CORRELATED MULTINOMIAL DATA

ANIKO SZABO

Abstract. We define a class for describing data from toxicology experiments with multinomial outcomes and implement fitting of a variety of existing models and trend tests.

1. Defining `CMData` – a class for Clustered Multinomial **Data**

We start with defining an S3 class describing data from toxicology experiments with multinomial outcomes of type $1, 2, \dots, K+1$. Here K denotes the "degrees of freedom" of the outcome. $K = 1$ corresponds to binary data. The class is a data frame with the following columns:

Trt: a factor defining (treatment) groups

ClusterSize: an integer-valued variable defining the cluster size

NResp.1–NResp.K+1: integer-valued variables defining the number of responses of type $1, 2, \dots, K+1$

Freq: an integer-valued variable defining frequency for each Trt/ClusterSize/NResp.1/$\cdots$/NResp.K combination

While having all the counts and the clustersize is somewhat redundant (sum of counts = clustersize), this can be helpful for many computations.

`CMData` converts a data frame to a CMData object. `x` is the input data frame; `trt`, `clustersize`, and `freq` could be strings or column indices defining the appropriate variable in `x`. `nresp` should be a vector of variable names or column indices of length K (if `clustersize` is given) or $K+1$ (in which case `clustersize` will be calculated). `freq` can also be NULL, in which case it is assumed that each combination has frequency 1.

"../R/CMData.R" 1≡

```
#’Create a ‘CMdata’ object from a data frame.
#’
#’The \code{CMData} function creates an object of class \dfn{CMData} that is
#’used in further analyses. It identifies the variables that define treatment
#’group, clustersize and the number of responses for each outcome type.
#’
#’@export
#’@importFrom stats aggregate
#’@param x a data frame with one row representing a cluster or potentially a
#’set of clusters of the same size and number of responses for each outcome
#’@param trt the name of the variable that defines treatment group
#’@param nresp either a character vector with the names or a numeric vector with indices
#’of the variables that define the number of responses in
#’the cluster for each outcome type. If \code{clustersize} is \code{NULL}, then it will be
#’calculated assuming that the \code{nresp} vector contains all the possible outcomes.
#’If \code{clustersize} is given, then an additional category is created for the excess cluster members.
#’@param clustersize the name or index of the variable that defines cluster size, or \code{NULL}. If \cod
#’its value will be calculated by adding the counts from the \code{nresp} variables. If defined,
#’an additional response type will be created for the excess cluster members.
#’@param freq the name or index of the variable that defines the number of clusters
#’represented by the data row. If \code{NULL}, then each row is assumed to
#’correspond to one cluster.
#’@return A data frame with each row representing all the clusters with the
```

Date: August 28, 2024.

```
#’same trt/size/number of responses, and standardized variable names:
#’@return \item{Trt}{factor, the treatment group}
#’@return \item{ClusterSize}{numeric, the cluster size}
#’@return \item{NResp.1--NResp.K+1}{numeric, the number of responses for each of the K+1 outcome types}
#’@return \item{Freq}{numeric, number of clusters with the same values}
#’@author Aniko Szabo
#’@seealso \code{\link{read.CMData}} for creating a \code{CMData} object
#’directly from a file.
#’@keywords classes manip
#’@examples
#’
#’data(dehp)
#’dehp <- CMData(dehp, trt="Trt", nresp=c("NResp.1","NResp.2","NResp.3"))
#’str(dehp)
#’
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `read.CMData` 4a.

"../R/CMData.R" 2≡

```
CMData <- function(x, trt, nresp, clustersize=NULL, freq=NULL){
  if (!is.data.frame(x)) stop("x has to be a data frame")
  nms <- names(x)
  K <- if (is.null(clustersize)) length(nresp)-1 else length(nresp)

  process.var <- function(var){
    if (is.character(var)){
       if (var %in% nms) res <- x[[var]]
       else stop(paste("Variable ’", var, "’ not found"))
    }
    else {
      if (is.numeric(var)){
         if (var %in% seq(along=nms)) res <- x[[var]]
         else stop(paste("Column", var, " not found"))
      }
      else stop(paste("Invalid variable specification:",var))
    }
  }
  trtvar <- factor(process.var(trt))
  nrespvar <- sapply(nresp, process.var)
  if (is.null(freq)) freqvar <- rep(1, nrow(x))
  else freqvar <- process.var(freq)

  if (!is.null(clustersize)){
     csvar <- process.var(clustersize) # read cluster sizes
     nrespvar <- cbind(nrespvar, csvar - rowSums(nrespvar))  # calculate last category
  }
  else {
    csvar <- rowSums(nrespvar) #calculate sample sizes
  }
  colnames(nrespvar) <- 1:(K+1)


  d <- data.frame(Trt=trtvar, ClusterSize=csvar, NResp=nrespvar, Freq=freqvar)
  nrespnames <- grep("NResp", names(d), value=TRUE)
  d <- aggregate(d$Freq, d[,c("Trt", "ClusterSize", nrespnames)], sum)
  names(d)[length(names(d))] <- "Freq"
  attr(d, "ncat") <- K+1
  class(d) <- c("CMData", "data.frame")
  d}
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `CMdata.data.frame` Never used.

The `read.CMData` function reads in clustered multinomial data from a tab-delimited text file. There are two basic data format options: either the counts of responses of all categories are given (and the cluster size is the sum of these counts), or the total cluster size is given with the counts of all but one category. The first column should always give the treatment group, then either the counts for each category (first option, chosen by setting `with.clustersize = FALSE`), or the size of the cluster followed by the counts for all but one category (second option, chosen by setting `with.clustersize = TRUE`). Optionally, a last column could give the number of times the given combination occurs in the data.

"../R/CMData.R" 3≡

```
#'Read data from external file into a CMData object
#'
#'A convenience function to read data from specially structured file directly
#'into a \code{CMData} object. There are two basic data format options:  either the counts of responses o
#'cluster size is the sum of these counts), or  the total cluster size is given with the counts of all bu
#'The first column should always give the treatment group, then either the counts for each category (firs
#'\code{with.clustersize = FALSE}), or the size of the cluster followed by the counts for all but one cat
#'chosen by setting \code{with.clustersize = TRUE}). Optionally, a last column could
#'give the number of times the given combination occurs in the data.
#'
#'@export
#'@importFrom utils read.table
#'@param file name of file with data. The data in the file should be structured as described above.
#'@param with.clustersize logical indicator of whether a cluster size variable is present
#'in the file
#'@param with.freq logical indicator of whether a frequency variable is present
#'in the file
#'@param ... additional arguments passed to \code{\link[utils]{read.table}}
#'@return a \code{CMData} object
#'@author Aniko Szabo
#'@seealso \code{\link{CMData}}
#'@keywords IO file
#'
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.

"../R/CMData.R" 4a≡

```
read.CMData <- function(file, with.clustersize=TRUE, with.freq=TRUE, ...){
  d <- read.table(file, ...)
  K <- ncol(d) - with.freq - 2  #subtracting Trt & either ClusterSize or last category column

  if (with.clustersize){
    d2 <- CMData(d, trt=1, clustersize=2, nresp=3:(K+2), freq=if (with.freq) "Freq" else NULL)
  }
  else {
    d2 <- CMData(d, trt=1, nresp=2:(K+2), freq=if (with.freq) "Freq" else NULL)
  }
  d2}
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `read.CMData` 1.

The `[.CMData` function defines subsetting of `CMData` objects. If the subsetting is only affecting the rows, then the appropriate attributes are preserved, and the unused levels of `Trt` are dropped. Otherwise the returned object does not have a `CMData` class anymore.

"../R/CMData.R" 4b≡

```
#'@rdname Extract
#'@export
#'@examples
#'
#'data(dehp)
#'str(dehp[1:5,])
#'str(dehp[1:5, 2:4])

"[.CMData" <- function(x, i, j, drop){
  res <- NextMethod("[")
  if (NCOL(res) == ncol(x)){
    res <- "[.data.frame"(x, i, )
    if (is.factor(res$Trt)) res$Trt <- droplevels(res$Trt)
    attr(res, "ncat") <- attr(x, "ncat")
    res
  }
  else {
    class(res) <- setdiff(class(res), "CMData")
  }
  res
}
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `[.CMData` Never used.

`unwrap.CMData` is a utility function that reformats a CMData object so that each row is one observation (instead of one cluster). A new 'ID' variable is added to indicate clusters.

"../R/CMData.R" 4c≡

```
#'@rdname unwrap
#'@method unwrap CMData
#'@export
#'@importFrom stats reshape
#'@return For \code{unwrap.CMData}: a data frame with one row for each cluster element (having a multinom
#'outcome) with the following standardized column names
```

```
#'@return \item{Trt}{factor, the treatment group}
#'@return \item{ClusterSize}{numeric, the cluster size}
#'@return \item{ID}{factor, each level representing a different cluster}
#'@return \item{Resp}{numeric with integer values giving the response type of the cluster
#'element}
#'@examples
#'
#'data(dehp)
#'dehp.long <- unwrap(dehp)
#'head(dehp.long)
#'
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `unwrap.CMData` 5a.

`"../R/CMData.R"` 5a≡

```
unwrap.CMData <- function(object,...){
  #unwrap Freq variable
  freqs <- rep(1:nrow(object), object$Freq)
  cm1 <- object[freqs,]
  cm1$Freq <- NULL

  #create ID
  cm1$ID <- factor(1:nrow(cm1))

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

  #reshape to have one row per category within cluster
  cm2 <- reshape(cm1, direction="long", varying=list(nrespvars), v.names="Count",
                 idvar="ID", timevar="Resp", times=1:ncat)

  #unwrap categories
  counts <- rep(1:nrow(cm2), cm2$Count)
  res <- cm2[counts,]
  res$Count <- NULL
  class(res) <- "data.frame"
  res <- res[order(res$ID),c("Trt","ID","ClusterSize","Resp")]
  rownames(res) <- NULL

  res
  }
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `unwrap.CMData` 4c.

2. Generating CMData objects

`"../R/CMData.R"` 5b≡

```
#' Generate a random CMData object
#'
#' Generates random exchangeably correlated multinomial data based on a parametric
#' distribution or using resampling. The Dirichlet-Multinomial, Logistic-Normal multinomial,
#' and discrete mixture multinomial parametric distributions are implemented.
#' All observations will be assigned to the same treatment group, and there is no
#' guarantee of a specific order of the observations in the output.
#'
#'@export
#'@param n number of independent clusters to generate
#'@param ncat number of response categories
#'@param clustersize.gen either an integer vector specifying the sizes of the clusters,
#' which will be recycled to achieve the target number of clusters \code{n}; or
#' a function with one parameter that returns an integer vector of cluster sizes when
#' the target number of clusters n is passed to it as a parameter
#'@param distribution a list with two components: "multinom.gen" and "param" that specifies
#' the generation process for each cluster. The "multinom.gen" component should be a function
#' of three parameters: number of clusters, vector of cluster sizes, and parameter list, that
#' a matrix of response counts where each row is a cluster and each column is the number of
#' responses of a given type. The "param" component should specify the list of parameters
#' needed by the multinom.gen function.
#'
#'@return a \code{CMData} object with randomly generated number of responses with
#' sample sizes specified in the call
#'
#'@author Aniko Szabo
#'@seealso \code{\link{CMData}} for details about \code{CMData} objects; \code{\link{multinom.gen}} for b
#'@keywords distribution
#'@examples
#'
#'# Resample from the dehp dataset
#' data(dehp)
#' ran.dehp <- ran.CMData(20, 3, 10, list(multinom.gen=mg.Resample, param=list(data=dehp)))
#'
#'# Dirichlet-Multinomial distribution with two treatment groups and random cluster sizes
#' binom.cs <- function(n){rbinom(n, p=0.3, size=10)+1}
#' dm1 <- ran.CMData(15, 4, binom.cs,
#'                    list(multinom.gen=mg.DirMult, param=list(shape=c(2,3,2,1))))
#' dm2 <- ran.CMData(15, 4, binom.cs,
#'                    list(multinom.gen=mg.DirMult, param=list(shape=c(1,1,4,1))))
#' ran.dm <- rbind(dm1, dm2)
#'
#'# Logit-Normal multinomial distribution
#' ran.ln <- ran.CMData(13, 3, 3,
#'                       list(multinom.gen=mg.LogitNorm,
#'                            param=list(mu=c(-1, 1), sigma=matrix(c(1,0.8,0.8,2), nr=2))))
#'
#'# Mixture of two multinomial distributions
#' unif.cs <- function(n){sample(5:9, size=n, replace=TRUE)}
#' ran.mm <- ran.CMData(6, 3, unif.cs,
#'                       list(multinom.gen=mg.MixMult,
#'                            param=list(q=c(0.8,0.2), m=rbind(c(-1,0), c(0,2)))))
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: mg.DirMult 8b, mg.LogitNorm 9a, mg.MixMult 9c, mg.Resample 7c, ran.CMData 6.

"../R/CMData.R" 6≡

```
ran.CMData <- function(n, ncat, clustersize.gen, distribution){

  respnames <- paste("NResp", 1:ncat, sep=".")
  if (is.numeric(clustersize.gen)){
    cs <- rep(clustersize.gen, length.out=n)
  }
  else if (is.function(clustersize.gen)){
    cs <- clustersize.gen(n)
  }
  else stop("clustersize.gen should be either numeric or a function.")
  counts <- distribution$multinom.gen(n, cs, distribution$param)
  colnames(counts) <- respnames

  dd <- data.frame(counts, ClusterSize=cs, Trt=0)
  res <- CMData(dd, trt="Trt", nresp=respnames)

  return(res)
}
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `ran.CMData` 5b, 7a.

`"../R/CMData.R"` 7a≡

```
#' Functions for generating multinomial outcomes
#'
#'These are built-in functions to be used by \code{\link{ran.CMData}} for generating
#' random multinomial data.
#'@name multinom.gen
#'@param n number of independent clusters to generate
#'@param clustersizes an integer vector specifying the sizes of the clusters
#'@param param a list of parameters for each specific generator
NULL
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `ran.CMData` 6.

2.1. **Resampling.**

`"../R/CMData.R"` 7b≡

```
#'@details For \bold{mg.Resample}: the \code{param} list should be \code{list(param=...)}, in which
#' the CMData object to be resampled is passed.
#'@export
#'@importFrom stats rmultinom
#'@rdname multinom.gen
#'
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `mg.Resample` 7c.

`"../R/CMData.R"` 7c≡

```
mg.Resample <- function(n, clustersizes, param){
  dd <- param$data
  nc <- attr(dd, "ncat")
  nrespvars <- paste("NResp", 1:nc, sep=".")
  datamat <- data.matrix(dd[, nrespvars])
  res <- matrix(NA, nrow=n, ncol=nc)

  #sample clusters
```

```
    idx <- sample(1:nrow(dd), size=n, replace=TRUE, prob=dd$Freq)
    #sample observations from clusters for target clustersize
    for (i in 1:n){
      res[i,] <- rmultinom(1, size=clustersizes[i], prob=datamat[idx[i],])
    }
    res
  }
  ◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `mg.Resample` 5b, 7b.

2.2. Dirichlet-Multinomial distribution.

`"../R/CMData.R"` 8a≡

```
#'@details For \bold{mg.DirMult}: the \code{param} list should be \code{list(shape=...)}, in which
#' the parameter vector of the Dirichlet distribution is passed
#'  (see \link[dirmult]{rdirichlet}).
#'
#'@importFrom dirmult rdirichlet
#'@export
#'@rdname multinom.gen
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `mg.DirMult` 8b.

`"../R/CMData.R"` 8b≡

```
mg.DirMult <- function(n, clustersizes, param){
  p <- rdirichlet(n, alpha=param$shape)

  ssize <- rep(clustersizes, length.out=n)
  x <- sapply(1:n, function(i)rmultinom(1, size=ssize[i], prob=p[i,]))

  return(t(x))
}
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `mg.DirMult` 5b, 8a.

2.3. Logit-Normal multinomial distribution.

`"../R/CMData.R"` 8c≡

```
#'@details For \bold{mg.LogitNorm}: the \code{param} list should be \code{list(mu=...,sigma=...)},
#'in which the mean vector and covariance matrix of the underlying Normal distribution
#'are passed. If \code{sigma} is NULL (or missing), then an identity matrix is assumed.
#'They should have \emph{K-1} dimensions for a \emph{K}-variate multinomial.
#'
#'@export
#'@importFrom mvtnorm rmvnorm
#'@rdname multinom.gen
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `mg.LogitNorm` 9a.

"../R/CMData.R" 9a≡

```
mg.LogitNorm <- function(n, clustersizes, param){
  if (is.null(param$sigma)) sigma <- diag(rep(1,length(param$mu)))
  z <- rmvnorm(n, mean=param$mu, sigma=param$sigma)
  z <- cbind(z, 0)
  p <- exp(z) / rowSums(exp(z))

  ssize <- rep(clustersizes, length.out=n)
  x <- sapply(1:n, function(i)rmultinom(1, size=ssize[i], prob=p[i,]))

  return(t(x))
}
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `mg.LogitNorm` 5b, 8c.

2.4. Discrete mixture of multinomials distribution.

"../R/CMData.R" 9b≡

```
#'@details For \bold{mg.MixMult}: the \code{param} list should be \code{list(q=...,m=...)},
#'in which the vector of mixture probabilities \code{q} and the matrix \code{m}
#' of logit-transformed means of each component are passed.
#'For a \emph{K}-variate multinomial, the matrix{m} should have \emph{K-1} columns
#' and \code{length(q)} rows.
#'
#'@export
#'@rdname multinom.gen
◇
```

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Uses: `mg.MixMult` 9c.

"../R/CMData.R" 9c≡

```
mg.MixMult <- function(n, clustersizes, param){

  class <- sample.int(length(param$q), size=n, prob=param$q, replace=TRUE)

  z <- cbind(param$m, 0)
  p <- apply(z, 1, function(zz){
    if (any(zz==Inf)){
      res <- rep(0, length(zz))
      res[zz==Inf] <- 1
      res
    }
    else {
      exp(zz)/sum(exp(zz))
    }
  })
  p <- t(p)  # apply transposes


  ssize <- rep(clustersizes, length.out=n)
  x <- sapply(1:n, function(i)rmultinom(1, size=ssize[i], prob=p[class[i],]))

  return(t(x))
}
```

◇

File defined by 1, 2, 3, 4abc, 5ab, 6, 7abc, 8abc, 9abc.
Defines: `mg.MixMult` 5b, 9b.