

Bayesian inference of an individual-based mutualistic network

02_01

Net 02_01

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

Data

Load dataset and sampling effort per individual plant:

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

```
## Rows: 35 Columns: 11
## -- Column specification -----
## Delimiter: ","
## chr (1): ind
## dbl (10): Turdus_philomelos, Vulpes_vulpes, Turdus_merula, Erithacus_rubecula, Turdus_iliacus, Sylvia_...
##
## 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.
```

```
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: 35 Columns: 31
## -- Column specification -----
## Delimiter: ","
## chr (4): ind, area, fruit_type, fruit_color
## dbl (27): canopy_cover_m2, height_cm, crop, d1_cm, d2_cm, x, y, fruit_d1_mm, fruit_d2_mm, fruit_mass,
```

```

##
## 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("01_01", "01_02", "02_01", "02_02", "02_03", "10_01", "11_01",
               "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 == "10_01") {
  effort <- effort |>
    mutate(se_cam_days = se_cam_h/24) |>
    pull(se_cam_days, name = "ind")
}

if (net == "11_01") {
  effort <- effort |>
    mutate(se_cam_months = se_cam_days/30) |>
    pull(se_cam_months, name = "ind")
}

if (net == "15_01") {
  effort <- effort |>
    mutate(se_cam_months = se_cam_days/30) |>
    pull(se_cam_months, 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")
}

# For Pistacia and Juniperus, use constant sampling effort
if (net %in% c("01_01", "01_02", "02_01", "02_02", "02_03")) {
  effort <- rep(10, nrow(mat))
  names(effort) <- web$ind
}

## Some nets may require adjusting of the count data or effort values
if (net %in% c("02_01")) {
  mat <- mat/10 # divide counts by 10 to make modelling feasible
  mat[mat > 0 & mat < 1] <- 1 # don't miss rare counts
  mat <- round(mat)
  mat <- apply(mat, c(1,2), as.integer)
}

```

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

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

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      0.00   0.00   0.00  15.68   4.00  749.00
```

```
# if (max(mat) > 1000) {
#   stop("More than 1000 counts in some cell(s)")
# }
```

```
summary(effort)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       10     10     10     10     10     10
```

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

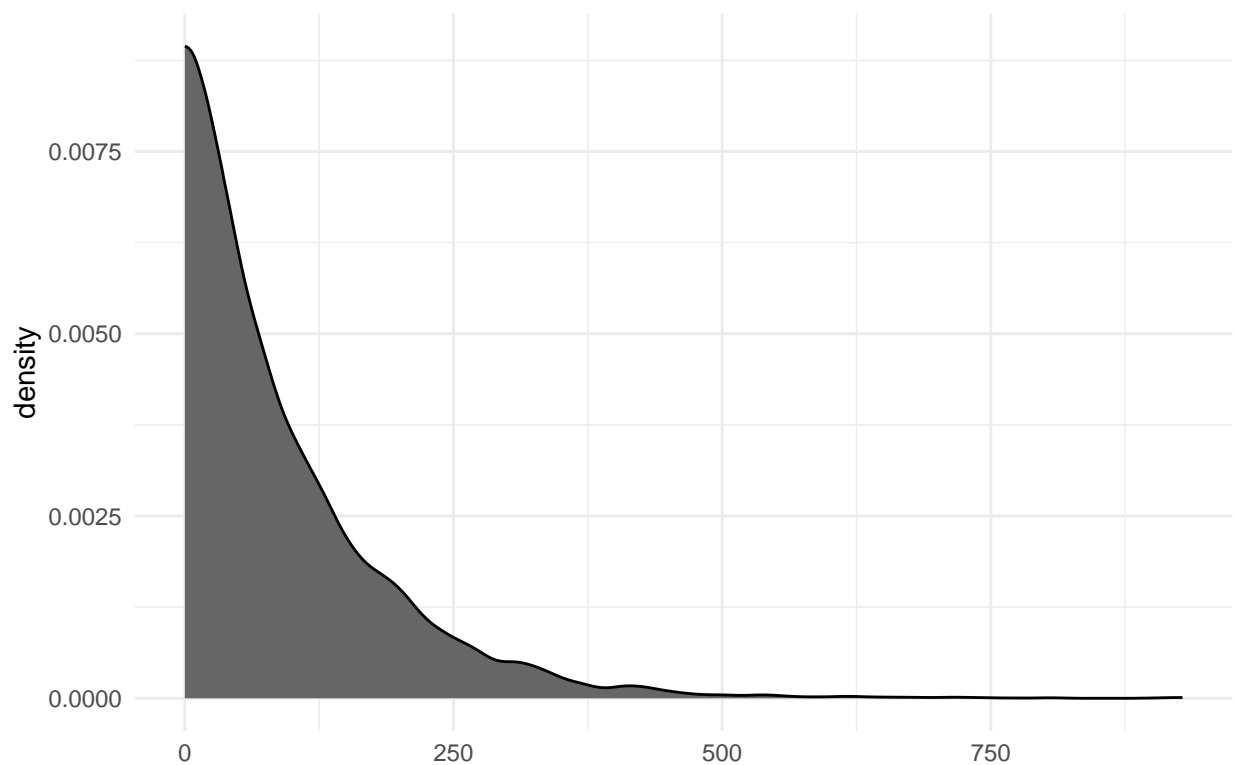
Bayesian inference of network structure

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

```
beta <- params$beta
```

```
plot_prior(beta)
```

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



```
fit <- fit_model(dt,
  refresh = 0,
  beta = 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)
```

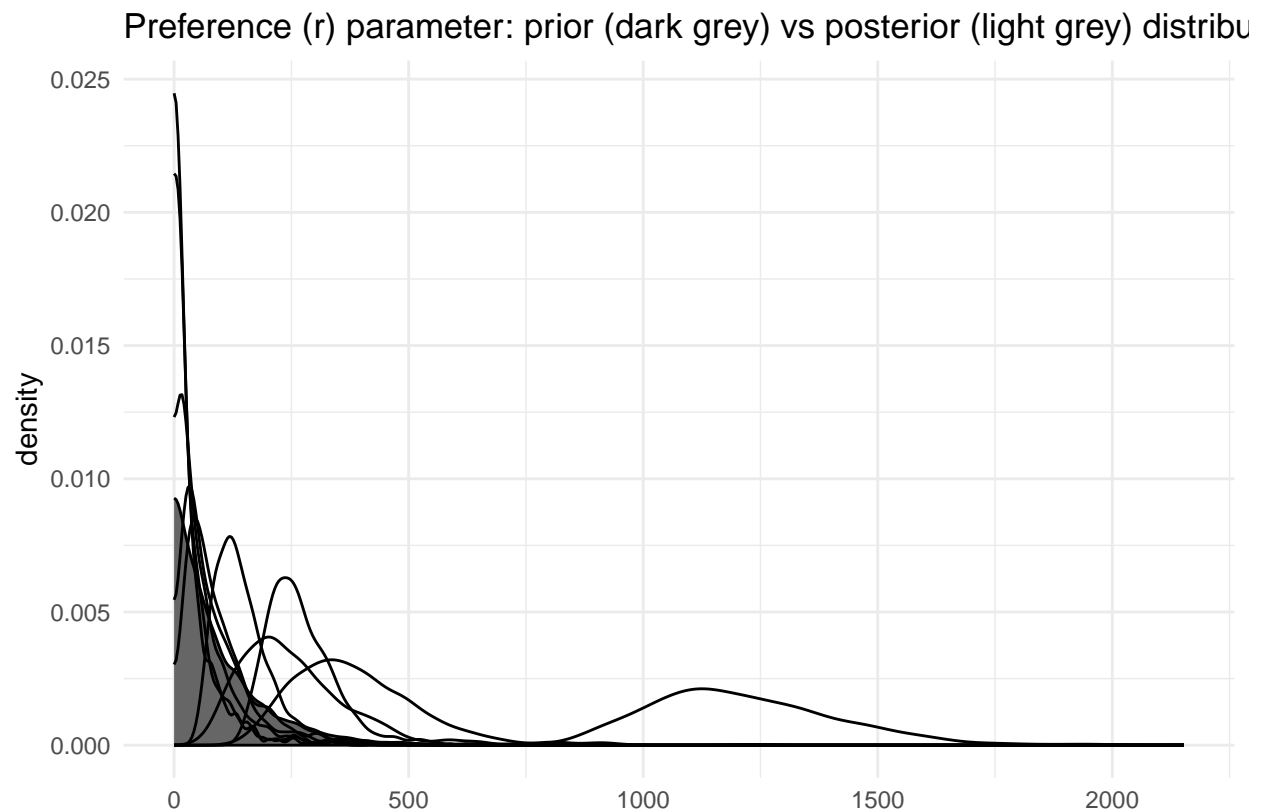
```
## Running MCMC with 4 parallel chains...
##
## Chain 4 finished in 500.6 seconds.
## Chain 1 finished in 514.0 seconds.
## Chain 2 finished in 516.9 seconds.
## Chain 3 finished in 518.1 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 512.4 seconds.
## Total execution time: 518.2 seconds.
```

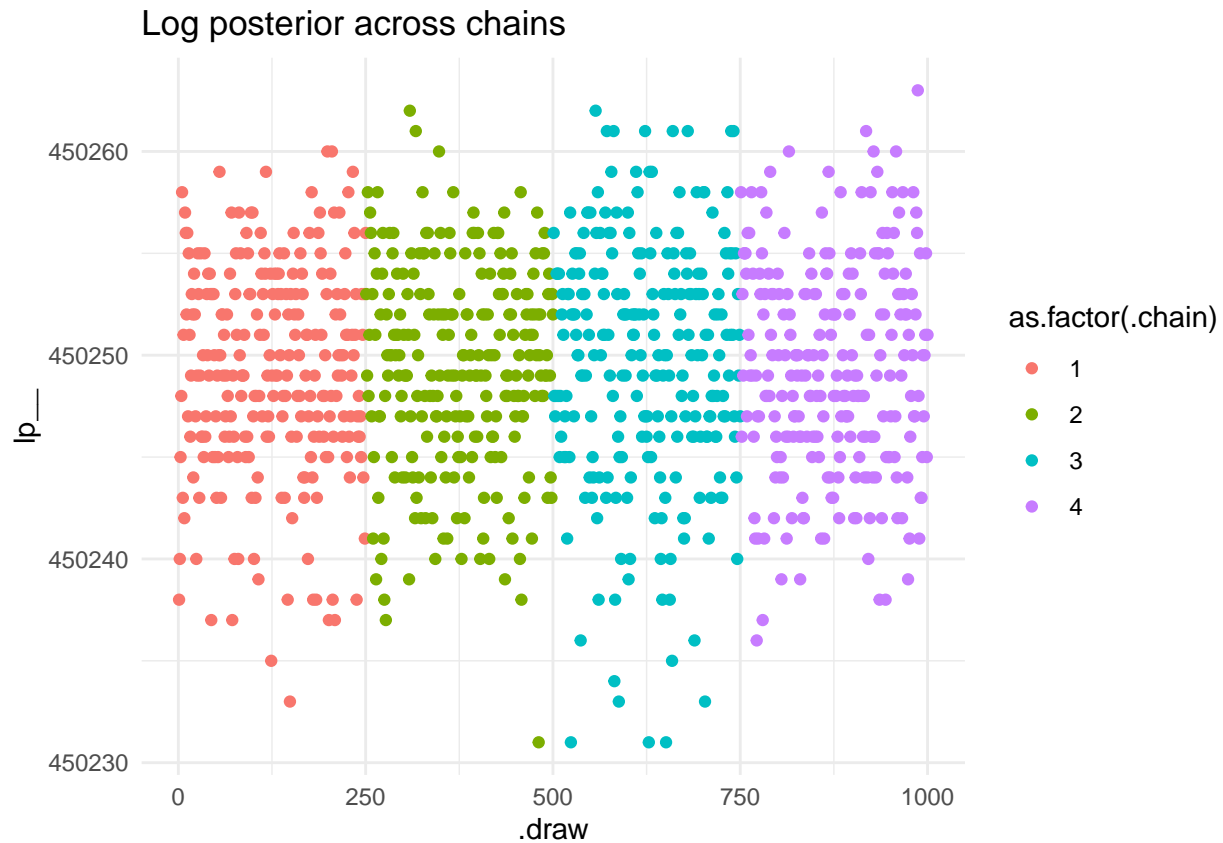
```
get_seed(fit)
```

```
## [1] 527933345
```

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

```
## Processing csv files: C:/Users/frodr/AppData/Local/Temp/Rtmp4ieGJu/varying_preferences-202406251325-  
##  
## 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.
```





Posteriors

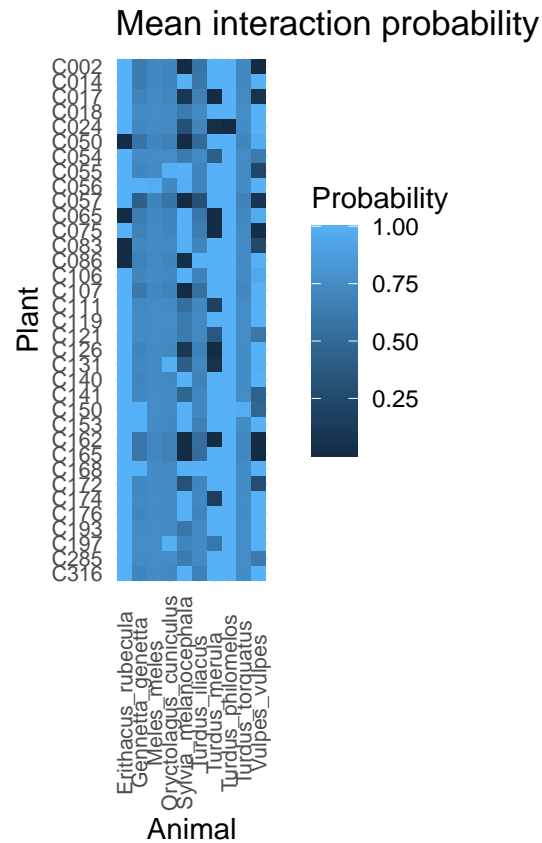
Get posterior distributions:

```
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.pr
##   <chr> <chr>      <int>      <int> <int>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 C002  Turdus_philo~    1          1     1         0.769      1061.      0.0502      0.436 1    e+
## 2 C014  Turdus_philo~    1          1     1         0.769      1061.      0.0502      0.436 1    e+
## 3 C017  Turdus_philo~    1          1     1         0.769      1061.      0.0319      0.436 1    e+
## 4 C018  Turdus_philo~    1          1     1         0.769      1061.      0.00419     0.436 1    e+
## 5 C024  Turdus_philo~    1          1     1         0.769      1061.      0.0111      0.436 2.39e-
## 6 C050  Turdus_philo~    1          1     1         0.769      1061.      0.0770      0.436 1    e+
```

Mean edge probability:

```
plot_interaction_prob(post)
```

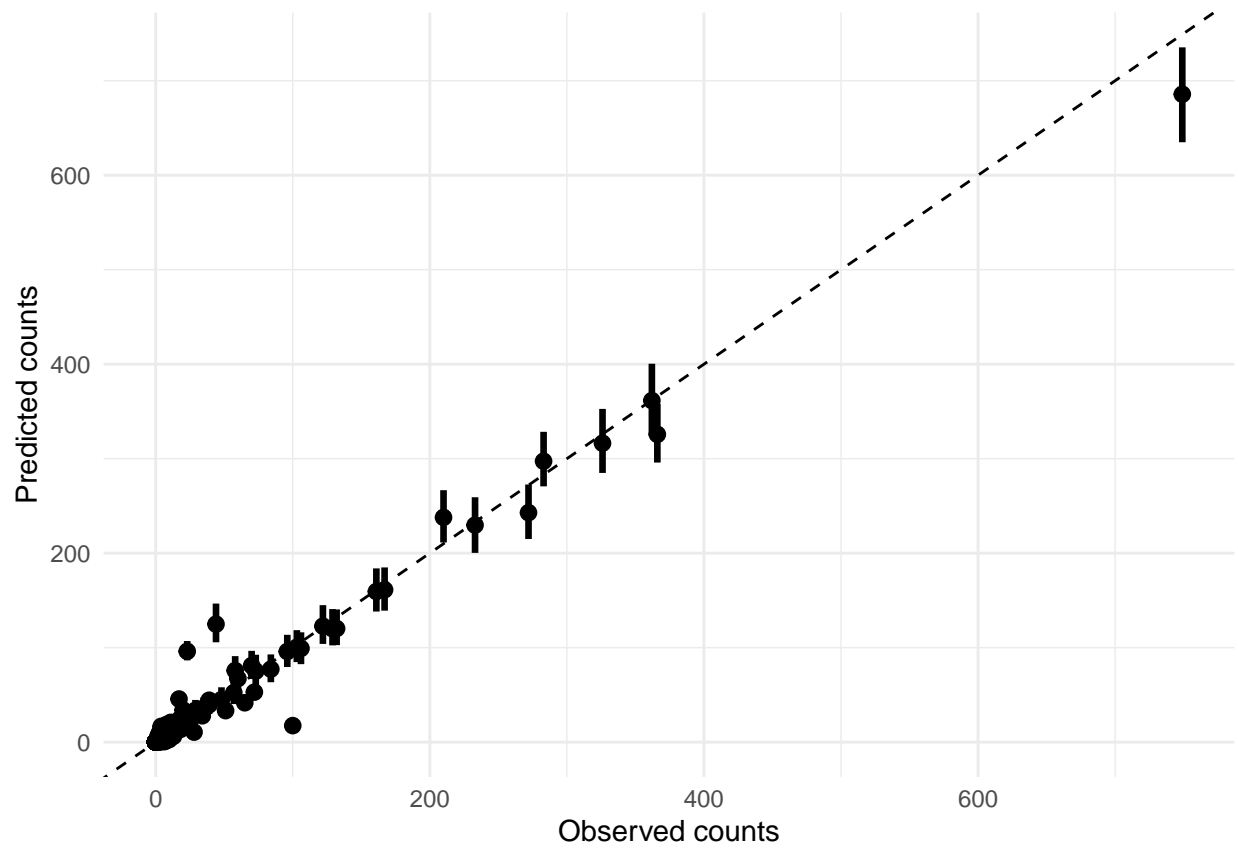


Generate predicted visits for each pairwise interaction

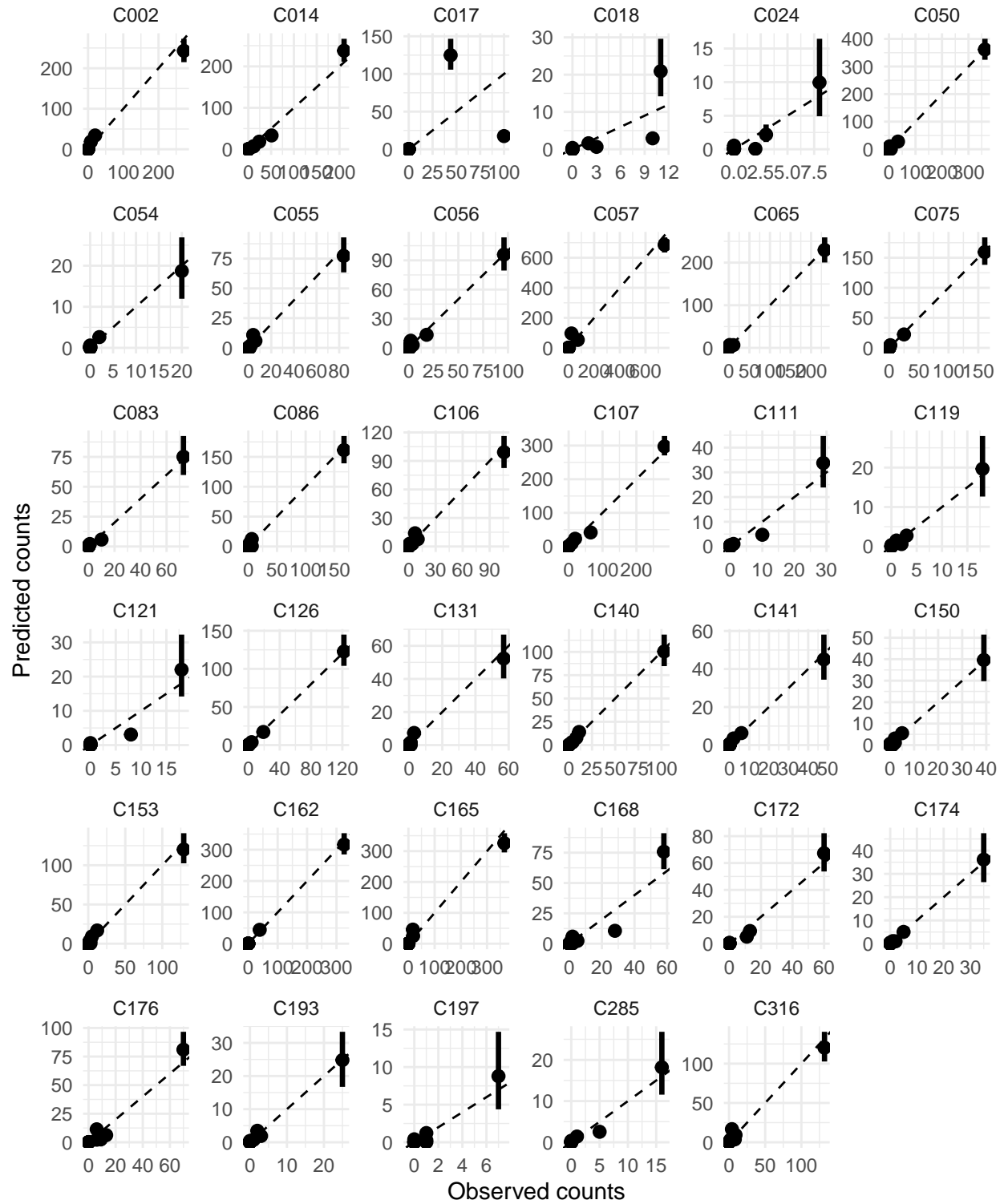
```
post.counts <- predict_counts(fit, dt)
```

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, byplant = TRUE, scales = "free")
```



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