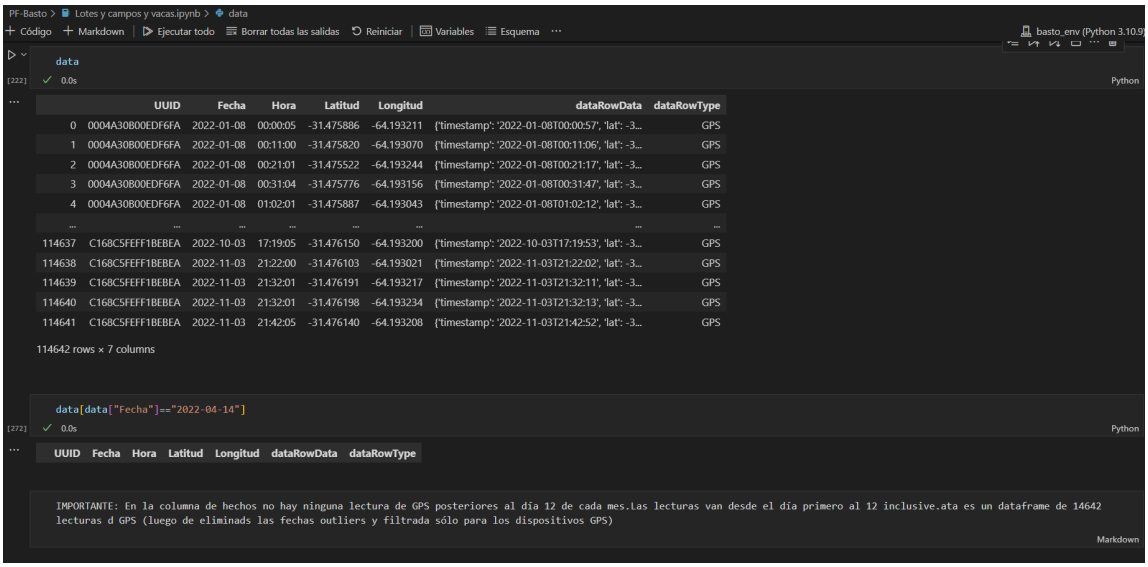
Inconsistencias de la base de prueba de Bastó

- Inexistencia de un diccionario de datos
- La información de cada animal carece de un timestamp, convirtiendo a los campos weight y isPregnant en poco confiables.

- Falta de lecturas de GPS de los dias posteriores al 12 de cada mes en todos los animales y durante todo el período de datos recolectados.



PF-Basto > Lotes y campos y vacas.ipynb > data

+ Código + Markdown | Ejecutar todo | Borrar todas las salidas | Reiniciar | Variables | Esquema ...

basto_env (Python 3.10.9)

```
data
```

[222] ✓ 0.0s Python

	UUID	Fecha	Hora	Latitud	Longitud	dataRowData	dataRowType
0	0004A30B00EDF6FA	2022-01-08	00:00:05	-31.475886	-64.193211	{'timestamp': '2022-01-08T00:00:57', 'lat': -3...	GPS
1	0004A30B00EDF6FA	2022-01-08	00:11:00	-31.475820	-64.193070	{'timestamp': '2022-01-08T00:11:06', 'lat': -3...	GPS
2	0004A30B00EDF6FA	2022-01-08	00:21:01	-31.475522	-64.193244	{'timestamp': '2022-01-08T00:21:17', 'lat': -3...	GPS
3	0004A30B00EDF6FA	2022-01-08	00:31:04	-31.475776	-64.193156	{'timestamp': '2022-01-08T00:31:47', 'lat': -3...	GPS
4	0004A30B00EDF6FA	2022-01-08	01:02:01	-31.475887	-64.193043	{'timestamp': '2022-01-08T01:02:12', 'lat': -3...	GPS
...	...	...	...	...	...	...	...
114637	C168C5FEFF1BEBEA	2022-10-03	17:19:05	-31.476150	-64.193200	{'timestamp': '2022-10-03T17:19:53', 'lat': -3...	GPS
114638	C168C5FEFF1BEBEA	2022-11-03	21:22:00	-31.476103	-64.193021	{'timestamp': '2022-11-03T21:22:02', 'lat': -3...	GPS
114639	C168C5FEFF1BEBEA	2022-11-03	21:32:01	-31.476191	-64.193217	{'timestamp': '2022-11-03T21:32:11', 'lat': -3...	GPS
114640	C168C5FEFF1BEBEA	2022-11-03	21:32:01	-31.476198	-64.193234	{'timestamp': '2022-11-03T21:32:13', 'lat': -3...	GPS
114641	C168C5FEFF1BEBEA	2022-11-03	21:42:05	-31.476140	-64.193208	{'timestamp': '2022-11-03T21:42:52', 'lat': -3...	GPS

114642 rows × 7 columns

```
data[data["Fecha"]=="2022-04-14"]
```

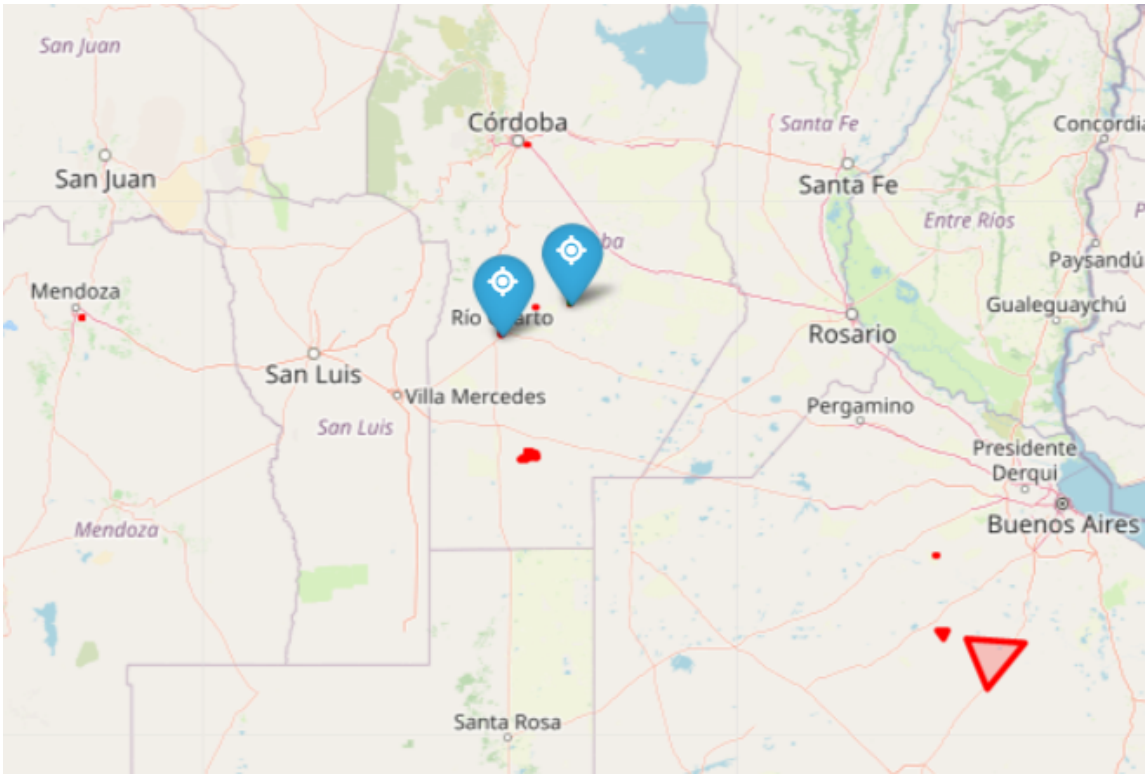
[272] ✓ 0.0s Python

UUID	Fecha	Hora	Latitud	Longitud	dataRowData	dataRowType

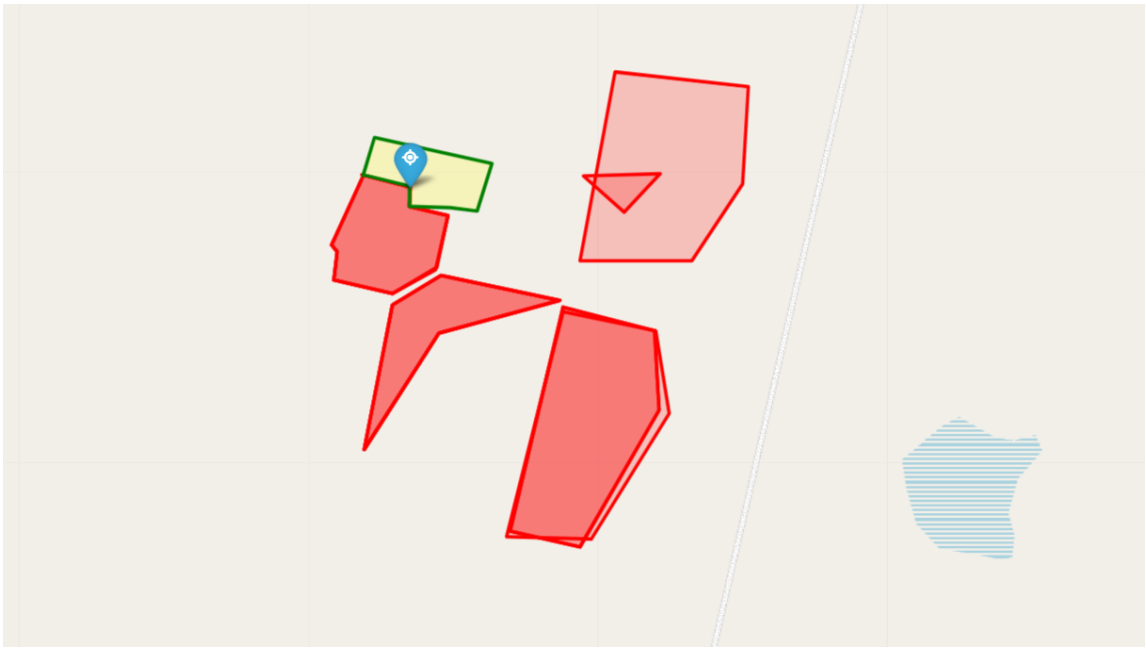
IMPORTANTE: En la columna de hechos no hay ninguna lectura de GPS posteriores al día 12 de cada mes.Las lecturas van desde el día primero al 12 inclusive.ata es un dataframe de 14642 lecturas d GPS (luego de eliminads las fechas outliers y filtrada sólo para los dispositivos GPS)

Markdown

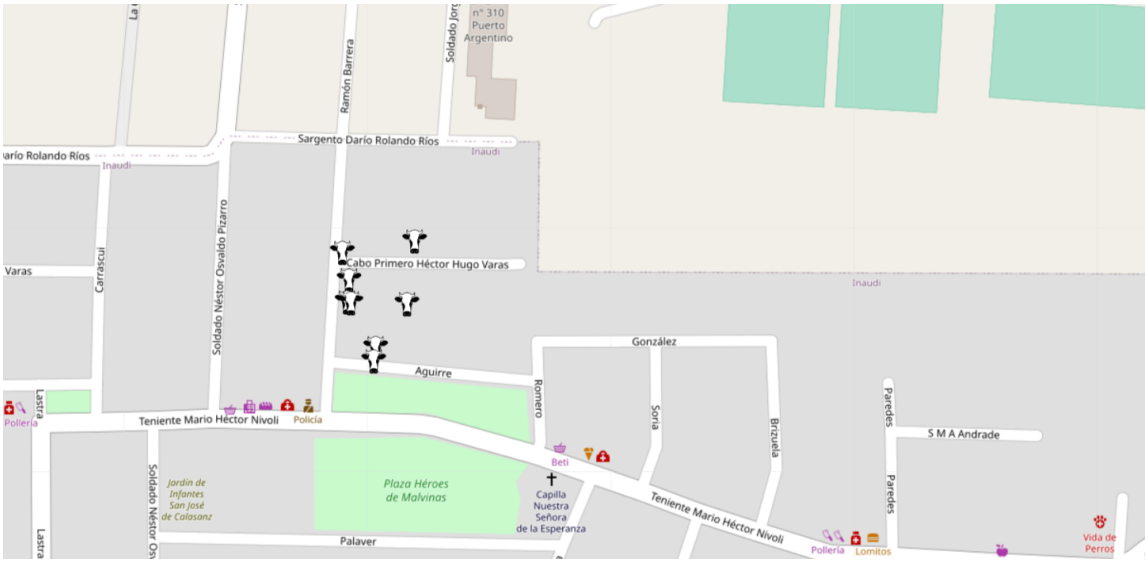
- Incosistencias con los con la ubicación y límites de los Establecimientos de la collección Settlements.En la gráfica se ve por ejemplo un lote triangular en Prov. de BsAs. que incluye a toda la ciudad de Las Flores, y muy pequeñas parcelas dentro de las ciudades de Córdoba, Río Cuarto, S. Miguel de Tucumán y Mendoza.También hay incosistencias en el campo centralPoints que idica la úbicación de los establecimientos.



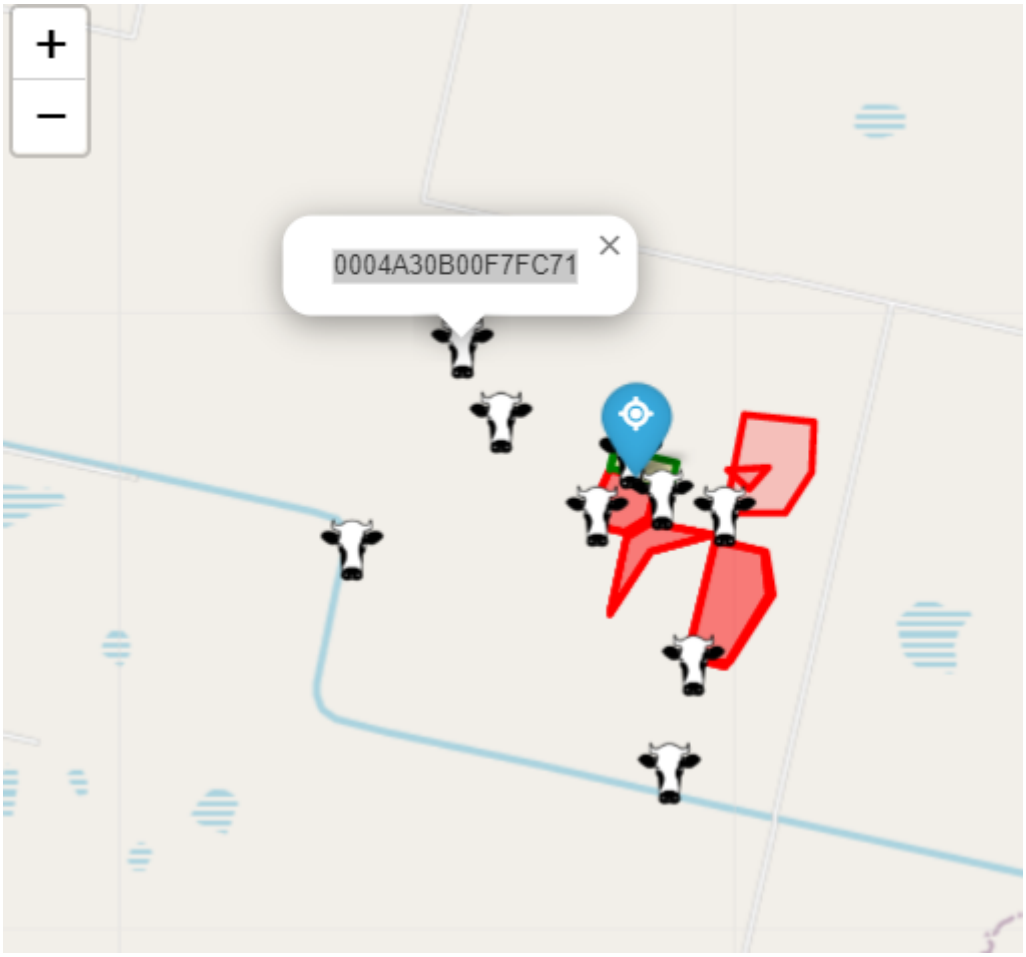
- Se detectaron varias incosistencias en los datos de la collección Plots en los límites de los lotes en el campo geoPoints. Como ejemplo podemos ver en rojo, los lotes de un establecimiento, y en verde y amarillo los límites del perímetro del mismo establecimiento obtenidos de la colección Settlements



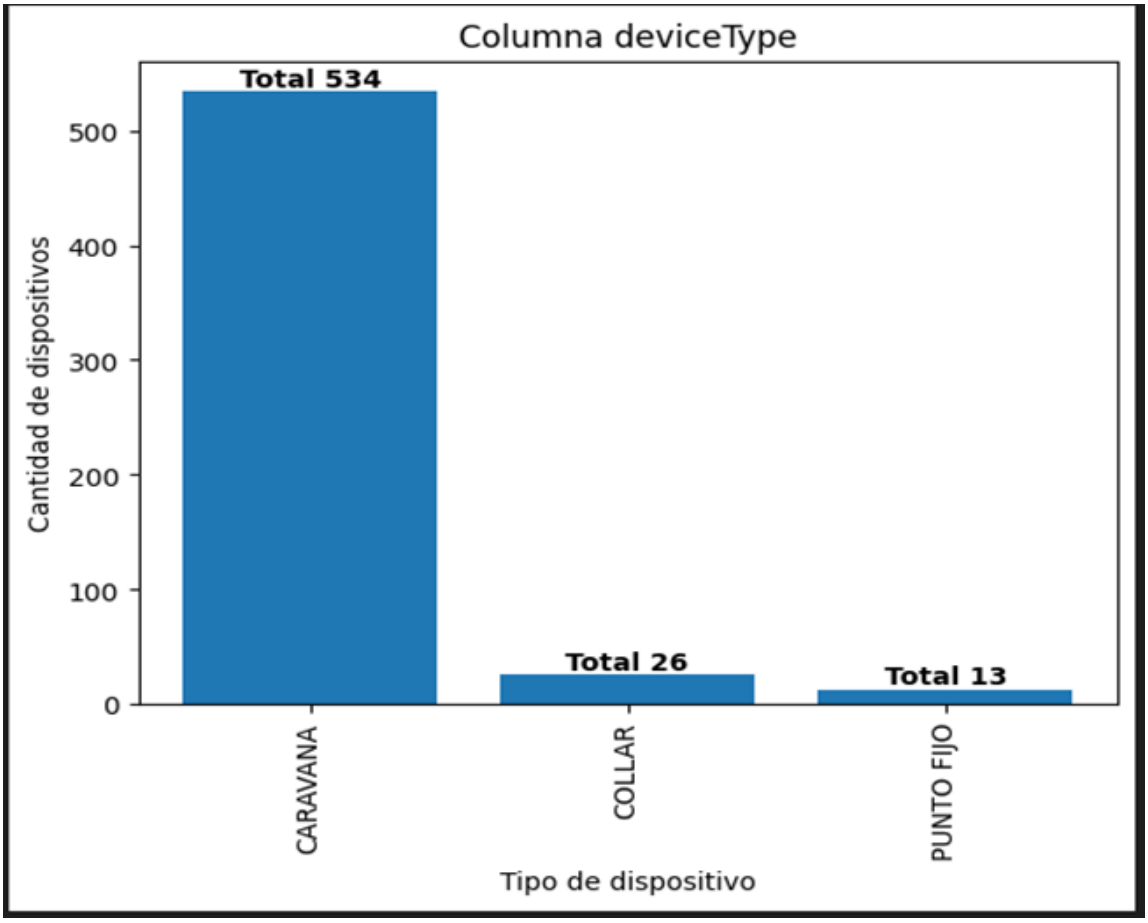
- La posición de algunas vacas con GPS están en lugares aparentemente no aptos par ganadería (en este caso un barrio en los suburbios de la ciudad de Córdoba).De las 30 vacas con GPS, sólo veinte se ubican aparentemente en establecimientos ganaderos, aptos para hacer un seguimiento.



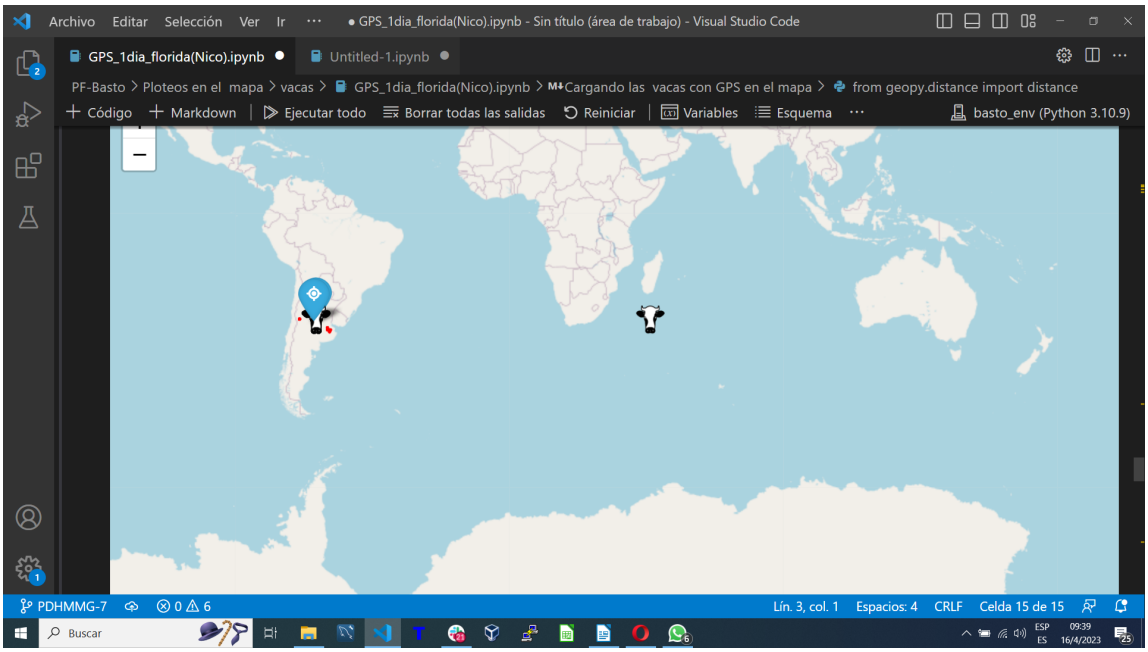
- La posición de los animales con GPS está totlmente afuera de los lotes del establecimineto(GeoPoints de Settlements) einclusive afuera de los límites del establecimiento



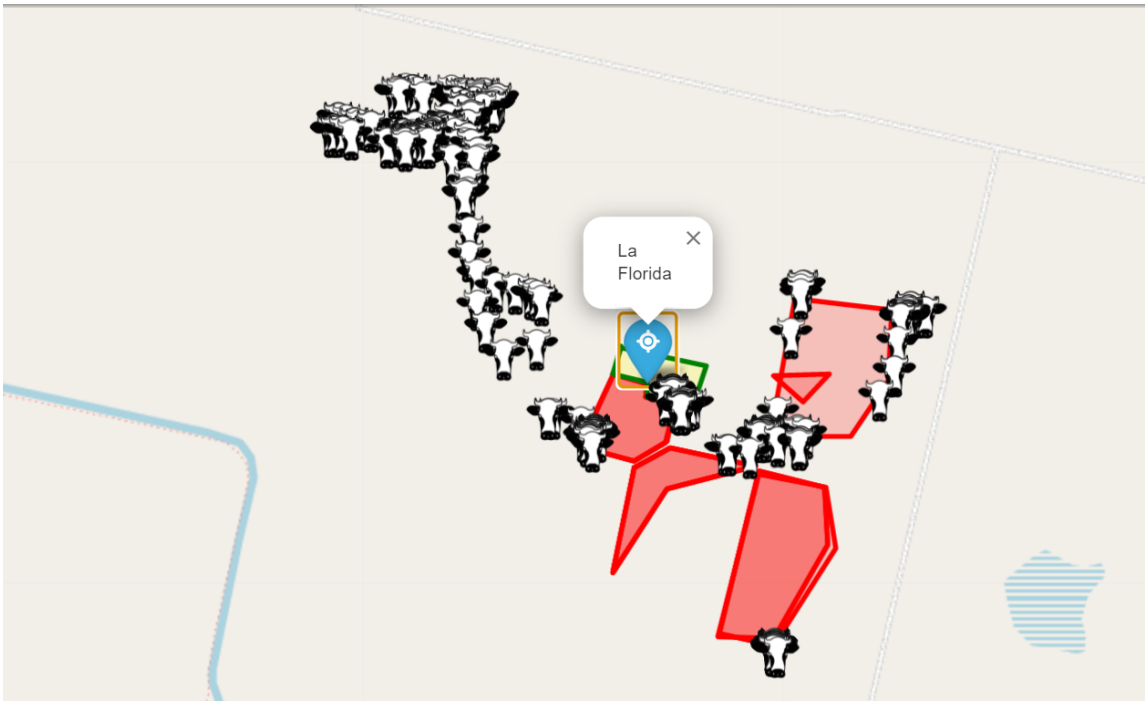
- Inconsistencia entre la cantidad total de animales en la collection “animals” (630), y el número total de dispositivos informado en “devices” en la columna deviceType (total entre collars y caravans: 560). Debería corresponder un dispositivo (ya sea collar o caravana), para cada animal.



Hay outliers en laposición de las vacas reportadas por GPS.Habría que implementar un código que excluya las mediciones que tienen lugar fuera del perímetro del campo.



Graficando un día completos de un único UUID con 313 registros perteneciente al establecimiento La Florida



distancia lineal máxima entre las coordenadas de los puntos más alejados entre si es de 3.

```python
# Calcular la distancia máxima y las coordenadas correspondientes
max_distance = 0
max_coords = None

for i in range(len(tuple_list)):
    for j in range(i+1, len(tuple_list)):
        distance_ij = haversine(*tuple_list[i], *tuple_list[j])
        if distance_ij > max_distance:
            max_distance = distance_ij
            max_coords = (tuple_list[i], tuple_list[j])

print("La distancia máxima es", max_distance, "metros, entre las coordenadas", max_coords)
```

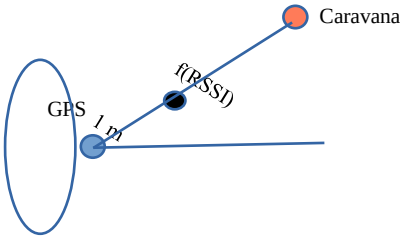
✓ 4.1s Python

```
La distancia máxima es 3374.385649472407 metros, entre las coordenadas ((-32.86435233333336, -63.6331905), (-32.840919, -63.65614333333333))
```

Distancias recorridas :
Para el calculo de las distancias recorridas se utilizan los datos generados por los collares con la geolocalización de los GPS (no las caravanas asociadas).Esto es debido a que son los únicos datos fidedignos de geolocalización que se disponen.
Hasta que se implemente un sistema de triangulación donde se pueda transmitir la posición de cada caravana, el valor de la misma es calculado a través de una función compleja que provee un valor aproximado de la posición.
Para aclarar este punto se explica a continuación la función de geolocalización “calculada” de las caravanas.

Cálculo posición de las caravanas:

ya que las localización de las caravanas está dada por una función compleja de aproximación en la la posición, es fijada en forma aleatoria alrededor del GPS que capta su señal, y en función de la última geolocalización que transmitió ese GPS de su posición.Utilizando una función que calcula un círculo de un metro de radio con centro en las coordenadas del GPS, genera un angulo aleatorio posicionando un punto sobre el círculo, lo que introduce una dirección aleatoria.Luego, una función de distancia basada en la proporcionalidad del valor de RSSI leido por el GPS , nos da la distancia a la que se encuentra la caravana del collar .



En virtud de lo antes expuesto, se ve la necesidad de calcular la distancia recorrida sólo en función de la posición de los GPS(collares).

Para obtener la distancia diaria recorrida, entendemos de que habría que hacer un promedio del recorrida diario de cada GPS. Pero al contar con un a base de datos de desarrollo con información diaria parcial, ya que en todos los casos diarios hay algunos gps que proveen muy poca información, las distancias que recorren no son representativas de la realidad. Por ello, si se promediaran, daría valores muy alterados respecto al real recorrido del ganado y nos imposibilitaria hacer una visualización lógica de los datos.
Por lo cuál, todas las distancias recorridas en forma diaria (diurna, nocturna y total), son obtenidas a través de la distancia recorrida por el GPS cuyo valor es mayor.

Cuando se utilice una base de datos operativa con la información suficiente, según criterio, se podría modificar el código en la función que se muestra en la figura, reemplazando el valor antes mencionado por un promedio de la distancias de todos los GPS.

```
----------> función para cálculo de dataframe de distancia diurna y nocturna

# función para calcular la distancia total a partir de una lista de puntos con latitud y longitud
'''
Para el cálculo del recorrido diario se tomaron sólo los GPS que transmitieron, y se tomó como distancia recorrida
el GPS de mayor recorrido, en lugar de tomar un promedio de todos los GPS como hubiera sido conveniente.
Esto se hizo así porque hay mucha disparidad en la cantidad de mediciones entre diferentes GPS. Muchos transmiten
muy pocas veces por día y muy espaciado, por lo que la función que genera puntos virtuales no los toma, y por otro
lado hay otros que tienen varias mediciones en poco tiempo y luego nada, lo que genera una distancia recorrida muy
corta, y otros GPS solo tienen una medición diaria, con lo cual la distancia recorrida es nula.
Por ello, hacer un promedio no reflejaría una distancia recorrida medianamente representativa de la realidad.
'''

def calcular_distancia_total(puntos):
    total_distance = 0
    for i in range(len(puntos) - 1):
        distancia = distance(puntos[i], puntos[i + 1]).meters
        total_distance += distancia
    return total_distance

def distancias(df):
    # inicializar variables
    uuid_anterior = None
    lat_anterior = None
    lon_anterior = None
    fecha_anterior = None
    puntos = []
    data = {'UUID': [], 'Fecha': [], 'distancia recorrida': []}

    # recorrer dataframe fila por fila
    for index, row in df.iterrows():
        uuid_actual = row['UUID']
        lat_actual = row['Latitud']
        lon_actual = row['Longitud']
        fecha_actual = row['Fecha']

        # si es la primera fila del recorrido, no calculamos distancia
        if uuid_anterior is None:
            uuid_anterior = uuid_actual
            lat_anterior = lat_actual
            lon_anterior = lon_actual
            fecha_anterior = fecha_actual
            puntos.append((lat_actual, lon_actual))
```