

# Bayesian inference of an individual-based mutualistic network

04\_01

## Net 04\_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: 18 Columns: 18
## -- Column specification -----
## Delimiter: ","
## dbl (18): ind, Parus_caeruleus, Regulus_ignicapillus, Turdus_philomelos, Parus_major, Sylvia_atricap
##
## 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: 18 Columns: 17
## -- Column specification -----
## Delimiter: ","
## chr (4): pop, years_sampled, fruit_type, fruit_color
## dbl (13): ind, x, y, height_cm, crop_2003, crop_2004, crop_2005, basal_area_cm2, altitude, 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.
##      0.000   0.000   0.000   3.118   1.000  124.000

```

```

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

summary(effort)

```

```

##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##      1.000   4.562   7.833  10.475  17.000  25.300

```

```

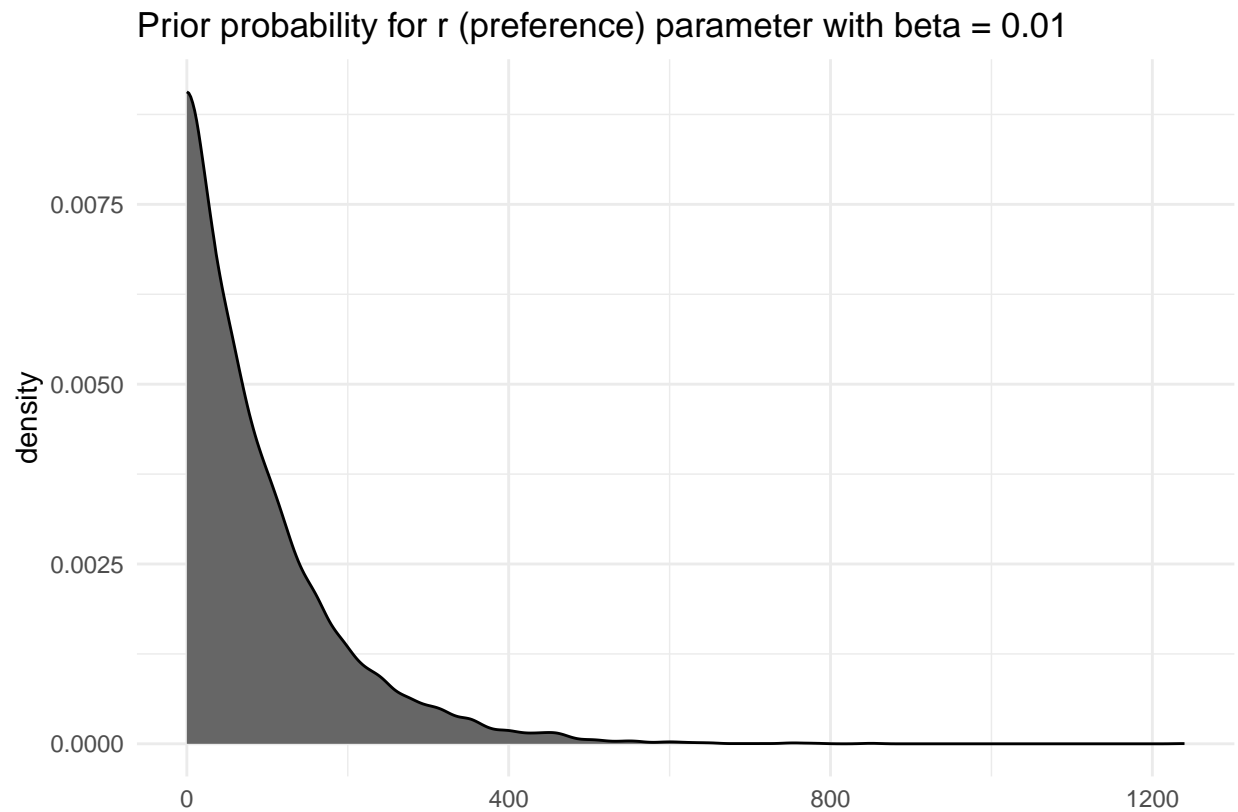
if (max(effort) > 500) {
  stop("Sampling effort > 500 for some plants")
}

```

## Bayesian inference of network structure

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

plot_prior(params$beta)
```



```
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 4 finished in 23.6 seconds.
## Chain 1 finished in 23.7 seconds.
## Chain 3 finished in 24.6 seconds.
## Chain 2 finished in 28.2 seconds.
##
## All 4 chains finished successfully.
```

```
## Mean chain execution time: 25.0 seconds.  
## Total execution time: 28.3 seconds.
```

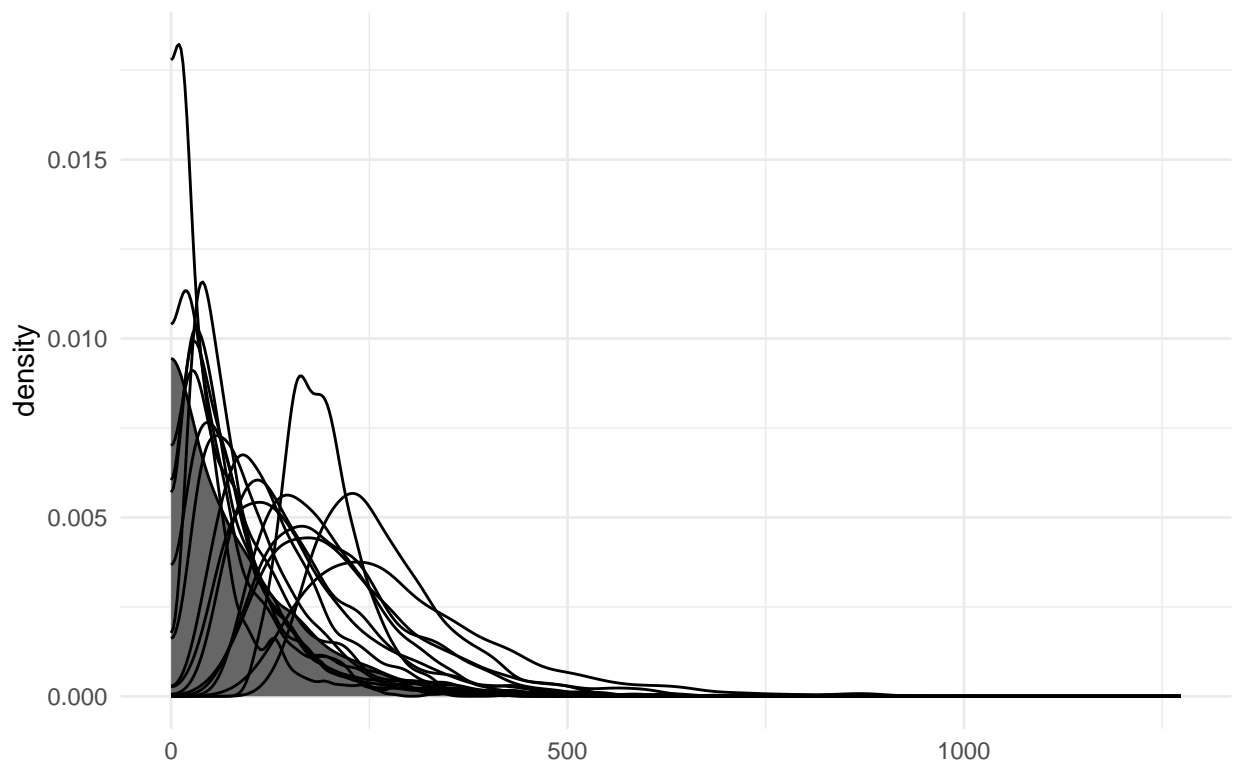
```
get_seed(fit)
```

```
## [1] 789035093
```

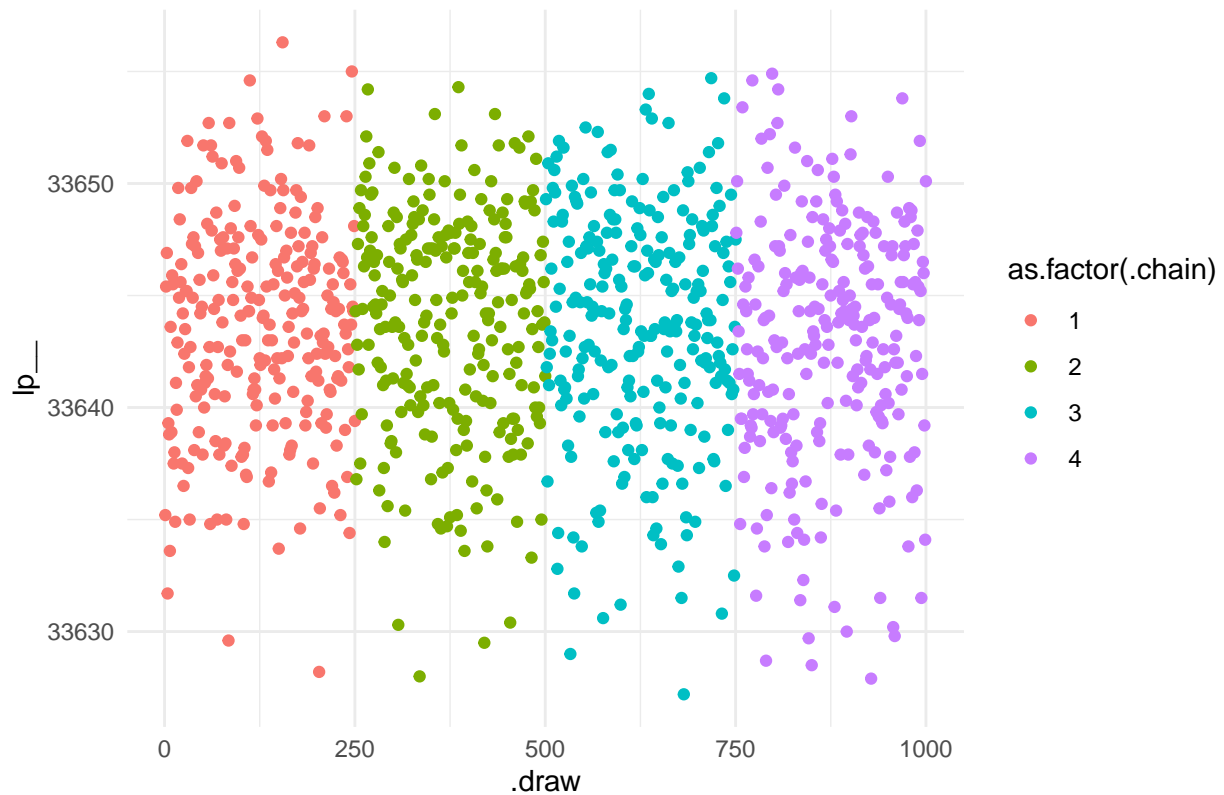
```
check_model(fit, data = dt)
```

```
## Processing csv files: C:/Users/frodr/AppData/Local/Temp/RtmpkDNfRE/varying_preferences-202406241406-  
##  
## 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) distribu



## Log posterior across chains



## Posteriors

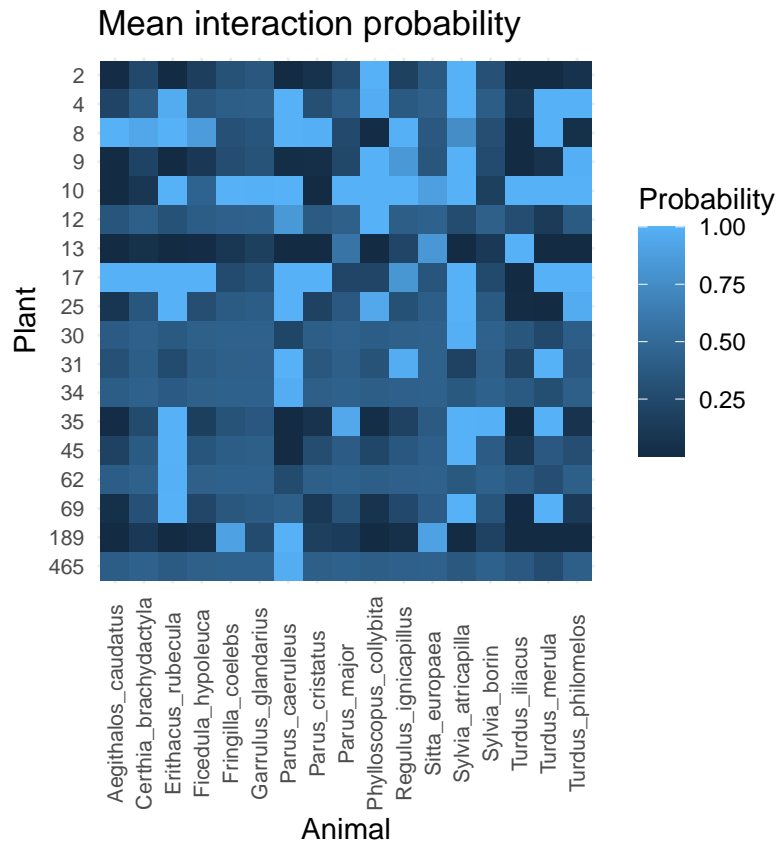
Get posterior distributions:

```
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.pr
##   <chr> <chr>      <int>      <int> <int>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 2     Parus_caerul~      1          1     1          0.419      317.      0.0433      0.160 1.16e-
## 2 4     Parus_caerul~      1          1     1          0.419      317.      0.00808     0.160 1 e+
## 3 8     Parus_caerul~      1          1     1          0.419      317.      0.0459      0.160 1 e+
## 4 9     Parus_caerul~      1          1     1          0.419      317.      0.0589      0.160 5.22e-
## 5 10    Parus_caerul~      1          1     1          0.419      317.      0.0868      0.160 1 e+
## 6 12    Parus_caerul~      1          1     1          0.419      317.      0.0212      0.160 7.35e-
```

Mean edge probability:

```
plot_interaction_prob(post)
```

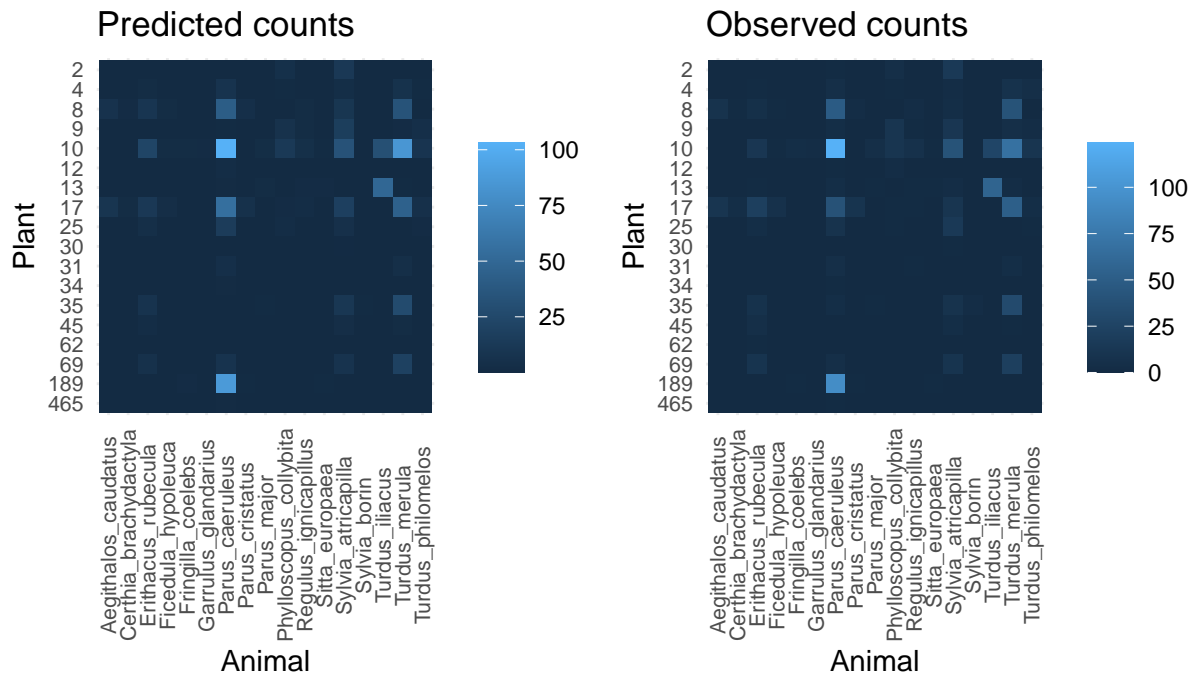


## Generate predicted visits for each pairwise interaction

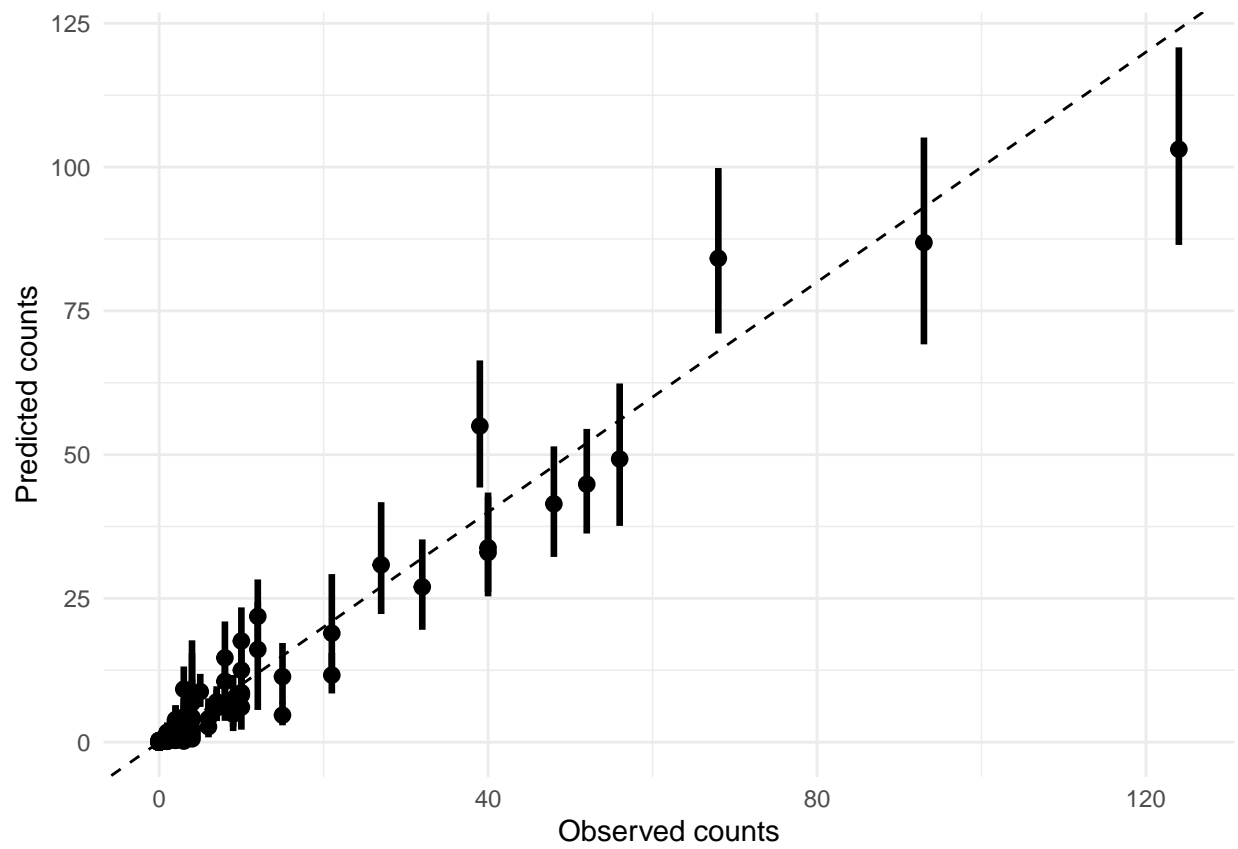
```
post.counts <- predict_counts(fit, dt)
```

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
```

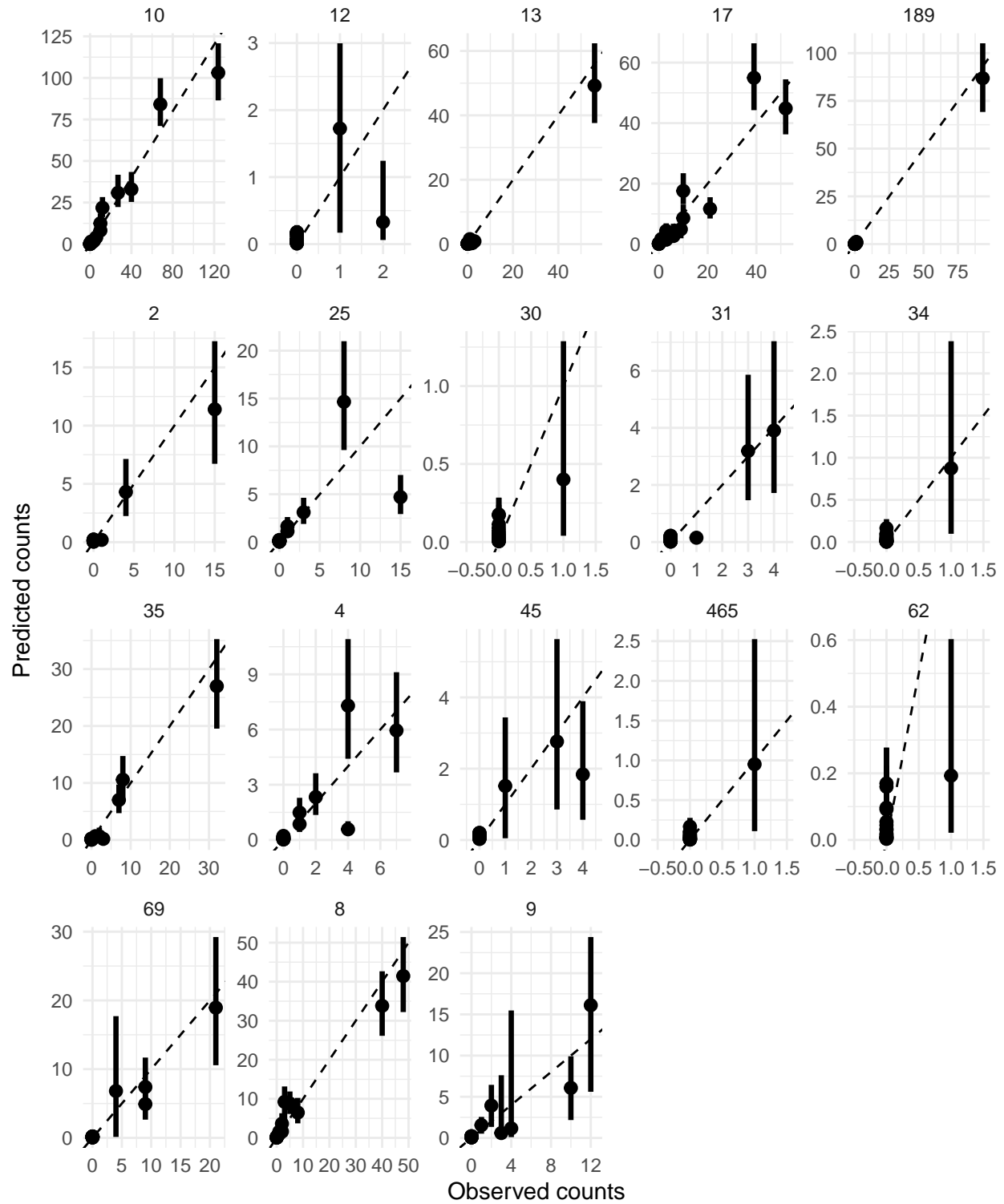


```
plot_counts_pred_obs(post.counts, dt)
```



```
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")))
```