

# EXCHANGEABLE MODEL FOR MULTINOMIAL DATA

ANIKO SZABO

**ABSTRACT.** We implement parameter estimation for exchangeable multinomial data, including estimation under marginal compatibility.

## 1. PRELIMINARIES

We will be using object of `CMData` class, which is defined in `CMData.w`.

We will also need to load support libraries.

```
"..\R\ExchMultinomial.R" 1≡
```

```
    #'@importFrom combinat hcube
```

```
    ◇
```

File defined by [1](#), [2](#), [4b](#), [5](#), [6g](#), [8b](#), [9ab](#), [10abc](#), [12](#), [14b](#).

## 2. EXCHANGEABLE MULTINOMIAL MODEL

**2.1. Definitions.** Let  $\mathbf{R} = (R_1, \dots, R_K)^T$  follow an exchangeable multinomial distribution with  $K + 1$  categories. We parameterize it by

$$\tau_{r_1, \dots, r_K | n} = \mathbb{P} [\mathcal{X}_{\{1, \dots, r_1\}}(O_1), \dots, \mathcal{X}_{\{\sum_{i=1}^{k-1} r_i + 1, \dots, \sum_{i=1}^k r_i\}}(O_k)] \quad (k = 1, \dots, K), \quad (1)$$

where  $r_i \geq 0$  and  $r_1 + \dots + r_K \leq n$ . For notational convenience, also let  $\tau_{0, \dots, 0} = 1$ .

**2.2. Estimation.** Consider  $\tau_{r_1, \dots, r_K | n}$  and its unconditional counterpart

$$\theta_{r_1, \dots, r_K} = \mathbb{P} [\mathcal{X}_{\{1, \dots, r_1\}}(O_1), \dots, \mathcal{X}_{\{\sum_{i=1}^{K-1} r_i + 1, \dots, \sum_{i=1}^K r_i\}}(O_K)] = \sum_{n=\sum r_i}^C \tau_{r_1, \dots, r_K | n} \mathbb{P}(N = n).$$

If  $A_{r_1, \dots, r_K | n}$  denotes the number of clusters of size  $n$  with response vector  $(r_1, \dots, r_K)$ , then their non-parametric estimates are

$$\hat{\tau}_{r_1, \dots, r_K | n} = \sum_{s_1, \dots, s_K} \frac{\binom{n - \sum r_i}{s_1, \dots, s_K}}{\binom{n}{r_1 + s_1, \dots, r_K + s_K}} \frac{A_{r_1, \dots, r_K | n}}{M_n}, \quad (2)$$

and

$$\hat{\theta}_{r_1, \dots, r_K} = \sum_{n=1}^M \sum_{s_1, \dots, s_K} \frac{\binom{n - \sum r_i}{s_1, \dots, s_K}}{\binom{n}{r_1 + s_1, \dots, r_K + s_K}} \frac{A_{r_1 + s_1, \dots, r_K + s_K | n}}{M}. \quad (3)$$

The function `tau` creates a “look-up table” for the MLEs. It returns either a list by treatment group of either  $K + 1$  or  $K$  dimensional arrays, depending on whether cluster-size specific estimates ( $\tau$ 's) or averaged estimates ( $\theta$ 's) are requested. For the cluster-size specific estimates the first dimension is the cluster size. The calculation of  $\theta$ 's is done separately for each dose level, and thus each dose level uses a different sample-size distribution for averaging.

```
"..\R\ExchMultinomial.R" 2≡
```

```
#'@rdname CorrBin-internal
< Define function for multinomial coefficient 14a >
#' Estimate joint event probabilities for multinomial data
#'
#' An exchangeable multinomial distribution with  $\text{eqn}\{K+1\}$  categories  $\text{eqn}\{0_1, \dots, 0_{K+1}\}$ , can be
#' parameterized by the joint probabilities of events
#'  $\text{eqn}\{\tau_{r_1, \dots, r_K} | n\} = P\{X_1 = \dots = X_{r_1} = 0_1, \dots, X_{\sum_{i=1}^{K-1} r_i + 1} = \dots =$ 
#' where  $\text{eqn}\{r_i \geq 0\}$  and  $\text{eqn}\{r_1 + \dots + r_K \leq n\}$ .
#' The {jointprobs} function estimates these probabilities under various settings.
#' Note that when some of the  $\text{eqn}\{r_i\}$ 's equal zero, then no restriction on the number of outcomes of the
#' corresponding type are imposed, so the resulting probabilities are marginal.
#'
#'@param cmdata a {CMDData} object
#'@param type character string describing the desired type of estimate:
#' \describe{
#' \item{"averaged"}{ - averaged over the observed cluster-size distribution within each treatment}
#' \item{"cluster"}{ - separately for each cluster size within each treatment}
#' \item{"mc"}{ - assuming marginal compatibility, ie that  $\text{eqn}\{\tau\}$  does not depend on the cluster-size}
#' }
#'@return a list with an array of estimates for each treatment. For a multinomial distribution with
#'  $\text{eqn}\{K+1\}$  categories the arrays will have either  $\text{eqn}\{K+1\}$  or  $\text{eqn}\{K\}$  dimensions, depending on whether
#' cluster-size specific estimates ({type="cluster"}) or pooled estimates
#' ({type="averaged"} or {type="mc"}) are requested. For the cluster-size specific estimates
#' the first dimension is the cluster-size. Each additional dimension is a possible outcome.
#'
#'@seealso {\link{mc.est}} for estimating the distribution under marginal compatibility,
#' {\link{uniprbs}} and {\link{multi.corr}} for extracting the univariate marginal event
#' probabilities, and the within-multinomial correlations from the joint probabilities.
#'@examples
#'data(dehp)
#'# averaged over cluster-sizes
#'tau.ave <- jointprobs(dehp, type="ave")
#'# averaged  $P(X_1=X_2=0_1, X_3=0_2)$  in the 1500 dose group
#'tau.ave[["1500"]][["2","1"]] # there are two type-1, and one type-2 outcome
#'
#'plot P( $X_1=0_1$ ) - the marginal probability of a type-1 event over cluster-sizes
#'tau <- jointprobs(dehp, type="cluster")
#'ests <- as.data.frame(lapply(tau, function(x)x[, "1", "0"]))
#'matplot(ests, type="b")
#'@export
#'@importFrom stats xtabs

jointprobs <- function(cmdata, type=c("averaged","cluster","mc")){
  type <- match.arg(type)

  < Extract info from cmdata into variables 3a >
  # multinomial lookup table
  mctab <- mChooseTable(M, nc, log=FALSE)

  res <- list()
  for (trt in levels(cmdata$Trt)){
    cm1 <- cmdata[cmdata$Trt==trt,]
    # observed freq lookup table
    atab <- array(0, dim=rep(M+1, nc))
    a.idx <- data.matrix(cm1[,nrespvars])
    atab[a.idx + 1] <- atab[a.idx + 1] + cm1$Freq

    if (type=="averaged"){
      Mn <- sum(cm1$Freq)
      < Calculate averaged thetas 3b >
    } else if (type=="cluster") {
      Mn <- xtabs(Freq ~ factor(ClusterSize, levels=1:M), data=cm1)
      < Calculate cluster-specific taus 3k >
    } else if (type=="mc") {
```

$\langle \text{Extract info from cmdata into variables 3a} \rangle \equiv$

```
nc <- attr(cmdata, "ncat")
nrespvars <- paste("NResp", 1:nc, sep=".")
M <- max(cmdata$ClusterSize)
```

◇

Fragment referenced in 2, 6g.

First, we define the MLE averaged over cluster sizes. The **Calculate averaged thetas** macro creates a  $K$ -dimensional array of  $\theta_{r_1, \dots, r_K}(d)$  values. The implementation is based on combining the two summations of the definition into one using  $n = \sum_{i=1}^K r_i + \sum_{i=1}^K s_i + s_{K+1}$ :

$$\begin{aligned} \hat{\theta}_{r_1, \dots, r_K} &= \sum_{n=1}^M \sum_{s_1, \dots, s_K} \frac{\binom{n - \sum r_i}{s_1, \dots, s_K}}{\binom{n}{r_1 + s_1, \dots, r_K + s_K}} \frac{A_{r_1 + s_1, \dots, r_K + s_K | n}}{M} \\ &= \sum_{s_1, \dots, s_{K+1}} \frac{\binom{\sum s_i}{s_1, \dots, s_K}}{\binom{\sum r_i + \sum s_i}{r_1 + s_1, \dots, r_K + s_K}} \frac{A_{r_1 + s_1, \dots, r_K + s_K | \sum r_i + \sum s_i}}{M}. \end{aligned} \quad (4)$$

$\langle \text{Calculate averaged thetas 3b} \rangle \equiv$

```
res.trt <- array(NA, dim=rep(M+1, nc-1))
dimnames(res.trt) <- rep.int(list(0:M), nc-1)
names(dimnames(res.trt)) <- paste("R", 1:(nc-1), sep="")
# indices for possible values of r
< Simplex with sums (3c idx ,3d M ,3e nc-1 ,3f idxsum ) 13 >
# indices for possible values of s
# (one more column than for r - ensures summation over all n's)
< Simplex with sums (3g sidx ,3h M ,3i nc ,3j sidxsum ) 13 >
for (i in 1:nrow(idxx)){
  r <- idx[i,]
  s.idx <- which(sidxsum <= M-sum(r))
  lower.idx <- sidx[s.idx, , drop=FALSE]
  upper.idx <- lower.idx + rep(c(r,0), each=nrow(lower.idx))
  res.trt[rbind(r)+1] <-
    sum(mctab[lower.idx+1] / mctab[upper.idx+1] * atab[upper.idx+1]) / Mn
}
```

◇

Fragment referenced in 2.

Next, we define the MLEs specific for each cluster size. The macro **Calculate cluster-specific taus** creates a  $K + 1$  dimensional array, with the cluster size as the first dimension.

$\langle \text{Calculate cluster-specific taus 3k} \rangle \equiv$

```
res.trt <- array(NA, dim=c(M, rep(M+1, nc-1))) #first dimension is 'n'
dimnames(res.trt) <- c(list(1:M), rep.int(list(0:M), nc-1))
names(dimnames(res.trt)) <- c("N", paste("R", 1:(nc-1), sep=""))
for (n in which(Mn > 0)){
  # indices for possible values of r
  < Simplex with sums (3l idx ,3m n ,3n nc-1 ,3o idxsum ) 13 >
  for (i in 1:nrow(idxx)){
    r <- idx[i,]
```

```

s.idx <- which(idxsum <= n-sum(r))
lower.idx <- idx[s.idx, , drop=FALSE]
upper.idx <- lower.idx + rep(r, each=nrow(lower.idx))
lower.idx <- cbind(lower.idx, n-sum(r)-idxsum[s.idx]) #add implied last column
upper.idx <- cbind(upper.idx, n-sum(r)-idxsum[s.idx]) #add implied last column
res.trt[cbind(n,rbind(r)+1)] <-
  sum(mctab[lower.idx+1] / mctab[upper.idx+1] * atab[upper.idx+1]) / Mn[n]
}
}

```

Fragment referenced in 2.

The code for calculating marginally-compatible *tau*'s is described in the next section.

(*Calculate MC taus 4a*)  $\equiv$

```

pim <- mc.estraw(cm1)[[1]] #only one treatment group
res.trt <- tau.from.pi(pim)

```

Fragment referenced in 2.

Uses: `mc.estraw` 6g, `tau` 2, `tau.from.pi` 8b.

### 3. MARGINAL COMPATIBILITY

Under marginal compatibility,

$$\pi_{\mathbf{r}|n} = \sum_{\mathbf{t} \in \mathcal{V}_M} h(\mathbf{r}, \mathbf{t}, n) \pi_{\mathbf{t}|M}, \quad (5)$$

where  $h(\mathbf{r}, \mathbf{t}, n) = \binom{\mathbf{t}}{\mathbf{r}} \binom{M - \sum r_i}{n - \sum r_i} / \binom{M}{n} = \prod_{i=1}^K \binom{t_i}{r_i} \binom{M - \sum r_i}{n - \sum r_i} / \binom{M}{n}$  and  $\mathcal{V}_n = \{(v_1, \dots, v_K) \in \mathbb{N}^K \mid v_i \geq 0, \sum v_i \leq n\}$  is a  $K$ -dimensional simplex lattice with maximum sum  $n$ .

**3.1. Estimation.** The following code implements the EM-algorithm for estimating the probabilities of response assuming marginal compatibility. Let  $(\mathbf{r}_i, n_i)$ ,  $i = 1, \dots, N$  denote the observed data for a given dose level, where  $i$  iterates through the clusters,  $n_i$  is the cluster size and  $\mathbf{r}_i = (r_1, \dots, r_K)$  is the observed number of responses of each type.

$$\pi_{\mathbf{t}|M}^{(t+1)} = \frac{1}{N} \sum_{i=1}^N h(\mathbf{r}_i, \mathbf{t}, n_i) \frac{\pi_{\mathbf{t}|M}^{(t)}}{\pi_{\mathbf{r}_i|n_i}^{(t)}}, \quad (6)$$

The `mc.est.CMData` function implements the `mc.est` S3 method for `CMData` objects, returning a data frame with all  $\pi_{\mathbf{r}|n}^{(g)}$ ,  $n = 1, \dots, M$  probabilities. The ‘hard’ work is done by the `mc.estraw` function, which returns a list of matrices with  $\pi_{\mathbf{r}|M}^{(g)}$  values.

"..\R\ExchMultinomial.R" 4b $\equiv$

```

#'@rdname mc.est
#'@method mc.est CMData
#'@export
#'@param eps numeric; EM iterations proceed until the sum of squared changes fall below \code{eps}
#'@return For \code{CMData}: A data frame giving the estimated pdf for each treatment and
#'clustersize. The probabilities add up to 1
#'for each \code{Trt}/\code{ClusterSize} combination. It has the following columns:
#'@return \item{Prob}{numeric, the probability of \code{NResp} responses in a
#'cluster of size \code{ClusterSize} in group \code{Trt}}
#'@return \item{Trt}{factor, the treatment group}
#'@return \item{ClusterSize}{numeric, the cluster size}
#'@return \item{NResp.1 - NResp.K}{numeric, the number of responses of each type}

```

```

#'
#'@note
#'For multinomial data, the implementation is currently written in R, so it is not very fast.
#'
#'@references George EO, Cheon K, Yuan Y, Szabo A (2016) On Exchangeable Multinomial Distributions. #'
#'@examples
#'data(dehp)
#'dehp.mc <- mc.est(subset(dehp, Trt=="0"))
#'subset(dehp.mc, ClusterSize==2)

mc.est.CMData <- function(object, eps=1E-6, ...){

  nc <- attr(object, "ncat")
  resp.vars1 <- paste("NResp", 1:(nc-1), sep=".")

  res <- mc.estraw(object=object, eps=eps, ...)
  margres <- lapply(res, Marginals) # has only NResp.1 - NResp.K

  mat.to.df <- function(idx, alist){
    dd <- as.data.frame.table(alist[[idx]], responseName="Prob")
    dd[c("N", resp.vars1)] <- lapply(dd[c("N", resp.vars1)], function(x)as.numeric(as.character(x)))
    dd$Trt <- names(alist)[idx]
    dd
  }
  margres <- lapply(1:length(margres), mat.to.df, alist=margres)
  fin <- do.call(rbind, margres)
  names(fin)[1] <- "ClusterSize"
  last.resp <- paste("NResp", nc, sep=".")
  fin[last.resp] <- fin$ClusterSize - rowSums(fin[resp.vars1]) # calculated omitted frequency
  fin$Trt <- factor(fin$Trt)
  fin <- fin[fin[last.resp] >= 0,] #remove impossible clusters
  fin[c("Trt", "ClusterSize", resp.vars1, last.resp, "Prob")]
}

```

File defined by [1](#), [2](#), [4b](#), [5](#), [6g](#), [8b](#), [9ab](#), [10abc](#), [12](#), [14b](#).

Defines: `mc.est.CMData` Never used.

Uses: `Marginals` [5](#), `mc.estraw` [6g](#).

First we write a help-function that calculates all the probabilities  $\pi_{\mathbf{r}|n}$  given the set of  $\theta_{\mathbf{r}} = \pi_{\mathbf{r}|M}$ . While there are a variety of ways doing this, we use a recursive formula:

$$\pi_{\mathbf{r}|n} = \sum_{i=1}^K \frac{r_i + 1}{n + 1} \pi_{\mathbf{r} + \mathbf{d}_i | n+1} + \frac{n - \sum_i r_i + 1}{n + 1} \pi_{\mathbf{r} | n+1}, \quad (7)$$

where  $\mathbf{d}_i$  is the  $i$ th coordinate basis vector (i.e. all its elements are 0, except the  $i$ th, which is 1).

The input for `Marginals` is a  $K$ -dimensional array of  $\pi_{\mathbf{r}|M}$ , and the output is a  $(K+1)$ -dimensional array with the values of  $\pi_{\mathbf{r}|n}$ ,  $n = 1, \dots, M$  with cluster size as the first dimension

"..\R\ExchMultinomial.R" 5≡

```

#'@rdname CorrBin-internal
Marginals <- function(theta){
  K <- length(dim(theta))
  M <- dim(theta)[1]-1

  res <- array(0, dim=c(M, rep(M+1, K)))
  dimnames(res) <- c(N=list(1:M), dimnames(theta))

  # indices for possible values of r

```

```

< Simplex with sums (6a idx ,6b M ,6c K+1 ,6d clustersize ) 13>
idx <- idx[ , -1, drop=FALSE] #remove (K+1)st category

< Initialize for cluster size M 6e>
for (cs in seq.int(M-1,1)){
  < Calculate values for cluster size cs given values for size cs+1 6f>
}

res
}

```

File defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.

The initialization just copies over the values from `theta` to the appropriate dimension. Note that when indexing the arrays, a “+1” is necessary since `idx` is 0-based.

< Initialize for cluster size M 6e>  $\equiv$

```

curridx <- idx[clustersize==M, ,drop=FALSE]
res[cbind(M, curridx+1)] <- theta[curridx+1]

```

Fragment referenced in 5.

The iterative step initializes with the last term (with  $\pi_{r|n+1}$ ) and loops over the basis vectors.

< Calculate values for cluster size cs given values for size cs+1 6f>  $\equiv$

```

curridx <- idx[clustersize==cs, , drop=FALSE]
res[cbind(cs, curridx+1)] <- (cs+1- rowSums(curridx))/(cs+1) * res[cbind(cs+1, curridx+1)]
for (j in 1:K){
  lookidx <- curridx
  lookidx[,j] <- lookidx[,j] + 1 #add 1 to the j-th coordinate
  res[cbind(cs, curridx+1)] <- res[cbind(cs, curridx+1)] +
    lookidx[,j]/(cs+1) * res[cbind(cs+1, lookidx+1)]
}

```

Fragment referenced in 5.

The actual EM iterations are performed in `mc.estraw`.

"..\R\ExchMultinomial.R" 6g $\equiv$

```

#'@rdname CorrBin-internal
#'@name mc.estraw
mc.estraw <- function(object, ...) UseMethod("mc.estraw")

#'@rdname CorrBin-internal
#'@method mc.estraw CMDData
mc.estraw.CMDData <- function(object, eps=1E-6, ...){
  cmdata <- object
  < Extract info from cmdata into variables 3a>

  # indices for possible values of r with clustersize = M
  < Simplex with sums (6h idx ,6i M ,6j nc-1 ,6k idxsum ) 13>

  res <- list()
  for (trt in levels(cmdata$Trt)){
    cm1 <- cmdata[cmdata$Trt==trt,]
  }
}

```

```

if (nrow(cm1) > 0){
  # observed freq lookup table
  atab <- array(0, dim=rep(M+1, nc))
  a.idx <- data.matrix(cm1[,nrespvars])
  atab[a.idx + 1] <- atab[a.idx + 1] + cm1$Freq
  Mn <- sum(cm1$Freq)

  < MC estimates for given dose group 7 >

  # append treatment-specific result to result list
  dimnames(res.trt) <- rep.int(list(0:M), nc-1)
  names(dimnames(res.trt)) <- paste("NResp", 1:(nc-1), sep=".")
  res.trt <- list(res.trt)
} else {
  res.trt <- list(c())
}
res <- c(res, res.trt)
}
names(res) <- levels(cmdata$Trt)
res
}

```

File defined by [1](#), [2](#), [4b](#), [5](#), [6g](#), [8b](#), [9ab](#), [10abc](#), [12](#), [14b](#).

Within each dose group, the algorithm iterates until the sum of squared changes of the parameters is smaller than the selected threshold `eps`.

*< MC estimates for given dose group 7 >*  $\equiv$

```

res.trt <- array(NA, dim=rep(M+1, nc-1))

#starting values
res.trt[idx + 1] <- 1/nrow(idx)

sqerror <- 1
#EM update
while (sqerror > eps){
  sqerror <- 0
  marg <- Marginals(res.trt)
  res.new <- array(NA, dim=rep(M+1, nc-1))
  res.new[idx + 1] <- 0

  < Calculate res.new - the value of res.trt for next iteration 8a >

  sqerror <- sum((res.new[idx+1] - res.trt[idx+1])^2)
  res.trt <- res.new
}

```

Fragment referenced in [6g](#).

Uses: `Marginals` [5](#).

The update of the  $\pi_{\mathbf{t}|M}$  is performed based on [\(6\)](#) rewritten to combine clusters of the same type:

$$\pi_{\mathbf{t}|M}^{(t+1)} = \frac{1}{N} \sum_{(\mathbf{r}, n)} \frac{A_{\mathbf{r}, n}}{\pi_{\mathbf{r}|n}^{(t)}} h(\mathbf{r}, \mathbf{t}, n) \pi_{\mathbf{t}|M}^{(t)}, \quad (8)$$

looping through each cluster type  $(\mathbf{r}, n)$ , and updating all  $\pi_{\mathbf{t}|M}$  values compatible with this type. The compatible  $\mathbf{t}$  vectors have  $t_i \geq r_i$ , so they can be written in the form  $\mathbf{t} = \mathbf{r} + \mathbf{s}$ , where  $s_i \geq 0$  and  $\sum s_i \leq M - \sum r_i$ .

*( Calculate res.new - the value of res.trt for next iteration 8a )  $\equiv$*

```
for (i in 1:nrow(cm1)){
  rlong <- data.matrix(cm1[,nrespvars])[i,]      #nc elements
  r <- rlong[-nc]                                #without the last category
  n <- cm1$ClusterSize[i]
  # indices to which this cluster type contributes
  s.idx <- which(idxsum <= M-sum(r))
  tidx <- idx[s.idx, , drop=FALSE] + rep(r, each=length(s.idx))

  hvals <- apply(tidx, 1, function(tvec)prod(choose(tvec, r)) * choose(M-sum(tvec), n-sum(r)))
  hvals <- hvals / choose(M, n)
  res.new[tidx+1] <- res.new[tidx+1] + atab[rbind(rlong)+1] / marg[rbind(c(n,r+1))] / Mn *
                                hvals * res.trt[tidx+1]
}
```

◇

Fragment referenced in 7.

**3.2. Manipulating estimates.** It is helpful to have functions that can convert the marginally compatible estimates from the  $\pi$ -based form obtained in the estimates to the  $\tau$ 's and to extract the variance-covariance matrix and the correlation parameters.

The `tau.from.pi` function takes a  $K$ -dimensional array of  $\pi_{\mathbf{r}}$  values, and returns a  $K$ -dimensional array of  $\tau_{\mathbf{r}}$  values using

$$\tau_{\mathbf{r}} = \sum_{\mathbf{s}} \frac{\binom{n-\sum r_i}{\mathbf{s}}}{\binom{n}{\mathbf{r}+\mathbf{s}}} \pi_{\mathbf{r}+\mathbf{s}}. \quad (9)$$

*"..\R\ExchMultinomial.R" 8b  $\equiv$*

```
#'@rdname CorrBin-internal
tau.from.pi <- function(pimat){
  K <- length(dim(pimat))
  n <- dim(pimat)[1] - 1
  res <- array(NA, dim=rep(n+1, K))
  dimnames(res) <- rep.int(list(0:n), K)
  names(dimnames(res)) <- paste("R", 1:K, sep="")

  # multinomial lookup table
  mctab <- mChooseTable(n, K+1, log=FALSE)

  # indices for possible values of r
  ( Simplex with sums (8c idx ,8d n ,8e K ,8f idxsum ) 13 )
  for (i in 1:nrow(idx)){
    r <- idx[i,]
    s.idx <- which(idxsum <= n-sum(r))
    lower.idx <- idx[s.idx, , drop=FALSE]
    upper.idx <- lower.idx + rep(r, each=nrow(lower.idx))
    lower.mc.idx <- cbind(lower.idx, n-sum(r)-idxsum[s.idx]) #add implied last column
    upper.mc.idx <- cbind(upper.idx, n-sum(r)-idxsum[s.idx]) #add implied last column
    res[rbind(r)+1] <-
      sum(mctab[lower.mc.idx+1] / mctab[upper.mc.idx+1] * pimat[upper.idx+1])
  }
  res
}
```

◇



File defined by [1](#), [2](#), [4b](#), [5](#), [6g](#), [8b](#), [9ab](#), [10abc](#), [12](#), [14b](#).

The `p.from.tau` function takes a  $K$ -dimensional array of  $\tau_r$  values, and returns a vector of marginal probabilities of success  $\tau_{d_i}$ .

"..\R\ExchMultinomial.R" 9a≡

```
#'@rdname CorrBin-internal
p.from.tau <- function(taumat){
  K <- length(dim(taumat))
  idx <- diag(nrow=K)
  taumat[rbind(idx+1)]
}
```

File defined by [1](#), [2](#), [4b](#), [5](#), [6g](#), [8b](#), [9ab](#), [10abc](#), [12](#), [14b](#).

Uses: `tau` [2](#).

The `uniprobs` function provides a wrapper to `p.from.tau` so that it works with the list output of the `jointprobs` function.

"..\R\ExchMultinomial.R" 9b≡

```
#'Extract univariate marginal probabilities from joint probability arrays
#
#Calculates the marginal probability of each event type for exchangeable correlated multinomial
#data based on joint probability estimates calculated by the \code{\link{jointprobs}} function.
# @param jp the output of \code{\link{jointprobs}} - a list of joint probability arrays by treatment
# @param type one of c("averaged","cluster","mc") - the type of joint probability. By default,
# the \code{type} attribute of \code{jp} is used.
# @return a list of estimated probability of each outcome by treatment group. The elements are either
# matrices or vectors depending on whether cluster-size specific estimates were requested
# (\code{type="cluster"}) or not.
# @export
# @seealso \code{\link{jointprobs}} for calculating the joint probability arrays
# @examples
# data(dehp)
# tau <- jointprobs(dehp, type="averaged")
# uniprobs(tau)
#
# #separately for each cluster size
# tau2 <- jointprobs(dehp, type="cluster")
# uniprobs(tau2)

uniprobs <- function(jp, type=attr(jp, "type")){
  type <- match.arg(type, c("averaged","cluster","mc"))

  get.probs <- function(tt){
    p <- p.from.tau(tt)
    c(p, 1-sum(p)) #add probability of last event type
  }

  if (type=="cluster") {
    res <- lapply(jp, function(x)apply(x, 1, get.probs))
  } else {
    res <- lapply(jp, get.probs)
  }
  res
}
```

◇  
 File defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.  
 Defines: `uniprbs` 2.  
 Uses: `tau` 2.

The function `corr.from.pi` calculates the within- and between-outcome correlation coefficients for the exchangeable model. It takes a  $K$ -dimensional array of  $\pi_r$  values, and returns a 2-dimensional matrix of  $\phi_{ij}$ ,  $i, j = 1, \dots, K$  values using

$$\phi_{ij} = \begin{cases} [\tau(2\mathbf{d}_i) - \tau(\mathbf{d}_i)^2] / [\tau(\mathbf{d}_i)(1 - \tau(\mathbf{d}_i))] & i = j \\ -[\tau(\mathbf{d}_i + \mathbf{d}_j) - \tau(\mathbf{d}_i)\tau(\mathbf{d}_j)] / [\tau(\mathbf{d}_i)\tau(\mathbf{d}_j)] & i \neq j, \end{cases} \quad (10)$$

where  $\mathbf{d}_i = (0, \dots, 0, \overbrace{1}^i, 0, \dots, 0)$ .

The function `corr.from.tau` does the same calculation, except it starts with  $\tau$ 's.

"..\R\ExchMultinomial.R" 10a≡

```
#'@rdname CorrBin-internal
corr.from.tau <- function(taumat){
  K <- length(dim(taumat))

  idx <- diag(nrow=K)
  numerator <- outer(1:K, 1:K, function(i,j){
    taumat[idx[i,]+idx[j,]+1] - taumat[idx[i,]+1] * taumat[idx[j,]+1]})
  denominator <- outer(1:K, 1:K, function(i,j){
    taumat[idx[i,]+1] * ifelse(i==j, 1-taumat[idx[i,]+1], -taumat[idx[j,]+1]))
  })
  res <- numerator / denominator #the negative sign is in the denominator
  res
}
```

◇  
 File defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.  
 Defines: `corr.from.tau` 10bc, 12.  
 Uses: `tau` 2.

"..\R\ExchMultinomial.R" 10b≡

```
#'@rdname CorrBin-internal
corr.from.pi <- function(pimat){
  tt <- tau.from.pi(pimat)
  res <- corr.from.tau(tt)
  res
}
```

◇  
 File defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.  
 Defines: `corr.from.pi` Never used.  
 Uses: `corr.from.tau` 10a, `tau` 2, `tau.from.pi` 8b.

Finally, `multi.corr` wraps these into an exported function useable on the list output of the `jointprobs` function.

"..\R\ExchMultinomial.R" 10c≡

```
#'Extract correlation coefficients from joint probability arrays
#'
```

```

#'Calculates the within- and between-outcome correlation coefficients for exchangeable correlated
#'multinomial data based on joint probability estimates calculated by the \code{\link{jointprobs}}
#'function. These determine the variance inflation due the cluster structure.
#'
#'If \eqn{R_i} and \eqn{R_j} is the number of events of type \eqn{i} and \eqn{j}, respectively, in a clus
#'size \eqn{n}, then
#'\deqn{Var(R_i)= n p_i (1-p_i)(1 + (n-1)\phi_{ii})}
#'\deqn{Cov(R_i,R_j)= -n p_i p_j (1 + (n-1)\phi_{ij})}
#'where \eqn{p_i} and \eqn{p_j} are the marginal event probabilities and \eqn{\phi_{ij}} are the correlat
#'coefficients computed by \code{multi.corr}.
#@param jp the output of \code{\link{jointprobs}} - a list of joint probability arrays by treatment
#@param type one of c("averaged","cluster","mc") - the type of joint probability. By default,
#'the \code{type} attribute of \code{jp} is used.
#@return a list of estimated correlation matrices by treatment group. If cluster-size specific
#'estimates were requested (\code{(type="cluster")}), then each list elements are a list of
#'these matrices for each cluster size.
#@export
#@seealso \code{\link{jointprobs}} for calculating the joint probability arrays
#@examples
#data(dehp)
#tau <- jointprobs(dehp, type="averaged")
#multi.corr(tau)
#'

multi.corr <- function(jp, type=attr(jp, "type")){
  type <- match.arg(type, c("averaged","cluster","mc"))

  if (type=="cluster") {
    K <- length(dim(jp[[1]])) - 1
    resmat <- lapply(jp, function(x)apply(x, 1, corr.from.tau))
    res <- lapply(resmat, function(x){
      lapply(1:ncol(x), function(idx)matrix(x[,idx], nrow=K)))
    }) else {
      res <- lapply(jp, corr.from.tau)
    }
    res
  }
}

```

File defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.

Defines: multi.corr 2.

Uses: corr.from.tau 10a, tau 2.

**3.3. Testing marginal compatibility.** The `mc.test.chisq` function implements a generalization of the Cochran-Armitage trend test for correlated multinomial data to test for marginal compatibility. Note that it only tests that the marginal probability of response  $p_i$  does not depend on the cluster size for any category.

First, we define the test statistic for one group, and then add the resulting  $\chi_K^2$ -distributed test statistics over the  $G$  groups for an overall  $GK$  degree of freedom test.

As above, let  $(\mathbf{r}_i, n_i)$ ,  $i = 1, \dots, N$  denote the observed data for a given dose level, where  $i$  iterates through the clusters,  $n_i$  is the cluster size and  $\mathbf{r}_i = (r_{i1}, \dots, r_{iK})$  is the observed number of responses of each type. Define the raw trend statistic for response  $j$  as

$$X_j = \sum_{i=1}^N r_{ij}(c_{n_i} - \bar{c}), \quad j = 1, \dots, K, \quad (11)$$

where  $c_n$  are the scores for the Cochran-Armitage test usually chosen as  $c_n = n - (M + 1)/2$ , and  $\bar{c}_g = (\sum_{i=1}^N n_i c_{n_i}) / (\sum_{i=1}^N n_i) = \sum_{n=1}^M M_n n c_n / \sum_{n=1}^M n M_n$  is the weighted average of the scores ( $M_n$  is the number of clusters of size  $n$ ).

The covariance of two of these test statistics is

$$\sigma_{jk} = \text{Cov}(X_j, X_k) = \begin{cases} \sum_{i=1}^N (c_{n_i} - \bar{c})^2 n_i p_{j|n} (1 - p_{j|n}) [1 + (n_i - 1) \phi_{jj|n_i}], & j = k; \\ - \sum_{i=1}^N (c_{n_i} - \bar{c})^2 n_i p_{j|n} p_{k|n} [1 + (n_i - 1) \phi_{jk|n_i}], & j \neq k, \end{cases} \quad (12)$$

where  $p_{j|n} = \tau_{\mathbf{d}_j|n}$  is the probability of event type  $O_j$  in clusters of size  $n$ . Under the null hypothesis of marginal compatibility, the dependence of  $p_{j|n}$  and  $\phi_{jk|n}$  on  $n$  can be removed:

$$\sigma_{jk} = \begin{cases} p_j(1 - p_j) \sum_{i=1}^N (c_{n_i} - \bar{c})^2 n_i [1 + (n_i - 1) \phi_{jj}] = p_j(1 - p_j) \sum_{n=1}^M n M_n (c_n - \bar{c})^2 [1 + (n - 1) \phi_{jj}], & j = k; \\ -p_j p_k \sum_{i=1}^N (c_{n_i} - \bar{c})^2 n_i [1 + (n_i - 1) \phi_{jk}] = -p_j p_k \sum_{n=1}^M n M_n (c_n - \bar{c})^2 [1 + (n - 1) \phi_{jk}], & j \neq k, \end{cases} \quad (13)$$

The combined test statistic for the given dose group  $g$  is

$$T_g^2 = X_g' \Sigma_g^{-1} X_g \sim \chi_K^2 \text{ under } H_0, \quad (14)$$

where  $X_g' = (X_{g1}, \dots, X_{gK})$ , and  $\Sigma_g = (\sigma_{gjk})_{K \times K}$  is its variance-covariance matrix defined by (11) and (13). The unknown values of  $p_j$  and  $\phi_{jk}$  will be replaced by their estimates under marginal compatibility  $\hat{\tau}_{g\mathbf{d}_j}$  and  $\hat{\phi}_{gjk}$ .

The final test statistic is an independent combination of the statistics for each dose group:

$$T^2 = \sum_{g=1}^G T_g^2 \sim \chi_{GK}^2 \text{ under } H_0. \quad (15)$$

"..\R\ExchMultinomial.R" 12≡

```
#'@rdname mc.test.chisq
#'@method mc.test.chisq CMDData
#'@export
#'@importFrom stats weighted.mean pchisq
#'@examples
#'#
#'#data(dehp)
#'#mc.test.chisq(dehp)
#'#

mc.test.chisq.CMDData <- function(object, ...){
  cmdata <- object[object$Freq > 0, ]
  K <- attr(object, "ncat")-1
  nrespvars <- paste("NResp", 1:K, sep=".")

  get.T <- function(x){
    x$Trt <- factor(x$Trt) #remove unused levels
    tt <- jointprobs(x, type="mc")[[1]] #only one treatment group
    p <- p.from.tau(tt)
    phi <- corr.from.tau(tt)
    xx <- x[rep(1:nrow(x), x$Freq),]
    xx$Freq <- 1

    M <- max(x$ClusterSize)
    Mn <- table(factor(xx$ClusterSize, levels=1:M))

    scores <- (1:M) - (M+1)/2

    Rmat <- data.matrix(xx[,nrespvars,drop=FALSE])
```

```

nvec <- xx$ClusterSize
cvec <- scores[nvec]
c.bar <- weighted.mean(cvec, w=nvec)
cvec <- cvec - c.bar

X <- t(Rmat) %*% cvec
Sigma <- diag(p, nrow=length(p)) - outer(p,p) #multinomial vcov
od.matrix <- matrix(0, nrow=K, ncol=K) #over-dispersion matrix
for (n in 1:M){
  od.matrix <- od.matrix + n * Mn[n] * (scores[n]-c.bar)^2 * (1+(n-1)*phi)
}
Sigma <- Sigma * od.matrix

Tstat <- t(X) %*% solve(Sigma) %*% X
Tstat
}

chis <- by(cmdata, cmdata$Trt, get.T)
chis <- chis[1:length(chis)]
chi.list <- list(chi.sq=chis, p=pchisq(chis, df=K, lower.tail=FALSE))
overall.chi <- sum(chis)
overall.df <- length(chis) * K
list(overall.chi=overall.chi, overall.p=pchisq(overall.chi, df=overall.df, lower.tail=FALSE),
     individual=chi.list)
}
◇

```

File defined by [1](#), [2](#), [4b](#), [5](#), [6g](#), [8b](#), [9ab](#), [10abc](#), [12](#), [14b](#).

Defines: `mc.test.chisq` Never used.

Uses: `corr.from.tau` [10a](#), `tau` [2](#).

#### 4. SUPPORT FUNCTIONS

The **Simplex with sums** macro creates a matrix (parameter 1) with rows containing the coordinates of an integer lattice within a  $d$ -dimensional (parameter 3) simplex of size  $n$  (parameter 2). That is all  $d$ -dimensional vectors with non-negative elements with sum not exceeding  $n$  are listed. The actual sums are saved in a vector (parameter 4). Since this is a parametrized macro, it will expand to code, so no actual function calls will be made by the program. This should reduce copying of the potentially large matrices.

*Simplex with sums 13*  $\equiv$

```

@1 <- hcube(rep(@2+1, @3))-1
@4 <- rowSums(@1)
@1 <- @1[@4 <= @2, ,drop=FALSE] #remove impossible indices
@4 <- @4[@4 <= @2]

```

◇  
Fragment referenced in [3bk](#), [5](#), [6g](#), [8b](#).

The **mChoose** function calculates the multinomial coefficient  $\binom{n}{r_1, \dots, r_K}$ . The lower part of the expression is passed as a vector. If its values add up to less than  $n$ , an additional value is added. The function is not vectorized.

$\langle$  Define function for multinomial coefficient 14a  $\rangle \equiv$

```
mChoose <- function(n, rvec, log=FALSE){
  rlast <- n - sum(rvec)
  rveclong <- c(rvec, rlast)
  if (any(rveclong < 0)) return(0)

  res <- lgamma(n + 1) - sum(lgamma(rveclong + 1))
  if (log) res else exp(res)
}
```

$\diamond$   
Fragment referenced in 2.

Defines: mChoose 14b.

The mChooseTable function creates a lookup table of the multinomial coefficients with the number of categories  $k$  and  $n = \max \sum r_i$  given. The results is a  $k$ -dimensional array, with element  $[r1, \dots, rK]$  corresponding to  $\binom{\sum (r_i - 1)}{r_1 - 1, \dots, r_k - 1}$  (because the array is 1-indexed, while  $r_i$  can go from 0). The values in the array with coordinate sum exceeding  $n$  are missing.

"..\R\ExchMultinomial.R" 14b $\equiv$

```
#'@rdname CorrBin-internal
mChooseTable <- function(n, k, log=FALSE){
  res <- array(NA, dim=rep.int(n+1, k))
  dimnames(res) <- rep.int(list(0:n), k)

  idx <- hcube(rep.int(n+1, k)) - 1
  idx <- idx[rowSums(idx) <= n, ,drop=FALSE]
  for (i in 1:nrow(idx)){
    r <- idx[i, ]
    res[rbind(r)+1] <- mChoose(n=sum(r), rvec=r, log=log)
  }
  res
}
```

$\diamond$   
File defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.

Uses: mChoose 14a.

## 5. FILES

"..\R\ExchMultinomial.R" Defined by 1, 2, 4b, 5, 6g, 8b, 9ab, 10abc, 12, 14b.

## 6. MACROS

$\langle$  Calculate averaged thetas 3b  $\rangle$  Referenced in 2.

$\langle$  Calculate cluster-specific taus 3k  $\rangle$  Referenced in 2.

$\langle$  Calculate MC taus 4a  $\rangle$  Referenced in 2.

$\langle$  Calculate res.new - the value of res.trt for next iteration 8a  $\rangle$  Referenced in 7.

$\langle$  Calculate values for cluster size cs given values for size cs+1 6f  $\rangle$  Referenced in 5.

$\langle$  Define function for multinomial coefficient 14a  $\rangle$  Referenced in 2.

$\langle$  Extract info from cmdata into variables 3a  $\rangle$  Referenced in 2, 6g.

$\langle$  Initialize for cluster size M 6e  $\rangle$  Referenced in 5.

$\langle$  MC estimates for given dose group 7  $\rangle$  Referenced in 6g.

$\langle$  Simplex with sums 13  $\rangle$  Referenced in 3bk, 5, 6g, 8b.

## 7. IDENTIFIERS

corr.from.pi: 10b.

corr.from.tau: 10a, 10bc, 12.

Marginals: [4b](#), [5](#), [7](#).  
mc.est.CMData: [4b](#).  
mc.estraw: [4ab](#), [6g](#).  
mc.test.chisq: [12](#).  
mChoose: [14a](#), [14b](#).  
multi.corr: [2](#), [10c](#).  
tau: [2](#), [4a](#), [8b](#), [9ab](#), [10abc](#), [12](#).  
tau.from.pi: [4a](#), [8b](#), [10b](#).  
uniprbs: [2](#), [9b](#).