pars = "prob_mean0_minus_sd")[["prob_mean0_minus_sd"]]
prob_mean0_plus_sd <- rstan::extract(fit, pars = "prob_mean0_plus_sd")[["prob_mean0_plus_sd"]]
# at mean deltaOEFF
post_grid_mean0_mean <- apply(post_grid_mean0, MARGIN = 2, FUN = mean)
post_grid_mean0_l95 <- apply(post_grid_mean0, MARGIN = 2, FUN = quantile, probs = 0.025)
post_grid_mean0_u95 <- apply(post_grid_mean0, MARGIN = 2, FUN = quantile, probs = 0.975)
# at mean deltaOEFF - sd
prob_mean0_minus_sd_mean <- apply(prob_mean0_minus_sd, MARGIN = 2, FUN = mean)
prob_mean0_minus_sd_l95 <- apply(prob_mean0_minus_sd, MARGIN = 2, FUN = quantile, probs = 0.025)
prob_mean0_minus_sd_u95 <- apply(prob_mean0_minus_sd, MARGIN = 2, FUN = quantile, probs = 0.975)
# at mean deltaOEFF + sd
prob_mean0_plus_sd_mean <- apply(prob_mean0_plus_sd, MARGIN = 2, FUN = mean)
prob_mean0_plus_sd_l95 <- apply(prob_mean0_plus_sd, MARGIN = 2, FUN = quantile, probs = 0.025)
prob_mean0_plus_sd_u95 <- apply(prob_mean0_plus_sd, MARGIN = 2, FUN = quantile, probs = 0.975)
# plot
plot(1, type= "n", xlim = c(-9, 9), ylim = c(0,1),
     main = "Probability with DEFF and fixed OEFF", xlab = "deltaDEFF", ylab = "Probability")
# mean
polygon(x = c(deltaDEFF_grid, rev(deltaDEFF_grid)),
        y = c(post_grid_mean0_l95, rev(post_grid_mean0_u95)),
        col = alpha("black", 0.8),
        border = NA)
lines(deltaDEFF_grid, post_grid_mean0_mean, lwd = 1.5)
# mean - sd
polygon(x = c(deltaDEFF_grid, rev(deltaDEFF_grid)),
        y = c(prob_mean0_minus_sd_l95, rev(prob_mean0_minus_sd_u95)),
        col = alpha("grey", 0.7),
        border = NA)
lines(deltaDEFF_grid, prob_mean0_minus_sd_mean, lwd = 1.5)
# mean + sd
polygon(x = c(deltaDEFF_grid, rev(deltaDEFF_grid)),
        y = c(prob_mean0_plus_sd_l95, rev(prob_mean0_plus_sd_u95)),
        col = alpha("red", 0.6),
        border = NA)
lines(deltaDEFF_grid, prob_mean0_plus_sd_mean, lwd = 1.5)
legend("bottomleft", legend = c("mean_deltaOEFF", "mean_minus_1sd", "mean_plus_1sd"),
       lty = c(1, 1, 1), lwd = c(2, 2, 2), col = c('black', 'grey', 'red'))
```



The probability of winning is roughly negatively associated with the predictor deltaDEFF.

Zsun: Simulation

```
source("zsun_playoff_simulator.R")
```

```
[1] "You can use function 'simu_final(team_name_list = team_list_2019,P)' to simulate one round of playo
[1] "The order of the team name is important!! Please use the default option for simulation of 2019 play
```

```
set.seed(20211129)
pairwise_data <- read.csv("pairwise_DEFF_OEFF_2019.csv")
champions_4000 <- rep()
prob_matrix_4000 <- list()
for (i in 1:4000){
  simu_i_prob <- pairwise_data %>%
    mutate(p = exp(d_OEFF * beta1_samples[i] + d_DEFF * beta2_samples[i])/ (1 + exp(d_OEFF * beta1_sampl
  prob_matrix_i <- matrix(simu_i_prob$p,nrow=16,ncol=16)
  name <- pairwise_data[1:16,1]
  colnames(prob_matrix_i) <- name
  rownames(prob_matrix_i) <- name
  prob_matrix_4000[[i]] <- prob_matrix_i # store the probability matrix
  champions_4000[i] <- simu_final(P = prob_matrix_i)[1] # store the simu_result
}
table(champions_4000)/4000
```

```
champions_4000
      Boston       Denver Golden State      Houston    Milwaukee Philadelphia
     0.00725      0.00850      0.23825      0.02775      0.60200      0.00050
```

```
    Portland      Toronto         Utah
     0.01775      0.08850      0.00950
```

```
# length(prob_matrix_4000)
```

MSE

MSE for 18-19 playoff prediction

```
Regular_19 <- read.csv("cleaned_data_v1_19.csv") # import the data
Regular_19$Success <- as.integer(Regular_19$Y)
deltaOEFF_19 <- Regular_19$deltaOEFF
deltaDEFF_19 <- Regular_19$deltaDEFF
predicted <- mean(beta1_samples) * deltaOEFF_19 + mean(beta2_samples) * deltaDEFF_19
pred_prob <- exp(predicted)/(exp(predicted) + 1)
observed <- Regular_19$Success
O_minus_P <- (observed - pred_prob)^2
(MSE_pred <- sum(O_minus_P)/length(pred_prob))
```

```
[1] 0.2127488
```

Cross validation

Split into 5 folds

```
set.seed(123)
idx <- sample(1:502)
fold1 <- idx[1:100]
fold2 <- idx[101:200]
fold3 <- idx[201:300]
fold4 <- idx[301:401]
fold5 <- idx[402:502]
```

Full Model

fold1 as test

```
train_cv1 <- Regular[-fold1,]
test_cv1 <- Regular[fold1,]
# train_cv1 <- Regular %>% filter(Date != 2013)
# test_cv1 <- Regular %>% filter(Date == 2013)
y <- train_cv1$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # length = 181
x <- cbind(train_cv1$deltaOEFF, train_cv1$deltaDEFF) # dim(502, 2)
K <- 1
N <- length(y) # N = 502
D <- dim(x)[2] # D = 2
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(K = K, # create a list to fill rstan model
                  N = N,
                  D = D,
                  y = y,
```

```
                  x = x,
                  n_grid = n_grid,
                  deltaOEFF_grid = deltaOEFF_grid,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "multi_logistic.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000224 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.24 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 2.64853 seconds (Warm-up)
Chain 1:                2.58075 seconds (Sampling)
Chain 1:                5.22928 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000223 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.23 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.63063 seconds (Warm-up)
Chain 2:                2.7603 seconds (Sampling)
```

```
Chain 2:                5.39093 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000236 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.36 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 2.63799 seconds (Warm-up)
Chain 3:                2.66589 seconds (Sampling)
Chain 3:                5.30388 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000228 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.28 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.66457 seconds (Warm-up)
Chain 4:                2.5699 seconds (Sampling)
Chain 4:                5.23447 seconds (Total)
Chain 4:
```

```
beta1_samples <- rstan::extract(fit, pars = "beta[1,1]")[["beta[1,1]"]] # extract 4000 coefficients
beta2_samples <- rstan::extract(fit, pars = "beta[2,1]")[["beta[2,1]"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv1)[1])
mean.p.test <- rep(0,dim(test_cv1)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv1$deltaOEFF + b2 * train_cv1$deltaDEFF
  log.pred.test <- b1 * test_cv1$deltaOEFF + b2 * test_cv1$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv1_train <- mean((train_cv1$Y - mean.p.train)^2))
```

```
[1] 0.2416489
```

```
(mse_cv1_test <- mean((test_cv1$Y - mean.p.test)^2))
```

```
[1] 0.2613469
```

fold2 as test

```
train_cv2 <- Regular[-fold2,]
test_cv2 <- Regular[fold2,]
# train_cv1 <- Regular %>% filter(Date != 2013)
# test_cv1 <- Regular %>% filter(Date == 2013)
y <- train_cv2$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # length = 181
x <- cbind(train_cv2$deltaOEFF, train_cv2$deltaDEFF) # dim(502, 2)
K <- 1
N <- length(y) # N = 502
D <- dim(x)[2] # D = 2
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(K = K, # create a list to fill rstan model
                  N = N,
                  D = D,
                  y = y,
                  x = x,
                  n_grid = n_grid,
                  deltaOEFF_grid = deltaOEFF_grid,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "multi_logistic.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000232 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.32 seconds.
Chain 1: Adjust your expectations accordingly!
```

```
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 2.65274 seconds (Warm-up)
Chain 1:                2.71737 seconds (Sampling)
Chain 1:                5.3701 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000247 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.47 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.68676 seconds (Warm-up)
Chain 2:                2.67975 seconds (Sampling)
Chain 2:                5.36651 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000234 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.34 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 2.66755 seconds (Warm-up)
Chain 3:                2.73033 seconds (Sampling)
Chain 3:                5.39788 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000238 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.38 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.68479 seconds (Warm-up)
Chain 4:                2.71638 seconds (Sampling)
Chain 4:                5.40118 seconds (Total)
Chain 4:
```

```r
beta1_samples <- rstan::extract(fit, pars = "beta[1,1]")[["beta[1,1]"]] # extract 4000 coefficients
beta2_samples <- rstan::extract(fit, pars = "beta[2,1]")[["beta[2,1]"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv2)[1])
mean.p.test <- rep(0,dim(test_cv2)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv2$deltaOEFF + b2 * train_cv2$deltaDEFF
  log.pred.test <- b1 * test_cv2$deltaOEFF + b2 * test_cv2$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
```

```
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv2_train <- mean((train_cv2$Y - mean.p.train)^2))
```

```
[1] 0.2483723
```

```
(mse_cv2_test <- mean((test_cv2$Y - mean.p.test)^2))
```

```
[1] 0.2200312
```

fold3 as test

```
train_cv3 <- Regular[-fold3,]
test_cv3 <- Regular[fold3,]
# train_cv1 <- Regular %>% filter(Date != 2013)
# test_cv1 <- Regular %>% filter(Date == 2013)
y <- train_cv3$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # length = 181
x <- cbind(train_cv3$deltaOEFF, train_cv3$deltaDEFF) # dim(502, 2)
K <- 1
N <- length(y) # N = 502
D <- dim(x)[2] # D = 2
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(K = K, # create a list to fill rstan model
                  N = N,
                  D = D,
                  y = y,
                  x = x,
                  n_grid = n_grid,
                  deltaOEFF_grid = deltaOEFF_grid,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "multi_logistic.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000226 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.26 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
```

```
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 2.6609 seconds (Warm-up)
Chain 1:                2.72082 seconds (Sampling)
Chain 1:                5.38172 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000224 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.24 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.73804 seconds (Warm-up)
Chain 2:                2.8467 seconds (Sampling)
Chain 2:                5.58473 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000233 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.33 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 2.67482 seconds (Warm-up)
Chain 3:                2.78673 seconds (Sampling)
```

```
Chain 3:                5.46155 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000227 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.27 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.69333 seconds (Warm-up)
Chain 4:                2.7957 seconds (Sampling)
Chain 4:                5.48903 seconds (Total)
Chain 4:
```

```r
beta1_samples <- rstan::extract(fit, pars = "beta[1,1]")[["beta[1,1]"]] # extract 4000 coefficients
beta2_samples <- rstan::extract(fit, pars = "beta[2,1]")[["beta[2,1]"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv3)[1])
mean.p.test <- rep(0,dim(test_cv3)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv3$deltaOEFF + b2 * train_cv3$deltaDEFF
  log.pred.test <- b1 * test_cv3$deltaOEFF + b2 * test_cv3$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv3_train <- mean((train_cv3$Y - mean.p.train)^2))
```

```
[1] 0.2445671
```

```r
(mse_cv3_test <- mean((test_cv3$Y - mean.p.test)^2))
```

```
[1] 0.2387924
```

fold4 as test

```
train_cv4 <- Regular[-fold4,]
test_cv4 <- Regular[fold4,]
# train_cv1 <- Regular %>% filter(Date != 2013)
# test_cv1 <- Regular %>% filter(Date == 2013)
y <- train_cv4$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # length = 181
x <- cbind(train_cv4$deltaOEFF, train_cv4$deltaDEFF) # dim(502, 2)
K <- 1
N <- length(y) # N = 502
D <- dim(x)[2] # D = 2
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(K = K, # create a list to fill rstan model
                  N = N,
                  D = D,
                  y = y,
                  x = x,
                  n_grid = n_grid,
                  deltaOEFF_grid = deltaOEFF_grid,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "multi_logistic.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000237 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.37 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 2.6746 seconds (Warm-up)
Chain 1:                2.62092 seconds (Sampling)
Chain 1:                5.29552 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000243 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.43 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
```

```
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.74086 seconds (Warm-up)
Chain 2:                2.59832 seconds (Sampling)
Chain 2:                5.33918 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000229 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.29 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 2.69999 seconds (Warm-up)
Chain 3:                2.78485 seconds (Sampling)
Chain 3:                5.48484 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000253 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.53 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
```

```
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.73625 seconds (Warm-up)
Chain 4:                2.73337 seconds (Sampling)
Chain 4:                5.46962 seconds (Total)
Chain 4:
```

```
beta1_samples <- rstan::extract(fit, pars = "beta[1,1]")[["beta[1,1]"]] # extract 4000 coefficients
beta2_samples <- rstan::extract(fit, pars = "beta[2,1]")[["beta[2,1]"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv4)[1])
mean.p.test <- rep(0,dim(test_cv4)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv4$deltaOEFF + b2 * train_cv4$deltaDEFF
  log.pred.test <- b1 * test_cv4$deltaOEFF + b2 * test_cv4$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv4_train <- mean((train_cv4$Y - mean.p.train)^2))
```

```
[1] 0.2391259
```

```
(mse_cv4_test <- mean((test_cv4$Y - mean.p.test)^2))
```

```
[1] 0.276788
```

fold5 as test

```
train_cv5 <- Regular[-fold5,]
test_cv5 <- Regular[fold5,]

y <- train_cv5$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # length = 181
x <- cbind(train_cv5$deltaOEFF, train_cv5$deltaDEFF) # dim(502, 2)
K <- 1
N <- length(y) # N = 502
D <- dim(x)[2] # D = 2
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(K = K, # create a list to fill rstan model
                  N = N,
                  D = D,
```

```
                  y = y,
                  x = x,
                  n_grid = n_grid,
                  deltaOEFF_grid = deltaOEFF_grid,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "multi_logistic.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.000232 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 2.32 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 2.71314 seconds (Warm-up)
Chain 1:                2.86278 seconds (Sampling)
Chain 1:                5.57592 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0.000232 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 2.32 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 2.67056 seconds (Warm-up)
```

```
Chain 2:                2.65619 seconds (Sampling)
Chain 2:                5.32676 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0.000227 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 2.27 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 2.72238 seconds (Warm-up)
Chain 3:                2.86651 seconds (Sampling)
Chain 3:                5.58888 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'multi_logistic' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 0.000227 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 2.27 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 2.71775 seconds (Warm-up)
Chain 4:                2.70979 seconds (Sampling)
Chain 4:                5.42754 seconds (Total)
Chain 4:
```

```
beta1_samples <- rstan::extract(fit, pars = "beta[1,1]")[["beta[1,1]"]] # extract 4000 coefficients
beta2_samples <- rstan::extract(fit, pars = "beta[2,1]")[["beta[2,1]"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv5)[1])
mean.p.test <- rep(0,dim(test_cv5)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv5$deltaOEFF + b2 * train_cv5$deltaDEFF
  log.pred.test <- b1 * test_cv5$deltaOEFF + b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv5_train <- mean((train_cv5$Y - mean.p.train)^2))
```

```
[1] 0.2461083
```

```
(mse_cv5_test <- mean((test_cv5$Y - mean.p.test)^2))
```

```
[1] 0.2328125
```

```
# train
mean(c(mse_cv1_train,mse_cv2_train,mse_cv3_train,mse_cv4_train,mse_cv5_train))
```

```
[1] 0.2439645
```

```
# test
mean(c(mse_cv1_test,mse_cv2_test,mse_cv3_test,mse_cv4_test,mse_cv5_test))
```

```
[1] 0.2459542
```

OEFF only

fold 1

```
train_cv1 <- Regular[-fold1,]
test_cv1 <- Regular[fold1,]

y <- train_cv1$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
x <- train_cv1$deltaOEFF
N <- length(y)
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaOEFF = x,
                  N = N,
                  deltaOEFF_grid = deltaOEFF_grid)
test_stan <- stan_model(file = "OEFF_Only.stan")
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/L
In file included from <built-in>:1:
```

```
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal
#include <complex>
         ^~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 7.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.71 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.172425 seconds (Warm-up)
Chain 1:                0.171497 seconds (Sampling)
Chain 1:                0.343922 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.8e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
```

```
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.177644 seconds (Warm-up)
Chain 2:                0.19832 seconds (Sampling)
Chain 2:                0.375964 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.6e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.46 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.176532 seconds (Warm-up)
Chain 3:                0.202131 seconds (Sampling)
Chain 3:                0.378663 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 5.9e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.59 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.179842 seconds (Warm-up)
Chain 4:                0.159337 seconds (Sampling)
Chain 4:                0.339179 seconds (Total)
Chain 4:
```

```
beta1_samples <- extract(fit, pars = "beta")[["beta"]] # extract 4000 coefficients each
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv1)[1])
mean.p.test <- rep(0,dim(test_cv1)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  # b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv1$deltaOEFF # + b2 * train_cv5$deltaDEFF
  log.pred.test <- b1 * test_cv1$deltaOEFF # + b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv1_train <- mean((train_cv1$Y - mean.p.train)^2))
```

```
[1] 0.2484997
```

```
(mse_cv1_test <- mean((test_cv1$Y - mean.p.test)^2))
```

```
[1] 0.306238
```

fold 2

```
train_cv2 <- Regular[-fold2,]
test_cv2 <- Regular[fold2,]

y <- train_cv2$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
x <- train_cv2$deltaOEFF
N <- length(y)
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaOEFF = x,
                  N = N,
                  deltaOEFF_grid = deltaOEFF_grid)
test_stan <- stan_model(file = "OEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

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

```
Chain 1: Gradient evaluation took 5.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.52 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.177569 seconds (Warm-up)
Chain 1:                0.167968 seconds (Sampling)
Chain 1:                0.345537 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.6e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.46 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.170711 seconds (Warm-up)
Chain 2:                0.171355 seconds (Sampling)
Chain 2:                0.342066 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.8e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
```

```
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.175469 seconds (Warm-up)
Chain 3:                0.155827 seconds (Sampling)
Chain 3:                0.331296 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4.6e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.46 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.171733 seconds (Warm-up)
Chain 4:                0.2267 seconds (Sampling)
Chain 4:                0.398433 seconds (Total)
Chain 4:
```

```r
beta1_samples <- extract(fit, pars = "beta")[["beta"]] # extract 4000 coefficients each
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv2)[1])
mean.p.test <- rep(0,dim(test_cv2)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  # b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv2$deltaOEFF # + b2 * train_cv5$deltaDEFF
  log.pred.test <- b1 * test_cv2$deltaOEFF # + b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
```

```
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv2_train <- mean((train_cv2$Y - mean.p.train)^2))
```

```
[1] 0.2586897
```

```
(mse_cv2_test <- mean((test_cv2$Y - mean.p.test)^2))
```

```
[1] 0.2417107
```

fold 3

```
train_cv3 <- Regular[-fold3,]
test_cv3 <- Regular[fold3,]

y <- train_cv3$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
x <- train_cv3$deltaOEFF
N <- length(y)
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaOEFF = x,
                  N = N,
                  deltaOEFF_grid = deltaOEFF_grid)
test_stan <- stan_model(file = "OEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.52 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.173081 seconds (Warm-up)
Chain 1:                0.172359 seconds (Sampling)
Chain 1:                0.34544 seconds (Total)
```

```
Chain 1:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.8e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.175397 seconds (Warm-up)
Chain 2:                0.165729 seconds (Sampling)
Chain 2:                0.341126 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.8e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.175103 seconds (Warm-up)
Chain 3:                0.172897 seconds (Sampling)
Chain 3:                0.348 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 4).
Chain 4:
```

```
Chain 4: Gradient evaluation took 4.5e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.45 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.178702 seconds (Warm-up)
Chain 4:                0.174768 seconds (Sampling)
Chain 4:                0.35347 seconds (Total)
Chain 4:
```

```
beta1_samples <- extract(fit, pars = "beta")[["beta"]] # extract 4000 coefficients each
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv3)[1])
mean.p.test <- rep(0,dim(test_cv3)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  # b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv3$deltaOEFF # + b2 * train_cv5$deltaDEFF
  log.pred.test <- b1 * test_cv3$deltaOEFF # + b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv3_train <- mean((train_cv3$Y - mean.p.train)^2))
```

```
[1] 0.2604634
```

```
(mse_cv3_test <- mean((test_cv3$Y - mean.p.test)^2))
```

```
[1] 0.2154315
```

fold 4

```
train_cv4 <- Regular[-fold4,]
test_cv4 <- Regular[fold4,]

y <- train_cv4$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
x <- train_cv4$deltaOEFF
N <- length(y)
```

```
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaOEFF = x,
                  N = N,
                  deltaOEFF_grid = deltaOEFF_grid)
test_stan <- stan_model(file = "OEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 4.8e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.176949 seconds (Warm-up)
Chain 1:                0.181983 seconds (Sampling)
Chain 1:                0.358932 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 5.4e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.54 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
```

```
Chain 2:  Elapsed Time: 0.180993 seconds (Warm-up)
Chain 2:                0.174767 seconds (Sampling)
Chain 2:                0.35576 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.7e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.47 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.176052 seconds (Warm-up)
Chain 3:                0.163743 seconds (Sampling)
Chain 3:                0.339795 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4.8e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.175384 seconds (Warm-up)
Chain 4:                0.163792 seconds (Sampling)
Chain 4:                0.339176 seconds (Total)
Chain 4:
```

```
beta1_samples <- extract(fit, pars = "beta")[["beta"]] # extract 4000 coefficients each
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv4)[1])
mean.p.test <- rep(0,dim(test_cv4)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  # b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv4$deltaOEFF # + b2 * train_cv5$deltaDEFF
  log.pred.test <- b1 * test_cv4$deltaOEFF # + b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv4_train <- mean((train_cv4$Y - mean.p.train)^2))
```

```
[1] 0.2532097
```

```
(mse_cv4_test <- mean((test_cv4$Y - mean.p.test)^2))
```

```
[1] 0.2712291
```

fold 5

```
train_cv5 <- Regular[-fold5,]
test_cv5 <- Regular[fold5,]

y <- train_cv5$Success # length = 502
deltaOEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
x <- train_cv5$deltaOEFF
N <- length(y)
n_grid = length(deltaOEFF_grid) # n_grid = 181
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaOEFF = x,
                  N = N,
                  deltaOEFF_grid = deltaOEFF_grid)
test_stan <- stan_model(file = "OEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5.1e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.51 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.175568 seconds (Warm-up)
Chain 1:                0.185545 seconds (Sampling)
Chain 1:                0.361113 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 5.5e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.55 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.176883 seconds (Warm-up)
Chain 2:                0.164498 seconds (Sampling)
Chain 2:                0.341381 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.7e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.47 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.178147 seconds (Warm-up)
Chain 3:                0.161034 seconds (Sampling)
Chain 3:                0.339181 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'OEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4.9e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.49 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.174565 seconds (Warm-up)
Chain 4:                0.169726 seconds (Sampling)
Chain 4:                0.344291 seconds (Total)
Chain 4:
```

```
beta1_samples <- extract(fit, pars = "beta")[["beta"]] # extract 4000 coefficients each
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv5)[1])
mean.p.test <- rep(0,dim(test_cv5)[1])
for ( i in 1:4000){
  b1 <- beta1_samples[i]
  # b2 <- beta2_samples[i]
  log.pred.train <- b1 * train_cv5$deltaOEFF # + b2 * train_cv5$deltaDEFF
  log.pred.test <- b1 * test_cv5$deltaOEFF # + b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv5_train <- mean((train_cv5$Y - mean.p.train)^2))
```

```
[1] 0.2569924
```

```
(mse_cv5_test <- mean((test_cv5$Y - mean.p.test)^2))
```

```
[1] 0.2496677
```

```
# train
mean(c(mse_cv1_train,mse_cv2_train,mse_cv3_train,mse_cv4_train,mse_cv5_train))
```

```
[1] 0.255571
```

```
# test
mean(c(mse_cv1_test,mse_cv2_test,mse_cv3_test,mse_cv4_test,mse_cv5_test))
```

```
[1] 0.2568554
```

DEFF only

fold 1

```
train_cv1 <- Regular[-fold1,]
test_cv1 <- Regular[fold1,]

y <- train_cv1$Success # length = 502
deltaDEFF_grid <- seq(-9, 9, by = 0.1) # grid for prediction
x <- train_cv1$deltaDEFF
N <- length(y)
n_grid = length(deltaDEFF_grid) # n_grid = 181
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaDEFF = x,
                  N = N,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "DEFF_Only.stan")
```

```
Running /Library/Frameworks/R.framework/Resources/bin/R CMD SHLIB foo.c
clang -mmacosx-version-min=10.13 -I"/Library/Frameworks/R.framework/Resources/include" -DNDEBUG   -I"/Li
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
^
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/src/Core/util/Ma
namespace Eigen {
               ^
               ;
In file included from <built-in>:1:
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/StanHeaders/include
In file included from /Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/
/Library/Frameworks/R.framework/Versions/4.1/Resources/library/RcppEigen/include/Eigen/Core:96:10: fatal
#include <complex>
         ^~~~~~~~~
3 errors generated.
make: *** [foo.o] Error 1
```

```
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 8.6e-05 seconds
```

```
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.86 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.170569 seconds (Warm-up)
Chain 1:                0.174616 seconds (Sampling)
Chain 1:                0.345185 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.5e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.45 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.169863 seconds (Warm-up)
Chain 2:                0.182927 seconds (Sampling)
Chain 2:                0.35279 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 5.1e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.51 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
```

```
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.172165 seconds (Warm-up)
Chain 3:                0.160415 seconds (Sampling)
Chain 3:                0.33258 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4.8e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.171903 seconds (Warm-up)
Chain 4:                0.156528 seconds (Sampling)
Chain 4:                0.328431 seconds (Total)
Chain 4:
```

```r
beta2_samples <- extract(fit, pars = "beta")[["beta"]] # extract 4000 coefficients each
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv1)[1])
mean.p.test <- rep(0,dim(test_cv1)[1])
for ( i in 1:4000){
  b2 <- beta2_samples[i]
  log.pred.train <- b2 * train_cv1$deltaDEFF
  log.pred.test <- b2 * test_cv1$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
```

```
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv1_train <- mean((train_cv1$Y - mean.p.train)^2))
```

```
[1] 0.2519422
```

```
(mse_cv1_test <- mean((test_cv1$Y - mean.p.test)^2))
```

```
[1] 0.2317686
```

fold 2

```
train_cv2 <- Regular[-fold2,]
test_cv2 <- Regular[fold2,]

y <- train_cv2$Success
deltaDEFF_grid <- seq(-9, 9, by = 0.1)
x <- train_cv2$deltaDEFF
N <- length(y)
n_grid = length(deltaDEFF_grid)
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaDEFF = x,
                  N = N,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "DEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.5 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.177012 seconds (Warm-up)
Chain 1:                0.161351 seconds (Sampling)
Chain 1:                0.338363 seconds (Total)
Chain 1:
```

```
SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.5e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.45 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.167135 seconds (Warm-up)
Chain 2:                0.175203 seconds (Sampling)
Chain 2:                0.342338 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.8e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.173124 seconds (Warm-up)
Chain 3:                0.185977 seconds (Sampling)
Chain 3:                0.359101 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 5.3e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.53 seconds.
```

```
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.173102 seconds (Warm-up)
Chain 4:                0.160282 seconds (Sampling)
Chain 4:                0.333384 seconds (Total)
Chain 4:
```

```
beta2_samples <- extract(fit, pars = "beta")[["beta"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv2)[1])
mean.p.test <- rep(0,dim(test_cv2)[1])
for (i in 1:4000){
  b2 <- beta2_samples[i]
  log.pred.train <- b2 * train_cv2$deltaDEFF
  log.pred.test <- b2 * test_cv2$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv2_train <- mean((train_cv2$Y - mean.p.train)^2))
```

```
[1] 0.2492772
```

```
(mse_cv2_test <- mean((test_cv2$Y - mean.p.test)^2))
```

```
[1] 0.2483159
```

fold 3

```
train_cv3 <- Regular[-fold3,]
test_cv3 <- Regular[fold3,]

y <- train_cv3$Success
deltaDEFF_grid <- seq(-9, 9, by = 0.1)
x <- train_cv3$deltaDEFF
N <- length(y)
n_grid = length(deltaDEFF_grid)
data_list <- list(n_grid = n_grid,
                  y = y,
```

```
                  deltaDEFF = x,
                  N = N,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "DEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 6.6e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.66 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.176972 seconds (Warm-up)
Chain 1:                0.1771 seconds (Sampling)
Chain 1:                0.354072 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.9e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.49 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.177453 seconds (Warm-up)
Chain 2:                0.190158 seconds (Sampling)
Chain 2:                0.367611 seconds (Total)
```

```
Chain 2:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 5.2e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.52 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.170679 seconds (Warm-up)
Chain 3:                0.188794 seconds (Sampling)
Chain 3:                0.359473 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4.6e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.46 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.171369 seconds (Warm-up)
Chain 4:                0.159176 seconds (Sampling)
Chain 4:                0.330545 seconds (Total)
Chain 4:
```

```r
beta2_samples <- extract(fit, pars = "beta")[["beta"]]
### Calculate MSE
```

```
mean.p.train <- rep(0,dim(train_cv3)[1])
mean.p.test <- rep(0,dim(test_cv3)[1])
for (i in 1:4000){
  b2 <- beta2_samples[i]
  log.pred.train <- b2 * train_cv3$deltaDEFF
  log.pred.test <- b2 * test_cv3$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv3_train <- mean((train_cv3$Y - mean.p.train)^2))
```

```
[1] 0.246551
```

```
(mse_cv3_test <- mean((test_cv3$Y - mean.p.test)^2))
```

```
[1] 0.2644499
```

fold 4

```
train_cv4 <- Regular[-fold4,]
test_cv4 <- Regular[fold4,]

y <- train_cv4$Success
deltaDEFF_grid <- seq(-9, 9, by = 0.1)
x <- train_cv4$deltaDEFF
N <- length(y)
n_grid = length(deltaDEFF_grid)
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaDEFF = x,
                  N = N,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "DEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5.2e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.52 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
```

```
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.173461 seconds (Warm-up)
Chain 1:                0.175808 seconds (Sampling)
Chain 1:                0.349269 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.7e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.47 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.176975 seconds (Warm-up)
Chain 2:                0.191084 seconds (Sampling)
Chain 2:                0.368059 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.5e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.45 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
```

```
Chain 3:  Elapsed Time: 0.174513 seconds (Warm-up)
Chain 3:                0.171994 seconds (Sampling)
Chain 3:                0.346507 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 4.4e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.44 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.169354 seconds (Warm-up)
Chain 4:                0.157854 seconds (Sampling)
Chain 4:                0.327208 seconds (Total)
Chain 4:
```

```r
beta2_samples <- extract(fit, pars = "beta")[["beta"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv4)[1])
mean.p.test <- rep(0,dim(test_cv4)[1])
for (i in 1:4000){
  b2 <- beta2_samples[i]
  log.pred.train <- b2 * train_cv4$deltaDEFF
  log.pred.test <- b2 * test_cv4$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv4_train <- mean((train_cv4$Y - mean.p.train)^2))
```

```
[1] 0.2483345
```

```r
(mse_cv4_test <- mean((test_cv4$Y - mean.p.test)^2))
```

```
[1] 0.2542012
```

fold 5

```
train_cv5 <- Regular[-fold5,]
test_cv5 <- Regular[fold5,]

y <- train_cv4$Success
deltaDEFF_grid <- seq(-9, 9, by = 0.1)
x <- train_cv5$deltaDEFF
N <- length(y)
n_grid = length(deltaDEFF_grid)
data_list <- list(n_grid = n_grid,
                  y = y,
                  deltaDEFF = x,
                  N = N,
                  deltaDEFF_grid = deltaDEFF_grid)
test_stan <- stan_model(file = "DEFF_Only.stan")
fit <- sampling(object = test_stan, data = data_list)
```

```
SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 5e-05 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0.5 seconds.
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 1: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 1: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 1: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 1: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 1: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 1: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 1:
Chain 1:  Elapsed Time: 0.169461 seconds (Warm-up)
Chain 1:                0.183738 seconds (Sampling)
Chain 1:                0.353199 seconds (Total)
Chain 1:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 4.9e-05 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0.49 seconds.
Chain 2: Adjust your expectations accordingly!
Chain 2:
Chain 2:
Chain 2: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 2: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 2: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 2: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 2: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 2: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

```
Chain 2: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 2: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 2: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 2: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 2: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 2: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 2:
Chain 2:  Elapsed Time: 0.16851 seconds (Warm-up)
Chain 2:                0.167773 seconds (Sampling)
Chain 2:                0.336283 seconds (Total)
Chain 2:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 4.8e-05 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0.48 seconds.
Chain 3: Adjust your expectations accordingly!
Chain 3:
Chain 3:
Chain 3: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 3: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 3: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 3: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 3: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 3: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 3: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 3: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 3: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 3: Iteration: 1600 / 2000 [ 80%]  (Sampling)
Chain 3: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 3: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 3:
Chain 3:  Elapsed Time: 0.169702 seconds (Warm-up)
Chain 3:                0.15824 seconds (Sampling)
Chain 3:                0.327942 seconds (Total)
Chain 3:

SAMPLING FOR MODEL 'DEFF_Only' NOW (CHAIN 4).
Chain 4:
Chain 4: Gradient evaluation took 5.1e-05 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0.51 seconds.
Chain 4: Adjust your expectations accordingly!
Chain 4:
Chain 4:
Chain 4: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 4: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 4: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 4: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 4: Iteration:  800 / 2000 [ 40%]  (Warmup)
Chain 4: Iteration: 1000 / 2000 [ 50%]  (Warmup)
Chain 4: Iteration: 1001 / 2000 [ 50%]  (Sampling)
Chain 4: Iteration: 1200 / 2000 [ 60%]  (Sampling)
Chain 4: Iteration: 1400 / 2000 [ 70%]  (Sampling)
Chain 4: Iteration: 1600 / 2000 [ 80%]  (Sampling)
```

```
Chain 4: Iteration: 1800 / 2000 [ 90%]  (Sampling)
Chain 4: Iteration: 2000 / 2000 [100%]  (Sampling)
Chain 4:
Chain 4:  Elapsed Time: 0.162739 seconds (Warm-up)
Chain 4:                0.15832 seconds (Sampling)
Chain 4:                0.321059 seconds (Total)
Chain 4:
```

```
beta2_samples <- extract(fit, pars = "beta")[["beta"]]
### Calculate MSE
mean.p.train <- rep(0,dim(train_cv5)[1])
mean.p.test <- rep(0,dim(test_cv5)[1])
for (i in 1:4000){
  b2 <- beta2_samples[i]
  log.pred.train <- b2 * train_cv5$deltaDEFF
  log.pred.test <- b2 * test_cv5$deltaDEFF
  p.train <- exp(log.pred.train) / (1 + exp(log.pred.train))
  p.test <- exp(log.pred.test) / (1 + exp(log.pred.test))
  mean.p.train <- mean.p.train + p.train/4000
  mean.p.test <- mean.p.test  + p.test/4000
}

(mse_cv5_train <- mean((train_cv5$Y - mean.p.train)^2))
```

```
[1] 0.2465516
```

```
(mse_cv5_test <- mean((test_cv5$Y - mean.p.test)^2))
```

```
[1] 0.2473444
```

```
# train
mean(c(mse_cv1_train,mse_cv2_train,mse_cv3_train,mse_cv4_train,mse_cv5_train))
```

```
[1] 0.2485313
```

```
# test
mean(c(mse_cv1_test,mse_cv2_test,mse_cv3_test,mse_cv4_test,mse_cv5_test))
```

```
[1] 0.249216
```