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 ith 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: September 1, 2016.

"../R/aaa-generics.R" 2a≡

```
#'Multinomial Cochran-Armitage trend test
#'
#'The \code{multiCA.test} 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. (2016) Test for trend with a multinomial outcome.
#'@keywords nonparametric
#'@examples
#'
#'data(stroke)
#'## using formula interface
#'multiCA.test(Type ~ Year, weights=Freq, data=stroke)
#'
#'## 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 is encapsulated in an internal function that operates on a matrix. No error control is provided here.

"../R/multiCA.R" 2b≡

```
#'@keywords internal

.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
  CAT <- X[nonz]^2 / (pdot[nonz] * (1-pdot[nonz])) / s2
```

```
  CAT.p.value <- pchisq(CAT, df=1, lower.tail=FALSE)

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

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

  res <- list(statistic = Tt, parameter = df, p.value = p.value,
              indiv.statistics = CAT, indiv.p.value = CAT.p.value)
  return(res)
}
```

◇

File defined by 2b, 3, 4a, 6c, 8a, 9.
Defines: `.multiCA.test` 3, 6a, 9, 10a, 12ac.

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

multiCA.test.default <- function(x, scores=1:ncol(x), outcomes=1:nrow(x),
  p.adjust.method=c("none","closed.set","holm-Shaffer"),...){
  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)

  Tt <- 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 = Tt, 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 5a ⟩
```

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

File defined by 2b, 3, 4a, 6c, 8a, 9.
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" 4a≡

```
#'@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

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, 4a, 6c, 8a, 9.
Defines: `multiCA.test.formula` Never used.

"../tests/testthat/test_overall.R" 4b≡

```
  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 4b, 5c, 6b.

3. Multiple testing adjusted inference for individual outcomes

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

```
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 6a ⟩
} else if (p.adjust.method=="holm-Shaffer") {
  ⟨ Holm-Shaffer adjustment 5b ⟩
}
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* 5b ⟩ ≡

```
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 5a.

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

```
  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 4b, 5c, 6b.

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 $p(\mathcal{J}) = P(W_j \geq \chi^2_{|\mathcal{J}|})$ 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* 6a ⟩ ≡

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

◇

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

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

```
  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 4b, 5c, 6b.

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" 6c≡

```
#' @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
.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, 4a, 6c, 8a, 9.

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 ν_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" 8a≡

```
#' 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

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, 4a, 6c, 8a, 9.
Defines: cnonct 8b, 9.

"../tests/testthat/test_power.R" 8b≡

```
  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 8b, 10a, 12ac.
Uses: cnonct 8a.

"../R/multiCA.R" 9≡

```
#' 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 distribution of the outcomes can be specified in two ways: either the full matrix of #' outcome pr
#'
#' @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 outcom
#' @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} wiht weights \code{n.prop}
#' @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))
#' @export

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* 10b ⟩

```
  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, 4a, 6c, 8a, 9.
Defines: `power.multiCA.test` 10a, 12ac.
Uses: `.multiCA.test` 2b, `cnonct` 8a.

"../tests/testthat/test_power.R" 10a≡

```
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 8b, 10a, 12ac.
Uses: `.multiCA.test` 2b, `power.multiCA.test` 9.

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* 10b ⟩ ≡

```
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 11 ⟩
  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 11 ⟩
}
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 11 ⟩
  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 11 ⟩
  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 11 ⟩
  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 11 ⟩
  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 11 ⟩
  p.ave <- p.end - slopes * (scores[G] - cbar)
}

  ⟨ Check validity of p.ave and slopes 12b ⟩
}
◇
```

Fragment referenced in 9.

⟨ *Get cbar and s2* 11 ⟩ ≡

```
  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 10b.

"../tests/testthat/test_power.R" 12a≡

```
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 8b, 10a, 12ac.
Uses: `.multiCA.test` 2b, `power.multiCA.test` 9.

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* 12b ⟩ ≡

```
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 10b.

"../tests/testthat/test_power.R" 12c≡

```
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 8b, 10a, 12ac.
Uses: `.multiCA.test` 2b, `power.multiCA.test` 9.

5. Files

`"../R/aaa-generics.R"` Defined by 2a.
`"../R/multiCA.R"` Defined by 2b, 3, 4a, 6c, 8a, 9.
`"../tests/testthat/test_overall.R"` Defined by 4b, 5c, 6b.
`"../tests/testthat/test_power.R"` Defined by 8b, 10a, 12ac.

6. Macros

⟨ Calculate adjusted p-values 5a ⟩ Referenced in 3.
⟨ Calculate p.ave and slopes from specification 10b ⟩ Referenced in 9.
⟨ Check validity of p.ave and slopes 12b ⟩ Referenced in 10b.
⟨ Closed set adjustment 6a ⟩ Referenced in 5a.
⟨ Get cbar and s2 11 ⟩ Referenced in 10b.
⟨ Holm-Shaffer adjustment 5b ⟩ Referenced in 5a.

7. Identifiers

`.multiCA.test`: 2b, 3, 6a, 9, 10a, 12ac.
`cnonct`: 8a, 8b, 9.
`multiCA.test.default`: 3.
`multiCA.test.formula`: 4a.
`power.multiCA.test`: 9, 10a, 12ac.