Reproducible Research in R Workshop

Marissa Dyck

2025-05-16

Script last updated: ENTER DATE by ENTER NAME

# Before you begin

## Notes

A few notes about this script.

This is a mock script that walks through the steps of analyzing mammal data using generalized linear models. It uses slightly modified data from a real study on brown bear livestock predation in Romania that was published in Conservation Science and Practice.

[Mihai I. Pop, Marissa A. Dyck, Silviu Chiriac, Berde Lajos, Szilárd Szabó, Cristian I. Iojă, Viorel D. Popescu. (2023). Predictors of brown bear predation events on livestock in the Romanian Carpathians](https://conbio.onlinelibrary.wiley.com/doi/full/10.1111/csp2.12884)

This code is derived from course materials developed by Dr. Marissa A. Dyck for undergraduate & graduate coursework at the University of Victoria, as well as international R workshops. The course is free and available online through [Dr. Dyck’s GitHub](https://marissadyck.github.io/R-crash-course.github.io/). You may find this a helpful resource for further developing your coding knowledge and skills beyond what we can cover in this workshop.

If you have question please email the author,

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

## R and RStudio

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

You can download R and RStudio [HERE](https://posit.co/download/rstudio-desktop/)

## 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](https://www.rstudio.com/wp-content/uploads/2015/02/rmarkdown-cheatsheet.pdf)

## 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('MuMIn')  
install.packages('car')  
install.packages('lme4')  
install.packages('PerformanceAnalytics')  
install.packages('broom')

## Load libraries

Then load the packages to your library so they are available for use during this current R session. I have this chode chunk set to message=FALSE so that my knitted doc doesn’t print all the info about each library that is normally printed in the console.

library(tidyverse) # for data formatting, cleaning, and much more!  
library(PerformanceAnalytics) # for generating correlation matrix plots   
library(lme4) # for fitting glms  
library(car) # companion package for glm analysis with additional functions  
library(MuMIn) # for model selection  
library(broom) # extracting odds ratios in a tidy format

# Data

## README

As previously mentioned, this data is a slightly modified version of data associated with [Pop et al., 2023](https://conbio.onlinelibrary.wiley.com/doi/full/10.1111/csp2.12884). There is a published GitHub repository on Dr. Dyck’s GitHub with the final data and analysis scripts from the publication if you are interested.

[Brown Bear Predation GitHub repository](https://github.com/marissadyck/Brown_bear_predation_RO)

Although the data is slightly different, the README that was published with the final analysis should serve as a good enough reference for the data we are using if you want more information about the data collected.

[Brown Bear README](data/bear_README.html)

## Import data

This code will read in the data as a tibble and save it to the environment with a descriptive and tidy name - this is essential for well organized reproducible research to avoid errors with coding. Avoid naming your data files as ‘data’ or ‘dat’ for example, because as your workflow gets more complex you may be importing several datasets for a single project or you may be working on several projects at a time in one R session - if all your data are named very similarly or the exact same thing you can easily reference the wrong data set.

In the same code chunk we will also do a bit of data tidying that I consider the standard now for all of my analyses to ease coding, reduce errors, and increase reproduce-ability.

* First, we will set all the column names to lowercase - this reduces keystrokes and possible case sensitive errors while coding
* Then we will specify how each of the variables should be read in (e.g. factor, numeric, etc.). This will also reduce potential errors later in the process as R often misinterprets how to read in data. You should always be familiar enough with your data before beginning any analysis to complete this section (i.e. you should know what each column is, how it was measured, and ideally what format you need it to be in for your analysis - some variables can be coded in several ways which are all correct depending on what you are doing with the data). **This is wear README files come in: You have to familiarize yourself with the data before beginning any analysis, and to have truly reproducible research when you publish or share your analysis you will need to have a thorough explanation of your data for someone else. This is all included in a README file and I recommend starting one at this phase in your process if not sooner!**
* Lastly, we will check the structure of the data and make sure things read in properly, and make any changes to this code if necessary

# read in the bear data and do some data tidying  
bear\_damage <- read\_csv('data/raw/pagube\_2008\_2016\_spatial.csv',  
   
 # specify how the columns are read in  
 col\_types = cols(Damage = col\_factor(),  
 Year = col\_factor(),  
 Month = col\_factor(),  
 Targetspp = col\_factor(),  
 Landcover\_code = col\_factor(),  
 .default = col\_number())) %>%   
   
 # set all column names to lowercase  
 rename\_with(tolower)  
  
# check the internal structure of the data   
str(bear\_damage)

## spc\_tbl\_ [3,024 × 24] (S3: spec\_tbl\_df/tbl\_df/tbl/data.frame)  
## $ damage : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 2 ...  
## $ year : Factor w/ 9 levels "2016","2009",..: 1 2 2 3 4 2 3 4 2 5 ...  
## $ month : Factor w/ 12 levels "0","7","9","8",..: 1 1 1 1 1 1 2 1 1 3 ...  
## $ targetspp : Factor w/ 3 levels "alte","ovine",..: NA NA NA NA NA NA 1 NA NA 2 ...  
## $ point\_x : num [1:3024] 489008 491321 491398 491758 492628 ...  
## $ point\_y : num [1:3024] 532778 542207 538028 536946 533069 ...  
## $ bear\_abund : num [1:3024] 0 0 26 26 0 26 26 26 26 26 ...  
## $ landcover\_code : Factor w/ 15 levels "311","112","231",..: 1 1 1 1 1 1 2 3 4 5 ...  
## $ altitude : num [1:3024] 549 596 506 485 530 551 437 527 467 561 ...  
## $ human\_population: num [1:3024] 0 0 54 32 0 0 229 0 0 0 ...  
## $ dist\_to\_forest : num [1:3024] 0 0 0 0 0 ...  
## $ dist\_to\_town : num [1:3024] 1558 2281.6 387.8 60.6 2076.3 ...  
## $ livestock\_killed: num [1:3024] 0 0 0 0 0 1 1 0 1 1 ...  
## $ shannondivindex : num [1:3024] 1.083 0.692 0.908 1.555 0.81 ...  
## $ prop\_arable : num [1:3024] 0 0 0 14.1 4.7 ...  
## $ prop\_orchards : num [1:3024] 0 0 0 0 0 0 0 0 0 0 ...  
## $ prop\_pasture : num [1:3024] 25.2 20.2 42.7 25.4 70.1 ...  
## $ prop\_ag\_mosaic : num [1:3024] 0 0 0 0 0 ...  
## $ prop\_seminatural: num [1:3024] 9.427 1.668 0.166 22.149 2.001 ...  
## $ prop\_deciduous : num [1:3024] 59.4 75.7 50.6 27.4 23.2 ...  
## $ prop\_coniferous : num [1:3024] 0 0 0 0 0 0 0 0 0 0 ...  
## $ prop\_mixedforest: num [1:3024] 0 0 0 0 0 0 0 0 0 0 ...  
## $ prop\_grassland : num [1:3024] 0 2.4 0.26 0 0 ...  
## $ prop\_for\_regen : num [1:3024] 4.17 0 0 0 0 ...  
## - attr(\*, "spec")=  
## .. cols(  
## .. .default = col\_number(),  
## .. Damage = col\_factor(levels = NULL, ordered = FALSE, include\_na = FALSE),  
## .. Year = col\_factor(levels = NULL, ordered = FALSE, include\_na = FALSE),  
## .. Month = col\_factor(levels = NULL, ordered = FALSE, include\_na = FALSE),  
## .. Targetspp = col\_factor(levels = NULL, ordered = FALSE, include\_na = FALSE),  
## .. POINT\_X = col\_number(),  
## .. POINT\_Y = col\_number(),  
## .. Bear\_abund = col\_number(),  
## .. Landcover\_code = col\_factor(levels = NULL, ordered = FALSE, include\_na = FALSE),  
## .. Altitude = col\_number(),  
## .. Human\_population = col\_number(),  
## .. Dist\_to\_forest = col\_number(),  
## .. Dist\_to\_town = col\_number(),  
## .. Livestock\_killed = col\_number(),  
## .. ShannonDivIndex = col\_number(),  
## .. prop\_arable = col\_number(),  
## .. prop\_orchards = col\_number(),  
## .. prop\_pasture = col\_number(),  
## .. prop\_ag\_mosaic = col\_number(),  
## .. prop\_seminatural = col\_number(),  
## .. prop\_deciduous = col\_number(),  
## .. prop\_coniferous = col\_number(),  
## .. prop\_mixedforest = col\_number(),  
## .. prop\_grassland = col\_number(),  
## .. prop\_for\_regen = col\_number()  
## .. )  
## - attr(\*, "problems")=<externalptr>

## Data checks and cleaning

Now we will do some mock data checks and data cleaning. This doesn’t necessarily reflect what you would need to do with this exact data but provides some examples of things to check and gives you practice with different code.

### Years

First we will check that the data represents all the years for the study and no years are missing and there isn’t data from any years outside the study timeframe.

From the README file we know that we should have data for years 2008-2016

# check that year is correct  
summary(bear\_damage$year)

## 2016 2009 2014 2012 2013 2011 2010 2015 2008   
## 348 408 396 620 392 160 140 324 236

*You’ll notice it isn’t ordered because we read it in as a factor not a numeric variable there are ways to fix this but I will leave that to you to explore*

### Months

We will also check that all the for months looks correct and is entered properly. We should have 1-12 with 1 being January and 12 December, we will also have zeros from the pseudoabsence data generated for this dataset (see README)

# check that month is correct   
summary(bear\_damage$month)

## 0 7 9 8 5 10 6 4 3 11 12 2   
## 2268 139 160 155 84 50 105 27 8 20 7 1

You’ll notice after checking that there aren’t any 1’s, that is because brown bears are hibernating during this period and thus there were no records of damage for January. These are important details to know, check, and make note of in your code and README for reproducible science!

## Filter data

Much of the spatial data for this dataset are represented as proportion (e.g., proportion of different types of habitat on the landscape). If we expect that our proportional data should all sum to 100 we can check that for each site and remove any sites (rows) that don’t as a data cleaning step. The code below will do this.

We also may be interested in filtering out observations where a lot of animals were involved (likely these are smaller livestock such as chickens etc.), if we only want to use data where a certain number of livestock were killed we can do that data cleaning step here as well.

We will assign a new data set to the environment for this step so we can compare with the old data

# create new data with prop\_check column and filter out observations that don't sum to 100  
bear\_damage\_tidy <- bear\_damage %>%   
   
 # create a column that sums across rows of spatial data  
 mutate(prop\_check = rowSums(across(contains('prop')))) %>%   
   
 # filter to 100 and only livestock events with 10 or fewer animals  
 filter(prop\_check == 100 &  
 livestock\_killed <= 10)   
  
# check new data  
summary(bear\_damage\_tidy)

## damage year month targetspp point\_x   
## 0:922 2012 :245 0 :922 alte : 16 Min. :491321   
## 1:198 2013 :160 9 : 42 ovine : 39 1st Qu.:547710   
## 2014 :142 6 : 38 bovine:143 Median :569859   
## 2015 :136 8 : 34 NA's :922 Mean :575292   
## 2009 :132 7 : 31 3rd Qu.:603439   
## 2016 :122 5 : 21 Max. :658416   
## (Other):183 (Other): 32   
## point\_y bear\_abund landcover\_code altitude   
## Min. :447121 Min. : 0.00 311 :261 Min. : 255.0   
## 1st Qu.:489899 1st Qu.:19.00 312 :254 1st Qu.: 697.0   
## Median :519150 Median :31.00 313 :159 Median : 915.0   
## Mean :524433 Mean :30.24 321 :135 Mean : 917.8   
## 3rd Qu.:552545 3rd Qu.:42.00 231 :129 3rd Qu.:1124.0   
## Max. :628579 Max. :77.00 211 : 80 Max. :1707.0   
## (Other):102   
## human\_population dist\_to\_forest dist\_to\_town livestock\_killed  
## Min. : 0.000 Min. : 0.00 Min. : 0 Min. :0.0000   
## 1st Qu.: 0.000 1st Qu.: 0.00 1st Qu.: 1790 1st Qu.:0.0000   
## Median : 0.000 Median : 0.00 Median : 2941 Median :0.0000   
## Mean : 2.546 Mean : 253.24 Mean : 3561 Mean :0.5634   
## 3rd Qu.: 0.000 3rd Qu.: 87.58 3rd Qu.: 4770 3rd Qu.:1.0000   
## Max. :369.000 Max. :7073.95 Max. :13140 Max. :7.0000   
##   
## shannondivindex prop\_arable prop\_orchards prop\_pasture   
## Min. :0.0000 Min. : 0.000 Min. :0 Min. : 0.00   
## 1st Qu.:0.5107 1st Qu.: 0.000 1st Qu.:0 1st Qu.: 0.00   
## Median :0.7850 Median : 0.000 Median :0 Median : 0.00   
## Mean :0.7633 Mean : 7.002 Mean :0 Mean : 8.88   
## 3rd Qu.:1.0451 3rd Qu.: 0.000 3rd Qu.:0 3rd Qu.:11.23   
## Max. :1.8951 Max. :100.000 Max. :0 Max. :96.11   
##   
## prop\_ag\_mosaic prop\_seminatural prop\_deciduous prop\_coniferous   
## Min. : 0.0000 Min. : 0.000 Min. : 0.00 Min. : 0.000   
## 1st Qu.: 0.0000 1st Qu.: 0.000 1st Qu.: 0.00 1st Qu.: 0.000   
## Median : 0.0000 Median : 0.000 Median : 3.13 Median : 6.774   
## Mean : 0.8042 Mean : 1.753 Mean : 27.40 Mean : 24.417   
## 3rd Qu.: 0.0000 3rd Qu.: 0.000 3rd Qu.: 56.59 3rd Qu.: 45.989   
## Max. :42.9740 Max. :44.391 Max. :100.00 Max. :100.000   
##   
## prop\_mixedforest prop\_grassland prop\_for\_regen prop\_check   
## Min. : 0.00 Min. : 0.000 Min. : 0.000 Min. :100   
## 1st Qu.: 0.00 1st Qu.: 0.000 1st Qu.: 0.000 1st Qu.:100   
## Median : 0.00 Median : 0.000 Median : 0.000 Median :100   
## Mean : 15.26 Mean : 8.732 Mean : 5.758 Mean :100   
## 3rd Qu.: 22.11 3rd Qu.:13.614 3rd Qu.: 8.162 3rd Qu.:100   
## Max. :100.00 Max. :86.540 Max. :61.234 Max. :100   
##

## Remove old data

Now, if we aren’t going to use the old version of the data any further, we should remove it from our environment. Keeping our environment clean helps ensure we don’t accidentally use the wrong data and also makes it organized and easier to inspect objects if we need.

# remove old data  
rm(bear\_damage)

# Summary statistics

Often times for publications, reports, or other deliverables we need to provide some summary information about the raw data we collected. Besides that, this is a great way to begin to explore your data prior to conducting any formal analyses.

## Total events

First we will calculate the total number of predation *events*, for this dataset, that is anything in the damage column that is coded as a one where zeros represent pseudoabsence events. We will also calculate the total number of livestock killed across all events.

Remember we are using the ‘tidy’ dataset which doesn’t include all of the raw data. Depending on your needs you may want to use the messier raw data to report your summary stats

# total number of events & total number of livestock killed  
  
# with summary we can look at the number of events (1s in the damage column)  
summary(bear\_damage\_tidy$damage)

## 0 1   
## 922 198

# or with summarise we can calculate both   
bear\_damage\_tidy %>%   
   
 # ensure to only count events of damage  
 filter(damage == '1') %>%   
   
 summarise(n\_events = n(),  
 total\_killed = sum(livestock\_killed))

## # A tibble: 1 × 2  
## n\_events total\_killed  
## <int> <dbl>  
## 1 198 470

## Events per livestock type

In this data there are several types of livestock that are affected, we may want to report specifics about each type of livestock. For that we can group our data before calculating some summary info.

# damage per livestock type (target species)  
bear\_damage\_tidy %>%   
   
 # ensure to only count events of damage not pseudoabsences  
 filter(damage == '1') %>%   
   
 # group by targetspp to get summaries for each species and year  
 group\_by(targetspp) %>%   
   
 # calculate total number of events (n)  
 summarise(n = n())

## # A tibble: 3 × 2  
## targetspp n  
## <fct> <int>  
## 1 alte 16  
## 2 ovine 39  
## 3 bovine 143

# bovine highest, alte lowest

## Events per year

This data was also collected over several years, so we may want to report some summary statistics for each year

# damage per year  
bear\_damage\_tidy %>%   
   
 # ensure to only count events of damage  
 filter(damage == '1') %>%   
   
 # group by targetspp to get summaries for each species and year  
 group\_by(year) %>%   
   
 # calculate total number of events (n)  
 summarise(n = n()) %>%   
   
 # sort by largest to smallest number of events  
 arrange(desc(n))

## # A tibble: 9 × 2  
## year n  
## <fct> <int>  
## 1 2012 42  
## 2 2013 34  
## 3 2014 28  
## 4 2016 26  
## 5 2015 25  
## 6 2009 12  
## 7 2011 11  
## 8 2010 11  
## 9 2008 9

# 2012 had highest number of events and 2008 had lowest number of events

## Events per month

We may also be interested in the monthly number of events

# damage per month  
bear\_damage\_tidy %>%   
   
 # ensure to only count events of damage  
 filter(damage == '1') %>%   
   
 # group by targetspp to get summaries for each species and year  
 group\_by(month) %>%   
   
 # calculate total number of events (n)  
 summarise(n = n()) %>%   
   
 # sort by largest to smallest number of events  
 arrange(desc(n))

## # A tibble: 10 × 2  
## month n  
## <fct> <int>  
## 1 9 42  
## 2 6 38  
## 3 8 34  
## 4 7 31  
## 5 5 21  
## 6 10 14  
## 7 4 12  
## 8 11 4  
## 9 3 1  
## 10 2 1

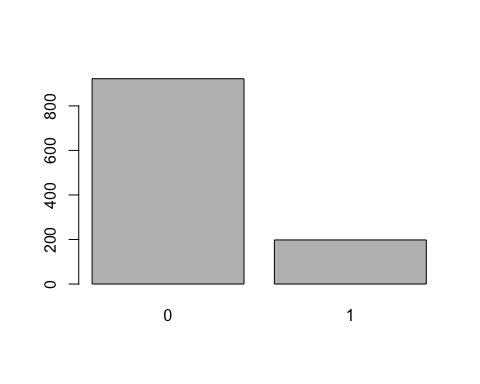
# september had the highest number of events and Dec/Jan had lowest with 0

# Analysis prep

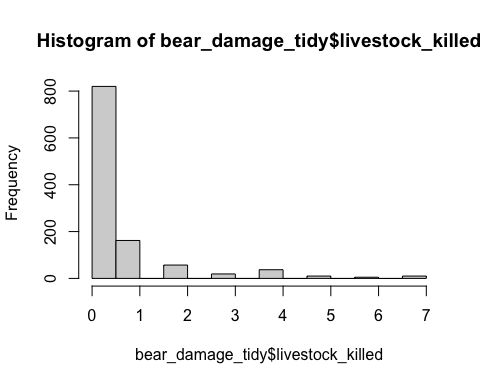
## Determine distribution

We are going to analyze this data in a generalized linear model framework, first we will take a look at our potential response variable/s and determine which to use if there are more than one option and which distribution is appropriate for our model based on the response variable

# damage is on epossible response variable and the appropriate distribution is binomial as it is 0/1 data  
plot(bear\_damage\_tidy$damage)



# or livestock killed is response variable which is count data so could be poisson or if highly zero-inflated causes overdispersion then negative binomial  
hist(bear\_damage\_tidy$livestock\_killed)



# lots of zeros lets do a quick test for dispersion with a simple glm  
test\_glm <-   
glm(livestock\_killed ~ bear\_abund,  
 data = bear\_damage\_tidy,  
 family = 'poisson')  
  
summary(test\_glm)

##   
## Call:  
## glm(formula = livestock\_killed ~ bear\_abund, family = "poisson",   
## data = bear\_damage\_tidy)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.047516 0.087835 -11.926 < 2e-16 \*\*\*  
## bear\_abund 0.014560 0.002248 6.477 9.34e-11 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for poisson family taken to be 1)  
##   
## Null deviance: 1958.6 on 1119 degrees of freedom  
## Residual deviance: 1916.5 on 1118 degrees of freedom  
## AIC: 2662.3  
##   
## Number of Fisher Scoring iterations: 6

# calculate dispersion which is residual deviance / degrees of freedom  
1916.5/1118

## [1] 1.714222

# 1.74 is high so over-dispersed - use negative binomial

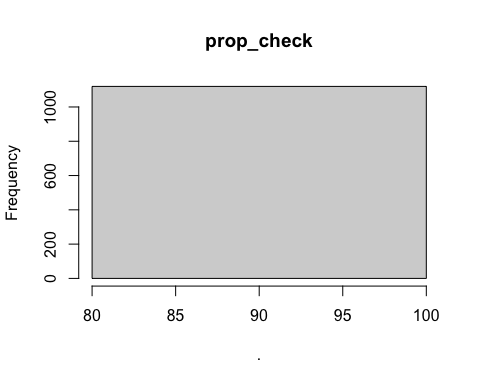
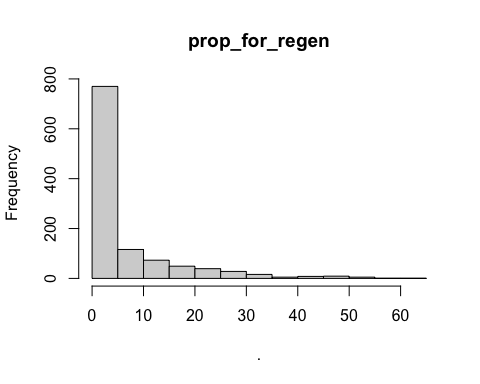
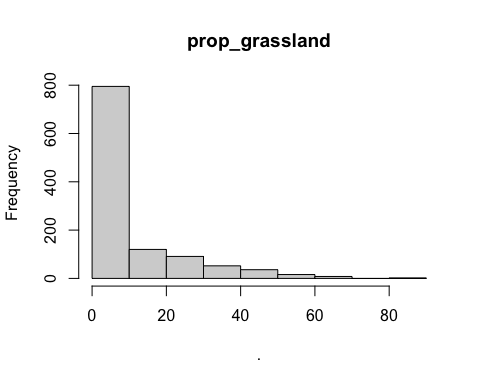
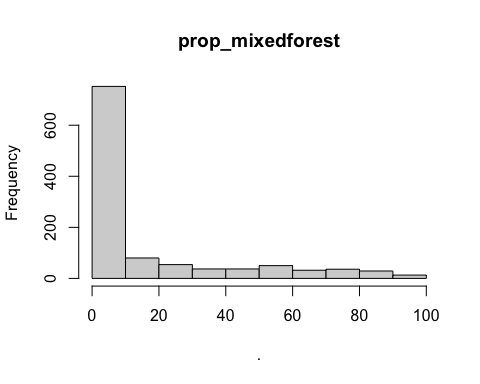
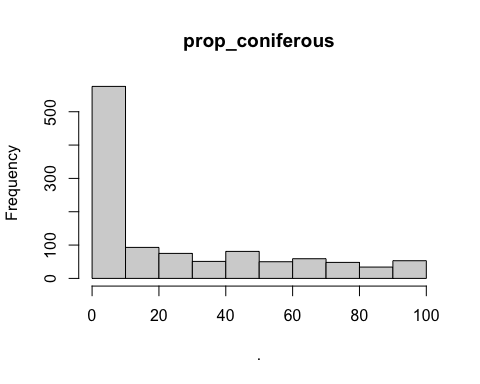
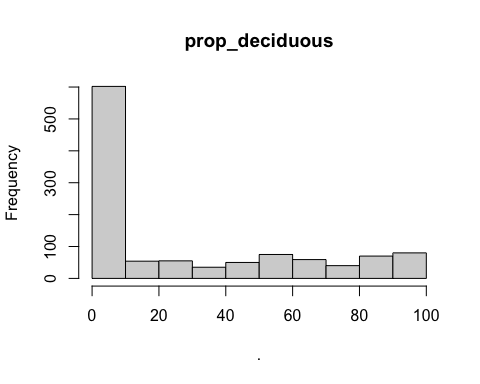
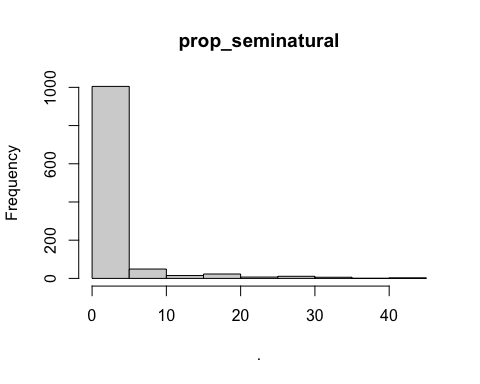
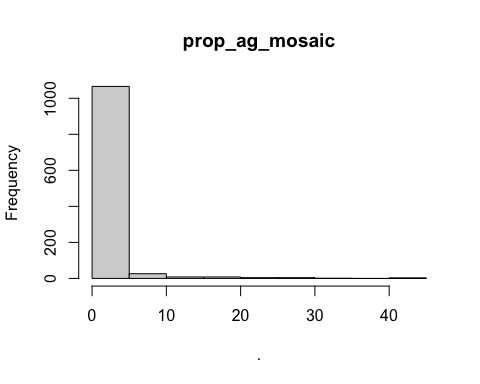
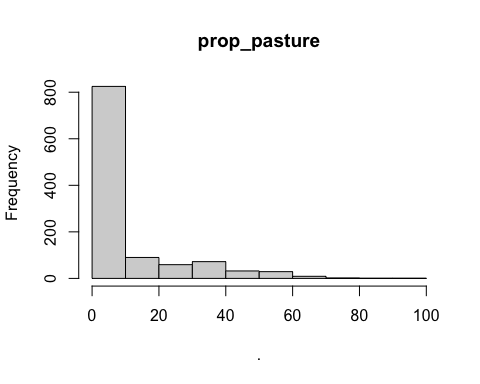
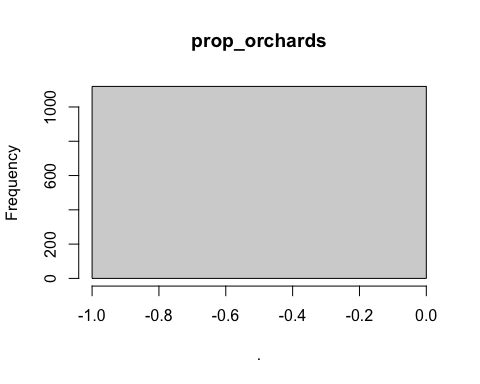
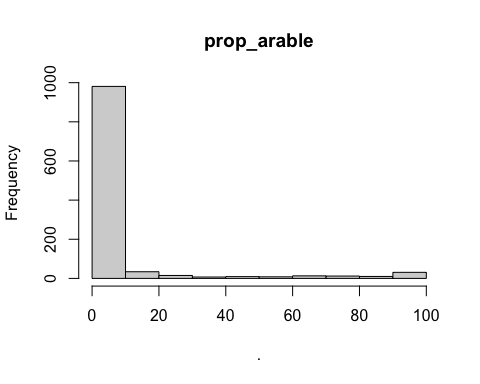
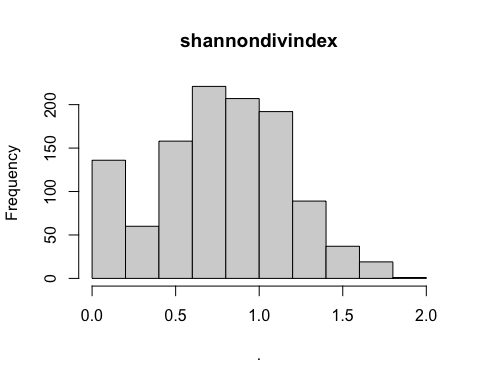
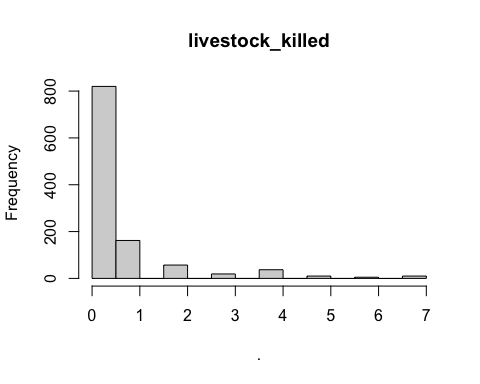
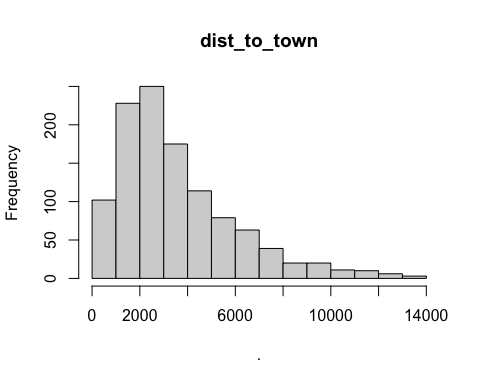
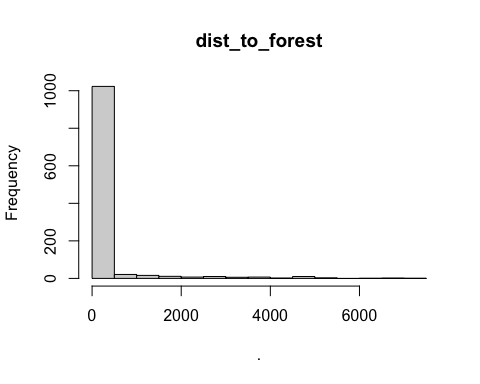
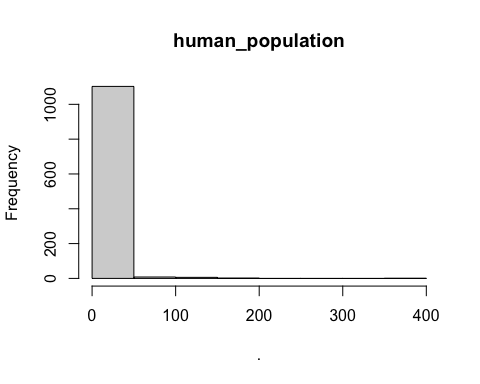
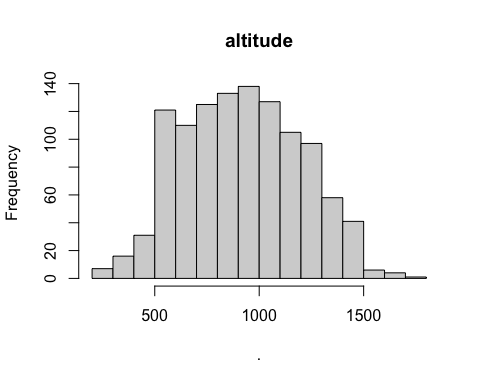
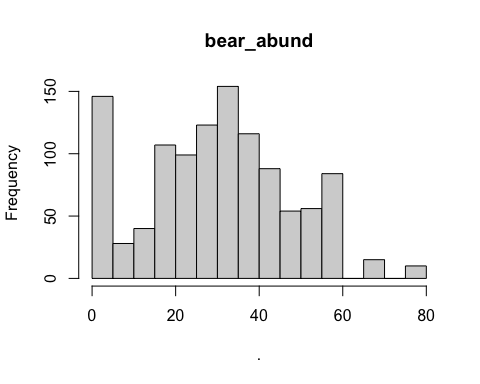
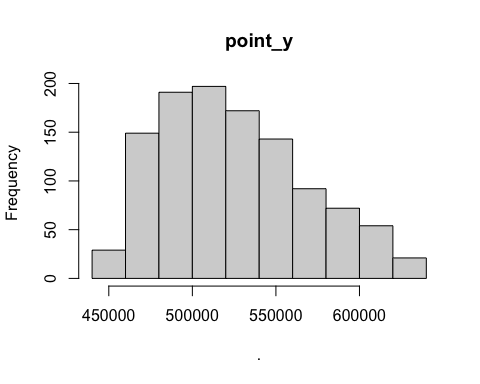
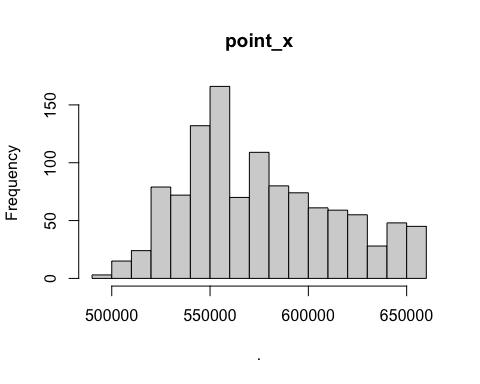
## Explanatory variable data exploration

Before we can run an analysis we need to do some exploration of our explanatory variables as well.

First we will plot histograms of each of our potential variables to insepct the data for issues and ensure there is enough variability to use each of our variables if we want

We could plot each with its own line of code or we could run an iteration using a handy tidyverse package called purrr; which is what the code below will do

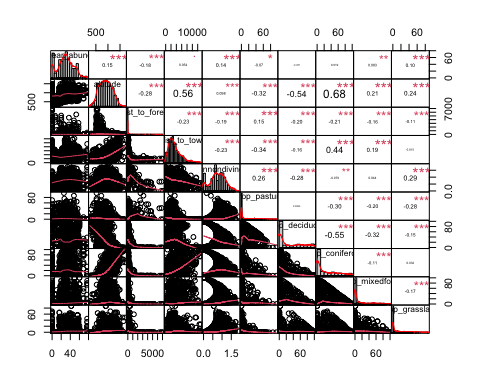
# using purr to generate histograms of each expl. variable  
bear\_damage\_tidy %>%   
   
 # select only numeric variables   
 select\_if(is.numeric) %>%   
   
 # use imap which will retain both the data (x) and the variable names (y)  
 imap(~.x %>%   
   
 # use the hist function on the data from previous pipe  
 hist(.,  
   
 # set the main title to y (each variable)  
 main = .y))



## $point\_x  
## $breaks  
## [1] 490000 500000 510000 520000 530000 540000 550000 560000 570000 580000  
## [11] 590000 600000 610000 620000 630000 640000 650000 660000  
##   
## $counts  
## [1] 3 15 24 79 72 132 166 70 109 80 74 61 59 55 28 48 45  
##   
## $density  
## [1] 2.678571e-07 1.339286e-06 2.142857e-06 7.053571e-06 6.428571e-06  
## [6] 1.178571e-05 1.482143e-05 6.250000e-06 9.732143e-06 7.142857e-06  
## [11] 6.607143e-06 5.446429e-06 5.267857e-06 4.910714e-06 2.500000e-06  
## [16] 4.285714e-06 4.017857e-06  
##   
## $mids  
## [1] 495000 505000 515000 525000 535000 545000 555000 565000 575000 585000  
## [11] 595000 605000 615000 625000 635000 645000 655000  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $point\_y  
## $breaks  
## [1] 440000 460000 480000 500000 520000 540000 560000 580000 600000 620000  
## [11] 640000  
##   
## $counts  
## [1] 29 149 191 197 172 143 92 72 54 21  
##   
## $density  
## [1] 1.294643e-06 6.651786e-06 8.526786e-06 8.794643e-06 7.678571e-06  
## [6] 6.383929e-06 4.107143e-06 3.214286e-06 2.410714e-06 9.375000e-07  
##   
## $mids  
## [1] 450000 470000 490000 510000 530000 550000 570000 590000 610000 630000  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $bear\_abund  
## $breaks  
## [1] 0 5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80  
##   
## $counts  
## [1] 146 28 40 107 99 123 154 116 88 54 56 84 0 15 0 10  
##   
## $density  
## [1] 0.026071429 0.005000000 0.007142857 0.019107143 0.017678571 0.021964286  
## [7] 0.027500000 0.020714286 0.015714286 0.009642857 0.010000000 0.015000000  
## [13] 0.000000000 0.002678571 0.000000000 0.001785714  
##   
## $mids  
## [1] 2.5 7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5 57.5 62.5 67.5 72.5  
## [16] 77.5  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $altitude  
## $breaks  
## [1] 200 300 400 500 600 700 800 900 1000 1100 1200 1300 1400 1500 1600  
## [16] 1700 1800  
##   
## $counts  
## [1] 7 16 31 121 110 125 133 138 127 105 97 58 41 6 4 1  
##   
## $density  
## [1] 6.250000e-05 1.428571e-04 2.767857e-04 1.080357e-03 9.821429e-04  
## [6] 1.116071e-03 1.187500e-03 1.232143e-03 1.133929e-03 9.375000e-04  
## [11] 8.660714e-04 5.178571e-04 3.660714e-04 5.357143e-05 3.571429e-05  
## [16] 8.928571e-06  
##   
## $mids  
## [1] 250 350 450 550 650 750 850 950 1050 1150 1250 1350 1450 1550 1650  
## [16] 1750  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $human\_population  
## $breaks  
## [1] 0 50 100 150 200 250 300 350 400  
##   
## $counts  
## [1] 1103 8 6 2 0 0 0 1  
##   
## $density  
## [1] 1.969643e-02 1.428571e-04 1.071429e-04 3.571429e-05 0.000000e+00  
## [6] 0.000000e+00 0.000000e+00 1.785714e-05  
##   
## $mids  
## [1] 25 75 125 175 225 275 325 375  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $dist\_to\_forest  
## $breaks  
## [1] 0 500 1000 1500 2000 2500 3000 3500 4000 4500 5000 5500 6000 6500 7000  
## [16] 7500  
##   
## $counts  
## [1] 1023 21 16 11 7 10 6 7 2 10 3 0 1 2 1  
##   
## $density  
## [1] 1.826786e-03 3.750000e-05 2.857143e-05 1.964286e-05 1.250000e-05  
## [6] 1.785714e-05 1.071429e-05 1.250000e-05 3.571429e-06 1.785714e-05  
## [11] 5.357143e-06 0.000000e+00 1.785714e-06 3.571429e-06 1.785714e-06  
##   
## $mids  
## [1] 250 750 1250 1750 2250 2750 3250 3750 4250 4750 5250 5750 6250 6750 7250  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $dist\_to\_town  
## $breaks  
## [1] 0 1000 2000 3000 4000 5000 6000 7000 8000 9000 10000 11000  
## [13] 12000 13000 14000  
##   
## $counts  
## [1] 102 228 250 175 114 79 63 39 20 20 11 10 6 3  
##   
## $density  
## [1] 9.107143e-05 2.035714e-04 2.232143e-04 1.562500e-04 1.017857e-04  
## [6] 7.053571e-05 5.625000e-05 3.482143e-05 1.785714e-05 1.785714e-05  
## [11] 9.821429e-06 8.928571e-06 5.357143e-06 2.678571e-06  
##   
## $mids  
## [1] 500 1500 2500 3500 4500 5500 6500 7500 8500 9500 10500 11500  
## [13] 12500 13500  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $livestock\_killed  
## $breaks  
## [1] 0.0 0.5 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0 5.5 6.0 6.5 7.0  
##   
## $counts  
## [1] 820 162 0 57 0 19 0 37 0 10 0 5 0 10  
##   
## $density  
## [1] 1.464285714 0.289285714 0.000000000 0.101785714 0.000000000 0.033928571  
## [7] 0.000000000 0.066071429 0.000000000 0.017857143 0.000000000 0.008928571  
## [13] 0.000000000 0.017857143  
##   
## $mids  
## [1] 0.25 0.75 1.25 1.75 2.25 2.75 3.25 3.75 4.25 4.75 5.25 5.75 6.25 6.75  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $shannondivindex  
## $breaks  
## [1] 0.0 0.2 0.4 0.6 0.8 1.0 1.2 1.4 1.6 1.8 2.0  
##   
## $counts  
## [1] 136 60 158 221 207 192 89 37 19 1  
##   
## $density  
## [1] 0.607142857 0.267857143 0.705357143 0.986607143 0.924107143 0.857142857  
## [7] 0.397321429 0.165178571 0.084821429 0.004464286  
##   
## $mids  
## [1] 0.1 0.3 0.5 0.7 0.9 1.1 1.3 1.5 1.7 1.9  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_arable  
## $breaks  
## [1] 0 10 20 30 40 50 60 70 80 90 100  
##   
## $counts  
## [1] 981 34 15 7 9 8 13 12 10 31  
##   
## $density  
## [1] 0.0875892857 0.0030357143 0.0013392857 0.0006250000 0.0008035714  
## [6] 0.0007142857 0.0011607143 0.0010714286 0.0008928571 0.0027678571  
##   
## $mids  
## [1] 5 15 25 35 45 55 65 75 85 95  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_orchards  
## $breaks  
## [1] -1 0  
##   
## $counts  
## [1] 1120  
##   
## $density  
## [1] 1  
##   
## $mids  
## [1] -0.5  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_pasture  
## $breaks  
## [1] 0 10 20 30 40 50 60 70 80 90 100  
##   
## $counts  
## [1] 825 90 59 72 32 29 9 2 1 1  
##   
## $density  
## [1] 7.366071e-02 8.035714e-03 5.267857e-03 6.428571e-03 2.857143e-03  
## [6] 2.589286e-03 8.035714e-04 1.785714e-04 8.928571e-05 8.928571e-05  
##   
## $mids  
## [1] 5 15 25 35 45 55 65 75 85 95  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_ag\_mosaic  
## $breaks  
## [1] 0 5 10 15 20 25 30 35 40 45  
##   
## $counts  
## [1] 1066 26 8 8 4 4 1 0 3  
##   
## $density  
## [1] 0.1903571429 0.0046428571 0.0014285714 0.0014285714 0.0007142857  
## [6] 0.0007142857 0.0001785714 0.0000000000 0.0005357143  
##   
## $mids  
## [1] 2.5 7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_seminatural  
## $breaks  
## [1] 0 5 10 15 20 25 30 35 40 45  
##   
## $counts  
## [1] 1005 49 15 23 7 11 6 1 3  
##   
## $density  
## [1] 0.1794642857 0.0087500000 0.0026785714 0.0041071429 0.0012500000  
## [6] 0.0019642857 0.0010714286 0.0001785714 0.0005357143  
##   
## $mids  
## [1] 2.5 7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_deciduous  
## $breaks  
## [1] 0 10 20 30 40 50 60 70 80 90 100  
##   
## $counts  
## [1] 602 54 55 35 50 75 59 40 70 80  
##   
## $density  
## [1] 0.053750000 0.004821429 0.004910714 0.003125000 0.004464286 0.006696429  
## [7] 0.005267857 0.003571429 0.006250000 0.007142857  
##   
## $mids  
## [1] 5 15 25 35 45 55 65 75 85 95  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_coniferous  
## $breaks  
## [1] 0 10 20 30 40 50 60 70 80 90 100  
##   
## $counts  
## [1] 576 93 75 51 81 50 59 48 34 53  
##   
## $density  
## [1] 0.051428571 0.008303571 0.006696429 0.004553571 0.007232143 0.004464286  
## [7] 0.005267857 0.004285714 0.003035714 0.004732143  
##   
## $mids  
## [1] 5 15 25 35 45 55 65 75 85 95  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_mixedforest  
## $breaks  
## [1] 0 10 20 30 40 50 60 70 80 90 100  
##   
## $counts  
## [1] 752 80 54 37 37 50 32 36 29 13  
##   
## $density  
## [1] 0.067142857 0.007142857 0.004821429 0.003303571 0.003303571 0.004464286  
## [7] 0.002857143 0.003214286 0.002589286 0.001160714  
##   
## $mids  
## [1] 5 15 25 35 45 55 65 75 85 95  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_grassland  
## $breaks  
## [1] 0 10 20 30 40 50 60 70 80 90  
##   
## $counts  
## [1] 795 120 91 52 36 16 8 0 2  
##   
## $density  
## [1] 0.0709821429 0.0107142857 0.0081250000 0.0046428571 0.0032142857  
## [6] 0.0014285714 0.0007142857 0.0000000000 0.0001785714  
##   
## $mids  
## [1] 5 15 25 35 45 55 65 75 85  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_for\_regen  
## $breaks  
## [1] 0 5 10 15 20 25 30 35 40 45 50 55 60 65  
##   
## $counts  
## [1] 770 116 73 49 39 28 16 5 8 9 5 1 1  
##   
## $density  
## [1] 0.1375000000 0.0207142857 0.0130357143 0.0087500000 0.0069642857  
## [6] 0.0050000000 0.0028571429 0.0008928571 0.0014285714 0.0016071429  
## [11] 0.0008928571 0.0001785714 0.0001785714  
##   
## $mids  
## [1] 2.5 7.5 12.5 17.5 22.5 27.5 32.5 37.5 42.5 47.5 52.5 57.5 62.5  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"  
##   
## $prop\_check  
## $breaks  
## [1] 80 100  
##   
## $counts  
## [1] 1120  
##   
## $density  
## [1] 0.05  
##   
## $mids  
## [1] 90  
##   
## $xname  
## [1] "."  
##   
## $equidist  
## [1] TRUE  
##   
## attr(,"class")  
## [1] "histogram"

Once we have looked at this we can drop any columns of data that aren’t usable or try to merge them with other variables if appropriate and check for multicolinearity between variables which is an assumption we need to meet for GLMs

bear\_damage\_tidy %>%   
   
 # choose specific variables of interest  
 select(bear\_abund,  
 altitude,  
 dist\_to\_forest,  
 dist\_to\_town,  
 shannondivindex,  
 prop\_pasture,  
 prop\_deciduous,  
 prop\_coniferous,  
 prop\_mixedforest,  
 prop\_grassland) %>%   
   
 chart.Correlation()

 Above the diagonal of this matrix shows us Pearson’s R correlation coefficient for each pairwise combination of our chosen explanatory variables, the diagonal has the name of each variable with a histogram of the raw data, and below the diagonal is a correlation plot of each pairwise combination of variables.

We don’t want to include any variables that are highly correlated (absolute value r > 0.6) in the same model, so we will write some notes here about any that violate this or are close

altitude, prop\_coniferous = 0.68  
altitude, dist\_to\_town = 0.56 prop\_deciduous, prop\_coniferous = -0.55 ( these are often inversely correlated in forested areas as they are the main two tree types so if there is a lot of one there’s less of the other, it’s not too high but if we know this ecological relationship exists we may want to be cautious about including them in the same model)

## Data formatting

Once we’ve explored both our explanatory and response variables, we may want to do some reformatting to our data. This can be for several reasons, if variables are correlated or lacking enough data to use one individually we may combine them if ecologically justified, etc.

Here we will create a new variable that combines all the forest cover data because we are interested in overall forest cover not the effect of specific forest types.

# formatting data to combine variables  
bear\_damage\_tidy <- bear\_damage\_tidy %>%   
   
 # add new column that groups all forest types   
 mutate(prop\_forest = rowSums(across(c(prop\_coniferous,   
 prop\_deciduous,  
 prop\_mixedforest))))  
  
  
# check new data  
summary(bear\_damage\_tidy$prop\_forest)

## Min. 1st Qu. Median Mean 3rd Qu. Max.   
## 0.00 48.67 73.72 67.07 93.29 100.00

# Analysis

## Fit GLMs

Here we will create a candidate set of models that represent hypotheses about what variables may explain our chosen response and fit these to a glm with the appropriate distribution. As this is a mock analysis and not a course on human-wildlife conflict with bears in eastern Europe we won’t spend a ton of time justifying these models we will just use a few as examples.

Note these are fit to a simple binomial distribution not a negative binomial distribution which is what we determined earlier would truly be the best-fit distribution

# going to scale data first for ease of coding  
bear\_damage\_tidy <- bear\_damage\_tidy %>%   
   
 # use mutate to change all numeric variables to scaled  
 mutate\_if(is.numeric,   
 scale)  
  
  
#I've done 3 models for demonstration  
  
# null model  
bear\_null <- glm(damage ~ 1,  
 data = bear\_damage\_tidy,  
 family = 'binomial')  
  
# interaction between distance to forest and distance to town (close to both town and forest would have high prob of damage)  
bear\_distance\_i <- glm(damage ~ dist\_to\_forest \* dist\_to\_town,  
 data = bear\_damage\_tidy,  
 family = 'binomial')  
  
# quick check that this model fit  
summary(bear\_distance\_i)

##   
## Call:  
## glm(formula = damage ~ dist\_to\_forest \* dist\_to\_town, family = "binomial",   
## data = bear\_damage\_tidy)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.1042 0.1433 -7.705 1.31e-14 \*\*\*  
## dist\_to\_forest 1.6345 0.4750 3.441 0.00058 \*\*\*  
## dist\_to\_town 0.2209 0.2179 1.014 0.31070   
## dist\_to\_forest:dist\_to\_town 3.5312 0.7857 4.494 6.99e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1044.92 on 1119 degrees of freedom  
## Residual deviance: 960.37 on 1116 degrees of freedom  
## AIC: 968.37  
##   
## Number of Fisher Scoring iterations: 7

# analog distance model w/o interaction  
bear\_distance <- glm(damage ~ dist\_to\_forest +  
 dist\_to\_town,  
 data = bear\_damage\_tidy,  
 family = 'binomial')  
  
# quick check that this model fit  
summary(bear\_distance)

##   
## Call:  
## glm(formula = damage ~ dist\_to\_forest + dist\_to\_town, family = "binomial",   
## data = bear\_damage\_tidy)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.69548 0.09292 -18.247 < 2e-16 \*\*\*  
## dist\_to\_forest -0.62093 0.18654 -3.329 0.000872 \*\*\*  
## dist\_to\_town -0.62435 0.10685 -5.843 5.12e-09 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1044.92 on 1119 degrees of freedom  
## Residual deviance: 991.73 on 1117 degrees of freedom  
## AIC: 997.73  
##   
## Number of Fisher Scoring iterations: 6

## Model selection

A common approach with GLMs is to use model selection and compare metrics like AIC to determine which set of explanatory variables best explains the observed data we collected.

We will do this using a function from the MuMIn package

# model selection on candidate set of models  
model.sel(bear\_null,  
 bear\_distance,  
 bear\_distance\_i)

## Model selection table   
## (Int) dst\_to\_frs dst\_to\_twn dst\_to\_frs:dst\_to\_twn df logLik  
## bear\_distance\_i -1.104 1.6340 0.2209 3.531 4 -480.183  
## bear\_distance -1.695 -0.6209 -0.6243 3 -495.865  
## bear\_null -1.538 1 -522.462  
## AICc delta weight  
## bear\_distance\_i 968.4 0.00 1  
## bear\_distance 997.8 29.35 0  
## bear\_null 1046.9 78.53 0  
## Models ranked by AICc(x)

Lowest AIC and highest model weight generally indicates the best fitting model. The MuMIn function will automatically sort your models in decreasing order of weight

Based on this our best-fit model is the one with our interaction term

## Check model assumptions

This section will vary depending on the response variable and other aspects of the data, models, etc.

For the best-fit model identified above we want to re-check assumption of independence with variance inflation factor (VIF), we also want to check for overdispersion, and any outliers

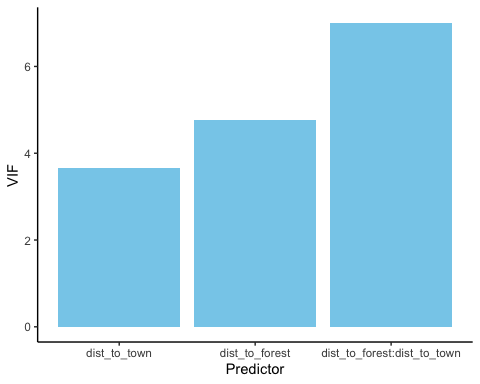
# check assumptions for top model  
  
vif(bear\_distance\_i)

## there are higher-order terms (interactions) in this model  
## consider setting type = 'predictor'; see ?vif

## dist\_to\_forest dist\_to\_town   
## 4.770770 3.652366   
## dist\_to\_forest:dist\_to\_town   
## 7.012544

# plot VIF  
vif(bear\_distance\_i) %>%  
   
 # Converts the named vector returned by vif() into a tidy tibble  
 enframe(name = 'Predictor',   
 value = 'VIF') %>%  
   
 # plot with ggplot  
 ggplot(aes(x = reorder(Predictor, VIF), # reorders from smallest VIF to largest   
 y = VIF)) +  
   
 # plot as bars  
 geom\_bar(stat = 'identity', fill = 'skyblue') +  
   
 # add labels  
 labs(x = 'Predictor',  
 y = 'VIF') +  
   
 # set theme  
 theme\_classic()

## there are higher-order terms (interactions) in this model  
## consider setting type = 'predictor'; see ?vif



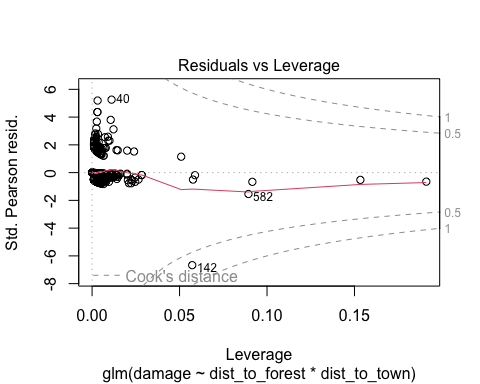
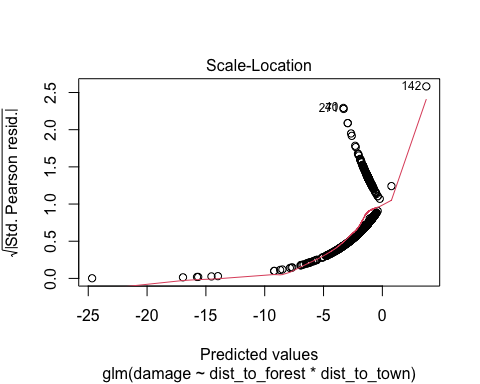
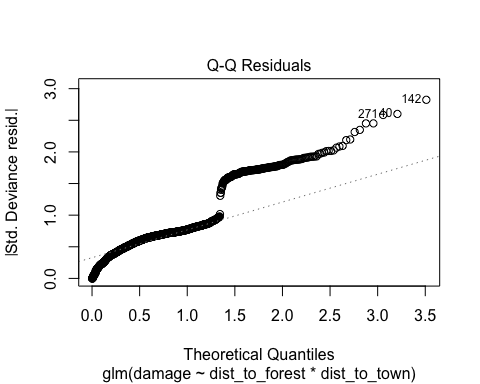
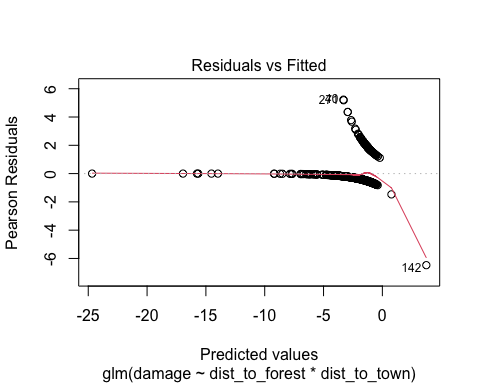
# dispersion  
summary(bear\_distance\_i)

##   
## Call:  
## glm(formula = damage ~ dist\_to\_forest \* dist\_to\_town, family = "binomial",   
## data = bear\_damage\_tidy)  
##   
## Coefficients:  
## Estimate Std. Error z value Pr(>|z|)   
## (Intercept) -1.1042 0.1433 -7.705 1.31e-14 \*\*\*  
## dist\_to\_forest 1.6345 0.4750 3.441 0.00058 \*\*\*  
## dist\_to\_town 0.2209 0.2179 1.014 0.31070   
## dist\_to\_forest:dist\_to\_town 3.5312 0.7857 4.494 6.99e-06 \*\*\*  
## ---  
## Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1  
##   
## (Dispersion parameter for binomial family taken to be 1)  
##   
## Null deviance: 1044.92 on 1119 degrees of freedom  
## Residual deviance: 960.37 on 1116 degrees of freedom  
## AIC: 968.37  
##   
## Number of Fisher Scoring iterations: 7

960.37/1116 # 0.86 slightly under dispersed but not a major issue for glm

## [1] 0.8605466

# check for observations with high leverage  
plot(bear\_distance\_i) # ignore first three



# Plot Results

There are several ways to plot results, for the purposes of this mock walk through we will plot odds ratios for the explanatory variables in our best-fit model. An odds ratio plot is a great way to show magnitude, confidence, and effect sizes for all your explanatory variables in one figure.

First we need to extract the odds ratios in a tidy format

# create a new data frame with the odds ratios  
model\_odds <-   
   
 # use the broom package to extract odds ratios into a tidy format  
broom::tidy(bear\_distance\_i,  
 exponentiate = TRUE,  
 conf.int = TRUE) %>%   
   
 # remove intercept and interaction term as we don't need to plot  
 filter(term %in% c('dist\_to\_forest',  
 'dist\_to\_town'))

## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred  
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred

# check data  
model\_odds

## # A tibble: 2 × 7  
## term estimate std.error statistic p.value conf.low conf.high  
## <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  
## 1 dist\_to\_forest 5.13 0.475 3.44 0.000580 2.08 13.2   
## 2 dist\_to\_town 1.25 0.218 1.01 0.311 0.829 1.94

Ignore warning for now as this is a mock analysis

Now we need to plot them in a visually pleasing and easily interpretable manner

# plot  
# specify data and mapping asesthetics  
ggplot(data = model\_odds,  
 aes(x = term,  
 y = estimate)) +  
   
 # add points for the odss  
 geom\_point() +  
   
 # add errorbars for the confidence intervals  
 geom\_errorbar(aes(ymin = conf.low,  
 ymax = conf.high),  
 linewidth = 0.5,  
 width = 0.4) +  
   
 geom\_hline(yintercept = 1,  
 alpha = 0.5) +  
   
 # rename the x axis labels  
 scale\_x\_discrete(labels = c('Distance to forest',  
 'Distance to town')) +  
   
 # rename y axis title  
 ylab('Odds ratio') +  
   
 # flip x and y axis   
 coord\_flip() +  
  
 # specify theme  
 theme\_bw() +  
   
 # specify theme elements  
 theme(panel.grid = element\_blank(),  
 axis.title.y = element\_blank())

