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

Cajo J. F. ter Braak

Appendix S2. R-code for the permutation tests with different data generating models of section 3.

1. Introduction

This appendix gives R-code for the four permutation tests in six different data generating models, used to create Fig. 1 of the paper. Section 2 gives the R code of functions and section 3 gives R-code of the script with example output.

2. R-functions for the permutation tests

The key functions are:

pmax_PermutationTest – implements the four different permutation tests for an arbitrary test statistic using the argument FUN_test_statistics, set by default to the function fourthcorner.

test_permutation_models – generate Monte Carlo *p* values for simulated data sets using the function pmax_PermutationTest and functions from Appendix S1 (in particular, generate_RC_community)

Save the following code as R-file with name: te_score_functions2.r

```
pmax_PermutationTest <- function(obj, FUN_test_statistics=fourthcorner, nrepet = 39
,  ...) {
  # obj from make_obj_for_traitenv(L,E,T, cut_off): matrices L,E and T
  # FUN_test_statistic : function that returns one or more test statistics
  # e.g.FUN_test_statistic = fourthcorner, trait_env__glm_LR or test_statistics_TE;
  # ... options for FUN_test_statistic
  obs <- FUN_test_statistics(obj, ...)
  sim.row <- matrix(NA, nrow = nrepet, ncol = length(obs))
  sim.col <- matrix(NA, nrow = nrepet, ncol = length(obs))
  sim.rc <- matrix(NA, nrow = nrepet, ncol = length(obs))
  for(i in 1:nrepet){
    per.row <- sample(nrow(obj$L))
    per.col <- sample(ncol(obj$L))
    obj.row <- obj.col <-  obj; obj.rc <- obj; obj.ran <- obj
    obj.row$E <-obj.row$E[per.row,,drop =FALSE]
    obj.col$T <-obj.col$T[per.col,,drop =FALSE]
    obj.rc$E <-obj.rc$E[per.row,,drop =FALSE]
    obj.rc$T <-obj.rc$T[per.col,,drop =FALSE]
    sim.row[i, ] <- FUN_test_statistics(obj.row, ...)
    sim.col[i, ] <- FUN_test_statistics(obj.col, ...)
    sim.rc[i, ] <- FUN_test_statistics(obj.rc, ...)
  }
  if (length(obs)==1){
    isna <-  sum(is.na(sim.row))
    pval.row <- (sum(abs(sim.row) >= abs(obs), na.rm = TRUE) + 1)  / (nrepet -isna
+ 1)
    isna <-  sum(is.na(sim.col))
```

```
    pval.col <- (sum(abs(sim.col) >= abs(obs), na.rm = TRUE) + 1)  / (nrepet -isna
+ 1)
    isna <-  sum(is.na(sim.rc))
    pval.rc <- (sum(abs(sim.rc) >= abs(obs), na.rm = TRUE) + 1)  / (nrepet -isna +
1)
  } else {
    obs.mat <- matrix(rep(abs(obs),each=nrepet), nrow= nrepet, ncol=length(obs))
    isna <-  colSums(is.na(sim.row))
    pval.row <- (colSums(abs(sim.row) >= obs.mat, na.rm=TRUE) + 1)/ (nrepet - isna
+ 1)
    isna <-  colSums(is.na(sim.col))
    pval.col <- (colSums(abs(sim.col) >= obs.mat, na.rm=TRUE) + 1)/ (nrepet - isna
+ 1)
    isna <-  colSums(is.na(sim.rc))
    pval.rc <- (colSums(abs(sim.rc) >= obs.mat, na.rm=TRUE) + 1)/ (nrepet - isna +
1)
   }
  result <- cbind(test_stat = obs, p.rc= pval.rc, prow = pval.row, pcol = pval.col,
pmax = pmax(pval.row, pval.col))
  return(result)
}

test_permutation_models <- function(n_species,n_communities, b_te=0, b_ze=0, b_tx =
0, b_zx=0, b_zx_star=0, sigma_epsilon_ij=0, mudat0= 30, distri_Y= "poisson", psort=
FALSE,  n_simul= 1000, nrepet = 199){

  score_vals <- matrix(NA, nrow= n_simul, ncol = 5)

  # one trait one environmental variable ----------------------------------

  for (i in 1:n_simul){
    T <- as.matrix(rnorm(n_species))
    E <- as.matrix(rnorm(n_communities))
    Y <- generate_RC_community(E= E, T= T, coefs=c(b_te, b_ze, b_tx, b_zx, b_zx_sta
r), sigma_epsilon_ij  = sigma_epsilon_ij, mu0=mudat0,
                               coefsE=c(ef.main,ef.main.sq, ef.main.ran.rows),
                               coefsT=c(ef.main,ef.main.sq, ef.main.ran.columns),
                               distribution = distri_Y)
    # give E and T a non-trivial mean and variance and remove any empty rows/column
s in Y
    obj <- make_obj_for_traitenv(Y,3+5*E,10+2*T)
    score_vals[i,] <- pmax_PermutationTest(obj, nrepet = nrepet, FUN_test_statistic
s=fourthcorner)
    score_vals[i, 1] <- pchisq(sum(obj$L)*score_vals[i, 1], df=1, lower.tail = FALS
E)
  }

  if (psort)     score_vals <- apply(score_vals,2,sort)
  score_vals
}
```

3. R-script with example output

```
rm(list=ls(all=TRUE))  # remove all existing items from the workspace
source("generate_trait_env_community_data_loglinear.r")
source("te_score_functions1.r")
source("te_score_functions2.r")

set.seed(13531)
```

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

# one trait one environmental variables Poisson data

n_communities <- 30 #n
n_species <- 30 # m
n_trait <- 1  #q
n_env <- 1 # p
cor_Tdat <- 0.5
cor_Edat <- 0.5

n_simul <-  100 #
nrepet <- 19

# global variables used in test_permutation_models
ef.main <- 0.2
ef.main.sq <-  0
ef.main.ran.rows <- 0
ef.main.ran.columns <- 0

distri_Y. <- "negbin" # distribution used for generating the response data Y given
E(Y)
coef_size <- 0.2


data_generating_models= c("poisson","negbin", "b_ze","b_tx","rc_1","b_te") # simula
tion models for the data

# simulate the sampling distribution of the permutation P value of the score test s
tatistic for different models for the data and different permutation sches --------
---------------------------

name.res <- c( "p.chi", "p.rc",  "prow",  "pcol",  "pmax") # chi-square pvalue and
pvalues of permutation schemes
performance.i <- array(NA,dim  = c(n_simul, length(data_generating_models),length(n
ame.res)) )
dimnames(performance.i) = list(simul= paste("i",1:n_simul),sim.model = data_generat
ing_models,perm.model = name.res)

psort <- TRUE
performance.i[,"poisson",]<- test_permutation_models(n_species,n_communities, b_te=
0, b_ze=0, b_tx = 0, b_zx=0, b_zx_star=0, sigma_epsilon_ij=0, mudat0= 30, distri_Y=
"poisson",  n_simul= n_simul, nrepet = nrepet, psort= psort)
performance.i[,"negbin",]<- test_permutation_models(n_species,n_communities, b_te=0
, b_ze=0, b_tx = 0, b_zx=0, b_zx_star=0, sigma_epsilon_ij=0, mudat0= 30, distri_Y=
"negbin",  n_simul= n_simul, nrepet = nrepet, psort= psort)
performance.i[,"b_ze",]<- test_permutation_models(n_species,n_communities, b_te=0,
b_ze=coef_size, b_tx = 0, b_zx=0, b_zx_star=0, sigma_epsilon_ij=0, mudat0= 30, dist
ri_Y= distri_Y.,  n_simul= n_simul, nrepet = nrepet, psort= psort)
performance.i[,"b_tx",]<- test_permutation_models(n_species,n_communities, b_te=0,
b_ze=0, b_tx = coef_size, b_zx=0, b_zx_star=0, sigma_epsilon_ij=0, mudat0= 30, dist
ri_Y= distri_Y.,  n_simul= n_simul, nrepet = nrepet, psort= psort)
performance.i[,"rc_1",]<- test_permutation_models(n_species,n_communities, b_te=0,
b_ze=0, b_tx = 0, b_zx=0, b_zx_star=coef_size, sigma_epsilon_ij=0, mudat0= 30, dist
ri_Y= distri_Y.,  n_simul= n_simul, nrepet = nrepet, psort= psort)
performance.i[,"b_te",]<- test_permutation_models(n_species,n_communities, b_te=coe
f_size, b_ze=coef_size, b_tx = 0, b_zx=0, b_zx_star=0, sigma_epsilon_ij=0, mudat0=
30, distri_Y= distri_Y.,  n_simul= n_simul, nrepet = nrepet, psort= psort)

nominal.level <- 0.05
signi <- performance.i < nominal.level
TypeIerror_rate <- colMeans(signi, na.rm = TRUE)
TypeIerror_rate
```

sim.mo	p.chi	p.ran	p.rc	prow	pcol	pmax
poisso	0.0487	0.0004	0.0476	0.0497	0.0519	0.0339
negbin	0.7477	0.0435	0.0517	0.0469	0.0473	0.0287
b_ze	0.8237	0.1575	0.1726	0.1723	0.0507	0.0484
b_tx	0.8228	0.1525	0.1702	0.0491	0.1681	0.0471
rc_1	0.7466	0.0472	0.0554	0.0518	0.0524	0.0316
b_te	0.9995	0.9680	0.9724	0.9689	0.9008	0.8992