# Supplementary Material - R code for reproducing the analysis

Filipe S. Dias (CIBIO-InBIO) Michael Betancourt (Symplectomorphic, LLC)
Patricia Rodríguez-González (CEF/ISA) Luís Borda-de-Água (CIBIO-InBIO)

## Contents

S1.Introduction . . . . . 1
S2.Preamble . . . . . 1
S3.Importing and processing the data . . . . . 2
S4.Mantel test . . . . . 2
S5.Generalized Dissimilarity Modeling . . . . . 3
S6.BetaBayes . . . . . 5
S6.1 Data preparation . . . . . 5
S6.1 The model . . . . . 6
S6.2 Prior predicitive checks . . . . . 7
S6.3 Run the model . . . . . 10
S6.4 Model validation . . . . . 11
S6.5 Parameter estimates . . . . . 12

## S1. Introduction

The analysis is based on a dataset collected by Condit et al 2002 that contains data from multiple 1-ha plots from Panama, Peru and Equador, where the authors identified and counted all plants with a stem diameter higher than 10 cm. In this analysis we worked with 43 plots from Panama and selected two covariates, geographic distance and elevation.

The original dataset can be found at: https://www.science.org/doi/10.1126/science.1066854

## S2. Preamble

To reproduce the analysis the reader should install the following R packages:

- readxl
- tidyverse
- otuSummary
- vegan
- ggpubr
- cmdstanr (and CmdStan)
- loo
- scales
- bayesplot
- gdm (>= 1.5.0-9, see: https://github.com/fitzLab-AL/GDM)

Please note that you will need to install

## S3. Importing and processing the data

```
library(readxl)
library(tidyverse)
library(otuSummary)
#Import species data
sp_table <- read_excel("conditwebtable.xls",sheet = "fullmatrix")
sp_table <- sp_table %>% select(BCI4,BCI0,BCI49,BCI45,starts_with("P"),S0,C3,C4)
sp_table <- sp_table %>% select(-P40,-P41,-P39,-P37,-P38)

sp_table <- data.frame(t(sp_table))
colnames(sp_table)<-paste(rep("sp",802),seq(1:802),sep="")
sp_table$site<-rownames(sp_table)

#Import covars
covars <- read_excel("conditwebtable.xls",sheet = "site info") %>%
  select(-...8) %>% rename(site='site no.',y="NS coord",x="EW coord")
covars<- covars %>% select(1:5)
site<-covars$site
site<-gsub("B", "BCI", site)
site[51:59]<-gsub("p", "P0", site[51:59])
site<-gsub("p", "P", site)
covars$site<-site

#Final dataset
data<-merge(sp_table,covars,by="site")
data$comm_id<-group_indices(data,site)
data<-data %>% select(site,comm_id,everything())
```

## S4. Mantel test

The first step is to use the mantel() from function from 'vegan' to calculate Bray-Curtis indices.

```
#Bray-Curtis
interm<-select(data,starts_with("sp"))
rownames(interm)<-data$comm_id
dist_sp<-vegdist(interm,method="bray")
```

Then, we need to create distance matrices for geographical distance and elevation.

```
#Distance
df<-data.frame(x=data$x,y=data$y)
rownames(df)<-data$comm_id
dist_euc<-dist(df)

#Elevation
df<-data.frame(x=data$elev)
rownames(df)<-data$comm_id
dist_elev<-dist(df)
```

Finally, we run the Mantel tests for both covariates:

```
mantel(dist_sp,dist_euc,method="pearson")
```

```
##
## Mantel statistic based on Pearson's product-moment correlation
```

```
##
## Call:
## mantel(xdis = dist_sp, ydis = dist_euc, method = "pearson")
##
## Mantel statistic r: 0.5861
##       Significance: 0.001
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.100 0.122 0.143 0.175
## Permutation: free
## Number of permutations: 999
```

```
mantel(dist_sp,dist_elev,method="pearson")
```

```
##
## Mantel statistic based on Pearson's product-moment correlation
##
## Call:
## mantel(xdis = dist_sp, ydis = dist_elev, method = "pearson")
##
## Mantel statistic r: 0.355
##       Significance: 0.001
##
## Upper quantiles of permutations (null model):
##   90%   95% 97.5%   99%
## 0.126 0.159 0.184 0.215
## Permutation: free
## Number of permutations: 999
```

The results suggest there is a statistically significant correlation between community dissimilarity, distance and differences in elevation.

## S5. Generalized Dissimilarity Modeling

We begin by formating our dataset according to 'gdm' data requirements.

```
library(gdm)
sp_gdm<-data
sp_gdm<- sp_gdm %>% select(-site,-elev) %>% rename(site=comm_id)
cov_gdm<- data %>% select(comm_id,elev)
cov_gdm <- cov_gdm %>% rename(site=comm_id)
data_gdm <- formatsitepair(sp_gdm, 1, siteColumn="site", XColumn="x",
              YColumn="y",predData=cov_gdm)
```

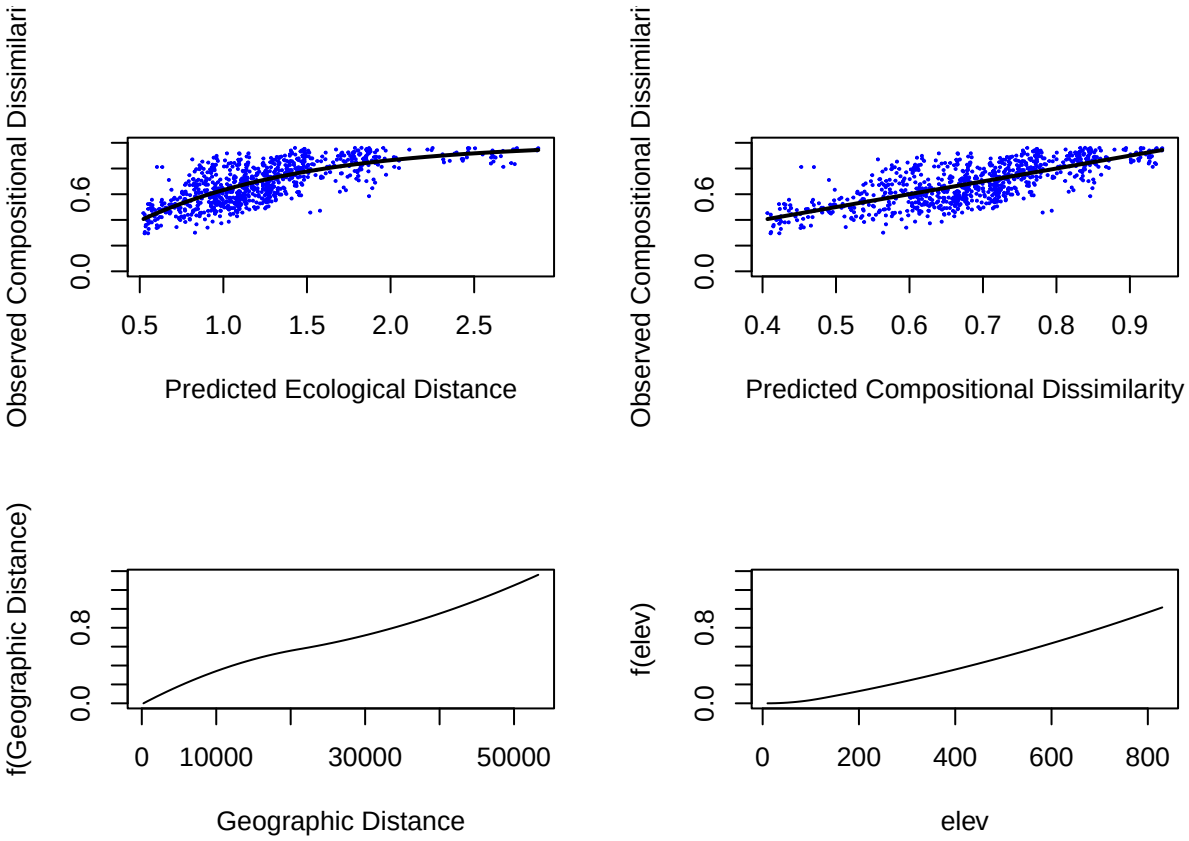
Now, we can run the analysis:

```
model <- gdm(data_gdm, geo=TRUE)
summary(model)
```

```
## [1]
## [1]
## [1] GDM Modelling Summary
## [1] Creation Date:  Fri Oct  7 15:53:23 2022
## [1]
## [1] Name:  model
## [1]
```

```
## [1] Data:  data_gdm
## [1]
## [1] Samples:  903
## [1]
## [1] Geographical distance used in model fitting?  TRUE
## [1]
## [1] NULL Deviance:  110.264
## [1] GDM Deviance:  52.364
## [1] Percent Deviance Explained:  52.51
## [1]
## [1] Intercept:  0.522
## [1]
## [1] PREDICTOR ORDER BY SUM OF I-SPLINE COEFFICIENTS:
## [1]
## [1] Predictor 1: Geographic
## [1] Splines: 3
## [1] Min Knot: 223.607
## [1] 50% Knot: 20965.009
## [1] Max Knot: 53257.125
## [1] Coefficient[1]: 0.432
## [1] Coefficient[2]: 0.36
## [1] Coefficient[3]: 0.579
## [1] Sum of coefficients for Geographic: 1.37
## [1]
## [1] Predictor 2: elev
## [1] Splines: 3
## [1] Min Knot: 10
## [1] 50% Knot: 120
## [1] Max Knot: 830
## [1] Coefficient[1]: 0
## [1] Coefficient[2]: 0.382
## [1] Coefficient[3]: 0.641
## [1] Sum of coefficients for elev: 1.023
```

```
plot(model,plot.layout = c(2, 2))
```




The model explains 52.51% of the deviance and there is a good match between observed and predicted compositional similarity, which indicates the model fits the data well. The spline function for geographical distance attained the highest maximum transformed value indicating it is the most important covariate. The sum of the coefficients was 1.37 which suggests it is an important predictor. The spline for elevation reached a slightly lower value as compared to geographic distance, but the sum of the coefficients was only slightly smaller, 1.023 indicating it also an important predictor.

Due to the non-independence of Bray-Curtis indices we need to perform permutation tests to assess their significance using gdm.varImp().

```
model_varimp<-gdm.varImp(data_gdm, geo=T, nPerm=50, parallel=T, cores=10, predSelect=T)
model_varimp$`Predictor p-values`
```

```
## Geographic       elev
##       0.00       0.02
```

Both p-values are <0.05, suggesting both terms are statistically significant.

## S6. BetaBayes

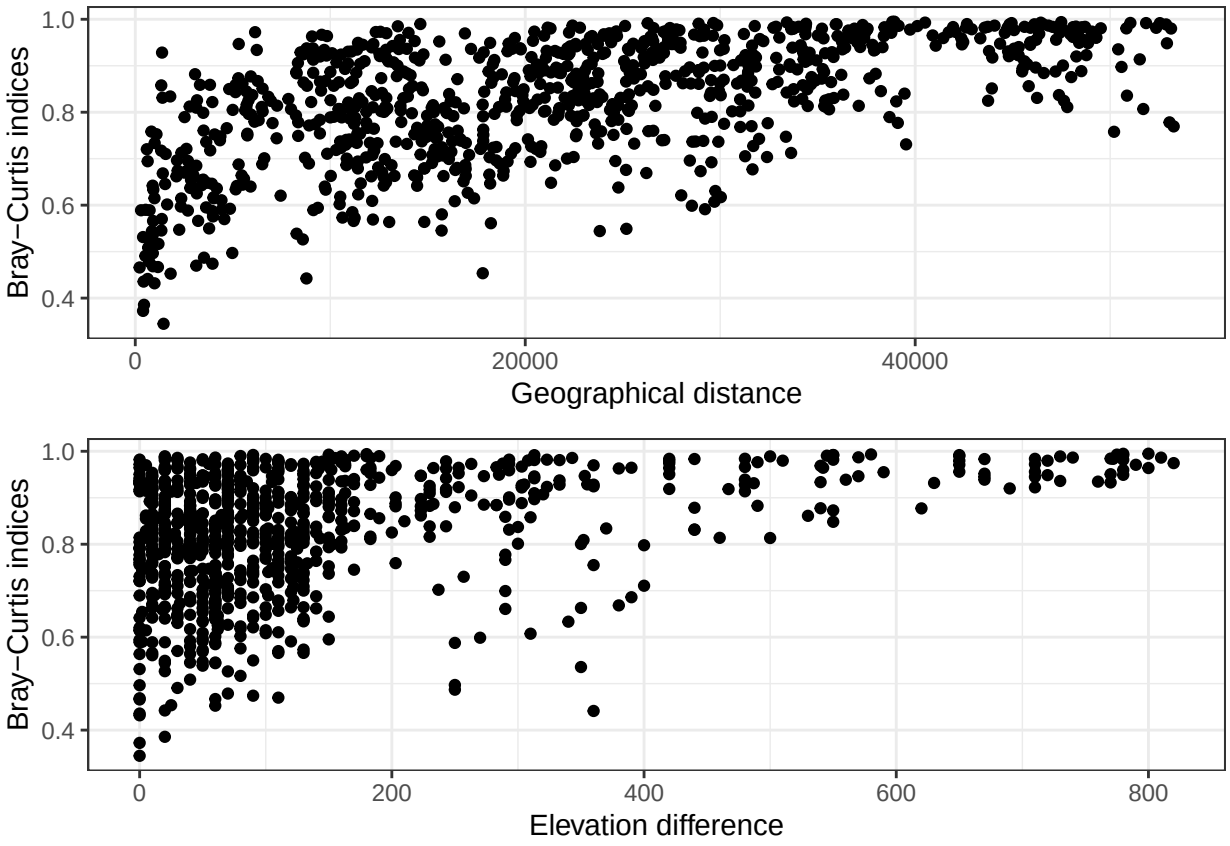
### S6.1 Data preparation

As before, we begin by preparing the data.

```
library(tidyverse)
spec<-matrixConvert(dist_sp, colname = c("comm_id1", "comm_id2", "bray"))
euc_dist<-matrixConvert(dist_euc, colname = c("comm_id1", "comm_id2", "euc_dist"))
elev_dist<-matrixConvert(dist_elev, colname = c("comm_id1", "comm_id2", "elev"))
data_model<-list(spec,euc_dist,elev_dist) %>% reduce(inner_join, by=c("comm_id1","comm_id2"))
```

Before we proceed, we plot Bray-Curtis indices against both covariates.

```
library(ggpubr)
ggarrange(
ggplot(data=data_model,mapping=aes(y=bray,x=euc_dist))+geom_point()+
  labs(y="Bray-Curtis indices",x="Geographical distance")+theme_bw(),
ggplot(data=data_model,mapping=aes(y=bray,x=elev))+geom_point()+
labs(y="Bray-Curtis indices",x="Elevation difference")+theme_bw(),
ncol=1)
```




### S6.1 The model

We model Bray-Curtis indices with a Beta distribution, which is a continuous probability distribution defined on the interval 0 and 1. We chose a Beta distribution parameterized by the mean (or location) "mu" and sample size "kappa". To make sure the parameter "mu" is bounded between 0 and 1 we modeled the logit of "mu" in a linear model of the covariates.

We save the corresponding Stan code in an object called "model_beta".

```
library(cmdstanr)
model_beta <- write_stan_file("
data{
int N;
int N_sample;
vector[N] s;
vector[N] x1;
vector[N] x2;
array[N] int<lower=1> idx1;
```

```
array[N] int<lower=1> idx2;
}
parameters{
real alpha;
real beta1;
real beta2;
vector[N_sample] alpha_s_nc;
real<lower=0> sigma_alpha_s;
real<lower=0> kappa;
}
transformed parameters{
vector[N_sample] alpha_s=sigma_alpha_s*alpha_s_nc;
vector[N] mu = alpha+alpha_s[idx1]+alpha_s[idx2]+beta1*x1+beta2*x2;
}
model{
// Likelihood:
s~beta_proportion(inv_logit(mu),kappa);
// Priors:
alpha~normal(0,0.5);
alpha_s_nc~normal(0,0.5);
sigma_alpha_s~exponential(2);
beta1~normal(0,0.5);
beta2~normal(0,0.5);
kappa ~ normal(0, 50);
}
generated quantities{
  array[N] real pred;
  array[N] real res;
  array[N] real log_lik;

  for(i in 1:N){
      pred[i]= beta_proportion_rng(inv_logit(mu[i]),kappa);
      log_lik[i]= beta_proportion_lpdf(s[i]|inv_logit(mu[i]),kappa);
  }
   for(i in 1:N){
      res[i]= pred[i]-s[i];
  }
}
",
dir="Models",basename="model_beta.stan")
```

### S6.2 Prior predicitive checks

To conduct prior predictive checks wee need to obtain samples from the prior model. We can do this using R functions, but we chose to use Stan instead. We can usese the above Stan code and comment out the section: "s~beta_proportion(inv_logit(mu),kappa)".

```
library(cmdstanr)
model_prior <- write_stan_file("
data{
int N;
int N_sample;
vector[N] s;
vector[N] x1;
```

```
vector[N] x2;
array[N] int<lower=1> idx1;
array[N] int<lower=1> idx2;
}
parameters{
real alpha;
real beta1;
real beta2;
vector[N_sample] alpha_s_nc;
real<lower=0> sigma_alpha_s;
real<lower=0> kappa;
}
transformed parameters{
vector[N_sample] alpha_s=sigma_alpha_s*alpha_s_nc;
vector[N] mu = alpha+alpha_s[idx1]+alpha_s[idx2]+beta1*x1+beta2*x2;
}
model{
// Likelihood:
//s~beta_proportion(inv_logit(mu),kappa);
// Priors:
alpha~normal(0,0.5);
alpha_s_nc~normal(0,0.5);
sigma_alpha_s~exponential(2);
beta1~normal(0,0.5);
beta2~normal(0,0.5);
kappa ~ normal(0, 50);
}
generated quantities{
  array[N] real pred;
  for(i in 1:N){
      pred[i]= beta_proportion_rng(inv_logit(mu[i]),kappa);
  }
}
",
dir="Models",basename="model_prior.stan")
```

To run Stan we need to create a list with the data. Noticed we transform both covariates.

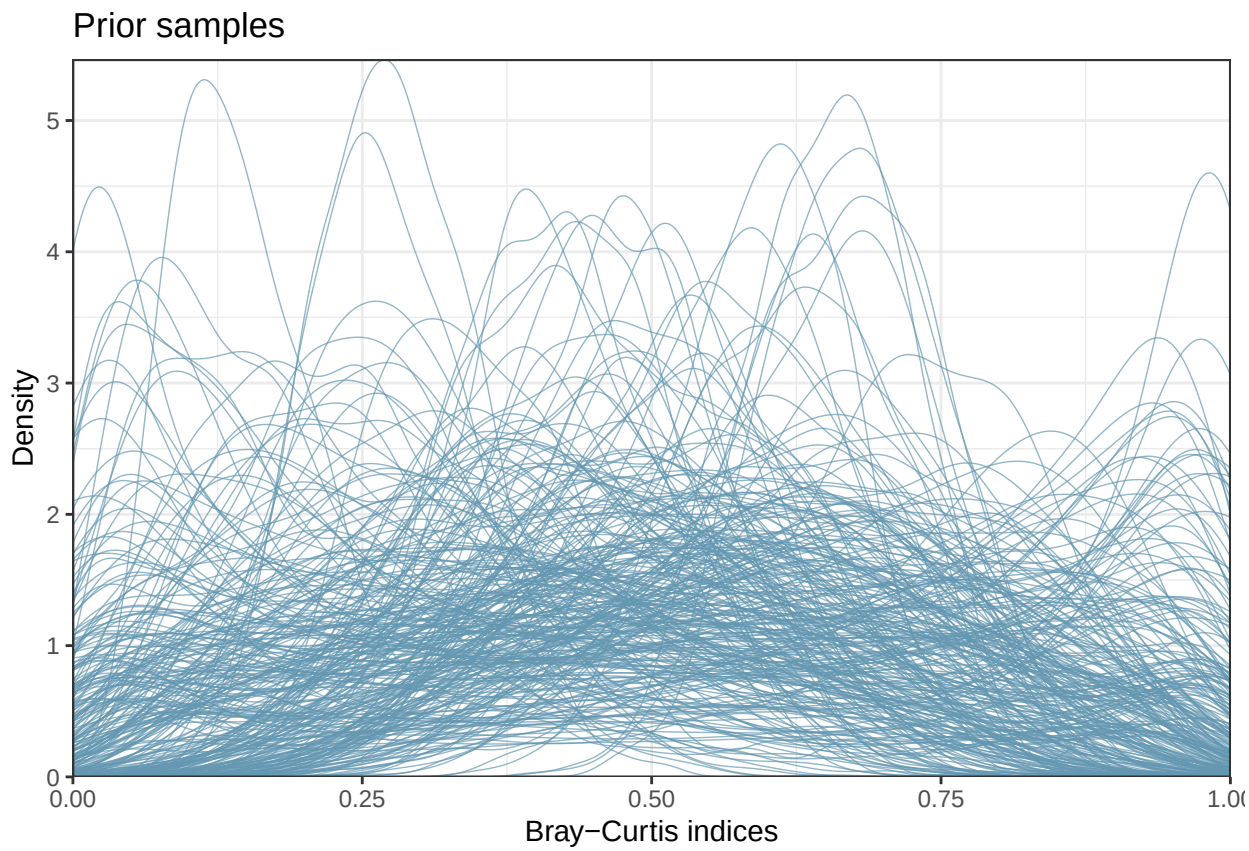
```
x1=(data_model$euc_dist-10000)/10000
x2=(data_model$elev-100)/100
dat<-list(N=nrow(data_model),
          N_sample=max(c(data_model$comm_id1,data_model$comm_id2)),
          s=data_model$bray,
          x1=x1,
          x2=x2,
          idx1=data_model$comm_id1,
          idx2=data_model$comm_id2)
```

```
prior<-cmdstan_model("Models/model_prior.stan")
fit_prior<-prior$sample(data=dat,chains=4,parallel_chains=4,
                        refresh=1000)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
```

```
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 1.2 seconds.
## Chain 3 finished in 1.2 seconds.
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 1.3 seconds.
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 4 finished in 1.4 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 1.3 seconds.
## Total execution time: 1.5 seconds.
```

```
library(bayesplot)
pred<-fit_prior$draws(c("pred"),format="matrix")
ppd_dens_overlay(pred[1:300,])+theme_bw()+legend_none()+labs(y="Density",x="Bray-Curtis indices")+
  labs(title="Prior samples")
```




### S6.3 Run the model

```
library(cmdstanr)
mod<-cmdstan_model("model_beta.stan")
fit<-mod$sample(data=dat,chains=4,parallel_chains=4,
                refresh=1000)
```

```
## Running MCMC with 4 parallel chains...
##
## Chain 1 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 2 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 3 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 4 Iteration:    1 / 2000 [  0%]  (Warmup)
## Chain 1 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 1 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 2 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 2 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 4 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 4 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 3 Iteration: 1000 / 2000 [ 50%]  (Warmup)
## Chain 3 Iteration: 1001 / 2000 [ 50%]  (Sampling)
## Chain 1 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 1 finished in 4.9 seconds.
## Chain 2 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 3 finished in 5.6 seconds.
```

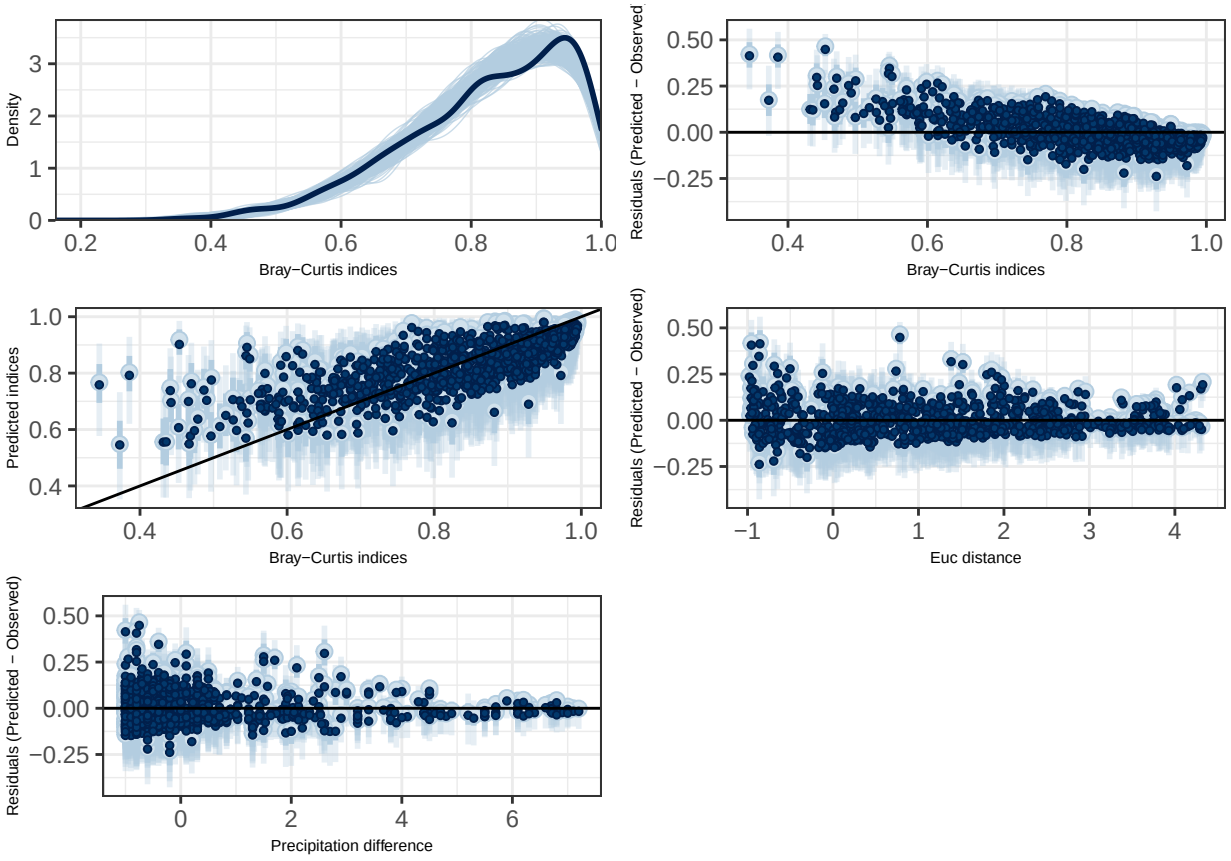
```
## Chain 4 Iteration: 2000 / 2000 [100%]  (Sampling)
## Chain 2 finished in 5.6 seconds.
## Chain 4 finished in 5.6 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 5.4 seconds.
## Total execution time: 5.7 seconds.
```

### S6.4 Model validation

To perform model validation we need to extract predicted values from the model, as well as the residuals.

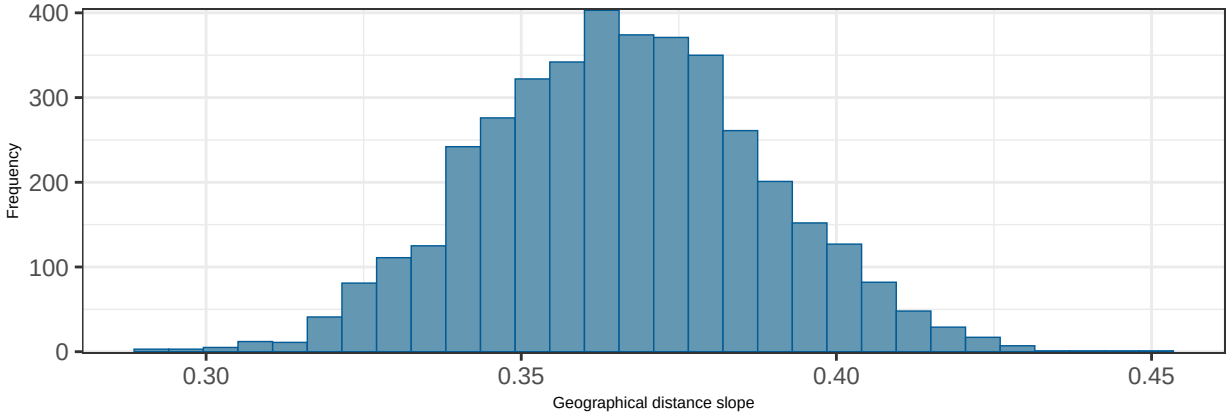
```
s<-dat$s
x1<-dat$x1
x2<-dat$x2
pred<-fit$draws(c("pred"),format="matrix")
mean_pred<-apply(pred,2,mean)
res<-fit$draws(c("res"),format="matrix")
res_mean<-apply(res,2,mean)
```

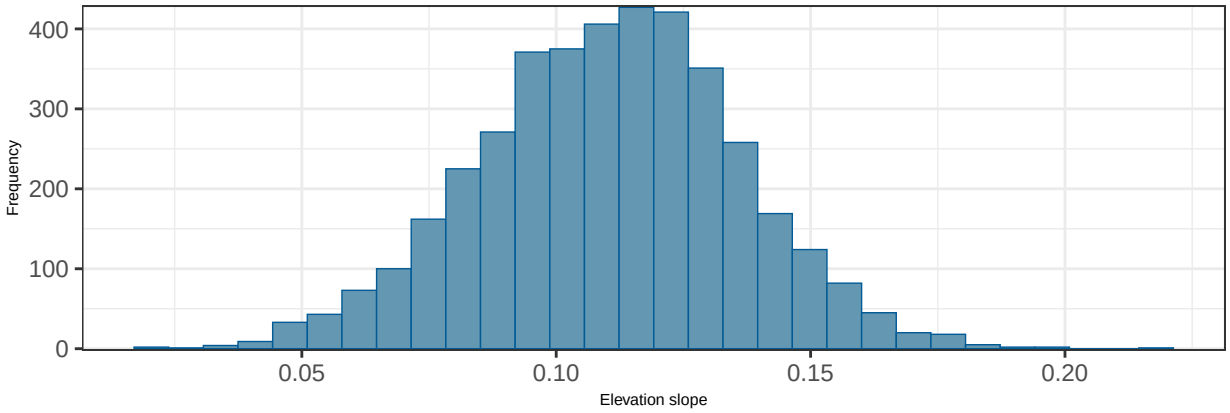
```
library(ggpubr)
library(bayesplot)
ggarrange(
  ppc_dens_overlay(dat$s,pred[1:300,])+theme_bw()+legend_none()+
    labs(y="Density",x="Bray-Curtis indices")+
    theme(axis.title = element_text(size = 6)),
  ppc_intervals(res_mean,res,s)+labs(y="Residuals (Predicted - Observed)",
                                     x="Bray-Curtis indices")+
    theme_bw()+
    theme(axis.title = element_text(size = 6)) +
    hline_0()+legend_none(),
  ppc_intervals(mean_pred,pred,s)+labs(y="Predicted indices",
                                       x="Bray-Curtis indices")+
    theme_bw()+
    theme(axis.title = element_text(size = 6))+
    abline_01()+legend_none(),
  ppc_intervals(res_mean,res,x1)+labs(y="Residuals (Predicted - Observed)",
                                     x="Euc distance")+
    theme_bw()+
    theme(axis.title = element_text(size = 6))+
    hline_0()+legend_none(),
   ppc_intervals(res_mean,res,x2)+labs(y="Residuals (Predicted - Observed)",
                                       x="Precipitation difference")+
    theme_bw()+
    theme(axis.title = element_text(size = 6))+
    hline_0()+legend_none(),

    ncol=2,nrow=3)
```




### 6.5 Parameter estimates

```
ggarrange(
  mcmc_hist(fit$draws(c("beta1")))+theme_bw()+
    theme(axis.title = element_text(size = 6))+
    labs(x="Geographical distance slope",y="Frequency"),
mcmc_hist(fit$draws(c("beta2")))+theme_bw()+
    theme(axis.title = element_text(size = 6))+
    labs(x="Elevation slope",y="Frequency"),
ncol=1,nrow=2)
```




```
fit$summary(c("alpha","beta1","beta2","kappa"))
```

```
## # A tibble: 4 x 10
##   variable   mean median     sd    mad      q5    q95  rhat ess_bulk ess_tail
##   <chr>     <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl> <dbl>    <dbl>    <dbl>
## 1 alpha     1.18   1.18  0.0887 0.0879  1.03    1.32   1.01     523.     947.
## 2 beta1     0.366  0.366 0.0222 0.0223  0.329   0.403  1.00    2192.    2687.
## 3 beta2     0.110  0.111 0.0254 0.0251  0.0675  0.152  1.00    1767.    2405.
## 4 kappa    18.1   18.1   0.892  0.906  16.7    19.6    1.00    3360.    2924.
```

```
#95% credibility interval
fit$summary(c("alpha","beta1","beta2","kappa"),~quantile(., probs = c(0.025, 0.975)))
```

```
## # A tibble: 4 x 3
##   variable  `2.5%` `97.5%`
##   <chr>      <dbl>   <dbl>
## 1 alpha     0.997    1.34
## 2 beta1     0.324    0.410
## 3 beta2     0.0592   0.159
## 4 kappa    16.4     19.9
```