# Mar.utils::assess_privacy()

**Mike McMahon**

**2020-05-14**

# Assessing Privacy/Rule of 5

May 8, 2020 - Mike McMahon (mike.mcmahon@dfo-mpo.gc.ca)

## Background

Fisheries catch and effort information for a corporate licence holder is considered to be sensitive, proprietary information and protected under Section 20 of the Access to Information Act. Without written consent, DFO is not permitted to release information or data products (e.g., maps and data layers) that might reveal personal or third party information such as catch, landed values, and vessel-specific fishing locations. It is possible to de-personalize information through aggregation, and the ATIP Secretariat recommends using a minimum of five when aggregating personal information.

This function facilitates the production of an aggregated product that meets the “Rule of 5” that can be publicly distributed.

## Assessing Permissability for Aggregation

The approach used by this function follows the guidelines outlined in the Atlantic Canadian Protocol on Mapping Fishing Activity. In short, sensitive point data is overlain on a polygon layer, and within each polygon, all of the sensitive fields are examined to check how many unique values exist within any given polygon. If there are fewer than 5 unique values of any of the sensitive fields, no data from that polygon can be shared. If there are at least 5 unique values, then all of the data within that polygon can be aggregated together, and the aggregate values can be distributed. The idea is that if at least 5 different entities are all combined, it it impossible to determine from the aggregated data which data is associated with each.

## Method

By default, this script uses the NAFO areas as the layer against which the sensitive data is overlain. More specifically, it assesses the raw data against the “NAFO Subunits” found on the 1995 Canadian Hydrographic Service Chart 10040 - “Fisheries and Oceans Statistical Chart” - “Gulf of Maine to Strait of Belle Isle”. If ALL of the sensitive fields meet the criteria of having at least 5 unique values within a polygon in a given dataset, that data can be aggregated and shared. If the data has multiple sensitive fields (e.g. Licenses and Vessels), the Rule of 5 must be met by all - if one field fails, NO data from within that polygon can be shared. In the example below, the hashed areas illustrate polygons that contain data, but the data does not meet the requirements of the Rule of 5, and cannot be aggregated and/or shared.

Once it is established which polygons contain sufficiently varied data (i.e. complies with the Rule of 5), all of the point data within those polygon can be aggregated. Following the Atlantic Canadian Protocol on Mapping Fishing Activity, the raw data is aggregated into hexagons, each with an area of 2min$^2$.

Two output shapefiles are generated for each run of the function:

1. The polygon file whose areas were used to assess the raw data is returned. These polygons indicate the results of the applying the Rule of 5, and indicate whether or not data within each polygon is sufficiently varied to allow aggregation. The polygons have a variable called `CAN_SHOW` that is populated with:
    - `YES` (i.e. this polygon has sufficient data to allow for aggregation of data)
    - `NO` (i.e. this polygon does NOT have enough data to aggregate. Data from this poly must not be shown)
    - <NULL> (i.e. no raw data exists in this polygon)

    For those polygons where `CAN_SHOW` is `YES`, aggregate values of the raw data are available (i.e. “MEAN”, “SUM” and “COUNT”). No such data exists for polygons that failed the Rule of 5 check.
2. The aggregated data is returned as a mesh of hexagons (or squares), each with an area of 2min $^2$. The mesh is only created for data falling within polygons that met the Rule of 5. Within each hexagon are the aggregate value for the raw data that fell within that hexagon.

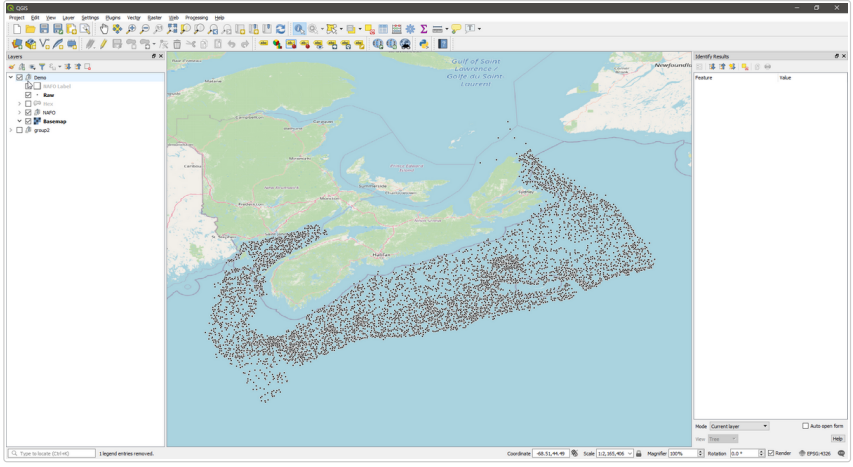
### Example

For the purpose of illustration, I will use non-sensitive RV vessel data for the Maritimes, specifically, I will use all of the Cod (species code = 10) catches for the summer survey (from 1970 to present).

```
# pull and filter the rv data (summer survey, cod only)
Mar.datawrangling::get_data('rv', data.dir = data.dir)
Mar.datawrangling::get_survey('rv', survey = "SUMMER")
GSSPECIES = GSSPECIES[GSSPECIES$CODE %in% c(10),]
Mar.datawrangling::self_filter(keep_nullsets = F)
all_cod = Mar.datawrangling::summarize_catches()

# save the raw data as a shapefile
Mar.utils::df_to_shp(all_cod, lat.field = "LATITUDE", lon.field = "LONGITUDE" , filename = "demo_raw")
```

At this point, we have data for all of the catches of cod during the RV survey.



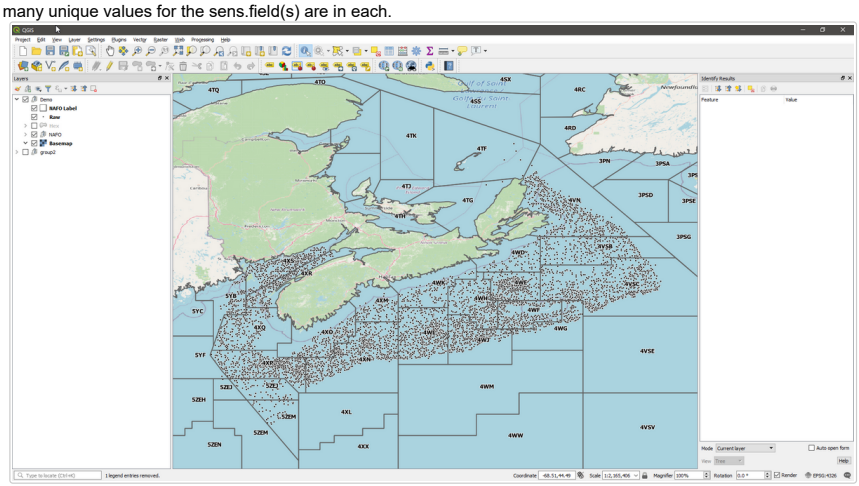
Raw Summer RV data for cod

### Applying the Rule of 5

But now, if we pretend that the field “YEAR” is sensitive (like “license”, or “vessel registration number”), we can illustrate how the script functions. The implication is that we want to obscure areas where an area has sets from less than 5 different years.

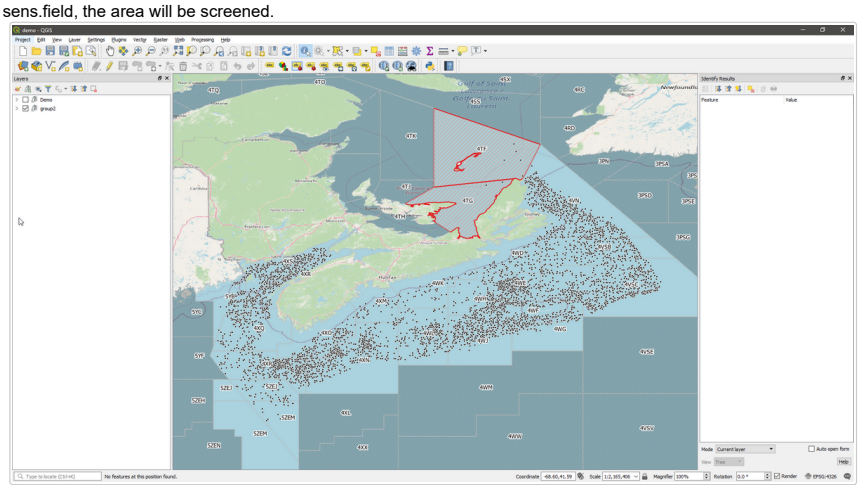
```
# run the script, pretending that "YEAR" is a sensitive field.
Mar.utils::assess_privacy(df=all_cod,
                          agg.fields = c("TOTNO","TOTWGT"),
                          sens.fields = "YEAR",
                          file.id = "demo")
```

Behind the scenes, the first thing that happens is that the raw data is overlain with the NAFO subunits. At this stage, the raw data is asessed to see which polygon it’s in, and then, for each polygon, the script counts how many unique values for the sens.field(s) are in each.



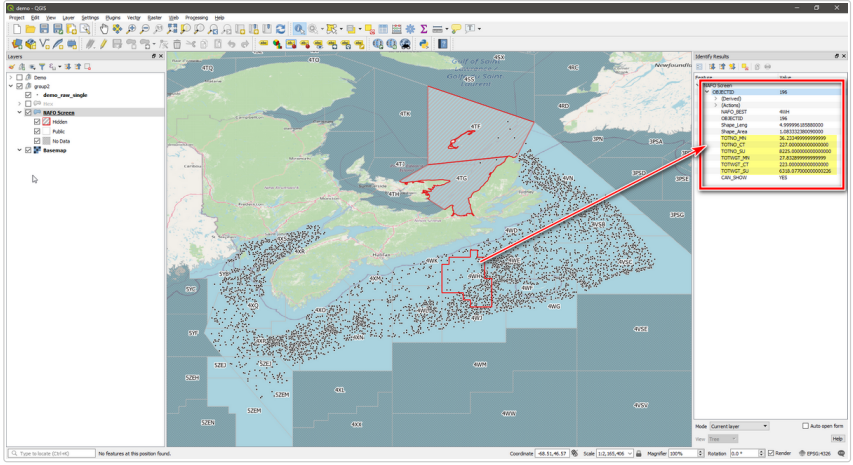
NAFO Subunits overlain over raw data

Below, you can see the NAFO subunits again, but this time, some have red hashed pattern, some are greyish, and some are transparent. The transparent ones are those that meet the Rule of 5, thes red hashed ones failed the Rule of 5, and the greyish ones contain no raw data. If you look carefully at the red hashed ones, you can see several raw data points, but not too many. For this example, there are less than 5 unique years associated with data in those areas. There could be 100s of points, but as long as they had less than 5 unique values for the sens.field, the area will be screened.



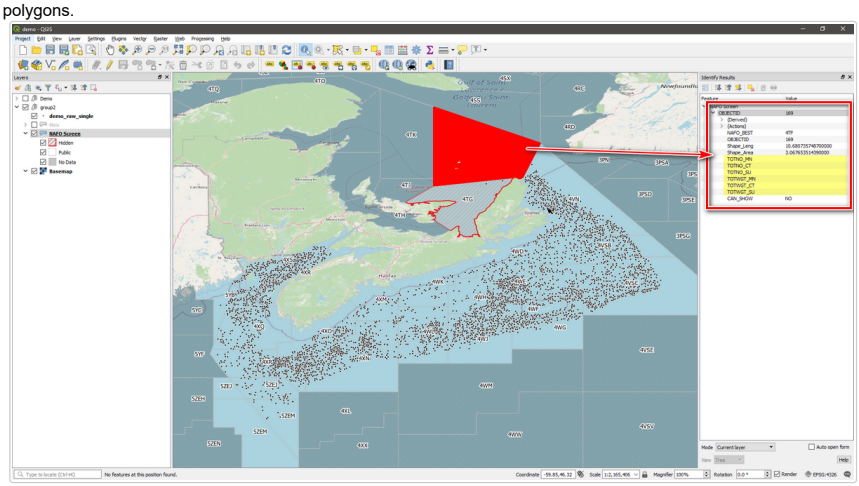
Results of checking the raw data against the Rule of 5 using the NAFO Subunits

Clicking on any of the NAFO subunits will reveal the attributes for the polygon, including aggregate values for those polygons where the Rule of 5 was met.



Aggregated Results from Screen (CAN_SHOW =“YES”)

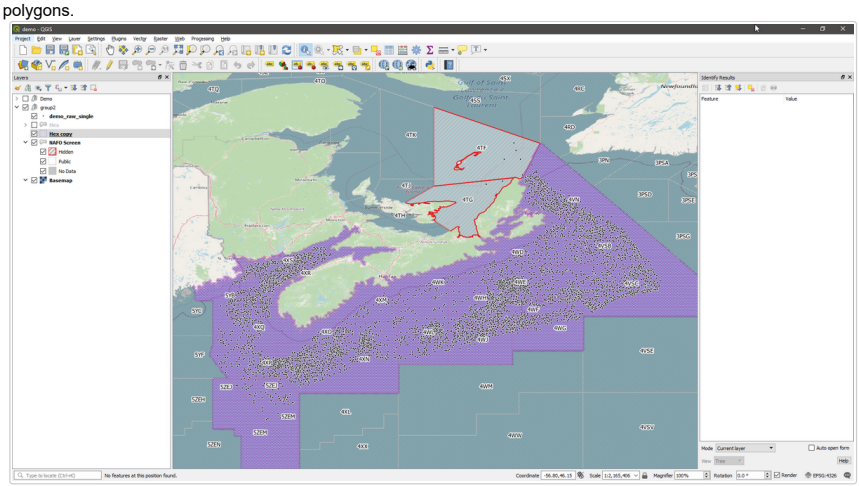
Conversely, those polygons without sufficiently varied data do not have aggregate values available for the polygons.



Aggregated Results from Screen (CAN_SHOW =“NO”)

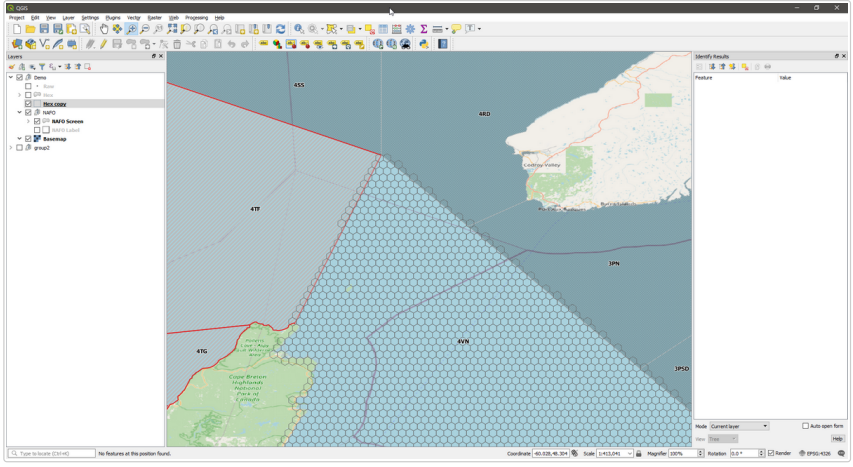
### Aggregating the Raw Data

At this point the script has identified the polygons with and without privacy concerns, and has determined which data can be aggregated,and which cannot. For those polygons that can be aggregated, a mesh layer is overlain (typically hexagons), and the raw data is then aggregated into the individual cells. In the image below, notice how the (purple) hexagonal mesh overlaps most of the raw data, but excludes the areas covered by the red, hashed polygons.



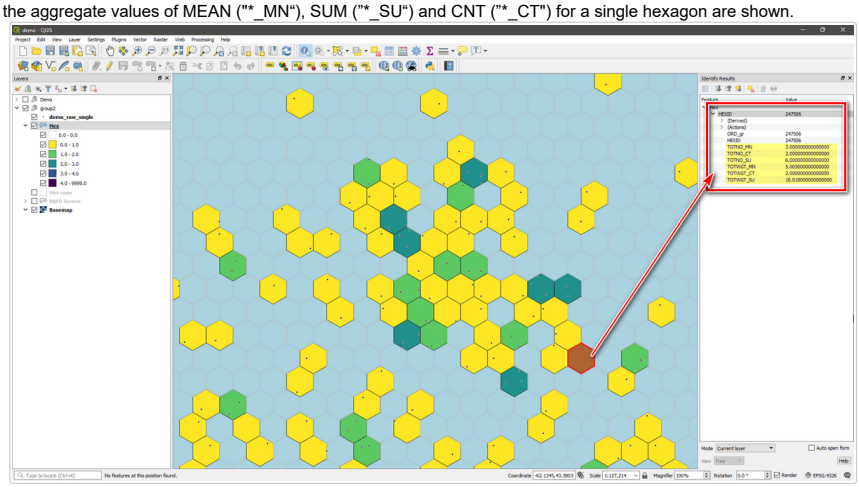
Hexagonal Mesh

Zooming in, you can see some details of the mesh along the borders of different classes of polygons (i.e. CAN_SHOW = “YES”, “NO” and <NULL>).



Hexagonal Mesh Details

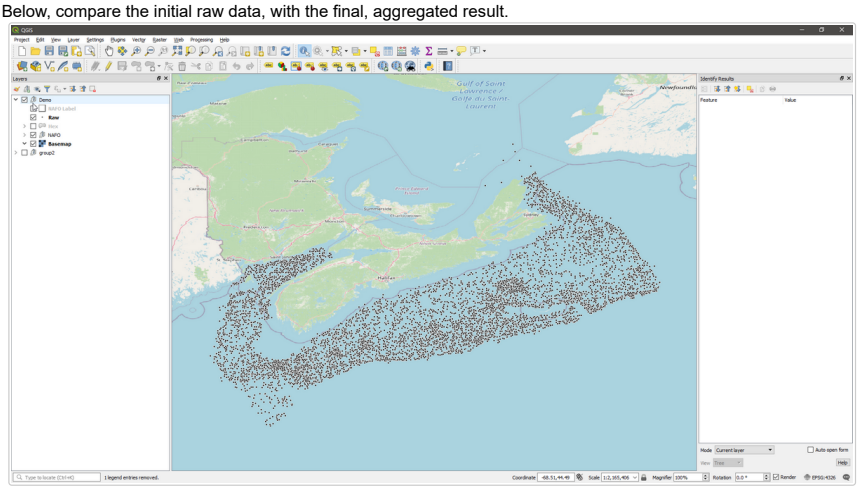
If raw data is overlain on the mesh, and the mesh is symbolized such that the colours correspond with the aggregate value of “COUNT”, the result of the aggregation is clear - note that colour of each hexagon corresponds with the number of raw data points that sit on top of it. Hexagons containing no data are empty, those with a single data point are yellow, and those with more are progressively darker. On the right of the image, the aggregate values of MEAN (“*_MN“), SUM (”*_SU“) and CNT (”*_CT") for a single hexagon are shown.



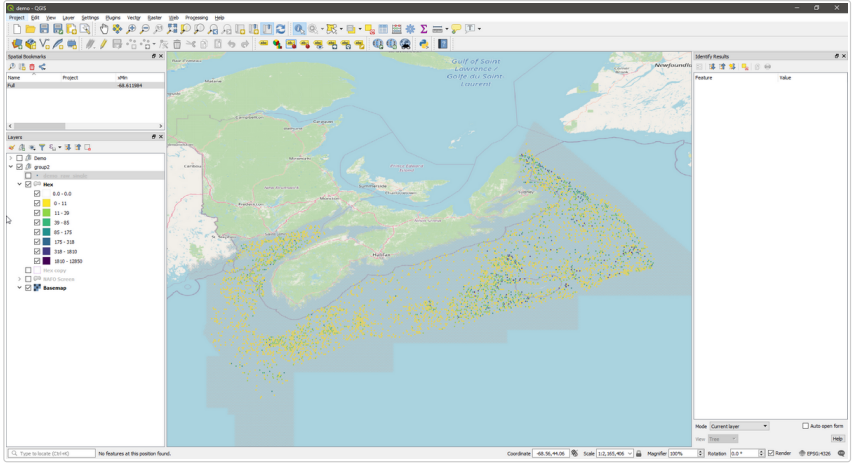
Hexagonal Mesh with Raw Data

### Final Results

Below, compare the initial raw data, with the final, aggregated result.



Raw Summer RV data for cod



Aggregated Summer RV data for cod, symbolized by MEAN TotNo

## Other Cases

This example above illustrates the simplest case - data where each row is a different measurement and all measurements are for a single species. Other cases also exist - such as bycatch data, where each row might contain data for a different species. While the parameters are all documented in the various help files, so will be gone over in more details here.

### Bycatch Data

For typical bycatch data, different rows may represent catches of different species. Some sets will have multiple records in the data- at least 1 for each species that was caught. To handle these sorts of data, the script reformats the data from “long” format to “wide” format.

For example - here’s what the default “wide” format data might look like:

| DATA1 | DATA2… | TRIP | SETNO | SPEC_CD | TOTWGT | TOTNO |
|---|---|---|---|---|---|---|
| blah | blah2 | TRIP1 | 1 | 10 | 2.5 | 1 |
| blah | blah2 | TRIP1 | 1 | 2550 | 16.2 | 4 |
| snarf | snarf2 | TRIP1 | 2 | 10 | 11.1 | 12 |
| blah | blah2 | TRIP1 | 1 | 11 | 2214.5 | 1023 |
| snarf | snarf2 | TRIP1 | 2 | 44 | 34.2 | 45 |
| snarf | snarf2 | TRIP1 | 2 | 11 | 0.5 | 1 |

And here’s the same data, but in “Wide” format.

| DATA1 | DATA2… | TRIP | SETNO | SPEC10_TOTWGT | SPEC10_TOTNO | SPEC11_TOTWGT | SPEC11_TOTNO | SPEC44_TOTWGT | SPEC44_TOTNO | SPEC |
|---|---|---|---|---|---|---|---|---|---|---|
| blah | blah2 | TRIP1 | 1 | 2.5 | 1 | 2214.5 | 1023 | NA | NA | 16.2 |
| snarf | snarf2 | TRIP1 | 2 | 11.1 | 12 | 0.5 | 1 | 34.2 | 45 | NA |

Now, each set only gets a single row, and each species has its own set of dedicated columns.

In order to make this happen, the script requires several fields be identified:

1. `agg.fields` - these are always required (not just for bycatch data). These are the fields on which aggregate calculations will be performed (i.e. sum, mean, count). In the example tables above, these would be identified as: `agg.fields=c("TOTWGT","TOTNO")`
2. `key.fields` - these are the fields which are required to identified a single row/set of data. In the example above, these would be identified as: `key.fields=c("TRIP","SETNO")`
3. `facet.field` - this is the field that contains the values that will become column headers. For bycatch data, this will likely be a species field.

**NOTE** It is important that the data provided to the script does not have more than one fields that *could* act as a facet.field. For example, if your data has SPECIES_CODE and SPECIES_COMMON_NAME, you should only provide one as the facet field, and you should drop the other prior to running the script.