Fitting a modified Connor-Mosimann Distribution to Elicited Quantiles of Multinomial Probabilities

Ed Wilson

2017-10-22

This package was created to fit a parametric distribution to multinomial probabilities elicited from experts.

Introduction

Expert elicitation is a common tool used when data to describe a phenomenon are either not yet available, or cannot be gathered for one reason or another. It has been employed in various fields including ecology, climate science, and more recently in Health Technology Assessment to plug data gaps in decision models.

Iglesias et al. recently published reporting guidelines for use of expert judgement in model based health economic evaluations, distinguishing between Delphi panels to collect and collate qualitative opinion and expert elicitation for quantitative opinion. This package is concerned with the latter.

Tools such as the SHEffield Elicitation Framework (SHELF), written by Tony O'Hagan and Jeremy Oakley focus on eliciting an expert's (or group of experts') range of plausible values for a quantity of interest, which is represented by a probability distribution showing the relative strength of belief supporting different values.

Recently, the SHELF approach has been adapted for elicitation of multinomial probabilities, for example to elicit proportions of patients in three or more disease states following a particular intervention. In this method, independent beta distributions are fitted around experts' beliefs about each proportion. A Dirichlet distribution is then fitted to all of these and the implicit sample size adjusted to provide the best overall replication of the marginal distributions.

However, the Dirichlet distribution is very restrictive, comprising only k parameters, equal to the number of dimensions / proportions to be elicited. (Specifically, the standard errors around the marginal distributions are functions of the overall implicit sample size alone.) A generalisation of the Dirichlet is the Connor-Mosimann (CM) distribution (see also Elfadaly & Garthwaite), a more flexible distribution described by a greater number of parameters. The proportions to be elicited are defined in terms of products of independent beta-distributed 'Z' variables. The CM distribution has 2(k-1) parameters allowing for more flexibility and thus in theory allowing a closer fit to the beliefs of the experts.

This vignette describes a package with a simple modification of the CM distribution, defining each Z variable as a scaled beta distribution. This modified-CM (mCM) is described by 4(k-1) parameters, allowing even greater flexibility and thus theoretically an improved fit over the CM and Dirichlet distributions.

Expert Elicitation

When devising the protocol for an expert elicitation exercise, two main modes of elicitation are generally used: the quantile and the roulette/histogram. In the roulette mode, experts place 'chips in bins' representing the relative strength of their belief about different values for a parameter. This is cognitively the least demanding and whilst commonly used to elicit continuous quantities and simple probabilities, it is less suitable for eliciting multinomial probabilities on the basis that a separate set of bins would need filling for each marginal probability. Relating these together into a multinomial distribution would be cognitively extremely challenging. Therefore, a quantile approach is preferred, where a minimum of 3 points along the distribution are elicited, typically the median and tertiles, quartiles or 95% credibility limits. The median is expressed as the value X at which the expert would place a 50:50 bet on the 'true value' being greater or less than X, and the lower and upper tertiles or quartiles being the values at which the expert would place a 2:1 or 3:1 bet respectively.

Fitting parametric distributions to elicited quantities

The Beta Distribution

Where data are binomial in nature, that is, a binary outcome such as respond/not respond or success/failure, a beta distribution can be fitted to elicited beliefs about the proportions in each group. The hyperparameters of the beta distribution are the number of successes (the alpha, denoted n_1 in equations 1 & 2) and failures (the beta, denoted n_2). The mean is the number of successes over the sum of successes and failures, and the variance a function of the parameters as per equations 1 & 2.

Equations 1 & 2: Beta distribution mean and variance

$$p = \frac{n_1}{(n_1 + n_2)}$$

$$Var(p) = \frac{n_1 n_2}{(n_1 + n_2)^2 (n_1 + n_2 + 1)}$$

Fitting a beta distribution to elicited points on the distribution (eg. median and tertiles) is a question of finding values for the hyperparameters n_1 and n_2 that provide the closest fit to those points. This can be achieved by defining a loss function. For example, the sum of the squared deviation (SSD) between the expert's median, lower and upper bounds compared with that yielded from a beta distribution with given values of n_1 and n_2 (equation 3). A numeric search algorithm is employed to find the values of n_1 and n_2 that minimises the SSD. Note the median of the beta distribution is not tractable, but can be approximated when n_1, $n_2 > 1$. Alternatively, it can be estimated empirically by Monte Carlo simulation: sampling many times from the distribution and calculating the resulting median, as well as lower and upper quantiles of interest.

Equation 3: Loss function (SSD)

$$SSD_(n_1, n_2) = (LL_T - LL_M)^2 + (MED_T - MED_M)^2 + (UL_T - UL_M)^2$$

Where LL = lower credible limit; MED = median; UL = upper credible limit; T = target (i.e. value elicited from expert); M = modelled (estimated from a given set of n_1 & n_2).

The Dirichlet Distribution

Where data are multinomial, that is where there are k outcomes (or dimensions) and k>2, the beta distribution can be generalised to the Dirichlet. The marginal means and variances are as per Equations 4 & 5. Note the parallels with the beta and where k=2, the Dirichlet reduces to the beta.

Equations 4 & 5: Dirichlet distribution (marginal) mean and variance

$$p_i = \frac{n_i}{n}$$

$$Var(p_i) = \frac{n_i(n - n_i)}{n^2(n+1)}$$

As with the beta, a loss function can be defined as the sum of the squared deviation between the elicited and computed values of the median and upper and lower bounds for a given set of hyperparameters $n_1 \dots n_k$ (Equation 6). For example, where there are three possible outcomes (dimensions), there will be 9 elements to the loss function (median, lower and upper credibility limit for each dimension). A search algorithm can once again be employed to identify n_1, n_2 and n_3 that minimises the loss function.

Equation 6: Loss function (SSD)

$$SSD_(n_1, n_2, \dots, n_k) = \sum_{i=1}^{k} [(LL_T - LL_M)^2 + (MED_T - MED_M)^2 + (UL_T - UL_M)^2]$$

Where $n = n_1 + n_2 + \dots + n_k$ *and i=1 to k*

The Connor-Mosimann distribution

The Dirichlet distribution may provide a poor fit to the elicited data, due to its limited number of parameters (equal to k, the number of dimensions). The Connor-Mosimann is a generalisation of the Dirichlet, formally described in full elsewhere (e.g. https://link.springer.com/article/10.1007/s11749-013-0336-4 and https://www.jstor.org/stable/2283728?seq=1#page_scan_tab_contents). Briefly, the CM defines a total of k-1 'Z' parameters as independent betas, each defined in terms of an alpha and beta hyperparameter, with finally $Z_k = 1$. The proportions (probabilities) are then defined as the product of the Z parameters as per Equation 7. The total number of hyperparameters is now 2(k-1) rather than k. An example with a three-dimensional problem is illustrated in Equation 8; a search algorithm is employed to identify a_1, a_2, b_1 and b_2 that minimises the loss function.

Equation 7: Connor-Mosimann distribution

$$Z_j \sim Beta(a_j, b_j); j = 1, \dots, (k-1)$$
$$Z_k = 1$$
$$P_1 = Z_1$$
$$P_j = Z_j \prod_{i=1}^{j-1} (1 - Z_i); j = 2, \dots, k$$

Equation 8: Illustration with k=3

$$Z_1 \sim Beta(a_1, b_1)$$
$$Z_2 \sim Beta(a_2, b_2)$$
$$Z_3 = 1$$
$$P_1 = Z_1$$
$$P_2 = Z_2(1 - Z_1)$$
$$P_3 = Z_3(1 - Z_2)(1 - Z_1) = (1 - Z_2)(1 - Z_1)$$

The modified Connor-Mosimann (mCM) distribution

To increase the flexibility of the Connor-Mosimann distribution, the Z parameters can be defined as scaled beta distributions, as per Equations 9 & 10. I refer to this as a modified Connor-Mosimann or mCM distribution. The scaled beta simply rescales a beta distribution between A and B, the lower and upper limits. Note the CM is a special case of the mCM where A = 0 and B = 1. The number of hyperparameters describing the distribution is now 4(k-1), providing added flexibility to more precisely model the opinions of the experts. The illustration with 3 dimensions is shown in Equation 11. A minimisation algorithm is again employed to find values of a_1, b_1, A_1, B_1, a_2, b_2, A_2, B_2 that provide the best fit to the elicited data (i.e. minimises the SSD).

Equation 9: Scaled beta distribution

$$X \sim ScaledBeta(a, b, A, B) = A + (B - A)Beta(a, b)$$

Equation 10: Modified Conor-Mosimann distribution

$$Z_j \sim ScaledBeta(a_j, b_j, A_j, B_j); j = 1, \ldots, (k-1), 0 > A_j, B_j > 1$$
$$Z_k = 1$$
$$P_1 = Z_1$$
$$P_j = Z_j \prod_{i=1}^{j-1}(1 - Z_i); j = 2, \ldots, k$$

Equation 11: Illustration with k=3

$$Z_1 \sim ScaledBeta(a_1, b_1, A_1, B_1)$$
$$Z_2 \sim ScaledBeta(a_2, b_2, A_2, B_2)$$
$$Z_3 = 1$$
$$P_1 = Z_1$$
$$P_2 = Z_2(1 - Z_1)$$
$$P_3 = Z_3(1 - Z_2)(1 - Z_1) = (1 - Z_2)(1 - Z_1)$$

Example application and use of the modcmfitr package

The remainder of this vignette illustrates the application of the Dirichlet, CM and mCM to some ficticious data from two experts. A comparison of the goodness of fit is shown by comparing the SSDs, and resulting distributions plotted graphically to allow visual comparison. (Note that a larger number of experts is usually desirable, but two is sufficient to illustrate the use of this package.)

In this example, a three-state Markov model has been designed to represent disease progression under no treatment. The states are remission, progression and dead. There are no objective data on the transition probabilities and it would be unethical to withhold treatment from patients in order to observe the natural history of the disease. Therefore an expert elicitation exercise was undertaken to estimate the annual probabilities of a patient either moving from the remission stage to the progressive stage, dying or remaining in remission.

Two experts provided their median beliefs and lower and upper 95% credibility bounds (i.e. 2.5th and 97.5th centile) as to the rates of progression based on their clinical experience. During the elicitation exercise, the experts were told that their medians had to sum to 100%, but that the credibility bounds were to represent the uncertainty in their belief, and so could vary anywhere between the the logical limits of 0 and 100%, and the median.

Table 1: Expert 1

	LL	MED	UL
Remission	0.43	0.55	0.65
Progression	0.16	0.27	0.46
Dead	0.03	0.18	0.23

Table 2: Expert 2

	LL	MED	UL
Remission	0.35	0.6	0.70
Progression	0.15	0.3	0.45
Dead	0.00	0.1	0.20

Expert 1 believes that a patient in the remission state has a 27% probability of being in the progression state in one year's time, but this could be anywhere between 16% and 46%. Expert 2's beliefs are virtually identical at 30% (15%, 45%). However, there is some disagreement between the experts in terms of the probability of death: Expert 1 gives a median probability of 18% (3%, 23%), whereas Expert 2 feels this is a little pessimistic and that 10% is more plausible, but the range of uncertainty is similar (0% to 20%).

Fitting the mCM distribution

There are five inputs to the fitModCM() function: Outcomes, RawData, SearchParams, ModCMorCM, and quantiles. The code to set up Expert 1's data is:

```
library(modcmfitr)
library(nloptr)

set.seed(12345)
Outcomes <- c("Remission","Progression","Dead")
RawData <- matrix(data = c(0.43, 0.55, 0.65,
                           0.16, 0.27, 0.46,
                           0.03, 0.18, 0.23
           ),ncol=3,byrow=TRUE)
SearchParams <- c(10000,100) # number of iterations, max number of searches
#                              (set to 100 here; you probably want more).
ModCMorCM <- 1 # flag to determine whether fitting mCM or CM distribution
Quantiles <- c(0.025,0.5,0.975) # example here is 95% credibility limits and median.

mCM <- fitModCM(Outcomes, RawData, SearchParams, ModCMorCM, Quantiles)
```

The code uses the controlled random search algorithm, crs2lm() in the nloptr package to find a set of parameters for the mCM distribution that minimises the SSD between the target and modelled quantiles. By default the routine spits out the SSD, which you should see get closer to zero as the algorithm homes in on a better fitting set of parameters. The output is a matrix with 10 columns and number of rows equal to the number of dimensions (health states), in this case 3:

```
print(mCM)
```

```
##                    a         b          L         U Tgt_LL Tgt_MED Tgt_UL
## Remission   2.265655 0.5268807 0.28119554 0.6493573   0.43    0.55   0.65
## Progression 7.832954 6.4716505 0.05040548 0.9866657   0.16    0.27   0.46
## Dead        0.000000 0.0000000 0.00000000 0.0000000   0.03    0.18   0.23
##             Mdl_LL Mdl_MED Mdl_UL
## Remission     0.39    0.61   0.65
## Progression   0.13    0.23   0.39
## Dead          0.08    0.18   0.32
```

The first four columns are the parameters of the mCM distribution. Note the last row is always zeros. These can be dropped from the reporting (remember the number of parameters is 4(k-1), where k is the number of dimensions), so the mCM distribution parameters in this case are:

```
##           a         b          L         U
## Z1 2.265655 0.5268807 0.28119554 0.6493573
## Z2 7.832954 6.4716505 0.05040548 0.9866657
```

The Zeds are scaled beta distributions with alpha and beta of a and b, scaled to an lower and upper limit of L and U respectively.

The goodness of fit can be assessed informally by comparing the target ('Tgt') lower limit, median and upper limit with the modelled ('Mdl') lower limit, median and upper limit, which is the final six columns of the output:

```
print(mCM[,5:10])
```

```
##             Tgt_LL Tgt_MED Tgt_UL Mdl_LL Mdl_MED Mdl_UL
## Remission     0.43    0.55   0.65   0.39    0.61   0.65
## Progression   0.16    0.27   0.46   0.13    0.23   0.39
## Dead          0.03    0.18   0.23   0.08    0.18   0.32
```

If these are identical then great, the fit is good. If the numbers are substantially different it may be worth re-running the function increasing the max searches. Note this will increase the time it takes to process. I found 10,000 to give reasonable results for one particular application, but it is important to experiment.

Sampling from the mCM distribution

The function rModCM() can be used to sample from the modified Connor-Mosimann distribution. The arguments are n, the number of samples required, and Z, the vector of scaled betas:

```
Z <- mCM[1:2,1:4]
rownames(Z) <- c("Z1","Z2")
print(Z)
```

```
##           a         b          L         U
## Z1 2.265655 0.5268807 0.28119554 0.6493573
## Z2 7.832954 6.4716505 0.05040548 0.9866657
```

```
mCMSamples <- rModCM(10,Z)
colnames(mCMSamples) <- c("Remission", "Progression", "Dead")
print(mCMSamples)
```

```
##       Remission Progression      Dead
##  [1,] 0.6347137   0.1645515 0.2007347
##  [2,] 0.4263337   0.3214777 0.2521886
##  [3,] 0.6491545   0.2277376 0.1231079
##  [4,] 0.6298489   0.1910991 0.1790520
##  [5,] 0.6491003   0.1624964 0.1884033
##  [6,] 0.6486226   0.1503900 0.2009874
##  [7,] 0.5926821   0.2991236 0.1081944
##  [8,] 0.6470179   0.2309131 0.1220690
##  [9,] 0.5232470   0.2802522 0.1965008
## [10,] 0.4044346   0.3064357 0.2891296
```

The rModCM function can be incorporated into a probabilistic decision model written in R, or else sets of sampled values can be copied and pasted into an Excel-based model as desired.

Fitting distributions to multiple experts at once

The fitMultipleCM() function takes the elicited quantiles of a number of experts simultaneously and runs fitModCM for each expert, so saves having to call fitModCM multiple times. The output is a matrix with the Zed parameters in the same format as per the fitModCM() function, but with a set of rows for each expert:

```
Quantiles <- c(0.025,0.5,0.975) # to fit median and 95% Credibility Intervals
SearchParams <- c(10000,100) # number of iterations, max number of searches
#                              (set to 100 here; you probably want more).

RawData <- data.frame(expert = as.character(c(1,1,1,2,2,2)),
                      Outcome = as.factor(c("Remission","Progression","Dead",
                                            "Remission","Progression","Dead")),
                      LL = as.numeric(c(0.43, 0.16, 0.03, 0.35, 0.15, 0.00)),
                     MED = as.numeric(c(0.55, 0.27, 0.18, 0.60, 0.30, 0.10)),
                      UL = as.numeric(c(0.65, 0.46, 0.23, 0.70, 0.45, 0.20)))

mCMMultipleExpert <- fitMultiplemCM(Quantiles, SearchParams, RawData)
```

```
print(mCMMultipleExpert)
```

```
##   Expert     Outcome        a        b          L         U Tgt_LL Tgt_MED
## 1      1   Remission 8.123141 7.160583 0.05117034 0.9297403   0.43    0.55
## 2      1 Progression 8.159584 6.218602 0.15625376 0.9387078   0.16    0.27
## 3      1        Dead 0.000000 0.000000 0.00000000 0.0000000   0.03    0.18
## 4      2   Remission 6.905326 3.088188 0.29295453 0.6741848   0.35    0.60
## 5      2 Progression 4.779386 3.030027 0.46405155 0.8922537   0.15    0.30
## 6      2        Dead 0.000000 0.000000 0.00000000 0.0000000   0.00    0.10
##   Tgt_UL Mdl_LL Mdl_MED Mdl_UL
## 1   0.65   0.31    0.52   0.73
## 2   0.46   0.15    0.28   0.46
## 3   0.23   0.08    0.19   0.33
## 4   0.70   0.44    0.56   0.64
## 5   0.45   0.23    0.32   0.43
## 6   0.20   0.06    0.12   0.20
```

Combining results from multiple experts

There are different approaches to combining the results from multiple experts. The SHELF process recommends a consensus-based approach in a face to face workshop, where a crude aggregation of the individual experts' beliefs is presented as a starting point. This is then modified until the group is satisfied that the resulting distribution is a satisfactory representation of its beliefs.

Other, mathematical, approaches are also possible. The mergeMultiplemCM() function uses a numeric linear pool approach. It samples from the individual experts' distributions many times, then calculates the overall quantiles and medians from the samples.

The function returns a matrix representing the lower, median and upper limits of the pooled distribution. This can then be fed into fitModCM() to generate a modified Connor-Mosimann distribution representing the overall spread of the experts' beliefs:

```
NrSamples <- 100000
RawData <- data.frame(expert = as.character(c(1,1,1,2,2,2)),
                      Outcome = as.factor(c("Remission","Progression","Dead",
                                            "Remission","Progression","Dead")),
                      a = mCMMultipleExpert$a,
                      b = mCMMultipleExpert$b,
                      L = mCMMultipleExpert$L,
                      U = mCMMultipleExpert$U)
mCMmerged <- mergeMultiplemCM(NrSamples,RawData)
```

```
##
##  Expert 1
##  Expert 2
```

```
mCMmerged <- round(mCMmerged,2)
print(mCMmerged)
```

```
##               LL  MED   UL
## Remission   0.34 0.55 0.69
## Progression 0.17 0.31 0.44
## Dead        0.07 0.14 0.31
```

Now using fitModCM() to generate the parameters of the pooled distribution:

```
Outcomes <- c("Remission","Progression","Dead")
RawData <- mCMmerged
SearchParams <- c(10000,100) #number of iterations, max number of searches
#                             (set to 100 here; you probably want more).
ModCMorCM <- 1
Quantiles <- c(0.025,0.5,0.975) # example here is 95% credibility limits and median.

mCMMultipleMerged <- fitModCM(Outcomes, RawData, SearchParams, ModCMorCM, Quantiles)
```

```
print(mCMMultipleMerged)
```

```
##                  a        b         L         U Tgt_LL Tgt_MED Tgt_UL
## Remission   8.1020 3.998697 0.1084331 0.6944812   0.34    0.55   0.69
## Progression 6.4675 5.149526 0.3419121 0.8844415   0.17    0.31   0.44
## Dead        0.0000 0.000000 0.0000000 0.0000000   0.07    0.14   0.31
##             Mdl_LL Mdl_MED Mdl_UL
## Remission     0.34    0.51   0.63
## Progression   0.21    0.32   0.46
## Dead          0.10    0.17   0.28
```

Note that the medians resulting from mergeMultiplemCM() will not necessarily sum to 100%. However, individual samples from the distribution that is fitted to this will.

Fitting a Dirichlet distribution

This package also includes code for fitting a Dirichlet distribution to elicited data. fitDirichlet() functions in the same manner as fitModCM(), but requires the rdirichlet() function from the gtools package:

```
library(gtools)
Outcomes <- c("Remission","Progression","Dead")
RawData <- matrix(data = c(0.43, 0.55, 0.65,
                           0.16, 0.27, 0.46,
                           0.03, 0.18, 0.23
            ),ncol=3,byrow=TRUE)
SearchParams <- c(10000,100) #number of iterations, max number of searches
#                              (set to 100 here; you probably want more).
Quantiles <- c(0.025,0.5,0.975) # example here is 95% credibility limits and median.

D <- fitDirichlet(Outcomes, RawData, SearchParams, Quantiles)
D <- round(D,2)
```

```
print(D)
```

```
##              Dirichlet Tgt_LL Tgt_MED Tgt_UL Mdl_LL Mdl_MED Mdl_UL
## Remission    446696.96   0.43    0.55   0.65   0.55    0.55   0.55
## Progression 290834.54   0.16    0.27   0.46   0.36    0.36   0.36
## Dead          78755.56   0.03    0.18   0.23   0.10    0.10   0.10
```

The Dirichlet is a very restrictive distribution with only k parameters, where k is the number of dimensions, so generally there will be a poor match between the target and modelled quantiles. Sampling from the Dirichlet can be done with the rdirichlet() function in the gtools package.

Fitting a Connor-Mosimann distribution

This is identical to fitting a modified Connor-Mosimann distribution, except setting the ModCMorCM flag to 0. The lower and upper limits default to 0 and 1 respectively, and the Zed parameters are defined as beta distributions with parameters a and b:

```
Outcomes <- c("Remission","Progression","Dead")
RawData <- matrix(data = c(0.43, 0.55, 0.65,
                          0.16, 0.27, 0.46,
                          0.03, 0.18, 0.23
           ),ncol=3,byrow=TRUE)
SearchParams <- c(10000,100)  #number of iterations, max number of searches
#                               (set to 100 here; you probably want more).
ModCMorCM <- 0 # flag to determine whether fitting mCM or CM distribution
Quantiles <- c(0.025,0.5,0.975) # example here is 95% credibility limits and median.

CM <- fitModCM(Outcomes, RawData, SearchParams, ModCMorCM, Quantiles)
```

```
print(CM)
```

```
##                    a        b L U Tgt_LL Tgt_MED Tgt_UL Mdl_LL Mdl_MED
## Remission    9.627784 8.495418 0 1   0.43    0.55   0.65   0.31    0.53
## Progression 5.562518 4.636343 0 1   0.16    0.27   0.46   0.10    0.25
## Dead         0.000000 0.000000 0 0   0.03    0.18   0.23   0.07    0.20
##              Mdl_UL
## Remission      0.75
## Progression    0.46
## Dead           0.42
```

To sample from the CM distribution, simply call the rModCM() function, including the 0 and 1 as lower and upper limits:

```
Z <- CM[1:2,1:4]
rownames(Z) <- c("Z1","Z2")
print(Z)
```

```
##           a        b L U
## Z1 9.627784 8.495418 0 1
## Z2 5.562518 4.636343 0 1
```

```
CMSamples <- rModCM(10,Z)
colnames(CMSamples) <- c("Remission", "Progression", "Dead")
print(CMSamples)
```

```
##       Remission Progression      Dead
##  [1,] 0.6001578   0.2780383 0.1218038
##  [2,] 0.4025043   0.1876086 0.4098871
##  [3,] 0.6312159   0.2139844 0.1547997
##  [4,] 0.4996642   0.2774382 0.2228976
##  [5,] 0.5081807   0.1362555 0.3555638
##  [6,] 0.5973754   0.2608341 0.1417906
##  [7,] 0.7507639   0.1340998 0.1151363
##  [8,] 0.4944196   0.2439033 0.2616771
##  [9,] 0.3834891   0.4421603 0.1743506
## [10,] 0.4148549   0.4401569 0.1449882
```

Comparison of fit between mCM, CM and Dirichlet distributions

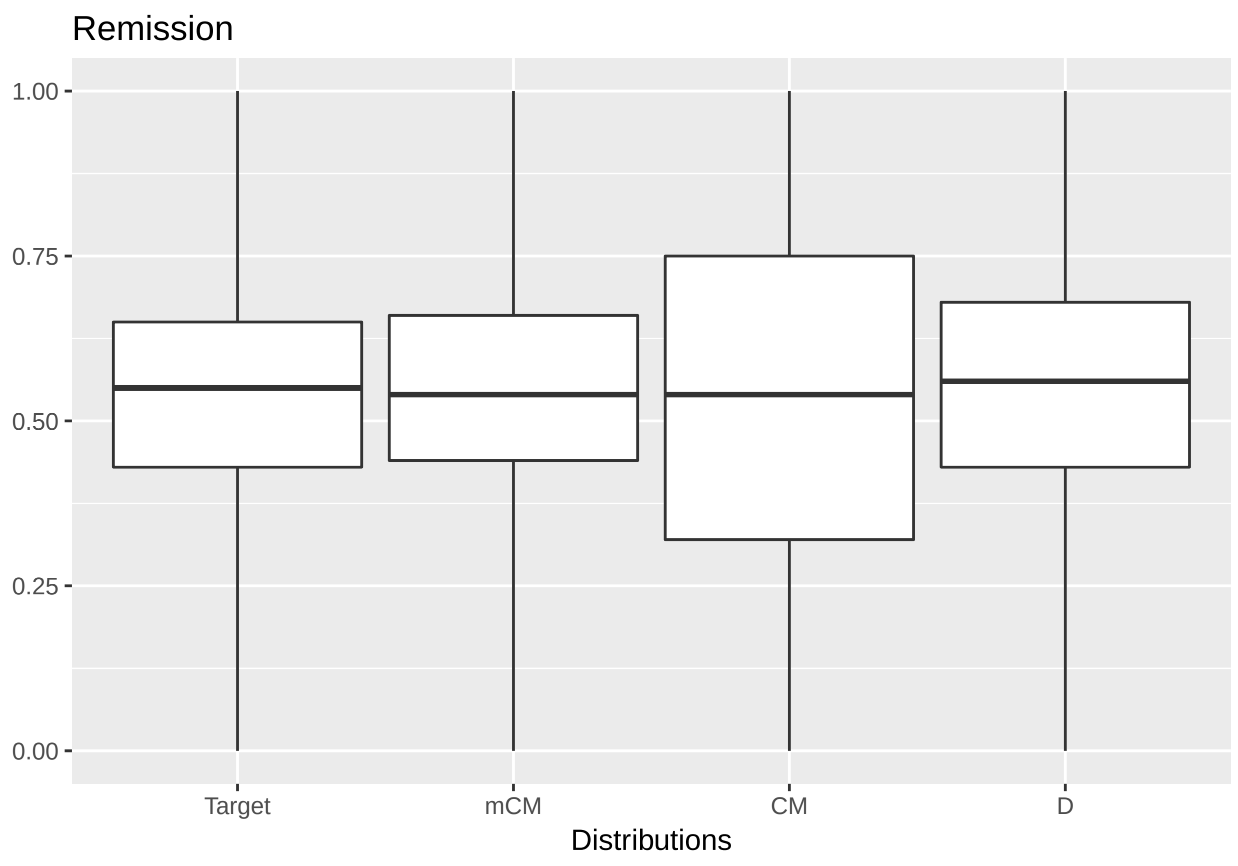
The final section of this vignette shows how well the three distributions fit to elicited data.

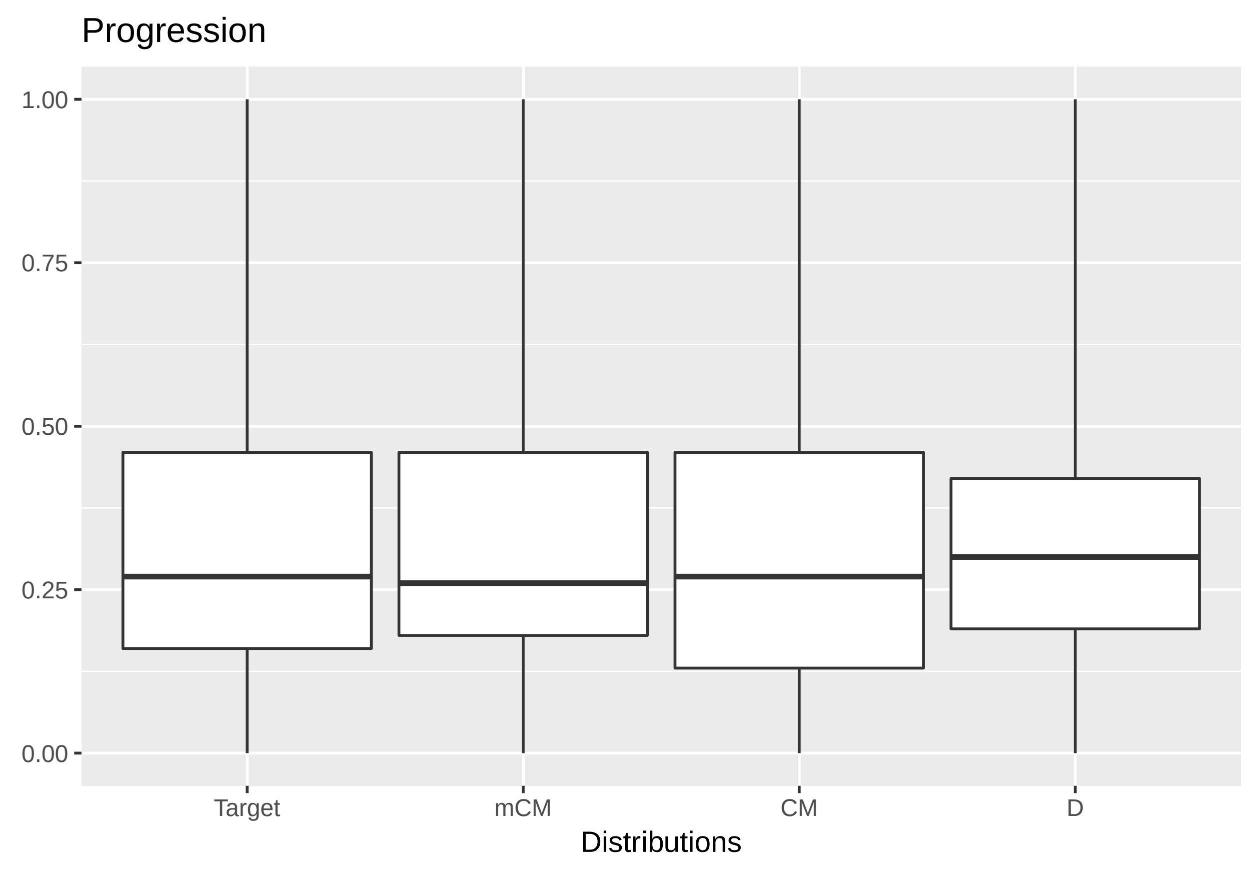
The boxes show the lower and upper 95% credibility bounds and median as elicited from the expert ('Target'), and for each of the mCM, CM and Dirichlet distributions for the example three dimensional problem used in this vignette. Finally the goodness of fit (i.e. SSD) is also shown in the final table.

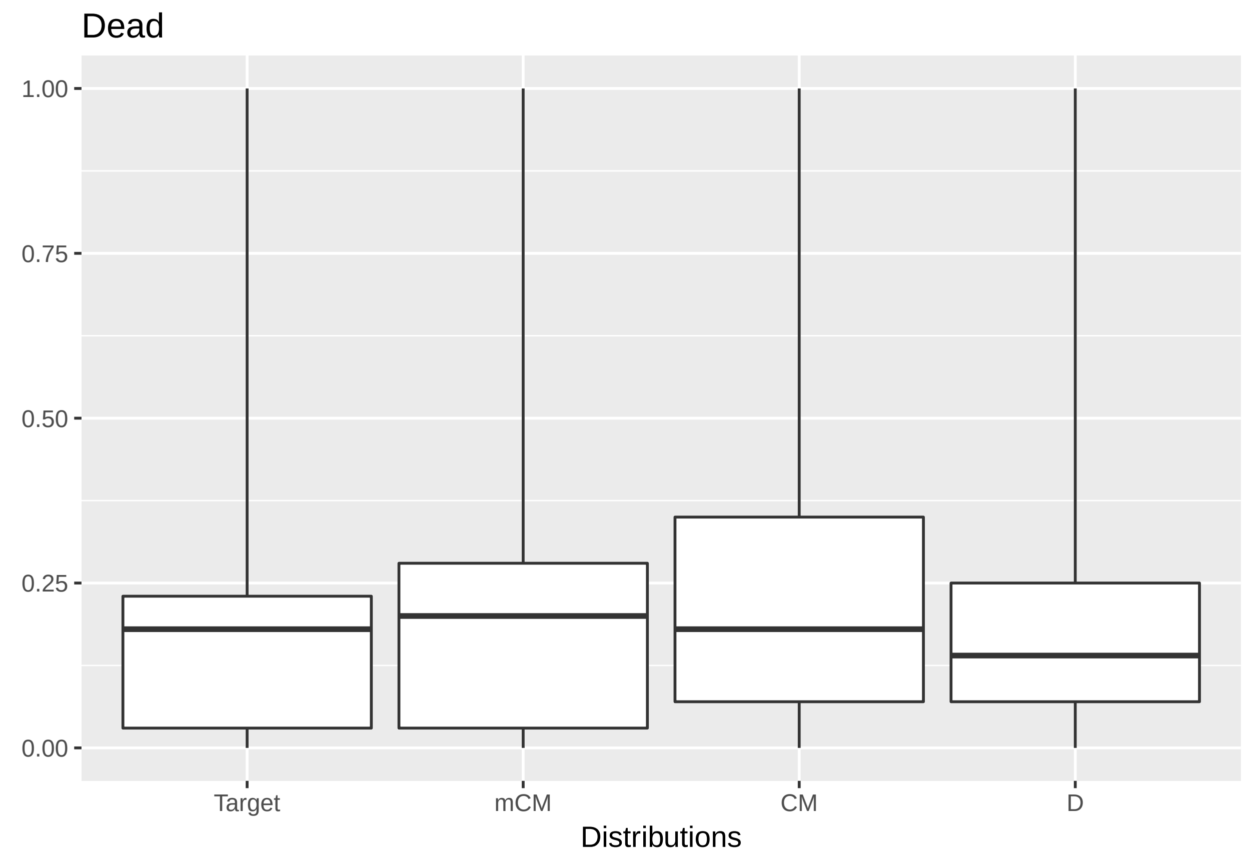
```
library(ggplot2)

#results from a run of fitModCM with 10,000 iterations
mCM <- matrix(data=c(6.1598312, 7.387483, 0.3443956, 0.7802353,
                                 0.43, 0.55, 0.65, 0.44, 0.54, 0.66,
                     0.4274036, 1.153597, 0.4664444, 0.9830223,
                                 0.16, 0.27, 0.46, 0.18, 0.26, 0.46,
                     0.0000000, 0.000000, 0.0000000, 0.0000000,
                                 0.03, 0.18, 0.23, 0.03, 0.20, 0.28),
               nrow=3, byrow=T,
              dimnames = list(c("Remission","Progression","Dead"),
                              c("a","b","L","U",
                                "Tgt_LL","Tgt_MED","Tgt_UL",
                                "Mdl_LL","Mdl_MED","Mdl_UL")))

#results from a run of fitModCM with 10,000 iterations, ModCMorCM=0
#(i.e. fitting a CM Distribution)
CM <- matrix(data=c(10, 8.589266, 0, 1,
                          0.43, 0.55, 0.65, 0.32, 0.54, 0.75,
                    10, 6.793693, 0, 1,
                            0.16, 0.27, 0.46, 0.13, 0.27, 0.46,
                     0, 0.000000, 0, 0,
                            0.03, 0.18, 0.23, 0.07, 0.18, 0.35),
               nrow=3, byrow=T,
              dimnames = list(c("Remission","Progression","Dead"),
                              c("a","b","L","U",
                                "Tgt_LL","Tgt_MED","Tgt_UL",
                                "Mdl_LL","Mdl_MED","Mdl_UL")))

#results from a run of fitDirichlet with 10,000 iterations
D <- matrix(data=c(30.51,
                        0.43, 0.55, 0.65, 0.43, 0.56, 0.68,
                   16.43,
                        0.16, 0.27, 0.46, 0.19, 0.30, 0.42,
                    8.12,
                        0.03, 0.18, 0.23, 0.07, 0.14, 0.25),
               nrow=3, byrow=T,
              dimnames = list(c("Remission","Progression","Dead"),
                              c("Dirichlet",
                                "Tgt_LL","Tgt_MED","Tgt_UL",
                                "Mdl_LL","Mdl_MED","Mdl_UL")))

df <- data.frame(
  Distributions = c("Target","mCM","CM","D"),
  y0 = c(0,0,0,0),
  y2_5 = c(0.43,mCM[1,8],CM[1,8],D[1,5]),
  y50 = c(0.55,mCM[1,9],CM[1,9],D[1,6]),
  y97_5 = c(0.65,mCM[1,10],CM[1,10],D[1,7]),
  y100 = c(1,1,1,1)
)
df$Distributions = factor(df$Distributions,levels=c("Target","mCM","CM","D")
                          ,ordered=TRUE) # order the distributions
ggplot(df, aes(Distributions)) +
  geom_boxplot(
   aes(ymin = y0, lower = y2_5, middle = y50, upper = y97_5, ymax = y100),
   stat = "identity") +
  ggtitle("Remission")

df$y2_5 = c(0.16,mCM[2,8],CM[2,8],D[2,5])
df$y50 = c(0.27,mCM[2,9],CM[2,9],D[2,6])
df$y97_5 = c(0.46,mCM[2,10],CM[2,10],D[2,7])
ggplot(df, aes(Distributions)) +
  geom_boxplot(
   aes(ymin = y0, lower = y2_5, middle = y50, upper = y97_5, ymax = y100),
   stat = "identity") +
  ggtitle("Progression")

df$y2_5 = c(0.03,mCM[3,8],CM[3,8],D[3,5])
df$y50 = c(0.18,mCM[3,9],CM[3,9],D[3,6])
df$y97_5 = c(0.23,mCM[3,10],CM[3,10],D[3,7])
ggplot(df, aes(Distributions)) +
  geom_boxplot(
    aes(ymin = y0, lower = y2_5, middle = y50, upper = y97_5, ymax = y100),
    stat = "identity") +
  ggtitle("Dead")

#calculate SSD for each fit:
SSD <- matrix(data=c(sum((mCM[,5:7]-mCM[,8:10])^2),
                     sum((CM[,5:7]-CM[,8:10])^2),
                     sum((D[,2:4]-D[,5:7])^2)),
                     ncol=3,
                     dimnames = (list(c("SSD"),c("mCM","CM","D"))))

kable(SSD, caption="Goodness of fit", align='l')
```

Table 3: Goodness of fit

	mCM	CM	D
SSD	0.0037	0.0391	0.008




Progression
1.00
0.75
0.50
0.25
0.00
Target
mCM
CM
D
Distributions



References

Connor R, Mosimann J. Concepts of independence for proportions with a generalisation of the Dirichlet distribution. J Am Stat Assoc 1969;64(325):194-206

Elfadaly F, Garthwaite P. Eliciting Dirichlet and Connor-Mosimann prior distributions for multinomial models. Test 2013;22(4):628-646

Iglesias et al. Reporting Guidelines for the Use of Expert Judgement in Model-Based Economic Evaluations. Pharmacoeconomics 2016;34(11):1161-72

Johnson SG, The NLopt nonlinear-optimization package, http://ab-initio.mit.edu/nlopt

Kerman J. A closed-form approximation for the median of the beta distribution. arXiv:1111.0433v1, November 2011.

Oakley, J (2017). SHELF: Tools to Support the Sheffield Elicitation Framework. R package version 1.2.3. https://CRAN.R-project.org/package=SHELF

O'Hagan T, Oakley J. The SHeffield Elicitation Framework. http://www.tonyohagan.co.uk/shelf/

Sonnenberg F, Beck J. Markov Models in Medical Decision Making: A practical guide. Medical Decision Making 1993;13:322-338)

Warnes GR, Bolker B, Lumley T (2015). gtools: Various R Programming Tools. R package version 3.5.0. https://CRAN.R-project.org/package=gtools

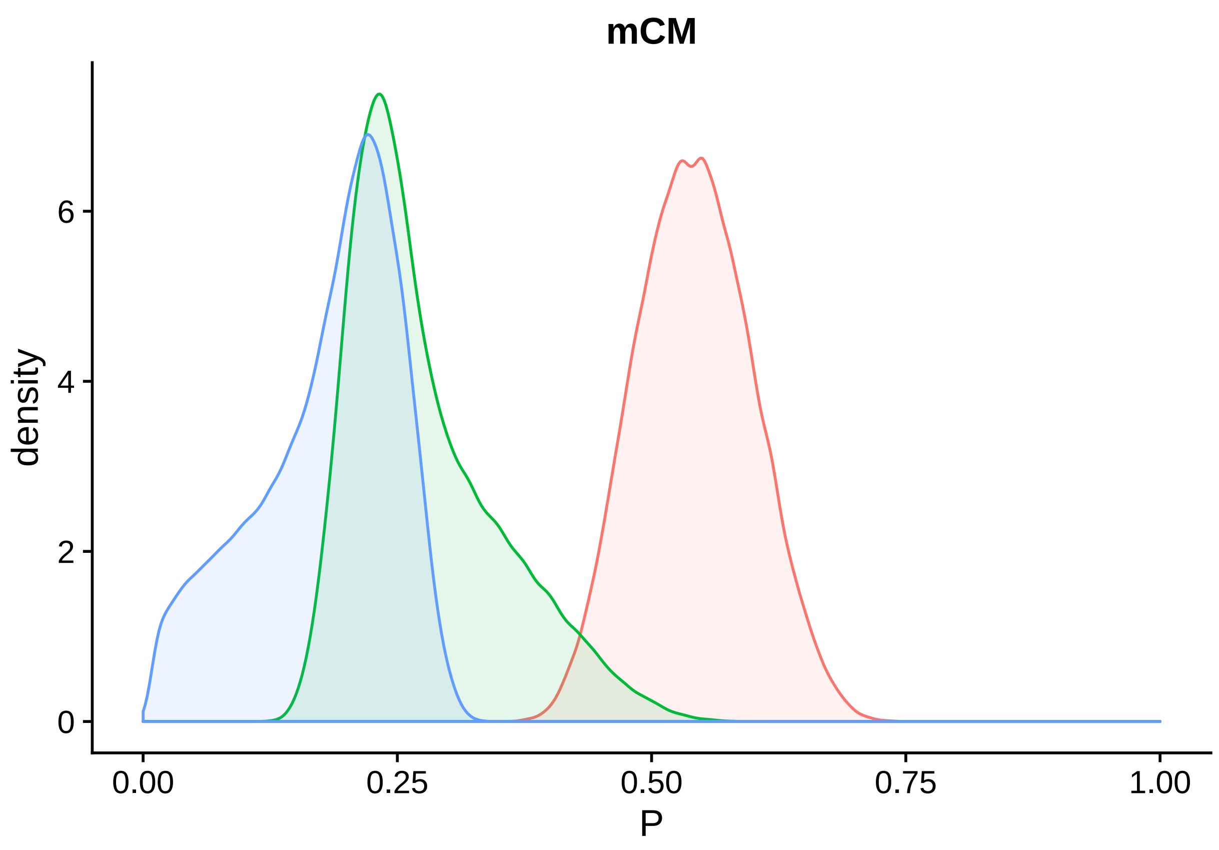
Appendix

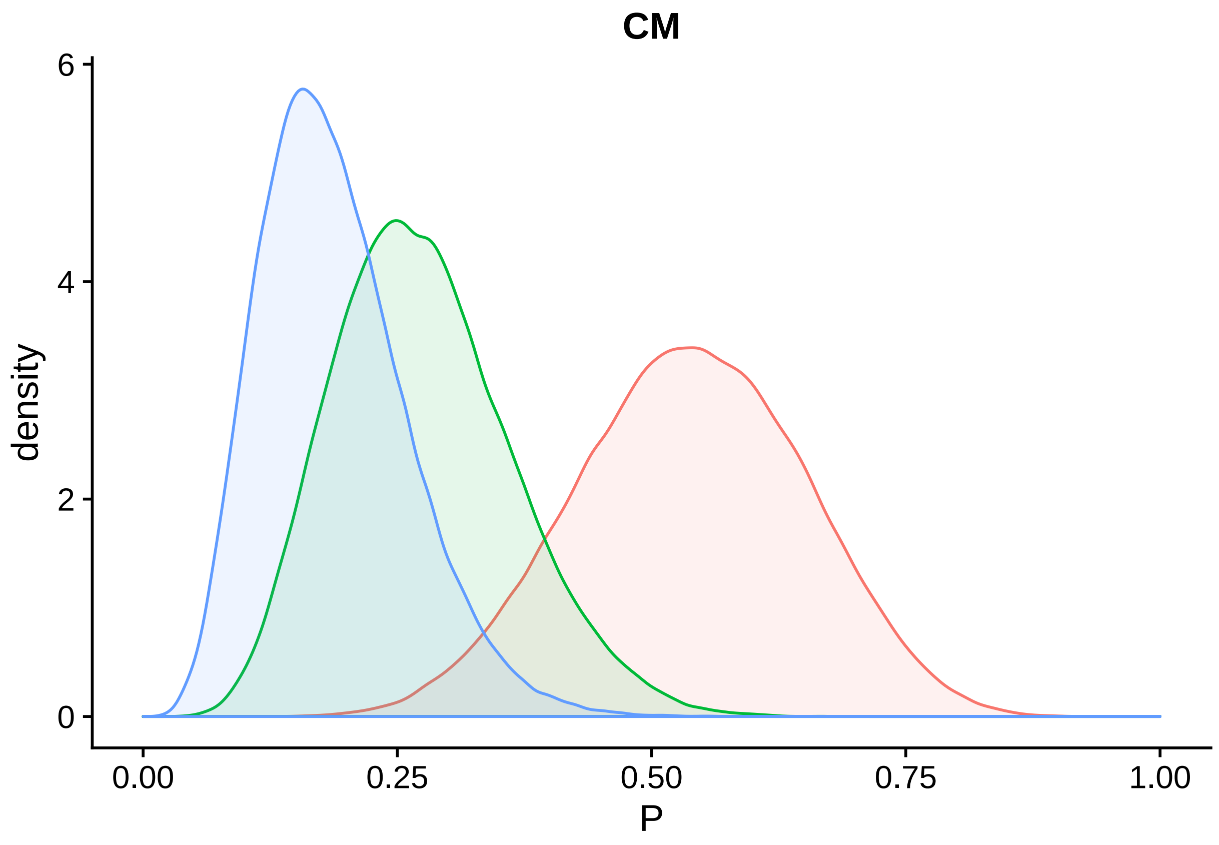
Plotting the densities is a useful visualisation. This shows the densities from the three-dimensional example used in this vignette.

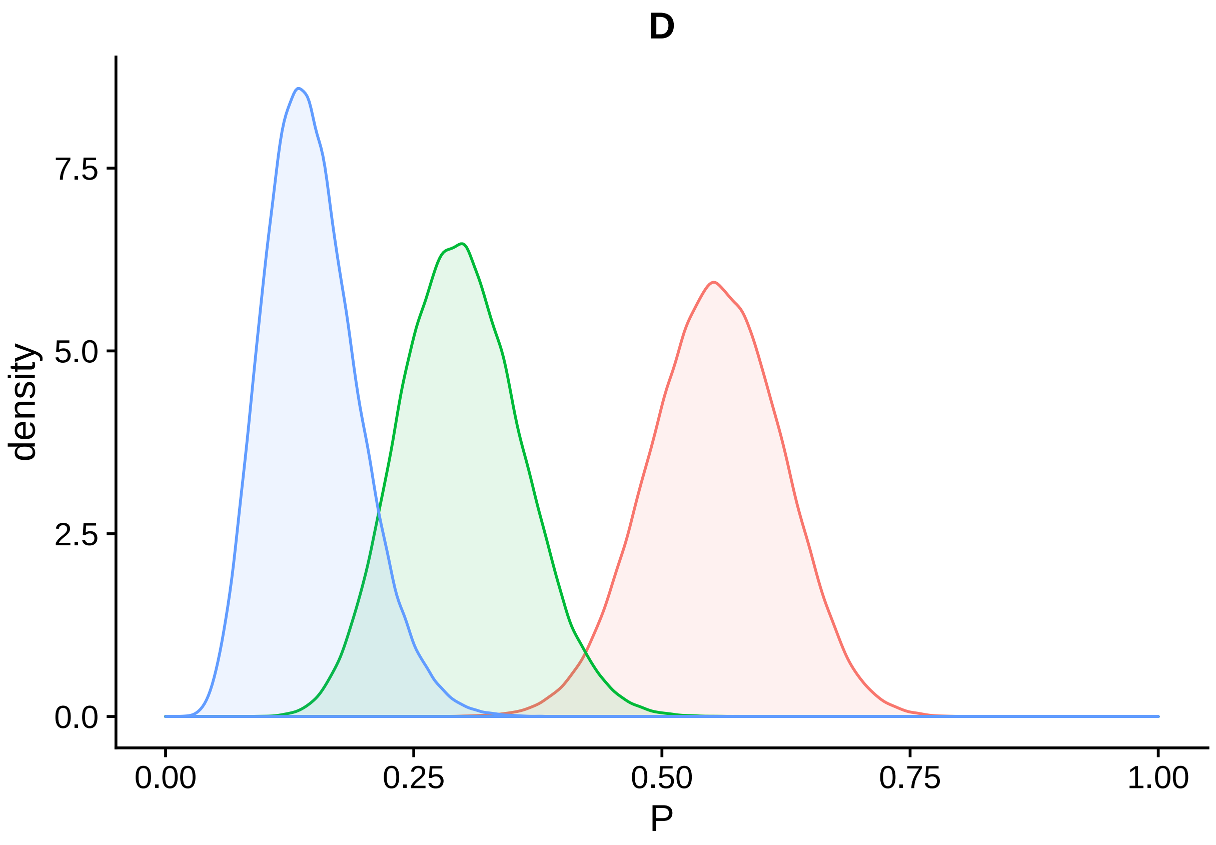
```r
library(tidyr)
library(cowplot)
```

```
##
## Attaching package: 'cowplot'

## The following object is masked from 'package:ggplot2':
##
##     ggsave
```

```r
samples <- 100000
mCMSamples <- cbind("mCM",rModCM(samples,mCM[1:2,1:4]))
CMSamples <- cbind("CM",rModCM(samples,CM[1:2,1:4]))
DSamples <- cbind("D",rdirichlet(samples,D[,1]))
sampled <- rbind(mCMSamples,CMSamples,DSamples)
sampled <- as.data.frame(sampled)
sampled[,2:4] <- apply(sampled[,2:4],c(1,2),as.numeric)
colnames(sampled) <- c("Distribution","Remission","Progression","Dead")

sampled <- gather(sampled,"State","P",2:4)
sampled$State <- factor(sampled$State,levels=c("Remission","Progression","Dead"),
                        ordered=TRUE)
sampled$Distribution <- factor(sampled$Distribution,levels=c("mCM","CM","D"),
                               ordered=TRUE)

mCMDens <- ggplot(data=subset(sampled, sampled$Distribution=="mCM"),
                  aes(x=P,colour=State, fill=State)) +
  geom_density(alpha=0.1, show.legend=T) +
  scale_x_continuous(name="P", limits= c(0,1)) +
  ggtitle("mCM")

legend <- get_legend(mCMDens)
mCMDens <- mCMDens + theme(legend.position='none')

CMDens <- ggplot(data=subset(sampled, sampled$Distribution=="CM"),
                 aes(x=P,colour=State, fill=State)) +
  geom_density(alpha=0.1, show.legend=F) +
  scale_x_continuous(name="P", limits= c(0,1)) +
  ggtitle("CM")

DDens <- ggplot(data=subset(sampled, sampled$Distribution=="D"),
                aes(x=P,colour=State, fill=State)) +
  geom_density(alpha=0.1, show.legend=F) +
  scale_x_continuous(name="P", limits= c(0,1)) +
  ggtitle("D")

ggdraw(mCMDens)
ggdraw(CMDens)
ggdraw(DDens)
ggdraw(legend)
```


mCM
density
6
4
2
0
0.00
0.25
0.50
0.75
1.00
P


CM
density
6
4
2
0
0.00
0.25
0.50
0.75
1.00
P


D
density
7.5
5.0
2.5
0.0
0.00
0.25
0.50
0.75
1.00
P

State

🟥 Remission

🟩 Progression

🟦 Dead