Bayesian inference of an individual-based mutualistic network
09_01

Net 09_01

```r
library(BayesianNetworks)
library(network.tools)
library(tidyverse)
theme_set(theme_minimal())
options(mc.cores = 4)
```

Data

Load dataset and sampling effort per individual plant:

```r
web <- readr::read_csv(here::here("data/nets_raw", paste0(params$net, "_int.csv"))) |>
  arrange(ind)
```

```
## Rows: 18 Columns: 23
## -- Column specification -----------------------------------------------------------------
## Delimiter: ","
## chr  (1): ind
## dbl (22): Piaya_cayana, Amazilia_sp, Euphonia_chlorotica, Euphonia_sp, Euphonia_violacea, Setophaga_
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
mat <- as.matrix(web[, -1])
mat <- apply(mat, c(1,2), as.integer)
rownames(mat) <- web$ind


# create numeric vector of sampling effort for each plant with names = plant id
effort <- readr::read_csv(here::here("data/nets_attr", paste0(params$net, "_attr.csv"))) |>
  select(ind, starts_with("se_")) |>
  filter(ind %in% web$ind) |>
  arrange(ind)
```

```
## Rows: 19 Columns: 11
## -- Column specification -----------------------------------------------------------------
## Delimiter: ","
## chr (4): ind, fruit_type, fruit_color, seeds_per_fruit
## dbl (7): fruit_d1_mm, fruit_d1_mm_sd, sucrose_mg_g, fructose_mg_g, glucose_mg_g, height_m, se_obs_h
```

```
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.

## If there is only one column with sampling effort, use it:
if (!net %in% c("15_01", "18_01", "18_02", "20_01", "21_01", "21_02")) {
  effort <- effort |>
    pull(starts_with("se_"), name = "ind")
}

# Otherwise, select sampling effort column in some specific nets:
if (net == "15_01") {
  effort <- effort |>
    pull(se_cam_days, name = "ind")
}

if (net %in% c("18_01", "18_02", "20_01")) {
  effort <- effort |>
    mutate(se_bc_months = se_bc_days/30) |>
    pull(se_bc_months, name = "ind")
}

if (net %in% c("21_01", "21_02")) {
  effort <- effort |>
    pull(se_obs_h, name = "ind")
}

## Some nets may require adjusting of the count data or effort values
## Insert that here eg.
# if (params$net == "01_01") {
#   mat <- round(mat/10)
#   mat <- apply(mat, c(1,2), as.integer)
# }

stopifnot(identical(length(effort), nrow(mat)))
stopifnot(identical(names(effort), rownames(mat)))

summary(as.numeric(mat))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.0000  0.0000  0.0000  0.6818  0.0000 29.0000
```

```
if (max(mat) > 500) {
  stop("More than 500 counts in some cell(s)")
}

summary(effort)
```
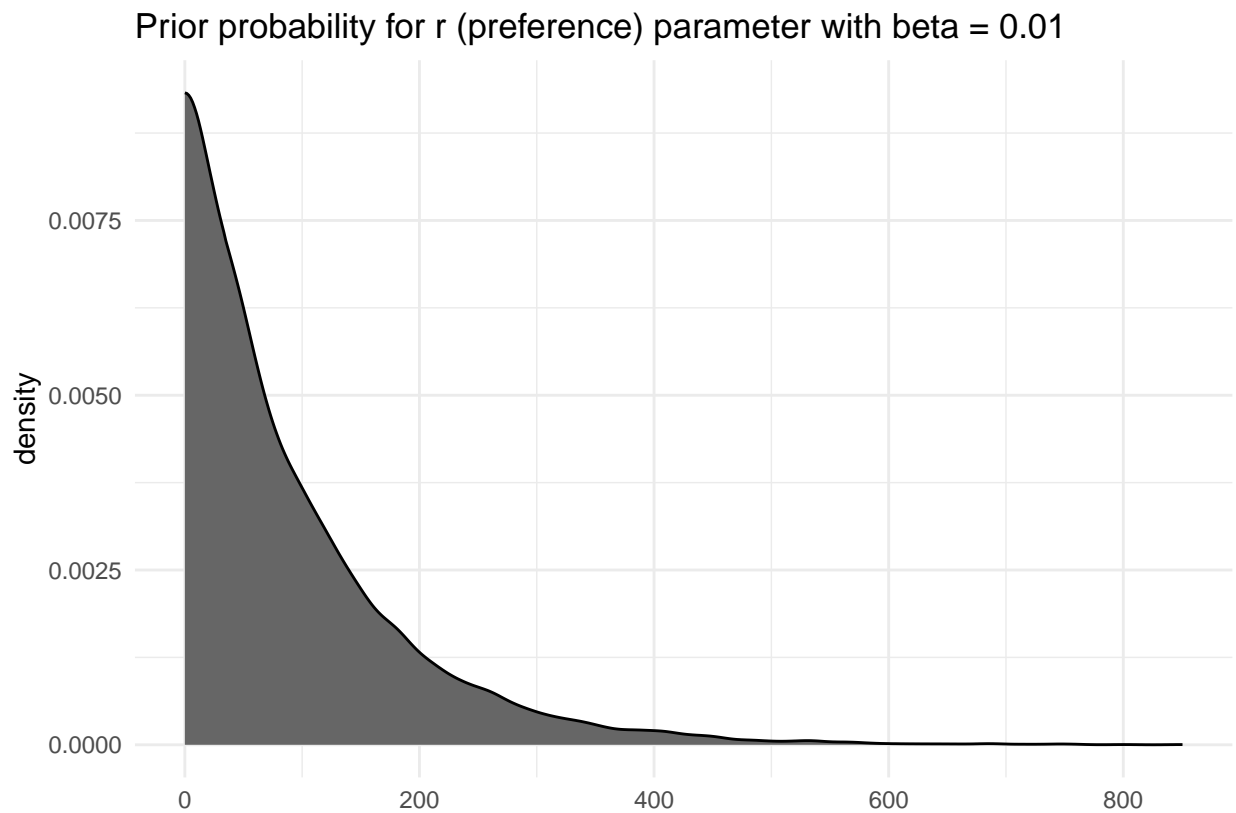
```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      20      20      20      20      20      20
```

2

```r
if (max(effort) > 500) {
  stop("Sampling effort > 500 for some plants")
}
```

Bayesian inference of network structure

```r
dt <- prepare_data(mat, sampl.eff = effort)

plot_prior(params$beta)
```

Prior probability for r (preference) parameter with beta = 0.01


```r
fit <- fit_model(dt,
                 refresh = 0,
                 beta = params$beta,
                 model = params$model,
                 # max_treedepth = 15,
                 # init = function() list(r = runif(1, 0, 20000)),
                 iter_warmup = params$iter,
                 iter_sampling = params$iter,
                 thin = 4 * params$iter / 1000)
```

```
## Running MCMC with 4 parallel chains...
##
```

```
## Chain 1 finished in 30.4 seconds.
## Chain 3 finished in 30.3 seconds.


## Warning: Chain 4 finished unexpectedly!


## Chain 4 finished in 30.4 seconds.
## Chain 2 finished in 30.8 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 30.5 seconds.
## Total execution time: 30.8 seconds.
```

```r
get_seed(fit)
```

```
## [1] 401813683
```

```r
check_model(fit, data = dt)
```

```
## Processing csv files: C:/Users/frodr/AppData/Local/Temp/RtmpkDNfRE/varying_preferences-202406241412-
##
## Checking sampler transitions treedepth.
## Treedepth satisfactory for all transitions.
##
## Checking sampler transitions for divergences.
## No divergent transitions found.
##
## Checking E-BFMI - sampler transitions HMC potential energy.
## E-BFMI satisfactory.
##
## Effective sample size satisfactory.
##
## Split R-hat values satisfactory all parameters.
##
## Processing complete, no problems detected.
```

Preference (r) parameter: prior (dark grey) vs posterior (light grey) distribut

density

0.03

0.02

0.01

0.00

0   250   500   750

Log posterior across chains
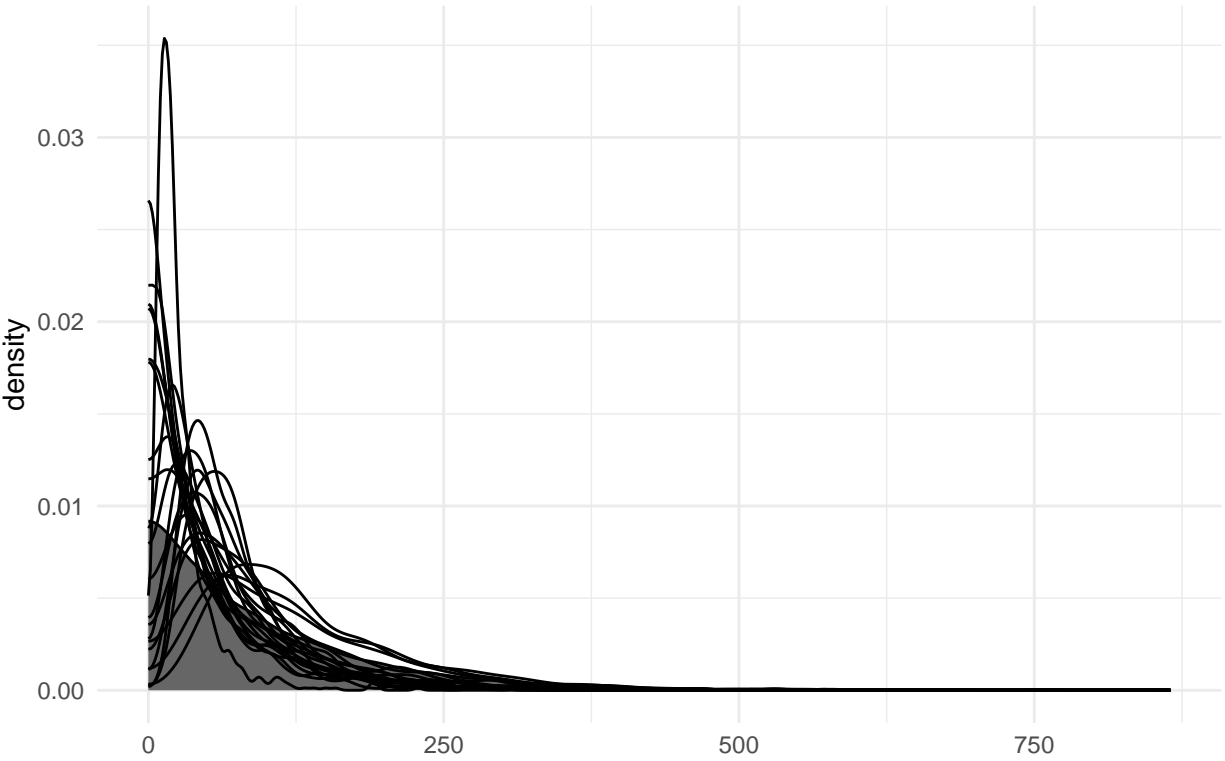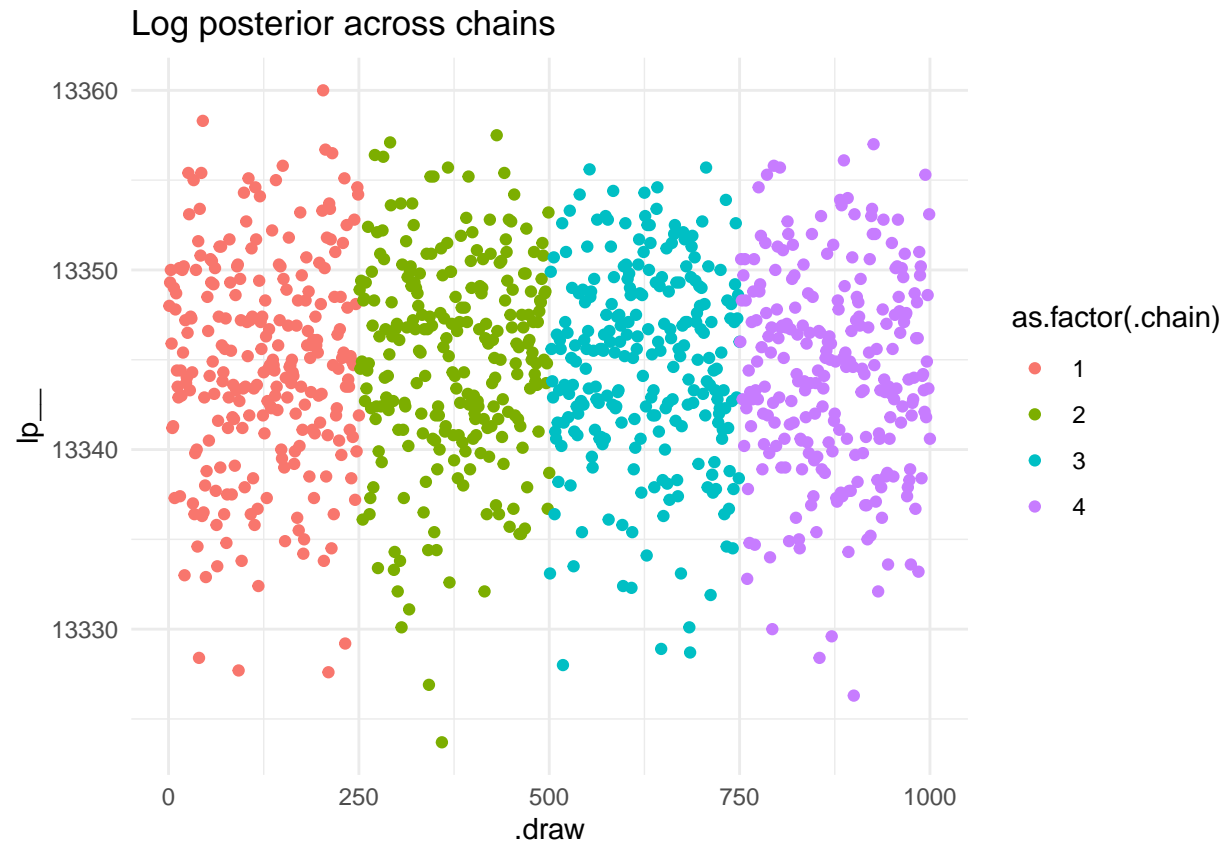
Posteriors

Get posterior distributions:

```
post <- get_posterior(fit, dt)
```

```
head(post)
```

```
## # A tibble: 6 x 11
## # Groups:   Animal, Plant [6]
##   Plant Animal        .chain .iteration .draw connectance preference plant.abund animal.abund int.prol
##   <chr> <chr>          <int>      <int> <int>       <dbl>      <dbl>       <dbl>        <dbl>       <dbl
## 1 A     Piaya_cayana       1          1     1       0.241       35.0      0.0212      0.00267       0.234
## 2 B     Piaya_cayana       1          1     1       0.241       35.0      0.0577      0.00267       0.222
## 3 C     Piaya_cayana       1          1     1       0.241       35.0      0.00477     0.00267       0.239
## 4 D     Piaya_cayana       1          1     1       0.241       35.0      0.196       0.00267       0.18
## 5 E     Piaya_cayana       1          1     1       0.241       35.0      0.0326      0.00267       0.230
## 6 F     Piaya_cayana       1          1     1       0.241       35.0      0.119       0.00267       0.203
```

Mean edge probability:

```
plot_interaction_prob(post)
```

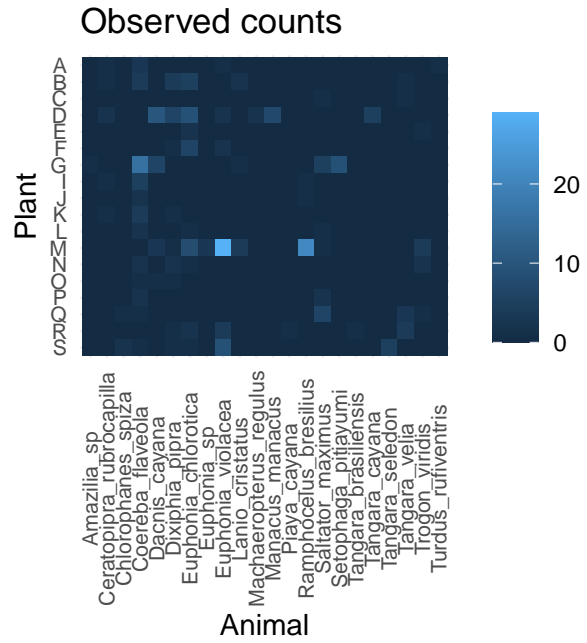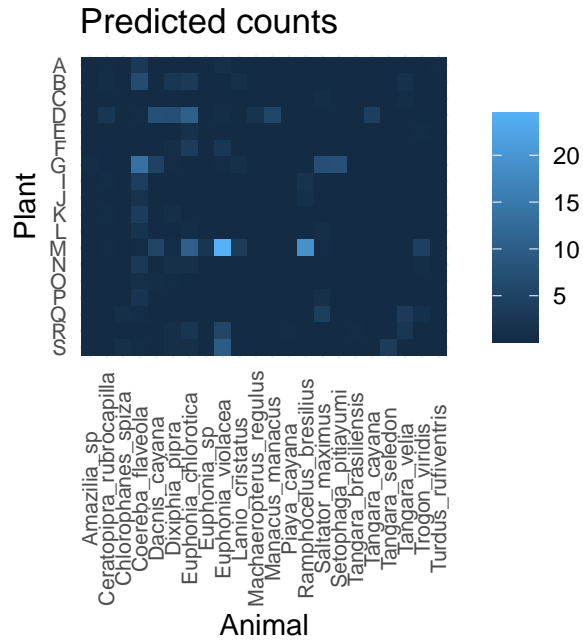Mean interaction probability
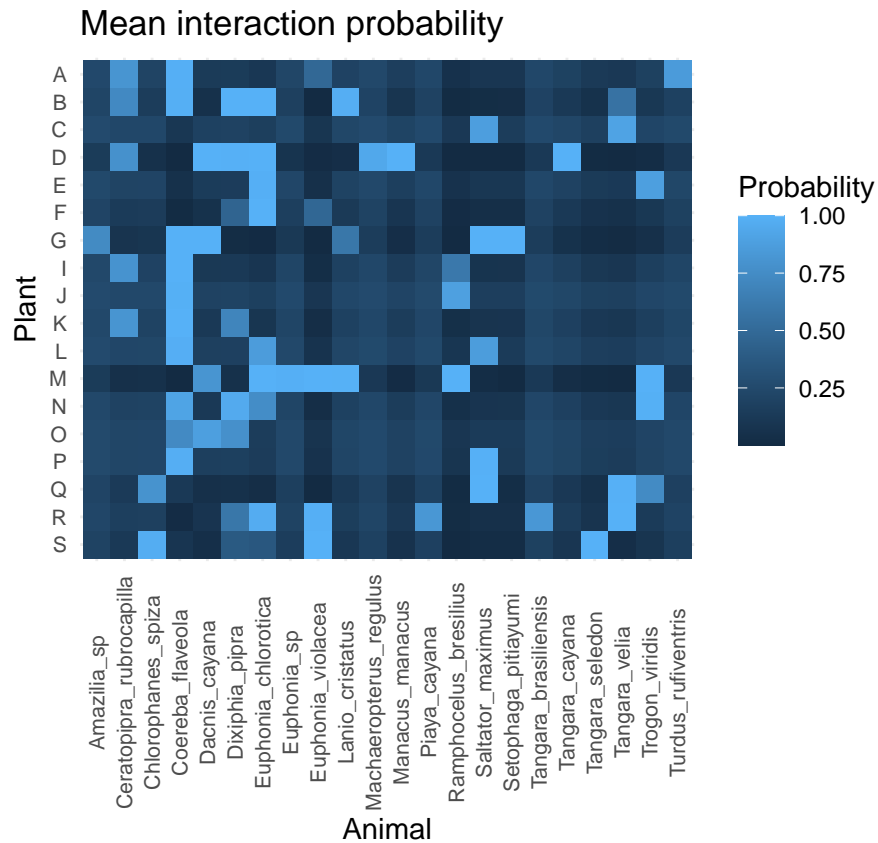
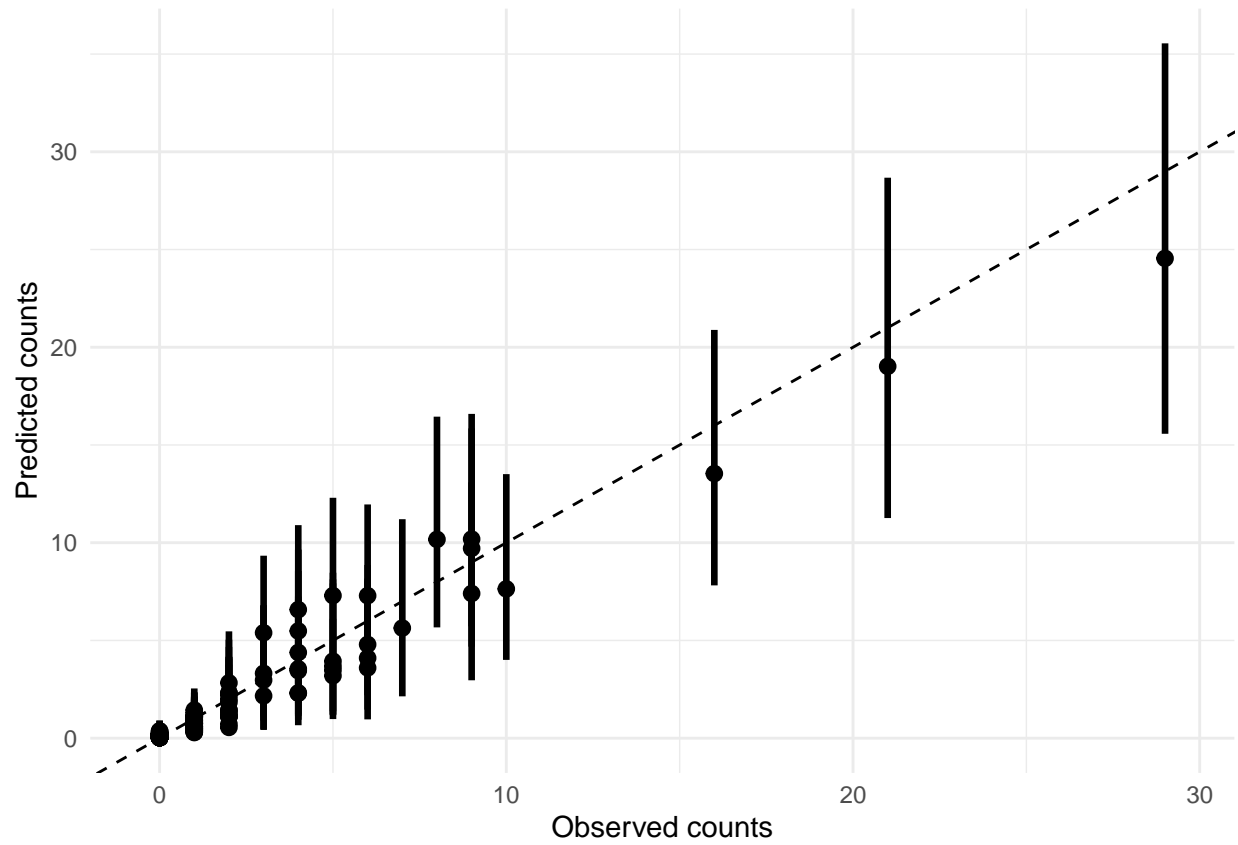## Generate predicted visits for each pairwise interaction

```
post.counts <- predict_counts(fit, dt)
```

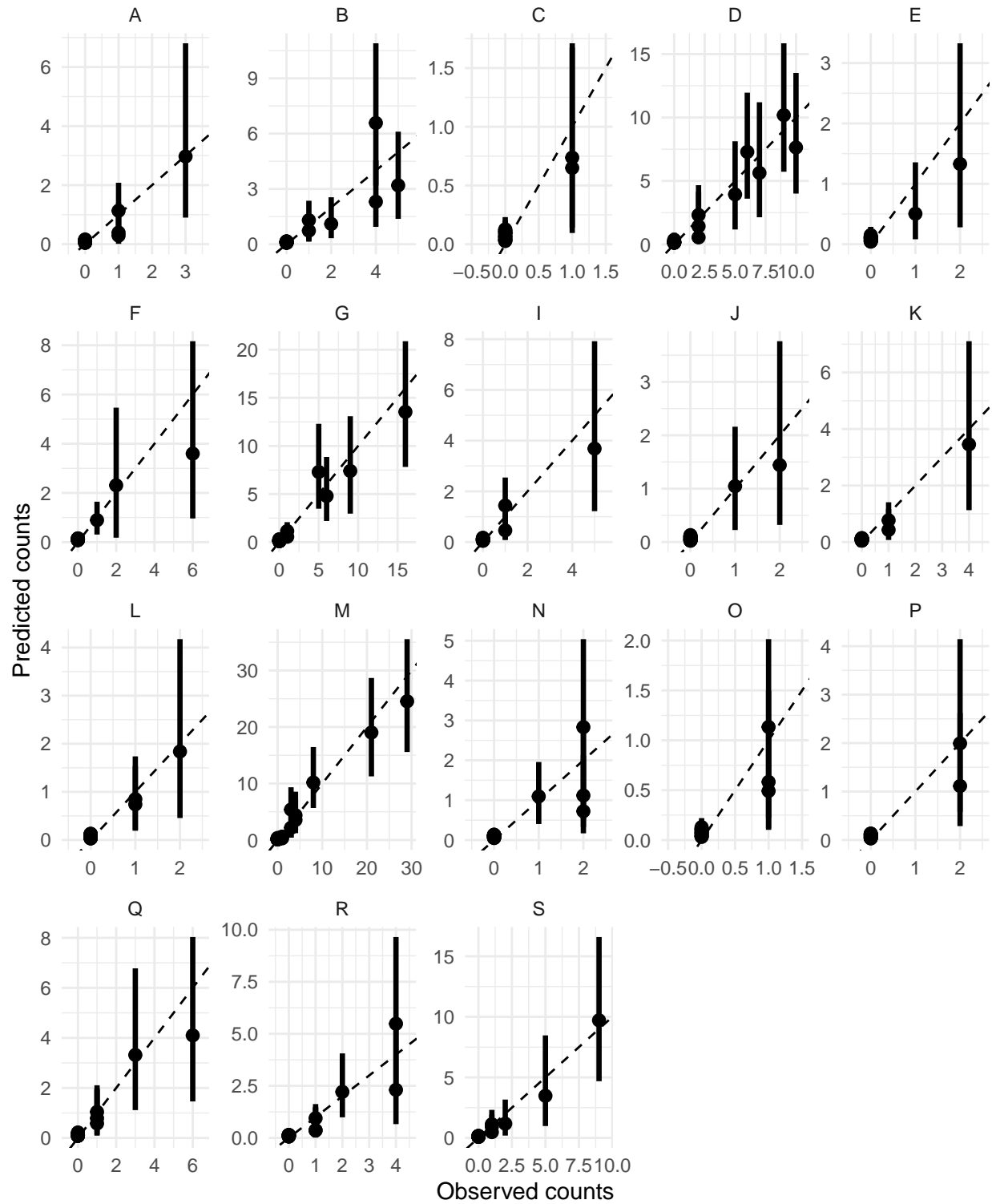Compare observed and predicted visits by the model:

```
p <- plot_counts_pred(post.counts, sort = FALSE)
o <- plot_counts_obs(mat, sort = FALSE, zero.na = FALSE)
library(patchwork)
p + o
```

```
plot_counts_pred_obs(post.counts, dt)
```

```
plot_counts_pred_obs(post.counts, dt, byplant = TRUE, scales = "free")
```

```
saveRDS(post.counts, here::here(paste0("data/nets_post/", params$net, "_post_counts.rds")))
```