Appendix 3: Estimating species interactions

Inferring species interactions from co-occurrence data with Markov networks

David J. Harris

This document describes how the different models were fit to the simulated data from Appendix 2 and how each model's performance was evaluated.[^1]

Note that the `pairs` program was run separately (outside of R) with the following options:

- Batch mode
- Sequential swap ("s")
- Printing all pairs ("y")
- C-score co-occurrence measure ("c")
- Default confidence limits (0.05)
- Default iterations (100)
- Maximum of 20 species

Initialization:

```
mc.cores = 8

library(dplyr)         # For manipulating data structures
library(corpcor)       # For regularized partial covariances
library(rosalia)       # For Markov networks
library(arm)           # For regularized logistic regression
library(BayesComm)     # For joint species distribution modeling
library(parallel)      # for mclapply
set.seed(1)
```

A function to import the data file and run each method on it:

```
fit_all = function(identifier){
  ######## Import ########

  data_filename = paste0("fakedata/matrices/", identifier, ".csv")
  truth_filename = paste0("fakedata/truths/", identifier, ".txt")

  # first column is row numbers; drop it
  raw_obs = as.matrix(read.csv(data_filename)[ , -1])

  # Identify species that are never present (or never absent) so they
  # can be dropped
  species_is_variable = diag(var(raw_obs)) > 0
  pair_is_variable = tcrossprod(species_is_variable) > 0
```

```
x = raw_obs[ , species_is_variable]
truth = unlist(read.table(truth_filename))[pair_is_variable[upper.tri(pair_is_variable)]]

splitname = strsplit(identifier, "/|-|\\.")[[1]]
n_sites = as.integer(splitname[[1]])
rep_name = splitname[[2]]

# Species IDs
sp1 = combn(colnames(x), 2)[1, ]
sp2 = combn(colnames(x), 2)[2, ]

######## Partial correlations ########
p_corr = pcor.shrink(x)

######## Correlations ########
corr = cor(x)
######## GLM ########
coef_matrix = matrix(0, ncol(x), ncol(x))
for(i in 1:ncol(x)){
  if(var(x[,i]) > 0){
    coefs = coef(bayesglm(x[,i] ~ x[ , -i], family = binomial))[-1]
    coef_matrix[i, -i] = coefs
  }
}
coef_matrix = (coef_matrix + t(coef_matrix)) / 2


######## Markov network ########
rosie = rosalia(x, maxit = 200, trace = 0,
                prior = make_logistic_prior(scale = 2), hessian = TRUE)

rosie_point = rosie$beta[upper.tri(rosie$beta)]
rosie_se = sqrt(diag(solve(rosie$opt$hessian)))[-(1:sum(species_is_variable))]



######## BayesComm and partial BayesComm ########
bc = BC(Y = x, model = "community", its = 1000)

bc_pcors = sapply(
  1:nrow(bc$trace$R),
  function(i){
    Sigma = matrix(0, nrow = ncol(x), ncol = ncol(x))
    Sigma[upper.tri(Sigma)] <- bc$trace$R[i, ]  # Fill in upper triangle
    Sigma <- Sigma + t(Sigma)                   # Fill in lower triangle
    diag(Sigma) <- 1  # Diagonal equals 1 in multivariate probit model
    pcor = cor2pcor(Sigma)
```

```
    pcor[upper.tri(pcor)]
  }
)




bind_rows(
  data_frame(
    rep_name = rep_name,
    n_sites = n_sites,
    sp1 = sp1,
    sp2 = sp2,
    method = "Markov network",
    truth = truth,
    estimate = rosie_point,
    lower = qnorm(.025, rosie_point, rosie_se),
    upper = qnorm(.975, rosie_point, rosie_se)
  ),
  data_frame(
    rep_name = rep_name,
    n_sites = n_sites,
    sp1 = sp1,
    sp2 = sp2,
    method = "GLM",
    truth = truth,
    estimate = coef_matrix[upper.tri(coef_matrix)],
    lower = NA,
    upper = NA
  ),
  data_frame(
    rep_name = rep_name,
    n_sites = n_sites,
    sp1 = sp1,
    sp2 = sp2,
    method = "correlation",
    truth = truth,
    estimate = corr[upper.tri(corr)],
    lower = NA,
    upper = NA
  ),
  data_frame(
    rep_name = rep_name,
    n_sites = n_sites,
    sp1 = sp1,
    sp2 = sp2,
```

```
      method = "partial correlation",
      truth = truth,
      estimate = p_corr[upper.tri(p_corr)],
      lower = NA,
      upper = NA
    ),
    data_frame(
      rep_name = rep_name,
      n_sites = n_sites,
      sp1 = sp1,
      sp2 = sp2,
      method = "partial BayesComm",
      truth = truth,
      estimate = rowMeans(bc_pcors),
      lower = apply(bc_pcors, 1, quantile, .025),
      upper = apply(bc_pcors, 1, quantile, .975)
    ),
    data_frame(
      rep_name = rep_name,
      n_sites = n_sites,
      sp1 = sp1,
      sp2 = sp2,
      method = "BayesComm",
      truth = truth,
      estimate = colMeans(bc$trace$R),
      lower = apply(bc$trace$R, 2, quantile, .025),
      upper = apply(bc$trace$R, 2, quantile, .975)
    )
  )
}
```

Run the above function on all the files:

```
# Find all the csv files in the fakedata/matrices folder,
# then drop .csv
identifiers = dir("fakedata/matrices", pattern = "\\.csv$") %>%
  gsub("\\.csv$", "", .)
```

```
mclapply(identifiers, fit_all, mc.cores = mc.cores,
         mc.preschedule = FALSE, mc.silent = TRUE) %>%
  bind_rows() %>%
  write.csv(file = "estimates.csv")
```

Logistic prior used for the Markov network

From the R help file for the logistic distribution:

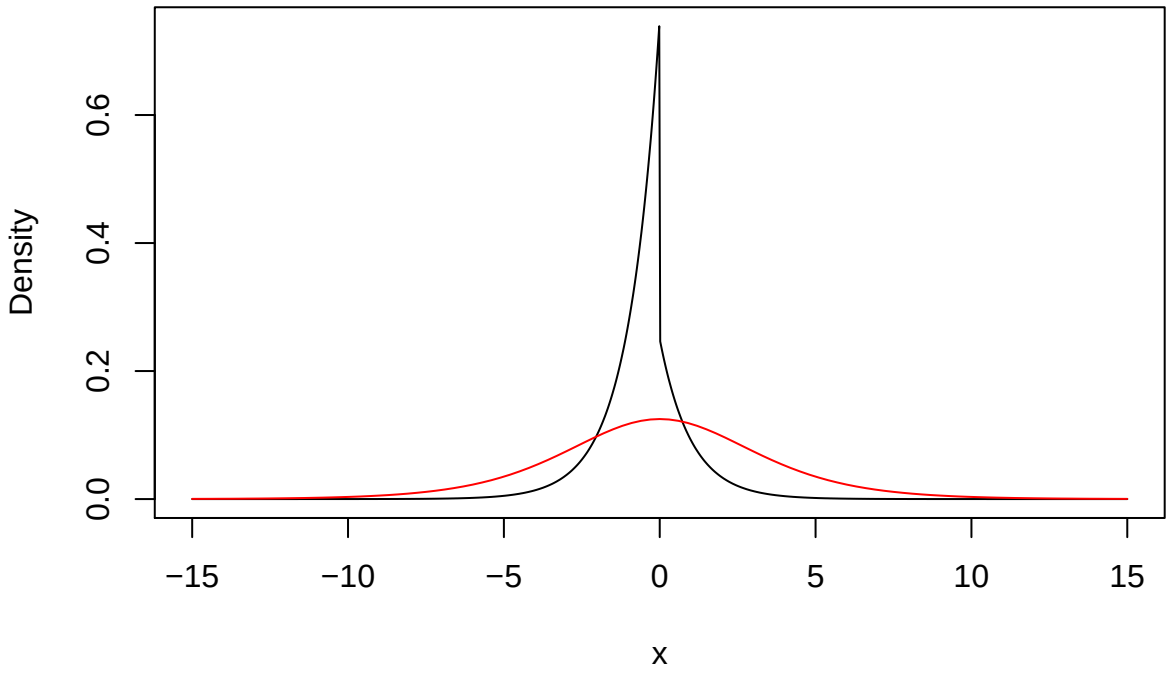
The Logistic distribution with location = m and scale = s has... density

f(x) = 1/s exp((x-m)/s) (1 + exp((x-m)/s))^-2.

It is a long-tailed distribution with mean m and variance π^2 /3 s^2.

As noted in the main text, I used location zero and scale 2 (plotted in red below). The "true" parameter distribution for β is drawn in black, for reference. These distributions have different means and the prior is substantially wider than the distribution of true parameters (mean absolute deviation from zero about 2.8 times larger).

```
curve(ifelse(x < 0, .75 * dexp(abs(x)), .25 * dexp(x)), from = -15, to = 15,
      n = 1000, ylab = "Density")
curve(dlogis(x, location = 0, scale = 2), add = TRUE, col = "red")
```

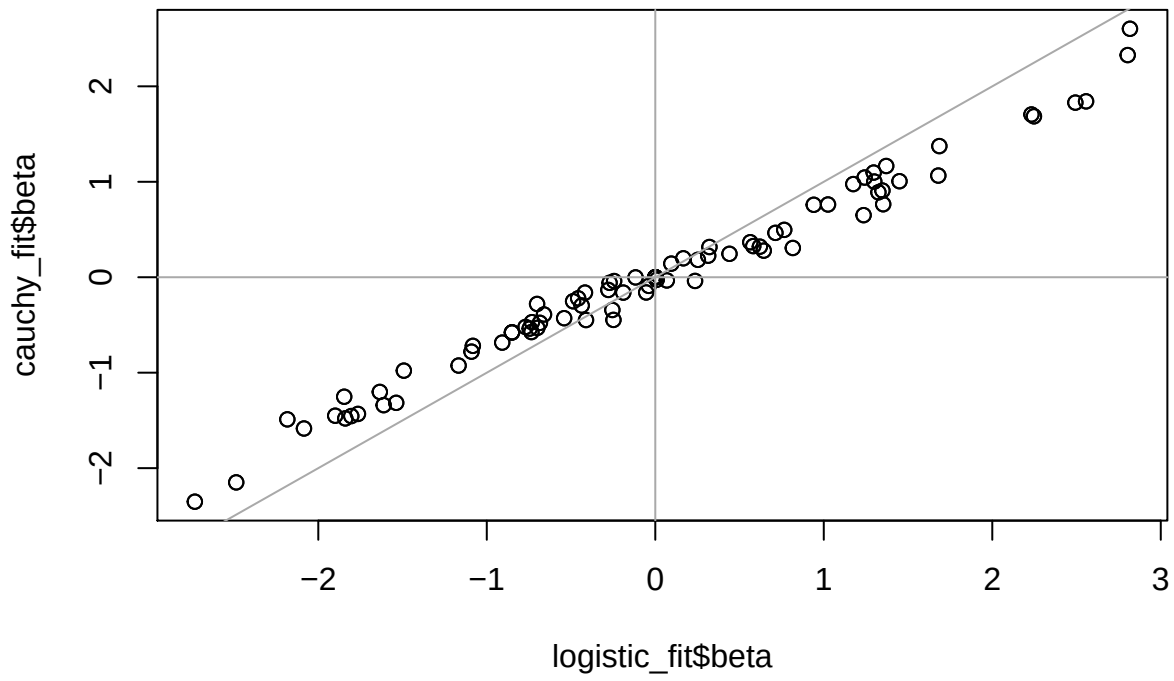


The log of this density was added to the log-likelihood to calculate an un-normalized log-posterior, which is optimized by the `optim` function in the `stats` package.

This prior distribution is "weakly informative" in the sense of Gelman et al. (2008. Annals of Applied Statistics, "A weakly informative default prior distribution for logistic and other regression models"), as shown below: the logistic regularizer pulls the estimated values toward zero less strongly than Gelman et al.'s weakly-informative Cauchy prior does.

```
x = as.matrix(
  read.csv("fakedata/matrices/25-no_env1.csv")[,-1]
)

# Don't estimate parameters for species that were never observed
```

```
x = x[ , colMeans(x) != 0]


# Estimate parameters using a weakly informative logistic prior
logistic_fit = rosalia(x, prior = make_logistic_prior(scale = 2),
                              trace = FALSE)

# Estimate parameters using Gelman et al.'s weakly informative Cauchy prior
cauchy_fit = rosalia(x, prior = make_cauchy_prior(scale = 2.5),
                            trace = FALSE)


plot(logistic_fit$beta, cauchy_fit$beta)
abline(0,1, h = 0, v = 0, col = "darkgray")
```



GLM coefficient correlations

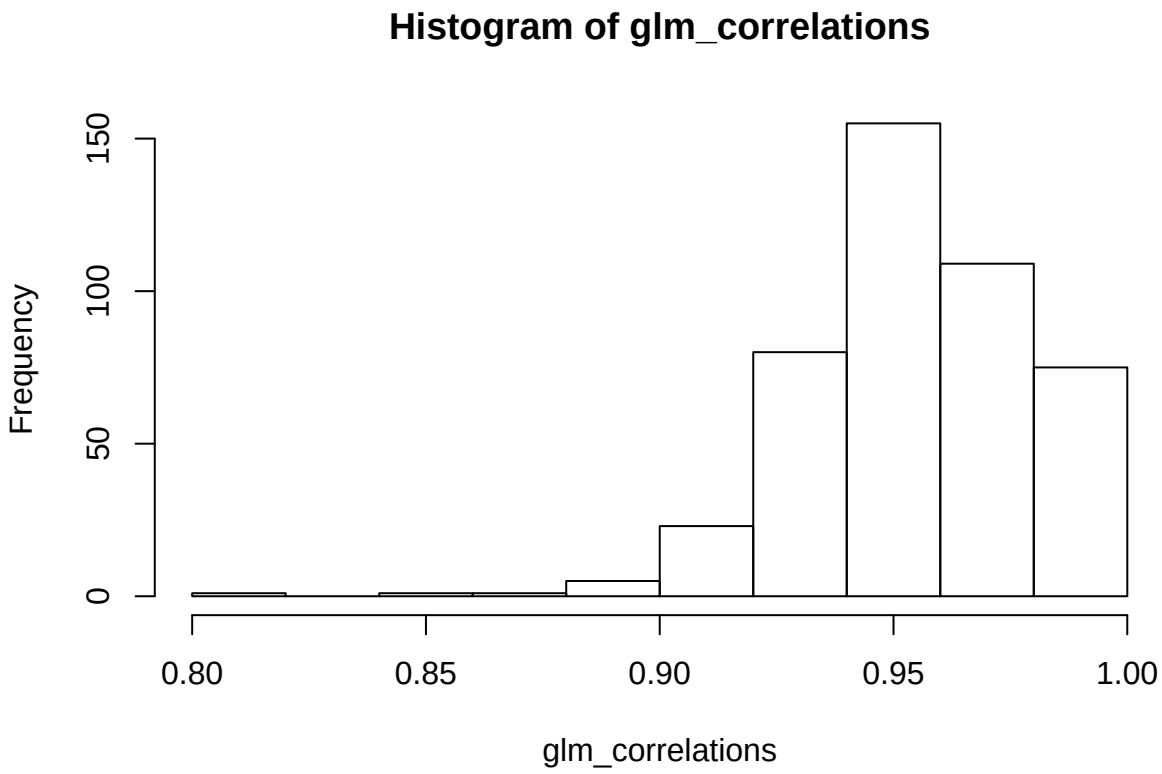
The GLM method produces two estimates for each species pair. These estimates tend to be very similar, so most reasonable methods for generating a consensus estimate should produce very similar results.

```
get_glm_cor = function(identifier){
  data_filename = paste0("fakedata/matrices/", identifier, ".csv")

  # first column is row numbers; drop it
  raw_obs = as.matrix(read.csv(data_filename)[ , -1])
```

```
  # Identify species that are never present (or never absent) so they
  # can be dropped
  species_is_variable = diag(var(raw_obs)) > 0
  pair_is_variable = tcrossprod(species_is_variable) > 0


  x = raw_obs[ , species_is_variable]

  coef_matrix = matrix(0, ncol(x), ncol(x))

  # Fill in the coefficient matrix, column-by-column
  for(i in 1:ncol(x)){
    if(var(x[,i]) > 0){
      coefs = coef(bayesglm(x[,i] ~ x[ , -i], family = binomial))[-1]
      coef_matrix[i, -i] = coefs
    }
  }

  # Compare the estimates from the upper triangle with the estimates from
  # the estimates in the lower triangle (which becomes the upper triangle after
  # transposing)
  cor(coef_matrix[upper.tri(coef_matrix)], t(coef_matrix)[upper.tri(coef_matrix)])
}

glm_correlations = unlist(mclapply(identifiers, get_glm_cor))

mean(glm_correlations)
```

```
## [1] 0.9546387
```

```
hist(glm_correlations)
```


Histogram of glm_correlations
Frequency
0
50
100
150
0.80
0.85
0.90
0.95
1.00
glm_correlations