CLAW: Comprehensive Look at Animal Welfare

Emily Thompson, Casey Hansen

2024-09-15

Welcome to CLAW: Comprehensive Look at Animal Welfare

CLAW is an RMarkdown script designed to analyze veterinary data and create useful visualizations that help track and identify trends in animal health and welfare. Through exploratory data analysis, CLAW enables veterinarians, researchers, and animal welfare organizations to visualize complex datasets, uncover patterns, and make data-driven decisions.

This script is structured to process various types of veterinary data into clear, useful visualizations. CLAW contains the capability to produce:

- standard and stacked bar charts
- detailed monthly summaries
- heatmaps to investgiate covariance
- pie charts
- GIS graphicalanalysis

CLAW is designed to be flexible and adaptable, allowing users to input their own data and customize visual outputs based on specific needs. Our aim for this tool is to help veterinary medicine professionals and researchers to better identify health trends towards improving care.

To cite CLAW, please use

> Thompson, E., Hansen, C. (2024) CLAW (2.0) [Source Code]. https://github.com/casey-hansen/IMPACT

Importing Dataset

First, upload your data file to the CLAW environment. **Accepted file types are CSV, XLS, or XLSX.** This import also does some minor data cleaning, converting all characters to lower case for consistency, and converts date data to date values in R.

```
###########################################################
# Define data file, including path if necessary
file_name = "CLAW_benchmark.csv"
###########################################################

# Upload data to a dataframe based on file extension
# CSV files
if (get.ext(file_name) == "csv"){
  data.df <- read_csv(file_name,col_names=TRUE)
  #Convert any dates into date data format
  if (TRUE %in% str_detect(tolower(colnames(data.df)), "date")){
  idx <- which(str_detect(tolower(colnames(data.df)), "date"))
  data.df[,idx] <- as.Date(data.df[,idx][[1]],"%d-%b")
  }

# Excel files
} else if((get.ext(file_name) == "xlsx")||((get.ext(file_name) == "xls"))){
  data.df <- read_excel(file_name)
  #Convert any dates into date data format
  if (TRUE %in% str_detect(tolower(colnames(data.df)), "date")){
  idx <- which(str_detect(tolower(colnames(data.df)), "date"))
  data.df[,idx] <- read_excel(file_name,col_types='date')[,idx]
  }

} else{
  stop("I'm sorry that file is unsupported. CLAW accepts data in CSV, XLS, or XLSX format")
}
```

```
## Rows: 149 Columns: 18
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr (16): Location, Intake.Date, Intake.Location, County, Habitat, Sex, Age,...
## dbl  (2): ID.Number, Number.in.Loc
##
## 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.
```

Optional: Age Simplification (Dogs and Cats)

Some times, we aim to analyze age data in a precise manner. In other cases, it's easier to visualize data using the common age groups for dogs and cats. We define the age groups for Cats as follows:

- neonate: less than 12 weeks
- kitten: 13 weeks - 6 months
- teen: 7 months - 2 years
- adult: 3 years - 7 years
- senior: 8+ years

We define the age groups for dogs following Harvey's definition:

- puppy:0–6 months
- juvenile: 6 months to 1 year
- young adult: 1 year are fully grown
- mature adult: 2–6 years can be considered Mature Adults
- senior: 7+ years

Harvey N. D. (2021). How Old Is My Dog? Identification of Rational Age Groupings in Pet Dogs Based Upon Normative Age-Linked Processes. Frontiers in veterinary science, 8, 643085. https://doi.org/10.3389/fvets.2021.643085

Assumptions: This code chunk assumes that you have a column in your data relating to age. The code also assumes that when species is specified in the data, they are specified as 'dog' or 'cat'. Defining the `species` option as `"multi"` will check for this attribute.

```r
############################################################
# "cat" - all data is for cat species
# "dog" - all data is for dog species
# "multi" - data is for both dog and cat species
species <- "cat"
############################################################

if (TRUE %in% str_detect(tolower(colnames(data.df)), "species")){
  spec_idx <- which(str_detect(tolower(colnames(data.df)), "species"))[[1]]
}

if (TRUE %in% str_detect(tolower(colnames(data.df)), "age")){
  age_idx <- which(str_detect(tolower(colnames(data.df)), "age"))[[1]]
}

cat_ages <- list("neonate" = c(c(paste(as.character(c(1:7)),"days")),
                               c("1 week)"),
                               c(paste(as.character(c(2:12)),"weeks")),
                               c("1 month"),
                               c(paste(as.character(c(2:3)),"months"))),
                 "kitten" = c(c(paste(as.character(c(13:26)),"weeks")),
                              c(paste(as.character(c(4:6)),"months"))),
                 "teen" = c(c(paste(as.character(c(7:24)),"months")),
                            c("1 year", "2 years")),
                 "adult" = c(paste(as.character(c(2:7)),"years")),
                 "senior" = c(paste(as.character(c(8:30)),"years")))

dog_ages <- list("puppy" = c(c(paste(as.character(c(1:182)),"day(s)")),
                             c("1 week"),
                             c(paste(as.character(c(0:12)),"weeks")),
                             c("1 month"),
                             c(paste(as.character(c(2:6)),"months"))),
                 "juvenile" = c(c(paste(as.character(c(27:52)),"weeks")),
                                c(paste(as.character(c(6:12)),"months"))),
                 "young adult" = c(c(paste(as.character(c(13:24)),"months")),
                                   c(paste(as.character(c(1:2)),"years"))),
                 "mature adult" = c(paste(as.character(c(2:6)),"years")),
                 "senior" = c(paste(as.character(c(7:30)),"years")))


#Simplify cat ages
if (species == "cat"){
  for (x in 1:length(names(cat_ages))){
  data.df[data.df[,age_idx][[1]] %in% cat_ages[x][[1]],][,age_idx] <- names(cat_ages)[x]
  }

# Simplify dog ages
}else if (species == "dog"){
  for (x in 1:length(names(dog_ages))){
  data.df[data.df[,age_idx][[1]] %in% dog_ages[x][[1]],][,age_idx] <- names(dog_ages)[x]
  }

# Simplify cat and dog ages
}else{
  for (x in 1:length(names(cat_ages))){
  data.df[(data.df[,age_idx][[1]] %in% cat_ages[x][[1]]) &
            (data.df[,species_idx][[1]] == "cat"),][,age_idx] <- names(cat_ages)[x]
  }
  for (x in 1:length(names(dog_ages))){
  data.df[(data.df[,age_idx][[1]] %in% dog_ages[x][[1]]) &
            (data.df[,species_idx][[1]] == "dog"),][,age_idx] <- names(dog_ages)[x]
  }
}
```

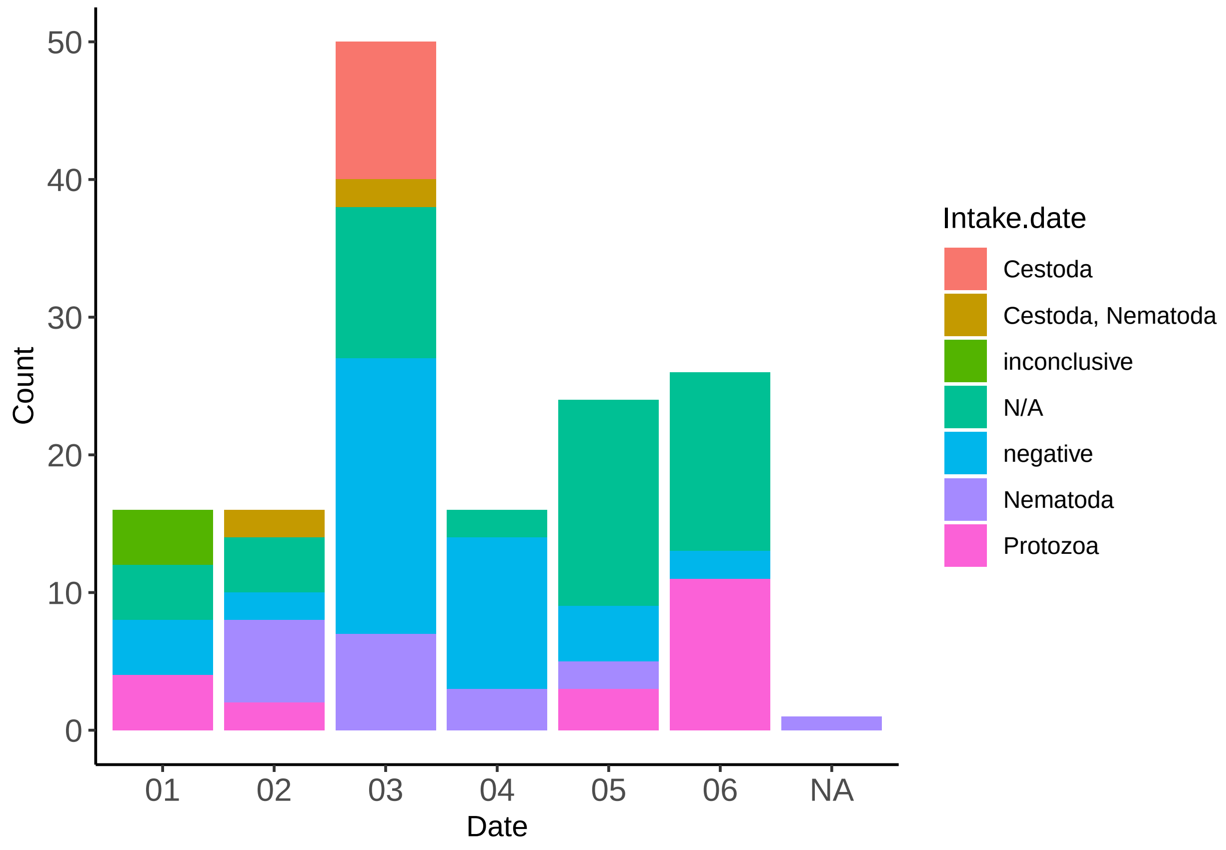
Bar Charts, Stacked or Otherwise

Stacked bar charts are useful for visualizing the breakdown of categorical data across different groups, comparing overall totals while showing the distribution of individual components within each category. This can help identify influencing factors in data trends.

The `categories` variable defines the column from which the categorical breakdown is created. To visualize a standard bar chart (i.e. a single category), set the value to be `categories <- "none"`

```
##########################################################
x_data <- "date"

# if categories <- "none", a simple bar graph is created
categories <- "fecal"
##########################################################


# Retrieve column for x_data
if (TRUE %in% str_detect(tolower(colnames(data.df)), x_data)){
  x_idx <- which(str_detect(tolower(colnames(data.df)), x_data))[[1]]
}else{
  print("I'm sorry - data not found. Please try again.")
}

# Make sure that dates are formatted correctly
if (x_data == "date"){
  aes_var <- format(data.df[,x_idx][[1]],"%m")
}else{
  aes_var <- data.df[,x_idx][[1]]
}

if (categories != "none"){
  cat_idx <- which(str_detect(tolower(colnames(data.df)), categories))[[1]]
  fill_var <- factor(data.df[,cat_idx][[1]])
} else{
  fill_var <- ""
}
```

```r
ggplot()+
  geom_bar(data = data.df, mapping = aes(aes_var,fill=fill_var))+
  theme_classic()+
  labs(
    x = str_to_title(x_data,),
    y = "Count",
    fill = str_to_title(colnames(data.df)[idx])
  )+
  theme(axis.text = element_text(size = 12))
```



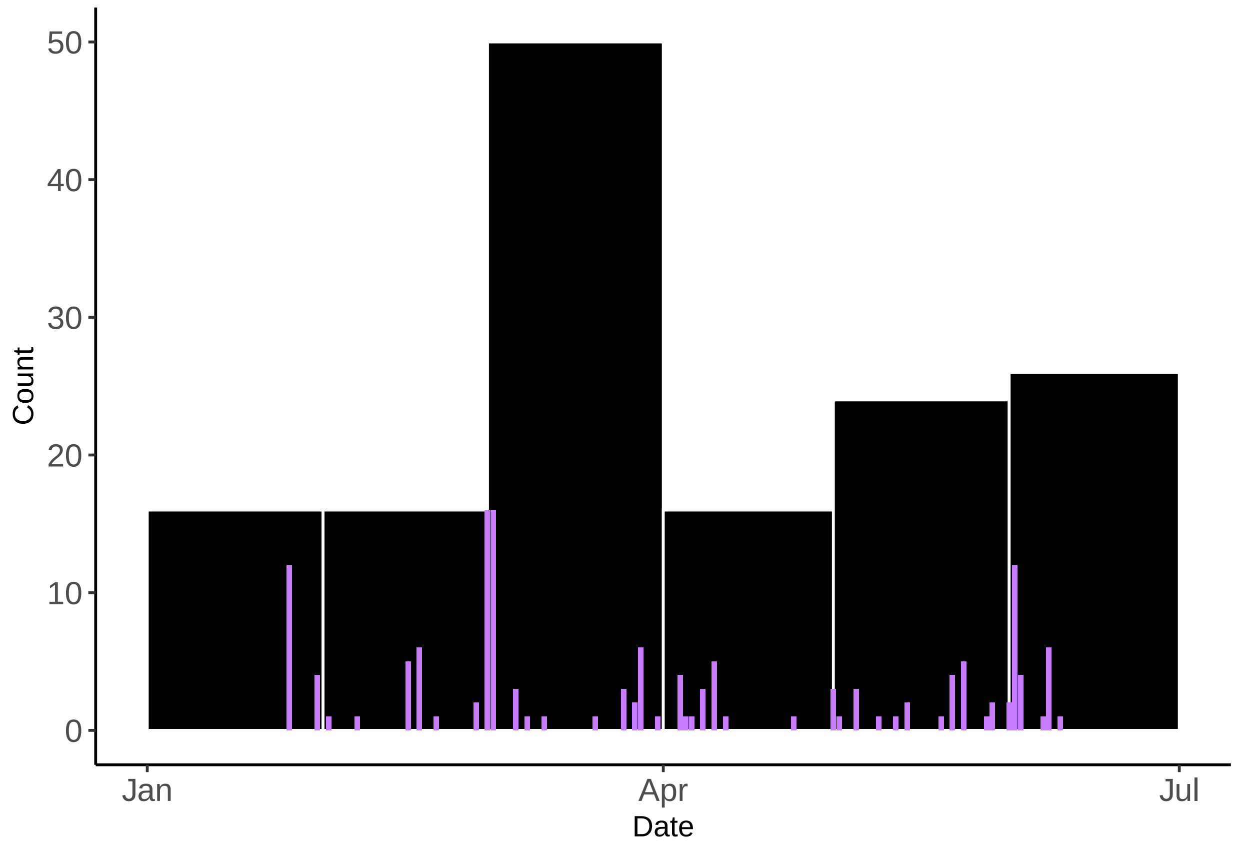
Timeline Summary by Month and Day

This graph shows not only long-term trends over time, but superimposed are the daily trends, to show how any sub-trends (for example - do people tend to surrender their pets at the beginning of the month, in the middle, or at the end?).

This code assumes there is a column related to date

```r
# Identify date column
date_idx <- which(str_detect(tolower(colnames(data.df)), "date"))[[1]]

# Rename date column for easier reference below
colnames(data.df)[date_idx] <- "Date"
```

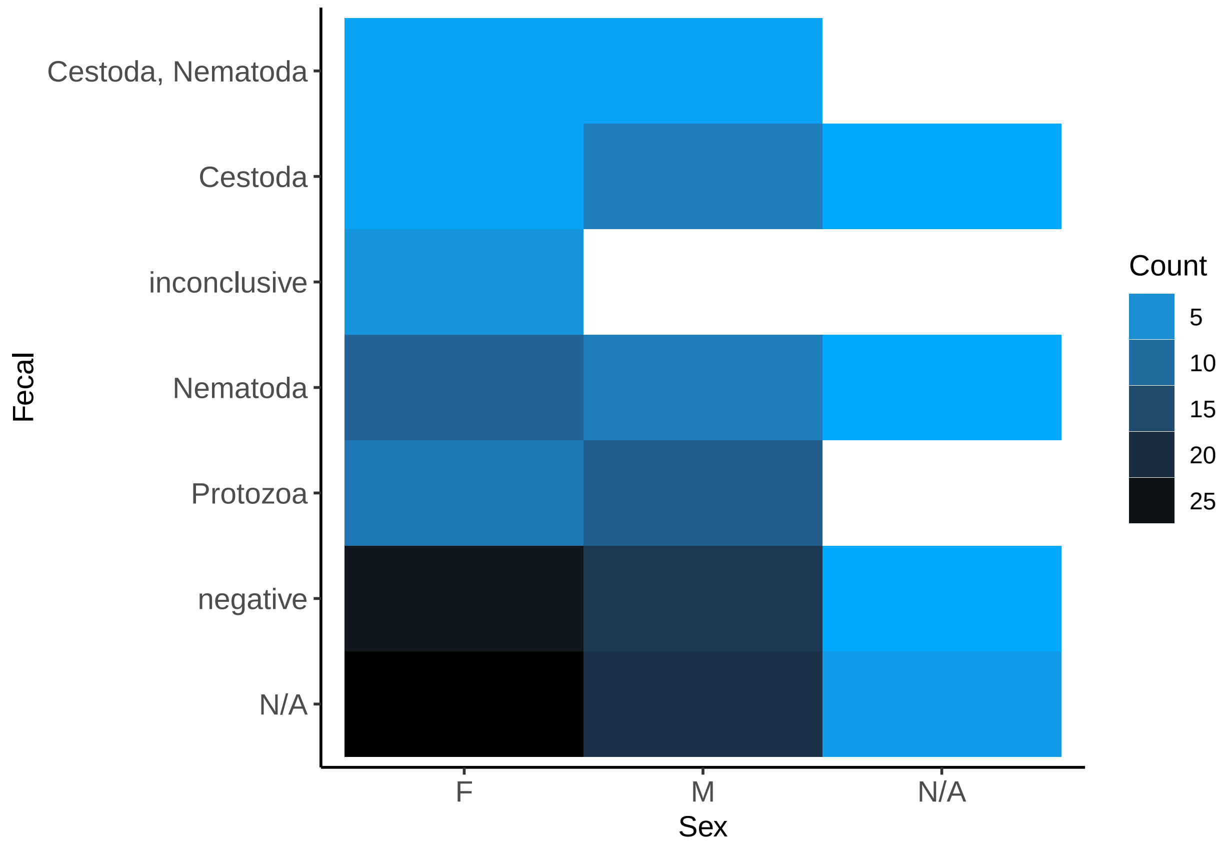
```r
# Force R to manually count the number of for each month
data.df_binned <- data.df |>
  mutate(
    month = format(Date, "%b")
  ) |>
  summarise(
    start = floor_date(min(Date), "month"),
    end = ceiling_date(max(Date), "month"),
    count = n(),
    .by = "month"
  )


#Plot
ggplot(data.df) +
  # Draw rectangles where the high is the number of cats surrendered each month
  geom_rect(data = data.df_binned,
            aes(xmin = start, xmax = end,ymin = 0, ymax = count),
            color = "white",fill = "black") +
  # Draw normal bar plot for intake
  geom_bar(aes(Date),fill = "#C77CFF" )+
  labs(x = "Date", y= "Count")+ theme_classic()+
  theme(axis.text = element_text(size = 12))
```



Heat Maps

Heat maps are commonly used with geospatial data or to represent relationship between variables. They are useful for “at a glance” data visualization, taking an initial investigation into covariance between variables, and for highlighting exceptional values (outliers).

```r
##########################################################
x_axis <- "sex"
y_axis <- "fecal"
##########################################################

# Check to make sure desired data exists
if (TRUE %in% str_detect(tolower(colnames(data.df)), x_axis) &
    TRUE %in% str_detect(tolower(colnames(data.df)), y_axis)){
  x_idx <- which(str_detect(tolower(colnames(data.df)), x_axis))[[1]]
  y_idx <- which(str_detect(tolower(colnames(data.df)), y_axis))[[1]]
}else{
  print("I'm sorry - data not found. Please try again.")
}

# Run a count of the covariance between variables
heat <- data.df %>% count(data.df[,x_idx], data.df[,y_idx])

# Plot the heatmap
ggplot(heat, aes(x = heat[,1][[1]], y = reorder(heat[,2][[1]], -n), fill = n)) +
  geom_tile() +
  theme_classic() +
  scale_fill_gradient(low = "#00A9FF", high = "black", guide = "legend") +
  labs(x = str_to_title(x_axis),
       y = str_to_title(y_axis),
       fill = "Count") +
  theme(axis.text = element_text(size = 11))
```

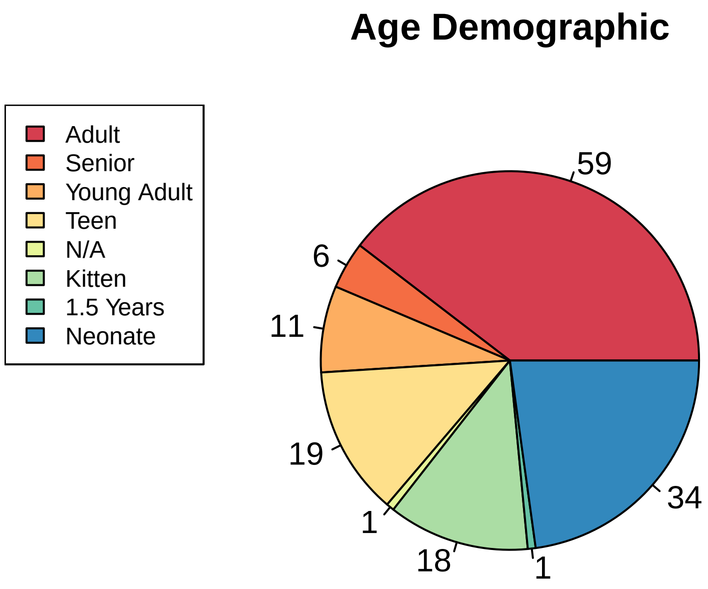


Demographic Pie Charts

While pie charts do have the potential to be misleading, they are a useful tool for quickly summarizing demographic breakdowns.

This code assumes you are visualizing no more than 11 distinct categories with your pie chart

```
##########################################################
demo <- "age"
##########################################################

d_idx <- which(str_detect(tolower(colnames(data.df)), demo))[[1]]

# Identify categories for pie chart
labs = unique(data.df[,d_idx][[1]])

# Count data for each category
value <- c()
for (each in labs){
  value <- append(value, c(colSums(sapply(each, `==`, data.df[,d_idx]))[[1]]))
}

# Create pie chart
pie(value,
    labels = value,
    col = brewer.pal(n = length(labs), name = "Spectral"),
    main="Age Demographic",
    cex=1)
legend("topleft", str_to_title(labs), cex=0.75, fill=brewer.pal(n = length(labs), name = "Spectral"))
```



Geospatial Map

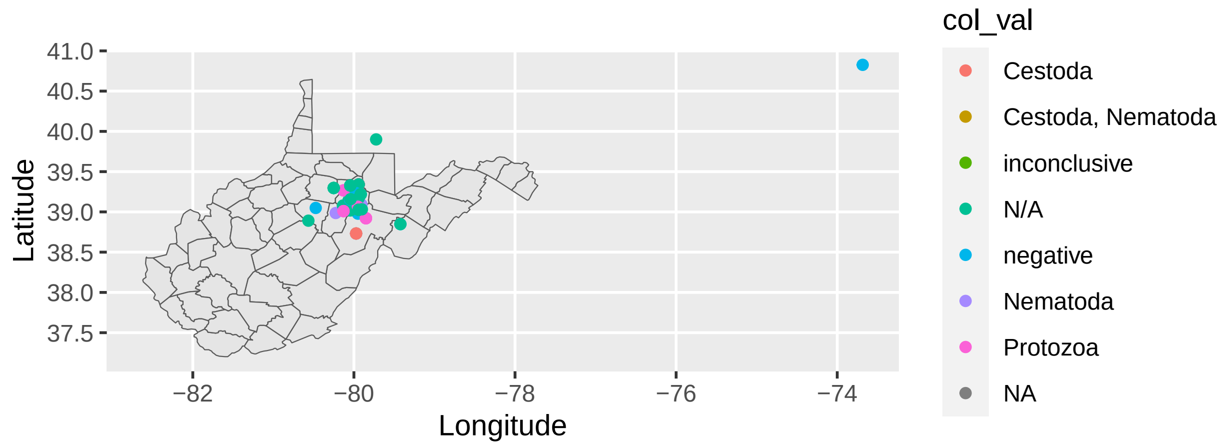
This power data visualization uses location coordinates to map out the geospatial distribution of your data - it can also color-code the location points to better help identify source locations of specific characteristics.

If you do not want categoried data points, set `categories <- "none"`

Currently, to keep CLAW completely open-sourced, the GIS package that plots the maps is only available for U.S. states. Stay up-to-date on the future of CLAW when this feature can be expanded for use in other countries

```
##########################################################
long_lat_data <- "afobc_locations.xlsx"
state <- "west virginia"
categories <- "fecal"
##########################################################

# If coordinate data is external to data set, import
if (long_lat_data != file_name){
  locations <- read_excel("afobc_locations.xlsx")
```

```r
  # Create longitude and latitude columns in dataframe
  data.df$long <- 0
  data.df$lat <- 0
  data.df[is.na(data.df$Intake.Location),]$Intake.Location <- "N/A"

  # Magic - use the locations dataframe to add in the long/lat values:
  for (each in locations$Intake.Location){
    # get index of location
    idx <- match(each,locations$Intake.Location)
    # add longitudes
    data.df[data.df$Intake.Location == each,]$long <- as.numeric(locations$Long[idx])
    # add latitudes
    data.df[data.df$Intake.Location == each,]$lat <- as.numeric(locations$Lat[idx])
  }

# If coordinate data is in original data file, identify longitude/latitude columns
}else{
  long_idx <- which(str_detect(tolower(colnames(data.df)), "long"))[[1]]
  lat_idx <- which(str_detect(tolower(colnames(data.df)), "lat"))[[1]]
  colnames(data.df)[long_idx] <- "long"
  colnames(data.df)[lat_idx] <- "lat"
}

# Set up map and plot
# Get rid of all the empty rows, otherwise errors
loc_data <- data.df[data.df$long != 0,]

# Determine color of data points
if (categories != "none"){
  col_idx <- which(str_detect(tolower(colnames(loc_data)), categories))[[1]]
  col_val <- loc_data[,col_idx][[1]]
}else{
  col_val = "blue"
}

# Define state map
us <- st_as_sf(maps::map("county", regions = state, plot = FALSE, fill = TRUE))
us_tbl <- as_tibble(us)

ggplot() +
  # Plot map
  geom_sf(data = us_tbl, aes(geometry = geom))+
  # Plot points on map
  geom_point(data = loc_data, aes(x = long, y = lat,color=col_val))+
  labs(x="Longitude", y="Latitude")
```



col_val
Cestoda
Cestoda, Nematoda
inconclusive
N/A
negative
Nematoda
Protozoa
NA
Latitude
41.0
40.5
40.0
39.5
39.0
38.5
38.0
37.5
Longitude
−82
−80
−78
−76
−74