Import and explore response variables

Aidan Brushett

2025-05-02

Contents

Before you begin

This script is number 4 of 6 in a series of scripts used to replicate the analyses presented in the paper: "Life on the edge: Industrial footprint and edge effects variably affect the distribution of a boreal small mammal"

This script was used to create and explore potential response variables describing the spatial distribution of red squirrels in our study area. This includes all steps, including importing raw Timelapse 2.0 data, calculating independent detections, binning data into a variety of response variables, and summary statistics (naive occupancy, camera-trap days, etc.)

When running these scripts, please ensure that you have downloaded the complete GitHub repository. This will ensure you have all the files, data, and proper folder structure you will need to run this code and associated analyses.

Also make sure you open RStudio through the R project (OSM_red_squirrel_distribution.Rproj). This will automatically set your working directory to the correct place (wherever you saved the repository) and ensure you don't have to change the file paths for some of the data. This analysis was initially run in R v4.3.0. If you have any questions or concerns, please contact one of the authors (in order):

Aidan Brushett M.Sc. Student University of Victoria
School of Environmental Studies
Email: aidanbrushett@uvic.ca

Emerald Arthurs M.Sc. Student University of Victoria
School of Environmental Studies

---

0. Setup

```r
rm(list=ls()) # start fresh :)

library(tidyverse)
library(sf)
library(ggplot2)
library(rphylopic)

# These are functions that can sometimes get masked by other packages. Shouldn't be an issue and
↪  could be removed.
select <- dplyr::select
filter <- dplyr::filter
mutate <- dplyr::mutate
summarize <- dplyr::summarize
# Cheeky function that means "not in"
`%nin%` = Negate(`%in%`)
```

1. Response variables

1.0. Import raw camera data and survey effort data

First the raw Timelapse2 data, which is stored in separate files from each year. These files came from the OSM_2023_2024 repositories from the ACME lab.

```r
osm_image <- list.files(path="./data/raw/",
                               pattern="OSM_timelapse",
                               full.names = TRUE) %>%

      # Import the timelapse data and bind them together using purrr
      purrr::map_dfr(.,
          ~read_csv(.x,

                   # Old code to specify column types, don't actually need it
                   col_types = cols(datetime = col_character(), # we will fix this later
                                    site = col_character(),
                                    total = col_integer(),
                                    group_count = col_integer(),
                                    comments = col_character(),
                                    species = col_factor(),
                                    otherspecify = col_character(),
                                    snow = col_factor(),
                                    empty = col_logical())
          ) %>%

            set_names(tolower(names(.))) %>%

            select(-array, -camera) %>%

            # Remove "Z" and "T" to brute force the timestamps where needed
            mutate(datetime = datetime %>% str_replace_all("Z", "") %>% str_replace_all("T", " "),
                   datetime = as_datetime(datetime),
                   datasource = paste0(.x),
                   year = year(datetime),
                   month = month(datetime),
                   day = day(datetime)) %>%

            mutate(
              site = gsub("-", "_", site) %>% # First, replace the hyphen with underscore
                gsub("_0+", "_", .) %>% # Then, remove leading zeros after the underscore
                gsub(" ", "", .) %>% # Then, remove any spaces
                gsub("LU0", "LU", .) %>% # Remove leading zeros from LU names for consistency
                as.factor(.), # Then, convert back to factor
               ) %>%

            # Create some extra columns for array and camera with the cleaned site names
            separate_wider_delim(site,
                                 delim = '_',
                                 names = c('array', 'camera'),
                                 cols_remove = FALSE) %>%

            # specify format of new columns
            mutate(
              array = as.factor(array),
              camera = as.factor(camera)
              ) %>%

            select(file, relativepath, array, camera, site, snow, datetime,
                   year, month, day, classifier, species, total, group_count, comments,
                   empty, datasource, cameramalfunction)
          )
```

```
## Warning: One or more parsing issues, call `problems()` on your data frame for details,
## e.g.:
##   dat <- vroom(...)
##   problems(dat)
```

```r
# Total images. This is all we really needed from the code above.
osm_image %>% nrow(.)
```

```
## [1] 683259
```

```r
# Images of a squirrel
osm_image %>%

  filter(species == "Red squirrel",
         empty == FALSE) %>%
  nrow(.)
```

```
## [1] 7399
```

Now the camera operability data, created in the OSM_2023_2024 repository by Aidan Brushett and Marissa Dyck.

```r
# List of days the cameras were running (excludes snow)
operating <- read_csv("./data/raw/OSM_operating_2021_2022_2023.csv") %>%

  mutate(year = year(date),
         month = month(date)) %>%

  select(-camera)
```

```
## Rows: 149274 Columns: 4
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): array, site
## dbl  (1): camera
## date (1): date
##
## 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.
```

1.1. Calculate independent detections

```r
# set the independent detection threshold to 30 minutes
mins <- 30

# prep the data for calculating independent detections
indet <- osm_image %>%

  # remove rows with no species info
  drop_na(species) %>%

  # We want to avoid using empty group images where no animal is actually present. This could
  ↪ falsely alter independent detections.
  filter(empty == FALSE) %>%

  # select only variables of interest
  select(array,
         site,
         species,
         datetime,
         month,
         year) %>%

  # now we need to create a new variable called timediff
  # first make sure data are arrange in proper order
  arrange(site, species, datetime) %>% # this will NOT work if not in correct order (early-late
  ↪  date)

  # create groups for each species at each site
  group_by(species, site) %>%

  # create new variable timediff that will calculate the difference
  mutate(timediff = as.numeric(difftime(datetime,lag(datetime),
                                      units = "mins")),
        # Then, create an event_ID that groups detections and identifies when photos are >30min
        ↪  apart.
        event_id = cumsum(if_else(is.na(timediff) | timediff > mins,  # Create episode IDs to
        ↪  distinguish detections
                                1,
                                0))
        )

# now create a new data frame with a single row for each event.
# The timestamp corresponds to the EARLIEST image in the event. Other code could be added
indet <- indet %>%

  group_by(array, species, site, event_id, year, month) %>%

  summarize(event_start = min(datetime)) %>%

  mutate(species = tolower(species))
```

```
## `summarise()` has grouped output by 'array', 'species', 'site', 'event_id',
## 'year'. You can override using the `.groups` argument.
```

How many independent detections of squirrels are there?

```
# Total independent detections
n_indet <- indet %>%

  group_by(species) %>%

  summarize(detections = n())

n_indet %>% filter(species == "red squirrel")
```

```
## # A tibble: 1 x 2
##   species      detections
##   <chr>             <int>
## 1 red squirrel       4747
```

1.2. Monthly independent detections and presence-absence

For plotting and formatting proportional monthly detections we need to create a subset of the species in the detections data to just include several focal species we are interested in

```
# create a list of focal species for filtering the data/plots
 focal_species <- c('black bear',
                    'caribou',
                    'cougar',
                    'coyote',
                    'fisher',
                    'grey wolf',
                    'lynx',
                    'moose',
                    'red fox',
                    'white-tailed deer',
                    'wolverine',
                    'snowshoe hare',
                    'red squirrel'
                    )
```

Bin the operating data monthly and retain months with >15 days of data.

```
operating_months <- operating %>%

  group_by(array, site, month, year) %>%

  summarize(n_days = n()) %>%

  filter(n_days >= 15) # retain months where there are 15 days of data only.
```

```
## `summarise()` has grouped output by 'array', 'site', 'month'. You can override
## using the `.groups` argument.
```

Then, let's bin the independent detections data into months. This data frame will only contain the positive detections. i.e., if a species was undetected for a given site/month, there will be no row.

```
indet_months <- indet %>%

  filter(species %in% focal_species) %>%

  group_by(array, site, species, month, year) %>%

  # Mark a 1 for presence and count the number of rows (i.e. # of detections)
  summarize(presence = 1,
            detections = n())
```

```
## `summarise()` has grouped output by 'array', 'site', 'species', 'month'. You
## can override using the `.groups` argument.
```

Then, let's merge this to the operating data using left_join. This will ensure that our 'absence' cases now show up in the dataset.

```r
presence_months <- operating_months %>%

  # Ensure that we have a new row for each species x month combination.
  crossing(species = focal_species) %>%

  left_join(indet_months, by = c("array", "site", "species", "month", "year")) %>%

  # Fill missing presence and detections with 0
  mutate(presence = replace_na(presence, 0),
         detections = replace_na(detections, 0),
         species = as.factor(species)) %>%

  select(-n_days)

summary(presence_months)
```

```
##     array               site               month             year     
##  Length:63934       Length:63934       Min.   : 1.000   Min.   :2021  
##  Class :character   Class :character   1st Qu.: 4.000   1st Qu.:2022  
##  Mode  :character   Mode  :character   Median : 7.000   Median :2023  
##                                        Mean   : 6.634   Mean   :2023  
##                                        3rd Qu.:10.000   3rd Qu.:2024  
##                                        Max.   :12.000   Max.   :2024  
##                                                                       
##        species         presence        detections     
##  black bear: 4918   Min.   :0.0000   Min.   : 0.0000  
##  caribou   : 4918   1st Qu.:0.0000   1st Qu.: 0.0000  
##  cougar    : 4918   Median :0.0000   Median : 0.0000  
##  coyote    : 4918   Mean   :0.1533   Mean   : 0.4597  
##  fisher    : 4918   3rd Qu.:0.0000   3rd Qu.: 0.0000  
##  grey wolf : 4918   Max.   :1.0000   Max.   :70.0000  
##  (Other)   :34426
```

Save the file:

```r
write_csv(presence_months, "./data/processed/OSM_monthly_detections_2021_2022_2023.csv")
```

1.3. Monthly proportional presence-absence

We will now create a proportional monthly variable for each species by determining how many months an animal was present, divided by the total months of data available.

```r
proportional_months <- presence_months %>%

  group_by(array, site, species) %>%

  # Count the number of rows (i.e. months)
  # Create a row that sums the number of months present (proportional presence!)
  summarize(months_active = n(),
            months_present = sum(presence),
            .groups = 'drop')
```

Let’s do a couple quick checks to make sure this worked. First, a visual inspection to make sure we have the same number of rows for each species and a reasonable range of active dates. Then, we’ll check if any of the presence values exceed the active values, which would indicate an error.

```r
summary(proportional_months)
```

```
##     array               site                    species     months_active
##  Length:5590        Length:5590        black bear: 430   Min.   : 1.00
##  Class :character   Class :character   caribou   : 430   1st Qu.:11.00
##  Mode  :character   Mode  :character   cougar    : 430   Median :12.00
##                                        coyote    : 430   Mean   :11.44
##                                        fisher    : 430   3rd Qu.:12.00
##                                        grey wolf : 430   Max.   :15.00
##                                        (Other)   :3010
##  months_present
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 0.000
##  Mean   : 1.754
##  3rd Qu.: 2.000
##  Max.   :15.000
##
```

```r
proportional_months %>%
  filter(months_present > months_active)
```

```
## # A tibble: 0 x 5
## # i 5 variables: array <chr>, site <chr>, species <fct>, months_active <int>,
## #   months_present <dbl>
```

Now let’s pivot the data to wide format :) We will also rename the species since they’re now becoming column names and spaces or hyphens aren’t acceptable.

```r
proportional_months <- proportional_months %>%

  # pivot the data wider so there is a column for each species and 1 row per site
  pivot_wider(names_from = species,
              values_from = months_present) %>%

  # replace NAs with zeros in all species columns
  mutate(across(
    where(is.numeric),
    ~ replace_na(., 0))) %>%

  # ensure all species columns are numeric not integer
  mutate_if(is.integer,
            as.numeric) %>%

  # set the column names to lower case and replace the spaces with '_' (these are both personal
  ↪ preferences of mine)
  set_names(
    names(.) %>%
      tolower()%>%
      str_replace_all(pattern = ' ',
                    replacement = '_') %>%
      str_replace_all(pattern = '-',
                      replacement = "_"))
```

Now we can run the function below to create a second column for each species that will represent the number of absences (months the species was not detected) at each camera from the active months. This function was adopted from Marissa Dyck's OSM 2022-2023 image processing script, found on the ACME GitHub.

```
# replace all non-letter digits with underscores to match the column names in the response
↪  variable.
focal_species_colnames <- focal_species %>%
  str_replace_all(., pattern = ' ', replacement = '_') %>%
  str_replace_all(., pattern = '-', replacement = '_')

# first convert data to data frame not a tibble for function to work
proportional_months <- as.data.frame(proportional_months)

# create a vector of the species columns for the loop
# update our focal species list to match the new column names
for (species in focal_species_colnames) {

  if (is.numeric(proportional_months[,species]) & is.numeric(proportional_months$months_active)) {
    col_absence <- paste0("absent_", species)

    proportional_months[col_absence] <- proportional_months$months_active -
↪   proportional_months[,species]

  }
}

str(proportional_months)
```

```
## 'data.frame':    430 obs. of  29 variables:
##  $ array                  : chr  "LU1" "LU1" "LU1" "LU1" ...
##  $ site                   : chr  "LU1_10" "LU1_11" "LU1_13" "LU1_22" ...
##  $ months_active          : num  5 14 15 14 14 15 15 15 15 14 ...
##  $ black_bear             : num  3 4 7 8 9 4 5 7 7 8 ...
##  $ caribou                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cougar                 : num  0 1 0 1 0 0 0 0 0 0 ...
##  $ coyote                 : num  4 8 10 11 9 11 0 9 4 7 ...
##  $ fisher                 : num  3 3 3 2 1 1 1 0 3 3 ...
##  $ grey_wolf              : num  0 2 0 0 0 1 0 0 0 0 ...
##  $ lynx                   : num  0 1 0 1 1 0 0 0 2 1 ...
##  $ moose                  : num  2 5 9 1 0 2 4 1 0 3 ...
##  $ red_fox                : num  0 2 0 0 0 0 0 4 0 0 ...
##  $ red_squirrel           : num  0 7 0 10 1 15 0 9 3 2 ...
##  $ snowshoe_hare          : num  1 3 0 8 2 2 0 12 4 7 ...
##  $ white_tailed_deer      : num  5 12 12 13 14 15 9 12 10 10 ...
##  $ wolverine              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ absent_black_bear      : num  2 10 8 6 5 11 10 8 8 6 ...
##  $ absent_caribou         : num  5 14 15 14 14 15 15 15 15 14 ...
##  $ absent_cougar          : num  5 13 15 13 14 15 15 15 15 14 ...
##  $ absent_coyote          : num  1 6 5 3 5 4 15 6 11 7 ...
##  $ absent_fisher          : num  2 11 12 12 13 14 14 15 12 11 ...
##  $ absent_grey_wolf        : num  5 12 15 14 14 14 15 15 15 14 ...
##  $ absent_lynx             : num  5 13 15 13 13 15 15 15 13 13 ...
##  $ absent_moose            : num  3 9 6 13 14 13 11 14 15 11 ...
##  $ absent_red_fox          : num  5 12 15 14 14 15 15 11 15 14 ...
##  $ absent_white_tailed_deer: num  0 2 3 1 0 0 6 3 5 4 ...
##  $ absent_wolverine        : num  5 14 15 14 14 15 15 15 15 14 ...
##  $ absent_snowshoe_hare    : num  4 11 15 6 12 13 15 3 11 7 ...
##  $ absent_red_squirrel     : num  5 7 15 4 13 0 15 6 12 12 ...
```

Let's adjust the black bear data to accommodate hibernation. We want bear data to represent only April to November.

```
operating_months_bears <- operating_months %>%

  # Bear data from April to November
  filter(month %in% c("4", "5", "6", "7", "8", "9", "10", "11")) %>%

  group_by(array, site) %>%

  # Count number of present months
  summarize(months_active_bears = n(),
            .groups = 'drop')

# Update the months_active field for bears
proportional_months <- proportional_months %>%

  left_join(operating_months_bears, by = c('array', 'site')) %>%

  # overwrite absent black bear column
  mutate(absent_black_bear = months_active_bears - black_bear) %>%

  # get rid of unnecessary columns for active months
  select(-months_active_bears)
```

Save the file:

```
write_csv(proportional_months,
↪  "./data/processed/OSM_proportional_monthly_presence_2021_2022_2023.csv")
```

Clean up our workspace:

```
rm(operating_months_bears, operating_months, indet_months)
```

1.4. Weekly independent detections and presence-absence

Bin the operating data weekly and retain weeks with >4 days of data. There is a handy function to do this!! `floor_date()` takes a date-time object and rounds it down to the nearest boundary of the specified time unit, starting on Sunday (default case). In this step, we will round down our operating data to the nearest week so we can bin it. We'll re-extract the month and year fields later.

```
operating_weeks <- operating %>%

  # Floor_date bins data into weeks starting on a Sunday
  mutate(week = floor_date(date, unit = "week")) %>%

  group_by(array, site, week) %>%

  # Number of days of data in a given week
  summarize(n_days = n()) %>%

  filter(n_days >= 4) # retain weeks where there are 15 days of data only.
```

```
## `summarise()` has grouped output by 'array', 'site'. You can override using the
## `.groups` argument.
```

Then, let's bin the independent detections data into weeks. This data frame will only contain the positive detections. i.e., if a species was undetected for a given site/week, there will be no row.

```
indet_weeks <- indet %>%

  filter(species %in% focal_species) %>%

  # What week does the independent detection belong to?
  mutate(week = floor_date(event_start, unit = "week")) %>%

  group_by(array, site, week, species) %>%

  # Mark a 1 for presence and count the number of rows (i.e. # of detections)
  summarize(presence = 1,
            detections = n(),
            .groups = 'drop')
```

Then, let's merge this to the operating data using left_join. This will ensure that our 'absence' cases now show up in the dataset.

```
presence_weeks <- operating_weeks %>%

  # Ensure that we have a new row for each species x week combination.
  crossing(species = focal_species) %>%

  left_join(indet_weeks, by = c("array", "site", "species", "week")) %>%

  # Fill missing presence and detections with 0
  mutate(presence = replace_na(presence, 0),
         detections = replace_na(detections, 0),
         species = as.factor(species)) %>%

  mutate(month = month(week),
         year = year(week)) %>%

  select(-n_days)

summary(presence_weeks)
```

```
##     array                site                  week
##  Length:277017      Length:277017      Min.   :2021-07-11 00:00:00.00
##  Class :character   Class :character   1st Qu.:2022-11-20 00:00:00.00
##  Mode  :character   Mode  :character   Median :2023-08-06 00:00:00.00
##                                         Mean   :2023-06-17 05:29:46.56
##                                         3rd Qu.:2024-03-03 00:00:00.00
##                                         Max.   :2024-09-22 00:00:00.00
##
##        species          presence          detections          month
##  black bear: 21309   Min.   :0.00000   Min.   : 0.0000   Min.   : 1.000
##  caribou   : 21309   1st Qu.:0.00000   1st Qu.: 0.0000   1st Qu.: 4.000
##  cougar    : 21309   Median :0.00000   Median : 0.0000   Median : 7.000
##  coyote    : 21309   Mean   :0.06325   Mean   : 0.1084   Mean   : 6.642
##  fisher    : 21309   3rd Qu.:0.00000   3rd Qu.: 0.0000   3rd Qu.:10.000
##  grey wolf : 21309   Max.   :1.00000   Max.   :24.0000   Max.   :12.000
##  (Other)   :149163
##       year
##  Min.   :2021
##  1st Qu.:2022
##  Median :2023
##  Mean   :2023
##  3rd Qu.:2024
##  Max.   :2024
##
```

Save the file:

```
write_csv(presence_weeks, "./data/processed/OSM_weekly_detections_2021_2022_2023.csv")
```

1.5. Weekly proportional presence-absence

We will now create a proportional weekly variable for each species by determining how many weeks an animal was present, divided by the total weeks of data available.

```
proportional_weeks <- presence_weeks %>%

  group_by(array, site, species) %>%

  # Total and present weeks, respectively
  summarize(weeks_active = n(),
            weeks_present = sum(presence),
            .groups = 'drop')
```

Let's do a couple quick checks to make sure this worked. First, a visual inspection to make sure we have the same number of rows for each species and a reasonable range of active dates. Then, we'll check if any of the presence values exceed the active values, which would indicate an error.

```
summary(proportional_weeks)
```

```
##     array                site                 species      weeks_active
##  Length:5590        Length:5590        black bear: 430   Min.   : 4.00
##  Class :character   Class :character   caribou   : 430   1st Qu.:49.00
##  Mode  :character   Mode  :character   cougar    : 430   Median :51.00
##                                        coyote    : 430   Mean   :49.56
##                                        fisher    : 430   3rd Qu.:53.00
##                                          grey wolf : 430   Max.   :65.00
##                                          (Other)   :3010
##  weeks_present
##  Min.   : 0.000
##  1st Qu.: 0.000
##  Median : 0.000
##  Mean   : 3.135
##  3rd Qu.: 3.000
##  Max.   :58.000
##
```

```
proportional_weeks %>%
  filter(weeks_present > weeks_active)
```

```
## # A tibble: 0 x 5
## # i 5 variables: array <chr>, site <chr>, species <fct>, weeks_active <int>,
## #   weeks_present <dbl>
```

Now let's pivot the data to wide format :)

```
proportional_weeks <- proportional_weeks %>%

  # pivot the data wider so there is a column for each species and 1 row per site
  pivot_wider(names_from = species,
              values_from = weeks_present) %>%

  # replace NAs with zeros in all species columns
  mutate(across(
    where(is.numeric),
    ~ replace_na(., 0))) %>%

  # ensure all species columns are numeric not integer
  mutate_if(is.integer,
            as.numeric) %>%

  # set the column names to lower case and replace the spaces with '_' (these are both personal
  ↪  preferences of mine)
  set_names(
    names(.) %>%
      tolower()%>%
      str_replace_all(pattern = ' ',
                      replacement = '_') %>%
      str_replace_all(pattern = '-',
                      replacement = "_"))
```

Now we can run the function below to create a second column for each species that will represent the number of absences (weeks the species was not detected) at each camera from the active weeks. This function was adopted from Marissa Dyck's OSM 2022-2023 image processing script, found on the ACME GitHub.

```
# first convert data to data frame not a tibble for function to work
proportional_weeks <- as.data.frame(proportional_weeks)

# create a vector of the species columns for the loop
# update our focal species list to match the new column names
```

```
for (species in focal_species_colnames) {

  if (is.numeric(proportional_weeks[,species]) & is.numeric(proportional_weeks$weeks_active)) {
    col_absence <- paste0("absent_", species)

    proportional_weeks[col_absence] <- proportional_weeks$weeks_active -
 ↪  proportional_weeks[,species]

  }
}

str(proportional_weeks)
```

```
## 'data.frame':    430 obs. of  29 variables:
##  $ array                    : chr  "LU1" "LU1" "LU1" "LU1" ...
##  $ site                     : chr  "LU1_10" "LU1_11" "LU1_13" "LU1_22" ...
##  $ weeks_active             : num  24 64 65 64 64 65 65 64 64 62 ...
##  $ black_bear               : num  4 7 12 11 20 4 9 19 12 10 ...
##  $ caribou                  : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ cougar                   : num  0 1 0 1 0 0 0 0 0 0 ...
##  $ coyote                   : num  4 11 12 18 13 22 0 14 5 13 ...
##  $ fisher                   : num  6 4 3 4 1 1 1 0 3 4 ...
##  $ grey_wolf                : num  0 2 0 0 0 1 0 0 0 0 ...
##  $ lynx                     : num  0 1 0 1 1 0 0 0 2 1 ...
##  $ moose                    : num  2 7 14 1 0 2 6 2 0 3 ...
##  $ red_fox                  : num  0 2 0 0 0 0 0 7 0 0 ...
##  $ red_squirrel             : num  0 9 0 26 1 47 0 19 5 3 ...
##  $ snowshoe_hare            : num  1 6 0 13 2 3 0 31 8 13 ...
##  $ white_tailed_deer        : num  22 34 41 28 43 40 17 20 13 20 ...
##  $ wolverine                : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ absent_black_bear        : num  20 57 53 53 44 61 56 45 52 52 ...
##  $ absent_caribou           : num  24 64 65 64 64 65 65 64 64 62 ...
##  $ absent_cougar            : num  24 63 65 63 64 65 65 64 64 62 ...
##  $ absent_coyote            : num  20 53 53 46 51 43 65 50 59 49 ...
##  $ absent_fisher            : num  18 60 62 60 63 64 64 64 61 58 ...
##  $ absent_grey_wolf         : num  24 62 65 64 64 64 65 64 64 62 ...
##  $ absent_lynx              : num  24 63 65 63 63 65 65 64 62 61 ...
##  $ absent_moose             : num  22 57 51 63 64 63 59 62 64 59 ...
##  $ absent_red_fox           : num  24 62 65 64 64 65 65 57 64 62 ...
##  $ absent_white_tailed_deer: num  2 30 24 36 21 25 48 44 51 42 ...
##  $ absent_wolverine         : num  24 64 65 64 64 65 65 64 64 62 ...
##  $ absent_snowshoe_hare     : num  23 58 65 51 62 62 65 33 56 49 ...
##  $ absent_red_squirrel      : num  24 55 65 38 63 18 65 45 59 59 ...
```

Let’s adjust the black bear data to accommodate hibernation. We want bear data to represent only April to November.

```
operating_weeks_bears <- operating_weeks %>%

  mutate(month = month(week)) %>%

  filter(month %in% c(4:11)) %>%

  group_by(array, site) %>%
  summarize(weeks_active_bears = n(),
            .groups = 'drop')

# Update the weeks_active field for bears
proportional_weeks <- proportional_weeks %>%

  left_join(operating_weeks_bears, by = c('array', 'site')) %>%

  # overwrite absent black bear column
  mutate(absent_black_bear = weeks_active_bears - black_bear) %>%

  # get rid of unnecessary columns for active weeks
  select(-weeks_active_bears)
```

Save the file:

```
write_csv(proportional_weeks,
 ↪  "./data/processed/OSM_proportional_weekly_presence_2021_2022_2023.csv")
```

Clean up our workspace:

```
rm(operating_weeks_bears, operating_weeks, indet_weeks)
```

2. Visualize the response variables

Let's make a couple quick figures to represent our detections and get a sense for the amount of variation between sites and between arrays.

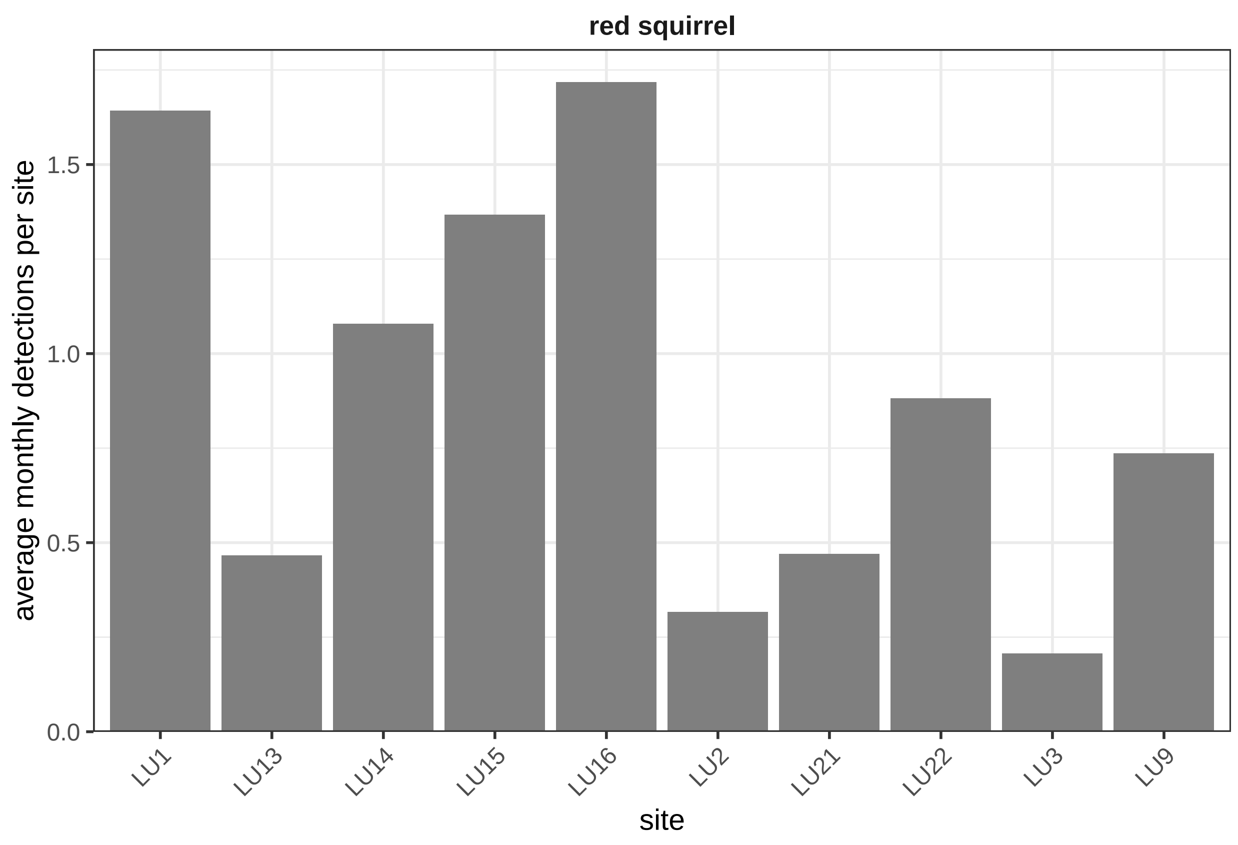
2.1. Monthly detections per site/array

This is a good proxy for overall density of a species across each landscape.

```
fig_month_det <- presence_months %>%

  filter(species %in% c('red squirrel')) %>%

  group_by(array, species) %>%

  # Total detections per array,
  # Adjusted by the total months of operation
  summarize(total_det = sum(detections),
            total_months = n(),
            average_det = total_det / total_months) %>%

  # Graph it!
  ggplot(., aes(x = array, y = average_det)) +

    # Bar plot side-by-side based on the number itself 'identity'
    geom_bar(stat = "identity", position = "dodge", fill = 'grey50') +

    facet_wrap(~species, scales = 'fixed') +
    labs(x = "site",
         y = "average monthly detections per site") +

    theme_bw() +

    theme(
         axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
         strip.background = element_blank(),  # Remove facet background
         strip.text = element_text(face = "bold", size = 10)
         ) +

    scale_y_continuous(expand = expansion(mult = c(0, 0.05)))  # Remove bottom space, keep small
    ↪  top space
```

```
## `summarise()` has grouped output by 'array'. You can override using the
## `.groups` argument.
```

```
fig_month_det
```



```
ggsave("./figures/red_squirrel_monthly_detection_rate.png", width = 5, height = 4)
```

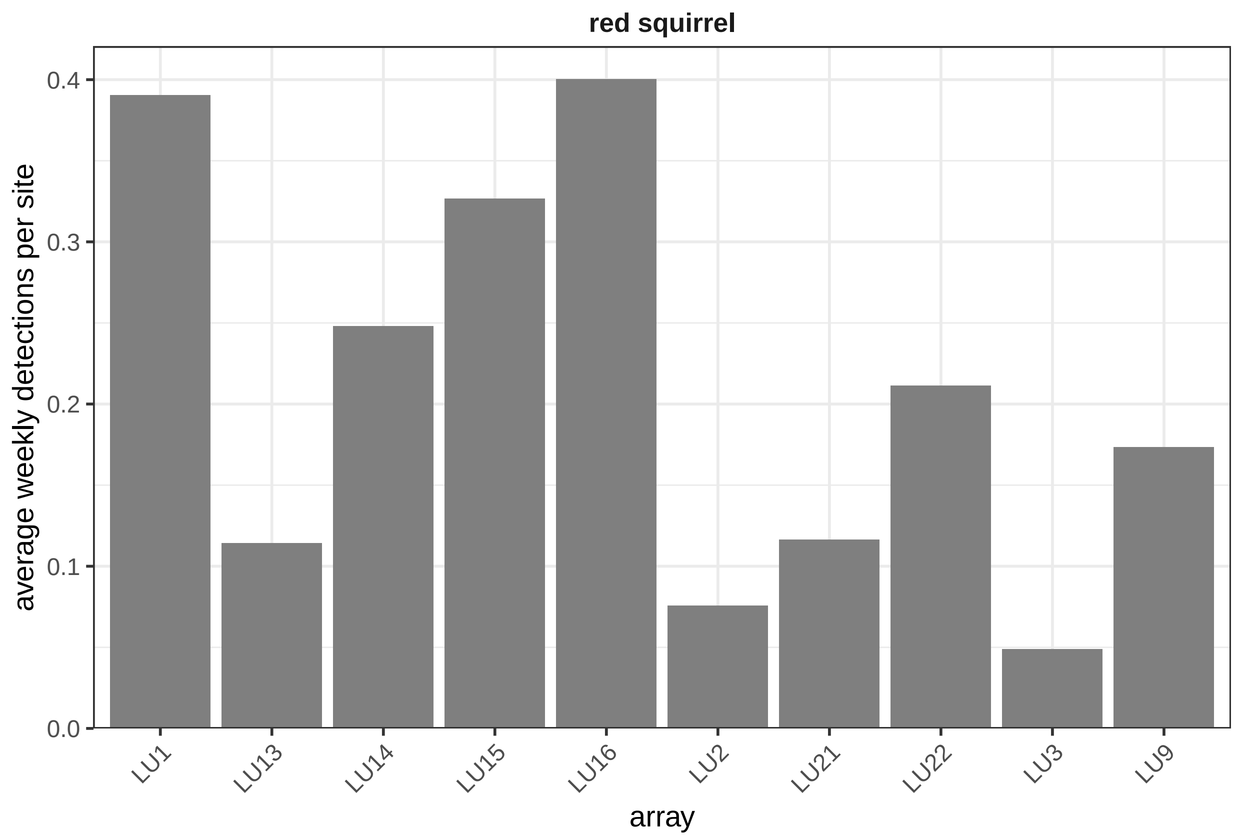
2.2. Weekly detections per site/array

This is a good proxy for overall density of a species across each landscape. Let's see if it varies much from our monthly metric.

```r
# Exact same code as above (4.1.) but with the weekly data
fig_week_det <- presence_weeks %>%

  filter(species %in% c('red squirrel')) %>%

  group_by(array, species) %>%

  summarize(total_det = sum(detections),
            total_weeks = n(),
            average_det = total_det / total_weeks) %>%

  ggplot(., aes(x = array, y = average_det)) +

    geom_bar(stat = "identity", position = "dodge", fill = 'grey50') +

    facet_wrap(~species, scales = 'fixed') +

    labs(x = "array",
         y = "average weekly detections per site") +

    theme_bw() +

    theme(
         axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
         strip.background = element_blank(),  # Remove facet background
         strip.text = element_text(face = "bold", size = 10)
         ) +

    scale_y_continuous(expand = expansion(mult = c(0, 0.05)))  # Remove bottom space, keep small
     ↪  top space
```

```
## `summarise()` has grouped output by 'array'. You can override using the
## `.groups` argument.
```

```r
fig_week_det
```



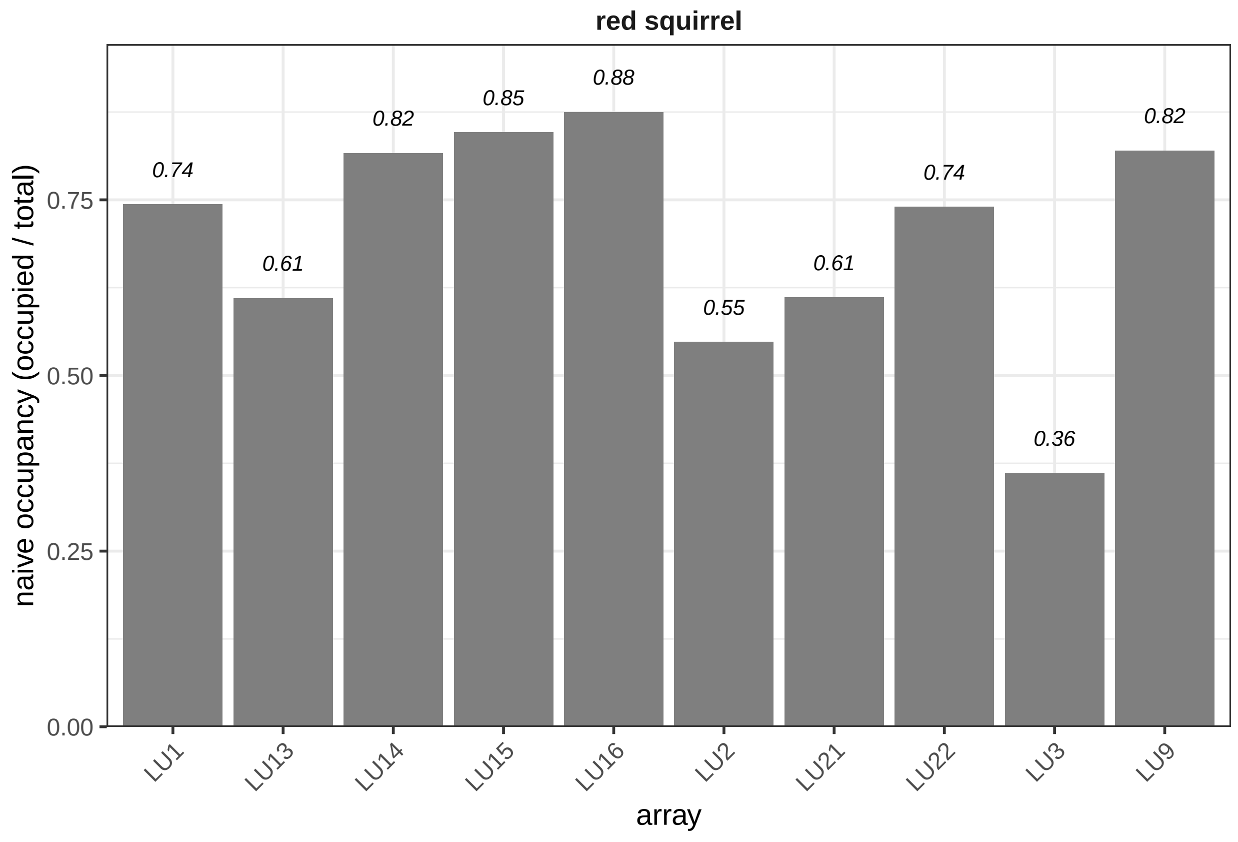
2.3. Naive occupancy per array

```
fig_occ_total <- presence_months %>%

  filter(species %in% c('red squirrel')) %>%

  group_by(array, site, species) %>%

  summarize(presence = max(presence)) %>%

  group_by(array, species) %>%

  # Number of sites present ('presence'),
  # Divided by the number of sites per array
  summarize(n_sites = n(),
            n_present = sum(presence),
            naive_occ = n_present / n_sites,
            .groups = 'drop') %>%

  ggplot(., aes(x = array)) +

    # Bar plot again
    geom_bar(aes(y = naive_occ), stat = "identity", fill = "grey50") +  # Total sites in grey

#     geom_bar(aes(y = n_present), stat = "identity", fill = "firebrick") +  # Present sites in red
```

```
    # Print a label just above the bars
    geom_text(aes(y = naive_occ + 0.05, label = round(naive_occ, 2)), size = 2.7, fontface =
     ↪  "italic") +

    theme_bw() +

    facet_wrap(~species, scales = 'fixed') +

    labs(x = "array",
         y = "naive occupancy (occupied / total)") +


    theme(
         axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
         strip.background = element_blank(),  # Remove facet background
         strip.text = element_text(face = "bold", size = 10)
         ) +

    scale_y_continuous(expand = expansion(mult = c(0, 0.05)))  # Remove bottom space, keep small
     ↪  top space
```

```
## `summarise()` has grouped output by 'array', 'site'. You can override using the
## `.groups` argument.
```

```
fig_occ_total
```



```
ggsave("./figures/red_squirrel_naive_occupancy.png", width = 5, height = 4)
```

What is the total naive occupancy for the study?

```
presence_months %>%

  filter(species %in% c('red squirrel')) %>%

  group_by(site) %>%

  summarize(presence = max(presence)) %>%

  summarize(n_sites = n(),
            n_occ = sum(presence))
```

```
## # A tibble: 1 x 2
##   n_sites n_occ
##     <int> <dbl>
## 1     430   305
```

```
305/430
```

```
## [1] 0.7093023
```

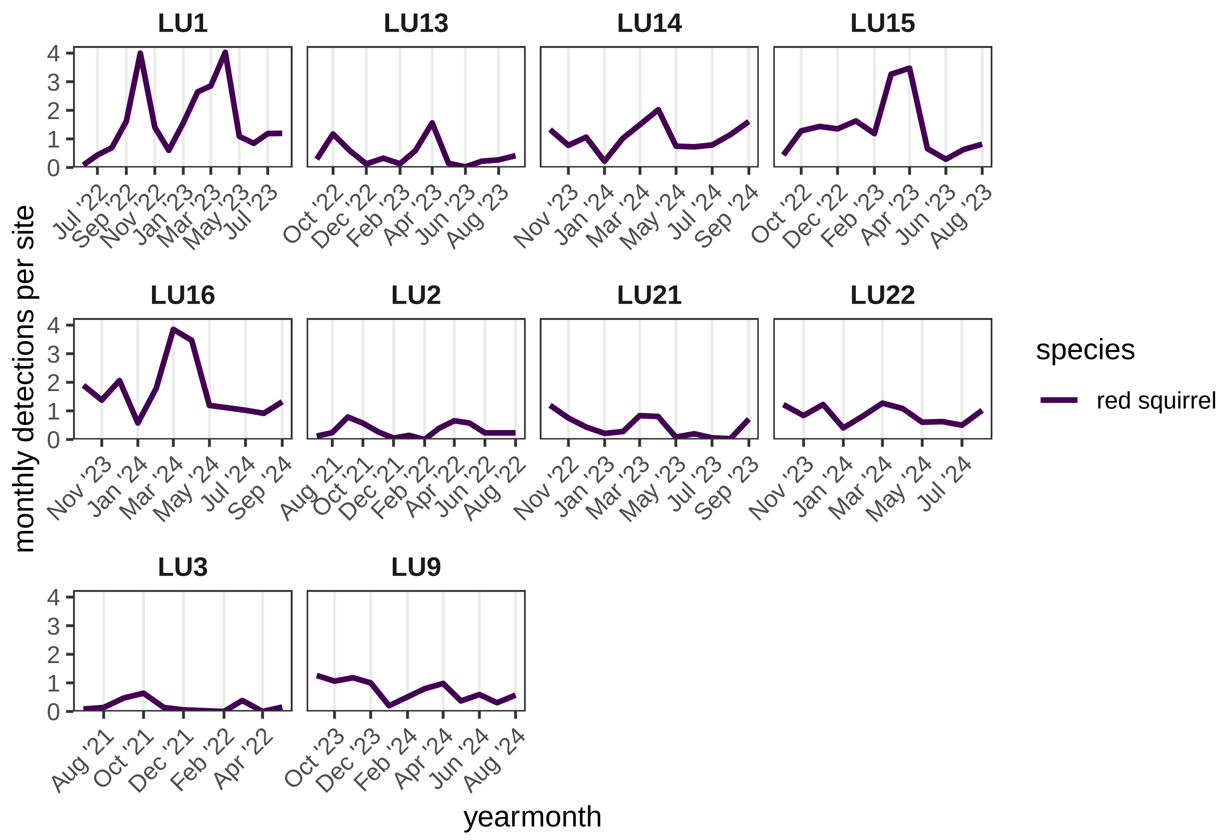
2.4. Temporal variation in detections

Does red squirrel density fluctuate substantially through the study period? If so, our uneven sampling among arrays (i.e., seasons and months that cameras were deployed) could create some issues.

```
fig_seasonal_det <- presence_months %>%

  filter(species %in% c('red squirrel')) %>%

  group_by(array, site, species, year, month) %>%

  summarize(det = sum(detections)) %>%

  group_by(array, species, year, month) %>%

  summarize(n_sites = n(),
            total_det = sum(det),
            avg_det = total_det / n_sites) %>%

  mutate(yearmonth = ym(paste(year, month, sep = "-"))) %>%  # Creates Year-Month column

  ggplot(., aes(x = yearmonth, y = avg_det, color = species)) +

    #geom_bar(stat = "identity", position = "dodge", fill = 'grey20') +

    geom_line(linewidth = 1) +  # Use lines instead of bars

    scale_color_viridis_d() +
    #geom_point(size = 1.5, color = 'grey20') +  # Optional: Add points to mark data

    facet_wrap(~array, scales = 'free_x') +

    labs(y = "monthly detections per site") +

    theme_bw() +

    theme(
         axis.text.x = element_text(angle = 45, hjust = 1),  # Rotate x-axis labels
         strip.background = element_blank(),  # Remove facet background
         strip.text = element_text(face = "bold", size = 10),
         panel.background = element_rect(fill = "white"),  # Set light gray background
         panel.grid.major.y = element_blank(),
         panel.grid.minor.y = element_blank()# Remove horizontal grid lines on the plot
         ) +

    scale_x_date(date_labels = "%b '%y", date_breaks = "2 months", minor_breaks = NULL) +  #
    ↪  6-month interval

    scale_y_continuous(expand = expansion(mult = c(0, 0.05)))  # Remove bottom space, keep small
    ↪  top space
```

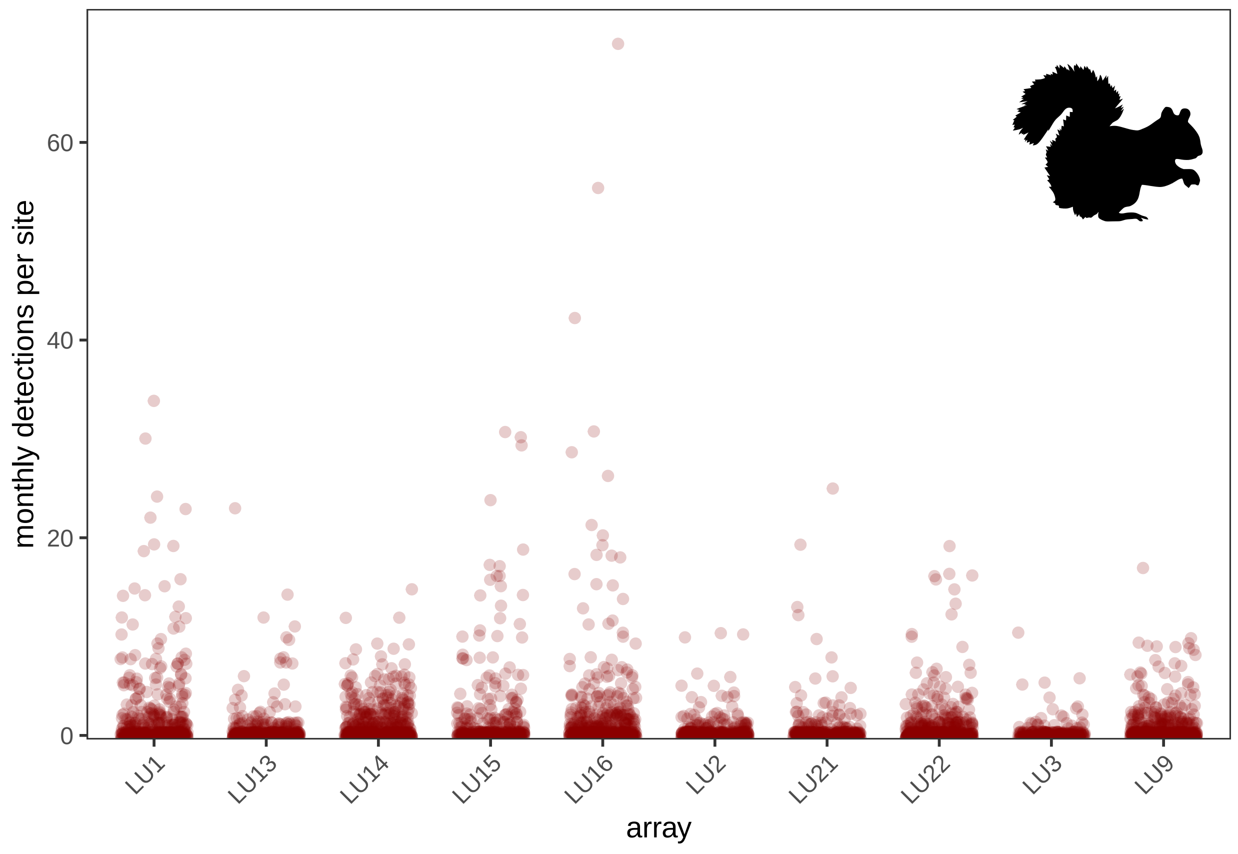
```
## `summarise()` has grouped output by 'array', 'site', 'species', 'year'. You can
## override using the `.groups` argument.
## `summarise()` has grouped output by 'array', 'species', 'year'. You can
## override using the `.groups` argument.
```

```
fig_seasonal_det
```



2.5. Figure of the response variable (monthly dets)

```
presence_months %>%
  filter(species == "red squirrel") %>%

ggplot(., aes(x = array, y = detections)) +

  geom_jitter(width = 0.3, alpha = 0.2, color = "darkred") +

  #geom_boxplot() +

  labs(x = "array",
       y = "monthly detections per site") +

  theme_bw() +

  theme(
       axis.text.x = element_text(angle = 45, hjust = 1),   # Rotate x-axis labels
       strip.background = element_blank(),  # Remove facet background
       panel.grid = element_blank(),
       strip.text = element_text(face = "bold", size = 10)
       ) +

  scale_y_continuous(expand = expansion(mult = c(0, 0.05))) +  # Remove bottom space, keep small
  ↪  top space
  rphylopic::add_phylopic(
    uuid = get_uuid(name = "Sciurus vulgaris"),
    x = 9.5,
    y = 60,
    height = 16
    )
```



```
ggsave(file = "./figures/allarrays_monthly_detections.png", width = 5, height = 3.2, dpi = 500)

# how many camera months and mean monthly detections?
presence_months %>%
  filter(species == "red squirrel") %>%
  summarize(n_months = n(),
            mean_detections = mean(detections))
```

```
## # A tibble: 1 x 2
##   n_months mean_detections
##      <int>           <dbl>
## 1     4918           0.934
```

It doesn't look too too variable to be quite honest, I think that we are likely not introducing any crazy bias into our data by using a proportional response with variable months among arrays/sites. The coolest thing to see is the huge difference in detections per site.

3. Summary statistics

3.1. Cameras and camera-trap survey effort

```
n_camera_days <- operating %>%

  group_by(array, site) %>%

  summarize(camera_days = n()) %>%

  # Any cams with <4 days of data were inevitably filtered out.
  # For now we have no filters applied based on the number of weeks.
  filter(camera_days >= 4)
```

```
## `summarise()` has grouped output by 'array'. You can override using the
## `.groups` argument.
```

```
sum(n_camera_days$camera_days)
```

```
## [1] 149273
```

```
sd(n_camera_days$camera_days)
```

```
## [1] 67.16344
```

```
mean(n_camera_days$camera_days)
```

```
## [1] 347.1465
```

```
min(n_camera_days$camera_days)
```

```
## [1] 32
```

```
max(n_camera_days$camera_days)
```

```
## [1] 455
```

```
n_cameras <- n_camera_days %>%

  group_by(array) %>%

  summarize(cams = n())

sd(n_cameras$cams)
```

```
## [1] 5.715476
```

```
sum(n_cameras$cams)
```

```
## [1] 430
```

```
min(n_cameras$cams)
```

```
## [1] 36
```

```
max(n_cameras$cams)
```

```
## [1] 50
```

```
mean(n_cameras$cams)
```

```
## [1] 43
```