Appendix 2: Simulation code

Simulation models

We simulate data from six models as follows.

Relevant libraries

```
library(mvtnorm)
library(clusterGeneration)
library(mgcv)
library(statmod)
library(corpcor)


# the ecoCopula package is availible on github
# devtools::install_github("gordy2x/ecoCopula")
library(ecoCopula)
```

1. Binary GCGM

For all the GCGM simulation models, we simulate direct interaction matrices by first simulating a correlation matrix using the `clusterGeneration` package, which we use for simulation, and inverting this to generate a precision matrix and hence "true" direct associations.

```
simulate_gcgmb<-function(N,P,sig,env,mmean,effect){

  #simulate latent Gaussian random variables with covariance matrix sig
  Z=rmvnorm(n=N,mean = rep(0,P),sigma = sig)
  #transform to uniform (copula) scale
  U=pnorm(Z)
  #add environmental heterogeneity and transform to binary scale
  meanmat=cbind(1,env)%*%rbind(mmean,effect)
  Y=qbinom(U,size=1,prob=plogis(meanmat))
  Y
}

# We use the following values for the parameters
# n_spp = 9
# n_sites =50
# env = matrix(rep(c(0,1),N/2), ncol = 1)
# env_het= 0 for uniform env and 5 for heterogeneous env
# effect=rnorm(P,0,env_het)
# mmean=rnorm(P, 0) #random Gaussian

# the covariance matrix is generated using library(clusterGeneration)
# The "onion" method was used as it gave precision matrices
# which were most distinct from the correlation matrix
# sig=cov2cor(genPositiveDefMat(P,covMethod="onion")$Sigma)

# The "true" interactions are the negative values of the precision matrix, i.e.
# my_gr=-solve(sig)
```

2. Binary Markov

For the binary Markov model, we adapt the code in Harris (2016). For both Markov simulation models, the "true" direct associations are simulated using the `make_coefficients()` function.

```
#functions from Harris (2016)
make_coefficients = function(n_spp, p_neg, mean_alpha){

  #simulate direct associations from exponential distribution
  true_beta_magnitudes = rexp(choose(n_spp, 2), rate = 1)
  #assign p_neg of the associations to be negative
  b = true_beta_magnitudes * sample(
    c(-1, 1),
    size = length(true_beta_magnitudes),
    prob = c(p_neg, 1 - p_neg),
    replace = TRUE
  )
  #generate species means from normal distribution
  a = rnorm(n_spp, mean_alpha)
  c(alpha = a, beta = b)
}

# Define a convenience function for Bernoulli random samples
rbern = function(n, prob){
  rbinom(n = n, size = 1, prob = prob)
}

#adapted from Harris (2016)
simulate_markov_b = function(n_spp, n_sites, n_gibbs, f,
                             rdist, p_neg, mean_alpha,alpha_env){
  #generate "true" direct associations and species means
  par = make_coefficients(n_spp, p_neg, mean_alpha)
  #save "true" direct associations
  truth = par[-(1:n_spp)]
  # "True" intercepts, possibly adjusted below by environment
  alpha = par[1:n_spp]
  # Turn the interaction values into an n-by-n matrix
  # Start with empty matrix; fill in upper triangle;
  # then fill in lower triangle with its transpose
  beta = matrix(0, n_spp, n_spp)
  beta[upper.tri(beta)] = truth
  beta = beta + t(beta)
  # The environmental gradient is binary, simulating a impact / control design
  env = matrix(rep(c(0,1),n_sites/2), ncol = 1)
  # Simulate the landscape from known process with Gibbs sampling
  # Landscape starts as if betas were all zero. Each species' occurrence
  # probability or abundance depends on its alpha value and on the
  # environment (assuming alpha_env is not set to zero).
  x = matrix(0,n_sites,n_spp)

  for(j in 1:n_spp){
    x[,j] = rdist(
      nrow(x),
      f(cbind(1,env)%*%(c(alpha[j], alpha_env[j])))
    )
```

```
  }
  # Gibbs sampling
  for(i in 1:n_gibbs){
    # Each round of Gibbs sampling updates one species (column) across all sites
    # according to its conditional probability (i.e. conditional on environment
    # and the other species that are present).
    for(j in 1:n_spp){
      x[,j] = rdist(
        nrow(x),
        f(x[,-j] %*% beta[-j, j] + cbind(1,env)%*%(c(alpha[j], alpha_env[j])))
      )
    }
  }
  out=list(x=x,beta=beta)
  out
}

#We use the following values for the parameters
# n_spp = 9
# n_sites =50
# n_gibbs = 2000
# f = plogis
# rdist = rbern
# p_neg = 0.75
# mean_alpha =-1
# alpha_env= 0 for uniform env and 5 for heterogeneous env
```

3. Count CGCM

```
simulate_gcgm_p<-function(N,P,sig,env,mmean,effect){
  #simulate latent Gaussian random variables with covariance matrix sig
  Z=rmvnorm(n=N,mean = rep(0,P),sigma = sig)
  #transform to uniform (copula) scale
  U=pnorm(Z)
  #add environmental heterogeneity and transform to Poisson scale
  meanmat=cbind(1,env)%*%rbind(mmean,effect)
  Y=qpois(U,lambda=exp(meanmat))
  Y
}

# We use the following values for the parameters
# n_spp = 9
# n_sites =50
# env = matrix(rep(c(0,1),N/2), ncol = 1)
# env_het= 0 for uniform env and 5 for heterogeneous env
# effect=rnorm(P,0,env_het)
# mmean=rnorm(P, 0) #random Gaussian

# the covariance matrix is generated using library(clusterGeneration)
# The "onion" method was used as it gave precisionmatrices
# which were most distinct from the correlation matrix
# sig=cov2cor(genPositiveDefMat(P,covMethod="onion")$Sigma)
```

```
# The "true" interactions are the negative values of the precision matrix, i.e.
# my_gr=-solve(sig)
```

4. Count Markov

This code is adapted from Harris (2016).

```
simulate_markov_p = function(n_spp, n_sites, n_gibbs, f,
                             rdist, p_neg, alpha,alpha_env){
  par = make_coefficients(n_spp, p_neg, 0)
  truth = par[-(1:n_spp)]
  # Turn the interaction values into an n-by-n matrix
  # Start with empty matrix; fill in upper triangle;
  # then fill in lower triangle with its transpose
  beta = matrix(0, n_spp, n_spp)
  beta[upper.tri(beta)] = truth
  beta = beta + t(beta)
  #environment is binary
  env = matrix(rep(c(0,1),n_sites/2), ncol = 1)
  # Simulate the landscape from known process with Gibbs sampling
  # Landscape starts as if betas were all zero. Each species' occurrence
  # probability or abundance depends on its alpha value and on the
  # environment (assuming alpha_env is not set to zero).
  x = matrix(0,n_sites,n_spp)

  for(j in 1:n_spp){
    x[,j] = rdist(
      nrow(x),
      f(cbind(1,env)%*%(c(alpha[j], alpha_env[j])))
    )
  }

  # Gibbs sampling
  for(i in 1:n_gibbs){
    # Each round of Gibbs sampling updates one species (column) across all sites
    # according to its conditional probability (i.e. conditional on environment
    # and the other species that are present).
    for(j in 1:n_spp){
      x[,j] = rdist(
        nrow(x),
        f(x[,-j] %*% beta[-j, j] + cbind(1,env)%*%(c(alpha[j], alpha_env[j])))
      )
    }
  }
  out=list(x=x,beta=beta)
  out
}

#We use the following values for the parameters
# n_spp = 9
# n_sites =50
# n_gibbs = 5000
# f = exp (log link)
```

```
# rdist = rpois
# p_neg = 1
# The alpha (speacies mean) of 3 was chosen as it lead to relatively stable simulations,
# with species being completely absent only rarely
# alpha =rep(3,P)
# env_het= 0 for uniform env and 5 for heterogeneous env
# alpha_env= rnorm(P,0,env_het)
```

5. Ordinal GCGM

```
#cumulative distribution function for cumulative link model
qord<-function(U,meanmat,theta){

  #initialize Y
  Y=U
  ncat=dim(theta)[1]
  #for each species
  for(j in 1:ncol(U)){
    #and each site
    for(i in 1:nrow(U)){
      #initialize cutoffs
      pi=c(rep(0,ncat+1),1)
      for(k in 1:ncat){
        #apply species mean and environment
        pi[k+1]=pnorm(meanmat[i,j]+theta[k,j])
      }
      pi=sort(unique(pi))
      #assign category for each data point
      Y[i,j]=as.numeric(cut(U[i,j],breaks=pi))-1
    }
  }
  Y
}

simulate_ordinal<-function(N,P,sig,env,effect,theta){

  #simulate latent Gaussian random variables with covariance matrix sig
  Z=rmvnorm(n=N,mean = rep(0,P),sigma = sig)
  #transform to uniform (copula) scale
  U=pnorm(Z)
  #add environmental heterogeneity and transform to ordinal scale
  meanmat=cbind(env)%*%rbind(effect)
  qord(U,meanmat,theta)

}

# We use the following values for the parameters
# P = 9
# N = 50
# env = matrix(rep(c(0,1),N/2), ncol = 1) #environmental covariate
# env_het= 0 for uniform env and 5 for heterogeneous env
# effect=rnorm(P,0,env_het)
# mmean=rnorm(P, 0) #random Gaussian
```

```
# the covariance matrix is generated using library(clusterGeneration)
# The "onion" method was used as it gave precisionmatrices
# which were most distinct from the correlation matrix
# sig=cov2cor(genPositiveDefMat(P,covMethod="onion")$Sigma)

# The "true" interactions are the negative values of the precision matrix, i.e.
# my_gr=-solve(sig)
```

6. Tweedie GCGM

```
simulate_Tweedie <-function(N,P,sig,env,mmean,effect){

  #simulate latent Gaussian random variables with covariance matrix sig
  Z=rmvnorm(n=N,mean = rep(0,P),sigma = sig)
  #transform to uniform (copula) scale
  U=pnorm(Z)
  #add environmental heterogeneity and transform to Tweedie  scale
  meanmat=cbind(1,env)%*%rbind(mmean,effect)
  matrix(qTweedie (U,mu=exp(meanmat),phi=2,power=1.5),N,P)

}

# We use the following values for the parameters
# P = 9
# N = 50
# env = matrix(rep(c(0,1),N/2), ncol = 1) #environmental covariate
# env_het= 0 for uniform env and 5 for heterogeneous env
# effect=rnorm(P,0,env_het)
# mmean=rnorm(P, 0) #random Gaussian

# the covariance matrix is generated using library(clusterGeneration)
# The "onion" method was used as it gave precisionmatrices
# which were most distinct from the correlation matrix
# sig=cov2cor(genPositiveDefMat(P,covMethod="onion")$Sigma)

# The "true" interactions are the negative values of the precision matrix, i.e.
# my_gr=-solve(sig)
```

Estimation models

We estimate direct associations with six methods, as follows.

1. gllvm

The `gllvm` package can accommodate all data types we simulated, however we only apply it to the binomial and count data as the function gave an error when using the ordinal family argument, and became stuck in an infinite loop for the Tweedie family argument on our simulated data.

```
library(gllvm)
# the data is in a matrix my_samp
# when the enviroment is heterogeneous we specify the following model
env = matrix(rep(c(0,1),N/2), ncol = 1) #environmental covariate
```

```
het_gllvm=gllvm(my_samp, NULL, formula = ~ env, num.lv = 5, family = data_fam)
# when the enviroment is uniform we specify the following model
unif_gllvm=gllvm(my_samp, NULL, formula = ~ 1 ,num.lv = 5, family = data_fam)
# data_fam = "binomial" for binary
# data_fam = "Poisson" for counts

# the covariance matrix obtained with the getResidualCor function is compared to direct species associa
```

2. Pairs

This is a FORTRAN package, and was run outside of R with the following arguments.

- Null model for randomization: Fixed row and column constraints (independent swap) (s)
- Pairwise co-occurence measure: C-score (c)
- Error benchmark of confidence limits: 0.05 (default)
- Number of iterations for computing standard deviations of the null model: 100 (default)
- Maximum number of species in matrix: 150 (default)

Estimated Z-scores were compared to direct species associations.

3. ggmlog

We use the `glasso` package on log transformed counts.

```
library(glasso)
# the data is in a matrix my_samp
unif_ggmlog=glasso(cor(log(my_samp+1)),rho = 0))
```

4. MRF cov

```
library(MRFcov)
# the data is in a matrix my_samp
# when the enviroment is heterogeneous we specify the following model
env = matrix(rep(c(0,1),N/2), ncol = 1) #environmental covariate
clark_dat=data.frame(my_samp)
het_gllvm=try(MRFcov(cbind(clark_dat,env),n_nodes = P,family = data_fam))
# when the enviroment is uniform we specify the following model
unif_gllvm=MRFcov(clark_dat,n_nodes = P,family = data_fam)
# data_fam = "binomial" for binary
# data_fam = "Poisson" for counts
```

5. rosalia

```
library(rosalia)
# the data is in a matrix my_samp
# for all except binary data, we first convert to presence/absence
my_samp_pa=(my_samp>0)*1
out_rosalia=rosalia(my_samp_pa)
```

6. ecoCopula

`ecoCopula` works by first estimating a model, and then using the fitted model to estimate direct associations. For binary and count data we use the `manyglm` function from the `mvabund` package to estimate the model.

```
library(ecoCopula)
# the data is in a matrix my_samp
# when the enviroment is heterogeneous we specify the following model
env = matrix(rep(c(0,1),N/2), ncol = 1) #environmental covariate
my_mod=manyglm(my_samp~env, family="data_fam")
# when the enviroment is uniform we specify the following model
my_mod=manyglm(my_samp~env, family="data_fam")#
# data_fam = "binomial" for binary
# data_fam = "negativ.binomial" for counts (default in manyglm)
```

For ordinal and Tweedie data we use the `manyany` function with either `clm` or `glm` with family `tweedie` respectively.

```
#ordinal
library(ordinal)
my_mod=manyany("clm",my_samp,my_samp~env,data=data.frame(int=1,env=env))#het env
my_mod=manyany("clm",my_samp,my_samp~1,data=data.frame(env=env))# unif env

#Tweedie
library(tweedie)
my_mod=manyany("glm", my_samp, data=data.frame(int=1,env=env), abund~env,
               family = tweedie (var.power=1.2),var.power=1.2) #het env
my_mod=manyany("glm", my_samp, data=data.frame(int=1,env=env), abund~1,
               family = tweedie (var.power=1.2),var.power=1.2) #unif env
# Notice we use a different `var.power` to simualte data (1.5) and estimate data (1.2)
# for a more realistic comparison
```

The direct species associations are then estimated with the `cgr` function from the `ecoCopula` package.

```
cgr(my_mod,lambda=0)
```

Author: Gordana Popovic, David Warton, Fiona Thomson, Francis Hui and Angela Moles