Convert Well-Known Binary (WKB) Location Fields to GeoJSON (Containing Latitude & Longitude Information)

The NABat data request output may contain fields for 'grts_geometry' and 'location_geometry'. These fields are output in the Well-Known Binary (WKB) format to allow locational data for points, lines, or polygons to all be stored within the same field within the "data.csv" file.

WKB format will appear as a string of numbers and letters.
For example: 0101000020E31000030BAFA23F34E15BD00B08AD872F0B4550...

*Note that the string length varies significantly depending on the type of location data it represents.

These instructions will guide the user through how to decode their WKB-formatted data into GeoJSON, resulting in surveys' associated Latitude/Longitude info becoming clearly represented within the updated 'data_edit.csv' file.

Installing and Loading the Necessary Packages

```
install.packages("sf")
install.packages("tidyverse")

library(sf)
library(tidyverse)
```

Reading In the 'data.csv' Data Request Output File

User Input Required

```
# Reads a CSV file from the specified path into a data frame named 'data'.  cSV
# file contains data request results obtained through the NABat Partner Portal.

data <- readr::read_csv("C:/Users/Your/Input/File/Path/data.csv")
```

Decoding the 'grts_geometry' Field

```
# Converts the 'grts_geometry' field from WKB format to a spatial object using
# the sf package, and stores it in the new data frame 'data2'.

data2 <- data %>%
    mutate(geometry = st_as_sfc(structure(grts_geometry, class = "WKB"), EWKB = TRUE)) %>%
    st_as_sf()
```

```
# Creates a new data frame 'data3' with converted grts_geometry data.  The
# output will display as geoJSON data for the GRTS cell square polygons.

data3 <- data.frame(data2$geometry)
colnames(data3) <- "grts.geometry"
```

Decoding the 'location_geometry' Field

```
# Converts the 'location_geometry' field from Well-Known Binary (WKB) format to
# a spatial object, then appends it to the 'data2' data frame.

data2 <- data %>%
    mutate(point.locations = st_as_sfc(structure(location_geometry, class = "WKB"),
        EWKB = TRUE)) %>%
    st_as_sf()


# Creates a new data frame 'data4' with converted location_geometry data.  The
# output will display as geoJSON data for the point locations.

# *Note that not all projects elect to share data at the point-location
# specificity.  In these situations, the 'location_geometry' data will either
# mirror the 'grts_geometry' data or appear as blank cells.

data4 <- data.frame(data2$point.locations)
colnames(data4) <- "point.geometry"
```

(Optional) Plotting the Data to Check for Obvious Errors

```
# Creates a plot to visually inspect the raw spatial data produced in 'data2'.

# Graphic should appear as square GRTS cells plotted in Lat/Long space.  If
# queried projects share data at the point-location specificity, then points
# should also appear on the plot.

ggplot(data2) + geom_sf()
```

Combining the Original 'data.csv' Data with the Decoded Locational Info

```
# Merges these new data frames to create a final data frame 'datafinal' which
# includes GRTS cell and point location information.  The newly-converted
# fields will appear as columns at the end of the data table.

datafinal <- bind_cols(data, data3, data4)
```

Exporting Data to a New File

User Input Required

```
# Writes the converted 'datafinal' to a new CSV file at the specified file
# path.
write.csv(datafinal, "C:/Users/Your/Output/File/Path/data_edit.csv")
```

For more tools & resources visit nabatmonitoring.org/resources

See below for complete code block sans R Markdown formatting:

```
library(sf)
library(tidyverse)

# Reads a CSV file from the specified path into a data frame named 'data'.  cSV
# file contains data request results obtained through the NABat Partner Portal.
# **USER INPUT REQUIRED**
data <- readr::read_csv("C:/Users/Your/Input/File/Path/data.csv")


# Converts the 'grts_geometry' field from WKB format to a spatial object using
# the sf package, and stores it in the new data frame 'data2'.

data2 <- data %>%
    mutate(geometry = st_as_sfc(structure(grts_geometry, class = "WKB"), EWKB = TRUE)) %>%
    st_as_sf()


# Creates a new data frame 'data3' with converted grts_geometry data The output
# will display as geoJSON data for the GRTS cell square polygons.

data3 <- data.frame(data2$geometry)
colnames(data3) <- "grts.geometry"


# Converts the 'location_geometry' field from Well-Known Binary (WKB) format to
# a spatial object, then appends it to the 'data2' data frame.

data2 <- data %>%
    mutate(point.locations = st_as_sfc(structure(location_geometry, class = "WKB"),
        EWKB = TRUE)) %>%
    st_as_sf()
```

```r
# Creates a new data frame 'data4' with converted location_geometry data.  The
# output will display as geoJSON data for the point locations.

# *Note that not all projects elect to share data at the point-location
# specificity.  In these situations, the 'location_geometry' data will either
# mirror the #'grts_geometry' data or appear as blank cells.

data4 <- data.frame(data2$point.locations)
colnames(data4) <- "point.geometry"


# Creates a plot to visually inspect the raw spatial data produced in 'data2'.
# Plot should appear as some combination of a square grid with points.

ggplot(data2) + geom_sf()


# Merges these new data frames to create a final data frame 'datafinal' which
# includes GRTS cell and point location information.  The newly-converted
# fields will appear as columns at the end of the data table.

datafinal <- bind_cols(data, data3, data4)


# Writes the converted 'datafinal' to a new CSV file at the specified file
# path.  **USER INPUT REQUIRED**

write.csv(datafinal, "C:/Users/Your/Output/File/Path/data_edit.csv")
```