Christmas_Island_probabilistic_approach

21 June 2019

In this document, we define the species interaction network for the Town region on Chirstmas Island. Then we perform the probabilistic modeling approach and species' response to different management perturbations are pooled.

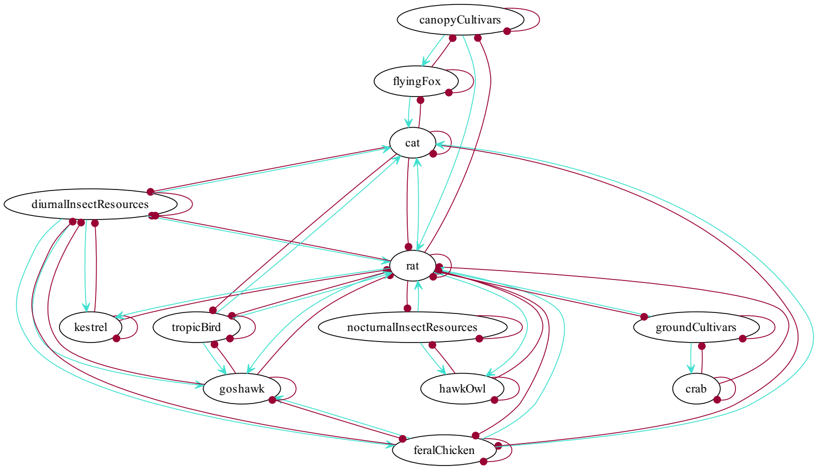
1. Define the species interaction network

We define the Christmas Island Town network by the species present, the interactions between them, and the signs of these interactions. The function initialise_foodweb returns a DiagrammeR graph obect, storing information of the network structure (nodes and edges) as dataframes (i.e. NDF: node data frame, and EDF: edge data frame). The network can be quickly plotted using function *render_graph()* from package *DiagrammeR* for a quick check of network structures. For a neat plot, use funciton *foodweb_neat_plot()*.

```r
spp_list = c(
  'cat',
  'rat',
  'crab',
  'goshawk',
  'hawkOwl',
  'tropicBird',
  'flyingFox',
  'feralChicken',
  'kestrel',
  'groundCultivars',
  'canopyCultivars',
  'diurnalInsectResources',
  'nocturnalInsectResources'
)

positive_edges_list = list(
    'cat'= c('tropicBird', 'flyingFox', 'diurnalInsectResources', 'feralChicken', 'rat'),
    'rat'= c('tropicBird', 'groundCultivars', 'canopyCultivars', 'diurnalInsectResources', 'nocturnalIns
    'crab'= c('groundCultivars'),
    'goshawk'= c('rat', 'tropicBird', 'diurnalInsectResources', 'feralChicken'),
    'hawkOwl'= c('rat', 'nocturnalInsectResources'),
    'flyingFox'= c('canopyCultivars'),
    'feralChicken'= c('diurnalInsectResources'),
    'kestrel'= c('rat', 'diurnalInsectResources')
)

negative_edges_list = list(
    'cat'= c('cat'),
    'rat'= c('rat', 'cat', 'crab', 'goshawk', 'hawkOwl', 'kestrel'),
    'crab'= c('crab'),
    'goshawk'= c('goshawk'),
    'hawkOwl'= c('hawkOwl'),
    'tropicBird'= c('tropicBird', 'cat', 'rat', 'goshawk'),
    'flyingFox'= c('flyingFox', 'cat'),
    'feralChicken'= c('feralChicken', 'cat', 'rat', 'goshawk'),
    'kestrel'= c('kestrel'),
    'groundCultivars'= c('groundCultivars', 'crab', 'rat'),
    'canopyCultivars'= c('canopyCultivars', 'flyingFox', 'rat'),
    'diurnalInsectResources'= c('diurnalInsectResources', 'cat', 'rat', 'goshawk', 'feralChicken', 'kes
    'nocturnalInsectResources'= c('nocturnalInsectResources', 'rat', 'hawkOwl')
)

unmonitored_spp_list = c(
    'crab',
    'kestrel',
    'groundCultivars',
    'canopyCultivars',
    'diurnalInsectResources',
    'nocturnalInsectResources')

unmonitored_spp_list_sim = c(
    'cat',
    'rat',
    'feralChicken',
    'crab',
    'kestrel',
    'groundCultivars',
    'canopyCultivars',
    'diurnalInsectResources',
    'nocturnalInsectResources')

control_list = c('cat','rat')

ambig_edges_list = list(
    list('rat'= c('crab')),
    list(
        'rat'= c('kestrel'),
        'kestrel'= c('rat')
    )
    # Remove interactions between rat and kestrel, which indicates the ambiguous links
    # between these two species were keeped or removed together. However, there could
    # be cases that the bidirectional link were considered as two ambiguous links respectively,
    # In such case, the two links will not be wrapped together, and thus they could be treated
    # respectively.
)
```

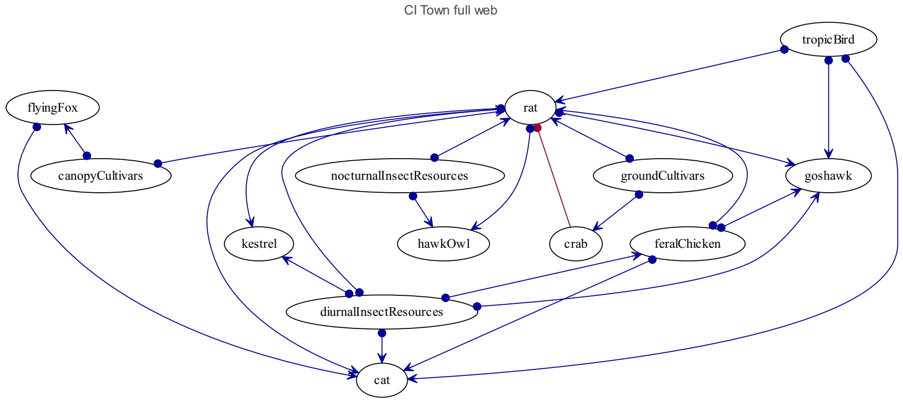
Initialise the full Town network.

```
full_web <- initialise_foodweb(positive_edges_list, negative_edges_list)
render_graph(full_web) # for a quick plot
```



Get a neat plot of the full Town network.

```r
foodweb_neat_plot(full_web, title = "CI Town full web")
```



Using the function *qualitative_community_matrix*, we can convert the interaction network into a qualitative community matrix (Mq), and get two named vectors *labelToIndex* and *indexToLabel* to map species labels (i.e. names) to indices of matrix and vice versa.

```
output <- qualitative_community_matrix(full_web)
Mq <- output$Mq
Mq
```

```
##       [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10] [,11] [,12] [,13]
##  [1,]   -1    0    0    0    0   -1    0    0    0     0     0    -1     0
##  [2,]    0   -1    0    1    1    1    0    0    0     0     0     1     1
##  [3,]    0    0   -1    0    0    0    0    1    0     0     0     0     0
##  [4,]    0   -1    0   -1   -1    0   -1    0    0    -1     0    -1     0
##  [5,]    0   -1    0    1   -1    0   -1    0    0     0     0    -1     0
##  [6,]    1   -1    0    0    0   -1    0    0    0     0     0     0     0
##  [7,]    0    0    0    1    1    0   -1    0    0     0     0     1     1
##  [8,]    0    0   -1    0    0    0    0   -1    0     0     0    -1     0
##  [9,]    0    0    0    0    0    0    0    0   -1     0     1     1     0
## [10,]    0    0    0    1    0    0    0    0    0    -1     0     1     0
## [11,]    0    0    0    0    0    0    0    0   -1     0    -1    -1     0
## [12,]    1   -1   -1    1    1    0   -1    1   -1    -1     1    -1     1
## [13,]    0   -1    0    0    0    0   -1    0    0     0     0    -1    -1
```

```
labelToIndex <- output$labelToIndex
indexToLabel <- output$indexToLabel

unname(indexToLabel[10])
```

```
## [1] "kestrel"
```

```
unname(labelToIndex["kestrel"])
```

```
## [1] 10
```

Get following vectors: - *spp_list_idx* represents indices of species in *spp_list* in order.. - *control_list_idx* represents indices of pest species in *control_list* in order. - *monitored_spp_list_sim* is a vector of species being monitored in the probabilistic approach. - *monitored_spp_idx_sim* represents indices of species in *monitored_spp_list_sim*.

```
spp_list_idx <- unname(labelToIndex[spp_list])
control_list_idx <- unname(labelToIndex[control_list])

monitored_spp_list_sim <- spp_list[!spp_list %in% unmonitored_spp_list_sim]
monitored_spp_idx_sim <- unname(labelToIndex[monitored_spp_list_sim])
```

2.The probabilistic approach: randomly sampling perturbation responses

The following *for* loop run simulations to collect species responses. To save time in this illustration, we run 10^5 times of simulations.

Ambiguous links are removed randomly from the network, which results in different network structures. Modelling outcomes of different network structures are pooled. All the simulation outcomes are stored in a list *collectedResponses*.

```
set.seed(178)

nSim <- 10000

collectedResponses = list()
sz <- dim(Mq)
n <- length(Mq)
```

```
noAmbig <- length(ambig_edges_list)

start_time <- Sys.time()

for (i in 1: nSim){

    selectAmbig <- as.logical(rbinom(noAmbig, 1, runif(1))) # randomly select ambig.links to remove

    if (all(!selectAmbig)) {

        Mq = Mq

    } else {

        ambig_df<-
            ambig_edges_list[selectAmbig] %>%
            melt() %>%
            setNames(., c("labelfrom", "labelto", "list")) %>%
            mutate_if(is.factor, as.character)

        dropEdge <- cbind(unname(labelToIndex[ambig_df[, "labelto"]]), unname(labelToIndex[ambig_df[, "l
        Mq[dropEdge] <- rep(0, nrow(ambig_df)) # set those dropped links as 0s.
    }


    valid <- FALSE

    while (!valid) {

      # find a random community matrix that is stable
      maxEig = 1

      while (maxEig > 0) {

          M = matrix(runif(n), sz[1], sz[2]) * Mq
          maxEig <- max(Re(eigen(M, symmetric=FALSE, only.values=TRUE)$values))
      }

      # Now have a valid stable matrix, find the sensitivity matrix
      Sq <- -solve(M)

      # check validation criteria: the pest respond negatively to management
      control_easy_cat = (Sq[labelToIndex['cat'],labelToIndex['cat']] > 0)
      control_easy_rat = (Sq[labelToIndex['rat'],labelToIndex['rat']] > 0)
      valid <- all(control_easy_cat, control_easy_rat)
    }

    #Now have a valid stable community matrix
    response <- vector()

    for (ps in control_list_idx) {

        resp <- ifelse(-Sq[monitored_spp_idx_sim, ps] < 0, "neg", "pos")
        response <- append(response, resp)
    }

    collectedResponses[[i]] <- response

}

end_time <- Sys.time()
time_elapsed = end_time - start_time
print(time_elapsed)
```

```
## Time difference of 11.7671 secs
```

Convert the list *collectedResponses* storing simulation outcomes into a dataframe *df_responses*.

```
df_responses <- do.call(rbind, collectedResponses) %>% as.data.frame() %>% mutate_if(is.factor, as.chara

colnames <- unlist(lapply(control_list, function(x) paste0(x, "_", monitored_spp_list_sim)))
colnames(df_responses) <- colnames
head(df_responses,3) # check the dataframe.
```

```
##   cat_goshawk cat_hawkOwl cat_tropicBird cat_flyingFox rat_goshawk
## 1         pos         neg            pos           pos         neg
## 2         pos         neg            pos           pos         pos
## 3         pos         neg            pos           pos         pos
##   rat_hawkOwl rat_tropicBird rat_flyingFox
## 1         neg            neg           pos
## 2         neg            pos           pos
## 3         neg            neg           neg
```

3.Aggregte outcomes for different managements

Aggregate outcomes for cat management only.

```
df_cat_resp <- df_responses %>%
    select(., starts_with("cat"))
count_cat <- sapply(df_cat_resp, table)
count_cat
```

```
##     cat_goshawk cat_hawkOwl cat_tropicBird cat_flyingFox
## neg         534        5001           3228          1486
## pos        9466        4999           6772          8514
```

Aggregate outcomes for the combined cat and rat management.

```
combined_resp_list <- list()

for (sp in monitored_spp_list_sim){

    # For each species, combine its response to cat and response to rat in each of the simulations.
    combined_response <-
        cbind(select(df_responses, contains(sp))) %>%
        apply(., 1 , paste , collapse = "&" )

    combined_resp_list[[length(combined_resp_list)+1]] <- combined_response
}
```

```
df_combined_resp <-
    do.call(cbind, combined_resp_list) %>%
    as.data.frame()

colnames_combined_df <- unlist(lapply(monitored_spp_list_sim, function(x) paste0('cr_', x)))
colnames(df_combined_resp) <- colnames_combined_df

lvls_cr <- c('pos&pos', 'pos&neg', 'neg&pos','neg&neg')
df_combined_resp[] <-  lapply(df_combined_resp, factor, levels=lvls_cr)

count_cr <- sapply(df_combined_resp, table)
count_cr
```

```
##         cr_goshawk cr_hawkOwl cr_tropicBird cr_flyingFox
## pos&pos       5052       1674          5217         6045
## pos&neg       4414       3325          1555         2469
## neg&pos        427       3149          2725         1485
## neg&neg        107       1852           503            1
```