Appendix C: Implementing the probabilistic approach and the Boolean approach on the Town network of Christmas Island

The following tutorials are available from the Github repository as R notebooks:

> https://github.com/yhan178/qualitative-modeling-r

The R code for performing the probabilistic approach and Boolean analysis was derived from the programming software accompanying Kristensen et al. (2019), which was developed in Python environment.

> Kristensen, N. P., R. A. Chisholm, and E. McDonald-Madden. 2019. Dealing with high uncertainty in qualitative network models using Boolean analysis. Methods in Ecology and Evolution 00:1–14. DOI: 10.1111/2041-210x.13179

C3.1 Implementing the probabilistic approach to the Town network of Christmas Island

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.

The tutorial R notebook also source two scripts with functions ('qualmod.R' and 'findPCU.R') to help produce the results. The source codes were all available on the Github repository:

> https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/qualmod.R

> https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/findPCU.R

```
library(devtools)
devtools::source_url("https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/qualmod
#url:"https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/qualmod.R?raw=TRUE")
```

C3.1.1 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()*.

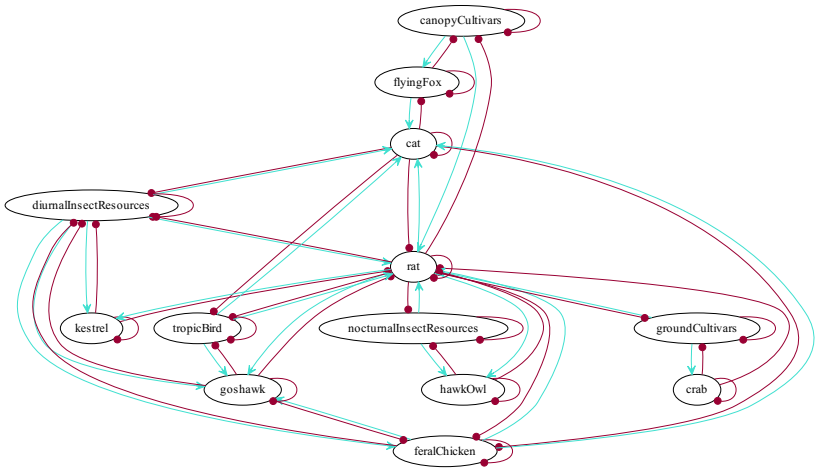
```
spp_list = c(
  'cat',
  'rat',
  'crab',
  'goshawk',
  'hawkOwl',
  'tropicBird',
  'flyingFox',
  'feralChicken',
  'kestrel',
```

```
  'groundCultivars',
  'canopyCultivars',
  'diurnalInsectResources',
  'nocturnalInsectResources'
)

positive_edges_list = list(
    'cat'= c('tropicBird', 'flyingFox', 'diurnalInsectResources', 'feralChicken', 'rat'),
    'rat'= c('tropicBird', 'groundCultivars', 'canopyCultivars', 'diurnalInsectResources',
             'nocturnalInsectResources', 'feralChicken'),
    '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', 'kestrel'),
    '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.
)
```

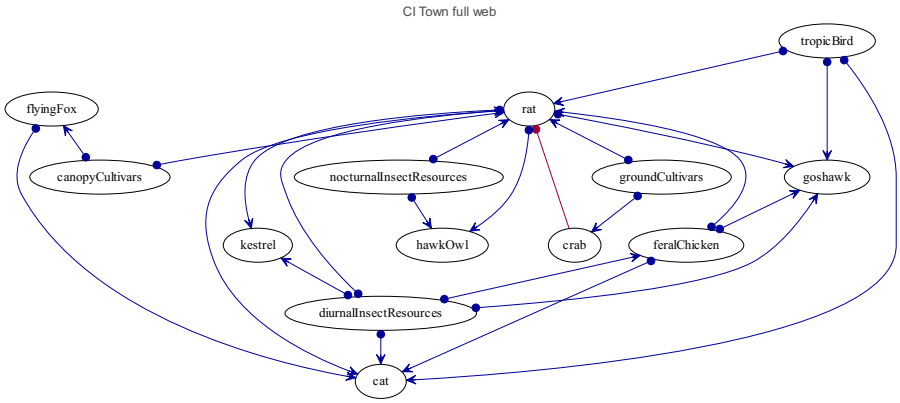
Initialise the full Town network.

```
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.

```
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])
```

C3.1.2 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[, "labelfrom"]]))
        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 12.65218 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.character)

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 rat_hawkOwl
## 1         pos         neg            pos           pos         neg         neg
## 2         pos         neg            pos           pos         pos         neg
## 3         pos         neg            pos           pos         pos         neg
##   rat_tropicBird rat_flyingFox
## 1            neg           pos
## 2            pos           pos
## 3            neg           neg
```

C3.1.3 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
```

C3.2 Implementing the Boolean approach on the full Town network of Christmas Island

We use the full Town network as an example to illustrate the Boolean approach.

Note: the Package *LogicOpt* was removed from the CRAN repository. An archived version can be downloaded from https://cran.r-project.org/src/contrib/Archive/LogicOpt/ and then installed froma local path. e.g. Using the code:

install.packages("C:/path to folder with the package", repos = NULL, type = "source"))

```
library(LogicOpt)
library(devtools)
devtools::source_url("https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/qualmod
devtools::source_url("https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/findPCU
#url:https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/qualmod.R?raw=TRUE
#https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/findPCU.R?raw=TRUE
```

C3.2.1 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()*.

```
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',
             'nocturnalInsectResources', 'feralChicken'),
    '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', 'kestrel'),
    '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')
```

Initialise the full Town network.

```
full_web <- initialise_foodweb(positive_edges_list, negative_edges_list)
```

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* is a vector of species being monitored in the Boolean approach.
- *monitored_spp_list_idx* represents indices of species in *monitored_spp_list*.

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

monitored_spp_list <- spp_list[!spp_list %in% unmonitored_spp_list]
monitored_spp_list_idx <- unname(labelToIndex[monitored_spp_list])
```

C3.2.2 The Boolean approach: parameter sweep on the full web

Storing and comparing string vectors of species-response combinations ('pos' and 'neg') can be extremly time comsuming in large while loop in R. Thus, we store the species-reponses calculated from each matrice as 1s and 0s corresponding to 'pos' and 'neg',and then concatenate this string of 1s and 0s as a binary digit. Lastly, the binary digit is converted as an integer and stored in a vector *collectedRespInt*.

Here for the parameter sweep, we set the maximum number of runs to 1,000,000 for demonstration, which takes about 3mins. In reality, we run 100,000,000 times, which need ~6 hours. Therefore, we use dataframe saved in advance to illustrate how to find implications rules in *C3.2.3 Boolean analysis and draw implication networks* and *C3.3 Boolean analysis and implication networks for combined cat and rat management*

```
set.seed(178)

search_terminator = 0.5
t_max = 1e06 # Here we use 1e6 for illustration, which takes about 3mins.
# t_max = 1e8 # ~ 6hours to run

k = 1
t = 0
t_last_updated = 0

collectedResponses = list()
```

```
collectedRespInt = NULL

sz <- dim(Mq)
n <- length(Mq)

start_time <- Sys.time()

while ((((t-t_last_updated) < search_terminator*t) | (t < 1000)) & (t < t_max)){

    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 <- ifelse(-Sq[spp_list_idx, control_list_idx] < 0, 0, 1)

    respInt <- strtoi(str_c(response, collapse = ""), base = 2L)

    existComb <- is.element(respInt, collectedRespInt)

    if (!existComb) {

        collectedRespInt[k] <- respInt

        k = k + 1
        t_last_updated = t
    }

    t = t + 1
}

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

```
## Time difference of 2.54255 mins
```

```
t_last_updated
```

```
## [1] 994945
```

```
t
```

```
## [1] 1e+06
```

```
length(collectedRespInt)
```

```
## [1] 7395
```

The integers respresenting possible species-response combinations are now stored in *collectedRespInt*. Now convert the *collectedRespInt* into a dataframe, where species responses are denoted as 1 (positive response) and 0 (negative response). Write the model responses to a csv file for later use.

```
colnames <- unlist(lapply(control_list, function(x) paste0(x, "_", spp_list)))

observedResponse <-
    append((2**length(colnames))-1, collectedRespInt) %>%
    intToBin(.) %>%   # convert the integer back into binary strings
    str_split(., "") %>%  # chop strings to 0s and 1s
    do.call(rbind, .) %>%
    as.data.frame() %>%
    setNames(., colnames) %>%
    slice(., -1)

write.csv(observedResponse, file = "town_search_full_10e6.csv", row.names=FALSE)
```

C3.2.3 Boolean analysis and draw implication networks

Read in the csv file of observed species-response combinations that we saved in advance for the full run of the parameter-sweep (number of matrices produced: 10^8).Data is available at: https://github.com/yhan178/qualitative-modeling-r/blob/master/case_study_CI_town_web/

```
observedResponse_df <- read.csv("town_search_full_10e8.csv", head = TRUE)
nrow(observedResponse_df)
```

```
## [1] 8930
```

```
# head(observedResponse_df) # Check the dataframe
```

C3.2.3a Cat management only

First, performing the Boolean analysis for cat management only.

We write a list of desired responses for this management. The *desiredResponses_c* list is then converted into a Boolean mask *desiredResponsesMask_c*, and passed to the function *getUnobservedInts2*.

Function *getUnobservedInts2* finds the complement of the set of observed responses (i.e. unobserved response combinations).The function first concatenate each observed response combination into a binary string and then convert this binary string into an integer. Then the complement corresponds to the list of integers that are missing from the full set of integers.

```
allResponse <- colnames(observedResponse_df)
# allResponse # check species response

str4true = 'pos'
```

```
str4flase = 'neg'

desiredResponses_c = c(
  'cat_tropicBird',
  'cat_goshawk',
  'cat_feralChicken',
  'cat_hawkOwl',
  'cat_rat',
  'cat_flyingFox')

boolLen_c <- length(desiredResponses_c)

desiredResponsesMask_c <- match(desiredResponses_c, allResponse)

unobservedInts_c <- getUnobservedInts2(observedResponse_df, desiredResponsesMask_c, boolLen_c)
length(unobservedInts_c)
```

```
## [1] 22
```

The function *getUnobservedBooldf* turns the list of integers *unobservedInts_c*, corresponding to unobserved species responses, back into a dataframe in Boolean expression (1s and 0s). The function *getUnobservedBooldf* also add an new column *unob* which is a vector of 1s (the dataframe therefore becomes a truth table) to allow funciton *logicopt()* in Package *LogicOpt* to perform Boolean minimization on the dataframe.

```
unobservedBooldf_c <- getUnobservedBooldf(unobservedInts_c, desiredResponses_c)
```

The function*logicopt()* in Package *LogicOpt* is used to perform the Boolean minimisation.

```
start_time <- Sys.time()

opt_c <- logicopt(unobservedBooldf_c, boolLen_c, 1, mode="espresso") # optimize the truth table
optEqn_c <- tt2eqn(opt_c[[1]], boolLen_c, 1) # show the optimized equations

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

```
## Time difference of 0.01600814 secs
```

The function *getPCUList* converts the optimized equations (*optEqn_c*) into a list of strings (PCUs) for further analysis.

```
optEqn_c # check the optimized equations
```

```
## [1] "unob = cat_goshawk*cat_feralchicken*cat_rat + cat_tropicbird*cat_goshawk*cat_rat + cat_rat*cat_
```

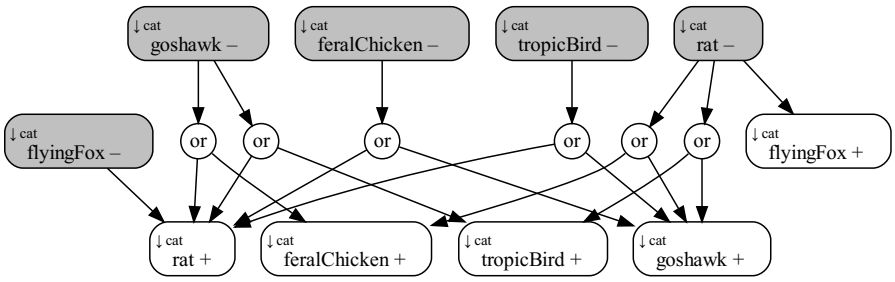
```
PCUList_c <- getPCUList(optEqn_c, str4true, str4flase, desiredResponses_c)
PCUList_c
```

```
## [[1]]
## [1] "negcat_rat"       "negcat_flyingFox"
##
## [[2]]
## [1] "negcat_goshawk"      "negcat_feralChicken" "negcat_rat"
##
## [[3]]
## [1] "negcat_tropicBird" "negcat_goshawk"    "negcat_rat"
```

PCULists are used as inputs to the functions *get_edgelist_singleAnte* and *get_edgelist_certainAnte*, which get an edgelist in a single-antecedent form or a certain-antecedent form. Then the edgelist will be passed to *draw_implication_network* function to plot the corresponding implication network.

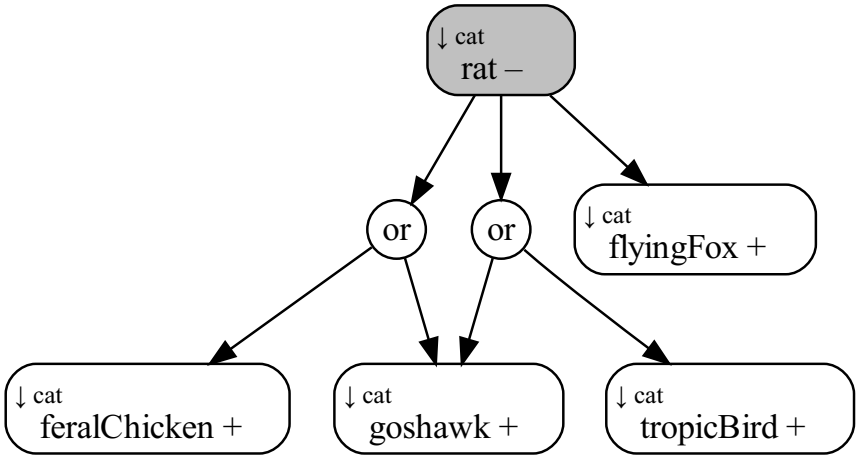
In the following attempt, we can see that species-responses are contingent upon the response of rat population to cat management.

```
edgelist_c1 <- get_edgelist_singleAnte(PCUList_c)
draw_implication_network(edgelist_c1)
```



Above network can be simplified as follows. It can be seen that if cat management has a negative effect on the rat population, certain positive outcomes are guaranteed; while if cat management has a positive effect on the rat population, we found no shorter implication rules.

```
alwaysAnteList_c1 = c('negcat_rat')
edgelist_c2 <- get_edgelist_certainAnte(PCUList_c, alwaysAnteList_c1)
draw_implication_network(edgelist_c2)
```



C3.2.3b Rat management only

Similar to above procedure, performing the Boolean analysis for rat management only. Write a list of desired responses for this management.Get the PCUList.

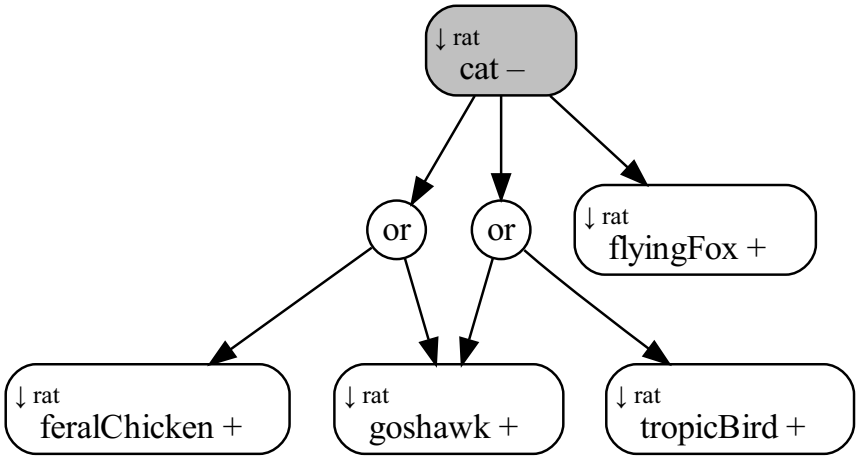
```
desiredResponses_r = c(
  'rat_tropicBird',
  'rat_goshawk',
  'rat_feralChicken',
  'rat_hawkOwl',
  'rat_cat',
  'rat_flyingFox')

boolLen_r <- length(desiredResponses_r)
desiredResponsesMask_r <- match(desiredResponses_r, allResponse)

unobservedInts_r <- getUnobservedInts2(observedResponse_df, desiredResponsesMask_r, boolLen_r)

unobservedBooldf_r <- getUnobservedBooldf(unobservedInts_r, desiredResponses_r)

opt_r <- logicopt(unobservedBooldf_r, boolLen_r, 1, mode="espresso")
optEqn_r <- tt2eqn(opt_r[[1]], boolLen_r, 1)

PCUList_r <- getPCUList(optEqn_r, str4true, str4flase, desiredResponses_r)
PCUList_r
```

```
## [[1]]
## [1] "negrat_cat"       "negrat_flyingFox"
##
## [[2]]
## [1] "negrat_goshawk"      "negrat_feralChicken" "negrat_cat"
```

```
##
## [[3]]
## [1] "negrat_tropicBird" "negrat_goshawk"    "negrat_cat"
```

We find that species-responses to rat management are contingent upon the response of cat population to rat management. If rat management has a negative effect on the cat population, certain positive outcomes are guaranteed; while if rat management has a positive effect on the rat population, we found no shorter implication rules.

```
alwaysAnteList_r1 = c('negrat_cat')
edgelist_r1 <- get_edgelist_certainAnte(PCUList_r, alwaysAnteList_r1)
draw_implication_network(edgelist_r1)
```



C3.2.3c Combined cat and rat management

Similar to above procedure, performing the Boolean analysis for combined cat and rat management. Write a list of desired responses for this management. Get the PCUList.

```
allResponse <- colnames(observedResponse_df)
# allResponse # check species response

str4true = 'pos'
str4flase = 'neg'

desiredResponses_cr = c(
  'cat_tropicBird',
  'cat_goshawk',
  'cat_feralChicken',
  'cat_hawkOwl',
  'cat_rat',
  'cat_flyingFox',
  'rat_tropicBird',
```

```
  'rat_goshawk',
  'rat_feralChicken',
  'rat_hawkOwl',
  'rat_cat',
  'rat_flyingFox')

boolLen_cr <- length(desiredResponses_cr)
desiredResponsesMask_cr <- match(desiredResponses_cr, allResponse)

unobservedInts_cr <- getUnobservedInts2(observedResponse_df, desiredResponsesMask_cr, boolLen_cr)

unobservedBooldf_cr <- getUnobservedBooldf(unobservedInts_cr, desiredResponses_cr)

opt_cr <- logicopt(unobservedBooldf_cr, boolLen_cr, 1, mode="espresso")
optEqn_cr <- tt2eqn(opt_cr[[1]], boolLen_cr, 1)

PCUList_cr <- getPCUList(optEqn_cr, str4true, str4flase, desiredResponses_cr)
# PCUList_cr  # check the PCUList
```

The generated PCUList is rather long and implication rules for the combined cat and rat management can be complicated. Thus we provide the another section for truncating and simplifying implication networks for the combined management. See section *C3.3 Boolean analysis and implication networks for combined cat and rat management*.

C3.3 Boolean analysis and implication networks for combined cat and rat management

Boolean analysis for the combined cat and rat management for the full Town Network.

```
library(LogicOpt)
library(devtools)
devtools::source_url("https://github.com/yhan178/qualitative-modeling-r/blob/master/qualmodfunc/findPCU
```

We continue from where we left in *C3.2.3c Combined cat and rat management*.

Read in the csv file of the observed species-response combinations that we saved in advance for the full run of the parameter-sweep (number of matrices produced: 10^8). Data is available at: https://github.com/yhan178/qualitative-modeling-r/blob/master/case_study_CI_town_web/

```
observedResponse_df <- read.csv("town_search_full_10e8.csv", header = TRUE)
# head(observedResponse_df)
```

C3.3.1 Perform Boolean minimization and get the PCUList

Performing the Boolean minimization on species responses to the combined management. Write a list of desired responses for this management. This section is identical to *C.3.2.3c Combined cat and rat management*.

```
allResponse <- colnames(observedResponse_df)
# allResponse # check species response

str4true = 'pos'
str4flase = 'neg'

desiredResponses_cr = c(
  'cat_tropicBird',
```

```
  'cat_goshawk',
  'cat_feralChicken',
  'cat_hawkOwl',
  'cat_rat',
  'cat_flyingFox',
  'rat_tropicBird',
  'rat_goshawk',
  'rat_feralChicken',
  'rat_hawkOwl',
  'rat_cat',
  'rat_flyingFox')

boolLen_cr <- length(desiredResponses_cr)
desiredResponsesMask_cr <- match(desiredResponses_cr, allResponse)

unobservedInts_cr <- getUnobservedInts2(observedResponse_df, desiredResponsesMask_cr, boolLen_cr)

unobservedBooldf_cr <- getUnobservedBooldf(unobservedInts_cr, desiredResponses_cr)

opt_cr <- logicopt(unobservedBooldf_cr, boolLen_cr, 1, mode="espresso")
optEqn_cr <- tt2eqn(opt_cr[[1]], boolLen_cr, 1)

PCUList_cr <- getPCUList(optEqn_cr, str4true, str4flase, desiredResponses_cr)
```

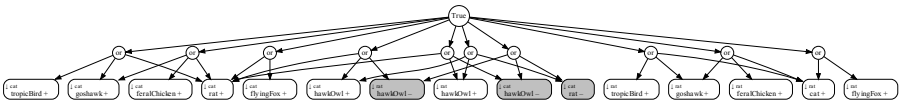
C3.3.2 Split the PCUList by length

Since the PCUList is quite large and complicated, we split the PCUList up by length.

```
PCUlength_cr <- sapply(PCUList_cr, function(i) length(i))
PCUList_cr1 <- PCUList_cr[PCUlength_cr <= 5]
PCUList_cr2 <- PCUList_cr[PCUlength_cr > 5]
```

Draw an implication network for all rules with length less than 5.

```
edgelist_cr1 <- get_edgelist_certainAnte(PCUList_cr1, c())
draw_implication_network(edgelist_cr1)
```



Above implication network shows that 'poscat_rat' 'negcat_rat' and 'posrat_cat' are central contingencies. Thus we analyze the implication rules under different scenarios based on the rat response to cat management and cat response to rat management.

C3.3.3 Implication netwoks under different scenarios

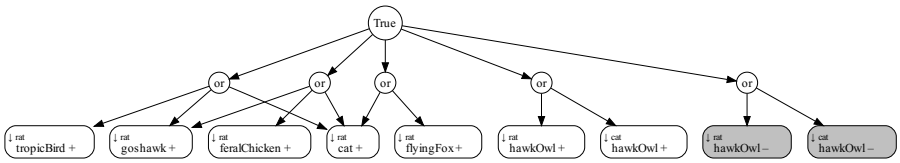
First, we split up the PCUs by the effects (positive/negtaive) of cat management on rat population. The always true conditions (here are "when cat management has a positive effect on rat population" and "when cat management has a negative effect on rat population" for PCULList_cr3 and PCUList_cr4 respectively) will be shown as "True" for a succinct presentation. There are rules only involving species-responses to cat management, which have been predicated by the cat only management scenario (see the document *Christmas_Island_Boolean_approach*). Thus, we can perform a further pruning, and these rules are predicated on "True".

```
PCUList_cr3 = list()
PCUList_cr4 = list()

for (PCU in PCUList_cr1){
    if (!('negcat_rat' %in% PCU)){
        if(!all(str_detect(PCU, "cat"))) {

            PCU <- str_subset(PCU, 'cat_rat', negate = TRUE)
            PCUList_cr3[[length(PCUList_cr3)+1]] <- PCU
        }
    }
}

for (PCU in PCUList_cr1){
    if (!('poscat_rat' %in% PCU)){
        if(!all(str_detect(PCU, "cat_"))) {
```

```
            PCU <- str_subset(PCU, 'cat_rat', negate = TRUE)
            PCUList_cr4[[length(PCUList_cr4)+1]] <- PCU
        }
    }
}
```

Scenario 1: when cat management has a positive effect on rat population.

```
edgelist_cr3 <- get_edgelist_certainAnte(PCUList_cr3, c())
draw_implication_network(edgelist_cr3)
```



From above implication network, we could further explore the implication rules under two scenarios given the effect of rat managemnet on cat population.

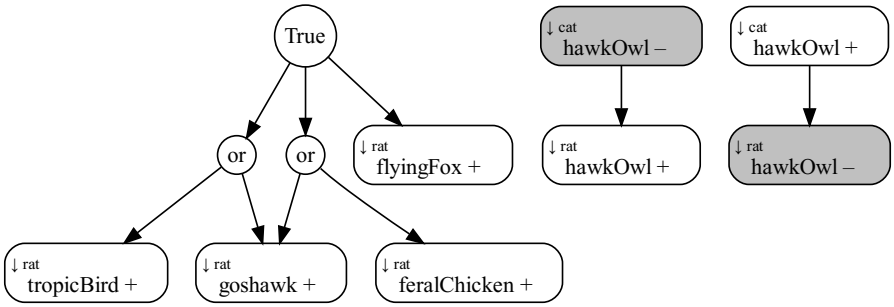
Scenario 1a: when cat management has a positive effect on rat population and rat management has a negative effect on cat population (this is the most likely scenario).

```
PCUList_cr5 = list()

for (PCU in PCUList_cr3){
    if (!('posrat_cat' %in% PCU)){
        PCU <- str_subset(PCU, 'rat_cat', negate = TRUE)
        PCUList_cr5[[length(PCUList_cr5)+1]] <- PCU
    }
}
```

The always true conditions (here are "when cat management has a positive effect on rat population and rat management has a negative effect on cat population") are shown as "True" for a succinct presentation.

In addtion, there is a better way to present the network given that the effect of rat management upon the hawk-owl was contingent upon the effect of cat management upon the hawk-owl.

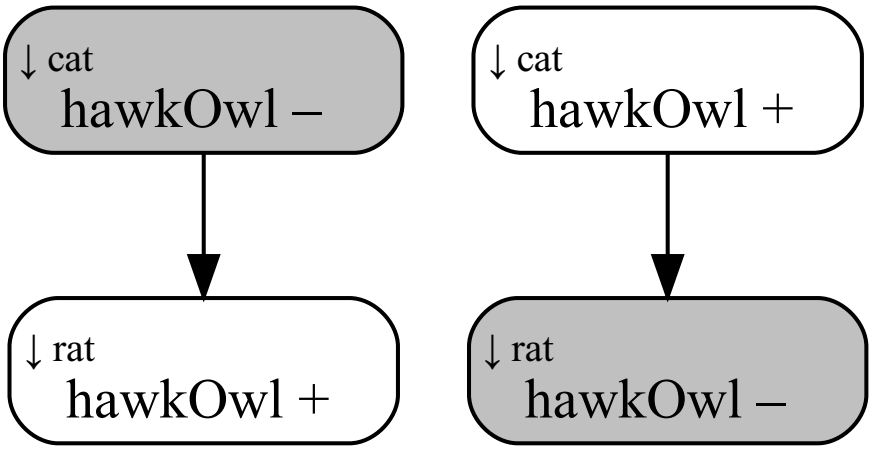
```
edgelist_cr5 <- get_edgelist_certainAnte(PCUList_cr5, c('poscat_hawkOwl', 'negcat_hawkOwl'))
draw_implication_network(edgelist_cr5)
```



Scenario 1b: when cat management has a positive effect on rat population and rat management has a positive effect on cat population (this is a less likely scenario).

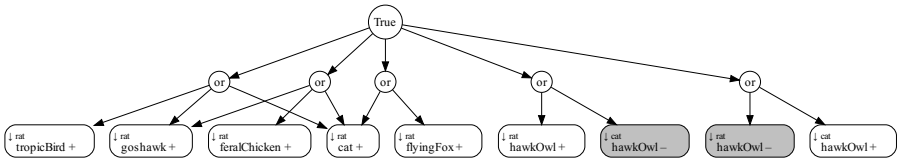
```
PCUList_cr6 = list()
for (PCU in PCUList_cr3){
    if (!('negrat_cat' %in% PCU)){
        PCU <- str_subset(PCU, 'rat_cat', negate = TRUE)
        PCUList_cr6[[length(PCUList_cr6)+1]] <- PCU
    }
}
```

```
edgelist_cr6 <- get_edgelist_certainAnte(PCUList_cr6, c('poscat_hawkOwl', 'negcat_hawkOwl'))
draw_implication_network(edgelist_cr6)
```



Scenario 2: when cat management has a negative effect on rat population.

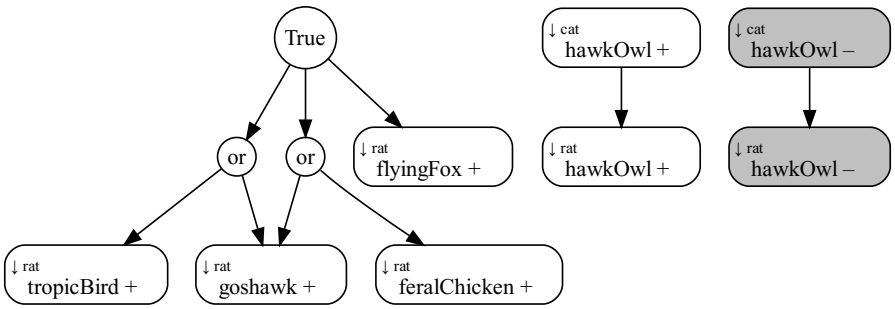
```
edgelist_cr4 <- get_edgelist_certainAnte(PCUList_cr4, c())
draw_implication_network(edgelist_cr4)
```



Scenario 2a: When cat management has a negative effect on rat population and rat management has a negative effect on cat population.

```
PCUList_cr7 = list()
for (PCU in PCUList_cr4){
    if (!('posrat_cat' %in% PCU)){
        PCU <- str_subset(PCU, 'rat_cat', negate = TRUE)
        PCUList_cr7[[length(PCUList_cr7)+1]] <- PCU
    }
}
```

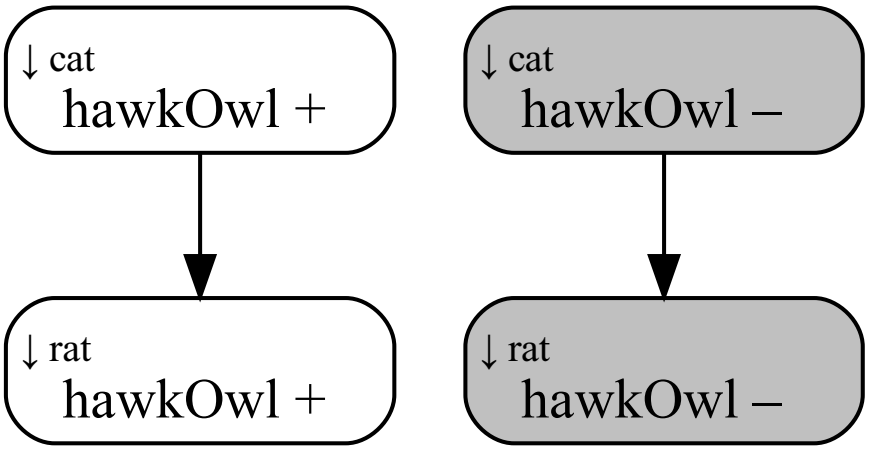
```
edgelist_cr7a <- get_edgelist_certainAnte(PCUList_cr7, c('poscat_hawkOwl', 'negcat_hawkOwl'))
draw_implication_network(edgelist_cr7a)
```



Scenario 2b: when cat management has a negative effect on rat population and rat management has a positive effect on cat population.

```
PCUList_cr8 = list()
for (PCU in PCUList_cr4){
    if (!('negrat_cat' %in% PCU)){
        PCU <- str_subset(PCU, 'rat_cat', negate = TRUE)
        PCUList_cr8[[length(PCUList_cr8)+1]] <- PCU
    }
}
```

```
edgelist_cr8a <- get_edgelist_certainAnte(PCUList_cr8, c('poscat_hawkOwl', 'negcat_hawkOwl'))
draw_implication_network(edgelist_cr8a)
```


↓ cat
hawkOwl +
↓ rat
hawkOwl +
↓ cat
hawkOwl –
↓ rat
hawkOwl –