Fourth-corner correlation is a score test statistic in a log-linear trait-environment model that is useful in permutation testing.

Cajo J. F. ter Braak

Appendix S1. R-code for the score tests of section 2 and comparison with the R package mdscore.

1. Introduction

This appendix gives R-code and results of simulations illustrating 1) the analytical equations (7), (10), (15) and (43) for the score test statistic for single and multiple traits and environmental variables, 2) their numerical equality to the values obtained using the R package mdscore (da Silva-Junior et al. 2015) and the difference with the log-likelihood ratio and 3) the approximate chi-square distribution of the score test under the Poisson null model that has row and column mains effects but no interactions.

Section 2 gives the R-code of the data generating function and functions for the fourth-corner and score statistic. Section 3 provides the R-script and section 4 gives example output.

2. R-functions

2.1 Data generating functions

The key functions are:

generate_Tdata – generates multivariate normal trait data and a linear combination thereof that is standard normally distributed.

generate_Edata – generates multivariate normal environmental data and a linear combination thereof that is standard normally distributed.

generate_RC_community – implements the basic log-linear model with different deviations from the Poisson such as interactions with unobserved traits (z) and environmental variables (x).

Save the following code as R-file with name:
generate_trait_env_community_data_loglinear.r

```
# function to generate one- and multi-trait data ---------------------------------
-----
generate_Tdata <-function(n_species,n_traits, correlation = 0, b){
  # n_species = number of species; n_traits = number of traits;
  # correlation = AR(1) autocorrelation between subsequent variables
  # b = coefficient of the linear combination (c in the paper)
  # generate n_traits correlated normally distibuted trait variables in Tdat
  # a linear combination of these  Tdat%*%b  in T
  # rescale b and T so that T ~ N(0,1),
  # value: list with Tdat, T and coefficients (rescaled b)
  ii <- 1:n_traits
```

```
  Sigma <-  correlation^abs(outer(ii,ii,  '-'))
  sdT <-  sqrt(b %*% Sigma %*%b)
  coefs <- b/sdT
  Tdat<-MASS::mvrnorm(n_species,mu=rep(0,n_traits),Sigma)
  T <- Tdat %*%coefs
  rownames(T)<- rownames(Tdat)<- paste("spec", seq_len(nrow(Tdat)), sep="")
  return(list(T = T, Tdat= Tdat, coefficients = coefs))
}

generate_Edata <-function(n,p, correlation = 0, b, mixfactor = 0){
  # generate p correlated normally distibuted E variables in Edat
  # a linear combination of these in E
  # rescale b and E so that E ~N(0,1),
  # value: list with Edat, E and coefficients (rescaled b)
  ii <- 1:p
  Sigma <-  correlation^abs(outer(ii,ii,  '-'))
  sdE <-  sqrt(b %*% Sigma %*%b)
  coefs <- b/sdE
  Edat<-MASS::mvrnorm(n,mu=rep(0,p),Sigma) + mixfactor * matrix(2*((runif(n*p) > 0.
5)-0.5),nrow = n,ncol =p)
  E <- Edat %*%coefs
  rownames(E)<- rownames(Edat)<- paste("site", seq_len(nrow(Edat)), sep="")
  return(list(E = E, Edat= Edat, coefficients = coefs))
}

# function to generate (zero inflated ) negative binomial data ------------
ZInbinom <- function(mu,sigma = 1, nu= 0){
  L <- rnbinom(length(mu), size = 1/sigma, mu = mu)
  # variance = mu + mu^2/size = mu + sigma * mu^2
  L <- ifelse(runif(length(mu)) < nu,  0,L)
  matrix(L,nrow = nrow(mu),ncol = ncol(mu))
}

# function to generate (zero inflated ) poisson data ------------
ZIpoisson <- function(mu, nu= 0){
  L <- rpois(length(mu), lambda = mu)
  L <- ifelse(runif(length(mu)) < nu,  0,L)
  matrix(L,nrow = nrow(mu),ncol = ncol(mu))
}

# function to generate data: RC-model/ loglinear model with y~NegBin eqn A.3 of ter
Braak et al 2016 PeerJ--------------------
generate_RC_community <- function(E,T, coefs=c(0,0.2,0,0,0),
                                  sigma_epsilon_ij = 0, mu0 =30,
                                  coefsE=c(0.2, 0, 0),  coefsx=c(0,0,0),
                                  coefsT=c(0.2, 0, 0),  coefsz=c(0,0,0),
                                  distribution = "negbin", link.mu = "log",
                                  sigma = 1, nu = 0,
                                  fct = 0.8, max.try = 5){
  #
  # one environmental variable E, one trait T
  # main effects:
  #    row and column effects functions
  #       f_row and f_col: a polynomial in E, T,x and z respectively
  #           d1*E + d2*E^2 + d3*rnorm(n, 0,1), with coefsE = c(d1,d2,d3)
  #           d1*T + d2*T^2 + d3*rnorm(n, 0,1), with coefsT = c(d1,d2,d3)
  #           d1*x + d2*x^2 + d3*rnorm(n, 0,1), with coefsx = c(d1,d2,d3)
  #           d1*z + d2*z^2 + d3*rnorm(n, 0,1), with coefsz = c(d1,d2,d3)
  #           added noise if d3 is nonzero
  # interaction effects
  #        b1*T*E + b2*z*E + b3*T*x + b4*x*z+ b5*xx*zz eps with coefs = c(b1,b2,b3,
b4)
  #        with z ~ N(0,1), x~N(0,1), zz ~ N(0,1), xx~N(0,1)and eps~sigma_epsilon_i
j*N(0,1)
  #        Note:  trait-environment association only if b1 !=0
  #        Note:  confounding random effects if coefs[2:3] are non-zero
```

```
  #
  # dist ="negbin", "ZINBI" , "pois", "binomial"
  # link =  link function "log", "logit"
  # if center. = TRUE then the interaction is double centered
  # the repeat loop is to ensure that the final number of  species and sites is wit
hin fct (0.8) of the number asked for in  argument

  # The equi-tolerance versions of the Gaussian models
  # of Peres-Neto, Dray & ter Braak (2016, Ecography http://dx.doi.org/10.1111/ecog
.02302) and
  # ter Braak, Peres-Neres & Dray (2016, PeerJ https://peerj.com/articles/2885/) ca
n be simulated
  # by setting coefsE-coefsz and coef as follows:
  # The environmentally structured, random trait case corresponds to
  # coefsE=c(0,-0.5/tolerance^2, 0);  coefsx=c(0,0,0)
  # coefsT=c(0,0, 0);  coefsz=c(0,-0.5/tolerance^2,0)
  # The trait structured, random environment case corresponds to
  # coefsE=c(0,0, 0);  coefsx=c(0,-0.5/tolerance^2,0)
  # coefsT=c(0,-0.5/tolerance^2, 0);  coefsz=c(0,0,0)
  # The both random case corresponds to
  # coefsE=c(0,0, 0);  coefsx=c(0,-0.5/tolerance^2,0)
  # coefsT=c(0,0, 0);  coefsz=c(0,-0.5/tolerance^2,0)

  #  local functions: marginal effects
  f_row <- function(E, coefsE){
    effects <- coefsE[1]*E + coefsE[2]*E*E + coefsE[3]*rnorm(length(E))
    effects
  }
  f_col <- function(T, coefsT){f_row(T,coefsE=coefsT)}
  # end local functions
  E <- c(E);
  T <- c(T)
  n_communities <- length(E)
  n_species <- length(T)
  i.try <- 1
  if (length(coefs)==3) coefs <- c(coefs,0,0)
  if (length(coefs)==4) coefs <- c(coefs,0)
  repeat {
    i.try <- i.try + 1
    x <-  rnorm(length(E)); z <-  rnorm(length(T))
    xx <-  rnorm(length(E)); zz <-  rnorm(length(T))
    mu.interaction <-
      coefs[1] * outer(E,T,FUN = "*") +
      coefs[2] * outer(E,z,FUN = "*") +
      coefs[3] * outer(x,T,FUN = "*") +
      coefs[4] * outer(x,z,FUN = "*") +
      coefs[5] * outer(xx,zz,FUN = "*") +
      sigma_epsilon_ij * matrix(rnorm(n_communities*n_species), nrow =n_communities
)
    mu.marginal <- outer(rep(0,n_communities), f_col(T,coefsT), FUN="+") +
      outer(f_row(E,coefsE), rep(0,n_species), FUN="+") +
      outer(rep(0,n_communities), f_col(z,coefsz), FUN="+") +
      outer(f_row(x,coefsx), rep(0,n_species), FUN="+")
    mu.lp <- mu.marginal + mu.interaction
    # for numerical stability we needed to limit mu.lp
    mu.lp <- ifelse(mu.lp > 10, matrix(10, nrow=nrow(mu.lp),ncol=ncol(mu.lp)), mu.l
p)
    mu.lp <- ifelse(mu.lp < -10,matrix(-10, nrow=nrow(mu.lp),ncol=ncol(mu.lp)), mu.
lp)
    if (mu0 > 0.9 && (distribution =="binomial"|| link.mu =="logit"))  mu0 <- 0.5 #
    mu <- switch(link.mu,
                 log =  mu0 * exp(mu.lp) ,
                 logit = 1/(1 + exp(-( mu.lp + log(mu0/(1-m0)))) )
    )
    L <- switch( distribution,
                 poisson = ZIpoisson(mu,nu=nu),
```

```
                  binomial = matrix(rbinom(prod(dim(mu)),1, mu),nrow = nrow(mu),ncol
= ncol(mu)),
                  negbin = ZInbinom(mu = mu, sigma= sigma, nu = 0),
                  ZINBI =  ZInbinom(mu = mu, sigma=sigma, nu =nu)
    )
    nspecies_c <- sum(colSums(L)!=0) # _c for check
    ncommunities_c <- sum(rowSums(L)!=0)
    if ((nspecies_c >= fct * n_species) && (ncommunities_c >=fct *n_communities ||
i.try> max.try)){break}
  }
  rownames(L)<- paste("site", seq_len(nrow(L)), sep="")
  colnames(L)<- paste("spec",seq_len(ncol(L)),sep="")
  return(L) # L = Y in the paper
}
```

2.2 R functions for the score test statistic

The key functions are:

make_obj_for_traitenv - makes object of class "TE_obj" from matrices or dataframes from L, E1 and T1; no factors allowed, as all is converted to matrices; the object is a list of three matrices: L, E, T where L takes the role of Y in the paper.

fourthcorner - calculates the squared fourth-corner correlation for quantitative traits and environmental variables. For more than one trait and one environmental variable, it gives both the score test statistic and the sum of the squared fourth-corner correlations. The former is the total inertia of a doubly constrained correspondence analysis (dCCA) and the latter is the total inertia in an RLQ analysis.

There are several auxiliary functions. Save the following code as R-file with name: te_score_functions1.r

```
# make object from data (L=Y) with checks on empty row/cols ----------------------
----------

make_obj_for_traitenv <- function(L,E1,T1, cut_off = 0){
  # makes object of class TE_obj from matrices or dataframes from L,E1 and T1
  # no factors allowed, as all is converted to matrices
  # the object is a list of three matrices: L, E, T
  L <- as.matrix(L)
  E1 <- as.matrix(E1)
  T1 <- as.matrix(T1)
  # check_L()
  rows<-seq_len(nrow(L))
  cols<-seq_len(ncol(L))
  rni <-which(rowSums(L)==0)
  repeat {
    if (length(rni)) {L <- L[-rni,,drop = FALSE]; rows <-rows[-rni]}
    ksi <- which(colSums(L)==0)
    if (length(ksi)) {L <- L[,-ksi, drop = FALSE]; cols <- cols[-ksi]}
    rni <-which(rowSums(L)==0)
    if ( length(rni)==0 & length(ksi)==0){break}
  }
  E1 <-as.matrix(E1)[rows,,drop = FALSE]
  T1 <-as.matrix(T1)[cols,,drop = FALSE]
  # end check_L()
  if(cut_off >0 ){
    abOcc <- apply(L>0,2,mean)
```

```
    L <- L[,abOcc>cut_off, drop =FALSE]
    T1 <- as.matrix(T1)[abOcc>cut_off,, drop = FALSE]
  }
  if(is.null(rownames(L)))rownames(L)<- paste("site", 1:nrow(L), sep="")
  if(is.null(colnames(L)))colnames(L)<- paste("spec",1:ncol(L),sep="")
  rownames(E1)= rownames(L); colnames(E1)=  paste("E", 1:ncol(E1), sep = "")
  rownames(T1)= colnames(L); colnames(T1)=  paste("T", 1:ncol(T1), sep = "")
  obj = list(L=L, E=E1,T = T1)
  class(obj) = c("TE_obj", class(obj))
  return(obj)
}


# convert lists of data frames to matrices and the reverse ----------------

df2matrix <- function(obj){
  # makes matrices from the data frames in obj
  lapply(obj,as.matrix)
}

matrix2df <- function(obj){
  # makes dataframes from the matrices in obj
   lapply(obj,as.data.frame)
}

# auxilary functions used in fourthcorner and score_test_ET

center_w <- function(X,w = rep(1/nrow(X),nrow(X))){ X - rep(1,length(w))%*%t(w)%*%
X }

standardize_w <- function(X,w = rep(1/nrow(X),nrow(X))){
  ones <- rep(1,length(w))
  Xc <- X - ones %*% t(w)%*% X
  Xc / ones%*%sqrt(t(ones)%*%(Xc*Xc*w))
}

wSVD <- function(Y,w=rep(1/nrow(Y),nrow(Y)) , eps = 1e-6){
  sw <- sqrt(w)
  Ystar <- Y*sw
  svdY <- svd(Ystar)
  nonzero <- which(svdY$d > eps)
  Ustar <- svdY$u[,nonzero]
  return(Ustar/sw)
} # returns w-orthogonalized Y

# powers of A: let us define a function for this for more general use
pow.matrix <- function(A, alpha, eps = 1.0e-10){
  eigA = eigen(A)
  Q = eigA$vectors
  l = eigA$values
  nonzero = which(l > eps)
  l.alpha = 0 * l
  l.alpha[nonzero] = l[nonzero]^alpha # avoid a power of a zero eigenvalue
  L = diag(l.alpha)
  Q %*% L %*%t(Q)  # power of A
}
# end of auxilary functions

# fourth corner test statistics -------------------------------------------
fourthcorner <- function(obj, dCCA = TRUE,...){
  # obj is from make_obj_for_traitenv for all preprocessing
  # ... added to allow for use in as argument in other functions with more
arguments
  # value:
  # for 1 trait and 1 environmental variable or if dCCA = FALSE:
```

```
  #       the sum of the squares of the fourthcorner correlation(s) between E and T
(can be matrices)
  # else two values
  # "dCCA" = total inertia of a doubly constrained correspondence analysis (dCCA)
  # "RLQ" = total inertia of an RLQ which is
  #                   1. the sum of the squares of the fourthcorner correlation(s)
between E and T
  #                   2. as dCCA, but ignoring the within set correlation among E
and among T variables
  with(obj,{
    L<-L/sum(L)
    Wn <- rowSums(L)
    Ws <- colSums(L)
    Estd_w <- standardize_w(E,Wn)
    Tstd_w <-  standardize_w(T,Ws)
    Fourthcorner <- t(Estd_w)%*%(L%*%Tstd_w)
    Fourthcorner <- sum(Fourthcorner*Fourthcorner)
    if (ncol(Estd_w)==1 && ncol(Tstd_w)==1 || !dCCA)
{return(c(Fourthcorner=Fourthcorner)) }
    Estd_w <- wSVD(Estd_w, Wn)
    Tstd_w <- wSVD(Tstd_w, Ws)
    dCCA <- t(Estd_w)%*%(L%*%Tstd_w)
    dCCA <- sum(dCCA*dCCA)
    res <- c(RLQ=Fourthcorner,dCCA=dCCA)
    return(c(dCCA=dCCA,RLQ=Fourthcorner))
  })
}

# native score test statistics using eqn for S(B=0) about eqn 10-----------------
-------------------------
score_test_ET <- function(obj){
  # function that implements equ for S(B=0) in eqn 10  and 11 [or (40) and (41)] in
  # doi: 10.1007/s10651-017-0368-0
  # obj is from make_obj_for_traitenv for all preprocessing
  # value: score test statistic for trait-environment interaction
    Y <- obj$L;
    Wn <- rowSums(Y)
    Ws <- colSums(Y)
    E <- center_w(obj$E,Wn/sum(Wn))
    T <- center_w(obj$T,Ws/sum(Ws))
    if (ncol(E)>1)  ERE_h <- pow.matrix(t(E)%*% diag(Wn)%*%E,-1/2) else ERE_h <- 1
/ sqrt(sum(Wn*E*E))
    if (ncol(T)>1)  TCT_h <- pow.matrix(t(T)%*% diag(Ws)%*%T,-1/2) else TCT_h <- 1
/ sqrt(sum(Ws*T*T))
    D <- ERE_h %*% t(E)%*% Y %*% T %*% TCT_h
    SB <- sum(Y)* sum(diag(t(D)%*%D))
    return(c(SB=SB))
}


Rten2_with_names <- function(X1,X2) {
  one.1 <- matrix(1,1,ncol(X1))
  one.2 <- matrix(1,1,ncol(X2))
  res  <- kronecker(X1,one.2)*kronecker(one.1,X2) #  column of X2 runs fastest
  colnames(res) <- kronecker(colnames(X1),colnames(X2),function(a,b){ paste(b, a,
sep = ":")} )
  rownames(res) <- rownames(X1)
  return(res)
}

#  expand the data in the obj (L,E,T) in vector form ----------------------
expand4glm <- function(obj){
  #if (class(obj)[1]=="TE_obj") obj <- matrix2df(obj)
  # adapted from Jamil et al 2013
  with(obj, {
```

```
    sitespec <- expand.grid(rownames(L),colnames(L))
    site <-sitespec[,1]; species<-sitespec[,2]
    y <- as.vector(as.matrix(L))
    Evec <-  E[site,, drop = FALSE];  rownames(Evec)= NULL
    Tvec <-  T[species,, drop = FALSE]; rownames(Tvec)= 1:nrow(Tvec)
    # #for one Evec!!!
    #ET <- Tvec * Evec[,1]; colnames(ET)= paste("E", colnames(T),sep="")
    # #for more environmental variables
    colnames(Evec)= paste("E", 1:ncol(Evec),sep="")
    colnames(Tvec)= paste("T", 1:ncol(Tvec),sep="")
    ET <- Rten2_with_names(Evec,Tvec)
    XYZ <- data.frame(y,site,species,ET)
    return(XYZ) # XYZ notation of Jamil et al 2013
  })
}
```

3. R-script

The script requires the R packages `mdscore` (da Silva-Junior et al. 2015) and `ade4` (Dray and Dufour 2007).

```
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("generate_trait_env_community_data_loglinear.r")
source("te_score_functions1.r")
library(mdscore)
library(ade4)

set.seed(1353)

# Section 1 simplest log-linear simulation model b_tx = 0 ------------------------
--------------------

n_communities <- 5 #n
n_species <- 7 # m
n_trait <- 1  #q
n_env <- 1 # p
cor_Tdat <- 0.5
cor_Edat <- 0.5

n_simul <-  100  #
distri_Y <- "poisson"  # distribution used for generating the response data Y given
E(Y)

mudat0 = 30
# effect sizes in log-linear model (b1,b2,b2 = b_te, b_ze, b_tx)
# with x and z random nuisance variables.
b_te <- 0   # the null model, without t-e interaction
b_ze <- 0
b_tx <- 0
b_zx <- 0
b_zx_star <- 0
sigma_epsilon_ij <- 0 # min(0.1, b_te + b_ze + b_tx)   # unstructured interaction


ef.main <- 0.2
ef.main.sq <-  0
ef.main.ran.rows <- 0
ef.main.ran.columns <- 0

if (ef.main.sq < 0){
  opt <-  -ef.main/(2*ef.main.sq)
  tol <- 1/sqrt(-2*ef.main.sq)
} else {opt=NA;tol= NA}
print(opt); print(tol)

score_vals <- matrix(NA, nrow = n_simul, ncol = 5);
colnames(score_vals)=c("fourth_corner","score", "mdscore", "LR",
"fourth_corner_ade4")

loglikPoi <- function(y,mu){sum(y*log(mu)+y-mu)}


# one trait one environmental variables Poisson data-----------------------------
----

for (i in 1:n_simul){
  T <- as.matrix(rnorm(n_species))
  E <- as.matrix(rnorm(n_communities))
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx,
```

```
b_zx_star), sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                                coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                                coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                                distribution = distri_Y)
  # give E and T a non-trivial mean and variance and remove any empty rows/columns
in Y
  obj <- make_obj_for_traitenv(Y,3+5*E,10+2*T)
  Y <- obj$L; E<-obj$E; T <- obj$T
  sumY <-sum(Y)
  fourth <- sumY * fourthcorner(obj)
  fourth_ade4 <- sumY *
(ade4::fourthcorner(as.data.frame(obj$E),as.data.frame(obj$L),as.data.frame(obj$T),
nrepet = 0)$tabD$obs)^2
  # score test statistics via library mdscore
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sumY
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, "T1.E1"])  # $Sr, $Sr_cor
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  modH1 <- glm(y~site+species+T1.E1, family = poisson, data = XYZ, mustart =
c(mu0))
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  score_vals[i,]= c(fourth,result$Sr,result$Sr_cor, LR, fourth_ade4)
}

max(abs(score_vals[,"fourth_corner"]-score_vals[,"fourth_corner_ade4"])) #<1.0e-10
max(abs(score_vals[,"fourth_corner"]-score_vals[,"score"])) #<1.0e-10
max(abs(score_vals[,"fourth_corner"]-score_vals[,"LR"]))    # ~0.25
max(abs(score_vals[,"score"]-score_vals[,"mdscore"]))       # ~0.05
max(abs(score_vals[,"LR"]-score_vals[,"mdscore"]))          # ~0.2

prob = (1:nrow(score_vals)) / nrow(score_vals)
xl = "chi-square based probability"; yl = "simulated significance"
x = pchisq(sort(score_vals[,"fourth_corner"], decreasing = TRUE), df = 1,
lower.tail = FALSE)
plot(x, prob, main ="score test: single trait & environment", xlab = xl, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")



# Multiple traits and multiple environmental variable --------------------------


n_trait <- 2  #q
n_env <- 3 # p

score_vals <- matrix(NA, nrow = n_simul, ncol = 6);
colnames(score_vals)=c("SB","score","dCCA","RLQ", "mdscore", "LR")


for (i in 1:n_simul){
  Tlist <- generate_Tdata(n_species, n_trait, correlation = cor_Tdat, b =
rep(1,n_trait))
  Tdat <- Tlist$Tdat; T <-Tlist$T; c.true <-Tlist$coefficients
  Elist <- generate_Edata(n_communities, n_env, correlation = cor_Edat, b =
rep(1,n_env))
  Edat <- Elist$Edat; E <-Elist$E; d.true <-Elist$coefficients
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx,
```

```
b_zx_star), sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                              coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                              coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                              distribution = distri_Y)
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  Y <- obj$L; E<-obj$E; T <- obj$T
  fourth <- sum(Y) * fourthcorner(obj)
  # score test statistics via library mdscore
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, 3+ (1:(n_trait*n_env))])  # $Sr, $Sr_cor
  SB <- score_test_ET(obj)
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  formulaH1 = as.formula(paste("y~", paste(c("site","species", names(XYZ)[3+
(1:(n_trait*n_env))]),collapse="+"), sep= ""))
  modH1 <- glm(formulaH1, family = poisson, data = XYZ, mustart = c(mu0))
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  #c("SB","score","dCCA","RLQ", "mdscore", "LR")
  score_vals[i,]= c(SB, result$Sr,fourth,result$Sr_cor, LR)
}

max(abs(score_vals[,"dCCA"]-score_vals[,"score"])) # <1.0e-10
max(abs(score_vals[,"SB"]-score_vals[,"score"]))   # <1.0e-10


prob = (1:nrow(score_vals)) / nrow(score_vals)
df = n_trait*n_env; df # df = 6
x = pchisq(sort(score_vals[,"score"], decreasing = TRUE), df = df,lower.tail =
FALSE)
plot(x, prob, main ="score test: multiple trait/environmental variables", xlab =
xl, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")



#  Traits and environmental variables are factors --------------------------

n_communities. <- 80 #n
n_species. <- 70 # m
nt = 3; ne = 4

score_vals <- matrix(NA, nrow = n_simul, ncol = 4);
colnames(score_vals)=c("chi_square_TE", "SB","dCCA","fourth_corner_ade4")

for (i in 1:n_simul){
  T <- as.matrix(rnorm(n_species.))
  E <- as.matrix(rnorm(n_communities.))
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx,
b_zx_star), sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                              coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                              coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                              distribution = distri_Y)
  # give E and T a non-trivial mean and variance and remove any empty rows/columns
in Y
  T_fact <- cut(T,nt)
  E_fact <- cut(E,ne)
  Tdat <- model.matrix(~0+T_fact, data= as.data.frame(T_fact))
```

```
  Edat <- model.matrix(~0+E_fact, data= as.data.frame(E_fact))
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  # calculate the usual chi-square statistic ( chi_squared_te) from the condensed
TE table (Y_te)
  Y <- obj$L; E<-obj$E; T <- obj$T
  Y_te <- t(E)%*% Y %*% T
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  mu0_te = rowSums(Y_te) %o% colSums(Y_te)/sum(Y_te)
  range(mu0_te - t(E)%*% mu0 %*% T) # <1.0e-10
  chi_squared_te <- sum((Y_te-mu0_te)^2 / mu0_te)
  # calculated using ade4
  Ef <- data.frame(E=factor(E %*% 1:ne));
  Tf <- data.frame(T=factor(T %*% 1:nt));
  res <- ade4::fourthcorner2(Ef,as.data.frame(Y),Tf,nrepet = 1)
  fourth_corner_factor <- sum(Y) * res$trRLQ$obs
  # calculate fourth corner and score in the usual way
  # (deleting the last category of the trait and the environmental factor)
  obj <- make_obj_for_traitenv(Y,Edat[,-ne],Tdat[,-nt])
  Y <- obj$L; E<-obj$E; T <- obj$T
  dCCA <- sum(Y) * fourthcorner(obj)["dCCA"]
  # score test statistics
  SB <- score_test_ET(obj)
  score_vals[i,]= c(chi_squared_te,SB, dCCA, fourth_corner_factor)
}
max(abs(score_vals[,"chi_square_TE"]-score_vals[,"SB"]))   # <1.0e-10
max(abs(score_vals[,"chi_square_TE"]-score_vals[,"dCCA"])) # <1.0e-10
max(abs(score_vals[,"chi_square_TE"]-score_vals[,"fourth_corner_ade4"])) #  <1.0e-
10


prob = (1:nrow(score_vals)) / nrow(score_vals)
df = (nt-1)*(ne-1); df
x = pchisq(sort(score_vals[,"chi_square_TE"], decreasing = TRUE), df =
df,lower.tail = FALSE)
plot(x, prob, main ="score test: trait and environment are both factors", xlab =
xl, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")



# Single trait and multiple environmental variables ----------------------

n_trait <- 1  #q
n_env <- 3 # p

score_vals <- matrix(NA, nrow = n_simul, ncol = 7);
colnames(score_vals)=c("score_via_CWM", "SB","score","dCCA","RLQ", "mdscore", "LR")



for (i in 1:n_simul){
  Tlist <- generate_Tdata(n_species, n_trait, correlation = cor_Tdat, b =
rep(1,n_trait))
  Tdat <- Tlist$Tdat; T <-Tlist$T; c.true <-Tlist$coefficients
  Elist <- generate_Edata(n_communities, n_env, correlation = cor_Edat, b =
rep(1,n_env))
  Edat <- Elist$Edat; E <-Elist$E; d.true <-Elist$coefficients
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx,
b_zx_star), sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                             coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                             coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                             distribution = distri_Y)
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
```

```
  Y <- obj$L; E<-obj$E; T <- obj$T
  fourth <- sum(Y) * fourthcorner(obj)
  # CWM: community weigthed trait avarage
  Wn <- rowSums(Y)
  Wm <- colSums(Y)
  T_c <- center_w(T, Wm/sum(Wm))
  # we could standardize here,
  # but also divide by the weighted variance later in line score_via_CWM
# T_std <- T_c / sqrt(sum(Wm*T_c*T_c))
  t_star <- diag(1/Wn)%*%Y%*% T_c # eqn 13
  lmt <- lm(t_star~E, weights = Wn)
  t_hat <- lmt$fitted
  score_via_CWM <- sum(Y) * sum(Wn*t_hat*t_hat)/sum(Wm*T_c*T_c)
  # score test statistics via library mdscore
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, 3+ (1:(n_trait*n_env))])  # $Sr, $Sr_cor
  SB <- score_test_ET(obj)
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  formulaH1 = as.formula(paste("y~", paste(c("site","species", names(XYZ)[3+
(1:(n_trait*n_env))]),collapse="+"), sep= ""))
  modH1 <- glm(formulaH1, family = poisson, data = XYZ, mustart = c(mu0))
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  score_vals[i,]= c(score_via_CWM,SB, result$Sr,fourth,result$Sr_cor, LR)
}

max(abs(score_vals[,"score_via_CWM"]-score_vals[,"score"])) # <1.0e-10

max(abs(score_vals[,"dCCA"]-score_vals[,"score"])) # <1.0e-10
max(abs(score_vals[,"SB"]-score_vals[,"score"]))   # <1.0e-10

prob = (1:nrow(score_vals)) / nrow(score_vals)
df = n_trait*n_env; df # df = 3
x = pchisq(sort(score_vals[,"score"], decreasing = TRUE), df = df,lower.tail =
FALSE)
plot(x, prob, main ="score test: single trait, multiple env", xlab = xl, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")



# Multiple traits and a single environmental variables ----------------------

n_trait <- 3  #q
n_env <- 1 # p


score_vals <- matrix(NA, nrow = n_simul, ncol = 7);
colnames(score_vals)=c("score_via_SNC", "SB","score","dCCA","RLQ", "mdscore", "LR")

for (i in 1:n_simul){
  Tlist <- generate_Tdata(n_species, n_trait, correlation = cor_Tdat, b =
rep(1,n_trait))
  Tdat <- Tlist$Tdat; T <-Tlist$T; c.true <-Tlist$coefficients
  Elist <- generate_Edata(n_communities, n_env, correlation = cor_Edat, b =
rep(1,n_env))
  Edat <- Elist$Edat; E <-Elist$E; d.true <-Elist$coefficients
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx,
```

```
b_zx_star), sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                              coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                              coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                              distribution = distri_Y)
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  Y <- obj$L; E<-obj$E; T <- obj$T
  fourth <- sum(Y) * fourthcorner(obj)
  # SNC: species niche centroid
  Wn <- rowSums(Y)
  Wm <- colSums(Y)
  E_c <- center_w(E, Wn/sum(Wn))
  # we could standardize here,
  # but also divide by the weighted variance later in line score_via_CWM
  # E_std <- E_c / sqrt(sum(Wn*E_c*E_c))
  e_star <- diag(1/Wm)%*%t(Y)%*% E_c # anologon of eqn 13
  lm_e <- lm(e_star~T, weights = Wm)
  e_hat <- lm_e$fitted
  score_via_SNC <- sum(Y) * sum(Wm*e_hat*e_hat)/sum(Wn*E_c*E_c)
  # score test statistics via library mdscore
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, 3+ (1:(n_trait*n_env))])  # $Sr, $Sr_cor
  SB <- score_test_ET(obj)
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  formulaH1 = as.formula(paste("y~", paste(c("site","species", names(XYZ)[3+
(1:(n_trait*n_env))]),collapse="+"), sep= ""))
  modH1 <- glm(formulaH1, family = poisson, data = XYZ, mustart = c(mu0))
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  score_vals[i,]= c(score_via_SNC,SB, result$Sr,fourth,result$Sr_cor, LR)
}

max(abs(score_vals[,"score_via_SNC"]-score_vals[,"score"])) # <1.0e-10
max(abs(score_vals[,"dCCA"]-score_vals[,"score"])) # <1.0e-10
max(abs(score_vals[,"SB"]-score_vals[,"score"]))   # <1.0e-10

prob = (1:nrow(score_vals)) / nrow(score_vals)
df = n_trait*n_env; df # df = 3
x = pchisq(sort(score_vals[,"score"], decreasing = TRUE), df = df,lower.tail =
FALSE)
plot(x, prob, main ="score test: multiple traits, single env", xlab = xl, ylab =
yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")
```

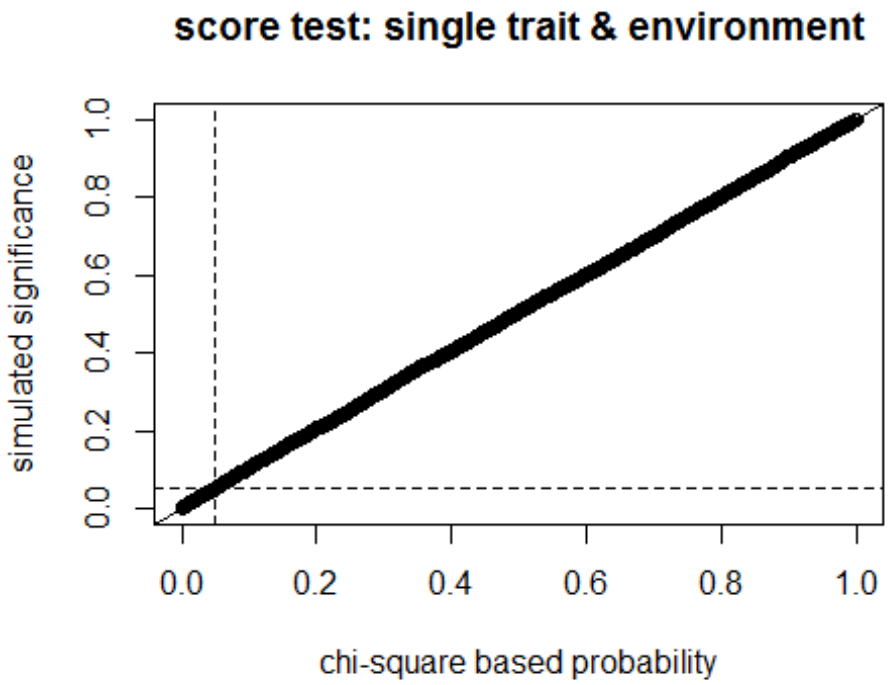
4. R-script with output

trait_env_score_test_examples1.R

```
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("generate_trait_env_community_data_loglinear.r")
source("te_score_functions1.r")
library(mdscore)

## Loading required package: MASS
```

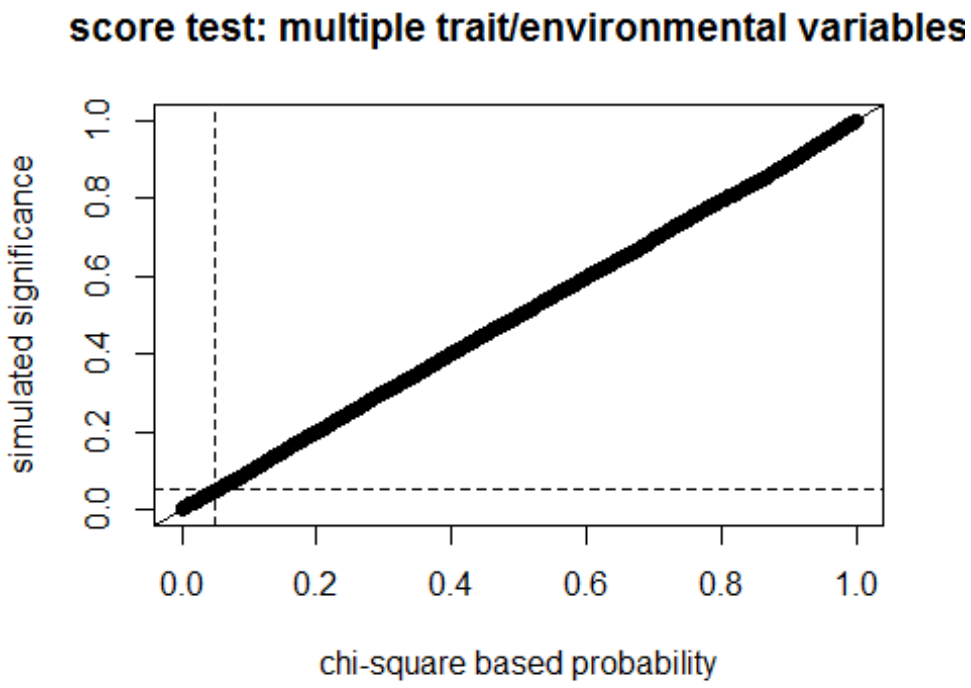
```r
library(ade4)

##
## Attaching package: 'ade4'
##
## The following object is masked _by_ '.GlobalEnv':
##
##     fourthcorner

set.seed(1353)

# Section 1 simplest log-linear simulation model b_tx = 0 ------------------------
--------------------

n_communities <- 5 #n
n_species <- 7 # m
n_trait <- 1  #q
n_env <- 1 # p
cor_Tdat <- 0.5
cor_Edat <- 0.5

n_simul <-  10000  #
distri_Y <- "poisson"  # distribution used for generating the response data Y given
E(Y)

mudat0 = 30
# effect sizes in log-linear model (b1,b2,b2 = b_te, b_ze, b_tx)
# with x and z random nuisance variables.
b_te <- 0   # the null model, without t-e interaction
b_ze <- 0
b_tx <- 0
b_zx <- 0
b_zx_star <- 0
sigma_epsilon_ij <- 0 # min(0.1, b_te + b_ze + b_tx)   # unstructured interaction


ef.main <- 0.2
ef.main.sq <-  0
ef.main.ran.rows <- 0
ef.main.ran.columns <- 0

if (ef.main.sq < 0){
  opt <-  -ef.main/(2*ef.main.sq)
  tol <- 1/sqrt(-2*ef.main.sq)
} else {opt=NA;tol= NA}
print(opt); print(tol)

## [1] NA

## [1] NA

score_vals <- matrix(NA, nrow = n_simul, ncol = 5);
colnames(score_vals)=c("fourth_corner","score", "mdscore", "LR", "fourth_corner_ade
4")

loglikPoi <- function(y,mu){sum(y*log(mu)+y-mu)}


# one trait one environmental variables Poisson data-----------------------------
----

for (i in 1:n_simul){
  T <- as.matrix(rnorm(n_species))
  E <- as.matrix(rnorm(n_communities))
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx, b_zx_star)
```

```
, sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                                  coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                                  coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                                  distribution = distri_Y)
  # give E and T a non-trivial mean and variance and remove any empty rows/columns
in Y
  obj <- make_obj_for_traitenv(Y,3+5*E,10+2*T)
  Y <- obj$L; E<-obj$E; T <- obj$T
  sumY <-sum(Y)
  fourth <- sumY * fourthcorner(obj)
  fourth_ade4 <- sumY * (ade4::fourthcorner(as.data.frame(obj$E),as.data.frame(obj$
L),as.data.frame(obj$T), nrepet = 0)$tabD$obs)^2
  # score test statistics via library mdscore
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sumY
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, "T1.E1"])  # $Sr, $Sr_cor
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  modH1 <- glm(y~site+species+T1.E1, family = poisson, data = XYZ, mustart = c(mu0)
)
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  score_vals[i,]= c(fourth,result$Sr,result$Sr_cor, LR, fourth_ade4)
}

max(abs(score_vals[,"fourth_corner"]-score_vals[,"fourth_corner_ade4"])) #<1.0e-10

## [1] 1.332268e-14

max(abs(score_vals[,"fourth_corner"]-score_vals[,"score"])) #<1.0e-10

## [1] 1.093559e-11

max(abs(score_vals[,"fourth_corner"]-score_vals[,"LR"]))    # ~0.25

## [1] 0.4660028

max(abs(score_vals[,"score"]-score_vals[,"mdscore"]))       # ~0.05

## [1] 0.06877659

max(abs(score_vals[,"LR"]-score_vals[,"mdscore"]))          # ~0.2

## [1] 0.4898645

prob = (1:nrow(score_vals)) / nrow(score_vals)
xl = "chi-square based probability"; yl = "simulated significance"
x = pchisq(sort(score_vals[,"fourth_corner"], decreasing = TRUE), df = 1, lower.tai
l = FALSE)
plot(x, prob, main ="score test: single trait & environment", xlab = xl, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")
```



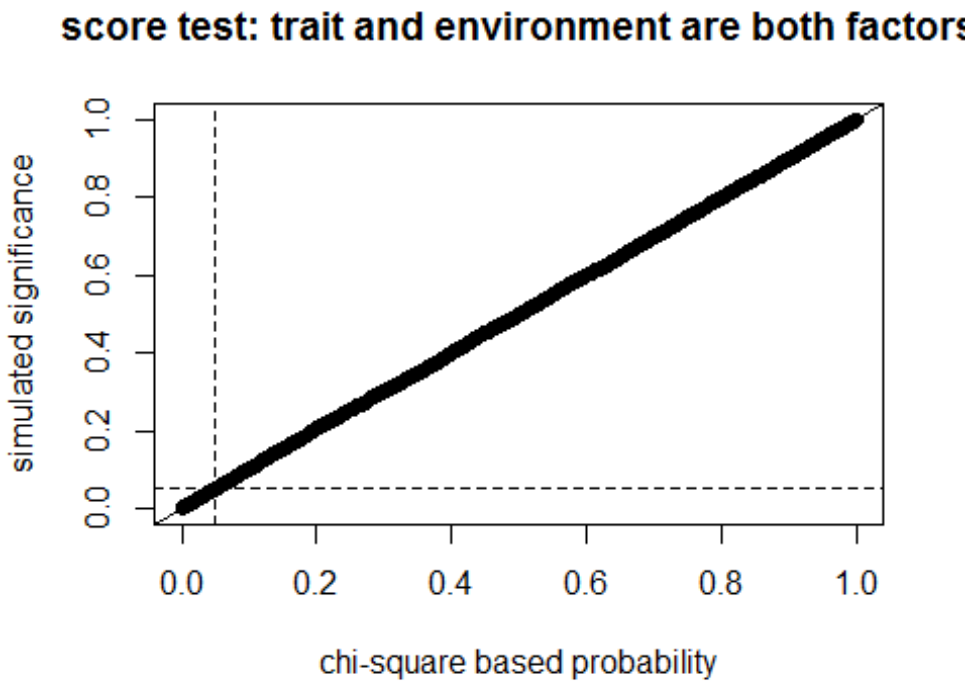

```
# Multiple traits and multiple environmental variable --------------------------


n_trait <- 2  #q
n_env <- 3 # p

score_vals <- matrix(NA, nrow = n_simul, ncol = 6);
colnames(score_vals)=c("SB","score","dCCA","RLQ", "mdscore", "LR")


for (i in 1:n_simul){
  Tlist <- generate_Tdata(n_species, n_trait, correlation = cor_Tdat, b = rep(1,n_t
rait))
  Tdat <- Tlist$Tdat; T <-Tlist$T; c.true <-Tlist$coefficients
  Elist <- generate_Edata(n_communities, n_env, correlation = cor_Edat, b = rep(1,n
_env))
  Edat <- Elist$Edat; E <-Elist$E; d.true <-Elist$coefficients
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx, b_zx_star)
, sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                             coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                             coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                             distribution = distri_Y)
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  Y <- obj$L; E<-obj$E; T <- obj$T
  fourth <- sum(Y) * fourthcorner(obj)
  # score test statistics via library mdscore
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, 3+ (1:(n_trait*n_env))])  # $Sr, $Sr_cor
  SB <- score_test_ET(obj)
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  formulaH1 = as.formula(paste("y~", paste(c("site","species", names(XYZ)[3+ (1:(n_
trait*n_env))]),collapse="+"), sep= ""))
  modH1 <- glm(formulaH1, family = poisson, data = XYZ, mustart = c(mu0))
  #   # LR via glm
```

```
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  #c("SB","score","dCCA","RLQ", "mdscore", "LR")
  score_vals[i,]= c(SB, result$Sr,fourth,result$Sr_cor, LR)
}

max(abs(score_vals[,"dCCA"]-score_vals[,"score"])) # <1.0e-10

## [1] 2.25775e-11

max(abs(score_vals[,"SB"]-score_vals[,"score"]))   # <1.0e-10

## [1] 7.389467e-11

prob = (1:nrow(score_vals)) / nrow(score_vals)
df = n_trait*n_env; df # df = 6

## [1] 6

x = pchisq(sort(score_vals[,"score"], decreasing = TRUE), df = df,lower.tail = FALS
E)
plot(x, prob, main ="score test: multiple trait/environmental variables", xlab = xl
, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")
```



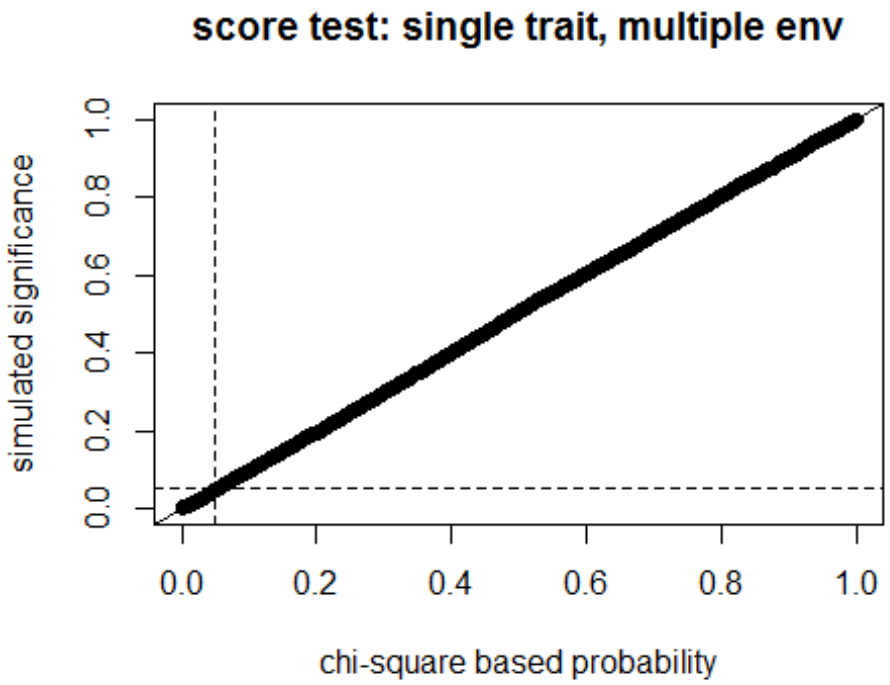

```
#  Traits and environmental variables are factors ---------------------------

n_communities. <- 80 #n
n_species. <- 70 # m
nt = 3; ne = 4

score_vals <- matrix(NA, nrow = n_simul, ncol = 4);
colnames(score_vals)=c("chi_square_TE", "SB","dCCA","fourth_corner_ade4")
```

```r
for (i in 1:n_simul){
  T <- as.matrix(rnorm(n_species.))
  E <- as.matrix(rnorm(n_communities.))
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx, b_zx_star)
, sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                             coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                             coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                             distribution = distri_Y)
  # give E and T a non-trivial mean and variance and remove any empty rows/columns
in Y
  T_fact <- cut(T,nt)
  E_fact <- cut(E,ne)
  Tdat <- model.matrix(~0+T_fact, data= as.data.frame(T_fact))
  Edat <- model.matrix(~0+E_fact, data= as.data.frame(E_fact))
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  # calculate the usual chi-square statistic ( chi_squared_te) from the condensed T
E table (Y_te)
  Y <- obj$L; E<-obj$E; T <- obj$T
  Y_te <- t(E)%*% Y %*% T
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  mu0_te = rowSums(Y_te) %o% colSums(Y_te)/sum(Y_te)
  range(mu0_te - t(E)%*% mu0 %*% T) # <1.0e-10
  chi_squared_te <- sum((Y_te-mu0_te)^2 / mu0_te)
  # calculated using ade4
  Ef <- data.frame(E=factor(E %*% 1:ne));
  Tf <- data.frame(T=factor(T %*% 1:nt));
  res <- ade4::fourthcorner2(Ef,as.data.frame(Y),Tf,nrepet = 1)
  fourth_corner_factor <- sum(Y) * res$trRLQ$obs
  # calculate fourth corner and score in the usual way
  # (deleting the last category of the trait and the environmental factor)
  obj <- make_obj_for_traitenv(Y,Edat[,-ne],Tdat[,-nt])
  Y <- obj$L; E<-obj$E; T <- obj$T
  dCCA <- sum(Y) * fourthcorner(obj)["dCCA"]
  # score test statistics
  SB <- score_test_ET(obj)
  score_vals[i,]= c(chi_squared_te,SB, dCCA, fourth_corner_factor)
}
max(abs(score_vals[,"chi_square_TE"]-score_vals[,"SB"]))   # <1.0e-10

## [1] 5.52447e-13

max(abs(score_vals[,"chi_square_TE"]-score_vals[,"dCCA"])) # <1.0e-10

## [1] 1.509903e-13

max(abs(score_vals[,"chi_square_TE"]-score_vals[,"fourth_corner_ade4"])) #  <1.0e-1
0

## [1] 5.329071e-15

prob = (1:nrow(score_vals)) / nrow(score_vals)
df = (nt-1)*(ne-1); df

## [1] 6

x = pchisq(sort(score_vals[,"chi_square_TE"], decreasing = TRUE), df = df,lower.tai
l = FALSE)
plot(x, prob, main ="score test: trait and environment are both factors", xlab = xl
, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")
```



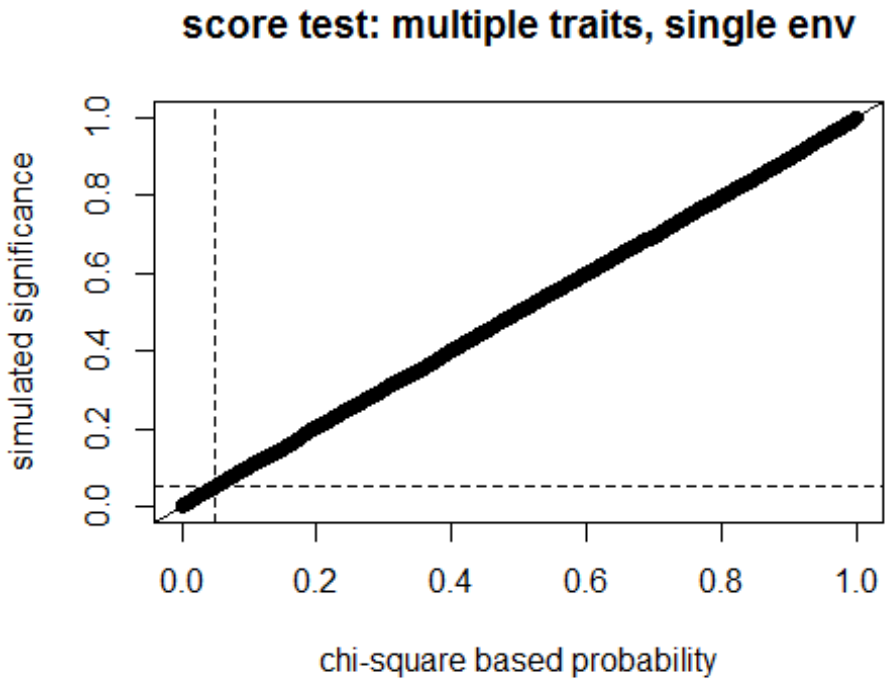

```
# Single trait and multiple environmental variables ----------------------

n_trait <- 1  #q
n_env <- 3 # p

score_vals <- matrix(NA, nrow = n_simul, ncol = 7);
colnames(score_vals)=c("score_via_CWM", "SB","score","dCCA","RLQ", "mdscore", "LR")



for (i in 1:n_simul){
  Tlist <- generate_Tdata(n_species, n_trait, correlation = cor_Tdat, b = rep(1,n_t
rait))
  Tdat <- Tlist$Tdat; T <-Tlist$T; c.true <-Tlist$coefficients
  Elist <- generate_Edata(n_communities, n_env, correlation = cor_Edat, b = rep(1,n
_env))
  Edat <- Elist$Edat; E <-Elist$E; d.true <-Elist$coefficients
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx, b_zx_star)
, sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                             coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                             coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                             distribution = distri_Y)
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  Y <- obj$L; E<-obj$E; T <- obj$T
  fourth <- sum(Y) * fourthcorner(obj)
  # CWM: community weigthed trait avarage
  Wn <- rowSums(Y)
  Wm <- colSums(Y)
  T_c <- center_w(T, Wm/sum(Wm))
  # we could standardize here,
  # but also divide by the weighted variance later in line score_via_CWM
# T_std <- T_c / sqrt(sum(Wm*T_c*T_c))
  t_star <- diag(1/Wn)%*%Y%*% T_c # eqn 13
  lmt <- lm(t_star~E, weights = Wn)
  t_hat <- lmt$fitted
  score_via_CWM <- sum(Y) * sum(Wn*t_hat*t_hat)/sum(Wm*T_c*T_c)
  # score test statistics via library mdscore
```

```
  # H0 via row*col
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, 3+ (1:(n_trait*n_env))])  # $Sr, $Sr_cor
  SB <- score_test_ET(obj)
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  formulaH1 = as.formula(paste("y~", paste(c("site","species", names(XYZ)[3+ (1:(n_
trait*n_env))]),collapse="+"), sep= ""))
  modH1 <- glm(formulaH1, family = poisson, data = XYZ, mustart = c(mu0))
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  score_vals[i,]= c(score_via_CWM,SB, result$Sr,fourth,result$Sr_cor, LR)
}

max(abs(score_vals[,"score_via_CWM"]-score_vals[,"score"])) # <1.0e-10

## [1] 7.285195e-11

max(abs(score_vals[,"dCCA"]-score_vals[,"score"])) # <1.0e-10

## [1] 7.279866e-11

max(abs(score_vals[,"SB"]-score_vals[,"score"]))   # <1.0e-10

## [1] 6.793677e-11

prob = (1:nrow(score_vals)) / nrow(score_vals)
df = n_trait*n_env; df # df = 3

## [1] 3

x = pchisq(sort(score_vals[,"score"], decreasing = TRUE), df = df,lower.tail = FALS
E)
plot(x, prob, main ="score test: single trait, multiple env", xlab = xl, ylab = yl)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")
```




```
# Multiple traits and a single environmental variables ----------------------

n_trait <- 3  #q
n_env <- 1 # p


score_vals <- matrix(NA, nrow = n_simul, ncol = 7);
colnames(score_vals)=c("score_via_SNC", "SB","score","dCCA","RLQ", "mdscore", "LR")

for (i in 1:n_simul){
  Tlist <- generate_Tdata(n_species, n_trait, correlation = cor_Tdat, b = rep(1,n_t
rait))
  Tdat <- Tlist$Tdat; T <-Tlist$T; c.true <-Tlist$coefficients
  Elist <- generate_Edata(n_communities, n_env, correlation = cor_Edat, b = rep(1,n
_env))
  Edat <- Elist$Edat; E <-Elist$E; d.true <-Elist$coefficients
  Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx, b_zx_star)
, sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                             coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                             coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                             distribution = distri_Y)
  obj <- make_obj_for_traitenv(Y,Edat,Tdat)
  Y <- obj$L; E<-obj$E; T <- obj$T
  fourth <- sum(Y) * fourthcorner(obj)
  # SNC: species niche centroid
  Wn <- rowSums(Y)
  Wm <- colSums(Y)
  E_c <- center_w(E, Wn/sum(Wn))
  # we could standardize here,
  # but also divide by the weighted variance later in line score_via_CWM
  # E_std <- E_c / sqrt(sum(Wn*E_c*E_c))
  e_star <- diag(1/Wm)%*%t(Y)%*% E_c # analogon of eqn 36
  lm_e <- lm(e_star~T, weights = Wm)
  e_hat <- lm_e$fitted
  score_via_SNC <- sum(Y) * sum(Wm*e_hat*e_hat)/sum(Wn*E_c*E_c)
  # score test statistics via library mdscore
  # H0 via row*col
```

```
  mu0 = rowSums(Y) %o% colSums(Y)/sum(Y)
  XYZ <- expand4glm(obj) # data frame XYZ
  modH0 <- glm(y~site+species, family = poisson, data = XYZ, mustart = c(mu0))
  result = mdscore(modH0, X1=XYZ[, 3+ (1:(n_trait*n_env))])  # $Sr, $Sr_cor
  SB <- score_test_ET(obj)
  devH0 = -2*(loglikPoi(Y,mu0)-loglikPoi(Y,Y))
  formulaH1 = as.formula(paste("y~", paste(c("site","species", names(XYZ)[3+ (1:(n_
trait*n_env))]),collapse="+"), sep= ""))
  modH1 <- glm(formulaH1, family = poisson, data = XYZ, mustart = c(mu0))
  #   # LR via glm
  #   range(modH0$fitted.values -mu0)
  #   deviance(modH0)
  # LR <-  deviance(modH0) - deviance(modH1)
  LR <-   devH0 - deviance(modH1)
  score_vals[i,]= c(score_via_SNC,SB, result$Sr,fourth,result$Sr_cor, LR)
}

max(abs(score_vals[,"score_via_SNC"]-score_vals[,"score"])) # <1.0e-10

## [1] 2.307488e-12

max(abs(score_vals[,"dCCA"]-score_vals[,"score"])) # <1.0e-10

## [1] 2.314593e-12

max(abs(score_vals[,"SB"]-score_vals[,"score"]))   # <1.0e-10

## [1] 2.275513e-12

prob = (1:nrow(score_vals)) / nrow(score_vals)
df = n_trait*n_env; df # df = 3

## [1] 3

x = pchisq(sort(score_vals[,"score"], decreasing = TRUE), df = df,lower.tail = FALS
E)
plot(x, prob, main ="score test: multiple traits, single env", xlab = xl, ylab = yl
)
abline(a= 0, b=1)
abline(v=0.05, lty = "dashed")
abline(h=0.05, lty = "dashed")
```



5. References

da Silva-Junior, A. H. M., D. N. da Silva, and S. L. P. Ferrari. 2015. mdscore: An R Package to Compute Improved Score Tests in Generalized Linear Models. http://www.jstatsoft.org/v61/c02/. Journal of Statistical Software **61**:1-16. DOI: 10.18637/jss.v18061.c18602

Dray, S., and A. B. Dufour. 2007. The ade4 package: implementing the duality diagram for ecologists. Journal of Statistical Software **22**:1-20.