

# Bayesian inference of an individual-based mutualistic network

13\_02

## Net 13\_02

```
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: 24 Columns: 7
## -- Column specification -----
## Delimiter: ","
## chr (1): ind
## dbl (6): Cervus_elaphus, Sus_scrofa, Oryctolagus_cuniculus, Rattus_sp, Vulpes_vulpes, Meles_meles
##
## 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: 24 Columns: 10
## -- Column specification -----
## Delimiter: ","
## chr (3): ind, fruit_type, fruit_color
## dbl (7): n_inf, canopy_cover_m2, crop, nearest_neigh, neigh_density_intra, neigh_radio, se_obs_days
```

```
##
## 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("10_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 == "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   1.493   2.000   12.000
```

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

summary(effort)
```

```
##      Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##       12      12      12      12     12     12
```

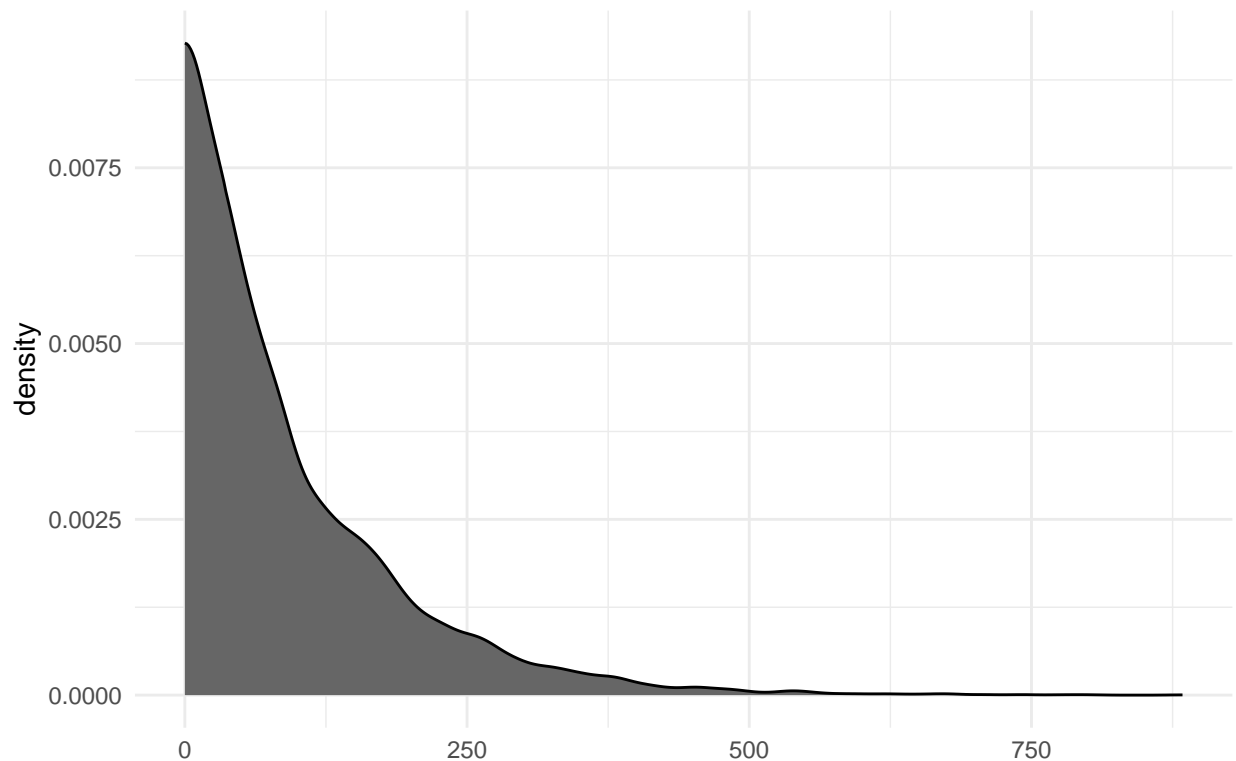
```
if (max(effort) > 500) {
  stop("Sampling effort > 500 for some plants")
}
```

## Bayesian inference of network structure

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

plot_prior(params$beta)
```

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



```
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)
```

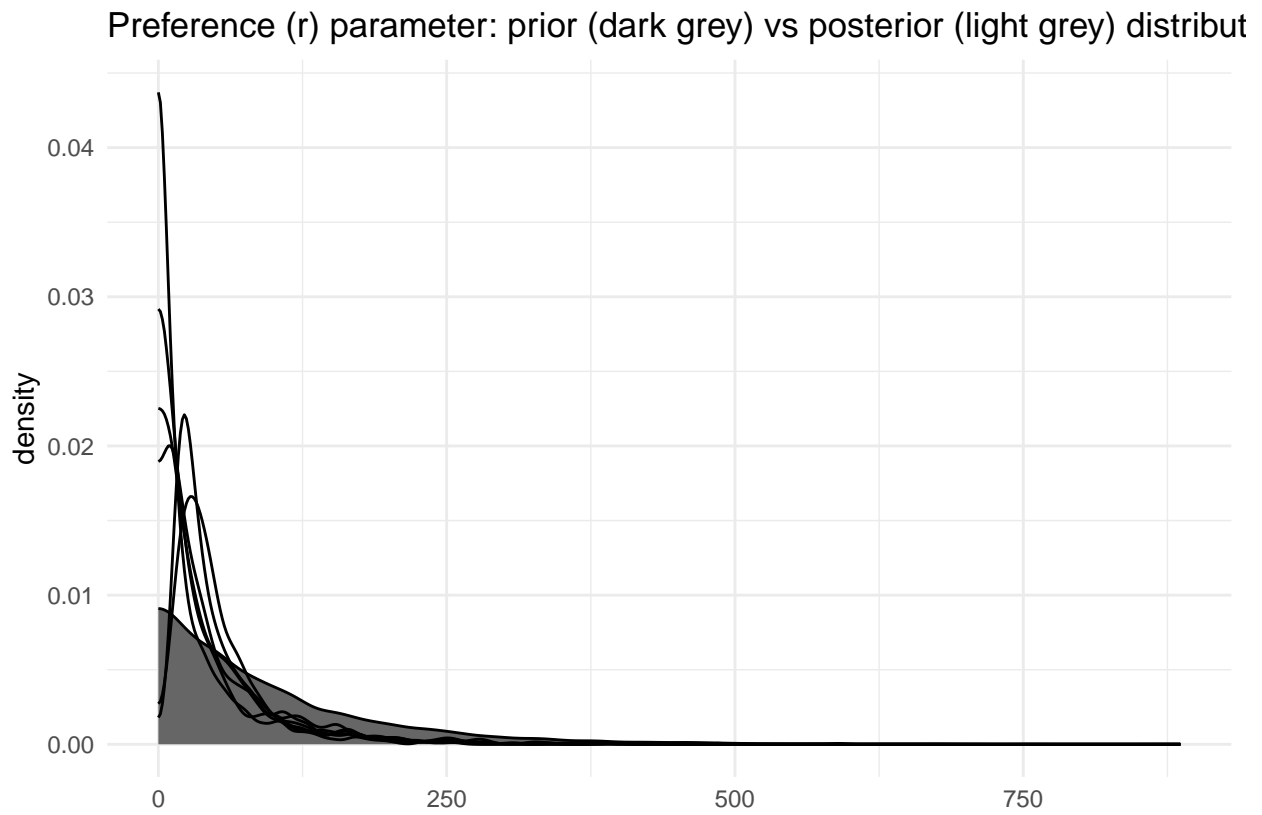
```
## Running MCMC with 4 parallel chains...
##
## Chain 3 finished in 14.9 seconds.
## Chain 1 finished in 17.3 seconds.
## Chain 2 finished in 17.6 seconds.
## Chain 4 finished in 18.6 seconds.
##
## All 4 chains finished successfully.
## Mean chain execution time: 17.1 seconds.
## Total execution time: 18.7 seconds.
```

```
get_seed(fit)
```

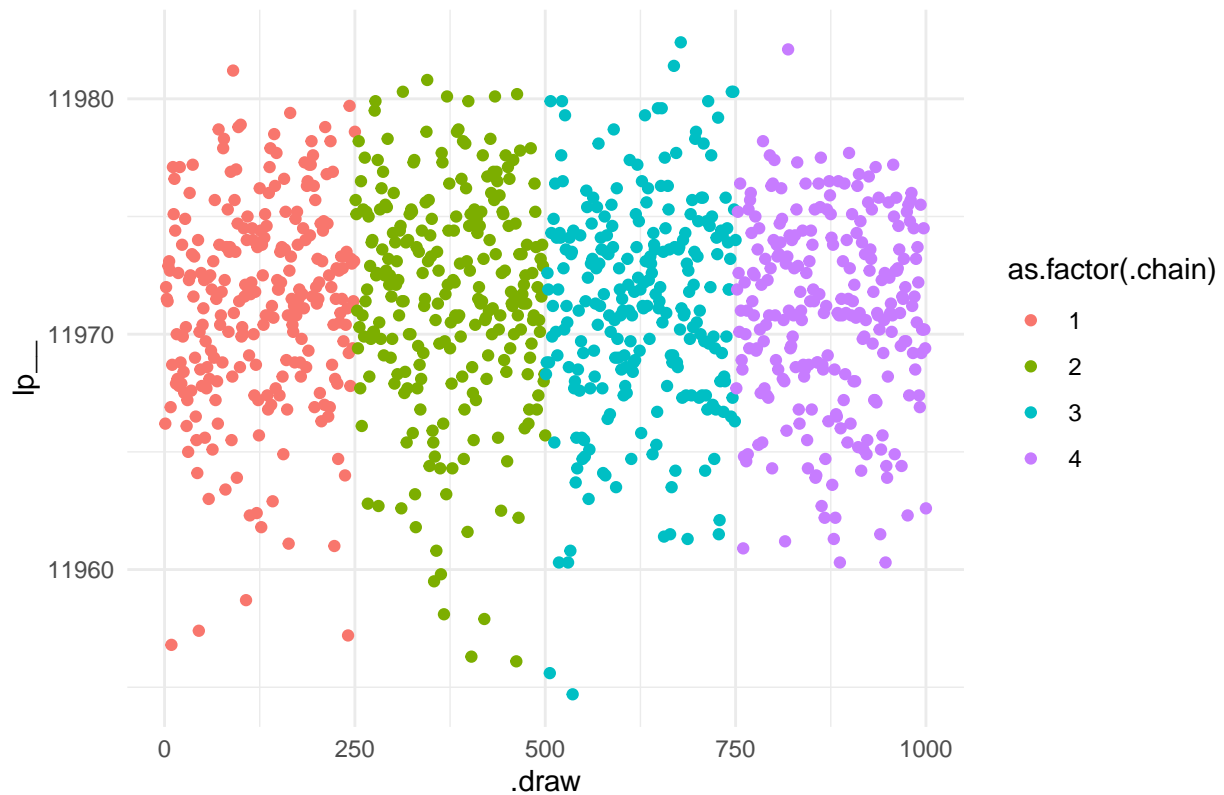
```
## [1] 714511546
```

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

```
## Processing csv files: C:/Users/frodr/AppData/Local/Temp/RtmpkDNfRE/varying_preferences-202406241428-
##
## 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.
```



## Log posterior across chains



## Posteriors

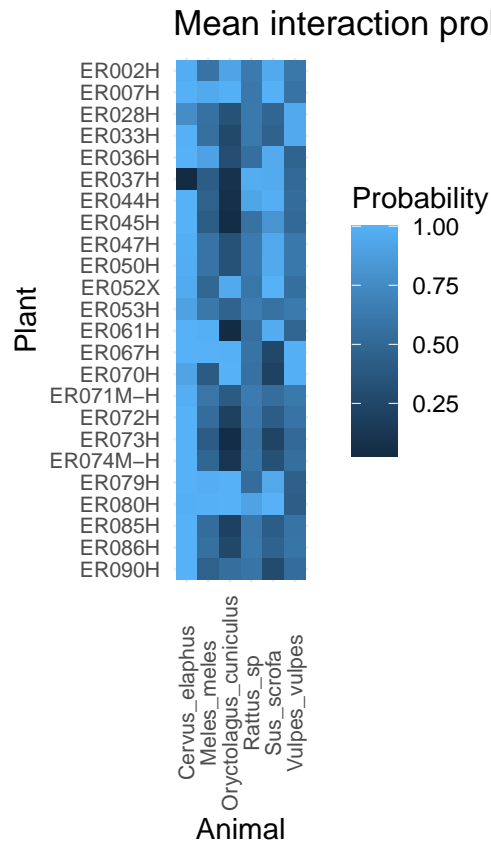
Get posterior distributions:

```
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 ER002H Cervus_elap~ 1         1     1      0.702    20.4     0.0263    0.637    9.47e
## 2 ER007H Cervus_elap~ 1         1     1      0.702    20.4     0.0162    0.637    9.99e
## 3 ER028H Cervus_elap~ 1         1     1      0.702    20.4     0.0107    0.637    9.05e
## 4 ER033H Cervus_elap~ 1         1     1      0.702    20.4     0.0328    0.637    9.93e
## 5 ER036H Cervus_elap~ 1         1     1      0.702    20.4     0.105     0.637    1.00e
## 6 ER037H Cervus_elap~ 1         1     1      0.702    20.4     0.0688    0.637    5.07e
```

Mean edge probability:

```
plot_interaction_prob(post)
```

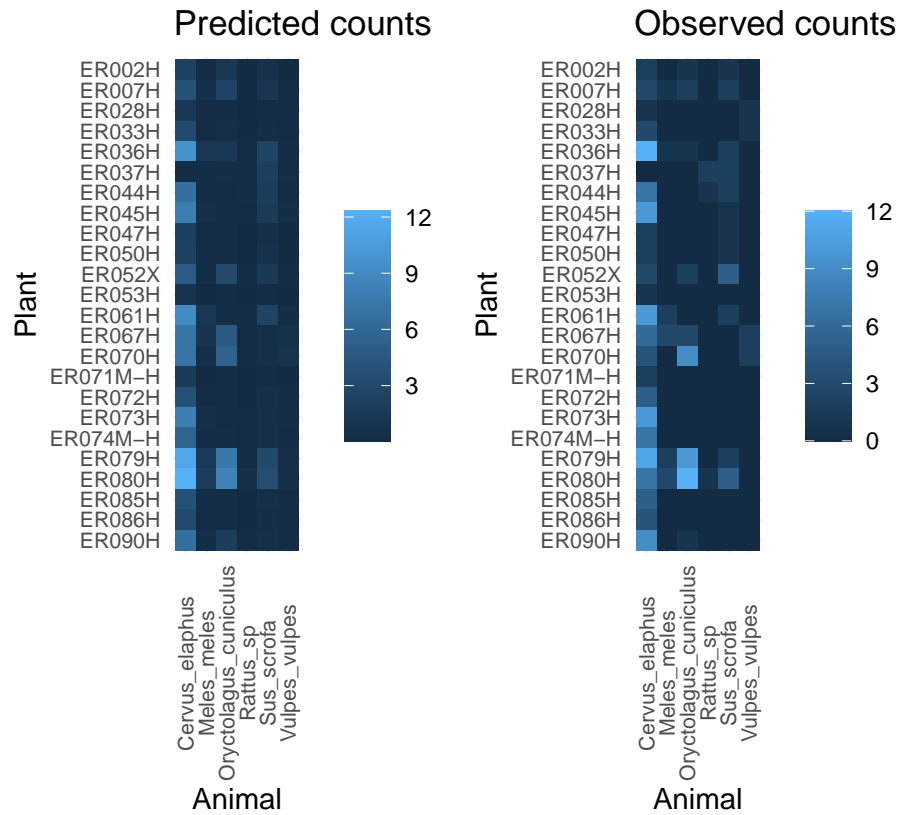


## Generate predicted visits for each pairwise interaction

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

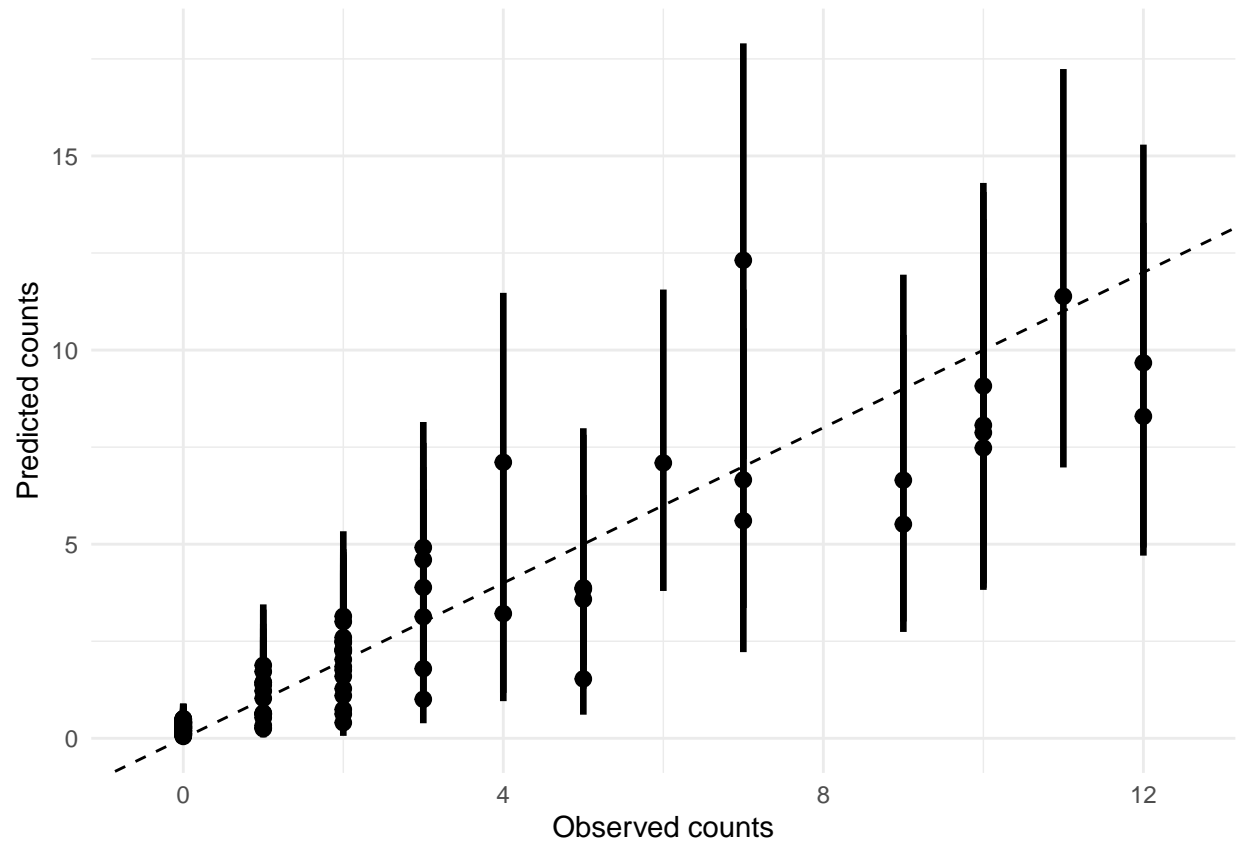
Compare observed and predicted visits by the model:

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

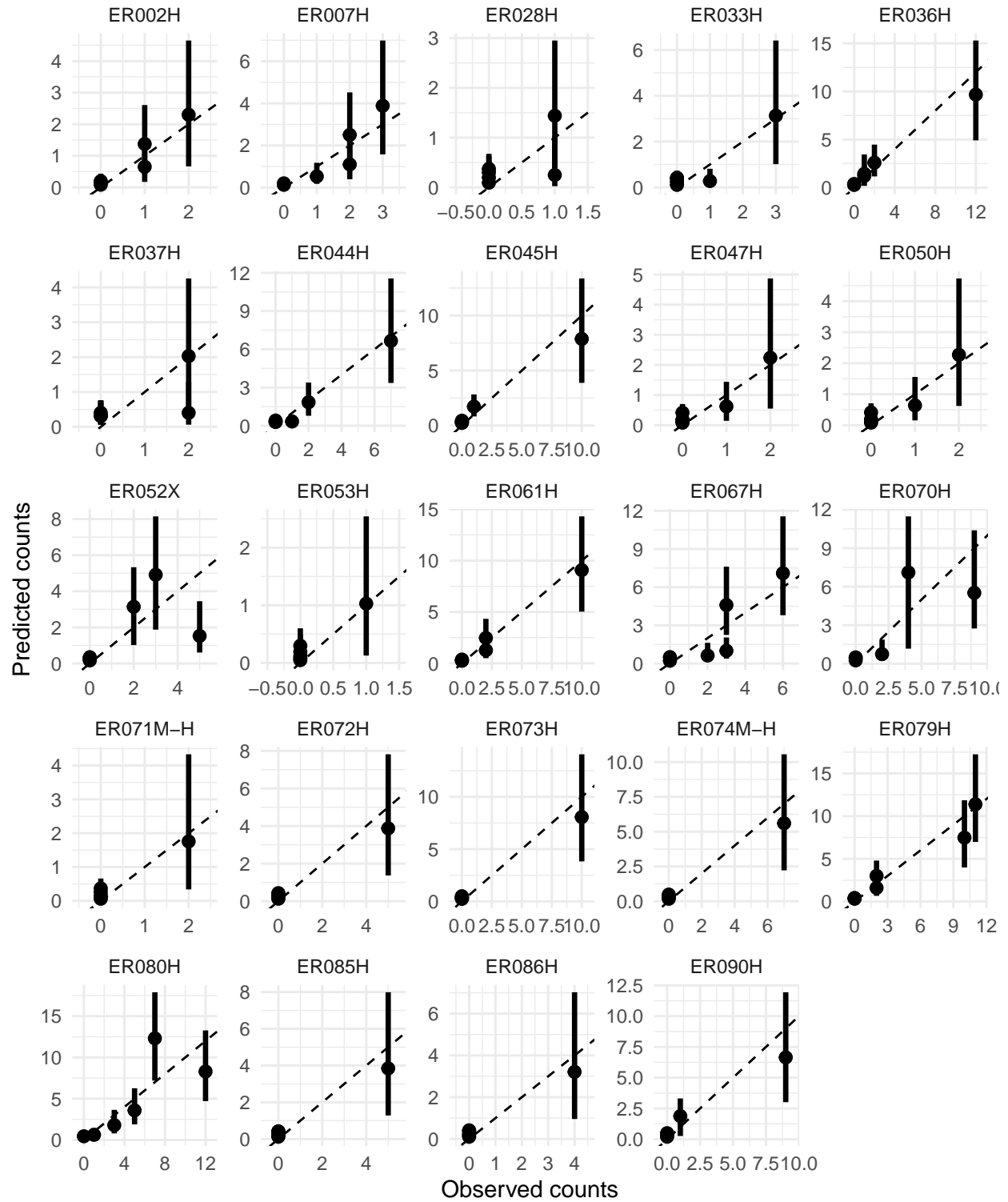


```
plot_counts_pred_obs(post.counts, dt)
```





```
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")))
```