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

Methods in Ecology and Evolution
https://doi.org/10.1111/2041-210X.13278

Cajo J. F. ter Braak

Appendices A1 and A9

A1. Tutorial

Sections A1.1. and A1.2 show how to apply the multilevel model MLM3 and N_2-weighted CWM/SNC regressions, respectively, with graphs and the max test for both and a bootstrap test for MLM3 using R (R Core Team 2018). The R code to run the tutorial can be found at https://doi.org/10.6084/m9.figshare.8152655. See the readme.txt. The scripts for applying MLM3 (section A1.1) are TutorialMLM3_Revisit.r and TutorialMLM3_Aravo.r. The script for applying N_2-weighted CWM/SNC regressions (section A1.2) is TutorialWA.r. Instead of including source files, you can install and use the R package TraitEnvMLMWA. You can install the latest version of TraitEnvMLMWA from github with:

```
remotes::install_github("CajoterBraak/TraitEnvMLMWA")
```

A1.1 Tutorial using multilevel method MLM3

This tutorial shows how to apply the multilevel model MLM3 using R with the library glmmTMB for overdispersed data and the library lme4 if the overdispersion is very minor. Multilevel model MLM2 is not illustrated as it is inferior to MLM3.

Before the release of glmmTMB, practitioners used lme4 for overdispersed data by including to the linear predictor an observation level random effect '(1|obs)', where obs is a factor with $n \times m$ levels. For estimation, such a model requires the (approximate) integration over an $n \times m$ dimensional space, which is challenging; approximations to the integral are likely not very good. The library glmmTMB avoids these problems by allowing response distributions with overdispersion parameters.

Section A1.1 shows analysis of the Revisit data using binomial data with overdispersion as in the main text. Section A1.2 show the analysis of the Aravo data (Appendix A7) using count data, first with overdispersion and then, using lme4, without, as these data show minor overdispersion.

Logit analysis using Revisit data

The data for this example is taken from Miller et al. (2018). This data file `whittakerrevisitdata.csv` is already nearly in the format needed for MLM3 analysis. See the section A1.2 for getting the data in such a format from the format in Figure 1 in the main text (the three items, Y, **t** and **e**). Note also that the trait and environmental variable are already standardized to zero mean and unit variance.

Main analyses

Load the data in to R.

```
dat <- read.csv("data/whittakerrevisitdata.csv")

str(dat)

## 'data.frame':    3900 obs. of  5 variables:
##  $ site   : int  109 113 12 156 157 160 161 163 164 185 ...
##  $ species: Factor w/ 75 levels "Abies concolor",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ trait  : num  1.4 1.4 1.4 1.4 1.4 ...
##  $ env    : num  -0.351 -0.774 1.667 1.016 1.342 ...
##  $ value  : int  0 0 0 25 7 4 4 0 0 0 ...
```

In the data, value is the response variable. For logit analysis in R, it must a matrix consisting of successes and failures.

```
dat$y <- with(dat, cbind(value,100-value))

dat$site <- factor(dat$site)
```

The main analysis is obtained by

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

resulting in many warnings. We neglect these warnings (see the subsection Warnings issued by glmmTMB). The summary of the analysis is

```
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) -5.42782    0.21827 -24.868  < 2e-16 ***
## env         -0.68239    0.12503  -5.458 4.82e-08 ***
## trait        1.00306    0.19788   5.069 4.00e-07 ***
## env:trait    0.25117    0.09583   2.621  0.00876 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The estimate of the parameter for trait-environment interaction β_{te} is the entry `env:trait 0.25117` and the Wald test in the last column is highly significant (p=0.00876). Estimates for the square-root of the variance components are $\hat{\sigma}_b$ = 0.3367 and $\hat{\sigma}_c$ =0.4214. There is considerable correlation between the (Intercept) and env in the stratum 'species' (0.41) and (Intercept) and trait in the stratum 'site' (-0.92). If one of these correlations is close to 1, it may be difficult to obtain stable estimates of the variance components.

It is wise to check for non-linear main effect. Here a polynomial of order 2 is fitted and summarized.

```r
# extension with polynomial main effects

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

MLM3.quad <- update(MLM3,formula.MLM3.quad)

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
```

Note that the model is an equi-width Gaussian logit response model (ter Braak & Looman 1986; Jamil & ter Braak 2013), because the coefficient for the squared term

for env is negative (-1.30393). But it is non-significant. The corresponding coefficient for trait is positive and non-significant. Comparing these models by ANOVA

```
anova(MLM3,MLM3.quad)

## Data: dat
## Models:
## MLM3: y ~ env * trait + (1 + env | species) + (1 + trait | site), zi=~0, disp=~1
## MLM3.quad: y ~ poly(env, 2) + poly(trait, 2) + (1 + env | species) + (1 + , zi=~
0, disp=~1
## MLM3.quad:     trait | site) + env:trait, zi=~0, disp=~1
##           Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## MLM3      11 6193.9 6262.9 -3086.0   6171.9
## MLM3.quad 13 6196.9 6278.4 -3085.4   6170.9 1.0501      2     0.5915
```

shows no evidence for the need for the quadratic terms (p= 0.59). The remainder of the analyses of the Revisit data uses the linear MLM3 model.

A likelihood ratio test (LRT) is often (slightly) superior to the Wald test, but it requires the fit of two models and thus takes more time to compute. The LRT test for the interaction is obtained by the code:

```
# LRT test ----------
# perhaps useful if there are less than 20 species or sites
formulaNULL <- update(formula(MLM3),  ~ . - trait:env )
MLM0 <- glmmTMB(formulaNULL, data=dat, family= family(MLM3))

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

[...]

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

anova(MLM0,MLM3)

## Data: dat
## Models:
## MLM0: y ~ env + trait + (1 + trait | site) + (1 + env | species), zi=~0, disp=~1
## MLM3: y ~ env * trait + (1 + trait | site) + (1 + env | species), zi=~0, disp=~1
##      Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## MLM0 10 6198.8 6261.5 -3089.4   6178.8
## MLM3 11 6193.9 6262.9 -3086.0   6171.9 6.8897      1    0.00867 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

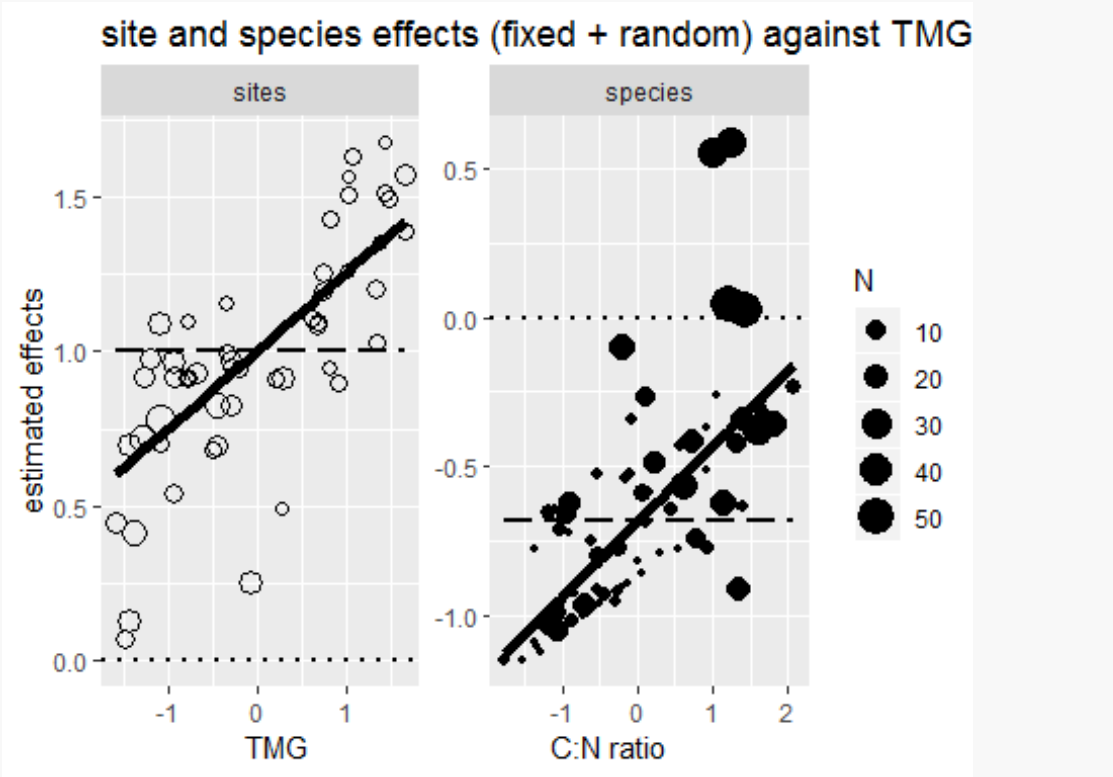
The LRT gives p-value 0.00867, which is close the Wald p-value (p=0.00876). In the initial work for the paper, both tests were used, but as I did not find much difference, I switched to the use of the Wald test only.

Graphs of fixed and random effects

Graphs like Figure 3 of the main text can be made using the function `plot.MLM3`.

```
library(ggplot2)
library(dplyr)
#source("Rfunctions/plot.MLM3.r")

library(TraitEnvMLMWA)
plot.MLM3(MLM3) +
```

```
  ggtitle('site and species effects (fixed + random) against TMG and C:N') +
  xlab("TMG                                C:N ratio") +
  ylab("estimated effects")

## Warning: Removed 75 rows containing missing values (geom_point).
```

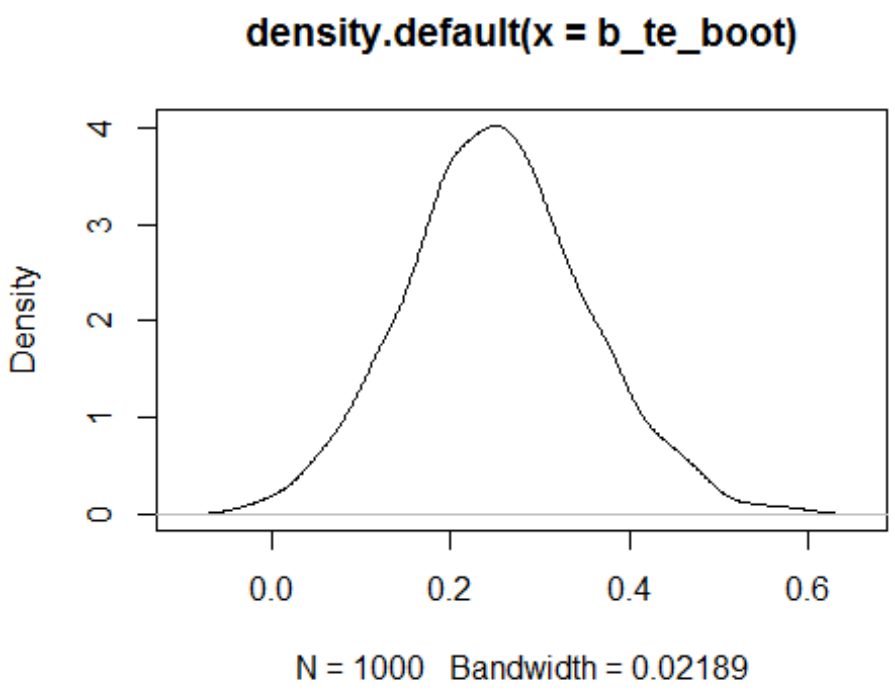


Bootstrap confidence interval for trait-environment parameter

A bootstrap confidence interval can adjust for bias in parametric MLM3. In theory - when glmmTMB with MLM3 works well - the scope for such a bootstrap is limited as MLM3 showed little bias in the simulations, except in logit models with large residual correlation. To deal with such bias, a non- or semi-parametric bootstrap would be needed (such as developed for GLM by Warton et al. 2017), but this is left for future work. The parametric bootstrap simulates without residual correlation, in which case MLM3 did not show bias (Table A8d). By consequence, the parametric bootstrap cannot correct for bias that is due to large residual correlation. However, in practice, the bootstrap is a good check whether glmmTMB with MLM3 worked well. It is a check whether it is warranted to ignore the many warnings glmmTMB issues (see also the subsection Warnings issued by glmmTMB). The code for a parametric bootstrap of the MLM3 model is:

```
getInteractionParameter <- function(MLM3){
  if (class(MLM3)[1] == "glmmTMB"){
        B <- summary(MLM3)$coefficients$'cond'
     } else if (class(MLM3)[1] == "lme4") {
        B <- summary(MLM3)$coefficients
     }
```

```
  coef_te <- B[nrow(B),-4]
  return(coef_te)
}
stats_data <- getInteractionParameter(MLM3)
# not implemented in glmmTMB
# bootMer(MLM3, FUN = getInteractionParameter, nsim = 4)
#

dat_boot <- model.frame(MLM3)
nsimul <- 1000
bootstrap_mat<- matrix(0, nrow = nsimul, ncol = length(stats_data))
colnames(bootstrap_mat) <- names(stats_data)
for (i in 1:nsimul){
  dat_boot$y <- as.matrix(simulate(MLM3))
  suppressWarnings(MLM3boot <- glmmTMB(formula(MLM3), family = betabinomial,  data=
dat_boot))
  bootstrap_mat[i, ] <- getInteractionParameter(MLM3boot)
}
summary(bootstrap_mat)

##     Estimate          Std. Error         z value
##  Min.   :-0.03181   Min.   :0.06529   Min.   :-0.2909
##  1st Qu.: 0.18604   1st Qu.:0.09012   1st Qu.: 1.9296
##  Median : 0.24980   Median :0.09720   Median : 2.5423
##  Mean   : 0.25215   Mean   :0.09762   Mean   : 2.5851
##  3rd Qu.: 0.31582   3rd Qu.:0.10472   3rd Qu.: 3.2402
##  Max.   : 0.59239   Max.   :0.13772   Max.   : 5.6388

# 95% naive/quantile/percentile  bootstap confidence interval (do not use..)
b_te_boot = bootstrap_mat[,1]
plot(density(b_te_boot))
```

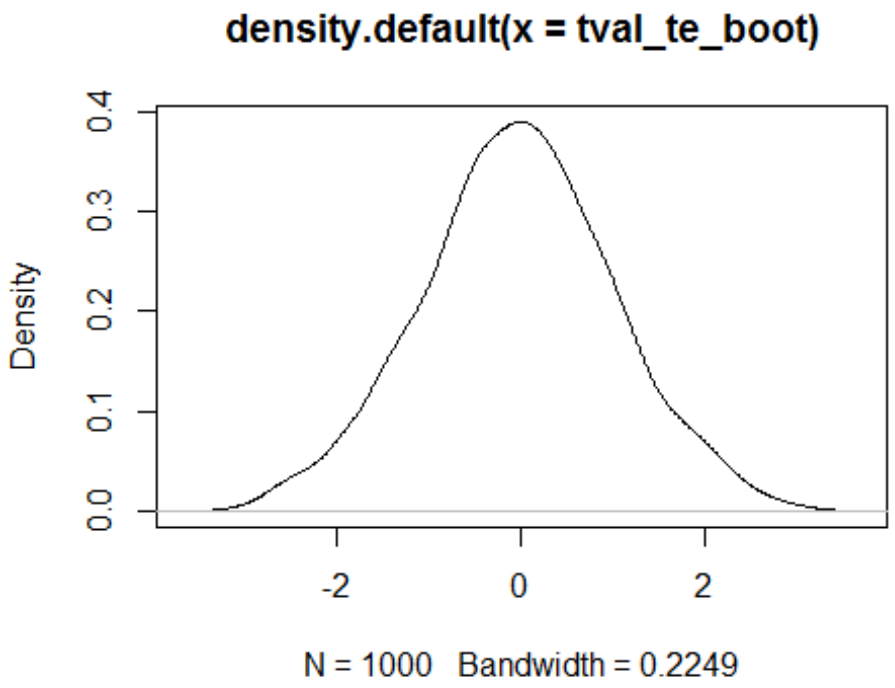



```
b_te_qu_ci = quantile(b_te_boot,probs = c(0.025,0.975)) # 95 % quantile / percentri
le bootstrap c.i.
b_te_qu_ci

##       2.5%      97.5%
## 0.05663358 0.46233038
```

```
mean(b_te_boot)

## [1] 0.2521484

sd(b_te_boot)

## [1] 0.10179
```

The mean and standard deviation of the bootstrap sample are about the same as the M LM3 estimates (0.25117 and 0.09583, respectively). A bootstrap studentized confide nce interval (Hall 1988, Davison & Hinkley 1997) is constructed as follows:

```
# 95% studentized bootstap confidence interval (use this one)
b_te_est <- stats_data[1]
se_b_te_est <- stats_data[2]
se_b_te_boot <- bootstrap_mat[,2]
# t-value when the true interaction coefficient = b_te_est
tval_te_boot = (b_te_boot - b_te_est)/se_b_te_boot
plot(density(tval_te_boot))
```




```
z_b_te_qu = quantile(tval_te_boot,probs = c(0.025,0.975), na.rm=TRUE)
z_b_te_qu # ideally symmetic -2 to 2, if not, the interval corrects for bias

##      2.5%     97.5%
## -2.081137  2.078249

b_te_Stu_boot_ci = b_te_est - rev(z_b_te_qu)*se_b_te_est
names(b_te_Stu_boot_ci)=rev(names(b_te_Stu_boot_ci))
b_te_Stu_boot_ci

##       2.5%      97.5%
## 0.05202084 0.45060189
```

The bootstrap intervals are almost equal and equal to the parametric one.

The rationale of the `rev(z_b_te_qu)` in the code above comes from the logic of the bootstrap and the hope that the z-value is asymptotically pivotal (Hall 1988,). The logic

is that the sample relates to the population in the same way as the bootstrap sample relates to the sample. The reason behind this logic is that the sample forms the truth from which the bootstrap samples. For a small number of bootstrap samples it is better to replace the quantiles `z_b_te_qu` by interpolated ones as in the function `boot.ci` of library, as follows.

```
library(boot)
t0 <- c(b_te_est, se_b_te_est^2)
t <- bootstrap_mat
t[,2] <- t[,2]^2
objboot <- list(t0 = t0, t = t, R = nrow(t))
boot.ci(objboot, type = "stud")

## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = objboot, type = "stud")
##
## Intervals :
## Level     Studentized
## 95%   ( 0.0512,  0.4530 )
## Calculations and Intervals on Original Scale
```

Testing trait-environment interaction by bootstrap

For small number of species or sites (<20, say) MLM3 shows type I error inflation (Appendix A8). A statistical test based on resampling can be useful in this case (by bootstrapping or permutation). This subsection presents the more customary bootstrap test, whereas the next presents the new model-based permutational max test. As there is no proven, reliable non- or semi-parametric bootstrap test, a parametric bootstrap is used. This bootstrap simulates from the model as fitted and is thus based on the same assumptions as MLM3. See Box A2 in Appendix A4. However, the parametric bootstrap test might be able to correct for type I error rate inflation inflation even in the case of structured residual correlation, as the simulations *without* structured residual correlation also show type I error rate inflation (Table A5)[1].

```
# Parametric bootstrap test-------------------------------------------------------
#source("Rfunctions/utility_functions.r")
#source("Rfunctions/Bootstrap_test.r")
library(TraitEnvMLMWA)

set.seed(123)
nrepet <- 19
res_boot <- Bootstrap_test_prmtrc(MLM3, test_stat = "Wald", nrepet = nrepet, Binomial_total = 100)
```

The output is

```
names(res_boot)

## [1] "p_values"  "nrepet"    "sim.boot"  "MLM0"      "obs"       "test_stat"

round(res_boot$p_values,3)

## p_prmtrc.Wald        p_boot
##         0.009         0.050
```

[1] If simulations without structured residual correlation would not show type I error rate inflation, the parametric bootstap test is unlikely to correct for type I error inflation, as it samples from such data.

```
## attr(,"nrepet")
## [1] 19
```

The observed test statistic is larger than any of the19 simulations, so that the Monte Carlo p-value is 1/(19+1) = 1/20 = 0.05. With 499 permutations, I obtained p =0.014.

Testing trait-environment interaction by permutation

The model-based permutation max test can be obtained as follows:

```
# model-based permutation max test ----------------------------------------
#source("Rfunctions/setoptions.r")
#source("Rfunctions/MLM3_p_max.r")
library(TraitEnvMLMWA)
set.seed(323)
res_perm <- MLM3_p_max(MLM3, test_stat = "Wald", nrepet = nrepet, Binomial_total =
100)
```

with output

```
names(res_perm)

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

round(res_perm$p_values,3)

##      obs.sites obs.species p.site.permut p.species.permut pmax.permut
## [1,]     0.009       0.009          0.05             0.05        0.05
## attr(,"nrepet")
## [1] 19
```

The observed test statistic is larger than any of the19 simulations in both the site-level and the species-level test, so that both Monte Carlo p-values are 0.05. The max test gives thus also p = 0.05. With 499 permutations, I obtained p =0.012.

Log-linear analysis using Aravo data

Analysis

This analysis starts from the data format in Figure 1: Y, trait, env. We scale the trait and environmental variable to zero mean and variance one, so as to make the parameter for trait-environment interaction and the variance components comparable among analyses.

```
# read and process data ---------------------------------------------------
library(ade4)
data(aravo)
Y <-aravo$spe
trait <- scale(aravo$traits$SLA)
env <- scale(aravo$env$Snow)
```

The function `make_obj_for_traitenv` checks for empty sites and species without occurrences and creates an object of class TE_obj. This object can then be used to create the expanded data frame for multilevel analysis using `expand4glmm`.

```r
#source("Rfunctions/utility_functions.r")

library(TraitEnvMLMWA)
obj <- make_obj_for_traitenv(env, Y, trait, cut_off=0)
str(obj)

## List of 3
##  $ L: int [1:75, 1:82] 0 0 3 0 0 0 2 0 0 3 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:75] "AR07" "AR71" "AR26" "AR54" ...
##   .. ..$ : chr [1:82] "Agro.rupe" "Alop.alpi" "Anth.nipp" "Heli.sede" ...
##  $ E: num [1:75, 1] -1.35 -1.35 -1.35 -1.35 -1.35 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:75] "AR07" "AR71" "AR26" "AR54" ...
##   .. ..$ : chr "env"
##  $ T: num [1:82, 1] -1.083 0.13 0.633 -0.65 -0.32 ...
##   ..- attr(*, "dimnames")=List of 2
##   .. ..$ : chr [1:82] "Agro.rupe" "Alop.alpi" "Anth.nipp" "Heli.sede" ...
##   .. ..$ : chr "trait"
##  - attr(*, "class")= chr [1:2] "TE_obj" "list"


dat <- expand4glmm(obj)
# 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"
```

We start by treating the abundance as negative binomial counts and the quadratic model:

```r
library(glmmTMB)

formula.MLM3 <- y ~ poly(env,2) + poly(trait,2) +  env : trait  + (1 + env|species)
+ (1 + trait| site)
MLM3 <- glmmTMB(formula.MLM3, family = "nbinom2",  data=dat)

## Warning in fitTMB(TMBStruc): Model convergence problem; non-positive-
## definite Hessian matrix. See vignette('troubleshooting')

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

summary(MLM3)

## Warning in sqrt(diag(vcov)): NaNs produced

[...]

## Overdispersion parameter for nbinom2 family (): 1.4e+14
```

The summary shows serious warnings; the estimate of the overdispersion parameter is hugh. In the nbinom2 family, the variance model is $\mu_{ij} + \mu_{ij}^2/\phi$, where ϕ is called the overdispersion parameter. However, the true overdispersion is $1/\phi$, so that a large value for ϕ means lack of overdispersion. Therefore, further analyses use the Poisson distribution. For this, we switch to library lme4 as it is quicker with the below-set options.

```r
# switching to lme4 -----------------------------------------------------
library(lme4)
```

```r
MLM3 <- glmer(formula.MLM3, family = poisson,  data=dat, nAGQ=0, control = glmerCon
trol(calc.derivs=F))
summary(MLM3)

## Generalized linear mixed model fit by maximum likelihood (Adaptive
##   Gauss-Hermite Quadrature, nAGQ = 0) [glmerMod]
##  Family: poisson  ( log )
## Formula:
## y ~ poly(env, 2) + poly(trait, 2) + env:trait + (1 + env | species) +
##     (1 + trait | site)
##    Data: dat
## Control: glmerControl(calc.derivs = F)
##
##      AIC      BIC   logLik deviance df.resid
##   6885.5   6966.2  -3430.8   6861.5     6138
##
## Scaled residuals:
##     Min      1Q  Median      3Q     Max
## -1.6329 -0.4272 -0.2099 -0.0212 10.1805
##
## Random effects:
##  Groups  Name        Variance Std.Dev. Corr
##  species (Intercept) 3.10803  1.7630
##          env         2.15746  1.4688   0.80
##  site    (Intercept) 0.06753  0.2599
##          trait       0.06291  0.2508   -0.78
## Number of obs: 6150, groups:  species, 82; site, 75
##
## Fixed effects:
##                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)       -2.7424     0.2127 -12.895  < 2e-16 ***
## poly(env, 2)1   -105.2570    14.2424  -7.390 1.46e-13 ***
## poly(env, 2)2    -51.5606     3.8193 -13.500  < 2e-16 ***
## poly(trait, 2)1   24.6091    16.3983   1.501  0.13343
## poly(trait, 2)2  -33.3940    11.2335  -2.973  0.00295 **
## env:trait          1.1782     0.1780   6.618 3.63e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Correlation of Fixed Effects:
##             (Intr) ply(n,2)1 ply(n,2)2 ply(t,2)1 ply(t,2)2
## poly(nv,2)1  0.779
## poly(nv,2)2  0.122  0.129
## ply(trt,2)1 -0.062 -0.074     0.034
## ply(trt,2)2  0.040 -0.010     0.090     0.077
## env:trait   -0.090 -0.100    -0.122     0.720    -0.087
```

The output shows large negative significant coefficients for the quadratic main effects.

The fitted model is an equi-width Gaussian logit response model (ter Braak & Looman 1986; Jamil & ter Braak 2013). The parameter for trait-environment interaction is 1.1782 and is highly significant (p = 3.63e-11).

A formal test of the quadratic versus the linear model is obtained via

```r
# could main effects be linear?
formula.MLM3.linear <- y ~ env * trait  + (1 + env|species) + (1 + trait| site)
MLM3.linear <- update(MLM3,formula.MLM3.linear)
anova(MLM3, MLM3.linear)

## Data: dat
## Models:
## MLM3.linear: y ~ env + trait + (1 + env | species) + (1 + trait | site) +
## MLM3.linear:     env:trait
```

```
## MLM3: y ~ poly(env, 2) + poly(trait, 2) + env:trait + (1 + env | species) +
## MLM3:     (1 + trait | site)
##             Df    AIC    BIC  logLik deviance  Chisq Chi Df Pr(>Chisq)
## MLM3.linear 10 7008.6 7075.8 -3494.3   6988.6
## MLM3        12 6885.5 6966.2 -3430.8   6861.5 127.05      2  < 2.2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The linear model is strongly rejected. It is of interest to note that the interaction parameter and its standard error in the linear model is almost the same as those in the quadratic.

Bootstrap confidence interval for trait-environment parameter

From an lme4 object, a bootstrap can be constructed with the function `bootMer`, as follows. The function `getInteractionParameter` is constructed for use of the function `boot.ci` in library boot for the construction of a studentized confidence interval.

```
# Bootstrap confidence interval -------------------------------------------

getInteractionParameter <- function(MLM3){
  if (class(MLM3)[1] == "glmmTMB"){
    B <- summary(MLM3)$coefficients$'cond'
  } else if (class(MLM3)[1] == "glmerMod") {
    B <- summary(MLM3)$coefficients
  }
  # estimate with variance for boot.ci
  coef_te <- c(B[nrow(B),1],B[nrow(B),2]^2)
  return(coef_te)
}

aa <- bootMer(MLM3, FUN = getInteractionParameter, nsim = 1000)


b_te_boot <- aa$t[,1]
mean(b_te_boot)

## [1] 1.152826

sd(b_te_boot)

## [1] 0.172938
```

The bootstrap mean and standard deviation are close to the parametric estimates for the interaction coefficient (1.1782) and its standard error (0.1780).

```
library(boot)
boot.ci(aa, index = c(1,2), type = "stud" )
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 1000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = aa, type = "stud", index = c(1, 2))
##
## Intervals :
## Level     Studentized
## 95%   ( 0.847,  1.556 )
## Calculations and Intervals on Original Scale
```

The 95% studentized bootstrap interval for the interaction coefficient is (0.847, 1.556), whereas the parametric interval is $1.1782 \pm 2 \times 0.1780 = (0.82, 1.53)$.

Warnings issued by glmmTMB

In fitting the MLM3 model, 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. This was warranted for the following reasons. 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, note that 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 (Appendix A4).

References

Davison, A.C. & Hinkley, D.V. (1997) *Bootstrap Methods and Their Applications*. Cambridge University Press, Cambridge .

Hall, P. (1988) Theoretical comparison of bootstrap confidence intervals (with discussion). *Annals of Statistics,* **16,** 927-981.

Jamil, T. & ter Braak, C.J.F. (2013) Generalized linear mixed models can detect unimodal species-environment relationships. *PeerJ,* **1,** e95.

Miller, J.E.D., Damschen, E.I. & Ives, A.R. (2018) Data from: Functional traits and community composition: a comparison among community-weighted means, weighted correlations, and multilevel models. Dryad Digital Repository.

ter Braak, C.J.F. & Looman, C.W.N. (1986) Weighted averaging, logistic regression and the Gaussian response model. *Vegetatio,* **65,** 3-11.

R Core Team (2018) *R: A language and environment for statistical computing, version 3.5.1*. R Foundation for Statistical Computing. www.R-project.org, Vienna, Austria.

Warton, D.I., Thibaut, L. & Wang, Y.A. (2017) The PIT-trap—A "model-free" bootstrap procedure for inference about regression models with discrete, multivariate responses. *Plos one,* **12,** e0181790.

A1.2 Tutorial using weighted averaging methods

This tutorial shows how to apply the weighted averaging (WA)-based methods using R with special purpose functions developed for this paper.

The first subsection shows the analysis of the Revisit data, with checks on the R-code. The second subsection show the analysis of the Aravo data as in Appendix A7. This subsection concludes with a multi-trait multi-environment test of significance, which is a weighted version of the score test developed in ter Braak (2017). The code shows how to treat categorical traits and environmental variables (factors in R).

N2-weighted lm: Revisit data

The data for this example is taken from Miller et al. (2018). This data file `whittakerrevisitdata.csv` is not in the format needed for WA-based analyses. We first extract the trait, environmental variable and the abundance matrix (**t**, **e** and **Y** in the main text, 'trait', 'env' and 'Y' in the R code) from this data, as follows.

Main analyses

Load the data in to R.

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

## adapt dataframe dat for WA-based analyses
n_sites <- with(dat, nlevels(factor(site)))
n_species <- with(dat, nlevels(factor(species)))
species <- dat$species[seq(from  = 1, by = n_sites, length.out = n_species)]
sites <- dat$site[1:n_sites]
trait <- dat$trait[seq(from  = 1, by = n_sites, length.out = n_species)]
env <- dat$env[1:n_sites]
Y <- matrix(dat$value, nrow = n_sites,ncol = n_species, dimnames = list(sites=
sites,species=species))
```

The main analysis is obtained with function `CWMSNC_regressions`. The function includes the permutation test and all results needed for plotting; it needs the utility functions so these are sourced first.

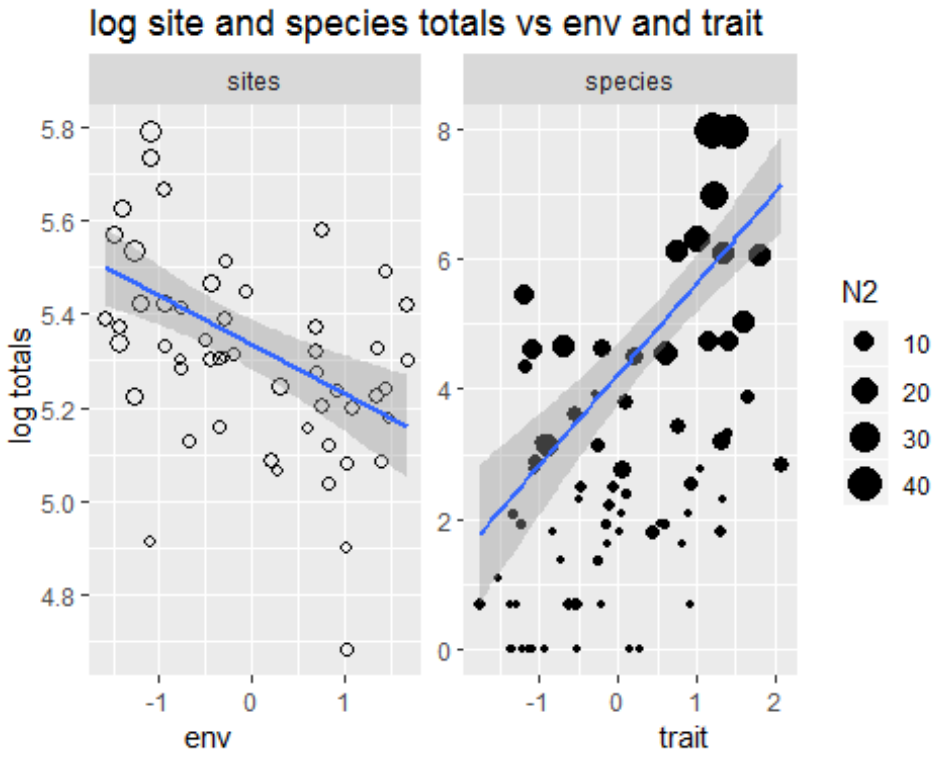
```
#  CWM/SNC regressions ----------------------------------------
#source("Rfunctions/utility_functions.r")
#source("Rfunctions/WAregressions.r")
library(TraitEnvMLMWA)

set.seed(1231)
result <- CWMSNC_regressions(env, Y, trait, weighing = "N2", nrepet = 499)
summary.CWMSNCr(result)

##                     sites species min/max
## correlations (N2w) 0.170   0.239   0.170
## p-values (N2w)     0.004   0.008   0.008
## attr(,"nrepet")
## [1] 499
```

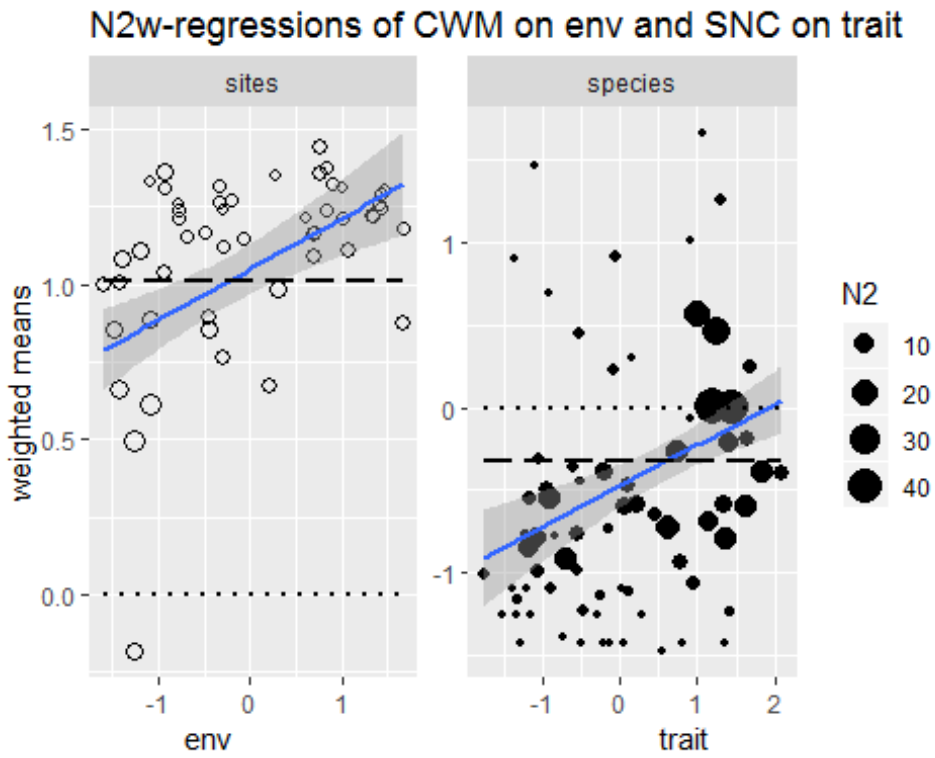
The summary shows the N_2-weighted fourth corner correlation (0.170), which is minimum of the site-level and species-level correlations (0.170 and 0.239,

respectively), and also the permutation-based *p*-value, which is the maximum of the site-level and species-level *p*-values (0.004 and 0.008, respectively). There is thus statistical evidence of trait-environment association between TMG and C:N ratio; the association is positive.

Plots are created with `plot.CWMSNCr` . The function gives two plots. The first one is about the main effects; it plots the logarithm of the row and column abundance totals against the environmental variable and trait, respectively. The second plot is about the association between the trait and the environmental variable; it plots the CWM and SNC against the environmental variable and the trait, respectively, as in Figure 4 in the main text.

```
plot.CWMSNCr(result)
```





The plots of the totals show that abundance decrease with the environmental variable (TMG) and that abundance increases with the trait (C:N ratio). The plots of CWM and SNC show that there is positive association between TMG and C:N ratio.

Analyses for showing that the short cuts in the R-code work

This subsection shows a check that the fast version of the permutation test, which has many short-cuts, gives the same result as the non-fast version, which recalculates the test statistic without any short-cut. The fast version of `WA_p_max` is used in the main function `CWMSNC_regressions`.

```
# Comparison of two versions of the max test-------------------------------------
-----------------------------
obj <- make_obj_for_traitenv(env,Y, trait, cut_off=0)
set.seed(1231)
# slow
system.time(aa<- WA_p_max(obj, nrepet =499, fast = FALSE))

##    user  system elapsed
##    2.33    0.00    2.33

round(aa$p_values,4)

##    p.site.permut p.species.permut      pmax.permut
##            0.004            0.008            0.008
## attr(,"nrepet")
## [1] 499

set.seed(1231)
# default
system.time(bb<- WA_p_max(obj, nrepet =499, fast = TRUE))
```

```
##    user  system elapsed
##    0.11    0.00    0.11

round(bb$p_values,4)

##    p.site.prmtrc p.species.prmtrc    p.site.permut p.species.permut
##            1e-04            1e-04            4e-03            8e-03
##      pmax.permut
##            8e-03
## attr(,"nrepet")
## [1] 499

# illustrating their equality
or1<- order(aa$sim.row)
or2<- order(bb$sim.row)
all.equal(or1,or2)

## [1] TRUE

or1<- order(aa$sim.col)
or2<- order(bb$sim.col)
all.equal(or1,or2)

## [1] TRUE
```

The order of the test statistics in the two versions are identical as the test statistic in the fast version is monotonic with the inverse of the parametric *p*-values that is used as test statistic in the non-fast version. The fast version is a factor 10 faster.

The permutation max test of 499 permutations using N2-weighted CWM/SNC regressions takes about 0.11 CPU units in the 'fast' version. A single fit of the MLM3 model took ~65 units (>500 times longer).

N2-weighted lm: Aravo data

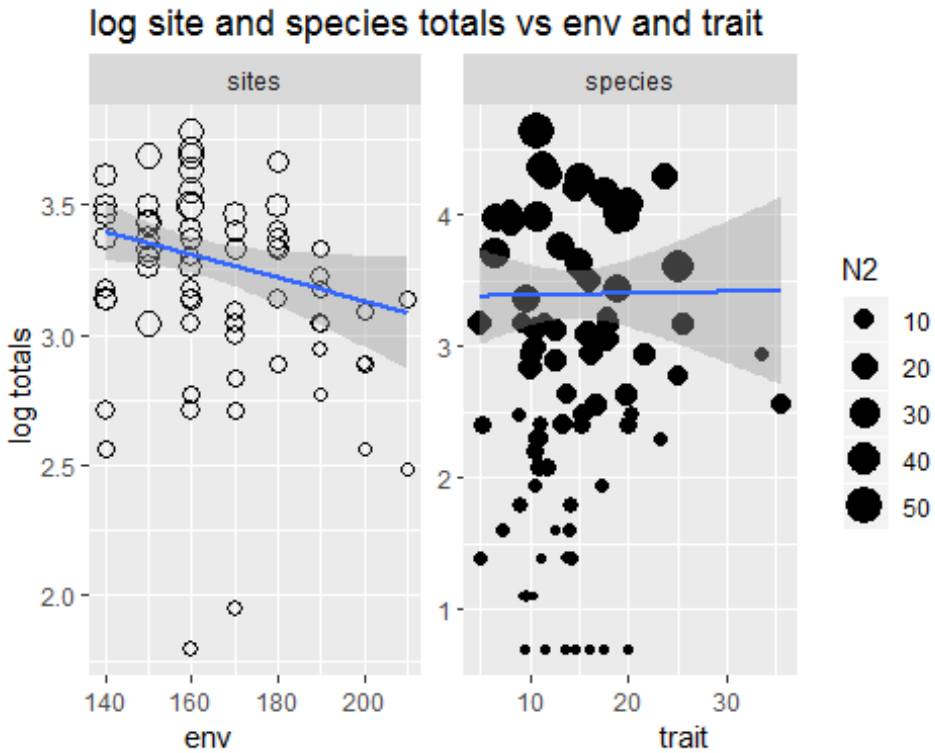
This section starts with the analysis of the variables Snow and SLA in the aravo data set. It continues with an analysis of Snow and Spread and with a multi-trait multi-environmental test of association.

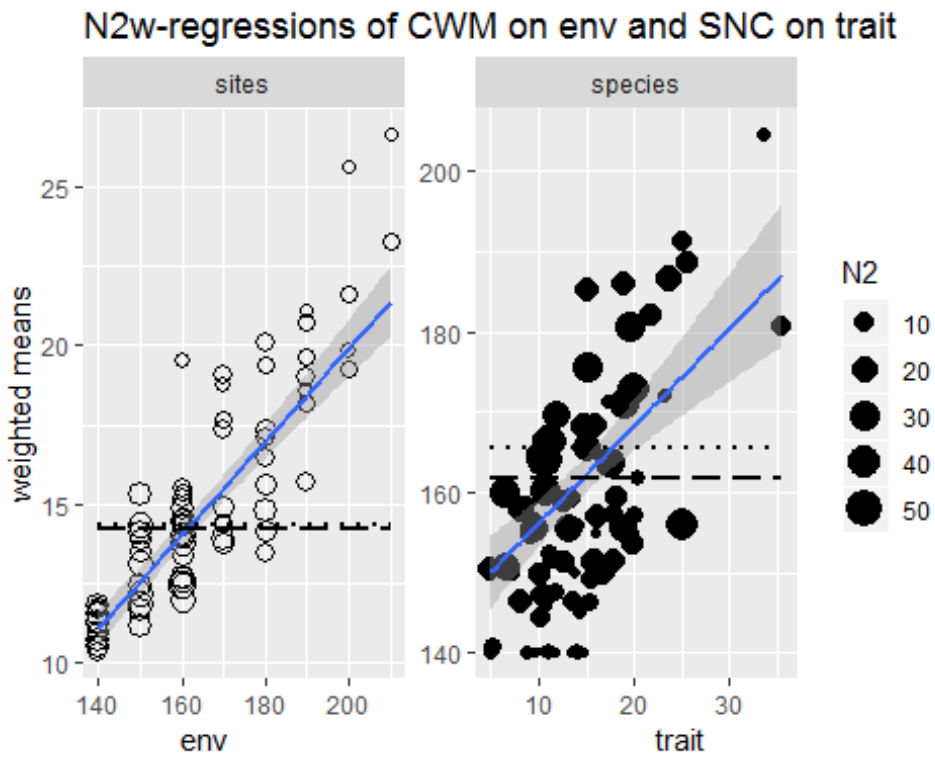
Main analyses

```
# Aravo data set ---------------------------------------------------------
library(ade4)

data("aravo")
Y <- aravo$spe
SLA <- aravo$traits$SLA
Snow <- aravo$env$Snow

result <- CWMSNC_regressions(Snow, Y, SLA, weighing = "N2", nrepet = 999)

summary.CWMSNCr(result)

##                     sites species min/max
## correlations (N2w) 0.438   0.406   0.406
## p-values (N2w)     0.001   0.001   0.001
## attr(,"nrepet")
## [1] 999
```

The association between Snow and SLA is positive and highly significant. Plots showing main effects and the association

```
plot.CWMSNCr(result)
```





The next analysis investigates the association between Spread and Snow. As the trait Spread is positive and very skew, it is log-transformed before analysis.

```
Spread <- log(aravo$traits$Spread)
result_SnowSpread <- CWMSNC_regressions(Snow, Y, Spread, weighing = "N2", nrepet =
999)
summary.CWMSNCr(result_SnowSpread)
```

```
##                     sites species min/max
## correlations (N2w) 0.080   0.072   0.072
## p-values (N2w)     0.002   0.425   0.425
## attr(,"nrepet")
## [1] 999
```

Despite that fact that the site-level permutation test is highly significant, there is no statistical evidence for assocation between Spread and Snow. In this analysis, the trait SLA takes the role of 'unobserved' trait that generates a significant CWM regression.

Multi-trait multi-environment test of association

For illustration, we conclude the WA-based method with a multi-trait multi-environment test. The test statistics is a weighted version of the score test (ter Braak 2017). After looking at histograms of each variable, I decided to log-transform variables that are nonnegative and skew so as to gain some robustness of the outcome. There is one factor in the data, which is transformed into a set of dummy variables. A variable with missing data (ZoogD) has been left out.

```
# Aravo data set: multi-trait multi-env test ------------------------------------
--------------------
summary(aravo$env)

##      Aspect          Slope       Form       PhysD        ZoogD
##  Min.   :1.000   Min.   : 0.00   1:19   Min.   : 0.00   no  :35
##  1st Qu.:5.000   1st Qu.: 0.00   2: 7   1st Qu.:20.00   some:28
##  Median :5.000   Median : 5.00   3:22   Median :30.00   high:12
##  Mean   :6.133   Mean   : 8.76   4:11   Mean   :32.07
##  3rd Qu.:8.000   3rd Qu.:15.00   5:16   3rd Qu.:42.50
##  Max.   :9.000   Max.   :35.00          Max.   :80.00
##       Snow
##  Min.   :140.0
##  1st Qu.:150.0
##  Median :160.0
##  Mean   :165.5
##  3rd Qu.:180.0
##  Max.   :210.0

E <- aravo$env[,-c(5,6)] # without Snow, without ZoogD (ZoogD has NAs)
# Slope is positive and very skew; therefore we logtransform, but there are zeroes
# add minimum non-zero value before taking logs
a <- min(E[E[,"Slope"]>0, "Slope"])
E[,"Slope"] <- log(E[,"Slope"]+ a)
# convert any factor into dummy variables
E <-  model.matrix(as.formula(paste("~ 0", paste(names(E),  collapse= "+"), sep= "+
")), data = E)
summary(E)

##      Aspect          Slope            Form1            Form2
##  Min.   :1.000   Min.   :0.6931   Min.   :0.0000   Min.   :0.00000
##  1st Qu.:5.000   1st Qu.:0.6931   1st Qu.:0.0000   1st Qu.:0.00000
##  Median :5.000   Median :1.9459   Median :0.0000   Median :0.00000
##  Mean   :6.133   Mean   :1.9515   Mean   :0.2533   Mean   :0.09333
##  3rd Qu.:8.000   3rd Qu.:2.8332   3rd Qu.:0.5000   3rd Qu.:0.00000
##  Max.   :9.000   Max.   :3.6109   Max.   :1.0000   Max.   :1.00000
##      Form3            Form4            Form5            PhysD
##  Min.   :0.0000   Min.   :0.0000   Min.   :0.0000   Min.   : 0.00
##  1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:0.0000   1st Qu.:20.00
##  Median :0.0000   Median :0.0000   Median :0.0000   Median :30.00
##  Mean   :0.2933   Mean   :0.1467   Mean   :0.2133   Mean   :32.07
##  3rd Qu.:1.0000   3rd Qu.:0.0000   3rd Qu.:0.0000   3rd Qu.:42.50
##  Max.   :1.0000   Max.   :1.0000   Max.   :1.0000   Max.   :80.00
```

```
T <- aravo$traits[, -6 ] # without SLA
# all traits are skew; therefore we logtransform
T[,c(1,3)] <- log(T[,c(1,3)]+ 1) # traits with zeroes
T[,-c(1,3)] <- log(T[,-c(1,3)]) # traits without zeroes
summary(T)

##      Height           Spread           Angle            Area
##  Min.   :0.0000   Min.   :0.6931   Min.   :0.000   Min.   :0.9163
##  1st Qu.:0.6931   1st Qu.:2.3026   1st Qu.:3.045   1st Qu.:3.8986
##  Median :1.3863   Median :2.9957   Median :3.829   Median :4.9647
##  Mean   :1.5121   Mean   :2.7350   Mean   :3.509   Mean   :4.7353
##  3rd Qu.:2.1972   3rd Qu.:3.3556   3rd Qu.:4.263   3rd Qu.:5.7177
##  Max.   :3.4340   Max.   :4.6052   Max.   :4.511   Max.   :7.4384
##      Thick             N_mass           Seed
##  Min.   :-2.5257   Min.   :4.528   Min.   :-4.6052
##  1st Qu.:-1.8326   1st Qu.:5.208   1st Qu.:-2.1855
##  Median :-1.6094   Median :5.447   Median :-1.1394
##  Mean   :-1.5147   Mean   :5.472   Mean   :-1.1732
##  3rd Qu.:-1.2040   3rd Qu.:5.752   3rd Qu.:-0.1173
##  Max.   : 0.3365   Max.   :6.139   Max.   : 1.5623

# convert any factor into dummy variables
T <-  model.matrix(as.formula(paste("~ 0", paste(names(T),  collapse= "+"), sep= "+
")), data = T)

obj <- make_obj_for_traitenv(E,Y, T, cut_off=0)
set.seed(1231)
aa<- WA_p_max(obj, nrepet =999, weighing = "N2", score_test = TRUE)
aa$p_values

##    p.site.prmtrc p.species.prmtrc    p.site.permut p.species.permut
##               NA               NA            0.001            0.001
##      pmax.permut
##            0.001
## attr(,"nrepet")
## [1] 999
```

There is evidence in the data for other associations than between SLA and Snow. If `score_test = TRUE` the function uses a score test which takes correlations between traits and between environmental variables (intra-set correlations) into account. If `weighing = "FC"` as well, it gives the test used in double constrained correspondence analysis (ter Braak, Šmilauer & Dray 2018). If `score_test = FALSE` the test ignores such correlations. If `weighing = "FC"` as well, it give the test used in RLQ (Dray *et al.* 2014). Plots of the multi-trait-multi-environment association can be made using RLQ and double constrained correspondence analysis (dc-CA); see Dray et al. 2014 and ter Braak *et al.* 2018. A procedure for selecting functional traits and environmental variables from a greater set of variables is available in Canoco 5.10 (ter Braak & Šmilauer 2018).

References

Dray, S., Choler, P., Dolédec, S., Peres-Neto, P.R., Thuiller, W., Pavoine, S. & ter Braak, C.J.F. (2014) Combining the fourth-corner and the RLQ methods for assessing trait responses to environmental variation. *Ecology,* **95,** 14-21.

Miller, J.E.D., Damschen, E.I. & Ives, A.R. (2018) Data from: Functional traits and community composition: a comparison among community-weighted means, weighted correlations, and multilevel models. Dryad Digital Repository.

ter Braak, C.J.F. (2017) Fourth-corner correlation is a score test statistic in a log-linear trait–environment model that is useful in permutation testing. *Environmental and Ecological Statistics,* **24,** 219-242.

ter Braak, C.J.F. & Šmilauer, P. (2018) *Canoco reference manual and user's guide: software for ordination (version 5.10).* Microcomputer Power, Ithaca, USA.

ter Braak, C.J.F., Šmilauer, P. & Dray, S. (2018) Algorithms and biplots for double constrained correspondence analysis. *Environmental and Ecological Statistics,* **25,** 171-197.

A9. R functions used in the tutorial

This appendix presents the R functions used in the Tutorial (Appendix A1).

A9.1 Utility functions in file utility_functions.r

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

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.

Code

```
# Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environ
ment relationships.

make_obj_for_traitenv <- function(E1, L, T1, cut_off = 0){
  # makes object of class TE_obj from matrices or dataframes from L,E1 and T1 (L=Y)
  # 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

  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]
  }
  #print(str(L))
  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); if(is.null(colnames(E1))){ if (ncol(E1)>1) colnames(E1
) <-  paste("E", 1:ncol(E1), sep = "") else colnames(E1) <- "env"}
  rownames(T1)= colnames(L); if(is.null(colnames(T1))){ if (ncol(T1)>1) colnames(T1
) <-  paste("T", 1:ncol(T1), sep = "") else colnames(T1) <- "trait"}
  obj = list(L=L, E=E1,T = T1)
  class(obj) = c("TE_obj", class(obj))
  return(obj)
}

expand4glmm <- function(obj, K= 0){
  # K  = 0 for Poisson/Negative binomial counts and Binomial total for (beta-)binom
ial
  # adapted from Jamil et al 2013
  if ('trait0' %in% names(obj)) inPermut_r_c = TRUE else inPermut_r_c = FALSE
  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 (inPermut_r_c){
      Evec0 <-  env0[site,, drop = FALSE];  rownames(Evec0)= NULL
      Tvec0 <-  trait0[species,, drop = FALSE]; rownames(Tvec0)= 1:nrow(Tvec0)
    } else {Evec0 =NULL; Tvec0 = NULL}
    if (is.null(colnames(E)))  {
      colnames(Evec) = paste("env", 1:ncol(Evec), sep = "")
    }
    if (is.null(colnames(T))) {
      colnames(Tvec) = paste("trait", 1:ncol(Tvec), sep = "")
    }
    ET <- Rten2_with_names(Evec,Tvec)
    if (inPermut_r_c){
      XYZ <- data.frame(y,site,species, obs = 1:length(y),Tvec, Evec, ET, Tvec0,  E
vec0)
      names(XYZ)[c(5:6, 8:9)] <- c("trait", "env","trait0", "env0")
    } else {
      XYZ <- data.frame(y,site,species, obs = 1:length(y),Tvec, Evec, ET)
      names(XYZ)[5:6] <- c("trait", "env")
    }
    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$trait[seq(from  = 1, by = n_sites, length.out = n_species)], nrow
= n_species, dimnames = list(species, "trait"))
  E <- matrix(dat$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,species=species))
  }
  else { # e.g. with count data
    L <- matrix(dat$y, nrow = n_sites,ncol = n_species, dimnames = list(sites= site
s,species=species))
  }
  # check the data for empty species and sites and bring them in standard format of
class TE_obj
  obj <- make_obj_for_traitenv(E1 = E, L = L,T1 = T, cut_off = cut_off)
}


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, s
ep = ":")} )
  rownames(res) <- rownames(X1)
  return(res)
}
```

A9.2 MLM3 plots in file plot.MLM3.r

The function `plot.MLM3` plots the fixid and random effect of an MLM3 object using ggplot2. The current R code has restrictions on the names of variables. The names in the data.frame (model.frame) of MLM3 should be in any order y, species, sites, trait, env. The data should be in column-wise order (first all the data of the first species, then the second....), as is the case when the data for an MLM3 fit is generated by the utility function `make_obj_for_traitenv` (see Appendix 1.1 Aravo data for an example). The function uses the libraries ggplot2 and dplyr.

```
plot.MLM3 <- function(MLM3, verbose = FALSE, title = paste( 'site and species effec
ts (fixed + random) against env and trait' )){
  # plot of a fitted MLM3 object
  # the names in the data.frame (model.frame) of MLM3 should be in any order
  # y, species, sites, trait, env
  # data should be in column-wise order (first all the data of the first species, t
hen the second....)
  library(ggplot2)
  library(dplyr)
  # get data in Y, trait, env form (Figure 1)-------------------------------------
--------------------------
  dat <-model.frame(MLM3)
  n_sites <- with(dat, nlevels(factor(site)))
  n_species <- with(dat, nlevels(factor(species)))
  species <- dat$species[seq(from  = 1, by = n_sites, length.out = n_species)]
  sites <- dat$site[1:n_sites]
  trait <- dat$trait[seq(from  = 1, by = n_sites, length.out = n_species)]
  env <- dat$env[1:n_sites]
  if (length(dat$y) > nrow(dat)){
    # binomial data
     Y <- matrix(dat$y[,1], nrow = n_sites,ncol = n_species, dimnames = list(sites=
sites,species=species))
  } else {
    # count data
     Y <- matrix(dat$y, nrow = n_sites,ncol = n_species, dimnames = list(sites= sit
es,species=species))
```

```
  }

  # extract random effect and b_te non-null ------------------------------------
---

  if (verbose) print(summary(MLM3))
  B <- summary(MLM3)$coefficients$'cond'

  b_te <- B[nrow(B),1]; gamma0 <- B["env",1]; beta0 <- B["trait",1]
  xx <- ranef(MLM3)$'cond'
  b_i <- xx$site$trait; c_j <-xx$species$env

  b_i_star <- beta0 + b_te*env + b_i      # Eq (2a)
  c_j_star <- gamma0 + b_te*trait + c_j   # Eq (2b)

  ETdat <- data.frame(env, b_i_star, N = rowSums(Y>0), level = "sites")
  names(ETdat)[2]<- "b_star"
  ETdat$beta0 = beta0
  ETdat$b_tslope = beta0 + b_te*env

  TEdat <- data.frame(trait, c_j_star= c_j_star, N = colSums(Y>0), level = "species
")
  names(TEdat)[2]<-"c_star"
  TEdat$gamma0 = gamma0
  TEdat$b_eslope = gamma0 + b_te*trait

  ETTEdat <- full_join(ETdat, TEdat, by= c("N","level"))
  # facet plot ------------------------------------------------------------
  pp <- ggplot(ETTEdat) +
    geom_point(aes(x= env, y = b_star, size = N), shape = 1) +
    geom_point(aes(x= trait, y = c_star, size = N), shape = 16) +
    geom_line(aes(x = env, y = b_tslope), size = 2) +
    geom_line(aes(x = trait, y = b_eslope), size = 2) +
    geom_hline(yintercept = 0, linetype = 'dotted',size = 1) +
    geom_line(aes(x = env, y = beta0), linetype = 'longdash', size = 1) +
    geom_line(aes(x = trait, y = gamma0), linetype = 'longdash', size = 1) +
    facet_wrap(~level, scales = "free") +
    ggtitle(title) +
    xlab("environmental variable                                    trait") +
    ylab("estimated effects")
  return(pp)
}
```

A9.3 Parametric bootstrap test in file Bootstrap_test_prmtrc.r

The function `Bootstrap_test_prmtrc` performs a bootstrap significance test of a fitted MLM3 object. The test statistic can be "Wald" or "LRT"; the latter is slower. The parameter `Binomial_total` is to decide whether the response should be matrix (as in logit models) or vector (as in log-linear models). The parameter `Binomial_total` should be set to the Binomial total N in the former case. The parameter `nAGQ` is used in lme4 to decide on the accuracy of numerical approximation to integrals; the value 0 gives the quickest fit.

Code

```
# Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environ
ment relationships.
Bootstrap_test_prmtrc <- function(MLM3, test_stat = "Wald", nrepet = 19, Binomial_t
otal = 0, nAGQ = 0, ...) {
  # Bootstrap test based on test_stat (Wald or LRT or both) according to Box A2 in
Appendix A3.
  # MLM3 is a fitted glmmTMB or lme4 objects
  # Binomial_total  = 0 for Poisson/Negative binomial counts and Binomial total for
(beta-)binomial
  # BEWARE the interaction coefficient is likely the last one, in the code nrow(B),
.... adapt if not

  # Step 1: arguments test_stat (Wald or LRT) and nrepet
  # Step 2: fit the null model
  formulaNULL <- update(formula(MLM3),  ~ . - trait:env )
  dat <- model.frame(MLM3)
  if (class(MLM3)[1] =="glmmTMB") {
  #  suppressWarnings(MLM0 <- glmmTMB(formulaNULL, data=dat, family= family(MLM3)))
    suppressWarnings(MLM0 <- update(MLM3, formulaNULL))
  } else {
    suppressWarnings(MLM0 <- update(MLM3, formulaNULL,  nAGQ=nAGQ, control = glmerC
ontrol(calc.derivs=F)))
  }
  # Step 3: get F0: the test statistic from the data
  if (test_stat %in% c("Wald", "both")) {
    if (class(MLM3)[1] == "glmmTMB"){
      B <- summary(MLM3)$coefficients$'cond'
    } else { # lme4
      B <- summary(MLM3)$coefficients
    }
    # BEWARE the interaction coefficient is likely the last one.... adapt if not
    obs1 <- 1/B[nrow(B),4] # 1/p-value
    names(obs1) <- "Wald"
  }
  if (test_stat %in% c("LRT", "both")) {
    obs2 <-  logLik(MLM3) - logLik(MLM0)
    names(obs2) <- "LRT"
  }
  if (test_stat=="Wald") obs <- obs1 else if (test_stat=="LRT") obs <- obs2 else ob
s<- c(obs1,obs2)
  if (test_stat == "both") id_LRT <-2 else id_LRT <- 1
  obj <- dat4MLM2TE_obj(dat) # TE_obj
  n_sites <- nrow(obj$L); n_species <- ncol(obj$L)

  sim.boot <- matrix(0, nrow = nrepet, ncol = length(obs))
  colnames(sim.boot) <- paste("boot",names(obs), sep ="")
  for (i in 1:nrepet){
    # step 4 and 5: simulated from the null model and calculate the test statistic
    Ysim <- simulate(MLM0)
    L <- matrix(Ysim[[1]], nrow = n_sites,ncol = n_species, dimnames = list(sites=
rownames(obj$L),species=colnames(obj$L)))
    obj.sim <- make_obj_for_traitenv(obj$E, L, obj$T)
    dat.sim <- expand4glmm(obj.sim, K = Binomial_total)
    if (class(MLM3)[1] == "glmmTMB"){
      suppressWarnings(MLM3.sim <- try(glmmTMB(formula(MLM3), data=dat.sim, family=
family(MLM3))))
      if (!class(MLM3.sim)[1]=="try-error"){
        B <- summary(MLM3.sim)$coefficients$'cond'}
      else {B <- matrix(NA, nrow =2, ncol = 4); MLM3.sim = 0}
    } else if (class(MLM3)[1] == "glmerMod") {
      suppressWarnings(MLM3.sim <- try(glmer(formula(MLM3), data=dat.sim, family= f
```

```
amily(MLM3),  nAGQ=nAGQ, control = glmerControl(calc.derivs=F))))
      if (!class(MLM3.sim)[1]=="try-error"){
        B <- summary(MLM3.sim)$coefficients
      } else {B <- matrix(NA, nrow =2, ncol = 4); MLM3.sim = 0}
    }
    if (test_stat %in% c("Wald","both")){
      sim.boot[i,1] <- 1/B[nrow(B),4]
    }
    if (test_stat %in% c("LRT","both")) { # LRT: refit null model
      if (class(MLM3.sim)[1] =="glmmTMB") {
        suppressWarnings(MLM0.sim <- try(glmmTMB(formulaNULL, data=dat.sim, family=
family(MLM3))))
      } else {
        suppressWarnings(MLM0.sim <- try(update(MLM3.sim, formulaNULL,  nAGQ=nAGQ,
control = glmerControl(calc.derivs=F))))
      }
      if (class(MLM0)[1]=="try-error") {
        sim.boot[i,id_LRT] <- NA
      } else sim.boot[i,id_LRT] <- logLik(MLM3.sim) - logLik(MLM0.sim)
    }
  }
  # step 6:     Compute the Monte Carlo significance level,
  #           i.e. compute the number of values F 0, F 1, F 2, ... , FK
  #           that is greater than or equal to F0 (this number is thus at least 1),
  #           and divide by K + 1 (with K = nrepet).
  if (length(obs)==1){
    isna.r <-  sum(is.na(sim.boot))
    p_boot <- (sum(abs(sim.boot) >= abs(obs), na.rm = TRUE) + 1)  / (nrepet- isna.r
+ 1)
  } else {
    obs.mat <- matrix(rep(abs(obs),each=nrepet), nrow= nrepet, ncol=length(obs))
    isna.r <-  colSums(is.na(sim.boot))
    p_boot <- (colSums(abs(sim.boot) >= obs.mat, na.rm=TRUE) + 1)/ (nrepet - isna.r
+ 1)
  }
  if (test_stat %in% c("Wald","both")) p_val1 <- 1/obs[1]
  if (test_stat %in% c("LRT","both"))  p_val2 <- pchisq(2*obs[id_LRT], df =1, lower
.tail = FALSE)
  if (test_stat=="Wald") p_val <- p_val1 else if (test_stat=="LRT") p_val <- p_val2
else p_val<- c(p_val1,p_val2)
  if (test_stat == "both") {
    result <- cbind(p_prmtrc= p_val, p_boot = p_boot)

  } else result <- c(p_prmtrc= p_val, p_boot = p_boot)
  attr(result, "nrepet")<- nrepet
  return(list(p_values=result, nrepet = nrepet, sim.boot=sim.boot, MLM0 = MLM0, obs
= obs, test_stat = test_stat))
}
```

A9.4 MLM3-based permutational max test in the file MLM3_p_max.r

The function `MLM3_p_max` performs a permutation-based max test of an fitted MLM3. The test statistic can be "Wald" or "LRT"; the latter is slower. The parameter `Binomial_total` is to decide whether the response should be matrix (as in logit models) or vector (as in log-linear models). The parameter `Binomial_total` should be set to the Binomial total N in the former case. The parameter `nAGQ` is used in lme4 to decide on the accuracy of numerical approximation to integrals; the value 0 gives the quickest fit.

The function `MLM3_p_max` depends on a number of functions of which the most important ones are:

MLM_Wald_bte – function that calculates the parametric *p*-value or its inverse for an MLM model.

PermutationTest_r_c_max - 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`, which is set by the options. Options are set by the argument `options = setoptions4pmax_test()`.

Code

```
# Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environ
ment relationships.
MLM3_p_max <- function(MLM3, nrepet = 19, Binomial_total = 0, test_stat = "Wald", n
AGQ =0) {
  #n_sites <- nrow(obj$L); n_species <- ncol(obj$L)

  options1 <- setoptions4pmax_test(verbose = TRUE,
            FUN_test_statistics =  list(init= init_MLM_pmax, test= MLM_Wald_bte),
                                   with.MLM= TRUE, nrepet = nrepet,
                                   perm.mat.spe = 0 , perm.mat.sit = 0, print = 10,
                                   test_statistics_are_1divp = c(TRUE), nmax = 6, f
ilek = 1)

  if (test_stat == "Wald") {with.LRT<-FALSE; LRT <- FALSE} else {with.LRT<-TRUE; LR
T <- TRUE; }

  if (class(MLM3)[1] =="glmmTMB") library = "glmmTMB" else library = "lme4"
  options2 <- setoptions4MLM( formula = formula(MLM3),
                              family = family(MLM3), K = Binomial_total,
                              model.name= c("MLM3"), estimation = FALSE,
                              library = library, output = "short", nAGQ =nAGQ,
                              with.LRT= with.LRT, LRT.only = LRT, verbose = FALSE)
  options3 <- c(options1,options2[-length(options2)])
  obj <- dat4MLM2TE_obj(model.frame(MLM3)) # TE_obj
  # aa <- MLM_Wald_bte(obj, options3)
  # aa
  result <- PermutationTest_r_c_max(obj, options = options3)
  return(result)
}

MLM_Wald_bte <- function(obj, options,  ...){
  #  function for use in PermutationTest_r_c_max
  # @param obj object of class TE_obj (made by function make_obj_for_traitenv(E, L,
T,cutoff))
  # @formula model formula using names created by expand4glmm(obj, K = K)[use verbo
se = TRUE to see these names]
  # @param K is binomial total; if K[1]==0 data are not binomial or presence / abse
nce
  #                       K can be a scalar or vector of n*m (number of sites * numb
er 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.
  # @param estimation if TRUE, the interaction coefficient is given, else testing i
s carried out via the Wald test
  # @value test statistic: 1/pWald_b_te
  result <- with(options,
```

```
 {
   if (estimation){
     col_b_te = 1 # column position of the interaction coefficient to select it
     invert_p_value = FALSE
   } else col_b_te = 4   # column position of the p-value to select it
   dat <- expand4glmm(obj, K = K)
   #names(dat)[c(5:6, 8:9)] <- c("trait", "env","trait0", "env0")
   #if (verbose) print(str(dat))
   if (library == "glmmTMB"){
     suppressWarnings( MLM <- try(glmmTMB(formula, data=dat, family= family)))
     if (!class(MLM)[1]=="try-error"){
       B <- summary(MLM)$coefficients$'cond'}
     else {B <- matrix(NA, nrow =2, ncol = 4); MLM = 0}
   } else if  (library == "lme4") {
     suppressWarnings(MLM <- try(glmer(formula, data=dat, family= family,  nAGQ=nAG
Q, control = glmerControl(calc.derivs=F))))
     if (!class(MLM)[1]=="try-error"){
       B <- summary(MLM)$coefficients
     } else {B <- matrix(NA, nrow =2, ncol = 4); MLM = 0}
   } 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),col_b_te]
   if (verbose){
     print(summary(MLM))
     if (estimation) {names(p_val_Wald) <- "b_te"} else 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 & !estimation) 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","col
s"), testnam) )
   if (with.LRT){
     formulaNULL <- update(formula(MLM),  ~ . - trait:env )
     if (class(MLM)[1] =="glmmTMB") {
       suppressWarnings(MLM0 <- try(update(MLM, formula = formulaNULL)))
     } else {
       suppressWarnings(MLM0 <- try(update(MLM, formulaNULL,  nAGQ=nAGQ, control =
glmerControl(calc.derivs=F))))
     }
     if (class(MLM0)[1]=="try-error") {
       if (class(MLM)[1] =="glmmTMB") suppressWarnings(MLM0 <- try(glmmTMB(formulaN
ULL, data=dat, family= family(MLM)))) else
         suppressWarnings(MLM0 <- try(glmer(formula, data=dat, family= family,  nAG
Q=nAGQ, control = glmerControl(calc.derivs=F))))
       if (class(MLM0)[1]=="try-error") {
         p_val_LRT <- NA
       } else {
         p_val_LRT <- pchisq(2*( logLik(MLM) - logLik(MLM0)), df =1, lower.tail = F
ALSE)
       }
     } else p_val_LRT <- pchisq(2*( logLik(MLM) - logLik(MLM0)), df =1, lower.tail
= FALSE)
     if (invert_p_value) test_stat2 <- 1/p_val_LRT else test_stat2 <- p_val_LRT
     names(test_stat2) <- "pval LRT"
     if (is.character(family) ) testnam2 <- paste(library, family, "MLM.LRT", sep =
".") else testnam2 <- paste(library, "MLM.LRT", sep = ".")

     #test_stat2 <- matrix(test_stat2, nrow = 2, ncol = 1, dimnames =list(c("rows",
"cols"), testnam2) )
     # print(test_stat2)
     if (LRT.only) {
       test_stat <- test_stat2
       } else {
```

```
      test_stat <- c(test_stat,test_stat2)
      names(test_stat) <- c("p_val_Wald", "p_val_LRT")
    }
  }
  #print(test_stat)
  if (estimation) {
    test_stat <- c(test_stat, B[nrow(B),2]) # standard error
    names(test_stat) <- c("b_te","se_b_te")
  } #else rownames(test_stat)<- c("teststat.site","teststat.species")
  if (verbose) print(test_stat)
  if (output =="short"){
    result <- test_stat
  } else {
    result <- list(test_stat = test_stat, model = MLM )
  }
  return(result)
                 })
  return(result)
}


init_MLM_pmax <- function(obj, options){list(obj=obj, options=options)}


PermutationTest_r_c_max <- function(obj, options = c(setoptions4pmax_test(),setopti
ons4MLM()),  ...) {
  # Specialized max test for WA-based and model-based MLM based methods
  #function to determine randomization p-values based on test statistics for rows a
nd columns
  # based on row and column based permutations of (possibly model-based) test-stati
stics,
  # 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-stati
stics, 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 or vector)
  #                        it should return 1/pval
  #
  #
  # obj from make_obj_for_traitenv(E, L, T, cut_off): matrices L,E and T (one envir
onmental variable, 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.)

  # for  permutation of model-based methods trait0 and env0 are not permuted, where
as E and T are.
  # the model formula can thus contain the terms env0 and trait0



  if (is.matrix(options$perm.mat.spe)) options$perm.mat.spe <- options$perm.mat.spe
[sample(nrow(options$perm.mat.spe)),]
  if (is.matrix(options$perm.mat.site)) options$perm.mat.site <- options$perm.mat.s
ite[sample(nrow(options$perm.mat.site)),]

  result <- options$FUN_test_statistics$init(obj,options)
  obj <- result$obj; options <- result$options
  rm(result)
  options_sites <- options_species <- options
  options_sites$verbose  <- options_species$verbose <- FALSE
```

```
  options_species$level <- "species"
  options_sites$level <- "sites"
  if (options$with.MLM){
    # add random T and E to data: trait0 and env0 and update formulas accordingly
    i_spe0 <- sample(1:nrow(obj$T), 1)
    i_sit0 <- sample(1:nrow(obj$E), 1)
    per.col <- syssample(i = i_spe0, options$perm.mat.spe, n =ncol(obj$L))
    obj$trait0 <- matrix(obj$T[per.col,,drop =FALSE],nrow= nrow(obj$T), ncol =ncol(
obj$T))  # used in species level
    per.row <- syssample(i = i_sit0, options$perm.mat.site, n =nrow(obj$L))
    obj$env0 <- matrix(obj$E[per.row,,drop =FALSE],nrow= nrow(obj$E), ncol =ncol(ob
j$E)) # used in site level

    #  e.g:  y ~  trait*env  + (1 + trait| site) + (1 + env|species) becomes
    #       y ~  trait*env  + trait0 + (1 + trait| site) + (1 + env|species) #for s
pecies
    # #     y ~  trait*env  + env0 + (1 + trait| site) + (1 + env|species) # for si
tes

    options_species$formula <-  update(options_species$formula, ~ . + trait0 )
    options_sites$formula  <-  update(options_sites$formula, ~ . + env0 )
  }
  obs.species <- options_species$FUN_test_statistics$test(obj, options = options_sp
ecies)
  obs.sites <- options_sites$FUN_test_statistics$test(obj, options = options_sites)
  obs <- rbind(obs.sites,obs.species)
  if (options$verbose) print(obs)
  if (length(options$test_statistics_are_1divp)==1) {
    options$test_statistics_are_1divp <- rep(options$test_statistics_are_1divp, nco
l(obs))
  } else if (length(options$test_statistics_are_1divp)!=ncol(obs)){
    print(paste("Warning: length of options$test_statistics_are_1divp  (", length(o
ptions$test_statistics_are_1divp) ,") is not equal to number of teststatistics: ",n
col(obs) ))
    options$test_statistics_are_1divp <- rep(options$test_statistics_are_1divp, nco
l(obs))
  }
  if (options$with.MLM){
    # change trait0 and env0 to T and E in data and update formulas so that only en
v:trait is permuted
    obj$trait0 <- obj$T
    obj$env0 <- obj$E
    #  e.g:  y ~  trait*env  + trait0 + (1 + trait| site) + (1 + env|species) becom
es
    #       y ~  trait*env  + trait0 + (1 + trait0| site) + (1 + env|species) #for
species
    # #     y ~  trait*env  + env0 + (1 + trait| site) + (1 + env0|species) # for s
ites

    options_species$formula <- update(options_species$formula,~ . -(1+trait|site)+(
1+trait0|site))
    options_sites$formula <- update(options_sites$formula,~ . -(1+env|species)+(1+e
nv0|species))

  }
  nrepet <- options$nrepet
  if(is.matrix(options$perm.mat.spe)|| is.matrix(options$perm.mat.sit)){
    nrepet <- min(c(nrow(options$perm.mat.spe)-1,nrow(options$perm.mat.sit)-1, nrep
et) )#  minus identity ispe_0 isit_0 permutation (which gives collinearity)
  } else { nrepet <- options$nrepet}
  sim.row <- matrix(0, nrow = nrepet, ncol = ncol(obs))
  sim.col <- matrix(0, nrow = nrepet, ncol = ncol(obs))


  for(i in 1:nrepet){
```

```
    per.row <- syssample(i = i, exclude = i_sit0, perm.mat = options$perm.mat.site,
n =nrow(obj$E))
    per.col <- syssample(i = i, exclude = i_spe0, perm.mat = options$perm.mat.speci
es, n =nrow(obj$T))
    #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]
    suppressWarnings(sim.row[i, ] <- options_sites$FUN_test_statistics$test(obj.row
,options = options_sites))
    suppressWarnings(sim.col[i, ] <- options_species$FUN_test_statistics$test(obj.c
ol,options = options_species))
    if (!i%%options$print){
      iteration <- i
      print(iteration)
      if (options$filek >=0 ) save(options, sim.row, sim.col, obs, iteration, file
=paste("perm_r_c",options$filek,".rdata", sep = ""))
    }
  }
  ialpha <- 1/options$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- i
sna.r  + 1)
    pval.col <- (sum(abs(sim.col) >= abs(obs[2,]), na.rm = TRUE) + 1)  / (nrepet- i
sna.c  + 1)
    if (options$falseSize){
      size.row <- (sum(sim.row >= ialpha, na.rm=TRUE))/ (nrepet - isna.r)
      size.col <- (sum(sim.col >= ialpha, na.rm=TRUE))/ (nrepet - isna.c)
      size.max <- (sum(max(sim.row,sim.col) >= ialpha, na.rm=TRUE))/ (nrepet - isna
.c)
    }
  } else {
    obs.mat.row <- matrix(rep(abs(obs[1,]),each=nrepet), nrow= nrepet, ncol=ncol(ob
s))
    isna.r <-  colSums(is.na(sim.row))
    pval.row <- (colSums(abs(sim.row) >= obs.mat.row, na.rm=TRUE) + 1)/ (nrepet - i
sna.r + 1)
    isna.c <-  colSums(is.na(sim.col))
    obs.mat.col <- matrix(rep(abs(obs[2,]),each=nrepet), nrow= nrepet, ncol=ncol(ob
s))
    pval.col <- (colSums(abs(sim.col) >= obs.mat.col, na.rm=TRUE) + 1)/ (nrepet - i
sna.c + 1)
    if(options$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_1di
vp]
  #result <- t(rbind(test_stat = obs, p.site.permut = pval.row, p.species.permut =
pval.col, pmax.permut = pmax(pval.row, pval.col)))
  result <- c(c(obj$p_values[-3]), p.site.permut = pval.row, p.species.permut = pva
l.col, pmax.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))
}

syssample <- function(i = 0, exclude = 0, perm.mat = 0, n = 10){
```

```
  # function to generate either random samples (if !is.matrix(perm.mat)) or
  # a systematic sample: i-th or for i>= exclude: i+1 th row of permat (i.e. withou
t the excluded sample)
  # exclude = 0 for no exclude, or permutation number to exclude
  if (is.matrix(perm.mat)){
    if (!exclude){ ii <- i} else{
      if(i<exclude) ii <- i else {ii <- i + 1}
    }
    sam <- perm.mat[ii,]
  } else {
    sam <-sample(x=n)
  }
  return(sam)
}

f_allPerms <- function(n){
  # all permutation matrix without identity and reverse order
  library(permute)
  result <- allPerms(n)[1:(numPerms(n)/2 - 1)  ,]
  result[sample(nrow(result)),]
}
```

Code for setting options (file setoptions.r)

These are functions to set options in the MLM3 fit and max test in the functions `MLM_Wald_bte` and `PermutationTest_r_c_max`.

```
setoptions4pmax_test <- function(verbose = TRUE, FUN_test_statistics = CWMSNC, with
.MLM= FALSE
 , nrepet = 49, level = 'sites', perm.mat.spe = 0 , perm.mat.sit = 0, print = 10,
 falseSize = FALSE, alpha = 0.05, test_statistics_are_1divp = TRUE, nmax = 6, filek
= -1){
  # set options for use in PermutationTest_r_c_max
  # perm.mat.spe =0 : random sample; else number of species,
  # if perm.mat.spe or sit >= nmax : random sampling only
  # or if the number of systematic samples
  if (perm.mat.spe > 0 & perm.mat.spe < nmax) perm.mat.spe <- f_allPerms(perm.mat.s
pe)
  if (perm.mat.sit > 0 & perm.mat.sit < nmax) perm.mat.sit <- f_allPerms(perm.mat.s
it)
  if (filek ==0 ) filek <- sample(1000,1)
  list(verbose = verbose, FUN_test_statistics = FUN_test_statistics, with.MLM= with
.MLM,
       nrepet= nrepet, level = level,perm.mat.spe=perm.mat.spe, perm.mat.sit= perm.
mat.sit,        print = print,
       falseSize = falseSize, alpha = alpha, test_statistics_are_1divp =
                      test_statistics_are_1divp, filek = filek )
}


setoptions4MLM<- function(   formula = y~ poly(trait,2) + poly(env,2) + trait:env +
(1+trait|site)+ (1+env|species),
                             family = "nbinom2", K = 0,
                             model.names= c("MLM3"),
                             library = "glmmTMB", nAGQ=0,
                             estimation = FALSE, invert_p_value = TRUE, output = "s
hort",
                             with.LRT= FALSE, LRT.only = FALSE, verbose = FALSE){
  # LRT: use the LRT test  instead of the Wald test
  # if LRT == FALSE and with.LRT == TRUE then LRT is an extra test statistic; name
```

```
it in model.names!!!
  if (LRT.only) with.LRT <- TRUE
  list(formula = formula,  family =  family, model.names = model.names, library=lib
rary, nAGQ=nAGQ,
       estimation = estimation, invert_p_value = invert_p_value, K= K, output = out
put,
       with.LRT= with.LRT, LRT.only =LRT.only,  verbose = verbose)
}
```

A9.5 WA-based analyses in file WAregressions.r

The functions for the weighted averaging(WA)-based methods are in the file WAregressions.r. There are three functions for users:

- `CWMSNC_regressions`: WA regressions (CWM/SNC regressions) with permutation testing.
- `summary.CWMSNCr` : summary of `CWMSNC_regressions`
- `plot.CWMSNCr` : Plots of `CWMSNC_regressions` (main effects and trait-env association)

There are three weighing options in `CWMSNC_regressions`,

1. `weighing = "N2"` for N_2-weighted CWM/SNC regression (N_2-weighted lm)
2. `weighing = "unw"` for unweighted, giving (permutation-based) lm CWM/SNC
3. `weighing = "FC"` for the fourth-corner correlation (fourth).

There is a cutoff option in `CWMSNC_regressions` to set a minimal number of occurrences of a species and in a site (default 0, which only removes only empty sites and species without any presence; see the utility function `make_obj_for_traitenv`;see section A9.1).

The remaining functions are for internal use in the above three main functions. The tutorial also uses `WA_p_max` for checking of the fast verson of the permutational max test used in `CWMSNC_regressions`. For details on the WA-based regression, see the function `CWMSNC_regressions0`, which is used once in the fast version but repeatedly in the non-fast version of the permutational max test. The R-code for the permutational max test (function `PermutationTest_r_c_max`) has been given in section A9.4.

Code

```
# Appendix to ter Braak 2019
#New robust weighted averaging- and model-based methods for assessing trait-environ
ment relationships.

# functions for the statistical analysis of trait-environment association
# 1 CWMSNC_regressions: Weighted averaging (WA) regressions (CWM/SNC regressions) w
ith permutation testing,
#                        with and without N2- or RK (Fourth-corner correlations) wei
ghting
# 2 summary.CWMSNCr   : summary of CWMSNC_regressions
```

```
# 3 plot.CWMSNCr      : Plots of CWMSNC_regressions (main effects and trait-env ass
ociation)
#
# the remaining functions are for internal use in the above three main functions



CWMSNC_regressions<-function(E, L=0, T=0, weighing = "N2", cutoff = 0, nrepet = 499
){
  # combination of CWM- and SNC-based regressions, possibly weighted.
  # @param E  object from class TE_obj from make_obj_for_traitenv(E, L, T,cutoff) o
r n-vector or nx1 matrix with the environmental values
  # @param L  n x m matrix of abundance values or nonspecified if E = object from f
unction make_obj_for_traitenv(E,L,T,cutoff)
  # @param T  m-vector or m x 1 matrix with trait values or nonspecified if E = obj
ect from function make_obj_for_traitenv(E,L,T,cutoff)
  # @ weighing = 0 or "unw": no weights
  #          = 1 or "RK" or "FC"  weights are the row and columns totals of L, as i
n the original fourth-corner
  #          = 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 spe
cies

  if(class(E)[1]!="TE_obj"){
    obj <- make_obj_for_traitenv(E,  L,  T,cut_off = cutoff)
  } else obj <- E
  result <- CWMSNC_regressions0(obj, weighing = weighing)
  # correlations
  result$wcorCWMSNC <- with(result, c(wcor(CWM,E),wcor(SNC,T)))
  names( result$wcorCWMSNC) <-  paste(result$weighing, c("_CWM_E","_SNC_T"),"_cor",
sep="")
  result$wFC <- with(result,FC_cor_generalized(obj, wE = wsites, wT = wspecies, R =
R, K = K))
  names(result$wFC) <- paste(result$weighing, c("_sites","_species", "_fourth_corne
r"),"_cor", sep="")
  # p-values of association
  result$p_values <- WA_p_max1(obj, weighing= weighing, nrepet =nrepet)$p_values
  attr(result$p_values, which = "nrepet") <- nrepet
  class(result) <- "CWMSNCr"
  return(result)
}

summary.CWMSNCr <- function(result, digits = 3){
  # result should be an object resulting from CWMSNC_regressions
  res <- rbind(result$wFC, result$p_values[-c(1,2)])
  rownames(res) <- paste(c("correlations", "p-values"), " (", result$weighing, ")"
, sep = "")
  colnames(res) <- c("sites","species","min/max")
  attr(res, "nrepet")<- attr(result$p_values, which = "nrepet")
  print(res, digits = digits)
}


plot.CWMSNCr <- function(result,title = paste(result$weighing, '-regressions of CWM
on env and SNC on trait',sep= "") ){
  # plot of a CWMSNCr object

  #title = paste(result$weighing, '-regressions of CWM on env and SNC on trait',sep
= "")

  library(ggplot2)

  ETdat <- with(result, data.frame( E,  CWM, R, wsites, meanT, wmeanCWM, level = "s
```

```
ites"))
  names(ETdat) <- c("E", "CWM", "R", "N2", "meanT","wmeanCWM" ,"level")

  TEdat <- with(result, data.frame( T,  SNC, K, wspecies, meanE, wmeanSNC, level =
"species"))
  names(TEdat) <- c("T", "SNC", "K", "N2", "meanE","wmeanSNC","level")


  ETTEdat <- suppressMessages(dplyr::full_join(ETdat,TEdat))

  # facet plot ------------------------------------------------------------
  suppressMessages(p.main <-  ggplot(data= ETTEdat) +
                      geom_point(aes(x= E, y = log(R), size = N2),shape = 1) +  stat
_smooth(aes(x= E, y = log(R), weight=N2), method = "gam") +
                      geom_point(aes(x= T, y = log(K), size = N2), shape = 16) + sta
t_smooth(aes(x= T, y = log(K), weight=N2), method = "gam") +
                      xlab(" env
trait")+
                      ylab(" log totals") +
                      facet_wrap(~level, scales = "free") +
                      ggtitle("log site and species totals vs env and trait"))
  suppressMessages(p.assoc <-  ggplot(data= ETTEdat) +
                      geom_point(aes(x= E, y = CWM, size = N2),shape = 1) +  stat_sm
ooth(aes(x= E, y = CWM, weight=N2), method = "gam") +
                      geom_point(aes(x= T, y = SNC, size = N2), shape = 16) + stat_s
mooth(aes(x= T, y = SNC, weight=N2), method = "gam") +
                      xlab(" env
trait")+
                      ylab("weighted means") +
                      geom_line(aes(x = E, y = meanT), linetype = 'dotted', size = 1
)+
                      geom_line(aes(x = T, y = meanE), linetype = 'dotted', size = 1
) +
                      geom_line(aes(x = E, y = wmeanCWM), linetype = 'longdash', siz
e = 1)+
                      geom_line(aes(x = T, y = wmeanSNC), linetype = 'longdash', siz
e = 1) +
                      facet_wrap(~level, scales = "free") +
                      ggtitle(title))
  #  print(p.main)
  #  print(p.assoc)
  return(list(p.main,p.assoc))
}

# the remaining functions are for internal use in the above three main functions

FC_cor_generalized <- function(E, L=0, T=0, wE=rep(1,nrow(L)) , wT= rep(1,ncol(L)),
wNam = "w", R = rowSums(L), K = colSums(L)){
  # generalized fourth-corner correlation: wE and wT are weights for sites and spec
ies respectively.
  # if wE == R and wT == K it gives the fourth-corner correlation
  # value: three correlations, one for sites and one for species and the signed min
imum (0 if different in sign)

  # local function
  f_cor_min <- function(x){
    # takes the signed minimum correlation if the correlations have the same sign (
close to min)
    # 0  if the correlations differ in sign
    if (sum(is.na(x))) return(NA)
    if (prod(x) < 0 ) { # the two elements in x diffent sign
      g <- 0
    } else {
      g <-  sign(x[1]) * min(abs(x)) # min
    }
```

```
    g
  }
  # end local function
  if(class(E)[1]!="TE_obj"){
    obj <- make_obj_for_traitenv(E,L, T,cutoff)
  } else obj <- E
  wE <- wE/sum(wE)
  wT <- wT/sum(wT)
  result <- with(obj, {
    Estd_w <- standardize_w(E,wE)
    Tstd_w <-  standardize_w(T,wT)
    Lw_si <- diag(wE/R)%*% L
    Lw_sp <-  L %*% diag(wT/K)
    #Fourthcorner  <- t(Estd_w)  %*% (L%*% Tstd_w)
    r_wE <- (t(Estd_w) %*% (Lw_si %*% Tstd_w))/sum(wE) # sum(wE)== sum(Lw_si)
    r_wT <- sum((t(Lw_sp)%*% Estd_w) *  Tstd_w)/sum(wT) # sum(wT)== sum(Lw_sp)
    result <- c(r_wE, r_wT)
    result <- c(result, f_cor_min(result))
    names(result) <-paste(wNam, c("wFC_r_sites", "wFC_r_species", "wFC_r_min"), sep
= "_")
    return(result)
  })
  return(result)
}



# Hill number of order 2: N2
fN2 <- function(x){x <- x/sum(x); 1/sum(x*x)}


CWMSNC_regressions0<-function(E,L=0,T=0, weighing = 0, cutoff = 0){
# combination of CWM- and SNC-based regressions, possibly weighted.
# @param E  object from class TE_obj from make_obj_for_traitenv(E,L,T,cutoff) or n-
vector or nx1 matrix with the environmental values
# @param L  n x m matrix of abundance values or nonspecified if E = object from fun
ction make_obj_for_traitenv(E,L,T,cutoff)
# @param T  m-vector or m x 1 matrix with trait values or nonspecified if E = objec
t from function make_obj_for_traitenv(E,L,T,cutoff)
# @ weighing = 0 or "unw": no weights
#          = 1 or "RK" or "FC" weights are the row and columns totals of L, as in t
he original fourth-corner
#          = 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 speci
es

  if(class(E)[1]!="TE_obj"){
    obj <- make_obj_for_traitenv(E1 = E, L = L,T1 = T,cutoff)
  } else obj <- E
  result <- with(obj,{
      R <- rowSums(L) # the site totals
      K <- colSums(L) # the species totals
      if (weighing %in% c(0,"unw") ){
        wsites <- rep(1, nrow(L))
        wspecies <-rep(1,ncol(L))
        weighing = "unw"
      } else if (weighing %in% c(1,"RK","FC")){
        wsites <- R
        wspecies <-K
        weighing = "RKw"
      } else if (weighing %in% c(2,"N2")){
        N2_si <- apply(L, 1, fN2)
        N2_sp <- apply(L, 2, fN2)
```

```
        wsites <- N2_si
        wspecies <- N2_sp
        weighing = "N2w"
      }
      CWM <- (L%*%T)/R #Community weighted mean wrt to trait T (mean trait value pe
r site)
      SNC <- (t(L)%*%E)/K  # Species niche centroid wrt to environmental variable E
(mean environmental value per species)
      meanE <- mean_w(E)
      meanT <- mean_w(T)
      wmeanCWM <- mean_w(CWM, w = wsites)
      wmeanSNC <- mean_w(SNC, w = wspecies)
      # site-level regression of CWM on E
      lm_CWMe <- lm(CWM~E, weights = wsites)
      if (ncol(CWM) ==1) {
        anova_site <- anova(lm_CWMe)
        F_sites <- anova_site$`F value`[1]# get F-value and p-value of site-level t
est
        Prob.site<- anova_site$"Pr(>F)"[1]
      } else {F_sites <-NA; Prob.site <- NA}
      # species-level regression of SNC on T
      lm_SNCt <- lm(SNC~T, weights =   wspecies)
      if (ncol(SNC)==1){
        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]
      } else {F_species <-NA; Prob.species <- NA}
      result<-list(p_values = c(p.site.prmtrc=Prob.site,p.species.prmtrc=Prob.speci
es,p.max.prmtrc=max(Prob.site,Prob.species)), F_values =  c(F_sites = F_sites, F_sp
ecies= F_species),
                   lm_CWMe = lm_CWMe, lm_SNCt = lm_SNCt, CWM = CWM, SNC = SNC,
                   E = E, T = T, L = L, meanE = meanE, meanT = meanT, wmeanCWM = wm
eanCWM, wmeanSNC = wmeanSNC,
                   R=R, K= K, wsites = wsites, wspecies=wspecies,weighing=weighing,
cutoff=cutoff)
    return(result)
    }
  )
  return(result)
}

CWMr<-function(E,L=0,T=0, weighing = "unw", cutoff = 0){
  # CWMr CWM-based regressions, possibly weighted.
  # @param E  object from class TE_obj from make_obj_for_traitenv(E,L,T,cutoff) or
n-vector or nx1 matrix with the environmental values
  # @param L  n x m matrix of abundance values or nonspecified if E = object from f
unction make_obj_for_traitenv(E,L,T,cutoff)
  # @param T  m-vector or m x 1 matrix with trait values or nonspecified if E = obj
ect from function make_obj_for_traitenv(E,L,T,cutoff)
  # @ weighing = 0 or "unw": no weights
  #          = 1 or "RK" or "FC" or "regular" weights are the row and columns total
s 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 spe
cies

  if(class(E)[1]!="TE_obj"){
    obj <- make_obj_for_traitenv(E,L,T,cutoff)
  } else obj <- E
  result <- with(obj,{
    R <- rowSums(L) # the site totals
    if (weighing %in% c(0,"unw") ){
      wsites <- NULL
```

```
    } else if (weighing %in% c(1,"RK", "FC")){
      wsites <- R
    } else if (weighing %in% c(2,"N2")){
      N2_si <- apply(L, 1, fN2)
      wsites <- N2_si
    }
    CWM <- (L%*%T)/R #Community weighted mean wrt to trait T (mean trait value per
site)
    # 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]
    result<-list(p_values =Prob.site, F_value = F_sites,
                 lm_CWMe = lm_CWMe,  CWM = CWM,  E = E, T = T, L = L, wsites = wsit
es,weighing=weighing,cutoff=cutoff)

    return(result)
  }
  )
  return(result)
}


SNCr<-function(E,L=0,T=0, weighing = 0, cutoff = 0){
  # CWMr CWM-based regressions, possibly weighted.
  # @param E  object from class TE_obj from make_obj_for_traitenv(E,L,T,cutoff) or
n-vector or nx1 matrix with the environmental values
  # @param L  n x m matrix of abundance values or nonspecified if E = object from f
unction make_obj_for_traitenv(E,L,T,cutoff)
  # @param T  m-vector or m x 1 matrix with trait values or nonspecified if E = obj
ect from function make_obj_for_traitenv(E,L,T,cutoff)
  # @ weighing = 0: no weights
  #          = 1 or "RK" or "FC" or  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 spe
cies

  if(class(E)[1]!="TE_obj"){
    obj <- make_obj_for_traitenv(E,L,T,cutoff)
  } else obj <- E
  result <- with(obj,{
    #R <- rowSums(L) # the site totals
    K <- colSums(L) # the species totals
    if (weighing %in% c(0,"unw") ){
      wsites <- NULL
      wspecies <-NULL
    } else if (weighing %in% c(1,"RK", "FC")){
      #wsites <- R
      wspecies <-K
    } else if (weighing %in% c(2,"N2")){
      N2_sp <- apply(L, 2, fN2)
      wspecies <-N2_sp
    }
    SNC <- (t(L)%*%E)/K  # Species niche centroid wrt to environmental variable E (
mean environmental value per species)
    #species-level regression of SNC on T
    lm_SNCt <- lm(SNC~T, weights =   wspecies)
    anova_species <- anova(lm_SNCt)
    F_species <- anova_species$`F value`[1]# get F-value and P-value of species-lev
el test
    Prob.species<- anova_species$"Pr(>F)"[1]
    result<-list(p_values =Prob.species, F_value = F_species,
```

```
                  lm_SNCt = lm_SNCt,  SNC = SNC,  E = E, T = T, L = L, wspecies = ws
pecies,weighing=weighing,cutoff=cutoff)

    return(result)
  }
  )
  return(result)
}

CWMSNCr <- function(obj, options){
  # level in c("species","sites","both")
  if(class(obj)[1]!="TE_obj") {print("obj must be of class TE_obj in CWMSNCr"); ret
urn(obj)}
  WA <- with(options,{
  if (level == "sites") WA <- c(site_p = CWMr(obj, weighing = weighing)$p_values) e
lse
    if (level == "species") {
      WA <- c(species_p =SNCr(obj, weighing =weighing )$p_values)
    } else if (level == "both") {
      WA <- c(CWMr(obj, weighing =weighing)$p_values,SNCr(obj, weighing =weighing )
$p_values)
      names(WA)<- c("site_p", "species_p")
    }
  return(WA)
  })
  if (options$invert_p_value) WA <- 1/WA
  return(WA)
}

WA_p_max <- function(obj, nrepet = 19,  weighing = "N2", score_test = TRUE, fast =
TRUE) {
  if (!fast){
    # following the description with test stat = 1/p_value of regressions performed
in full
    result <- WA_p_max0(obj, nrepet = nrepet,  weighing = weighing)
  } else {
    # short cut, which is even more general (allows multi-trait multi-envi)
    if(ncol(obj$E)==1 && ncol(obj$T == 1)) score_test = FALSE
    result <- WA_p_max1(obj, nrepet = nrepet,  weighing = weighing, wsvd = score_te
st)
  }
  return(result)
}

WA_p_max0 <- function(obj, nrepet = 19,  weighing = "N2") {
  #n_sites <- nrow(obj$L); n_species <- ncol(obj$L)

  options1 <- setoptions4pmax_test(verbose = FALSE,
                       FUN_test_statistics = list(init=function(obj,options){list(o
bj =obj, options = options)}, test=CWMSNCr),
                       with.MLM= FALSE, nrepet = nrepet,
                       perm.mat.spe = 0 , perm.mat.sit = 0, print = 1000,
                       test_statistics_are_1divp = c(TRUE), nmax = 6, filek = -1)
  options2 <- setoptions4WA(weighing = weighing , invert_p_value = TRUE, fast = FAL
SE)
  options3 <- c(options1,options2[-length(options2)])
  result <- PermutationTest_r_c_max(obj, options = options3)
  return(result)
}



# local functions
mean_w <- function(X,w = rep(1/nrow(X),nrow(X))){t(w/sum(w))%*% X}
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)), wsvd = FALSE){
  # NB requires w to be have sum 1
  ones <- rep(1,length(w))
  Xc <- X - ones %*% t(w)%*% X
  Xstd <- Xc / ones%*%sqrt(t(ones)%*%(Xc*Xc*w))
  if (wsvd) Xstd <- wSVD(Xstd, w)
return(Xstd)
}

wcor <- function(X, Y=X, w = rep(1,nrow(X))){
  # weighted correlation between matrix X and Y
  w <- w/sum(w)
  Xstd <- standardize_w(X, w)
  Ystd <- standardize_w(Y, w)
  t(Xstd) %*% diag(w) %*% Ystd
  }

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

init_WA4_pmax <- function(obj, options){
 res <- CWMSNC_regressions0(obj, weighing = options$weighing)
 Wn <- res$wsites/sum(res$wsites)
 Ws <- res$wspecies/sum(res$wspecies)
 names(res)
 E <-standardize_w(res$E, Wn, wsvd = options$wsvd)
 T <- standardize_w(res$T, Ws, wsvd = options$wsvd)
 CWM <- res$L%*%T/res$R #CWM wrt to standardized T (trait), and with wSVD orthogona
lized
 SNC <- t(res$L)%*%E/res$K # SNC wrt to standardized E (environment) and with wSVD
orthogonalized
 result <- list(E=E,T=T, CWM=CWM, SNC = SNC, p_values = res$p_values, Wn = Wn, Ws=W
s)
 return(list(obj = result, options = options))
}

#obj_init <- init_WA4pmax(obj,options)

test_stat_WA <- function(obj, options){
  test_stat <- with(obj, {
    if (options$level =="sites"){
      Estd <- standardize_w(E, Wn, wsvd = options$wsvd)
      test_stat <- t(Estd)%*%diag(Wn)%*% CWM
      #test_stat <- sum(test_stat*test_stat)
      return(t(rep(1,ncol(Estd))) %*% (test_stat*test_stat)%*% rep(1,ncol(CWM)))

    } else {
      Tstd <- standardize_w(T, Ws, wsvd = options$wsvd)
      test_stat <- t(Tstd)%*% diag(Ws)%*% SNC
      return(t(rep(1,ncol(Tstd))) %*% (test_stat*test_stat)%*% rep(1,ncol(SNC)))

    }
     #return(sum(test_stat*test_stat))
    })
 return(test_stat)
}

#test_stat_WA(obj_init,options)
```

```
WA_p_max1 <- function(obj, nrepet = 19,  weighing = "N2", wsvd = TRUE) {
  #n_sites <- nrow(obj$L); n_species <- ncol(obj$L)

  options1 <- setoptions4pmax_test(verbose = FALSE,
                                   FUN_test_statistics = list(init= init_WA4_pmax,
test= test_stat_WA),
                                   with.MLM= FALSE, nrepet = nrepet,
                                   perm.mat.spe = 0 , perm.mat.sit = 0, print = 500
0,
                                   test_statistics_are_1divp = c(FALSE), nmax = 6,
filek = -1)
  options2 <- setoptions4WA(weighing = weighing, invert_p_value = FALSE, wsvd = wsv
d, fast = TRUE)
  options3 <- c(options1,options2[-length(options2)])
  result <- PermutationTest_r_c_max(obj, options = options3)
  return(result)
}
setoptions4WA<- function(weighing = 0, level = "both", invert_p_value = TRUE, fast
= TRUE, wsvd=TRUE,  verbose = FALSE){
  list(weighing = weighing, level = level, invert_p_value = invert_p_value, fast =
fast, wsvd= wsvd, verbose = verbose)
}
```