Romania SEM Data Formatting

Marissa Dyck

2024-08-07

Contents

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)[] from tMarissa Dyck's 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) 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.

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.

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

Load libraries

Then load the packages to your library.

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

```
## -- 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 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*.

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 approporiate for analysis
5. seelct only 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
```

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 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.

```r
# 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: 25
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (16): GMU, TrapSite, TrapCode, Camera, Type, Date, RDate, Time, Occasio...
## dbl   (6): Sort, Session, X, Y, Z, Sequence
## lgl   (1): Detector
## date  (2): StartDate, EndDate
##
## 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

```r
# 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/ 452 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    3/22/19:  40   249     : 142   V:   0   Red deer  :773
##  3:2595    3/24/19:  40   124     : 114            Wild boar:694
##             3/18/19:  38   289     : 112            Bear      :604
##             2/5/19 :  37   170     : 104            Roe deer  :386
##             1/13/20:  37   153     : 102            Lynx      :332
##             (Other):4442   (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 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 retainig 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 I think we are good

Species occurrence

Now we need to calculate species occurrence from the animals data, the code chunk below will

1. create a new object called species occurrence that is a product of the animals data
2. group the data
3. count the number of observations in each group
4. pivot the data to wide formate for use in SEM

```
# count occurrences per species per season and trap

species_occurrences <- animals %>%

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

  # use summarize to count data
  summarize(n = n()) %>%

  pivot_wider(names_from = species,
              values_from = n,
              values_fill = 0)
```

```
## ‘summarise()‘ has grouped output by ’session’, ’trapcode’. You can override
## using the ‘.groups‘ argument.
```

Now let’s remove the old data we don’t need anymore

```r
# remove animals data from global env

rm(animals)
```

Merge camera and occurrence

Let’s merge the camera and occurrence data into one file we can use for analysis

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

ro_sem_dat <- species_occurrences %>%

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

Now we need to export and save this data as a csv for later use

```r
write_csv(ro_sem_dat,
          'data/processed/ro_sem_dat_2018-2019.csv')
```