**Supplementary Information**

to

“R package douconca: multi-trait multi-environment ordination of ecological communities by double constrained correspondence analysis.”

# Appendix S1. Skeleton dc-CA algorithm

|  |
| --- |
| library(vegan) # Step 0: Preprocess : divide by site totals. Y <- abun / rowSums(abun) # Step 1: CCA on traits and get scores which are CWMs. CCAonTraits <- cca( t(Y) ~ SLA + Seedmass, data= traitData) mlt <- sqrt((nrow(Y) - 1) / nrow(Y)) CWMs\_orthonormal\_traits <-   scores(CCAonTraits, display = "species", scaling = "species")\*mlt # Step 2: RDA on environment and get constrained site scores. RDAonEnv <- rda(  CWMs\_orthonormal\_traits ~ Moisture + Management, data = envData) eigenvals(RDAonEnv)  ## RDA1 RDA2 PC1 PC2  ## 0.22700462 0.03037791 0.06161181 0.03110378  dcCA\_constrained\_site\_scores <- scores(RDAonEnv, display="lc") # obtain the unconstrained species scores # by weighted averaging (giving species-niche centroids, SNC) # and regress these on the traits for the constrained species scores |
| Skeleton of the dc-CA algorithm with the default division by the site totals so as to obtain equal site weights (abun = site × species abundance matrix; CCA = canonical correspondence analysis, RDA = redundancy analysis). SLA and Seedmass must be traits in the trait data (traitData). Moisture and Management must be names in the environmental data (envData). The algorithm is taken from ter Braak et al. (2018). |

# Appendix S2. R code of the case studies

## Dune meadow data

### R code

rm(list=ls())  
library(douconca)  
data("dune\_trait\_env")  
  
abun <-dune\_trait\_env$comm[, -1] # must delete column/variable "Sites"  
envData <- dune\_trait\_env$envir  
traitData <- dune\_trait\_env$traits  
names(envData)[c(3,4)] <- c("Moisture","Management")  
  
mod\_funct\_traits <- dc\_CA(  
 formulaEnv = ~ Moisture + Management,  
 formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,  
 response = abun, envData, traitData)  
mod\_ecol\_traits <- dc\_CA(  
 formulaEnv = ~ Moisture + Management,  
 formulaTraits = ~F+R+N+L,  
 response = abun, envData, traitData, verbose = FALSE)  
Y <- abun / rowSums(abun)  
mod\_cca <-cca(Y~Moisture+Management, data = envData)  
mod\_CCA <- dc\_CA(formulaEnv = ~ Moisture+Management,  
 formulaTraits = ~Species,  
 response = abun, envData, traitData, verbose = FALSE)  
sum(eigenvals(mod\_cca, model = "constrained")) # 0.967  
sum(mod\_CCA$eigenvalues) # 0.967  
sum(mod\_ecol\_traits$eigenvalues) # 0.567  
sum(mod\_funct\_traits$eigenvalues) # 0.348  
#   
set.seed(1457)  
anova(mod\_ecol\_traits, by = "axis") # axis 1 P = 0.001 axis 2 0.003  
anova(mod\_funct\_traits, by = "axis")# axis 1 P = 0.048 axis 2 0.427

### Output

rm(list=ls())  
library(douconca)  
data("dune\_trait\_env")  
  
abun <-dune\_trait\_env$comm[, -1] # must delete column/variable "Sites"  
envData <- dune\_trait\_env$envir  
traitData <- dune\_trait\_env$traits  
names(envData)[c(3,4)] <- c("Moisture","Management")  
  
mod\_funct\_traits <- dc\_CA(  
 formulaEnv = ~ Moisture + Management,  
 formulaTraits = ~ SLA + Height + LDMC + Seedmass + Lifespan,  
 response = abun, envData, traitData)

## Step 1: the CCA ordination of the transposed matrix with trait constraints,  
## useful in itself and also yielding CWMs of the orthonormalized traits for step 2.  
## Call: cca(formula = tY ~ SLA + Height + LDMC +  
## Seedmass + Lifespan, data = dataTraits)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 2.3490 1.0000   
## Constrained 0.6776 0.2885 5  
## Unconstrained 1.6714 0.7115 19  
##   
## Inertia is scaled Chi-square  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## CCA1 CCA2 CCA3 CCA4 CCA5   
## 0.26839 0.19597 0.12356 0.07003 0.01967   
##   
## Eigenvalues for unconstrained axes:  
## CA1 CA2 CA3 CA4 CA5 CA6 CA7 CA8   
## 0.4386 0.2396 0.1938 0.1750 0.1429 0.1112 0.0854 0.0601   
## (Showing 8 of 19 unconstrained eigenvalues)  
##   
## Step 2: the RDA ordination of CWMs of the orthonormalized traits   
## of step 1 with environmental constraints:  
## Call: rda(formula = out1$CWMs\_orthonormal\_traits ~  
## Moisture + Management, data = out1$data$dataEnv)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 0.6776 1.0000   
## Constrained 0.3485 0.5143 4  
## Unconstrained 0.3291 0.4857 5  
##   
## Inertia is variance  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## RDA1 RDA2 RDA3 RDA4   
## 0.23379 0.09752 0.01330 0.00387   
##   
## Eigenvalues for unconstrained axes:  
## PC1 PC2 PC3 PC4 PC5   
## 0.15878 0.07151 0.05197 0.02864 0.01823   
##   
## mean, sd, VIF and canonical coefficients with their optimistic [!] t-values:  
## Avg SDS VIF Regr1 tval1  
## Moisture 2.90 7.8613 1.3001 0.3391 6.3237  
## ManagementBF 0.15 1.5969 1.4553 0.0348 0.6131  
## ManagementHF 0.25 1.9365 1.4214 0.0402 0.7170  
## ManagementNM 0.30 2.0494 1.4393 0.2792 4.9488  
## Avg SDS VIF Regr1  
## SLA 24.6468 6.3438 1.1888 -0.8181  
## Height 25.1272 15.6848 1.3033 -0.1122  
## LDMC 244.5084 70.9729 1.1791 -0.0821  
## Seedmass 0.6543 0.6688 1.0784 -0.7661  
## Lifespanperennial 0.9607 0.1944 1.0964 0.1399  
## tval1  
## SLA -3.5373  
## Height -0.4635  
## LDMC -0.3562  
## Seedmass -3.4781  
## Lifespanperennial 0.6301  
##   
## weighted variance  
## total 2.349  
## traits\_explain 0.678  
## constraintsTE 0.348  
## attr(,"meaning")  
## meaning   
## total "total inertia"   
## traits\_explain "trait-constrained inertia"   
## constraintsTE "trait-constrained inertia explained by the predictors in formulaEnv"

mod\_ecol\_traits <- dc\_CA(  
 formulaEnv = ~ Moisture + Management,  
 formulaTraits = ~F+R+N+L,  
 response = abun, envData, traitData, verbose = FALSE)  
Y <- abun / rowSums(abun)  
mod\_cca <-cca(Y~Moisture+Management, data = envData)  
mod\_CCA <- dc\_CA(formulaEnv = ~ Moisture+Management,  
 formulaTraits = ~Species,  
 response = abun, envData, traitData, verbose = FALSE)

##   
## The model is overfitted with no unconstrained  
## (residual) component

sum(eigenvals(mod\_cca, model = "constrained")) # 0.967

## [1] 0.9670062

sum(mod\_CCA$eigenvalues) # 0.967

## [1] 0.9670062

sum(mod\_ecol\_traits$eigenvalues) # 0.567

## [1] 0.5674187

sum(mod\_funct\_traits$eigenvalues) # 0.348

## [1] 0.3484763

#   
set.seed(1457)  
anova(mod\_ecol\_traits, by = "axis") # axis 1 P = 0.001 axis 2 0.003

## $species  
## Species-level permutation test using dc-CA  
## Model: dc\_CA(formulaEnv = ~Moisture + Management, formulaTraits = ~F + R + N + L, response = abun, dataEnv = envData, dataTraits = traitData, verbose = FALSE)   
## Residualized predictor permutation  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.34359 0.35531 19.7768 0.001 \*\*\*  
## dcCA2 1 0.21586 0.22322 12.4246 0.001 \*\*\*  
## dcCA3 1 0.00790 0.00817 0.4547 0.964   
## dcCA4 1 0.00007 0.00007 0.0042 1.000   
## Residual 23 0.39959   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $sites  
## Df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.34359 0.39491 17.0301 0.001 \*\*\*  
## dcCA2 1 0.21586 0.24810 10.6990 0.003 \*\*   
## dcCA3 1 0.00790 0.00908 0.3915 0.964   
## dcCA4 1 0.00007 0.00008 0.0036 1.000   
## Residual 15 0.30263   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $max  
## Max test combining the community- and species- level tests   
## Model: dc\_CA(formulaEnv = ~Moisture + Management, formulaTraits = ~F + R + N + L, response = abun, dataEnv = envData, dataTraits = traitData, verbose = FALSE)   
##   
## a mix the species- (traits) and community- (environment) levels:  
##   
## df ChiSquare F traitP envP Pr(>F)   
## dcCA1 1 0.34359 17.0301 0.001 0.001 0.001 \*\*\*  
## dcCA2 1 0.21586 12.4246 0.001 0.003 0.003 \*\*   
## dcCA3 1 0.00790 0.4547 0.964 0.964 0.964   
## dcCA4 1 0.00007 0.0042 1.000 1.000 1.000   
## Residual 23 0.39959   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

anova(mod\_funct\_traits, by = "axis")# axis 1 P = 0.048 axis 2 0.427

## $species  
## Species-level permutation test using dc-CA  
## Model: dc\_CA(formulaEnv = ~Moisture + Management, formulaTraits = ~SLA + Height + LDMC + Seedmass + Lifespan, response = abun, dataEnv = envData, dataTraits = traitData)   
## Residualized predictor permutation  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.23379 0.241767 8.3155 0.048 \*  
## dcCA2 1 0.09752 0.100851 3.4687 0.427   
## dcCA3 1 0.01330 0.013751 0.4730 0.998   
## dcCA4 1 0.00387 0.003998 0.1375 1.000   
## Residual 22 0.61853   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $sites  
## Df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.23379 0.34502 10.6545 0.001 \*\*\*  
## dcCA2 1 0.09752 0.14392 4.4444 0.044 \*   
## dcCA3 1 0.01330 0.01962 0.6060 0.916   
## dcCA4 1 0.00387 0.00571 0.1762 0.957   
## Residual 15 0.32914   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $max  
## Max test combining the community- and species- level tests   
## Model: dc\_CA(formulaEnv = ~Moisture + Management, formulaTraits = ~SLA + Height + LDMC + Seedmass + Lifespan, response = abun, dataEnv = envData, dataTraits = traitData)   
##   
## Taken from the species-level test:  
## Residualized predictor permutation  
## Permutation: free  
## Number of permutations: 999  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.23379 0.241767 8.3155 0.048 \*  
## dcCA2 1 0.09752 0.100851 3.4687 0.427   
## dcCA3 1 0.01330 0.013751 0.4730 0.998   
## dcCA4 1 0.00387 0.003998 0.1375 1.000   
## Residual 22 0.61853   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Wisconsin forest understory vegetation

### R code

**rm**(list=**ls**(all=TRUE)) *# remove all existing items from the workspace*  
  
resp<- **read.csv**("data/Wisconsin\_response.csv")  
**names**(resp)[1**:**10]  
Y <- resp[,**-**1]; **rownames**(resp) <- resp[,"X"]  
envir<- **read.csv**("data/Wisconsin\_envir.csv")[,**-**1]  
envir**$**site<- **paste0**("site", **round**(envir**$**site,0))  
traits<- **read.csv**("data/Wisconsin\_traits.csv")  
**names**(envir)  
**names**(traits)  
envir**$**BA <- **sqrt**(envir**$**BA)  
envir**$**X.N <- **log**(envir**$**X.N)  
**summary**(traits)  
nams<- **c**("SLA","VH","LS","CN","LT") *#log transformed*  
**for** (k **in** nams) traits[[k]] <- **log**(traits[[k]])  
**names**(traits)[3] <- "LMA"; traits[,3] <- **-** traits[,3]  
**names**(traits)  
keytraits <- **c**("VH","LS","LMA","LCC")  
keyenv <- **c**("MAT","MAP","TSD","X.Sand","X.N","BA")  
  
formulaEnvkey <- **as.formula**(**paste0**("~", **paste**(keyenv,collapse = "+")))  
formulaTraitskey <- **as.formula**(**paste0**("~", **paste**(keytraits,collapse = "+")))  
  
  
**library**(douconca)  
divide = TRUE  
mod1 <- **dc\_CA**(formulaEnv = formulaEnvkey,formulaTraits = formulaTraitskey,  
 response = Y, dataEnv = envir, dataTraits = traits, divideBySiteTotals = divide)  
**set.seed**(1457)  
**anova**(mod1,by="axis")  
*# the default plot*  
**plot**(mod1) *# gradient\_description = "correlation"*  
  
**plot**(mod1, gradient\_description = "tvalues")  
  
# nitty-gritty, adapt names, change widths, and flip the axis -----  
*# adapt names*  
newnams <- **getPlotdata**(mod1)**$**newNameList**$**newnames  
newnams**$**env[4**:**5] <- **c**("%Sand", "%N")  
**plot**(mod1, gradient\_description = "tvalues", widths = **c**(3,1,1), newnames = newnams, flip\_axis = TRUE, expand =1.5)

### Output

dcCAWisconsin.r

rm(list=ls(all=TRUE)) # remove all existing items from the workspace  
  
resp<- read.csv("data/Wisconsin\_response.csv")  
names(resp)[1:10]

## [1] "X" "Actaepachy" "Adianpedat" "Ageraaltis" "Agrimgrypo"  
## [6] "Alliapetio" "Alliutrico" "Ambroartem" "Ambrotrifi" "Amphibract"

Y <- resp[,-1]; rownames(resp) <- resp[,"X"]  
envir<- read.csv("data/Wisconsin\_envir.csv")[,-1]  
envir$site<- paste0("site", round(envir$site,0))  
traits<- read.csv("data/Wisconsin\_traits.csv")  
names(envir)

## [1] "site" "quads" "Lat" "Long" "MAT" "MAP" "TSD"   
## [8] "PCV" "X.Sand" "X.N" "BA"

names(traits)

## [1] "X" "sp" "SLA" "VH" "LCC" "LS" "CN" "LT" "LNC"   
## [10] "LDMC"

envir$BA <- sqrt(envir$BA)  
envir$X.N <- log(envir$X.N)  
summary(traits)

## X sp SLA VH   
## Min. : 1 Length:185 Min. : 102.8 Min. : 5.333   
## 1st Qu.: 47 Class :character 1st Qu.: 273.6 1st Qu.: 21.750   
## Median : 93 Mode :character Median : 362.1 Median : 42.667   
## Mean : 93 Mean : 389.5 Mean : 49.636   
## 3rd Qu.:139 3rd Qu.: 473.3 3rd Qu.: 65.417   
## Max. :185 Max. :1140.8 Max. :285.667   
## LCC LS CN   
## Min. :37.41 Min. : 0.0486 Min. : 7.609   
## 1st Qu.:42.77 1st Qu.: 13.7969 1st Qu.:13.606   
## Median :44.33 Median : 31.2816 Median :16.742   
## Mean :44.16 Mean : 110.0357 Mean :17.727   
## 3rd Qu.:45.83 3rd Qu.: 80.6262 3rd Qu.:20.738   
## Max. :48.27 Max. :2472.8583 Max. :43.459   
## LT LNC LDMC   
## Min. :0.07106 Min. :0.01036 Min. :0.06807   
## 1st Qu.:0.15967 1st Qu.:0.02146 1st Qu.:0.17952   
## Median :0.20324 Median :0.02571 Median :0.22549   
## Mean :0.22066 Mean :0.02738 Mean :0.23668   
## 3rd Qu.:0.25200 3rd Qu.:0.03308 3rd Qu.:0.29316   
## Max. :0.68700 Max. :0.06047 Max. :0.54234

nams<- c("SLA","VH","LS","CN","LT") #log transformed  
for (k in nams) traits[[k]] <- log(traits[[k]])  
names(traits)[3] <- "LMA"; traits[,3] <- - traits[,3]  
names(traits)

## [1] "X" "sp" "LMA" "VH" "LCC" "LS" "CN" "LT" "LNC"   
## [10] "LDMC"

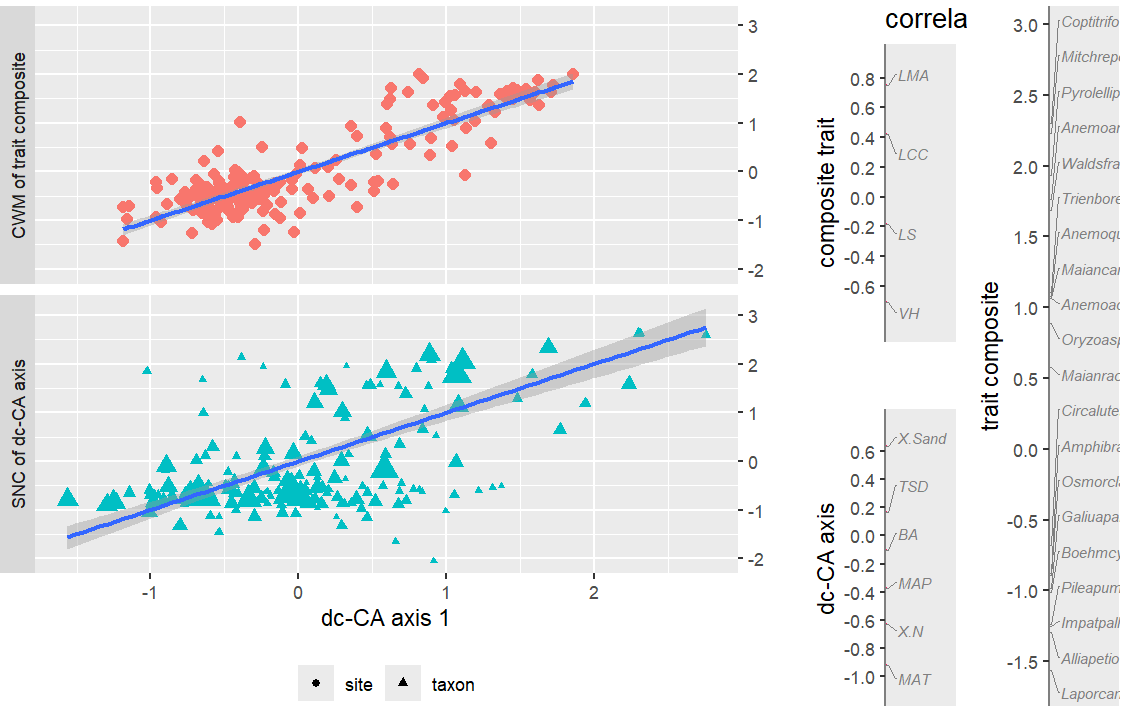
keytraits <- c("VH","LS","LMA","LCC")  
keyenv <- c("MAT","MAP","TSD","X.Sand","X.N","BA")  
  
formulaEnvkey <- as.formula(paste0("~", paste(keyenv,collapse = "+")))  
formulaTraitskey <- as.formula(paste0("~", paste(keytraits,collapse = "+")))  
  
  
library(douconca)  
divide = TRUE  
mod1 <- dc\_CA(formulaEnv = formulaEnvkey,formulaTraits = formulaTraitskey,  
 response = Y, dataEnv = envir, dataTraits = traits, divideBySiteTotals = divide)

## Step 1: the CCA ordination of the transposed matrix with trait constraints,  
## useful in itself and also yielding CWMs of the orthonormalized traits for step 2.  
## Call: cca(formula = tY ~ VH + LS + LMA + LCC, data = dataTraits)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 8.00948 1.00000   
## Constrained 0.69794 0.08714 4  
## Unconstrained 7.31154 0.91286 179  
##   
## Inertia is scaled Chi-square  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## CCA1 CCA2 CCA3 CCA4   
## 0.4492 0.1395 0.0700 0.0392   
##   
## Eigenvalues for unconstrained axes:  
## CA1 CA2 CA3 CA4 CA5 CA6 CA7 CA8   
## 0.5359 0.4100 0.3389 0.2853 0.2578 0.2482 0.2342 0.2172   
## (Showing 8 of 179 unconstrained eigenvalues)  
##   
## Step 2: the RDA ordination of CWMs of the orthonormalized traits   
## of step 1 with environmental constraints:  
## Call: rda(formula = out1$CWMs\_orthonormal\_traits ~ MAT + MAP + TSD  
## + X.Sand + X.N + BA, data = out1$data$dataEnv)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 0.6979 1.0000   
## Constrained 0.3731 0.5346 4  
## Unconstrained 0.3248 0.4654 4  
##   
## Inertia is variance  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## RDA1 RDA2 RDA3 RDA4   
## 0.3358 0.0315 0.0055 0.0003   
##   
## Eigenvalues for unconstrained axes:  
## PC1 PC2 PC3 PC4   
## 0.11926 0.10911 0.06018 0.03626   
##   
## mean, sd, VIF and canonical coefficients with their optimistic [!] t-values:  
## Avg SDS VIF Regr1 tval1  
## MAT 7.4346 18.6444 1.8605 -0.4633 -13.6170  
## MAP 871.1496 671.3967 1.3141 -0.0373 -1.3032  
## TSD 11.1728 6.1386 1.3080 -0.1323 -4.6380  
## X.Sand 51.3152 290.7371 1.5450 0.0704 2.2707  
## X.N -1.3053 7.2435 1.4969 -0.1861 -6.0980  
## BA 4.8269 12.2454 1.0815 0.0184 0.7076  
## Avg SDS VIF Regr1 tval1  
## VH 3.4328 0.6483 1.2462 -0.5975 -7.7126  
## LS 4.0828 1.5063 1.3598 -0.0276 -0.3414  
## LMA -6.0860 0.3689 1.0817 0.5688 7.8794  
## LCC 44.4614 2.0866 1.1830 0.3462 4.5858  
##   
## weighted variance  
## total 8.009  
## traits\_explain 0.698  
## constraintsTE 0.373  
## attr(,"meaning")  
## meaning   
## total "total inertia"   
## traits\_explain "trait-constrained inertia"   
## constraintsTE "trait-constrained inertia explained by the predictors in formulaEnv"

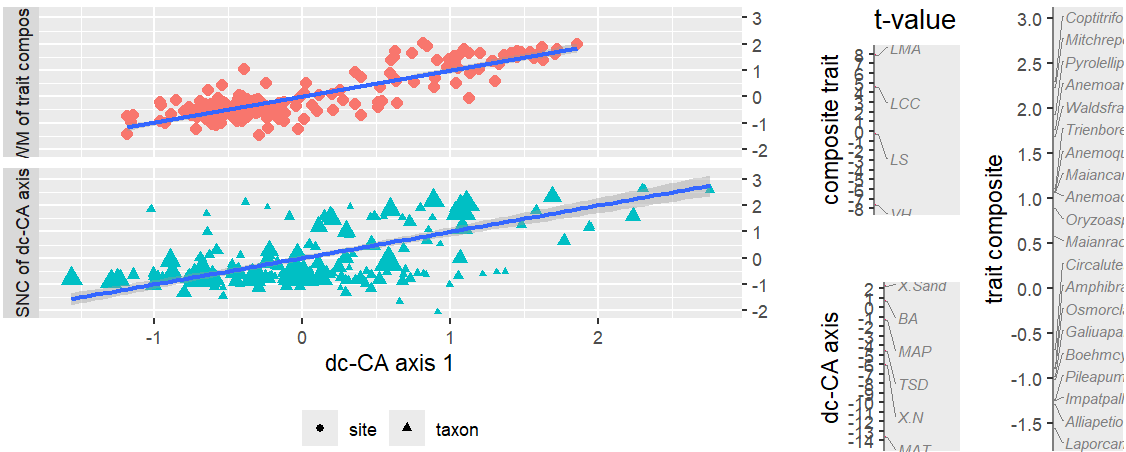
set.seed(1457)  
anova(mod1,by="axis")

## $species  
## Species-level permutation test using dc-CA  
## Model: dc\_CA(formulaEnv = formulaEnvkey, formulaTraits = formulaTraitskey, response = Y, dataEnv = envir, dataTraits = traits, divideBySiteTotals = divide)   
## Residualized predictor permutation  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.33581 0.259882 65.7702 0.001 \*\*\*  
## dcCA2 1 0.03147 0.024356 6.1641 0.502   
## dcCA3 1 0.00550 0.004253 1.0764 0.984   
## dcCA4 1 0.00034 0.000262 0.0664 1.000   
## Residual 180 0.91906   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $sites  
## Df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.33581 0.48115 190.2322 0.001 \*\*\*  
## dcCA2 1 0.03147 0.04509 17.8288 0.001 \*\*\*  
## dcCA3 1 0.00550 0.00787 3.1133 0.319   
## dcCA4 1 0.00034 0.00049 0.1921 1.000   
## Residual 184 0.32481   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $max  
## Max test combining the community- and species- level tests   
## Model: dc\_CA(formulaEnv = formulaEnvkey, formulaTraits = formulaTraitskey, response = Y, dataEnv = envir, dataTraits = traits, divideBySiteTotals = divide)   
##   
## Taken from the species-level test:  
## Residualized predictor permutation  
## Permutation: free  
## Number of permutations: 999  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.33581 0.259882 65.7702 0.001 \*\*\*  
## dcCA2 1 0.03147 0.024356 6.1641 0.502   
## dcCA3 1 0.00550 0.004253 1.0764 0.984   
## dcCA4 1 0.00034 0.000262 0.0664 1.000   
## Residual 180 0.91906   
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

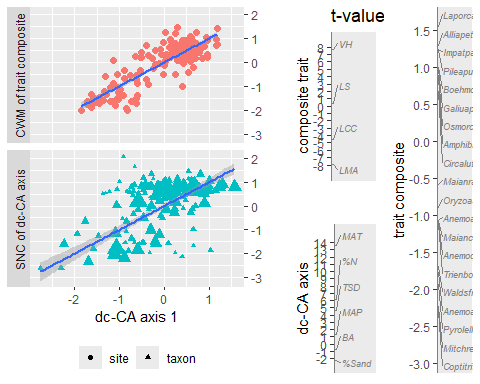
# the default plot  
plot(mod1) # gradient\_description = "correlation"



plot(mod1, gradient\_description = "tvalues")



# nitty-gritty, adapt names, change widths, and flip the axis --------------  
# adapt names  
newnams <- getPlotdata(mod1)$newNameList$newnames  
newnams$env[4:5] <- c("%Sand", "%N")  
plot(mod1, gradient\_description = "tvalues", widths = c(3,1,1), newnames = newnams, flip\_axis = TRUE, expand =1.5)



## Amazonian forest trees

### R code

rm(list=ls(all=TRUE))   
  
#Pinho, Bruno; ter Braak, Cajo; P. L. Melo, Felipe;  
#Bauman, David; Barlow, Jos (2024).   
#Data and code from Pinho et al. - Winner-Loser plant trait replacements in  
#human-modified tropical forests.   
#figshare. Dataset. https://doi.org/10.6084/m9.figshare.25565169  
  
# see README.txt for further use of this data  
  
# Libraries----  
library(data.table)  
library(tidyr)  
library(dplyr)  
library(vegan)  
library(tibble)  
  
#remotes::install\_github("CajoterBraak/douconca") # or:  
#install.packages("douconca") # preferred   
library(douconca)#dc-CA  
  
# Data----  
comm <- fread("Data/comm.csv")  
comm <- dcast.data.table(data = comm, Region\_Plot ~ Binomial\_correct)  
comm <- column\_to\_rownames(comm, "Region\_Plot")  
  
env <- fread("Data/env.csv")   
env$invBA\_ha <- 1/ env$BA\_ha  
  
traits <- fread("Data/traits.csv")  
traits <- column\_to\_rownames(traits, "Binomial\_correct")  
  
#Create community matrix  
  
# Analysis----  
  
# an initial analysis required   
# all traits are being used in this selection as formulaTraits =~.  
names(traits)  
  
out1 <- dc\_CA(formulaEnv = ~lnFL2000 + Condition(Region), formulaTraits =~.,   
 response = comm, dataEnv = env, dataTraits = traits)  
  
out1$eigenvalues  
  
## Fitting models according to the DAG   
## (as in Fig.3), but now using dc-CA   
  
### step 1 Forest loss------------------------------------------------------------------  
  
considered <- NULL  
consider <- c("lnFL500","lnFL1000", "lnFL2000")  
names(consider) <- consider  
consider  
  
fit\_measures <- matrix(NA, nrow = length(consider), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- consider  
  
cntr <- how(within = Within(type = "free"), plots = Plots(strata= env$Region, type = "none"), nper= 1999)  
set.seed(123)  
  
test <- TRUE  
  
for (k in seq\_along(consider)){  
   
 formulaE\_FS <- as.formula(paste("~", consider[k], "+Condition(Region)" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
  
pvaladj3 <- c(p.adjust(fit\_measures[1:3,"pval1"], method = "holm"), rep(NA, nrow(fit\_measures)-3))  
#pvaladj <- p.adjust(fit\_measures[,"pval1"], method = "holm")  
fit\_measures <- cbind(fit\_measures,pvaladj3)  
round(fit\_measures, 5)   
# best = lnFL2000 ; all significant with and without correction for multiple testing  
  
fit\_measuresL <- list()  
fit\_measuresL[[1]] <- fit\_measures  
  
considered <- c(considered, consider["lnFL2000"])  
considered  
  
### step 2 Fragmentation------------------------------------------------------------------  
stepk <-2  
  
considerk <- c("lnNfrag500","lnNfrag1000", "lnNfrag2000")  
  
fit\_measures <- matrix(NA, nrow = length(considerk), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- considerk  
  
for (k in seq\_along(considerk)){  
   
 formulaE\_FS <- as.formula(paste("~", considerk[k], "+Condition(Region +", paste(considered, collapse = "+") ,")", sep ="" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
pvaladj3 <- c(p.adjust(fit\_measures[1:3,"pval1"], method = "holm"), rep(NA, nrow(fit\_measures)-3))  
#pvaladj <- p.adjust(fit\_measures[,"pval1"], method = "holm")  
fit\_measures <- cbind(fit\_measures,pvaladj3 )  
round(fit\_measures, 5)   
# best = lnNfrag1000 ; not significant, neither with nor without multiple testing correction  
  
fit\_measuresL[[stepk]] <- fit\_measures  
  
considered <- c(considered, "lnNfrag1000")  
considered  
  
### step 3 Edge density------------------------------------------------------------------  
  
stepk <-3  
considerk <- c("ln\_edg500" , "ln\_edg1000" ,"ln\_edg2000")  
  
fit\_measures <- matrix(NA, nrow = length(considerk), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- considerk  
  
for (k in seq\_along(considerk)){  
   
 formulaE\_FS <- as.formula(paste("~", considerk[k], "+Condition(Region +", paste(considered, collapse = "+") ,")", sep ="" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
pvaladj3 <- c(p.adjust(fit\_measures[1:3,"pval1"], method = "holm"), rep(NA, nrow(fit\_measures)-3))  
fit\_measures <- cbind(fit\_measures,pvaladj3)  
round(fit\_measures, 5) #  
# best ln\_edg500  
  
fit\_measuresL[[stepk]] <- fit\_measures #  
fit\_measuresL  
considered <- c(considered, "ln\_edg500")  
considered  
### step 4 Local degradation------------------------------------------------------------------  
  
stepk <-4  
considerk <- c("invBA\_ha")  
  
fit\_measures <- matrix(NA, nrow = length(considerk), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- considerk  
  
for (k in seq\_along(considerk)){  
   
 formulaE\_FS <- as.formula(paste("~", considerk[k], "+Condition(Region +", paste(considered, collapse = "+") ,")", sep ="" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
round(fit\_measures, 5)  
  
fit\_measuresL[[stepk]] <- fit\_measures  
fit\_measuresL  
  
  
# Final model with lnFL2000 and local degradation -------------------------  
  
  
outF <- dc\_CA(formulaEnv = ~lnFL2000 + invBA\_ha + Condition(Region), formulaTraits =~.,  
 response = comm, dataEnv = env, dataTraits = traits)  
  
anova(outF, by= "axis", permutations = list(999,cntr))  
  
plot(outF, widths = c(3,1,1.2), remove\_centroids = TRUE)  
  
  
# Extra code showing log(invBA\_ha) ----------------------------------------  
  
# avoids an outlier  
  
outF2 <- dc\_CA(formulaEnv = ~lnFL2000 + log(invBA\_ha) + Condition(Region), formulaTraits =~.,  
 response = comm, dataEnv = env, dataTraits = traits)  
  
anova(outF2, by= "axis", permutations = list(999,cntr))  
  
  
plot(outF2, widths = c(3,1,1.2), remove\_centroids = TRUE, flip\_axis = TRUE)

### Output

multivariate\_analysis.R

rm(list=ls(all=TRUE))   
  
#Pinho, Bruno; ter Braak, Cajo; P. L. Melo, Felipe;  
#Bauman, David; Barlow, Jos (2024).   
#Data and code from Pinho et al. - Winner-Loser plant trait replacements in  
#human-modified tropical forests.   
#figshare. Dataset. https://doi.org/10.6084/m9.figshare.25565169  
  
# see README.txt for further use of this data  
  
# Libraries----  
library(data.table)

## data.table 1.14.4 using 2 threads (see ?getDTthreads). Latest news: r-datatable.com

library(tidyr)  
library(dplyr)

##   
## Attaching package: 'dplyr'

## The following objects are masked from 'package:data.table':  
##   
## between, first, last

## The following objects are masked from 'package:stats':  
##   
## filter, lag

## The following objects are masked from 'package:base':  
##   
## intersect, setdiff, setequal, union

library(vegan)

## Loading required package: permute

## Loading required package: lattice

## This is vegan 2.6-8

library(tibble)  
  
#remotes::install\_github("CajoterBraak/douconca") # or:  
#install.packages("douconca") # preferred   
library(douconca)#dc-CA  
  
# Data----  
comm <- fread("Data/comm.csv")  
comm <- dcast.data.table(data = comm, Region\_Plot ~ Binomial\_correct)

## Using 'BA' as value column. Use 'value.var' to override

## Aggregate function missing, defaulting to 'length'

comm <- column\_to\_rownames(comm, "Region\_Plot")  
  
env <- fread("Data/env.csv")   
env$invBA\_ha <- 1/ env$BA\_ha  
  
traits <- fread("Data/traits.csv")  
traits <- column\_to\_rownames(traits, "Binomial\_correct")  
  
#Create community matrix  
  
# Analysis----  
  
# an initial analysis required   
# all traits are being used in this selection as formulaTraits =~.  
names(traits)

## [1] "WD" "lnSM" "LMA" "Hmax" "DS"

out1 <- dc\_CA(formulaEnv = ~lnFL2000 + Condition(Region), formulaTraits =~.,   
 response = comm, dataEnv = env, dataTraits = traits)

## Warning in dc\_CA(formulaEnv = ~lnFL2000 + Condition(Region), formulaTraits = ~., : variableCWD has missing values and is deleted from the environmental data.

## Step 1: the CCA ordination of the transposed matrix with trait constraints,  
## useful in itself and also yielding CWMs of the orthonormalized traits for step 2.  
## Call: cca(formula = tY ~ WD + lnSM + LMA + Hmax +  
## DS, data = dataTraits)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 28.23366 1.00000   
## Constrained 0.47361 0.01677 6  
## Unconstrained 27.76005 0.98323 270  
##   
## Inertia is scaled Chi-square  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## CCA1 CCA2 CCA3 CCA4 CCA5 CCA6   
## 0.20958 0.07642 0.06127 0.04692 0.04579 0.03364   
##   
## Eigenvalues for unconstrained axes:  
## CA1 CA2 CA3 CA4 CA5 CA6 CA7   
## 0.8026 0.6536 0.6198 0.6021 0.5610 0.5342 0.4894   
## CA8   
## 0.4346   
## (Showing 8 of 270 unconstrained eigenvalues)  
##   
## Step 2: the RDA ordination of CWMs of the orthonormalized traits   
## of step 1 with environmental constraints:  
## Call: rda(formula = out1$CWMs\_orthonormal\_traits ~  
## lnFL2000 + Condition(Region), data =  
## out1$data$dataEnv)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 0.47361 1.00000   
## Conditional 0.06598 0.13930 5  
## Constrained 0.04775 0.10082 1  
## Unconstrained 0.35989 0.75988 6  
##   
## Inertia is variance  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## RDA1   
## 0.04775   
##   
## Eigenvalues for unconstrained axes:  
## PC1 PC2 PC3 PC4 PC5 PC6   
## 0.15012 0.05697 0.05161 0.03921 0.03550 0.02648   
##   
## mean, sd, VIF and canonical coefficients with their optimistic [!] t-values:  
## Avg SDS VIF Regr1 tval1  
## RegionPB 0.0627 3.9917 1.3706 0.0114 0.4110  
## RegionPGM 0.3469 7.8355 2.5118 -0.0976 -2.6019  
## RegionSGD 0.1365 5.6523 1.7273 0.0040 0.1282  
## RegionSTM 0.2325 6.9537 2.3089 -0.1058 -2.9433  
## RegionUna 0.0849 4.5878 1.4942 -0.0221 -0.7649  
## lnFL2000 -1.7400 25.7541 1.2629 -0.2456 -9.2355  
## Avg SDS VIF Regr1 tval1  
## WD 0.6374 0.1729 1.4905 0.5438 9.5432  
## lnSM 5.6438 2.4597 1.4762 0.5540 9.7676  
## LMA 82.9936 24.1529 1.0145 -0.0197 -0.4194  
## Hmax 28.3753 8.7913 1.0652 -0.0711 -1.4764  
## DSNonZoo 0.1693 0.3750 1.0911 0.2793 5.7275  
## DSSynZoo 0.1216 0.3269 1.2521 0.0814 1.5587  
##   
## weighted variance  
## total 28.234  
## traits\_explain 0.474  
## conditionE 0.066  
## constraintsTE 0.048  
## attr(,"meaning")  
## meaning   
## total "total inertia"   
## traits\_explain "trait-constrained inertia"   
## conditionE "trait-constrained inertia explained by the condition in formulaEnv"   
## constraintsTE "trait-constrained inertia explained by the predictors in formulaEnv"

out1$eigenvalues

## dcCA1   
## 0.04774874

## Fitting models according to the DAG   
## (as in Fig.3), but now using dc-CA   
  
### step 1 Forest loss------------------------------------------------------------------  
  
considered <- NULL  
consider <- c("lnFL500","lnFL1000", "lnFL2000")  
names(consider) <- consider  
consider

## lnFL500 lnFL1000 lnFL2000   
## "lnFL500" "lnFL1000" "lnFL2000"

fit\_measures <- matrix(NA, nrow = length(consider), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- consider  
  
cntr <- how(within = Within(type = "free"), plots = Plots(strata= env$Region, type = "none"), nper= 1999)  
set.seed(123)  
  
test <- TRUE  
  
for (k in seq\_along(consider)){  
   
 formulaE\_FS <- as.formula(paste("~", consider[k], "+Condition(Region)" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
  
pvaladj3 <- c(p.adjust(fit\_measures[1:3,"pval1"], method = "holm"), rep(NA, nrow(fit\_measures)-3))  
#pvaladj <- p.adjust(fit\_measures[,"pval1"], method = "holm")  
fit\_measures <- cbind(fit\_measures,pvaladj3)  
round(fit\_measures, 5)

## variance pval1 pvaladj3  
## lnFL500 0.03328 5e-04 0.0015  
## lnFL1000 0.04092 5e-04 0.0015  
## lnFL2000 0.04775 5e-04 0.0015

# best = lnFL2000 ; all significant with and without correction for multiple testing  
  
fit\_measuresL <- list()  
fit\_measuresL[[1]] <- fit\_measures  
  
considered <- c(considered, consider["lnFL2000"])  
considered

## lnFL2000   
## "lnFL2000"

### step 2 Fragmentation------------------------------------------------------------------  
stepk <-2  
  
considerk <- c("lnNfrag500","lnNfrag1000", "lnNfrag2000")  
  
fit\_measures <- matrix(NA, nrow = length(considerk), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- considerk  
  
for (k in seq\_along(considerk)){  
   
 formulaE\_FS <- as.formula(paste("~", considerk[k], "+Condition(Region +", paste(considered, collapse = "+") ,")", sep ="" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
pvaladj3 <- c(p.adjust(fit\_measures[1:3,"pval1"], method = "holm"), rep(NA, nrow(fit\_measures)-3))  
#pvaladj <- p.adjust(fit\_measures[,"pval1"], method = "holm")  
fit\_measures <- cbind(fit\_measures,pvaladj3 )  
round(fit\_measures, 5)

## variance pval1 pvaladj3  
## lnNfrag500 0.00274 0.0840 0.1680  
## lnNfrag1000 0.00341 0.0415 0.1245  
## lnNfrag2000 0.00246 0.1345 0.1680

# best = lnNfrag1000 ; not significant, neither with nor without multiple testing correction  
  
fit\_measuresL[[stepk]] <- fit\_measures  
  
considered <- c(considered, "lnNfrag1000")  
considered

## lnFL2000   
## "lnFL2000" "lnNfrag1000"

### step 3 Edge density------------------------------------------------------------------  
  
stepk <-3  
considerk <- c("ln\_edg500" , "ln\_edg1000" ,"ln\_edg2000")  
  
fit\_measures <- matrix(NA, nrow = length(considerk), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- considerk  
  
for (k in seq\_along(considerk)){  
   
 formulaE\_FS <- as.formula(paste("~", considerk[k], "+Condition(Region +", paste(considered, collapse = "+") ,")", sep ="" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
pvaladj3 <- c(p.adjust(fit\_measures[1:3,"pval1"], method = "holm"), rep(NA, nrow(fit\_measures)-3))  
fit\_measures <- cbind(fit\_measures,pvaladj3)  
round(fit\_measures, 5) #

## variance pval1 pvaladj3  
## ln\_edg500 0.00323 0.0420 0.1260  
## ln\_edg1000 0.00165 0.2665 0.2665  
## ln\_edg2000 0.00281 0.0685 0.1370

# best ln\_edg500  
  
fit\_measuresL[[stepk]] <- fit\_measures #  
fit\_measuresL

## [[1]]  
## variance pval1 pvaladj3  
## lnFL500 0.03328161 5e-04 0.0015  
## lnFL1000 0.04092097 5e-04 0.0015  
## lnFL2000 0.04774874 5e-04 0.0015  
##   
## [[2]]  
## variance pval1 pvaladj3  
## lnNfrag500 0.002742606 0.0840 0.1680  
## lnNfrag1000 0.003408878 0.0415 0.1245  
## lnNfrag2000 0.002458787 0.1345 0.1680  
##   
## [[3]]  
## variance pval1 pvaladj3  
## ln\_edg500 0.003231004 0.0420 0.1260  
## ln\_edg1000 0.001647057 0.2665 0.2665  
## ln\_edg2000 0.002810637 0.0685 0.1370

considered <- c(considered, "ln\_edg500")  
considered

## lnFL2000   
## "lnFL2000" "lnNfrag1000" "ln\_edg500"

### step 4 Local degradation------------------------------------------------------------------  
  
stepk <-4  
considerk <- c("invBA\_ha")  
  
fit\_measures <- matrix(NA, nrow = length(considerk), ncol = 2)  
colnames(fit\_measures) <- c("variance","pval1")  
rownames(fit\_measures) <- considerk  
  
for (k in seq\_along(considerk)){  
   
 formulaE\_FS <- as.formula(paste("~", considerk[k], "+Condition(Region +", paste(considered, collapse = "+") ,")", sep ="" ))  
   
 out\_FS <- dc\_CA(formulaE\_FS,  
 dc\_CA\_object = out1, verbose = FALSE)  
   
 if (test) {  
 an <- anova(out\_FS$RDAonEnv, permutations = cntr)  
 pval <- an$`Pr(>F)`[1]} else pval <- NA  
 fit\_measures[k,] <- c(out\_FS$inertia["constraintsTE","weighted variance"], pval)  
}  
  
round(fit\_measures, 5)

## variance pval1  
## invBA\_ha 0.00991 5e-04

fit\_measuresL[[stepk]] <- fit\_measures  
fit\_measuresL

## [[1]]  
## variance pval1 pvaladj3  
## lnFL500 0.03328161 5e-04 0.0015  
## lnFL1000 0.04092097 5e-04 0.0015  
## lnFL2000 0.04774874 5e-04 0.0015  
##   
## [[2]]  
## variance pval1 pvaladj3  
## lnNfrag500 0.002742606 0.0840 0.1680  
## lnNfrag1000 0.003408878 0.0415 0.1245  
## lnNfrag2000 0.002458787 0.1345 0.1680  
##   
## [[3]]  
## variance pval1 pvaladj3  
## ln\_edg500 0.003231004 0.0420 0.1260  
## ln\_edg1000 0.001647057 0.2665 0.2665  
## ln\_edg2000 0.002810637 0.0685 0.1370  
##   
## [[4]]  
## variance pval1  
## invBA\_ha 0.009907882 5e-04

# Final model with lnFL2000 and local degradation -------------------------  
  
  
outF <- dc\_CA(formulaEnv = ~lnFL2000 + invBA\_ha + Condition(Region), formulaTraits =~.,  
 response = comm, dataEnv = env, dataTraits = traits)

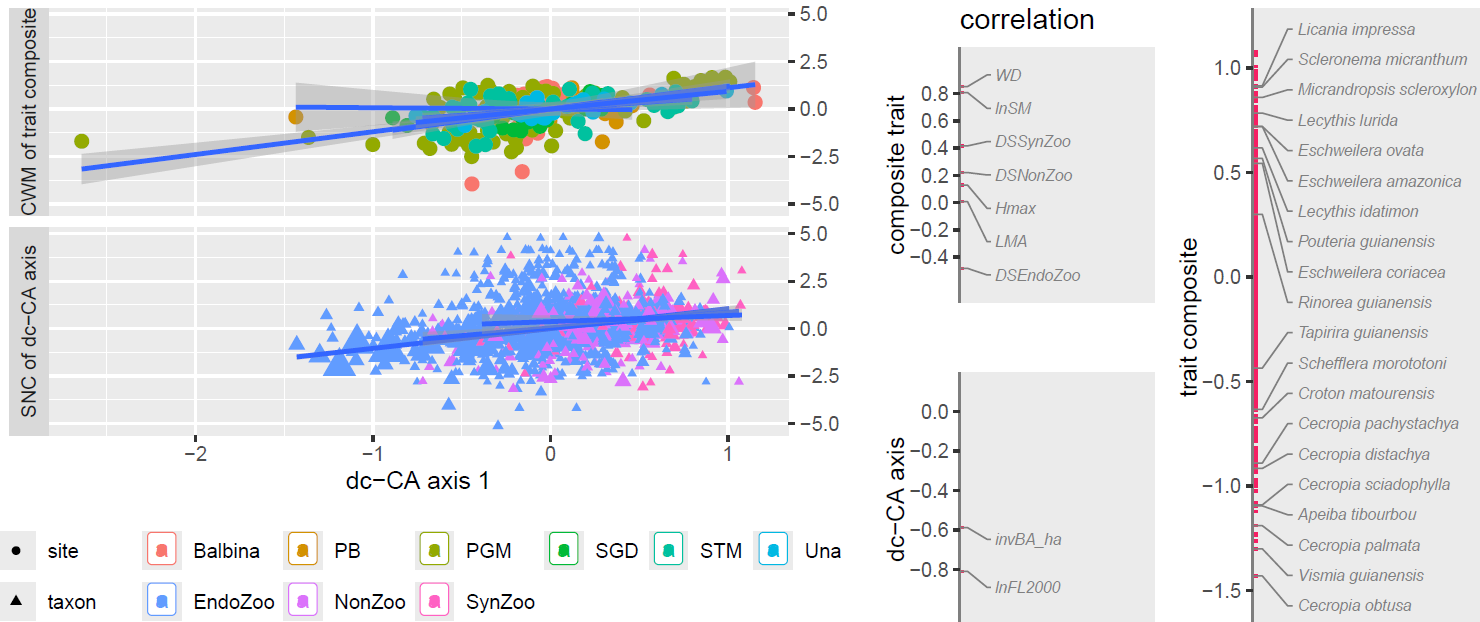
## Warning in dc\_CA(formulaEnv = ~lnFL2000 + invBA\_ha + Condition(Region), : variableCWD has missing values and is deleted from the environmental data.

## Step 1: the CCA ordination of the transposed matrix with trait constraints,  
## useful in itself and also yielding CWMs of the orthonormalized traits for step 2.  
## Call: cca(formula = tY ~ WD + lnSM + LMA + Hmax +  
## DS, data = dataTraits)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 28.23366 1.00000   
## Constrained 0.47361 0.01677 6  
## Unconstrained 27.76005 0.98323 270  
##   
## Inertia is scaled Chi-square  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## CCA1 CCA2 CCA3 CCA4 CCA5 CCA6   
## 0.20958 0.07642 0.06127 0.04692 0.04579 0.03364   
##   
## Eigenvalues for unconstrained axes:  
## CA1 CA2 CA3 CA4 CA5 CA6 CA7   
## 0.8026 0.6536 0.6198 0.6021 0.5610 0.5342 0.4894   
## CA8   
## 0.4346   
## (Showing 8 of 270 unconstrained eigenvalues)  
##   
## Step 2: the RDA ordination of CWMs of the orthonormalized traits   
## of step 1 with environmental constraints:  
## Call: rda(formula = out1$CWMs\_orthonormal\_traits ~  
## lnFL2000 + invBA\_ha + Condition(Region), data =  
## out1$data$dataEnv)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 0.47361 1.00000   
## Conditional 0.06598 0.13930 5  
## Constrained 0.05861 0.12376 2  
## Unconstrained 0.34902 0.73694 6  
##   
## Inertia is variance  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## RDA1 RDA2   
## 0.05733 0.00128   
##   
## Eigenvalues for unconstrained axes:  
## PC1 PC2 PC3 PC4 PC5 PC6   
## 0.14106 0.05679 0.05154 0.03805 0.03549 0.02608   
##   
## mean, sd, VIF and canonical coefficients with their optimistic [!] t-values:  
## Avg SDS VIF Regr1 tval1  
## RegionPB 0.0627 3.9917 1.3713 0.0140 0.5221  
## RegionPGM 0.3469 7.8355 2.8648 -0.0385 -0.9949  
## RegionSGD 0.1365 5.6523 1.7282 0.0009 0.0304  
## RegionSTM 0.2325 6.9537 2.4016 -0.0754 -2.1278  
## RegionUna 0.0849 4.5878 1.4970 -0.0273 -0.9744  
## lnFL2000 -1.7400 25.7541 1.3627 -0.2137 -8.0033  
## invBA\_ha 0.0478 0.5157 1.2939 -0.1126 -4.3281  
## Avg SDS VIF Regr1 tval1  
## WD 0.6374 0.1729 1.4905 0.5245 9.4324  
## lnSM 5.6438 2.4597 1.4762 0.5579 10.0812  
## LMA 82.9936 24.1529 1.0145 -0.0141 -0.3073  
## Hmax 28.3753 8.7913 1.0652 -0.0088 -0.1868  
## DSNonZoo 0.1693 0.3750 1.0911 0.2883 6.0592  
## DSSynZoo 0.1216 0.3269 1.2521 0.0949 1.8615  
##   
## weighted variance  
## total 28.234  
## traits\_explain 0.474  
## conditionE 0.066  
## constraintsTE 0.059  
## attr(,"meaning")  
## meaning   
## total "total inertia"   
## traits\_explain "trait-constrained inertia"   
## conditionE "trait-constrained inertia explained by the condition in formulaEnv"   
## constraintsTE "trait-constrained inertia explained by the predictors in formulaEnv"

anova(outF, by= "axis", permutations = list(999,cntr))

## $species  
## Species-level permutation test using dc-CA  
## Model: dc\_CA(formulaEnv = ~lnFL2000 + invBA\_ha + Condition(Region), formulaTraits = ~., response = comm, dataEnv = env, dataTraits = traits)   
## Residualized predictor permutation  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.057333 0.186197 275.9672 0.001 \*\*\*  
## dcCA2 1 0.001279 0.004154 6.1569 0.902   
## Residual 1200 0.249303   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $sites  
## Df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.05733 0.140647 43.2020 0.0005 \*\*\*  
## dcCA2 1 0.00128 0.003138 0.9638 0.4590   
## Residual 263 0.34902   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $max  
## Max test combining the community- and species- level tests   
## Model: dc\_CA(formulaEnv = ~lnFL2000 + invBA\_ha + Condition(Region), formulaTraits = ~., response = comm, dataEnv = env, dataTraits = traits)   
##   
## Taken from the species-level test:  
## Residualized predictor permutation  
## Permutation: free  
## Number of permutations: 999  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.057333 0.186197 275.9672 0.001 \*\*\*  
## dcCA2 1 0.001279 0.004154 6.1569 0.902   
## Residual 1200 0.249303   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

plot(outF, widths = c(3,1,1.2), remove\_centroids = TRUE)



# Extra code showing log(invBA\_ha) ----------------------------------------  
  
# avoids an outlier  
  
outF2 <- dc\_CA(formulaEnv = ~lnFL2000 + log(invBA\_ha) + Condition(Region), formulaTraits =~.,  
 response = comm, dataEnv = env, dataTraits = traits)

## Warning in dc\_CA(formulaEnv = ~lnFL2000 + log(invBA\_ha) + Condition(Region), : variableCWD has missing values and is deleted from the environmental data.

## Step 1: the CCA ordination of the transposed matrix with trait constraints,  
## useful in itself and also yielding CWMs of the orthonormalized traits for step 2.  
## Call: cca(formula = tY ~ WD + lnSM + LMA + Hmax +  
## DS, data = dataTraits)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 28.23366 1.00000   
## Constrained 0.47361 0.01677 6  
## Unconstrained 27.76005 0.98323 270  
##   
## Inertia is scaled Chi-square  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## CCA1 CCA2 CCA3 CCA4 CCA5 CCA6   
## 0.20958 0.07642 0.06127 0.04692 0.04579 0.03364   
##   
## Eigenvalues for unconstrained axes:  
## CA1 CA2 CA3 CA4 CA5 CA6 CA7   
## 0.8026 0.6536 0.6198 0.6021 0.5610 0.5342 0.4894   
## CA8   
## 0.4346   
## (Showing 8 of 270 unconstrained eigenvalues)  
##   
## Step 2: the RDA ordination of CWMs of the orthonormalized traits   
## of step 1 with environmental constraints:  
## Call: rda(formula = out1$CWMs\_orthonormal\_traits ~  
## lnFL2000 + log(invBA\_ha) + Condition(Region), data  
## = out1$data$dataEnv)  
##   
## -- Model Summary --  
##   
## Inertia Proportion Rank  
## Total 0.47361 1.00000   
## Conditional 0.06598 0.13930 5  
## Constrained 0.05929 0.12519 2  
## Unconstrained 0.34834 0.73551 6  
##   
## Inertia is variance  
##   
## -- Eigenvalues --  
##   
## Eigenvalues for constrained axes:  
## RDA1 RDA2   
## 0.05755 0.00174   
##   
## Eigenvalues for unconstrained axes:  
## PC1 PC2 PC3 PC4 PC5 PC6   
## 0.14089 0.05670 0.05161 0.03788 0.03548 0.02579   
##   
## mean, sd, VIF and canonical coefficients with their optimistic [!] t-values:  
## Avg SDS VIF Regr1 tval1  
## RegionPB 0.0627 3.9917 1.3830 0.0002 0.0065  
## RegionPGM 0.3469 7.8355 3.1489 -0.0170 -0.4194  
## RegionSGD 0.1365 5.6523 1.7552 -0.0128 -0.4243  
## RegionSTM 0.2325 6.9537 2.5439 -0.0567 -1.5570  
## RegionUna 0.0849 4.5878 1.5408 -0.0437 -1.5422  
## lnFL2000 -1.7400 25.7541 1.4340 -0.2032 -7.4266  
## log(invBA\_ha) -3.1720 8.1671 1.7647 -0.1336 -4.4008  
## Avg SDS VIF Regr1 tval1  
## WD 0.6374 0.1729 1.4905 0.5203 8.7554  
## lnSM 5.6438 2.4597 1.4762 0.5580 9.4358  
## LMA 82.9936 24.1529 1.0145 -0.0291 -0.5943  
## Hmax 28.3753 8.7913 1.0652 0.0012 0.0245  
## DSNonZoo 0.1693 0.3750 1.0911 0.2840 5.5857  
## DSSynZoo 0.1216 0.3269 1.2521 0.1021 1.8755  
##   
## weighted variance  
## total 28.234  
## traits\_explain 0.474  
## conditionE 0.066  
## constraintsTE 0.059  
## attr(,"meaning")  
## meaning   
## total "total inertia"   
## traits\_explain "trait-constrained inertia"   
## conditionE "trait-constrained inertia explained by the condition in formulaEnv"   
## constraintsTE "trait-constrained inertia explained by the predictors in formulaEnv"

anova(outF2, by= "axis", permutations = list(999,cntr))

## $species  
## Species-level permutation test using dc-CA  
## Model: dc\_CA(formulaEnv = ~lnFL2000 + log(invBA\_ha) + Condition(Region), formulaTraits = ~., response = comm, dataEnv = env, dataTraits = traits)   
## Residualized predictor permutation  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.057552 0.158522 227.3577 0.001 \*\*\*  
## dcCA2 1 0.001741 0.004795 6.8765 0.821   
## Residual 1200 0.303759   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $sites  
## Df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.05755 0.14118 43.4516 0.0005 \*\*\*  
## dcCA2 1 0.00174 0.00427 1.3142 0.2890   
## Residual 263 0.34834   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## $max  
## Max test combining the community- and species- level tests   
## Model: dc\_CA(formulaEnv = ~lnFL2000 + log(invBA\_ha) + Condition(Region), formulaTraits = ~., response = comm, dataEnv = env, dataTraits = traits)   
##   
## Taken from the species-level test:  
## Residualized predictor permutation  
## Permutation: free  
## Number of permutations: 999  
##   
## df ChiSquare R2 F Pr(>F)   
## dcCA1 1 0.057552 0.158522 227.3577 0.001 \*\*\*  
## dcCA2 1 0.001741 0.004795 6.8765 0.821   
## Residual 1200 0.303759   
## ---  
## Signif. codes:   
## 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

plot(outF2, widths = c(3,1,1.2), remove\_centroids = TRUE, flip\_axis = TRUE)

