Simulate a binomial response and random effects

Eamonn O'Brien

17 July, 2019

Contents

Contents

List of Figures

List of Tables

Some functions

Introduction

```r
cat("\nI simulate between country and between site SD with an underlying mean (see reference where code
```

```
I simulate between country and between site SD with an underlying mean (see reference where code was hel
```

Contents

```r
cat("* Directories
 * Population parameters  & Design Paramters for simulation
 * Examine simulation datan
 * Create unbalanced data. Trial and error to program this!
 * Frequentist random effects model
 * Use approach in reference , although I think it can only handle one cluster
 * Use sandwich approach in reference , although I think it can only handle one cluster
 * Analysis ignoring clustering
 * Bayesian
 * Load Bayesian analysis
 * Check Model
 * Predictions
 * Plot Bayesian site level predictions
 * Plot Bayesian country level predictions
 * Exploring
 * Plot country SD estimates
 * Plot site SD estimates
 * Crude estimates
 * Plot crude estimates country level
 * Plot crude site estimates
 * lmer estimates sites
 * lmer estimates countries
 * References
 * Computing environment")
```

```
* Directories
 * Population parameters  & Design Paramters for simulation
 * Examine simulation datan
 * Create unbalanced data. Trial and error to program this!
 * Frequentist random effects model
 * Use approach in reference , although I think it can only handle one cluster
 * Use sandwich approach in reference , although I think it can only handle one cluster
 * Analysis ignoring clustering
 * Bayesian
 * Load Bayesian analysis
 * Check Model
 * Predictions
 * Plot Bayesian site level predictions
 * Plot Bayesian country level predictions
 * Exploring
 * Plot country SD estimates
 * Plot site SD estimates
 * Crude estimates
```

```
* Plot crude estimates country level
* Plot crude site estimates
* lmer estimates sites
* lmer estimates countries
* References
* Computing environment
```

Population parameters & Design Paramters for simulation

```r
# mu: underlying mean of the outcome in the control group
# beta1: covariate not used sdcountry: sd of random effect at
# the country level (sd for the bi) sdsite: sd of random
# effect at the site level (sd for the bij)

# Design parameters countries: number of countries sites:
# number of sites per country persons : number of persons per
# site

mu = 0.8  # 0.5 means odds = 1, so intercept should be zero (log odds)
(reg.intercept <- log(mu/(1 - mu)))  #   the intercept of the regression model should approximate this
```

```
[1] 1.386294
```

```r
# get back to prob
exp(reg.intercept)/(1 + exp(reg.intercept))  # convert log odds to prob
```

```
[1] 0.8
```

```r
beta1 = 1  # if this was 1, means no difference, log(1)==0 * trt

sdcountry = 0.2  # this is on the log odds scale
sdsite = 0.4  # this is on the log odds scale

# Design Parameters
countries = 25  # no of countries

# no of sites in countries
sites <- MASS::rnegbin(countries, mu = 10, theta = 1.6)
sites <- ifelse(sites == 0, 1, sites)  # dont want 0s

# no persons at each site
persons <- MASS::rnegbin(sum(sites), mu = 11, theta = 2) + 1  # dont want any 0s hence the + 1
```

Examine simulation data

```r
sum(persons)
```

```
[1] 2042
```

```r
sites  # shows the number of sites in each of the countries
```

```
 [1]  7  1 10 24 10  1  1 19 12  6  7  8 23  9  5  5  4  1  8  1  1  1  5  4  6
```

```r
sequence(sites)  # expand the sites
```

```
  [1]  1  2  3  4  5  6  7  1  1  2  3  4  5  6  7  8  9 10  1  2  3  4  5  6  7  8  9 10
```

```
 [29] 11 12 13 14 15 16 17 18 19 20 21 22 23 24  1  2  3  4  5  6  7  8  9 10  1  1  1  2
 [57]  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19  1  2  3  4  5  6  7  8  9 10 11
 [85] 12  1  2  3  4  5  6  1  2  3  4  5  6  7  1  2  3  4  5  6  7  8  1  2  3  4  5  6
[113]  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23  1  2  3  4  5  6  7  8  9  1  2
[141]  3  4  5  1  2  3  4  5  1  2  3  4  1  1  2  3  4  5  6  7  8  1  1  1  1  2  3  4
[169]  5  1  2  3  4  1  2  3  4  5  6
```

```
persons  # shows the persons at each site
```

```
  [1] 24 23 10  2  1  4 17 12  3 40  1 15 11  5 13 26  4  6  4 12  6  1 10 10 14  9  9  6
 [29] 21 19 10 10  8  7  4 27  9 21 10 14 24 10 21  5 10 14 12  9  8 17  7 11 14 12  4 11
 [57] 17 15 12 10 29 17  8 10  4 14 11 18  8  4 17  3 11  7 15  7  5 17  9 14  7  4  7  2
 [85] 14 15 12 11  5 34  4 32  7  9 12  3  1 18 11 10 18  7 20 12 21  6 34 10  3  3 31 15
[113]  8  4  8 10 28  7 10  7  8 15 11 42  8 13  9  4 11  6  4  4 13  3  8  3  3 14 14  9
[141]  2 18 16  2  7  1  3  2  8 26 26  8  4  7 18  9  9 22 32 24  5 20 18  6 14 11  3 14
[169] 12 10 11  6  1  7  7 11 21  5  5
```

```
rep(1:length(sites), sites)  # grouping indicator for sites
```

```
  [1]  1  1  1  1  1  1  1  2  3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4  4  4  4  4
 [29]  4  4  4  4  4  4  4  4  4  4  4  4  4  4  5  5  5  5  5  5  5  5  5  5  6  7  8  8
 [57]  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  8  9  9  9  9  9  9  9  9  9  9  9
 [85]  9 10 10 10 10 10 10 11 11 11 11 11 11 11 12 12 12 12 12 12 12 12 13 13 13 13 13 13
[113] 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 14 14 14 14 14 14 14 14 14 15 15
[141] 15 15 15 16 16 16 16 16 17 17 17 17 18 19 19 19 19 19 19 19 19 20 21 22 23 23 23 23
[169] 23 24 24 24 24 25 25 25 25 25 25
```

Create unbalanced data. Trial and error to program this!

```
# pp <- seq_along(rep(persons, persons))# count of persons
# (not really needed)
pp <- rep(1:sum(persons))  # count of persons (not really needed) but simpler
g <- rep(1:length(persons), persons)  # person in each site is 'flattened'
x <- rep(1:length(sites), sites)  # sites ditto
country <- rep(x, persons)  # countries

# put tx in there for now although it does not vary
tx = 1
d <- cbind(country = country, site = g, person = pp, tx = tx)  # create a data frame
summary(d)
```

```
    country           site            person             tx
 Min.   : 1.00   Min.   :  1.00   Min.   :   1.0   Min.   :1
 1st Qu.: 5.00   1st Qu.: 43.00   1st Qu.: 511.2   1st Qu.:1
 Median :10.00   Median : 90.00   Median :1021.5   Median :1
 Mean   :10.63   Mean   : 88.63   Mean   :1021.5   Mean   :1
 3rd Qu.:14.00   3rd Qu.:130.00   3rd Qu.:1531.8   3rd Qu.:1
 Max.   :25.00   Max.   :179.00   Max.   :2042.0   Max.   :1
```

```
# draw random effects for clusters
countryRE <- rnorm(countries, 0, sdcountry)
siteRE <- rnorm(sum(sites), 0, sdsite)


# create outcome
(prob <- 1/(1 + exp(-(log(mu/(1 - mu)) + log(beta1) * tx + countryRE[d[,
```

```
    1]] + siteRE[d[, 2]]))))
```

```
  [1] 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991
  [9] 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991
 [17] 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991 0.7810991
 [25] 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514
 [33] 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514
 [41] 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.8505514 0.7774309
 [49] 0.7774309 0.7774309 0.7774309 0.7774309 0.7774309 0.7774309 0.7774309 0.7774309
 [57] 0.7774309 0.8316754 0.8316754 0.7744956 0.7217244 0.7217244 0.7217244 0.7217244
 [65] 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378
 [73] 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378 0.8669378
 [81] 0.8669378 0.8126269 0.8126269 0.8126269 0.8126269 0.8126269 0.8126269 0.8126269
 [89] 0.8126269 0.8126269 0.8126269 0.8126269 0.8126269 0.9019021 0.9019021 0.9019021
 [97] 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198
[105] 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198
[113] 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198
[121] 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198
[129] 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198 0.8256198
[137] 0.8785570 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386
[145] 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386 0.9104386
[153] 0.8047486 0.8047486 0.8047486 0.8047486 0.8047486 0.8047486 0.8047486 0.8047486
[161] 0.8047486 0.8047486 0.8047486 0.7309907 0.7309907 0.7309907 0.7309907 0.7309907
[169] 0.8073830 0.8073830 0.8073830 0.8073830 0.8073830 0.8073830 0.8073830 0.8073830
[177] 0.8073830 0.8073830 0.8073830 0.8073830 0.8073830 0.8089526 0.8089526 0.8089526
[185] 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526
[193] 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526
[201] 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8089526 0.8305468
[209] 0.8305468 0.8305468 0.8305468 0.9014781 0.9014781 0.9014781 0.9014781 0.9014781
[217] 0.9014781 0.8134814 0.8134814 0.8134814 0.8134814 0.8522617 0.8522617 0.8522617
[225] 0.8522617 0.8522617 0.8522617 0.8522617 0.8522617 0.8522617 0.8522617 0.8522617
[233] 0.8522617 0.8193834 0.8193834 0.8193834 0.8193834 0.8193834 0.8193834 0.8336098
[241] 0.8492681 0.8492681 0.8492681 0.8492681 0.8492681 0.8492681 0.8492681 0.8492681
[249] 0.8492681 0.8492681 0.8942148 0.8942148 0.8942148 0.8942148 0.8942148 0.8942148
[257] 0.8942148 0.8942148 0.8942148 0.8942148 0.8391310 0.8391310 0.8391310 0.8391310
[265] 0.8391310 0.8391310 0.8391310 0.8391310 0.8391310 0.8391310 0.8391310 0.8391310
[273] 0.8391310 0.8391310 0.8685714 0.8685714 0.8685714 0.8685714 0.8685714 0.8685714
[281] 0.8685714 0.8685714 0.8685714 0.8715567 0.8715567 0.8715567 0.8715567 0.8715567
[289] 0.8715567 0.8715567 0.8715567 0.8715567 0.8775207 0.8775207 0.8775207 0.8775207
[297] 0.8775207 0.8775207 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437
[305] 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437
[313] 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.7979437 0.9049952
[321] 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952
[329] 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952 0.9049952
[337] 0.9049952 0.9049952 0.8101318 0.8101318 0.8101318 0.8101318 0.8101318 0.8101318
[345] 0.8101318 0.8101318 0.8101318 0.8101318 0.8264876 0.8264876 0.8264876 0.8264876
[353] 0.8264876 0.8264876 0.8264876 0.8264876 0.8264876 0.8264876 0.8970501 0.8970501
[361] 0.8970501 0.8970501 0.8970501 0.8970501 0.8970501 0.8970501 0.8929164 0.8929164
[369] 0.8929164 0.8929164 0.8929164 0.8929164 0.8929164 0.7626481 0.7626481 0.7626481
[377] 0.7626481 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951
[385] 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951
[393] 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951 0.7779951
[401] 0.7779951 0.7779951 0.7779951 0.7779951 0.7426939 0.7426939 0.7426939 0.7426939
[409] 0.7426939 0.7426939 0.7426939 0.7426939 0.7426939 0.8423726 0.8423726 0.8423726
```

```
[417] 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726
[425] 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726 0.8423726
[433] 0.8423726 0.8423726 0.8414672 0.8414672 0.8414672 0.8414672 0.8414672 0.8414672
[441] 0.8414672 0.8414672 0.8414672 0.8414672 0.8518134 0.8518134 0.8518134 0.8518134
[449] 0.8518134 0.8518134 0.8518134 0.8518134 0.8518134 0.8518134 0.8518134 0.8518134
[457] 0.8518134 0.8518134 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589
[465] 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589
[473] 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589 0.8610589
[481] 0.8610589 0.8610589 0.7961645 0.7961645 0.7961645 0.7961645 0.7961645 0.7961645
[489] 0.7961645 0.7961645 0.7961645 0.7961645 0.6327000 0.6327000 0.6327000 0.6327000
[497] 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000
[505] 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000 0.6327000
[513] 0.6327000 0.7944582 0.7944582 0.7944582 0.7944582 0.7944582 0.7758635 0.7758635
[521] 0.7758635 0.7758635 0.7758635 0.7758635 0.7758635 0.7758635 0.7758635 0.7758635
[529] 0.5836148 0.5836148 0.5836148 0.5836148 0.5836148 0.5836148 0.5836148 0.5836148
[537] 0.5836148 0.5836148 0.5836148 0.5836148 0.5836148 0.5836148 0.7116144 0.7116144
[545] 0.7116144 0.7116144 0.7116144 0.7116144 0.7116144 0.7116144 0.7116144 0.7116144
[553] 0.7116144 0.7116144 0.6417385 0.6417385 0.6417385 0.6417385 0.6417385 0.6417385
[561] 0.6417385 0.6417385 0.6417385 0.5282191 0.5282191 0.5282191 0.5282191 0.5282191
[569] 0.5282191 0.5282191 0.5282191 0.7313229 0.7313229 0.7313229 0.7313229 0.7313229
[577] 0.7313229 0.7313229 0.7313229 0.7313229 0.7313229 0.7313229 0.7313229 0.7313229
[585] 0.7313229 0.7313229 0.7313229 0.7313229 0.7129208 0.7129208 0.7129208 0.7129208
[593] 0.7129208 0.7129208 0.7129208 0.7114320 0.7114320 0.7114320 0.7114320 0.7114320
[601] 0.7114320 0.7114320 0.7114320 0.7114320 0.7114320 0.7114320 0.8481152 0.8481152
[609] 0.8481152 0.8481152 0.8481152 0.8481152 0.8481152 0.8481152 0.8481152 0.8481152
[617] 0.8481152 0.8481152 0.8481152 0.8481152 0.8162390 0.8162390 0.8162390 0.8162390
[625] 0.8162390 0.8162390 0.8162390 0.8162390 0.8162390 0.8162390 0.8162390 0.8162390
[633] 0.8262798 0.8262798 0.8262798 0.8262798 0.7739365 0.7739365 0.7739365 0.7739365
[641] 0.7739365 0.7739365 0.7739365 0.7739365 0.7739365 0.7739365 0.7739365 0.7655458
[649] 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458
[657] 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458 0.7655458
[665] 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471
[673] 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471 0.8060471 0.8321937
[681] 0.8321937 0.8321937 0.8321937 0.8321937 0.8321937 0.8321937 0.8321937 0.8321937
[689] 0.8321937 0.8321937 0.8321937 0.7429534 0.7429534 0.7429534 0.7429534 0.7429534
[697] 0.7429534 0.7429534 0.7429534 0.7429534 0.7429534 0.8027426 0.8027426 0.8027426
[705] 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426
[713] 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426
[721] 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426 0.8027426
[729] 0.8027426 0.8027426 0.8657805 0.8657805 0.8657805 0.8657805 0.8657805 0.8657805
[737] 0.8657805 0.8657805 0.8657805 0.8657805 0.8657805 0.8657805 0.8657805 0.8657805
[745] 0.8657805 0.8657805 0.8657805 0.8075139 0.8075139 0.8075139 0.8075139 0.8075139
[753] 0.8075139 0.8075139 0.8075139 0.7681474 0.7681474 0.7681474 0.7681474 0.7681474
[761] 0.7681474 0.7681474 0.7681474 0.7681474 0.7681474 0.7968740 0.7968740 0.7968740
[769] 0.7968740 0.8724846 0.8724846 0.8724846 0.8724846 0.8724846 0.8724846 0.8724846
[777] 0.8724846 0.8724846 0.8724846 0.8724846 0.8724846 0.8724846 0.8724846 0.8451773
[785] 0.8451773 0.8451773 0.8451773 0.8451773 0.8451773 0.8451773 0.8451773 0.8451773
[793] 0.8451773 0.8451773 0.8778170 0.8778170 0.8778170 0.8778170 0.8778170 0.8778170
[801] 0.8778170 0.8778170 0.8778170 0.8778170 0.8778170 0.8778170 0.8778170 0.8778170
[809] 0.8778170 0.8778170 0.8778170 0.8778170 0.8224284 0.8224284 0.8224284 0.8224284
[817] 0.8224284 0.8224284 0.8224284 0.8224284 0.8377197 0.8377197 0.8377197 0.8377197
[825] 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683
[833] 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683 0.7779683
[841] 0.7779683 0.8480511 0.8480511 0.8480511 0.7815488 0.7815488 0.7815488 0.7815488
```

```
 [849] 0.7815488 0.7815488 0.7815488 0.7815488 0.7815488 0.7815488 0.7815488 0.7211250
 [857] 0.7211250 0.7211250 0.7211250 0.7211250 0.7211250 0.7211250 0.8277083 0.8277083
 [865] 0.8277083 0.8277083 0.8277083 0.8277083 0.8277083 0.8277083 0.8277083 0.8277083
 [873] 0.8277083 0.8277083 0.8277083 0.8277083 0.8277083 0.9085963 0.9085963 0.9085963
 [881] 0.9085963 0.9085963 0.9085963 0.9085963 0.8627879 0.8627879 0.8627879 0.8627879
 [889] 0.8627879 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469
 [897] 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469 0.8791469
 [905] 0.8791469 0.8791469 0.8404818 0.8404818 0.8404818 0.8404818 0.8404818 0.8404818
 [913] 0.8404818 0.8404818 0.8404818 0.7815722 0.7815722 0.7815722 0.7815722 0.7815722
 [921] 0.7815722 0.7815722 0.7815722 0.7815722 0.7815722 0.7815722 0.7815722 0.7815722
 [929] 0.7815722 0.8080484 0.8080484 0.8080484 0.8080484 0.8080484 0.8080484 0.8080484
 [937] 0.8036043 0.8036043 0.8036043 0.8036043 0.7909721 0.7909721 0.7909721 0.7909721
 [945] 0.7909721 0.7909721 0.7909721 0.8581419 0.8581419 0.7387206 0.7387206 0.7387206
 [953] 0.7387206 0.7387206 0.7387206 0.7387206 0.7387206 0.7387206 0.7387206 0.7387206
 [961] 0.7387206 0.7387206 0.7387206 0.8446003 0.8446003 0.8446003 0.8446003 0.8446003
 [969] 0.8446003 0.8446003 0.8446003 0.8446003 0.8446003 0.8446003 0.8446003 0.8446003
 [977] 0.8446003 0.8446003 0.8444840 0.8444840 0.8444840 0.8444840 0.8444840 0.8444840
 [985] 0.8444840 0.8444840 0.8444840 0.8444840 0.8444840 0.8444840 0.7040935 0.7040935
 [993] 0.7040935 0.7040935 0.7040935 0.7040935 0.7040935 0.7040935 0.7040935 0.7040935
[1001] 0.7040935 0.8323135 0.8323135 0.8323135 0.8323135 0.8323135 0.9103901 0.9103901
[1009] 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901
[1017] 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901
[1025] 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901
[1033] 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901 0.9103901
[1041] 0.7546244 0.7546244 0.7546244 0.7546244 0.8313511 0.8313511 0.8313511 0.8313511
[1049] 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511
[1057] 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511
[1065] 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511 0.8313511
[1073] 0.8313511 0.8313511 0.8313511 0.8313511 0.7199857 0.7199857 0.7199857 0.7199857
[1081] 0.7199857 0.7199857 0.7199857 0.8457379 0.8457379 0.8457379 0.8457379 0.8457379
[1089] 0.8457379 0.8457379 0.8457379 0.8457379 0.7953045 0.7953045 0.7953045 0.7953045
[1097] 0.7953045 0.7953045 0.7953045 0.7953045 0.7953045 0.7953045 0.7953045 0.7953045
[1105] 0.8719195 0.8719195 0.8719195 0.6623274 0.7749982 0.7749982 0.7749982 0.7749982
[1113] 0.7749982 0.7749982 0.7749982 0.7749982 0.7749982 0.7749982 0.7749982 0.7749982
[1121] 0.7749982 0.7749982 0.7749982 0.7749982 0.7749982 0.7749982 0.7700491 0.7700491
[1129] 0.7700491 0.7700491 0.7700491 0.7700491 0.7700491 0.7700491 0.7700491 0.7700491
[1137] 0.7700491 0.7925644 0.7925644 0.7925644 0.7925644 0.7925644 0.7925644 0.7925644
[1145] 0.7925644 0.7925644 0.7925644 0.7626977 0.7626977 0.7626977 0.7626977 0.7626977
[1153] 0.7626977 0.7626977 0.7626977 0.7626977 0.7626977 0.7626977 0.7626977 0.7626977
[1161] 0.7626977 0.7626977 0.7626977 0.7626977 0.7626977 0.7475940 0.7475940 0.7475940
[1169] 0.7475940 0.7475940 0.7475940 0.7475940 0.8390056 0.8390056 0.8390056 0.8390056
[1177] 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056
[1185] 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056 0.8390056
[1193] 0.7279658 0.7279658 0.7279658 0.7279658 0.7279658 0.7279658 0.7279658 0.7279658
[1201] 0.7279658 0.7279658 0.7279658 0.7279658 0.8821522 0.8821522 0.8821522 0.8821522
[1209] 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522
[1217] 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522 0.8821522
[1225] 0.8821522 0.7200835 0.7200835 0.7200835 0.7200835 0.7200835 0.7200835 0.8197256
[1233] 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256
[1241] 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256
[1249] 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256
[1257] 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256 0.8197256
[1265] 0.8197256 0.8438627 0.8438627 0.8438627 0.8438627 0.8438627 0.8438627 0.8438627
[1273] 0.8438627 0.8438627 0.8438627 0.8466744 0.8466744 0.8466744 0.7946783 0.7946783
```

```
[1281] 0.7946783 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467
[1289] 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467
[1297] 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467
[1305] 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467 0.8184467
[1313] 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295
[1321] 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295 0.8651295 0.7091938
[1329] 0.7091938 0.7091938 0.7091938 0.7091938 0.7091938 0.7091938 0.7091938 0.7615801
[1337] 0.7615801 0.7615801 0.7615801 0.8323283 0.8323283 0.8323283 0.8323283 0.8323283
[1345] 0.8323283 0.8323283 0.8323283 0.7851855 0.7851855 0.7851855 0.7851855 0.7851855
[1353] 0.7851855 0.7851855 0.7851855 0.7851855 0.7851855 0.8830938 0.8830938 0.8830938
[1361] 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938
[1369] 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938
[1377] 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938 0.8830938
[1385] 0.8830938 0.8511133 0.8511133 0.8511133 0.8511133 0.8511133 0.8511133 0.8511133
[1393] 0.8180103 0.8180103 0.8180103 0.8180103 0.8180103 0.8180103 0.8180103 0.8180103
[1401] 0.8180103 0.8180103 0.7049444 0.7049444 0.7049444 0.7049444 0.7049444 0.7049444
[1409] 0.7049444 0.8152510 0.8152510 0.8152510 0.8152510 0.8152510 0.8152510 0.8152510
[1417] 0.8152510 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193
[1425] 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193 0.7937193
[1433] 0.8073869 0.8073869 0.8073869 0.8073869 0.8073869 0.8073869 0.8073869 0.8073869
[1441] 0.8073869 0.8073869 0.8073869 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405
[1449] 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405
[1457] 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405
[1465] 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405
[1473] 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405
[1481] 0.7313405 0.7313405 0.7313405 0.7313405 0.7313405 0.8420447 0.8420447 0.8420447
[1489] 0.8420447 0.8420447 0.8420447 0.8420447 0.8420447 0.7423991 0.7423991 0.7423991
[1497] 0.7423991 0.7423991 0.7423991 0.7423991 0.7423991 0.7423991 0.7423991 0.7423991
[1505] 0.7423991 0.7423991 0.7995636 0.7995636 0.7995636 0.7995636 0.7995636 0.7995636
[1513] 0.7995636 0.7995636 0.7995636 0.8357072 0.8357072 0.8357072 0.8357072 0.7614511
[1521] 0.7614511 0.7614511 0.7614511 0.7614511 0.7614511 0.7614511 0.7614511 0.7614511
[1529] 0.7614511 0.7614511 0.8576798 0.8576798 0.8576798 0.8576798 0.8576798 0.8576798
[1537] 0.7381453 0.7381453 0.7381453 0.7381453 0.6080315 0.6080315 0.6080315 0.6080315
[1545] 0.8518193 0.8518193 0.8518193 0.8518193 0.8518193 0.8518193 0.8518193 0.8518193
[1553] 0.8518193 0.8518193 0.8518193 0.8518193 0.8518193 0.8697977 0.8697977 0.8697977
[1561] 0.8097855 0.8097855 0.8097855 0.8097855 0.8097855 0.8097855 0.8097855 0.8097855
[1569] 0.8537855 0.8537855 0.8537855 0.7503703 0.7503703 0.7503703 0.8193715 0.8193715
[1577] 0.8193715 0.8193715 0.8193715 0.8193715 0.8193715 0.8193715 0.8193715 0.8193715
[1585] 0.8193715 0.8193715 0.8193715 0.8193715 0.5496408 0.5496408 0.5496408 0.5496408
[1593] 0.5496408 0.5496408 0.5496408 0.5496408 0.5496408 0.5496408 0.5496408 0.5496408
[1601] 0.5496408 0.5496408 0.8097366 0.8097366 0.8097366 0.8097366 0.8097366 0.8097366
[1609] 0.8097366 0.8097366 0.8097366 0.8481020 0.8481020 0.8030444 0.8030444 0.8030444
[1617] 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444
[1625] 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444 0.8030444 0.7240994
[1633] 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994
[1641] 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994 0.7240994 0.7399457
[1649] 0.7399457 0.6112552 0.6112552 0.6112552 0.6112552 0.6112552 0.6112552 0.6112552
[1657] 0.7982146 0.7260603 0.7260603 0.7260603 0.6891870 0.6891870 0.7516748 0.7516748
[1665] 0.7516748 0.7516748 0.7516748 0.7516748 0.7516748 0.7516748 0.8252702 0.8252702
[1673] 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702
[1681] 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702
[1689] 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702 0.8252702
[1697] 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952
[1705] 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952
```

```
[1713] 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952 0.7912952
[1721] 0.7912952 0.7912952 0.8195886 0.8195886 0.8195886 0.8195886 0.8195886 0.8195886
[1729] 0.8195886 0.8195886 0.7518643 0.7518643 0.7518643 0.7518643 0.8955180 0.8955180
[1737] 0.8955180 0.8955180 0.8955180 0.8955180 0.8955180 0.8125935 0.8125935 0.8125935
[1745] 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935
[1753] 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935 0.8125935 0.8702680
[1761] 0.8702680 0.8702680 0.8702680 0.8702680 0.8702680 0.8702680 0.8702680 0.8702680
[1769] 0.6065116 0.6065116 0.6065116 0.6065116 0.6065116 0.6065116 0.6065116 0.6065116
[1777] 0.6065116 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979
[1785] 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979
[1793] 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7844979 0.7690863
[1801] 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863
[1809] 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863
[1817] 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863
[1825] 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7690863 0.7279372
[1833] 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372
[1841] 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372
[1849] 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.7279372 0.8901228
[1857] 0.8901228 0.8901228 0.8901228 0.8901228 0.7237655 0.7237655 0.7237655 0.7237655
[1865] 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655
[1873] 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655 0.7237655
[1881] 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537
[1889] 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537 0.8451537
[1897] 0.8451537 0.8451537 0.7672208 0.7672208 0.7672208 0.7672208 0.7672208 0.7672208
[1905] 0.8428550 0.8428550 0.8428550 0.8428550 0.8428550 0.8428550 0.8428550 0.8428550
[1913] 0.8428550 0.8428550 0.8428550 0.8428550 0.8428550 0.8428550 0.7787243 0.7787243
[1921] 0.7787243 0.7787243 0.7787243 0.7787243 0.7787243 0.7787243 0.7787243 0.7787243
[1929] 0.7787243 0.8791936 0.8791936 0.8791936 0.8424518 0.8424518 0.8424518 0.8424518
[1937] 0.8424518 0.8424518 0.8424518 0.8424518 0.8424518 0.8424518 0.8424518 0.8424518
[1945] 0.8424518 0.8424518 0.7654779 0.7654779 0.7654779 0.7654779 0.7654779 0.7654779
[1953] 0.7654779 0.7654779 0.7654779 0.7654779 0.7654779 0.7654779 0.6756994 0.6756994
[1961] 0.6756994 0.6756994 0.6756994 0.6756994 0.6756994 0.6756994 0.6756994 0.6756994
[1969] 0.8932125 0.8932125 0.8932125 0.8932125 0.8932125 0.8932125 0.8932125 0.8932125
[1977] 0.8932125 0.8932125 0.8932125 0.7345625 0.7345625 0.7345625 0.7345625 0.7345625
[1985] 0.7345625 0.6334551 0.8383120 0.8383120 0.8383120 0.8383120 0.8383120 0.8383120
[1993] 0.8383120 0.8258520 0.8258520 0.8258520 0.8258520 0.8258520 0.8258520 0.8258520
[2001] 0.7090726 0.7090726 0.7090726 0.7090726 0.7090726 0.7090726 0.7090726 0.7090726
[2009] 0.7090726 0.7090726 0.7090726 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892
[2017] 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892
[2025] 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892 0.6581892
[2033] 0.7999364 0.7999364 0.7999364 0.7999364 0.7999364 0.8685474 0.8685474 0.8685474
[2041] 0.8685474 0.8685474
```

```r
(y <- rbinom(sum(persons), 1, prob))
```

```
  [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [43] 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
 [85] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1
[127] 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1
[169] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1
[211] 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[253] 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1
[295] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[337] 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1
[379] 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 0
```

```
 [421] 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1
 [463] 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1
 [505] 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1
 [547] 1 0 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0
 [589] 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 1
 [631] 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1
 [673] 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
 [715] 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1
 [757] 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1
 [799] 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1
 [841] 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 [883] 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1
 [925] 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1
 [967] 1 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 1
[1009] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1051] 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1
[1093] 1 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1
[1135] 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1
[1177] 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
[1219] 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1
[1261] 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1
[1303] 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0
[1345] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
[1387] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1
[1429] 1 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0
[1471] 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 0 1
[1513] 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1
[1555] 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1
[1597] 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1
[1639] 0 1 0 1 1 1 1 1 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0
[1681] 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1
[1723] 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0
[1765] 1 1 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 1 1 0 1
[1807] 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0
[1849] 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1891] 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1 1
[1933] 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1
[1975] 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1
[2017] 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1
```

```r
# create
d <- as.data.frame(cbind(y, d))
d$country <- as.factor(d$country)
d$site <- as.factor(d$site)
```

Frequentist random effects model

```r
require(lme4)
fit0 = glmer(data = d, formula = y ~ (1 | country/site), family = binomial)
print(fit0)
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace Approximation) [glmerMod
]
 Family: binomial  ( logit )
```

```
Formula: y ~ (1 | country/site)
   Data: d
      AIC       BIC    logLik  deviance  df.resid
 2050.808  2067.673 -1022.404  2044.808      2039
Random effects:
 Groups       Name        Std.Dev.
 site:country (Intercept) 0.1978
 country      (Intercept) 0.2604
Number of obs: 2042, groups:  site:country, 179; country, 25
Fixed Effects:
(Intercept)
      1.389
```

```r
# (fit0fci <- confint(fit0)) # takes some time

coef(fit0)
```

```
$`site:country`
       (Intercept)
1:1       1.385498
2:1       1.484694
3:1       1.412282
4:1       1.401705
5:1       1.395458
6:1       1.337437
7:1       1.452336
8:2       1.401972
9:3       1.411550
10:3      1.382191
11:3      1.396704
12:3      1.422013
13:3      1.357623
14:3      1.426051
15:3      1.480903
16:3      1.256839
17:3      1.380666
18:3      1.395436
19:4      1.378525
20:4      1.395229
21:4      1.354507
22:4      1.396165
23:4      1.455959
24:4      1.418930
25:4      1.408233
26:4      1.412376
27:4      1.375200
28:4      1.430101
29:4      1.451735
30:4      1.474995
31:4      1.418930
32:4      1.381948
33:4      1.443166
34:4      1.361482
35:4      1.416751
36:4      1.384893
```

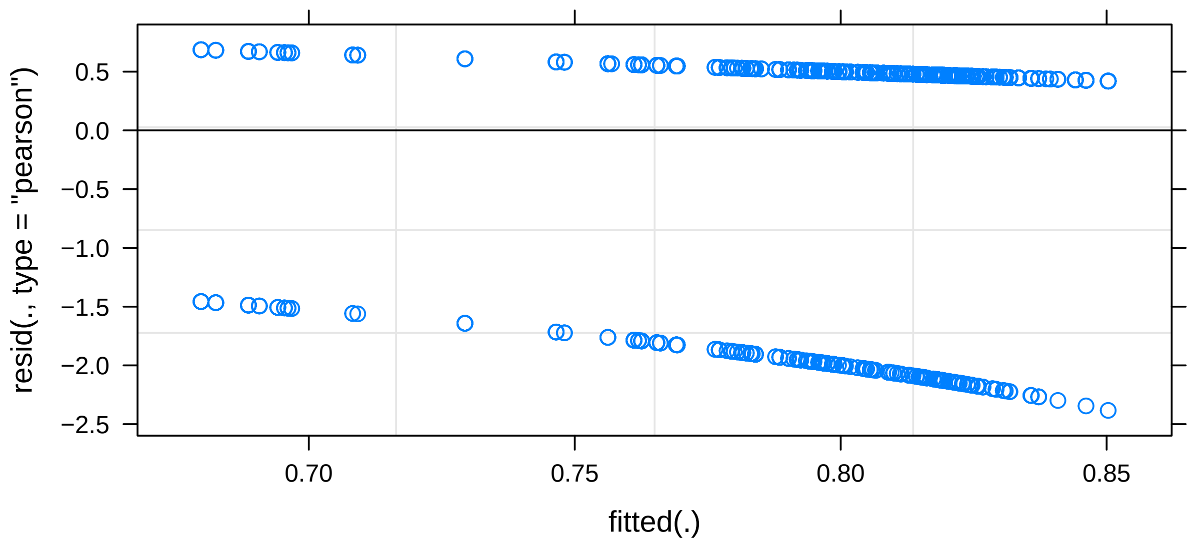
37:4	1.412376
38:4	1.243125
39:4	1.345014
40:4	1.408233
41:4	1.297880
42:4	1.345014
43:5	1.331987
44:5	1.370221
45:5	1.425029
46:5	1.290742
47:5	1.303499
48:5	1.341580
49:5	1.366906
50:5	1.357830
51:5	1.429645
52:5	1.363743
53:6	1.414292
54:7	1.401972
55:8	1.415758
56:8	1.349147
57:8	1.423328
58:8	1.411116
59:8	1.429003
60:8	1.342502
61:8	1.458086
62:8	1.387599
63:8	1.441254
64:8	1.379491
65:8	1.377507
66:8	1.368668
67:8	1.385946
68:8	1.393784
69:8	1.441254
70:8	1.377507
71:8	1.387599
72:8	1.409213
73:8	1.312399
74:9	1.399472
75:9	1.415477
76:9	1.437073
77:9	1.385747
78:9	1.392559
79:9	1.412899
80:9	1.336768
81:9	1.437073
82:9	1.416989
83:9	1.399472
84:9	1.364571
85:9	1.336768
86:10	1.276615
87:10	1.399456
88:10	1.319278
89:10	1.387289
90:10	1.568286

91:10	1.418207
92:11	1.432745
93:11	1.403594
94:11	1.418095
95:11	1.366411
96:11	1.373551
97:11	1.396838
98:11	1.338661
99:12	1.353986
100:12	1.383836
101:12	1.401208
102:12	1.400388
103:12	1.414079
104:12	1.397452
105:12	1.420419
106:12	1.393440
107:13	1.388563
108:13	1.391112
109:13	1.412787
110:13	1.374414
111:13	1.401096
112:13	1.499391
113:13	1.375858
114:13	1.420475
115:13	1.375858
116:13	1.464733
117:13	1.414049
118:13	1.442991
119:13	1.427899
120:13	1.368092
121:13	1.375858
122:13	1.285218
123:13	1.362077
124:13	1.194877
125:13	1.450322
126:13	1.413336
127:13	1.309667
128:13	1.382335
129:13	1.398606
130:14	1.323578
131:14	1.344865
132:14	1.344865
133:14	1.451395
134:14	1.413257
135:14	1.339952
136:14	1.413257
137:14	1.413257
138:14	1.422622
139:15	1.263577
140:15	1.360500
141:15	1.408156
142:15	1.404317
143:15	1.352232
144:16	1.405953

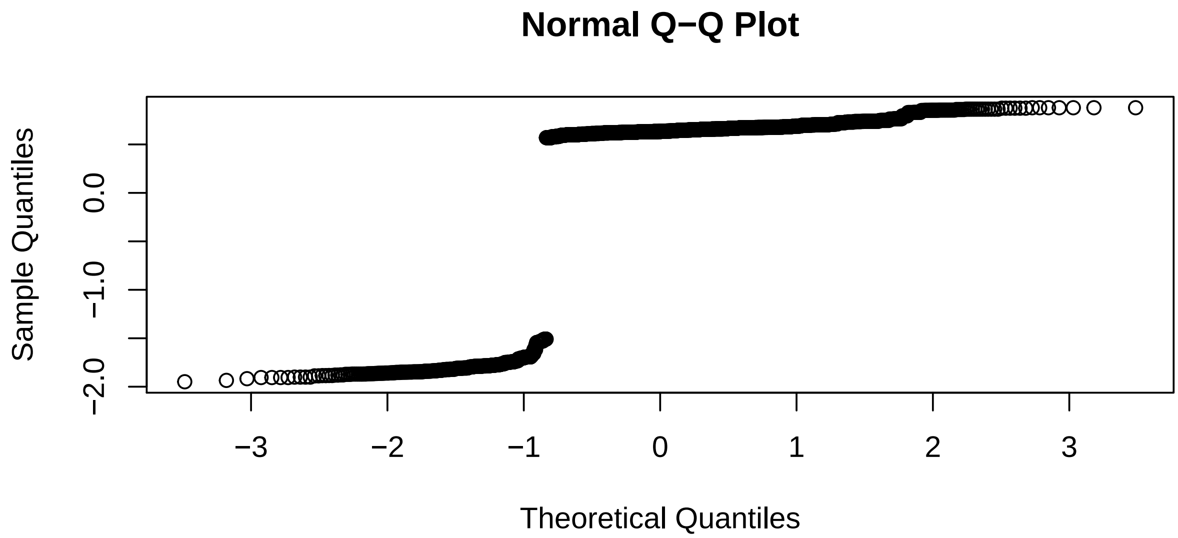
```
145:16    1.296750
146:16    1.397605
147:16    1.375852
148:16    1.405953
149:17    1.374801
150:17    1.431138
151:17    1.330403
152:17    1.412035
153:18    1.345254
154:19    1.410635
155:19    1.356145
156:19    1.426963
157:19    1.316491
158:19    1.319677
159:19    1.427829
160:19    1.335559
161:19    1.431722
162:20    1.388880
163:21    1.498872
164:22    1.360733
165:23    1.408772
166:23    1.389059
167:23    1.410088
168:23    1.372606
169:23    1.432293
170:24    1.322949
171:24    1.404122
172:24    1.363403
173:24    1.397660
174:25    1.366097
175:25    1.366097
176:25    1.395595
177:25    1.359924
178:25    1.426676
179:25    1.426676

$country
   (Intercept)
1    1.6413322
2    1.4113943
3    1.4212727
4    1.5127757
5    0.8509702
6    1.4327620
7    1.4113943
8    1.5580859
9    1.5021000
10   1.4485697
11   1.3993389
12   1.4787266
13   1.3531880
14   1.3279862
15   1.1169206
16   1.2788070
```

```
17   1.3749135
18   1.3130145
19   1.2362595
20   1.3886845
21   1.5794687
22   1.3398641
23   1.5055160
24   1.2704204
25   1.3998884

attr(,"class")
[1] "coef.mer"
```

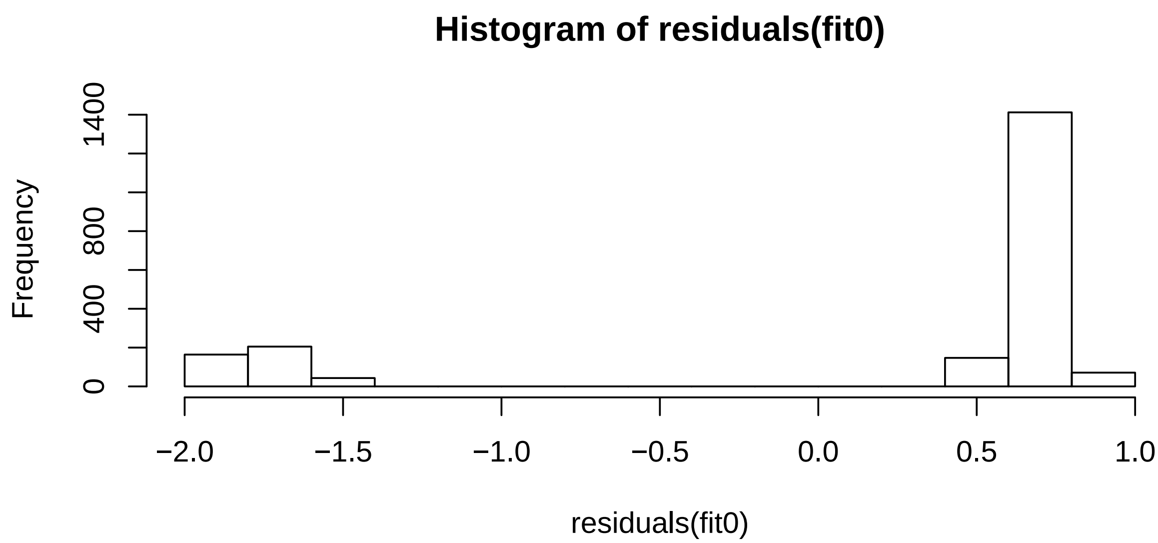
```
# are these sensible from binomial?
plot(fit0)
```




```
qqnorm(residuals(fit0))
```




```
hist(residuals(fit0))
```




```
fit0r <- ranef(fit0, condVar = TRUE)  # frequentist estimates of random effects

coef(fit0)
```

```
$`site:country`
       (Intercept)
1:1       1.385498
2:1       1.484694
3:1       1.412282
4:1       1.401705
5:1       1.395458
6:1       1.337437
```

```
7:1      1.452336
8:2      1.401972
9:3      1.411550
10:3     1.382191
11:3     1.396704
12:3     1.422013
13:3     1.357623
14:3     1.426051
15:3     1.480903
16:3     1.256839
17:3     1.380666
18:3     1.395436
19:4     1.378525
20:4     1.395229
21:4     1.354507
22:4     1.396165
23:4     1.455959
24:4     1.418930
25:4     1.408233
26:4     1.412376
27:4     1.375200
28:4     1.430101
29:4     1.451735
30:4     1.474995
31:4     1.418930
32:4     1.381948
33:4     1.443166
34:4     1.361482
35:4     1.416751
36:4     1.384893
37:4     1.412376
38:4     1.243125
39:4     1.345014
40:4     1.408233
41:4     1.297880
42:4     1.345014
43:5     1.331987
44:5     1.370221
45:5     1.425029
46:5     1.290742
47:5     1.303499
48:5     1.341580
49:5     1.366906
50:5     1.357830
51:5     1.429645
52:5     1.363743
53:6     1.414292
54:7     1.401972
55:8     1.415758
56:8     1.349147
57:8     1.423328
58:8     1.411116
59:8     1.429003
60:8     1.342502
```

61:8	1.458086
62:8	1.387599
63:8	1.441254
64:8	1.379491
65:8	1.377507
66:8	1.368668
67:8	1.385946
68:8	1.393784
69:8	1.441254
70:8	1.377507
71:8	1.387599
72:8	1.409213
73:8	1.312399
74:9	1.399472
75:9	1.415477
76:9	1.437073
77:9	1.385747
78:9	1.392559
79:9	1.412899
80:9	1.336768
81:9	1.437073
82:9	1.416989
83:9	1.399472
84:9	1.364571
85:9	1.336768
86:10	1.276615
87:10	1.399456
88:10	1.319278
89:10	1.387289
90:10	1.568286
91:10	1.418207
92:11	1.432745
93:11	1.403594
94:11	1.418095
95:11	1.366411
96:11	1.373551
97:11	1.396838
98:11	1.338661
99:12	1.353986
100:12	1.383836
101:12	1.401208
102:12	1.400388
103:12	1.414079
104:12	1.397452
105:12	1.420419
106:12	1.393440
107:13	1.388563
108:13	1.391112
109:13	1.412787
110:13	1.374414
111:13	1.401096
112:13	1.499391
113:13	1.375858
114:13	1.420475

115:13	1.375858
116:13	1.464733
117:13	1.414049
118:13	1.442991
119:13	1.427899
120:13	1.368092
121:13	1.375858
122:13	1.285218
123:13	1.362077
124:13	1.194877
125:13	1.450322
126:13	1.413336
127:13	1.309667
128:13	1.382335
129:13	1.398606
130:14	1.323578
131:14	1.344865
132:14	1.344865
133:14	1.451395
134:14	1.413257
135:14	1.339952
136:14	1.413257
137:14	1.413257
138:14	1.422622
139:15	1.263577
140:15	1.360500
141:15	1.408156
142:15	1.404317
143:15	1.352232
144:16	1.405953
145:16	1.296750
146:16	1.397605
147:16	1.375852
148:16	1.405953
149:17	1.374801
150:17	1.431138
151:17	1.330403
152:17	1.412035
153:18	1.345254
154:19	1.410635
155:19	1.356145
156:19	1.426963
157:19	1.316491
158:19	1.319677
159:19	1.427829
160:19	1.335559
161:19	1.431722
162:20	1.388880
163:21	1.498872
164:22	1.360733
165:23	1.408772
166:23	1.389059
167:23	1.410088
168:23	1.372606

```
169:23    1.432293
170:24    1.322949
171:24    1.404122
172:24    1.363403
173:24    1.397660
174:25    1.366097
175:25    1.366097
176:25    1.395595
177:25    1.359924
178:25    1.426676
179:25    1.426676

$country
   (Intercept)
1    1.6413322
2    1.4113943
3    1.4212727
4    1.5127757
5    0.8509702
6    1.4327620
7    1.4113943
8    1.5580859
9    1.5021000
10   1.4485697
11   1.3993389
12   1.4787266
13   1.3531880
14   1.3279862
15   1.1169206
16   1.2788070
17   1.3749135
18   1.3130145
19   1.2362595
20   1.3886845
21   1.5794687
22   1.3398641
23   1.5055160
24   1.2704204
25   1.3998884

attr(,"class")
[1] "coef.mer"
```

```
ranef(fit0, condVar = TRUE)
```

```
$`site:country`
          (Intercept)
1:1    -0.00364748371
2:1     0.09554850929
3:1     0.02313631379
4:1     0.01255957288
5:1     0.00631278534
6:1    -0.05170803803
7:1     0.06319073587
8:2     0.01282708240
```

9:3	0.02240498693
10:3	-0.00695476438
11:3	0.00755844437
12:3	0.03286772761
13:3	-0.03152242004
14:3	0.03690554107
15:3	0.09175727162
16:3	-0.13230644031
17:3	-0.00847915025
18:3	0.00629104005
19:4	-0.01062051991
20:4	0.00608315180
21:4	-0.03463802379
22:4	0.00701920542
23:4	0.06681402684
24:4	0.02978447115
25:4	0.01908753722
26:4	0.02323088451
27:4	-0.01394519983
28:4	0.04095526940
29:4	0.06258957527
30:4	0.08584954532
31:4	0.02978447115
32:4	-0.00719745443
33:4	0.05402060170
34:4	-0.02766376126
35:4	0.02760569061
36:4	-0.00425242410
37:4	0.02323088451
38:4	-0.14602043256
39:4	-0.04413160851
40:4	0.01908753722
41:4	-0.09126564876
42:4	-0.04413160851
43:5	-0.05715862221
44:5	-0.01892481676
45:5	0.03588349488
46:5	-0.09840355128
47:5	-0.08564616931
48:5	-0.04756546075
49:5	-0.02223928538
50:5	-0.03131505340
51:5	0.04049997191
52:5	-0.02540271041
53:6	0.02514613365
54:7	0.01282708240
55:8	0.02661259496
56:8	-0.03999853759
57:8	0.03418273515
58:8	0.02197026630
59:8	0.03985802799
60:8	-0.04664363614
61:8	0.06894025110
62:8	-0.00154683068

63:8	0.05210849029
64:8	-0.00965414331
65:8	-0.01163796575
66:8	-0.02047729144
67:8	-0.00319897811
68:8	0.00463834824
69:8	0.05210849029
70:8	-0.01163796575
71:8	-0.00154683068
72:8	0.02006808187
73:8	-0.07674644385
74:9	0.01032619797
75:9	0.02633196644
76:9	0.04792739615
77:9	-0.00339845734
78:9	0.00341316202
79:9	0.02375340977
80:9	-0.05237779366
81:9	0.04792739615
82:9	0.02784377922
83:9	0.01032619797
84:9	-0.02457410840
85:9	-0.05237779366
86:10	-0.11253021751
87:10	0.01031082037
88:10	-0.06986711534
89:10	-0.00185600101
90:10	0.17914088411
91:10	0.02906132164
92:11	0.04359944425
93:11	0.01444818461
94:11	0.02894976158
95:11	-0.02273397323
96:11	-0.01559450281
97:11	0.00769226008
98:11	-0.05048434842
99:12	-0.03515909891
100:12	-0.00530913804
101:12	0.01206261530
102:12	0.01124237895
103:12	0.02493412749
104:12	0.00830648910
105:12	0.03127362213
106:12	0.00429496504
107:13	-0.00058198100
108:13	0.00196661399
109:13	0.02364177045
110:13	-0.01473154708
111:13	0.01195072411
112:13	0.11024537757
113:13	-0.01328776557
114:13	0.03132930904
115:13	-0.01328776557
116:13	0.07558781422

```
117:13  0.02490413010
118:13  0.05384570838
119:13  0.03875328400
120:13 -0.02105292127
121:13 -0.01328776557
122:13 -0.10392727795
123:13 -0.02706834678
124:13 -0.19426807669
125:13  0.06117637879
126:13  0.02419074661
127:13 -0.07947880143
128:13 -0.00681020481
129:13  0.00946023396
130:14 -0.06556723037
131:14 -0.04428050574
132:14 -0.04428050574
133:14  0.06224941502
134:14  0.02411208746
135:14 -0.04919354618
136:14  0.02411208746
137:14  0.02411208746
138:14  0.03347623590
139:15 -0.12556821230
140:15 -0.02864557759
141:15  0.01901085096
142:15  0.01517134997
143:15 -0.03691318913
144:16  0.01680807077
145:16 -0.09239541004
146:16  0.00845925179
147:16 -0.01329297684
148:16  0.01680807077
149:17 -0.01434474665
150:17  0.04199292005
151:17 -0.05874276555
152:17  0.02288952360
153:18 -0.04389143138
154:19  0.02148996044
155:19 -0.03300049557
156:19  0.03781714641
157:19 -0.07265445102
158:19 -0.06946863203
159:19  0.03868339424
160:19 -0.05358602756
161:19  0.04257636748
162:20 -0.00026568636
163:21  0.10972642545
164:22 -0.02841193945
165:23  0.01962642262
166:23 -0.00008648411
167:23  0.02094272612
168:23 -0.01653965316
169:23  0.04314770990
170:24 -0.06619681133
```

```
171:24  0.01497616259
172:24 -0.02574206986
173:24  0.00851461344
174:25 -0.02304834554
175:25 -0.02304834554
176:25  0.00644993670
177:25 -0.02922175537
178:25  0.03753107228
179:25  0.03753107228

$country
     (Intercept)
1   0.2521868580
2   0.0222489051
3   0.0321272970
4   0.1236303552
5  -0.5381751343
6   0.0436166171
7   0.0222489051
8   0.1689404917
9   0.1129546649
10  0.0594243194
11  0.0101935063
12  0.0895812509
13 -0.0359573482
14 -0.0611591617
15 -0.2722247636
16 -0.1103383772
17 -0.0142319029
18 -0.0761308192
19 -0.1528858507
20 -0.0004608398
21  0.1903233137
22 -0.0492812414
23  0.1163705858
24 -0.1187250030
25  0.0107430192

with conditional variances for "site:country" "country"
```

```r
predFun <- function(fit0) {
    predict(fit0)
}
bb <- bootMer(fit0, nsim = 200, FUN = predFun, seed = 101)

# https://stats.stackexchange.com/questions/147836/prediction-interval-for-lmer-mixed-effects-model-in-
c(attr(ranef(fit0, condVar = TRUE)[[1]], "postVar"))  # site variances
```

```
  [1] 0.03519550 0.03551873 0.03726197 0.03870246 0.03890121 0.03828261 0.03624946
  [8] 0.03671605 0.03841970 0.03260337 0.03887015 0.03608905 0.03672189 0.03799565
 [15] 0.03652405 0.03403638 0.03818404 0.03776379 0.03822557 0.03665667 0.03778847
 [22] 0.03888270 0.03710603 0.03706096 0.03631403 0.03724346 0.03720190 0.03784766
 [29] 0.03522934 0.03559362 0.03706096 0.03701578 0.03746805 0.03758806 0.03824620
 [36] 0.03416130 0.03724346 0.03478082 0.03697056 0.03631403 0.03440850 0.03697056
 [43] 0.03386322 0.03759758 0.03633487 0.03525120 0.03572912 0.03650483 0.03678373
```

```
 [50] 0.03468702 0.03709420 0.03603084 0.03643868 0.03671605 0.03827158 0.03685426
 [57] 0.03593088 0.03623971 0.03678190 0.03703344 0.03424914 0.03586573 0.03751756
 [64] 0.03707814 0.03825117 0.03634232 0.03690232 0.03571410 0.03751756 0.03825117
 [71] 0.03586573 0.03847217 0.03680616 0.03763445 0.03622196 0.03766719 0.03802396
 [78] 0.03586061 0.03726680 0.03626908 0.03766719 0.03824778 0.03763445 0.03865331
 [85] 0.03626908 0.03596130 0.03666053 0.03674010 0.03799473 0.03400070 0.03822289
 [92] 0.03376499 0.03755575 0.03717376 0.03654726 0.03839732 0.03886728 0.03547253
 [99] 0.03680167 0.03702370 0.03568304 0.03761593 0.03540122 0.03667699 0.03526586
[106] 0.03780991 0.03267784 0.03682438 0.03838735 0.03837133 0.03314818 0.03600856
[113] 0.03723352 0.03816043 0.03723352 0.03691516 0.03363638 0.03751414 0.03686971
[120] 0.03744613 0.03723352 0.03564576 0.03657885 0.03102117 0.03730949 0.03624790
[127] 0.03694337 0.03813956 0.03662744 0.03765291 0.03812196 0.03812196 0.03640749
[134] 0.03838580 0.03723502 0.03838580 0.03838580 0.03619325 0.03567498 0.03682919
[141] 0.03855936 0.03514551 0.03541714 0.03860651 0.03740400 0.03885098 0.03835675
[148] 0.03860651 0.03731024 0.03449743 0.03429771 0.03734593 0.03812986 0.03740342
[155] 0.03512202 0.03698266 0.03686492 0.03436684 0.03312898 0.03407490 0.03787594
[162] 0.03546341 0.03621589 0.03772050 0.03644678 0.03691875 0.03845749 0.03639498
[169] 0.03680364 0.03678597 0.03668513 0.03765465 0.03884953 0.03753345 0.03753345
[176] 0.03679215 0.03512218 0.03799256 0.03799256
```

```
c(attr(ranef(fit0, condVar = TRUE)[[2]], "postVar"))  # country variances
```

```
 [1] 0.04076525 0.06063998 0.03128437 0.01913590 0.02730659 0.05980552 0.06063998
 [8] 0.02263608 0.03352534 0.03898749 0.03796602 0.03422382 0.01716208 0.04201540
[15] 0.04026641 0.05787861 0.04095207 0.06489354 0.02925684 0.05687134 0.05913522
[22] 0.06366196 0.04508305 0.05178061 0.04342486
```

Use sandwich approach in reference , although I think it can only handle one cluster

```
require(rms)
dd <- datadist(d)  #  Run for all potential vars.
options(datadist = "dd")

d$tx <- rnorm(sum(persons), 0, 1)  # need to do this otherwise intercept model has no covar matrix
o <- try(lrm(y ~ 1 + tx, x = TRUE, y = TRUE, d))
(v <- robcov(fit = o, cluster = d[, c(2)]))  # can I cluster on multiple variances - no?
```

```
Logistic Regression Model

 lrm(formula = y ~ 1 + tx, data = d, x = TRUE, y = TRUE)

                         Model Likelihood     Discrimination    Rank Discrim.
                            Ratio Test           Indexes           Indexes
 Obs              2042    LR chi2      3.08    R2       0.002    C       0.525
  0                412    d.f.            1    g        0.109    Dxy     0.049
  1               1630    Pr(> chi2) 0.0794    gr       1.115    gamma   0.049
 Cluster on   d[, c(2)]                        gp       0.018    tau-a   0.016
 Clusters           25                         Brier    0.161
 max |deriv|     5e-10

           Coef    S.E.   Wald Z Pr(>|Z|)
 Intercept  1.3745 0.0825 16.66  <0.0001
 tx        -0.0976 0.0443 -2.20  0.0275
```

```r
summary(v)
```

```
             Effects              Response : y

 Factor      Low High Diff. Effect S.E. Lower 0.95 Upper 0.95
 tx          1   1    0     0        0  0          0
  Odds Ratio 1   1    0     1       NA  1          1
```

Analysis ignoring clustering

```r
binom::binom.confint(sum(d$y == 1), length(d$y))
```

```
          method    x    n     mean     lower     upper
1  agresti-coull 1630 2042 0.798237 0.7802691 0.8150850
2     asymptotic 1630 2042 0.798237 0.7808307 0.8156433
3          bayes 1630 2042 0.798091 0.7806143 0.8154041
4        cloglog 1630 2042 0.798237 0.7801683 0.8150013
5          exact 1630 2042 0.798237 0.7801632 0.8154503
6          logit 1630 2042 0.798237 0.7802690 0.8150839
7         probit 1630 2042 0.798237 0.7803848 0.8151907
8        profile 1630 2042 0.798237 0.7804630 0.8152632
9            lrt 1630 2042 0.798237 0.7804371 0.8152611
10     prop.test 1630 2042 0.798237 0.7800258 0.8153125
11        wilson 1630 2042 0.798237 0.7802780 0.8150760
```

Bayesian

```r
require(brms)
require(rstan)

rstan_options(auto_write = TRUE)
# options(mc.cores = parallel::detectCores())

fit = brm(formula = y ~ (1 | country/site), family = bernoulli,
    data = d, seed = 123, )
```

```
SAMPLING FOR MODEL '5eccc82172b36984db900788f26aa760' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 11.409 seconds (Warm-up)
Chain 1:                9.911 seconds (Sampling)
Chain 1:                21.32 seconds (Total)
Chain 1:

SAMPLING FOR MODEL '5eccc82172b36984db900788f26aa760' NOW (CHAIN 2).
Chain 2:
Chain 2: Gradient evaluation took 0 seconds
Chain 2: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 11.082 seconds (Warm-up)
Chain 2:                20.149 seconds (Sampling)
Chain 2:                31.231 seconds (Total)
Chain 2:

SAMPLING FOR MODEL '5eccc82172b36984db900788f26aa760' NOW (CHAIN 3).
Chain 3:
Chain 3: Gradient evaluation took 0 seconds
Chain 3: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 11.149 seconds (Warm-up)
Chain 3:                12.038 seconds (Sampling)
Chain 3:                23.187 seconds (Total)
Chain 3: 

SAMPLING FOR MODEL '5eccc82172b36984db900788f26aa760' NOW (CHAIN 4).
Chain 4: 
Chain 4: Gradient evaluation took 0 seconds
Chain 4: 1000 transitions using 10 leapfrog steps per transition would take 0 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: 11.29 seconds (Warm-up)
Chain 4:                5.424 seconds (Sampling)
Chain 4:                16.714 seconds (Total)
Chain 4: 
```

```r
print(fit)
```

```
 Family: bernoulli 
  Links: mu = logit 
Formula: y ~ (1 | country/site) 
   Data: d (Number of observations: 2042) 
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects: 
~country (Number of levels: 25) 
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.29      0.11     0.08     0.51       1241 1.00

~country:site (Number of levels: 179) 
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.20      0.12     0.01     0.43        902 1.00

Population-Level Effects: 
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     1.39      0.10     1.21     1.59       2611 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample 
is a crude measure of effective sample size, and Rhat is the potential 
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
f <- fit  # selected model, default priors used

# save the workspace save.image(file= 'brms bernouli
# model.RData' )
```

Load Bayesian analysis

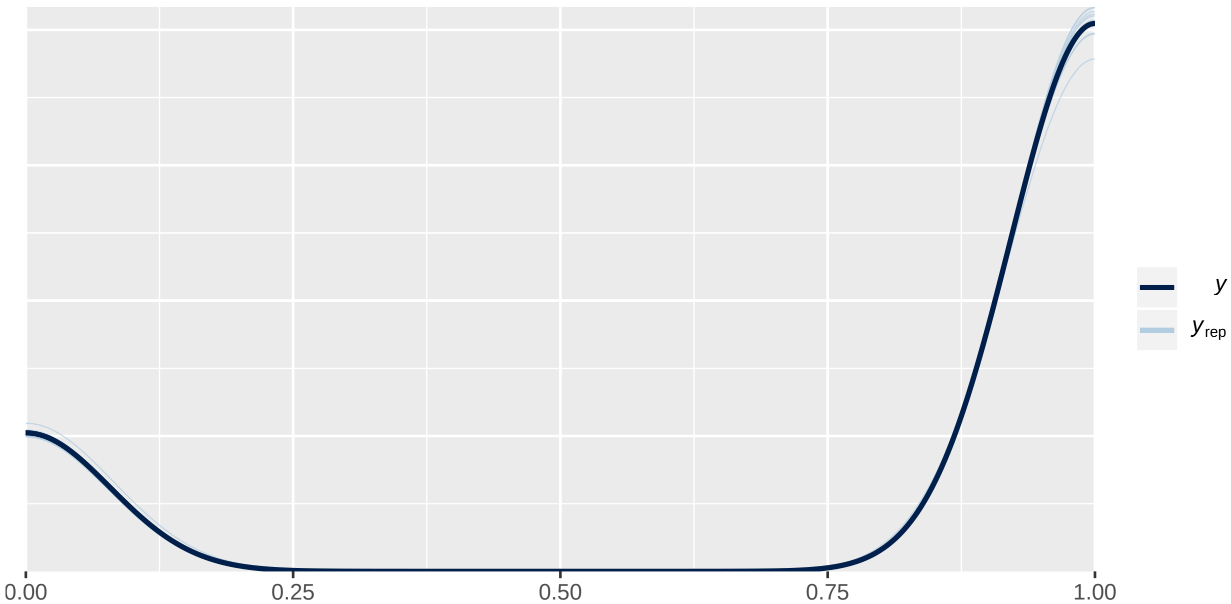
```r
# load(file='brms bernouli model.RData' ) #site sd=2.5,
# country=0.5 library(brms)
```

Check Model

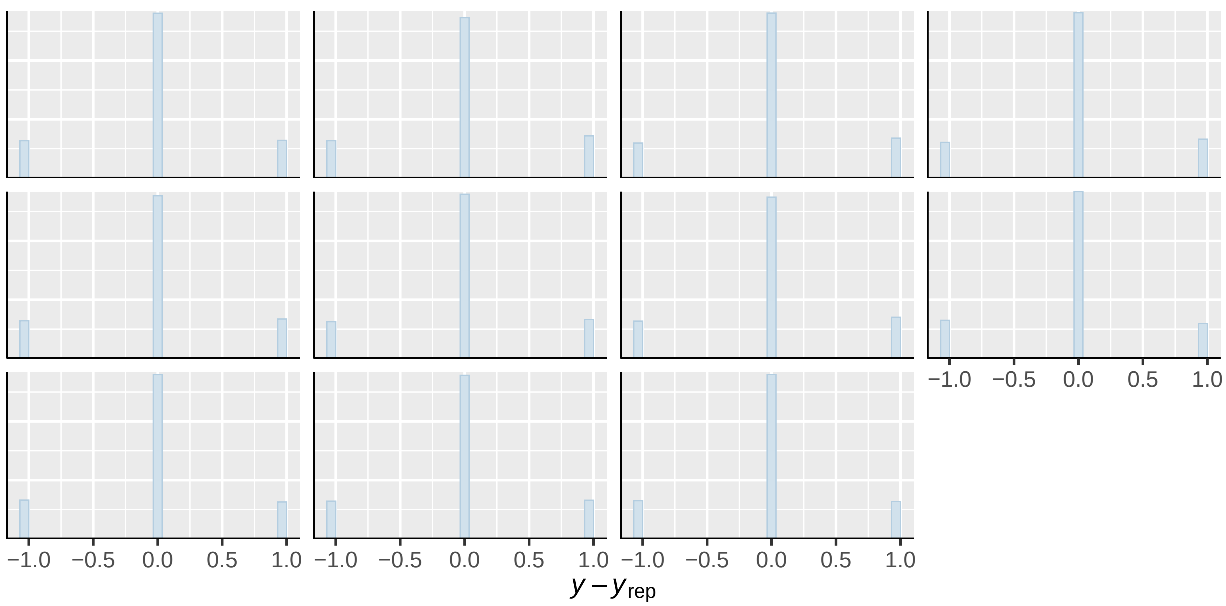
```r
print(f)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: y ~ (1 | country/site)
   Data: d (Number of observations: 2042)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000

Group-Level Effects:
~country (Number of levels: 25)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.29      0.11     0.08     0.51       1241 1.00

~country:site (Number of levels: 179)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.20      0.12     0.01     0.43        902 1.00

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     1.39      0.10     1.21     1.59       2611 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

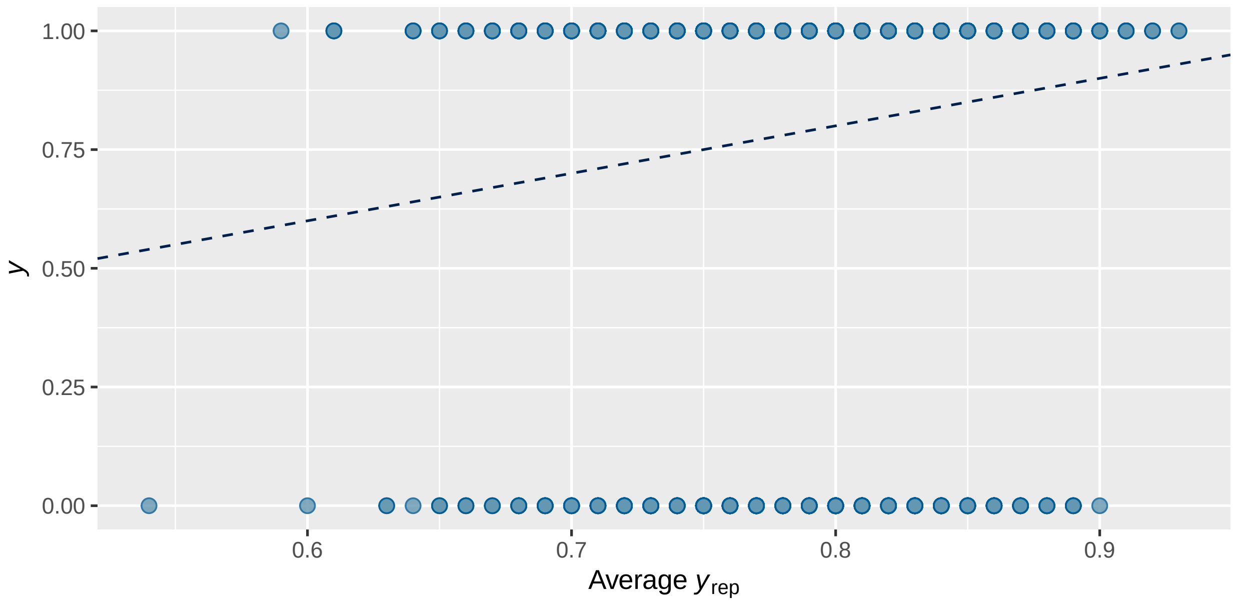
```r
pp_check(f)  # shows dens_overlay plot by default
```



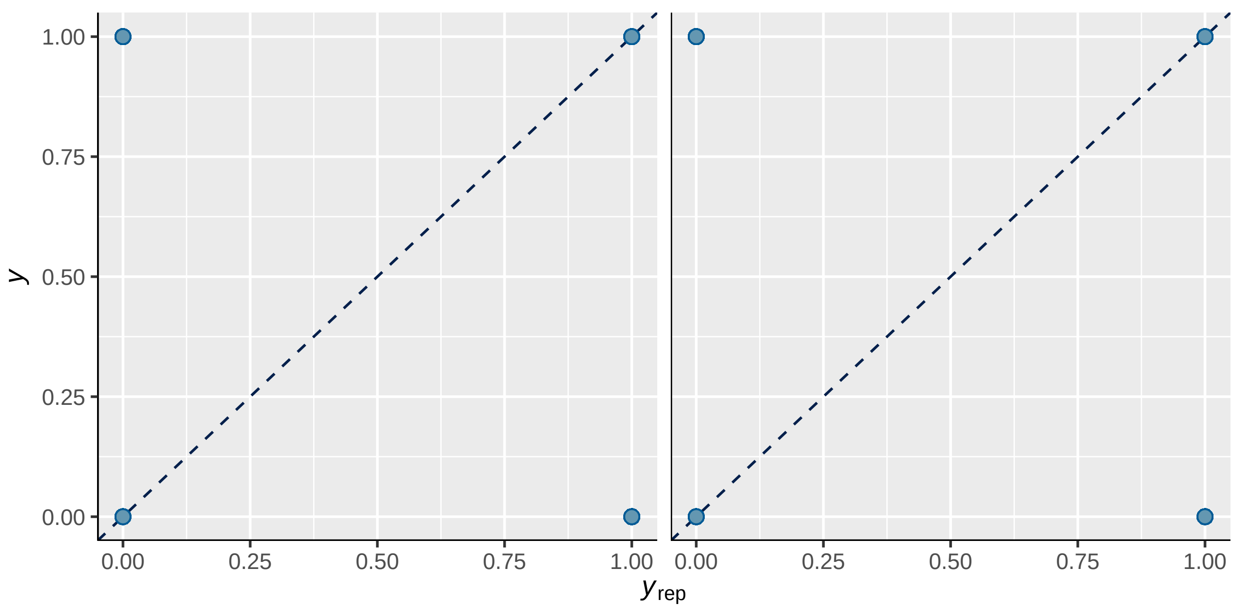

```r
pp_check(f, type = "error_hist", nsamples = 11)   # 11 draws
```



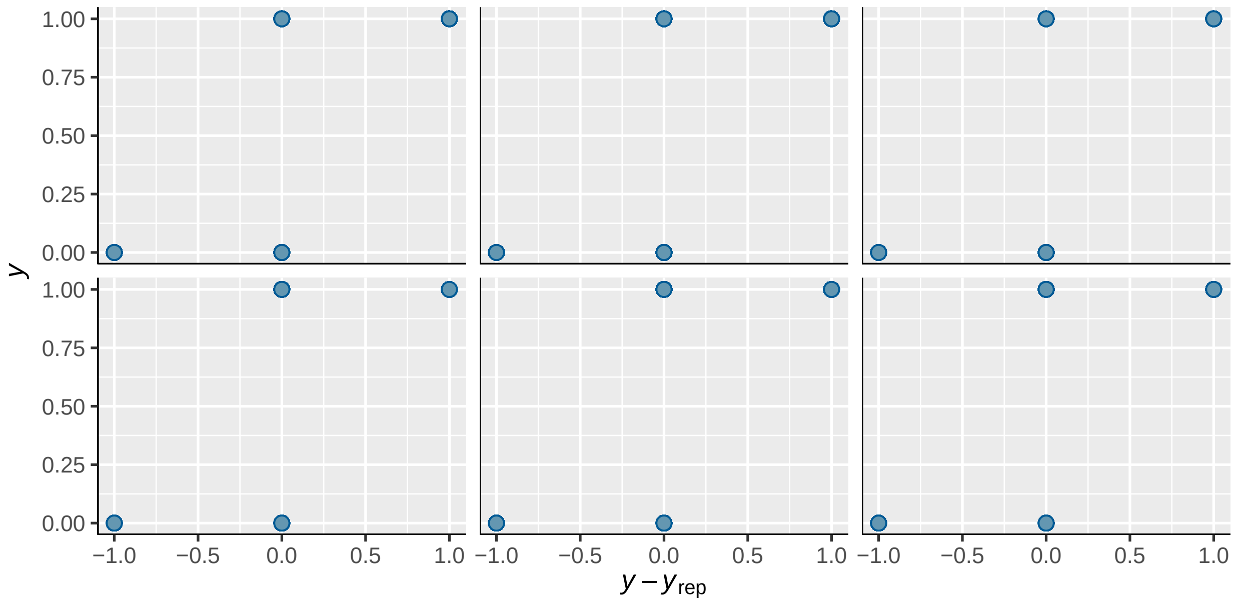

```r
pp_check(f, type = "scatter_avg", nsamples = 100)   # mean on x axis y observed
```



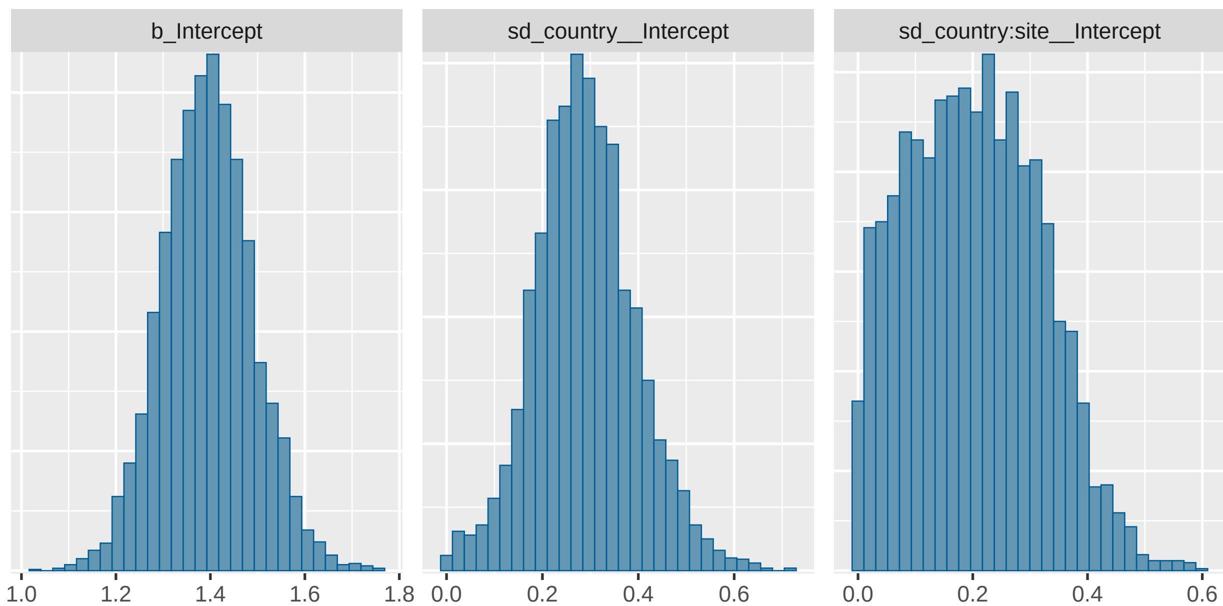

```
# pp_check(f, type = 'stat_2d') ## took a long time so
# cancelled this pp_check(f ,x='vas', type = 'intervals') ##
# took a long time so cancelled this
pp_check(f, type = "scatter", nsamples = 2)
```




```
# pp_check(f,x='avisit',type = 'error_scatter_avg_vs_x',
# nsamples = 10) #throwing error pp_check(f,x='vas', type =
# 'ribbon', nsamples = 20) #throwing error
pp_check(f, type = "error_scatter", nsamples = 6)
```




```r
stanplot(f, type = "hist")
```



Predictions

```r
print(f)
```

```
 Family: bernoulli
  Links: mu = logit
Formula: y ~ (1 | country/site)
   Data: d (Number of observations: 2042)
Samples: 4 chains, each with iter = 2000; warmup = 1000; thin = 1;
         total post-warmup samples = 4000
```

```
Group-Level Effects:
~country (Number of levels: 25)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.29      0.11     0.08     0.51       1241 1.00

~country:site (Number of levels: 179)
              Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
sd(Intercept)     0.20      0.12     0.01     0.43        902 1.00

Population-Level Effects:
          Estimate Est.Error l-95% CI u-95% CI Eff.Sample Rhat
Intercept     1.39      0.10     1.21     1.59       2611 1.00

Samples were drawn using sampling(NUTS). For each parameter, Eff.Sample
is a crude measure of effective sample size, and Rhat is the potential
scale reduction factor on split chains (at convergence, Rhat = 1).
```

```r
f1 <- fitted(f, re_formula = NULL)  # include all random effects
head(f1)  #site level
```

```
      Estimate  Est.Error      Q2.5     Q97.5
[1,] 0.8382657 0.03916884 0.7534587 0.9064717
[2,] 0.8382657 0.03916884 0.7534587 0.9064717
[3,] 0.8382657 0.03916884 0.7534587 0.9064717
[4,] 0.8382657 0.03916884 0.7534587 0.9064717
[5,] 0.8382657 0.03916884 0.7534587 0.9064717
[6,] 0.8382657 0.03916884 0.7534587 0.9064717
```

```r
f2 <- fitted(f, re_formula = NA)  # no random effects
head(f2)  # intercept only
```

```
      Estimate  Est.Error      Q2.5     Q97.5
[1,] 0.8008768 0.01522577 0.7701308 0.8301442
[2,] 0.8008768 0.01522577 0.7701308 0.8301442
[3,] 0.8008768 0.01522577 0.7701308 0.8301442
[4,] 0.8008768 0.01522577 0.7701308 0.8301442
[5,] 0.8008768 0.01522577 0.7701308 0.8301442
[6,] 0.8008768 0.01522577 0.7701308 0.8301442
```

```r
f3 <- predict(f, re_formula = NULL)  # include all random effects
head(f3)  #individual
```

```
     Estimate Est.Error Q2.5 Q97.5
[1,]  0.83900 0.3675769    0     1
[2,]  0.83525 0.3710011    0     1
[3,]  0.82775 0.3776448    0     1
[4,]  0.83450 0.3716777    0     1
[5,]  0.84700 0.3600325    0     1
[6,]  0.84125 0.3654885    0     1
```

```r
f4 <- predict(f, re_formula = NA)  # no random effects
head(f4)  #individual
```

```
     Estimate Est.Error Q2.5 Q97.5
[1,]  0.79775 0.4017279    0     1
[2,]  0.80025 0.3998624    0     1
[3,]  0.79875 0.4009846    0     1
```

```
[4,]  0.79750 0.4019131    0     1
[5,]  0.80600 0.3954783    0     1
[6,]  0.79850 0.4011708    0     1
```

```
f5 <- predict(f, re_formula = ~(1 | country))
head(f5)  #not sure
```

```
     Estimate Est.Error Q2.5 Q97.5
[1,]  0.83450 0.3716777    0     1
[2,]  0.83475 0.3714525    0     1
[3,]  0.83400 0.3721272    0     1
[4,]  0.83900 0.3675769    0     1
[5,]  0.85425 0.3528995    0     1
[6,]  0.84050 0.3661875    0     1
```

```
f6 <- predict(f, re_formula = ~(1 | site))
head(f6)  # individual level
```

```
     Estimate Est.Error Q2.5 Q97.5
[1,]  0.80125 0.3991093    0     1
[2,]  0.80500 0.3962502    0     1
[3,]  0.81150 0.3911598    0     1
[4,]  0.79875 0.4009846    0     1
[5,]  0.79750 0.4019131    0     1
[6,]  0.81525 0.3881431    0     1
```

```
f8 <- fitted(f, re_formula = ~(1 | country))
head(f8)  #country level
```

```
      Estimate  Est.Error      Q2.5    Q97.5
[1,] 0.8388855 0.03201261 0.7747917 0.899565
[2,] 0.8388855 0.03201261 0.7747917 0.899565
[3,] 0.8388855 0.03201261 0.7747917 0.899565
[4,] 0.8388855 0.03201261 0.7747917 0.899565
[5,] 0.8388855 0.03201261 0.7747917 0.899565
[6,] 0.8388855 0.03201261 0.7747917 0.899565
```

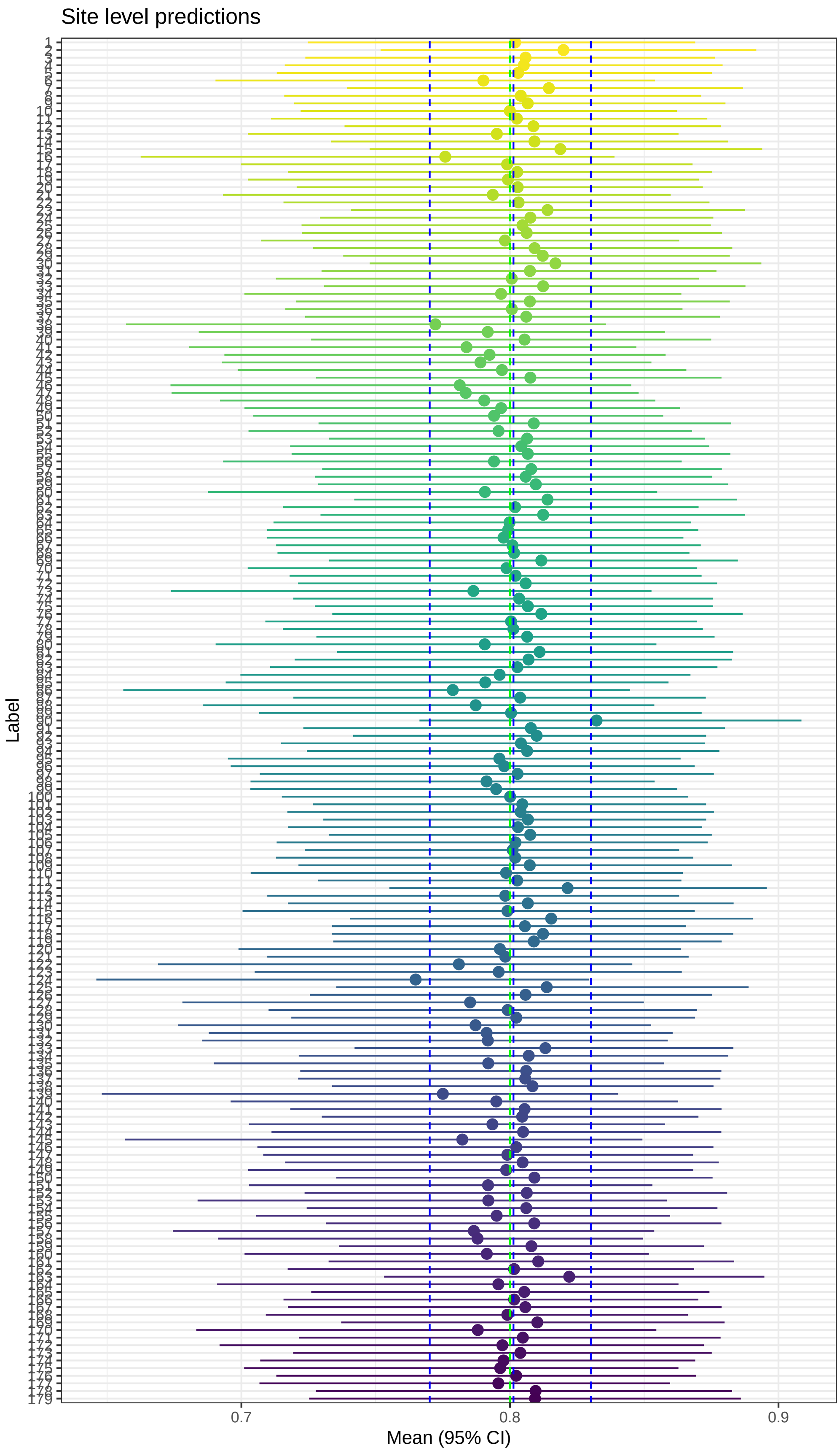
```
f9 <- fitted(f, re_formula = ~(1 | country/site))
head(f9)  #site level only same as f1
```

```
      Estimate  Est.Error      Q2.5     Q97.5
[1,] 0.8382657 0.03916884 0.7534587 0.9064717
[2,] 0.8382657 0.03916884 0.7534587 0.9064717
[3,] 0.8382657 0.03916884 0.7534587 0.9064717
[4,] 0.8382657 0.03916884 0.7534587 0.9064717
[5,] 0.8382657 0.03916884 0.7534587 0.9064717
[6,] 0.8382657 0.03916884 0.7534587 0.9064717
```

Plot Bayesian site level predictions

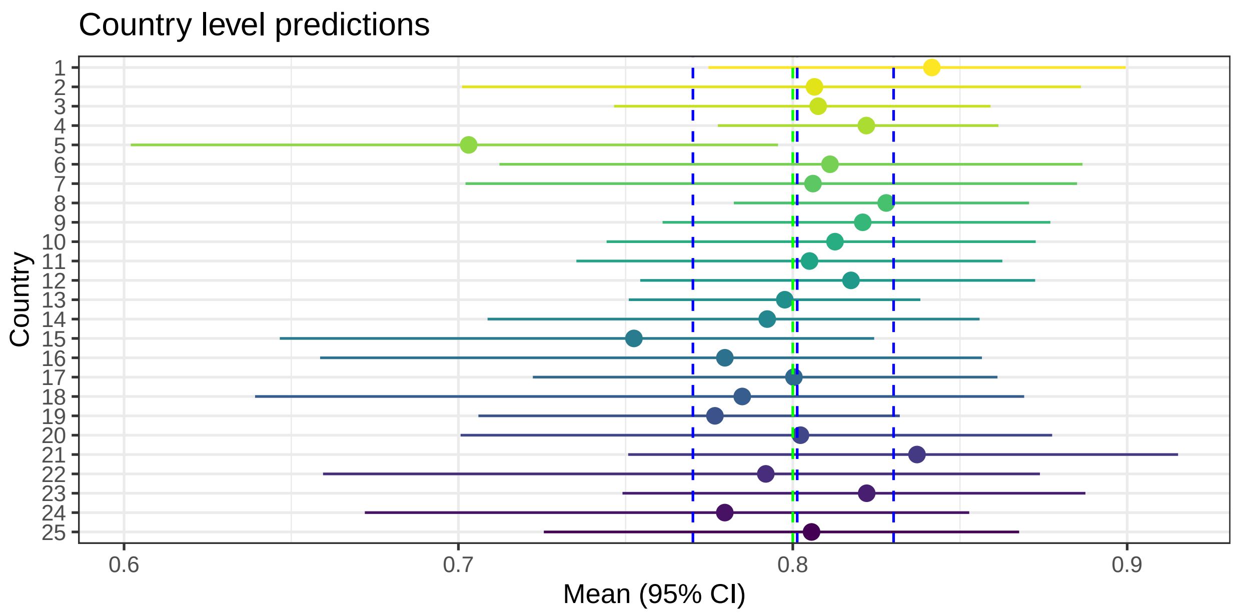
```
    pred <- f1 # f9
    df <- data.frame(cbind(pred, d))
    df$label <- df$site

 # start here use coefficients from model rather than predictions
    A <- function(x) 1/(1+exp(-x))
```

```
  e1 <- A(fixef(fit)[,'Estimate'])
  e2 <- A(fixef(fit)[,'Q2.5'])
  e3 <- A(fixef(fit)[,'Q97.5'])

  bc <- coef(fit, old=FALSE, summary=TRUE)$`country:site`
  df <- as.data.frame(bc)
  names(df) <- dimnames(bc)[[2]]
  df$sites <-     gsub(".*_", "", rownames(df))
  df$countries <- gsub("_.*", "", rownames(df))

  df <- data.frame(df[,c(2,5,6)], apply(df[,c(1,3,4)],2, A) )

  df <- df[,c(3,2,4,1,5,6)]

  df$label <- df$sites
  df$label <- as.numeric(as.character(df$label))
  df <- df[order(df$label),]
  # reverses the factor level ordering for labels after coord_flip()
 # df <- df[order(sites),]
  df$label <- factor(df$label, levels=rev(unique(df$label)), ordered = T)

  # df$label <- factor(   df$label, levels=unique(as.character(   df$label)) )

  fp <- NULL
  fp <- ggplot(data=df, aes(x=label, y=Estimate, ymin=Q2.5, ymax=Q97.5, colour=label)) +
    geom_pointrange() +
       geom_hline(yintercept=mu, color="green",lty=2) +  # add a dotted line

    geom_hline(yintercept=e1,  color="blue", linetype="dashed") +  # estimate
    geom_hline(yintercept=e2,   color="blue", linetype="dashed") +
    geom_hline(yintercept=e3,    color="blue", linetype="dashed") +
    coord_flip() +  # flip coordinates (puts labels on y axis)
    xlab("Label") + ylab("Mean (95% CI)") +
    theme_bw() + # use a white background
    theme(legend.position="none") +
   ggtitle("Site level predictions")
  print(fp)
```


Site level predictions
Label
Mean (95% CI)
0.7
0.8
0.9

Plot Bayesian country level predictions

```r
#these are predictions
pred <- f8
df <- data.frame(cbind(pred, d))
df$label <- df$country

# start here use coefficients from model rather than predictions
bc <- coef(fit, old=FALSE, summary=TRUE)$country
df <- as.data.frame(bc)
names(df) <- dimnames(bc)[[2]]
df$label <- unique(d$country)


df <- data.frame(df[,c(2,5)], apply(df[,c(1,3,4)],2, A) )
df <- df[,c(2,3,1,4,5)]

# reverses the factor level ordering for labels after coord_flip()
df$label <- factor(df$label, levels=rev(unique(df$label)), ordered = T)

fp <- NULL
fp <- ggplot(data=df, aes(x=label, y=Estimate,
                          ymin=df[,4], ymax=df[,5], colour=label)) +
  geom_pointrange() +
  geom_hline(yintercept=mu, color="green",lty=2) +  # add a dotted line

  geom_hline(yintercept=e1, color="blue", linetype="dashed") +  # estimate
  geom_hline(yintercept=e2,    color="blue", linetype="dashed") +
  geom_hline(yintercept=e3,    color="blue", linetype="dashed") +

  coord_flip() +  # flip coordinates (puts labels on y axis)
  xlab("Country") +
  ylab("Mean (95% CI)") +
  theme_bw() + # use a white background
  theme(legend.position="none") +
  ggtitle("Country level predictions")
print(fp)
```




```
# fitted_values <- fitted(fit)
# head(fitted_values)
# plot fitted means against actual response
# dat <- as.data.frame(cbind(Y = standata(fit)$Y, fitted_values))
# ggplot(dat) + geom_point(aes(x = Estimate, y = Y))
```

Exploring

```
newdata <- data.frame(country = 1, site = 1, person = 1)
predict(fit, newdata = newdata)
```

```
     Estimate Est.Error Q2.5 Q97.5
[1,]  0.82675 0.3785106    0     1
```

```
f7 <- predict(f, re_formula = NULL, summary = F)  # include all random effects
f7[1:10, 1:10]  #columns are samples, rows predictions
```

```
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    1    1    1    1    1    0    1    1     1
 [2,]    1    1    1    1    0    1    1    1    1     1
 [3,]    1    0    0    0    1    0    0    1    1     1
 [4,]    1    1    1    1    1    1    1    1    1     1
 [5,]    1    1    1    0    1    1    1    1    1     0
 [6,]    1    0    0    1    1    1    1    1    0     1
 [7,]    1    1    0    0    0    1    1    0    1     1
 [8,]    1    1    1    1    1    0    1    0    1     1
 [9,]    0    1    1    1    0    1    1    1    1     1
[10,]    1    1    1    1    1    1    1    1    1     1
```

```
x <- pp_check(f, nsamples = 1)
head(x$data)
```

```
# A tibble: 6 x 6
   y_id rep_id rep_label              is_y  is_y_label      value
  <int>  <int> <fct>                  <lgl> <fct>           <dbl>
```

```
1     1      1 italic(y)[rep]  ( 1 ) FALSE italic(y)[rep]     1
2     2      1 italic(y)[rep]  ( 1 ) FALSE italic(y)[rep]     1
3     3      1 italic(y)[rep]  ( 1 ) FALSE italic(y)[rep]     0
4     4      1 italic(y)[rep]  ( 1 ) FALSE italic(y)[rep]     1
5     5      1 italic(y)[rep]  ( 1 ) FALSE italic(y)[rep]     1
6     6      1 italic(y)[rep]  ( 1 ) FALSE italic(y)[rep]     1
```

```r
data1 <- make_standata(formula = y ~ (1 | country/site), family = bernoulli,
    data = d, )

data1$Y
```

```
   [1] 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
  [43] 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
  [85] 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1
 [127] 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1
 [169] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 1 0 1
 [211] 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [253] 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1
 [295] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 [337] 1 0 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1
 [379] 1 1 1 0 1 0 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 0 0
 [421] 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 0 1 1 1
 [463] 0 1 1 0 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 0 0 1 0 0 0 0 1 1 1 1 1 1 0 1
 [505] 1 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 0 1 1 0 0 0 1 1 0 1 1 1 0 1
 [547] 1 0 1 0 1 0 0 0 0 1 0 1 1 1 0 1 0 0 1 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1 1 0
 [589] 1 1 1 0 1 1 1 0 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0 1 1
 [631] 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1
 [673] 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1
 [715] 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1
 [757] 1 1 1 1 1 0 0 1 1 1 1 1 0 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 0 1 0 1 1
 [799] 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 1 1 1
 [841] 1 1 1 1 0 1 1 1 0 1 0 1 1 0 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1
 [883] 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1
 [925] 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 0 1 0 0 1 1 1 1 0 1 1 0 1 1
 [967] 1 0 0 0 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 0 1 1
[1009] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[1051] 1 1 1 1 1 1 1 1 1 1 1 0 0 0 1 1 0 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
[1093] 1 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1 0 1 1 0 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 0 0 1 1 1 1 1
[1135] 0 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1
[1177] 1 1 0 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
[1219] 1 1 0 1 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 0 1 1 1 0 1
[1261] 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1
[1303] 1 1 1 1 0 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0
[1345] 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1
[1387] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 0 1 1 1 0 1 0 1 1 1 1 1 0 1 0 1 0 1 1 1 1
[1429] 1 0 0 0 0 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 0 0 0 1 1 1 1 0 0 1 0 1 1 0 1 0 1 1 1 1 0 0
[1471] 1 1 1 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 0 0 1 0 1
[1513] 1 0 1 1 1 1 0 1 1 1 1 0 1 1 1 1 0 1 1 0 1 0 0 1 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 1 1 1
[1555] 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 0 1 1 0 1
[1597] 1 1 0 0 1 1 1 0 1 1 0 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 0 1 1 0 1 1 0 1
[1639] 0 1 0 1 1 1 1 1 0 1 1 1 0 0 0 0 1 1 1 1 0 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 0 1 1 0
[1681] 1 1 1 1 0 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 0 1 0 1
[1723] 1 1 1 0 1 1 1 1 1 1 0 0 0 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 0 0 1 0 1 1 1 1 1 1 1 1 1 0
[1765] 1 1 1 1 1 0 1 0 1 0 1 1 0 1 1 0 1 1 1 0 0 1 1 1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 1 1 0 1
```

```
[1807] 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 0 0
[1849] 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1
[1891] 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 0 1 1 1 1 1 1 1 1
[1933] 1 1 1 1 0 1 1 1 0 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 0 0 1 1 0 1 1 1
[1975] 1 1 1 1 0 1 1 0 0 1 1 1 1 1 1 0 0 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 0 1 0 1 0 1 1 1
[2017] 1 1 1 1 0 1 1 0 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1
```

```r
##########################################################################

samples1 <- posterior_samples(fit, "^b")
head(samples1)
```

```
  b_Intercept
1    1.407710
2    1.522305
3    1.491709
4    1.477533
5    1.395959
6    1.420230
```

```r
# extract posterior samples of group-level standard
# deviations
samples2 <- posterior_samples(fit, "^sd_")
head(samples2)
```

```
  sd_country__Intercept sd_country:site__Intercept
1             0.2426742                  0.1791958
2             0.4068697                  0.1326295
3             0.3521017                  0.1424982
4             0.3230643                  0.2744405
5             0.3482391                  0.2717645
6             0.4572454                  0.4284394
```

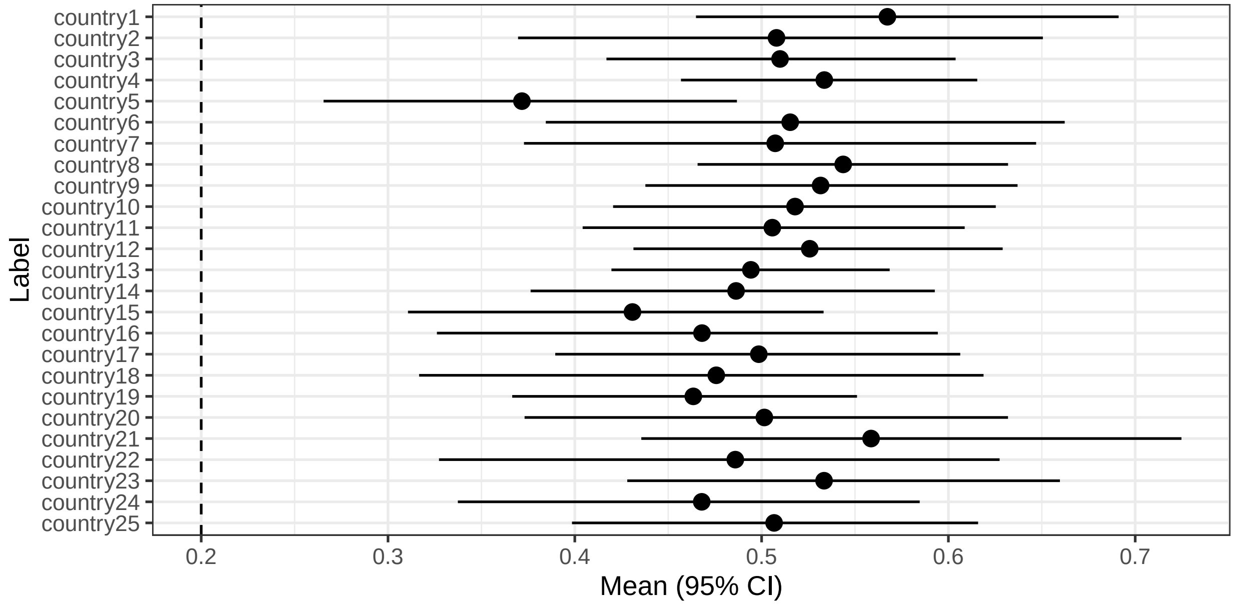
```r
samples3 <- posterior_samples(fit, "^r_country")
head(samples3)[, 1:10]  # show forst 10 columns
```

```
  r_country[1,Intercept] r_country[2,Intercept] r_country[3,Intercept]
1            -0.04795804             -0.1554772            -0.08151334
2            -0.10334515             -0.2969273            -0.11299682
3             0.09323888              0.4227618            -0.22317217
4             0.35973691             -0.2338941             0.23653282
5             0.47979934             -0.1471704            -0.06263322
6             0.03215335             -0.2562508             0.24280972
  r_country[4,Intercept] r_country[5,Intercept] r_country[6,Intercept]
1             0.02361763             -0.6172457            -0.06859248
2             0.13517266             -0.9729814             0.22326579
3             0.12432123             -0.4292339            -0.35883153
4             0.14890283             -0.8228707             0.44421784
5             0.37083394             -0.7380799             0.29143035
6             0.01899304             -0.7179125             0.11536524
  r_country[7,Intercept] r_country[8,Intercept] r_country[9,Intercept]
1            -0.15518422              0.2304349             0.04245634
2            -0.21353095              0.1172642            -0.16999984
3             0.07238551              0.1887413            -0.36992973
4            -0.03655802              0.2530969             0.64643374
5             0.04660996              0.2510266             0.52326010
```

```
6            0.24239495               0.3450241           -0.08077318
  r_country[10,Intercept]
1             0.11782431
2            -0.03751289
3             0.23757448
4            -0.03646897
5             0.01813214
6             0.37727621
```

Plot country SD estimates

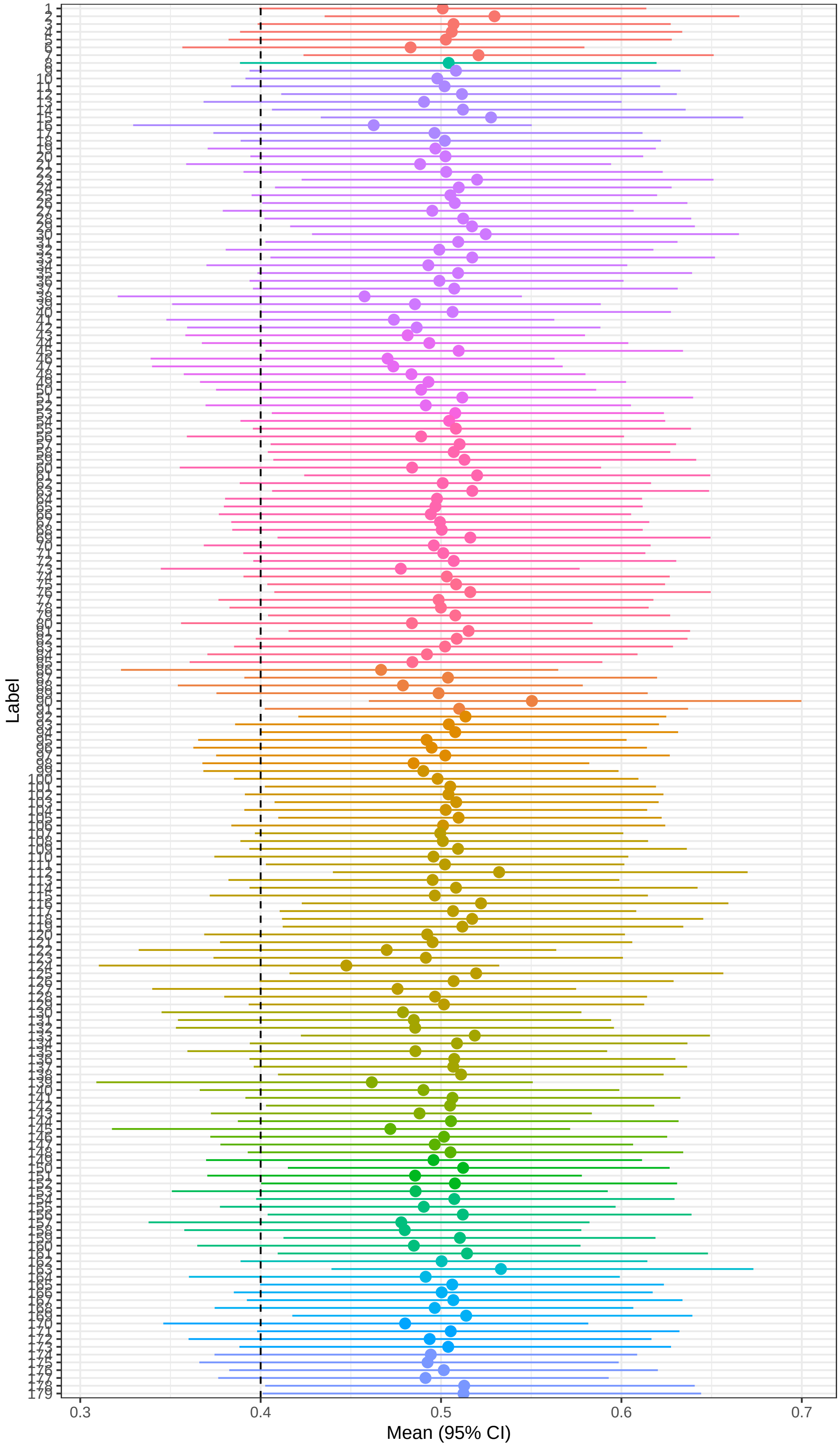
```
 # posterior samples
 mc.1 <- as.mcmc(f, pars = NA, exact_match = TRUE,
                 combine_chains = TRUE, inc_warmup = FALSE)

 # get specific estimates
 # names(mc.1[1,])
 mc_country <- mc.1[, grep("r_country[", names(mc.1[1,]), fixed=TRUE) ]
 # names(mc_country[1,])

 # convert to probabilities
 prob <- apply(mc_country,c(2),function(x) exp(x)/(1+exp(x)))
 # function to calculate summary stats
 statz <- function(x) {
                 t(cbind(c(mean(x), quantile(x, c(0.025,0.975))) ))
        }

 #here are the country specific estimates
 est <- apply(prob,2,statz)

# forest plot see refernce
label <- paste0("country", 1:25)
mean  <-  est[1,]
lower <-  est[2,]
upper <-  est[3,]

df <- data.frame(label, mean, lower, upper)

# reverses the factor level ordering for labels after coord_flip()
df$label <- factor(df$label, levels=rev(df$label))


library(ggplot2)

fp <- NULL
fp <- ggplot(data=df, aes(x=label, y=mean, ymin=lower, ymax=upper)) +
        geom_pointrange() +
        geom_hline(yintercept=sdcountry, lty=2) +  # add a dotted line at x=1 after flip
        coord_flip() +  # flip coordinates (puts labels on y axis)
        xlab("Label") + ylab("Mean (95% CI)") +
        theme_bw()  # use a white background
print(fp)
```



Plot site SD estimates

```
  # posterior samples
  mc.1 <- as.mcmc(f, pars = NA, exact_match = TRUE,
                  combine_chains = TRUE, inc_warmup = FALSE)

  # get specific estimates
  # names(mc.1[1,])
  mc_country <- mc.1[, grep("r_country:", names(mc.1[1,]), fixed=TRUE) ]
  #names(mc_country[1,])

  # convert to probabilities
  prob <- apply(mc_country, c(2) ,function(x) exp(x)/(1+exp(x)))

  # function to calculate summary stats
  statz <- function(x) {
                  t(cbind(c(mean(x), quantile(x, c(0.025,0.975))) ))
          }

  #here are the specific estimates
  est <- apply(prob,2,statz)

 # forest plot see reference
 # label <- paste0("site", 1:dim(mc_country)[2])

 #get labbeling info

 x1 <- gsub("[^0-9\\_]", "",  names(mc_country[1,]))
 x2 <- gsub("\\_", "",  x1)

 x2a <- gsub("\\_", ".",  x1)
 co <- gsub(".*\\.(.*)\\..*", "\\1", x2a)
```

```
site.only <- sub('.*\\.', '', x2a)
label <-  as.numeric(site.only)
mean  <-  est[1,]
lower <-  est[2,]
upper <-  est[3,]


df <- data.frame(co, label, mean, lower, upper)
df <- df[order(label),]

# reverses the factor level ordering for labels after coord_flip()
df$label <- factor(df$label, levels=rev(df$label))

fp <- NULL
fp <- ggplot(data=df, aes(x=label, y=mean, ymin=lower, ymax=upper, colour=co)) +
        geom_pointrange() +
        geom_hline(yintercept=sdsite, lty=2) +  # add a dotted line at x=1 after flip
        coord_flip() +  # flip coordinates (puts labels on y axis)
        xlab("Label") + ylab("Mean (95% CI)") +
        theme_bw() + # use a white background +
  theme(legend.position="none")

print(fp)
```


Label
Mean (95% CI)
0.3
0.4
0.5
0.6
0.7

Comparing frequentist and Bayesian, what the difference between fitted and coef from model output?

```
# fixef(fit, old=FALSE) object$fit@sim country predictions
# https://github.com/paul-buerkner/brms/issues/82

#########################################
f8 <- fitted(f, re_formula = ~(1 | country))  # is this the correct way for country level?
ba <- unique(f8)[, 1]  # Bayesian predictions
ba[1:10]


fr <- as.vector(unlist(1/(1 + exp(-coef(fit0)$country))))  # freq blups
plot(fr, ba)
abline(0, 1)  # compare Bayesian and Frequentist

######################################### above differs to model coefficients?
bc <- as.vector(coef(fit, old = TRUE)$country)
bc <- 1/(1 + exp(-bc))
bc

plot(fr, bc)
abline(0, 1)  # compare Bayesian and Frequentist

#########################################
f <- as.vector(unlist(coef(fit0)$country))  #frequentist
b <- as.vector(coef(fit)$country[, 1, ])  #bayes
plot(f, b)
abline(0, 1)
cor(f, b)


f <- as.vector(unlist(coef(fit0)$site))  #frequentist
# b <- as.vector(coef(fit)$`country:site`[,1,]) bayes , needs
# ordering
bc <- coef(fit, old = FALSE, summary = TRUE)$`country:site`
df <- as.data.frame(bc)
names(df) <- dimnames(bc)[[2]]
df$sites <- gsub(".*_", "", rownames(df))
# df$countries <- gsub('_.*', '', rownames(df))
df$sites <- as.numeric(as.character(df$sites))
df <- df[order(df$sites), ]
b <- df[, 1]
plot(f, b)
abline(0, 1)
cor(f, b)
```

Crude estimates

```
foo <- d

library(dplyr)
foox <- foo %>% group_by(country, site) %>% summarise(N = length(y),
```

```
    ones = (mean(y)) * length(y), zero = (1 - mean(y)) * length(y),
    mean = mean(y), )

foox <- data.frame(foox)
names(foox) <- c("Country", "site", "N", "Yes", "No", "Mean")

foox <- cbind(foox, binconf(foox$Yes, foox$N))
foox
```

```
     Country site  N Yes No      Mean  PointEst      Lower     Upper
X          1    1 24  20  4 0.8333333 0.8333333 0.64146929 0.9332132
X.1        1    2 23  22  1 0.9565217 0.9565217 0.79008845 0.9977699
X.2        1    3 10   9  1 0.9000000 0.9000000 0.59584997 0.9948707
X.3        1    4  2   2  0 1.0000000 1.0000000 0.34238023 1.0000000
X.4        1    5  1   1  0 1.0000000 1.0000000 0.05129329 1.0000000
X.5        1    6  4   2  2 0.5000000 0.5000000 0.15003899 0.8499610
X.6        1    7 17  16  1 0.9411765 0.9411765 0.73017969 0.9969827
X.7        2    8 12  10  2 0.8333333 0.8333333 0.55196914 0.9530349
X.8        3    9  3   3  0 1.0000000 1.0000000 0.43850297 1.0000000
X.9        3   10 40  32  8 0.8000000 0.8000000 0.65242694 0.8950001
X.10       3   11  1   1  0 1.0000000 1.0000000 0.05129329 1.0000000
X.11       3   12 15  13  2 0.8666667 0.8666667 0.62118017 0.9626387
X.12       3   13 11   8  3 0.7272727 0.7272727 0.43435470 0.9025394
X.13       3   14  5   5  0 1.0000000 1.0000000 0.56551754 1.0000000
X.14       3   15 13  13  0 1.0000000 1.0000000 0.77190463 1.0000000
X.15       3   16 26  17  9 0.6538462 0.6538462 0.46220571 0.8058777
X.16       3   17  4   3  1 0.7500000 0.7500000 0.30064184 0.9871767
X.17       3   18  6   5  1 0.8333333 0.8333333 0.43649718 0.9914511
X.18       4   19  4   3  1 0.7500000 0.7500000 0.30064184 0.9871767
X.19       4   20 12  10  2 0.8333333 0.8333333 0.55196914 0.9530349
X.20       4   21  6   4  2 0.6666667 0.6666667 0.29999332 0.9032286
X.21       4   22  1   1  0 1.0000000 1.0000000 0.05129329 1.0000000
X.22       4   23 10  10  0 1.0000000 1.0000000 0.72246720 1.0000000
X.23       4   24 10   9  1 0.9000000 0.9000000 0.59584997 0.9948707
X.24       4   25 14  12  2 0.8571429 0.8571429 0.60058621 0.9599061
X.25       4   26  9   8  1 0.8888889 0.8888889 0.56500029 0.9943007
X.26       4   27  9   7  2 0.7777778 0.7777778 0.45258897 0.9367749
X.27       4   28  6   6  0 1.0000000 1.0000000 0.60966571 1.0000000
X.28       4   29 21  19  2 0.9047619 0.9047619 0.71085861 0.9734812
X.29       4   30 19  18  1 0.9473684 0.9473684 0.75361269 0.9973004
X.30       4   31 10   9  1 0.9000000 0.9000000 0.59584997 0.9948707
X.31       4   32 10   8  2 0.8000000 0.8000000 0.49016247 0.9433178
X.32       4   33  8   8  0 1.0000000 1.0000000 0.67559244 1.0000000
X.33       4   34  7   5  2 0.7142857 0.7142857 0.35893445 0.9177811
X.34       4   35  4   4  0 1.0000000 1.0000000 0.51010916 1.0000000
X.35       4   36 27  22  5 0.8148148 0.8148148 0.63301316 0.9181929
X.36       4   37  9   8  1 0.8888889 0.8888889 0.56500029 0.9943007
X.37       4   38 21  13  8 0.6190476 0.6190476 0.40878654 0.7924899
X.38       4   39 10   7  3 0.7000000 0.7000000 0.39677815 0.8922087
X.39       4   40 14  12  2 0.8571429 0.8571429 0.60058621 0.9599061
X.40       4   41 24  17  7 0.7083333 0.7083333 0.50832306 0.8508535
X.41       4   42 10   7  3 0.7000000 0.7000000 0.39677815 0.8922087
X.42       5   43 21  13  8 0.6190476 0.6190476 0.40878654 0.7924899
X.43       5   44  5   3  2 0.6000000 0.6000000 0.23072428 0.8823792
```

X.44	5	45	10	8	2	0.8000000	0.8000000	0.49016247	0.9433178
X.45	5	46	14	7	7	0.5000000	0.5000000	0.26799202	0.7320080
X.46	5	47	12	6	6	0.5000000	0.5000000	0.25378160	0.7462184
X.47	5	48	9	5	4	0.5555556	0.5555556	0.26665129	0.8112215
X.48	5	49	8	5	3	0.6250000	0.6250000	0.30574239	0.8631557
X.49	5	50	17	11	6	0.6470588	0.6470588	0.41300363	0.8269028
X.50	5	51	7	6	1	0.8571429	0.8571429	0.48687217	0.9926724
X.51	5	52	11	7	4	0.6363636	0.6363636	0.35380117	0.8483353
X.52	6	53	14	12	2	0.8571429	0.8571429	0.60058621	0.9599061
X.53	7	54	12	10	2	0.8333333	0.8333333	0.55196914	0.9530349
X.54	8	55	4	4	0	1.0000000	1.0000000	0.51010916	1.0000000
X.55	8	56	11	8	3	0.7272727	0.7272727	0.43435470	0.9025394
X.56	8	57	17	15	2	0.8823529	0.8823529	0.65663649	0.9671202
X.57	8	58	15	13	2	0.8666667	0.8666667	0.62118017	0.9626387
X.58	8	59	12	11	1	0.9166667	0.9166667	0.64612009	0.9957256
X.59	8	60	10	7	3	0.7000000	0.7000000	0.39677815	0.8922087
X.60	8	61	29	26	3	0.8965517	0.8965517	0.73614919	0.9641851
X.61	8	62	17	14	3	0.8235294	0.8235294	0.58970541	0.9380887
X.62	8	63	8	8	0	1.0000000	1.0000000	0.67559244	1.0000000
X.63	8	64	10	8	2	0.8000000	0.8000000	0.49016247	0.9433178
X.64	8	65	4	3	1	0.7500000	0.7500000	0.30064184	0.9871767
X.65	8	66	14	11	3	0.7857143	0.7857143	0.52410769	0.9242861
X.66	8	67	11	9	2	0.8181818	0.8181818	0.52301944	0.9486323
X.67	8	68	18	15	3	0.8333333	0.8333333	0.60777962	0.9416342
X.68	8	69	8	8	0	1.0000000	1.0000000	0.67559244	1.0000000
X.69	8	70	4	3	1	0.7500000	0.7500000	0.30064184	0.9871767
X.70	8	71	17	14	3	0.8235294	0.8235294	0.58970541	0.9380887
X.71	8	72	3	3	0	1.0000000	1.0000000	0.43850297	1.0000000
X.72	8	73	11	7	4	0.6363636	0.6363636	0.35380117	0.8483353
X.73	9	74	7	6	1	0.8571429	0.8571429	0.48687217	0.9926724
X.74	9	75	15	13	2	0.8666667	0.8666667	0.62118017	0.9626387
X.75	9	76	7	7	0	1.0000000	1.0000000	0.64566956	1.0000000
X.76	9	77	5	4	1	0.8000000	0.8000000	0.37553463	0.9897413
X.77	9	78	17	14	3	0.8235294	0.8235294	0.58970541	0.9380887
X.78	9	79	9	8	1	0.8888889	0.8888889	0.56500029	0.9943007
X.79	9	80	14	10	4	0.7142857	0.7142857	0.45350916	0.8827862
X.80	9	81	7	7	0	1.0000000	1.0000000	0.64566956	1.0000000
X.81	9	82	4	4	0	1.0000000	1.0000000	0.51010916	1.0000000
X.82	9	83	7	6	1	0.8571429	0.8571429	0.48687217	0.9926724
X.83	9	84	2	1	1	0.5000000	0.5000000	0.02564665	0.9743534
X.84	9	85	14	10	4	0.7142857	0.7142857	0.45350916	0.8827862
X.85	10	86	15	9	6	0.6000000	0.6000000	0.35746830	0.8017550
X.86	10	87	12	10	2	0.8333333	0.8333333	0.55196914	0.9530349
X.87	10	88	11	7	4	0.6363636	0.6363636	0.35380117	0.8483353
X.88	10	89	5	4	1	0.8000000	0.8000000	0.37553463	0.9897413
X.89	10	90	34	33	1	0.9705882	0.9705882	0.85084427	0.9984914
X.90	10	91	4	4	0	1.0000000	1.0000000	0.51010916	1.0000000
X.91	11	92	32	27	5	0.8437500	0.8437500	0.68245850	0.9313558
X.92	11	93	7	6	1	0.8571429	0.8571429	0.48687217	0.9926724
X.93	11	94	9	8	1	0.8888889	0.8888889	0.56500029	0.9943007
X.94	11	95	12	9	3	0.7500000	0.7500000	0.46769467	0.9110583
X.95	11	96	3	2	1	0.6666667	0.6666667	0.20765960	0.9829022
X.96	11	97	1	1	0	1.0000000	1.0000000	0.05129329	1.0000000
X.97	11	98	18	13	5	0.7222222	0.7222222	0.49127343	0.8750025

X.98	12	99	11	8	3	0.7272727	0.7272727	0.43435470	0.9025394
X.99	12	100	10	8	2	0.8000000	0.8000000	0.49016247	0.9433178
X.100	12	101	18	15	3	0.8333333	0.8333333	0.60777962	0.9416342
X.101	12	102	7	6	1	0.8571429	0.8571429	0.48687217	0.9926724
X.102	12	103	20	17	3	0.8500000	0.8500000	0.63958114	0.9476313
X.103	12	104	12	10	2	0.8333333	0.8333333	0.55196914	0.9530349
X.104	12	105	21	18	3	0.8571429	0.8571429	0.65363940	0.9501899
X.105	12	106	6	5	1	0.8333333	0.8333333	0.43649718	0.9914511
X.106	13	107	34	27	7	0.7941176	0.7941176	0.63201612	0.8965047
X.107	13	108	10	8	2	0.8000000	0.8000000	0.49016247	0.9433178
X.108	13	109	3	3	0	1.0000000	1.0000000	0.43850297	1.0000000
X.109	13	110	3	2	1	0.6666667	0.6666667	0.20765960	0.9829022
X.110	13	111	31	25	6	0.8064516	0.8064516	0.63719742	0.9081299
X.111	13	112	15	15	0	1.0000000	1.0000000	0.79611670	1.0000000
X.112	13	113	8	6	2	0.7500000	0.7500000	0.40927543	0.9285208
X.113	13	114	4	4	0	1.0000000	1.0000000	0.51010916	1.0000000
X.114	13	115	8	6	2	0.7500000	0.7500000	0.40927543	0.9285208
X.115	13	116	10	10	0	1.0000000	1.0000000	0.72246720	1.0000000
X.116	13	117	28	23	5	0.8214286	0.8214286	0.64408575	0.9212150
X.117	13	118	7	7	0	1.0000000	1.0000000	0.64566956	1.0000000
X.118	13	119	10	9	1	0.9000000	0.9000000	0.59584997	0.9948707
X.119	13	120	7	5	2	0.7142857	0.7142857	0.35893445	0.9177811
X.120	13	121	8	6	2	0.7500000	0.7500000	0.40927543	0.9285208
X.121	13	122	15	9	6	0.6000000	0.6000000	0.35746830	0.8017550
X.122	13	123	11	8	3	0.7272727	0.7272727	0.43435470	0.9025394
X.123	13	124	42	27	15	0.6428571	0.6428571	0.49166366	0.7701081
X.124	13	125	8	8	0	1.0000000	1.0000000	0.67559244	1.0000000
X.125	13	126	13	11	2	0.8461538	0.8461538	0.57765369	0.9567418
X.126	13	127	9	5	4	0.5555556	0.5555556	0.26665129	0.8112215
X.127	13	128	4	3	1	0.7500000	0.7500000	0.30064184	0.9871767
X.128	13	129	11	9	2	0.8181818	0.8181818	0.52301944	0.9486323
X.129	14	130	6	3	3	0.5000000	0.5000000	0.18761631	0.8123837
X.130	14	131	4	2	2	0.5000000	0.5000000	0.15003899	0.8499610
X.131	14	132	4	2	2	0.5000000	0.5000000	0.15003899	0.8499610
X.132	14	133	13	12	1	0.9230769	0.9230769	0.66686049	0.9960544
X.133	14	134	3	3	0	1.0000000	1.0000000	0.43850297	1.0000000
X.134	14	135	8	5	3	0.6250000	0.6250000	0.30574239	0.8631557
X.135	14	136	3	3	0	1.0000000	1.0000000	0.43850297	1.0000000
X.136	14	137	3	3	0	1.0000000	1.0000000	0.43850297	1.0000000
X.137	14	138	14	12	2	0.8571429	0.8571429	0.60058621	0.9599061
X.138	15	139	14	7	7	0.5000000	0.5000000	0.26799202	0.7320080
X.139	15	140	9	6	3	0.6666667	0.6666667	0.35420214	0.8794162
X.140	15	141	2	2	0	1.0000000	1.0000000	0.34238023	1.0000000
X.141	15	142	18	14	4	0.7777778	0.7777778	0.54785416	0.9099907
X.142	15	143	16	11	5	0.6875000	0.6875000	0.44404356	0.8583536
X.143	16	144	2	2	0	1.0000000	1.0000000	0.34238023	1.0000000
X.144	16	145	7	3	4	0.4285714	0.4285714	0.15821986	0.7495416
X.145	16	146	1	1	0	1.0000000	1.0000000	0.05129329	1.0000000
X.146	16	147	3	2	1	0.6666667	0.6666667	0.20765960	0.9829022
X.147	16	148	2	2	0	1.0000000	1.0000000	0.34238023	1.0000000
X.148	17	149	8	6	2	0.7500000	0.7500000	0.40927543	0.9285208
X.149	17	150	26	22	4	0.8461538	0.8461538	0.66468801	0.9384997
X.150	17	151	26	19	7	0.7307692	0.7307692	0.53916960	0.8629555
X.151	17	152	8	7	1	0.8750000	0.8750000	0.52911182	0.9935883

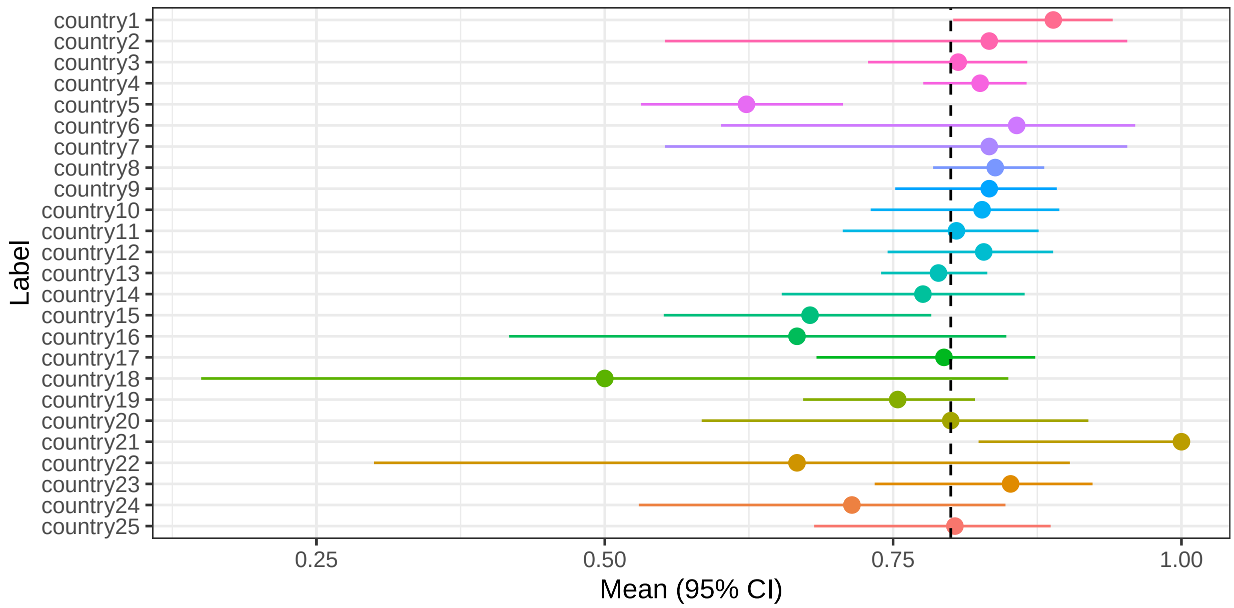
```
X.152      18  153  4   2  2 0.5000000 0.5000000 0.15003899 0.8499610
X.153      19  154  7   6  1 0.8571429 0.8571429 0.48687217 0.9926724
X.154      19  155 18  13  5 0.7222222 0.7222222 0.49127343 0.8750025
X.155      19  156  9   8  1 0.8888889 0.8888889 0.56500029 0.9943007
X.156      19  157  9   5  4 0.5555556 0.5555556 0.26665129 0.8112215
X.157      19  158 22  15  7 0.6818182 0.6818182 0.47318598 0.8363941
X.158      19  159 32  26  6 0.8125000 0.8125000 0.64690845 0.9111046
X.159      19  160 24  17  7 0.7083333 0.7083333 0.50832306 0.8508535
X.160      19  161  5   5  0 1.0000000 1.0000000 0.56551754 1.0000000
X.161      20  162 20  16  4 0.8000000 0.8000000 0.58398257 0.9193423
X.162      21  163 18  18  0 1.0000000 1.0000000 0.82412078 1.0000000
X.163      22  164  6   4  2 0.6666667 0.6666667 0.29999332 0.9032286
X.164      23  165 14  12  2 0.8571429 0.8571429 0.60058621 0.9599061
X.165      23  166 11   9  2 0.8181818 0.8181818 0.52301944 0.9486323
X.166      23  167  3   3  0 1.0000000 1.0000000 0.43850297 1.0000000
X.167      23  168 14  11  3 0.7857143 0.7857143 0.52410769 0.9242861
X.168      23  169 12  11  1 0.9166667 0.9166667 0.64612009 0.9957256
X.169      24  170 10   6  4 0.6000000 0.6000000 0.31267377 0.8318197
X.170      24  171 11   9  2 0.8181818 0.8181818 0.52301944 0.9486323
X.171      24  172  6   4  2 0.6666667 0.6666667 0.29999332 0.9032286
X.172      24  173  1   1  0 1.0000000 1.0000000 0.05129329 1.0000000
X.173      25  174  7   5  2 0.7142857 0.7142857 0.35893445 0.9177811
X.174      25  175  7   5  2 0.7142857 0.7142857 0.35893445 0.9177811
X.175      25  176 11   9  2 0.8181818 0.8181818 0.52301944 0.9486323
X.176      25  177 21  16  5 0.7619048 0.7619048 0.54908826 0.8937199
X.177      25  178  5   5  0 1.0000000 1.0000000 0.56551754 1.0000000
X.178      25  179  5   5  0 1.0000000 1.0000000 0.56551754 1.0000000
```

```
foox1 <- foo %>% group_by(country) %>% summarise(N = length(y),
    ones = (mean(y)) * length(y), zero = (1 - mean(y)) * length(y),
    mean = mean(y), )

foox1 <- data.frame(foox1)
names(foox1) <- c("Country", "N", "Yes", "No", "Mean")
```

Plot crude estimates country level

```
  foox1 <- cbind(foox1,binconf(foox1$Yes, foox1$N))
  est <- foox1

  label <- paste0("country", 1:25)
  mean  <-  est[,6]
  lower <-  est[,7]
  upper <-  est[,8]

  df <- data.frame(label, mean, lower, upper)

  # reverses the factor level ordering for labels after coord_flip()
  df$label <- factor(df$label, levels=rev(df$label))


  library(ggplot2)

  fp <- NULL
```

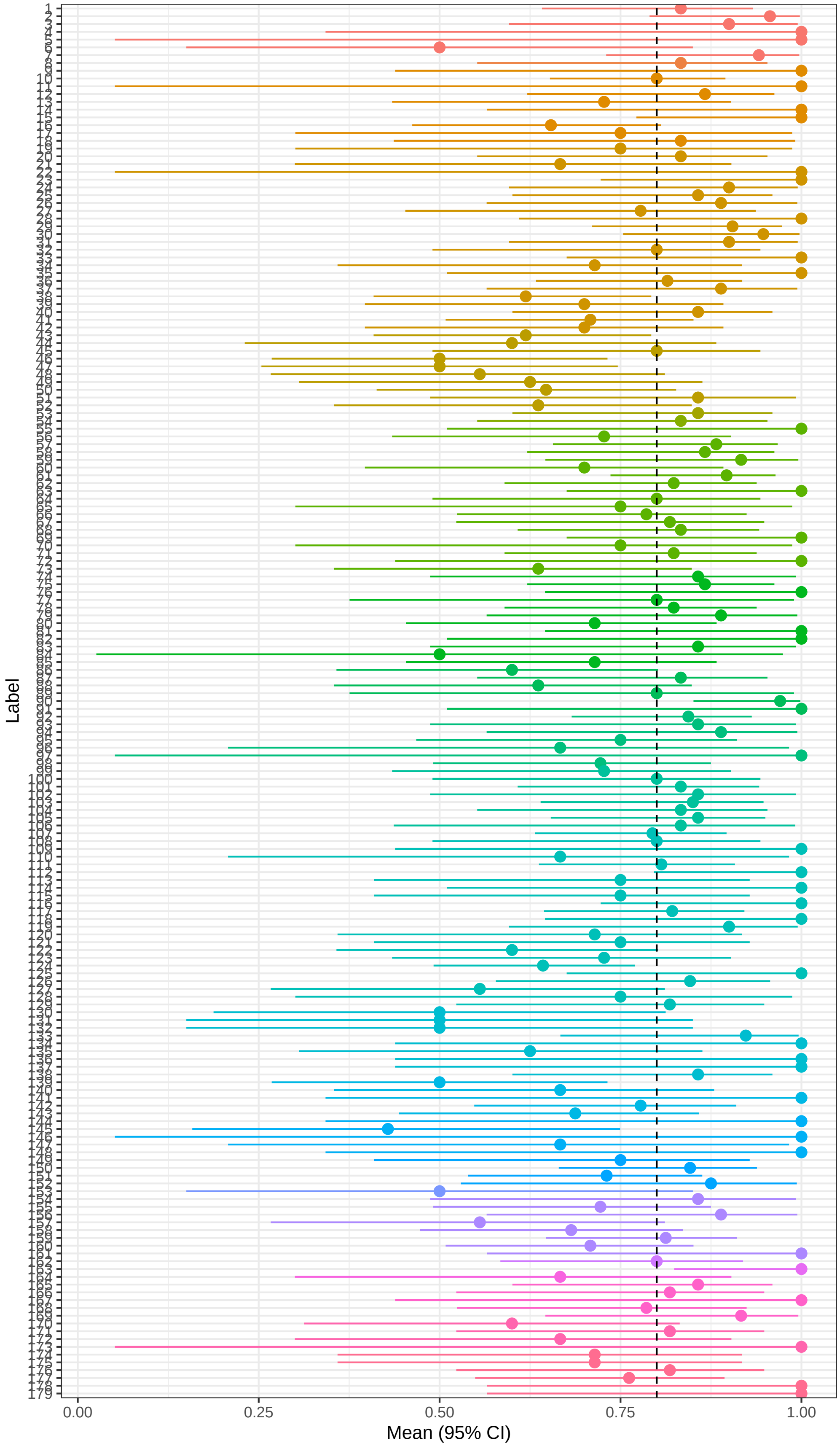
```
fp <- ggplot(data=df, aes(x=label, y=mean, ymin=lower, ymax=upper, colour=label)) +
        geom_pointrange() +
        geom_hline(yintercept=mu, lty=2) +  # add a dotted line at x=1 after flip
        coord_flip() +  # flip coordinates (puts labels on y axis)
        xlab("Label") + ylab("Mean (95% CI)") +
        theme_bw() +   # use a white background
        theme(legend.position="none")
print(fp)
```



Plot crude site estimates

```
est <- foox
label <-  est$site
mean  <-  est[,6]
lower <-  est[,8]
upper <-  est[,9]


df <- data.frame(co=foox$Country, label, mean, lower, upper)
#df <- df[order(label),]

# reverses the factor level ordering for labels after coord_flip()
df$label <- factor(df$label, levels=rev(df$label))

fp <- NULL
fp <- ggplot(data=df, aes(x=label, y=mean, ymin=lower, ymax=upper, colour=co)) +
        geom_pointrange() +
        geom_hline(yintercept=mu, lty=2) +  # add a dotted line at x=1 after flip
        coord_flip() +  # flip coordinates (puts labels on y axis)
        xlab("Label") + ylab("Mean (95% CI)") +
        theme_bw() + # use a white background +
  theme(legend.position="none")
```

```
print(fp)
```


Label
Mean (95% CI)
0.00
0.25
0.50
0.75
1.00

```
# cat("\n")
# #cat("Summary Statistics")
# cat("\n")
# print(kable(foox, format="pandoc", digits=c(0,0,0,4),
#             caption = "crude estimates"))
# cat("\n")
```

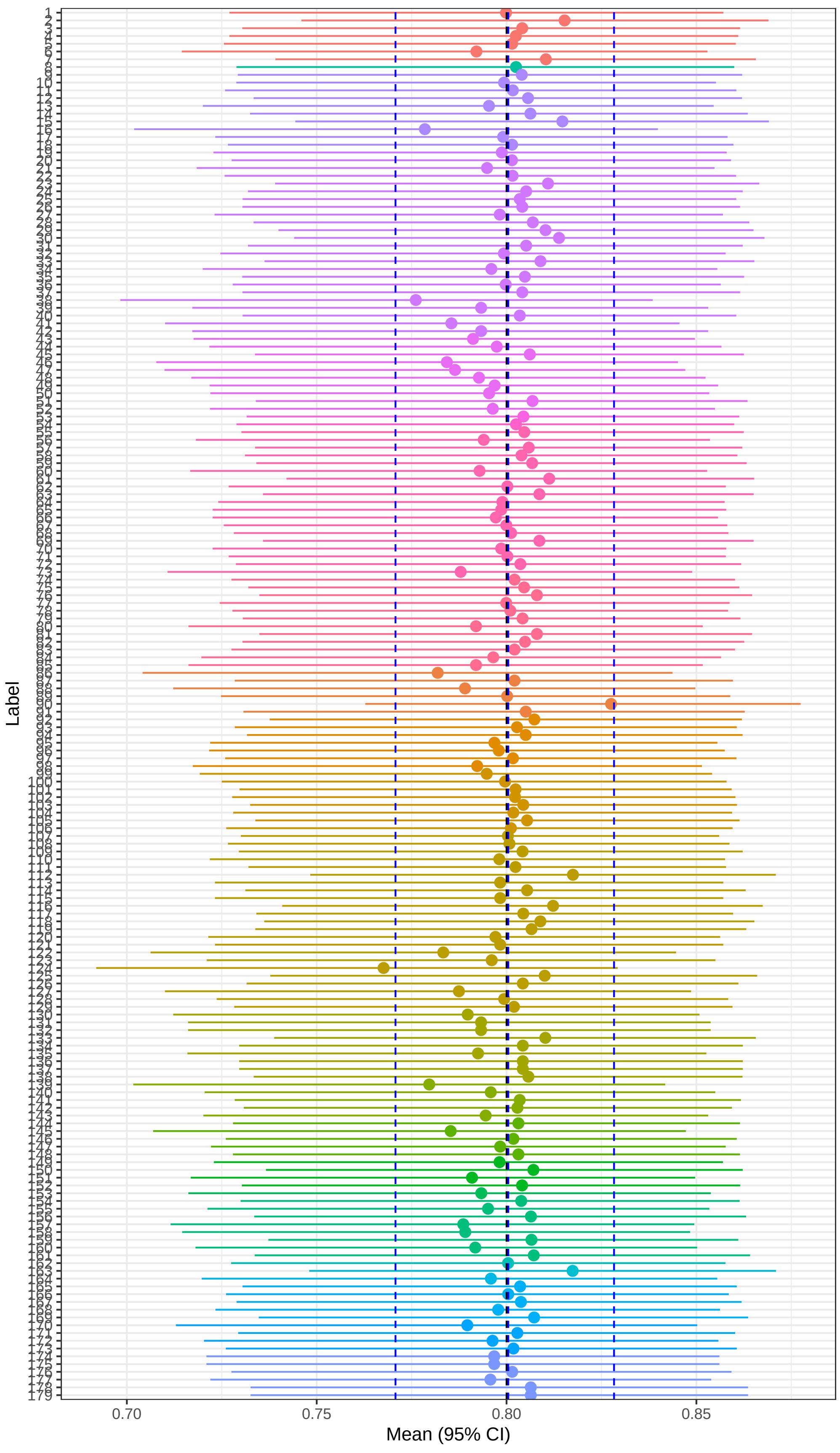
lmer estimates sites

```
  # manage the data
  # fco <- coef(fit0)$`site:country`
  # c(attr(ranef(fit0,condVar=TRUE)[[1]],"postVar"))  # co
  str(rr1 <- ranef(fit0, condVar = TRUE))
```

```
List of 2
 $ site:country:'data.frame':   179 obs. of  1 variable:
  ..$ (Intercept): num [1:179] -0.00365 0.09555 0.02314 0.01256 0.00631 ...
  ..- attr(*, "postVar")= num [1, 1, 1:179] 0.0352 0.0355 0.0373 0.0387 0.0389 ...
 $ country     :'data.frame':   25 obs. of  1 variable:
  ..$ (Intercept): num [1:25] 0.2522 0.0222 0.0321 0.1236 -0.5382 ...
  ..- attr(*, "postVar")= num [1, 1, 1:25] 0.0408 0.0606 0.0313 0.0191 0.0273 ...
 - attr(*, "class")= chr "ranef.mer"
```

```
  # dotplot(rr1)                    ## default  #
  # cV <- ranef(fit0, condVar = TRUE)
  # # ranvar <- attr(cV[[1]], "postVar")
  # # sqrt(diag(ranvar[,,]))

  # get random effects from lmer and calculate confidence intervals too
  # http://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#lme4
  # intercept
  fix <- fixef(fit0)              # intercept on log scale
  fix.var <- sqrt(diag(vcov(fit0)))   # intercept associated standard error^2
  # blups
  s.c <- rr1[1][1]                # site country random effects
  s.c.var <- c(attr(ranef(fit0,condVar=TRUE)[[1]],"postVar"))^0.5  # site country sds
  # make a datset
  vars <- as.data.frame(cbind(intercept=fix, intercept.var=fix.var ,
                              blup=as.vector(unlist(s.c)), blup.var=s.c.var ))
  # calculate CI for the random effects
  vars$est <- vars$intercept+vars$blup  # shift from intercept
  vars$lower <- vars$est+c(-1)* qnorm(0.975)*sqrt(vars$blup.var^2+vars$intercept.var^2)  #lower CI
  vars$upper <- vars$est+c( 1)* qnorm(0.975)*sqrt(vars$blup.var^2+vars$intercept.var^2)  #upper CI

  #log odds to probabilities
  A <- function(x) 1/(1+exp(-x))
  df <- vars
  df <- data.frame(df[,c(1:4)], apply(df[,c(5:7)],2, A) )


  label = rownames(s.c$`site:country`)
  df$sites <-     gsub(":.*","", label)
  df$countries <- gsub(".*:","", label)
```

```
#plot
est <- df
label <-  est$sites
mean  <-  est$est
lower <-  est$lower
upper <-  est$upper

# intercept CI
fit0fci <- confint(fit0)
L <- fit0fci["(Intercept)",][1][[1]]
U <- fit0fci["(Intercept)",][2][[1]]

df <- data.frame(co=df$countries, label, mean, lower, upper)
#df <- df[order(label),]

# reverses the factor level ordering for labels after coord_flip()
df$label <- factor(df$label, levels=rev(df$label))

fp <- NULL
fp <- ggplot(data=df, aes(x=label, y=mean, ymin=lower, ymax=upper, colour=co)) +
        geom_pointrange() +
        geom_hline(yintercept=mu, lty=2) +  # add a dotted line at x=1 after flip
        geom_hline(yintercept= A(fix),  color="blue", linetype="dashed") +  # estimate
        geom_hline(yintercept=A(L),  color="blue", linetype="dashed") +
        geom_hline(yintercept=A(U),  color="blue", linetype="dashed") +
        coord_flip() +  # flip coordinates (puts labels on y axis)
        xlab("Label") + ylab("Mean (95% CI)") +
        theme_bw() + # use a white background +
  theme(legend.position="none")

print(fp)
```


Label
Mean (95% CI)
0.70
0.75
0.80
0.85

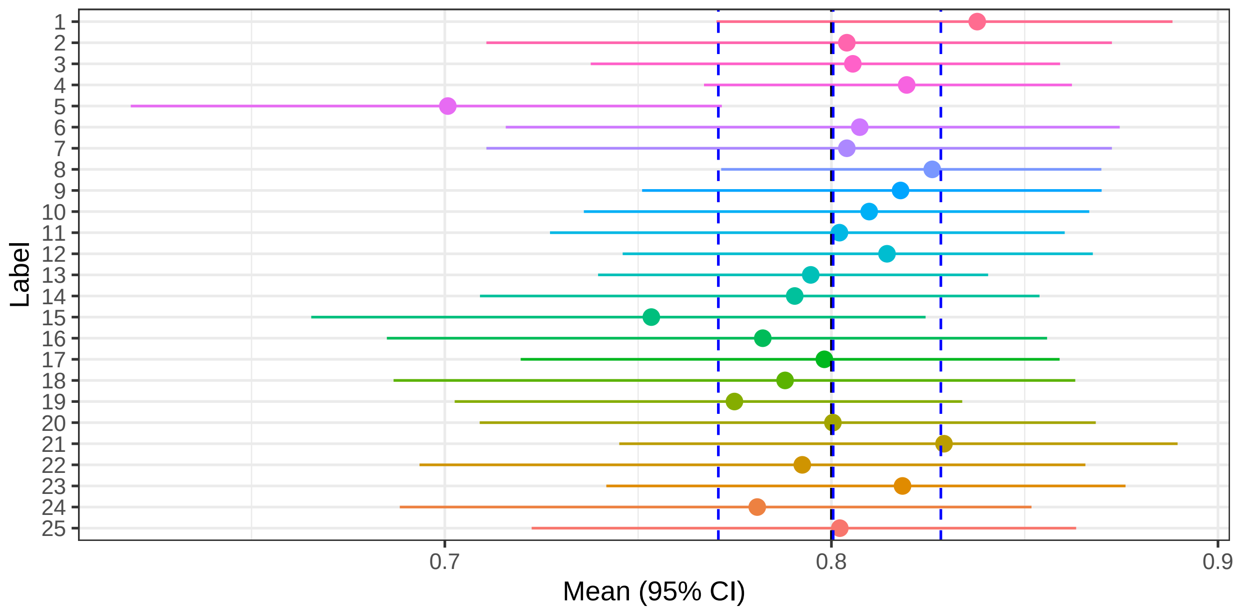
lmer estimates countries

```r
str(rr1 <- ranef(fit0, condVar = TRUE))
```

```
List of 2
 $ site:country:'data.frame':   179 obs. of  1 variable:
  ..$ (Intercept): num [1:179] -0.00365 0.09555 0.02314 0.01256 0.00631 ...
  ..- attr(*, "postVar")= num [1, 1, 1:179] 0.0352 0.0355 0.0373 0.0387 0.0389 ...
 $ country      :'data.frame':   25 obs. of  1 variable:
  ..$ (Intercept): num [1:25] 0.2522 0.0222 0.0321 0.1236 -0.5382 ...
  ..- attr(*, "postVar")= num [1, 1, 1:25] 0.0408 0.0606 0.0313 0.0191 0.0273 ...
 - attr(*, "class")= chr "ranef.mer"
```

```r
  # get random effects from lmer and calculate confidence intervals too
  # http://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#lme4
  # blups
  s.c <- rr1[2][1]                # country random effects
  s.c.var <- c(attr(ranef(fit0,condVar=TRUE)[[2]],"postVar"))^0.5  # country sds
  # make a datset
  vars <- as.data.frame(cbind(intercept=fix, intercept.var=fix.var ,
                              blup=as.vector(unlist(s.c)), blup.var=s.c.var ))
  # calculate CI for the random effects
  vars$est <- vars$intercept+vars$blup  # shift from intercept
  vars$lower <- vars$est+c(-1)* qnorm(0.975)*sqrt(vars$blup.var^2+vars$intercept.var^2)  #lower CI
  vars$upper <- vars$est+c( 1)* qnorm(0.975)*sqrt(vars$blup.var^2+vars$intercept.var^2)  #upper CI

  #log odds to probabilities
  A <- function(x) 1/(1+exp(-x))
  df <- vars
  df <- data.frame(df[,c(1:4)], apply(df[,c(5:7)],2, A) )


  label = rownames(s.c$`country`)
  df$countries <- gsub(".*:","", label)

  #plot
  est <- df
  label <-  est$countries
  mean  <-  est$est
  lower <-  est$lower
  upper <-  est$upper

  # intercept CI
# fit0fci <- confint(fit0)
#  L <- fit0fci["(Intercept)",][1][[1]]
#  U <- fit0fci["(Intercept)",][2][[1]]

  df <- data.frame( label, mean, lower, upper)
  #df <- df[order(label),]

  # reverses the factor level ordering for labels after coord_flip()
  df$label <- factor(df$label, levels=rev(df$label))

  fp <- NULL
  fp <- ggplot(data=df, aes(x=label, y=mean, ymin=lower, ymax=upper, colour=label)) +
```

```
        geom_pointrange() +
        geom_hline(yintercept=mu, lty=2) +  # add a dotted line at x=1 after flip
        geom_hline(yintercept= A(fix),   color="blue", linetype="dashed") +  # estimate
        geom_hline(yintercept=A(L),   color="blue", linetype="dashed") +
        geom_hline(yintercept=A(U),  color="blue", linetype="dashed") +
        coord_flip() +  # flip coordinates (puts labels on y axis)
        xlab("Label") + ylab("Mean (95% CI)") +
        theme_bw() + # use a white background +
  theme(legend.position="none")

print(fp)
```



REFERENCES

1 paper http://bmcmedresmethodol.biomedcentral.com/track/pdf/10.1186/1471-2288-11-94?site=bmcmedresmethodol.biomedcentral.com 1 code http://www.biomedcentral.com/content/supplementary/1471-2288-11-94-S1.PDF 1 code https://bmcmedresmethodol.biomedcentral.com/articles/10.1186/1471-2288-11-94#MOESM1 2 forrest plot https://stackoverflow.com/questions/38062650/forest-plot-for-a-beginner-simple-exa 3 https://stackoverflow.com/questions/14639892/how-to-extract-words-between-two-period-using-rs-gsub 4 https://stackoverflow.com/questions/31774086/extracting-text-after-last-period-in-string-in-r 5 https://stats.stackexchange.com/questions/147836/prediction-interval-for-lmer-mixed-effects-model-in-r 6 http://bbolker.github.io/mixedmodels-misc/glmmFAQ.html#lme4

COMPUTING ENVIRONMENT

```
R version 3.6.1 (2019-07-05)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 17134)

Matrix products: default

locale:
[1] LC_COLLATE=English_United Kingdom.1252
[2] LC_CTYPE=English_United Kingdom.1252
[3] LC_MONETARY=English_United Kingdom.1252
[4] LC_NUMERIC=C
[5] LC_TIME=English_United Kingdom.1252

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods
[7] base

other attached packages:
 [1] dplyr_0.8.3           rstan_2.19.2          StanHeaders_2.18.1-10
 [4] brms_2.9.0            Rcpp_1.0.1            rms_5.1-3.1
 [7] SparseM_1.77          Hmisc_4.2-0           ggplot2_3.2.0
[10] Formula_1.2-3         survival_2.44-1.1     lattice_0.20-38
[13] lme4_1.1-21           Matrix_1.2-17         knitr_1.23

loaded via a namespace (and not attached):
  [1] TH.data_1.0-10       minqa_1.2.4          colorspace_1.4-1
  [4] ggridges_0.5.1       rsconnect_0.8.13     htmlTable_1.13.1
  [7] markdown_1.0         base64enc_0.1-3      rstudioapi_0.10
 [10] MatrixModels_0.4-1   DT_0.7               fansi_0.4.0
 [13] mvtnorm_1.0-11       bridgesampling_0.6-0 codetools_0.2-16
 [16] splines_3.6.1        shinythemes_1.1.2    zeallot_0.1.0
 [19] bayesplot_1.7.0      nloptr_1.2.1         binom_1.1-1
 [22] cluster_2.1.0        shiny_1.3.2          compiler_3.6.1
 [25] backports_1.1.4      assertthat_0.2.1     lazyeval_0.2.2
 [28] cli_1.1.0            later_0.8.0          formatR_1.7
 [31] prettyunits_1.0.2    acepack_1.4.1        htmltools_0.3.6
 [34] quantreg_5.41        tools_3.6.1          igraph_1.2.4.1
 [37] coda_0.19-3          gtable_0.3.0         glue_1.3.1
 [40] reshape2_1.4.3       vctrs_0.2.0          nlme_3.1-140
 [43] crosstalk_1.0.0      xfun_0.8             stringr_1.4.0
 [46] ps_1.3.0             mime_0.7             miniUI_0.1.1.1
 [49] gtools_3.8.1         polspline_1.1.15     MASS_7.3-51.4
 [52] zoo_1.8-6            scales_1.0.0         colourpicker_1.0
 [55] promises_1.0.1       Brobdingnag_1.2-6    parallel_3.6.1
 [58] sandwich_2.5-1       inline_0.3.15        shinystan_2.5.0
 [61] RColorBrewer_1.1-2   yaml_2.2.0           gridExtra_2.3
 [64] loo_2.1.0            rpart_4.1-15         latticeExtra_0.6-28
 [67] stringi_1.4.3        dygraphs_1.1.1.6     checkmate_1.9.4
 [70] boot_1.3-22          pkgbuild_1.0.3       rlang_0.4.0
 [73] pkgconfig_2.0.2      matrixStats_0.54.0   evaluate_0.14
 [76] purrr_0.3.2          labeling_0.3         rstantools_1.5.1
 [79] htmlwidgets_1.3      tidyselect_0.2.5     processx_3.4.0
```

```
 [82] plyr_1.8.4         magrittr_1.5        R6_2.4.0
 [85] multcomp_1.4-10    pillar_1.4.2        foreign_0.8-71
 [88] withr_2.1.2        xts_0.11-2          abind_1.4-5
 [91] nnet_7.3-12        tibble_2.1.3        crayon_1.3.4
 [94] utf8_1.1.4         rmarkdown_1.14      grid_3.6.1
 [97] data.table_1.12.2  callr_3.3.0         threejs_0.3.1
[100] digest_0.6.20      xtable_1.8-4        httpuv_1.5.1
[103] stats4_3.6.1       munsell_0.5.0       viridisLite_0.3.0
[106] shinyjs_1.0
```

```
[1] "C:/Users/eam2018/Documents/hierarchical-binomial-random-effects"
```

This took 349.89 seconds to execute.

```
[1] FALSE
```