# TEST FOR TREND WITH A MULTINOMIAL OUTCOME

ANIKO SZABO

## 1. Introduction

Consider a study in which a multinomial outcome with $K$ possible unordered values is measured in subjects belonging to one of $G$ ordered groups. The size of each group, $n_{i\cdot}$, is defined by the study design, and will be treated as fixed. Let $\mathbf{p}_i = (p_{i1}, \ldots, p_{iK})^{\mathrm{T}}$ denote the probabilities of the multinomial outcomes in the $i$th group. The hypothesis of interest is to evaluate the homogeneity of these probabilities across the groups with a targeted alternative of a trend in at least one of the categories. Formally, we consider testing $H_0 = \bigcap_{j=1}^{K} H_{0j}$ versus $H_1 = \bigcup_{j=1}^{K} H_{1j}$, where

$$\begin{aligned} &H_{0j} : p_{1j} = \cdots = p_{Gj} \\ &H_{1j} : p_{1j} \leq \cdots \leq p_{Gj} \text{ or } p_{1j} \geq \cdots \geq p_{Gj} \text{ with at least one inequality} \end{aligned} \tag{1}$$

The test is based on the following result:

**Theorem 1.** *Let $\mathcal{J} \subset \{1, \ldots, K\}$, then under $H_{0\mathcal{J}} = \bigcap_{j\in\mathcal{J}} H_{0j}$ as $N \to \infty$*

$$W_{\mathcal{J}} = \sum_{j\in\mathcal{J}} (1 - p_{\cdot j}) T_j^2 + \Big(\sum_{j\in\mathcal{J}} p_{\cdot j}\Big) T_{\mathcal{J}}^2 \xrightarrow{d} \chi_d^2, \tag{2}$$

*where $d = \min(|\mathcal{J}|, K-1)$, $T_{\mathcal{J}} = \big[\sum_{i=1}^{G} \sum_{j\in\mathcal{J}} n_{ij}(c_i - \bar{c})\big] \big/ \sqrt{p_{\cdot\mathcal{J}}(1 - p_{\cdot\mathcal{J}}) s^2}$ denotes the Cochran-Armitage trend test statistic for testing for marginal trend in $p_{i\mathcal{J}} = \sum_{j\in\mathcal{J}} p_{ij}$, $i = 1, \ldots, G$.*

## 2. Implementing the overall test

The main `multiCA.test` function is a generic, with methods for a matrix and formula input.

*Date*: March 25, 2025.

"../R/aaa-generics.R" 2a≡

```
#'Multinomial Cochran-Armitage trend test
#'
#'The \code{multiCA.test} function performs a multinomial generalization of the
#' Cochran-Armitage trend test.
#'
#'
#'@export
#'@param x a two-dimensional matrix of event counts with the outcomes as rows and ordered groups as colum
#'@param \dots other arguments
#'@return a list with two components
#' \item{overall}{an object of class "htest" with the results of the overall test}
#' \item{individual}{a vector with adjusted p-values for individual outcomes}
#'@author Aniko Szabo
#'@references Szabo, A. (2018). Test for Trend With a Multinomial Outcome. The American Statistician, 73(
#'@keywords nonparametric
#'@examples
#'
#'data(stroke)
#'## using formula interface
#'multiCA.test(Type ~ Year, weights=Freq, data=stroke)
#'
#'##using Westfall's multiple testing adjustment
#'multiCA.test(Type ~ Year, weights=Freq, data=stroke, p.adjust.method="Westfall")
#'
#'## using matrix interface and testing only the first 3 outcomes
#'strk.mat <- xtabs(Freq ~ Type + Year, data=stroke)
#'multiCA.test(strk.mat, outcomes=1:3)
#'
#'@name multiCA.test

multiCA.test <- function(x,...) UseMethod("multiCA.test")

◇
```

The actual calculation of the test statitistic, overall and unadjusted individual p-values, and correlation/contrast matrices that will be useful for adjusted p-value calculation, is encapsulated in an internal function that operates on a matrix. No error control is provided here.

"../R/multiCA.R" 2b≡

```
#' @keywords internal
#' @importFrom stats terms xtabs


.multiCA.test <- function(x, scores, outcomes){
  K <- nrow(x)
  full <- length(outcomes) == K  #full test

  nidot <- apply(x, 2, sum)
  n <- sum(nidot)

  cbar <- sum(nidot * scores)/n

  s2 <- sum(nidot * (scores - cbar)^2)
  pdot <- prop.table(rowSums(x))[outcomes]
  nonz <- (pdot > 0)
```

```
  if (!any(nonz)) return(1)

  X <- x[outcomes, ,drop=FALSE] %*% (scores - cbar)

  #individual tests
  Tt <-  X[nonz] / sqrt(pdot[nonz] * (1-pdot[nonz])* s2)
  CAT <- Tt^2
  CAT.p.value <- pchisq(CAT, df=1, lower.tail=FALSE)

  #overall test
  if (full || sum(pdot) >= 1){
    W <- ( sum(X[nonz]^2 / pdot[nonz])) / s2
  } else {
    W <- (sum(X)^2 / (1-sum(pdot)) + sum(X[nonz]^2 / pdot[nonz])) / s2
  }

  df <- length(outcomes) - full
  p.value <- pchisq(W, df=df, lower.tail=FALSE)

  ⟨ Calculate correlation and contrast matrices 8b ⟩

  res <- list(statistic = W, parameter = df, p.value = p.value,
              indiv.statistics = Tt, indiv.p.value = CAT.p.value,
              sigma0 = Sigma0, contrast = C)
  return(res)
}
◇
```

File defined by 2b, 3, 4, 7b, 9a, 10, 15.
Defines: `.multiCA.test` 3, 6c, 10, 11, 13b, 14.

The default method uses a two-dimensional contingency matrix with the outcomes as rows and ordered groups as columns.

`"../R/multiCA.R"` 3≡

```
#'@rdname multiCA.test
#'@method multiCA.test default
#'@param scores non-decreaseing numeric vector of the same length as the number of ordered groups. Defaul
#'@param outcomes integer or character vector defining the set of outcomes (by row index or row name) ove
#'@param p.adjust.method character string defining the correction method for individual outcome p-values.
#'@export
#' @importFrom utils str
#' @importFrom multcomp glht adjusted parm

multiCA.test.default <- function(x, scores=1:ncol(x), outcomes=1:nrow(x),
  p.adjust.method=c("none","closed.set","Holm-Shaffer", "single-step", "Westfall"),...){
  if (!is.matrix(x)) {
    cat(str(x))
    stop("x should be a two-dimensional matrix")
}
  if (length(scores) != ncol(x)) stop("The length of the score vector should equal the number of columns

  testres <- .multiCA.test(x=x, scores=scores, outcomes=outcomes)

  W <- c(W = testres$statistic)
  df <- c(df = testres$parameter)

  p.value <- testres$p.value
```

```
  null.value <- 0
  names(null.value) <- sprintf("slope for outcomes %s", deparse(substitute(outcomes)))

  res <- list(statistic = W, parameter = df, p.value = p.value,
              method="Multinomial Cochran-Armitage trend test",
              alternative="two.sided",
              null.value=null.value,
              data.name = deparse(substitute(x)))
  class(res) <- "htest"

  ⟨ Calculate adjusted p-values 5b ⟩

  return(list(overall = res, individual = indiv.res))
}
◇
```

File defined by 2b, 3, 4, 7b, 9a, 10, 15.
Defines: `multiCA.test.default` Never used.
Uses: `.multiCA.test` 2b.

The formula interface converts data into the appropriate contingency matrix for use with the default method. The code is based on `t.test.formula`.

"../R/multiCA.R" 4≡

```
#'@rdname multiCA.test
#'@method multiCA.test formula
#'@param formula a formula of the form \code{outcome ~ group} where \code{outcome} is a factor representi
#'@param data  an optional matrix or data frame containing the variables in the formula \code{formula}. B
#'@param subset  an optional vector specifying a subset of observations to be used.
#'@param na.action       a function which indicates what should happen when the data contain NAs. Default
#'@param weights an integer-valued variable representing the number of times each \code{outcome} - \code{
#'@export
#' @importFrom stats terms xtabs

multiCA.test.formula <- function(formula, data, subset, na.action,  weights, ...){
    if (missing(formula) || (length(formula) != 3L) || (length(attr(terms(formula[-2L]),
        "term.labels")) != 1L))
        stop("'formula' missing or incorrect")
    m <- match.call(expand.dots = FALSE)
    if (is.matrix(eval(m$data, parent.frame())))
        m$data <- as.data.frame(data)
    m[[1L]] <- quote(stats::model.frame)
    m$... <- NULL
    mf <- eval(m, parent.frame())
    responsevar <- attr(attr(mf, "terms"), "response")
    response <- mf[[responsevar]]
    weightvar <- which(names(mf)=="(weights)")
    w <- if(length(weightvar) > 0)  mf[[weightvar]] else rep(1L, nrow(mf))
    g <- factor(mf[,-c(responsevar, weightvar)])

    tab <- xtabs(w ~ response + g)
    multiCA.test(tab, ...)
}
◇
```

File defined by 2b, 3, 4, 7b, 9a, 10, 15.
Defines: `multiCA.test.formula` Never used.

"../tests/testthat/test_overall.R" 5a≡

```
context("Multinomial CA test")
test_that("Overall test works on stroke data", {
  data(stroke)
  res0 <- multiCA.test(Type ~ Year, weights=Freq, data=stroke, p.adjust="none")
  expect_equivalent(res0$overall$statistic, 40.06580869)

  strk.mat <- xtabs(Freq ~ Type + Year, data=stroke)
  res1 <- multiCA.test(strk.mat, p.adjust="none")
  expect_equal(res0$overall[c("statistic","paramater", "p.value")],
               res1$overall[c("statistic","paramater", "p.value")])
  expect_equivalent(res0$individual, res1$individual)

  res2 <- multiCA.test(strk.mat, outcomes=1:5, p.adjust="none")
  expect_equal(res1$overall[c("statistic","paramater", "p.value")],
               res2$overall[c("statistic","paramater", "p.value")])
  expect_equivalent(res1$individual, res2$individual)
})
```

◇

File defined by 5a, 6b, 7a.

## 3. Multiple testing adjusted inference for individual outcomes

⟨ *Calculate adjusted p-values* 5b ⟩ ≡

```
if (missing(p.adjust.method)){
  if (length(outcomes)<=3) p.adjust.method <- "closed.set"
  else p.adjust.method <- "Holm-Shaffer"
} else {
  p.adjust.method <- match.arg(p.adjust.method)
}

full.set <- (length(outcomes) == nrow(x))
if (p.adjust.method=="none") {
  indiv.res <- testres$indiv.p.value
} else if (p.adjust.method=="closed.set") {
  ⟨ Closed set adjustment 6c ⟩
} else if (p.adjust.method=="Holm-Shaffer") {
  ⟨ Holm-Shaffer adjustment 6a ⟩
} else if (p.adjust.method %in% c("single-step", "Westfall")) {
  ⟨ glht adjustment 8a ⟩
}
attr(indiv.res, "method") <- p.adjust.method
```

◇

Fragment referenced in 3.

3.1. **Holm-Shaffer approach.** Shaffer's modification of Holm's adjustment involves multiplying the ordered p-values by $t_s$, the maximum number of possibly true hypotheses, given that at least $s-1$ hypotheses are false. In our case the logical restriction means that if there is at least one false null hypothesis, then no

more than $K-2$ null hypotheses could be true. So

$$p^{HS}_{(j)} = \max_{s \le j}(\min(t_s p_{(s)}, 1))$$

$$\text{where } t_s = \begin{cases} K-s+1, & s \neq 2 \\ K-2, & s = 2 \end{cases}$$

⟨ *Holm-Shaffer adjustment* 6a ⟩ ≡

```
s <- seq_along(testres$indiv.p.value)
if (full.set) s[2] <- 3
o <- order(testres$indiv.p.value)
ro <- order(o)
indiv.res <- pmin(1, cummax((length(outcomes) - s + 1L) * testres$indiv.p.value[o]))[ro]
```

◇

Fragment referenced in 5b.

"../tests/testthat/test_overall.R" 6b≡

```
test_that("Holm-Shaffer consistent with Holm", {
  data(stroke)
  res0 <- multiCA.test(Type ~ Year, weights=Freq, data=stroke, p.adjust="none")
  expect_equal(attr(res0$individual, "method"), "none")

  res1 <- multiCA.test(Type ~ Year, weights=Freq, data=stroke, p.adjust="Holm-Shaffer")
  expect_equal(attr(res1$individual, "method"), "Holm-Shaffer")
  expect_equivalent(sort(p.adjust(res0$individual, method="holm"))[-2],
                    sort(res1$individual)[-2])

  res0a <- multiCA.test(Type ~ Year, weights=Freq, data=stroke, p.adjust="none",
                        outcomes=1:4)
  res1a <- multiCA.test(Type ~ Year, weights=Freq, data=stroke, p.adjust="Holm-Shaffer",
                        outcomes=1:4)
  expect_equivalent(sort(p.adjust(res0a$individual, method="holm")),
                    sort(res1a$individual))
})
```

◇

File defined by 5a, 6b, 7a.

3.2. **Closed set adjustment.** In a closed testing procedure an elementary hypothesis $H_{0j}$ is rejected if and only if all composite hypotheses $H_{0\mathcal{J}}$, where $j \in \mathcal{J}$ are rejected. The process can be rewritten using adjusted p-values for $H_{0j}, j = 1, \ldots K$:

$$p^*_j = \max_{\mathcal{J}: j \in \mathcal{J}} p(\mathcal{J}), \tag{3}$$

where $p(\mathcal{J}) = P(W_j \ge \chi^2_{|\mathcal{J}|})$ is the unadjusted p-value for testing $H_{0\mathcal{J}}$. From the logical constraints sets $\mathcal{J}$ of cardinality $K-1$ do not need to be considered.

⟨ *Closed set adjustment* 6c ⟩ ≡

```
mytest <- function(hypotheses){
  .multiCA.test(x, scores, hypotheses)$p.value
}
indiv.res <- .p.adjust.closed(mytest, outcomes, remove=full.set)
```

◇

Fragment referenced in 5b.
Uses: `.multiCA.test` 2b.

"../tests/testthat/test_overall.R" 7a≡

```
test_that("Closed set works with 3 outcomes", {
  data(stroke)
  strk.mat <- xtabs(Freq ~ Type + Year, data=stroke)
  res0 <- multiCA.test(strk.mat[1:3,], p.adjust="none")
  res1 <- multiCA.test(strk.mat[1:3,], p.adjust="closed.set")
  expect_equivalent(pmax(res0$individual, res0$overall$p.value),
                    res1$individual)
})
```

◇

File defined by 5a, 6b, 7a.

The actual adjustment calculation is based on code from `cherry::closed`, removing the $K-1$ element sets if the full set of hypotheses is being tested.

"../R/multiCA.R" 7b≡

```
#'Internal functions
#'
#' These internal functions perform the closed set p-value adjustment calculation
#' for the multivariate Cochran-Armitage trend test. The logical constraint
#' on the possible number of true null hypotheses is incorporated.
#'
#'
#' @name internal
#' @importFrom bitops bitAnd
#' @keywords internal
.bit2boolean <- function (x, N)
{
  base <- 2^(1:N - 1)
  bitAnd(x, base) != 0
}

#' @param test function that performs the local test. The function should accept a subvector of the hypot
#' @param  hypotheses identifiers of the collection of elementary hypotheses.
#' @param remove logical indicator of whether hypotheses of length N-1 should be removed
#' @param  ...  additional parameters to the 'test' function
#' @return  numeric vector of adjusted p-values for each hypothesis
#' @keywords internal
#' @name internal
.p.adjust.closed <- function (test, hypotheses, remove=FALSE, ...)
{
  N <- length(hypotheses)
  Nmax <- log2(.Machine$integer.max + 1)
  if (N > Nmax)
    stop("no more than ", Nmax, " hypotheses supported in full closed testing.\n Use a shortcut-based tes
  closure <- 1:(2^N - 1)
  base <- 2^(1:N - 1)
  offspring <- function(x) {
    res <- bitAnd(x, closure)
    res[res != 0]
  }
  lengths <- rowSums(sapply(base, function(bs) bitAnd(closure, bs) != 0))

  idx <- sort.list(lengths, decreasing = TRUE)
  closure <- closure[idx]
  lengths <- lengths[idx]
```

```
  if (remove)  closure <- closure[lengths != (N-1)]

  adjusted <- numeric(2^N - 1)
  for (i in closure) {
    if (adjusted[i] < 1) {
      localtest <- test(hypotheses[.bit2boolean(i,N)], ...)
      if (localtest > adjusted[i]) {
        offs <- offspring(i)
        adjusted[offs] <- pmax(adjusted[offs], localtest)
      }
    }
  }

  out <- adjusted[base]
  names(out) <- hypotheses
  return(out)
}
```
◇

File defined by 2b, 3, 4, 7b, 9a, 10, 15.

3.3. **Multivariate normal based adjustment.** The "single-step" and "Westfall" adjustments are based on the (asympotic) multivariate normality of the test statistics, and are implemented in the `multcomp` package.

⟨ *glht adjustment* 8a ⟩ ≡

```
  if (full.set) {
      testparm <- parm(testres$indiv.statistics[-1], testres$sigma0[-1,-1])
  } else {
      testparm <-  parm(testres$indiv.statistics, testres$sigma0)
      }
  g1 <- glht(model = testparm,  linfct = testres$contrast)
  indiv.res <- summary(g1, test=adjusted(type=p.adjust.method,...))$test$pvalues
```
◇

Fragment referenced in 5b.

⟨ *Calculate correlation and contrast matrices* 8b ⟩ ≡

```
# correlation of T
sqrt.or.pdot <- sqrt(pdot[nonz]/(1-pdot[nonz]))
Sigma0 <- -outer(sqrt.or.pdot, sqrt.or.pdot)
diag(Sigma0) <- 1

# contrast matrix
if (full){
  coefs <- sqrt(pdot[nonz] * (1-pdot[nonz]))
  C <- rbind(coefs[-1], diag(K-1))
} else {
  C <- diag(length(nonz))
}
```
◇

Fragment referenced in 2b.

## 4. Power and sample size calculation

The calculation is based on the following result: Let $\nu_i = n_{i\cdot}/N$ denote the proportion of subjects in group $i$.

**Theorem 2.** *Under $H_a$, the asymptotic distribution of $W$ is approximately $\chi^2_{K-1}(\lambda)$ with non-centrality parameter*

$$\lambda = N s_\nu^2 \sum_{j=1}^{K} \frac{\beta_j^2}{p_{\cdot j}}, \tag{4}$$

*where $s_\nu^2 = \sum_{i=1}^{G} \nu_i (c_i - \bar{c})^2 = s^2/N$ and $\beta_j = \left[\sum_{i=1}^{G} \nu_i (p_{ij} - p_{\cdot j})(c_i - \bar{c})\right]/s_\nu^2$ is the slope of $p_{ij}$, $i = 1, \ldots, G$ regressed on $c_i$ with weights $\nu_i$.*

A non-centrality parameter calculation function can be useful by itself. It calculates the non-centrality parameter for a chi-square distribution that achieves the target power at a given significance level.

"../R/multiCA.R" 9a≡

```
#' Non-centrality parameter for chi-square distribution
#'
#' Calculates the non-centrality parameter for a chi-square distribution for a given
#' quantile. This is often needed for sample size calculation for chi-square based tests.
#'
#'@details The function is modeled after the SAS function CNONCT. If \code{p} is larger
#' than the cumulative probability of the central chi-square distribution at \code{x}, then
#' there is no solution and NA is returned.
#'
#'@param x a numeric value at which the distribution was evaluated
#'@param p a numeric value giving the cumulative probability at \code{x}
#'@param df an integer giving the degrees of freedom of the chi-square variable
#'@examples
#' (ncp <- cnonct(qchisq(0.95, df=10), 0.8, df=10))
#' ## check
#' pchisq(qchisq(0.95, df=10), df=10, ncp=ncp)  ## 0.8
#'@export
#'@importFrom stats pchisq uniroot

cnonct <- function(x, p, df){

  if (pchisq(x, df=df) < p) return(NA)

  f <- function(ncp){pchisq(x, df=df, ncp=pmax(0,ncp)) - p}

  res <- uniroot(f, interval=c(0, 100), extendInt="downX", tol=.Machine$double.eps^0.5)
  res$root
}
```

◇

File defined by 2b, 3, 4, 7b, 9a, 10, 15.
Defines: cnonct 9b, 10, 15.

"../tests/testthat/test_power.R" 9b≡

```
context("Power calculations")

test_that("non-centrality calculation works", {
  x <- qchisq(0.75, df=10)
  expect_equal(cnonct(x, df=10, p=0.75), 0)
  expect_equal(cnonct(x, df=10, p=0.9), NA)
  expect_equal(pchisq(x, df=10, ncp=cnonct(x, p=0.6, df=10)), 0.6)
```

```
  })
```

◇

File defined by 9b, 11, 13b, 14, ?, ?.
Uses: cnonct 9a.

"../R/multiCA.R" 10≡

```
#' Power calculations for the multinomial Cochran-Armitage trend test
#'
#' Given the probabilities of outcomes, compute the power of the overall multinomial
#' Cochran-Armitage trend test or determine the sample size to obtain a target power.
#'
#'@details
#' The sample size calculation depends only on \code{p.ave} - the weighted average probability of
#' each outcome, and \code{slopes} - the weighted regression slope of each outcome.
#'
#' The values of these two key inputs can be specified in three ways:
#'
#' 1. directly passing \code{p.ave} and  \code{slopes}, or
#'
#' 2. specifying exactly two of the parameters \code{p.ave}, \code{slopes}, \code{p.start}, and \code{p.e
#' In this case the full matrix of outcome probabilites will be inferred
#' assuming linearity within each outcome.
#'
#' 3. specifying the full matrix of outcome probabilities \code{pmatrix}.
#'
#' The calculation is based on approximating the distribution of the test statistic
#' under the alternative with a non-central chi-squared distribution instead of the correct
#' weighted mixture of multiple non-central chi-squares. This results in bias in the power
#' away from 50% - values above it are somewhat overestimated, while values under it are
#' underestimated.
#'
#' @param N integer, the total sample size of the study. If \code{NULL} then \code{power} needs to be spe
#' @param power target power. If \code{NULL} then \code{N} needs to be specified.
#' @param pmatrix numeric matrix of hypothesized outcome probabilities in each group,  with
#' the outcomes as rows and ordered groups as columns. The columns should add up to 1.
#' @param p.ave numeric vector of average probability of each outcome over the groups
#' weighted by \code{n.prop}.
#' @param p.start,p.end numeric vectors of the probability of each outcome for the
#' first / last ordered group
#' @param slopes numeric vector of the hypothesized slope of each outcome when regressed
#' against the column \code{scores} with weights \code{n.prop}. The values should add up to zero,
#' as the total probability is always 1 and has no trend.
#' @param scores non-decreasing numeric vector of the same length as the number of ordered groups
#' giving the trend test scores. Defaults to linearly increasing values.
#' @param n.prop numeric vector describing relative sample sizes of the ordered groups.
#' Will be normalized to sum to 1. Defaults to equal sample sizes.
#' @param G integer, number of ordered groups
#' @param sig.level significance level
#' @return object of class "power.htest"
#'
#' @examples
#' power.multiCA.test(power=0.8, p.start=c(0.1,0.2,0.3,0.4), p.end=c(0.4, 0.3, 0.2, 0.1),
#'                       G=5, n.prop=c(3,2,1,2,3))
#'
#' ## Power of stroke study with 100 subjects per year and observed trends
#' data(stroke)
#' strk.mat <- xtabs(Freq ~ Type + Year, data=stroke)
```

```
#' power.multiCA.test(N=900, pmatrix=prop.table(strk.mat, margin=2))
#' @seealso \code{\link{power.CA.test}} for simpler (and more precise) power calculation
#' with a binomial outcome
#' @references Szabo, A. (2018). Test for Trend With a Multinomial Outcome. The American Statistician, 73
#' @export
#' @importFrom stats pchisq qchisq weighted.mean


power.multiCA.test <- function(N=NULL, power=NULL, pmatrix=NULL, p.ave=NULL, p.start=NULL,
                               p.end=NULL, slopes=NULL, scores=1:G, n.prop=rep(1, G),
                               G=length(p.ave), sig.level=0.05){
  if (sum(sapply(list(N, power), is.null)) != 1)
        stop("exactly one of 'N',  and 'power' must be NULL")
  if (!is.numeric(sig.level) || any(0 > sig.level | sig.level > 1))
        stop("'sig.level' must be numeric in [0, 1]")
```

⟨ *Calculate p.ave and slopes from specification* 12 ⟩

```
  df <- K - 1
  crit <- qchisq(sig.level, df=df, lower.tail=FALSE)
  ncp0 <- sum(slopes^2 / p.ave) * s2
  if (missing(power)){
    ncp <- ncp0 * N
    power <- pchisq(crit, df=df, ncp=ncp, lower.tail=FALSE)
   }
   else {
     ncp <- cnonct(crit, p=1-power, df=df)
     N <-  ncp / ncp0
   }

   res <- structure(list(n = N, n.prop = n.prop, p.ave=p.ave, slopes = slopes, G = G,
                         sig.level = sig.level, power = power,
                         method = "Multinomial Cochran-Armitage trend test"),
                    class = "power.htest")
   res
   }
```

◇

File defined by 2b, 3, 4, 7b, 9a, 10, 15.
Defines: `power.multiCA.test` 11, 13b, 14.
Uses: `.multiCA.test` 2b, `cnonct` 9a.

"../tests/testthat/test_power.R" 11≡

```
test_that("calculated power is independent of the input format", {
  pmat <- rbind(seq(0.1, 0.4, length=5),
                seq(0.2, 0.3, length=5),
                seq(0.3, 0.1, length=5),
                seq(0.4, 0.2, length=5))
  res0 <- power.multiCA.test(N=100, pmatrix=pmat)
  expect_equal(res0, power.multiCA.test(N=100, p.start=pmat[,1], p.end=pmat[,5], G=5))
  expect_equal(res0, power.multiCA.test(N=100, p.start=pmat[,1], p.ave=rowMeans(pmat),
               G=5))
  expect_equal(res0, power.multiCA.test(N=100, p.end=pmat[,5], p.ave=rowMeans(pmat),
               G=5))
  expect_equal(res0, power.multiCA.test(N=100, p.ave=rowMeans(pmat),
               slopes=pmat[,2]-pmat[,1], G=5))
  expect_equal(res0, power.multiCA.test(N=100, p.start=pmat[,1],
               slopes=pmat[,2]-pmat[,1], G=5))
```

```
  expect_equal(res0, power.multiCA.test(N=100, p.end=pmat[,5],
               slopes=pmat[,2]-pmat[,1], G=5))
})

test_that("Power is computed correctly", {
  pmat <- rbind(seq(0.1, 0.4, length=5),
                seq(0.2, 0.3, length=5),
                seq(0.3, 0.1, length=5),
                seq(0.4, 0.2, length=5))
  res0 <- power.multiCA.test(N=100, pmatrix=pmat)
  expect_equal(100, power.multiCA.test(power=res0$power, pmatrix=pmat)$n)
  expect_equal(0.1, power.multiCA.test(N=100, p.ave=c(0.5, rep(0.1, 5)),
               slopes=rep(0,6), G=6, sig.level=0.1)$power)
})
```

◇

File defined by 9b, 11, 13b, 14, ?, ?.
Uses: `.multiCA.test` 2b, `power.multiCA.test` 10.

When `slopes` is not specified, then a linear trend for each outcome is assumed:

$$p_{ij} = \bar{p}_j + \beta_j(c_i - \bar{c})$$

⟨ *Calculate p.ave and slopes from specification* 12 ⟩ ≡

```
if (!is.null(pmatrix)){
  K <- nrow(pmatrix)
  G <- ncol(pmatrix)
  if (!isTRUE(all.equal(colSums(pmatrix), rep(1, G),
                        check.attributes=FALSE, use.names=FALSE)))
    stop("pmatrix should have column sums of 1.")
  ⟨ Get cbar and s2 13a ⟩
  slopes <- as.vector(pmatrix %*% (n.prop * (scores-cbar))) / s2
  p.ave <- as.vector(pmatrix %*% n.prop)
}
else {
 if (sum(sapply(list(p.ave, slopes, p.start, p.end), is.null)) != 2)
      stop("Either  pmatrix, or exactly two of 'p.ave', 'slopes', 'p.start', and 'p.end' must be specif

if (!is.null(p.ave) & !is.null(slopes)){
  if (length(p.ave) != length(slopes))
    stop("p.ave and slopes should have the same length")
  K <- length(p.ave)
  ⟨ Get cbar and s2 13a ⟩
}
else if (!is.null(p.ave) & !is.null(p.start)){
  if (length(p.ave) != length(p.start))
    stop("p.ave and p.start should have the same length")
  K <- length(p.ave)
  ⟨ Get cbar and s2 13a ⟩
  slopes <- (p.start - p.ave) / (scores[1] - cbar)
}
else if (!is.null(p.ave) & !is.null(p.end)){
  if (length(p.ave) != length(p.end))
    stop("p.ave and p.end should have the same length")
  K <- length(p.ave)
  ⟨ Get cbar and s2 13a ⟩
  slopes <- (p.end - p.ave) / (scores[G] - cbar)
}
else if (!is.null(p.start) & !is.null(p.end)){
```

```
  if (length(p.start) != length(p.end))
    stop("p.start and p.end should have the same length")
  K <- length(p.start)
  ⟨ Get cbar and s2 13a ⟩
  slopes <- (p.end - p.start) / (scores[G] - scores[1])
  p.ave <- p.start - slopes * (scores[1] - cbar)
}
else if (!is.null(p.start) & !is.null(slopes)){
  if (length(p.start) != length(slopes))
    stop("p.start and slopes should have the same length")
  K <- length(p.start)
  ⟨ Get cbar and s2 13a ⟩
  p.ave <- p.start - slopes * (scores[1] - cbar)
}
else if (!is.null(p.end) & !is.null(slopes)){
  if (length(p.end) != length(slopes))
    stop("p.end and slopes should have the same length")
  K <- length(p.end)
  ⟨ Get cbar and s2 13a ⟩
  p.ave <- p.end - slopes * (scores[G] - cbar)
}

⟨ Check validity of p.ave and slopes 13c ⟩
}
◇
```

Fragment referenced in 10.

⟨ *Get cbar and s2* 13a ⟩ ≡

```
  if (missing(G)){
    if (!missing(scores)) G <- length(scores)
    else if (!missing(n.prop)) G <- length(n.prop)
    else stop("The number of groups G needs to be specified explicitly or implicitly through the dimens
  }
  if (sum(n.prop) != 1) n.prop <- n.prop/sum(n.prop)
  cbar <- weighted.mean(scores, w=n.prop)
  s2 <- sum(n.prop * (scores-cbar)^2)
◇
```

Fragment referenced in 12.

"../tests/testthat/test_power.R" 13b≡

```
test_that("G is properly identified", {
  expect_error(power.multiCA.test(N=100, p.start=c(0.1, 0.9), p.end=c(0.8, 0.2)),
               "G needs to be specified")
  expect_equal(power.multiCA.test(N=100, p.start=c(0.1, 0.9), p.end=c(0.8, 0.2),
               n.prop=rep(1,4))$G, 4)
  expect_equal(power.multiCA.test(N=100, p.start=c(0.1, 0.9), p.end=c(0.8, 0.2),
               scores=1:4)$G, 4)
})

test_that("Scaling of n.prop does not matter", {
  expect_equal(
     power.multiCA.test(N=100, p.start=c(0.1, 0.9), p.end=c(0.8, 0.2), G=6,
                        n.prop=rep(1,6)),
     power.multiCA.test(N=100, p.start=c(0.1, 0.9), p.end=c(0.8, 0.2), G=6,
                        n.prop=rep(2,6)))
```

```
  })
```
◇

File defined by 9b, 11, 13b, 14, ?, ?.
Uses: `.multiCA.test` 2b, `power.multiCA.test` 10.

To ensure a valid setup, slopes should add up to 0, and all of the $p_{ij}$'s implied by a linear trend should be between 0 and 1.

⟨ *Check validity of p.ave and slopes* 13c ⟩ ≡

```
if (!isTRUE(all.equal(sum(slopes), 0, check.attributes=FALSE, use.names=FALSE)))
    stop("Implied or specified values of slopes should sum to 0.")
if (!isTRUE(all.equal(sum(p.ave), 1, check.attributes=FALSE, use.names=FALSE)))
    stop("Implied or specified values of p.ave should sum to 1.")
check <- outer(1:K, 1:G, function(j,i)p.ave[j] + slopes[j]*(scores[i]-cbar))
if (!all(check >= 0) || !(all(check <=1)))
  stop("The parameters do not define a valid probability matrix")
```
◇

Fragment referenced in 12.

`"../tests/testthat/test_power.R"` 14≡

```
test_that("p.ave and slopes are checked for validity", {
  expect_error(power.multiCA.test(N=100, p.start=c(0.1, 0.3), p.end=c(0.8, 0.2), G=3),
               "slopes should sum to 0")
  expect_error(power.multiCA.test(N=100, p.ave=c(0.1, 0.8), slopes=c(0.1, -0.1), G=4),
                "p.ave should sum to 1")
  expect_error(power.multiCA.test(N=100, p.ave=c(0.1, 0.9), slopes=c(0.1, -0.1), G=4),
                "valid probability matrix")
  expect_error(power.multiCA.test(N=100, p.ave=c(0.4, 0.6), slopes=c(0.1, 0.1), G=3),
               "slopes should sum to 0")
})
```
◇

File defined by 9b, 11, 13b, 14, ?, ?.
Uses: `.multiCA.test` 2b, `power.multiCA.test` 10.

4.1. **Power for Cochran-Armitage trend test.** For $K = 2$ the multinomial trend test reduces to the regular Cochran-Armitage trend test. For the power calculation, however, we do not need the approximation used for the multinomial test, and a better formula can be derived.

Simplifying notations, let $p_i$, $i = 1, \dots, G$ denote the probability of an event in group $i$, and $x_i$ events out of $n_i$ trials are observed in this group. Then the unnormalized CA test statistic is $U = \sum_i x_i(c_i - \bar{c})$ with $\bar{c} = \sum_i \nu_i c_i$, $\nu_i = n_i/N$ and

$$U \mid H_0 \sim N(0, N\sigma_0^2)$$
$$U \mid H_a \sim N(N\mu, N\sigma^2)$$

where $\mu = \sum_i \nu_i p_i (c_i - \bar{c})$ is the mean, $\sigma_0^2 = \bar{p}(1-\bar{p}) \sum_i \nu_i (c_i - \bar{c})^2$ is the null variance, and $\sigma^2 = \sum_i \nu_i p_i (1 - p_i)(c_i - \bar{c})^2$ is the alternative variance.

The traditional CA test statistic $T = U^2/(N\sigma_0^2) \sim \chi_1^2$ under $H_0$. Under $H_a$ we have

$$T = \frac{U^2}{N\sigma_0^2} = \frac{\sigma^2}{\sigma_0^2} \frac{U^2}{N\sigma^2}$$

so

$$\frac{\sigma_0^2}{\sigma^2} T = \frac{U^2}{N\sigma^2} \sim \chi_1^2\Big(N\frac{\mu^2}{\sigma^2}\Big),$$

since $\frac{U}{\sqrt{N}\sigma} \sim N\Big(\frac{\sqrt{N}\mu}{\sigma}, 1\Big)$.

The non-centrality parameter

$$\lambda = N\frac{\mu^2}{\sigma^2} = \frac{\big[\sum \nu_i p_i (c_i - \bar{c})\big]^2}{\sum_i \nu_i p_i (1-p_i)(c_i-\bar{c})^2} = N\frac{\beta^2 s_\nu^2}{\bar{p}(1-\bar{p})}\frac{\sigma_0^2}{\sigma^2} = \lambda_{\text{approx}}\frac{\sigma_0^2}{\sigma^2},$$

where $\beta$ is the slope of regressing $p_i$ on $c_i$ with weights $\nu_i$ and $s_\nu^2 = \sum_i \nu_i (c_i - \bar{c})^2$, as $\mu = \beta s_\nu^2$ and $\sigma_0^2 = \bar{p}(1-\bar{p})s_\nu^2$.

In addition to the traditional test statistic, a $Z$-based version that allows directional testing is also implemented. This is based on the test statistic

$$Z = \frac{U}{\sqrt{N}\sigma_0},$$

which has a $N(0,1)$ distribution under $H_0$ and $N(\mu\sqrt{N}/\sigma_0, \sigma^2/\sigma_0^2)$ distribution under $H_a$.

For one-sided testing, we have

$$\text{Power} = 1 - \beta = P(Z > z_\alpha) = 1 - \Phi\Big(\frac{z_\alpha - \mu\sqrt{N}/\sigma_0}{\sigma/\sigma_0} = 1 - \Phi\Big(\frac{\sigma_0 z_\alpha - \mu\sqrt{N}}{\sigma}\Big)$$

$$N = \Big[\frac{\sigma_0 z_\alpha + \sigma z_\beta}{\mu}\Big]^2$$

"../R/multiCA.R" 15≡

```
#' Power calculations for the Cochran-Armitage trend test
#'
#' @param N integer, the total sample size of the study. If \code{NULL} then \code{power} needs to be spe
#' @param power target power. If \code{NULL} then \code{N} needs to be specified.
#' @param pvec numeric vector of hypothesized outcome probabilities in each group.
#' @param scores non-decreasing numeric vector of the same length as the number of ordered groups
#' giving the trend test scores. Defaults to linearly increasing values.
#' @param n.prop numeric vector describing relative sample sizes of the ordered groups.
#' Will be normalized to sum to 1. Defaults to equal sample sizes.
#' @param sig.level significance level
#' @param alternative character string specifying the alternative hypothesis
#' @return object of class "power.htest"
#'
#' @examples
#' # sample size required to detect with 80% power a decreasing trend over 4 groups
#' # with 3:2:1:2 sample-size distribution at a 2.5% significance level
#' power.CA.test(power=0.8, pvec=c(0.4, 0.3, 0.2, 0.1), n.prop=c(3,2,1,2),
#'               alternative = "less", sig.level=0.025)
#'
#' # power of a 2-sided test to detect a logistic increase with slope 0.2 over 5 groups
#' # with groups of size 10 with unequal dose spacing
#' doses <- c(0,1,2,4,8)
#' p0 <- 0.05 # event probability at lowest dose
#' logit.props <- log(p0/(1-p0)) + doses * 0.2
#' p <- 1 / (1 + exp(-logit.props)) # hypothesized probabilities at each dose
#' power.CA.test(N = 10 * 5, pvec=p, scores = doses)
#'
#' @export
#' @references Nam, J. (1987). A Simple Approximation for Calculating Sample Sizes for Detecting Linear T
#' Biometrics, 43(3), 701-705.
#' @importFrom stats pnorm qnorm
#'

power.CA.test <- function(N=NULL, power=NULL, pvec=NULL, scores=seq_along(pvec),
                          n.prop=rep(1, length(pvec)), sig.level=0.05,
                          alternative = c("two.sided", "less", "greater")){
```

```
if (sum(sapply(list(N, power), is.null)) != 1)
      stop("exactly one of 'N',  and 'power' must be NULL")
if (!is.numeric(sig.level) || any(0 > sig.level | sig.level > 1))
      stop("'sig.level' must be numeric in [0, 1]")
if (any(pvec < 0) | any(pvec > 1))
      stop("All probabilities in 'pvec' should be between 0 and 1")
if (length(pvec) != length(scores) | length(pvec) != length(n.prop))
    stop("Vectors 'pvec', 'scores', and 'n.prop', if specified, should have the same lengths.")


alternative <- match.arg(alternative)

n.prop <- n.prop / sum(n.prop)


sbar <- sum(scores * n.prop)
pbar <- sum(pvec * n.prop)
v.nu <- sum(n.prop * (scores-sbar)^2)
v0 <- pbar * (1-pbar) * v.nu
v <- sum(n.prop * pvec * (1-pvec) * (scores-sbar)^2)
mu <- sum(n.prop * pvec * (scores - sbar))

if (alternative == "two.sided"){
  crit <- qchisq(sig.level, df=1, lower.tail=FALSE)
  ncp0 <- mu^2 / v
} else {
  crit <- qnorm(sig.level, lower.tail=FALSE)
  if (alternative == "less") crit <- (-1) * crit
}


if (missing(power)){
  if (alternative == "two.sided"){
    ncp <- ncp0 * N
    power <- pchisq(v0/v * crit, df=1, ncp=ncp, lower.tail=FALSE)
  } else {
    term <- (sqrt(v0)*crit - mu * sqrt(N)) / sqrt(v)
    power <- pnorm(term, lower.tail = (alternative == "less"))
  }
 }
 else {
  if (alternative == "two.sided"){
    ncp <- cnonct(v0/v *crit, p=1-power, df=1)
    N <-  ncp / ncp0
  } else {
    zb <- qnorm(power, lower.tail=TRUE)
    N <- (sqrt(v0) * crit + sqrt(v) * zb)^2 / mu^2
  }
 }

 res <- structure(list(n = N, n.prop = n.prop, p = pvec,
                       alternative = alternative,
                       sig.level = sig.level, power = power,
                        method = "Cochran-Armitage trend test"),
                  class = "power.htest")
 res
 }
```

◇

File defined by 2b, 3, 4, 7b, 9a, 10, 15.
Uses: cnonct 9a.

"../tests/testthat/test_power.R" ?≡

```
test_that("Binomial power calculation works", {
  pvec0 <- seq(0.1, 0.2, length.out = 5)
  res0 <- power.CA.test(N=100, pvec = pvec0)
  expect_equal(100, power.CA.test(power=res0$power, pvec = pvec0)$n)
  res_lo <- power.CA.test(N=100, pvec = pvec0, alternative = "less",
        sig.level = res0$sig.level/2)
  res_up <- power.CA.test(N=100, pvec = pvec0, alternative = "greater",
        sig.level = res0$sig.level/2)
  expect_equal(res0$power, res_lo$power + res_up$power)
  expect_equal(0.1, power.CA.test(N=100, pvec = rep(0.2, 4), sig.level=0.1)$power)
  })
```

◇
File defined by 9b, 11, 13b, 14, ?, ?.

"../tests/testthat/test_power.R" ?≡

```
test_that("power.CA.test inputs are checked for validity", {
  expect_error(power.CA.test(N=100, pvec = c(-0.5, 0.4)),
               "should be between 0 and 1")
  expect_error(power.CA.test(N=100, pvec = c(0.5, 1.4)),
               "should be between 0 and 1")
  expect_error(power.CA.test(N=100, pvec=c(0.1, 0.8), scores=1:3),
                 "same lengths")
  expect_error(power.CA.test(N=100, pvec=c(0.1, 0.8), scores=1:2, n.prop=1:3),
                 "same lengths")
  expect_error(power.CA.test(N=100, pvec = c(0.1, 0.2), power=0.8),
                 "must be NULL")
  expect_error(power.CA.test(pvec = c(0.1, 0.2)),
                 "must be NULL")
})
```

◇
File defined by 9b, 11, 13b, 14, ?, ?.

## 5. Files

"../R/aaa-generics.R" Defined by 2a.
"../R/multiCA.R" Defined by 2b, 3, 4, 7b, 9a, 10, 15.
"../tests/testthat/test_overall.R" Defined by 5a, 6b, 7a.
"../tests/testthat/test_power.R" Defined by 9b, 11, 13b, 14, ?, ?.

## 6. Macros

⟨ Calculate adjusted p-values 5b ⟩ Referenced in 3.
⟨ Calculate correlation and contrast matrices 8b ⟩ Referenced in 2b.
⟨ Calculate p.ave and slopes from specification 12 ⟩ Referenced in 10.
⟨ Check validity of p.ave and slopes 13c ⟩ Referenced in 12.
⟨ Closed set adjustment 6c ⟩ Referenced in 5b.
⟨ Get cbar and s2 13a ⟩ Referenced in 12.
⟨ glht adjustment 8a ⟩ Referenced in 5b.
⟨ Holm-Shaffer adjustment 6a ⟩ Referenced in 5b.

## 7. Identifiers