OSM report 2022-2023

Marissa Dyck

2024-05-08

Contents

Before you begin **2**

Notes . . . 2

R and RStudio . . . 3

R markdown . . . 3

Install packages . . . 3

Load libraries . . . 3

Data **4**

Load detection data . . . 4

Merge data . . . 5

Data checks . . . 6

Summary . . . 8

Data Formatting . . . 8

Subset data . . . 10

Detection plots **12**

Detection data . . . 12

Detection plots . . . 12

Save detection plots . . . 19

Naive occupancy **20**

Data . . . 20

Occupancy plots . . . 20

Save occupancy plots . . . 27

Final combined plots for report **27**

Finish with detection data . . . 34

Global Analysis prep **34**

Response metrics 34

Covariates 38

Global Analysis **41**

Black bear 43

Caribou 54

Coyote 59

Fisher 63

Grey wolf 67

Lynx 72

Moose 76

Red fox 80

White-tailed deer 84

Top buffers 89

Analysis Prep **90**

Summary 90

Data prep 90

Subset Analysis **176**

Anthropogenic 176

Landscape 210

The first two chunks of this r markdown file after the r setup allow for plot zooming, but it also means that the html file must be opened in a browser to view the document properly. When it knits in RStudio the preview will appear empty but the html when opened in a browser will have all the info and you can click on each plot to Zoom in on it.

Before you begin

Notes

A few notes about this script.

If you are running this with the 2022-2023 data make sure you download the whole (OSM_2022-2023 GitHub repository)[https://github.com/ACMElabUvic/OSM_2022-2023] from the ACMElabUvic 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 (OSM_2022-2023.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.

Lastly, if you are looking to adapt this code for a future year of data, you will want to ensure you have run the 2_ACME_landscape_covariate_exploration_script.Rmd with your data as there is much data formatting,

cleaning, and restructuring that has to be done before this code will work. *Helpful note: The files are numbered in the order they are used for this analysis.*

If you have question please email the most recent author, currently

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

(*update/add authors as needed*)

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. *NOTE this will not run automatically as eval=FALSE is included in the chunk setup (i.e. I don't want it to run every time I run this code since I have the packages installed).

```
install.packages('tidyverse')
install.packages('ggpubr')
install.packages('corrplot')
install.packages('Hmisc')
install.packages('glmmTMB')
install.packages('MuMIn')
install.packages('TMB', type = 'source')
install.packages('rphylopic')
install.packages('broom')
```

Load libraries

Then load the packages to your library so they are usable for this session.

```
library(tidyverse) # data tidying, visualization, and much more; this will load all tidyverse packages,
library(ggpubr) # make modifications to plot for publication (arrange plots)
library(PerformanceAnalytics)    # Used to generate a correlation plot
library(Hmisc) # used to generate histograms for all variables in data frame
library(glmmTMB)      # Constructing GLMMs
```

```
## Warning in checkMatrixPackageVersion(): Package version inconsistency detected.
## TMB was built with Matrix version 1.4.1
## Current Matrix version is 1.5.3
## Please re-install 'TMB' from source using install.packages('TMB', type = 'source') or ask CRAN for a
```

```
library(MuMIn) # for model selection
library(rphylopic) # add animal silhouettes to graphs
library(broom) # extracting odds ratios in a tidy format
```

Data

Load detection data

Read in saved and cleaned detection data for the 6 LUs from 2021-2022 and 2022-2023 e.g., the 1_ACME_camera_script_9-2-2024.R.

These are two separate files from the two different fiscal years, so they need to be imported and then merged into one data file for plotting. Since both are stored in the data/processed/ folder we can read them both in as a list with *purrr*.

```
# detection data
# read in saved and cleaned detection data from the 1_ACME_camera_script_9-2-2024.R
detections <- file.path('data/processed',

                        c('OSM_ind_det_2021.csv',
                          'OSM_ind_det_2022.csv')) %>%

 map(~.x %>%
       read_csv(.) %>%

       # ensure the array, site, species, and event_id read in as factors
       mutate_if(is.character,
                 as.factor)) %>%

  # give names to each data frame in list
  purrr::set_names('dets_2021',
                   'dets_2022') # R doesn't like when they are just numbers, you can make it work but i
```

```
## Rows: 6696 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): array, site, species, event_id
## dbl  (3): month, year, timediff
## dttm (1): datetime
##
## 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: 14063 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (4): array, site, species, event_id
## dbl  (3): month, year, timediff
```

```
## dttm (1): datetime
##
## 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.
```

```
# look at data structure
str(detections)
```

```
## List of 2
##  $ dets_2021: tibble [6,696 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ array   : Factor w/ 2 levels "LU2","LU3": 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ site    : Factor w/ 78 levels "LU2_03","LU2_05",..: 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ species : Factor w/ 35 levels "Arctic hare",..: 28 28 35 35 35 35 35 35 35 35 ...
##   ..$ datetime: POSIXct[1:6696], format: "2021-07-15 11:40:07" "2022-02-06 15:11:06" ...
##   ..$ month   : num [1:6696] 7 2 7 8 8 8 8 8 8 9 ...
##   ..$ year    : num [1:6696] 2021 2022 2021 2021 2021 ...
##   ..$ timediff: num [1:6696] NA 296839 NA 28519 13230 ...
##   ..$ event_id: Factor w/ 6696 levels "E1000000","E1000001",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ dets_2022: tibble [14,063 x 8] (S3: tbl_df/tbl/data.frame)
##   ..$ array   : Factor w/ 4 levels "LU01","LU13",..: 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ site    : Factor w/ 155 levels "LU01_06","LU01_10",..: 1 1 1 1 1 1 1 1 1 1 ...
##   ..$ species : Factor w/ 39 levels "ATVer","Beaver",..: 31 31 38 38 38 38 38 38 38 38 ...
##   ..$ datetime: POSIXct[1:14063], format: "2022-06-17 10:01:52" "2023-09-10 12:51:15" ...
##   ..$ month   : num [1:14063] 6 9 6 7 7 7 8 8 8 8 ...
##   ..$ year    : num [1:14063] 2022 2023 2022 2022 2022 ...
##   ..$ timediff: num [1:14063] NA 648166 NA 31847 21429 ...
##   ..$ event_id: Factor w/ 14063 levels "E0","E1","E10",..: 1 2 5176 6287 7398 8509 9620 10731 11842 1
```

Merge data

Now we need to merge these two files to make plotting them easier.

They have the same columns so we could just the base R `rbind()` function, but in case there are differences in the columns in the future, let's use the cleaner `dplyr::bind_rows()` function

```
# Join two years of detection data
detections_merged <- dplyr::bind_rows(detections$dets_2021,
                                      detections$dets_2022)

# check structure of new data
str(detections_merged)
```

```
## tibble [20,759 x 8] (S3: tbl_df/tbl/data.frame)
##  $ array   : Factor w/ 6 levels "LU2","LU3","LU01",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ site    : Factor w/ 233 levels "LU2_03","LU2_05",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ species : Factor w/ 42 levels "Arctic hare",..: 28 28 35 35 35 35 35 35 35 35 ...
##  $ datetime: POSIXct[1:20759], format: "2021-07-15 11:40:07" "2022-02-06 15:11:06" ...
##  $ month   : num [1:20759] 7 2 7 8 8 8 8 8 8 9 ...
##  $ year    : num [1:20759] 2021 2022 2021 2021 2021 ...
##  $ timediff: num [1:20759] NA 296839 NA 28519 13230 ...
##  $ event_id: Factor w/ 20759 levels "E1000000","E1000001",..: 1 2 3 4 5 6 7 8 9 10 ...
```

Data checks

Check to make sure the data looks good after merging before moving on.

Arrays and sites

Let's check that there are the correct number of levels for arrays and sites, there should be 6 arrays and 233 sites (155 from 2022-2023 and 78 from 2021-2022).

```
# this will provide a summary of the levels
str(detections_merged)
```

```
## tibble [20,759 x 8] (S3: tbl_df/tbl/data.frame)
##  $ array   : Factor w/ 6 levels "LU2","LU3","LU01",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ site    : Factor w/ 233 levels "LU2_03","LU2_05",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ species : Factor w/ 42 levels "Arctic hare",..: 28 28 35 35 35 35 35 35 35 35 ...
##  $ datetime: POSIXct[1:20759], format: "2021-07-15 11:40:07" "2022-02-06 15:11:06" ...
##  $ month   : num [1:20759] 7 2 7 8 8 8 8 8 8 9 ...
##  $ year    : num [1:20759] 2021 2022 2021 2021 2021 ...
##  $ timediff: num [1:20759] NA 296839 NA 28519 13230 ...
##  $ event_id: Factor w/ 20759 levels "E1000000","E1000001",..: 1 2 3 4 5 6 7 8 9 10 ...
```

```
# this will list all unique entries (levels) for each variable
levels(detections_merged$array)
```

```
## [1] "LU2"  "LU3"  "LU01" "LU13" "LU15" "LU21"
```

```
levels(detections_merged$site)
```

```
##   [1] "LU2_03"   "LU2_05"   "LU2_100"  "LU2_101"  "LU2_106"  "LU2_110"
##   [7] "LU2_119"  "LU2_123"  "LU2_13"   "LU2_137"  "LU2_143"  "LU2_15"
##  [13] "LU2_153"  "LU2_155"  "LU2_17"   "LU2_18"   "LU2_20"   "LU2_21"
##  [19] "LU2_24"   "LU2_25"   "LU2_26"   "LU2_27"   "LU2_29"   "LU2_30"
##  [25] "LU2_31"   "LU2_32"   "LU2_33"   "LU2_34"   "LU2_36"   "LU2_38"
##  [31] "LU2_39"   "LU2_40"   "LU2_42"   "LU2_44"   "LU2_46"   "LU2_47"
##  [37] "LU2_50"   "LU2_51"   "LU2_52"   "LU2_54"   "LU2_56"   "LU2_57"
##  [43] "LU3_05"   "LU3_07"   "LU3_10"   "LU3_102"  "LU3_103"  "LU3_111"
##  [49] "LU3_126"  "LU3_129"  "LU3_13"   "LU3_131"  "LU3_137"  "LU3_14"
##  [55] "LU3_147"  "LU3_15"   "LU3_17"   "LU3_18"   "LU3_19"   "LU3_20"
##  [61] "LU3_21"   "LU3_25"   "LU3_27"   "LU3_28"   "LU3_32"   "LU3_36"
##  [67] "LU3_38"   "LU3_39"   "LU3_40"   "LU3_41"   "LU3_44"   "LU3_45"
##  [73] "LU3_46"   "LU3_48"   "LU3_49"   "LU3_50"   "LU3_51"   "LU3_52"
##  [79] "LU01_06"  "LU01_10"  "LU01_11"  "LU01_13"  "LU01_22"  "LU01_25"
##  [85] "LU01_27"  "LU01_30"  "LU01_32"  "LU01_36"  "LU01_40"  "LU01_41"
##  [91] "LU01_43"  "LU01_44"  "LU01_45"  "LU01_46"  "LU01_47"  "LU01_48"
##  [97] "LU01_60"  "LU01_63"  "LU01_64"  "LU01_66"  "LU01_67"  "LU01_70"
## [103] "LU01_71"  "LU01_72"  "LU01_73"  "LU01_74"  "LU01_75"  "LU01_76"
## [109] "LU01_77"  "LU01_78"  "LU01_79"  "LU01_80"  "LU01_82"  "LU01_83"
## [115] "LU01_84"  "LU01_85"  "LU01_86"  "LU13_03"  "LU13_05"  "LU13_06"
## [121] "LU13_08"  "LU13_11"  "LU13_12"  "LU13_128" "LU13_13"  "LU13_131"
## [127] "LU13_14"  "LU13_15"  "LU13_16"  "LU13_17"  "LU13_18"  "LU13_19"
```

```
## [133] "LU13_20"  "LU13_21"  "LU13_22"  "LU13_26"  "LU13_27"  "LU13_30"
## [139] "LU13_32"  "LU13_33"  "LU13_34"  "LU13_35"  "LU13_36"  "LU13_37"
## [145] "LU13_38"  "LU13_41"  "LU13_43"  "LU13_45"  "LU13_47"  "LU13_49"
## [151] "LU13_51"  "LU13_52"  "LU13_53"  "LU13_55"  "LU13_56"  "LU13_57"
## [157] "LU13_59"  "LU13_70"  "LU15_01"  "LU15_02"  "LU15_03"  "LU15_04"
## [163] "LU15_07"  "LU15_08"  "LU15_09"  "LU15_10"  "LU15_11"  "LU15_12"
## [169] "LU15_14"  "LU15_15"  "LU15_16"  "LU15_17"  "LU15_18"  "LU15_19"
## [175] "LU15_20"  "LU15_21"  "LU15_22"  "LU15_23"  "LU15_24"  "LU15_25"
## [181] "LU15_26"  "LU15_27"  "LU15_28"  "LU15_29"  "LU15_30"  "LU15_31"
## [187] "LU15_32"  "LU15_34"  "LU15_36"  "LU15_37"  "LU15_40"  "LU15_41"
## [193] "LU15_43"  "LU15_44"  "LU15_46"  "LU15_58"  "LU15_61"  "LU21_06"
## [199] "LU21_09"  "LU21_10"  "LU21_100" "LU21_105" "LU21_106" "LU21_107"
## [205] "LU21_109" "LU21_114" "LU21_116" "LU21_119" "LU21_122" "LU21_126"
## [211] "LU21_14"  "LU21_153" "LU21_16"  "LU21_164" "LU21_21"  "LU21_23"
## [217] "LU21_27"  "LU21_32"  "LU21_36"  "LU21_41"  "LU21_52"  "LU21_56"
## [223] "LU21_57"  "LU21_59"  "LU21_63"  "LU21_68"  "LU21_74"  "LU21_78"
## [229] "LU21_82"  "LU21_871" "LU21_93"  "LU21_97"  "LU21_98"
```

Everything looks good.

NAs

Let's ensure no NAs were introduced where there shouldn't be during the joining process.

```
# check for NAs introduced during data merge
summary(detections_merged)
```

```
##   array           site                    species
##  LU2 :4711   LU01_47:  405   White-tailed deer:6141
##  LU3 :1985   LU01_84:  396   Snowshoe hare    :4571
##  LU01:6283   LU01_66:  383   Red squirrel     :2201
##  LU13:1972   LU01_27:  320   Black bear       :1997
##  LU15:2951   LU2_39 :  310   Coyote           :1285
##  LU21:2857   LU2_50 :  292   Moose            : 693
##              (Other):18653   (Other)          :3871
##     datetime                         month            year
##  Min.   :2021-07-08 10:21:40.00   Min.   : 1.000   Min.   :2021
##  1st Qu.:2022-04-25 01:14:32.50   1st Qu.: 5.000   1st Qu.:2022
##  Median :2022-11-03 19:10:30.00   Median : 8.000   Median :2022
##  Mean   :2022-10-09 18:09:20.27   Mean   : 7.204   Mean   :2022
##  3rd Qu.:2023-05-05 01:40:45.50   3rd Qu.: 9.000   3rd Qu.:2023
##  Max.   :2023-10-02 11:08:57.00   Max.   :12.000   Max.   :2023
##
##     timediff          event_id
##  Min.   :    30   E1000000:    1
##  1st Qu.:  1427   E1000001:    1
##  Median :  5270   E1000002:    1
##  Mean   : 30292   E1000003:    1
##  3rd Qu.: 18443   E1000004:    1
##  Max.   :653883   E1000005:    1
##  NA's   :2446     (Other) :20753
```

THe only NAs are in the timediff column which is what we expect since any of the first observations won't have a a value for timediff. If you are confused by this re-visit the 1_ACME_camera_script.

Summary

Some summaries of the merged detection data

```
focal_species <- c('Black bear',
                   'Caribou',
                   'Coyote',
                   'Fisher',
                   'Grey wolf',
                   'Lynx',
                   'Moose',
                   'Red fox',
                   'White-tailed deer')

focal_mammals <- detections_merged %>%

  filter(species %in% focal_species)
```

Data Formatting

In order to get plots that have the same formatting as last years' report we have to do a bit of data formatting. First we need to make sure we are including the same relevant species (some were ignored for last years' report or grouped together).

Last years report had the following species

- white-tailed deer
- snowshoe hare
- black bear
- coyote
- red squirrel
- fisher
- unknown
- moose
- lynx
- spruce grouse
- red fox
- striped skunk
- ruffed grouse
- owl
- grey wolf
- domestic dog
- cougar
- raven
- other
- mule deer

And they grouped all humans except for staff as 'Humans'. Let's look at the species we have in the combined years of data and try to format it the same way.

```
detections_merged %>%

  # group by array and species
```

```r
  group_by(species) %>%
  summarise(n = n()) %>%

  # sort from greatest to least
  arrange(desc(n)) %>%

  # have R print everything
  print(n = nrow(.))
```

```
## # A tibble: 42 x 2
##    species                n
##    <fct>              <int>
##  1 White-tailed deer   6141
##  2 Snowshoe hare       4571
##  3 Red squirrel        2201
##  4 Black bear          1997
##  5 Coyote              1285
##  6 Moose                693
##  7 Unknown              663
##  8 Lynx                 525
##  9 Staff                461
## 10 Unknown deer         340
## 11 Fisher               257
## 12 Grey wolf            224
## 13 Marten               220
## 14 Other birds          219
## 15 Red fox              127
## 16 Caribou              115
## 17 Spruce grouse         93
## 18 Ruffed grouse         90
## 19 Unknown canid         76
## 20 Grey jay              66
## 21 Unknown mustelid      66
## 22 Striped skunk         65
## 23 ATVer                 41
## 24 Cougar                37
## 25 Unknown ungulate      34
## 26 Short-tailed weasel   19
## 27 Long-tailed weasel    17
## 28 Human                 16
## 29 Snowmobiler           16
## 30 Domestic dog          15
## 31 Owl                   15
## 32 Other                 11
## 33 Raven                 10
## 34 Wolverine              8
## 35 Otter                  7
## 36 Porcupine              5
## 37 Beaver                 4
## 38 Canada goose           4
## 39 Mule deer              2
## 40 Arctic hare            1
## 41 Elk                    1
```

```
## 42 Hunter                   1
```

> Hmmm there is one instance of an arctic hare, check that this isn't meant to be a snowshoe hare and fix later if needed.

Now let's create a new data frame (tibble) to work with for the OSM figure summaries specifically

I personally would lump all the unknown together and all the birds together but for the sake of consistency with last year's figures we will remove the same entries and keep the birds separate, let's create a vector of entries to drop

```
# create vector of entries to drop for plots
species_drop <- c('Staff',
                  'Unknown deer',
                  'Unknown ungulate',
                  'Unknown canid',
                  'Unknown mustelid',
                  'Other birds',
                  'Arctic hare')

# now we can create the new data frame with some changes consistent w/ choices made for 2021-2022
detections_merged <- detections_merged %>%

  # for summarizing, lets lump all the recreational humans into "Humans"
  mutate(species = recode_factor(species,
                                 "Snowmobiler" = "Human",
                                 "ATVer" = "Human",
                                 'Hunter' = 'Human')) %>%

  # remove species we don't want to plot
  filter(!species %in% species_drop)

# look at data
str(detections_merged)
```

```
## tibble [19,562 x 8] (S3: tbl_df/tbl/data.frame)
##  $ array   : Factor w/ 6 levels "LU2","LU3","LU01",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ site    : Factor w/ 233 levels "LU2_03","LU2_05",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ species : Factor w/ 39 levels "Human","Arctic hare",..: 33 33 33 33 33 33 33 33 33 33 ...
##  $ datetime: POSIXct[1:19562], format: "2021-07-18 08:59:30" "2021-08-07 04:21:14" ...
##  $ month   : num [1:19562] 7 8 8 8 8 8 8 9 9 9 ...
##  $ year    : num [1:19562] 2021 2021 2021 2021 2021 ...
##  $ timediff: num [1:19562] NA 28519 13230 4290 7337 ...
##  $ event_id: Factor w/ 20759 levels "E1000000","E1000001",..: 3 4 5 6 7 8 9 10 11 12 ...
```

Subset data

We will also want to subset the data by landscape unit (LU) and generate a new data frame for each LU to use for plotting

I'm not great at writing loops, so let's see how this shit goes... probably bad but who knows

```
array_frames <- list()

for (i in unique(detections_merged$array)){

   #Subset data based on radius
  df <- detections_merged %>%
    filter(array == i)

 # list of dataframes
  array_frames <- c(array_frames, list(df))


}

# inspect one data frame
print(array_frames[[1]]) # this is for LU2
```

```
## # A tibble: 4,592 x 8
##    array site   species        datetime            month  year timediff event_id
##    <fct> <fct>  <fct>          <dttm>              <dbl> <dbl>    <dbl> <fct>
##  1 LU2   LU2_03 White-tailed ~ 2021-07-18 08:59:30     7  2021       NA  E1000002
##  2 LU2   LU2_03 White-tailed ~ 2021-08-07 04:21:14     8  2021   28519. E1000003
##  3 LU2   LU2_03 White-tailed ~ 2021-08-16 08:51:21     8  2021   13230. E1000004
##  4 LU2   LU2_03 White-tailed ~ 2021-08-19 08:21:29     8  2021    4290. E1000005
##  5 LU2   LU2_03 White-tailed ~ 2021-08-24 10:39:23     8  2021    7337. E1000006
##  6 LU2   LU2_03 White-tailed ~ 2021-08-26 09:17:02     8  2021    2797. E1000007
##  7 LU2   LU2_03 White-tailed ~ 2021-08-31 19:13:33     8  2021    7796  E1000008
##  8 LU2   LU2_03 White-tailed ~ 2021-09-10 05:03:31     9  2021   13550. E1000009
##  9 LU2   LU2_03 White-tailed ~ 2021-09-16 17:10:41     9  2021    9363. E1000010
## 10 LU2   LU2_03 White-tailed ~ 2021-09-16 19:19:24     9  2021     127  E1000011
## # i 4,582 more rows
```

... I think this worked

Now let's change names of list items using purrr, couldn't figure out how to name them in the loop, you don't necessarily need to do this because we change the names in the next section, but I like having things named

```
array_frames <- array_frames %>%

  purrr::set_names('LU02',
                   'LU03',
                   'LU01',
                   'LU13',
                   'LU15',
                   'LU21')

# inspect each data frame
head(array_frames$LU01)
```

```
## # A tibble: 6 x 8
##   array site   species        datetime            month  year timediff event_id
##   <fct> <fct>  <fct>          <dttm>              <dbl> <dbl>    <dbl> <fct>
```

```
## 1 LU01  LU01_06 White-tailed ~ 2022-06-18 11:09:19     6  2022      NA  E2
## 2 LU01  LU01_06 White-tailed ~ 2022-07-10 13:56:10     7  2022   31847. E3
## 3 LU01  LU01_06 White-tailed ~ 2022-07-25 11:04:44     7  2022   21429. E4
## 4 LU01  LU01_06 White-tailed ~ 2022-07-31 06:38:06     7  2022    8356. E5
## 5 LU01  LU01_06 White-tailed ~ 2022-08-01 09:45:28     8  2022    1627. E6
## 6 LU01  LU01_06 White-tailed ~ 2022-08-01 15:51:01     8  2022     364. E7
```

Detection plots

Detection data

Now we can apply the same data formatting for each LUs' data frame using purrr.

We want to count the number of independent detections per species per LU to use in the detection plots

```r
# apply the same formatting to each LU data frame using purrr map
detection_data <- array_frames %>%

  purrr::map(
    ~.x %>%

      # group by species
      group_by(species) %>%

      # calculate a column with unique accounts of each species
      mutate(count = n_distinct(event_id)) %>%

      # keep just the columns we need
      select(species, count) %>%

      # keep only unique (distinct) rows so we should be left with one row per species, this helps with
      distinct()) %>%

  # set names of list objects
  purrr::set_names('Detections LU02',
                   'Detections LU03',
                   'Detections LU01',
                   'Detections LU13',
                   'Detections LU15',
                   'Detections LU21')
```

Detection plots

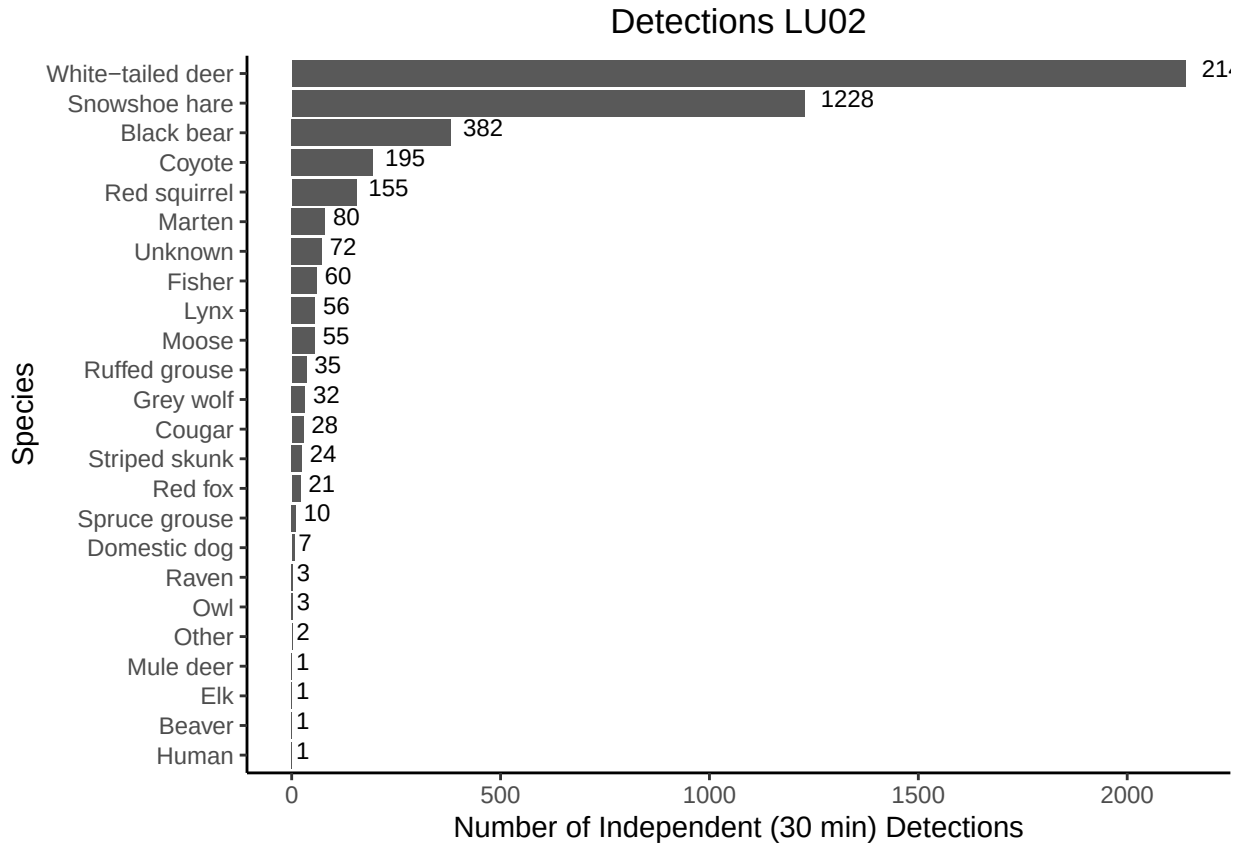
Now to graph independent detections for each LU using purrr, this avoids a TON of code repetition needed to plot each one individually

We use `purrr::imap()` instead of `purrr::map()` because imap maintains the variable names in our list (e.g. Detections LU01, Detections LU13, etc.) which we can then use to title each plot.

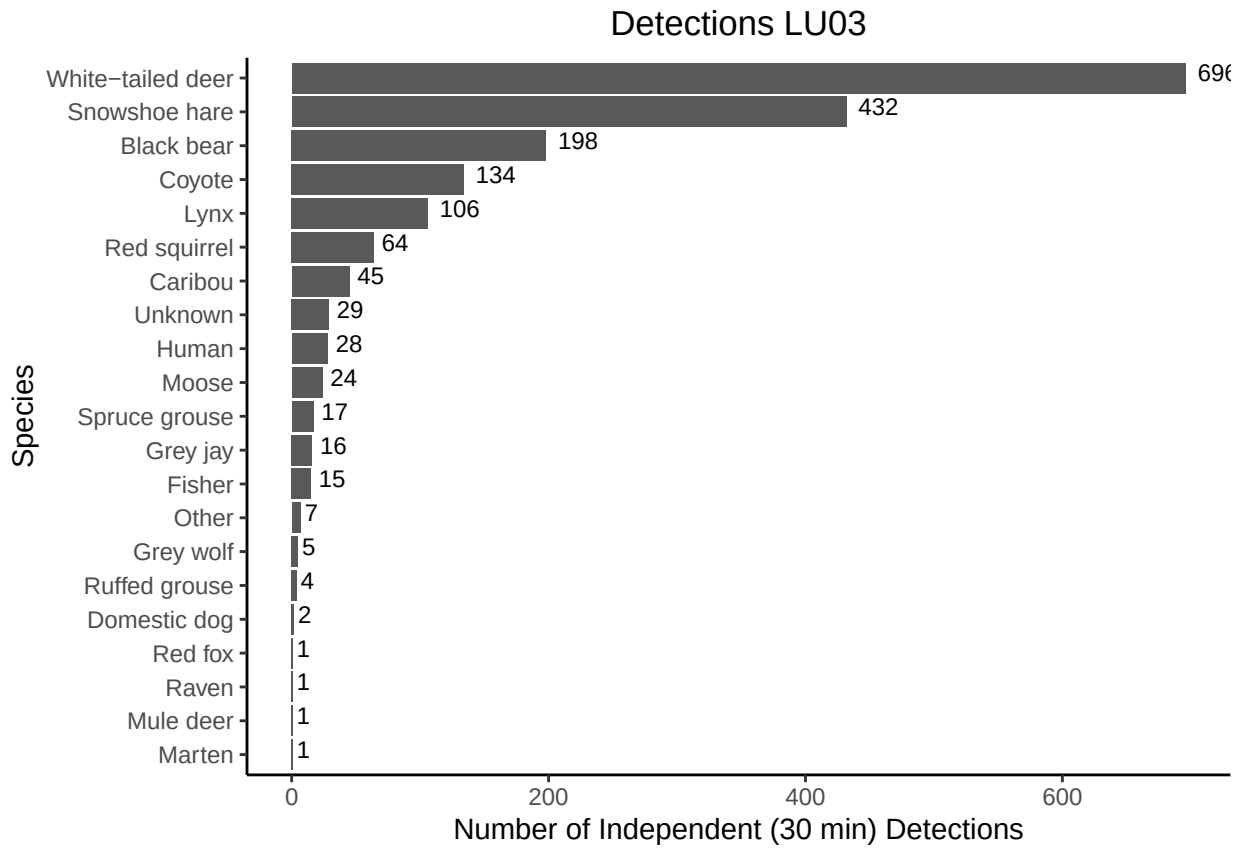
Within `purrr::imap()` we just paste the code we would use for a single ggplot since all the graphical elements (except the title which we change with the file name [.y]) are the same

```r
# create object detection plots which uses the detection_data list (w/ all 4 LUs)
detection_plots <- detection_data %>%

  # use imap instead of map as it allows us to use .y to paste the list element names as the plot title
  purrr::imap(
    ~.x %>%

      # now just copy and paste the ggplot code for the detection graphs
      ggplot(.,
             aes(x = reorder(species, count), y = count)) +

      # plot as bar graph using geom_col so we don't have to provide a y aesthetic
      geom_col() +

      # switch the x and y axis
      coord_flip() +

      # add the number of detections at the end of each bar
      geom_text(aes(label = count),
                color = "black",
                size = 3,
                hjust = -0.3,
                vjust = 0.2) +

      # label x and y axis with informative titles
      labs(x = 'Species',
           y = 'Number of Independent (30 min) Detections') +

      # add title to plot with LU name the .y will take the name of whatever you named each list elemen
      ggtitle(.y) +

      # set the theme
      theme_classic() +
      theme(plot.title = element_text(hjust = 0.5)))

# view plots, this will print each in it's own window so you have to scroll back in the plot viewer pan
detection_plots
```

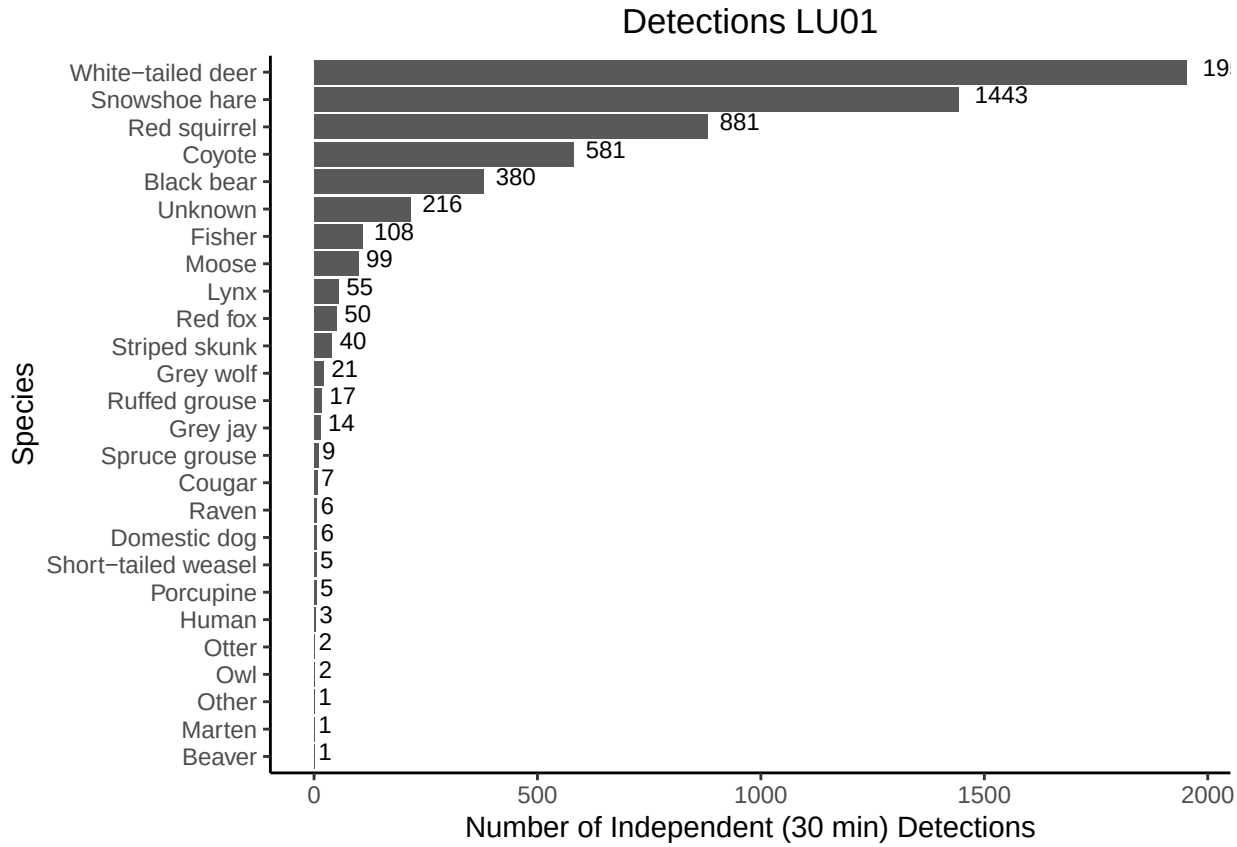
```
## $`Detections LU02`
```



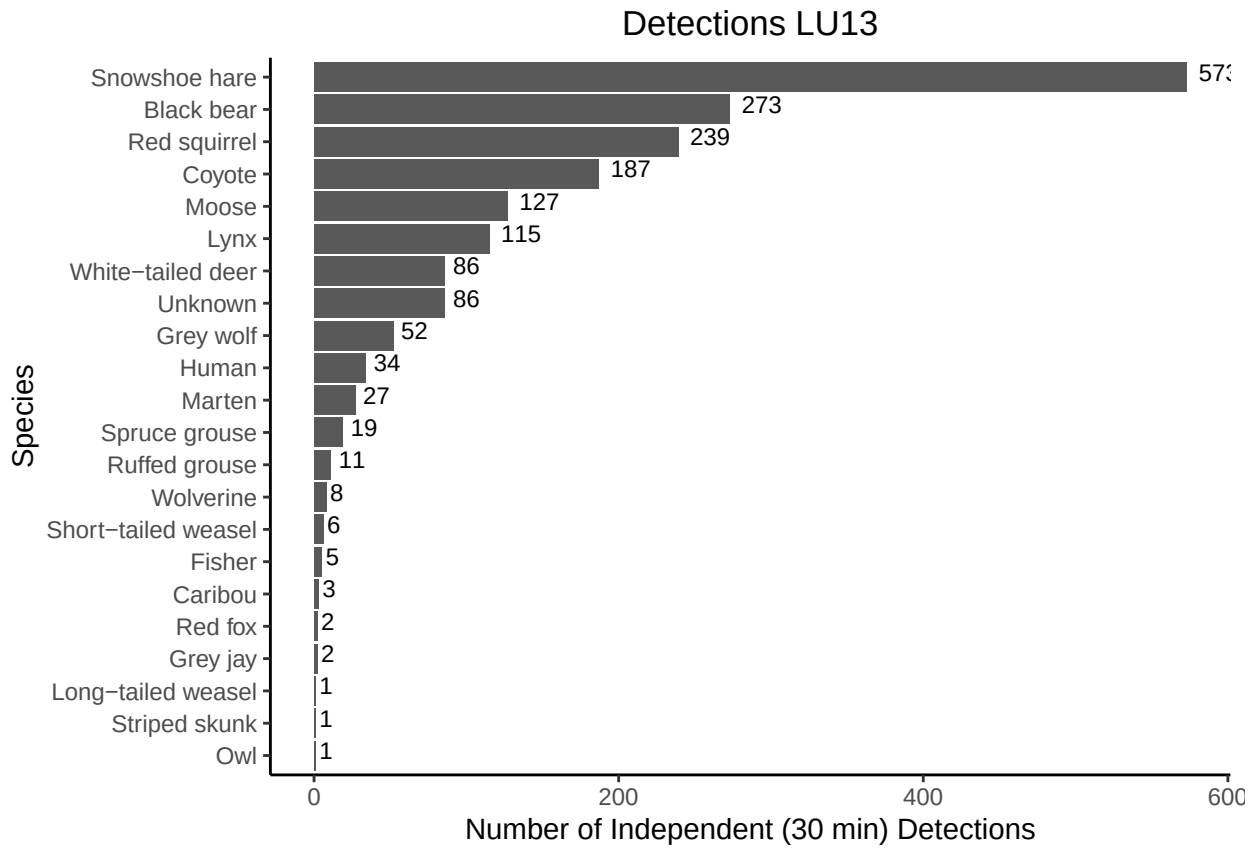

```
##
## $`Detections LU03`
```



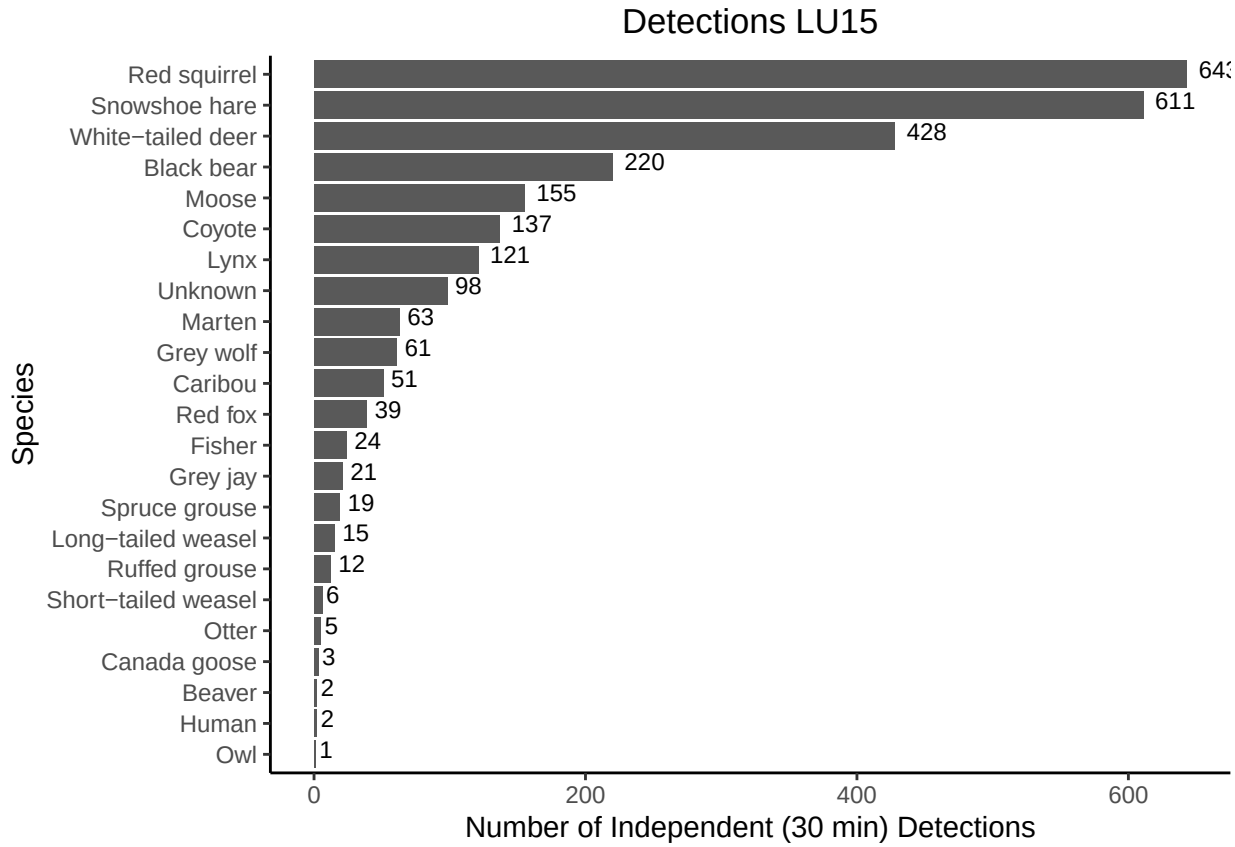

```
##
## $`Detections LU01`
```



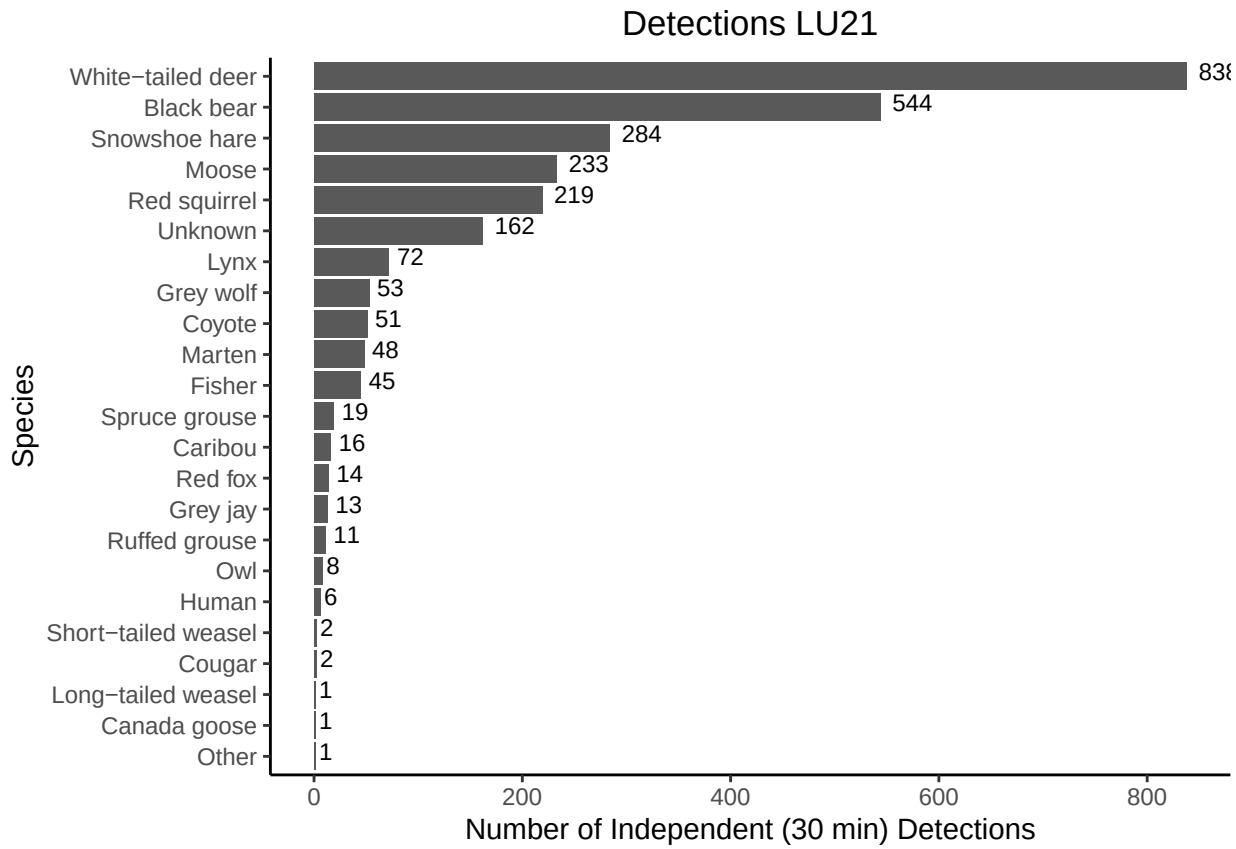

```
##
## $`Detections LU13`
```




```
##
## $`Detections LU15`
```




```
##
## $`Detections LU21`
```



Save detection plots

Now we want to save these plots in case we need each individual one (we will combine the detection and naive occ plots into a single figure for each LU later and use those for the OSM report, but we may want these standalone plots later so let's save them while they are here).

We can save all the plots from the purrr iteration above using `purrr::imap`. imap is used instead of map because it allows us to retain the list object names (plot names) to paste as the file name with the .y command.

> IMPORTANT if you are using this code for a future github repo, DO NOT use .tiff as the file extension. This will cause issues when trying to push any changes to the github repo as the files are too large to meet githubs requirements

```
# save plots only use if needed
purrr::imap(
  detection_plots,
  ~ggsave(.x,
            file = paste0("figures/OSM_",
                          .y,
                          '.jpg'), # avoid using .tiff extension in the github repo, those files are t
          dpi = 600,
          width = 11,
          height = 9,
          units = 'in'))
```

Naive occupancy

Data

We also need to alter the detection data a bit to use for naive occupancy plots.

We will use the individual LU detection data like we did before and use `purrr::map()` to apply the dame data formatting to all 4 data frames.

Here we want to calculate the total number of sites in each LU, the number of sites each species was detected at in each LU and then use both those numbers to calculate naive occupancy for each species in each LU

```
# First we need to alter the data frame a bit for these plots, let's create a data frame for each LU (I


# apply the same formatting to each data frame using purrr
occupancy_data <- array_frames %>%

  purrr::map(
    ~.x %>%

      # calculate the total number of sites for each LU
      mutate(total_sites = n_distinct(site)) %>%

      # group by species to calculate the number of sites each spp occurred at
      group_by(species) %>%

      # add columns to count the number of sites each spp occurred at and then the naive occupancy
  reframe(count = n_distinct(site),
          naive_occ = count/total_sites,
          ind_det = n_distinct(event_id)) %>%

    # keep just the columns we need
  select(species, naive_occ, ind_det) %>%

    # keep only unique (distinct) rows so we should be left with one row per species, this helps with p
  distinct()) %>%

  purrr::set_names('Naive Occupancy LU02',
                   'Naive Occupancy LU03',
                   'Naive Occupancy LU01',
                   'Naive Occupancy LU13',
                   'Naive Occupancy LU15',
                   'Naive Occupancy LU21')
```

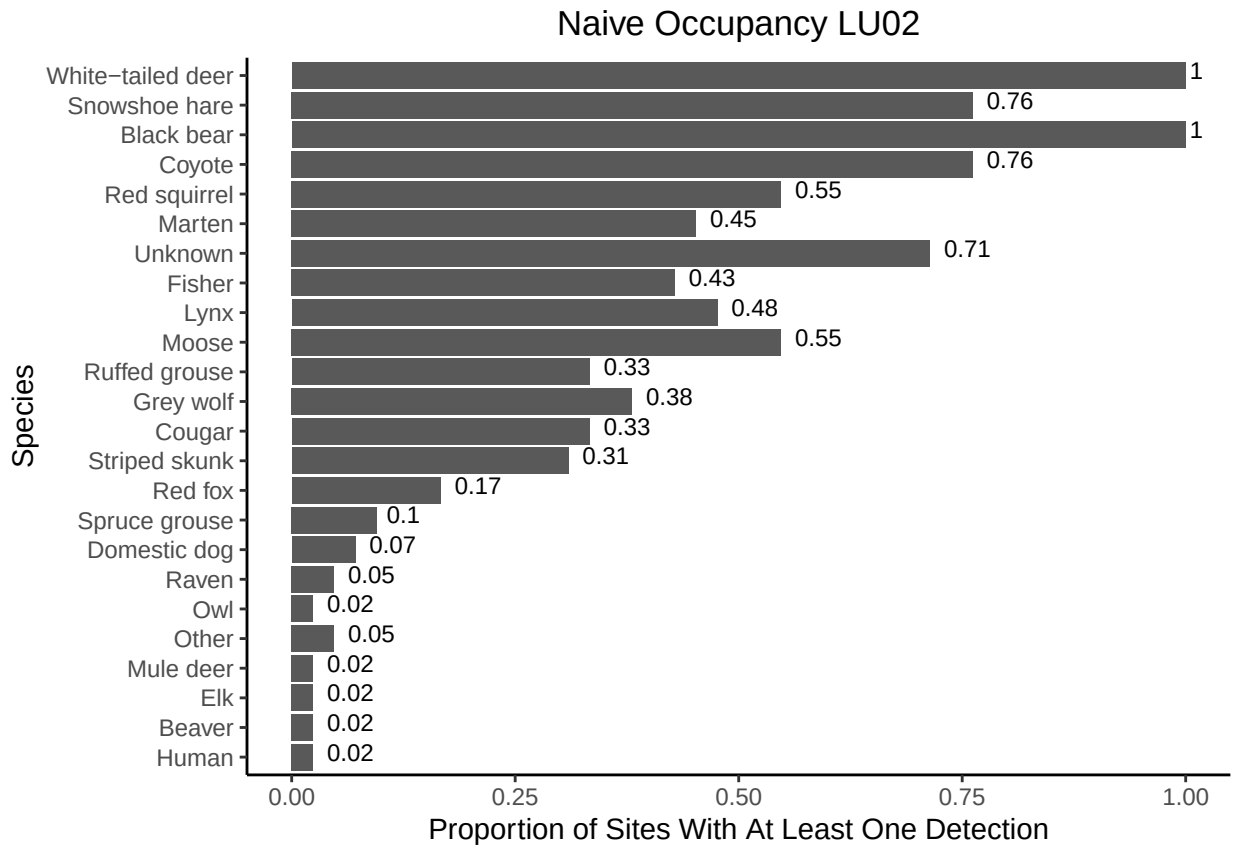
Occupancy plots

Now we can graph naive occupancy for each LU using purrr, and as with the detection plots this saves a massive amount of coding using purrr to run an iteration on the data files and produce four plots at once instead of copying and pasting code for each individually

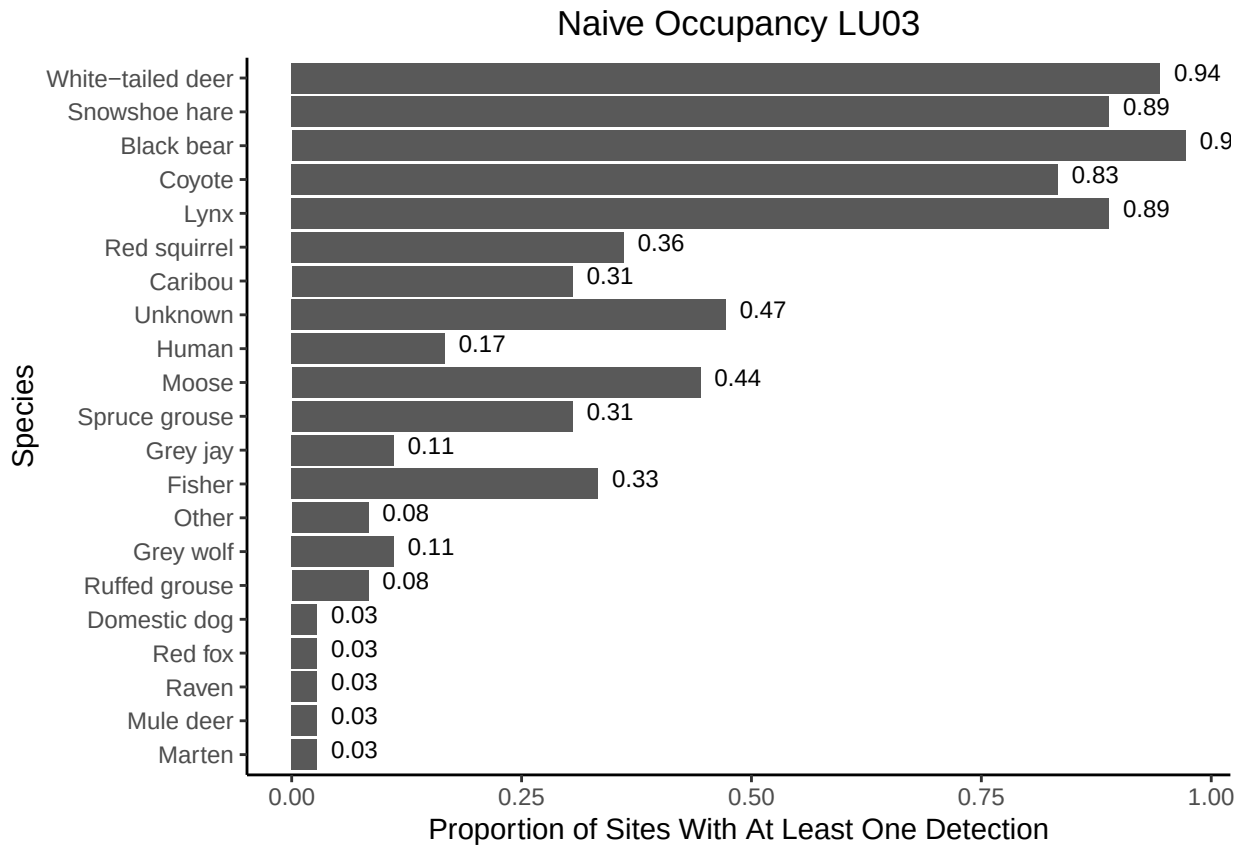
```
# create object occupancy_plots which uses the occupancy_data list (w/ all 4 LUs)
occupancy_plots <- occupancy_data %>%
```

```r
  # use imap instead of map as it allows us to use .y to paste the list element names as the plot title
  purrr::imap(
    ~.x %>%

      # now just copy and paste the ggplot code for the occupancy graphs
      ggplot(.,
             aes(x = fct_reorder(species,
                                 ind_det), # this reorders the species so they match the order of the d
                 y = naive_occ)) +

      # plot as bars using geom_col() which uses stat = 'identity', instead of geom_bar() which will co
      geom_col() +

      # flip x and y axis
      coord_flip() +

      # add text to end of bars that provides naive occ value
      geom_text(aes(label = round(naive_occ, 2)),
                size = 3,
                hjust = -0.3,
                vjust = 0.2) +

      # relabel x and y axis and title
      labs(x = 'Species',
           y = 'Proportion of Sites With At Least One Detection') +

      # set plot title using .y (name of list object)
      ggtitle(.y) +

      # set. theme elements
      theme_classic()+
      theme(plot.title = element_text(hjust = 0.5)))

# view plots
occupancy_plots
```

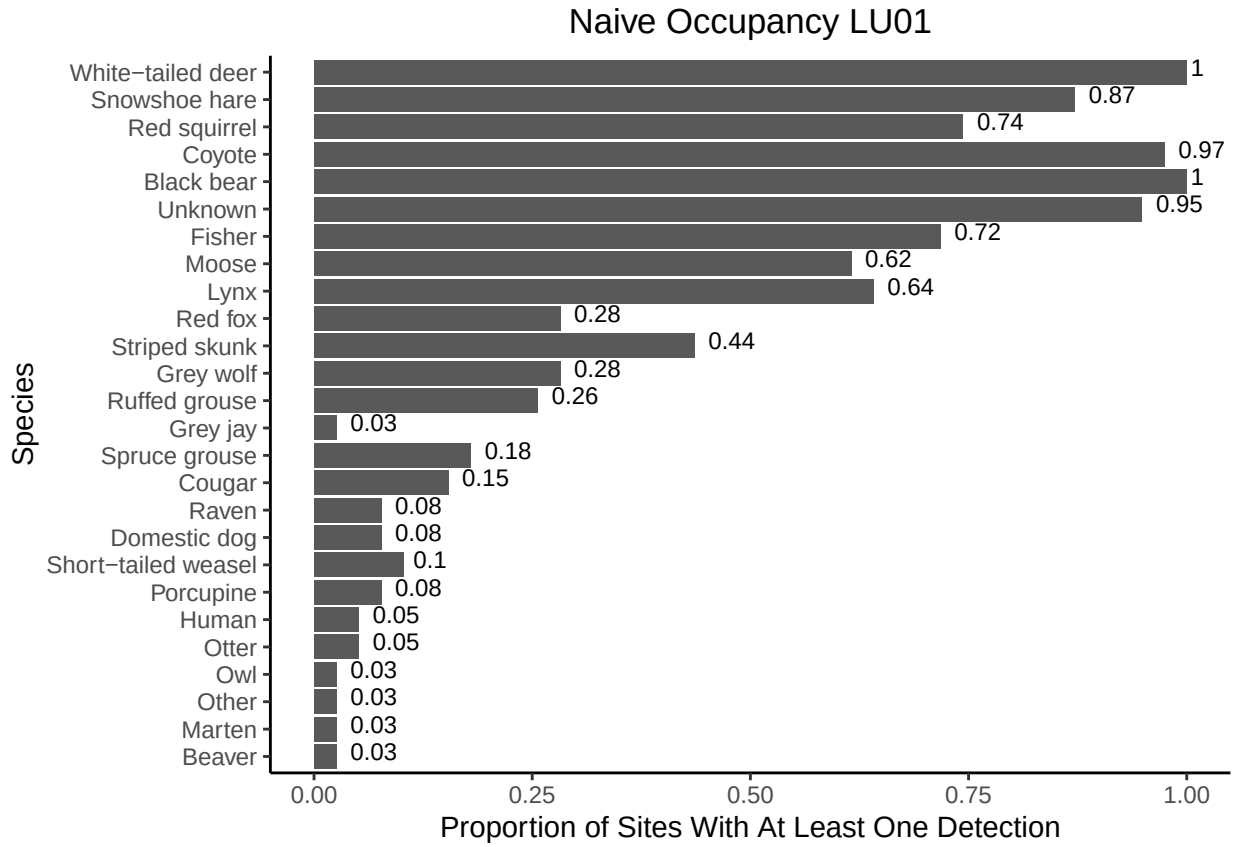
```
## $`Naive Occupancy LU02`
```



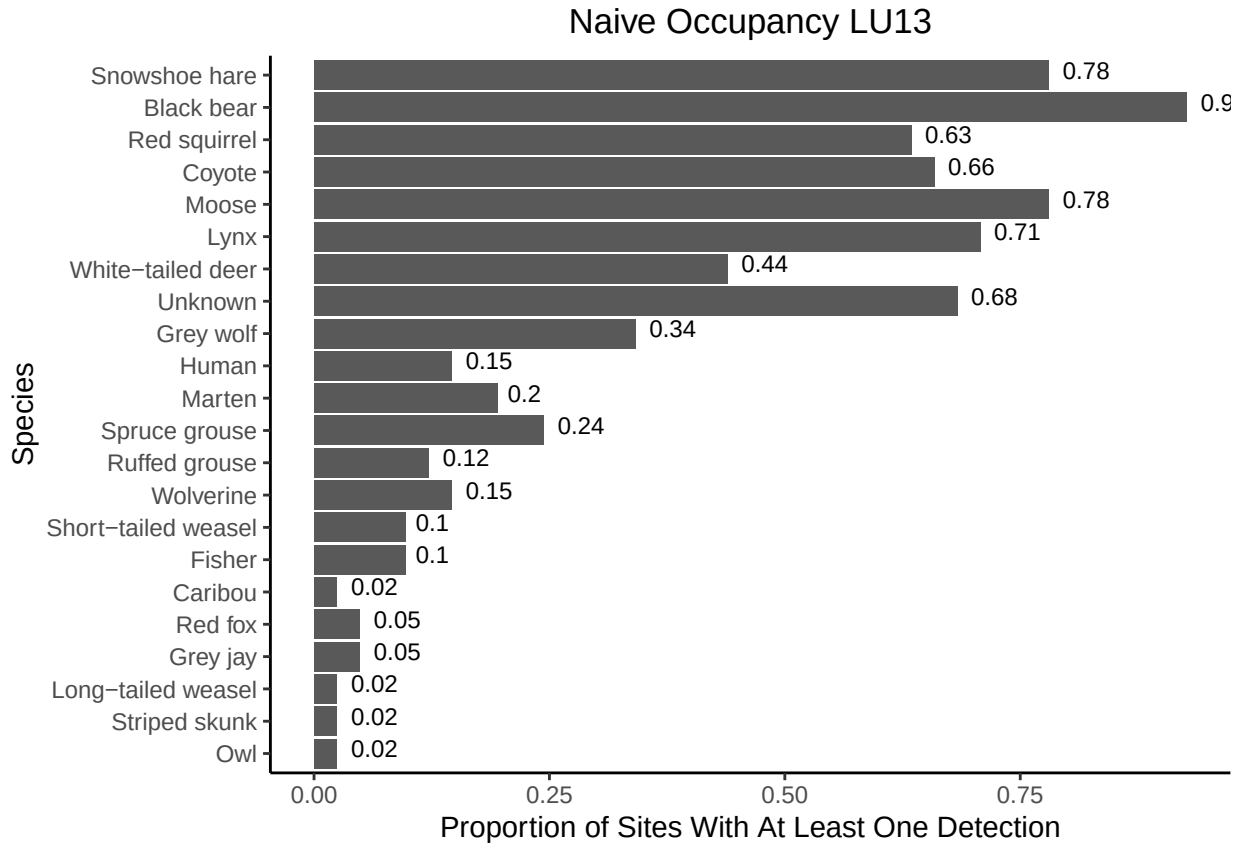

```
##
## $`Naive Occupancy LU03`
```



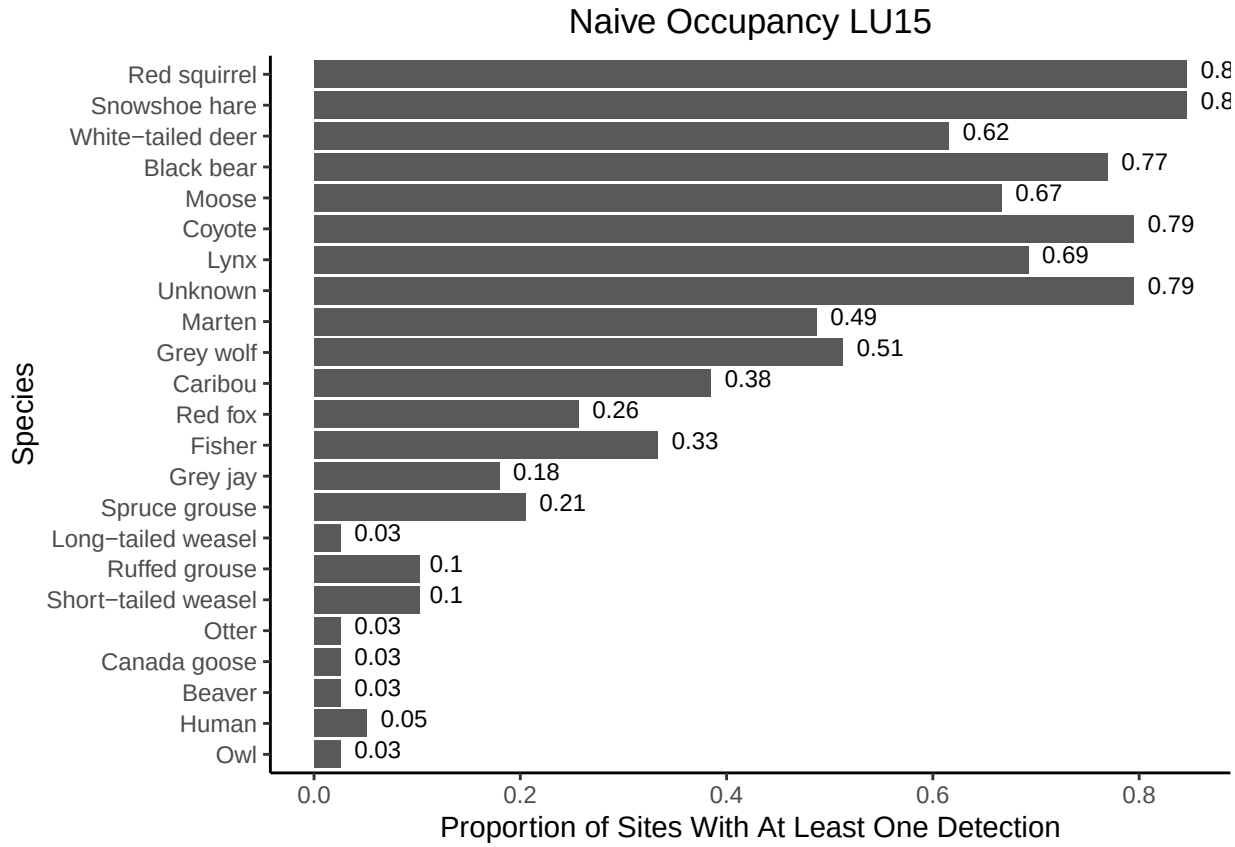

```
##
## $`Naive Occupancy LU01`
```



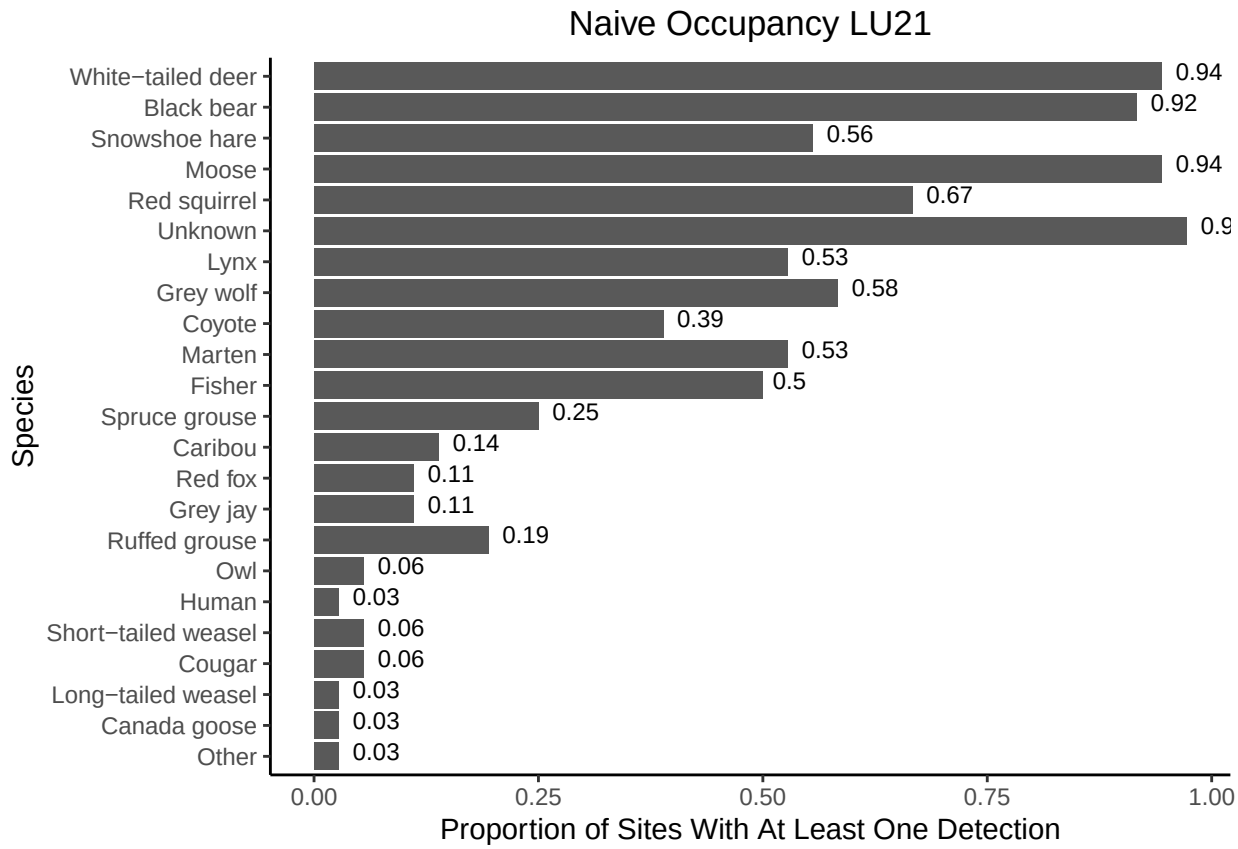

```
##
## $‘Naive Occupancy LU13‘
```




```
##
## $‘Naive Occupancy LU15‘
```




```
##
## $`Naive Occupancy LU21`
```



Save occupancy plots

As with the detection plots, we might want these individual plots later for something so we can use `purrr::imap()` to save them to the figures folder

Again avoid using the .tiff extension in github

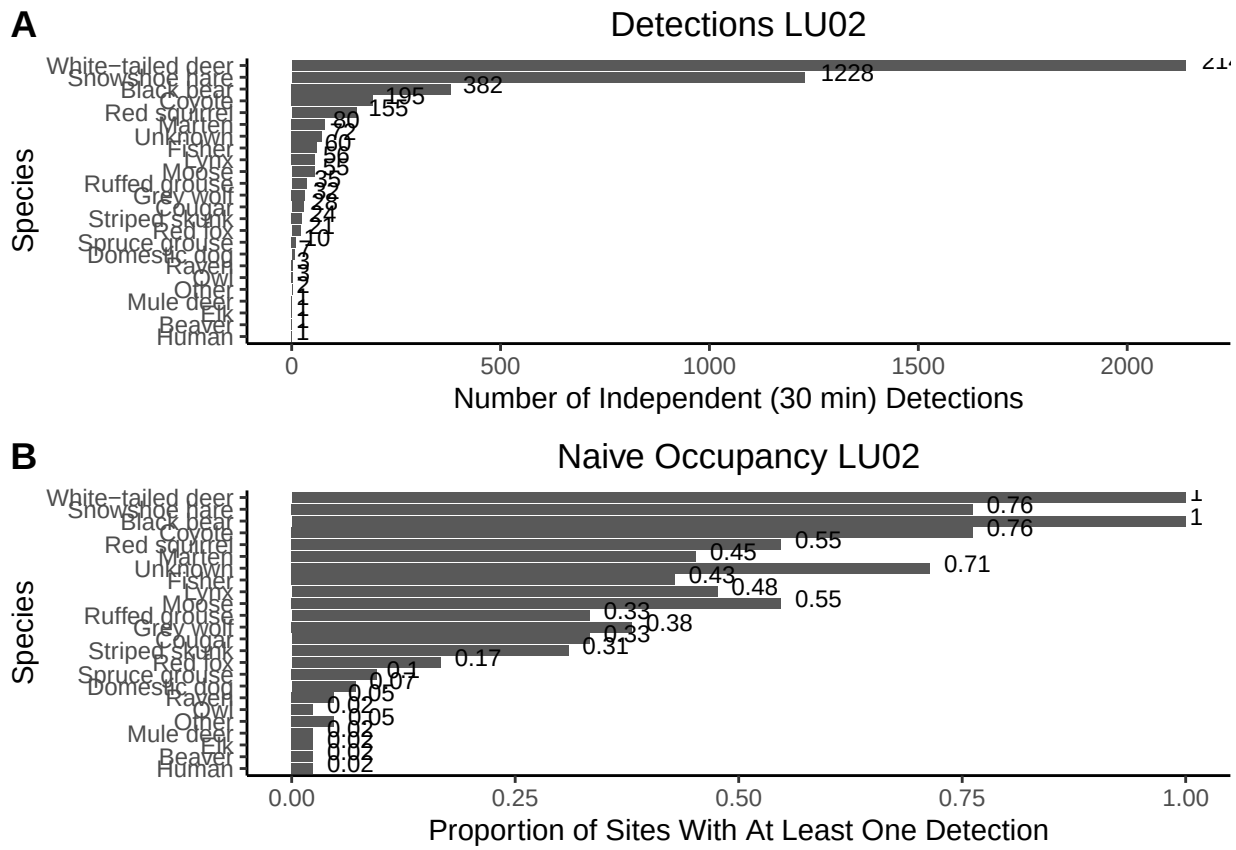
```
# save plots
purrr::imap(
  occupancy_plots,
  ~ggsave(.x,
          file = paste0("figures/OSM_",
                        .y,
                        '.jpg'), # avoid using .tiff extension in the github repo, those files are too
          dpi = 600,
          width = 11,
          height = 9,
          units = 'in'))
```

Final combined plots for report

The previous year's report had a figure for each LU with the detections plot on the top and the occupancy plot on the bottom so we will recreate these for this year using `ggarrange()`.

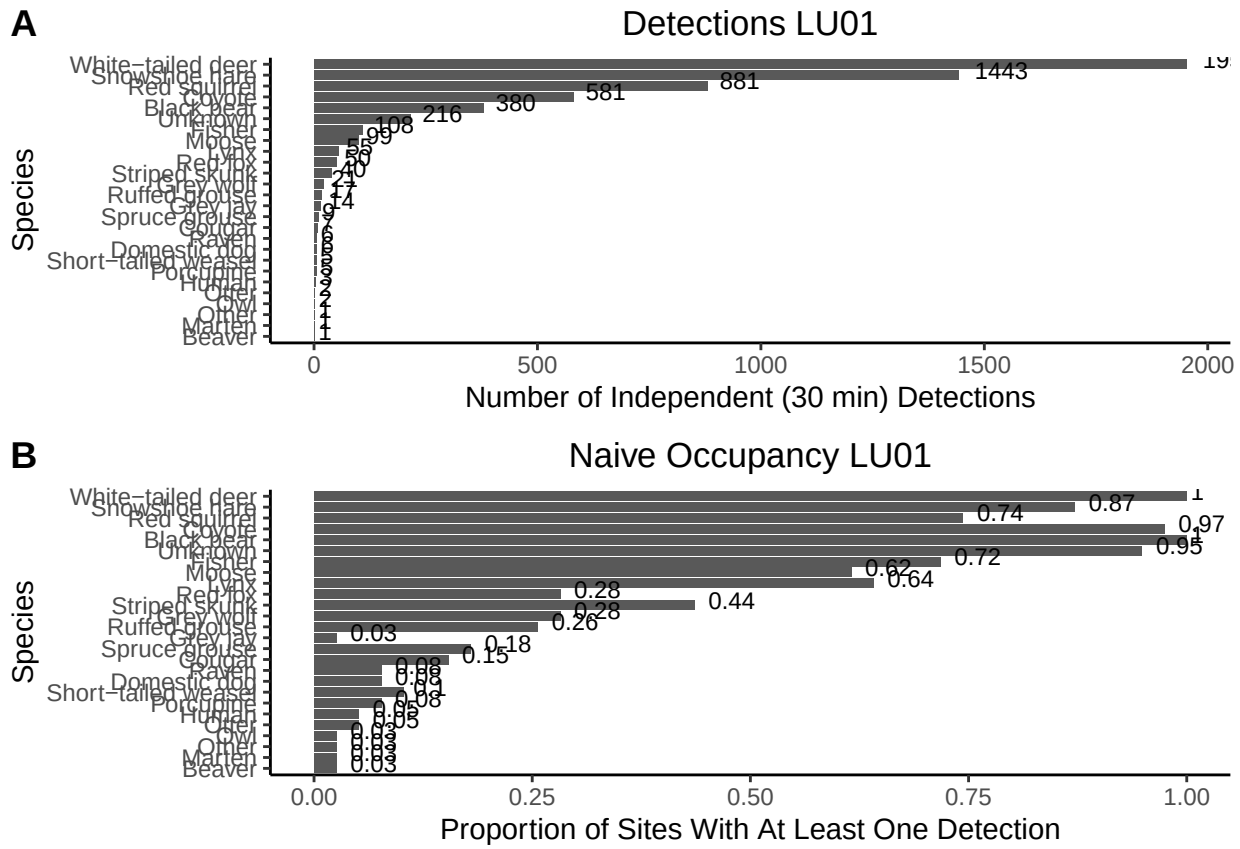
Unfortunately I could not figure out how to do this in purrr to reduce coding but luckily it isn't too much repetition

```
# not sure I know how to do the following section in purrr just yet, but we've saved a ton of coding so

# LU2
LU02_det_occ_plots <- ggarrange(detection_plots$`Detections LU02`,
                                occupancy_plots$`Naive Occupancy LU02`,
                                labels = c("A", "B"),
                                nrow = 2)

# view plot
LU02_det_occ_plots
```

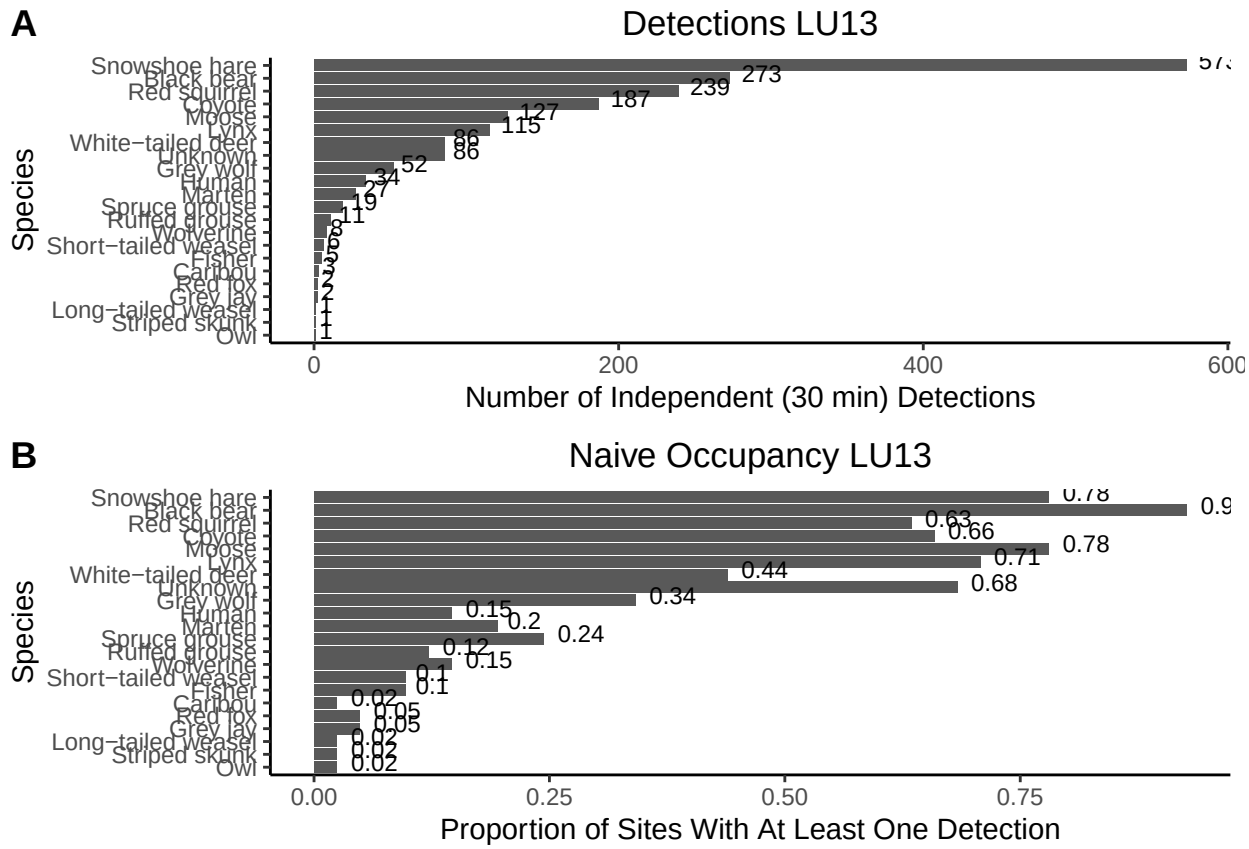



```
# LU3
LU03_det_occ_plots <- ggarrange(detection_plots$`Detections LU03`,
                                occupancy_plots$`Naive Occupancy LU03`,
                                labels = c("A", "B"),
                                nrow = 2)

# view plot
LU03_det_occ_plots
```

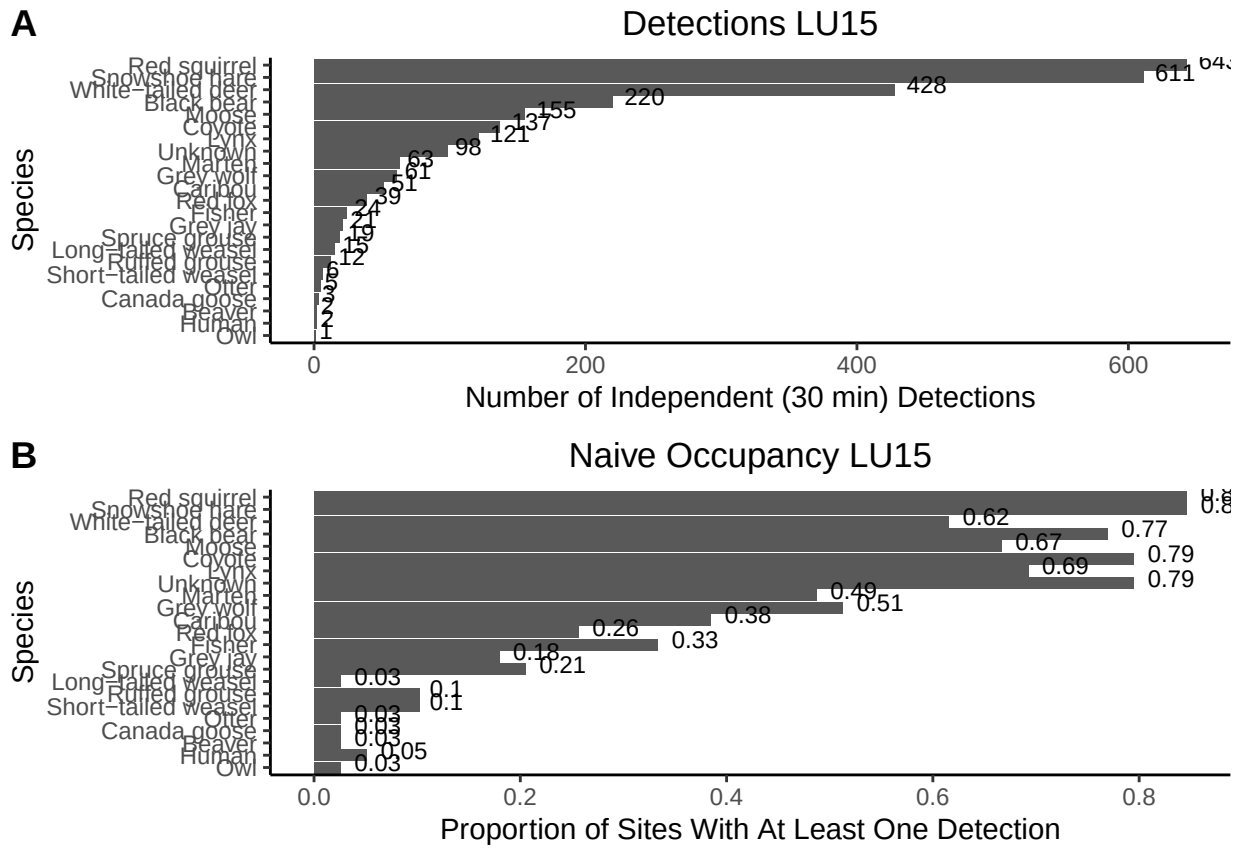

```
# LU1

# arrange the plots so each LU has a figure with detections on top and naive occ on bottom
LU01_det_occ_plots <- ggarrange(detection_plots$`Detections LU01`,
                                occupancy_plots$`Naive Occupancy LU01`,
                                labels = c("A", "B"),
                                nrow = 2)

# view plot
LU01_det_occ_plots
```



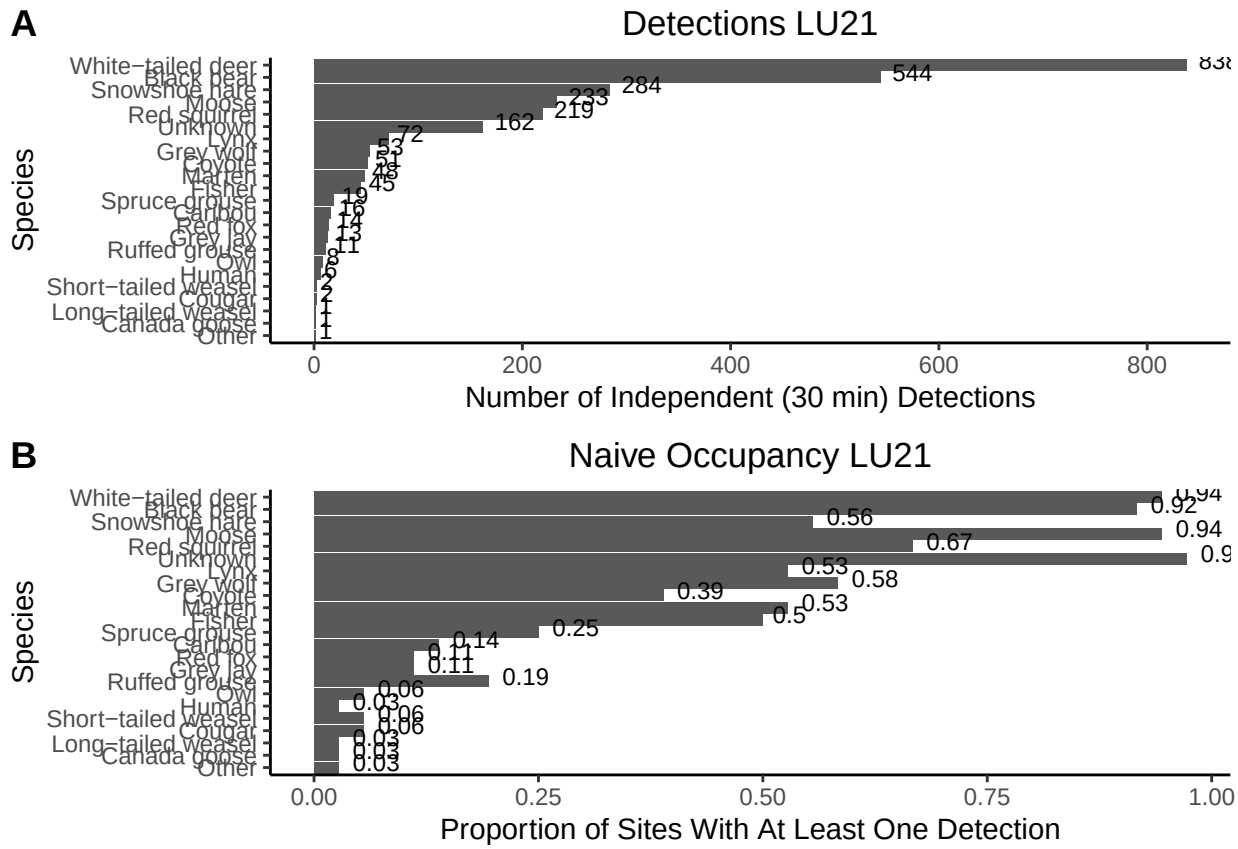

```
# LU13

# arrange the plots so each LU has a figure with detections on top and naive occ on bottom
LU13_det_occ_plots <- ggarrange(detection_plots$`Detections LU13`,
                                occupancy_plots$`Naive Occupancy LU13`,
                                labels = c("A", "B"),
                                nrow = 2)

# view plot
LU13_det_occ_plots
```




```
# LU15

# arrange the plots so each LU has a figure with detections on top and naive occ on bottom
LU15_det_occ_plots <- ggarrange(detection_plots$`Detections LU15`,
                                occupancy_plots$`Naive Occupancy LU15`,
                                labels = c("A", "B"),
                                nrow = 2)

# view plot
LU15_det_occ_plots
```




```
# LU21

# arrange the plots so each LU has a figure with detections on top and naive occ on bottom
LU21_det_occ_plots <- ggarrange(detection_plots$`Detections LU21`,
                                occupancy_plots$`Naive Occupancy LU21`,
                                labels = c("A", "B"),
                                nrow = 2)

# view plot
LU21_det_occ_plots
```



We can however, save all the figures again using purrr

```
# save all figures at once using purrr
final_det_occ_plots <- list(LU02_det_occ_plots,
                            LU03_det_occ_plots,
                            LU01_det_occ_plots,
                            LU13_det_occ_plots,
                            LU15_det_occ_plots,
                            LU21_det_occ_plots) %>%


  purrr::set_names('LU02_det_occ_plots',
                   'LU03_det_occ_plots',
                   'LU01_det_occ_plots',
                   'LU13_det_occ_plots',
                   'LU15_det_occ_plots',
                   'LU21_det_occ_plots') %>%

  purrr::imap(
    ~ggsave(.x,
            file = paste0("figures/OSM_",
                          .y,
                          '.jpg'), # avoid using .tiff extension in the github repo, those files are to
            dpi = 600,
            width = 12,
            height = 15,
```

```
            units = 'in'))
```

Finish with detection data

Before proceeding let's clear the objects currently in our environment since we don't need them for the analysis

```
rm(list = ls(all.names = TRUE))
```

Global Analysis prep

Now we can start the analysis prep.

First we need to read in the proportional detection (response metrics) and covariate (explanatory metrics) data files for all 6 LUs (fiscal years 2021-2022 and 2022-2023)

Response metrics

Read in data files

We haven't joined the response metrics from 2021-2022 and 2022-2023 in any previous scripts yet, so let's read those in using *purrr* and then join them and take a look at the data to make sure it looks good.

```
# response metric (proportional detections from the from the ACME_camera_script_9-2-2024.R or .Rmd)

prop_detections <-  file.path('data/processed',

                        c('OSM_proportional_detections_2021.csv',
                          'OSM_proportional_detections_2022.csv')) %>%

  map(~.x %>%
        read_csv(.,

                 # set the column types to read in correctly
                col_types = cols(site = col_factor(),
                                 .default = col_number()))) %>%

  # give names to each data frame in list
  purrr::set_names('prop_dets_2021',
                   'prop_dets_2022') # R doesn't like when they are just numbers, you can make it work

# check variable structure
str(prop_detections)
```

```
## List of 2
##  $ prop_dets_2021: spc_tbl_ [78 x 23] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##   ..$ site                   : Factor w/ 78 levels "LU2_03","LU2_05",..: 1 2 3 4 5 6 7 8 9 10 ...
##   ..$ black_bear             : num [1:78] 2 2 1 3 2 3 4 3 1 2 ...
##   ..$ coyote                 : num [1:78] 1 0 3 5 6 3 8 0 3 4 ...
##   ..$ fisher                 : num [1:78] 2 0 3 1 0 2 1 1 1 0 ...
```

```
##   ..$ snowshoe_hare           : num [1:78] 1 0 2 5 6 9 14 0 7 3 ...
##   ..$ white-tailed_deer       : num [1:78] 7 3 6 7 6 7 9 6 5 13 ...
##   ..$ cougar                  : num [1:78] 0 1 0 0 0 0 1 0 0 1 ...
##   ..$ lynx                    : num [1:78] 0 0 3 2 0 4 6 0 1 0 ...
##   ..$ red_fox                 : num [1:78] 0 0 3 0 0 0 0 1 0 0 ...
##   ..$ moose                   : num [1:78] 0 0 0 0 1 0 3 3 0 1 ...
##   ..$ grey_wolf               : num [1:78] 0 0 0 0 0 0 0 0 1 0 ...
##   ..$ caribou                 : num [1:78] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ absent_black_bear       : num [1:78] 3 3 4 2 3 8 7 2 4 9 ...
##   ..$ absent_coyote           : num [1:78] 6 7 4 2 1 11 6 7 4 10 ...
##   ..$ absent_fisher           : num [1:78] 5 7 4 6 7 12 13 6 6 14 ...
##   ..$ absent_snowshoe_hare    : num [1:78] 6 7 5 2 1 5 0 7 0 11 ...
##   ..$ absent_white-tailed_deer: num [1:78] 0 4 1 0 1 7 5 1 2 1 ...
##   ..$ absent_cougar           : num [1:78] 7 6 7 7 7 14 13 7 7 13 ...
##   ..$ absent_lynx             : num [1:78] 7 7 4 5 7 10 8 7 6 14 ...
##   ..$ absent_red_fox          : num [1:78] 7 7 4 7 7 14 14 6 7 14 ...
##   ..$ absent_moose            : num [1:78] 7 7 7 7 6 14 11 4 7 13 ...
##   ..$ absent_grey_wolf        : num [1:78] 7 7 7 7 7 14 14 7 6 14 ...
##   ..$ absent_caribou          : num [1:78] 7 7 7 7 7 14 14 7 7 14 ...
##   ..- attr(*, "spec")=
##   .. .. cols(
##   .. ..   .default = col_number(),
##   .. ..   site = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
##   .. ..   black_bear = col_number(),
##   .. ..   coyote = col_number(),
##   .. ..   fisher = col_number(),
##   .. ..   snowshoe_hare = col_number(),
##   .. ..   `white-tailed_deer` = col_number(),
##   .. ..   cougar = col_number(),
##   .. ..   lynx = col_number(),
##   .. ..   red_fox = col_number(),
##   .. ..   moose = col_number(),
##   .. ..   grey_wolf = col_number(),
##   .. ..   caribou = col_number(),
##   .. ..   absent_black_bear = col_number(),
##   .. ..   absent_coyote = col_number(),
##   .. ..   absent_fisher = col_number(),
##   .. ..   absent_snowshoe_hare = col_number(),
##   .. ..   `absent_white-tailed_deer` = col_number(),
##   .. ..   absent_cougar = col_number(),
##   .. ..   absent_lynx = col_number(),
##   .. ..   absent_red_fox = col_number(),
##   .. ..   absent_moose = col_number(),
##   .. ..   absent_grey_wolf = col_number(),
##   .. ..   absent_caribou = col_number()
##   .. .. )
##   ..- attr(*, "problems")=<externalptr>
##  $ prop_dets_2022: spc_tbl_ [154 x 27] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##   ..$ site                    : Factor w/ 154 levels "LU01_06","LU01_10",..: 1 2 3 4 5 6 7 8 9 10 ..
##   ..$ black_bear              : num [1:154] 7 3 4 7 8 9 4 5 7 7 ...
##   ..$ coyote                  : num [1:154] 4 4 8 10 11 9 11 0 9 4 ...
##   ..$ fisher                  : num [1:154] 4 3 3 3 2 1 1 2 0 3 ...
##   ..$ moose                   : num [1:154] 3 2 5 9 1 0 2 4 1 0 ...
##   ..$ red_squirrel            : num [1:154] 3 0 8 0 10 1 15 0 9 3 ...
```

```
##   ..$ snowshoe_hare           : num [1:154] 4 1 3 0 8 2 2 0 12 4 ...
##   ..$ white-tailed_deer       : num [1:154] 12 5 12 12 13 14 15 9 12 10 ...
##   ..$ cougar                  : num [1:154] 0 0 1 0 1 0 0 0 0 0 ...
##   ..$ grey_wolf               : num [1:154] 0 0 2 0 0 0 1 0 0 0 ...
##   ..$ lynx                    : num [1:154] 0 0 1 0 1 1 0 0 0 2 ...
##   ..$ red_fox                 : num [1:154] 0 0 2 0 0 0 0 0 4 0 ...
##   ..$ wolverine               : num [1:154] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ caribou                 : num [1:154] 0 0 0 0 0 0 0 0 0 0 ...
##   ..$ absent_black_bear       : num [1:154] 5 3 8 5 4 3 8 7 5 5 ...
##   ..$ absent_coyote           : num [1:154] 10 1 6 5 3 5 4 15 6 11 ...
##   ..$ absent_fisher           : num [1:154] 10 2 11 12 12 13 14 13 15 12 ...
##   ..$ absent_moose            : num [1:154] 11 3 9 6 13 14 13 11 14 15 ...
##   ..$ absent_red_squirrel     : num [1:154] 11 5 6 15 4 13 0 15 6 12 ...
##   ..$ absent_snowshoe_hare    : num [1:154] 10 4 11 15 6 12 13 15 3 11 ...
##   ..$ absent_white-tailed_deer: num [1:154] 2 0 2 3 1 0 0 6 3 5 ...
##   ..$ absent_cougar           : num [1:154] 14 5 13 15 13 14 15 15 15 15 ...
##   ..$ absent_grey_wolf        : num [1:154] 14 5 12 15 14 14 14 15 15 15 ...
##   ..$ absent_lynx             : num [1:154] 14 5 13 15 13 13 15 15 15 13 ...
##   ..$ absent_red_fox          : num [1:154] 14 5 12 15 14 14 15 15 11 15 ...
##   ..$ absent_wolverine        : num [1:154] 14 5 14 15 14 14 15 15 15 15 ...
##   ..$ absent_caribou          : num [1:154] 14 5 14 15 14 14 15 15 15 15 ...
##   ..- attr(*, "spec")=
##   .. .. cols(
##   .. ..   .default = col_number(),
##   .. ..   site = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
##   .. ..   black_bear = col_number(),
##   .. ..   coyote = col_number(),
##   .. ..   fisher = col_number(),
##   .. ..   moose = col_number(),
##   .. ..   red_squirrel = col_number(),
##   .. ..   snowshoe_hare = col_number(),
##   .. ..   `white-tailed_deer` = col_number(),
##   .. ..   cougar = col_number(),
##   .. ..   grey_wolf = col_number(),
##   .. ..   lynx = col_number(),
##   .. ..   red_fox = col_number(),
##   .. ..   wolverine = col_number(),
##   .. ..   caribou = col_number(),
##   .. ..   absent_black_bear = col_number(),
##   .. ..   absent_coyote = col_number(),
##   .. ..   absent_fisher = col_number(),
##   .. ..   absent_moose = col_number(),
##   .. ..   absent_red_squirrel = col_number(),
##   .. ..   absent_snowshoe_hare = col_number(),
##   .. ..   `absent_white-tailed_deer` = col_number(),
##   .. ..   absent_cougar = col_number(),
##   .. ..   absent_grey_wolf = col_number(),
##   .. ..   absent_lynx = col_number(),
##   .. ..   absent_red_fox = col_number(),
##   .. ..   absent_wolverine = col_number(),
##   .. ..   absent_caribou = col_number()
##   .. .. )
##   ..- attr(*, "problems")=<externalptr>
```

Merge data files

Now we need to merge the two data files for analysis. We can do this with *dplyr*.

```
# merge the proportional detections files so there are rows for both fiscal years
prop_dets_all <- dplyr::bind_rows(prop_detections$prop_dets_2021,
                                  prop_detections$prop_dets_2022)

print(prop_dets_all)
```

```
## # A tibble: 232 x 27
##    site  black_bear coyote fisher snowshoe_hare `white-tailed_deer` cougar  lynx
##    <fct>      <dbl>  <dbl>  <dbl>         <dbl>                <dbl>  <dbl> <dbl>
##  1 LU2_~          2      1      2             1                    7      0     0
##  2 LU2_~          2      0      0             0                    3      1     0
##  3 LU2_~          1      3      3             2                    6      0     3
##  4 LU2_~          3      5      1             5                    7      0     2
##  5 LU2_~          2      6      0             6                    6      0     0
##  6 LU2_~          3      3      2             9                    7      0     4
##  7 LU2_~          4      8      1            14                    9      1     6
##  8 LU2_~          3      0      1             0                    6      0     0
##  9 LU2_~          1      3      1             7                    5      0     1
## 10 LU2_~          2      4      0             3                   13      1     0
## # i 222 more rows
## # i 19 more variables: red_fox <dbl>, moose <dbl>, grey_wolf <dbl>,
## #   caribou <dbl>, absent_black_bear <dbl>, absent_coyote <dbl>,
## #   absent_fisher <dbl>, absent_snowshoe_hare <dbl>,
## #   `absent_white-tailed_deer` <dbl>, absent_cougar <dbl>, absent_lynx <dbl>,
## #   absent_red_fox <dbl>, absent_moose <dbl>, absent_grey_wolf <dbl>,
## #   absent_caribou <dbl>, red_squirrel <dbl>, wolverine <dbl>, ...
```

Format data

This looks good except since there were no wolverines in the first fiscal year of monitoring (LU02 and LU03) those columns have NAs for both arrays, we want to replace those NAs with Zeros and move the wolverine column to the correct location

Let's do that now.

```
prop_dets_all <- prop_dets_all %>%

   # replace NAs introduced from joining data to zeros
  replace(is.na(.),
          0) %>%

  # relocate wolverine column
  relocate(.,
           wolverine,
           .after = caribou)

# check data
head(prop_dets_all)
```

```
## # A tibble: 6 x 27
##   site   black_bear coyote fisher snowshoe_hare `white-tailed_deer` cougar  lynx
##   <fct>       <dbl>  <dbl>  <dbl>         <dbl>                <dbl>  <dbl> <dbl>
## 1 LU2_03          2      1      2             1                    7      0     0
## 2 LU2_05          2      0      0             0                    3      1     0
## 3 LU2_1~          1      3      3             2                    6      0     3
## 4 LU2_1~          3      5      1             5                    7      0     2
## 5 LU2_1~          2      6      0             6                    6      0     0
## 6 LU2_1~          3      3      2             9                    7      0     4
## # i 19 more variables: red_fox <dbl>, moose <dbl>, grey_wolf <dbl>,
## #   caribou <dbl>, wolverine <dbl>, absent_black_bear <dbl>,
## #   absent_coyote <dbl>, absent_fisher <dbl>, absent_snowshoe_hare <dbl>,
## #   `absent_white-tailed_deer` <dbl>, absent_cougar <dbl>, absent_lynx <dbl>,
## #   absent_red_fox <dbl>, absent_moose <dbl>, absent_grey_wolf <dbl>,
## #   absent_caribou <dbl>, red_squirrel <dbl>, absent_red_squirrel <dbl>,
## #   absent_wolverine <dbl>
```

Looks good!

Save data

Let's save the merged and formatted detection data from 2021 and 2022 for future use

```
write_csv(prop_dets_all,
          'data/processed/OSM_proportional_detections_merged_2021_2022.csv')
```

Covariates

In the previous script, 2_ACME_landscape_covariate_exploration_script.Rmd we joined the two fiscal years of data and grouped the covariates for analysis and saved this data as a csv file, so we can read in this file now and we shouldn't have to do any further formatting at the moment

Read data

We will check the data structure after reading in the file just to make sure everything looks good.

```
covariates_all <- read_csv('data/processed/OSM_covariates_grouped_2021_2022.csv',

                           # set the column types to read in correctly
                           col_types = cols(array = col_factor(),
                                            camera = col_factor(),
                                            site = col_factor(),
                                            buff_dist = col_factor(),
                                            .default = col_number()))
```

```
## Warning: The following named parsers don't match the column names: camera
```

```
str(covariates_all)
```

```
## spc_tbl_ [4,660 x 19] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ array             : Factor w/ 6 levels "LU13","LU15",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ site              : Factor w/ 233 levels "LU13_18","LU13_15",..: 1 2 3 4 5 6 7 8 9 10 ...
##  $ buff_dist         : Factor w/ 20 levels "250","500","750",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ harvest           : num [1:4660] 0 0 0.687 0.337 0 ...
##  $ pipeline          : num [1:4660] 0 0.068 0 0 0.0301 ...
##  $ roads             : num [1:4660] 0 0.0174 0 0 0 ...
##  $ seismic_lines     : num [1:4660] 0 0.03277 0 0.00889 0.01144 ...
##  $ seismic_lines_3D  : num [1:4660] 0 0 0 0 0.0523 ...
##  $ trails            : num [1:4660] 0.00588 0.0028 0 0.01591 0 ...
##  $ transmission_lines: num [1:4660] 0.0642 0 0 0 0.091 ...
##  $ veg_edges         : num [1:4660] 0 0.0858 0 0 0 ...
##  $ wells             : num [1:4660] 0 0 0 0 0.0322 ...
##  $ lc_grassland      : num [1:4660] 0.193 0.348 0 0 0.178 ...
##  $ lc_coniferous     : num [1:4660] 0.456 0.358 0.186 1 0.822 ...
##  $ lc_broadleaf      : num [1:4660] 0 0 0 0 0 ...
##  $ lc_mixed          : num [1:4660] 0 0.101 0.255 0 0 ...
##  $ lc_developed      : num [1:4660] 0 0.0916 0 0 0 ...
##  $ lc_shrub          : num [1:4660] 0.316 0 0.559 0 0 ...
##  $ osm_industrial    : num [1:4660] 0.383 0.157 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   .default = col_number(),
##   ..   array = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
##   ..   site = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
##   ..   buff_dist = col_factor(levels = NULL, ordered = FALSE, include_na = FALSE),
##   ..   harvest = col_number(),
##   ..   pipeline = col_number(),
##   ..   roads = col_number(),
##   ..   seismic_lines = col_number(),
##   ..   seismic_lines_3D = col_number(),
##   ..   trails = col_number(),
##   ..   transmission_lines = col_number(),
##   ..   veg_edges = col_number(),
##   ..   wells = col_number(),
##   ..   lc_grassland = col_number(),
##   ..   lc_coniferous = col_number(),
##   ..   lc_broadleaf = col_number(),
##   ..   lc_mixed = col_number(),
##   ..   lc_developed = col_number(),
##   ..   lc_shrub = col_number(),
##   ..   osm_industrial = col_number()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

Everything looks good!

###Subset data by buffer

We do need to subset the data so we have separate data frames for each buffer width to work with in the analysis **AND** to explore correlations between variables at each buffer width, as these may very with spatial scales

Let's use a for loop to subset the data, thanks Andrew!

```
buffer_frames <- list()

for (i in unique(covariates_all$buff_dist)){

  print(i)

  # Subset data based on radius
  df <- covariates_all %>%
    filter(buff_dist == i)

  # list of dataframes
  buffer_frames <-c (buffer_frames, list(df))
}
```

```
## [1] "250"
## [1] "500"
## [1] "750"
## [1] "1000"
## [1] "1250"
## [1] "1500"
## [1] "1750"
## [1] "2000"
## [1] "2250"
## [1] "2500"
## [1] "2750"
## [1] "3000"
## [1] "3250"
## [1] "3500"
## [1] "3750"
## [1] "4000"
## [1] "4250"
## [1] "4500"
## [1] "4750"
## [1] "5000"
```

```
# name list objects so we can extract names for plotting

buffer_frames <- buffer_frames %>%

  # absurdly long way to do this but for sake of time fuck it
  purrr::set_names('250 meter buffer',
                   '500 meter buffer',
                   '750 meter buffer',
                   '1000 meter buffer',
                   '1250 meter buffer',
                   '1500 meter buffer',
                   '1750 meter buffer',
                   '2000 meter buffer',
                   '2250 meter buffer',
                   '2500 meter buffer',
                   '2750 meter buffer',
                   '3000 meter buffer',
                   '3250 meter buffer',
```

```
                    '3500 meter buffer',
                    '3750 meter buffer',
                    '4000 meter buffer',
                    '4250 meter buffer',
                    '4500 meter buffer',
                    '4750 meter buffer',
                    '5000 meter buffer')
```

Now we have a list with data frames for each buffer width which we can work with later.

Add response metric

Now that we have the covariate data formatted we need to add the response metric (monthly proportional presence/absence) to the data frames

```
osm_final_df_2021_2022 <- buffer_frames %>%

  purrr::map(
    ~.x %>%

      left_join(prop_dets_all,
                by = 'site'))
```

Finish with data formatting

Let's remove the objects we no longer need from the environment to keep our work space clean

```
rm(covariates_all,
   prop_detections,
   df,
   i)
```

Global Analysis

Now we are going to run a global model which includes all HFI and LC variables that at first glance (will do a more thorough check later) seem to have enough data to include as covariates for each buffer width, and then we will compare these models see which buffer width best fit the data for each species. After that we will optimize models so they don't includes any variables that are highly correlated.

> Edit: we added subset analsyis (anthropogenic and landscape) where we subset the data into anthropogenic features (HFI) and landscape (landcover classes) to see how optimum buffer size was influenced based on the type of variables included in the models. For these subset analyses we also looked more closely at which variables were correlated and have adjusted the global models accordingly (i.e. dropped/merged the same variables) to make them more accurate and comparable to the subset models. If there are variables commented out of the models, these were used in the initial run-through of the analysis but were dropped in the subset models so I've dropped them accordingly in a re-run of the global models.

This almost means we have to add one quick data formatting step which wasn't here in the initial run through of the data

```r
osm_final_df_2021_2022 <- osm_final_df_2021_2022 %>%

  # use purr so all changes are made to all data frames
  purrr::map(
    ~.x %>%

  # mutate data to merge variables that were correlated and similar enough to count as one for the purp
  mutate(pipeline_transmission_lines = pipeline + transmission_lines,
         lc_forest = lc_coniferous + lc_broadleaf + lc_mixed))

# view structure of one data frame
str(osm_final_df_2021_2022$`250 meter buffer`)
```

```
## tibble [233 x 47] (S3: tbl_df/tbl/data.frame)
##  $ array                       : Factor w/ 6 levels "LU13","LU15",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ site                        : Factor w/ 233 levels "LU13_18","LU13_15",..: 1 2 3 4 5 6 7 8 9 10 ..
##  $ buff_dist                   : Factor w/ 20 levels "250","500","750",..: 1 1 1 1 1 1 1 1 1 1 ...
##  $ harvest                     : num [1:233] 0 0 0.687 0.337 0 ...
##  $ pipeline                    : num [1:233] 0 0.068 0 0 0.0301 ...
##  $ roads                       : num [1:233] 0 0.0174 0 0 0 ...
##  $ seismic_lines               : num [1:233] 0 0.03277 0 0.00889 0.01144 ...
##  $ seismic_lines_3D            : num [1:233] 0 0 0 0 0.0523 ...
##  $ trails                      : num [1:233] 0.00588 0.0028 0 0.01591 0 ...
##  $ transmission_lines          : num [1:233] 0.0642 0 0 0 0.091 ...
##  $ veg_edges                   : num [1:233] 0 0.0858 0 0 0 ...
##  $ wells                       : num [1:233] 0 0 0 0 0.0322 ...
##  $ lc_grassland                : num [1:233] 0.193 0.348 0 0 0.178 ...
##  $ lc_coniferous               : num [1:233] 0.456 0.358 0.186 1 0.822 ...
##  $ lc_broadleaf                : num [1:233] 0 0 0 0 0 ...
##  $ lc_mixed                    : num [1:233] 0 0.101 0.255 0 0 ...
##  $ lc_developed                : num [1:233] 0 0.0916 0 0 0 ...
##  $ lc_shrub                    : num [1:233] 0.316 0 0.559 0 0 ...
##  $ osm_industrial              : num [1:233] 0.383 0.157 0 0 0 ...
##  $ black_bear                  : num [1:233] 1 4 6 2 0 3 1 2 1 1 ...
##  $ coyote                      : num [1:233] 9 8 5 3 2 7 9 0 0 1 ...
##  $ fisher                      : num [1:233] 0 0 0 1 0 0 0 0 0 0 ...
##  $ snowshoe_hare               : num [1:233] 11 9 0 0 2 3 8 3 0 3 ...
##  $ white-tailed_deer           : num [1:233] 4 5 3 0 0 3 4 0 0 0 ...
##  $ cougar                      : num [1:233] 0 0 0 0 0 0 0 0 0 0 ...
##  $ lynx                        : num [1:233] 4 1 0 0 1 4 1 1 0 2 ...
##  $ red_fox                     : num [1:233] 0 0 0 0 0 0 0 0 0 0 ...
##  $ moose                       : num [1:233] 0 0 4 0 1 1 0 1 0 3 ...
##  $ grey_wolf                   : num [1:233] 0 3 0 3 0 0 0 0 0 0 ...
##  $ caribou                     : num [1:233] 0 0 0 0 0 0 0 0 0 0 ...
##  $ wolverine                   : num [1:233] 0 0 0 0 0 0 0 0 0 0 ...
##  $ absent_black_bear           : num [1:233] 8 5 3 7 9 6 8 7 8 8 ...
##  $ absent_coyote               : num [1:233] 3 4 7 9 10 5 3 12 12 11 ...
##  $ absent_fisher               : num [1:233] 12 12 12 11 12 12 12 12 12 12 ...
##  $ absent_snowshoe_hare        : num [1:233] 1 3 12 12 10 9 4 9 12 9 ...
##  $ absent_white-tailed_deer    : num [1:233] 8 7 9 12 12 9 8 12 12 12 ...
##  $ absent_cougar               : num [1:233] 12 12 12 12 12 12 12 12 12 12 ...
##  $ absent_lynx                 : num [1:233] 8 11 12 12 11 8 11 11 12 10 ...
##  $ absent_red_fox              : num [1:233] 12 12 12 12 12 12 12 12 12 12 ...
```

```
##  $ absent_moose               : num [1:233] 12 12 8 12 11 11 12 11 12 9 ...
##  $ absent_grey_wolf           : num [1:233] 12 9 12 9 12 12 12 12 12 12 ...
##  $ absent_caribou             : num [1:233] 12 12 12 12 12 12 12 12 12 12 ...
##  $ red_squirrel               : num [1:233] 10 3 0 7 0 6 5 1 0 0 ...
##  $ absent_red_squirrel        : num [1:233] 2 9 12 5 12 6 7 11 12 12 ...
##  $ absent_wolverine           : num [1:233] 12 12 12 12 12 12 12 12 12 12 ...
##  $ pipeline_transmission_lines: num [1:233] 0.0642 0.068 0 0 0.1211 ...
##  $ lc_forest                  : num [1:233] 0.456 0.459 0.441 1 0.822 ...
```

We don't need to do ALL the species since many don't have enough data.

> Refer to the 1_ACME_camera_script_9-2-2024.html or .Rmd the plot for proportional monthly detections should provide info on which species we have enough data for, can be found under Response metrics/3.Proportion monthly detections

A brief look at this fig indicates that we have enough for all the mammals in the prop_detections data frame **except**

- cougar
- wolverine
- caribou??? (may have enough, may not)

Black bear

> there is probably a way to shorten the following code to select particular species, I saw Andrew's for loop in the draft script he wrote but couldn't quite figure out how to adapt it to my purposes with the data formatted the way I have it, so I did this instead, maybe we can merge approaches later to clean this up if deemed necessary? But it certainly functions for now and is understandable... I think.

Global models

Let's start with bears and use purrr to create a global model for every buffer distance

Recall `purrr::map()` is magical for iterations and will apply all the functions within the `map()` function to each item of the list supplied before the the `map()` function.

```
# create models for black bears at each buffer size
black_bear_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
```

```
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))
```

Model selection We will use the `model.sel()` function from the *MuMIn* package to compare the global models for each buffer width and see which buffer fits the black bear data best

```
# run model selection and save the results as a tibble for graphing use later
black_bear_global_model.sel <- model.sel(black_bear_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
black_bear_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1   250 meter buffer  -0.5980169           +          0.007994002
## 2  3750 meter buffer  -0.5985159           +          0.063737243
## 3  4000 meter buffer  -0.5988958           +          0.066912736
## 4  3500 meter buffer  -0.5983175           +          0.055592270
## 5  3250 meter buffer  -0.5979959           +          0.057684143
## 6  4250 meter buffer  -0.5985903           +          0.074157040
## 7  2750 meter buffer  -0.5964813           +          0.051593393
## 8  3000 meter buffer  -0.5969842           +          0.054352443
## 9  2500 meter buffer  -0.5953182           +          0.043643124
## 10 4500 meter buffer  -0.5981399           +          0.084455835
## 11 2250 meter buffer  -0.5949003           +          0.034920531
## 12 2000 meter buffer  -0.5957579           +          0.035769335
```

```
## 13 1750 meter buffer  -0.5965510           +          0.037729143
## 14 1500 meter buffer  -0.5968317           +          0.038047627
## 15 4750 meter buffer  -0.5982737           +          0.084943338
## 16 5000 meter buffer  -0.5977571           +          0.089220581
## 17 1250 meter buffer  -0.5961642           +          0.028780877
## 18 1000 meter buffer  -0.5953469           +          0.019342371
## 19  500 meter buffer  -0.5967796           +          0.032865831
## 20  750 meter buffer  -0.5945338           +          0.031272604
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1                -0.18817816          -0.117329550              -0.001633920
## 2                -0.12238956          -0.018577823              -0.033644857
## 3                -0.11741932          -0.019160942              -0.021771808
## 4                -0.11954337          -0.004563477              -0.027074375
## 5                -0.11757691           0.005321373              -0.027887991
## 6                -0.11001361          -0.030798057              -0.017549031
## 7                -0.12577734           0.024428922              -0.039484929
## 8                -0.11949519           0.016129793              -0.035466584
## 9                -0.12320851           0.028731402              -0.060955671
## 10               -0.09582794          -0.041341629              -0.022064400
## 11               -0.12626177           0.039872849              -0.065079993
## 12               -0.12271892           0.041853052              -0.051017515
## 13               -0.12031961           0.031947677              -0.038185093
## 14               -0.13837424           0.016515660              -0.033476364
## 15               -0.08954523          -0.047390826              -0.027063637
## 16               -0.09662684          -0.058542128              -0.030799609
## 17               -0.13591024           0.015875842              -0.027751477
## 18               -0.10213770           0.036216575              -0.016250799
## 19               -0.11199818          -0.022960834               0.051657322
## 20               -0.10855727           0.028620814               0.007246516
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                  0.033049532                                -0.07254973
## 2                 -0.110610223                                -0.11564161
## 3                 -0.110931416                                -0.12731595
## 4                 -0.093471225                                -0.12514482
## 5                 -0.080988433                                -0.13400657
## 6                 -0.110641347                                -0.11108057
## 7                 -0.049156036                                -0.13673851
## 8                 -0.064566331                                -0.13648825
## 9                 -0.038812955                                -0.11762028
## 10                -0.117727387                                -0.10938327
## 11                -0.024962780                                -0.10897596
## 12                -0.009151511                                -0.10615428
## 13                 0.002659282                                -0.10943393
## 14                 0.017318953                                -0.09956684
## 15                -0.117008201                                -0.10327526
## 16                -0.116378237                                -0.09421258
## 17                 0.034924606                                -0.08951920
## 18                 0.013738406                                -0.09475402
## 19                 0.018231490                                -0.12202041
## 20                 0.030198632                                -0.08703427
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.1168960               -0.01782996          0.12649868
## 2                     -0.1216505               -0.12116452          0.10384016
## 3                     -0.1160259               -0.12741813          0.10412186
```

```
## 4                    -0.1136758               -0.10577448          0.09673126
## 5                    -0.1098018               -0.09997384          0.09012450
## 6                    -0.1191445               -0.14162573          0.10153308
## 7                    -0.1125162               -0.09440463          0.07928799
## 8                    -0.1103608               -0.09113884          0.07835177
## 9                    -0.1117889               -0.09321001          0.08614513
## 10                   -0.1253820               -0.15659265          0.10042965
## 11                   -0.1122646               -0.09143361          0.09190455
## 12                   -0.1146880               -0.10762045          0.08801546
## 13                   -0.1198787               -0.12237009          0.07879741
## 14                   -0.1286891               -0.10071781          0.07856426
## 15                   -0.1283342               -0.15907323          0.09693842
## 16                   -0.1358831               -0.16134235          0.09090227
## 17                   -0.1341083               -0.06911243          0.08315494
## 18                   -0.1301128               -0.05777813          0.08347821
## 19                   -0.1183573               -0.02208990          0.07081656
## 20                   -0.1314970               -0.03521406          0.04347846
##    cond(scale(wells)) df    logLik     AICc     delta      weight
## 1         -0.01118399 12 -448.9609 923.3464  0.000000 0.678117929
## 2          0.28508132 12 -451.5357 928.4960  5.149550 0.051652889
## 3          0.28817305 12 -451.6143 928.6532  5.306750 0.047748422
## 4          0.26178688 12 -452.1408 929.7063  6.359888 0.028201561
## 5          0.26161862 12 -452.2350 929.8946  6.548196 0.025667433
## 6          0.27289486 12 -452.4707 930.3660  7.019633 0.020277352
## 7          0.25930355 12 -452.4741 930.3728  7.026364 0.020209226
## 8          0.25974061 12 -452.6879 930.8006  7.454142 0.016317699
## 9          0.24078162 12 -452.8100 931.0447  7.698294 0.014442482
## 10         0.28167893 12 -452.8168 931.0582  7.711804 0.014345258
## 11         0.23007384 12 -452.8612 931.1471  7.800727 0.013721416
## 12         0.19780467 12 -452.9433 931.3112  7.964820 0.012640568
## 13         0.17750600 12 -453.0539 931.5324  8.185981 0.011317280
## 14         0.16480949 12 -453.1899 931.8044  8.458003 0.009878095
## 15         0.27294373 12 -453.2413 931.9073  8.560848 0.009382978
## 16         0.27590212 12 -453.4016 932.2279  8.881446 0.007993258
## 17         0.13017251 12 -453.4538 932.3323  8.985866 0.007586634
## 18         0.11152605 12 -453.8759 933.1765  9.830099 0.004974236
## 19         0.08990772 12 -454.0540 933.5326 10.186168 0.004163006
## 20         0.12258499 12 -455.1710 935.7667 12.420327 0.001362277
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
```

```
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Looks like the smallest buffer (250) fits the data best for black bears.

> Note about re-run global models: when we removed the correlated variables the top model satyed the same (250m) but the weight increased (from 0.159 to 0.678)

Let's look at this model closer

Model summary Since we aren't interpreting the magnitude and direction of effect for individual variables in the model as you normally would, the model summary serves primarily to see that there are no issues with the top model (convergence issues, large SE, etc.) that could cause us to quetion the results of the model.sel() function.

```
summary(black_bear_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(black_bear, absent_black_bear) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    921.9    963.3   -449.0    897.9      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.08116  0.2849
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -0.598017   0.125793  -4.754 1.99e-06 ***
## scale(harvest)                   0.007994   0.049036   0.163  0.87050
## scale(seismic_lines)            -0.017830   0.052626  -0.339  0.73476
## scale(seismic_lines_3D)         -0.116896   0.058268  -2.006  0.04484 *
## scale(trails)                    0.126499   0.046332   2.730  0.00633 **
## scale(wells)                    -0.011184   0.053722  -0.208  0.83509
## scale(osm_industrial)            0.033050   0.051185   0.646  0.51848
## scale(pipeline_transmission_lines) -0.072550   0.063235  -1.147  0.25125
## scale(lc_grassland)             -0.001634   0.059234  -0.028  0.97799
## scale(lc_developed)             -0.188178   0.065107  -2.890  0.00385 **
## scale(lc_forest)                -0.117330   0.070534  -1.663  0.09622 .
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Nothing looks fishy in the model summary for now, we will look at this more closely once we have a true top model.

Code to remove a model

> At one point the 250 meter buffer was giving us issues so we had to remove it. After re-extracting the data and re-doing some data formatting this is no longer an issue but I've saved the code here in case it's needed in the future

```
# create models for black bears at each buffer size
black_bear_mods_no250 <- osm_final_df_2021_2022 %>%

# remove 250 meter buffer width
  purrr::discard_at('250 meter buffer') %>%

  # use purrr map to fun the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                        # HFI
                        scale(harvest) +
                        scale(pipeline) +
                        scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(transmission_lines) +
                        scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        scale(lc_coniferous) +
                        scale(lc_broadleaf) +
                        scale(lc_mixed) +
                        scale(lc_developed) +
                        scale(lc_shrub) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))
```

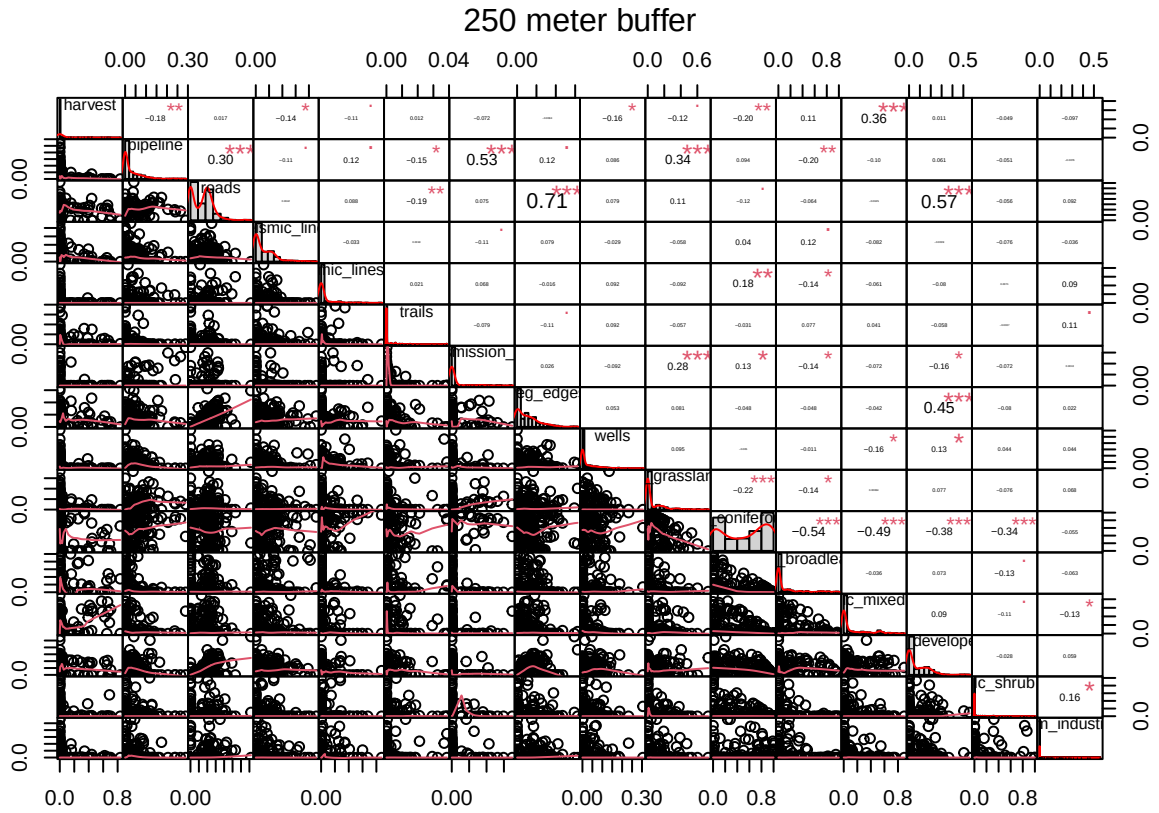
Subset models

> NOTE this code was not used for publication but was an initial idea for how we could further use the data. We only explored subset models for Bears thus far, but have left the headings for the other species in case we want to add in these subset models later

Autocorrelation 250m Before we can develop model subsets we need to see what variables can be included in the same model at this buffer width.

Let's use the `chart.Correlation()` function in the *Performance Analytics* package to look at this.

```
buffer_frames$`250 meter buffer` %>%

  select_if(is.numeric) %>%

   # use chart.correlation
      chart.Correlation(.,
                        histogram = TRUE,
                        method = "pearson")

mtext('250 meter buffer', side = 3, line = 3)
```



> You can click on this fig to zoom in!

List of correlated variables:

- pipeline & transmission_lines 0.53

- roads & veg_edges 0.71
- roads & lc_developed 0.57

Model list Let's create another global model without these correlated variables. I'm going to select transmission_lines over pipeline because the summary from earlier showed transmission lines had larger effect on black bear presence, and I'm going to choose to keep roads instead of veg edges and the developed landcover class because we are interested in the effect of roads more than these other two variables.

```
# global model w/ non-correlated variables
bear_global_250 <-  glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                        # HFI
                        scale(harvest) +
                        scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(transmission_lines) +
                        scale(wells) +
                        scale(osm_industrial) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        scale(lc_coniferous) +
                        scale(lc_broadleaf) +
                        scale(lc_mixed) +
                        scale(lc_shrub) +

                        # Random effect of array
                        (1|array),
                    data = osm_final_df_2021_2022$`250 meter buffer`,
                    family = 'binomial')

# null model to compare
bear_null_250 <- glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~ 1 +

                        # Random effect of array
                        (1|array),
                    data = osm_final_df_2021_2022$`250 meter buffer`,
                    family = 'binomial')

# second model is based on linear features providing easier movement through boreal forest
bear_linear_250 <- glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                        # HFI
                        scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(transmission_lines) +

                        # Random effect of array
                        (1|array),
```

```
                  data = osm_final_df_2021_2022$`250 meter buffer`,
                  family = 'binomial')
```

Model selection Now let's look at a model selection table with our subset models.

```
model.sel(bear_global_250,
          bear_null_250,
          bear_linear_250)
```

```
## Model selection table
##                 cnd((Int)) dsp((Int)) cnd(scl(hrv)) cnd(scl(lc_brd))
## bear_linear_250    -0.5986          +
## bear_global_250    -0.5995          +       0.01223           0.2326
## bear_null_250      -0.5884          +
##                 cnd(scl(lc_cnf)) cnd(scl(lc_grs)) cnd(scl(lc_mxd))
## bear_linear_250
## bear_global_250           0.1823           0.1248           0.1455
## bear_null_250
##                 cnd(scl(lc_shr)) cnd(scl(osm_ind)) cnd(scl(rds))
## bear_linear_250                                          -0.1586
## bear_global_250           0.1891           0.04922       -0.1215
## bear_null_250
##                 cnd(scl(ssm_lns_3D)) cnd(scl(ssm_lns)) cnd(scl(trl))
## bear_linear_250              -0.1311          -0.04436       0.10250
## bear_global_250              -0.1135          -0.04271       0.09811
## bear_null_250
##                 cnd(scl(trn_lns)) cnd(scl(wll)) df   logLik  AICc delta weight
## bear_linear_250          -0.09901                7 -449.674 913.8  0.00  0.972
## bear_global_250          -0.10520      -0.01348 15 -444.383 921.0  7.14  0.027
## bear_null_250                                    2 -463.090 930.2 16.38  0.000
## Models ranked by AICc(x)
## Random terms (all models):
##   cond(1 | array)
```

Looks like the linear feature model is best, SO FAR

Top model

250 meter buffer - linear features

```
summary(bear_linear_250)
```

Summary

```
##  Family: binomial  ( logit )
## Formula:
## cbind(black_bear, absent_black_bear) ~ scale(roads) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(transmission_lines) +
```

```
##     (1 | array)
## Data: osm_final_df_2021_2022$`250 meter buffer`
##
##      AIC      BIC   logLik deviance df.resid
##    913.3    937.5   -449.7    899.3      225
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.05426  0.2329
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                            Estimate Std. Error z value Pr(>|z|)
## (Intercept)                -0.59856    0.10644  -5.624 1.87e-08 ***
## scale(roads)               -0.15864    0.05694  -2.786  0.00534 **
## scale(seismic_lines)       -0.04436    0.05006  -0.886  0.37556
## scale(seismic_lines_3D)    -0.13114    0.05635  -2.327  0.01995 *
## scale(trails)               0.10247    0.04641   2.208  0.02724 *
## scale(transmission_lines) -0.09901    0.05653  -1.751  0.07987 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Odds ratios Let's extract the odds ratios for the top model so we can plot them for data vis later.

```
bear_model_odds <- bear_linear_250 %>%

  # extract the coefficients and upper and lower CI
      confint() %>%

      # format resulting object as a tibble data frame
      as_tibble() %>%


      # add a column where we can put the feature names
      rowid_to_column() %>%

      # rename the columns for plotting
      rename('lower' = `2.5 %`,
             'upper' = `97.5 %`,
             'estimate' = Estimate,
             'feature' = rowid) %>%

      # rename the entries to features, need to look at the order the features are in from the model su
      mutate(feature = as.factor(feature),
             feature = recode(feature,
                              '1' = 'intercept',
                              '2' = 'roads',
                              '3' = 'seismic_lines',
                              '4' = 'seismic_lines_3D',
                              '5' = 'trails',
                              '6' = 'transmission_lines',
```

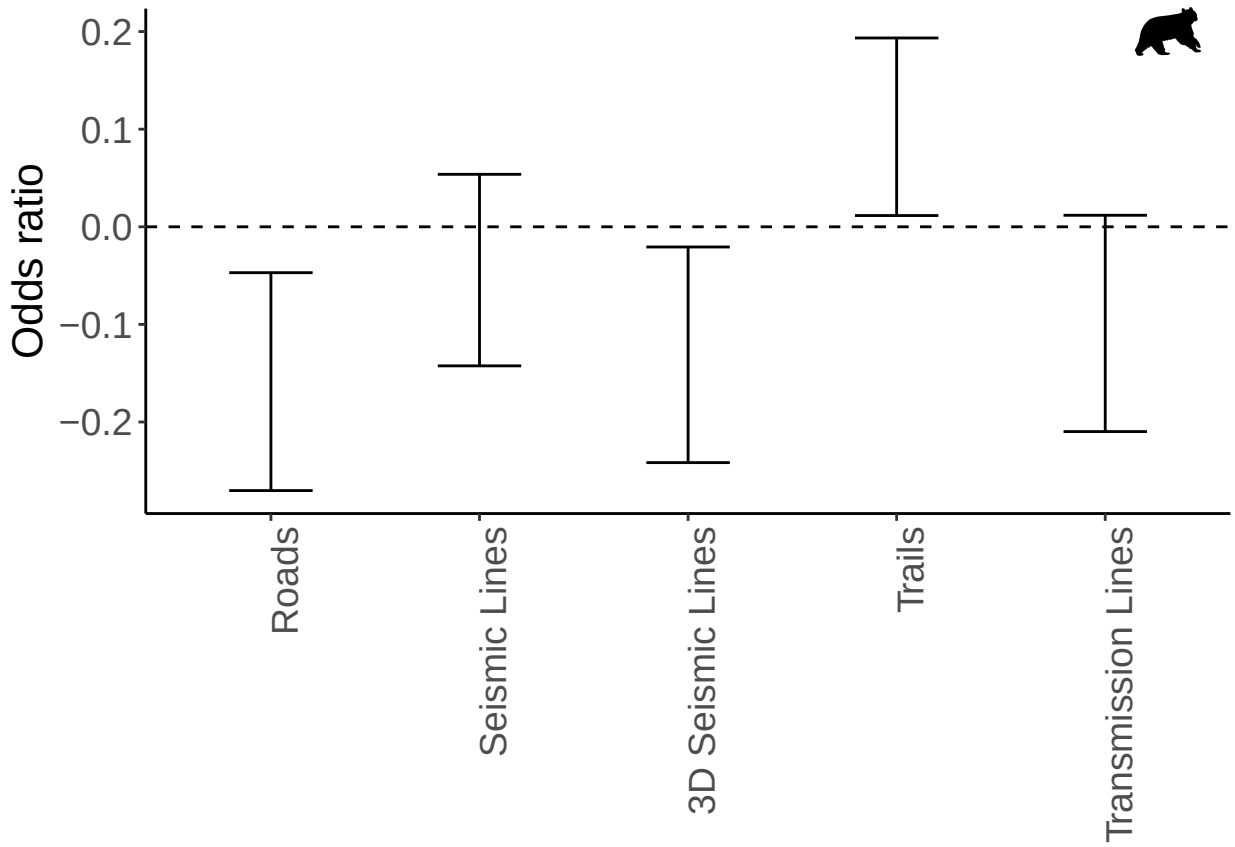
```
                              '7' = 'intercept_array')) %>%

  # remove intercepts
  filter(!grepl('intercept',
                feature))
```

Plot odds ratios First let's get a silhouette for this graphy from phylopic

```
black_bear_img <- get_phylopic(get_uuid(name = 'Ursus americanus'))
```

Now let's use ggplot to plot the odds ratios for each feature in the top model

```
# provide data and mapping aesthetics
ggplot(bear_model_odds, aes(x = feature,
                            y = estimate)) +

   geom_errorbar(aes(ymin = lower,
                     ymax = upper),
                 width = 0.4,
                 linewidth = 0.5,
                 position = position_dodge(width = 0.9)) +

   geom_hline(yintercept = 0, linetype = "dashed") +

  labs(y = 'Odds ratio') +

  scale_x_discrete(labels = c('Roads',
                              'Seismic Lines',
                              '3D Seismic Lines',
                              'Trails',
                              'Transmission Lines')) +

  add_phylopic(black_bear_img,
               x = 5.3,
               y = 0.2,
               ysize = 0.05) +

  theme_classic() +

  theme(axis.title.x = element_blank(),
        axis.text.x = element_text(angle = 90,
                                   hjust = 1),
        axis.title = element_text(size = 16),
        axis.text = element_text(size = 14))
```



Let's repeat this process for each species that we have enough data for.

Caribou

We may or may not have enough data for caribou but let's try it at least for this preliminary report

We can use the same code from black bears (above) to run global models for each buffer width

And in the same chunk to save time let's also run the `model.sel()` function

Global models

```
caribou_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

     glmmTMB::glmmTMB(cbind(caribou, absent_caribou) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
```

```
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))
```

```
## Warning in finalizeTMB(TMBStruc, obj, fit, h, data.tmb.old): Model convergence
## problem; singular convergence (7). See vignette('troubleshooting'),
## help('diagnose')
```

```
# run model selection and save the results as a tibble for graphing use later
caribou_global_model.sel <- model.sel(caribou_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
caribou_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  1000 meter buffer   -5.541691           +          -0.65725167
## 2  1250 meter buffer   -5.389313           +          -0.36118018
## 3   750 meter buffer   -5.765763           +          -1.56407319
## 4  1500 meter buffer   -5.302534           +          -0.19770713
## 5  2500 meter buffer   -5.603878           +          -0.16267341
## 6  2000 meter buffer   -5.379371           +          -0.21428286
## 7  1750 meter buffer   -5.282614           +          -0.21810476
## 8  2750 meter buffer   -5.573419           +          -0.13425099
## 9  2250 meter buffer   -5.475323           +          -0.19149017
## 10  500 meter buffer   -5.749800           +          -2.30534167
## 11 3000 meter buffer   -5.603490           +          -0.04353968
## 12  250 meter buffer -204.202683           +        -599.03806642
## 13 3250 meter buffer   -5.702496           +           0.02383525
```

```
## 14 3500 meter buffer   -5.674306           +          -0.01945111
## 15 3750 meter buffer   -5.640270           +          -0.08865803
## 16 4750 meter buffer   -5.192625           +          -1.17614785
## 17 5000 meter buffer   -5.222123           +          -1.38062579
## 18 4000 meter buffer   -5.527983           +          -0.19871032
## 19 4500 meter buffer   -5.133141           +          -0.89512309
## 20 4250 meter buffer   -5.195888           +          -0.54075400
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1                 0.06847734            0.205338004               -0.06587403
## 2                -0.09234293            0.190861812               -0.14997595
## 3                -0.04325940            0.253684235               -0.06969995
## 4                -0.30868340            0.207730850               -0.24014413
## 5                 0.32758898            0.176629676               -0.34144706
## 6                 0.10330649            0.192238820               -0.27845455
## 7                -0.31571093            0.180899772               -0.26739770
## 8                 0.30491896            0.140374009               -0.31461343
## 9                 0.24039981            0.191954202               -0.34027327
## 10               -0.05809455            0.159390735               -0.13692140
## 11                0.44238834            0.098769628               -0.25212109
## 12               -0.30358218            0.177228128               -0.14613326
## 13                0.61272256            0.062340559               -0.24326333
## 14                0.62165873            0.046119302               -0.26295003
## 15                0.62134322            0.033176341               -0.28878257
## 16               -0.59518026           -0.072105246               -0.38774229
## 17               -0.65944116           -0.094553901               -0.38136360
## 18                0.51387299            0.006902339               -0.31771379
## 19               -0.25990904           -0.032875304               -0.36765369
## 20                0.02969350           -0.015897931               -0.33420234
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                   -1.2929550                                -0.3002017
## 2                   -1.1419093                                -0.2206548
## 3                   -1.0232651                                -0.2924596
## 4                   -0.9155964                                -0.2617808
## 5                   -1.4878903                                -0.7149004
## 6                   -1.5029489                                -0.4076253
## 7                   -1.0907323                                -0.2710183
## 8                   -1.1069871                                -0.8976197
## 9                   -1.6242628                                -0.4375504
## 10                  -0.3164276                                -0.1656729
## 11                  -0.9702550                                -0.9864143
## 12                  -0.1913004                                -0.2072504
## 13                  -0.9267209                                -0.9570913
## 14                  -0.8173317                                -0.9677301
## 15                  -0.7580180                                -0.9200752
## 16                   0.0774661                                -0.6926807
## 17                   0.0916440                                -0.6611871
## 18                  -0.6265348                                -0.8016489
## 19                  -0.0755507                                -0.7022723
## 20                  -0.2868306                                -0.7415273
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.1386450               0.143810555          0.04559153
## 2                     -0.1474092               0.176342455          0.05588960
## 3                     -0.1412665               0.219454946          0.03604664
## 4                     -0.2001540               0.086853722         -0.04956552
```

```
## 5                    -0.1131389               0.163996802         -0.19603058
## 6                    -0.1097337               0.104234593         -0.06962252
## 7                    -0.1938458               0.022045525         -0.04380021
## 8                    -0.1277863               0.115521182         -0.26147205
## 9                    -0.1162640               0.176184157         -0.09751303
## 10                   -0.1700043               0.209347341         -0.04656235
## 11                   -0.1276584               0.009939888         -0.37206550
## 12                   -0.2941479               0.229333639         -0.29452119
## 13                   -0.1511563              -0.071569637         -0.46886471
## 14                   -0.1273545              -0.088015843         -0.50306198
## 15                   -0.1116802              -0.116771313         -0.48833303
## 16                   -0.1672512               0.051263313         -0.50796175
## 17                   -0.1574435               0.078616191         -0.52941063
## 18                   -0.1043717              -0.121080522         -0.42295765
## 19                   -0.1232611              -0.011530570         -0.43142000
## 20                   -0.1172785              -0.065326214         -0.41312655
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1           0.7342746 12 -131.8196 289.0640  0.000000 7.963951e-01
## 2           0.7962446 12 -134.1331 293.6908  4.626890 7.877930e-02
## 3           0.6896422 12 -134.2288 293.8823  4.818342 7.158779e-02
## 4           0.8722577 12 -135.5227 296.4701  7.406097 1.962976e-02
## 5           1.5625358 12 -136.3080 298.0406  8.976616 8.951200e-03
## 6           1.1628716 12 -136.6219 298.6685  9.604506 6.539380e-03
## 7           0.9260111 12 -136.9531 299.3310 10.266997 4.695463e-03
## 8           1.5513888 12 -136.9576 299.3399 10.275898 4.674613e-03
## 9           1.3500289 12 -136.9868 299.3982 10.334214 4.540280e-03
## 10          0.4975402 12 -137.7936 301.0118 11.947861 2.026206e-03
## 11          1.4900746 12 -138.5356 302.4958 13.431846 9.648066e-04
## 12          0.3695857 12 -138.8885 303.2016 14.137627 6.779253e-04
## 13          1.4380997 12 -140.0395 305.5037 16.439735 2.144300e-04
## 14          1.4153473 12 -140.5539 306.5325 17.468558 1.281975e-04
## 15          1.3861851 12 -141.1210 307.6667 18.602748 7.270984e-05
## 16          1.1279565 12 -141.7487 308.9221 19.858169 3.881343e-05
## 17          1.1220463 12 -141.7695 308.9636 19.899640 3.801690e-05
## 18          1.3090163 12 -142.3013 310.0273 20.963332 2.233567e-05
## 19          1.1030771 12 -142.8833 311.1913 22.127307 1.248087e-05
## 20          1.1761871 12 -142.9935 311.4116 22.347619 1.117905e-05
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
```

```
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

We get a warning that there are some model convergence problems, I expect this is because we don't have enough data for caribou, so we may choose to remove the caribou results from the manuscript or provide that caveat.

For now let's examine the top buffer width and see if the model seems reasonable.

> Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for caribou does change (3000m to 1000m) which now matches the top model for anthropogenic features, this could be because there are more anthro features in the global models and/or because of lack of data/model convergence issues. If we see this same trend with other global models it will be worth mentioning as a caveat in the discussion.

Model summary 1000m Let's take a closer look at the top model summary

```
summary(caribou_mods$`1000 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(caribou, absent_caribou) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    287.6    329.0   -131.8    263.6      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 4.767    2.183
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -5.54169    1.07006  -5.179 2.23e-07 ***
## scale(harvest)                   -0.65725    0.64637  -1.017  0.30924
## scale(seismic_lines)              0.14381    0.16616   0.865  0.38678
## scale(seismic_lines_3D)          -0.13865    0.21047  -0.659  0.51006
## scale(trails)                     0.04559    0.20832   0.219  0.82676
## scale(wells)                      0.73427    0.17137   4.285 1.83e-05 ***
## scale(osm_industrial)            -1.29295    0.47653  -2.713  0.00666 **
## scale(pipeline_transmission_lines) -0.30020    0.30963  -0.970  0.33227
## scale(lc_grassland)              -0.06587    0.16616  -0.396  0.69178
## scale(lc_developed)               0.06848    0.40830   0.168  0.86681
## scale(lc_forest)                  0.20534    0.14382   1.428  0.15336
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

There's nothing that catches my eye immediately as being sus about this particular model so it may not have been one with convergence issues. We will keep it in report for now

Subset Models

Add later if deemed necessary.

Coyote

Global models

```
coyote_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

     glmmTMB::glmmTMB(cbind(coyote, absent_coyote) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))


# run model selection and save the results as a tibble for graphing use later
```

```
coyote_global_model.sel <- model.sel(coyote_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
coyote_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  5000 meter buffer   -1.387405           +         -0.084683809
## 2  4750 meter buffer   -1.386435           +         -0.080093043
## 3  4500 meter buffer   -1.384919           +         -0.070055963
## 4  4250 meter buffer   -1.383120           +         -0.061963914
## 5  1250 meter buffer   -1.387643           +          0.041423041
## 6  1500 meter buffer   -1.382727           +          0.040671391
## 7  2250 meter buffer   -1.377760           +         -0.011873066
## 8  1750 meter buffer   -1.379182           +          0.026285665
## 9  2000 meter buffer   -1.376632           +          0.003103946
## 10 3000 meter buffer   -1.379774           +         -0.024974850
## 11 2500 meter buffer   -1.378784           +         -0.016181448
## 12 2750 meter buffer   -1.378749           +         -0.021552916
## 13 4000 meter buffer   -1.381507           +         -0.059528492
## 14 1000 meter buffer   -1.391280           +          0.042799808
## 15 3250 meter buffer   -1.379660           +         -0.031487002
## 16 3500 meter buffer   -1.378673           +         -0.043315893
## 17 3750 meter buffer   -1.379494           +         -0.049725718
## 18  750 meter buffer   -1.392533           +          0.073423108
## 19  500 meter buffer   -1.390117           +          0.038748362
## 20  250 meter buffer   -1.386069           +         -0.056699954
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1                  0.3723829            -0.1599505              -0.201565893
## 2                  0.3626378            -0.1687627              -0.197723406
## 3                  0.3539336            -0.1750140              -0.192288881
## 4                  0.3337514            -0.1807675              -0.171215491
## 5                  0.3791189            -0.1453668              -0.022652122
## 6                  0.3471607            -0.1458746              -0.033376423
## 7                  0.3432047            -0.1490561              -0.092859888
## 8                  0.3454284            -0.1416329              -0.053434037
## 9                  0.3411187            -0.1375864              -0.073698581
## 10                 0.3105397            -0.1719981              -0.060974060
## 11                 0.3309993            -0.1559528              -0.077812366
## 12                 0.3161171            -0.1623329              -0.060873299
## 13                 0.3263633            -0.1865309              -0.140912758
## 14                 0.3792161            -0.1430275              -0.011126504
## 15                 0.3142354            -0.1785523              -0.076444084
## 16                 0.3314600            -0.1843575              -0.107448755
## 17                 0.3304539            -0.1920417              -0.127281904
## 18                 0.3436510            -0.1184575               0.007329282
## 19                 0.2172943            -0.1126626               0.020854512
## 20                 0.1531684            -0.1636688              -0.022135648
```

```
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                    0.3536675                                0.1136902698
## 2                    0.3539307                                0.1075698503
## 3                    0.3470543                                0.0934759992
## 4                    0.3402821                                0.0789617270
## 5                    0.2821016                                0.0194312986
## 6                    0.3035874                                0.0311907903
## 7                    0.3043412                                0.0261521144
## 8                    0.3033371                                0.0253843467
## 9                    0.3109182                                0.0109109544
## 10                   0.3340890                               -0.0344420146
## 11                   0.3135146                               -0.0003666942
## 12                   0.3360739                               -0.0308855950
## 13                   0.3177742                                0.0469077523
## 14                   0.2735007                                0.0301858715
## 15                   0.3277301                               -0.0137064626
## 16                   0.3026731                                0.0135309452
## 17                   0.2960900                                0.0342291463
## 18                   0.2877806                                0.0875596873
## 19                   0.2931698                                0.0656488312
## 20                   0.2168276                                0.1311700564
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.07121340                  0.2668503       -1.666319e-02
## 2                     -0.07461374                  0.2590864       -2.444740e-02
## 3                     -0.07683031                  0.2565248       -2.321689e-02
## 4                     -0.07192316                  0.2365325       -1.811958e-02
## 5                     -0.14059392                  0.2701879        1.330279e-02
## 6                     -0.12764212                  0.2595679        3.690687e-02
## 7                     -0.08821219                  0.2749065        6.001013e-02
## 8                     -0.11131888                  0.2521583        4.822123e-02
## 9                     -0.09834766                  0.2497815        4.571074e-02
## 10                    -0.05023398                  0.2206073        2.613415e-02
## 11                    -0.07148100                  0.2642255        4.629794e-02
## 12                    -0.05897457                  0.2322501        2.767566e-02
## 13                    -0.06100372                  0.2270799       -2.052174e-03
## 14                    -0.13165027                  0.2595744       -6.606805e-05
## 15                    -0.04805539                  0.2130992        1.474503e-02
## 16                    -0.05270340                  0.2203164        1.938682e-02
## 17                    -0.05709709                  0.2199482        1.475222e-02
## 18                    -0.13414118                  0.2508119       -3.534228e-02
## 19                    -0.12495868                  0.1928322       -7.749655e-02
## 20                    -0.02725795                  0.1533772        3.497855e-02
##    cond(scale(wells)) df    logLik      AICc      delta       weight
## 1          0.00949774 12 -460.0687  945.5622  0.0000000 4.442975e-01
## 2          0.01628066 12 -460.2716  945.9678  0.4056534 3.627333e-01
## 3          0.03685148 12 -461.2274  947.8795  2.3173121 1.394684e-01
## 4          0.04561015 12 -463.7125  952.8496  7.2874875 1.162027e-02
## 5          0.09660289 12 -463.7166  952.8579  7.2957714 1.157224e-02
## 6          0.09859273 12 -464.0414  953.5074  7.9452171 8.363574e-03
## 7          0.09614335 12 -464.4395  954.3036  8.7414206 5.616924e-03
## 8          0.09473482 12 -464.7146  954.8538  9.2916655 4.265936e-03
## 9          0.09525575 12 -465.1718  955.7683 10.2061087 2.700509e-03
## 10         0.09630945 12 -465.4910  956.4067 10.8445222 1.962528e-03
## 11         0.09437106 12 -465.5461  956.5169 10.9547310 1.857309e-03
```

```
## 12         0.09051201 12 -465.9488  957.3223 11.7601157 1.241644e-03
## 13         0.05943287 12 -466.0326  957.4899 11.9277674 1.141805e-03
## 14         0.13733603 12 -466.0661  957.5568 11.9946770 1.104238e-03
## 15         0.08004487 12 -466.2721  957.9688 12.4066773 8.986657e-04
## 16         0.08138492 12 -466.7154  958.8555 13.2933960 5.768324e-04
## 17         0.07878630 12 -466.7895  959.0036 13.4414311 5.356785e-04
## 18         0.19147174 12 -469.3204  964.0655 18.5033626 4.263043e-05
## 19         0.20430702 12 -485.6540  996.7327 51.1705500 3.436609e-12
## 20         0.03063854 12 -503.9044 1033.2334 87.6712544 4.074768e-20
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for coyote did change (2250-5000). As with caribou this result is also more similar to the top anthro model (4750m)

Model summary 5000m Let's get the model summary for this model

```
summary(coyote_mods$`5000 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(coyote, absent_coyote) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    944.1    985.5   -460.1    920.1      220
##
## Random effects:
##
```

```
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1189   0.3449
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -1.387405   0.150996  -9.188  < 2e-16 ***
## scale(harvest)                    -0.084684   0.094828  -0.893 0.371845
## scale(seismic_lines)               0.266850   0.077153   3.459 0.000543 ***
## scale(seismic_lines_3D)           -0.071213   0.074950  -0.950 0.342037
## scale(trails)                     -0.016663   0.072397  -0.230 0.817965
## scale(wells)                       0.009498   0.111273   0.085 0.931979
## scale(osm_industrial)              0.353667   0.064260   5.504 3.72e-08 ***
## scale(pipeline_transmission_lines) 0.113690   0.106001   1.073 0.283479
## scale(lc_grassland)               -0.201566   0.091179  -2.211 0.027059 *
## scale(lc_developed)                0.372383   0.090102   4.133 3.58e-05 ***
## scale(lc_forest)                  -0.159951   0.070220  -2.278 0.022735 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Everything looks good here

Subset models

Add later if deemed necessary.

Fisher

Global models

```
fisher_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

     glmmTMB::glmmTMB(cbind(fisher, absent_fisher) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +
```

```
                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
fisher_global_model.sel <- model.sel(fisher_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
fisher_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  1000 meter buffer   -3.099759           +           -0.2427972
## 2   750 meter buffer   -3.078768           +           -0.2584359
## 3   500 meter buffer   -3.063430           +           -0.3150145
## 4  2500 meter buffer   -3.013959           +           -0.1365859
## 5  1250 meter buffer   -3.061498           +           -0.2543493
## 6  2750 meter buffer   -3.005513           +           -0.1379150
## 7  2250 meter buffer   -3.026091           +           -0.1485053
## 8  3000 meter buffer   -2.994056           +           -0.1425289
## 9  3250 meter buffer   -2.987314           +           -0.1566246
## 10 2000 meter buffer   -3.029025           +           -0.1699645
## 11 3500 meter buffer   -2.979809           +           -0.1801562
## 12 1500 meter buffer   -3.031835           +           -0.2348054
## 13 4250 meter buffer   -2.979184           +           -0.2075828
## 14 1750 meter buffer   -3.024476           +           -0.2122054
## 15 4500 meter buffer   -2.983539           +           -0.2265939
## 16 4000 meter buffer   -2.973866           +           -0.1987231
## 17 3750 meter buffer   -2.972995           +           -0.1957554
## 18 4750 meter buffer   -2.990213           +           -0.2585855
## 19 5000 meter buffer   -2.988975           +           -0.2811262
## 20  250 meter buffer   -3.001628           +           -0.3662532
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1                  0.5727364             0.7061622                0.21162467
## 2                  0.5548374             0.7433866                0.22503536
## 3                  0.5009800             0.6974368                0.26369034
## 4                  0.5125109             0.5430848               -0.11199246
## 5                  0.5113902             0.6093210                0.15921449
```

```
## 6                 0.4877380             0.5311597              -0.16384457
## 7                 0.5459942             0.5793126              -0.01138773
## 8                 0.4720191             0.4993427              -0.21700795
## 9                 0.4500998             0.4830171              -0.24140682
## 10                0.5262478             0.5935377               0.01940845
## 11                0.4491744             0.4816302              -0.21503001
## 12                0.4637941             0.5691423               0.10923414
## 13                0.4305839             0.4670011              -0.18053345
## 14                0.4536740             0.5800992               0.05581789
## 15                0.4128843             0.4763959              -0.15342957
## 16                0.4323847             0.4704532              -0.16090702
## 17                0.4323407             0.4689907              -0.17660125
## 18                0.3990150             0.4827507              -0.12926129
## 19                0.3666465             0.4737121              -0.10186128
## 20                0.2779568             0.4160728               0.25525233
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                -0.020861054                                -0.5915789
## 2                -0.057031272                                -0.4844852
## 3                -0.195261742                                -0.3401411
## 4                 0.031305000                                -0.4934016
## 5                -0.044018282                                -0.4645780
## 6                 0.046908733                                -0.4606777
## 7                 0.037879489                                -0.5188282
## 8                 0.053226750                                -0.3992715
## 9                 0.001688767                                -0.3516691
## 10                0.027615191                                -0.4856637
## 11               -0.055968602                                -0.3216140
## 12               -0.017429274                                -0.3868388
## 13               -0.218543081                                -0.3576092
## 14                0.029346500                                -0.3844161
## 15               -0.266645809                                -0.3674282
## 16               -0.139473041                                -0.3664399
## 17               -0.102542445                                -0.3284374
## 18               -0.325432645                                -0.3376598
## 19               -0.327155081                                -0.2853743
## 20               -0.243500329                                -0.1912888
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.3745712              2.617337e-02        0.094363213
## 2                     -0.3410204              1.612390e-02        0.065717590
## 3                     -0.3199229             -5.498808e-02        0.060860337
## 4                     -0.1680831             -8.696346e-02       -0.017205281
## 5                     -0.3740559             -9.887550e-03        0.065767668
## 6                     -0.1506485             -1.201066e-01       -0.035119085
## 7                     -0.1975101             -4.348857e-02       -0.025004158
## 8                     -0.1325913             -1.209405e-01       -0.046217325
## 9                     -0.1280096             -1.117693e-01       -0.051492836
## 10                    -0.2378094             -5.514954e-02       -0.022471592
## 11                    -0.1314974             -8.438546e-02       -0.005697268
## 12                    -0.3431068             -3.310497e-02        0.067488764
## 13                    -0.1193407             -1.915035e-02        0.009935865
## 14                    -0.3046761             -5.883176e-02        0.059906796
## 15                    -0.1199883             -7.801534e-03        0.028683036
## 16                    -0.1226651             -2.482316e-02        0.015423612
## 17                    -0.1344931             -5.581889e-02        0.018250417
```

```
## 18                   -0.1395798               6.659563e-03         0.067162756
## 19                   -0.1712571              -9.998825e-05         0.085457257
## 20                   -0.1936371              -7.432649e-02         0.008109032
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1          0.04743872 12 -269.7496 564.9238  0.000000 0.5256777832
## 2          0.06888444 12 -270.6270 566.6786  1.754847 0.2186047130
## 3          0.09495003 12 -271.3187 568.0620  3.138205 0.1094629257
## 4          0.18732329 12 -272.2356 569.8958  4.971987 0.0437589018
## 5          0.04406055 12 -272.6819 570.7884  5.864595 0.0280052152
## 6          0.21778468 12 -272.7780 570.9807  6.056933 0.0254374379
## 7          0.12024773 12 -273.3664 572.1576  7.233776 0.0141229401
## 8          0.22405348 12 -273.6817 572.7881  7.864355 0.0103037824
## 9          0.26657039 12 -273.8947 573.2141  8.290296 0.0083273102
## 10         0.10881300 12 -273.9343 573.2932  8.369465 0.0080041160
## 11         0.24999722 12 -275.3799 576.1845 11.260698 0.0018857757
## 12         0.03112278 12 -275.8137 577.0522 12.128376 0.0012220143
## 13         0.34950353 12 -275.9045 577.2336 12.309862 0.0011160072
## 14         0.06251461 12 -276.0899 577.6045 12.680747 0.0009271079
## 15         0.37661718 12 -276.2361 577.8968 12.973067 0.0008010395
## 16         0.30773340 12 -276.3047 578.0341 13.110305 0.0007479162
## 17         0.27658548 12 -276.3785 578.1816 13.257781 0.0006947507
## 18         0.37585967 12 -276.5576 578.5399 13.616081 0.0005807981
## 19         0.33599631 12 -277.5640 580.5526 15.628803 0.0002123088
## 20        -0.09731483 12 -278.2477 581.9201 16.996315 0.0001071560
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for fisher did not change but the weight did increase (0.266 - 0.526)

Model summary 1000m Let's print the summary for this model

```
summary(fisher_mods$`1000 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(fisher, absent_fisher) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    563.5    604.9   -269.7    539.5      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.2993   0.5471
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -3.09976    0.25568 -12.124  < 2e-16 ***
## scale(harvest)                    -0.24280    0.10744  -2.260  0.02383 *
## scale(seismic_lines)               0.02617    0.09586   0.273  0.78482
## scale(seismic_lines_3D)           -0.37457    0.16863  -2.221  0.02633 *
## scale(trails)                      0.09436    0.09191   1.027  0.30457
## scale(wells)                       0.04744    0.11174   0.425  0.67117
## scale(osm_industrial)             -0.02086    0.13613  -0.153  0.87821
## scale(pipeline_transmission_lines) -0.59158    0.19212  -3.079  0.00208 **
## scale(lc_grassland)                0.21162    0.11357   1.863  0.06240 .
## scale(lc_developed)                0.57274    0.12387   4.624 3.77e-06 ***
## scale(lc_forest)                   0.70616    0.17333   4.074 4.62e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Everything looks good

Subset models

Add later if deemed necessary.

Grey wolf

Global models

```
wolf_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%
```

```
    glmmTMB::glmmTMB(cbind(grey_wolf, absent_grey_wolf) ~

                         # HFI
                         scale(harvest) +
                         # scale(pipeline) +
                         # scale(roads) +
                         scale(seismic_lines) +
                         scale(seismic_lines_3D) +
                         scale(trails) +
                         # scale(transmission_lines) +
                         # scale(veg_edges) +
                         scale(wells) +
                         scale(osm_industrial) +
                         scale(pipeline_transmission_lines) +

                         # VEG covariates in numerical order
                         scale(lc_grassland) +
                         # scale(lc_coniferous) +
                         # scale(lc_broadleaf) +
                         # scale(lc_mixed) +
                         scale(lc_developed) +
                         # scale(lc_shrub) +
                         scale(lc_forest) +

                         # Random effect of array
                         (1|array),
                   data = .,
                   family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
wolf_global_model.sel <- model.sel(wolf_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
wolf_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  3500 meter buffer   -3.241080           +          0.194223499
## 2  3750 meter buffer   -3.240742           +          0.204414444
## 3  3250 meter buffer   -3.238011           +          0.180711889
## 4  4000 meter buffer   -3.228914           +          0.184087788
## 5  3000 meter buffer   -3.237625           +          0.188638735
## 6  4250 meter buffer   -3.210211           +          0.122195256
## 7  2250 meter buffer   -3.270770           +          0.136419264
## 8  2750 meter buffer   -3.244702           +          0.190325800
## 9  2000 meter buffer   -3.264612           +          0.104390609
## 10 2500 meter buffer   -3.253495           +          0.165116931
## 11 4500 meter buffer   -3.187178           +         -0.023640642
```

```
## 12 1750 meter buffer   -3.248178           +          0.076996670
## 13 4750 meter buffer   -3.190901           +          0.076094150
## 14 5000 meter buffer   -3.187831           +          0.077807145
## 15 1500 meter buffer   -3.223890           +          0.072341544
## 16 1250 meter buffer   -3.189647           +          0.081338565
## 17 1000 meter buffer   -3.172840           +          0.067841875
## 18  250 meter buffer   -3.212090           +          0.024559128
## 19  500 meter buffer   -3.158348           +         -0.002271218
## 20  750 meter buffer   -3.149762           +          0.069943373
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1               0.0266379039           0.130807264               0.56008161
## 2               0.0335144227           0.140248688               0.59059966
## 3               0.0108659999           0.127471908               0.49866982
## 4              -0.0353683654           0.145464449               0.58873246
## 5               0.0003671201           0.103144877               0.43691208
## 6              -0.0919821895           0.144099863               0.54808555
## 7               0.3242142495           0.197838750               0.26464102
## 8               0.0534622598           0.119950805               0.38445326
## 9               0.3375102315           0.218123998               0.21946042
## 10              0.1836977775           0.141899075               0.30712638
## 11             -0.2649012675           0.119859963               0.47802293
## 12              0.3188754516           0.206844464               0.15508903
## 13             -0.0583890028           0.153772791               0.51131542
## 14             -0.0061590542           0.155431790               0.50951315
## 15              0.2962775957           0.173769189               0.09912091
## 16              0.1545603950           0.093511055               0.08397110
## 17              0.0476211815           0.044090850               0.06164422
## 18             -0.1363311460          -0.016463872              -0.06204844
## 19             -0.0650089131          -0.110366774              -0.12392074
## 20             -0.0863735367          -0.008995222               0.04169226
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                  -0.18346209                              -0.163264614
## 2                  -0.17143390                              -0.204513608
## 3                  -0.17058828                              -0.111780422
## 4                  -0.13712128                              -0.239481764
## 5                  -0.17488419                              -0.073017237
## 6                  -0.09837332                              -0.237529692
## 7                  -0.33712071                              -0.016122713
## 8                  -0.18851666                              -0.053470219
## 9                  -0.37561093                               0.002686450
## 10                 -0.25878440                              -0.017175620
## 11                 -0.04722503                              -0.162486880
## 12                 -0.34617300                               0.007224684
## 13                 -0.08593282                              -0.306618307
## 14                 -0.09069219                              -0.359842722
## 15                 -0.32426234                              -0.000869049
## 16                 -0.27584786                              -0.006260505
## 17                 -0.19445813                               0.008650959
## 18                  0.07144123                               0.028608625
## 19                 -0.06535686                               0.016396072
## 20                 -0.10912202                               0.037890982
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.7268749              -0.024568131          0.09243789
## 2                     -0.6842827              -0.011821369          0.07760084
```

```
## 3                    -0.7746113              -0.012520438          0.10367713
## 4                    -0.6795778              -0.028688382          0.06075201
## 5                    -0.8349898               0.008779830          0.09972455
## 6                    -0.7003180              -0.034948126          0.07843057
## 7                    -0.8264356               0.038137186          0.19985299
## 8                    -0.8662871               0.026599838          0.11656928
## 9                    -0.8083138               0.019235289          0.22958289
## 10                   -0.8666287               0.044026474          0.16640039
## 11                   -0.8085713              -0.056138944          0.11889953
## 12                   -0.7935841              -0.067953607          0.22576119
## 13                   -0.6687589              -0.042685508          0.08722261
## 14                   -0.6356450              -0.039324251          0.08120180
## 15                   -0.7546658              -0.044327801          0.21076902
## 16                   -0.7537462              -0.002609989          0.15811719
## 17                   -0.7956591               0.046574433          0.10346726
## 18                   -1.1088557               0.013091754          0.22047546
## 19                   -0.8088531               0.067420837          0.10091086
## 20                   -0.8334033               0.116771888          0.03224661
##    cond(scale(wells)) df    logLik     AICc      delta       weight
## 1         -0.40673585 12 -241.0302 507.4851  0.0000000 3.026598e-01
## 2         -0.43915770 12 -241.2530 507.9307  0.4456331 2.422070e-01
## 3         -0.39186371 12 -241.8104 509.0454  1.5602873 1.387211e-01
## 4         -0.37871183 12 -241.9762 509.3770  1.8919361 1.175238e-01
## 5         -0.34473727 12 -242.9303 511.2853  3.8002410 4.526295e-02
## 6         -0.33626172 12 -243.2758 511.9763  4.4911795 3.204110e-02
## 7         -0.49617187 12 -243.4311 512.2869  4.8017985 2.743199e-02
## 8         -0.33948678 12 -243.5383 512.5013  5.0161828 2.464362e-02
## 9         -0.45732440 12 -243.6632 512.7510  5.2658950 2.175104e-02
## 10        -0.40503696 12 -243.9819 513.3885  5.9034177 1.581408e-02
## 11        -0.23365281 12 -244.1118 513.6482  6.1631581 1.388807e-02
## 12        -0.42117886 12 -244.6348 514.6942  7.2091136 8.232193e-03
## 13        -0.29135251 12 -245.2127 515.8501  8.3649668 4.618758e-03
## 14        -0.29237162 12 -245.8419 517.1085  9.6234240 2.461811e-03
## 15        -0.36824066 12 -245.9238 517.2723  9.7871929 2.268260e-03
## 16        -0.28705839 12 -247.9546 521.3339 13.8488585 2.976551e-04
## 17        -0.27582360 12 -249.3812 524.1871 16.7020003 7.147619e-05
## 18        -0.06846463 12 -249.7165 524.8576 17.3725038 5.111677e-05
## 19        -0.30235731 12 -249.8445 525.1136 17.6285078 4.497517e-05
## 20        -0.12850018 12 -251.4317 528.2880 20.8029240 9.197231e-06
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
```

```
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for wolf did change (500 - 3500), as with coyote and caribou this is more similar to top anthro modelthan it was before AND actually the same as the landscape model

Model summary 3500m Let's get the model summary for this model

```
summary(wolf_mods$`3500 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(grey_wolf, absent_grey_wolf) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    506.1    547.4   -241.0    482.1      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1303   0.3609
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -3.24108    0.19902 -16.285  < 2e-16 ***
## scale(harvest)                     0.19422    0.16181   1.200 0.230006
## scale(seismic_lines)              -0.02457    0.13065  -0.188 0.850840
## scale(seismic_lines_3D)           -0.72687    0.29274  -2.483 0.013026 *
## scale(trails)                      0.09244    0.11377   0.812 0.416513
## scale(wells)                      -0.40674    0.27482  -1.480 0.138866
## scale(osm_industrial)             -0.18346    0.16334  -1.123 0.261349
## scale(pipeline_transmission_lines) -0.16326    0.19750  -0.827 0.408440
## scale(lc_grassland)                0.56008    0.15213   3.682 0.000232 ***
## scale(lc_developed)                0.02664    0.28028   0.095 0.924282
## scale(lc_forest)                   0.13081    0.19835   0.659 0.509581
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Subset models

Add later if deemed necessary.

Lynx

Global models

```
lynx_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

     glmmTMB::glmmTMB(cbind(lynx, absent_lynx) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
# run model selection and save the results as a tibble for graphing use later
lynx_global_model.sel <- model.sel(lynx_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
lynx_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
```

```
## 1   250 meter buffer   -1.898428           +         -0.302883298
## 2   500 meter buffer   -1.884363           +         -0.217519901
## 3  3250 meter buffer   -1.887696           +          0.169175844
## 4  3000 meter buffer   -1.887112           +          0.159699656
## 5  1000 meter buffer   -1.872601           +         -0.106823652
## 6  3500 meter buffer   -1.886305           +          0.172124772
## 7  1250 meter buffer   -1.873243           +         -0.089823808
## 8   750 meter buffer   -1.874501           +         -0.149993834
## 9  3750 meter buffer   -1.885218           +          0.175625754
## 10 2750 meter buffer   -1.883255           +          0.138658729
## 11 1500 meter buffer   -1.872891           +         -0.059072154
## 12 4250 meter buffer   -1.881576           +          0.180076272
## 13 4000 meter buffer   -1.882743           +          0.177327069
## 14 5000 meter buffer   -1.874674           +          0.129414209
## 15 4750 meter buffer   -1.875063           +          0.145949568
## 16 4500 meter buffer   -1.877739           +          0.170751654
## 17 2000 meter buffer   -1.872898           +          0.040975998
## 18 2500 meter buffer   -1.877010           +          0.105105374
## 19 1750 meter buffer   -1.870128           +         -0.006825369
## 20 2250 meter buffer   -1.873867           +          0.079934227
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1                -0.13488291          -0.018564129                0.01084969
## 2                -0.09147433           0.046896290                0.16345357
## 3                 0.14348731           0.012155250                0.16577245
## 4                 0.16203947           0.020307082                0.17251629
## 5                -0.01345159           0.077618131                0.13004422
## 6                 0.13143181          -0.000369871                0.13617464
## 7                 0.07801732           0.087879200                0.14980992
## 8                -0.04746628           0.075406071                0.12483804
## 9                 0.12877075          -0.006577550                0.09997960
## 10                0.16534886           0.021998047                0.16321158
## 11                0.14371928           0.090204396                0.14671108
## 12                0.09101602           0.002492953                0.06487366
## 13                0.10850337          -0.003900798                0.07577853
## 14                0.04349666           0.029509102               -0.01200343
## 15                0.05866018           0.019609829               -0.01060870
## 16                0.06515203           0.012416363                0.02204842
## 17                0.18325868           0.080492043                0.10730576
## 18                0.17998278           0.038024831                0.11925861
## 19                0.15908625           0.083789380                0.10560206
## 20                0.18793882           0.060207261                0.09285951
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                   0.100523215                                0.003730106
## 2                   0.122347187                               -0.090041949
## 3                   0.008736217                               -0.103725280
## 4                   0.002711300                               -0.105178103
## 5                   0.081673167                               -0.035018474
## 6                   0.001590802                               -0.087949023
## 7                   0.043882423                               -0.051076813
## 8                   0.107609728                               -0.045020960
## 9                   0.000424834                               -0.079782167
## 10                  0.003943819                               -0.092814796
## 11                  0.019547119                               -0.051093766
## 12                  0.025073675                               -0.076595967
```

```
## 13                0.010306852                            -0.081497476
## 14                0.078185372                            -0.074204591
## 15                0.062992452                            -0.064440637
## 16                0.047234375                            -0.076090157
## 17                0.014519556                            -0.046310037
## 18                0.007504534                            -0.082022647
## 19                0.018253166                            -0.038411426
## 20                0.014027316                            -0.068024487
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     0.07553842               -0.172539784          0.04983365
## 2                     0.07648397               -0.090876141         -0.01788460
## 3                     0.11650660               -0.137848940          0.09311238
## 4                     0.12049832               -0.106940450          0.09253440
## 5                     0.07623316               -0.001316084          0.10100901
## 6                     0.10397270               -0.158838354          0.09991898
## 7                     0.09034415                0.061223149          0.11412406
## 8                     0.07205674               -0.038617170          0.03278183
## 9                     0.09587350               -0.175627853          0.09758097
## 10                    0.11958993               -0.083159007          0.08370259
## 11                    0.10155633                0.067106873          0.12622060
## 12                    0.09533020               -0.177536746          0.09542228
## 13                    0.09408521               -0.175549224          0.09327548
## 14                    0.10169883               -0.167344366          0.09053452
## 15                    0.09295081               -0.163797883          0.09616824
## 16                    0.09271259               -0.169661588          0.09431668
## 17                    0.10571156                0.026895854          0.11284790
## 18                    0.11113085               -0.062454010          0.08584787
## 19                    0.10152780                0.043044697          0.11681140
## 20                    0.10682084               -0.035860471          0.09529744
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1          -0.2172774 12 -436.0433 897.5113  0.000000 9.818848e-01
## 2          -0.2136561 12 -440.2247 905.8741  8.362756 1.500068e-02
## 3          -0.2843353 12 -443.6425 912.7096 15.198320 4.917986e-04
## 4          -0.3064524 12 -443.6498 912.7243 15.212972 4.882090e-04
## 5          -0.2315831 12 -444.1067 913.6381 16.126814 3.091486e-04
## 6          -0.2578488 12 -444.1116 913.6479 16.136540 3.076489e-04
## 7          -0.2692156 12 -444.2216 913.8678 16.356502 2.756076e-04
## 8          -0.1956949 12 -444.4866 914.3978 16.886454 2.114529e-04
## 9          -0.2372007 12 -444.5271 914.4788 16.967488 2.030567e-04
## 10         -0.3160955 12 -444.6874 914.7995 17.288225 1.729697e-04
## 11         -0.2865068 12 -444.8712 915.1671 17.655828 1.439283e-04
## 12         -0.1956705 12 -445.1794 915.7835 18.272220 1.057544e-04
## 13         -0.2017007 12 -445.3383 916.1012 18.589894 9.022281e-05
## 14         -0.1283398 12 -445.7359 916.8965 19.385148 6.062183e-05
## 15         -0.1401878 12 -445.7923 917.0092 19.497911 5.729847e-05
## 16         -0.1480991 12 -445.8236 917.0719 19.560561 5.553141e-05
## 17         -0.3164331 12 -445.9946 917.4138 19.902508 4.680430e-05
## 18         -0.3023679 12 -446.2781 917.9808 20.469475 3.525092e-05
## 19         -0.2939078 12 -446.3129 918.0504 20.539115 3.404460e-05
## 20         -0.2986471 12 -446.6142 918.6530 21.141701 2.518828e-05
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
```

```
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for lynx did change (1250 - 250), as with others this is more similar to top anthro model (acutally the same as anthro model).

Model summary 250m Let's get the model summary

```
summary(lynx_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(lynx, absent_lynx) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    896.1    937.4   -436.0    872.1      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1573   0.3967
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -1.89843    0.17291 -10.980  < 2e-16 ***
## scale(harvest)                   -0.30288    0.09009  -3.362 0.000774 ***
## scale(seismic_lines)             -0.17254    0.07219  -2.390 0.016840 *
## scale(seismic_lines_3D)           0.07554    0.05476   1.379 0.167755
## scale(trails)                     0.04983    0.05307   0.939 0.347748
## scale(wells)                     -0.21728    0.07372  -2.947 0.003204 **
```

```
## scale(osm_industrial)               0.10052    0.05392   1.864 0.062295 .
## scale(pipeline_transmission_lines)  0.00373    0.06477   0.058 0.954079
## scale(lc_grassland)                 0.01085    0.07148   0.152 0.879349
## scale(lc_developed)                -0.13488    0.07953  -1.696 0.089878 .
## scale(lc_forest)                   -0.01856    0.07703  -0.241 0.809563
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Everything looks good so far

Subset models

Add later if deemed necessary.

Moose

Global model

```
moose_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

     glmmTMB::glmmTMB(cbind(moose, absent_moose) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                      data = .,
```

```
                  family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
moose_global_model.sel <- model.sel(moose_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
moose_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1   250 meter buffer   -1.849716           +         -0.010623977
## 2   500 meter buffer   -1.845364           +          0.007943112
## 3   750 meter buffer   -1.842638           +          0.022287853
## 4  1000 meter buffer   -1.839316           +          0.002888910
## 5  1250 meter buffer   -1.831966           +          0.022185769
## 6  1500 meter buffer   -1.832178           +          0.017689199
## 7  1750 meter buffer   -1.829419           +          0.005967840
## 8  2500 meter buffer   -1.827361           +         -0.020626181
## 9  2750 meter buffer   -1.827564           +         -0.020803203
## 10 3000 meter buffer   -1.827156           +         -0.024106844
## 11 3250 meter buffer   -1.824838           +         -0.033285815
## 12 2000 meter buffer   -1.826526           +         -0.010088945
## 13 2250 meter buffer   -1.826150           +         -0.008670335
## 14 3500 meter buffer   -1.822505           +         -0.046408038
## 15 3750 meter buffer   -1.820495           +         -0.058906593
## 16 4000 meter buffer   -1.817861           +         -0.067566986
## 17 5000 meter buffer   -1.812549           +         -0.107067715
## 18 4250 meter buffer   -1.814628           +         -0.080320286
## 19 4750 meter buffer   -1.812394           +         -0.100470793
## 20 4500 meter buffer   -1.812833           +         -0.092628237
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1               -0.400092436           -0.28907134              -0.113092601
## 2               -0.247704490           -0.20865494              -0.028001319
## 3               -0.171449659           -0.12682403               0.012860985
## 4               -0.171733651           -0.15426764              -0.005877747
## 5               -0.153240510           -0.16737297               0.026716591
## 6               -0.078239025           -0.16386674               0.026176801
## 7               -0.067752008           -0.15424500               0.037204866
## 8               -0.011378206           -0.13542131              -0.005974313
## 9                0.003767231           -0.13334998               0.007117298
## 10               0.032777308           -0.12002779              -0.015150884
## 11               0.047794505           -0.10987427              -0.029267529
## 12              -0.018673900           -0.13426643               0.015523748
## 13              -0.025813352           -0.14134645              -0.004633224
## 14               0.047442646           -0.10794539              -0.053202040
## 15               0.046925176           -0.10971632              -0.085059615
## 16               0.043235063           -0.09741685              -0.113922120
## 17               0.047949825           -0.06937349              -0.227470774
```

```
## 18               0.041375005            -0.08017510              -0.123150351
## 19               0.062711363            -0.06349064              -0.160060453
## 20               0.043464043            -0.07208039              -0.128062905
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                   -0.1807698                                0.04241050
## 2                   -0.2957918                                0.07822214
## 3                   -0.3382014                                0.10344898
## 4                   -0.3006834                                0.10277282
## 5                   -0.2423918                                0.10130518
## 6                   -0.2504926                                0.10059183
## 7                   -0.1992674                                0.11300329
## 8                   -0.2374117                                0.20235694
## 9                   -0.2614937                                0.21296301
## 10                  -0.2740314                                0.21857662
## 11                  -0.2843232                                0.21815978
## 12                  -0.1878836                                0.13048425
## 13                  -0.2027139                                0.16861613
## 14                  -0.2894988                                0.22713706
## 15                  -0.2964070                                0.23506486
## 16                  -0.2822088                                0.22773893
## 17                  -0.1979284                                0.25777804
## 18                  -0.2577785                                0.20753102
## 19                  -0.2124468                                0.20998896
## 20                  -0.2348222                                0.19885008
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                  0.1464444607                0.105069523          0.08050523
## 2                  0.1163658557                0.073922718          0.05705261
## 3                  0.0916389522                0.003155747          0.04281248
## 4                  0.0680339795               -0.065833553          0.06115504
## 5                  0.0580559161               -0.046035692          0.07610496
## 6                  0.0567591650               -0.060175421          0.08446437
## 7                  0.0397587118               -0.068108737          0.06578645
## 8                 -0.0171895911               -0.014274674          0.08757601
## 9                 -0.0211132958               -0.014686468          0.08737380
## 10                -0.0239053783               -0.032256841          0.09076933
## 11                -0.0276048704               -0.032759875          0.09238102
## 12                 0.0213802946               -0.037162312          0.06642959
## 13                -0.0006894857               -0.022439828          0.07148073
## 14                -0.0353599264               -0.031967269          0.09331236
## 15                -0.0442201256               -0.028397004          0.09740065
## 16                -0.0519282356               -0.028003814          0.10050589
## 17                -0.0745985386                0.026541061          0.10798249
## 18                -0.0498499039               -0.014473141          0.10404817
## 19                -0.0522331227                0.020226622          0.10470023
## 20                -0.0486323061                0.003291123          0.10496804
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1         -0.03034155 12 -427.0835 879.5916  0.000000 9.408427e-01
## 2         -0.05981923 12 -430.5929 886.6104  7.018784 2.814541e-02
## 3         -0.12909875 12 -431.1488 887.7223  8.130656 1.614237e-02
## 4         -0.16432321 12 -431.3512 888.1271  8.535477 1.318444e-02
## 5         -0.14408258 12 -434.2785 893.9816 14.389964 7.059534e-04
## 6         -0.19605163 12 -434.9712 895.3671 15.775486 3.531130e-04
## 7         -0.25988910 12 -435.7040 896.8326 17.241033 1.696970e-04
## 8         -0.27095540 12 -436.4442 898.3131 18.721480 8.094671e-05
```

```
## 9         -0.25671814 12 -436.6093 898.6433 19.051689 6.862703e-05
## 10        -0.26306339 12 -436.6754 898.7755 19.183847 6.423882e-05
## 11        -0.24348919 12 -437.0982 899.6211 20.029473 4.208936e-05
## 12        -0.29839896 12 -437.1559 899.7365 20.144839 3.973021e-05
## 13        -0.27641726 12 -437.2031 899.8309 20.239324 3.789690e-05
## 14        -0.21965224 12 -437.2852 899.9951 20.403517 3.490998e-05
## 15        -0.17824883 12 -437.4496 900.3240 20.732340 2.961737e-05
## 16        -0.15067915 12 -437.7959 901.0164 21.424760 2.095027e-05
## 17        -0.14820778 12 -438.0026 901.4298 21.838219 1.703758e-05
## 18        -0.13819369 12 -438.6585 902.7417 23.150119 8.841632e-06
## 19        -0.17151672 12 -438.9892 903.4031 23.811493 6.352095e-06
## 20        -0.14419802 12 -439.2096 903.8438 24.252202 5.095869e-06
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for moose did not change but the weight did increase (0.504 - 0.941). This top model is the same as the landscape model.

Model summary 250m Let's get the model summary

```
summary(moose_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(moose, absent_moose) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    878.2    919.5   -427.1    854.2      220
```

```
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.3014   0.549
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -1.84972    0.23284  -7.944 1.95e-15 ***
## scale(harvest)                   -0.01062    0.06092  -0.174 0.861546
## scale(seismic_lines)              0.10507    0.06305   1.666 0.095643 .
## scale(seismic_lines_3D)           0.14644    0.05514   2.656 0.007909 **
## scale(trails)                     0.08051    0.05258   1.531 0.125766
## scale(wells)                     -0.03034    0.06728  -0.451 0.652026
## scale(osm_industrial)            -0.18077    0.07146  -2.530 0.011420 *
## scale(pipeline_transmission_lines)  0.04241    0.06953   0.610 0.541879
## scale(lc_grassland)              -0.11309    0.06932  -1.631 0.102817
## scale(lc_developed)              -0.40009    0.08750  -4.572 4.82e-06 ***
## scale(lc_forest)                 -0.28907    0.08148  -3.548 0.000388 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Looks good for now

Subset models

Add later if deemed necessary

Red fox

Global models

```
fox_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

     glmmTMB::glmmTMB(cbind(red_fox, absent_red_fox) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
```

```
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))


# run model selection and save the results as a tibble for graphing use later
fox_global_model.sel <- model.sel(fox_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
fox_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  4750 meter buffer   -4.710660           +         0.1719449706
## 2  5000 meter buffer   -4.438342           +         0.1962039685
## 3  4500 meter buffer   -4.704824           +         0.1479952639
## 4  1500 meter buffer   -4.533097           +        -0.2459263767
## 5  1750 meter buffer   -4.593444           +        -0.1464567284
## 6  4250 meter buffer   -4.582045           +         0.1111945290
## 7  4000 meter buffer   -4.307353           +         0.0898085271
## 8  1250 meter buffer   -4.413307           +        -0.2390297495
## 9  3750 meter buffer   -4.259312           +         0.0325518386
## 10 2000 meter buffer   -4.463057           +        -0.0959096352
## 11 2250 meter buffer   -4.206357           +        -0.0516943115
## 12 2500 meter buffer   -4.185386           +        -0.0026727758
## 13 1000 meter buffer   -4.423257           +        -0.2049562281
## 14 2750 meter buffer   -4.165725           +         0.0474697062
## 15  500 meter buffer   -4.331585           +        -0.1191625937
## 16 3500 meter buffer   -4.159307           +         0.0000326624
## 17 3000 meter buffer   -4.150423           +         0.0533881937
## 18 3250 meter buffer   -4.148094           +         0.0346435735
## 19  750 meter buffer   -4.350467           +        -0.2256966123
## 20  250 meter buffer   -4.314438           +        -0.0755865998
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
```

```
## 1                  0.3922000           -0.79371553                0.4438821
## 2                  0.6080074           -0.62825324                0.4975331
## 3                  0.3742119           -0.79675787                0.3943830
## 4                  0.4050962           -0.12634316                0.5745372
## 5                  0.2375217           -0.16775374                0.5892874
## 6                  0.2584423           -0.76448355                0.2998805
## 7                  0.5899049           -0.45664133                0.4669497
## 8                  0.4211023           -0.10873136                0.5176597
## 9                  0.6150069           -0.31440316                0.5523548
## 10                 0.3378334           -0.10968856                0.6123616
## 11                 0.7396066           -0.01560293                0.7021326
## 12                 0.7367246           -0.06084646                0.6474266
## 13                 0.5975677           -0.04113826                0.5440339
## 14                 0.6695471           -0.11748636                0.5551106
## 15                 0.6214101            0.36552881                0.3195532
## 16                 0.6637446           -0.20005475                0.5829544
## 17                 0.6574876           -0.13367496                0.5417045
## 18                 0.5885693           -0.13511957                0.5874901
## 19                 0.4363296            0.11421495                0.5122062
## 20                 0.5478319            0.14059910                0.2997906
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                  0.211508231                                0.85133918
## 2                  0.024189383                                0.36202796
## 3                  0.204850332                                0.83388673
## 4                  0.167290751                               -0.26360179
## 5                  0.303308901                               -0.09302959
## 6                  0.039526359                                0.75986990
## 7                 -0.497790096                                0.20961822
## 8                  0.165419273                               -0.58176206
## 9                 -0.567369169                                0.10938966
## 10                 0.234716202                               -0.22122562
## 11                -0.006097739                               -0.47668309
## 12                -0.019592999                               -0.39525624
## 13                 0.130117162                               -0.92467132
## 14                -0.053907992                               -0.29316939
## 15                 0.363842690                               -0.34869971
## 16                -0.253684805                               -0.05723367
## 17                -0.108209737                               -0.18213745
## 18                -0.139049284                               -0.17195934
## 19                 0.221464242                               -0.37059674
## 20                 0.192562170                               -0.40468512
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.64074029              -0.566613856         -0.8160982
## 2                     -1.29237310              -0.236530689         -0.8335380
## 3                     -0.59594635              -0.517839825         -0.8812319
## 4                     -0.38786539              -0.121790744         -0.7753648
## 5                     -0.47177828              -0.191262989         -0.8771643
## 6                     -0.72239778              -0.414262359         -0.7283896
## 7                     -1.01320298              -0.054081886         -0.6360047
## 8                     -0.39443681              -0.092135933         -0.7080355
## 9                     -0.89167770              -0.001070778         -0.4701606
## 10                    -0.50567915              -0.141011833         -0.7710160
## 11                    -0.65924372               0.162949023         -0.6500051
## 12                    -0.62464367               0.136696676         -0.6584918
```

```
## 13                  -0.27191167               -0.067027615          -0.6733136
## 14                  -0.65566887                0.110873539          -0.6456739
## 15                  -0.25272386               -0.173350818          -0.6186467
## 16                  -0.69819491                0.040427738          -0.4202069
## 17                  -0.64932567                0.102741540          -0.5829771
## 18                  -0.70826784                0.088240847          -0.4496556
## 19                  -0.16543996               -0.052969716          -0.6205387
## 20                   0.04142461               -0.087317739          -0.8328836
##    cond(scale(wells)) df    logLik     AICc       delta       weight
## 1          -0.8512377 12 -163.4806 352.3859  0.00000000 3.667349e-01
## 2          -0.4373352 12 -163.4941 352.4129  0.02703244 3.618113e-01
## 3          -0.8090613 12 -164.1207 353.6660  1.28007890 1.933689e-01
## 4          -0.5035217 12 -165.9899 357.4045  5.01857219 2.982518e-02
## 5          -0.4425342 12 -166.5185 358.4616  6.07571924 1.758031e-02
## 6          -0.4445907 12 -166.7989 359.0224  6.63645281 1.328204e-02
## 7          -0.1573813 12 -167.5841 360.5929  8.20696478 6.056649e-03
## 8          -0.1627423 12 -168.5672 362.5591 10.17320378 2.266048e-03
## 9          -0.1406339 12 -168.6182 362.6611 10.27517847 2.153404e-03
## 10         -0.2947945 12 -168.7688 362.9622 10.57633247 1.852383e-03
## 11         -0.1999088 12 -168.8712 363.1670 10.78114201 1.672080e-03
## 12         -0.2297243 12 -169.1973 363.8192 11.43326206 1.206843e-03
## 13         -0.1583967 12 -169.2443 363.9133 11.52736861 1.151373e-03
## 14         -0.1508334 12 -170.5275 366.4796 14.09369785 3.191131e-04
## 15          0.2243238 12 -170.8717 367.1680 14.78206875 2.261865e-04
## 16         -0.2561072 12 -170.9352 367.2951 14.90922512 2.122536e-04
## 17         -0.1699877 12 -171.4053 368.2353 15.84943374 1.326452e-04
## 18         -0.1065284 12 -171.8659 369.1564 16.77051761 8.369135e-05
## 19         -0.1641260 12 -172.4404 370.3054 17.91950105 4.711747e-05
## 20          0.1147348 12 -173.4257 372.2761 19.89023178 1.758908e-05
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for fox did not change but the weight dropped (0.855 - 0.367). This top model is the same as the landscape model.

Model summary 4750m Let's get the model summary

```
summary(fox_mods$`4750 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(red_fox, absent_red_fox) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    351.0    392.3   -163.5    327.0      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 1.334    1.155
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -4.7107     0.6049  -7.788 6.83e-15 ***
## scale(harvest)                     0.1719     0.2779   0.619  0.53603
## scale(seismic_lines)              -0.5666     0.2969  -1.909  0.05632 .
## scale(seismic_lines_3D)           -0.6407     0.6589  -0.972  0.33086
## scale(trails)                     -0.8161     0.4000  -2.040  0.04131 *
## scale(wells)                      -0.8512     0.4063  -2.095  0.03617 *
## scale(osm_industrial)              0.2115     0.3085   0.686  0.49298
## scale(pipeline_transmission_lines)  0.8513     0.4980   1.710  0.08736 .
## scale(lc_grassland)                0.4439     0.2471   1.796  0.07242 .
## scale(lc_developed)                0.3922     0.3527   1.112  0.26617
## scale(lc_forest)                  -0.7937     0.2942  -2.698  0.00697 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Looks good so far

Subset Models

Add later if deemed necessary.

White-tailed deer

Global models

```
deer_mods <- osm_final_df_2021_2022 %>%

  # use purrr map to make global models for all other buffer sizes
  purrr::map(
    ~.x %>%

      # have to include the `` around the white-tailed_deer or R won't recognize it as a variable becau
     glmmTMB::glmmTMB(cbind(`white-tailed_deer`, `absent_white-tailed_deer`) ~

                        # HFI
                        scale(harvest) +
                        # scale(pipeline) +
                        # scale(roads) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        # scale(transmission_lines) +
                        # scale(veg_edges) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # VEG covariates in numerical order
                        scale(lc_grassland) +
                        # scale(lc_coniferous) +
                        # scale(lc_broadleaf) +
                        # scale(lc_mixed) +
                        scale(lc_developed) +
                        # scale(lc_shrub) +
                        scale(lc_forest) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))


# run model selection and save the results as a tibble for graphing use later
deer_global_model.sel <- model.sel(deer_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_final_df_2021_2022)))

# look at model selection results
deer_global_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  1500 meter buffer  -0.2118018           +          -0.08697915
## 2  4000 meter buffer  -0.2228334           +          -0.20497662
## 3  4500 meter buffer  -0.2264998           +          -0.23139410
```

```
## 4  4250 meter buffer  -0.2244901           +          -0.22030347
## 5  3750 meter buffer  -0.2191663           +          -0.17859059
## 6  4750 meter buffer  -0.2282169           +          -0.25375531
## 7  3500 meter buffer  -0.2173055           +          -0.15909038
## 8  5000 meter buffer  -0.2304881           +          -0.27597320
## 9  1250 meter buffer  -0.2057161           +          -0.08675304
## 10 1750 meter buffer  -0.2067686           +          -0.09977419
## 11 3250 meter buffer  -0.2116719           +          -0.13329466
## 12 3000 meter buffer  -0.2055097           +          -0.11846276
## 13 2000 meter buffer  -0.2057676           +          -0.10146795
## 14 2250 meter buffer  -0.2033035           +          -0.11370795
## 15 2750 meter buffer  -0.2032782           +          -0.10732094
## 16 2500 meter buffer  -0.2043346           +          -0.10765711
## 17 1000 meter buffer  -0.1895958           +          -0.11220105
## 18  750 meter buffer  -0.1780650           +          -0.07972438
## 19  500 meter buffer  -0.1829497           +          -0.01119863
## 20  250 meter buffer  -0.1844108           +           0.04014501
##    cond(scale(lc_developed)) cond(scale(lc_forest)) cond(scale(lc_grassland))
## 1               -0.183436547           -0.06969916               0.14372747
## 2               -0.087805884           -0.08854160               0.30653918
## 3               -0.031435450           -0.09763181               0.35504370
## 4               -0.066398213           -0.09771188               0.32574618
## 5               -0.110402993           -0.09364101               0.27871118
## 6                0.022210383           -0.08592438               0.35768242
## 7               -0.117084906           -0.09228863               0.26598138
## 8                0.043026444           -0.08800771               0.35136995
## 9               -0.126352063           -0.05513991               0.13263971
## 10              -0.172632016           -0.07167220               0.14085256
## 11              -0.129052360           -0.09846753               0.25219343
## 12              -0.120867001           -0.10008698               0.22913746
## 13              -0.151532753           -0.07911291               0.15437899
## 14              -0.137393389           -0.08353000               0.16053268
## 15              -0.120869553           -0.09510206               0.20523539
## 16              -0.123086455           -0.09080592               0.18918194
## 17              -0.016106577           -0.01412560               0.08257206
## 18               0.033503242           -0.01587439               0.05410520
## 19               0.019784091           -0.05613558               0.05070313
## 20              -0.004756961           -0.11237215               0.02677214
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                    0.3569561                              0.046930837
## 2                    0.2042716                             -0.086433912
## 3                    0.1960369                             -0.148546289
## 4                    0.1993662                             -0.113231511
## 5                    0.2060455                             -0.039671964
## 6                    0.1891270                             -0.203305588
## 7                    0.2096988                             -0.034266880
## 8                    0.1822756                             -0.253638260
## 9                    0.3665603                              0.036414972
## 10                   0.3224017                              0.059218440
## 11                   0.2181027                             -0.011204600
## 12                   0.2272504                             -0.001410895
## 13                   0.2878043                              0.039226254
## 14                   0.2665277                              0.020149450
## 15                   0.2355525                              0.003639011
```

```
## 16                   0.2470561                                0.008483685
## 17                   0.3388702                                0.052858673
## 18                   0.3299672                                0.098927530
## 19                   0.3135973                                0.116682599
## 20                   0.2171382                                0.093551391
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.4547173                -0.08996001          0.16851613
## 2                     -0.3458370                -0.04041973          0.22183271
## 3                     -0.2999718                -0.04713276          0.20995443
## 4                     -0.3221222                -0.05741096          0.21309017
## 5                     -0.3671571                -0.03409730          0.22323381
## 6                     -0.2724234                -0.02822194          0.20937160
## 7                     -0.3659442                -0.03980814          0.21630321
## 8                     -0.2488776                -0.02063040          0.21272948
## 9                     -0.4348374                -0.07623988          0.15513595
## 10                    -0.4230096                -0.10163402          0.16782175
## 11                    -0.3567852                -0.02783274          0.20691023
## 12                    -0.3422108                -0.01609253          0.18994593
## 13                    -0.4014505                -0.09492539          0.16499698
## 14                    -0.3672114                -0.05274901          0.17912160
## 15                    -0.3422297                -0.02910964          0.18797904
## 16                    -0.3449757                -0.04236590          0.18781294
## 17                    -0.3421214                -0.06818239          0.14888925
## 18                    -0.2814914                -0.01456565          0.08956161
## 19                    -0.2644586                 0.03644965          0.10641335
## 20                    -0.2163116                -0.02892613          0.19492800
##    cond(scale(wells)) df    logLik     AICc      delta       weight
## 1          0.35761341 12 -559.5356 1144.496  0.0000000 2.336945e-01
## 2          0.35280602 12 -559.6139 1144.652  0.1566346 2.160906e-01
## 3          0.31476467 12 -559.7405 1144.906  0.4098307 1.903947e-01
## 4          0.33546987 12 -559.8449 1145.114  0.6186809 1.715156e-01
## 5          0.36811564 12 -560.4516 1146.328  1.8320648 9.350197e-02
## 6          0.30114312 12 -561.6082 1148.641  4.1452665 2.941138e-02
## 7          0.37020871 12 -562.0942 1149.613  5.1172155 1.809086e-02
## 8          0.31845188 12 -562.3844 1150.194  5.6977115 1.353338e-02
## 9          0.31896401 12 -562.4469 1150.318  5.8226493 1.271383e-02
## 10         0.34862826 12 -562.5917 1150.608  6.1122272 1.100007e-02
## 11         0.37930253 12 -562.9868 1151.398  6.9024579 7.409670e-03
## 12         0.38040277 12 -565.0241 1155.473 10.9770750 9.660665e-04
## 13         0.35382030 12 -565.3652 1156.155 11.6592803 6.868597e-04
## 14         0.37886200 12 -565.9253 1157.275 12.7795451 3.922885e-04
## 15         0.38338991 12 -566.0391 1157.503 13.0071132 3.500981e-04
## 16         0.36992273 12 -566.3869 1158.198 13.7026388 2.472626e-04
## 17         0.27620629 12 -572.1280 1169.681 25.1847907 7.940370e-07
## 18         0.25604451 12 -583.5892 1192.603 48.1072488 8.361655e-12
## 19         0.24464000 12 -588.8826 1203.190 58.6940411 4.201453e-14
## 20         0.06152785 12 -595.6261 1216.677 72.1809510 4.951699e-17
##                   Dataset
## 1  osm_final_df_2021_2022
## 2  osm_final_df_2021_2022
## 3  osm_final_df_2021_2022
## 4  osm_final_df_2021_2022
## 5  osm_final_df_2021_2022
## 6  osm_final_df_2021_2022
```

```
## 7  osm_final_df_2021_2022
## 8  osm_final_df_2021_2022
## 9  osm_final_df_2021_2022
## 10 osm_final_df_2021_2022
## 11 osm_final_df_2021_2022
## 12 osm_final_df_2021_2022
## 13 osm_final_df_2021_2022
## 14 osm_final_df_2021_2022
## 15 osm_final_df_2021_2022
## 16 osm_final_df_2021_2022
## 17 osm_final_df_2021_2022
## 18 osm_final_df_2021_2022
## 19 osm_final_df_2021_2022
## 20 osm_final_df_2021_2022
```

Note about re-run global models: when we re-ran the global models with dropped correlated variables our top model for deer did not change but the weight dropped (0.974 - 0.234). This top model is the same as the landscape model.

Model summary 1500m Let's get the model summary

```
summary(deer_mods$`1500 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(‘white-tailed_deer‘, ‘absent_white-tailed_deer‘) ~ scale(harvest) +
##     scale(seismic_lines) + scale(seismic_lines_3D) + scale(trails) +
##     scale(wells) + scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     scale(lc_grassland) + scale(lc_developed) + scale(lc_forest) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##   1143.1   1184.4   -559.5   1119.1      220
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 1.974    1.405
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -0.21180    0.57592  -0.368  0.71305
## scale(harvest)                   -0.08698    0.06553  -1.327  0.18440
## scale(seismic_lines)             -0.08996    0.06637  -1.355  0.17530
## scale(seismic_lines_3D)          -0.45472    0.09650  -4.712 2.45e-06 ***
## scale(trails)                     0.16852    0.05189   3.247  0.00117 **
## scale(wells)                      0.35761    0.07963   4.491 7.10e-06 ***
## scale(osm_industrial)             0.35696    0.05902   6.048 1.47e-09 ***
## scale(pipeline_transmission_lines)  0.04693    0.08430   0.557  0.57771
## scale(lc_grassland)               0.14373    0.06313   2.277  0.02281 *
```

```
## scale(lc_developed)                -0.18344    0.08172  -2.245  0.02479 *
## scale(lc_forest)                   -0.06970    0.05879  -1.186  0.23578
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Looks okay for now

Subset models

Add later if deemed necessary.

Top buffers

For convenience here is a list of the top buffer widths for each species

Initial run
* Black bear - 250m
* Caribou - 3000m
* Coyote - 2250m
* Fisher - 1000m
* Grey wolf - 500m
* Lynx - 1250m
* Moose - 250m
* Red fox - 4750
* White-tailed deer - 1500m

Re-run (without correlated variables)
* Black bear - 250m
* Caribou - 1000m
* Coyote - 5000m
* Fisher - 1000m
* Grey wolf - 3500m
* Lynx - 250m
* Moose - 250m
* Red fox - 4750m
* White-tailed deer - 1500m

I also want to create a data frame with the model.sel outputs from each species so I can plot these results later

```
# provide list of model.sel data frames from global analysis

global_model.sel_data <- list(
  bear = black_bear_global_model.sel,
  caribou = caribou_global_model.sel,
  coyote = coyote_global_model.sel,
  fisher = fisher_global_model.sel,
  wolf = wolf_global_model.sel,
  lynx = lynx_global_model.sel,
  moose = moose_global_model.sel,
  fox = fox_global_model.sel,
  deer = deer_global_model.sel
) %>%
```

```
  # use purrr to combine data and extract species names to use as a column
  map_dfr(~.x %>%

            mutate(species = deparse(substitute(.x))),
          .id = "species")
```

And save this for use later

```
write_csv(global_model.sel_data,
          'data/processed/OSM_glm_global_model_sel_data.csv')
```

Analysis Prep

Summary

While the global models are interesting and yield relevant results there is an issue of addressing autocorrelation with such large models that we don't have the capacity to address at each spatial scale while still keeping the models the same so the only thing being compared is the buffer size.

It's also relevant to wonder if these results would be different depending on if we look at exclusively anthropogenic or landscape features in a model and by reducing the variables in each model we can address this issue of autocorrelation much easier. So the next analysis will repeat what we've done above for each species but instead of using one global model for everything we will divide the data into disturbance and landscape covariates and run two separate models per species and see if the results differ.

Data prep

First let's subset the data into the two new groups we will use for this analysis so we can check for auto correlations within the data sets

Anthropogenic

First let's generate the developed data set which includes only the covariates for the human factors indices

```
osm_anthro_df_2021_2022 <- osm_final_df_2021_2022 %>%

  # use purr to apply following data manipulation steps to all the buffer data frames
  purrr::map(

    ~.x %>%

      # de-select columns for landscape data (e.g. those that have the prefix 'lc_')
      select(!starts_with('lc_'))
  )
```

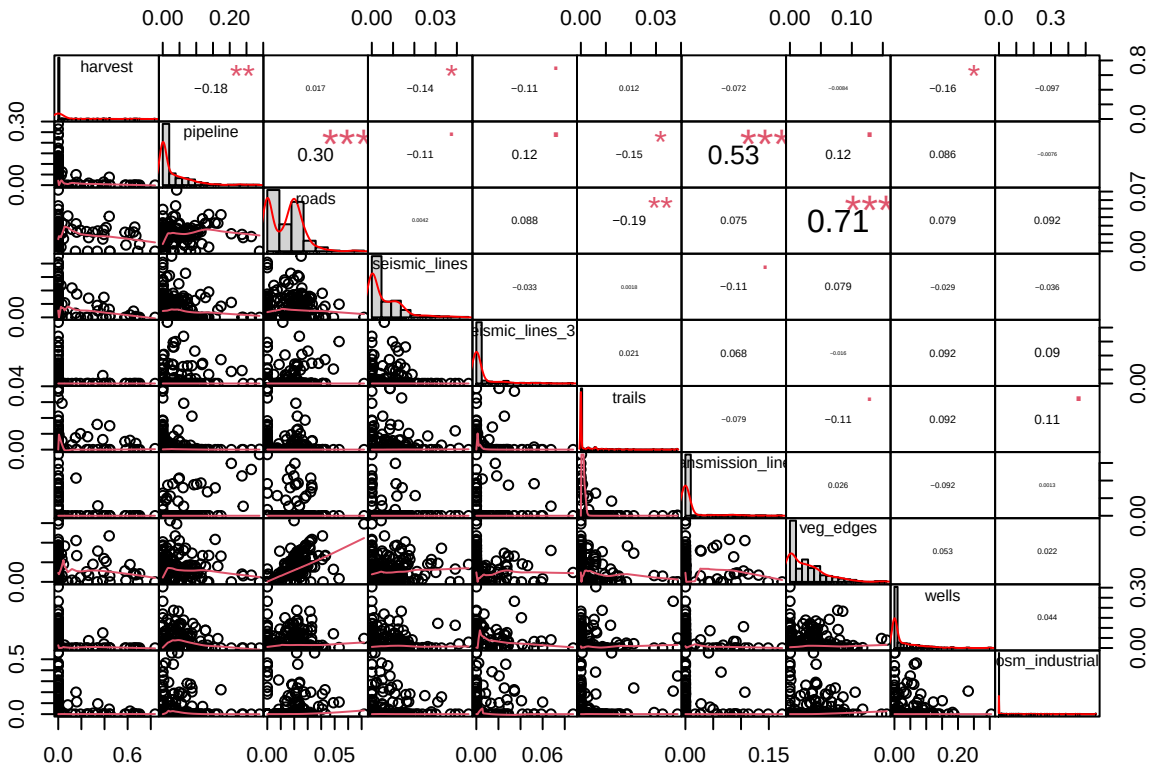
Now let's explore the correlations between variables at each buffer scale and see what features we may need to remove or group together (if appropriate)

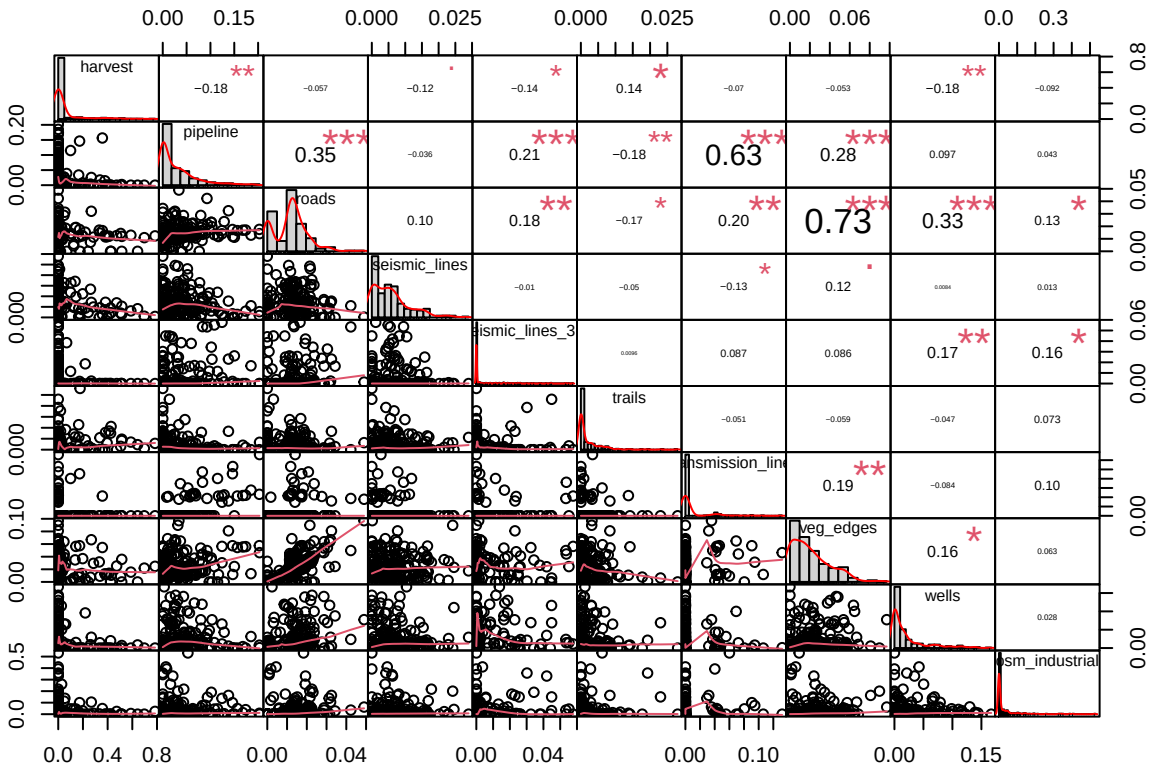
Correlation plots

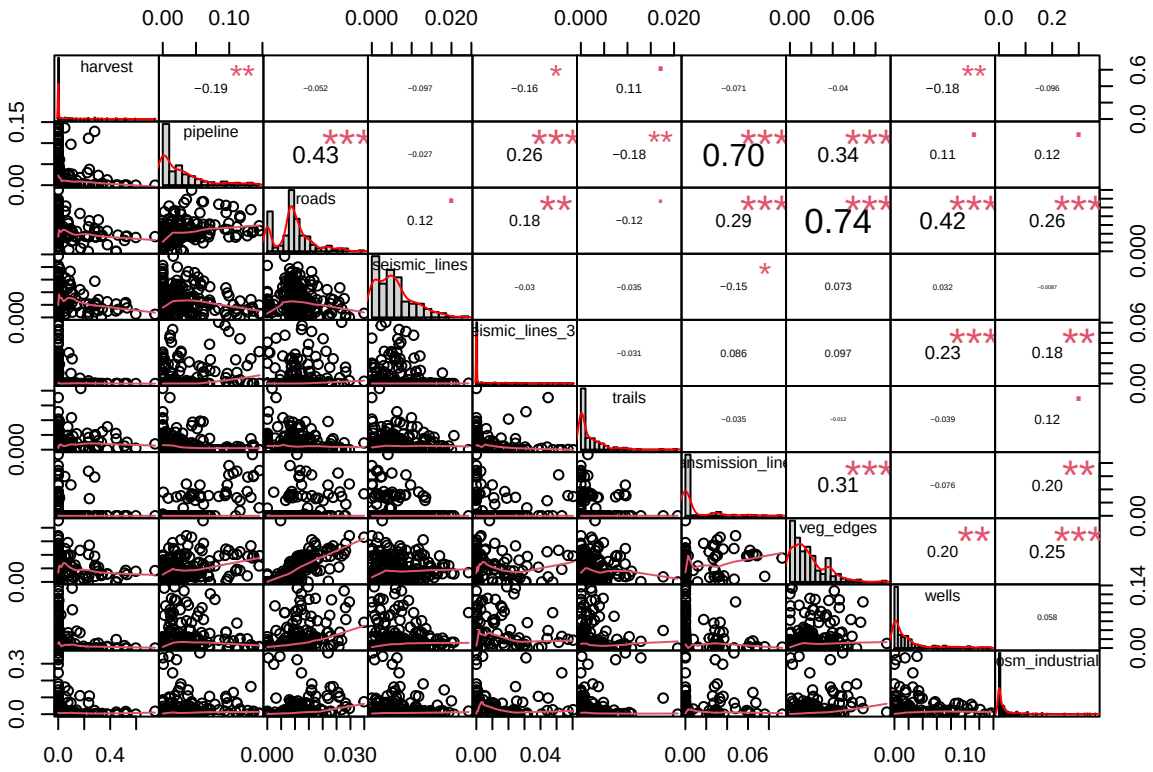
Now we need to make correlation plots for each buffer width to see what variables are correlated at a given spatial scale. We can use `purrr::map()` with the `chart.Correlation()` function from the *PerformanceAnalytics* package to make correlation plots with a specified method (e.g., pearson, spearman, etc.) That also show histograms and scatterplots of each variable.

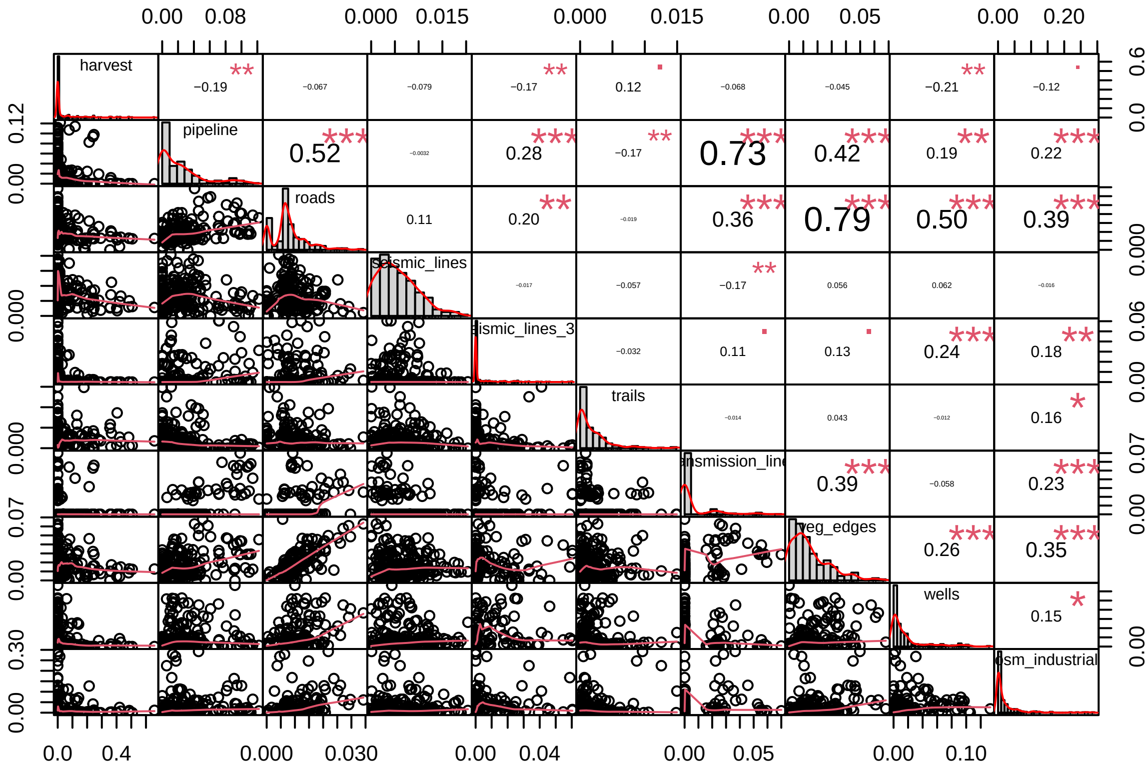
Let's do this for the anthropogenic data first

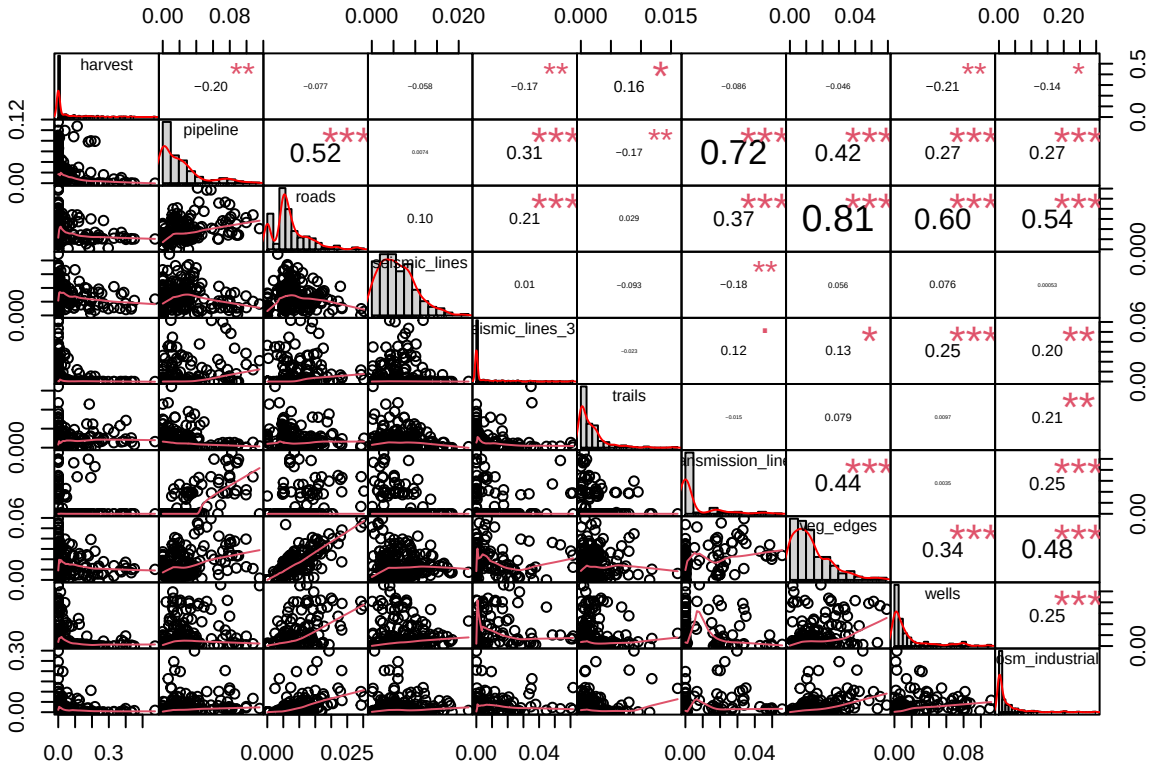
```
osm_anthro_df_2021_2022 %>%

  purrr::map(
    ~.x %>%

      # select only columns with covaraites not other info
      select(harvest:osm_industrial) %>%

      # use chart.correlation
      chart.Correlation(.,
                        histogram = TRUE,
                        method = "pearson")
  )
```

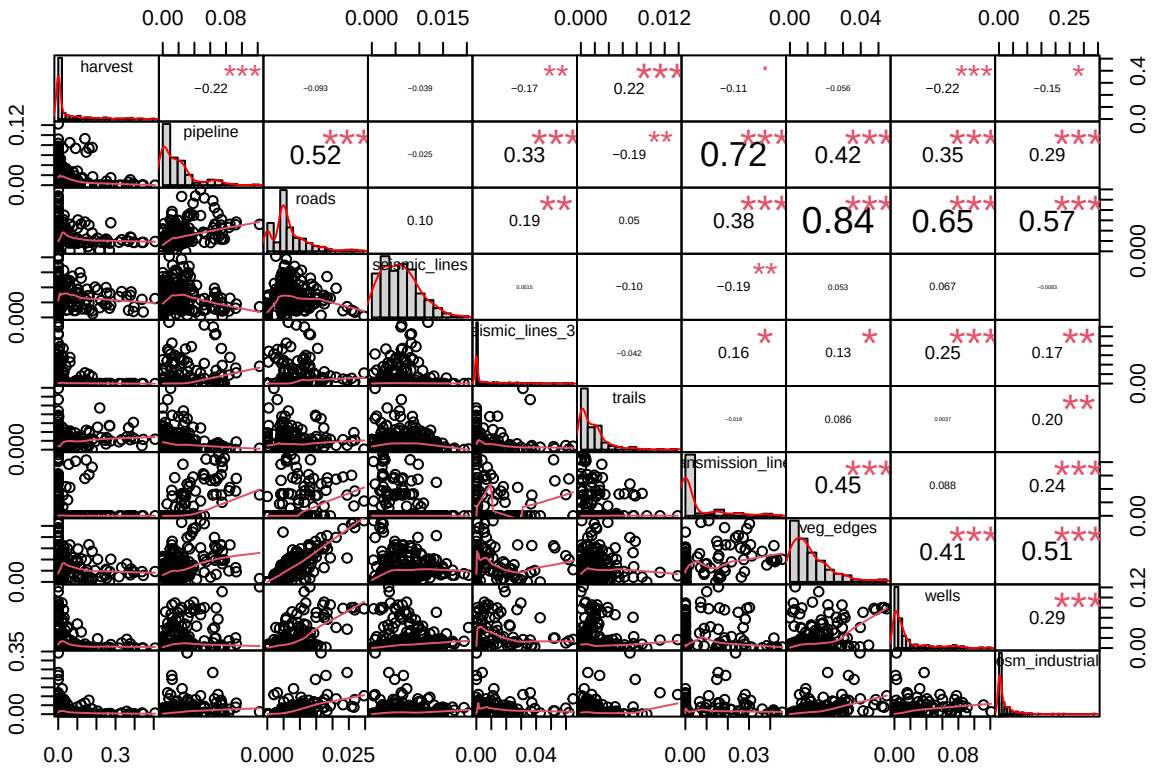


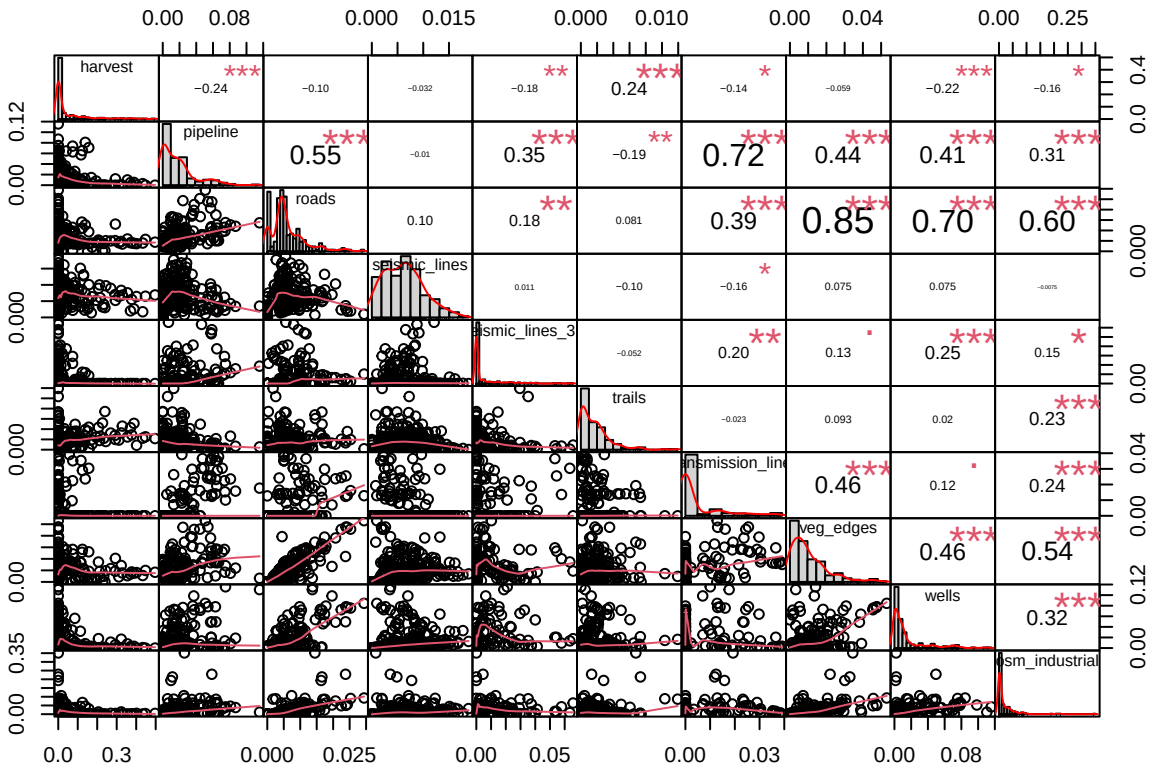

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
0.35
0.21
0.63
0.28
0.73
0.33
0.17
0.16
0.19
0.20
0.18
0.14
0.12
0.13
0.10
0.087
0.086
0.097
0.043
0.063
0.073
0.028
0.00
0.15
0.000
0.025
0.06
0.3
0.8
0.4
0.04
0.5

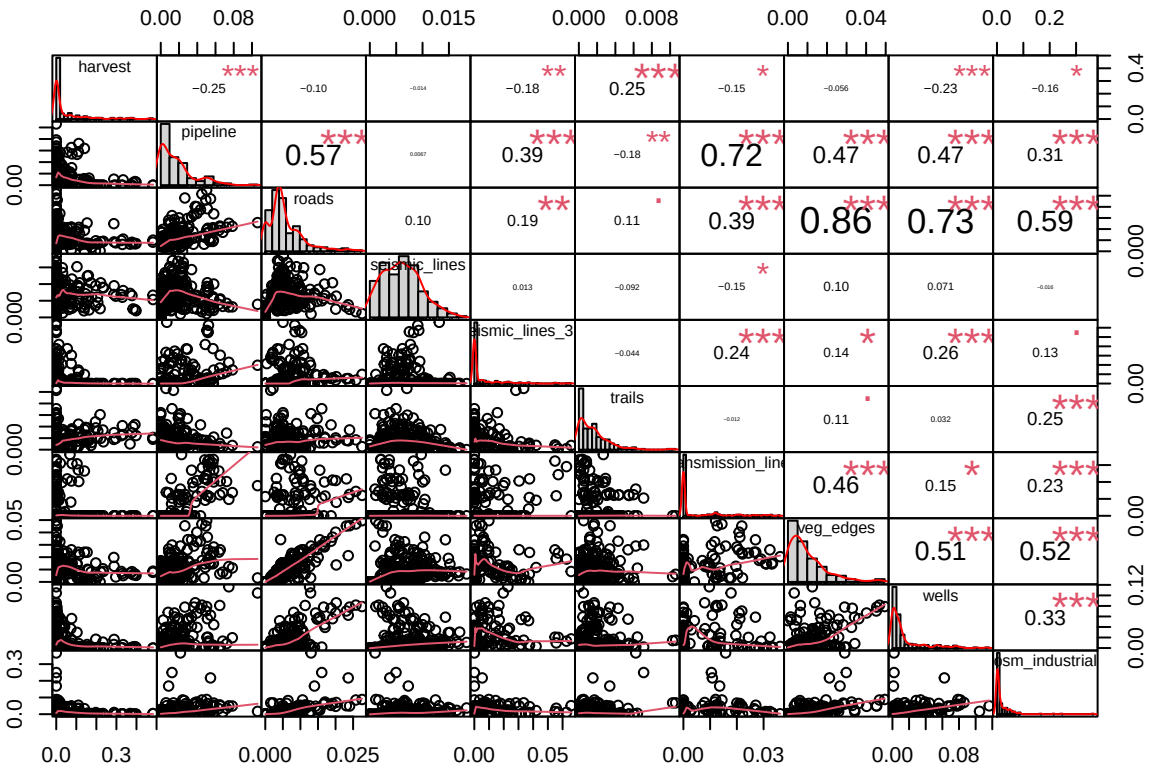

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
0.43
0.26
0.70
0.34
0.74
0.42
0.23
0.31
0.20
0.25

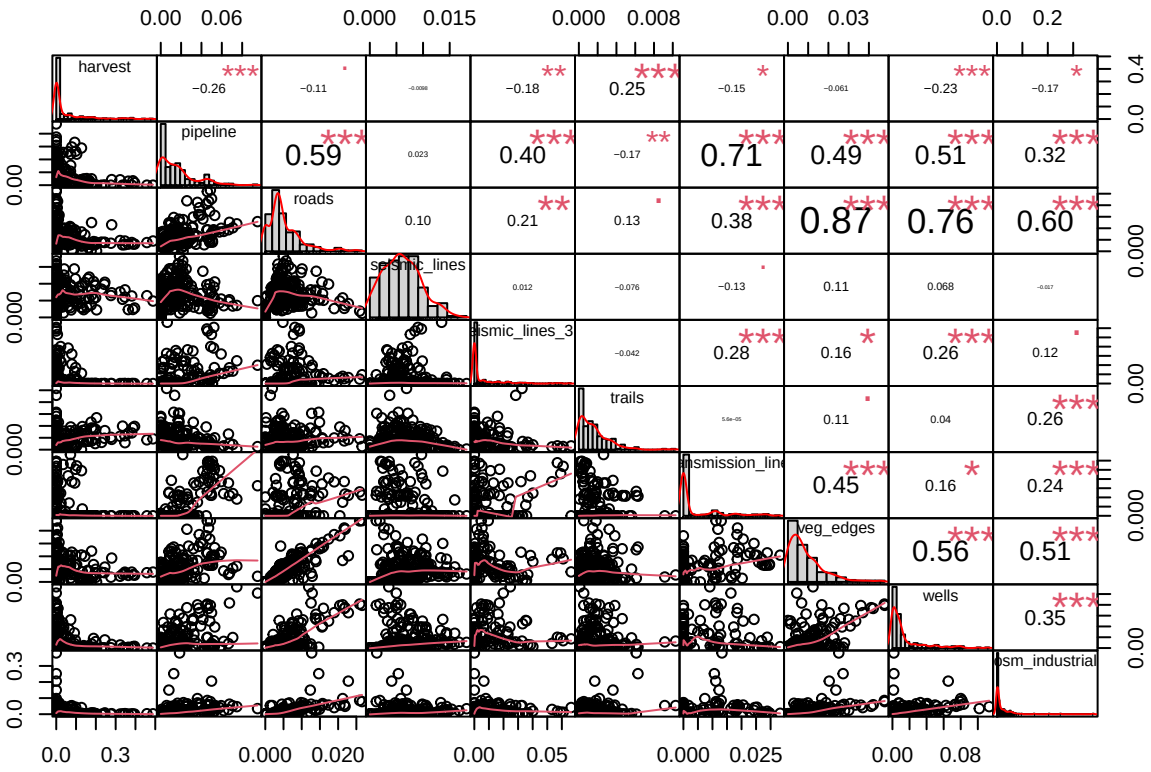

harvest
pipeline
roads
seismic_lines
seismic_lines_3
trails
transmission_line
veg_edges
wells
osm_industrial

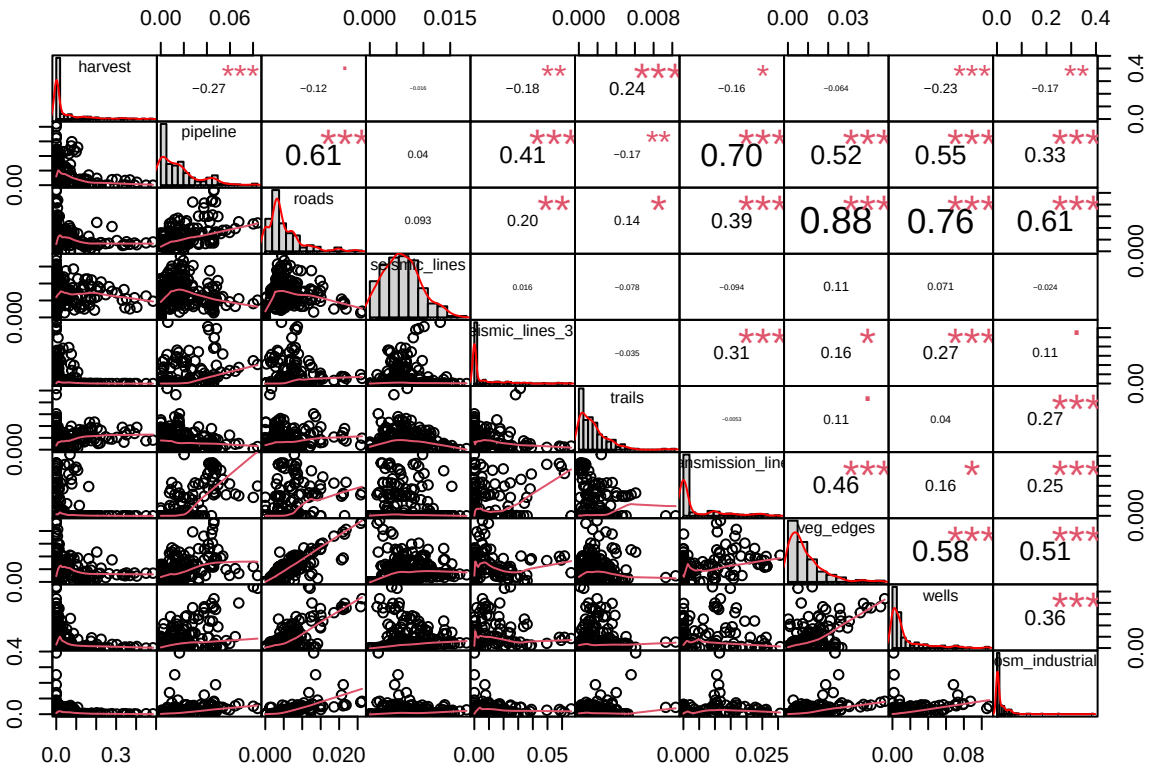

harvest
pipeline
roads
seismic_lines
trails
wells
0.52
0.72
0.81
0.60
0.54
0.44
0.48
0.25
0.27
0.31
0.21
0.10
0.01
0.12
0.13
0.20
0.34

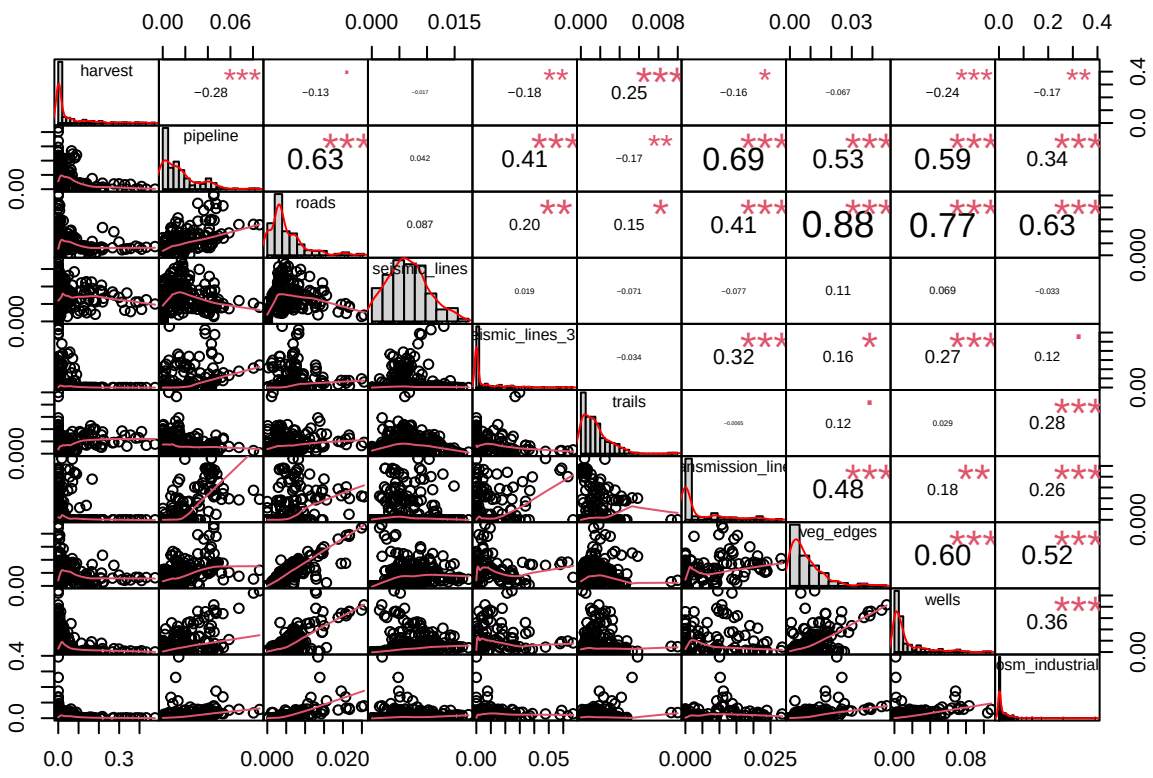

harvest
pipeline
roads
seismic_lines
ismic_lines_3
trails
nsmission_lin
veg_edges
wells
sm_industrial
0.00 0.08
0.000 0.015
0.000 0.012
0.00 0.04
0.00 0.25
0.0 0.3
0.000 0.025
0.00 0.04
0.00 0.03
0.00 0.08
0.00 0.12
0.000
0.00
0.35
0.00
0.0 0.4
0.000
0.00
0.00 0.12
−0.22
−0.093
−0.039
−0.17
0.22
−0.11
−0.056
−0.22
−0.15
0.52
−0.025
0.33
−0.19
0.72
0.42
0.35
0.29
0.10
0.19
0.05
0.38
0.84
0.65
0.57
0.0015
−0.10
−0.19
0.053
0.067
−0.0083
−0.042
0.16
0.13
0.25
0.17
−0.019
0.086
0.0027
0.20
0.45
0.088
0.24
0.41
0.51
0.29

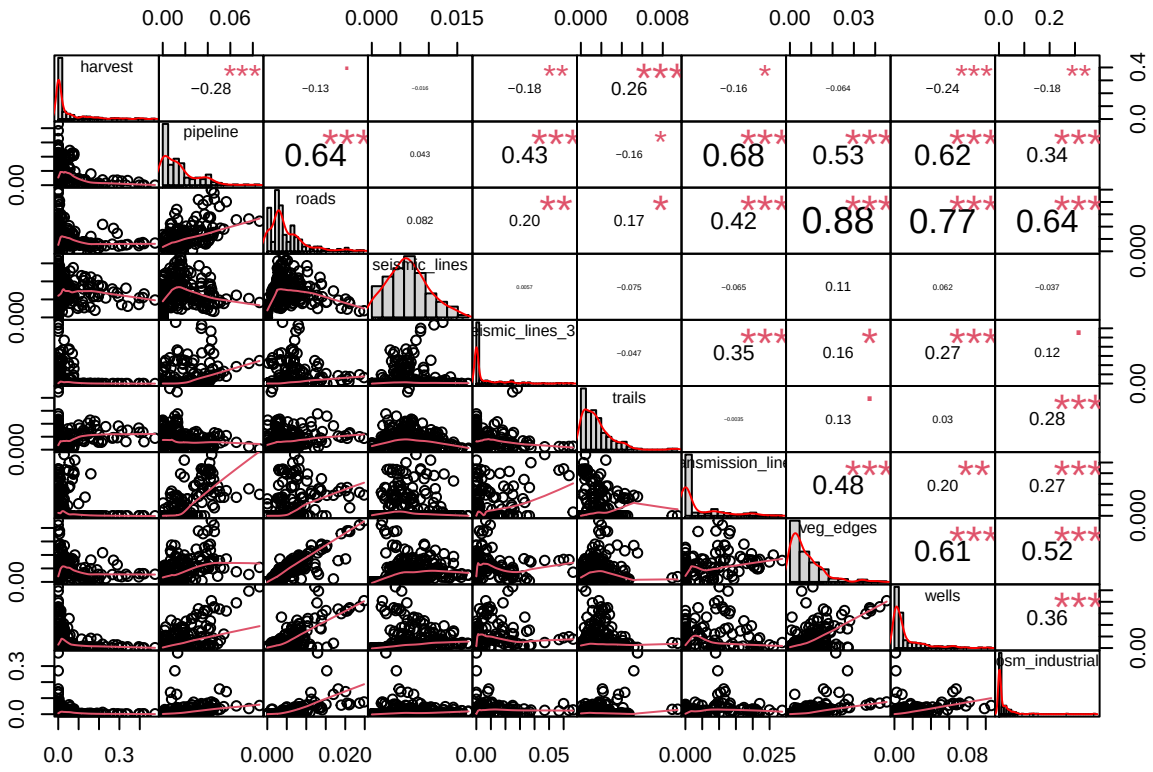

harvest
pipeline
roads
seismic_lines
seismic_lines_3
trails
transmission_line
veg_edges
wells
wsm_industrial
0.55
0.35
0.72
0.44
0.41
0.31
0.10
0.18
0.39
0.85
0.70
0.60
0.20
0.13
0.25
0.15
0.23
0.46
0.12
0.24
0.46
0.54
0.32

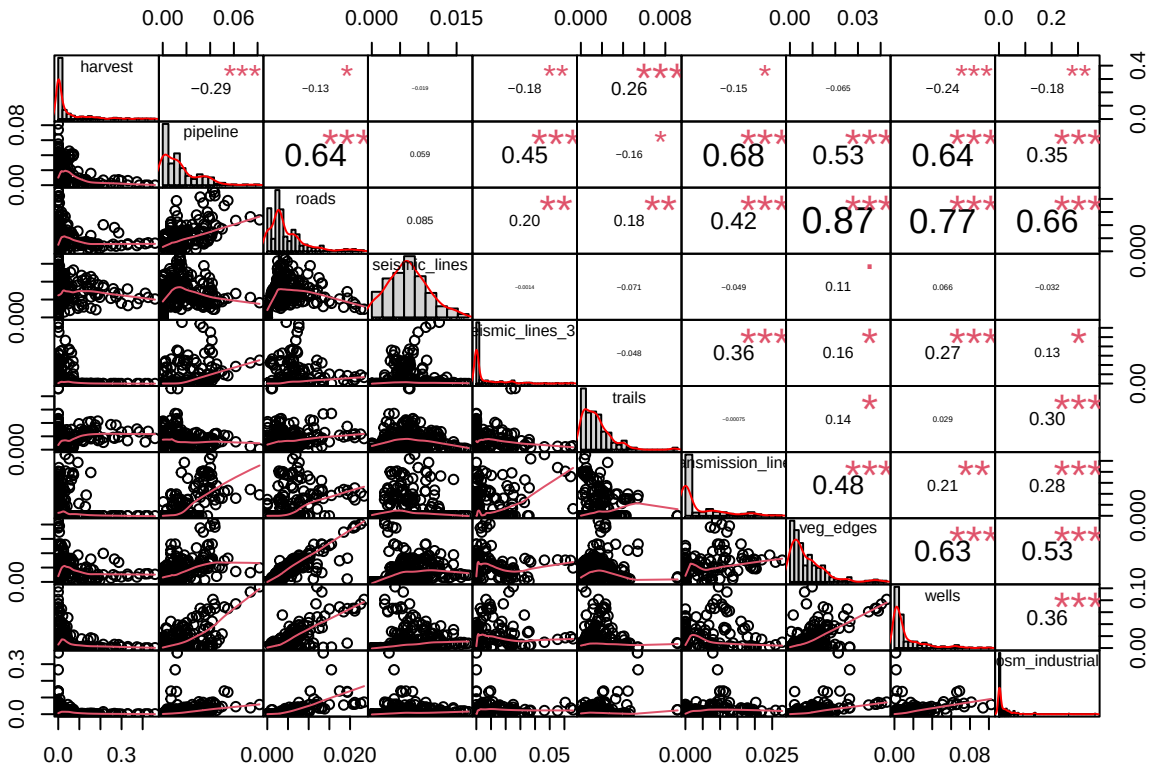

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
industrial
0.57
0.39
0.72
0.47
0.47
0.31
0.10
0.19
0.11
0.39
0.86
0.73
0.59
0.24
0.14
0.26
0.46
0.15
0.23
0.51
0.52
0.33

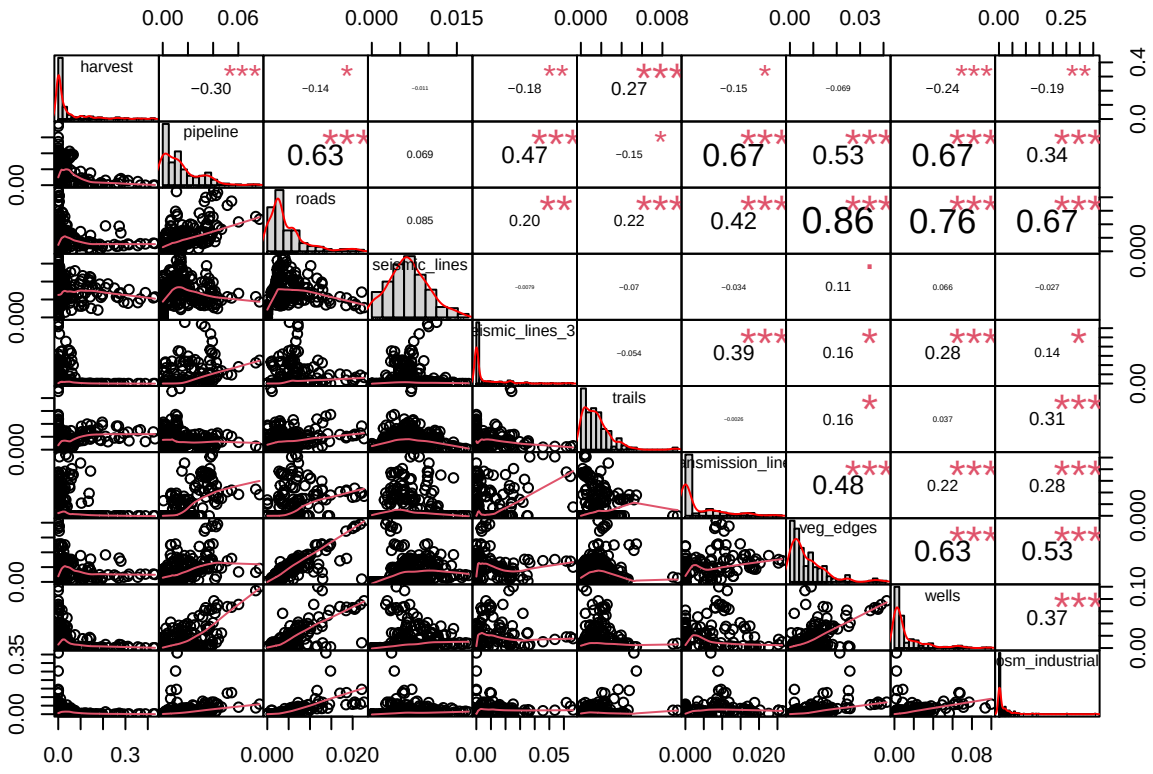

harvest
pipeline
roads
seismic_lines
seismic_lines_3
trails
transmission_line
veg_edges
wells
osm_industrial

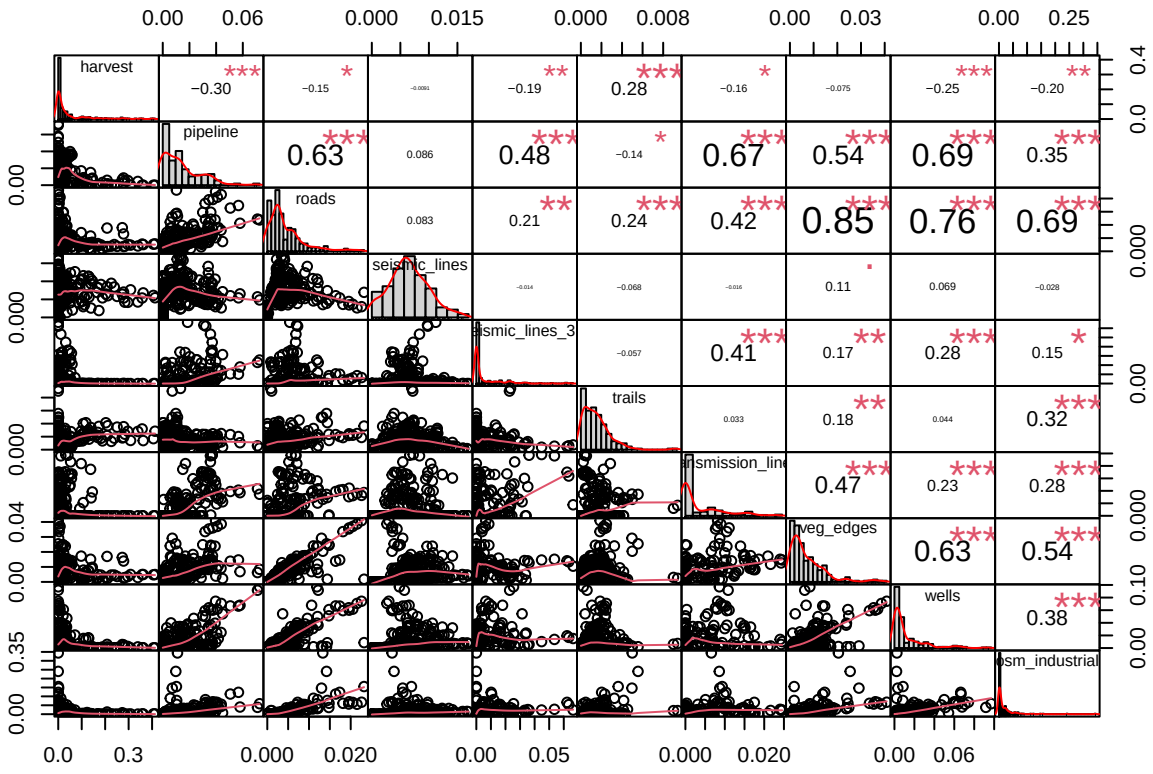

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
0.61
0.41
0.70
0.52
0.55
0.33
0.88
0.76
0.61
0.46
0.58
0.51
0.36

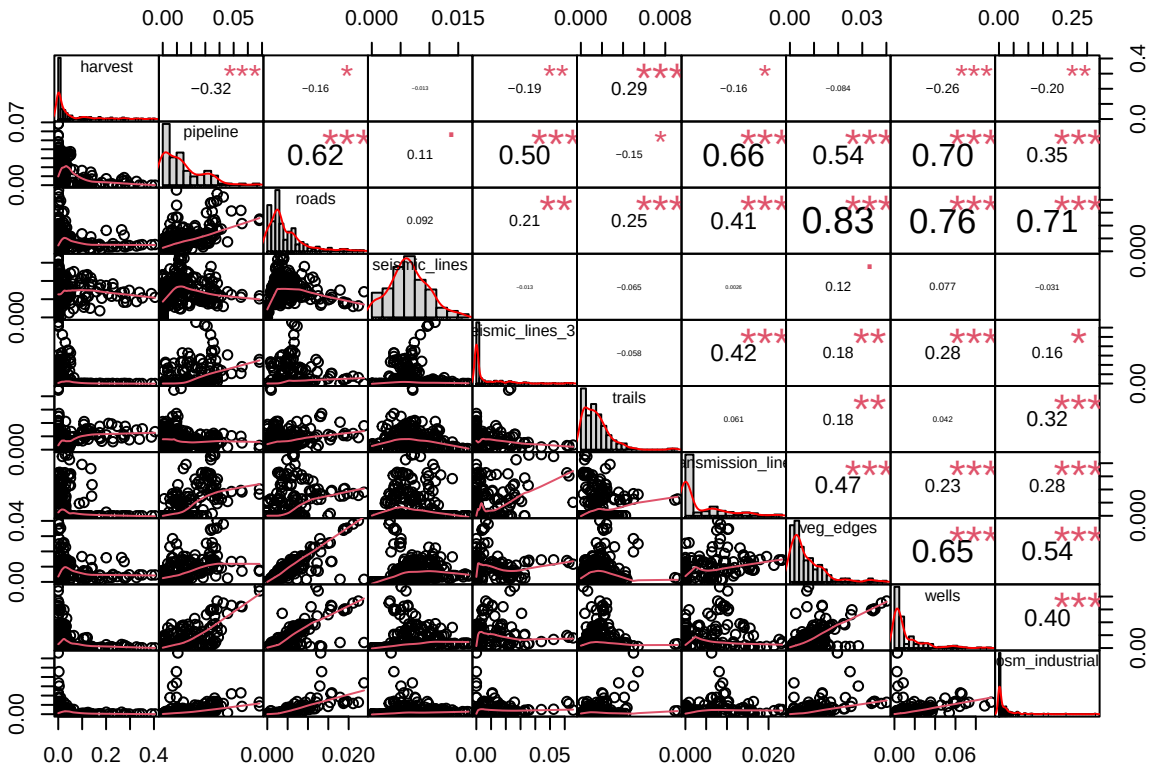

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial

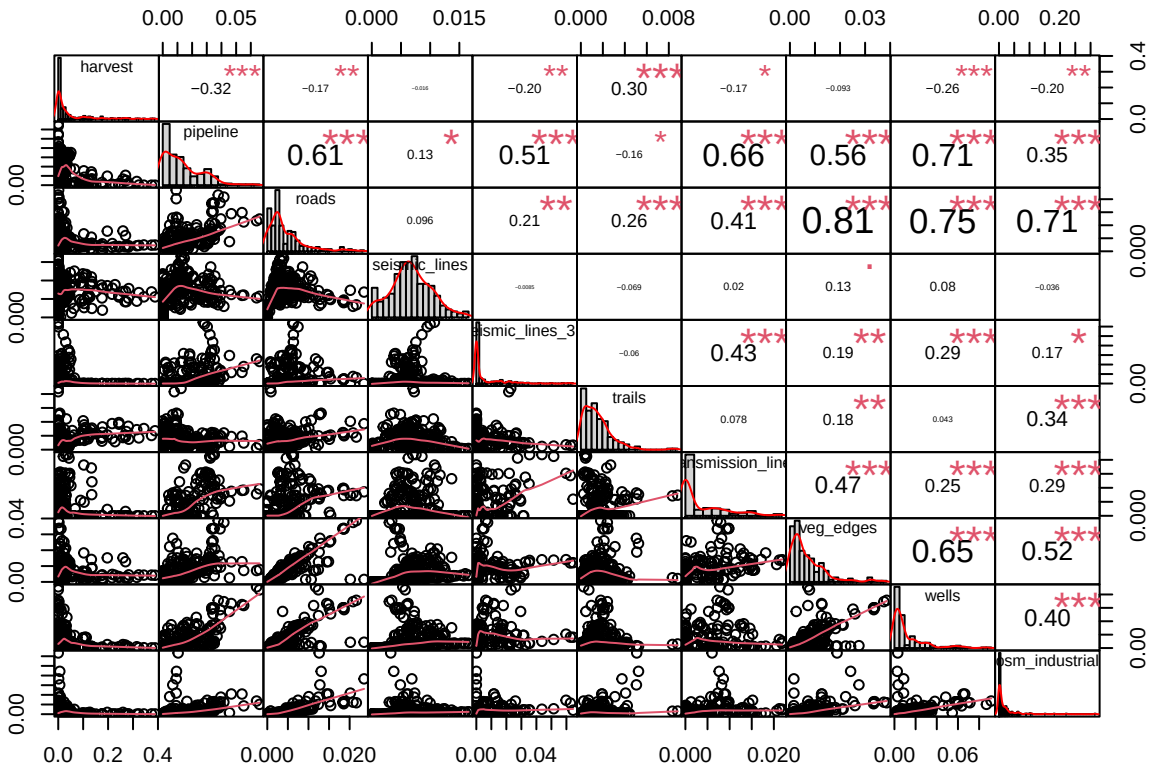

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
0.64
0.43
0.68
0.53
0.62
0.34
0.20
0.17
0.42
0.88
0.77
0.64
0.35
0.16
0.27
0.28
0.48
0.20
0.27
0.61
0.52
0.36

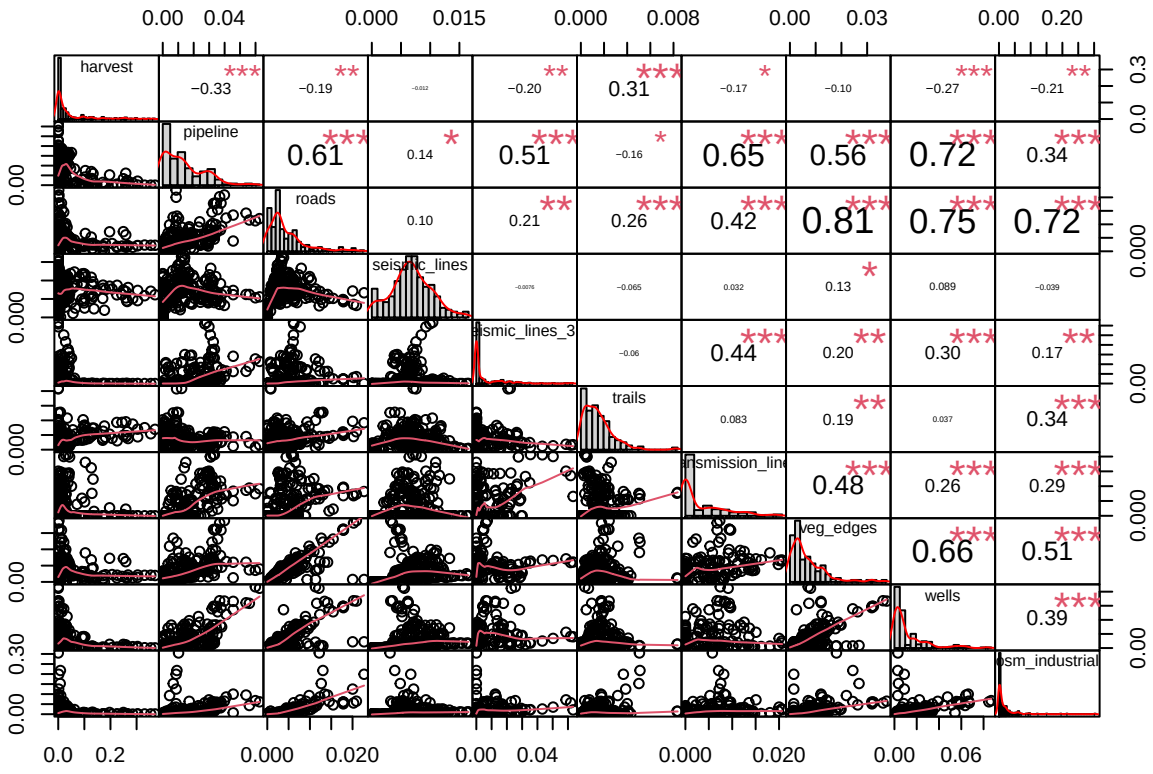

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial

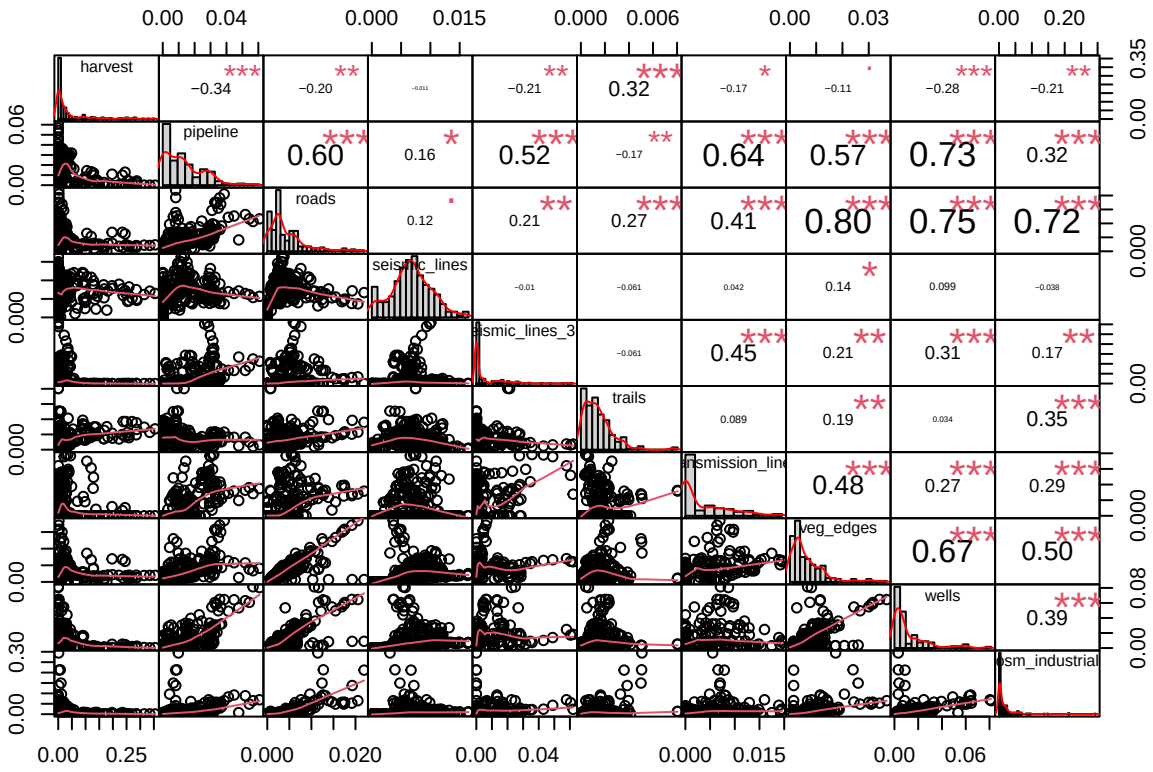

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial

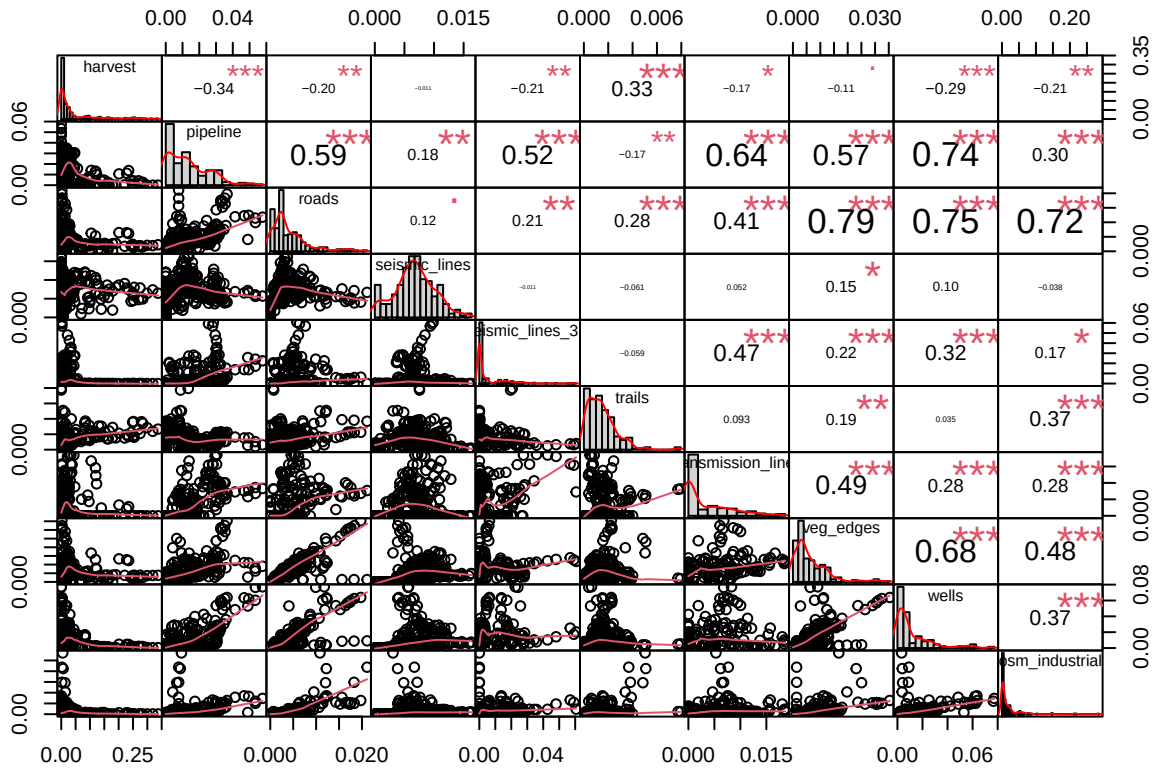

harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial


harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial
0.00 0.05
0.000 0.015
0.000 0.008
0.00 0.03
0.00 0.25
0.0 0.2 0.4
0.000 0.020
0.00 0.05
0.000 0.020
0.00 0.06
0.00 0.07
0.000 0.000
0.04 0.00
0.000 0.00
0.4 0.0
0.000 0.00
0.000 0.00


harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial


harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial


harvest
pipeline
roads
seismic_lines
trails
veg_edges
wells
osm_industrial
0.00 0.04
0.000 0.015
0.000 0.006
0.00 0.03
0.00 0.20
0.00 0.25
0.000 0.020
0.00 0.04
0.000 0.015
0.00 0.06




```
## $`250 meter buffer`
## NULL
##
## $`500 meter buffer`
## NULL
##
## $`750 meter buffer`
## NULL
##
## $`1000 meter buffer`
## NULL
##
## $`1250 meter buffer`
## NULL
##
## $`1500 meter buffer`
## NULL
##
## $`1750 meter buffer`
## NULL
##
## $`2000 meter buffer`
## NULL
##
## $`2250 meter buffer`
## NULL
```

```
##
## $`2500 meter buffer`
## NULL
##
## $`2750 meter buffer`
## NULL
##
## $`3000 meter buffer`
## NULL
##
## $`3250 meter buffer`
## NULL
##
## $`3500 meter buffer`
## NULL
##
## $`3750 meter buffer`
## NULL
##
## $`4000 meter buffer`
## NULL
##
## $`4250 meter buffer`
## NULL
##
## $`4500 meter buffer`
## NULL
##
## $`4750 meter buffer`
## NULL
##
## $`5000 meter buffer`
## NULL
```

This creates a lot of plots (1 for each buffer size) and I haven't found a way to label them by the buffer size yet unfortunately. But hopefully there will be trends among the buffer sizes to simplify the process of choosing covariates

> in Rstudio you can click on the white square icon in the upper right-hand corner of the figures to open a new window with the plots so you can see the values easier

In order of buffer size I'm going to summarize covariates that are correlated below with their respective r2 values. I'm only listing correlations above 0.6 for each buffer size

- 250m roads + veg edges 0.71 *this is to be expected as veg edges occur along the sides of roads (primarily) as well as other disturbance features. We will choose roads in the analysis over veg edges*
- 500m pipeline + transmission lines 0.63
  we may be able to combine these roads + veg edges 0.73
- 750m
  pipeline + transmission lines 0.70
  roads + veg edges 0.74

- 1000m
  pipeline + transmission lines 0.73
  roads + veg edges 0.79
- 1250m
  pipeline + transmission lines 0.72
  roads + veg edges 0.81
- 1500m
  pipeline + transmission lines 0.72
  roads + veg edges 0.84
  roads + wells 0.64
- 1750m pipeline + transmission lines 0.72
  roads + veg edges 0.85
  roads + wells 0.70
- 2000m
  pipeline + transmission lines 0.72
  roads + veg edges 0.86
  roads + wells 0.73
- 2250m
  pipeline + transmission lines 0.71
  roads + veg edges 0.87
  roads + wells 0.76
- 2500m
  pipleine + roads 0.61
  pipeline + transmission lines 0.70
  roads + veg edges 0.88
  roads + wells 0.76
  roads + osm_industrial 0.61
- 2750m
  pipeline + roads 0.61
  pipeline + transmission lines 0.70
  roads + veg edges 0.88
  roads + wells 0.76
  roads + osm_industrial 0.61
- 3000m
  pipeline + roads 0.63
  pipeline + transmission lines 0.69
  roads + veg edges 0.88
  roads + wells 0.77
  roads + osm_industrial 0.63

  at this point there are some obvious trends between roads and several features and pipelines and transmission lines. for simplicity sake I am only going to report correlations for variables not correlated with roads, as we will have to drop this covariate, and new pairs of highly correlated variables. I will also not specify the buffer size if it's found in more than one buffer

- 3250m +
  veg edges + wells 0.61
  pipeline + wells 0.67
  pipeline + wells 0.69

In summary we should merge pipelines and transmission lines, remove roads, and remove veg edges from the analysis to insure we don't majorly violate assumptions of independence for our models. We will re-assess correlations after making these changes.

List of final variables
I first pasted the full list here and then deleted or added combined variables as I went through each buffer's correlation plot

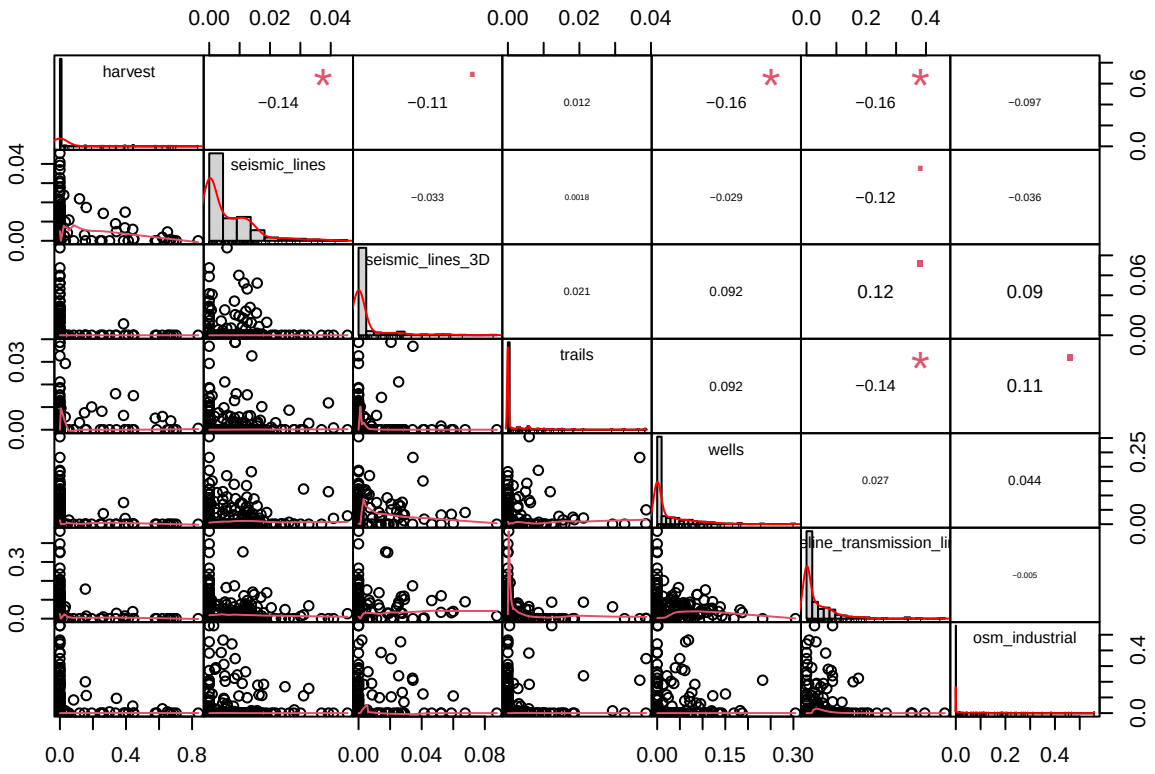
harvest
seismic lines
seismic lines 3D
trails
pipeline and transmission lines
wells
osm_industrial

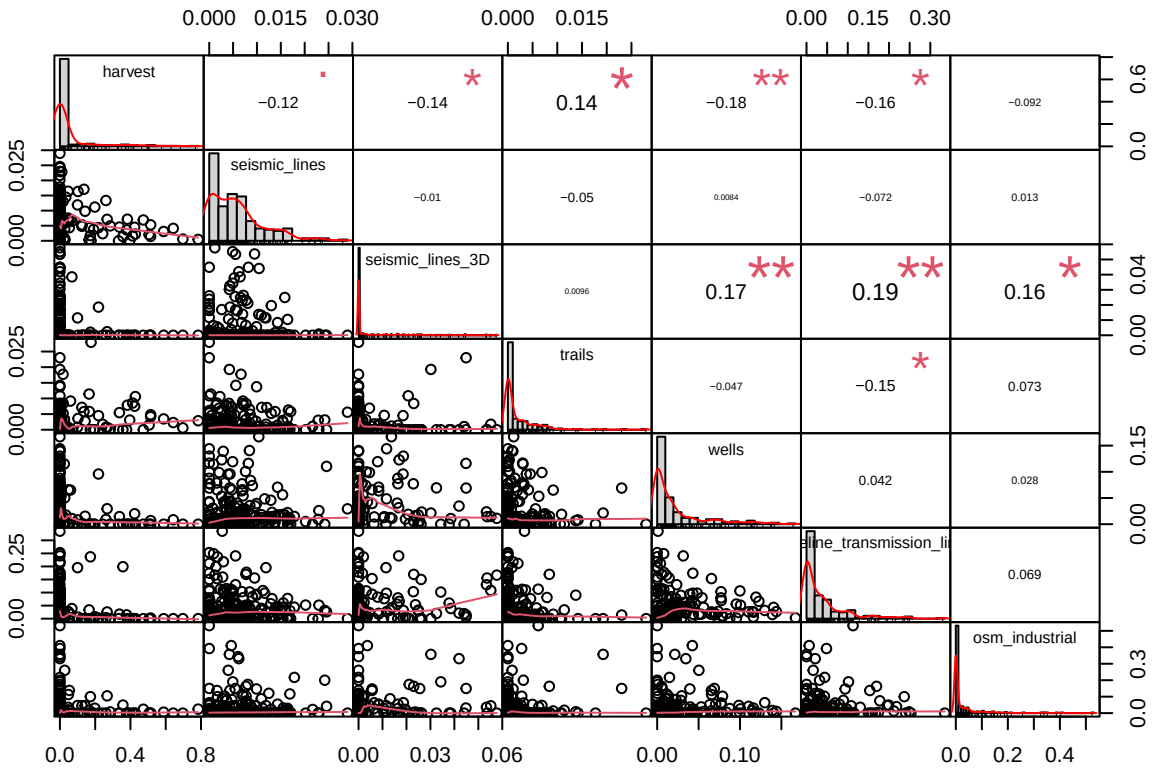
Reformat data We'll reformat the anthropogenicsubset data here and re-run the correlation plots

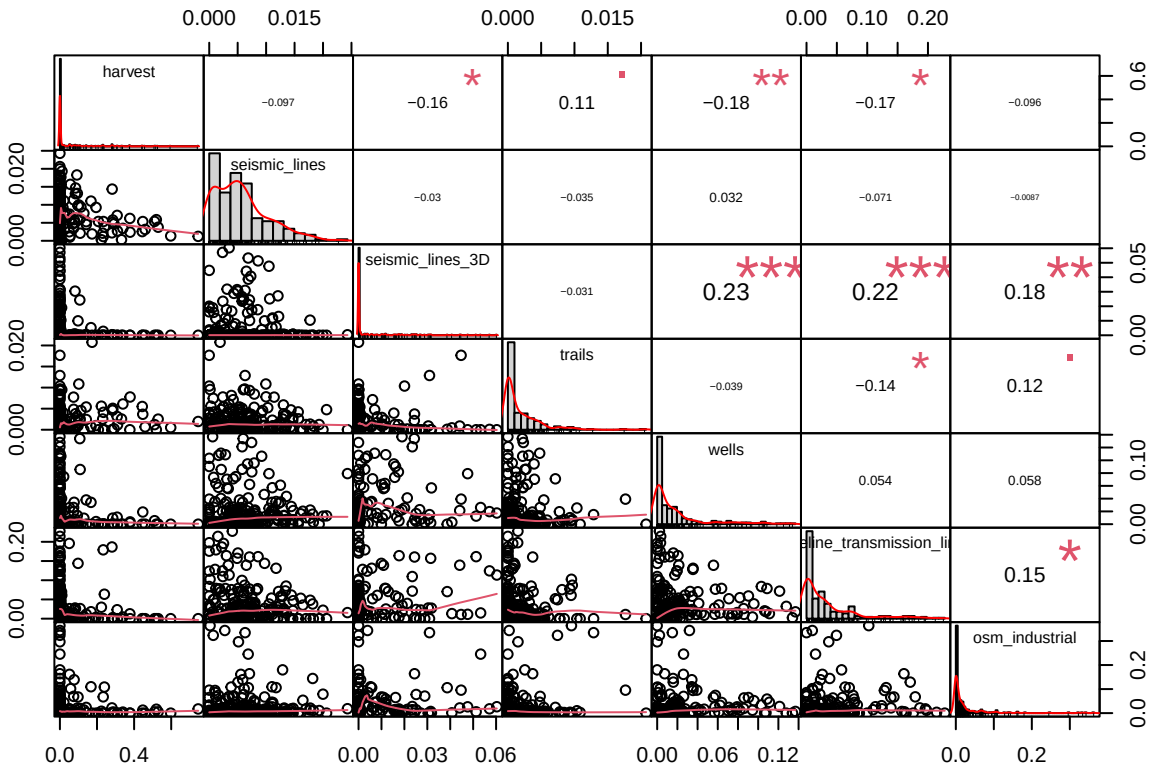
```
osm_anthro_df_2021_2022 <- osm_anthro_df_2021_2022 %>%

  # use purrr
  purrr::map(

    ~.x %>%

      # mutate data to combine pipeline and transmission line
      mutate(pipeline_transmission_lines = (pipeline + transmission_lines)) %>%

      # remove correlated variables we won't include in subset models
      select(! c(pipeline, roads, veg_edges, transmission_lines)) %>%

      # relocate column so new covariate is with others
      relocate(pipeline_transmission_lines,
               .after = wells)
  )
```

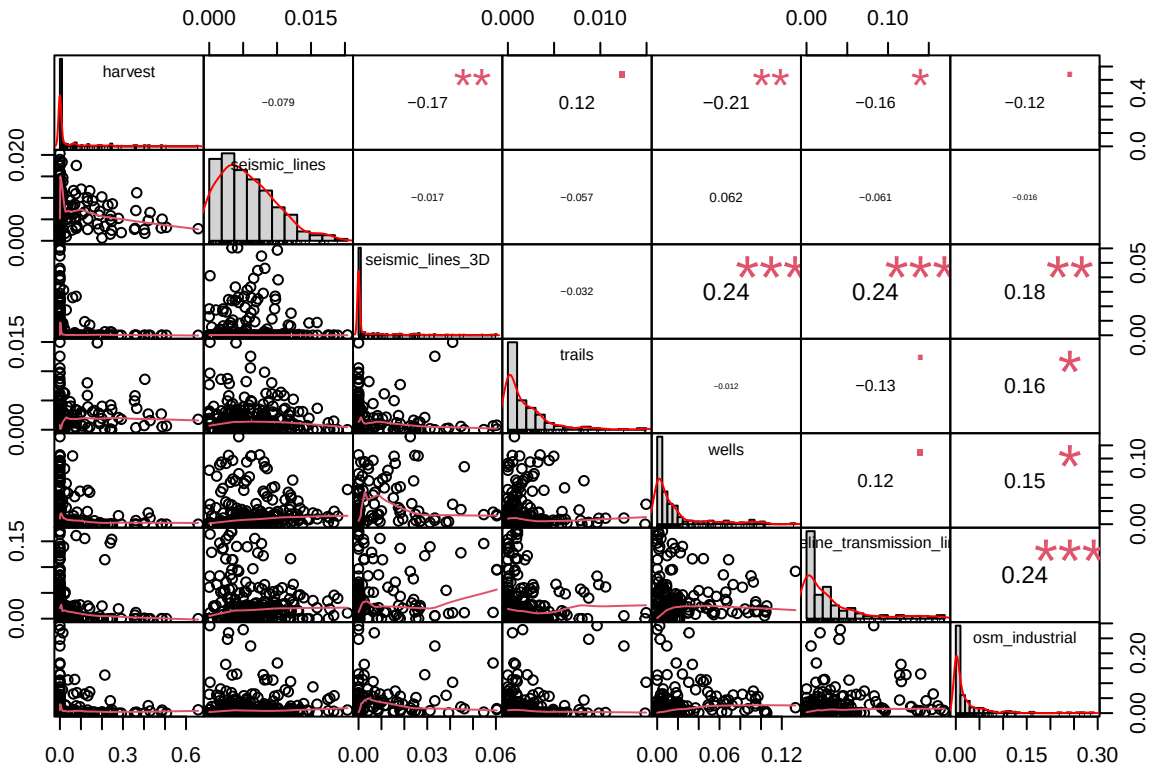
Re-run correlation plots Now we should double check that this solves any issues of highly correlated variables by re-running the plots from before

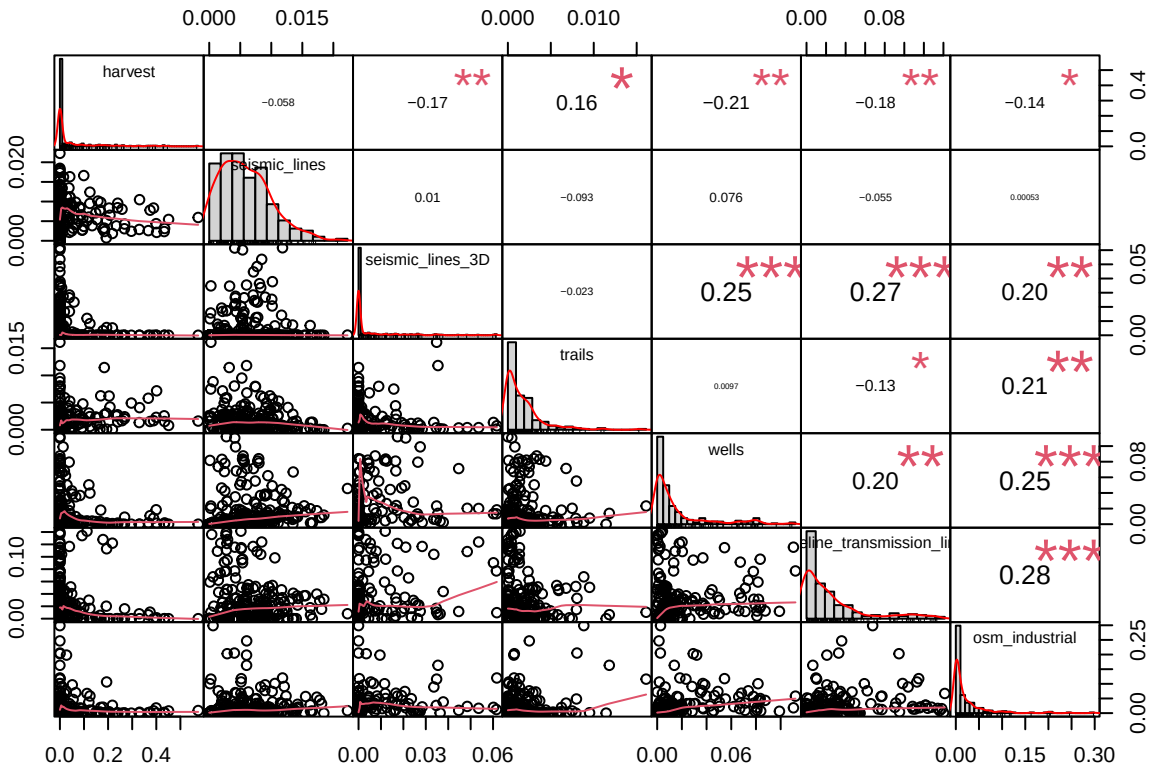
```
osm_anthro_df_2021_2022 %>%

  purrr::map(
    ~.x %>%

      # select only columns with covariates not other info
      select(harvest:osm_industrial) %>%

      # use chart.correlation
      chart.Correlation(.,
                        histogram = TRUE,
                        method = "pearson")
  )
```

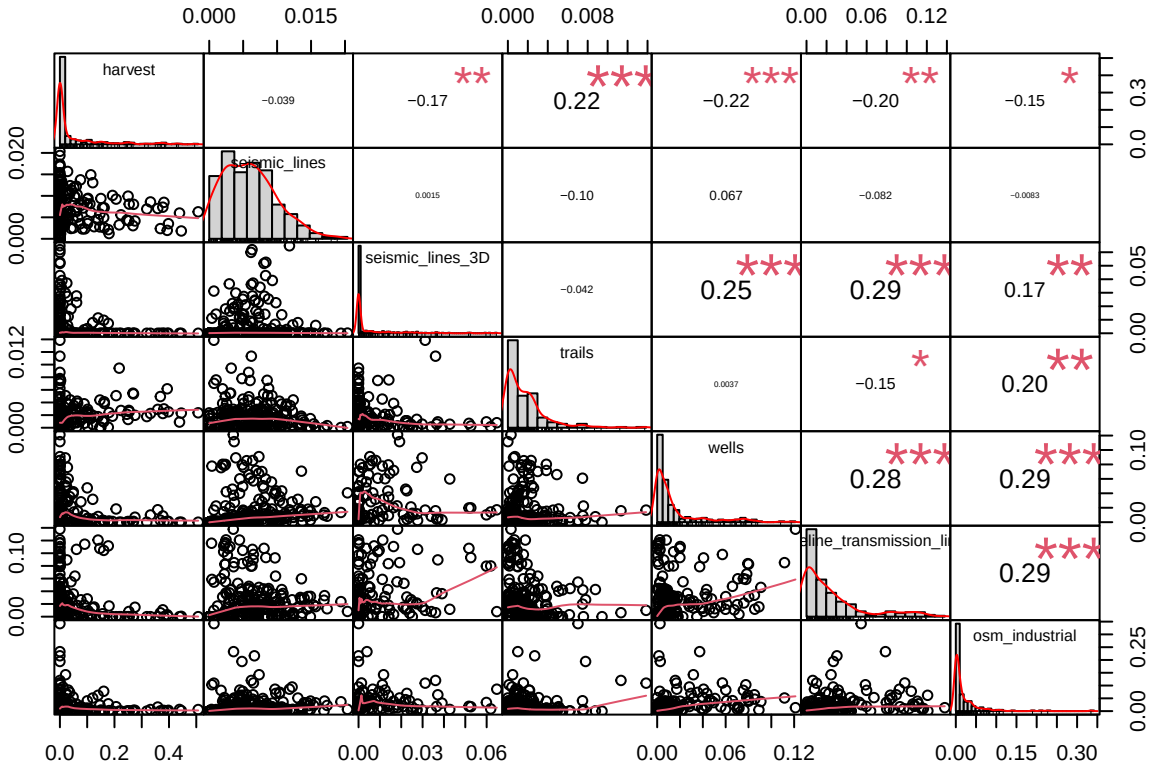

harvest
seismic_lines
seismic_lines_3D
trails
wells
osm_industrial
−0.14
−0.11
0.012
−0.16
−0.16
−0.097
−0.033
0.0018
−0.029
−0.12
−0.036
0.021
0.092
0.12
0.09
0.092
−0.14
0.11
0.027
0.044
−0.005

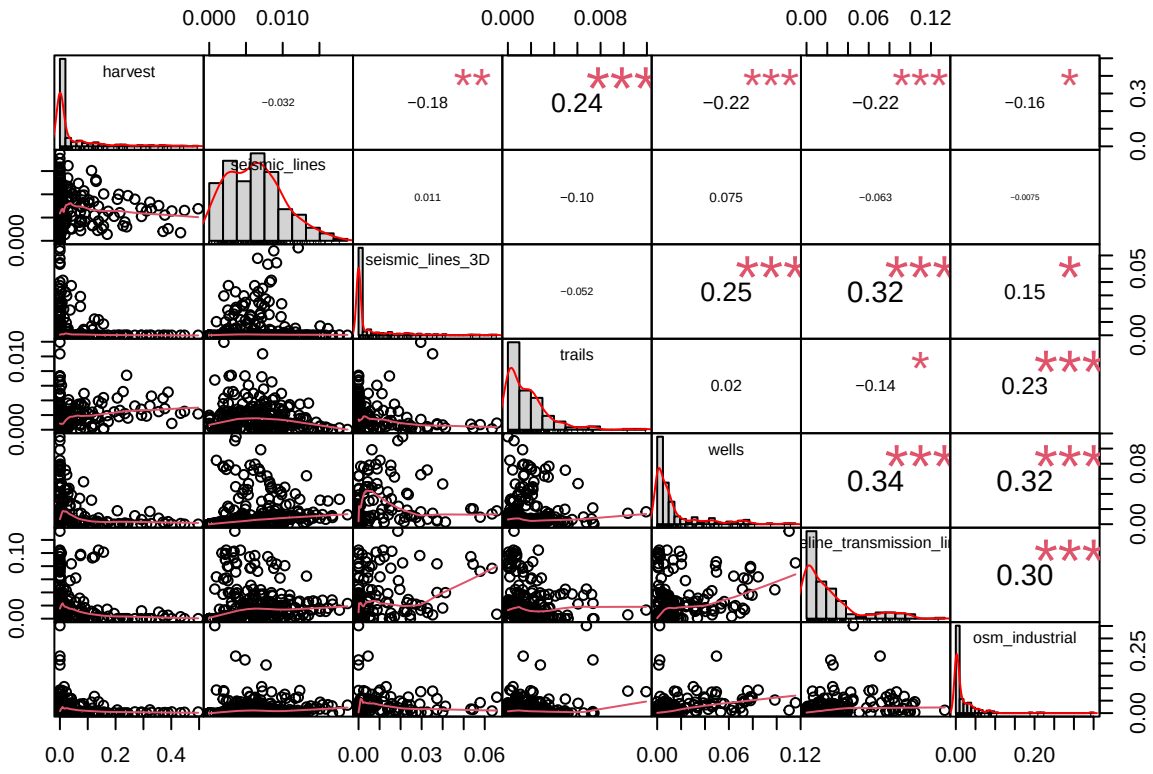

harvest
seismic_lines
seismic_lines_3D
trails
wells
osm_industrial
−0.12
−0.14
0.14
−0.18
−0.16
−0.092
−0.01
−0.05
0.0084
−0.072
0.013
0.0096
0.17
0.19
0.16
−0.047
−0.15
0.073
0.042
0.028
0.069
0.000 0.015 0.030
0.000 0.015
0.00 0.15 0.30
0.0 0.4 0.8
0.00 0.03 0.06
0.00 0.10
0.0 0.2 0.4

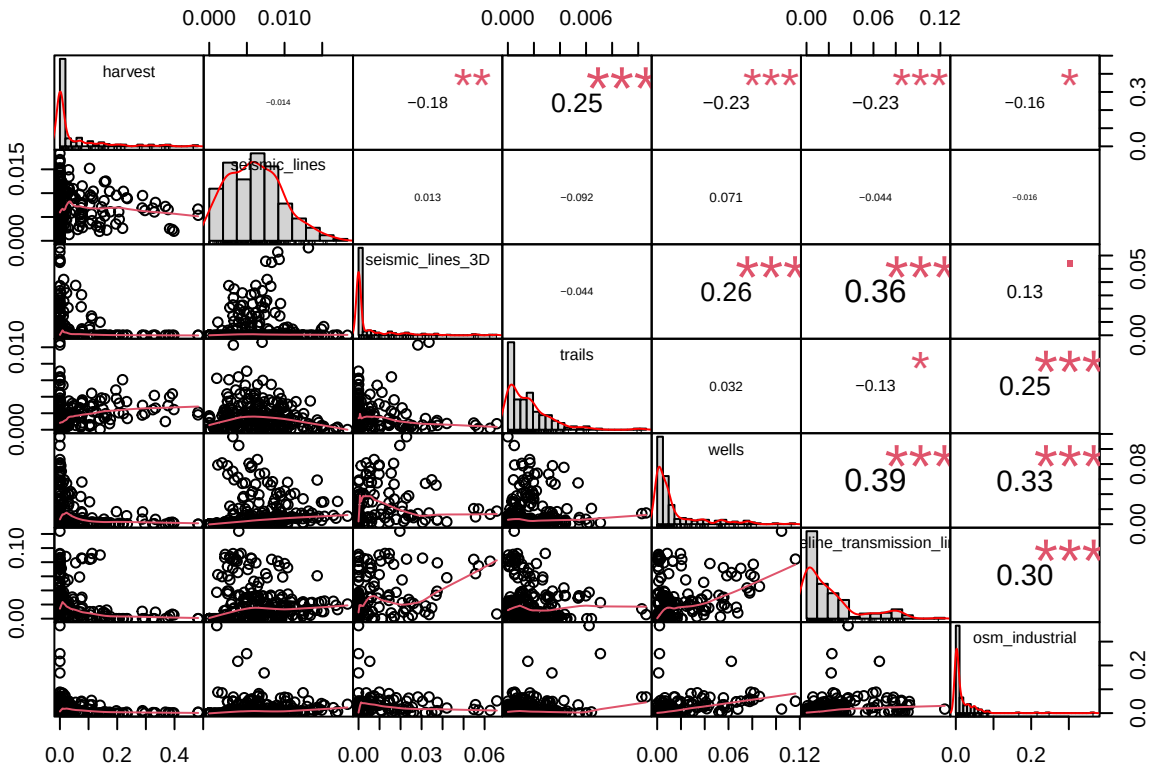

harvest
seismic_lines
seismic_lines_3D
trails
wells
osm_industrial
−0.097
−0.16 *
0.11 .
−0.18 **
−0.17 *
−0.096
−0.03
−0.035
0.032
−0.071
−0.0087
−0.031
0.23 ***
0.22 ***
0.18 **
−0.039
−0.14 *
0.12 .
0.054
0.058
0.15 *
0.000 0.015
0.000 0.015
0.00 0.10 0.20
0.020
0.000
0.020
0.000
0.20
0.00
0.6
0.0
0.05
0.00
0.10
0.00
0.2
0.0
0.0 0.4
0.00 0.03 0.06
0.00 0.06 0.12
0.0 0.2

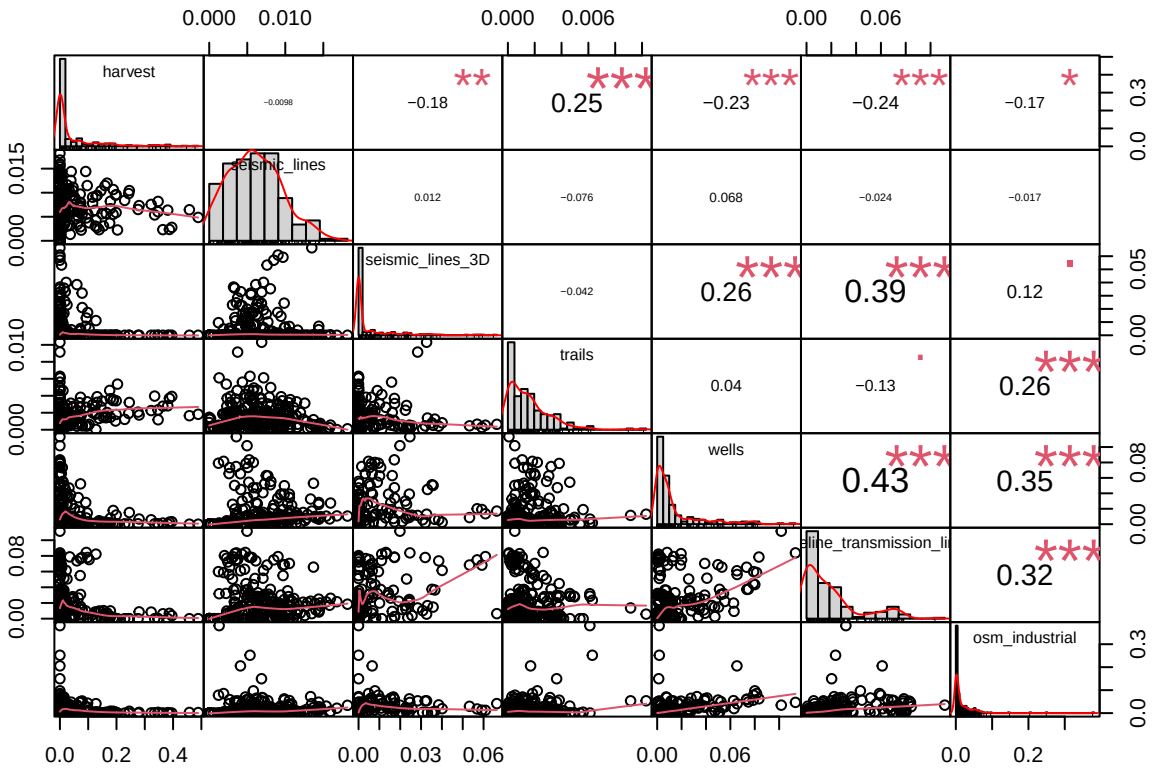

harvest
seismic_lines
seismic_lines_3D
trails
wells
osm_industrial
−0.079
−0.17
0.12
−0.21
−0.16
−0.12
−0.017
−0.057
0.062
−0.061
−0.016
−0.032
0.24
0.24
0.18
−0.012
−0.13
0.16
0.12
0.15
0.24
0.000
0.015
0.000
0.010
0.00
0.10
0.0
0.4
0.020
0.000
0.05
0.015
0.000
0.00
0.10
0.00
0.15
0.00
0.20
0.0
0.3
0.6
0.00
0.03
0.06
0.00
0.06
0.12
0.00
0.15
0.30

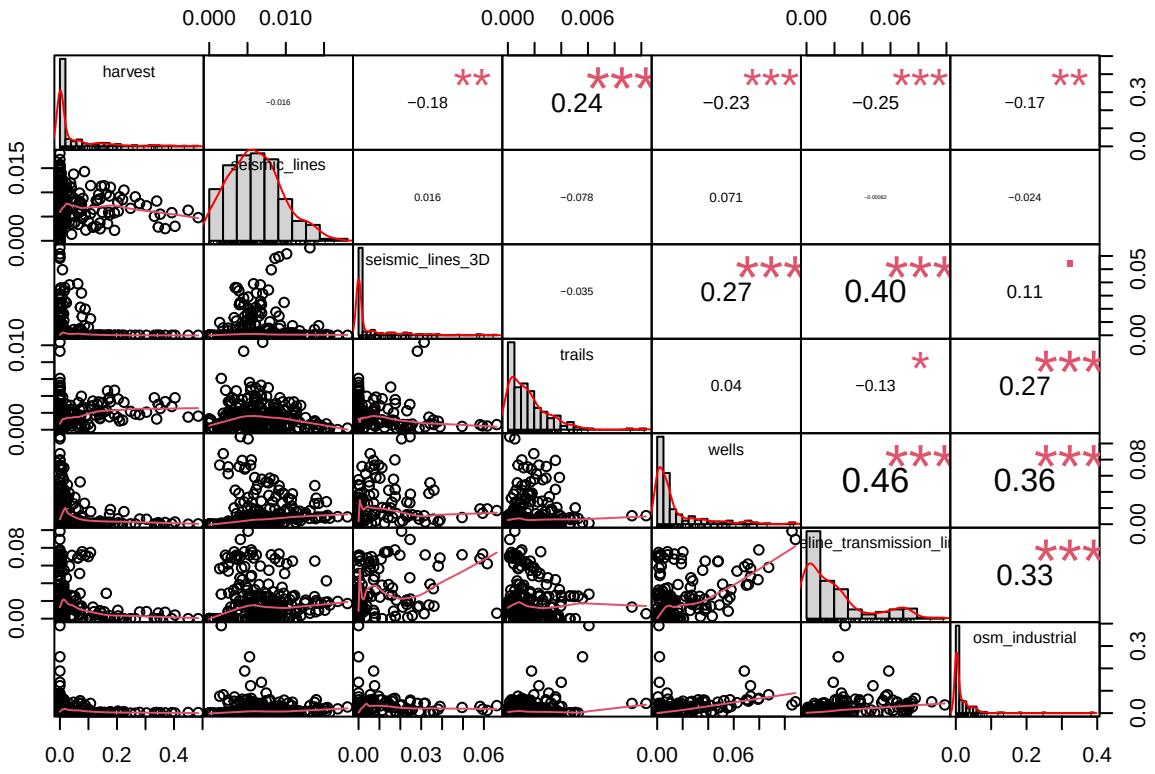

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.058
−0.17
0.16
−0.21
−0.18
−0.14
0.01
−0.093
0.076
−0.055
0.00053
−0.023
0.25
0.27
0.20
0.0097
−0.13
0.21
0.20
0.25
0.28
0.000
0.015
0.10
0.05
0.08
0.25
0.0
0.2
0.4
0.03
0.06
0.15
0.30

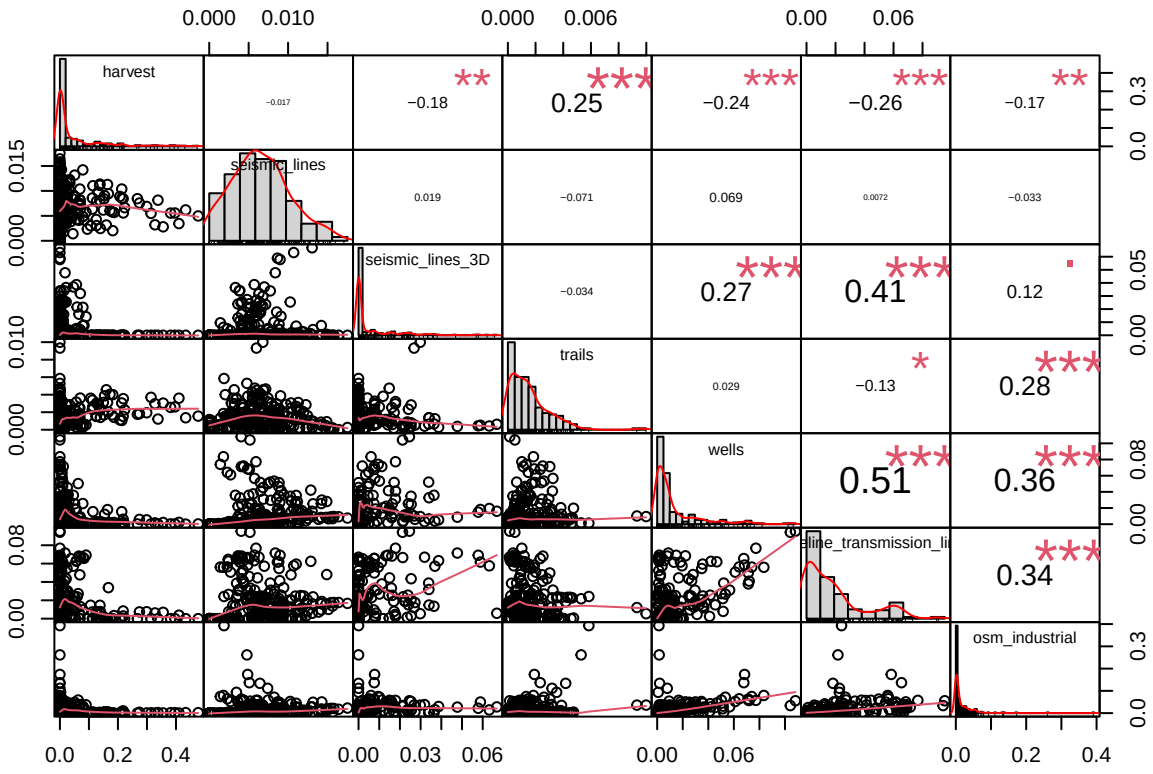

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.039
−0.17
0.22
−0.22
−0.20
−0.15
0.0015
−0.10
0.067
−0.082
−0.0083
−0.042
0.25
0.29
0.17
0.0037
−0.15
0.20
0.28
0.29
0.29
0.000
0.015
0.000
0.008
0.00
0.06
0.12
0.020
0.000
0.012
0.000
0.10
0.00
0.3
0.0
0.05
0.00
0.10
0.00
0.25
0.00
0.0
0.2
0.4
0.00
0.03
0.06
0.00
0.06
0.12
0.00
0.15
0.30

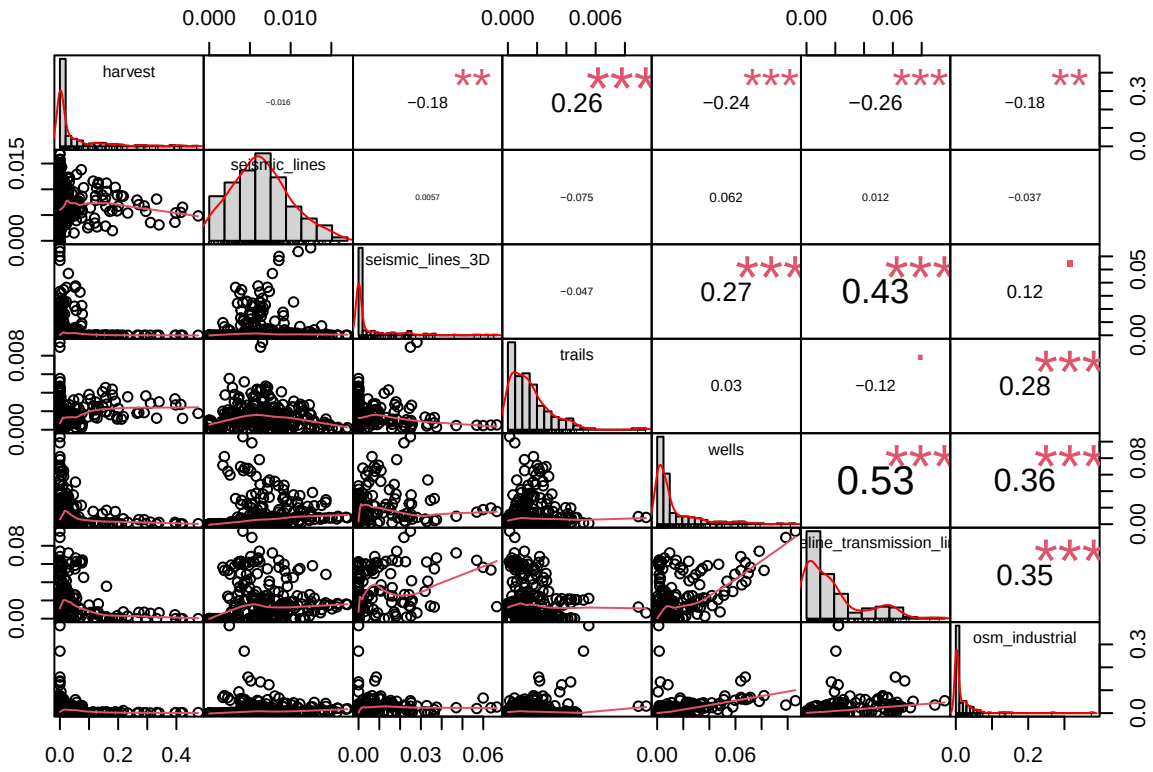

harvest
seismic_lines
seismic_lines_3D
trails
wells
osm_industrial
−0.032
−0.18
0.24
−0.22
−0.22
−0.16
0.011
−0.10
0.075
−0.063
−0.052
0.25
0.32
0.15
0.02
−0.14
0.23
0.34
0.32
0.30

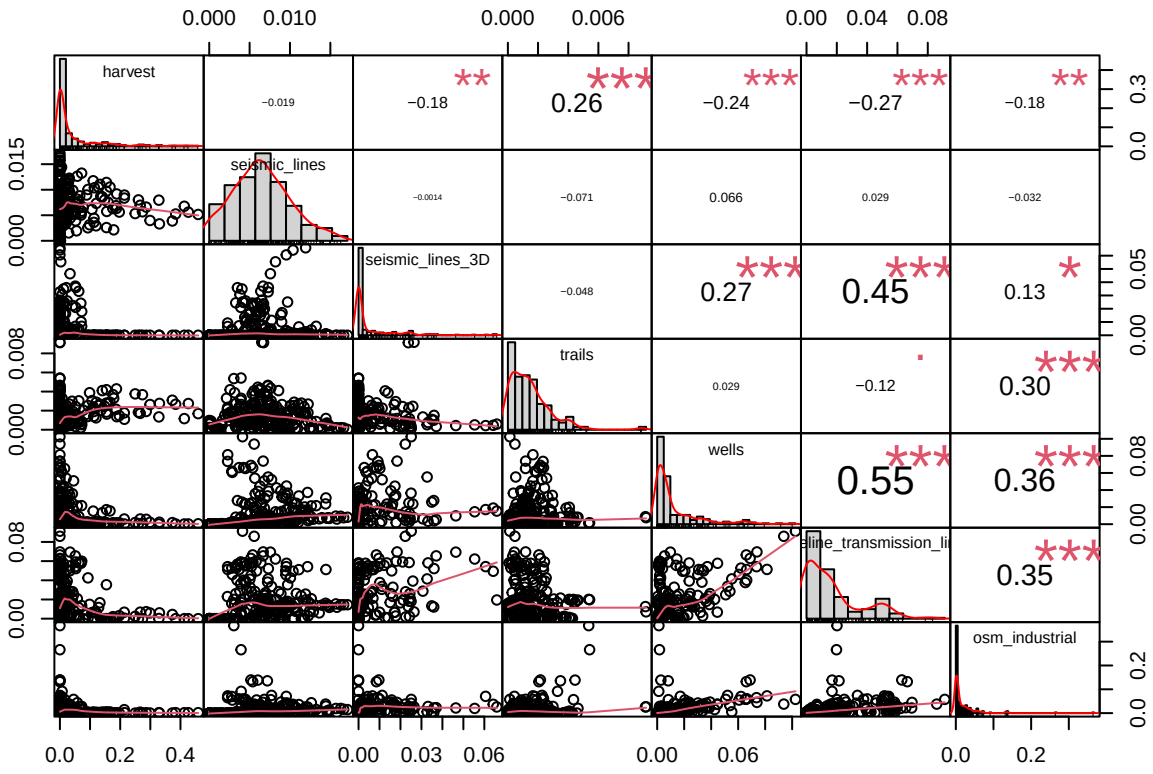

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.014
−0.18
0.25
−0.23
−0.23
−0.16
0.013
−0.092
0.071
−0.044
−0.016
−0.044
0.26
0.36
0.13
0.032
−0.13
0.25
0.39
0.33
0.30
0.000 0.010
0.000 0.006
0.00 0.06 0.12
0.0 0.2 0.4
0.00 0.03 0.06
0.00 0.06 0.12
0.0 0.2

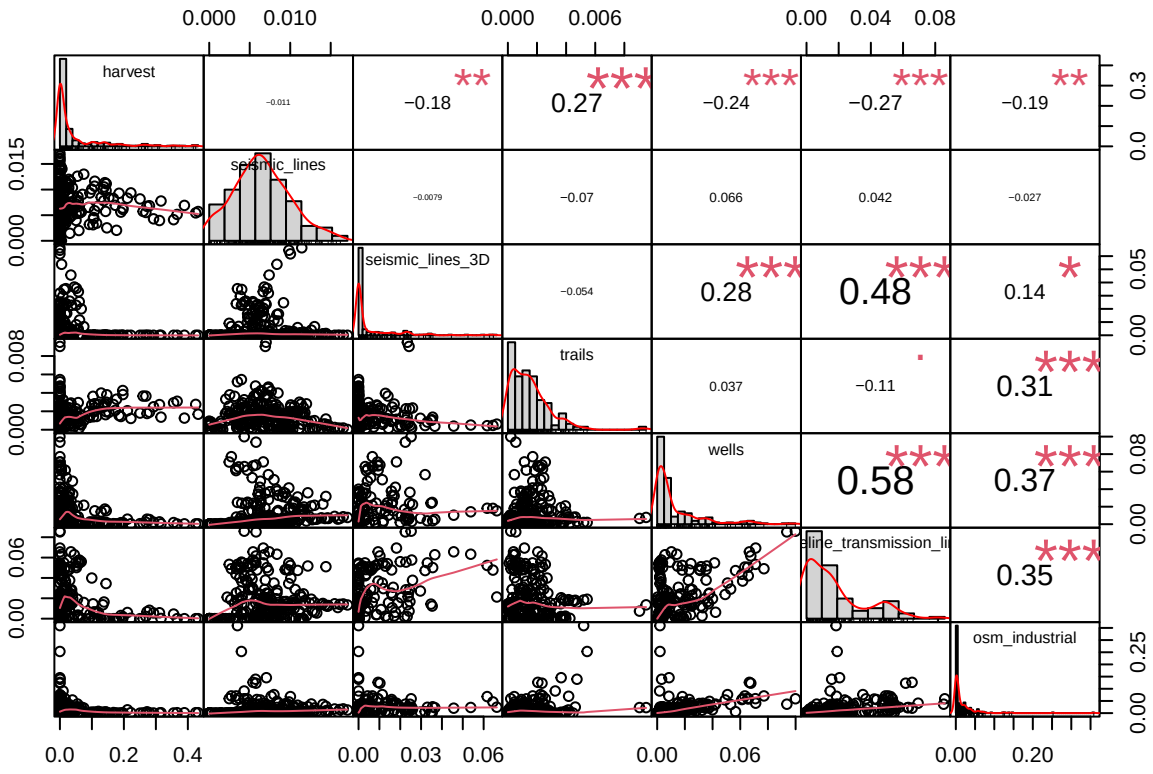

harvest
seismic_lines
seismic_lines_3D
trails
wells
osm_industrial
−0.18
0.25
−0.23
−0.24
−0.17
0.012
−0.076
0.068
−0.024
−0.017
−0.042
0.26
0.39
0.12
0.04
−0.13
0.26
0.43
0.35
0.32

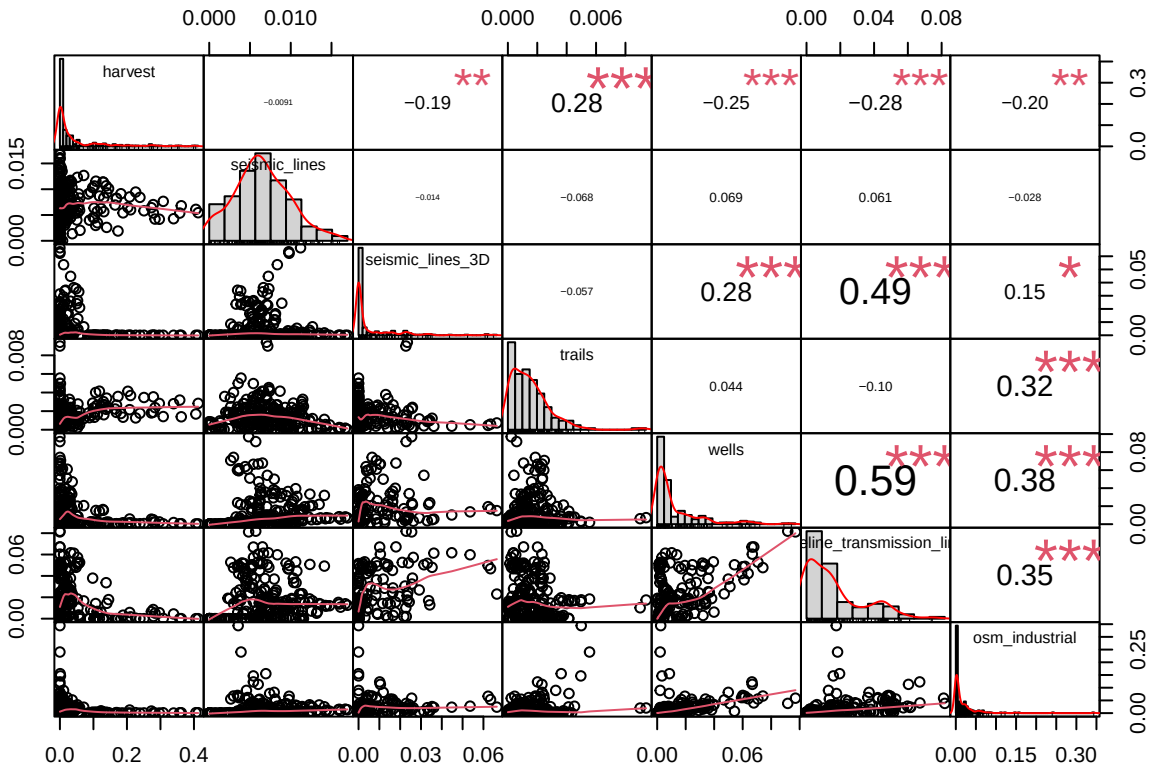

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.18
0.24
−0.23
−0.25
−0.17
0.016
−0.078
0.071
−0.024
0.27
0.40
0.11
0.04
−0.13
0.27
0.46
0.36
0.33

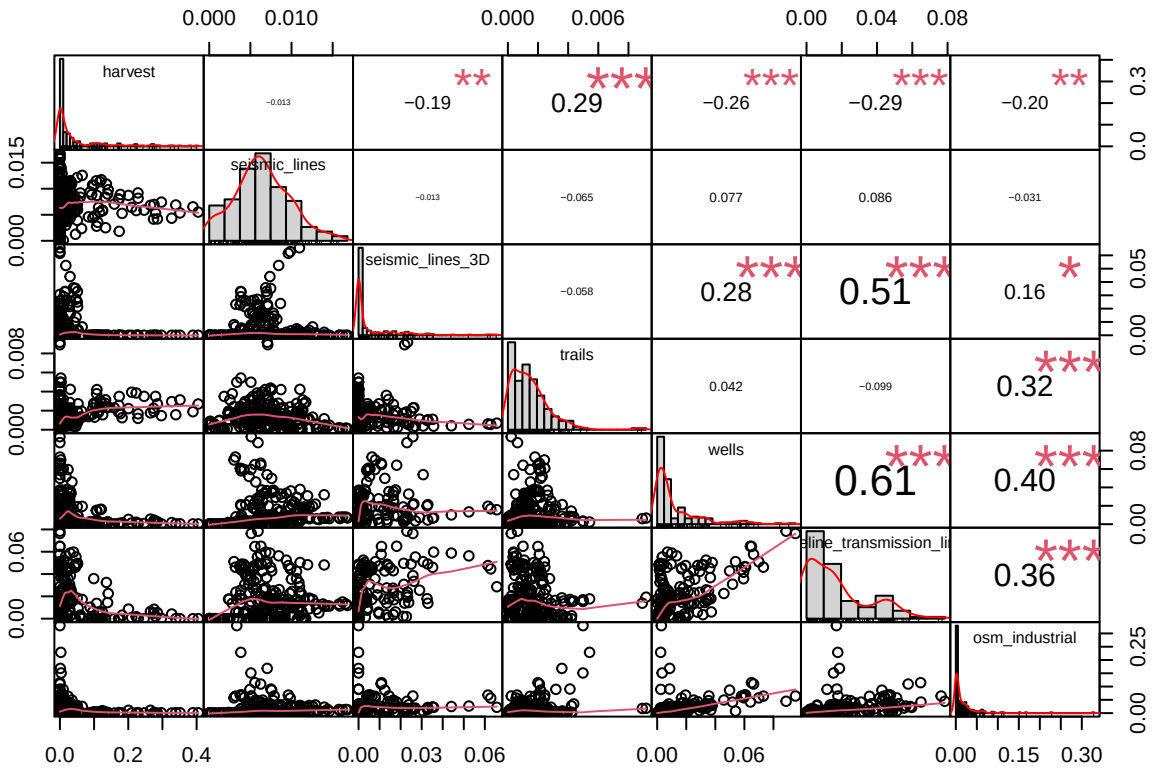

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.017
−0.18
0.25
−0.24
−0.26
−0.17
0.019
−0.071
0.069
0.0072
−0.033
−0.034
0.27
0.41
0.12
0.029
−0.13
0.28
0.51
0.36
0.34
0.000 0.010
0.000 0.006
0.00 0.06
0.000 0.015
0.000 0.010
0.00 0.08
0.0 0.3
0.00 0.05
0.00 0.08
0.0 0.3
0.0 0.2 0.4
0.00 0.03 0.06
0.00 0.06
0.0 0.2 0.4

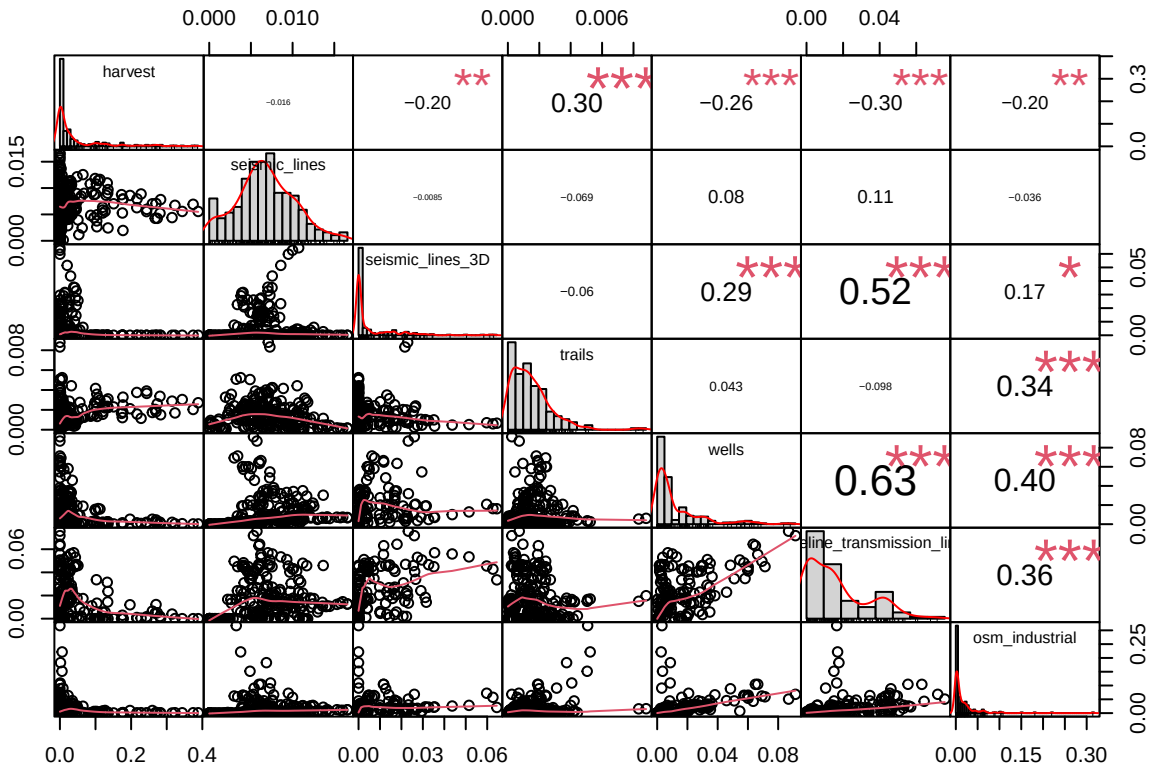

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.016
−0.18
0.26
−0.24
−0.26
−0.18
0.0057
−0.075
0.062
0.012
−0.037
−0.047
0.27
0.43
0.12
0.03
−0.12
0.28
0.53
0.36
0.35
0.000
0.010
0.006
0.00
0.06
0.015
0.05
0.008
0.08
0.2
0.4
0.03

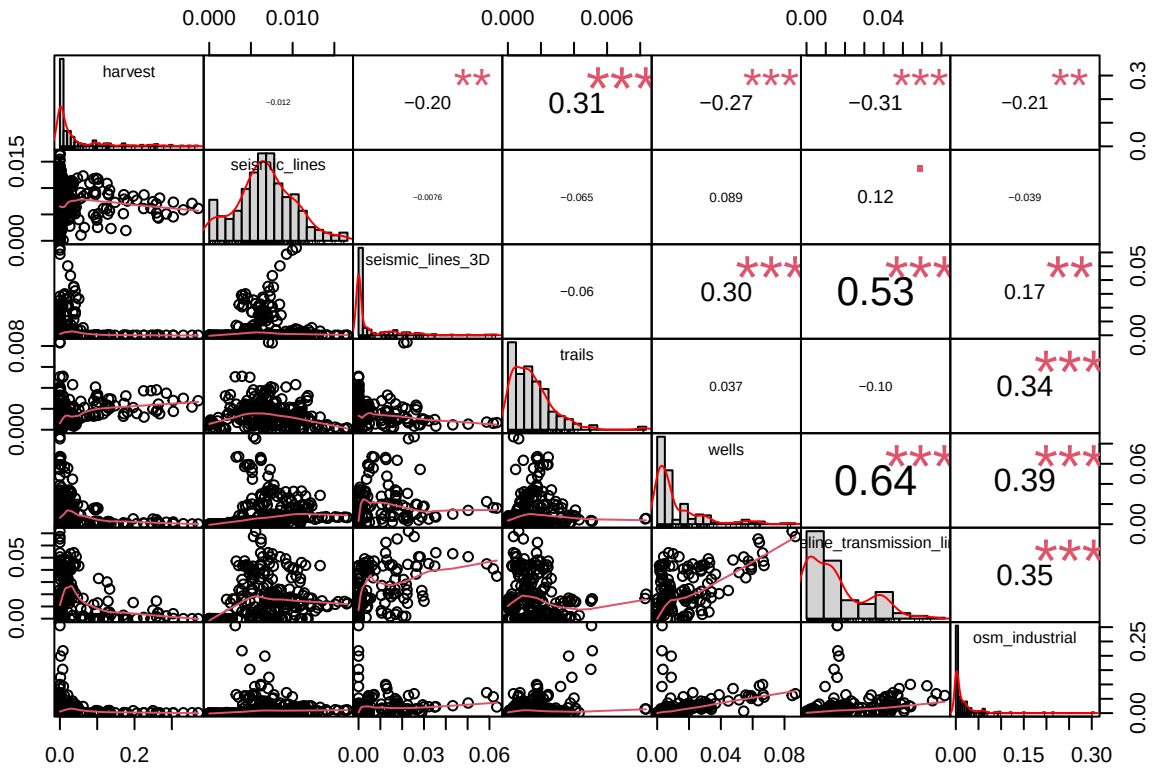

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.019
−0.18
0.26
−0.24
−0.27
−0.18
−0.0014
−0.071
0.066
0.029
−0.032
−0.048
0.27
0.45
0.13
0.029
−0.12
0.30
0.55
0.36
0.35
0.000 0.010
0.000 0.006
0.00 0.04 0.08
0.000 0.015
0.000 0.008
0.00 0.08
0.0 0.3
0.00 0.05
0.00 0.08
0.0 0.2
0.0 0.2 0.4
0.00 0.03 0.06
0.00 0.06
0.0 0.2

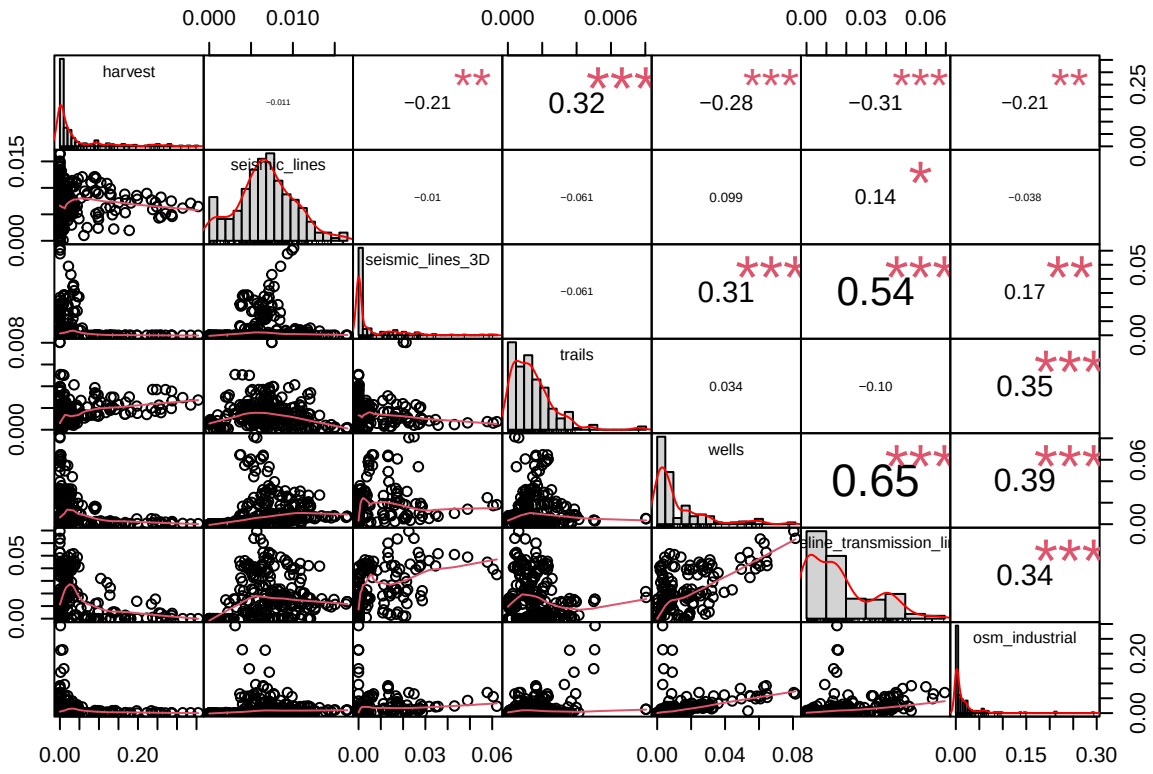

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.18
0.27
−0.24
−0.27
−0.19
−0.07
0.066
0.042
−0.054
0.28
0.48
0.14
0.037
−0.11
0.31
0.58
0.37
0.35

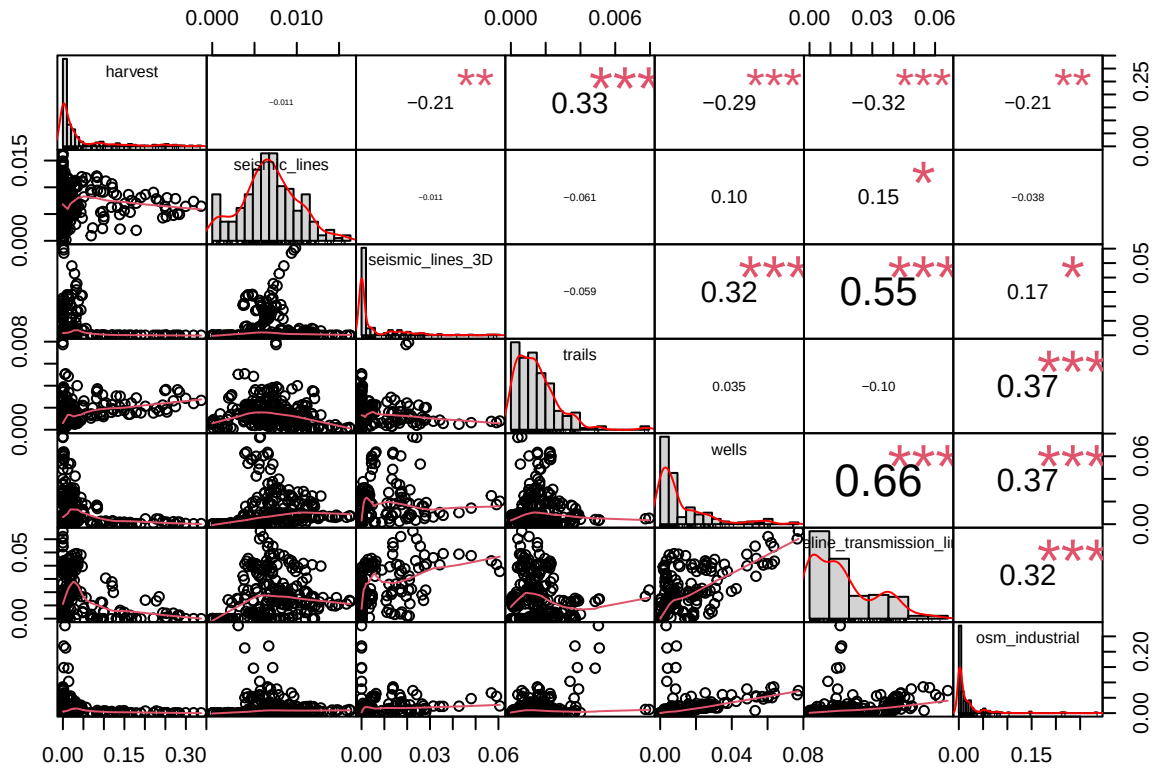

harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.0091
−0.19
0.28
−0.25
−0.28
−0.20
−0.014
−0.068
0.069
0.061
−0.028
−0.057
0.28
0.49
0.15
0.044
−0.10
0.32
0.59
0.38
0.35
0.000 0.010
0.000 0.006
0.00 0.04 0.08
0.000 0.015
0.000 0.008
0.00 0.06
0.0 0.3
0.00 0.05
0.00 0.08
0.00 0.25
0.0 0.2 0.4
0.00 0.03 0.06
0.00 0.06
0.00 0.15 0.30


harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.013
** −0.19
*** 0.29
*** −0.26
*** −0.29
** −0.20
−0.013
−0.065
0.077
0.086
−0.031
*** 0.28
*** 0.51
* 0.16
−0.058
0.042
−0.099
*** 0.32
*** 0.61
*** 0.40
*** 0.36
0.000 0.010
0.000 0.006
0.00 0.04 0.08
0.000 0.015
0.000 0.008
0.00 0.06
0.0 0.3
0.00 0.05
0.00 0.08
0.00 0.25
0.0 0.2 0.4
0.00 0.03 0.06
0.00 0.06
0.00 0.15 0.30


harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.20
0.30
−0.26
−0.30
−0.20
0.08
0.11
−0.036
−0.06
0.29
0.52
0.17
0.043
−0.098
0.34
0.63
0.40
0.36
0.0
0.2
0.4
0.00
0.03
0.06
0.00
0.04
0.08
0.00
0.15
0.30


harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.20
0.31
−0.27
−0.31
−0.21
0.089
0.12
−0.039
−0.06
0.30
0.53
0.17
0.037
−0.10
0.34
0.64
0.39
0.35


harvest
seismic_lines
seismic_lines_3D
trails
wells
line_transmission_li
osm_industrial
−0.011
−0.21
0.32
−0.28
−0.31
−0.21
−0.01
−0.061
0.099
0.14
−0.038
−0.061
0.31
0.54
0.17
0.034
−0.10
0.35
0.65
0.39
0.34
0.000 0.010
0.000 0.006
0.00 0.03 0.06
0.000 0.015
0.000 0.008
0.00 0.05
0.00 0.25
0.00 0.05
0.00 0.06
0.00 0.20
0.00 0.20
0.00 0.03 0.06
0.00 0.04 0.08
0.00 0.15 0.30




```
## $‘250 meter buffer‘
## NULL
##
## $‘500 meter buffer‘
## NULL
##
## $‘750 meter buffer‘
## NULL
##
## $‘1000 meter buffer‘
## NULL
##
## $‘1250 meter buffer‘
## NULL
##
## $‘1500 meter buffer‘
## NULL
##
## $‘1750 meter buffer‘
## NULL
##
## $‘2000 meter buffer‘
## NULL
##
## $‘2250 meter buffer‘
## NULL
```

```
##
## $`2500 meter buffer`
## NULL
##
## $`2750 meter buffer`
## NULL
##
## $`3000 meter buffer`
## NULL
##
## $`3250 meter buffer`
## NULL
##
## $`3500 meter buffer`
## NULL
##
## $`3750 meter buffer`
## NULL
##
## $`4000 meter buffer`
## NULL
##
## $`4250 meter buffer`
## NULL
##
## $`4500 meter buffer`
## NULL
##
## $`4750 meter buffer`
## NULL
##
## $`5000 meter buffer`
## NULL
```

Wells and the new pipelines and transmission lines variable seem to become increasingly correlated as the buffer size increases but the max r squared value is 0.66 so we will leave it in for now and discuss with others about whether or not to remove it since it's on the cusp.

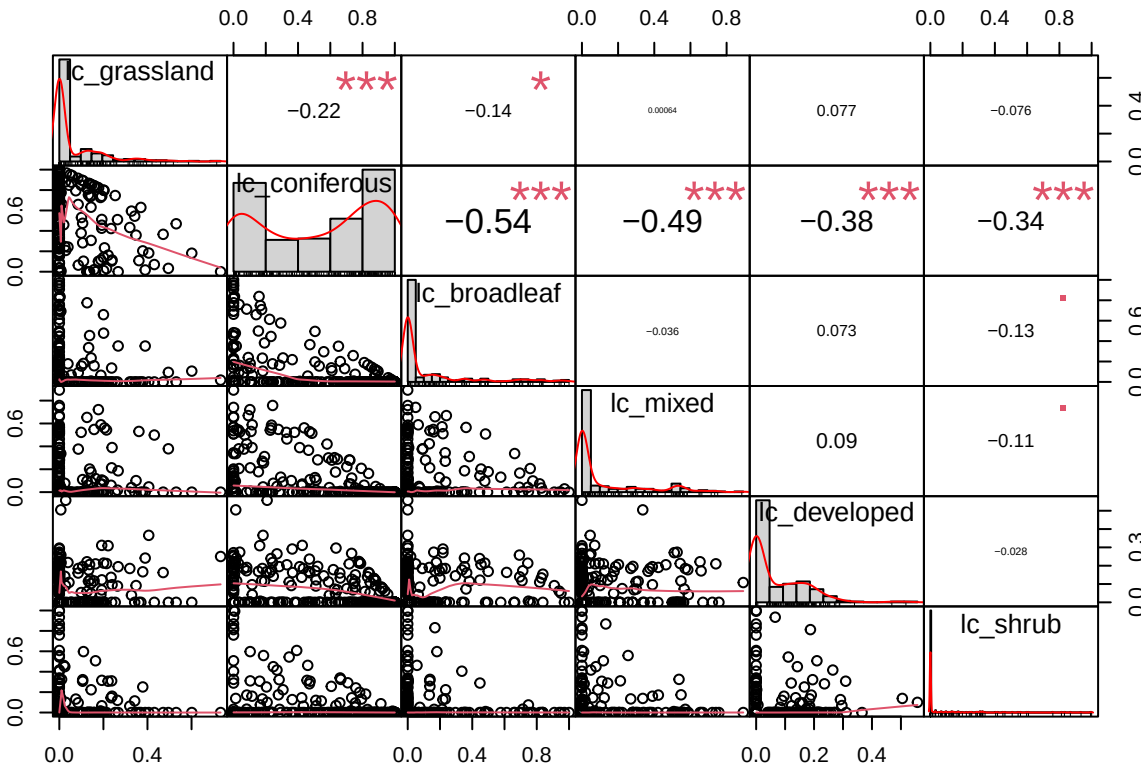
Landscape

Now let's do the same for the landscape data

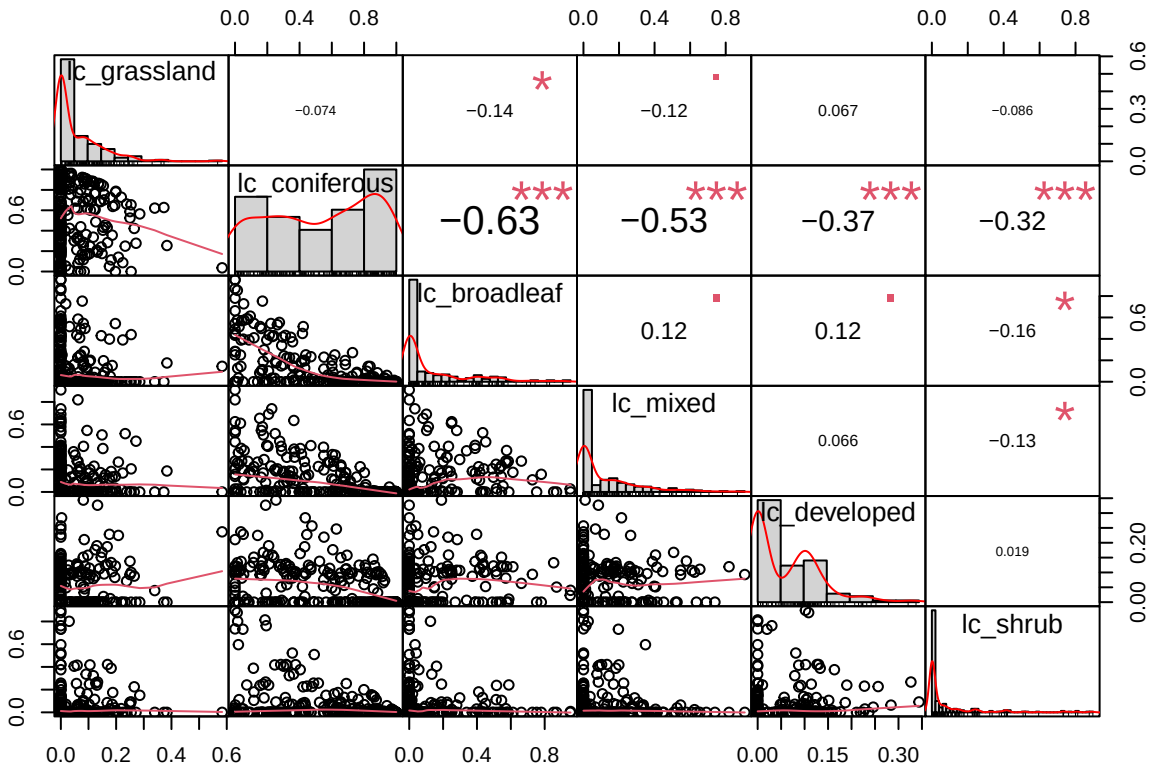
```
osm_landscape_df_2021_2022 <- osm_final_df_2021_2022 %>%

   # use purr to apply following data manipulation steps to all the buffer data frames
  purrr::map(

    ~.x %>%

      # de-select columns for developed data
      select(!harvest:wells &
               !osm_industrial)
  )
```

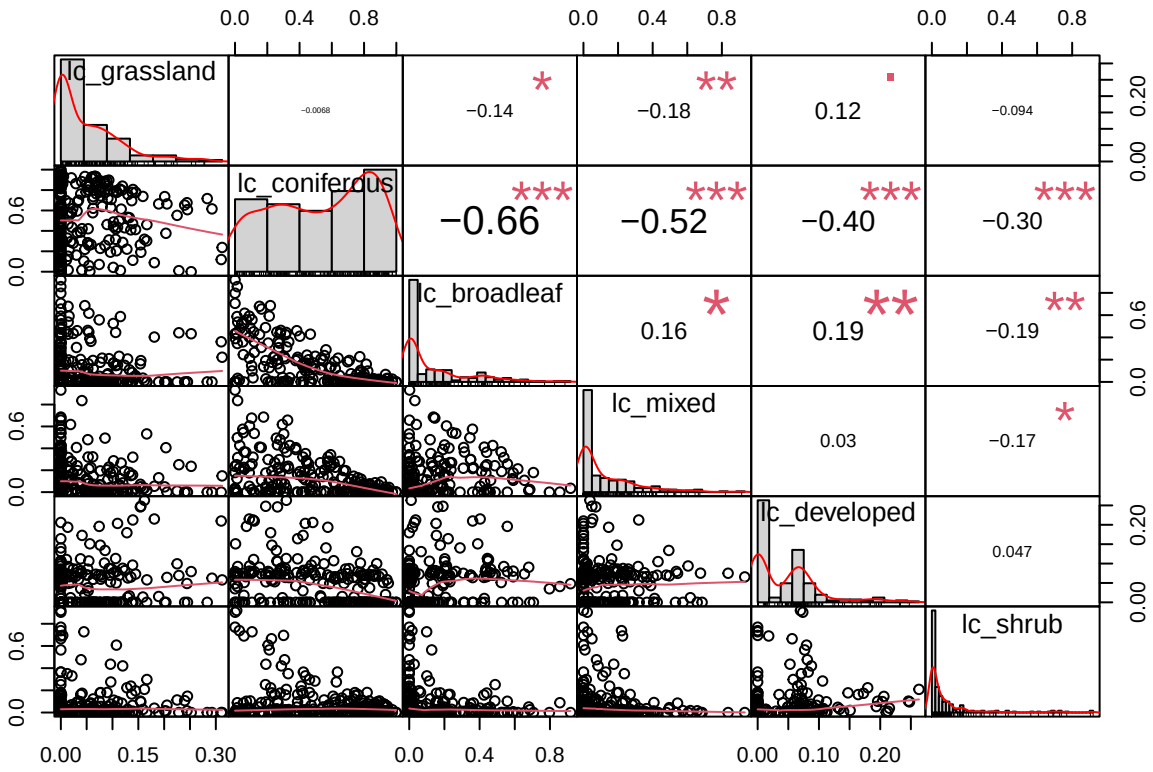
Now we will follow the same process with the landscape data for generating correlation plots and assessing whaich variables need to be removed or combined

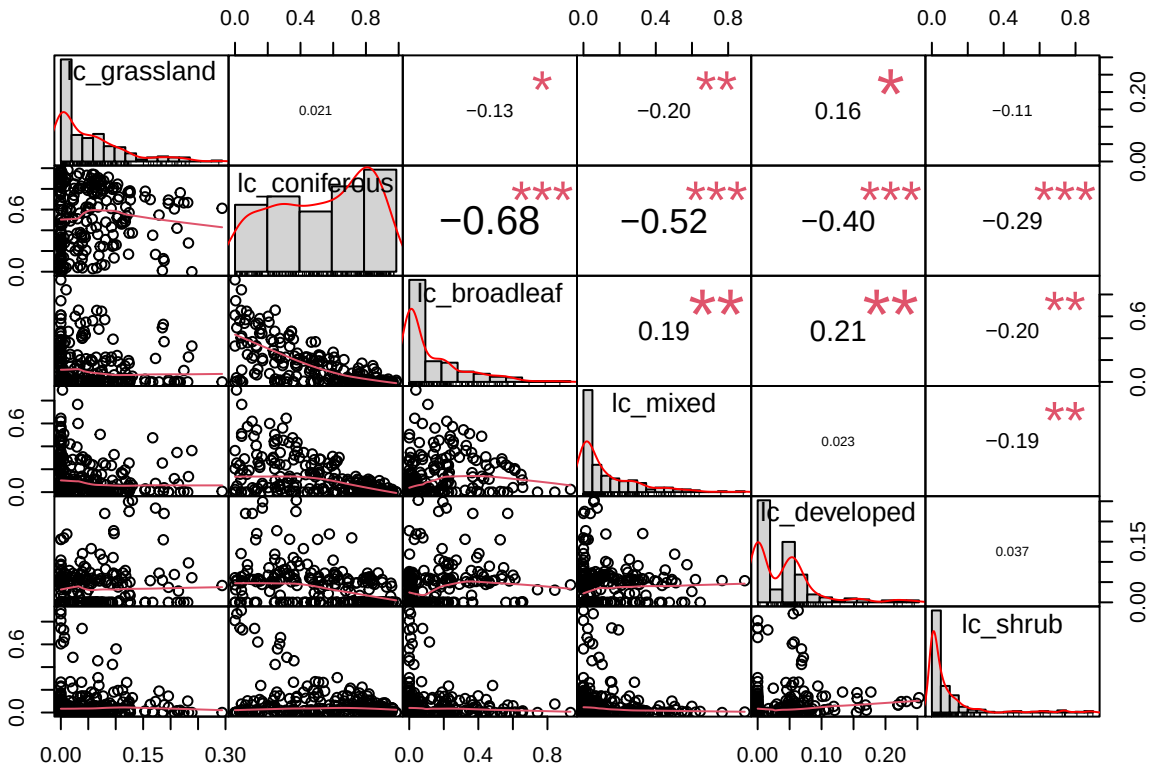
```
osm_landscape_df_2021_2022 %>%

  purrr::map(
    ~.x %>%

      # select only columns with covariates not other info
      select(lc_grassland:lc_shrub) %>%

      # use chart.correlation
      chart.Correlation(.,
                        histogram = TRUE,
                        method = "pearson")
  )
```

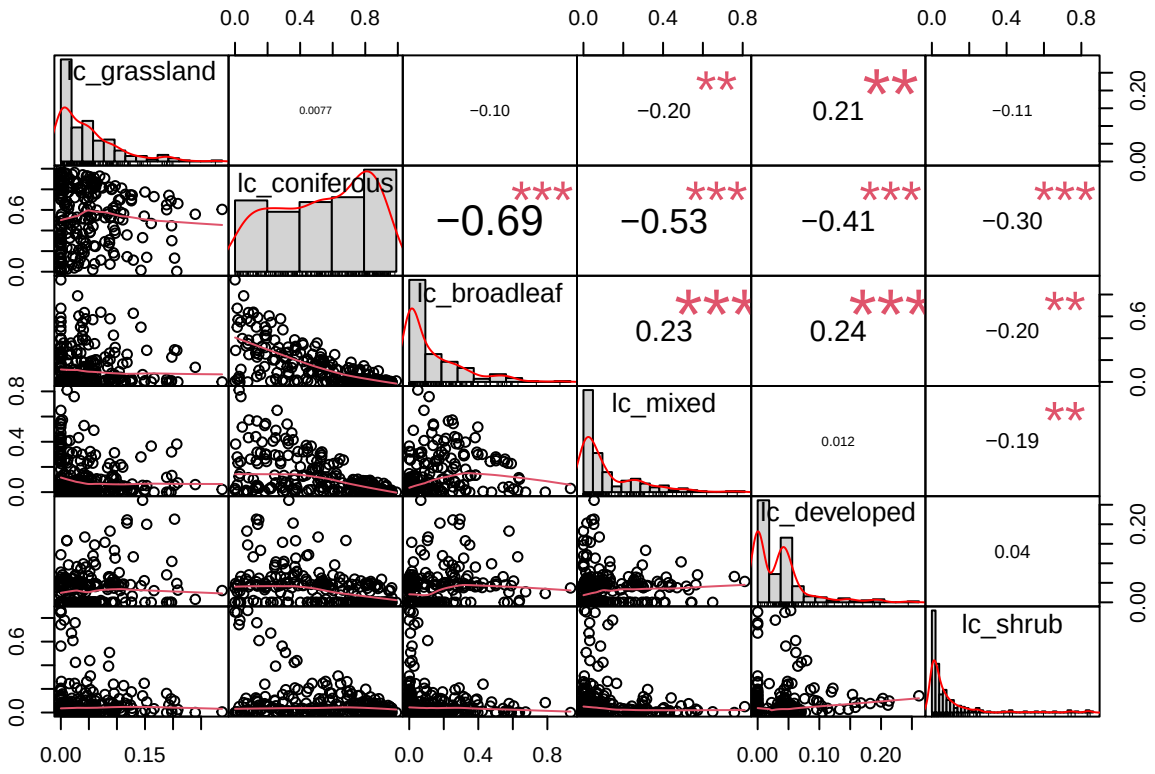


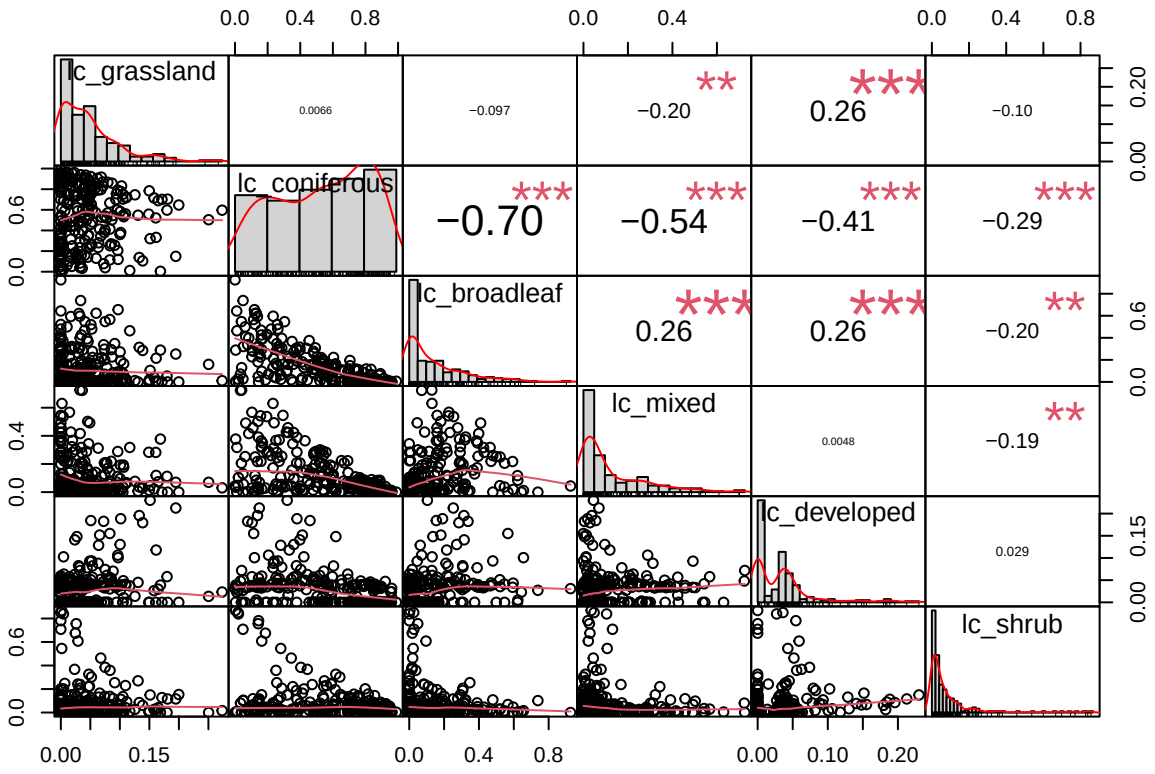
Correlation plots

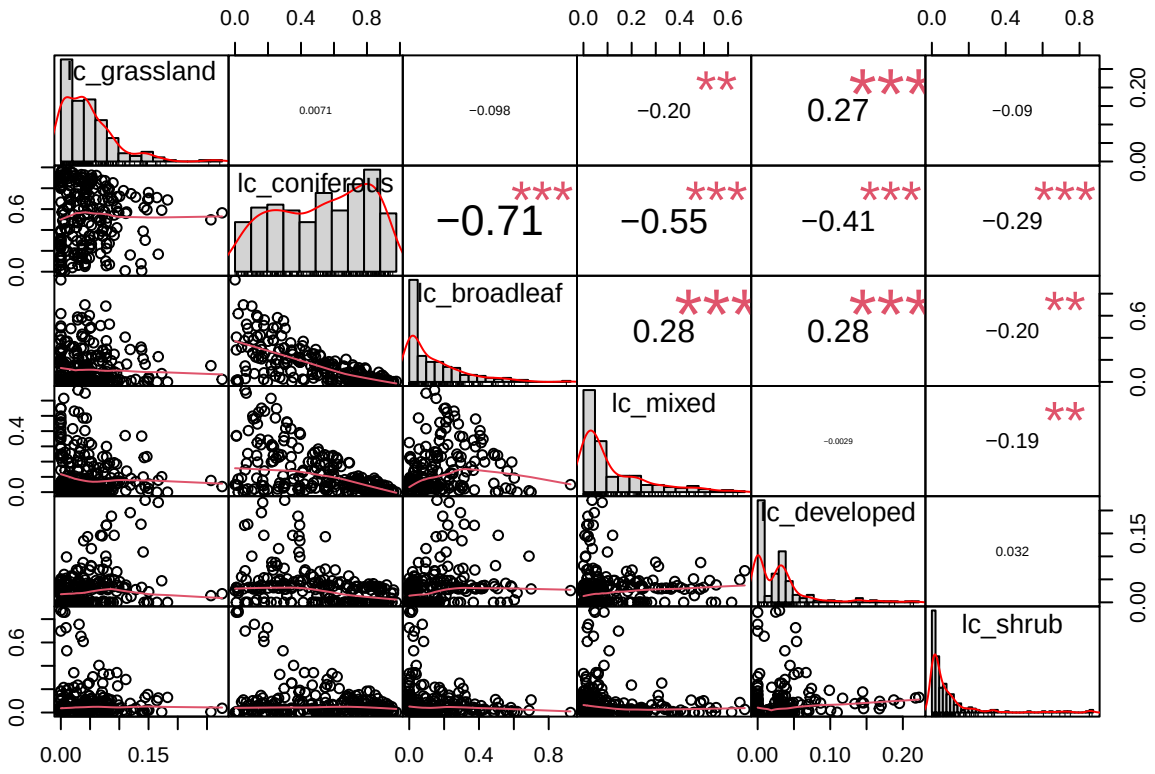


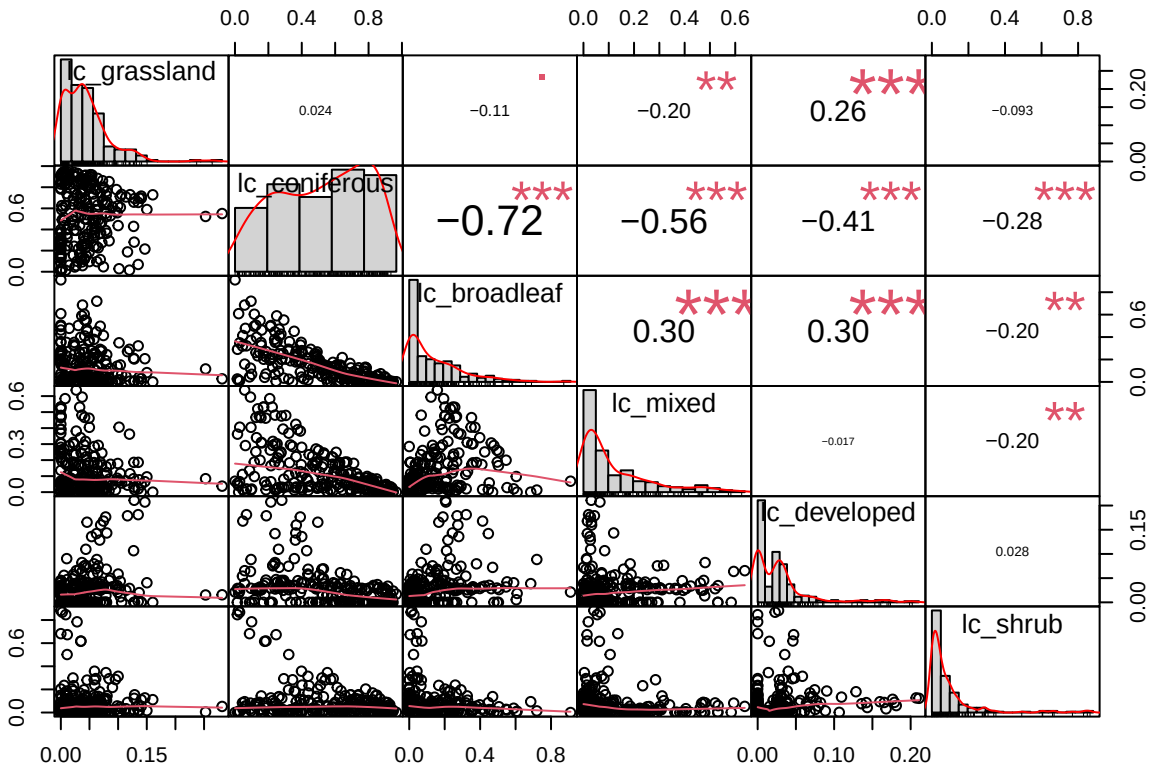

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
−0.14
−0.18
0.12
−0.094
−0.66
−0.52
−0.40
−0.30
0.16
0.19
−0.19
0.03
−0.17
0.047
0.0
0.4
0.8
0.6
0.20
0.00
0.15
0.30
0.10

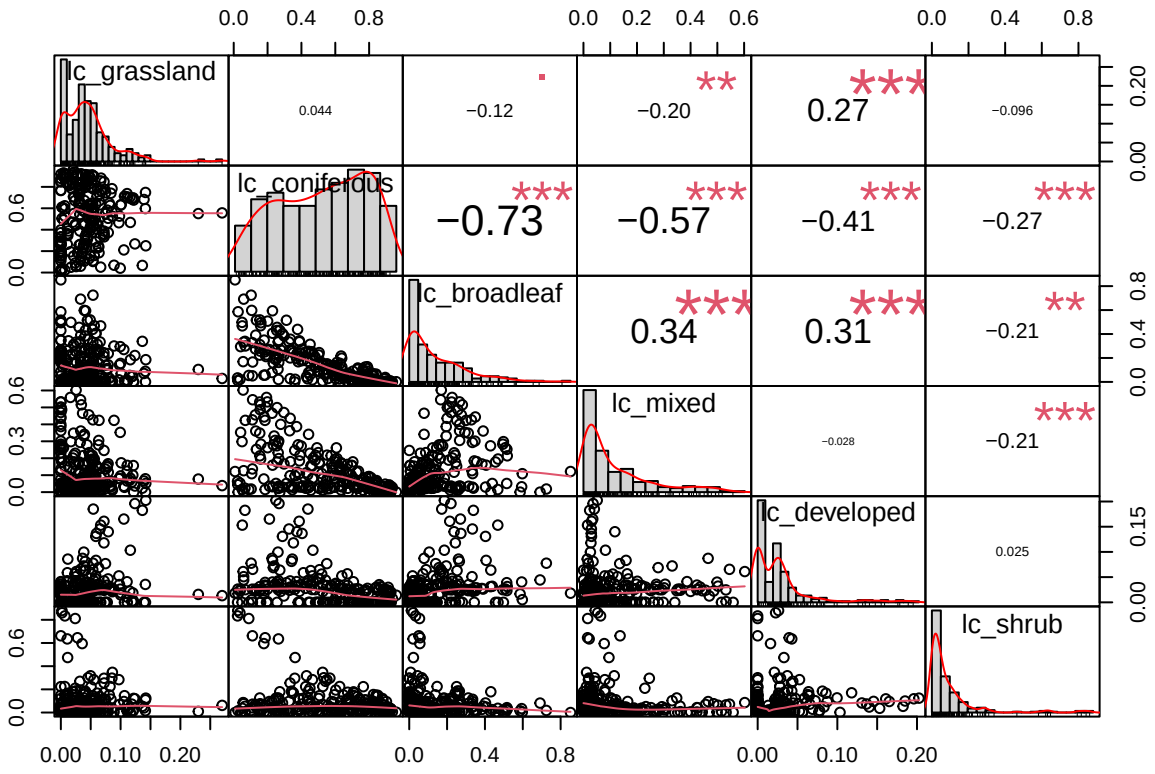

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.021
−0.13
−0.20
0.16
−0.11
−0.68
−0.52
−0.40
−0.29
0.19
0.21
−0.20
0.023
−0.19
0.037
*
**
***
0.0
0.4
0.8
0.6
0.00
0.15
0.30
0.10
0.20

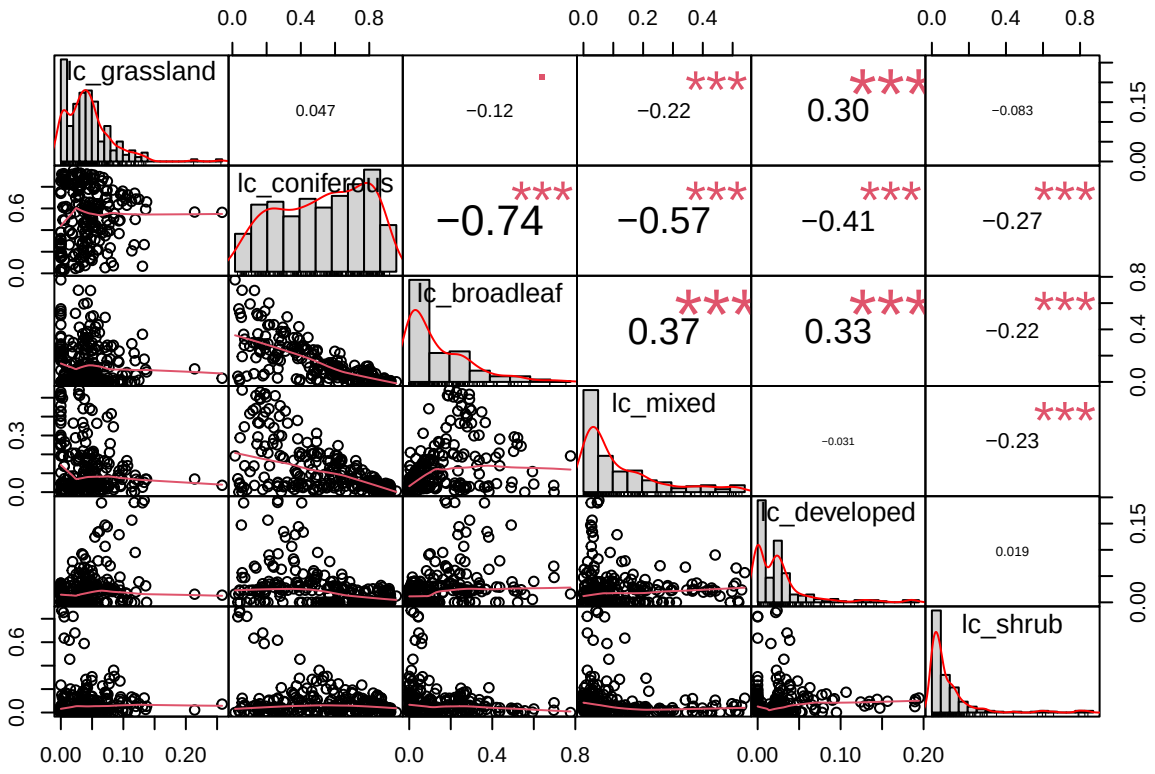

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.0077
−0.10
−0.20
0.21
−0.11
−0.69
−0.53
−0.41
−0.30
0.23
0.24
−0.20
0.012
−0.19
0.04
0.0
0.4
0.8
0.6
0.00
0.15
0.10
0.20

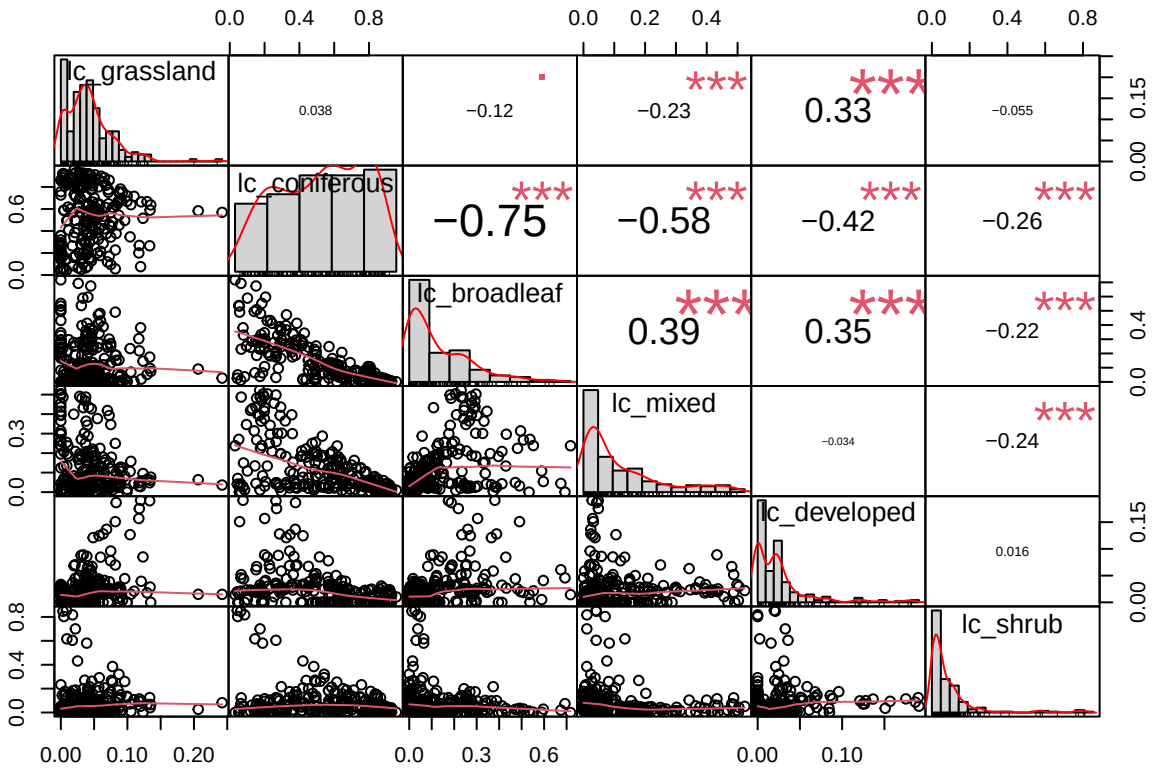

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.0066
−0.097
−0.20
0.26
−0.10
−0.70
−0.54
−0.41
−0.29
0.26
0.26
−0.20
0.0048
−0.19
0.029
0.0
0.4
0.8
0.00
0.15
0.10
0.20
0.6

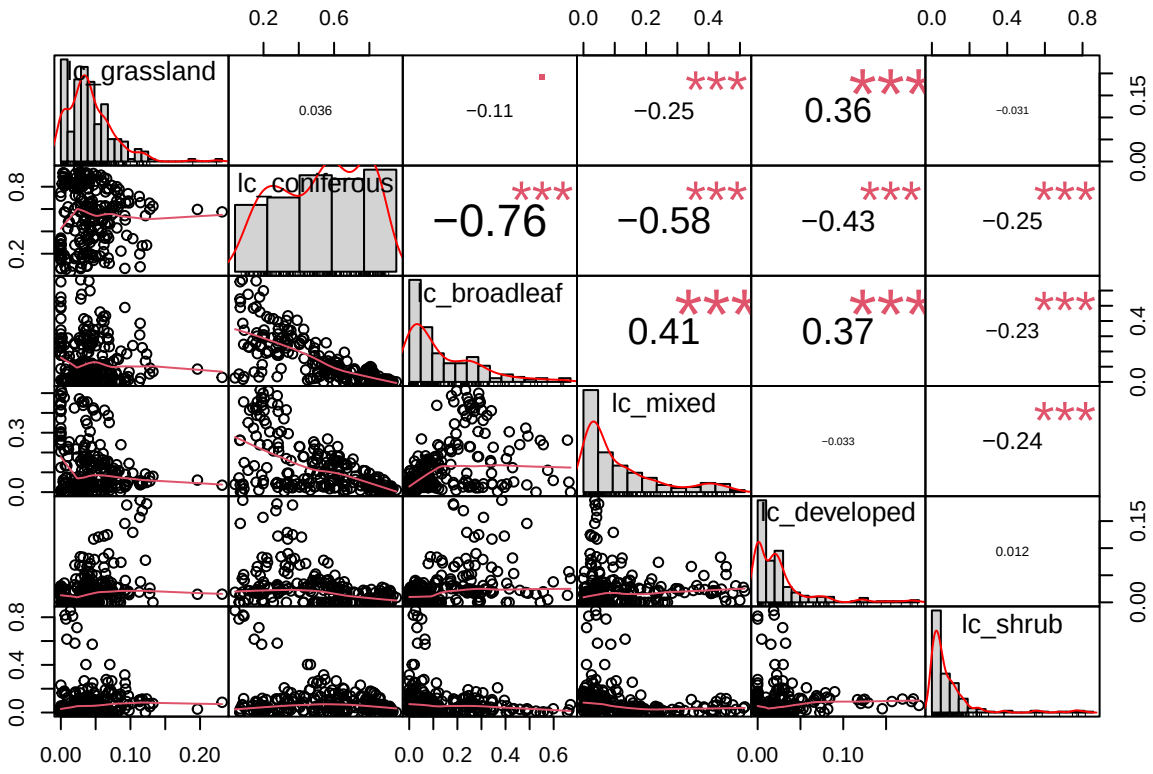

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.0071
−0.098
−0.20
0.27
−0.09
−0.71
−0.55
−0.41
−0.29
0.28
0.28
−0.20
−0.0029
−0.19
0.032
0.0
0.4
0.8
0.0
0.2
0.4
0.6
0.0
0.4
0.8
0.00
0.20
0.6
0.0
0.6
0.0
0.4
0.0
0.6
0.0
0.6
0.0
0.15
0.00
0.15
0.0
0.4
0.8
0.00
0.10
0.20

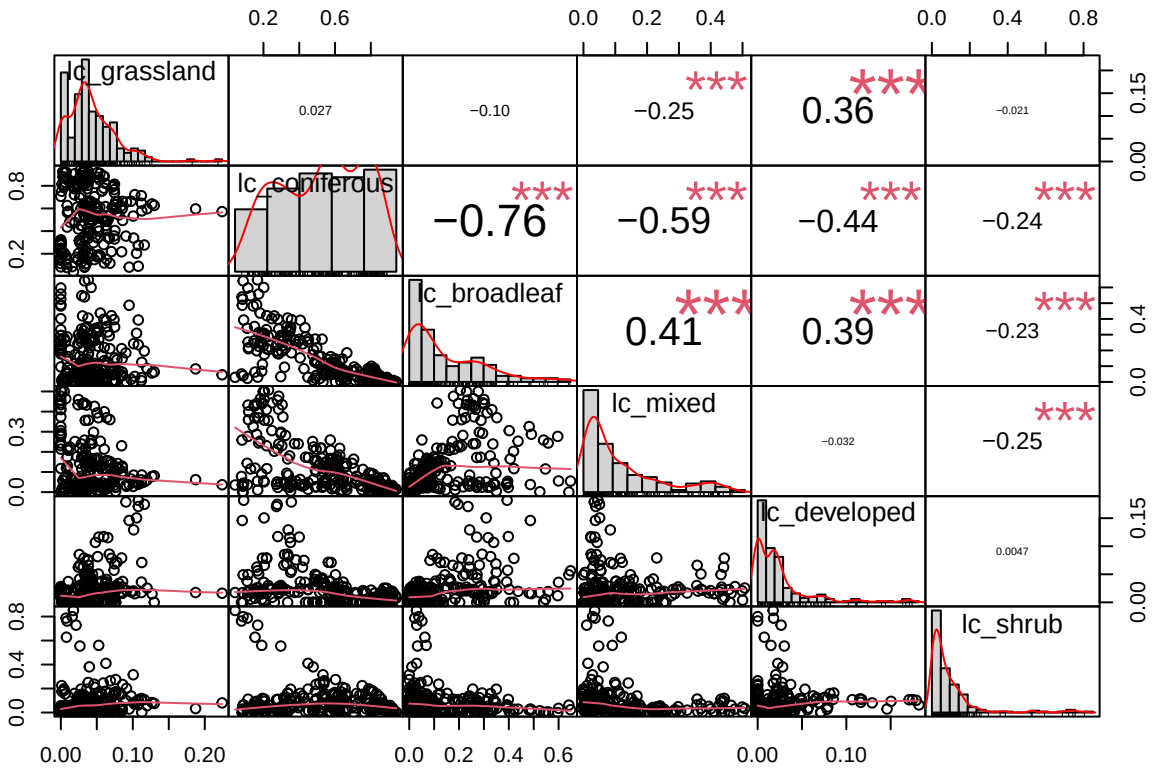

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.024
−0.11
−0.20
0.26
−0.093
−0.72
−0.56
−0.41
−0.28
0.30
0.30
−0.20
−0.017
−0.20
0.028

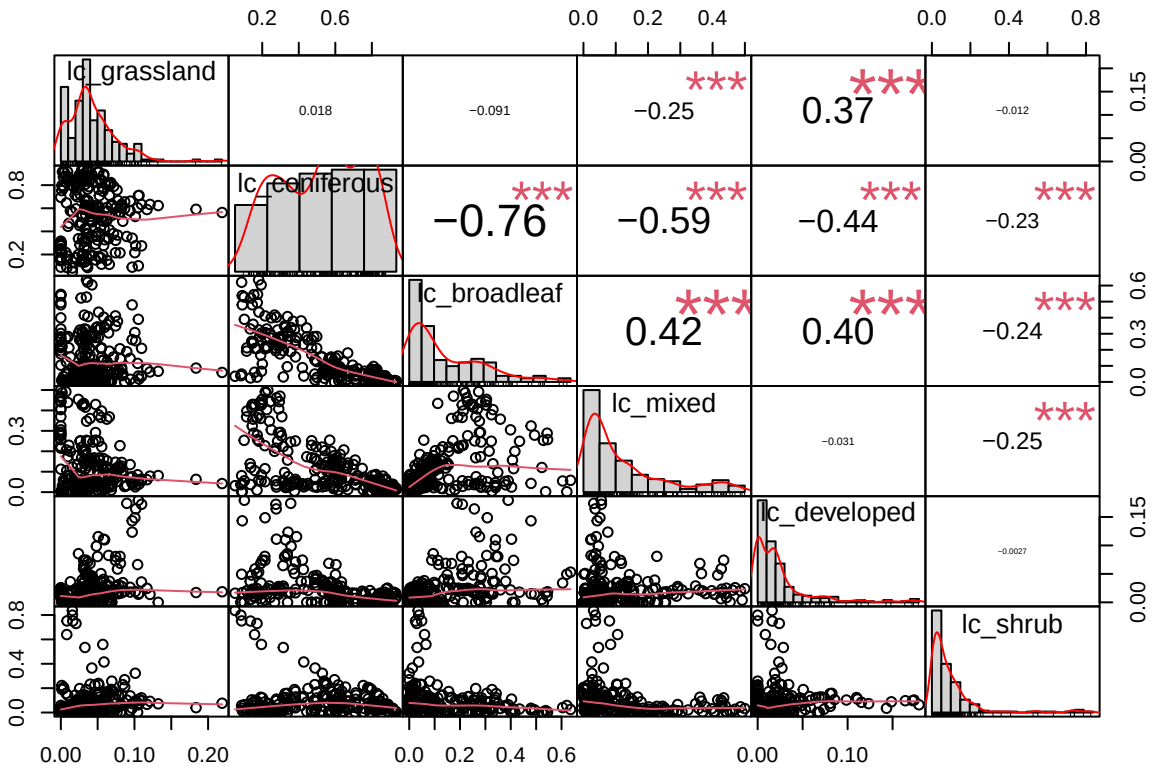

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.044
−0.12
−0.20
0.27
−0.096
−0.73
−0.57
−0.41
−0.27
0.34
0.31
−0.21
−0.028
−0.21
0.025

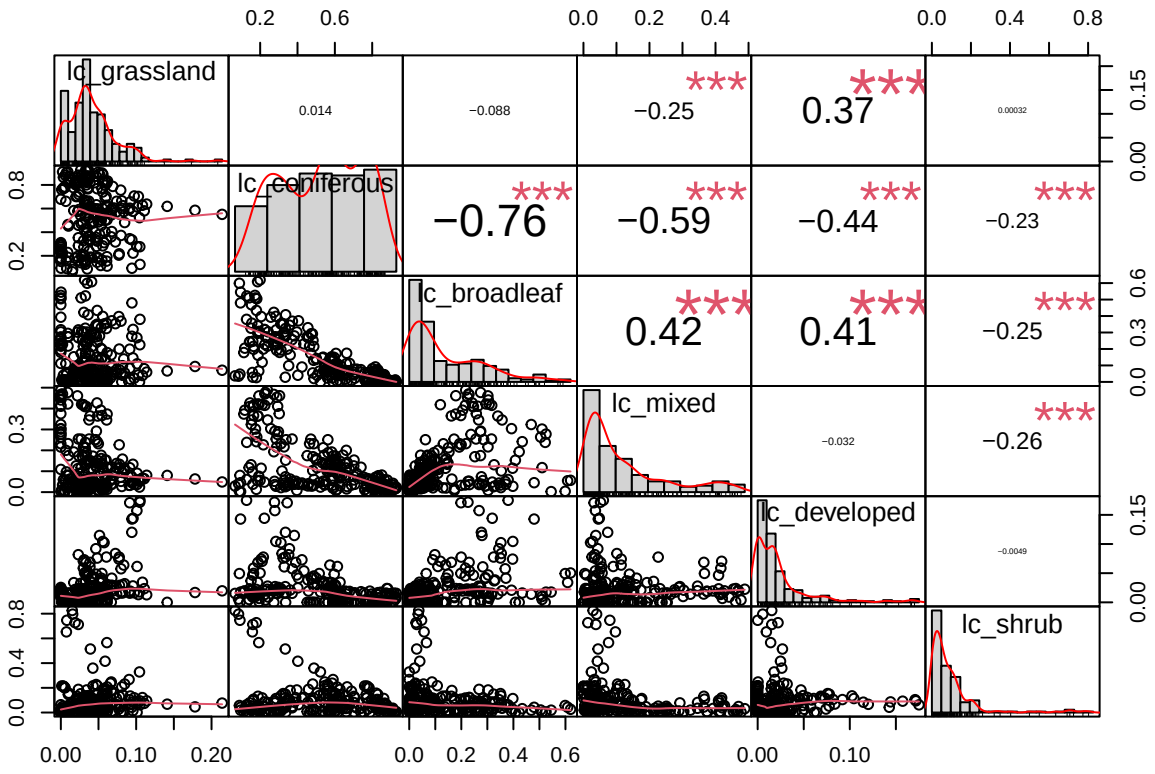

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.047
−0.12
−0.22
0.30
−0.083
−0.74
−0.57
−0.41
−0.27
0.37
0.33
−0.22
−0.031
−0.23
0.019
0.0 0.4 0.8
0.0 0.2 0.4
0.0 0.4 0.8
0.0 0.6
0.0 0.3
0.0 0.8
0.00 0.15
0.00 0.10 0.20
0.0 0.4 0.8
0.00 0.10 0.20

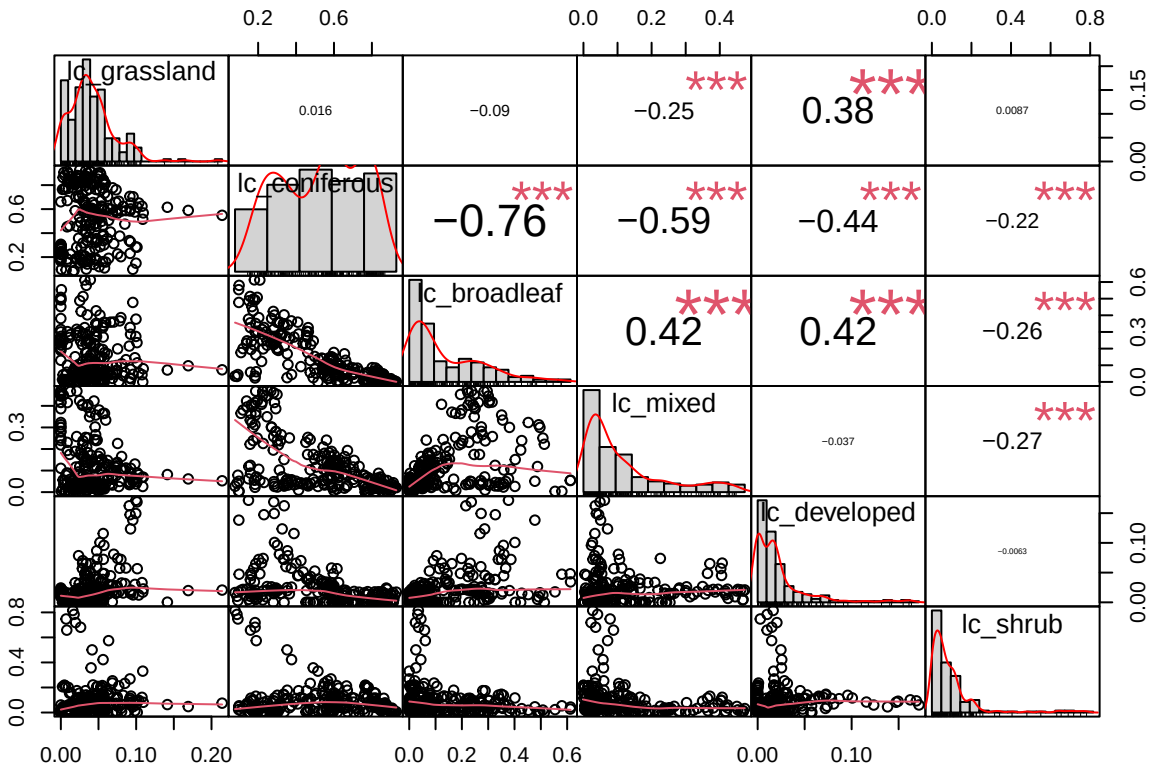

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.038
−0.12
−0.23
0.33
−0.055
−0.75
−0.58
−0.42
−0.26
0.39
0.35
−0.22
−0.034
−0.24
0.016
0.0 0.4 0.8
0.0 0.2 0.4
0.0 0.4 0.8
0.0 0.6
0.0 0.3
0.0 0.4 0.8
0.00 0.15
0.0 0.4
0.00 0.15
0.00 0.10 0.20
0.0 0.3 0.6
0.00 0.10

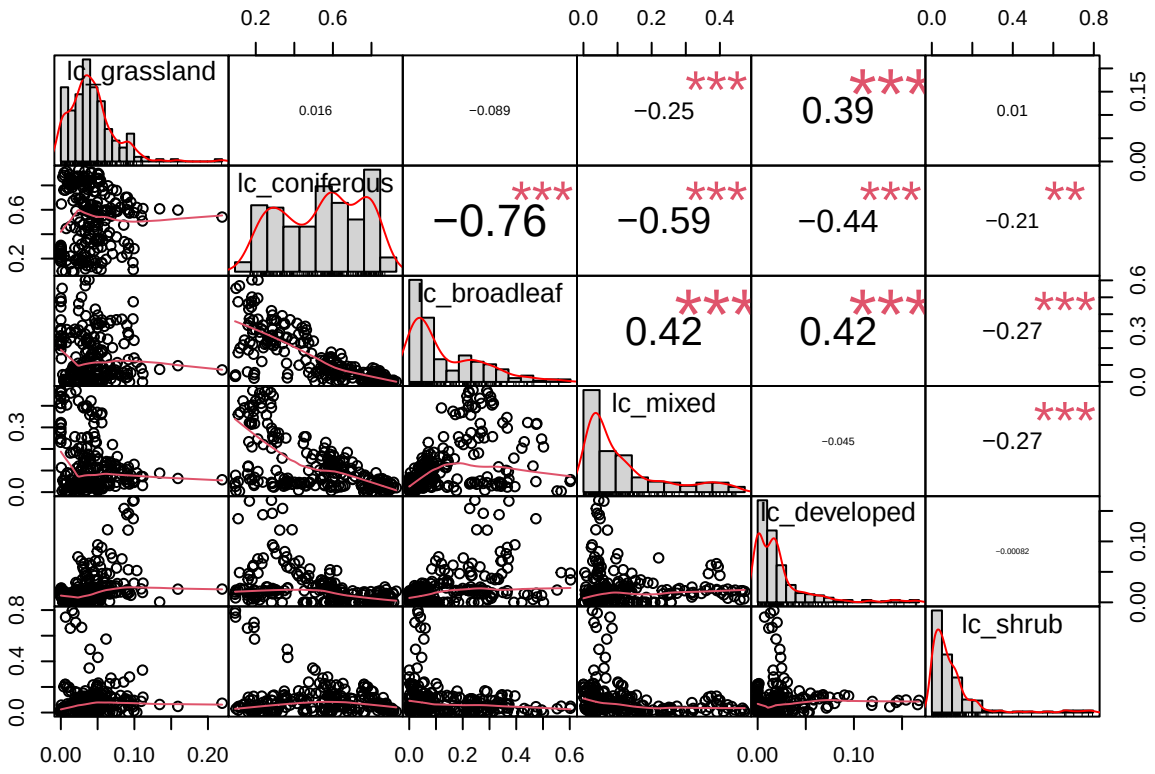

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.036
−0.11
−0.25
0.36
−0.031
−0.76
−0.58
−0.43
−0.25
0.41
0.37
−0.23
−0.033
−0.24
0.012

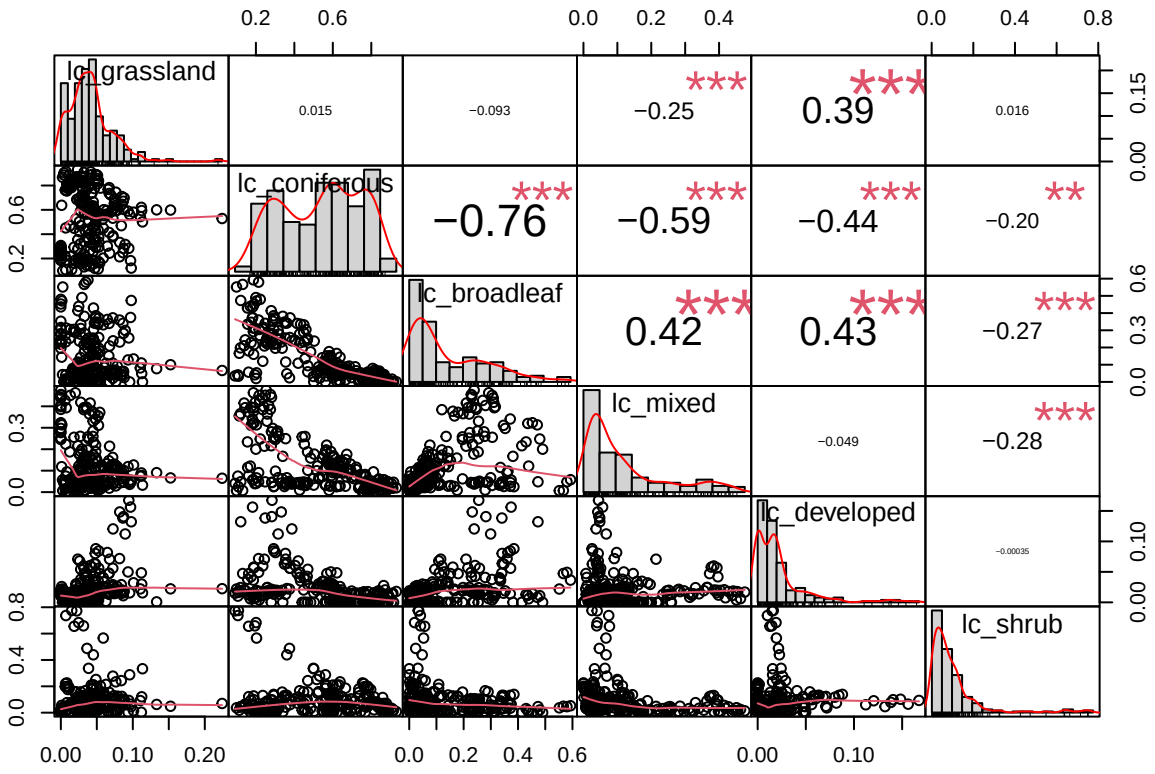

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.027
−0.10
−0.25
0.36
−0.021
−0.76
−0.59
−0.44
−0.24
0.41
0.39
−0.23
−0.032
−0.25
0.0047

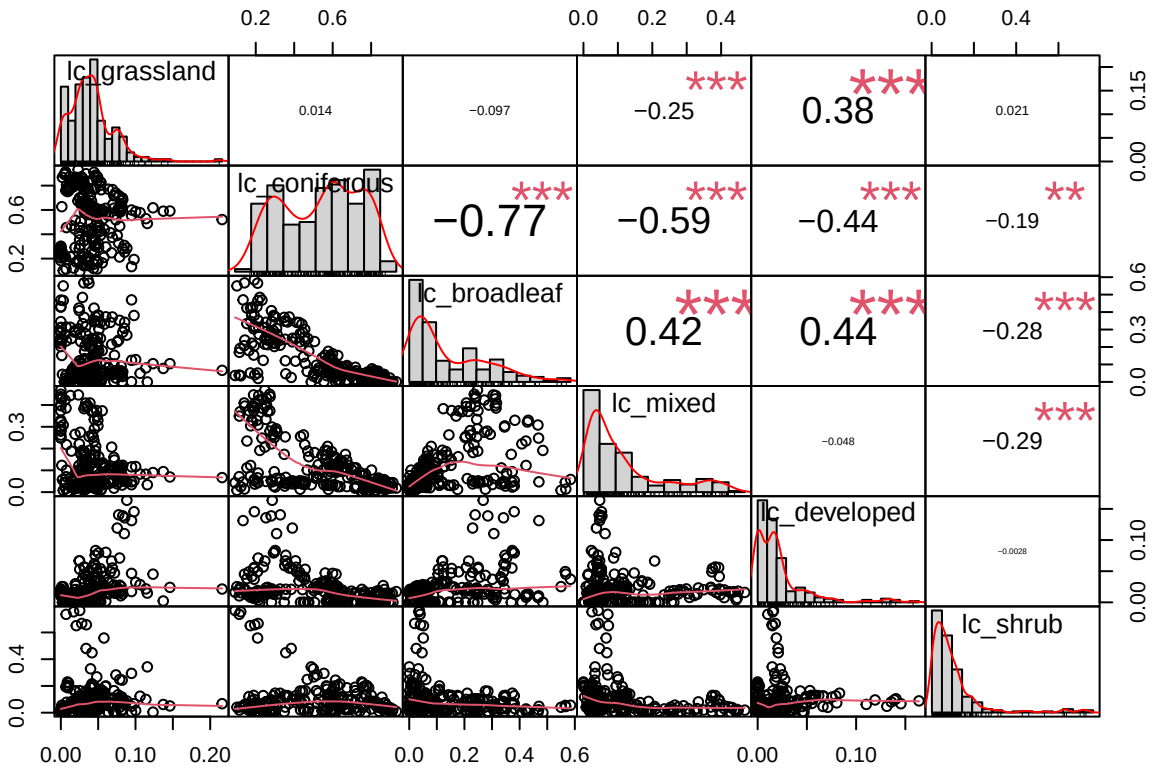

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.018
−0.091
−0.25
0.37
−0.012
−0.76
−0.59
−0.44
−0.23
0.42
0.40
−0.24
−0.031
−0.25
−0.0027
***

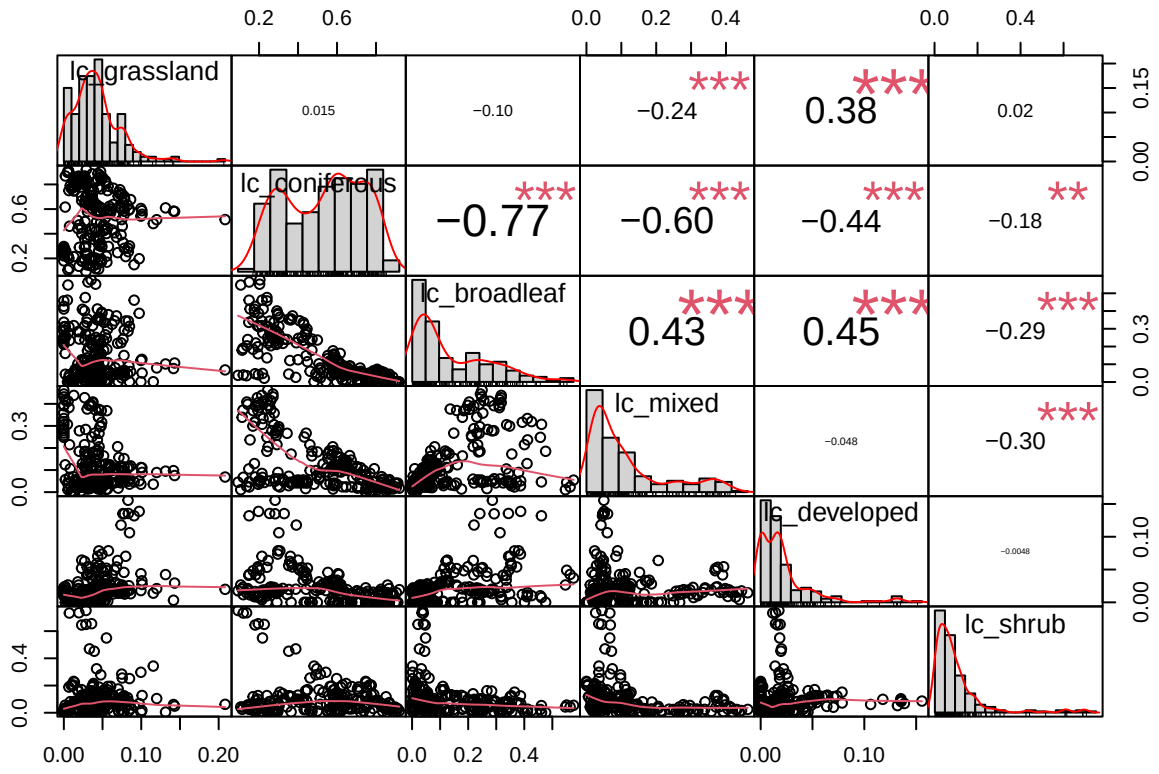

lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.014
−0.088
−0.25
0.37
0.00032
−0.76
−0.59
−0.44
−0.23
0.42
0.41
−0.25
−0.032
−0.26
−0.0049
***


lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.016
−0.09
−0.25
0.38
0.0087
−0.76
−0.59
−0.44
−0.22
0.42
0.42
−0.26
−0.037
−0.27
−0.0063
***


lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.016
−0.089
−0.25
0.39
0.01
−0.76
−0.59
−0.44
−0.21
0.42
0.42
−0.27
−0.045
−0.27
−0.00082
***
**


lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.015
−0.093
−0.25
0.39
0.016
−0.76
−0.59
−0.44
−0.20
0.42
0.43
−0.27
−0.049
−0.28
−0.00035
***
**


lc_grassland
lc_coniferous
lc_broadleaf
lc_mixed
lc_developed
lc_shrub
0.014
−0.097
−0.25
0.38
0.021
−0.77
−0.59
−0.44
−0.19
0.42
0.44
−0.28
−0.048
−0.29
−0.0028
***
**




```
## $`250 meter buffer`
## NULL
##
## $`500 meter buffer`
## NULL
##
## $`750 meter buffer`
## NULL
##
## $`1000 meter buffer`
## NULL
##
## $`1250 meter buffer`
## NULL
##
## $`1500 meter buffer`
## NULL
##
## $`1750 meter buffer`
## NULL
##
## $`2000 meter buffer`
## NULL
##
## $`2250 meter buffer`
## NULL
```

```
##
## $`2500 meter buffer`
## NULL
##
## $`2750 meter buffer`
## NULL
##
## $`3000 meter buffer`
## NULL
##
## $`3250 meter buffer`
## NULL
##
## $`3500 meter buffer`
## NULL
##
## $`3750 meter buffer`
## NULL
##
## $`4000 meter buffer`
## NULL
##
## $`4250 meter buffer`
## NULL
##
## $`4500 meter buffer`
## NULL
##
## $`4750 meter buffer`
## NULL
##
## $`5000 meter buffer`
## NULL
```

Below is a summary of the different buffer sizes and correlated variables

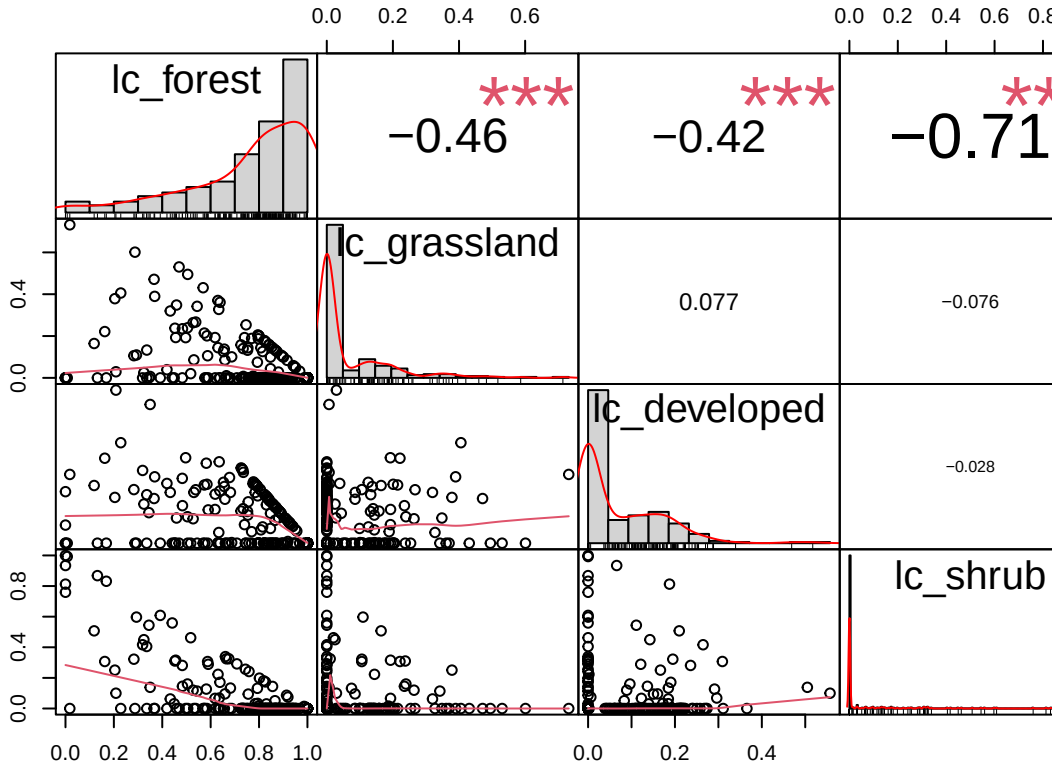
- 250m
  None >0.6
- 500m coniferous + broadleaf 0.63
- 750m
  coniferous + broadleaf 0.66
- 1000m
  coniferous + braodleaf 0.68

  Coniferous and broadleaf appear to be the only variables highly correlated so I think what we will do is combine all 3 forest types into one category, because we really aren't interested in interpreting these individual variables but rather seeing if there are differences in which buffer is the best fit between development and landscape features.

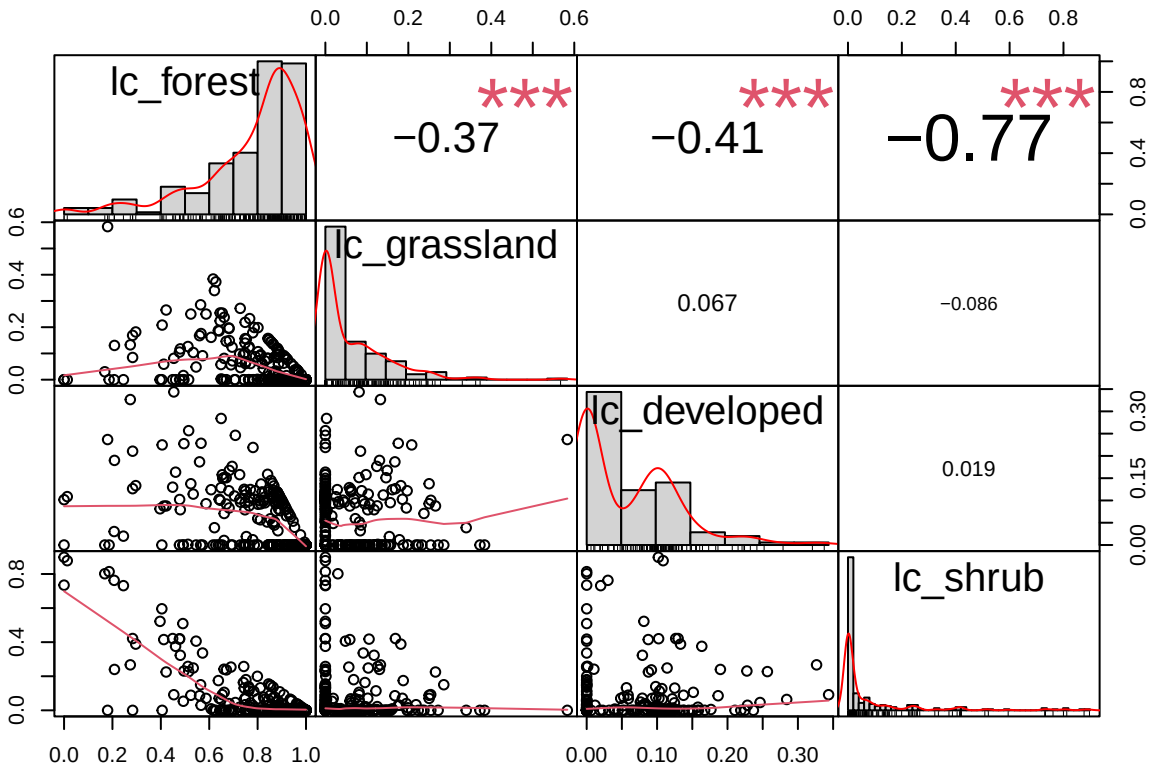
```
osm_landscape_df_2021_2022 <- osm_landscape_df_2021_2022 %>%

  # use purrr to apply the functions to all data frames
  purrr::map(

    ~.x %>%

      # use mutate to combine forest types into one variable
      mutate(lc_forest = lc_coniferous + lc_broadleaf + lc_mixed) %>%

      # remove old variables
      select(! c(lc_coniferous, lc_broadleaf, lc_mixed)) %>%

      # relocate the new column with. the other covariates
      relocate(lc_forest,
               .after = buff_dist)
  )
```

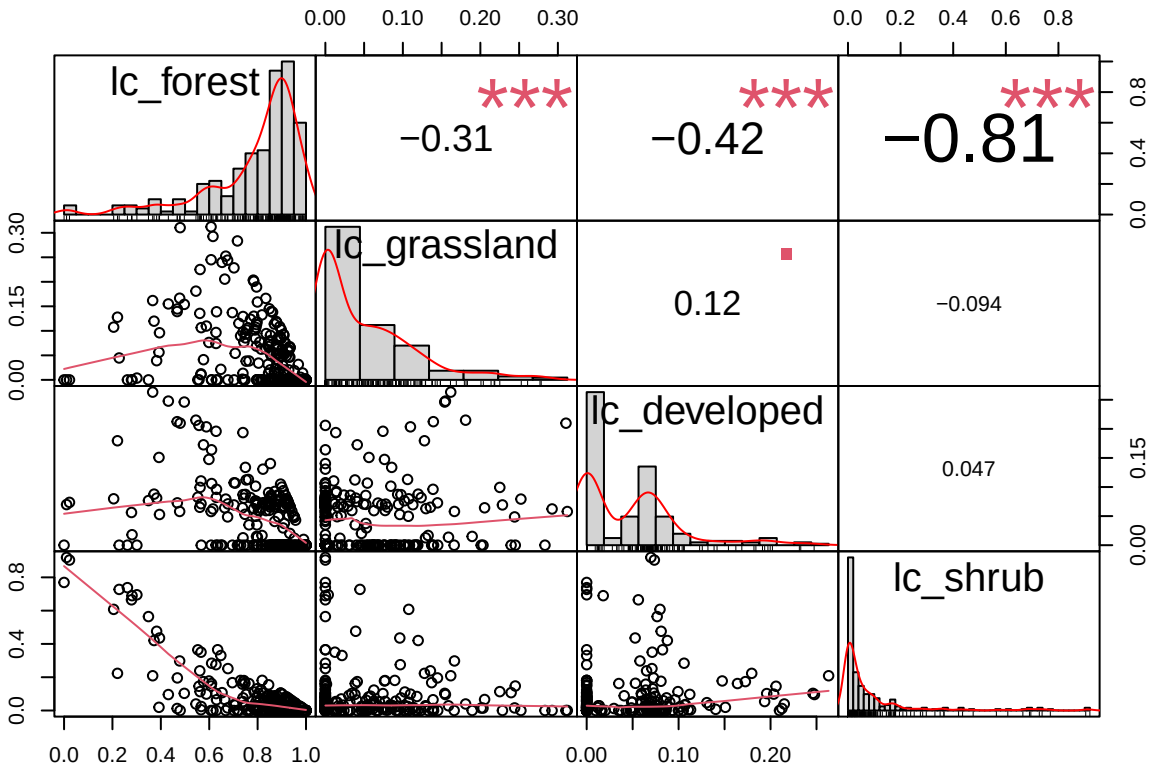
Reformat data Looks good! Now let's re-run the correlation plots

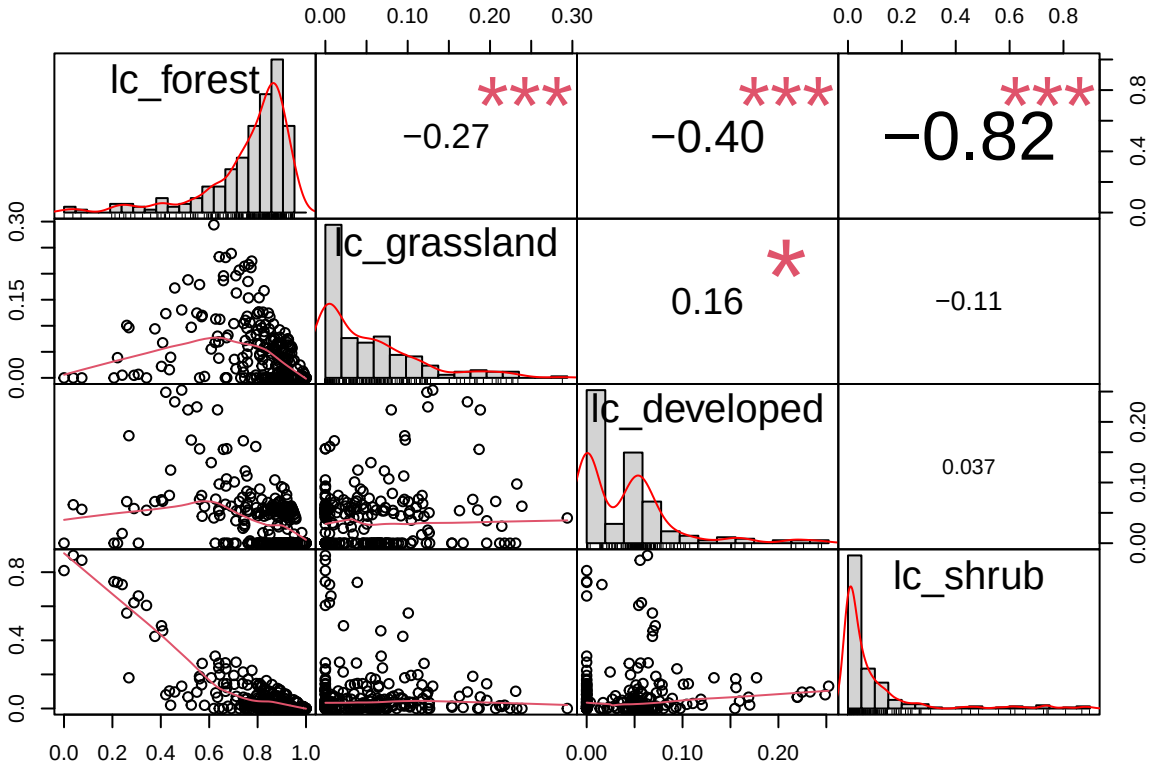
```
osm_landscape_df_2021_2022 %>%

  purrr::map(
    ~.x %>%

       # select only columns with covariates not other info
      select(lc_forest:lc_shrub) %>%

      # use chart.correlation
      chart.Correlation(.,
                        histogram = TRUE,
                        method = "pearson")
  )
```

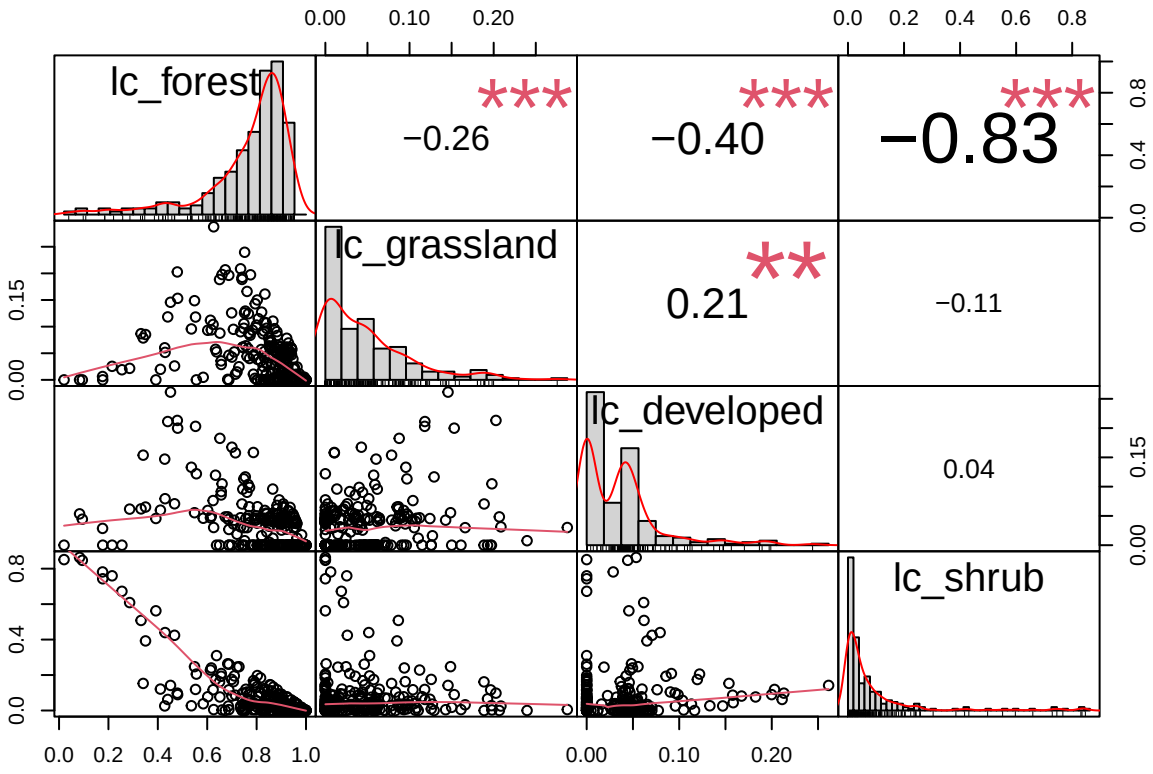


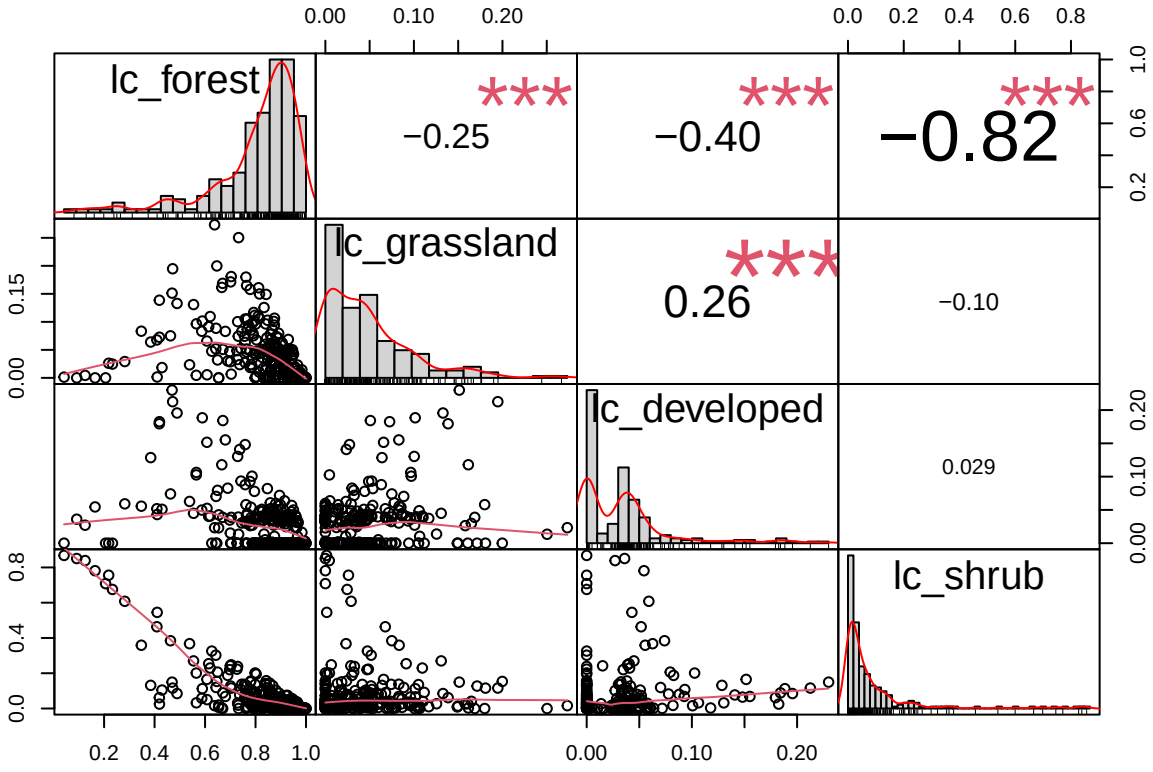
Re-run correlation plots

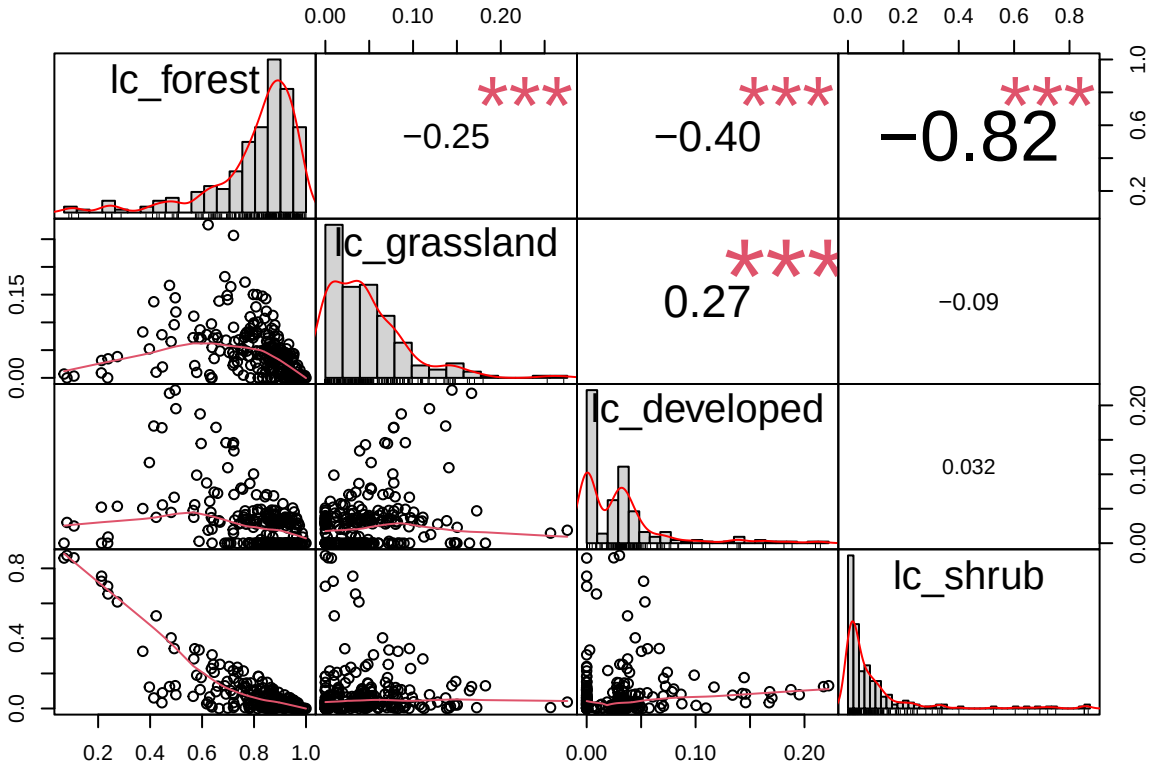


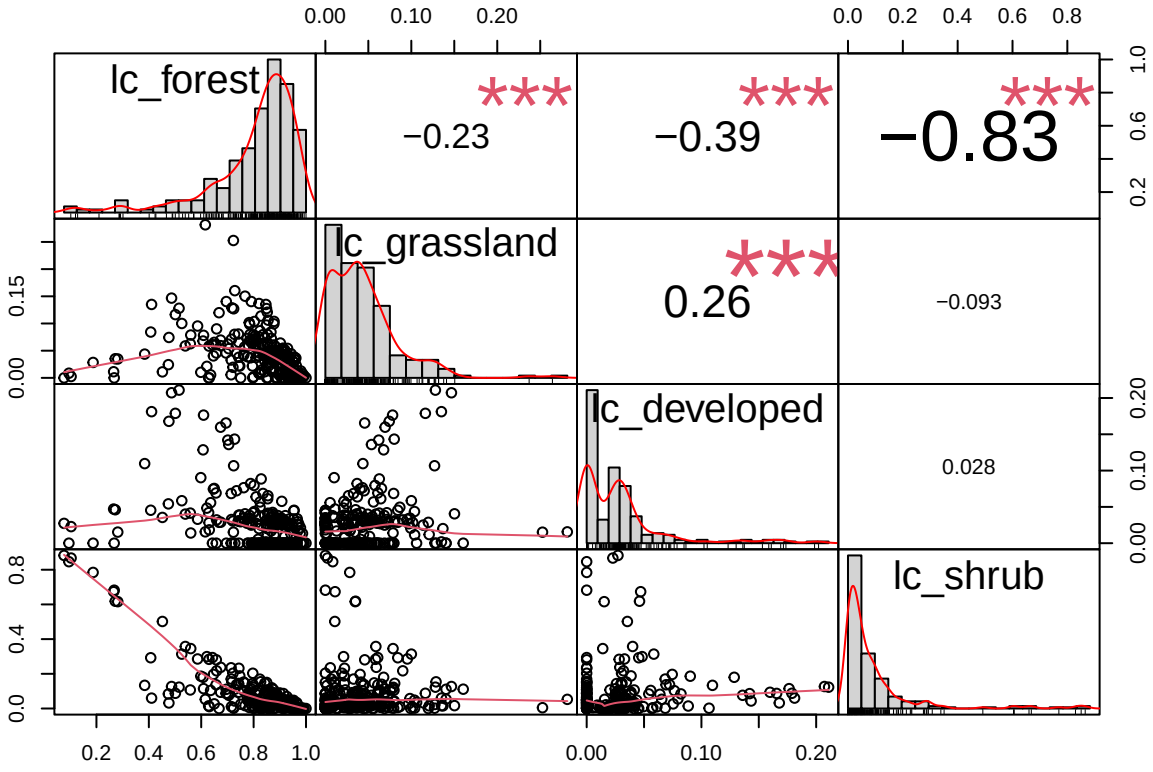

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.31
−0.42
−0.81
0.12
−0.094
0.047
***
***
***
0.00 0.10 0.20 0.30
0.0 0.2 0.4 0.6 0.8
0.00 0.15 0.30
0.0 0.4 0.8
0.0 0.2 0.4 0.6 0.8 1.0
0.00 0.10 0.20

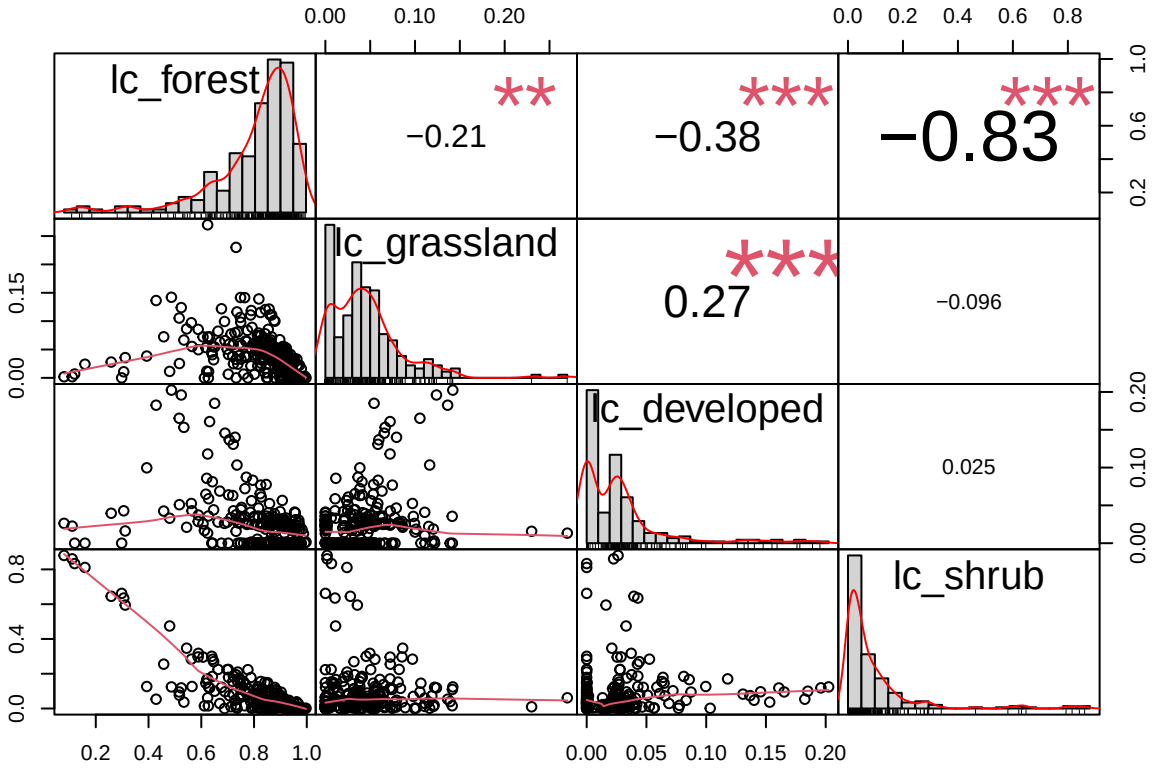

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.27
−0.40
−0.82
0.16
−0.11
0.037
***
***
***
*
0.00 0.10 0.20 0.30
0.0 0.2 0.4 0.6 0.8
0.00 0.15 0.30
0.0 0.4 0.8
0.00 0.10 0.20
0.0 0.2 0.4 0.6 0.8 1.0
0.00 0.10 0.20

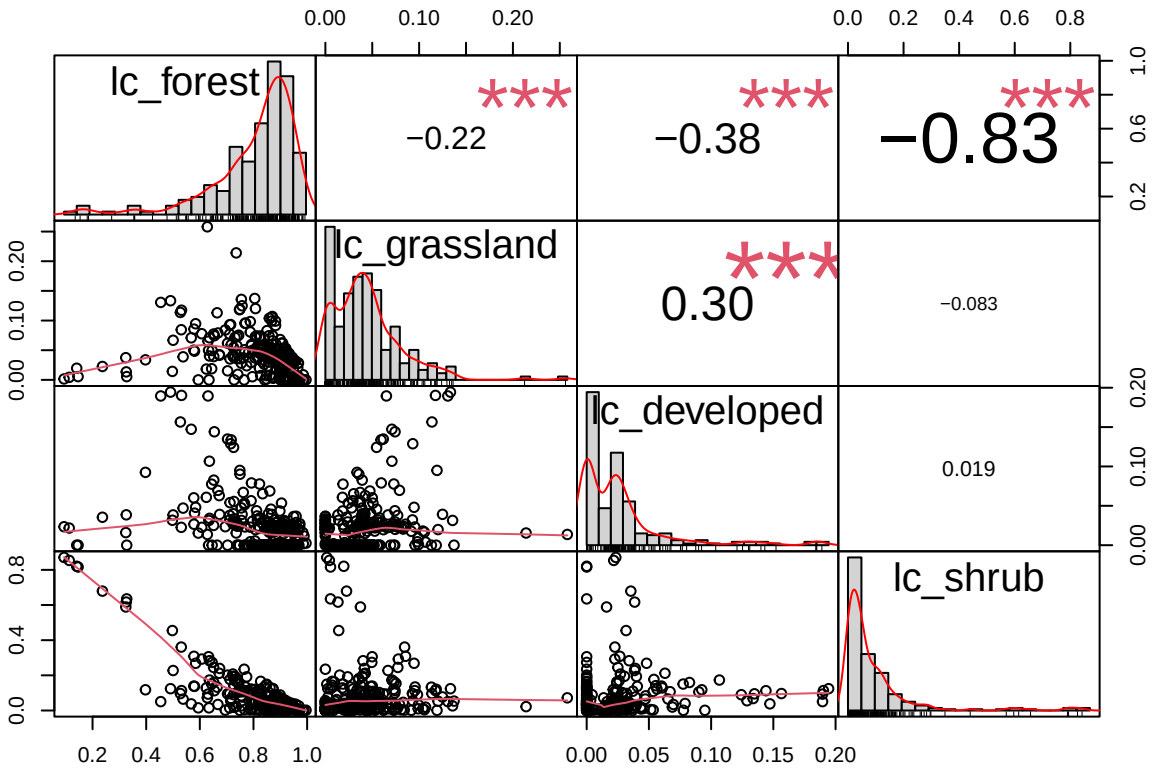

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.26
−0.40
−0.83
0.21
−0.11
0.04
***
**
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.15

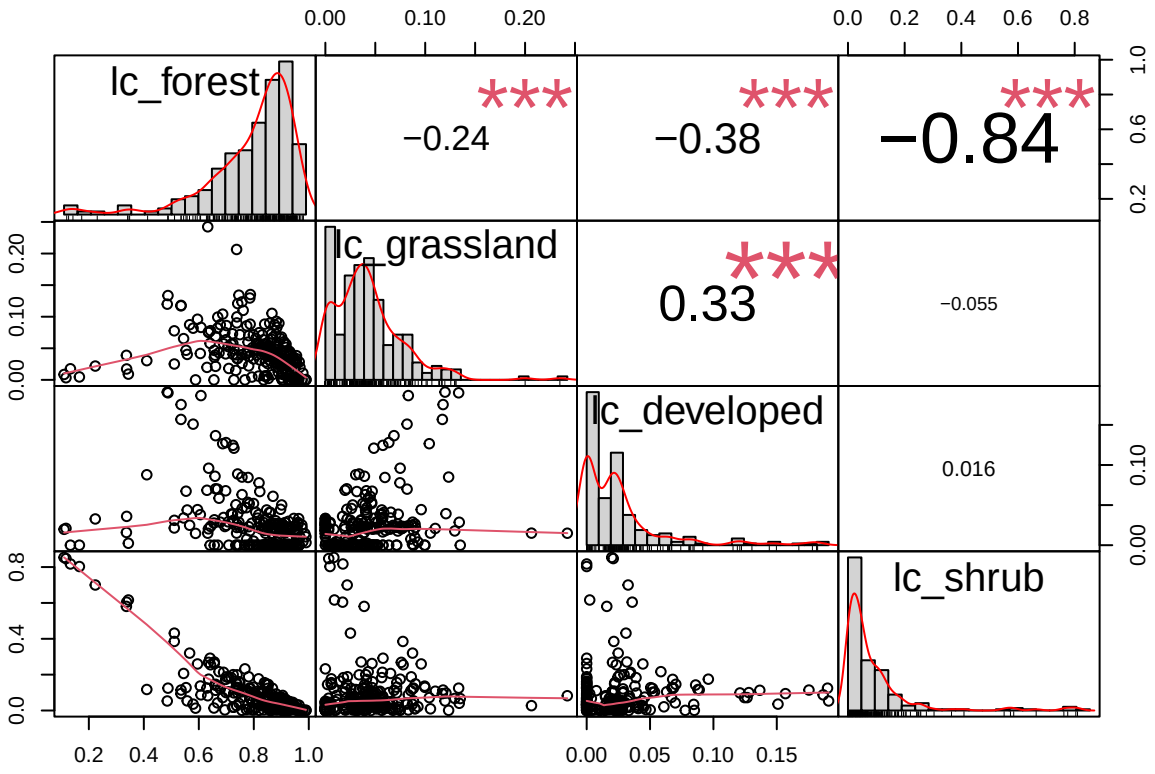

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.25
−0.40
−0.82
0.26
−0.10
0.029
***
***
***
***
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.15
0.00
0.20
0.10
0.00
0.8
0.4
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.10
0.20

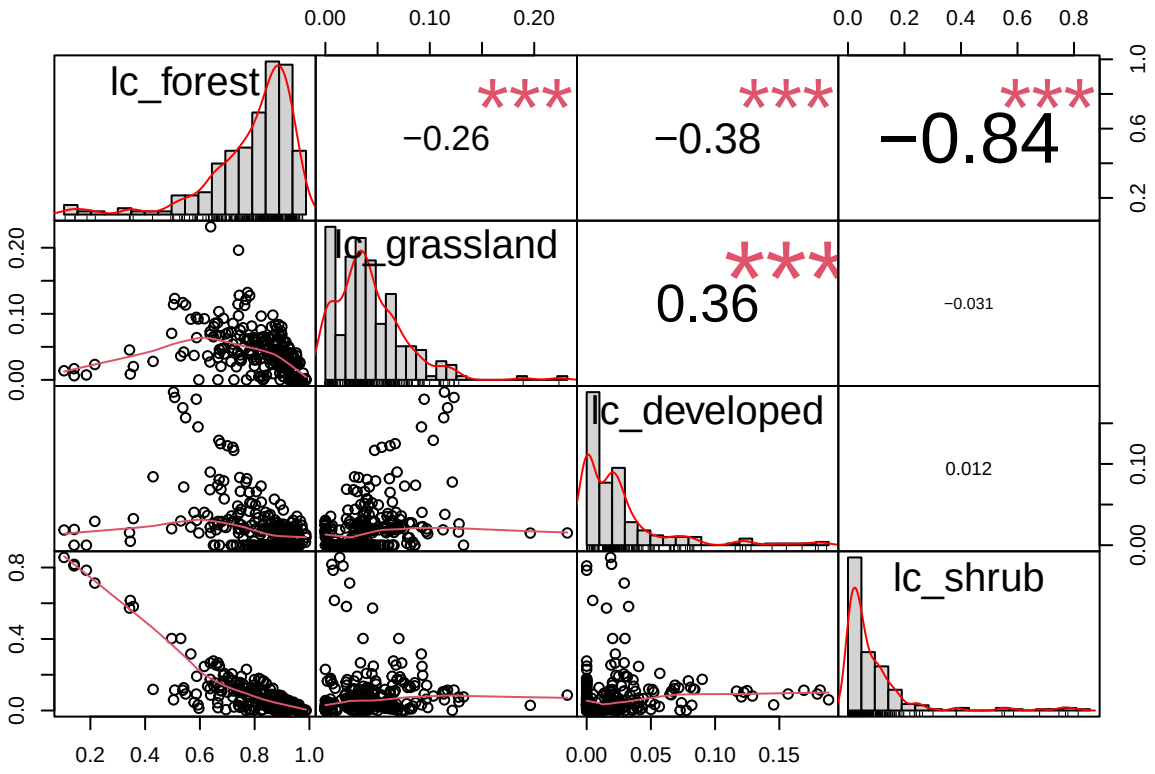

lc_forest
lc_grassland
lc_developed
lc_shrub
***
−0.25
***
−0.40
***
−0.82
***
0.27
−0.09
0.032
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.15
0.0
0.4
0.8
0.2
0.4
0.6
0.8
1.0
0.00
0.10
0.20

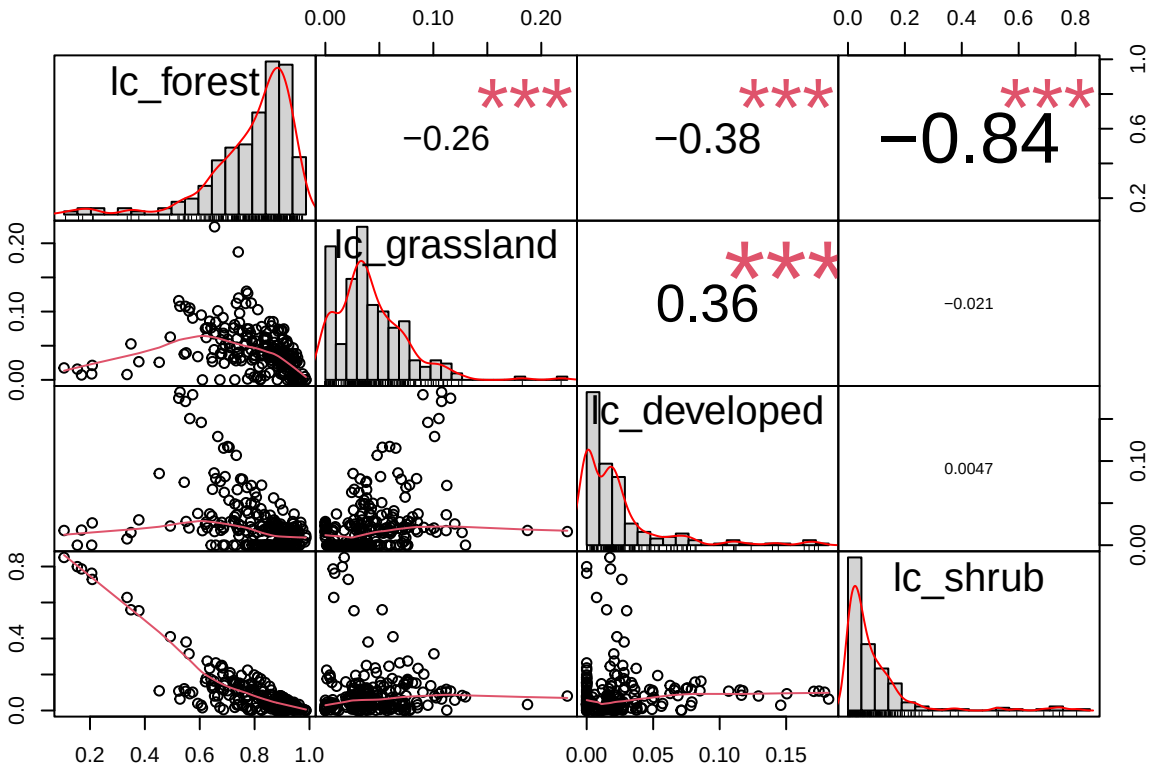

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.23
−0.39
−0.83
0.26
−0.093
0.028
***

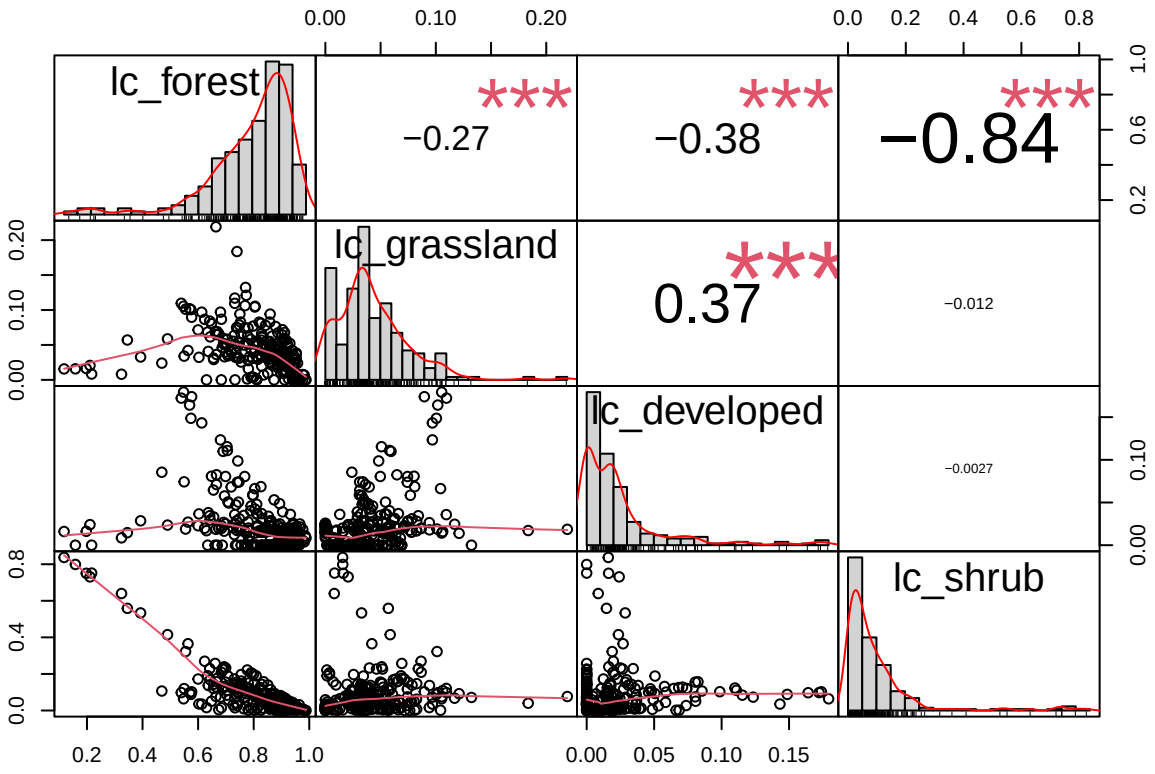

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.21
−0.38
−0.83
0.27
−0.096
0.025
**
***
***
***
0.00 0.10 0.20
0.0 0.2 0.4 0.6 0.8
0.2 0.4 0.6 0.8 1.0
0.00 0.05 0.10 0.15 0.20
0.0 0.4 0.8
0.00 0.15
0.2 0.6 1.0
0.00 0.10 0.20

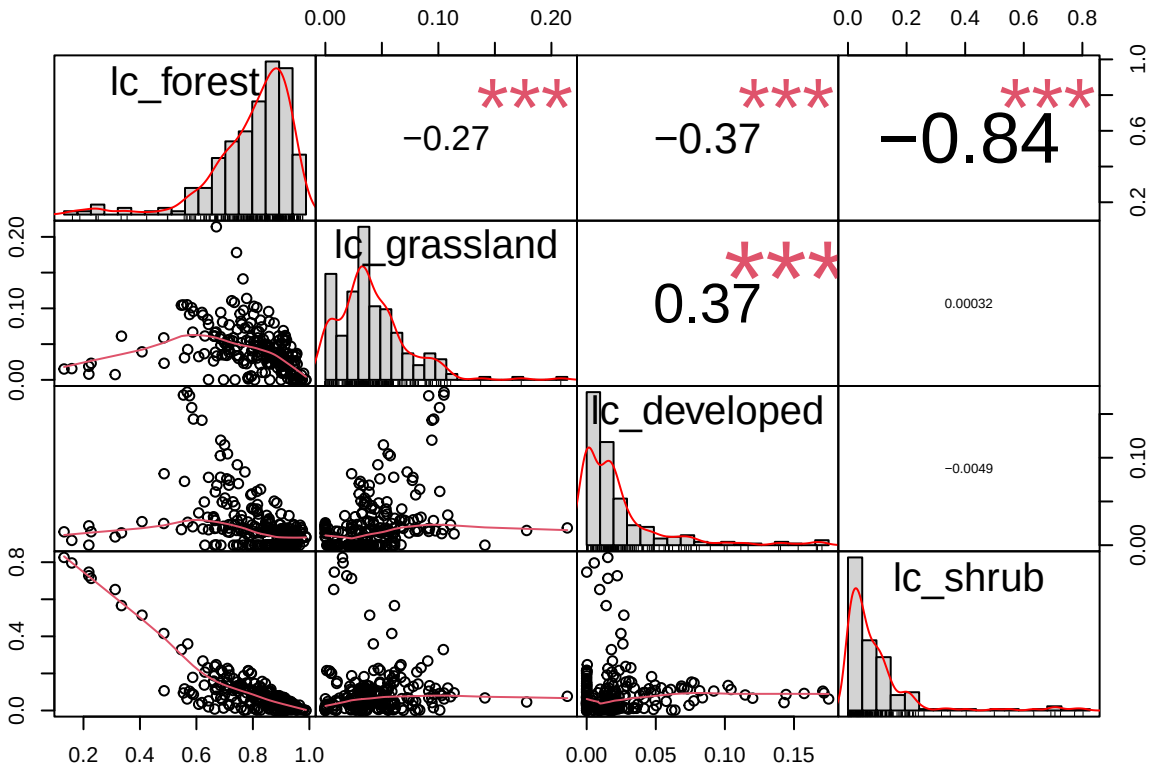

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.22
−0.38
−0.83
0.30
−0.083
0.019
***

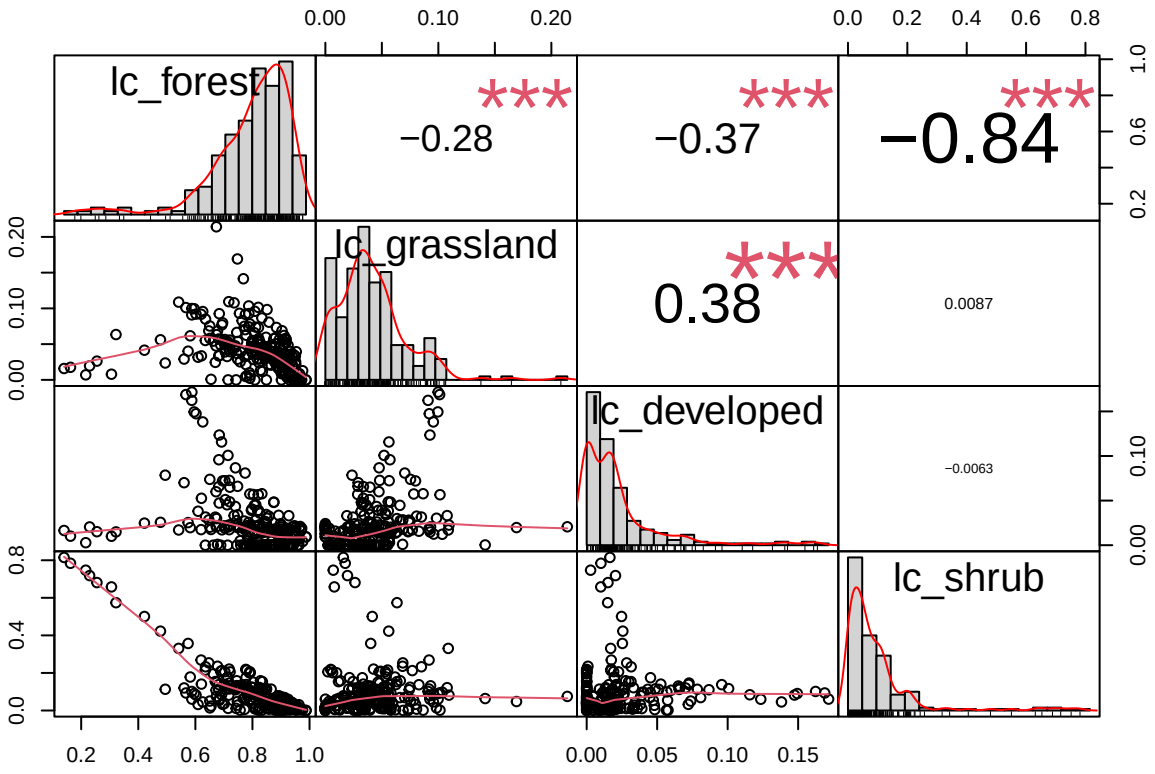

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.24
−0.38
−0.84
0.33
−0.055
0.016
***

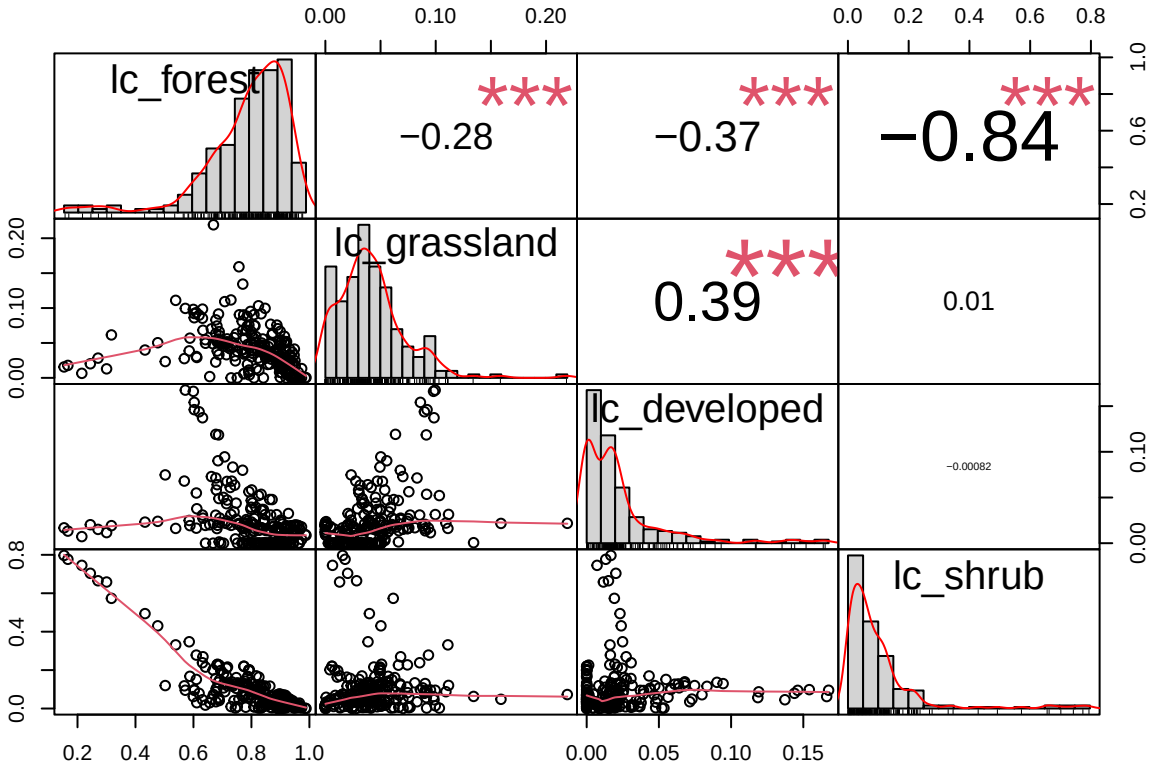

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.26
−0.38
−0.84
0.36
−0.031
0.012
***
***
***
***
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.05
0.10
0.15

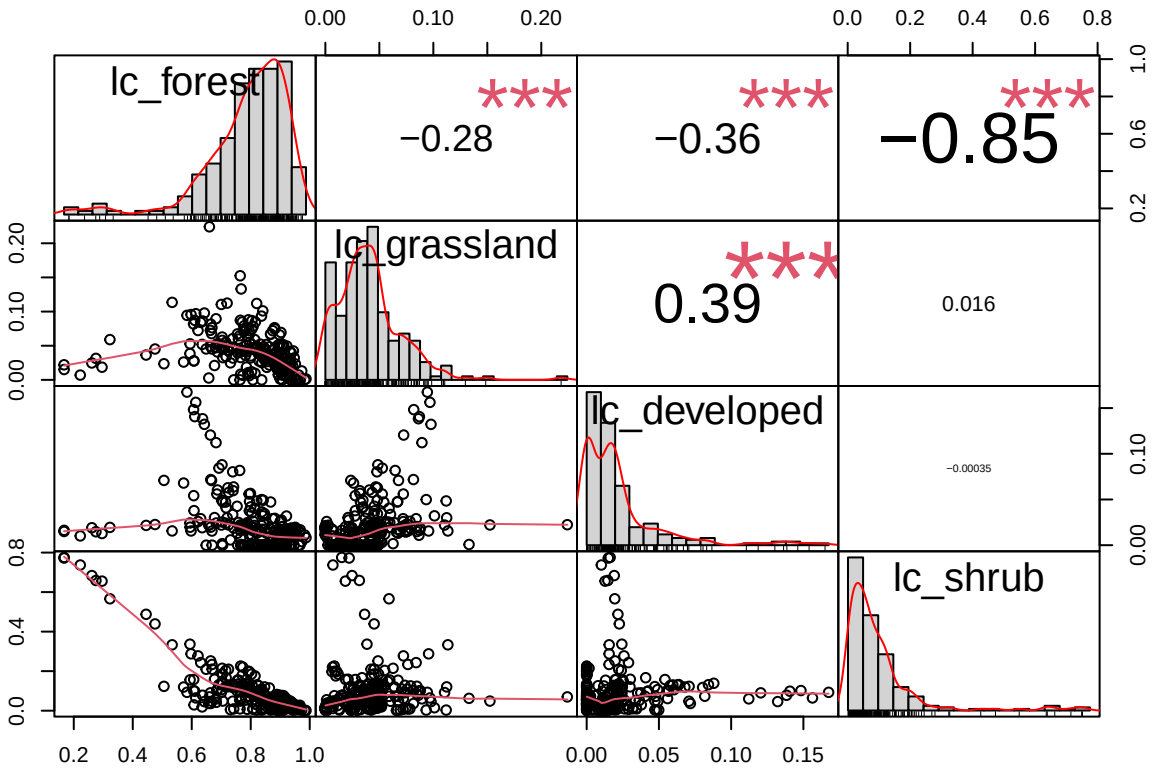

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.26
−0.38
−0.84
0.36
−0.021
0.0047
***

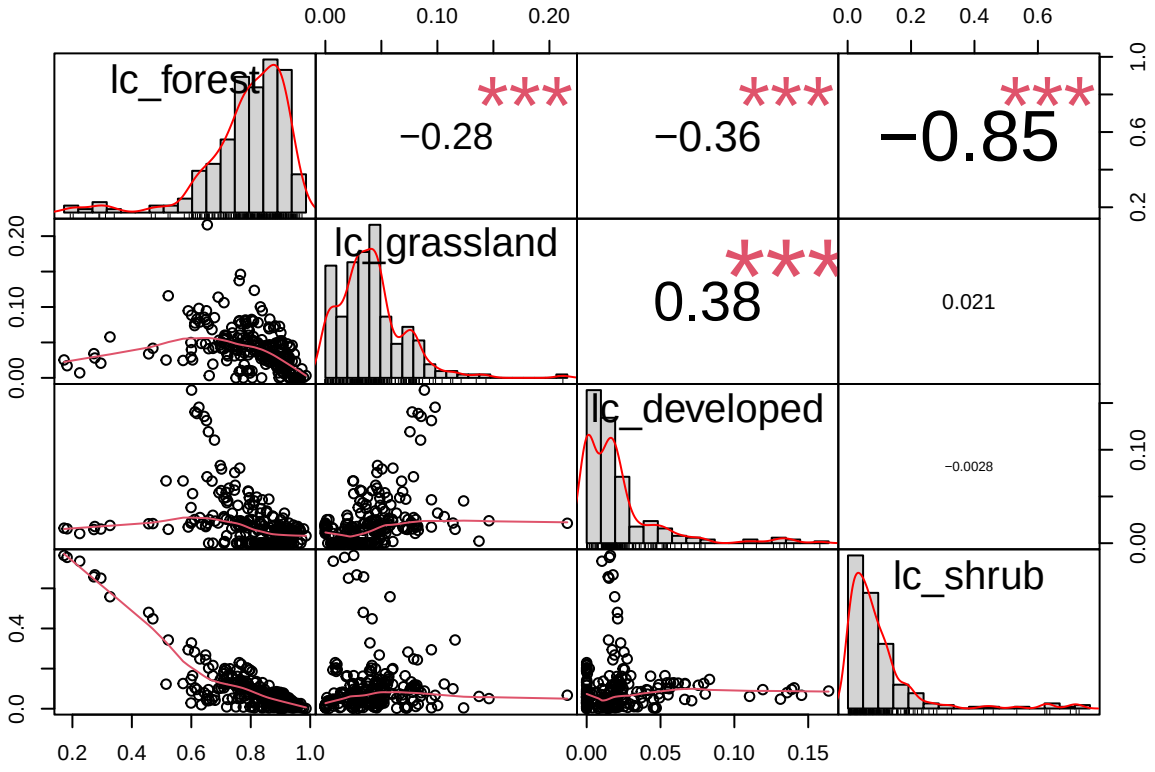

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.27
−0.38
−0.84
0.37
−0.012
−0.0027
***
***
***
***
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.2
0.4
0.6
0.8
1.0
0.00
0.05
0.10
0.15

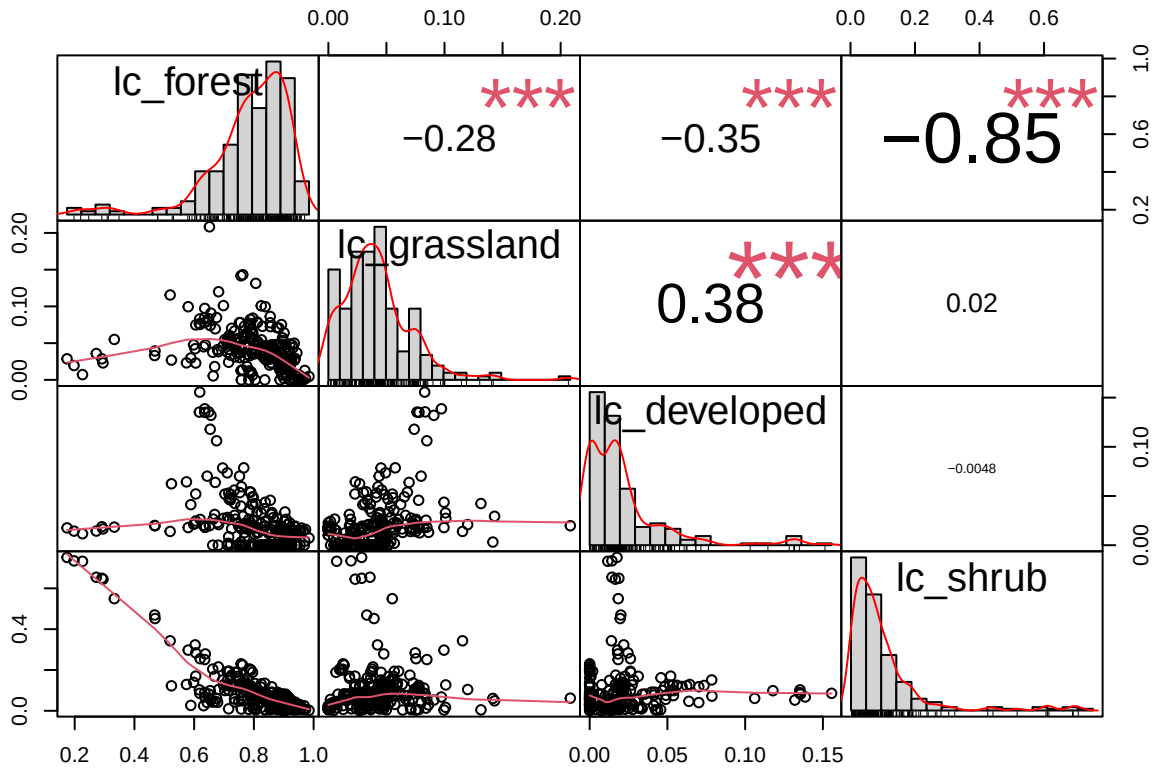

lc_forest
lc_grassland
lc_developed
lc_shrub
−0.27
−0.37
−0.84
0.37
0.00032
−0.0049
***


lc_forest
lc_grassland
lc_developed
lc_shrub
−0.28
−0.37
−0.84
0.38
0.0087
−0.0063
***
0.00
0.10
0.20
0.0
0.2
0.4
0.6
0.8
1.0
0.00
0.05
0.10
0.15


lc_forest
lc_grassland
lc_developed
lc_shrub
−0.28
−0.37
−0.84
0.39
0.01
−0.00082
***


lc_forest
lc_grassland
lc_developed
lc_shrub
−0.28
−0.36
−0.85
0.39
0.016
−0.00035
***
***
***
***


lc_forest
lc_grassland
lc_developed
lc_shrub
−0.28
−0.36
−0.85
0.38
0.021
−0.0028
***




```
## $`250 meter buffer`
## NULL
##
## $`500 meter buffer`
## NULL
##
## $`750 meter buffer`
## NULL
##
## $`1000 meter buffer`
## NULL
##
## $`1250 meter buffer`
## NULL
##
## $`1500 meter buffer`
## NULL
##
## $`1750 meter buffer`
## NULL
##
## $`2000 meter buffer`
## NULL
##
## $`2250 meter buffer`
## NULL
```

```
##
## $`2500 meter buffer`
## NULL
##
## $`2750 meter buffer`
## NULL
##
## $`3000 meter buffer`
## NULL
##
## $`3250 meter buffer`
## NULL
##
## $`3500 meter buffer`
## NULL
##
## $`3750 meter buffer`
## NULL
##
## $`4000 meter buffer`
## NULL
##
## $`4250 meter buffer`
## NULL
##
## $`4500 meter buffer`
## NULL
##
## $`4750 meter buffer`
## NULL
##
## $`5000 meter buffer`
## NULL
```

Well crap. Turns out the primary landscape types are forest and shrub so when we combine all three forest types we now have an issue with autocorrelation with the shrub landcover type. Might need to rethink the approach here.

For now will use one or the other

Subset Analysis

Now that the two separate data files are ready we can run an analysis with each, similar to what we did with the global models and see if this changes our findings from the global model or if there are differences in results between the two model subsets

Anthropogenic

We will do another analysis for each species as we did previously.

Black bear

```
bear_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
bear_anthro_model.sel <- model.sel(bear_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
bear_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  4000 meter buffer  -0.6015232           +           0.05541942
## 2  3750 meter buffer  -0.6013650           +           0.05112073
## 3  3500 meter buffer  -0.6006633           +           0.04790540
## 4  4250 meter buffer  -0.6006769           +           0.06464311
## 5  3250 meter buffer  -0.6003573           +           0.05221540
## 6   250 meter buffer  -0.5947386           +           0.01448212
## 7  4500 meter buffer  -0.6006909           +           0.06537875
## 8  4750 meter buffer  -0.6004806           +           0.06567352
## 9  3000 meter buffer  -0.5993263           +           0.05157993
## 10 2750 meter buffer  -0.5986616           +           0.05077139
## 11 5000 meter buffer  -0.5999002           +           0.06687890
## 12 2500 meter buffer  -0.5977169           +           0.04560308
## 13 1750 meter buffer  -0.5973738           +           0.04413308
## 14 2250 meter buffer  -0.5970675           +           0.04059526
## 15 2000 meter buffer  -0.5971866           +           0.04329195
```

```
## 16 1500 meter buffer  -0.5970707           +           0.04253157
## 17 1000 meter buffer  -0.5946072           +           0.02594788
## 18 1250 meter buffer  -0.5956455           +           0.03523915
## 19  500 meter buffer  -0.5940999           +           0.03240523
## 20  750 meter buffer  -0.5926719           +           0.03607808
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                -0.138073054                                -0.16132550
## 2                -0.139534500                                -0.15194464
## 3                -0.125564825                                -0.15595894
## 4                -0.132449593                                -0.15004063
## 5                -0.114284284                                -0.16171484
## 6                 0.040705860                                -0.07483023
## 7                -0.126194869                                -0.14824552
## 8                -0.121661850                                -0.14588034
## 9                -0.100714392                                -0.16372884
## 10               -0.087575474                                -0.16571955
## 11               -0.118360568                                -0.14452364
## 12               -0.074669119                                -0.15458549
## 13               -0.040286637                                -0.14611903
## 14               -0.063582502                                -0.14929281
## 15               -0.050383211                                -0.14757252
## 16               -0.027985048                                -0.13526727
## 17               -0.018972820                                -0.12706675
## 18               -0.008794519                                -0.12775540
## 19                0.011470208                                -0.10182932
## 20                0.004609023                                -0.10065339
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                    -0.08229338                -0.12772024          0.10905497
## 2                    -0.08645588                -0.11545829          0.10791338
## 3                    -0.08425655                -0.10015801          0.09872115
## 4                    -0.08576510                -0.14671796          0.10781803
## 5                    -0.08254602                -0.09434335          0.09166207
## 6                    -0.13575399                -0.01958648          0.13376102
## 7                    -0.08715346                -0.14886063          0.10723718
## 8                    -0.08945536                -0.15039805          0.10538003
## 9                    -0.08421909                -0.08098251          0.07872619
## 10                   -0.08584863                -0.08104548          0.07847708
## 11                   -0.09168571                -0.15026428          0.10128633
## 12                   -0.08322838                -0.07426624          0.08318239
## 13                   -0.09228563                -0.10733592          0.07712904
## 14                   -0.08316945                -0.07224762          0.08658669
## 15                   -0.08609534                -0.09301424          0.08296612
## 16                   -0.09866618                -0.08144812          0.07839530
## 17                   -0.11114535                -0.04636697          0.08467478
## 18                   -0.10642820                -0.05029657          0.08329980
## 19                   -0.12610340                -0.01986758          0.07753893
## 20                   -0.11981538                -0.02425698          0.05208192
##    cond(scale(wells)) df    logLik     AICc     delta      weight
## 1         0.210212826  9 -452.8033 924.4174 0.0000000 0.174063282
## 2         0.194585969  9 -452.8904 924.5917 0.1742734 0.159538017
## 3         0.174881985  9 -453.4032 925.6171 1.1996825 0.095543120
## 4         0.214459814  9 -453.5006 925.8119 1.3944910 0.086675686
## 5         0.172225184  9 -453.5363 925.8834 1.4660072 0.083631090
## 6        -0.007107041  9 -453.6497 926.1102 1.6927697 0.074666691
```

```
## 7         0.214946235  9 -453.7219 926.2546 1.8371651 0.069465927
## 8         0.211261958  9 -454.0837 926.9781 2.5607052 0.048379025
## 9         0.162990824  9 -454.2466 927.3040 2.8866041 0.041104471
## 10        0.157018861  9 -454.3500 927.5108 3.0934027 0.037066647
## 11        0.210985374  9 -454.4272 927.6652 3.2477757 0.034313233
## 12        0.130950427  9 -454.9214 928.6536 4.2361253 0.020933615
## 13        0.098847376  9 -455.1992 929.2093 4.7918653 0.015855022
## 14        0.119175813  9 -455.3339 929.4786 5.0611217 0.013857936
## 15        0.105368769  9 -455.3362 929.4833 5.0658284 0.013825362
## 16        0.082659690  9 -455.7318 930.2743 5.8569115 0.009308828
## 17        0.066083924  9 -455.9711 930.7531 6.3356474 0.007327213
## 18        0.063730888  9 -456.0551 930.9210 6.5035624 0.006737155
## 19        0.080648560  9 -456.2005 931.2118 6.7943782 0.005825412
## 20        0.083262566  9 -457.3303 933.4713 9.0538827 0.001882268
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

We will use the `model.sel()` function from the *MuMIn* package to compare the global models for each buffer width and see which buffer fits the black bear data best

```
model.sel(bear_anthro_mods)
```

```
## Model selection table
##                    cnd((Int)) dsp((Int)) cnd(scl(hrv)) cnd(scl(osm_ind))
## 4000 meter buffer    -0.6015          +       0.05542         -0.138100
## 3750 meter buffer    -0.6014          +       0.05112         -0.139500
## 3500 meter buffer    -0.6007          +       0.04791         -0.125600
## 4250 meter buffer    -0.6007          +       0.06464         -0.132400
## 3250 meter buffer    -0.6004          +       0.05222         -0.114300
## 250 meter buffer     -0.5947          +       0.01448          0.040710
## 4500 meter buffer    -0.6007          +       0.06538         -0.126200
## 4750 meter buffer    -0.6005          +       0.06567         -0.121700
## 3000 meter buffer    -0.5993          +       0.05158         -0.100700
## 2750 meter buffer    -0.5987          +       0.05077         -0.087580
```

```
## 5000 meter buffer    -0.5999          +       0.06688         -0.118400
## 2500 meter buffer    -0.5977          +       0.04560         -0.074670
## 1750 meter buffer    -0.5974          +       0.04413         -0.040290
## 2250 meter buffer    -0.5971          +       0.04060         -0.063580
## 2000 meter buffer    -0.5972          +       0.04329         -0.050380
## 1500 meter buffer    -0.5971          +       0.04253         -0.027990
## 1000 meter buffer    -0.5946          +       0.02595         -0.018970
## 1250 meter buffer    -0.5956          +       0.03524         -0.008795
## 500 meter buffer     -0.5941          +       0.03241          0.011470
## 750 meter buffer     -0.5927          +       0.03608          0.004609
##                   cnd(scl(ppl_trn_lns)) cnd(scl(ssm_lns_3D)) cnd(scl(ssm_lns))
## 4000 meter buffer              -0.16130             -0.08229          -0.12770
## 3750 meter buffer              -0.15190             -0.08646          -0.11550
## 3500 meter buffer              -0.15600             -0.08426          -0.10020
## 4250 meter buffer              -0.15000             -0.08577          -0.14670
## 3250 meter buffer              -0.16170             -0.08255          -0.09434
## 250 meter buffer               -0.07483             -0.13580          -0.01959
## 4500 meter buffer              -0.14820             -0.08715          -0.14890
## 4750 meter buffer              -0.14590             -0.08946          -0.15040
## 3000 meter buffer              -0.16370             -0.08422          -0.08098
## 2750 meter buffer              -0.16570             -0.08585          -0.08105
## 5000 meter buffer              -0.14450             -0.09169          -0.15030
## 2500 meter buffer              -0.15460             -0.08323          -0.07427
## 1750 meter buffer              -0.14610             -0.09229          -0.10730
## 2250 meter buffer              -0.14930             -0.08317          -0.07225
## 2000 meter buffer              -0.14760             -0.08610          -0.09301
## 1500 meter buffer              -0.13530             -0.09867          -0.08145
## 1000 meter buffer              -0.12710             -0.11110          -0.04637
## 1250 meter buffer              -0.12780             -0.10640          -0.05030
## 500 meter buffer               -0.10180             -0.12610          -0.01987
## 750 meter buffer               -0.10070             -0.11980          -0.02426
##                   cnd(scl(trl)) cnd(scl(wll)) df   logLik  AICc delta weight
## 4000 meter buffer       0.10910      0.210200  9 -452.803 924.4  0.00  0.174
## 3750 meter buffer       0.10790      0.194600  9 -452.890 924.6  0.17  0.160
## 3500 meter buffer       0.09872      0.174900  9 -453.403 925.6  1.20  0.096
## 4250 meter buffer       0.10780      0.214500  9 -453.501 925.8  1.39  0.087
## 3250 meter buffer       0.09166      0.172200  9 -453.536 925.9  1.47  0.084
## 250 meter buffer        0.13380     -0.007107  9 -453.650 926.1  1.69  0.075
## 4500 meter buffer       0.10720      0.214900  9 -453.722 926.3  1.84  0.069
## 4750 meter buffer       0.10540      0.211300  9 -454.084 927.0  2.56  0.048
## 3000 meter buffer       0.07873      0.163000  9 -454.247 927.3  2.89  0.041
## 2750 meter buffer       0.07848      0.157000  9 -454.350 927.5  3.09  0.037
## 5000 meter buffer       0.10130      0.211000  9 -454.427 927.7  3.25  0.034
## 2500 meter buffer       0.08318      0.131000  9 -454.921 928.7  4.24  0.021
## 1750 meter buffer       0.07713      0.098850  9 -455.199 929.2  4.79  0.016
## 2250 meter buffer       0.08659      0.119200  9 -455.334 929.5  5.06  0.014
## 2000 meter buffer       0.08297      0.105400  9 -455.336 929.5  5.07  0.014
## 1500 meter buffer       0.07840      0.082660  9 -455.732 930.3  5.86  0.009
## 1000 meter buffer       0.08467      0.066080  9 -455.971 930.8  6.34  0.007
## 1250 meter buffer       0.08330      0.063730  9 -456.055 930.9  6.50  0.007
## 500 meter buffer        0.07754      0.080650  9 -456.200 931.2  6.79  0.006
## 750 meter buffer        0.05208      0.083260  9 -457.330 933.5  9.05  0.002
## Models ranked by AICc(x)
## Random terms (all models):
```

```
##   cond(1 | array)
```

This is quite a bit different from the global results which indicated that the 250m buffer was best fit for black bears, whereas this one it is the 4000m buffer

```
summary(bear_anthro_mods$`4000 meter buffer`)
```

Top model summary

```
##  Family: binomial  ( logit )
## Formula:
## cbind(black_bear, absent_black_bear) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    923.6    954.6   -452.8    905.6      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.003319 0.05762
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -0.60152    0.05278 -11.398   <2e-16 ***
## scale(harvest)                    0.05542    0.06258   0.886   0.3759
## scale(seismic_lines)             -0.12772    0.08033  -1.590   0.1119
## scale(seismic_lines_3D)          -0.08229    0.06340  -1.298   0.1943
## scale(trails)                     0.10905    0.05301   2.057   0.0397 *
## scale(wells)                      0.21021    0.09903   2.123   0.0338 *
## scale(osm_industrial)            -0.13807    0.06539  -2.112   0.0347 *
## scale(pipeline_transmission_lines) -0.16133    0.07616  -2.118   0.0342 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Quck test on scaling Going to run the models again WITHOUT scaling variables as recent literature suggests this may influence model results.

```
test_bear_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
```

```
    glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                         # HFI that aren't correlated
                         harvest +
                         seismic_lines +
                         seismic_lines_3D +
                         trails +
                         wells +
                         osm_industrial +
                         pipeline_transmission_lines +

                         # Random effect of array
                         (1|array),
                     data = .,
                     family = 'binomial'))
```

```
## Warning in finalizeTMB(TMBStruc, obj, fit, h, data.tmb.old): Model convergence
## problem; non-positive-definite Hessian matrix. See vignette('troubleshooting')
```

```
model.sel(test_bear_anthro_mods)
```

```
## Model selection table
##                    cnd((Int)) dsp((Int)) cnd(hrv) cnd(osm_ind) cnd(ppl_trn_lns)
## 3750 meter buffer    -0.3935          +   0.7610     -4.06200           -8.432
## 3500 meter buffer    -0.4274          +   0.5648     -3.46600           -8.004
## 4250 meter buffer    -0.3758          +   0.8226     -3.83500           -8.574
## 3250 meter buffer    -0.4354          +   0.5984     -3.08000           -8.013
## 250 meter buffer     -0.5559          +   0.0956      0.40430           -1.096
## 4500 meter buffer    -0.3724          +   0.8539     -3.66900           -8.731
## 4750 meter buffer    -0.3672          +   0.8794     -3.55700           -8.858
## 3000 meter buffer    -0.4453          +   0.5780     -2.64100           -7.755
## 2750 meter buffer    -0.4431          +   0.5598     -2.27400           -7.557
## 5000 meter buffer    -0.3637          +   0.9151     -3.47500           -9.010
## 2500 meter buffer    -0.4561          +   0.4954     -1.96300           -6.772
## 1750 meter buffer    -0.4044          +   0.4531     -1.05300           -5.135
## 2250 meter buffer    -0.4645          +   0.4344     -1.67800           -6.143
## 2000 meter buffer    -0.4287          +   0.4542     -1.32400           -5.640
## 1500 meter buffer    -0.4506          +   0.4292     -0.70590           -4.310
## 1000 meter buffer    -0.4979          +   0.2331     -0.40810           -3.166
## 1250 meter buffer    -0.4983          +   0.3435     -0.21260           -3.680
## 500 meter buffer     -0.5637          +   0.2399      0.15480           -1.773
## 750 meter buffer     -0.5451          +   0.2963      0.07942           -2.118
## 4000 meter buffer    -0.3865          +   0.7698     -4.03600           -9.096
##                    cnd(ssm_lns) cnd(ssm_lns_3D) cnd(trl) cnd(wll) df   logLik
## 3750 meter buffer      -37.410          -7.630    80.15  12.3800  9 -452.958
## 3500 meter buffer      -27.100          -7.226    71.71   9.5940  9 -453.403
## 4250 meter buffer      -41.040          -7.557    82.40  12.5900  9 -453.501
## 3250 meter buffer      -25.080          -7.046    65.78   9.2670  9 -453.536
## 250 meter buffer        -2.274         -10.600    21.61  -0.1549  9 -453.650
## 4500 meter buffer      -42.150          -7.784    83.98  12.9400  9 -453.722
## 4750 meter buffer      -42.870          -8.093    84.71  13.1000  9 -454.084
## 3000 meter buffer      -21.300          -7.146    55.37   8.5810  9 -454.247
```

```
## 2750 meter buffer      -21.250          -7.255    53.48   8.0170  9 -454.350
## 5000 meter buffer      -43.150          -8.404    83.25  13.3700  9 -454.427
## 2500 meter buffer      -19.300          -7.010    54.05   6.4690  9 -454.921
## 1750 meter buffer      -26.160          -7.751    41.56   4.4000  9 -455.199
## 2250 meter buffer      -18.560          -6.970    54.29   5.7130  9 -455.334
## 2000 meter buffer      -23.350          -7.220    48.93   4.8850  9 -455.336
## 1500 meter buffer      -19.110          -8.372    37.74   3.5080  9 -455.732
## 1000 meter buffer      -10.280          -9.509    33.28   2.4940  9 -455.971
## 1250 meter buffer      -11.420          -9.124    36.02   2.6370  9 -456.055
## 500 meter buffer        -3.477         -11.040    20.00   2.3810  9 -456.200
## 750 meter buffer        -4.862         -10.420    17.61   2.8880  9 -457.330
## 4000 meter buffer      -38.440          -7.300    81.98  12.9300  9
##                     AICc delta
## 3750 meter buffer 924.7  0.00
## 3500 meter buffer 925.6  0.89
## 4250 meter buffer 925.8  1.09
## 3250 meter buffer 925.9  1.16
## 250 meter buffer  926.1  1.38
## 4500 meter buffer 926.3  1.53
## 4750 meter buffer 927.0  2.25
## 3000 meter buffer 927.3  2.58
## 2750 meter buffer 927.5  2.78
## 5000 meter buffer 927.7  2.94
## 2500 meter buffer 928.7  3.93
## 1750 meter buffer 929.2  4.48
## 2250 meter buffer 929.5  4.75
## 2000 meter buffer 929.5  4.76
## 1500 meter buffer 930.3  5.55
## 1000 meter buffer 930.8  6.03
## 1250 meter buffer 930.9  6.19
## 500 meter buffer  931.2  6.49
## 750 meter buffer  933.5  8.74
## 4000 meter buffer
## Models ranked by AICc(x)
## Random terms (all models):
##   cond(1 | array)
```

Interesting, you get almost the same results but the 4000m buffer is the worst fit. There was a warning about model convergence issues which I suspect is the problem, this is fairly common with unscaled data. We will proceed with scaling the data as is standard practice still, but consider in discussion on ms how this may affect results.

Caribou

```
caribou_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(caribou, absent_caribou) ~
```

```
                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))
```

```
## Warning in finalizeTMB(TMBStruc, obj, fit, h, data.tmb.old): Model convergence
## problem; singular convergence (7). See vignette('troubleshooting'),
## help('diagnose')
```

```
# run model selection and save the results as a tibble for graphing use later
caribou_anthro_model.sel <- model.sel(caribou_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
caribou_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  1000 meter buffer   -5.548202           +           -0.7434272
## 2  1250 meter buffer   -5.414419           +           -0.3827273
## 3   750 meter buffer   -5.861136           +           -1.8466582
## 4  1500 meter buffer   -5.316414           +           -0.1687568
## 5  2000 meter buffer   -5.302913           +           -0.1953843
## 6  2500 meter buffer   -5.357304           +           -0.1833198
## 7  2750 meter buffer   -5.356840           +           -0.1653046
## 8   500 meter buffer   -5.731118           +           -2.2269204
## 9  1750 meter buffer   -5.267118           +           -0.1693908
## 10 2250 meter buffer   -5.286203           +           -0.1932869
## 11 3000 meter buffer   -5.325367           +           -0.1264651
## 12 3250 meter buffer   -5.269542           +           -0.1375689
## 13 3500 meter buffer   -5.222246           +           -0.2107347
## 14  250 meter buffer -205.478643           +         -603.1262423
## 15 3750 meter buffer   -5.176264           +           -0.3112598
## 16 5000 meter buffer   -5.112732           +           -1.1953009
## 17 4750 meter buffer   -5.094802           +           -0.9945439
## 18 4000 meter buffer   -5.124358           +           -0.4164396
## 19 4250 meter buffer   -5.081197           +           -0.5981531
## 20 4500 meter buffer   -5.069085           +           -0.8296411
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
```

```
## 1                  -1.28956739                                -0.3131129
## 2                  -1.25756529                                -0.3097483
## 3                  -1.09329936                                -0.2940728
## 4                  -1.08717216                                -0.4410655
## 5                  -1.32940469                                -0.5624233
## 6                  -0.98599062                                -0.9072482
## 7                  -0.65443205                                -1.0623980
## 8                  -0.35599956                                -0.2392163
## 9                  -1.20635579                                -0.4705007
## 10                 -1.17561085                                -0.6466125
## 11                 -0.52741655                                -1.1040122
## 12                 -0.45679026                                -1.0464903
## 13                 -0.35285436                                -1.0622802
## 14                 -0.26819231                                -0.4221266
## 15                 -0.29414398                                -1.0294607
## 16                  0.02831963                                -0.9535882
## 17                  0.01072871                                -0.9732911
## 18                 -0.21849650                                -0.9482116
## 19                 -0.13794582                                -0.9275871
## 20                 -0.06007709                                -0.9202266
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                    -0.18749404                 0.15459501          0.03898567
## 2                    -0.19535612                 0.21040880          0.02258331
## 3                    -0.12132902                 0.22122254          0.06621093
## 4                    -0.21181586                 0.14586261         -0.12468209
## 5                    -0.21414456                 0.14382342         -0.15185239
## 6                    -0.20674375                 0.15923967         -0.25533265
## 7                    -0.18116466                 0.10498796         -0.30340030
## 8                    -0.12408563                 0.20969474         -0.03229518
## 9                    -0.19671517                 0.09651472         -0.13096070
## 10                   -0.22651171                 0.17175204         -0.17320021
## 11                   -0.17061300                 0.02473656         -0.39107391
## 12                   -0.18673037                -0.03486376         -0.44815026
## 13                   -0.13735234                -0.04570776         -0.48144176
## 14                   -0.17968245                 0.21911311         -0.26721143
## 15                   -0.10411236                -0.08400064         -0.45833715
## 16                    0.03660485                -0.03071875         -0.54129254
## 17                    0.01761321                -0.04810542         -0.53944909
## 18                   -0.07631351                -0.10206367         -0.40605803
## 19                   -0.02956579                -0.08693773         -0.42725566
## 20                    0.00300806                -0.07011857         -0.45824025
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1           0.6921299  9 -133.0946 285.0001  0.000000 8.840982e-01
## 2           0.7364424  9 -135.6750 290.1607  5.160649 6.696993e-02
## 3           0.6323499  9 -136.4419 291.6947  6.694596 3.110194e-02
## 4           0.8065110  9 -138.0477 294.9062  9.906126 6.243276e-03
## 5           0.9663138  9 -139.0421 296.8950 11.894887 2.309717e-03
## 6           1.2408913  9 -139.1457 297.1021 12.102056 2.082440e-03
## 7           1.2757101  9 -139.2793 297.3695 12.369416 1.821864e-03
## 8           0.5188765  9 -139.3748 297.5605 12.560392 1.655946e-03
## 9           0.8359218  9 -139.4659 297.7427 12.742568 1.511775e-03
## 10          1.0391623  9 -139.9962 298.8032 13.803158 8.895756e-04
## 11          1.2519807  9 -140.1523 299.1154 14.115343 7.610141e-04
## 12          1.1743808  9 -141.5352 301.8812 16.881131 1.909015e-04
```

```
## 13          1.1335767  9 -142.0319 302.8746 17.874512 1.161715e-04
## 14          0.3501608  9 -142.1019 303.0146 18.014554 1.083153e-04
## 15          1.0852318  9 -142.6347 304.0802 19.080079 6.357905e-05
## 16          0.8192121  9 -143.7337 306.2782 21.278124 2.118433e-05
## 17          0.8414263  9 -143.8595 306.5297 21.529651 1.868083e-05
## 18          0.9949019  9 -143.9159 306.6426 21.642557 1.765546e-05
## 19          0.9080571  9 -144.5263 307.8633 22.863238 9.589840e-06
## 20          0.8303706  9 -144.6750 308.1608 23.160747 8.264341e-06
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

Let's look at the summary for the top model

```
summary(caribou_anthro_mods$`1000 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(caribou, absent_caribou) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    284.2    315.2   -133.1    266.2      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 4.648    2.156
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
```

```
##                                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -5.54820    1.05969  -5.236 1.64e-07 ***
## scale(harvest)                    -0.74343    0.66903  -1.111   0.2665
## scale(seismic_lines)               0.15460    0.16295   0.949   0.3428
## scale(seismic_lines_3D)           -0.18749    0.19623  -0.955   0.3393
## scale(trails)                      0.03899    0.20948   0.186   0.8524
## scale(wells)                       0.69213    0.15941   4.342 1.41e-05 ***
## scale(osm_industrial)             -1.28957    0.45284  -2.848   0.0044 **
## scale(pipeline_transmission_lines) -0.31311    0.28058  -1.116   0.2645
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Coyote

```
coyote_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(coyote, absent_coyote) ~

                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
coyote_anthro_model.sel <- model.sel(coyote_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
coyote_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  4750 meter buffer   -1.392504           +         -0.168455867
## 2  5000 meter buffer   -1.394096           +         -0.178552842
```

```
## 3  4500 meter buffer   -1.390544           +         -0.149430735
## 4  3000 meter buffer   -1.381542           +         -0.073710735
## 5  2750 meter buffer   -1.380048           +         -0.067891675
## 6  4250 meter buffer   -1.388380           +         -0.132438631
## 7  3250 meter buffer   -1.382392           +         -0.082540861
## 8  2250 meter buffer   -1.378863           +         -0.049801658
## 9  2500 meter buffer   -1.379342           +         -0.057112417
## 10 4000 meter buffer   -1.386264           +         -0.124148068
## 11 2000 meter buffer   -1.378030           +         -0.034052913
## 12 1750 meter buffer   -1.379266           +         -0.017100267
## 13 3500 meter buffer   -1.383006           +         -0.099848855
## 14 3750 meter buffer   -1.384199           +         -0.110751481
## 15 1500 meter buffer   -1.381316           +         -0.004779912
## 16 1250 meter buffer   -1.382910           +         -0.009979531
## 17  750 meter buffer   -1.387961           +          0.019511213
## 18 1000 meter buffer   -1.385038           +         -0.013772602
## 19  500 meter buffer   -1.387937           +          0.008353305
## 20  250 meter buffer   -1.376811           +         -0.091722768
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                    0.4460095                              0.0492885780
## 2                    0.4403746                              0.0554699985
## 3                    0.4497606                              0.0288879296
## 4                    0.4563655                             -0.0458324383
## 5                    0.4562499                             -0.0360959773
## 6                    0.4488958                              0.0122997745
## 7                    0.4497229                             -0.0367862183
## 8                    0.4118956                              0.0258917111
## 9                    0.4265502                              0.0004534265
## 10                   0.4384295                             -0.0152152021
## 11                   0.4143112                              0.0341437667
## 12                   0.4133428                              0.0524894279
## 13                   0.4311573                             -0.0254287541
## 14                   0.4269129                             -0.0200063062
## 15                   0.4113818                              0.0590109789
## 16                   0.3993337                              0.0761563708
## 17                   0.3722183                              0.1323948034
## 18                   0.3735062                              0.1117975234
## 19                   0.3456310                              0.0713110747
## 20                   0.2663284                              0.1284211047
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                    -0.05030671                 0.1597048        0.021534287
## 2                    -0.04633720                 0.1629008        0.030254427
## 3                    -0.04988250                 0.1557877        0.011283848
## 4                    -0.05317002                 0.1214172        0.020233072
## 5                    -0.06249270                 0.1267952        0.018038495
## 6                    -0.05041775                 0.1395795        0.004724674
## 7                    -0.04979732                 0.1167598        0.013729476
## 8                    -0.10057216                 0.1737527        0.058867307
## 9                    -0.07836984                 0.1647012        0.040199115
## 10                   -0.04670352                 0.1303833        0.009784945
## 11                   -0.12410064                 0.1496142        0.050715263
## 12                   -0.14991651                 0.1417368        0.049734431
## 13                   -0.04930477                 0.1189339        0.021716258
## 14                   -0.04822736                 0.1191147        0.020433831
```

```
## 15                   -0.17358676                  0.1518348         0.038895595
## 16                   -0.18776428                  0.1577051         0.018111061
## 17                   -0.15778008                  0.1663454        -0.073215761
## 18                   -0.16523423                  0.1549719         0.002132869
## 19                   -0.12698933                  0.1540626        -0.113309983
## 20                   -0.05136722                  0.1033641         0.008134052
##    cond(scale(wells)) df    logLik      AICc     delta       weight
## 1          0.15121332  9 -478.8307  976.4721  0.000000 2.677073e-01
## 2          0.15506108  9 -478.9603  976.7314  0.259249 2.351609e-01
## 3          0.16170342  9 -479.3838  977.5784  1.106249 1.539719e-01
## 4          0.22806529  9 -479.6742  978.1592  1.687030 1.151666e-01
## 5          0.22267665  9 -480.4034  979.6175  3.145408 5.554482e-02
## 6          0.16963788  9 -480.4602  979.7312  3.259061 5.247642e-02
## 7          0.21790659  9 -480.6905  980.1918  3.719705 4.168086e-02
## 8          0.24090134  9 -481.6381  982.0870  5.614863 1.615877e-02
## 9          0.23514655  9 -481.6522  982.1152  5.643089 1.593232e-02
## 10         0.19206688  9 -481.8575  982.5257  6.053585 1.297601e-02
## 11         0.23595126  9 -482.0702  982.9512  6.479104 1.048916e-02
## 12         0.23854223  9 -482.3840  983.5788  7.106717 7.664015e-03
## 13         0.21732776  9 -482.4476  983.7060  7.233908 7.191793e-03
## 14         0.21147612  9 -482.9246  984.6600  8.187889 4.463575e-03
## 15         0.25465521  9 -483.2020  985.2148  8.742685 3.382287e-03
## 16         0.23525786  9 -487.8267  994.4643 17.992175 3.316723e-05
## 17         0.28845742  9 -493.6053 1006.0213 29.549198 1.025967e-07
## 18         0.24339094  9 -493.9776 1006.7660 30.293868 7.070178e-08
## 19         0.24886776  9 -499.5576 1017.9260 41.453864 2.667277e-10
## 20         0.03331044  9 -516.5012 1051.8131 75.340993 1.168346e-17
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(coyote_anthro_mods$`4750 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(coyote, absent_coyote) ~ scale(harvest) + scale(seismic_lines) +
```

```
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    975.7   1006.7   -478.8    957.7      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.3289   0.5735
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -1.39250    0.24038  -5.793 6.92e-09 ***
## scale(harvest)                   -0.16846    0.09498  -1.774   0.0761 .
## scale(seismic_lines)              0.15970    0.07739   2.064   0.0391 *
## scale(seismic_lines_3D)          -0.05031    0.07036  -0.715   0.4746
## scale(trails)                     0.02153    0.06545   0.329   0.7421
## scale(wells)                      0.15121    0.08582   1.762   0.0781 .
## scale(osm_industrial)             0.44601    0.06039   7.386 1.51e-13 ***
## scale(pipeline_transmission_lines)  0.04929    0.08745   0.564   0.5730
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fisher

```
fisher_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(fisher, absent_fisher) ~

                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))
```

```r
# run model selection and save the results as a tibble for graphing use later
fisher_anthro_model.sel <- model.sel(fisher_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
fisher_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1   250 meter buffer   -2.970098           +          -0.37281000
## 2   500 meter buffer   -2.968812           +          -0.32480059
## 3  2500 meter buffer   -2.951236           +          -0.06195582
## 4   750 meter buffer   -2.953649           +          -0.25235733
## 5  2750 meter buffer   -2.947957           +          -0.05096300
## 6  3000 meter buffer   -2.942918           +          -0.04421143
## 7  3250 meter buffer   -2.940482           +          -0.06051081
## 8  1000 meter buffer   -2.956293           +          -0.24611026
## 9  1250 meter buffer   -2.959535           +          -0.26157041
## 10 4500 meter buffer   -2.953143           +          -0.14596094
## 11 4750 meter buffer   -2.957786           +          -0.17650433
## 12 5000 meter buffer   -2.960696           +          -0.20226078
## 13 4250 meter buffer   -2.947593           +          -0.12067896
## 14 4000 meter buffer   -2.942035           +          -0.10750579
## 15 2250 meter buffer   -2.946490           +          -0.09139343
## 16 3500 meter buffer   -2.937845           +          -0.08977742
## 17 3750 meter buffer   -2.939503           +          -0.10755446
## 18 2000 meter buffer   -2.944134           +          -0.12836671
## 19 1500 meter buffer   -2.951164           +          -0.21612239
## 20 1750 meter buffer   -2.947460           +          -0.17881373
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                -0.261515339                                -0.1093808
## 2                -0.235679393                                -0.2236027
## 3                 0.058014363                                -0.3525791
## 4                -0.099587887                                -0.2419767
## 5                 0.075523849                                -0.3390423
## 6                 0.080274665                                -0.3117343
## 7                 0.044189029                                -0.2765482
## 8                -0.034303787                                -0.2149241
## 9                -0.018855638                                -0.1834547
## 10               -0.238014725                                -0.1913936
## 11               -0.290950939                                -0.1599280
## 12               -0.321410734                                -0.1182718
## 13               -0.176119696                                -0.2046227
## 14               -0.085537745                                -0.2241321
## 15                0.066759523                                -0.3130683
## 16                0.002052139                                -0.2189673
## 17               -0.040318968                                -0.2047163
## 18                0.059915768                                -0.2581474
## 19                0.028076185                                -0.1903796
```

```
## 20                 0.063695038                                -0.2025935
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.1563917                -0.05936497       -0.0042085333
## 2                     -0.2158947                -0.05955326        0.0046489457
## 3                     -0.2064242                -0.12605138       -0.0271136892
## 4                     -0.2619768                -0.02602934       -0.0096097628
## 5                     -0.1925482                -0.15164212       -0.0487836831
## 6                     -0.1849352                -0.14381325       -0.0692987305
## 7                     -0.1894938                -0.15018407       -0.0786455578
## 8                     -0.3244353                -0.03178542        0.0784057208
## 9                     -0.3590342                -0.07089473        0.0780837981
## 10                    -0.2182959                -0.12833971        0.0268554192
## 11                    -0.2316126                -0.11391831        0.0639452090
## 12                    -0.2549898                -0.11199744        0.0840044947
## 13                    -0.2153749                -0.13468010       -0.0005200369
## 14                    -0.2091631                -0.12611769       -0.0089314620
## 15                    -0.2203454                -0.09884631       -0.0336466982
## 16                    -0.1962564                -0.15263131       -0.0332706835
## 17                    -0.2081682                -0.14087529       -0.0043050036
## 18                    -0.2525273                -0.10545942       -0.0233358596
## 19                    -0.3316512                -0.07759202        0.0611248729
## 20                    -0.2944688                -0.09131610        0.0420946737
##    cond(scale(wells)) df    logLik     AICc     delta      weight
## 1          -0.1090989  9 -282.4616 583.7339 0.0000000 0.394989486
## 2           0.1248778  9 -282.6540 584.1187 0.3848032 0.325856641
## 3           0.4010348  9 -284.7050 588.2207 4.4868056 0.041907145
## 4           0.1641634  9 -284.9030 588.6169 4.8829771 0.034376411
## 5           0.3795159  9 -285.0007 588.8121 5.0782177 0.031179172
## 6           0.3748553  9 -285.4104 589.6316 5.8977266 0.020697146
## 7           0.3791443  9 -285.4133 589.6375 5.9035591 0.020636876
## 8           0.1735040  9 -285.5955 590.0018 6.2678730 0.017200228
## 9           0.2060493  9 -285.7685 590.3478 6.6139213 0.014467409
## 10          0.4651678  9 -285.8575 590.5257 6.7918106 0.013236178
## 11          0.4716751  9 -285.8599 590.5307 6.7967645 0.013203433
## 12          0.4532900  9 -285.9048 590.6205 6.8865488 0.012623811
## 13          0.4438610  9 -285.9627 590.7361 7.0022011 0.011914529
## 14          0.4188396  9 -286.1362 591.0833 7.3493725 0.010015891
## 15          0.3562711  9 -286.1864 591.1836 7.4496634 0.009526024
## 16          0.3558264  9 -286.2814 591.3737 7.6397943 0.008662142
## 17          0.3783061  9 -286.2829 591.3767 7.6427417 0.008649386
## 18          0.3050207  9 -286.9352 592.6813 8.9473658 0.004504948
## 19          0.2194890  9 -287.1646 593.1400 9.4060500 0.003581688
## 20          0.2365684  9 -287.4210 593.6529 9.9189729 0.002771456
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
```

```
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(fisher_anthro_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(fisher, absent_fisher) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    582.9    613.9   -282.5    564.9      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.7269   0.8526
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -2.970098   0.365121  -8.135 4.13e-16 ***
## scale(harvest)                    -0.372810   0.115254  -3.235  0.00122 **
## scale(seismic_lines)              -0.059365   0.081191  -0.731  0.46467
## scale(seismic_lines_3D)           -0.156392   0.127570  -1.226  0.22023
## scale(trails)                     -0.004209   0.091425  -0.046  0.96328
## scale(wells)                      -0.109099   0.081448  -1.339  0.18041
## scale(osm_industrial)             -0.261515   0.119236  -2.193  0.02829 *
## scale(pipeline_transmission_lines) -0.109381   0.130572  -0.838  0.40220
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Grey wolf

```
wolf_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%
```

```
      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(grey_wolf, absent_grey_wolf) ~

                         # HFI that aren't correlated
                         scale(harvest) +
                         scale(seismic_lines) +
                         scale(seismic_lines_3D) +
                         scale(trails) +
                         scale(wells) +
                         scale(osm_industrial) +
                         scale(pipeline_transmission_lines) +

                         # Random effect of array
                         (1|array),
                     data = .,
                     family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
wolf_anthro_model.sel <- model.sel(wolf_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
wolf_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  2000 meter buffer   -3.220596           +          0.103368649
## 2  2250 meter buffer   -3.225892           +          0.124362457
## 3  1750 meter buffer   -3.211318           +          0.084437026
## 4  2500 meter buffer   -3.223418           +          0.143458198
## 5  1500 meter buffer   -3.193621           +          0.072583037
## 6  2750 meter buffer   -3.213796           +          0.164955175
## 7  3000 meter buffer   -3.202014           +          0.160254805
## 8  1250 meter buffer   -3.175858           +          0.072686636
## 9  3250 meter buffer   -3.193875           +          0.153819497
## 10 3500 meter buffer   -3.181859           +          0.148471242
## 11 3750 meter buffer   -3.174576           +          0.138642186
## 12 1000 meter buffer   -3.167931           +          0.063166488
## 13 4000 meter buffer   -3.168011           +          0.132027971
## 14 4250 meter buffer   -3.162301           +          0.103950000
## 15 4500 meter buffer   -3.155558           +          0.085337746
## 16 4750 meter buffer   -3.149556           +          0.069320656
## 17  500 meter buffer   -3.154998           +         -0.005899598
## 18  250 meter buffer   -3.196617           +          0.031519912
## 19 5000 meter buffer   -3.144870           +          0.052728101
## 20  750 meter buffer   -3.145970           +          0.070424917
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                  -0.32843085                              0.11513789
## 2                  -0.29216457                              0.11632821
```

```
## 3                  -0.29027507                                0.08698004
## 4                  -0.23782864                                0.11847349
## 5                  -0.27173295                                0.05125234
## 6                  -0.19798364                                0.09800387
## 7                  -0.19161392                                0.09266036
## 8                  -0.24850710                                0.02968595
## 9                  -0.18538528                                0.09092028
## 10                 -0.18725239                                0.07583360
## 11                 -0.17400949                                0.04668447
## 12                 -0.18986613                                0.03246979
## 13                 -0.15774713                               -0.01008217
## 14                 -0.13342854                               -0.04789767
## 15                 -0.11929088                               -0.08091858
## 16                 -0.11027967                               -0.11996861
## 17                 -0.04928729                               -0.02624667
## 18                  0.05258660                               -0.00551844
## 19                 -0.10220916                               -0.15301803
## 20                 -0.11805238                                0.05155455
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                      -0.8979572               -0.02981141          0.22937183
## 2                      -0.9238416               -0.02200388          0.21716331
## 3                      -0.8790757               -0.10098454          0.21204923
## 4                      -0.9636481               -0.01863653          0.20784810
## 5                      -0.8376476               -0.06988959          0.19667254
## 6                      -0.9548616               -0.02822308          0.17715161
## 7                      -0.9213182               -0.03185417          0.16119033
## 8                      -0.7954583               -0.01897317          0.15413495
## 9                      -0.8858472               -0.04028454          0.15591811
## 10                     -0.8552462               -0.04462247          0.13722150
## 11                     -0.8152510               -0.02610660          0.11178624
## 12                     -0.8060020                0.03935814          0.10145368
## 13                     -0.7768847               -0.02802684          0.08157248
## 14                     -0.7534444               -0.02355878          0.07508420
## 15                     -0.7363116               -0.03009705          0.06653922
## 16                     -0.7114120               -0.02722074          0.05308246
## 17                     -0.8061073                0.06533123          0.10579168
## 18                     -1.0378013                0.01477629          0.22235651
## 19                     -0.6967970               -0.02332284          0.04543569
## 20                     -0.8092120                0.11843684          0.03785004
##    cond(scale(wells)) df    logLik     AICc       delta       weight
## 1          -0.1941208  9 -245.9326 510.6761  0.00000000 0.2452700598
## 2          -0.2213759  9 -245.9783 510.7673  0.09123205 0.2343331600
## 3          -0.1894311  9 -246.3421 511.4950  0.81887275 0.1628653055
## 4          -0.2079877  9 -246.5974 512.0056  1.32946159 0.1261698596
## 5          -0.1718898  9 -247.2201 513.2510  2.57489060 0.0676883841
## 6          -0.2047941  9 -247.5040 513.8189  3.14276088 0.0509568628
## 7          -0.2197413  9 -248.0947 515.0003  4.32417953 0.0282267503
## 8          -0.1994415  9 -248.4946 515.8000  5.12395786 0.0189230539
## 9          -0.2563423  9 -248.5318 515.8745  5.19837377 0.0182319036
## 10         -0.2519521  9 -249.1444 517.0996  6.42347195 0.0098811007
## 11         -0.2755337  9 -249.4667 517.7442  7.06813258 0.0071584509
## 12         -0.2528909  9 -249.5529 517.9167  7.24058974 0.0065670520
## 13         -0.2591065  9 -249.7164 518.2436  7.56753821 0.0055766642
## 14         -0.2632859  9 -249.9975 518.8059  8.12978781 0.0042100140
```

```
## 15         -0.2413098  9 -250.3088 519.4284  8.75231613 0.0030839210
## 16         -0.2210611  9 -250.4485 519.7077  9.03164485 0.0026819321
## 17         -0.3217369  9 -250.4932 519.7971  9.12105046 0.0025646824
## 18         -0.0940118  9 -250.5437 519.8982  9.22212852 0.0024382867
## 19         -0.1976244  9 -250.5647 519.9403  9.26416148 0.0023875772
## 20         -0.1413821  9 -251.6771 522.1650 11.48891619 0.0007849791
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(wolf_anthro_mods$`2000 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(grey_wolf, absent_grey_wolf) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    509.9    540.9   -245.9    491.9      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1494   0.3865
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                 Estimate Std. Error z value Pr(>|z|)
## (Intercept)                     -3.22060    0.20918 -15.396  < 2e-16 ***
## scale(harvest)                   0.10337    0.11432   0.904  0.36588
## scale(seismic_lines)            -0.02981    0.11606  -0.257  0.79729
## scale(seismic_lines_3D)         -0.89796    0.29493  -3.045  0.00233 **
```

```
## scale(trails)                       0.22937    0.10300   2.227  0.02595 *
## scale(wells)                       -0.19412    0.19809  -0.980  0.32711
## scale(osm_industrial)              -0.32843    0.18340  -1.791  0.07333 .
## scale(pipeline_transmission_lines)  0.11514    0.13458   0.856  0.39225
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lynx

```
lynx_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(lynx, absent_lynx) ~

                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
lynx_anthro_model.sel <- model.sel(lynx_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
lynx_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1   250 meter buffer   -1.894836           +          -0.29467069
## 2   500 meter buffer   -1.873101           +          -0.21569198
## 3  4250 meter buffer   -1.878132           +           0.17708328
## 4  5000 meter buffer   -1.875329           +           0.13109992
## 5  4750 meter buffer   -1.875487           +           0.14567041
## 6  3750 meter buffer   -1.877733           +           0.16606727
## 7  4500 meter buffer   -1.876813           +           0.17013676
```

```
## 8  3500 meter buffer   -1.876248           +           0.15982760
## 9  4000 meter buffer   -1.877722           +           0.17156337
## 10 1000 meter buffer   -1.866105           +          -0.11440623
## 11 3250 meter buffer   -1.875161           +           0.15400246
## 12  750 meter buffer   -1.866773           +          -0.15052077
## 13 3000 meter buffer   -1.873577           +           0.14246691
## 14 1250 meter buffer   -1.865800           +          -0.10474335
## 15 2750 meter buffer   -1.870552           +           0.12220359
## 16 1500 meter buffer   -1.863552           +          -0.07129125
## 17 2000 meter buffer   -1.865072           +           0.03121712
## 18 2500 meter buffer   -1.867720           +           0.09170598
## 19 1750 meter buffer   -1.863455           +          -0.01426631
## 20 2250 meter buffer   -1.866236           +           0.06938032
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                   0.09425034                                0.006616863
## 2                   0.10293881                               -0.019673189
## 3                   0.05193274                               -0.035048833
## 4                   0.08080414                               -0.058032732
## 5                   0.07244795                               -0.050452241
## 6                   0.04198486                               -0.022294739
## 7                   0.06234069                               -0.050635027
## 8                   0.04247588                               -0.012373846
## 9                   0.04479930                               -0.037003527
## 10                  0.07094306                                0.019786653
## 11                  0.04846624                               -0.012902275
## 12                  0.09380222                                0.002609593
## 13                  0.04266891                               -0.012630281
## 14                  0.05541717                                0.022198793
## 15                  0.04221334                               -0.006783697
## 16                  0.04894335                                0.026627004
## 17                  0.04712056                                0.028358338
## 18                  0.04379083                               -0.007383012
## 19                  0.04890621                                0.028220546
## 20                  0.04817181                                0.000730868
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     0.07457654               -0.165728236          0.06159282
## 2                     0.06937945               -0.081084614         -0.01431302
## 3                     0.07412847               -0.190623564          0.09286601
## 4                     0.10098593               -0.175148432          0.08851419
## 5                     0.09101677               -0.172064735          0.09413081
## 6                     0.06482959               -0.195478981          0.09471726
## 7                     0.08297493               -0.178906873          0.09220457
## 8                     0.06458772               -0.181911915          0.09696013
## 9                     0.06961254               -0.190886604          0.09032131
## 10                    0.07077167               -0.006769057          0.09299693
## 11                    0.07139784               -0.166652239          0.08938725
## 12                    0.06941278               -0.037171319          0.03048031
## 13                    0.07540087               -0.145048537          0.08869826
## 14                    0.07156556                0.039075752          0.10145198
## 15                    0.07739493               -0.123695521          0.07960387
## 16                    0.07478875                0.036136764          0.10924981
## 17                    0.08069574               -0.001349671          0.10336588
## 18                    0.07900523               -0.097932646          0.08088922
## 19                    0.07889843                0.016555868          0.10430295
```

```
## 20                    0.08147357               -0.063072555          0.08930522
##    cond(scale(wells)) df    logLik     AICc    delta       weight
## 1          -0.2236997  9 -437.7189 894.2487  0.00000 9.959526e-01
## 2          -0.2004746  9 -444.1866 907.1839 12.93525 1.546624e-03
## 3          -0.1309606  9 -445.7671 910.3449 16.09628 3.184023e-04
## 4          -0.1158957  9 -445.8725 910.5558 16.30715 2.865400e-04
## 5          -0.1196413  9 -445.9482 910.7072 16.45849 2.656579e-04
## 6          -0.1410049  9 -445.9694 910.7496 16.50088 2.600861e-04
## 7          -0.1130591  9 -446.0114 910.8337 16.58499 2.493750e-04
## 8          -0.1499100  9 -446.1870 911.1848 16.93617 2.092160e-04
## 9          -0.1231175  9 -446.2240 911.2587 17.01007 2.016268e-04
## 10         -0.2121816  9 -446.3792 911.5692 17.32049 1.726399e-04
## 11         -0.1602172  9 -446.5210 911.8529 17.60421 1.498075e-04
## 12         -0.1859588  9 -446.9218 912.6544 18.40568 1.003449e-04
## 13         -0.1684109  9 -447.0492 912.9092 18.66056 8.833821e-05
## 14         -0.2184514  9 -447.1562 913.1231 18.87448 7.937745e-05
## 15         -0.1788555  9 -447.9887 914.7882 20.53956 3.452463e-05
## 16         -0.2021069  9 -448.3550 915.5208 21.27218 2.393555e-05
## 17         -0.2143190  9 -448.7222 916.2552 22.00651 1.658006e-05
## 18         -0.1774811  9 -448.7574 916.3255 22.07686 1.600697e-05
## 19         -0.2096346  9 -448.8379 916.4866 22.23789 1.476872e-05
## 20         -0.1881892  9 -448.9228 916.6565 22.40780 1.356586e-05
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(lynx_anthro_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(lynx, absent_lynx) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
```

```
##      AIC      BIC   logLik deviance df.resid
##    893.4    924.5   -437.7    875.4      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1606   0.4007
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                       -1.894836   0.174410 -10.864  < 2e-16 ***
## scale(harvest)                    -0.294671   0.090226  -3.266  0.00109 **
## scale(seismic_lines)              -0.165728   0.071320  -2.324  0.02014 *
## scale(seismic_lines_3D)            0.074577   0.053207   1.402  0.16102
## scale(trails)                      0.061593   0.052606   1.171  0.24166
## scale(wells)                      -0.223700   0.073888  -3.028  0.00247 **
## scale(osm_industrial)              0.094250   0.051713   1.823  0.06837 .
## scale(pipeline_transmission_lines) 0.006617   0.059660   0.111  0.91169
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Moose

```
moose_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(moose, absent_moose) ~

                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
moose_anthro_model.sel <- model.sel(moose_anthro_mods) %>%

  as.data.frame() %>%
```

```
  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
moose_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1   750 meter buffer   -1.834851           +          0.0183680775
## 2  1000 meter buffer   -1.830798           +         -0.0007321436
## 3   500 meter buffer   -1.826986           +          0.0069889217
## 4  1250 meter buffer   -1.821919           +          0.0138909705
## 5  1500 meter buffer   -1.821835           +          0.0004848259
## 6  2500 meter buffer   -1.818893           +         -0.0374481328
## 7  3000 meter buffer   -1.818311           +         -0.0411993488
## 8  1750 meter buffer   -1.819386           +         -0.0147424390
## 9  2750 meter buffer   -1.817997           +         -0.0409496051
## 10 3250 meter buffer   -1.816784           +         -0.0479719094
## 11 3500 meter buffer   -1.815764           +         -0.0602364139
## 12 3750 meter buffer   -1.814890           +         -0.0717997559
## 13 2000 meter buffer   -1.817848           +         -0.0282153014
## 14 2250 meter buffer   -1.817706           +         -0.0252919481
## 15 4000 meter buffer   -1.813877           +         -0.0782905080
## 16  250 meter buffer   -1.813660           +         -0.0025584285
## 17 4250 meter buffer   -1.811849           +         -0.0881673185
## 18 4500 meter buffer   -1.810556           +         -0.0990084223
## 19 5000 meter buffer   -1.810027           +         -0.1123890697
## 20 4750 meter buffer   -1.809936           +         -0.1049721696
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                   -0.3433885                             0.0966437760
## 2                   -0.3138647                             0.0793705338
## 3                   -0.2876972                             0.0700746093
## 4                   -0.2539251                             0.0877363064
## 5                   -0.2434553                             0.0886337280
## 6                   -0.2052588                             0.1786350775
## 7                   -0.2303478                             0.1952696928
## 8                   -0.1902955                             0.1020286967
## 9                   -0.2232597                             0.1921100396
## 10                  -0.2406866                             0.1915315588
## 11                  -0.2485917                             0.1884764137
## 12                  -0.2566540                             0.1806878808
## 13                  -0.1659176                             0.1176421042
## 14                  -0.1769857                             0.1449500609
## 15                  -0.2502905                             0.1638160723
## 16                  -0.1620849                            -0.0008116912
## 17                  -0.2331126                             0.1441355832
## 18                  -0.2127881                             0.1356954426
## 19                  -0.1832783                             0.1556111815
## 20                  -0.1909256                             0.1413337033
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                    0.087406667                0.015413734          0.05467494
## 2                    0.068487407               -0.050993190          0.07085907
## 3                    0.100326796                0.069436351          0.06798361
```

```
## 4                    0.054913317               -0.038167047          0.08574822
## 5                    0.045153145               -0.065373593          0.09490797
## 6                   -0.024933718               -0.012025228          0.09334724
## 7                   -0.032250519               -0.030413547          0.09251953
## 8                    0.027391544               -0.075249170          0.07995651
## 9                   -0.032164018               -0.014056939          0.09192659
## 10                  -0.033865855               -0.030785970          0.09264045
## 11                  -0.035857910               -0.029600929          0.09262267
## 12                  -0.037737671               -0.026343553          0.09552988
## 13                   0.008666257               -0.042813700          0.07710227
## 14                  -0.008776108               -0.022815706          0.08022005
## 15                  -0.037312659               -0.026489429          0.09752037
## 16                   0.111910277                0.090396184          0.09089820
## 17                  -0.032047910               -0.013893253          0.10077367
## 18                  -0.029145450                0.002179302          0.10271458
## 19                  -0.033124806                0.019215524          0.10460411
## 20                  -0.028302305                0.014932788          0.10208187
##    cond(scale(wells)) df    logLik     AICc      delta       weight
## 1         -0.15573120  9 -433.5507 885.9122  0.0000000 0.5257873895
## 2         -0.20550426  9 -434.0008 886.8123  0.9001188 0.3352369244
## 3         -0.09245836  9 -435.7866 890.3839  4.4717229 0.0562066712
## 4         -0.18125650  9 -437.3077 893.4263  7.5140624 0.0122786961
## 5         -0.20159559  9 -437.3623 893.5354  7.6231891 0.0116266791
## 6         -0.25697587  9 -437.7120 894.2348  8.3226226 0.0081955035
## 7         -0.22388644  9 -437.7451 894.3010  8.3887584 0.0079289270
## 8         -0.25642441  9 -437.8704 894.5516  8.6394094 0.0069949765
## 9         -0.22684062  9 -437.8975 894.6058  8.6936392 0.0068078568
## 10        -0.19943281  9 -438.0525 894.9158  9.0035676 0.0058305603
## 11        -0.18518977  9 -438.2267 895.2642  9.3520104 0.0048983171
## 12        -0.15718134  9 -438.5155 895.8419  9.9296893 0.0036694884
## 13        -0.27734814  9 -438.5776 895.9659 10.0537201 0.0034488362
## 14        -0.26706105  9 -438.5926 895.9961 10.0838984 0.0033971869
## 15        -0.14629008  9 -438.8747 896.5603 10.6480836 0.0025621715
## 16        -0.03078054  9 -439.1295 897.0699 11.1576591 0.0019858904
## 17        -0.14194295  9 -439.6408 898.0924 12.1801654 0.0011910250
## 18        -0.14892773  9 -440.1562 899.1232 13.2109628 0.0007113544
## 19        -0.18440736  9 -440.2555 899.3218 13.4096253 0.0006440907
## 20        -0.17376918  9 -440.3307 899.4721 13.5599468 0.0005974549
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
```

```
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(moose_anthro_mods$`750 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(moose, absent_moose) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    885.1    916.1   -433.6    867.1      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.2396   0.4895
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -1.83485    0.20929  -8.767  < 2e-16 ***
## scale(harvest)                      0.01837    0.06765   0.272 0.785981
## scale(seismic_lines)                0.01541    0.06597   0.234 0.815250
## scale(seismic_lines_3D)             0.08741    0.06029   1.450 0.147104
## scale(trails)                       0.05467    0.05245   1.042 0.297238
## scale(wells)                       -0.15573    0.08944  -1.741 0.081644 .
## scale(osm_industrial)              -0.34339    0.08957  -3.834 0.000126 ***
## scale(pipeline_transmission_lines)  0.09664    0.06979   1.385 0.166093
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Red fox

```
fox_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(red_fox, absent_red_fox) ~

                        # HFI that aren't correlated
```

```
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
fox_anthro_model.sel <- model.sel(fox_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
fox_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  1750 meter buffer   -4.592612           +          -0.26286812
## 2  2000 meter buffer   -4.532780           +          -0.21717820
## 3  1500 meter buffer   -4.486297           +          -0.37666940
## 4  2250 meter buffer   -4.490386           +          -0.24493178
## 5  2500 meter buffer   -4.508543           +          -0.19353507
## 6  4250 meter buffer   -4.480454           +          -0.13874649
## 7  4500 meter buffer   -4.514180           +          -0.14027307
## 8  1250 meter buffer   -4.370437           +          -0.39906687
## 9  4000 meter buffer   -4.452845           +          -0.15279827
## 10 3750 meter buffer   -4.457178           +          -0.16131488
## 11 4750 meter buffer   -4.511307           +          -0.12339492
## 12 2750 meter buffer   -4.467354           +          -0.14086498
## 13 5000 meter buffer   -4.468562           +          -0.08202983
## 14 3000 meter buffer   -4.452072           +          -0.12976118
## 15  500 meter buffer   -4.374015           +          -0.18812253
## 16 3250 meter buffer   -4.408018           +          -0.14975688
## 17 3500 meter buffer   -4.427207           +          -0.17301465
## 18 1000 meter buffer   -4.310002           +          -0.38185222
## 19  750 meter buffer   -4.324336           +          -0.37675609
## 20  250 meter buffer   -4.307371           +          -0.15799042
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                   0.44701079                                0.324038108
## 2                   0.39515293                                0.345598267
## 3                   0.36805304                                0.217907576
## 4                   0.36720262                                0.403817550
## 5                   0.39306084                                0.466149400
## 6                   0.28236079                                0.714919142
```

```
## 7                  0.42598222                              0.753243079
## 8                  0.31616394                              0.059675277
## 9                  0.11613881                              0.719388279
## 10                 0.07542929                              0.758485305
## 11                 0.44767922                              0.766823832
## 12                 0.35289758                              0.502462046
## 13                 0.42312798                              0.655295194
## 14                 0.30135658                              0.600864352
## 15                 0.38674306                              0.053999230
## 16                 0.24888709                              0.562203488
## 17                 0.21850584                              0.702367548
## 18                 0.19832465                              0.008758208
## 19                 0.30654177                              0.130319615
## 20                 0.24807610                             -0.033861558
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                    -0.22085658                 -0.4868721          -0.8834670
## 2                    -0.29498741                 -0.4768147          -0.7528324
## 3                    -0.12128691                 -0.4365305          -0.7755132
## 4                    -0.33791853                 -0.3919251          -0.6415524
## 5                    -0.36657708                 -0.4114253          -0.6323124
## 6                    -0.69654461                 -0.2709903          -0.3811354
## 7                    -0.62738237                 -0.3166534          -0.4311867
## 8                    -0.13259309                 -0.3583284          -0.6933834
## 9                    -0.67057909                 -0.2413443          -0.2684029
## 10                   -0.63565483                 -0.2503769          -0.1879981
## 11                   -0.65749766                 -0.3208702          -0.3783411
## 12                   -0.41890581                 -0.3420415          -0.4959890
## 13                   -0.71136367                 -0.2512876          -0.3243299
## 14                   -0.42734023                 -0.2604621          -0.3654257
## 15                   -0.01503751                 -0.2116350          -0.6258047
## 16                   -0.47190799                 -0.1987774          -0.2426482
## 17                   -0.50117583                 -0.2699965          -0.1972197
## 18                   -0.09753932                 -0.3227392          -0.6295377
## 19                   -0.07750341                 -0.1868773          -0.6375869
## 20                    0.12026800                 -0.1493282          -0.8489739
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1         -0.08828480  9 -173.6990 366.2088  0.000000 7.325857e-01
## 2          0.01458861  9 -175.2407 369.2923  3.083426 1.567838e-01
## 3         -0.03182992  9 -176.2325 371.2759  5.067047 5.815179e-02
## 4          0.04205289  9 -177.8491 374.5089  8.300087 1.154828e-02
## 5          0.01523485  9 -178.3298 375.4705  9.261660 7.140268e-03
## 6          0.21501072  9 -178.6243 376.0595  9.850678 5.318765e-03
## 7          0.09272988  9 -178.7863 376.3835 10.174668 4.523320e-03
## 8          0.19525576  9 -178.7877 376.3862 10.177371 4.517211e-03
## 9          0.27955167  9 -178.8836 376.5781 10.369233 4.104008e-03
## 10         0.25873331  9 -178.9330 376.6768 10.468018 3.906224e-03
## 11         0.08300829  9 -179.1461 377.1030 10.894142 3.156644e-03
## 12         0.11544595  9 -179.2526 377.3159 11.107095 2.837811e-03
## 13         0.21082390  9 -179.9893 378.7895 12.580679 1.358309e-03
## 14         0.16092921  9 -180.0568 378.9244 12.715571 1.269718e-03
## 15         0.47561059  9 -180.1740 379.1589 12.950056 1.129249e-03
## 16         0.25467588  9 -180.5557 379.9223 13.713472 7.709314e-04
## 17         0.14450934  9 -180.5757 379.9621 13.753312 7.557265e-04
## 18         0.25200724  9 -182.7290 384.2688 18.059938 8.773903e-05
```

```
## 19         0.29356934  9 -183.2344 385.2795 19.070684 5.293123e-05
## 20         0.13914895  9 -186.7350 392.2808 26.071959 1.597366e-06
##                   Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(fox_anthro_mods$`1750 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(red_fox, absent_red_fox) ~ scale(harvest) + scale(seismic_lines) +
##     scale(seismic_lines_3D) + scale(trails) + scale(wells) +
##     scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    365.4    396.4   -173.7    347.4      223
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 2.963    1.721
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)                        -4.59261    0.76036  -6.040 1.54e-09 ***
## scale(harvest)                     -0.26287    0.24399  -1.077 0.281304
## scale(seismic_lines)               -0.48687    0.15878  -3.066 0.002167 **
## scale(seismic_lines_3D)            -0.22086    0.29037  -0.761 0.446900
## scale(trails)                      -0.88347    0.25707  -3.437 0.000589 ***
## scale(wells)                       -0.08828    0.24232  -0.364 0.715610
## scale(osm_industrial)               0.44701    0.17367   2.574 0.010054 *
## scale(pipeline_transmission_lines)  0.32404    0.23535   1.377 0.168557
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

White-tailed deer

```
deer_anthro_mods <- osm_anthro_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(`white-tailed_deer`, `absent_white-tailed_deer`) ~

                        # HFI that aren't correlated
                        scale(harvest) +
                        scale(seismic_lines) +
                        scale(seismic_lines_3D) +
                        scale(trails) +
                        scale(wells) +
                        scale(osm_industrial) +
                        scale(pipeline_transmission_lines) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
deer_anthro_model.sel <- model.sel(deer_anthro_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
deer_anthro_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(harvest))
## 1  1500 meter buffer  -0.1949688           +          -0.08753506
## 2  1250 meter buffer  -0.1941255           +          -0.08802588
## 3  1750 meter buffer  -0.1896187           +          -0.10391011
## 4  4000 meter buffer  -0.1970629           +          -0.23130893
## 5  3750 meter buffer  -0.1931356           +          -0.20818608
## 6  2000 meter buffer  -0.1887216           +          -0.11206117
## 7  2250 meter buffer  -0.1851221           +          -0.12969512
## 8  4250 meter buffer  -0.2003849           +          -0.24413573
## 9  3500 meter buffer  -0.1920635           +          -0.18816085
## 10 3250 meter buffer  -0.1878558           +          -0.16527370
## 11 4500 meter buffer  -0.2039623           +          -0.25288839
```

```
## 12 2500 meter buffer  -0.1853276           +          -0.13080511
## 13 2750 meter buffer  -0.1840471           +          -0.13375845
## 14 1000 meter buffer  -0.1872159           +          -0.11662570
## 15 3000 meter buffer  -0.1846154           +          -0.15093998
## 16 4750 meter buffer  -0.2081727           +          -0.27156534
## 17 5000 meter buffer  -0.2115667           +          -0.29101335
## 18  750 meter buffer  -0.1784914           +          -0.08545025
## 19  500 meter buffer  -0.1836003           +          -0.01531539
## 20  250 meter buffer  -0.1848293           +           0.02890545
##    cond(scale(osm_industrial)) cond(scale(pipeline_transmission_lines))
## 1                    0.3266929                                0.068918042
## 2                    0.3531894                                0.071355953
## 3                    0.2887577                                0.073991666
## 4                    0.2006858                                0.016824965
## 5                    0.1935731                                0.046181915
## 6                    0.2602571                                0.058618826
## 7                    0.2433420                                0.048601940
## 8                    0.2086745                               -0.003816363
## 9                    0.1937264                                0.047904809
## 10                   0.1997387                                0.055874076
## 11                   0.2178106                               -0.019732642
## 12                   0.2278389                                0.048843146
## 13                   0.2179251                                0.045337615
## 14                   0.3464785                                0.091748708
## 15                   0.2113551                                0.050276531
## 16                   0.2228619                               -0.046201932
## 17                   0.2222918                               -0.087870812
## 18                   0.3466307                                0.127212159
## 19                   0.3323732                                0.136149032
## 20                   0.2452461                                0.109232041
##    cond(scale(seismic_lines_3D)) cond(scale(seismic_lines)) cond(scale(trails))
## 1                     -0.4286887                -0.07579262          0.16605451
## 2                     -0.4239948                -0.06642569          0.14971158
## 3                     -0.4028014                -0.09030177          0.17126342
## 4                     -0.3928509                -0.03262868          0.23454814
## 5                     -0.4027613                -0.02544128          0.23608550
## 6                     -0.3905780                -0.08964066          0.17165898
## 7                     -0.3650659                -0.05132720          0.19006195
## 8                     -0.3770892                -0.05001198          0.22406437
## 9                     -0.3998331                -0.03283989          0.23055478
## 10                    -0.3866642                -0.02725456          0.22161964
## 11                    -0.3705582                -0.04690424          0.21741979
## 12                    -0.3605450                -0.05163172          0.20204683
## 13                    -0.3666263                -0.03938735          0.20047128
## 14                    -0.3536116                -0.07066707          0.14494716
## 15                    -0.3691539                -0.02600635          0.20431238
## 16                    -0.3561955                -0.03453529          0.21628748
## 17                    -0.3352126                -0.02876257          0.21748385
## 18                    -0.2964609                -0.02424617          0.08335996
## 19                    -0.2827067                 0.02861321          0.09759079
## 20                    -0.2470869                -0.04876880          0.18617039
##    cond(scale(wells)) df    logLik     AICc     delta       weight
## 1           0.3433686  9 -565.1554 1149.122  0.000000 7.322431e-01
## 2           0.3240821  9 -566.4778 1151.766  2.644627 1.951560e-01
```

```
## 3           0.3455140  9 -567.7221 1154.255  5.133266 5.623164e-02
## 4           0.4280328  9 -570.4370 1159.685 10.563062 3.723193e-03
## 5           0.4250871  9 -570.4419 1159.695 10.572899 3.704925e-03
## 6           0.3599137  9 -570.6865 1160.184 11.062160 2.900933e-03
## 7           0.3895186  9 -571.2485 1161.308 12.186166 1.653723e-03
## 8           0.4254652  9 -571.5245 1161.860 12.738132 1.254888e-03
## 9           0.4152409  9 -571.7318 1162.274 13.152770 1.019923e-03
## 10          0.4150440  9 -572.5072 1163.825 14.703526 4.697051e-04
## 11          0.4238344  9 -572.5605 1163.932 14.810105 4.453300e-04
## 12          0.3944214  9 -572.8046 1164.420 15.298292 3.488779e-04
## 13          0.4158101  9 -573.1861 1165.183 16.061241 2.382325e-04
## 14          0.2919624  9 -573.2028 1165.216 16.094653 2.342857e-04
## 15          0.4168705  9 -573.3591 1165.529 16.407254 2.003851e-04
## 16          0.4201377  9 -574.0336 1166.878 17.756217 1.020803e-04
## 17          0.4348209  9 -574.3720 1167.555 18.433195 7.276758e-05
## 18          0.2746656  9 -584.3733 1187.557 38.435784 3.299369e-09
## 19          0.2579680  9 -590.4102 1199.631 50.509463 7.882514e-12
## 20          0.0664280  9 -598.5895 1215.990 66.868014 2.210296e-15
##                    Dataset
## 1  osm_anthro_df_2021_2022
## 2  osm_anthro_df_2021_2022
## 3  osm_anthro_df_2021_2022
## 4  osm_anthro_df_2021_2022
## 5  osm_anthro_df_2021_2022
## 6  osm_anthro_df_2021_2022
## 7  osm_anthro_df_2021_2022
## 8  osm_anthro_df_2021_2022
## 9  osm_anthro_df_2021_2022
## 10 osm_anthro_df_2021_2022
## 11 osm_anthro_df_2021_2022
## 12 osm_anthro_df_2021_2022
## 13 osm_anthro_df_2021_2022
## 14 osm_anthro_df_2021_2022
## 15 osm_anthro_df_2021_2022
## 16 osm_anthro_df_2021_2022
## 17 osm_anthro_df_2021_2022
## 18 osm_anthro_df_2021_2022
## 19 osm_anthro_df_2021_2022
## 20 osm_anthro_df_2021_2022
```

```
summary(deer_anthro_mods$`1500 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(‘white-tailed_deer‘, ‘absent_white-tailed_deer‘) ~ scale(harvest) +
##     scale(seismic_lines) + scale(seismic_lines_3D) + scale(trails) +
##     scale(wells) + scale(osm_industrial) + scale(pipeline_transmission_lines) +
##     (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##   1148.3   1179.3   -565.2   1130.3      223
##
## Random effects:
```

```
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 1.779    1.334
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                                  Estimate Std. Error z value Pr(>|z|)
## (Intercept)                      -0.19497    0.54694  -0.356  0.72149
## scale(harvest)                   -0.08754    0.06511  -1.344  0.17879
## scale(seismic_lines)             -0.07579    0.06319  -1.199  0.23039
## scale(seismic_lines_3D)          -0.42869    0.09018  -4.754 2.00e-06 ***
## scale(trails)                     0.16605    0.05173   3.210  0.00133 **
## scale(wells)                      0.34337    0.07246   4.739 2.15e-06 ***
## scale(osm_industrial)             0.32669    0.05488   5.953 2.63e-09 ***
## scale(pipeline_transmission_lines)  0.06892    0.07549   0.913  0.36126
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Save model results

We also want to save the output from the model.sel funciton for each species as one csv file for use plotting later like we did with the global analysis

```
# provide list of model.sel data frames from global analysis

anthro_model.sel_data <- list(
  bear = bear_anthro_model.sel,
  caribou = caribou_anthro_model.sel,
  coyote = coyote_anthro_model.sel,
  fisher = fisher_anthro_model.sel,
  wolf = wolf_anthro_model.sel,
  lynx = lynx_anthro_model.sel,
  moose = moose_anthro_model.sel,
  fox = fox_anthro_model.sel,
  deer = deer_anthro_model.sel
) %>%

  # use purrr to combine data and extract species names to use as a column
  map_dfr(~.x %>%

            mutate(species = deparse(substitute(.x))),
          .id = "species")
```

And save this for use later

```
write_csv(anthro_model.sel_data,
          'data/processed/OSM_glm_anthro_model_sel_data.csv')
```

Landscape

Now we will duplicate this analysis but for data that only includes landscape features.

For now we have to select EITHER lc_forest or lc_shrub as they are highly correlated

Black bear

```
bear_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(black_bear, absent_black_bear) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
bear_land_model.sel <- model.sel(bear_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
bear_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1   250 meter buffer  -0.5940062           +               -0.22455623
## 2   500 meter buffer  -0.5921419           +               -0.12596244
## 3  1000 meter buffer  -0.5902570           +               -0.08992757
## 4  1250 meter buffer  -0.5899130           +               -0.09248092
## 5   750 meter buffer  -0.5913420           +               -0.10334653
## 6  1500 meter buffer  -0.5892872           +               -0.08210297
## 7  2000 meter buffer  -0.5885890           +               -0.06772306
## 8  2250 meter buffer  -0.5881483           +               -0.06175796
## 9  2500 meter buffer  -0.5882895           +               -0.06528588
## 10 1750 meter buffer  -0.5888792           +               -0.06971220
## 11 2750 meter buffer  -0.5889258           +               -0.07668298
## 12 3000 meter buffer  -0.5889580           +               -0.07805944
## 13 3500 meter buffer  -0.5892280           +               -0.08398031
## 14 3750 meter buffer  -0.5894918           +               -0.08584491
## 15 3250 meter buffer  -0.5890142           +               -0.07954595
## 16 4000 meter buffer  -0.5898104           +               -0.08629769
```

```
## 17 4250 meter buffer  -0.5900128           +               -0.08116356
## 18 4500 meter buffer  -0.5900676           +               -0.07534982
## 19 5000 meter buffer  -0.5896616           +               -0.06409369
## 20 4750 meter buffer  -0.5896032           +               -0.06501601
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1            -0.166545966             -0.0338886818  5 -456.3951 923.0557
## 2            -0.062020095              0.0189681862  5 -460.9899 932.2453
## 3             0.008256353             -0.0321157828  5 -461.0813 932.4281
## 4            -0.009789065             -0.0352047456  5 -461.1460 932.5576
## 5            -0.005930996             -0.0007372826  5 -461.4152 933.0958
## 6            -0.007234420             -0.0284842932  5 -461.5450 933.3554
## 7             0.020710391             -0.0290792061  5 -461.6240 933.5134
## 8             0.020953730             -0.0354869272  5 -461.6914 933.6482
## 9             0.017800095             -0.0274581274  5 -461.7985 933.8625
## 10            0.008760172             -0.0223054735  5 -461.8327 933.9309
## 11            0.014033317             -0.0054353999  5 -461.8822 934.0299
## 12            0.008553426             -0.0036752402  5 -461.9418 934.1491
## 13           -0.015144122              0.0004516173  5 -462.0006 934.2667
## 14           -0.024249011              0.0057615924  5 -462.0108 934.2870
## 15           -0.004495071             -0.0014421209  5 -462.0464 934.3584
## 16           -0.028752071              0.0147876169  5 -462.0493 934.3642
## 17           -0.034458698              0.0192829301  5 -462.1781 934.6216
## 18           -0.042564861              0.0195784137  5 -462.2521 934.7696
## 19           -0.055883047              0.0020862330  5 -462.2547 934.7750
## 20           -0.048356628              0.0046211298  5 -462.3337 934.9329
##        delta      weight                  Dataset
## 1   0.000000 0.914581004 osm_anthro_df_2021_2022
## 2   9.189548 0.009241386 osm_anthro_df_2021_2022
## 3   9.372341 0.008434206 osm_anthro_df_2021_2022
## 4   9.501854 0.007905345 osm_anthro_df_2021_2022
## 5  10.040106 0.006040055 osm_anthro_df_2021_2022
## 6  10.299733 0.005304735 osm_anthro_df_2021_2022
## 7  10.457723 0.004901810 osm_anthro_df_2021_2022
## 8  10.592532 0.004582295 osm_anthro_df_2021_2022
## 9  10.806775 0.004116811 osm_anthro_df_2021_2022
## 10 10.875201 0.003978344 osm_anthro_df_2021_2022
## 11 10.974181 0.003786248 osm_anthro_df_2021_2022
## 12 11.093416 0.003567119 osm_anthro_df_2021_2022
## 13 11.211013 0.003363424 osm_anthro_df_2021_2022
## 14 11.231311 0.003329461 osm_anthro_df_2021_2022
## 15 11.302670 0.003212762 osm_anthro_df_2021_2022
## 16 11.308445 0.003203498 osm_anthro_df_2021_2022
## 17 11.565912 0.002816543 osm_anthro_df_2021_2022
## 18 11.713873 0.002615694 osm_anthro_df_2021_2022
## 19 11.719248 0.002608674 osm_anthro_df_2021_2022
## 20 11.877193 0.002410586 osm_anthro_df_2021_2022
```

```
summary(bear_land_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(black_bear, absent_black_bear) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
```

```
##
##      AIC      BIC   logLik deviance df.resid
##    922.8    940.0   -456.4    912.8      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1092   0.3304
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -0.59401    0.14308  -4.151  3.3e-05 ***
## scale(lc_forest)    -0.16655    0.06698  -2.487 0.012898 *
## scale(lc_grassland) -0.03389    0.05487  -0.618 0.536799
## scale(lc_developed) -0.22456    0.06436  -3.489 0.000485 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Caribou

```
caribou_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(caribou, absent_caribou) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
caribou_land_model.sel <- model.sel(caribou_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
caribou_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1   250 meter buffer   -5.039744           +               -0.07920529
## 2  1250 meter buffer   -5.073363           +               -0.47573584
## 3  1500 meter buffer   -5.073111           +               -0.46442173
## 4   500 meter buffer   -5.020600           +                0.02154320
## 5   750 meter buffer   -5.022980           +               -0.10813319
## 6  1750 meter buffer   -5.073948           +               -0.52212117
## 7  1000 meter buffer   -5.039413           +               -0.33123049
## 8  2000 meter buffer   -5.036072           +               -0.41612035
## 9  4750 meter buffer   -5.016494           +                0.02212472
## 10 4500 meter buffer   -5.021824           +                0.09100185
## 11 2250 meter buffer   -5.017428           +               -0.33761804
## 12 5000 meter buffer   -5.020685           +                0.09191273
## 13 4250 meter buffer   -5.017737           +                0.10874223
## 14 2500 meter buffer   -5.007387           +               -0.29407088
## 15 2750 meter buffer   -5.002110           +               -0.26287406
## 16 4000 meter buffer   -5.014757           +                0.12757463
## 17 3750 meter buffer   -4.995640           +               -0.01349521
## 18 3500 meter buffer   -4.993128           +               -0.05918197
## 19 3000 meter buffer   -4.993432           +               -0.19813164
## 20 3250 meter buffer   -4.988324           +               -0.10344989
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1             0.221768285              -0.15411729  5 -150.5854 311.4363
## 2             0.118284339              -0.12061144  5 -151.1989 312.6634
## 3             0.121616392              -0.14279265  5 -151.2090 312.6835
## 4             0.239995535              -0.12290176  5 -151.3883 313.0422
## 5             0.197592620              -0.15224614  5 -151.5661 313.3977
## 6             0.106181455              -0.10122006  5 -151.6441 313.5536
## 7             0.142141484              -0.09792318  5 -151.9191 314.1037
## 8             0.090467148              -0.08011251  5 -152.5766 315.4186
## 9            -0.050299764              -0.20895635  5 -152.9150 316.0955
## 10           -0.038887723              -0.21200637  5 -152.9523 316.1700
## 11            0.074887644              -0.08964834  5 -152.9810 316.2274
## 12           -0.059831519              -0.19529503  5 -153.0109 316.2872
## 13           -0.035484610              -0.18805158  5 -153.1334 316.5322
## 14            0.057850255              -0.09180166  5 -153.2063 316.6782
## 15            0.043151889              -0.09703247  5 -153.3160 316.8975
## 16           -0.038377535              -0.15636930  5 -153.3353 316.9361
## 17           -0.023504821              -0.13742783  5 -153.4555 317.1764
## 18           -0.001796785              -0.13123674  5 -153.4796 317.2247
## 19            0.032108527              -0.09638687  5 -153.4846 317.2347
## 20            0.015118354              -0.10318028  5 -153.6017 317.4689
##       delta     weight                Dataset
## 1  0.000000 0.22151625 osm_anthro_df_2021_2022
## 2  1.227057 0.11993711 osm_anthro_df_2021_2022
## 3  1.247209 0.11873469 osm_anthro_df_2021_2022
## 4  1.605845 0.09924320 osm_anthro_df_2021_2022
## 5  1.961364 0.08308081 osm_anthro_df_2021_2022
## 6  2.117316 0.07684866 osm_anthro_df_2021_2022
## 7  2.667427 0.05836887 osm_anthro_df_2021_2022
## 8  3.982290 0.03024561 osm_anthro_df_2021_2022
## 9  4.659132 0.02156195 osm_anthro_df_2021_2022
## 10 4.733686 0.02077298 osm_anthro_df_2021_2022
## 11 4.791080 0.02018533 osm_anthro_df_2021_2022
```

```
## 12 4.850896 0.01959056 osm_anthro_df_2021_2022
## 13 5.095873 0.01733208 osm_anthro_df_2021_2022
## 14 5.241848 0.01611212 osm_anthro_df_2021_2022
## 15 5.461181 0.01443860 osm_anthro_df_2021_2022
## 16 5.499741 0.01416289 osm_anthro_df_2021_2022
## 17 5.740115 0.01255901 osm_anthro_df_2021_2022
## 18 5.788329 0.01225987 osm_anthro_df_2021_2022
## 19 5.798337 0.01219867 osm_anthro_df_2021_2022
## 20 6.032527 0.01085073 osm_anthro_df_2021_2022
```

```
summary(caribou_land_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(caribou, absent_caribou) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    311.2    328.4   -150.6    301.2      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 5.351    2.313
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -5.03974    1.10127  -4.576 4.73e-06 ***
## scale(lc_forest)     0.22177    0.15873   1.397    0.162
## scale(lc_grassland) -0.15412    0.16528  -0.932    0.351
## scale(lc_developed) -0.07921    0.20030  -0.395    0.693
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Coyote

```
coyote_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(coyote, absent_coyote) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
```

```
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
coyote_land_model.sel <- model.sel(coyote_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
coyote_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1  3750 meter buffer   -1.368761          +                 0.4529572
## 2  4500 meter buffer   -1.365998          +                 0.4741382
## 3  4750 meter buffer   -1.364961          +                 0.4811141
## 4  5000 meter buffer   -1.363739          +                 0.4857252
## 5  3500 meter buffer   -1.368307          +                 0.4483529
## 6  4250 meter buffer   -1.367207          +                 0.4594487
## 7  4000 meter buffer   -1.368569          +                 0.4494386
## 8  3250 meter buffer   -1.368265          +                 0.4331079
## 9  3000 meter buffer   -1.367635          +                 0.4248031
## 10 1250 meter buffer   -1.366635          +                 0.4172910
## 11 1750 meter buffer   -1.363617          +                 0.4389814
## 12 2750 meter buffer   -1.366238          +                 0.4196682
## 13 2000 meter buffer   -1.363295          +                 0.4372304
## 14 1500 meter buffer   -1.365343          +                 0.4235781
## 15 2250 meter buffer   -1.364285          +                 0.4356357
## 16 2500 meter buffer   -1.365244          +                 0.4240073
## 17 1000 meter buffer   -1.368914          +                 0.3838178
## 18  750 meter buffer   -1.370036          +                 0.3340027
## 19  500 meter buffer   -1.369698          +                 0.2110103
## 20  250 meter buffer   -1.373489          +                 0.1435200
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik      AICc
## 1              -0.2504571              -0.075337957  5 -486.1135  982.4925
## 2              -0.2354329              -0.092235973  5 -486.1745  982.6145
## 3              -0.2305974              -0.090665973  5 -486.1813  982.6280
## 4              -0.2254750              -0.091480668  5 -486.4594  983.1844
## 5              -0.2468286              -0.068584943  5 -486.7078  983.6810
## 6              -0.2404484              -0.083466339  5 -486.9957  984.2569
## 7              -0.2481005              -0.074203486  5 -487.0113  984.2881
## 8              -0.2468445              -0.052268277  5 -488.0951  986.4557
## 9              -0.2440525              -0.050349030  5 -489.1893  988.6441
## 10             -0.1826078              -0.017412912  5 -490.4368  991.1391
## 11             -0.1670040              -0.050528762  5 -490.5229  991.3113
## 12             -0.2305268              -0.055643903  5 -490.9381  992.1416
```

```
## 13             -0.1757053              -0.072877408  5 -491.2952  992.8558
## 14             -0.1715850              -0.028573541  5 -491.3307  992.9268
## 15             -0.1913409              -0.083067018  5 -491.4137  993.0930
## 16             -0.2120977              -0.070989096  5 -491.6408  993.5471
## 17             -0.1960226               0.009004420  5 -491.8013  993.8681
## 18             -0.1861901               0.048059099  5 -497.9676 1006.2006
## 19             -0.1916414               0.039140690  5 -513.3686 1037.0028
## 20             -0.1977173               0.006203179  5 -519.6501 1049.5657
##         delta       weight                  Dataset
## 1   0.0000000 1.924163e-01 osm_anthro_df_2021_2022
## 2   0.1219884 1.810308e-01 osm_anthro_df_2021_2022
## 3   0.1354810 1.798136e-01 osm_anthro_df_2021_2022
## 4   0.6918619 1.361463e-01 osm_anthro_df_2021_2022
## 5   1.1885301 1.062076e-01 osm_anthro_df_2021_2022
## 6   1.7643536 7.963745e-02 osm_anthro_df_2021_2022
## 7   1.7955616 7.840443e-02 osm_anthro_df_2021_2022
## 8   3.9632395 2.652377e-02 osm_anthro_df_2021_2022
## 9   6.1516030 8.880517e-03 osm_anthro_df_2021_2022
## 10  8.6465481 2.550749e-03 osm_anthro_df_2021_2022
## 11  8.8188190 2.340236e-03 osm_anthro_df_2021_2022
## 12  9.6490889 1.545143e-03 osm_anthro_df_2021_2022
## 13 10.3632991 1.081135e-03 osm_anthro_df_2021_2022
## 14 10.4343382 1.043408e-03 osm_anthro_df_2021_2022
## 15 10.6004605 9.602429e-04 osm_anthro_df_2021_2022
## 16 11.0545476 7.652042e-04 osm_anthro_df_2021_2022
## 17 11.3755846 6.517260e-04 osm_anthro_df_2021_2022
## 18 23.7081158 1.368012e-06 osm_anthro_df_2021_2022
## 19 54.5102766 2.802113e-13 osm_anthro_df_2021_2022
## 20 67.0732387 5.241711e-16 osm_anthro_df_2021_2022
```

```
summary(coyote_land_mods$`3750 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(coyote, absent_coyote) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    982.2    999.5   -486.1    972.2      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1374   0.3707
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.36876    0.16060  -8.523  < 2e-16 ***
## scale(lc_forest)    -0.25046    0.05751  -4.355 1.33e-05 ***
## scale(lc_grassland) -0.07534    0.06625  -1.137    0.255
## scale(lc_developed)  0.45296    0.06442   7.032 2.04e-12 ***
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Fisher

```
fisher_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(fisher, absent_fisher) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
fisher_land_model.sel <- model.sel(fisher_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
fisher_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1  2500 meter buffer   -2.968379          +                 0.4462479
## 2  3250 meter buffer   -2.964730          +                 0.4788544
## 3  2750 meter buffer   -2.967005          +                 0.4573525
## 4  3000 meter buffer   -2.965289          +                 0.4730753
## 5  2250 meter buffer   -2.968784          +                 0.4207583
## 6  3500 meter buffer   -2.959554          +                 0.4584811
## 7  2000 meter buffer   -2.970962          +                 0.4006872
## 8  3750 meter buffer   -2.951848          +                 0.4343198
## 9  4250 meter buffer   -2.949502          +                 0.4292259
## 10 4000 meter buffer   -2.949604          +                 0.4244698
## 11 4500 meter buffer   -2.946984          +                 0.4115209
## 12 4750 meter buffer   -2.945720          +                 0.3971584
## 13 1750 meter buffer   -2.963963          +                 0.3641566
## 14 5000 meter buffer   -2.943661          +                 0.3765220
## 15 1250 meter buffer   -2.968150          +                 0.3575414
```

```
## 16 1000 meter buffer   -2.980891           +                0.3499200
## 17  750 meter buffer   -2.988100           +                0.3899083
## 18  500 meter buffer   -2.988047           +                0.4346195
## 19 1500 meter buffer   -2.959700           +                0.3511111
## 20  250 meter buffer   -2.942422           +                0.2841211
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1               0.4626473              -0.18883198  5 -278.7345 567.7345
## 2               0.4208578              -0.26410640  5 -278.8745 568.0145
## 3               0.4539606              -0.20704398  5 -279.0308 568.3271
## 4               0.4321033              -0.24437884  5 -279.0621 568.3897
## 5               0.4861173              -0.11844450  5 -279.7041 569.6737
## 6               0.4082150              -0.24854140  5 -279.7154 569.6963
## 7               0.5042086              -0.07794105  5 -280.3559 570.9773
## 8               0.3841964              -0.22070440  5 -280.8247 571.9149
## 9               0.3683729              -0.24125958  5 -280.8461 571.9576
## 10              0.3701224              -0.22623898  5 -281.0735 572.4125
## 11              0.3672068              -0.21980574  5 -281.5093 573.2840
## 12              0.3680595              -0.19849247  5 -281.8277 573.9209
## 13              0.4814612              -0.04320923  5 -282.3687 575.0029
## 14              0.3686674              -0.16747161  5 -282.4228 575.1112
## 15              0.5177533               0.02741755  5 -282.5875 575.4406
## 16              0.5883085               0.07927383  5 -282.6993 575.6640
## 17              0.6379614               0.15948496  5 -282.7427 575.7509
## 18              0.6500935               0.24801315  5 -282.9647 576.1949
## 19              0.4675019              -0.00922415  5 -283.0227 576.3109
## 20              0.4274624               0.22361368  5 -288.8675 588.0004
##         delta       weight                  Dataset
## 1   0.0000000 2.052411e-01 osm_anthro_df_2021_2022
## 2   0.2799928 1.784286e-01 osm_anthro_df_2021_2022
## 3   0.5926252 1.526080e-01 osm_anthro_df_2021_2022
## 4   0.6552383 1.479044e-01 osm_anthro_df_2021_2022
## 5   1.9392371 7.783309e-02 osm_anthro_df_2021_2022
## 6   1.9618019 7.695988e-02 osm_anthro_df_2021_2022
## 7   3.2428047 4.056002e-02 osm_anthro_df_2021_2022
## 8   4.1804337 2.538018e-02 osm_anthro_df_2021_2022
## 9   4.2231469 2.484389e-02 osm_anthro_df_2021_2022
## 10  4.6780181 1.978999e-02 osm_anthro_df_2021_2022
## 11  5.5495272 1.279970e-02 osm_anthro_df_2021_2022
## 12  6.1863565 9.309235e-03 osm_anthro_df_2021_2022
## 13  7.2684161 5.419357e-03 osm_anthro_df_2021_2022
## 14  7.3766708 5.133819e-03 osm_anthro_df_2021_2022
## 15  7.7060792 4.354220e-03 osm_anthro_df_2021_2022
## 16  7.9295393 3.893917e-03 osm_anthro_df_2021_2022
## 17  8.0163988 3.728425e-03 osm_anthro_df_2021_2022
## 18  8.4603649 2.986203e-03 osm_anthro_df_2021_2022
## 19  8.5764341 2.817833e-03 osm_anthro_df_2021_2022
## 20 20.2659042 8.157913e-06 osm_anthro_df_2021_2022
```

```
summary(fisher_land_mods$`2500 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(fisher, absent_fisher) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
```

```
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    567.5    584.7   -278.7    557.5      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.5454   0.7385
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -2.96838    0.32100  -9.247  < 2e-16 ***
## scale(lc_forest)     0.46265    0.14544   3.181  0.00147 **
## scale(lc_grassland) -0.18883    0.12061  -1.566  0.11745
## scale(lc_developed)  0.44625    0.09245   4.827 1.38e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Grey wolf

```
wolf_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(grey_wolf, absent_grey_wolf) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
wolf_land_model.sel <- model.sel(wolf_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
```

```
wolf_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1  3500 meter buffer   -3.046161           +                -0.27916703
## 2  3750 meter buffer   -3.046275           +                -0.30137364
## 3  4000 meter buffer   -3.042264           +                -0.33186520
## 4  3250 meter buffer   -3.034100           +                -0.26426890
## 5  4250 meter buffer   -3.033446           +                -0.33765275
## 6  4500 meter buffer   -3.027710           +                -0.33980293
## 7  3000 meter buffer   -3.021901           +                -0.23722348
## 8  4750 meter buffer   -3.019626           +                -0.30765699
## 9  5000 meter buffer   -3.015489           +                -0.28889918
## 10 2750 meter buffer   -3.011930           +                -0.20286706
## 11 2500 meter buffer   -2.998351           +                -0.14740111
## 12  500 meter buffer   -3.000899           +                -0.09243200
## 13 2250 meter buffer   -2.991719           +                -0.10049864
## 14  250 meter buffer   -2.990126           +                -0.14612407
## 15 2000 meter buffer   -2.988606           +                -0.05723996
## 16 1750 meter buffer   -2.986130           +                -0.01932916
## 17 1500 meter buffer   -2.984354           +                 0.01408029
## 18 1250 meter buffer   -2.982215           +                -0.02108745
## 19  750 meter buffer   -2.984944           +                -0.07707025
## 20 1000 meter buffer   -2.982631           +                -0.03963140
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1             0.005707166               0.353789595  5 -259.3789 529.0232
## 2            -0.014531404               0.348327601  5 -259.5091 529.2836
## 3            -0.025042871               0.332041018  5 -259.7174 529.7003
## 4             0.022662089               0.320367395  5 -260.1464 530.5584
## 5            -0.041091273               0.298933049  5 -260.3860 531.0375
## 6            -0.057995075               0.270972883  5 -260.8489 531.9632
## 7             0.021704146               0.276410841  5 -261.2570 532.7794
## 8            -0.064113125               0.240307685  5 -261.6511 533.5677
## 9            -0.079158920               0.216040190  5 -262.1049 534.4754
## 10            0.037407907               0.237442870  5 -262.1129 534.4912
## 11            0.055446663               0.172816949  5 -263.3431 536.9516
## 12           -0.137015483              -0.167191129  5 -263.6123 537.4901
## 13            0.088961442               0.133213377  5 -263.9000 538.0655
## 14           -0.090579964              -0.070318797  5 -264.2212 538.7079
## 15            0.105697539               0.108193337  5 -264.2396 538.7447
## 16            0.123668480               0.075176813  5 -264.4673 539.2000
## 17            0.116839723               0.029148982  5 -264.6449 539.5554
## 18            0.071713342               0.013028003  5 -264.7813 539.8281
## 19           -0.018946655               0.002585728  5 -264.8062 539.8779
## 20            0.035650149               0.003798323  5 -264.8428 539.9511
##         delta      weight                 Dataset
## 1   0.0000000 0.242663947 osm_anthro_df_2021_2022
## 2   0.2603872 0.213040859 osm_anthro_df_2021_2022
## 3   0.6770892 0.172972561 osm_anthro_df_2021_2022
## 4   1.5351417 0.112629838 osm_anthro_df_2021_2022
## 5   2.0142701 0.088636391 osm_anthro_df_2021_2022
## 6   2.9399753 0.055795315 osm_anthro_df_2021_2022
## 7   3.7562011 0.037098516 osm_anthro_df_2021_2022
## 8   4.5445063 0.025013718 osm_anthro_df_2021_2022
```

```
## 9   5.4521307 0.015888764 osm_anthro_df_2021_2022
## 10  5.4679641 0.015763474 osm_anthro_df_2021_2022
## 11  7.9283727 0.004606605 osm_anthro_df_2021_2022
## 12  8.4668785 0.003519216 osm_anthro_df_2021_2022
## 13  9.0422911 0.002639348 osm_anthro_df_2021_2022
## 14  9.6846391 0.001914311 osm_anthro_df_2021_2022
## 15  9.7214771 0.001879375 osm_anthro_df_2021_2022
## 16 10.1767862 0.001496733 osm_anthro_df_2021_2022
## 17 10.5321307 0.001253090 osm_anthro_df_2021_2022
## 18 10.8048964 0.001093331 osm_anthro_df_2021_2022
## 19 10.8546794 0.001066453 osm_anthro_df_2021_2022
## 20 10.9278230 0.001028155 osm_anthro_df_2021_2022
```

```
summary(wolf_land_mods$`3500 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(grey_wolf, absent_grey_wolf) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    528.8    546.0   -259.4    518.8      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.3526   0.5938
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -3.046161   0.267243 -11.398  < 2e-16 ***
## scale(lc_forest)     0.005707   0.149326   0.038  0.96951
## scale(lc_grassland)  0.353790   0.117829   3.003  0.00268 **
## scale(lc_developed) -0.279167   0.148037  -1.886  0.05932 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lynx

```
lynx_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(lynx, absent_lynx) ~
```

```
                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                    data = .,
                    family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
lynx_land_model.sel <- model.sel(lynx_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
lynx_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1   500 meter buffer   -1.859195           +               -0.10277645
## 2   750 meter buffer   -1.857541           +               -0.06674319
## 3   250 meter buffer   -1.856118           +               -0.12883312
## 4  1000 meter buffer   -1.855711           +               -0.04117640
## 5  1250 meter buffer   -1.854201           +                0.01610288
## 6  1500 meter buffer   -1.853651           +                0.04786925
## 7  4750 meter buffer   -1.851128           +                0.04169626
## 8  5000 meter buffer   -1.850956           +                0.03511441
## 9  3000 meter buffer   -1.854755           +                0.01805685
## 10 4500 meter buffer   -1.851382           +                0.03533651
## 11 3250 meter buffer   -1.854671           +                0.01480914
## 12 1750 meter buffer   -1.852403           +                0.05850045
## 13 2750 meter buffer   -1.854269           +                0.01892344
## 14 2000 meter buffer   -1.852213           +                0.05903098
## 15 3500 meter buffer   -1.853872           +                0.02250146
## 16 3750 meter buffer   -1.852866           +                0.03749073
## 17 4000 meter buffer   -1.852226           +                0.03489236
## 18 4250 meter buffer   -1.851847           +                0.02934343
## 19 2250 meter buffer   -1.851994           +                0.05816141
## 20 2500 meter buffer   -1.852701           +                0.03872636
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1           -0.0125606132              0.095236806  5 -452.4006 915.0666
## 2            0.0417150035              0.088067909  5 -453.1013 916.4681
## 3           -0.0552563579              0.005895648  5 -453.3180 916.9016
## 4            0.0430111755              0.078393893  5 -453.5708 917.4072
## 5            0.0536610540              0.070893539  5 -453.7861 917.8377
## 6            0.0516615485              0.051569724  5 -453.9317 918.1289
## 7           -0.0215377485             -0.090991719  5 -453.9398 918.1452
## 8           -0.0158485214             -0.092856771  5 -453.9734 918.2123
## 9           -0.0246544317              0.050075665  5 -454.1851 918.6357
```

```
## 10          -0.0272417603              -0.065995812  5 -454.2043 918.6740
## 11          -0.0321049253               0.045353702  5 -454.2210 918.7074
## 12           0.0488104989               0.011149529  5 -454.2696 918.8047
## 13          -0.0171477231               0.044975812  5 -454.2949 918.8553
## 14           0.0424713873               0.002747558  5 -454.3382 918.9419
## 15          -0.0371501581               0.023662900  5 -454.3396 918.9447
## 16          -0.0357463085              -0.006762491  5 -454.3770 919.0196
## 17          -0.0346749204              -0.025899560  5 -454.3956 919.0567
## 18          -0.0315343454              -0.035689015  5 -454.4167 919.0989
## 19           0.0232694641              -0.006766036  5 -454.4480 919.1614
## 20          -0.0002094453               0.014422878  5 -454.5261 919.3178
##       delta     weight                  Dataset
## 1  0.000000 0.20670727 osm_anthro_df_2021_2022
## 2  1.401436 0.10257410 osm_anthro_df_2021_2022
## 3  1.834957 0.08258474 osm_anthro_df_2021_2022
## 4  2.340569 0.06413686 osm_anthro_df_2021_2022
## 5  2.771080 0.05171581 osm_anthro_df_2021_2022
## 6  3.062316 0.04470768 osm_anthro_df_2021_2022
## 7  3.078553 0.04434621 osm_anthro_df_2021_2022
## 8  3.145664 0.04288282 osm_anthro_df_2021_2022
## 9  3.569069 0.03470102 osm_anthro_df_2021_2022
## 10 3.607410 0.03404212 osm_anthro_df_2021_2022
## 11 3.640773 0.03347897 osm_anthro_df_2021_2022
## 12 3.738109 0.03188862 osm_anthro_df_2021_2022
## 13 3.788661 0.03109270 osm_anthro_df_2021_2022
## 14 3.875328 0.02977412 osm_anthro_df_2021_2022
## 15 3.878118 0.02973261 osm_anthro_df_2021_2022
## 16 3.952939 0.02864086 osm_anthro_df_2021_2022
## 17 3.990121 0.02811330 osm_anthro_df_2021_2022
## 18 4.032276 0.02752695 osm_anthro_df_2021_2022
## 19 4.094789 0.02667987 osm_anthro_df_2021_2022
## 20 4.251159 0.02467335 osm_anthro_df_2021_2022
```

```
summary(lynx_land_mods$`500 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(lynx, absent_lynx) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    914.8    932.0   -452.4    904.8      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.1629   0.4037
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)         -1.85919    0.17480 -10.636   <2e-16 ***
```

```
## scale(lc_forest)    -0.01256    0.06974  -0.180    0.857
## scale(lc_grassland)  0.09524    0.05849   1.628    0.103
## scale(lc_developed) -0.10278    0.08084  -1.271    0.204
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Moose

```
moose_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(moose, absent_moose) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
moose_land_model.sel <- model.sel(moose_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
moose_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1   250 meter buffer   -1.824370          +               -0.38907200
## 2   500 meter buffer   -1.811440          +               -0.26517617
## 3  1250 meter buffer   -1.811622          +               -0.24854908
## 4  1000 meter buffer   -1.809467          +               -0.24500592
## 5   750 meter buffer   -1.807044          +               -0.22419410
## 6  1500 meter buffer   -1.807871          +               -0.20492713
## 7  1750 meter buffer   -1.807510          +               -0.20460227
## 8  2000 meter buffer   -1.803424          +               -0.17250345
## 9  2250 meter buffer   -1.802767          +               -0.17451218
## 10 5000 meter buffer   -1.794867          +               -0.03135968
## 11 2500 meter buffer   -1.801511          +               -0.16473681
## 12 2750 meter buffer   -1.800720          +               -0.15117920
```

```
## 13 4750 meter buffer   -1.794206           +               -0.04654219
## 14 3000 meter buffer   -1.798543           +               -0.13010136
## 15 4500 meter buffer   -1.794334           +               -0.06239861
## 16 4250 meter buffer   -1.794461           +               -0.06812523
## 17 4000 meter buffer   -1.794621           +               -0.07759938
## 18 3250 meter buffer   -1.797146           +               -0.11404443
## 19 3500 meter buffer   -1.796185           +               -0.10191078
## 20 3750 meter buffer   -1.795196           +               -0.08647021
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1           -0.1894191876              -0.084560017  5 -435.5390 881.3435
## 2           -0.1252092690               0.012836269  5 -441.1781 892.6216
## 3           -0.1302412537               0.045986762  5 -441.2800 892.8256
## 4           -0.0972965295               0.019964245  5 -441.8496 893.9646
## 5           -0.0569759026               0.032593961  5 -442.5540 895.3735
## 6           -0.1308807651               0.041118183  5 -442.5726 895.4107
## 7           -0.1225948081               0.048505398  5 -442.6991 895.6637
## 8           -0.0978660613               0.023709654  5 -443.9016 898.0687
## 9           -0.0896738480               0.012999835  5 -443.9410 898.1476
## 10           0.0229644406              -0.150138388  5 -444.2650 898.7955
## 11          -0.0705255139               0.020287163  5 -444.2935 898.8524
## 12          -0.0541213807               0.037062218  5 -444.6862 899.6378
## 13           0.0170187161              -0.108580891  5 -444.9070 900.0794
## 14          -0.0322270115               0.022790635  5 -445.1435 900.5525
## 15           0.0152642073              -0.078029954  5 -445.1668 900.5992
## 16           0.0144043611              -0.067810754  5 -445.2068 900.6790
## 17           0.0091484173              -0.049862667  5 -445.3283 900.9221
## 18          -0.0159558138               0.012893112  5 -445.3831 901.0317
## 19          -0.0070072556              -0.002757181  5 -445.4782 901.2219
## 20          -0.0006681693              -0.022075498  5 -445.5523 901.3701
##       delta       weight                  Dataset
## 1   0.00000 9.876445e-01 osm_anthro_df_2021_2022
## 2  11.27815 3.512226e-03 osm_anthro_df_2021_2022
## 3  11.48212 3.171684e-03 osm_anthro_df_2021_2022
## 4  12.62118 1.794512e-03 osm_anthro_df_2021_2022
## 5  14.03005 8.871843e-04 osm_anthro_df_2021_2022
## 6  14.06727 8.708273e-04 osm_anthro_df_2021_2022
## 7  14.32023 7.673662e-04 osm_anthro_df_2021_2022
## 8  16.72522 2.305506e-04 osm_anthro_df_2021_2022
## 9  16.80412 2.216323e-04 osm_anthro_df_2021_2022
## 10 17.45207 1.602990e-04 osm_anthro_df_2021_2022
## 11 17.50895 1.558050e-04 osm_anthro_df_2021_2022
## 12 18.29438 1.052028e-04 osm_anthro_df_2021_2022
## 13 18.73592 8.436207e-05 osm_anthro_df_2021_2022
## 14 19.20908 6.658884e-05 osm_anthro_df_2021_2022
## 15 19.25569 6.505515e-05 osm_anthro_df_2021_2022
## 16 19.33556 6.250832e-05 osm_anthro_df_2021_2022
## 17 19.57864 5.535454e-05 osm_anthro_df_2021_2022
## 18 19.68821 5.240351e-05 osm_anthro_df_2021_2022
## 19 19.87843 4.764907e-05 osm_anthro_df_2021_2022
## 20 20.02667 4.424501e-05 osm_anthro_df_2021_2022
```

```
summary(moose_land_mods$`250 meter buffer`)
```

```
##  Family: binomial  ( logit )
```

```
## Formula:
## cbind(moose, absent_moose) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    881.1    898.3   -435.5    871.1      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 0.2284   0.4779
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                      Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -1.82437    0.20458  -8.917  < 2e-16 ***
## scale(lc_forest)     -0.18942    0.07609  -2.489   0.0128 *
## scale(lc_grassland) -0.08456    0.06399  -1.321   0.1864
## scale(lc_developed) -0.38907    0.08615  -4.516  6.3e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Red fox

```
fox_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%

      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
     glmmTMB::glmmTMB(cbind(red_fox, absent_red_fox) ~

                        # Landscape classes that aren't correlated
                        scale(lc_forest) +
                        scale(lc_grassland) +
                        scale(lc_developed) +

                        # Random effect of array
                        (1|array),
                      data = .,
                      family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
fox_land_model.sel <- model.sel(fox_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%
```

```
  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
fox_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1  4750 meter buffer   -4.311701           +                 0.2865729
## 2  5000 meter buffer   -4.314446           +                 0.2946127
## 3  4500 meter buffer   -4.305607           +                 0.2758245
## 4  4250 meter buffer   -4.288722           +                 0.2665661
## 5  3750 meter buffer   -4.268422           +                 0.2491588
## 6  4000 meter buffer   -4.264669           +                 0.2688970
## 7  3500 meter buffer   -4.258079           +                 0.2469422
## 8  3250 meter buffer   -4.258905           +                 0.2281469
## 9  1500 meter buffer   -4.286422           +                 0.1720749
## 10 2750 meter buffer   -4.250647           +                 0.2308787
## 11 2500 meter buffer   -4.280069           +                 0.2042168
## 12 2250 meter buffer   -4.292690           +                 0.2029512
## 13 3000 meter buffer   -4.238552           +                 0.2446116
## 14 1750 meter buffer   -4.291311           +                 0.1625072
## 15 1250 meter buffer   -4.242640           +                 0.2204567
## 16 2000 meter buffer   -4.282952           +                 0.2067221
## 17 1000 meter buffer   -4.226176           +                 0.2698492
## 18  750 meter buffer   -4.197991           +                 0.3321050
## 19  500 meter buffer   -4.125760           +                 0.6055379
## 20  250 meter buffer   -4.125957           +                 0.4755105
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1             -0.47066300                0.4440924  5 -172.6120 355.4895
## 2             -0.49014388                0.4374597  5 -172.7404 355.7463
## 3             -0.46005814                0.4369906  5 -173.1568 356.5791
## 4             -0.43624853                0.4273371  5 -174.1172 358.4999
## 5             -0.31343664                0.4867189  5 -174.8190 359.9035
## 6             -0.37452035                0.4290864  5 -175.0954 360.4563
## 7             -0.28257620                0.4828959  5 -175.6674 361.6002
## 8             -0.24611112                0.5047879  5 -176.0282 362.3219
## 9             -0.14489943                0.5400933  5 -176.1631 362.5917
## 10            -0.25524006                0.4595279  5 -176.6450 363.5555
## 11            -0.21586397                0.5171330  5 -176.7956 363.8568
## 12            -0.17676341                0.5379829  5 -176.8160 363.8975
## 13            -0.24897156                0.4498939  5 -176.8312 363.9278
## 14            -0.18913438                0.5304592  5 -177.2477 364.7609
## 15            -0.14633021                0.4692529  5 -177.3753 365.0160
## 16            -0.17391068                0.5171511  5 -177.4997 365.2649
## 17            -0.11051461                0.4438864  5 -178.2805 366.8265
## 18             0.04517619                0.4519380  5 -178.4801 367.2256
## 19             0.25203544                0.3412639  5 -180.4476 371.1607
## 20             0.02398044                0.2553588  5 -181.3288 372.9231
##         delta       weight                 Dataset
## 1   0.0000000 3.248046e-01 osm_anthro_df_2021_2022
## 2   0.2567835 2.856685e-01 osm_anthro_df_2021_2022
## 3   1.0896523 1.883680e-01 osm_anthro_df_2021_2022
## 4   3.0103732 7.209878e-02 osm_anthro_df_2021_2022
## 5   4.4140392 3.573763e-02 osm_anthro_df_2021_2022
```

```
## 6   4.9668439 2.710727e-02 osm_anthro_df_2021_2022
## 7   6.1107532 1.529991e-02 osm_anthro_df_2021_2022
## 8   6.8323776 1.066572e-02 osm_anthro_df_2021_2022
## 9   7.1022529 9.319391e-03 osm_anthro_df_2021_2022
## 10  8.0660043 5.755878e-03 osm_anthro_df_2021_2022
## 11  8.3672793 4.950973e-03 osm_anthro_df_2021_2022
## 12  8.4079915 4.851209e-03 osm_anthro_df_2021_2022
## 13  8.4383115 4.778220e-03 osm_anthro_df_2021_2022
## 14  9.2713792 3.150415e-03 osm_anthro_df_2021_2022
## 15  9.5265670 2.773029e-03 osm_anthro_df_2021_2022
## 16  9.7753938 2.448625e-03 osm_anthro_df_2021_2022
## 17 11.3370387 1.121542e-03 osm_anthro_df_2021_2022
## 18 11.7361119 9.186664e-04 osm_anthro_df_2021_2022
## 19 15.6712359 1.284271e-04 osm_anthro_df_2021_2022
## 20 17.4335926 5.320662e-05 osm_anthro_df_2021_2022
```

```
summary(fox_land_mods$`4750 meter buffer`)
```

```
##  Family: binomial  ( logit )
## Formula:
## cbind(red_fox, absent_red_fox) ~ scale(lc_forest) + scale(lc_grassland) +
##     scale(lc_developed) + (1 | array)
## Data: .
##
##      AIC      BIC   logLik deviance df.resid
##    355.2    372.5   -172.6    345.2      227
##
## Random effects:
##
## Conditional model:
##  Groups Name        Variance Std.Dev.
##  array  (Intercept) 1.299    1.14
## Number of obs: 232, groups:  array, 6
##
## Conditional model:
##                     Estimate Std. Error z value Pr(>|z|)
## (Intercept)          -4.3117     0.5237  -8.233   <2e-16 ***
## scale(lc_forest)     -0.4707     0.1928  -2.441   0.0147 *
## scale(lc_grassland)   0.4441     0.1794   2.476   0.0133 *
## scale(lc_developed)   0.2866     0.1200   2.388   0.0170 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

White-tailed deer

```
deer_land_mods <- osm_landscape_df_2021_2022 %>%

  # use purrr map to run the same functions for all buffer sizes ((all objects in list))
  purrr::map(
    ~.x %>%
```

```
      # glmmTMB function let's us run the proportional binomial model using cbind to combine the presen
    glmmTMB::glmmTMB(cbind(`white-tailed_deer`, `absent_white-tailed_deer`) ~

                       # Landscape classes that aren't correlated
                       scale(lc_forest) +
                       scale(lc_grassland) +
                       scale(lc_developed) +

                       # Random effect of array
                       (1|array),
                     data = .,
                     family = 'binomial'))

# run model selection and save the results as a tibble for graphing use later
deer_land_model.sel <- model.sel(deer_land_mods) %>%

  as.data.frame() %>%

  rownames_to_column(var = 'Model')  %>%

  mutate(Dataset = deparse(substitute(osm_anthro_df_2021_2022)))

# look at model selection results
deer_land_model.sel
```

```
##                Model cond((Int)) disp((Int)) cond(scale(lc_developed))
## 1  4500 meter buffer  -0.1623813           +                0.20187598
## 2  4250 meter buffer  -0.1625427           +                0.19591693
## 3  4750 meter buffer  -0.1620244           +                0.22303896
## 4  4000 meter buffer  -0.1635226           +                0.19381588
## 5  5000 meter buffer  -0.1623011           +                0.23067503
## 6  3750 meter buffer  -0.1641042           +                0.19652830
## 7  3500 meter buffer  -0.1641705           +                0.19465875
## 8  3250 meter buffer  -0.1641689           +                0.19223412
## 9  3000 meter buffer  -0.1631662           +                0.19472744
## 10 2750 meter buffer  -0.1625679           +                0.19934091
## 11 2500 meter buffer  -0.1625756           +                0.19855991
## 12 2250 meter buffer  -0.1623364           +                0.20390809
## 13 2000 meter buffer  -0.1617604           +                0.19848624
## 14 1750 meter buffer  -0.1616216           +                0.18838780
## 15 1500 meter buffer  -0.1628144           +                0.16268304
## 16 1250 meter buffer  -0.1615885           +                0.16221466
## 17 1000 meter buffer  -0.1598991           +                0.16354671
## 18  750 meter buffer  -0.1614725           +                0.10971171
## 19  250 meter buffer  -0.1670141           +               -0.04939048
## 20  500 meter buffer  -0.1652200           +                0.01759524
##    cond(scale(lc_forest)) cond(scale(lc_grassland)) df    logLik     AICc
## 1             -0.20538963               0.37068831  5 -587.9308 1186.127
## 2             -0.19218942               0.36213244  5 -589.2947 1188.855
## 3             -0.21075061               0.34441889  5 -589.3299 1188.925
## 4             -0.17740529               0.35435141  5 -590.5906 1191.447
## 5             -0.22074357               0.31717359  5 -591.1398 1192.545
## 6             -0.16596432               0.34093999  5 -592.0095 1194.284
```

```
## 7            -0.15778893               0.32926474  5 -593.4363 1197.138
## 8            -0.15234248               0.32505954  5 -593.8628 1197.991
## 9            -0.14746745               0.30920697  5 -595.4702 1201.206
## 10           -0.14242312               0.28807224  5 -597.4595 1205.184
## 11           -0.13978800               0.27618068  5 -599.0260 1208.318
## 12           -0.13605774               0.25025524  5 -601.1461 1212.558
## 13           -0.12928230               0.23720599  5 -602.3431 1214.952
## 14           -0.12274787               0.22306219  5 -603.8423 1217.950
## 15           -0.12319161               0.21265628  5 -606.0623 1222.390
## 16           -0.10650333               0.19348297  5 -608.7951 1227.856
## 17           -0.07477042               0.14695476  5 -614.2262 1238.718
## 18           -0.08782437               0.12648591  5 -617.4000 1245.065
## 19           -0.18923240               0.03773632  5 -618.6593 1247.584
## 20           -0.13121110               0.08187522  5 -620.2648 1250.795
##        delta       weight                  Dataset
## 1   0.000000 6.108356e-01 osm_anthro_df_2021_2022
## 2   2.727884 1.561607e-01 osm_anthro_df_2021_2022
## 3   2.798154 1.507693e-01 osm_anthro_df_2021_2022
## 4   5.319680 4.273371e-02 osm_anthro_df_2021_2022
## 5   6.417984 2.467612e-02 osm_anthro_df_2021_2022
## 6   8.157393 1.034116e-02 osm_anthro_df_2021_2022
## 7  11.010977 2.482682e-03 osm_anthro_df_2021_2022
## 8  11.863979 1.620668e-03 osm_anthro_df_2021_2022
## 9  15.078879 3.247785e-04 osm_anthro_df_2021_2022
## 10 19.057402 4.442855e-05 osm_anthro_df_2021_2022
## 11 22.190518 9.275017e-06 osm_anthro_df_2021_2022
## 12 26.430608 1.113245e-06 osm_anthro_df_2021_2022
## 13 28.824693 3.362961e-07 osm_anthro_df_2021_2022
## 14 31.823070 7.509873e-08 osm_anthro_df_2021_2022
## 15 36.262932 8.156966e-09 osm_anthro_df_2021_2022
## 16 41.728719 5.304545e-10 osm_anthro_df_2021_2022
## 17 52.590774 2.322645e-12 osm_anthro_df_2021_2022
## 18 58.938417 9.718735e-14 osm_anthro_df_2021_2022
## 19 61.457022 2.758682e-14 osm_anthro_df_2021_2022
## 20 64.667930 5.539388e-15 osm_anthro_df_2021_2022
```

Save model results

We also want to save the output from the model.sel funciton for each species as one csv file for use plotting later like we did with the previous analyses

```
# provide list of model.sel data frames from global analysis

land_model.sel_data <- list(
  bear = bear_land_model.sel,
  caribou = caribou_land_model.sel,
  coyote = coyote_land_model.sel,
  fisher = fisher_land_model.sel,
  wolf = wolf_land_model.sel,
  lynx = lynx_land_model.sel,
  moose = moose_land_model.sel,
  fox = fox_land_model.sel,
  deer = deer_land_model.sel
```

```
) %>%

  # use purrr to combine data and extract species names to use as a column
  map_dfr(~.x %>%

            mutate(species = deparse(substitute(.x))),
          .id = "species")
```

And save this for use later

```
write_csv(land_model.sel_data,
          'data/processed/OSM_glm_land_model_sel_data.csv')
```

> In summary it does matter whether you are looking at landscape or anthropogenic features which spatial scale you use. Only two species (lynx and moose) had spatial scales for both top models within 1000m (1km). There were also more well-defined top models (delta AIC greater than 2.00) when you subset data to specifically look at anthro or landscape features.