Romania SEM Data Formatting

Marissa Dyck

2024-08-07

Contents

Before you begin **1**
- Notes . . . 1
- R and RStudio . . . 1
- R markdown . . . 2
- Install packages . . . 2
- Load libraries . . . 2

Data **2**
- Camera data . . . 2
- Animal data . . . 5
- Trap effort . . . 9

Before you begin

Notes

A few notes about this script.

If you are want to run through the full analysis with the published data make sure you download the whole (Romania_SEM repository)[https://github.com/marissadyck/Romania_SEM] from the author's (Marissa A. Dyck) GitHub. 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 (Romania_SEM.Rproj) in the repo 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 the data.

If you have question please email the author,

Marissa A. Dyck
Postdoctoral research fellow
University of Victoria
School of Environmental Studies
Email: marissadyck17@gmail.com

R and RStudio

Before starting you should ensure you have the latest version of R and RStudio downloaded. This code was generated under R version 4.2.3 and with RStudio version 2024.04.2+764.

You can download R and RStudio HERE

R markdown

This script is written in R markdown and thus uses a mix of coding markup languages and R. If you are planning to run this script with new data or make any modifications you will want to be familiar with some basics of R markdown or transfer code in the code chunks to a script.

Below is an R markdown cheatsheet to help you get started,
R markdown cheatsheet

Install packages

If you don't already have the following packages installed, use the code below to install them. *Note this code chunk will NOT automatically run when the file is knit since eval=FALSE is set, because I already have the packages and don't want to reinsall them every time. You will need to run this chunk separately if you need to install these packages.* You only need to do this once.

```
install.packages('tidyverse') # data tidying, visualization, and much more; this will load all tidyvers
install.packages('janitor') # used for cleaning up data
```

Load libraries

Then load the packages to your library. You will need to do this anytime you start a new session in R.

```
library(tidyverse) # data tidying, visualization, and much more; this will load all tidyverse packages,
```

```
## -- Attaching core tidyverse packages ------------------------ tidyverse 2.0.0 --
## v dplyr     1.1.2     v readr     2.1.4
## v forcats   1.0.0     v stringr   1.5.0
## v ggplot2   3.5.1     v tibble    3.2.1
## v lubridate 1.9.3     v tidyr     1.3.0
## v purrr     1.0.2
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
## i Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts to become error
```

```
library(janitor) # used for cleaning up data
```

```
##
## Attaching package: 'janitor'
##
## The following objects are masked from 'package:stats':
##
##     chisq.test, fisher.test
```

Data

Camera data

We will start with the camera data from our remote camera traps that were deployed in Romania.

Import data

We can load both data files at once and rowbind them since they have the same columns using a function in the *Purrr* package.

```
cameras <-

# provide file path (e.g. folders to find the data)
  file.path('data/raw',

            # provide the file names
            c('cams_data_winter_2018-2019.csv',
              'cams_data_autumn_2018-2019.csv')) %>%

   # use purrr map to read in files, the ~.x is a placeholder that refers to the object before the last
  map_dfr(~.x %>%
        read_csv(.))
```

```
## Rows: 64 Columns: 27
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): TrapCode, Impact
## dbl (25): Session, X, Y, Z, distnatlro, distsettle, diststream, denslocalr, ...
##
## 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.
## Rows: 76 Columns: 27
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (2): TrapCode, Impact
## dbl (25): Session, X, Y, Z, distnatlro, distsettle, diststream, denslocalr, ...
##
## 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.
```

View this data, by using `View(cameras)` or clicking on the object in the *environment*.

Format data

We need to do a bit of data cleaning to make this file usable for our analysis. The code chunk below will,

1. read in the data again
2. specify how to read in the columns
3. set the column names to lowercase for easier coding later
4. reformat some columns so data is appropriate for analysis
5. and finally select only to columns of data we need

```
cameras <-

# provide file path (e.g. folders to find the data)
  file.path('data/raw',

            # provide the file names
            c('cams_data_winter_2018-2019.csv',
              'cams_data_autumn_2018-2019.csv')) %>%

   # use purrr map to read in files, the ~.x is a placeholder that refers to the object before the last
```

```
  map_dfr(~.x %>%
        read_csv(.,

                 col_types = cols(Session = col_factor(),
                                  TrapCode = col_factor(),
                                      Impact = col_factor(),
                                      .default = col_number()))) %>%

   # set column names to lowercase
  set_names(
    names(.) %>%
      tolower()) %>%

  # Combine specific CORINE Land Cover types for forest into one

  mutate(clc_forest = (clc311 + clc312 + clc313)) %>%

  # select only columns of interest for SEM analysis and merginf with animals data frame

  select(session, trapcode, z, denslocalr, tri5, clc_forest, distnatlro, distsettle, diststream, distlo
```

Data checks

Let's take a look at this data using a few common functions

```
# check data structure
str(cameras)
```

```
## tibble [140 x 10] (S3: tbl_df/tbl/data.frame)
##  $ session   : Factor w/ 2 levels "2","3": 1 1 1 1 1 1 1 1 1 1 ...
##  $ trapcode  : Factor w/ 140 levels "118","119","120",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ z         : num [1:140] 1133 1189 1357 1257 902 ...
##  $ denslocalr: num [1:140] 0.289 0.291 0.295 0.286 0.228 ...
##  $ tri5      : num [1:140] 223 213 234 178 233 ...
##  $ clc_forest: num [1:140] 0.626 0.97 0.878 0.708 1 ...
##  $ distnatlro: num [1:140] 2900 5805 2163 1581 6360 ...
##  $ distsettle: num [1:140] 1780 5728 1118 1556 5692 ...
##  $ diststream: num [1:140] 1140 500 283 0 300 ...
##  $ distlocalr: num [1:140] 1063 1005 200 412 361 ...
```

```
# summary stats for columns
summary(cameras)
```

```
##  session    trapcode         z          denslocalr          tri5
##  2:64    118    :  1   Min.   : 663   Min.   :0.2127   Min.   : 66.38
##  3:76    119    :  1   1st Qu.:1030   1st Qu.:0.2413   1st Qu.:178.80
##          120    :  1   Median :1168   Median :0.2705   Median :221.61
##          122    :  1   Mean   :1169   Mean   :0.2700   Mean   :220.43
##          124    :  1   3rd Qu.:1295   3rd Qu.:0.2939   3rd Qu.:257.41
##          125    :  1   Max.   :1617   Max.   :0.3434   Max.   :494.01
##          (Other):134
##    clc_forest        distnatlro      distsettle      diststream
##  Min.   :0.08505   Min.   :  100   Min.   :    0   Min.   :   0.0
##  1st Qu.:0.64883   1st Qu.: 2148   1st Qu.: 1814   1st Qu.: 100.0
##  Median :0.82853   Median : 4433   Median : 4325   Median : 223.6
```

```
##  Mean   :0.76375   Mean   : 5179   Mean   : 5273   Mean   : 262.8
##  3rd Qu.:0.91221   3rd Qu.: 7441   3rd Qu.: 7263   3rd Qu.: 360.6
##  Max.   :1.00000   Max.   :15516   Max.   :17786   Max.   :1300.0
##
##    distlocalr
##  Min.   :   0.0
##  1st Qu.: 316.2
##  Median : 880.1
##  Mean   :1030.2
##  3rd Qu.:1633.2
##  Max.   :3101.6
##
```

```
# print first few rows
head(cameras,
     n = 25)
```

```
## # A tibble: 25 x 10
##    session trapcode     z denslocalr  tri5 clc_forest distnatlro distsettle
##    <fct>   <fct>    <dbl>      <dbl> <dbl>      <dbl>      <dbl>      <dbl>
##  1 2       118       1133      0.289  223.      0.626      2900       1780.
##  2 2       119       1189      0.291  213.      0.970      5805.      5728
##  3 2       120       1357      0.295  234.      0.878      2163.      1118.
##  4 2       122       1257      0.286  178.      0.708      1581.      1556.
##  5 2       124        902      0.228  233.      1          6360.      5692.
##  6 2       125        867      0.223  266.      1          5092.      4847.
##  7 2       127        795      0.232  145.      0.837      2816.      3833.
##  8 2       129       1098      0.279  210.      0.649      1044.      1170.
##  9 2       130       1278      0.270  183.      0.535      6083.      1703.
## 10 2       132       1173      0.266  254.      0.859      3569.      9729.
## # i 15 more rows
## # i 2 more variables: diststream <dbl>, distlocalr <dbl>
```

Everything looks good. 140 trap sites (trapcode), no NAs

Animal data

Now let's import the detection data from the camera traps.

Import data

This is a csv file with information from the camera traps provided by (Foundation Conservation Carpathia) [https://www.carpathia.org/], their staff and volunteers have already tagged the images and identified the species present in each.

```
# load species occurrence data
animals <- read_csv('data/raw/animals_on_cameras_2018-2019.csv')
```

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

```
## Rows: 6621 Columns: 24
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (18): GMU, TrapSite, StartDate, EndDate, TrapCode, Camera, Type, Date, R...
```

```
## dbl  (6): Sort, Session, X, Y, Z, Sequence
##
## 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.
```

View this data, by using `View(animals)` or clicking on the object in the *environment*.

Format data

We also need to do a bit of data cleaning to make this file usable for our analysis. The code chunk below will,

1. read in the data again
2. specify how to read in the columns
3. set the column names to lowercase for easier coding later
4. recode some of the entries
5. remove data that we won't use

```
# load species occurrence data

animals <- read_csv('data/raw/animals_on_cameras_2018-2019.csv',

                    # specify how to read in columns
                    col_types = cols(X = col_number(),
                                     Y = col_number(),
                                     Z = col_number(),
                                     StartDate = col_date(format = '%Y-%m-%d'),
                                     EndDate = col_date(format = '%Y-%m-%d'),
                                     RDate = col_date(format = '%Y-%m-%d'),
                                     Time = col_time(format = '%H:%M:%S'),
                                     NoAnimals = col_integer(),
                                     Sequence = col_integer(),
                                     Comments = col_character(),
                                     .default = col_factor()  #.default sets any unspecified columns
                    )) %>%

  # set column names to lowercase
  set_names(
    names(.) %>%
      tolower()) %>%

  # select just the columns of data we need to count species observations per season and trap
  select(session, date, trapcode, type, species) %>%

  # recode species data to group some types
  mutate( species = recode(species,
                           'Vehicle' = 'Human',
                           'Cow' = 'Livestock',
                           'Goat' = 'Livestock',
                           'Horse' = 'Livestock',
                           'Beech marten' = 'Mustelid',
                           'Pine marten' = 'Mustelid',
                           'Least weasel' = 'Mustelid',
```

```
                          'European polecat' = 'Mustelid')) %>%


  # remove  session 1 (trial period) and unknown species
  filter(! session == '1',
         ! is.na(species),
         ! species == 'Unknown',

         # select just the pictures to avoid duplicate occurrences at the same site
         type == 'P')
```

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

Will get an error about parsing issues just ignore.

Data checks

Overall Let's take a look at this data using a few common functions

```
# check data structure
str(animals)
```

```
## tibble [4,684 x 5] (S3: tbl_df/tbl/data.frame)
##  $ session : Factor w/ 3 levels "1","2","3": 2 2 2 2 2 2 2 2 2 2 ...
##  $ date    : Factor w/ 360 levels "1/1/18","1/13/18",..: 68 70 70 71 71 72 72 73 73 74 ...
##  $ trapcode: Factor w/ 186 levels "40","41","64",..: 46 46 48 46 48 46 48 46 48 46 ...
##  $ type    : Factor w/ 2 levels "P","V": 1 1 1 1 1 1 1 1 1 1 ...
##  $ species : Factor w/ 22 levels "Bear","European hare",..: 3 3 3 3 3 3 3 3 3 3 ...
```

```
# summary stats for columns
summary(animals)
```

```
##  session        date         trapcode    type          species
##  1:   0   3/30/19 :  50   127    : 212   P:4684   Fox      :985
##  2:2089   10/12/19:  44   249    : 142   V:   0   Red deer :773
##  3:2595   10/13/19:  41   124    : 114            Wild boar:694
##           3/22/19 :  40   289    : 112            Bear     :604
##           3/24/19 :  40   170    : 104            Roe deer :386
##           3/18/19 :  38   153    : 102            Lynx     :332
##           (Other) :4431   (Other):3898            (Other)  :910
```

```
# print first few rows
head(animals,
     n = 25)
```

```
## # A tibble: 25 x 5
##    session date     trapcode    type  species
##    <fct>   <fct>    <fct>       <fct> <fct>
##  1 2       4/12/18  123_189_195 P     Fox
##  2 2       10/21/18 123_189_195 P     Fox
##  3 2       10/21/18 173         P     Fox
##  4 2       10/23/18 123_189_195 P     Fox
##  5 2       10/23/18 173         P     Fox
##  6 2       10/25/18 123_189_195 P     Fox
```

```
##  7 2       10/25/18 173         P     Fox
##  8 2       11/2/18  123_189_195 P     Fox
##  9 2       11/2/18  173         P     Fox
## 10 2       11/15/18 123_189_195 P     Fox
## # i 15 more rows
```

Let's do a few more specific data checks to make sure everything is correct

Species names Let's make sure the species entries are correct and no spelling mistakes

```
levels(animals$species)
```

```
##  [1] "Bear"          "European hare" "Fox"           "Unknown"
##  [5] "Wolf"          "Badger"        "Roe deer"      "Wild cat"
##  [9] "Lynx"          "Red deer"      "Dog"           "Wild boar"
## [13] "Mustelid"      "Squirrel"      "Bird"          "Chamois"
## [17] "Human"         "Hedgehog"      "Livestock"     "Mouse"
## [21] "Domestic cat"  "Otter"
```

Everything looks good here

Trapcodes Let's check that no trap codes are mis-entered or repeated etc.

```
levels(animals$trapcode)
```

```
##   [1] "40"          "41"          "64"          "22"          "66"
##   [6] "17"          "18"          "19"          "36"          "37"
##  [11] "24_86"       "27"          "28"          "31"          "48"
##  [16] "33"          "54"          "25_77"       "43"          "49"
##  [21] "53"          "45_84"       "29"          "55"          "70"
##  [26] "30"          "67"          "20_21_23"    "44"          "68"
##  [31] "56"          "61"          "46_78"       "50"          "52"
##  [36] "58"          "63"          "32"          "60"          "35_47"
##  [41] "42_75"       "51"          "39_71"       "65"          "76"
##  [46] "123_189_195" "119"         "173"         "120"         "118"
##  [51] "127"         "124"         "121_176"     "130"         "126_194"
##  [56] "134"         "131_192"     "137_203"     "142"         "144"
##  [61] "179"         "136_207"     "139_193"     "132"         "135"
##  [66] "146"         "161"         "128_208"     "145_185_196" "122"
##  [71] "157"         "167"         "164_200"     "147"         "171"
##  [76] "149"         "170"         "172"         "129"         "174"
##  [81] "155"         "148"         "153"         "133"         "158_205"
##  [86] "162_198_202" "140"         "143"         "165_177"     "175"
##  [91] "178"         "154"         "156"         "163"         "150"
##  [96] "160"         "188"         "125"         "151"         "265"
## [101] "138_187"     "168_181"     "152"         "184"         "182"
## [106] "191"         "166_199"     "183"         "169_201"     "190"
## [111] "266_294_309" "212_291"     "219"         "213_290"     "211_235"
## [116] "218"         "220"         "217"         "284"         "287"
## [121] "216"         "225"         "227"         "242"         "243"
## [126] "248"         "249"         "230"         "246"         "239"
## [131] "245"         "259"         "237"         "253"         "247_295"
## [136] "224"         "241"         "251"         "262"         "228"
## [141] "260"         "257"         "232"         "233"         "240"
## [146] "254"         "226"         "261"         "263_320"     "236"
## [151] "264"         "273"         "229"         "285"         "269"
```

```
## [156] "272"         "258"         "275"         "279_292"     "280"
## [161] "244_293"     "252_313"     "250"         "255"         "283"
## [166] "282"         "289"         "277_301"     "274"         "231"
## [171] "270"         "268"         "281"         "276"         "238"
## [176] "306"         "256"         "271"         "278"         "234_316"
## [181] "288"         "267_310"     "311"         "303"         "299"
## [186] "234"
```

Hmm there seem to be more sites than in the camera data, this may be because it's still counting the trapcodes from session 1 even though we filtered those out in the data formatting steps. Let's try something else to check if the trapcodes match the camera data

```
# check which trapcodes are in animals that are not in the cameras data
setdiff(levels(animals$trapcode),
        levels(cameras$trapcode))
```

```
##  [1] "40"       "41"       "64"       "22"       "66"       "17"
##  [7] "18"       "19"       "36"       "37"       "24_86"    "27"
## [13] "28"       "31"       "48"       "33"       "54"       "25_77"
## [19] "43"       "49"       "53"       "45_84"    "29"       "55"
## [25] "70"       "30"       "67"       "20_21_23" "44"       "68"
## [31] "56"       "61"       "46_78"    "50"       "52"       "58"
## [37] "63"       "32"       "60"       "35_47"    "42_75"    "51"
## [43] "39_71"    "65"       "76"       "234"
```

```
# and vice versa
setdiff(levels(cameras$trapcode),
        levels(animals$trapcode))
```

```
## character(0)
```

It looks like the animals data has extra trapcodes but no missing ones from the cameras data. I think it is still retaining the codes from session 1 in Rs memory for some reason after we used the filter function to remove them, let's check by looking for any data from the trapcodes printed above

```
animals %>%

  filter(trapcode == '40')
```

```
## # A tibble: 0 x 5
## # i 5 variables: session <fct>, date <fct>, trapcode <fct>, type <fct>,
## #   species <fct>
```

No data, so we are good

Trap effort

Now we need to load in the trap effort so we can check that we can quanitfy camera effort for each site.

```
effort <- file.path('data/raw',

                    # provide file names
                    c('Trap_Effort_S2.csv',
                    'Trap_Effort_S3.csv')) %>%

  map(~.x %>%

        read_csv(.) %>%
```

```
        # set column names to lowercase for coding ease
        set_names(
          names(.) %>%
            tolower())) %>%

  purrr::set_names(c('session_2',
                     'session_3'))
```

```
## Rows: 64 Columns: 9
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): TrapCode
## dbl (8): Occasion_1, Occasion_2, Occasion_3, Occasion_4, Occasion_5, Occasio...
##
## 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.
## Rows: 76 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (1): TrapCode
## dbl (7): Occasion_1, Occasion_2, Occasion_3, Occasion_4, Occasion_5, Occasio...
##
## 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.
```

Data checks

```
str(effort)
```

```
## List of 2
##  $ session_2: spc_tbl_ [64 x 9] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##   ..$ trapcode  : chr [1:64] "118" "119" "120" "122" ...
##   ..$ occasion_1: num [1:64] 14 14 14 14 14 14 14 0 14 14 ...
##   ..$ occasion_2: num [1:64] 14 14 14 14 12 14 14 9 14 14 ...
##   ..$ occasion_3: num [1:64] 14 14 14 14 14 11 14 14 14 14 ...
##   ..$ occasion_4: num [1:64] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_5: num [1:64] 14 14 14 14 13 14 13 14 14 14 ...
##   ..$ occasion_6: num [1:64] 14 14 14 14 14 14 14 12 14 14 ...
##   ..$ occasion_7: num [1:64] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_8: num [1:64] 11 11 14 9 11 14 8 14 14 10 ...
##   ..- attr(*, "spec")=
##   .. .. cols(
##   .. ..   TrapCode = col_character(),
##   .. ..   Occasion_1 = col_double(),
##   .. ..   Occasion_2 = col_double(),
##   .. ..   Occasion_3 = col_double(),
##   .. ..   Occasion_4 = col_double(),
##   .. ..   Occasion_5 = col_double(),
##   .. ..   Occasion_6 = col_double(),
##   .. ..   Occasion_7 = col_double(),
##   .. ..   Occasion_8 = col_double()
##   .. .. )
##   ..- attr(*, "problems")=<externalptr>
```

```
##  $ session_3: spc_tbl_ [76 x 8] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##   ..$ trapcode  : chr [1:76] "216" "217" "218" "219" ...
##   ..$ occasion_1: num [1:76] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_2: num [1:76] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_3: num [1:76] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_4: num [1:76] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_5: num [1:76] 14 12 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_6: num [1:76] 14 14 14 14 14 14 14 14 14 14 ...
##   ..$ occasion_7: num [1:76] 6 14 14 14 14 14 14 14 14 14 ...
##   ..- attr(*, "spec")=
##   .. .. cols(
##   .. ..   TrapCode = col_character(),
##   .. ..   Occasion_1 = col_double(),
##   .. ..   Occasion_2 = col_double(),
##   .. ..   Occasion_3 = col_double(),
##   .. ..   Occasion_4 = col_double(),
##   .. ..   Occasion_5 = col_double(),
##   .. ..   Occasion_6 = col_double(),
##   .. ..   Occasion_7 = col_double()
##   .. .. )
##   ..- attr(*, "problems")=<externalptr>
```

What I need to do is add a column to each data frame that specifies the session, and add a column that totals the number of active camera days from each occasion since this data were originally formatted for a different occupancy analysis with 2-week camera occasions

Format data

Add columns The code below adds two new columns to each data frame separately that specifies the session, and totals the number of days the camera was functioning from the raw data.

```
# add session column to each and total effort column
effort$session_2 <- effort$session_2 %>%

  add_column(.,
             session = 2) %>%

  mutate(trap_effort = rowSums(across(occasion_1:occasion_8)))

head(effort$session_2)
```

```
## # A tibble: 6 x 11
##   trapcode occasion_1 occasion_2 occasion_3 occasion_4 occasion_5 occasion_6
##   <chr>         <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 118              14         14         14         14         14         14
## 2 119              14         14         14         14         14         14
## 3 120              14         14         14         14         14         14
## 4 122              14         14         14         14         14         14
## 5 124              14         12         14         14         13         14
## 6 125              14         14         11         14         14         14
## # i 4 more variables: occasion_7 <dbl>, occasion_8 <dbl>, session <dbl>,
## #   trap_effort <dbl>
```

```
effort$session_3 <- effort$session_3 %>%

  add_column(.,
```

```
            session = 3) %>%

   mutate(trap_effort = rowSums(across(occasion_1:occasion_7)))

head(effort$session_3)
```

```
## # A tibble: 6 x 10
##   trapcode occasion_1 occasion_2 occasion_3 occasion_4 occasion_5 occasion_6
##   <chr>         <dbl>      <dbl>      <dbl>      <dbl>      <dbl>      <dbl>
## 1 216              14         14         14         14         14         14
## 2 217              14         14         14         14         12         14
## 3 218              14         14         14         14         14         14
## 4 219              14         14         14         14         14         14
## 5 220              14         14         14         14         14         14
## 6 224              14         14         14         14         14         14
## # i 3 more variables: occasion_7 <dbl>, session <dbl>, trap_effort <dbl>
```

Bind data Now with both select just the columns we need and the we will rowbind the data so we have one data frame with info for both sessions

```
effort_tidy <- effort %>%

  map(~.x %>%

        select(trapcode,
               session,
               trap_effort) %>%

         # make session a factor to join with other data
  mutate(session = as.factor(session)))


# check dta
head(effort_tidy)
```

```
## $session_2
## # A tibble: 64 x 3
##    trapcode session trap_effort
##    <chr>    <fct>         <dbl>
##  1 118      2               109
##  2 119      2               109
##  3 120      2               112
##  4 122      2               107
##  5 124      2               106
##  6 125      2               109
##  7 127      2               105
##  8 129      2                91
##  9 130      2               112
## 10 132      2               108
## # i 54 more rows
##
## $session_3
## # A tibble: 76 x 3
##    trapcode session trap_effort
```

```
##    <chr>    <fct>         <dbl>
##  1 216      3                90
##  2 217      3                96
##  3 218      3                98
##  4 219      3                98
##  5 220      3                98
##  6 224      3                98
##  7 225      3                98
##  8 226      3                98
##  9 227      3                98
## 10 228      3                98
## # i 66 more rows
```

```
# bind data
trap_effort <-
bind_rows(effort_tidy)
```

Remove messy data lists that we don't need

```
rm(effort)
rm(effort_tidy)
```

Join animals and effort data and count species occurrences

Now we want to join the animals and effort data frames so we have a single data frame for our analysis, and also we will count the number of species occurrences (independent detections) from the animal data and add a column for this as it will be the response variable in our analysis

```
animals_effort <- animals %>%

  # join data frames
  left_join(trap_effort,
            by = c('session', 'trapcode')) %>%

   # group data
  group_by(session, trapcode, species, trap_effort) %>%

  # use summarize to species ocurrences
  summarize(n = n(),
            .groups = 'drop') %>%

  # ensure only one row per unique combination is returned
  distinct(session, trapcode, species, .keep_all = TRUE) %>%

  # pivot the data to wide format so each species has a column
  pivot_wider(names_from = species,
              values_from = n,
              values_fill = list(n = 0))

# check data
head(animals_effort)
```

```
## # A tibble: 6 x 22
##   session trapcode    trap_effort `European hare`   Fox  Wolf Badger  Lynx Human
##   <fct>   <chr>             <dbl>           <int> <int> <int>  <int> <int> <int>
## 1 2       118                 109               1     9     2      7    10     1
```

```
## 2 2       119                 109                 0     3     0      1     7     0
## 3 2       120                 112                 0     4     0      2     2     0
## 4 2       121_176             104                 0    12     0      0     0     0
## 5 2       122                 107                 0    14     3      0     8     0
## 6 2       123_189_195         107                 0    24     3      6     2     0
## # i 13 more variables: `Red deer` <int>, Bear <int>, `Roe deer` <int>,
## #   `Wild boar` <int>, Mustelid <int>, Chamois <int>, Bird <int>,
## #   `Wild cat` <int>, Dog <int>, Squirrel <int>, Livestock <int>,
## #   Hedgehog <int>, Otter <int>
```

Merge camera and occurrence

Finally let's merge the camera and occurrence data so we have all the data (response variables, covariates, summary info) in one data frame

```
# merge species occurrence data with the camera site variables from cams

ro_sem_data_effort <- animals_effort %>%

  # join data
  left_join(cameras,
    by = c('session', 'trapcode')) %>%

  # remove any NAS
  drop_na() %>%

  # set the species column names to lowercase and clean them up
  set_names(
    names(.) %>%
      tolower()) %>%
  clean_names() %>%
  rename(hare = european_hare)
```

Save new cleaned data

```
write_csv(ro_sem_data_effort,
          'data/processed/ro_sem_data_effort_2018-2019.csv')
```