New robust weighted averaging- and model-based methods for assessing trait-environment relationships.

Cajo J. F. ter Braak

Appendix

1. R functions and scripts

This appendix gives R functions, code and output (by Rmarkdown) supporting the main conclusions of the paper.

The main scripts are

- AnalyseData.r
- DoSimulations.r

which call functions that are in the files

- WAregressionsMLM_rand_max_test.r
- PowerSimulation_functions.r

in the subfolder Rfunctions.

The script AnalyseData.r provides the numerical results of the Dataset sections of the main text. The script DoSimulations.r allows to see the essentials of the simulation study. With patience (computer time!), the complete study can be redone with slight adaptations of the code.

1.1 Functions in WAregressionsMLM_rand_max_test.r

The key functions are:

CWMSNC_regressions - Weighted averaging (WA) regressions (CWM/SNC regressions) with and without N_2- or RK- weighting (R=row total, K=column total; R and K are the weights in the fourth-corner correlation).

PermutationTest_r_c - max test by randomizing sites and species; the function differs from pmax_PermutationTest (ter Braak et al. 2017 PeerJ) (from which is adapted) by allowing different test statistics for the site-level and species-level test; the test statistic(s) must be specified by the function argument `FUN_test_statistics`. The default function is `CWMSNC_N2` which gives the reciprocal of the p values of the N_2- weighted WA regression as test statistics. Options are set by the argument `options = setoptions4pmax_test()`.

Three utility functions to make trait-environment data suitable for GLMM, which are used to fit the MLM models to data:

make_obj_for_traitenv - forms an object of class `TE_obj` (Trait-Environment object), with testing for empty sites and species

expand4glmm - function to change a `TE_obj` into a dataframe for use in glm and glmm (MLM); suitable for multiple traits and environmental variables

dat4MLM2TE_obj - the reverse of expand4glmm (single trait and single env variable only

There are some more functions for internal usage. The full code is as follows:

WAregressionsMLM_rand_max_test.r

```
[...]

CWMSNC_regressions<-function(E,L=0,T=0, weighing = 2, cutoff = 0){
# combination of CWM- and SNC-based regressions, possibly weighted.
# @param E  object from class TE_obj from make_obj_for_traitenv(L,E,T,cutoff) or n-vector or n
x1 matrix with the environmental values
# @param L  n x m matrix of abundance values or nonspecified if E = object from function make_
obj_for_traitenv(L,E,T,cutoff)
# @param T  m-vector or m x 1 matrix with trait values or nonspecified if E = object from func
tion make_obj_for_traitenv(L,E,T,cutoff)
# @ weighing = 0: no weights
#          = 1 or "RK"  weights are the row and columns totals of L
#          = 2 or "N2" weights are the Hill N2-effective numbers
#               (effective number of occurrences of a species and N2-diversity of a site)
# @cutoff  if E is not a TE_obj, cutoff on minimal number of occurrences of a species

  if(class(E)[1]!="TE_obj"){
    obj <- make_obj_for_traitenv(L,E,T,cutoff)
  } else obj <- E
  result <- with(obj,{
      R <- rowSums(L) # the site totals
      K <- colSums(L) # the species totals
      if (weighing == 0 ){
        wsites <- NULL
        wspecies <-NULL
      } else if (weighing %in% c(1,"RK")){
        wsites <- R
        wspecies <-K
      } else if (weighing %in% c(2,"N2")){
        N2_si <- apply(L, 1, function(x){x <- x/sum(x); 1/sum(x*x)})
        N2_sp <- apply(L, 2, function(x){x <- x/sum(x); 1/sum(x*x)})
        wsites <- N2_si
        wspecies <-N2_sp
      }
      CWM <- (L%*%T)/R #Community weighted mean wrt to trait T (mean trait value per site)
      SNC <- (t(L)%*%E)/K  # Species niche centroid wrt to environmental variable E (mean envi
ronmental value per species)
      # site-level regression of CWM on E
      lm_CWMe <- lm(CWM~E, weights = wsites)
      anova_site <- anova(lm_CWMe)
```

```
      F_sites <- anova_site$`F value`[1]# get F-value and p-value of site-level test
      Prob.site<- anova_site$"Pr(>F)"[1]
      # species-level regression of SNC on T
      lm_SNCt <- lm(SNC~T, weights =   wspecies)
      anova_species <- anova(lm_SNCt)
      F_species <- anova_species$`F value`[1]# get F-value and P-value of species-level test
      Prob.species<- anova_species$"Pr(>F)"[1]
      result<-list(p_values = c(p.site.prmtrc=Prob.site,p.species.prmtrc=Prob.species,p.max.pr
mtrc=max(Prob.site,Prob.species)), F_values =  c(F_sites = F_sites, F_species= F_species),
                   lm_CWMe = lm_CWMe, lm_SNCt = lm_SNCt, CWM = CWM, SNC = SNC, E = E, T = T, L
= L, wsites = wsites, wspecies=wspecies,weighing=weighing,cutoff=cutoff)
    return(result)
    }
  )
  return(result)
}


make_obj_for_traitenv <- function(L,E1,T1, cut_off = 0, forBootstrap = FALSE){
  # 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
  #' @param cut_off species occurring strictly less then cut_off are deleted
  #' @param forBootstrap adapts the row and column names of L to site1.. siten and species1 ..
speciesm
  #'                                so that they are unique (which they are not in a bootstra
p of L)
  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))|| forBootstrap) rownames(L)<- paste("site", 1:nrow(L), sep="")
  if(is.null(colnames(L))|| forBootstrap) colnames(L)<- paste("spec",1:ncol(L),sep="")
  rownames(E1)= rownames(L); if(is.null(colnames(E1))) colnames(E1)=  paste("E", 1:ncol(E1), s
ep = "")
  rownames(T1)= colnames(L); if(is.null(colnames(T1)))colnames(T1)=  paste("T", 1:ncol(T1), se
p = "")
  obj = list(L=L, E=E1,T = T1)
  class(obj) = c("TE_obj", class(obj))
  return(obj)
}

setoptions4pmax_test <- function(verbose = TRUE, falseSize = FALSE, alpha = 0.05, test_statist
```

```
ics_are_1divp = c(TRUE)){
# set options for use in PermutationTest_r_c
  list(verbose = verbose, falseSize = falseSize, alpha = alpha, test_statistics_are_1divp = te
st_statistics_are_1divp )
}


CWMSNC_N2 <- function(obj, ...){
# function for use in PermutationTest_r_c
  #FF <- CWMSNC_regressions(obj)$F_values;
  # equivalent, but handier as it shows 1/p in result of PermutationTest_r_c :
  FF <- 1/CWMSNC_regressions(obj)$p_values[-3]
  FF <- cbind(FF); colnames(FF)<- "N2-weighted lm";
  rownames(FF)<- c("teststat.site","teststat.species")
  return(FF)
}


PermutationTest_r_c <- function(obj, FUN_test_statistics=CWMSNC_N2, nrepet = 199, options = se
toptions4pmax_test(),  ...) {
  # function to determine randomization p-values based on test statistics for rows and columns
  # based on row and column based permutations of (possibly model-based) test-statistics,
  # such as the anova  F-value or chisq-value or equivalently, 1/p-values
  # The function examimes exceedance and takes the absolute value of the test-statistics, that
explains why 1/p-value can be used.
  #
  # @param obj object of class TE_obj (from make_obj_for_traitenv)
  # @param FUN_test_statistics  function that returns one or more test statistics (value, vect
or or matrix with one of two rows)
  #                         it should return 1/pval, because the
  #
  #
  # The value of FUN_test_statistics should have
  #   2 rows (containing row and column test statistics) and K named columns, with K number of
methods used in FUN)
  #     the column names should be the name of the test
  # obj from make_obj_for_traitenv(L,E,T, cut_off): matrices L,E and T (one environmental vari
able, 1-many traits)
  # e.g.FUN_test_statistic =
  # ... options for FUN_test_statistic
  # BEWARE: the size calculation is adapted from Miller et al 2018 MEE but
  # is only trustworthy if the null model holds true!!!!
  # For a non-null model the size is biased upwards (ter Braak, 2019 in prep.)
  obs <- FUN_test_statistics(obj,...)
  if (options$verbose) print(obs)
  if (length(options$test_statistics_are_1divp)==1) {
    options$test_statistics_are_1divp <- rep(options$test_statistics_are_1divp, ncol(obs))
  } else if (length(options$test_statistics_are_1divp)!=ncol(obs)){
    print(paste("Warning: length of options$test_statistics_are_1divp  (", length(options$test
_statistics_are_1divp) ,") is not equal to number of teststatistics: ",ncol(obs) ))
    options$test_statistics_are_1divp <- rep(options$test_statistics_are_1divp, ncol(obs))
  }
  sim.row <- matrix(0, nrow = nrepet, ncol = ncol(obs))
  sim.col <- matrix(0, nrow = nrepet, ncol = ncol(obs))

  for(i in 1:nrepet){
    # if (i/10 - round(i/10)==0)print(i)
    per.row <- sample(nrow(obj$L))
    per.col <- sample(ncol(obj$L))
    #permute_rows_columns <- function(obj, per.row,per.col){
    obj.row <- obj.col <-  obj  # obj.rc <- obj
```

```
    obj.row$E <-obj.row$E[per.row,,drop =FALSE]
    obj.col$T <-obj.col$T[per.col,,drop =FALSE]
    # sim.row[i, ] <- FUN_test_statistics(obj.row)[1,, drop = FALSE]
    # sim.col[i, ] <- FUN_test_statistics(obj.col)[2,,drop = FALSE]
    suppressWarnings(sim.row[i, ] <- FUN_test_statistics(obj.row,...)[1,, drop = FALSE])
    suppressWarnings(sim.col[i, ] <- FUN_test_statistics(obj.col,...)[2,,drop = FALSE])
    if (options$verbose && !i%%10) print(i)
  }
  ialpha <- 1/options$alpha
  if (ncol(obs)==1){
    isna.r <-  sum(is.na(sim.row))
    isna.c <-  sum(is.na(sim.col))
    pval.row <- (sum(abs(sim.row) >= abs(obs[1,]), na.rm = TRUE) + 1)  / (nrepet- isna.r  + 1)
    pval.col <- (sum(abs(sim.col) >= abs(obs[2,]), na.rm = TRUE) + 1)  / (nrepet- isna.c  + 1)
    if (options$falseSize){
      size.row <- (sum(sim.row >= ialpha, na.rm=TRUE))/ (nrepet - isna.r)
      size.col <- (sum(sim.col >= ialpha, na.rm=TRUE))/ (nrepet - isna.c)
      size.max <- (sum(max(sim.row,sim.col) >= ialpha, na.rm=TRUE))/ (nrepet - isna.c)
    }
  } else {
    obs.mat.row <- matrix(rep(abs(obs[1,]),each=nrepet), nrow= nrepet, ncol=ncol(obs))
    isna.r <-  colSums(is.na(sim.row))
    pval.row <- (colSums(abs(sim.row) >= obs.mat.row, na.rm=TRUE) + 1)/ (nrepet - isna.r + 1)
    isna.c <-  colSums(is.na(sim.col))
    obs.mat.col <- matrix(rep(abs(obs[2,]),each=nrepet), nrow= nrepet, ncol=ncol(obs))
    pval.col <- (colSums(abs(sim.col) >= obs.mat.col, na.rm=TRUE) + 1)/ (nrepet - isna.c + 1)
    if(options$falseSize){
      size.row <- (colSums(sim.row >= ialpha, na.rm=TRUE))/ (nrepet - isna.r)
      size.col <- (colSums(sim.col >= ialpha, na.rm=TRUE))/ (nrepet - isna.c)
      colnames(sim.row) = colnames(obs)
      colnames(sim.col) = colnames(obs)
    }
  }
  obs[options$test_statistics_are_1divp] <- 1/obs[options$test_statistics_are_1divp]
  result <- t(rbind(test_stat = obs, p.site.permut = pval.row, p.species.permut = pval.col, pm
ax.permut = pmax(pval.row, pval.col)))
  attr(result, "nrepet")<- nrepet
  if (options$falseSize) result <- cbind(result, t(rbind(size.row = size.row, size.col = size.
col)))
  return(list(p_values=result, nrepet = nrepet, obs = obs,sim.row=sim.row, sim.col = sim.col))
}

#  functions for glm and glmm (MLM)
#  expand the data in the obj (L,E,T) in vector form -----------------------
expand4glmm <- function(obj, K= 0){
  #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)
    if (ncol(Evec)==1) envnames <- "env" else if (is.null(colnames(E))) {
      envnames = paste("env", 1:ncol(Evec), sep = "")
    } else envnames <- colnames(E)
    if (ncol(Tvec)==1) traitnames <- "trait" else if (is.null(colnames(T))) {
      traitnames = paste("trait", 1:ncol(Tvec), sep = "")
    } else traitnames <- colnames(T)
    ET <- Rten2_with_names(Evec,Tvec)
    XYZ <- data.frame(y,site,species, obs = 1:length(y),Tvec, Evec, ET)
```

```
    if (K[1]>0) XYZ$y = cbind(y, K-y)
    return(XYZ) # XYZ notation of Jamil et al 2013
  })
}

dat4MLM2TE_obj<- function(dat, cut_off = 0){
# create abundance matrix TE_obj from a dataframe dat that is in standard order
#     first all data from site 1, then site 2,
#        with the all species listed in each site (so include 0 abundances) and
#        species having the same order in each site
# @param dat dataframe with names y, sites, species
# create abundance matrix L, environmental variable vector E and trait vector T
# BEWARE: currently works for a single trait and environmental variable only!!!!
  dat$site=factor(dat$site)
  dat$species=factor(dat$species)
  n_sites <-nlevels(factor(dat$site))
  n_species <-nlevels(dat$species)
  species <- dat$species[seq(from  = 1, by = n_sites, length.out = n_species)]
  sites <- dat$site[1:n_sites]
  # create the usual trait T, environment E and abundance matrix L;
  # these would normally the original data.. but are here derived from the existing data.frame
dat
  T <- matrix(dat$trait[seq(from  = 1, by = n_sites, length.out = n_species)], nrow = n_specie
s, dimnames = list(species, "trait"))
  E <- matrix(dat$env[1:n_sites], nrow = n_sites, dimnames = list(sites, "env"))
  if (length(dat$y)>nrow(dat)){ # e.g. with binomial response
    L <- matrix(dat$y[,1], nrow = n_sites,ncol = n_species, dimnames = list(sites= sites,speci
es=species))
  }
  else { # e.g. with count data
    L <- matrix(dat$y, nrow = n_sites,ncol = n_species, dimnames = list(sites= sites,species=s
pecies))
  }
  # check the data for empty species and sites and bring them in standard format of class TE_o
bj
  obj <- make_obj_for_traitenv(L = L,E1 = E,T1 = T, cut_off = cut_off)
}

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

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

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)
}
# End expand  ------------------------------------------------------------
```

1.2 Script AnalyseData.r

The script fits the WA-regressions and multilevel models to the Revisit data. It can be used to obtain the *p*-values in Table 1 of the main text. Note that these are Monte Carlo estimates, so independent simulations with the same number of repetitions (`nrepet`) will give slightly different answers.

The script also contains functions, some of which are contained in `Rfunctions/DoSimulations.r` described more fully in the section "Functions in DoSimulations".

```
#Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environment
relationships.
#Code used to analyze the Whittaker Siskiyou Mountain revisit dataset subset at
#  https://datadryad.org/resource/doi:10.5061/dryad.7gj0s3b
# using MLM2 and MLM3 with the glmmTMB library and betabinomial(logit) response
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("Rfunctions/WAregressionsMLM_rand_max_test.r")

# read and process data ------------------------------------------------

dat=read.csv("data/whittakerrevisitdata.csv")

## adapt dataframe dat for MLM analyses using glmmTMB or lme4
dat$site=factor(dat$site)
## create abundance matrix L, environmental variable vector E and trait vector T
n_sites <-nlevels(factor(dat$site))
n_species <-nlevels(dat$species)
species <- dat$species[seq(from  = 1, by = n_sites, length.out = n_species)]
sites <- dat$site[1:n_sites]
# create the usual trait T, environment E and abundance matrix L;
# these would normally the original data.. but are here derived from the existing data.frame
dat
T <- matrix(dat$trait[seq(from  = 1, by = n_sites, length.out = n_species)], nrow = n_species,
dimnames = list(species, "trait"))
E <- matrix(dat$env[1:n_sites], nrow = n_sites, dimnames = list(sites, "env"))
L <- matrix(dat$value, nrow = n_sites,ncol = n_species, dimnames = list(sites=
sites,species=species))
# check the data for empty species and sites and bring them in standard format
obj <- make_obj_for_traitenv(L,E,T, cut_off=0)
K  = 100
dat <- expand4glmm(obj, K = K)
# note that the formula s below should only contain names available in dat, thus:
names(dat)

# parametric max test using CWM/SNC regressions ------------------------------------------

# unweighted WA regressions == lm CWM/SNC
E_T0 = CWMSNC_regressions(E,L,T, weighing = 0)
```

```r
round(E_T0$p_values,5)
coef(E_T0$lm_CWMe)

# row-column totals (RK)-weighted WA regressions == RK-weighted lm
# The weights are as in Fourth Corner (FC); its permutations version equals FC
E_T1 = CWMSNC_regressions(E,L,T, weighing = 1)
round(E_T1$p_values,5)
coef(E_T1$lm_CWMe)

# N2-weigthed  WA regeressions == N2-weighted lm
E_T2 = CWMSNC_regressions(E,L,T, weighing = 2)
round(E_T2$p_values,5)
coef(E_T2$lm_CWMe)

# the default is N2-weighted regression
E_T = CWMSNC_regressions(E,L,T)
round(E_T$p_values,5)
coef(E_T$lm_CWMe)

# alternatively, perform checks only once by creating an object of class TE_obj
# using function make object of class TE_obj
obj <- make_obj_for_traitenv(L,E,T, cut_off=0)
E_Tobj = CWMSNC_regressions(obj)
round(E_Tobj$p_values,5)
coef(E_Tobj$lm_CWMe)
all.equal(E_Tobj, E_T)

# nonparametric max test using CWM/SNC regressions  ----------------------------------------


## define a function creating test statistics for use in PermutationTest_r_c
# its value must be a matrix with two rows:
#                first row contains sites-level test statistics,
#                second row contains species-level test statistcics
# the cbind in CWMSNC_N2 does the job. The ... is required.


set.seed(145)
# the default test is N2-weighted CWM/SNC regressions
timeN2 <- system.time(test_result <-PermutationTest_r_c(obj, nrepet = 19))
timeN2
names(test_result)
round(test_result$p_values, 5)

# all three weigthed regressions in one analysis
CWMSNC_RKN2Unw <- function(obj, invert_p_value = TRUE,  ...){
  FF <- matrix(c(CWMSNC_regressions(obj, weighing = 2)$p_values[-3],
              CWMSNC_regressions(obj, weighing = 1)$p_values[-3],
              CWMSNC_regressions(obj, weighing = 0)$p_values[-3]), nrow= 2);
  colnames(FF)<- c("N2-weighted lm","FC=RK-weighted lm","lm CWM/SNC");
  rownames(FF)<- c("teststat.site","teststat.species")
  if (invert_p_value) FF <- 1/FF
  return(FF)
}
# check
CWMSNC_RKN2Unw(obj,invert_p_value = FALSE)

timeWA199 <- system.time(test_result <-PermutationTest_r_c(obj, FUN_test_statistics =
CWMSNC_RKN2Unw, nrepet = 199, options = setoptions4pmax_test(verbose = FALSE)))
timeWA199
names(test_result)
```

```r
round(test_result$p_values,4)

# for graphs see DoWAplots.r

# parametric MLM analyses -------------------

library(glmmTMB)

formula.MLM3.linear <- y ~ env * trait  + (1 + env|species) + (1 + trait| site)
timeMLM3 <- system.time(MLM3 <- glmmTMB(formula.MLM3.linear, family = betabinomial,
data=dat))

timeMLM3
summary(MLM3)
pMLM3BBWald <- summary(MLM3)$coefficients$'cond'[4,4]
pMLM3BBWald

save(MLM3, file = "MLM3.Rdata")
# for graphs see DoMLM3plots.r

# MLM2
formula.MLM2.linear <- y ~ env * trait  + (1 + env|species) + (1 | site)
MLM2 <- glmmTMB(formula.MLM2.linear, family = betabinomial,  data=dat)
pMLM2BBWald <- summary(MLM2)$coefficients$'cond'[4,4]
pMLM2BBWald

pvalue.MLM3vMLM2 <- anova(MLM2,MLM3)$'Pr(>Chisq)'[2]
pvalue.MLM3vMLM2

# extension with polynomial main effects

formula.MLM3.quad <- y ~ poly(env,2) + poly(trait,2) +  env : trait  + (1 + env|species) + (1
+ trait| site)

timeMLM3.quad <- system.time(MLM3.quad <- update(MLM3,formula.MLM3.quad))
timeMLM3.quad

summary(MLM3.quad)
pvalue.MLM3quad <- anova(MLM3,MLM3.quad)$'Pr(>Chisq)'[2]
pvalue.MLM3quad

# nonparametric MLM3 max test -------------------

MLM_Wald_bte <- function(obj, formula, family,  K= 0, library = "glmmTMB", invert_p_value =
TRUE, verbose = FALSE, nAGQ =0,  ...){
#  function for use in PermutationTest_r_c
# @param obj object of class TE_obj (made by function make_obj_for_traitenv(L,E,T,cutoff))
# @formula model formula using names created by expand4glmm(obj, K = K)[use verbose = TRUE to
see these names]
# @param K is binomial total; if K[1]==0 data are not binomial or presence / absence
#                      K can be a scalar or vector of n*m (number of sites * number of
species)
# @param family glm family; use character form
# @param library use "glmmTMB" or else (e.g. "lme4")
# @param verbose If TRUE, prints the names for use in formula.
# @value test statistic: 1/pWald_b_te
  dat <- expand4glmm(obj, K = K)
  if (verbose) print(str(dat))
   if (library == "glmmTMB"){
     MLM <- glmmTMB(formula, data=dat, family= family)
     B <- summary(MLM)$coefficients$'cond'
```

```
  } else if  (library == "lme4") {
    MLM <- glmer(formula, data=dat, family= family,  nAGQ=nAGQ, control =
glmerControl(calc.derivs=F))
    B <- summary(MLM)$coefficients$'cond'
  } else {print(paste("library", library ,"not implemented"))}
# BEWARE the interaction coefficient is likely the last one.... adapt if not
  p_val_Wald <- B[nrow(B),4]
  if (verbose){
    print(summary(MLM))
    names(p_val_Wald) <- "p_val_Wald"
    print(p_val_Wald)
  }
  if (is.character(family) ) testnam <- paste(library, family, "MLMWald", sep = ".") else
testnam <- paste(library, "MLMWald", sep = ".")
  if (invert_p_value) test_stat <- 1/p_val_Wald else test_stat <- p_val_Wald
  test_stat <- matrix(test_stat, nrow = 2, ncol = 1, dimnames =list(c("rows","cols"),
testnam) )
  rownames(test_stat)<- c("teststat.site","teststat.species")
  if (verbose) print(test_stat)
  return(test_stat)
}

names(expand4glmm(obj, K = 100))
timeFUNMLM <- system.time(inv_pWald <- MLM_Wald_bte(obj, formula = formula.MLM3.linear,family
= "betabinomial", K = 100, verbose = TRUE))
timeFUNMLM
(p_Wald <- 1/inv_pWald)


# nonparametric WA regessions- and MLM3-max tests  -------------------


WA_MLM <- function(obj, formula.list, family.list,  K= 0, invert_p_value = TRUE,  model.names=
unlist(formula.list), library.list = list("glmmTMB"), verbose = FALSE, ...){
  # creates test statistics of the weighted averaging methods (WA) and MLM methods
  nlib <- length(library.list)
  nfam <- length(family.list)
  nfor <- length(formula.list)
  MLM_test <- NULL

  for (i in seq_along(formula.list)){
    MLM_testi <- MLM_Wald_bte(obj,
                              formula = formula.list[[min(c(i,nfor))]],
                              family= family.list[[min(c(i,nfam))]],  K= K, invert_p_value =
invert_p_value,
                              library.list = library.list[[min(c(i,nlib))]], verbose =
verbose)
    MLM_test <- cbind(MLM_test,MLM_testi)
  }
  colnames(MLM_test) <- rep(model.names, length(formula.list))[1:ncol(MLM_test)]
  WA <- CWMSNC_RKN2Unw(obj, invert_p_value = invert_p_value)
  return(cbind(MLM_test,WA))
}

nrepet <- 19 # number of random site permuations and random species permutations
timeFUNPerm_r_c <- system.time(test_result <-PermutationTest_r_c(obj,
                                  FUN_test_statistics = WA_MLM,
                                  model.names = c("MLM2BB", "MLM3BB"),
                                  formula.list =list(formula.MLM2.linear,formula.MLM3.linear),
                                  family.list = list("betabinomial"),
                                  library.list =c("glmmTMB"), K= 100, nrepet = nrepet))
```

```
timeFUNPerm_r_c
names(test_result)
test_result$p_values

save.image("AnalysesRevisit.Rdata")
```

1.3 Output of AnalyseData.r

In fitting the MLM2 and MLM3 model to the Revisit data with Betabinomial response distributions and logit link, the library glmmTMB gives numerous warnings. These come in two types:

```
## Warning in f(par, order = order, ...): value out of range in 'lgamma'

## Warning in fitTMB(TMBStruc): Model convergence problem; false convergence
## (8). See vignette('troubleshooting')
```

All results in the paper have been obtained ignoring these warnings. First of all, the parametric bootstrap resulted in estimates corresponding to the estimates of the real data, so that the parametric and bootstrap confidence intervals nearly coincided. Moreover, as an extra assurance, the results of the permutation test are not necessarily jeopardized by an unstable fitting process. An unstable fitting process might give a lower power but not an invalid estimate of the significance level. In a permutation test, the test statistic for the permutated data should be calculated exactly in the same way as the test statistic is calculated from the data, because, under the null hypothesis, the test statistic derived from the data follows the distribution of the test statistic in the permuted data. Under the null hypothesis, the data is simply another permuted (random) data set; that is why numbers of permutations are 19, 99, 499 instead of 20, 100, 500. So, if the test statistic is calculated in the same way from the randomly permutated data as from the observed data, the permutation test maintains the Type I error rate. Note that it is tempting to make the permutation test quicker by taking 'hot starts' of the fitting process (e.g. by specifying starting values), but this does jeopardize the validity of the test.

The permutation max test of 499 permutations using N2-weighted WA regressions takes about 5 CPU units. A single fit of the MLM3 model took 68 units (15 times longer).

The output also contains the summary of the fitted MLM3 model.

```
#Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environment relati
onships.
#Code used to analyze the Whittaker Siskiyou Mountain revisit dataset subset at
#  https://datadryad.org/resource/doi:10.5061/dryad.7gj0s3b
# using MLM2 and MLM3 with the glmmTMB library and betabinomial(logit) response
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("Rfunctions/WAregressionsMLM_rand_max_test.r")
```

```r
# read and process data ------------------------------------------------

dat=read.csv("data/whittakerrevisitdata.csv")

## adapt dataframe dat for MLM analyses using glmmTMB or lme4
dat$site=factor(dat$site)
## create abundance matrix L, environmental variable vector E and trait vector T
n_sites <-nlevels(factor(dat$site))
n_species <-nlevels(dat$species)
species <- dat$species[seq(from  = 1, by = n_sites, length.out = n_species)]
sites <- dat$site[1:n_sites]
# create the usual trait T, environment E and abundance matrix L;
# these would normally the original data.. but are here derived from the existing data.frame d
at
T <- matrix(dat$trait[seq(from  = 1, by = n_sites, length.out = n_species)], nrow = n_species,
dimnames = list(species, "trait"))
E <- matrix(dat$env[1:n_sites], nrow = n_sites, dimnames = list(sites, "env"))
L <- matrix(dat$value, nrow = n_sites,ncol = n_species, dimnames = list(sites= sites,species=s
pecies))
# check the data for empty species and sites and bring them in standard format
obj <- make_obj_for_traitenv(L,E,T, cut_off=0)
K  = 100
dat <- expand4glmm(obj, K = K)
# note that the formula s below should only contain names available in dat, thus:
names(dat)

## [1] "y"         "site"      "species"   "obs"       "trait"     "env"
## [7] "trait.env"

# parametric max test using CWM/SNC regressions ------------------------------------------

# unweighted WA regressions == lm CWM/SNC
E_T0 = CWMSNC_regressions(E,L,T, weighing = 0)
round(E_T0$p_values,5)

##    p.site.prmtrc p.species.prmtrc     p.max.prmtrc
##          0.00176          0.03476          0.03476

coef(E_T0$lm_CWMe)

## (Intercept)           E
##   1.1009611   0.1178507

# row-column totals (RK)-weighted WA regressions == RK-weighted lm
# The weights are as in Fourth Corner (FC); its permutations version equals FC
E_T1 = CWMSNC_regressions(E,L,T, weighing = 1)
round(E_T1$p_values,5)

##    p.site.prmtrc p.species.prmtrc     p.max.prmtrc
##          0.00111          0.00004          0.00111

coef(E_T1$lm_CWMe)

## (Intercept)           E
##   1.0944621   0.1216659

# N2-weigthed  WA regeressions == N2-weighted lm
E_T2 = CWMSNC_regressions(E,L,T, weighing = 2)
round(E_T2$p_values,5)

##    p.site.prmtrc p.species.prmtrc     p.max.prmtrc
##          0.00012          0.00006          0.00012
```

```r
coef(E_T2$lm_CWMe)
```

```
## (Intercept)           E
##   1.0498269   0.1636838
```

```r
# the default is N2-weighted regression
E_T = CWMSNC_regressions(E,L,T)
round(E_T$p_values,5)
```

```
##    p.site.prmtrc p.species.prmtrc     p.max.prmtrc
##          0.00012          0.00006          0.00012
```

```r
coef(E_T$lm_CWMe)
```

```
## (Intercept)           E
##   1.0498269   0.1636838
```

```r
# alternatively, perform checks only once by creating an object of class TE_obj
# using function make object of class TE_obj
obj <- make_obj_for_traitenv(L,E,T, cut_off=0)
E_Tobj = CWMSNC_regressions(obj)
round(E_Tobj$p_values,5)
```

```
##    p.site.prmtrc p.species.prmtrc     p.max.prmtrc
##          0.00012          0.00006          0.00012
```

```r
coef(E_Tobj$lm_CWMe)
```

```
## (Intercept)           E
##   1.0498269   0.1636838
```

```r
all.equal(E_Tobj, E_T)
```

```
## [1] TRUE
```

```r
# nonparametric max test using CWM/SNC regressions  ----------------------------------------


## define a function creating test statistics for use in PermutationTest_r_c
# its value must be a matrix with two rows:
#                first row contains sites-level test statistics,
#                second row contains species-level test statistcics
# the cbind in CWMSNC_N2 does the job. The ... is required.


set.seed(145)
# the default test is N2-weighted CWM/SNC regressions
timeN2 <- system.time(test_result <-PermutationTest_r_c(obj, nrepet = 19))
```

```
##                  N2-weighted lm
## teststat.site          8334.502
## teststat.species      16362.793
## [1] 10
```

```r
timeN2
```

```
##    user  system elapsed
##    0.23    0.00    0.23
```

```r
names(test_result)
```

```
## [1] "p_values" "nrepet"   "obs"      "sim.row"  "sim.col"
```

```r
round(test_result$p_values, 5)
```

```
##                 teststat.site teststat.species p.site.permut
## N2-weighted lm        0.00012            6e-05          0.05
##                 p.species.permut pmax.permut
## N2-weighted lm             0.05        0.05
## attr(,"nrepet")
## [1] 19
```

```
# all three weigthed regressions in one analysis
CWMSNC_RKN2Unw <- function(obj, invert_p_value = TRUE,  ...){
  FF <- matrix(c(CWMSNC_regressions(obj, weighing = 2)$p_values[-3],
              CWMSNC_regressions(obj, weighing = 1)$p_values[-3],
              CWMSNC_regressions(obj, weighing = 0)$p_values[-3]), nrow= 2);
  colnames(FF)<- c("N2-weighted lm","FC=RK-weighted lm","lm CWM/SNC");
  rownames(FF)<- c("teststat.site","teststat.species")
  if (invert_p_value) FF <- 1/FF
  return(FF)
}
# check
CWMSNC_RKN2Unw(obj,invert_p_value = FALSE)
```

```
##                   N2-weighted lm FC=RK-weighted lm  lm CWM/SNC
## teststat.site       1.199832e-04      1.106409e-03 0.001764364
## teststat.species    6.111426e-05      3.636984e-05 0.034760173
```

```
timeWA199 <- system.time(test_result <-PermutationTest_r_c(obj, FUN_test_statistics = CWMSNC_R
KN2Unw, nrepet = 199, options = setoptions4pmax_test(verbose = FALSE)))
timeWA199
```

```
##    user  system elapsed
##    4.99    0.00    5.04
```

```
names(test_result)
```

```
## [1] "p_values" "nrepet"   "obs"      "sim.row"  "sim.col"
```

```
round(test_result$p_values,4)
```

```
##                   teststat.site teststat.species p.site.permut
## N2-weighted lm           0.0001           0.0001          0.01
## FC=RK-weighted lm        0.0011           0.0000          0.01
## lm CWM/SNC               0.0018           0.0348          0.01
##                   p.species.permut pmax.permut
## N2-weighted lm               0.010       0.010
## FC=RK-weighted lm            0.065       0.065
## lm CWM/SNC                   0.055       0.055
## attr(,"nrepet")
## [1] 199
```

```
# for graphs see DoWAplots.r

# parametric MLM analyses -------------------
```

```
library(glmmTMB)
```

```
formula.MLM3.linear <- y ~ env * trait  + (1 + env|species) + (1 + trait| site)
timeMLM3 <- system.time(MLM3 <- glmmTMB(formula.MLM3.linear, family = betabinomial,  data=dat)
)
```

```
## Warning in f(par, order = order, ...): value out of range in 'lgamma'
[...]
```

```
## Warning in f(par, order = order, ...): value out of range in 'lgamma'
```

```
## Warning in fitTMB(TMBStruc): Model convergence problem; false convergence
## (8). See vignette('troubleshooting')

timeMLM3

##    user  system elapsed
##   67.76    0.46   68.32

summary(MLM3)

##  Family: betabinomial  ( logit )
## Formula:
## y ~ env * trait + (1 + env | species) + (1 + trait | site)
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##   6193.9   6262.9  -3086.0   6171.9     3889
##
## Random effects:
##
## Conditional model:
##  Groups  Name        Variance Std.Dev. Corr
##  species (Intercept) 2.1786   1.4760
##          env         0.1776   0.4214   0.41
##  site    (Intercept) 0.2255   0.4749
##          trait       0.1134   0.3367   -0.92
## Number of obs: 3900, groups:  species, 75; site, 52
##
## Overdispersion parameter for betabinomial family (): 3.35
##
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.42786    0.21826 -24.868  < 2e-16 ***
## env         -0.68238    0.12503  -5.458 4.82e-08 ***
## trait        1.00305    0.19788   5.069 4.00e-07 ***
## env:trait    0.25120    0.09583   2.621  0.00876 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

pMLM3BBWald <- summary(MLM3)$coefficients$'cond'[4,4]
pMLM3BBWald

## [1] 0.008756279

save(MLM3, file = "MLM3.Rdata")
# for graphs see DoMLM3plots.r

# MLM2
formula.MLM2.linear <- y ~ env * trait  + (1 + env|species) + (1 | site)
MLM2 <- glmmTMB(formula.MLM2.linear, family = betabinomial,  data=dat)

## Warning in f(par, order = order, ...): value out of range in 'lgamma'

[..]

## Warning in f(par, order = order, ...): value out of range in 'lgamma'

timeMLM3.quad

##    user  system elapsed
##   72.55    0.35   73.05

summary(MLM3.quad)
```

```
##  Family: betabinomial  ( logit )
## Formula:
## y ~ poly(env, 2) + poly(trait, 2) + (1 + env | species) + (1 +
##     trait | site) + env:trait
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##   6196.9   6278.4  -3085.4   6170.9     3887
##
## Random effects:
##
## Conditional model:
##  Groups  Name        Variance Std.Dev. Corr
##  species (Intercept) 2.1772   1.4755
##          env         0.1806   0.4249   0.43
##  site    (Intercept) 0.2260   0.4754
##          trait       0.1109   0.3330   -0.92
## Number of obs: 3900, groups:  species, 75; site, 52
##
## Overdispersion parameter for betabinomial family (): 3.35
##
## Conditional model:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -5.43000    0.21858 -24.842  < 2e-16 ***
## poly(env, 2)1   -43.12954    7.88765  -5.468 4.55e-08 ***
## poly(env, 2)2    -1.30393    3.46687  -0.376  0.70683
## poly(trait, 2)1  61.99530   12.35436   5.018 5.22e-07 ***
## poly(trait, 2)2  10.45304   10.95875   0.954  0.34016
## env:trait         0.25310    0.09592   2.639  0.00832 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
pvalue.MLM3quad <- anova(MLM3,MLM3.quad)$'Pr(>Chisq)'[2]
pvalue.MLM3quad
```

```
## [1] 0.5915321
```

```
# nonparametric MLM3 max test -------------------

MLM_Wald_bte <- function(obj, formula, family,  K= 0, library = "glmmTMB", invert_p_value = TR
UE, verbose = FALSE, nAGQ =0,  ...){
#  function for use in PermutationTest_r_c
# @param obj object of class TE_obj (made by function make_obj_for_traitenv(L,E,T,cutoff))
# @formula model formula using names created by expand4glmm(obj, K = K)[use verbose = TRUE to
see these names]
# @param K is binomial total; if K[1]==0 data are not binomial or presence / absence
#                      K can be a scalar or vector of n*m (number of sites * number of species
)
# @param family glm family; use character form
# @param library use "glmmTMB" or else (e.g. "lme4")
# @param verbose If TRUE, prints the names for use in formula.
# @value test statistic: 1/pWald_b_te
  dat <- expand4glmm(obj, K = K)
  if (verbose) print(str(dat))
   if (library == "glmmTMB"){
     MLM <- glmmTMB(formula, data=dat, family= family)
     B <- summary(MLM)$coefficients$'cond'
  } else if  (library == "lme4") {
    MLM <- glmer(formula, data=dat, family= family,  nAGQ=nAGQ, control = glmerControl(calc.de
rivs=F))
    B <- summary(MLM)$coefficients$'cond'
```

```
  } else {print(paste("library", library ,"not implemented"))}
# BEWARE the interaction coefficient is likely the last one.... adapt if not
  p_val_Wald <- B[nrow(B),4]
  if (verbose){
    print(summary(MLM))
    names(p_val_Wald) <- "p_val_Wald"
    print(p_val_Wald)
  }
  if (is.character(family) ) testnam <- paste(library, family, "MLMWald", sep = ".") else test
nam <- paste(library, "MLMWald", sep = ".")
  if (invert_p_value) test_stat <- 1/p_val_Wald else test_stat <- p_val_Wald
  test_stat <- matrix(test_stat, nrow = 2, ncol = 1, dimnames =list(c("rows","cols"), testnam)
)
  rownames(test_stat)<- c("teststat.site","teststat.species")
  if (verbose) print(test_stat)
  return(test_stat)
}

names(expand4glmm(obj, K = 100))

## [1] "y"         "site"      "species"   "obs"       "trait"     "env"
## [7] "trait.env"

timeFUNMLM <- system.time(inv_pWald <- MLM_Wald_bte(obj, formula = formula.MLM3.linear,family
= "betabinomial", K = 100, verbose = TRUE))

## 'data.frame':    3900 obs. of  7 variables:
##  $ y        : num [1:3900, 1:2] 0 0 0 25 7 4 4 0 0 0 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : NULL
##   .. ..$ : chr  "y" ""
##  $ site     : Factor w/ 52 levels "109","113","12",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ species  : Factor w/ 75 levels "Abies concolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ obs      : int  1 2 3 4 5 6 7 8 9 10 ...
##  $ trait    : num  1.4 1.4 1.4 1.4 1.4 ...
##  $ env      : num  -0.351 -0.774 1.667 1.016 1.342 ...
##  $ trait.env: num  -0.493 -1.087 2.342 1.427 1.885 ...
## NULL

## Warning in f(par, order = order, ...): value out of range in 'lgamma'

[..]

## Warning in f(par, order = order, ...): value out of range in 'lgamma'

## Warning in fitTMB(TMBStruc): Model convergence problem; false convergence
## (8). See vignette('troubleshooting')

##  Family: betabinomial  ( logit )
## Formula:
## y ~ env * trait + (1 + env | species) + (1 + trait | site)
## Data: dat
##
##      AIC      BIC   logLik deviance df.resid
##   6193.9   6262.9  -3086.0   6171.9     3889
##
## Random effects:
##
## Conditional model:
##  Groups  Name        Variance Std.Dev. Corr
##  species (Intercept) 2.1786   1.4760
##          env         0.1776   0.4214   0.41
```

```
##  site    (Intercept) 0.2255   0.4749
##          trait       0.1134   0.3367   -0.92
## Number of obs: 3900, groups:  species, 75; site, 52
##
## Overdispersion parameter for betabinomial family (): 3.35
##
## Conditional model:
##             Estimate Std. Error z value Pr(>|z|)
## (Intercept) -5.42786    0.21826 -24.868  < 2e-16 ***
## env         -0.68238    0.12503  -5.458 4.82e-08 ***
## trait        1.00305    0.19788   5.069 4.00e-07 ***
## env:trait    0.25120    0.09583   2.621  0.00876 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##  p_val_Wald
## 0.008756279
##                   glmmTMB.betabinomial.MLMWald
## teststat.site                        114.2038
## teststat.species                     114.2038

timeFUNMLM

##    user  system elapsed
##   66.53    0.37   66.97

(p_Wald <- 1/inv_pWald)

##                   glmmTMB.betabinomial.MLMWald
## teststat.site                     0.008756279
## teststat.species                  0.008756279

# nonparametric WA regessions- and MLM3-max tests  -------------------


WA_MLM <- function(obj, formula.list, family.list,  K= 0, invert_p_value = TRUE,  model.names=
unlist(formula.list), library.list = list("glmmTMB"), verbose = FALSE, ...){
  # creates test statistics of the weighted averaging methods (WA) and MLM methods
  nlib <- length(library.list)
  nfam <- length(family.list)
  nfor <- length(formula.list)
  MLM_test <- NULL

  for (i in seq_along(formula.list)){
    MLM_testi <- MLM_Wald_bte(obj,
                              formula = formula.list[[min(c(i,nfor))]],
                              family= family.list[[min(c(i,nfam))]],  K= K, invert_p_value = i
nvert_p_value,
                              library.list = library.list[[min(c(i,nlib))]], verbose = verbose
)
    MLM_test <- cbind(MLM_test,MLM_testi)
  }
  colnames(MLM_test) <- rep(model.names, length(formula.list))[1:ncol(MLM_test)]
  WA <- CWMSNC_RKN2Unw(obj, invert_p_value = invert_p_value)
  return(cbind(MLM_test,WA))
}

nrepet <- 19 # number of random site permuations and random species permutations
timeFUNPerm_r_c <- system.time(test_result <-PermutationTest_r_c(obj,
                                  FUN_test_statistics = WA_MLM,
                                  model.names = c("MLM2BB", "MLM3BB"),
                                  formula.list =list(formula.MLM2.linear,formula.MLM3.linear),
```

```
                                family.list = list("betabinomial"),
                                library.list =c("glmmTMB"), K= 100, nrepet = nrepet))

## Warning in f(par, order = order, ...): value out of range in 'lgamma'

[...]

## Warning in f(par, order = order, ...): value out of range in 'lgamma'

## Warning in fitTMB(TMBStruc): Model convergence problem; false convergence
## (8). See vignette('troubleshooting')

##                    MLM2BB   MLM3BB N2-weighted lm FC=RK-weighted lm
## teststat.site    472.7057 114.2038       8334.502           903.825
## teststat.species 472.7057 114.2038      16362.793         27495.308
##                   lm CWM/SNC
## teststat.site      566.77658
## teststat.species    28.76856
## [1] 10

timeFUNPerm_r_c

##    user  system elapsed
## 4489.19   23.77 4529.07

names(test_result)

## [1] "p_values" "nrepet"   "obs"      "sim.row"  "sim.col"

test_result$p_values

##                   teststat.site teststat.species p.site.permut
## MLM2BB             0.0021154812     2.115481e-03          0.05
## MLM3BB             0.0087562786     8.756279e-03          0.05
## N2-weighted lm     0.0001199832     6.111426e-05          0.05
## FC=RK-weighted lm  0.0011064089     3.636984e-05          0.05
## lm CWM/SNC         0.0017643637     3.476017e-02          0.05
##                   p.species.permut pmax.permut
## MLM2BB                        0.05        0.05
## MLM3BB                        0.05        0.05
## N2-weighted lm                0.05        0.05
## FC=RK-weighted lm             0.05        0.05
## lm CWM/SNC                    0.05        0.05
## attr(,"nrepet")
## [1] 19

save.image("AnalysesRevisit.Rdata")
```

1.4 Functions in DoSimulations.r

The key functions are:

test_models_MLM – simulates from an existing fitted MLM model after the interaction parameter `b_te` (β_{te}) has be set to a user-determined value. It then simulates data and determines of each simulated data set *p*-value(s) using one or more methods, either parametric or non-parametric using the `PermutationTest_r_c` function in section 1.1. Which models to fit and which test statistics are supplied to `test_models_MLM` is set by the options argument, the default of which is given by `setoptions4simul()`. The argument K should be nonzero for binomial distributions, for which is the binomial total. Also the number of simulations and permutations is set by the option argument.

f_rej_rate – function to analyse the output of `test_models_MLM`. It calculated the fraction rejections for the nominal level specified (default 0.05).

Functions that fit models and calculate test statistics:

CWMSNC_RKN2Unw – function that calculates the parametric *p*-values or their reprocical for the three variants of WA regression.

MLM_Wald_bte – function that calculates the parametric *p*-value or its inverse for an MLM model. It is for use in the `PermutationTest_r_c` function.

WA_MLM – function that calculates the parametric *p*-values or their inverse for list of MLM model and combines it with the parametric *p*-value of unweighted WA regressions. It is for use in the `PermutationTest_r_c` function.

```
test_models_MLM <- function(mod, K= 0, options = setoptions4simul(),...){
  # Simulation from an existing MLM model for which simulate(mod) exists(lme4 or glmm.TMB mode
l)
  dat<- model.frame(mod)
  obj <- dat4MLM2TE_obj(dat) # TE_obj with c
  n_sites <- nrow(obj$L); n_species <- ncol(obj$L)
  options4perm <- setoptions4pmax_test(verbose = FALSE)
  n_simul <- options$n_simul
  # one trait one environmental variables  data----------------------------------
  for (i in 1:n_simul){
      Ysim <- simulate(mod,...)
      if (length(unlist(Ysim))>length(obj$L)) Ysim <- matrix(Ysim[,1]) # for binomial data
      L <- matrix(Ysim, nrow = n_sites,ncol = n_species, dimnames = list(sites= rownames(obj$L
),species=colnames(obj$L)))
      obj.sim <- make_obj_for_traitenv(L,obj$E,obj$T)
    if (options$with.prmtr){
      suppressWarnings(p_values_WA_MLM0 <- WA_MLM(obj.sim, with.MLM = options$with.MLM, formul
a.list = options$formula.list, family.list= options$family.list,  K= K, invert_p_value = FALSE
,  model.names= options$model.names, library.list = options$library.list, verbose = FALSE))
      p_max <- apply(p_values_WA_MLM0,2,max)
      p_values_WA_MLM <- t(rbind(p_values_WA_MLM0, p_max =  p_max))
      nam_WA_MLM <- rownames(p_values_WA_MLM)
    } else {nam_WA_MLM <- NULL;p_values_WA_MLM <- NULL}

    if (options$with.perm) {
      rand_p_row_col_max <-PermutationTest_r_c(obj, FUN_test_statistics = CWMSNC_RKN2Unw, nrep
et = options$nrepet, options = options4perm)$p_values[,c("p.site.permut","p.species.permut","p
max.permut")]
      nam_perm <- paste("perm",rownames(rand_p_row_col_max), sep = "_")
      } else {rand_p_row_col_max <- NULL; nam_perm <- NULL}
```

```
    pval <- rbind(rand_p_row_col_max,p_values_WA_MLM)
    if (!i%%10) print(i)
    if (i==1){
      n.test <- nrow(pval)
      rownames(pval)<- c(nam_perm,nam_WA_MLM)
      colnames(pval) <- c("p_site","p_species","p_max")
      if(options$verbose){
        cat("\np-values in the first simulation; you can check (names of) simulated methods fr
om next output\n")
        cat("Names starting with perm are now subjected to permutation using " ,options$n_simu
l," simulations \n")
        cat(" with progress printed every 10 simulations\n")
        print(pval)
        }
      score_vals <- array(NA, dim= c(n.test, 3,n_simul)) # number of tests, 3 (row/col/max), n
umber of simulations
    }
    score_vals[,,i] <- pval
  }

  if (options$psort)    {
    score_vals <- apply(score_vals,c(1,2),sort)
    score_vals <- aperm( score_vals,c(3,2,1))
  }

  dimnames(score_vals) <-  list(methods=c(nam_perm,nam_WA_MLM),perm=c("p_site","p_species","p_
max"),simul=1:n_simul)
  return(score_vals)
}

setoptions4simul<- function(with.MLM = FALSE, with.perm=FALSE, with.prmtr=TRUE, psort=FALSE,
n_simul= 20, nrepet = 199,
                            formula.list = list(y~ trait*env + (1+trait|site)+ (1+env|species)
),
                            family.list = list(nbinom2()),  K= 0,
                            model.names= "MLM3",
                            library.list = list("glmmTMB"), verbose = FALSE){

  list(with.MLM = with.MLM, with.perm= with.perm, with.prmtr = with.prmtr, psort = psort,  n_s
imul= n_simul, nrepet = nrepet,
       formula.list = formula.list,  family.list =  family.list ,model.names = model.names, li
brary.list=library.list, verbose = verbose )
}

# all three weighted averaging regressions in one analysis, starting from a TE_obj object
CWMSNC_RKN2Unw <- function(obj, invert_p_value = TRUE, ...){
  FF <- matrix(c(CWMSNC_regressions(obj, weighing = 2)$p_values[-3],
                 CWMSNC_regressions(obj, weighing = 1)$p_values[-3],
                 CWMSNC_regressions(obj, weighing = 0)$p_values[-3]), nrow= 2);
  colnames(FF)<- c("N2-weighted lm","FC=RK-weighted lm","lm CWM/SNC");
  if (invert_p_value)rownames(FF)<- c("teststat.site","teststat.species") else rownames(FF)<-
c("p_row","p_col")
  if (invert_p_value) FF <- 1/FF
  return(FF)
}

# function for MLM analysis starting from a TE_obj object
MLM_Wald_bte <- function(obj, formula, family,  K= 0, library = "glmmTMB", invert_p_value = TR
UE, verbose = FALSE, nAGQ =0,  ...){
  #  function for use in PermutationTest_r_c
  # @param obj object of class TE_obj (made by function make_obj_for_traitenv(L,E,T,cutoff))
```

```r
  # @formula model formula using names created by expand4glmm(obj, K = K)[use verbose = TRUE t
o see these names]
  # @param K is binomial total; if K[1]==0 data are not binomial or presence / absence
  #                       K can be a scalar or vector of n*m (number of sites * number of speci
es)
  # @param family glm family; use character form
  # @param library use "glmmTMB" or else (e.g. "lme4")
  # @param verbose If TRUE, prints the names for use in formula.
  # @value test statistic: 1/pWald_b_te
  dat <- expand4glmm(obj, K = K)
  if (verbose) print(str(dat))
  if (library == "glmmTMB"){
    MLM <- glmmTMB(formula, data=dat, family= family)
    B <- summary(MLM)$coefficients$'cond'
  } else if  (library == "lme4") {
    MLM <- glmer(formula, data=dat, family= family,  nAGQ=nAGQ, control = glmerControl(calc.de
rivs=F))
    B <- summary(MLM)$coefficients$'cond'
  } else {print(paste("library", library ,"not implemented"))}
  # BEWARE the interaction coefficient is likely the last one.... adapt if not
  p_val_Wald <- B[nrow(B),4]
  if (verbose){
    print(summary(MLM))
    names(p_val_Wald) <- "p_val_Wald"
    print(p_val_Wald)
  }
  if (is.character(family) ) testnam <- paste(library, family, "MLMWald", sep = ".") else test
nam <- paste(library, "MLMWald", sep = ".")
  if (invert_p_value) test_stat <- 1/p_val_Wald else test_stat <- p_val_Wald
  test_stat <- matrix(test_stat, nrow = 2, ncol = 1, dimnames =list(c("rows","cols"), testnam)
)
  rownames(test_stat)<- c("teststat.site","teststat.species")
  if (verbose) print(test_stat)
  return(test_stat)
}


WA_MLM <- function(obj, with.MLM = TRUE, formula.list, family.list,  K= 0, invert_p_value = TR
UE,  model.names= unlist(formula.list), library.list = list("glmmTMB"), verbose = FALSE, ...){
  # creates test statistics of the weighted averaging methods (WA) and MLM methods
  MLM_test <- NULL
  if (with.MLM){
    nlib <- length(library.list)
    nfam <- length(family.list)
    nfor <- length(formula.list)

    for (i in seq_along(formula.list)){
      MLM_testi <- MLM_Wald_bte(obj,
                                formula = formula.list[[min(c(i,nfor))]],
                                family= family.list[[min(c(i,nfam))]],  K= K, invert_p_value =
invert_p_value,
                                library.list = library.list[[min(c(i,nlib))]], verbose = verbo
se)
      MLM_test <- cbind(MLM_test,MLM_testi)
    }
    colnames(MLM_test) <- rep(model.names, length(formula.list))[1:ncol(MLM_test)]
  }
  WA <- CWMSNC(obj, invert_p_value = invert_p_value)
  return(cbind(MLM_test,WA))
}
```

```
f_rej_rate <- function(score_vals,nominal.level=0.05, drop.methods="", select.methods =""){
  methods <-dimnames(score_vals)[1]
  perm <-dimnames(score_vals)[2]# c("prow","pcol","pmax")
  n_simul <- dim(score_vals)[3]
  #print(methods)
  if(any(methods%in%drop.methods)) {
    methods = methods[-which(methods%in%drop.methods)]
    score_vals = score_vals[-which(methods%in% drop.methods),,, drop = FALSE]
  } else  if(any(methods%in%select.methods)) {
    methods = methods[which(methods%in%select.method)]
    score_vals = score_vals[which(methods%in% select.method),,, drop = FALSE]
  }
  signi <- score_vals <= nominal.level
  Rejection_rate <- apply(signi,c(1,2), mean, na.rm = TRUE)
  return(Rejection_rate)
}
```

1.5 Script DoSimulations.r

The script starts from the MLM3 model fitted by glmmTMB to the Revisit data using the betabinomial distribution with logit link. In the code, model object is modified by setting the interaction parameter β_{te} to a specified value. Data are then simulated with this changed parameters. The script is self-explanatory.

```
#Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environment relati
onships.
#Code to perform the power simulations based on the MLM3 Betabinomial model
# for the Whittaker Siskiyou Mountain revisit dataset subset at
#  https://datadryad.org/resource/doi:10.5061/dryad.7gj0s3b
# Run AnalyseData.r at least up to the line:
#                                save(MLM3, file = "MLM3.Rdata")
#and move the file MLM3.Rdata to the Rdata subfolder or use and existing glmmTMB MLM3 model
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("Rfunctions/WAregressionsMLM_rand_max_test.r")
source("Rfunctions/PowerSimulation_functions.r")


library(glmmTMB)

load("Rdata/MLM3.Rdata")
MLM <-MLM3

b_te_coefs = c(0, 0.1, 0.2, 0.3, 0.4, 0.6)

i = 5
(b_te <-b_te_coefs[i])
MLM$sdr$par.fixed[4] <- b_te # not needed, but to be sure
```

```
MLM$fit$par[4]<- b_te
MLM$fit$parfull[4] <- b_te


options4testmodels <- setoptions4simul(n_simul = 20, with.prmtr = TRUE,  with.MLM=TRUE,
                 with.perm = TRUE, nrepet = 19,
                 family.list = list("betabinomial"), verbose = TRUE)
score_vals <- test_models_MLM(MLM, K= 100, options = options4testmodels)

Fraction_rejected <- f_rej_rate(score_vals, nominal.level = 0.05)
Fraction_rejected
save.image(paste("PowerSimulations_for_b_te_", b_te,".Rdata ", sep = ""))
```

1.6 Output of DoSimulations.r

In fitting the MLM2 and MLM3 model to the simulated data with Betabinomial response distributions and logit link, the library glmmTMB gives numerous warnings. These come in two types:

```
## Warning in f(par, order = order, ...): value out of range in 'lgamma'

## Warning in fitTMB(TMBStruc): Model convergence problem; false convergence
## (8). See vignette('troubleshooting')
```

All results in the paper have been obtained ignoring these warnings. The MLM3 BB Wald test maintains the Type I error rate in the null situation (see section 1.3). In the non-null case, the power increases sigmoidal with the size of the effect as expected. The estimated powers are higher than Miller et al. (2018) report for MLM2 under the MLM2 Binomial Normal model, despite the extra variance component in MLM3.

Output for a very small number of simulations is as follows:

```
#Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environment relati
onships.
#Code to perform the power simulations based on the MLM3 Betabinomial model
# for the Whittaker Siskiyou Mountain revisit dataset subset at
#  https://datadryad.org/resource/doi:10.5061/dryad.7gj0s3b
# Run AnalyseData.r at least up to the line:
#                               save(MLM3, file = "MLM3.Rdata")
#and move the file MLM3.Rdata to the Rdata subfolder or use and existing glmmTMB MLM3 model
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("Rfunctions/WAregressionsMLM_rand_max_test.r")
source("Rfunctions/PowerSimulation_functions.r")


library(glmmTMB)

load("Rdata/MLM3.Rdata")
```

```r
MLM <-MLM3

b_te_coefs = c(0, 0.1, 0.2, 0.3, 0.4, 0.6)

i = 5
(b_te <-b_te_coefs[i])

## [1] 0.4

MLM$sdr$par.fixed[4] <- b_te # not needed, but to be sure
MLM$fit$par[4]<- b_te
MLM$fit$parfull[4] <- b_te


options4testmodels <- setoptions4simul(n_simul = 2, with.prmtr = TRUE,  with.MLM=TRUE,
                with.perm = TRUE, nrepet = 20,
                family.list = list("betabinomial"), verbose = TRUE)
score_vals <- test_models_MLM(MLM, K= 100, options = options4testmodels)

##
## p-values in the first simulation; you can check (names of) simulated methods from next outp
ut
## Names starting with perm are now subjected to permutation using  2  simulations
##  with progress printed every 10 simulations
##                               p_site    p_species        p_max
## perm_FC=RK-weighted lm 5.000000e-02 5.000000e-02 5.000000e-02
## perm_N2-weighted lm    5.000000e-02 5.000000e-02 5.000000e-02
## perm_lm CWM/SNC        5.000000e-02 5.000000e-02 5.000000e-02
## MLM3                   4.325276e-12 4.325276e-12 4.325276e-12
## lm CWM/SNC             4.169902e-07 3.441460e-08 4.169902e-07

Fraction_rejected <- f_rej_rate(score_vals, nominal.level = 0.05)
Fraction_rejected

##                          perm
## methods                   p_site p_species p_max
##   perm_N2-weighted lm       1.00      0.75  0.75
##   perm_FC=RK-weighted lm    1.00      0.15  0.15
##   perm_lm CWM/SNC           1.00      0.45  0.45
##   MLM3                      0.95      0.95  0.95
##   lm CWM/SNC                0.90      1.00  0.90

save.image(paste("PowerSimulations_for_b_te_", b_te,".Rdata ", sep = ""))
```