Webinar Series

Data Use Skills
Featuring Data from Natural History Collections

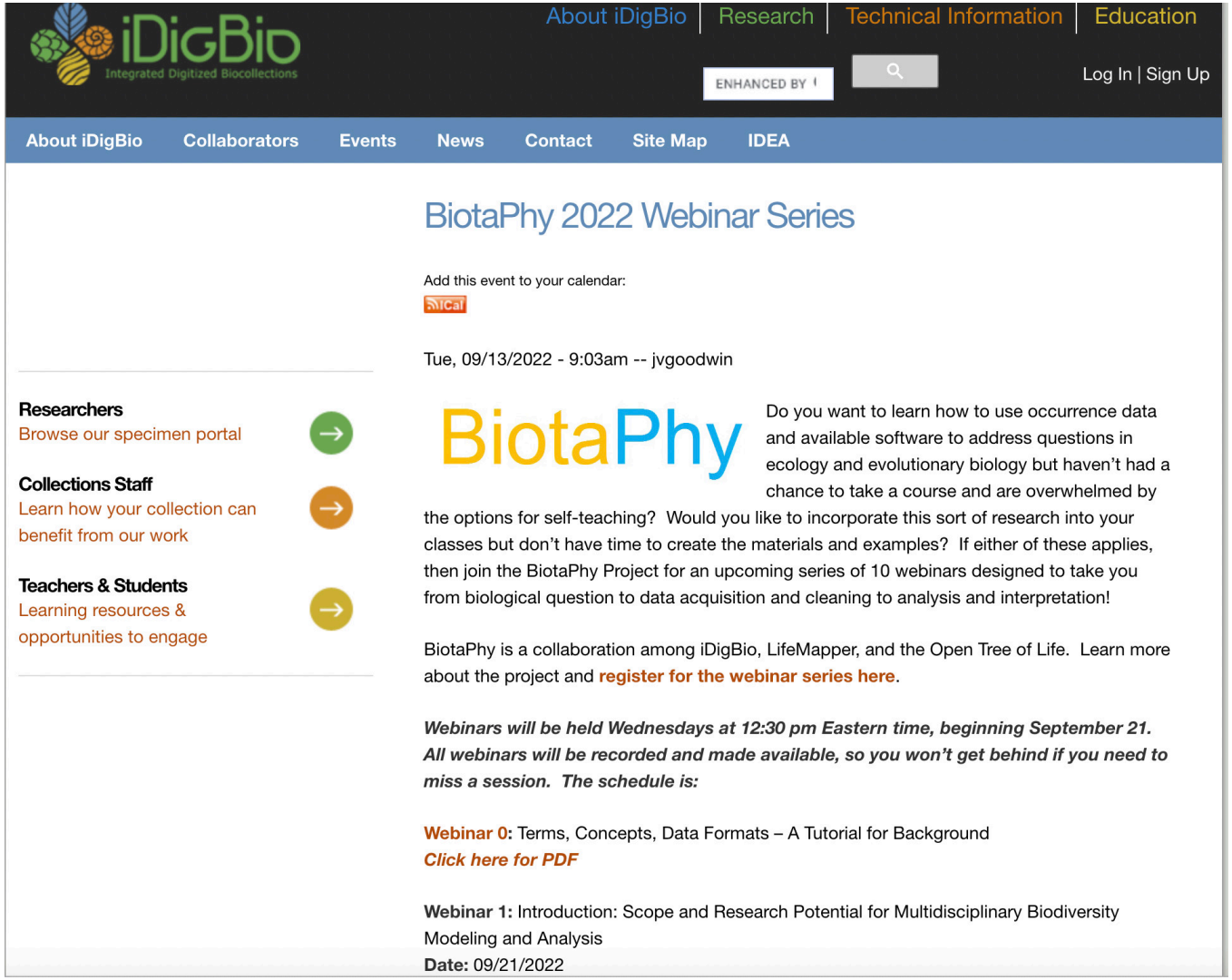
September 21-November 30, 2022

https://www.idigbio.org/content/biotaphy-2022-webinar-series

iDigBio:

https://www.idigbio.org/content/biotaphy-2022-webinar-series

iDigBio.org



Thank You

Maria Cortez
Aimee Stewart

Jill Goodwin
Gil Nelson

Webinar 3

Clean Your Dirty Data

Session Summary Webinar 2

1. Species list must contain: **ACCEPTED, UNIQUE and CONSISTENT names!**
2. Manual ways of **treating nomenclature**: useful for small datasets and to subset large datasets.
3. Automated ways of treating nomenclature: useful for large datasets.
4. **BiotaPhy framework is a great toolkit for automated pathway!**

Goals

Introduce the importance of cleaning data before processing biodiversity analyses and practice a few ways to do so

Learning Objectives

Biological Objectives:

- ✓ **How can we use occurrence data, and where can we find them?**

- ✓ **Introduce why cleaning data is an essential step to ensure biodiversity analyses are as sound as possible.**

Learning Objectives

Technical Objectives :

- ✓ **How to download occurrence data from iDigBio (and GBIF)**

- ✓ **How to clean large data sets effortlessly, consistently and rapidly using industrial-strength tools**

Webinar organization

1. **Exploring Concepts: How can we use occurrence data, and why is cleaning data an important step?**
2. **Demonstration: How to download data from iDigBio**
3. **On Your Own: More options for data downloads**
4. **Exercises: Using Biotaphy Tools for cleaning occurrence data**
5. **Session Summary, Q&A and Discussion**

Exploring Concepts

How can we use occurrence data, and where can we find them?

Exploring Concepts

Occurrence data are essential for producing species distribution models, estimating phylogenetic diversity, and more!!

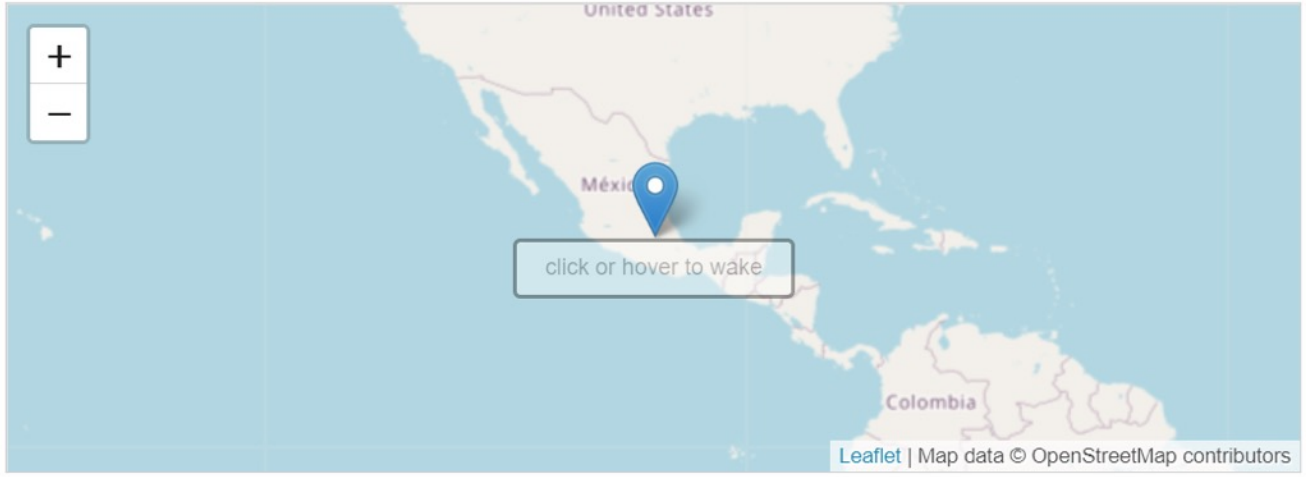
Lat: -19.065

Lon: -98.6336111

Heuchera mexicana W. Schaffn. ex Small & Rydb., 1905

From Computarización del Herbario ENCB, IPN. Fase IV. Base de datos de la familia Pinaceae y de distintas familias de la clase Magnoliopsida depositadas en el Herbario de la Escuela Nacional de Ciencias Biológicas, IPN

Continent	North America	Institution Code	Encb-ipn
Country	Mexico	Collection Code	Encb
State/Province	Mexico	Catalog Number	No Disponible
County/Parish	Ecatzingo	Collected By	Rosario Vázquez
Locality	Campamento Tlamacas, Amecameca	Date Collected	1962-05-29
Latitude	19.065		
Longitude	-98.6336111		

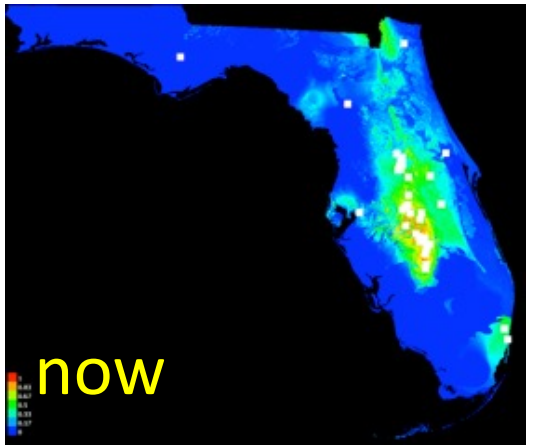


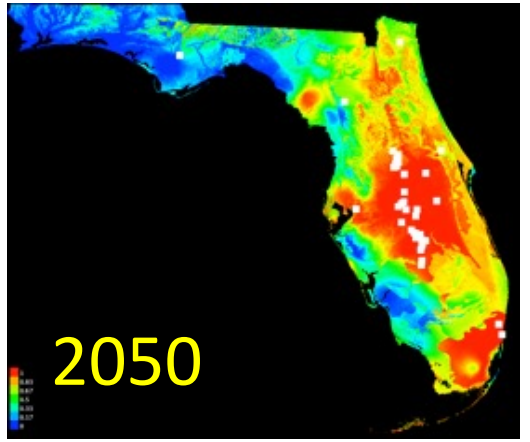
iDigBio
Integrated Digitized Biocollections

Exploring Concepts

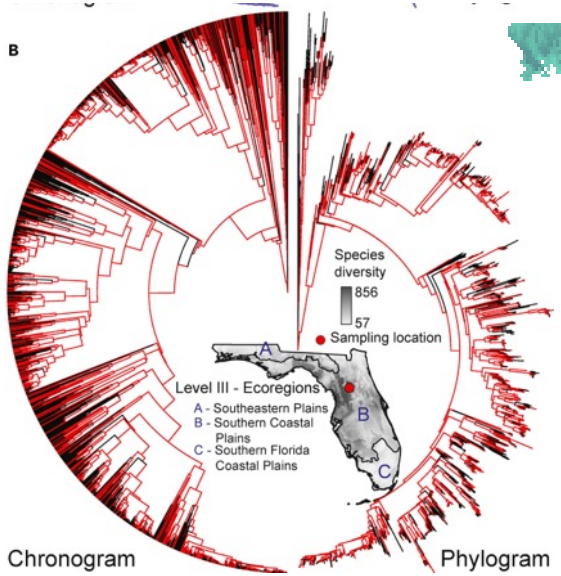
Species distribution models

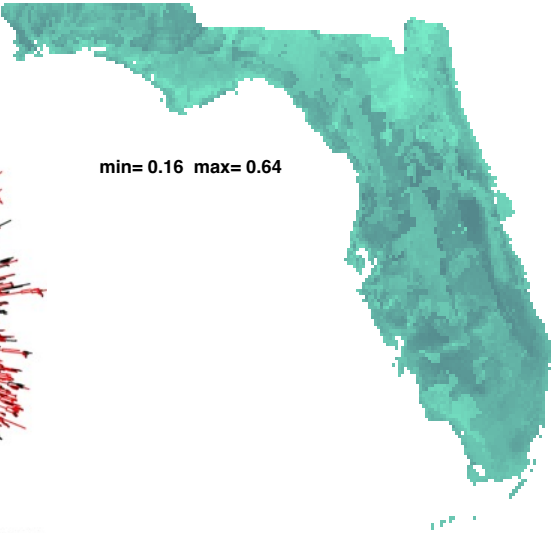
Prunus geniculata





Phylogenetic diversity





Where do we get occurrence data?

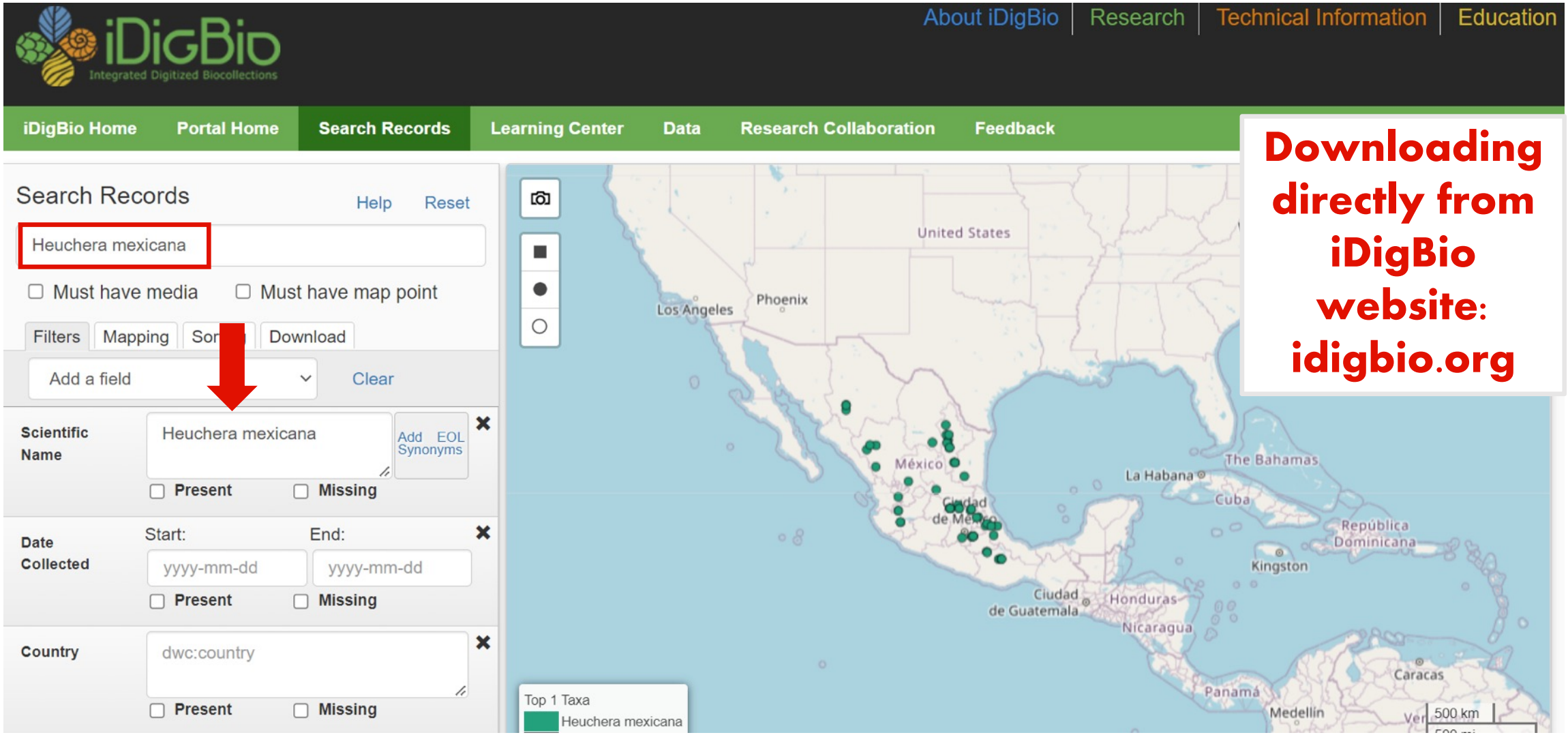
Where do we get occurrence data?

GBIF

iDigBio

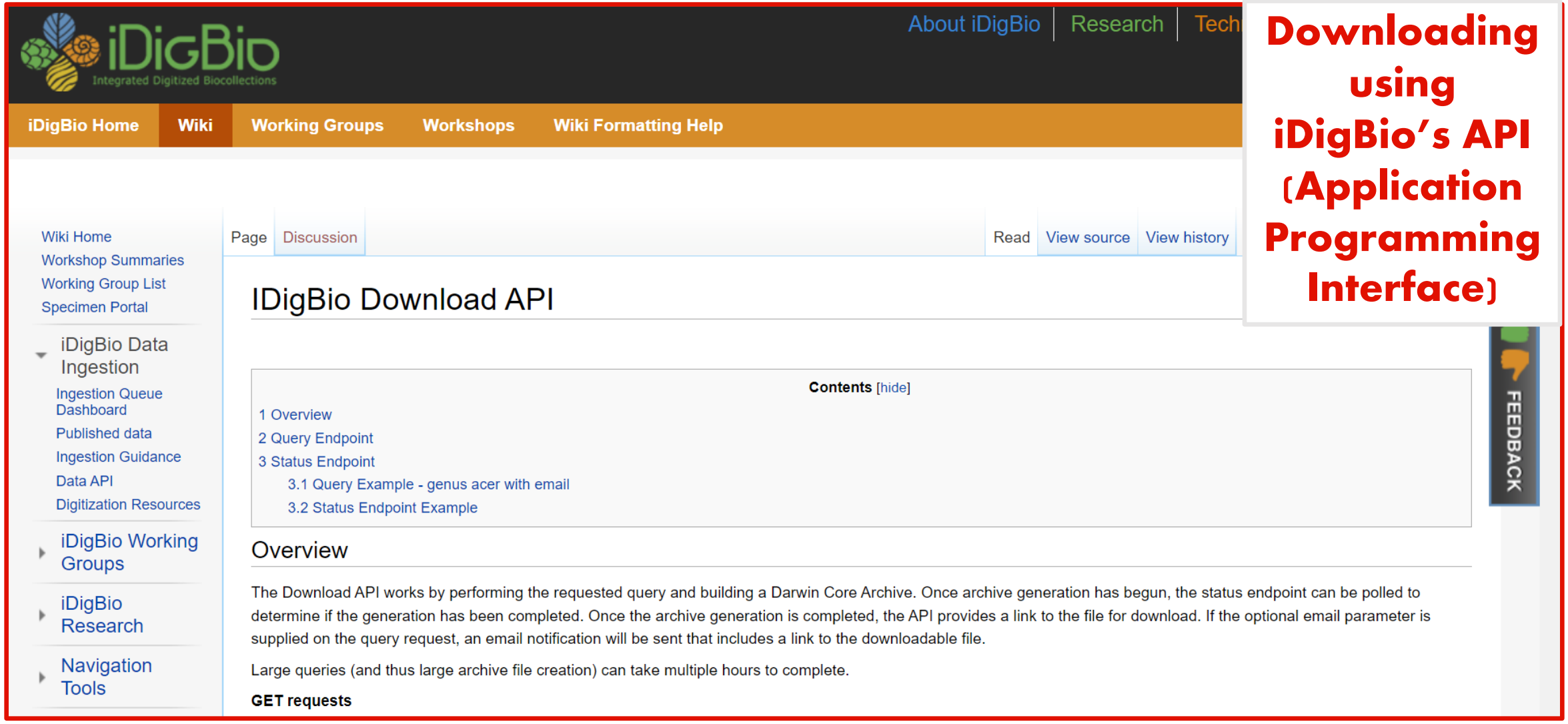
Downloading data from portals: iDigBio

Downloading data from portals: iDigBio



Downloading directly from iDigBio website: idigbio.org

On your own: Downloading data: iDigBio



Downloading using iDigBio's API (Application Programming Interface)

On your own: Downloading data: iDigBio

Download occurrence data from iDigBio

iDigBio's API

To download from iDigBio, full instructions are at the Download API reference.

To pull data from the command prompt, use the `curl` command to pull text response directly to terminal with the example query_url: Euphorbia

```
$ curl https://api.idigbio.org/v2/download/?rq=%7B%22genus%22%3A%22euphorbia%22%7D&email=donotreply%40idigbio.org
[1] 58979
astewart@murderbot:~/git/tutorials$ {
  "complete": false,
  "created": "2022-05-02T15:28:41.730968+00:00",
  "expires": "2022-06-01T15:28:41.628063+00:00",
  "hash": "18911492e8517926cb8693fc9f971cf066107016",
  "query": {
    "core_source": "indexterms",
    "core_type": "records",
    "form": "dwca-csv",
    "mediarecord_fields": null,
    "mq": null,
    "record_fields": null,
    "rq": {
      "genus": "euphorbia"
    }
  },
  "status_url": "https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38",
  "task_status": "PENDING"
}
```

On your own: Downloading data: iDigBio

Then use `curl` on the resulting **status_url** field:

iDigBio's API

```
$ curl https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38
{
  "complete": false,
  "created": "2022-05-02T15:28:41.730968+00:00",
  "expires": "2022-06-01T15:28:41.735029+00:00",
  "hash": "18911492e8517926cb8693fc9f971cf066107016",
  "query": {
    "core_source": "indexterms",
    "core_type": "records",
    "form": "dwca-csv",
    "mediarecord_fields": null,
    "mq": null,
    "record_fields": null,
    "rq": {
      "genus": "euphorbia"
    }
  },
  "status_url": "https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38",
  "task_status": "PENDING"
}
```

On your own: Downloading data: iDigBio

When the task_status shows "SUCCESS":

iDigBio's API

```
$ curl https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38
{
  "complete": true,
  "created": "2022-05-02T15:28:41.730968+00:00",
  "download_url": "http://s.idigbio.org/idigbio-downloads/d54c0ad7-6697-4096-9f11-b2a9a6041a38.zip",
  "expires": "2022-06-01T15:28:41.552351+00:00",
  "hash": "18911492e8517926cb8693fc9f971cf066107016",
  "query": {
    "core_source": "indexterms",
    "core_type": "records",
    "form": "dwca-csv",
    "mediarecord_fields": null,
    "mq": null,
    "record_fields": null,
    "rq": {
      "genus": "euphorbia"
    }
  },
  "status_url": "https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38",
  "task_status": "SUCCESS"
}
```

On your own: Downloading data: iDigBio

When the task_status shows "SUCCESS":

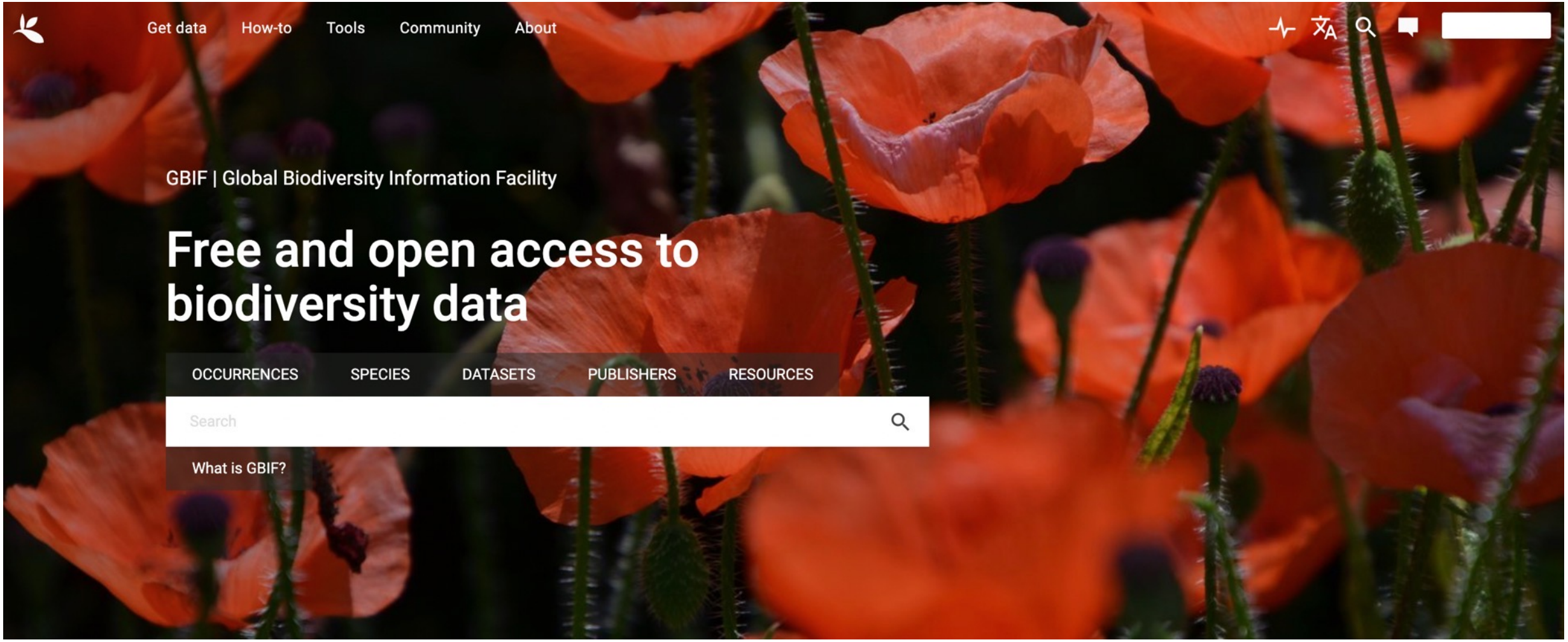
iDigBio's API

```
$ curl https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38
{
  "complete": true,
  "created": "2022-05-02T15:28:41.730968+00:00",
  "download_url": "http://s.idigbio.org/idigbio-downloads/d54c0ad7-6697-4096-9f11-b2a9a6041a38.zip",
  "expires": "2022-06-01T15:28:41.552351+00:00",
  "hash": "18911492e8517926cb8693fc9f971cf066107016",
  "query": {
    "core_source": "indexterms",
    "core_type": "records",
    "form": "dwca-csv",
    "mediarecord_fields": null,
    "mq": null,
    "record_fields": null,
    "rq": {
      "genus": "euphorbia"
    }
  },
  "status_url": "https://api.idigbio.org/v2/download/d54c0ad7-6697-4096-9f11-b2a9a6041a38",
  "task_status": "SUCCESS"
}
```

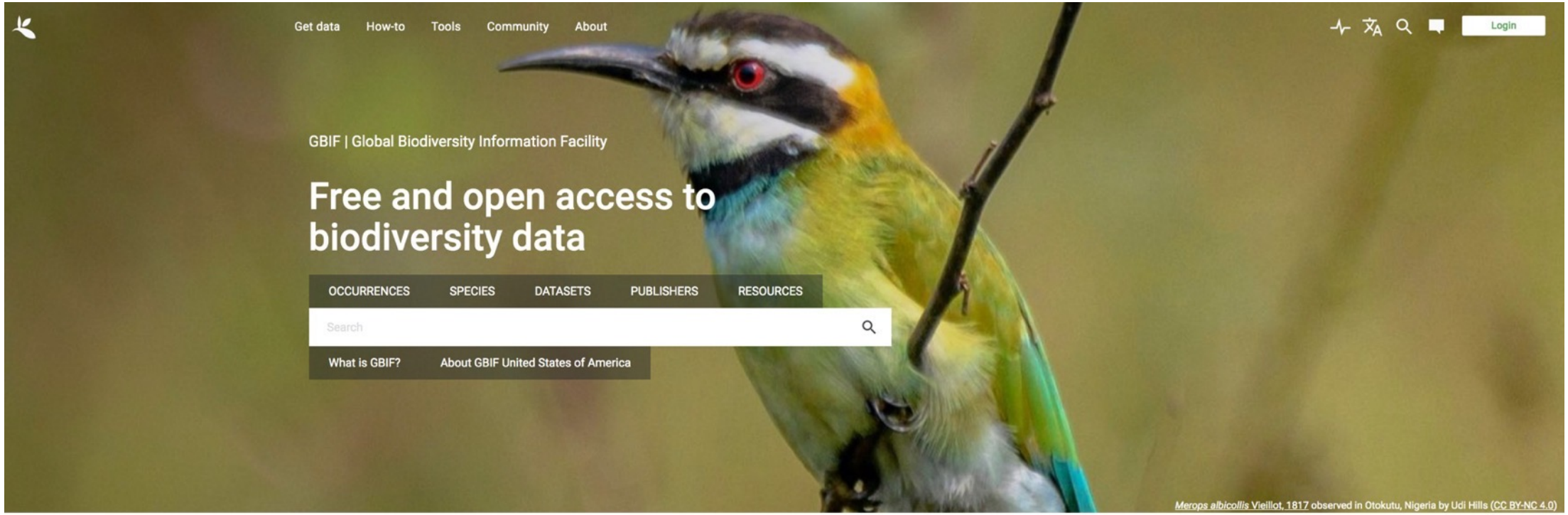
Save the response into a file with the `wget` command and the **download_url** field:

```
wget http://s.idigbio.org/idigbio-downloads/d54c0ad7-6697-4096-9f11-b2a9a6041a38.zip
```

On your own: Downloading data: GBIF



On your own: Downloading data: GBIF



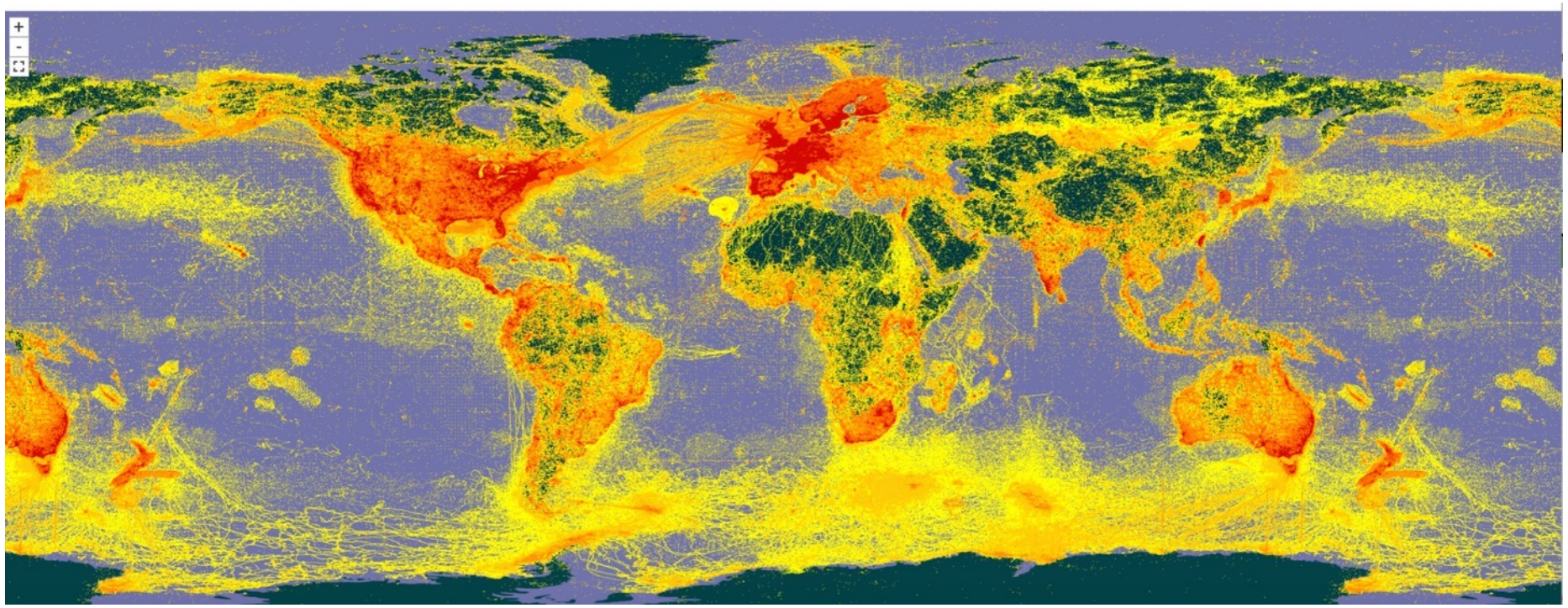
2,169,156,841
Occurrence records

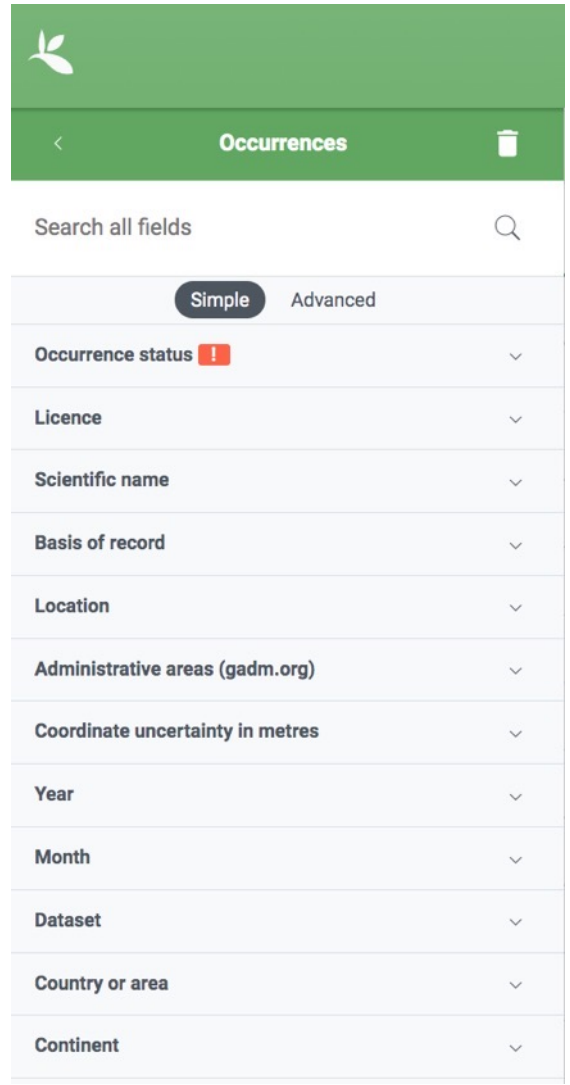
69,336
Datasets

1,835
Publishing institutions

7,227
Peer-reviewed papers using data

On your own: Downloading data: GBIF





On your own: Downloading data: iDigBio & GBIF

- ridigbio
- rgbif
- spocc: Interface to Species Occurrence Data Sources

Why is cleaning data an important step, and why and how do we clean our data?

Exploring Concepts

BiotaPhy

Occurrence Records:

- Specimen information: typically reproduced directly from labels
- Aggregators do only minimal or no cleaning
- Errors may result
- May need to modify set of records

Exploring Concepts

Examples of Data Cleaning Issues:

- Resolve taxon names
- Remove duplicates (physical & electronic)
- Clean localities
  - Round up the latitude/longitude
  - Remove coordinates at 0,0
  - Remove coordinates in cultivated zones, botanical gardens, etc.
  - Remove coordinates outside of the desired range
- Spatial correction to optimize data set for modeling

Exploring Concepts

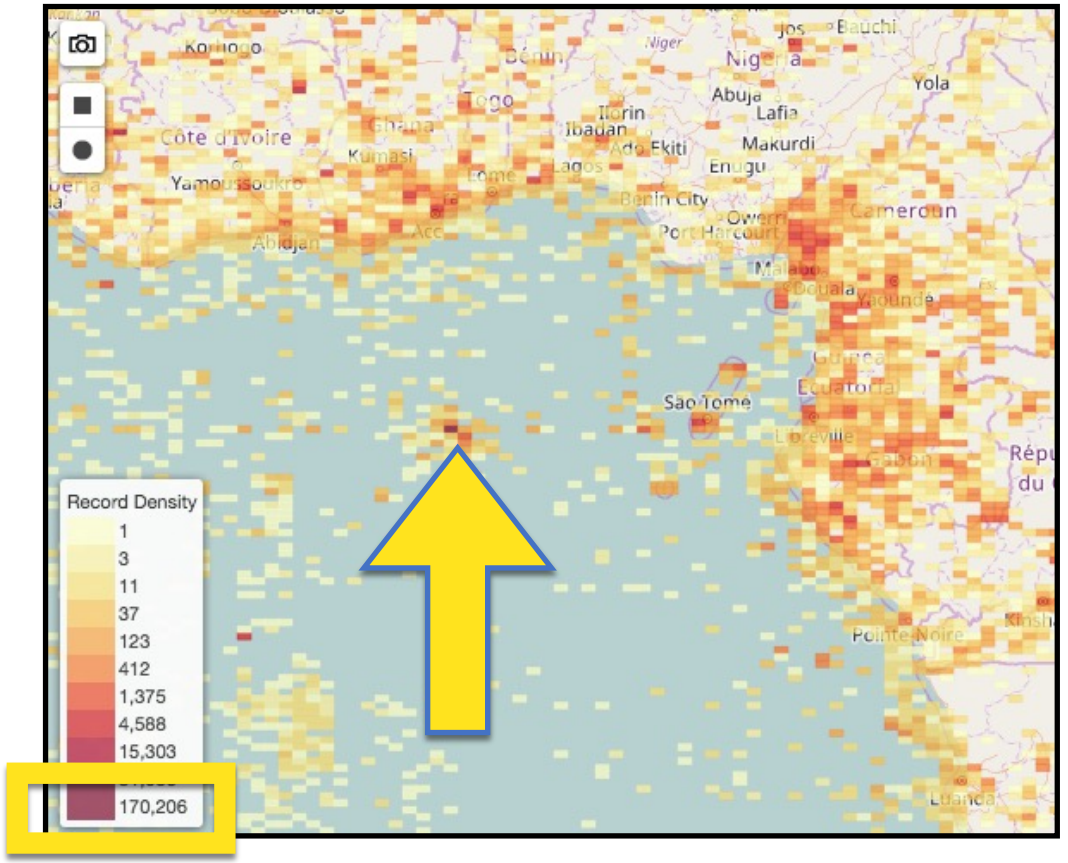
Examples of Data Cleaning Issues:

- ✓ Resolve taxon names
- ✓ Remove duplicates (physical & electronic)
- Clean localities
  - Round up the latitude/longitude
  - Remove coordinates at 0,0
  - Remove coordinates in cultivated zones, botanical gardens, etc.
  - Remove coordinates outside of the desired range
- Spatial correction to optimize data set for modeling

Exploring Concepts

Examples of Data Cleaning Issues:

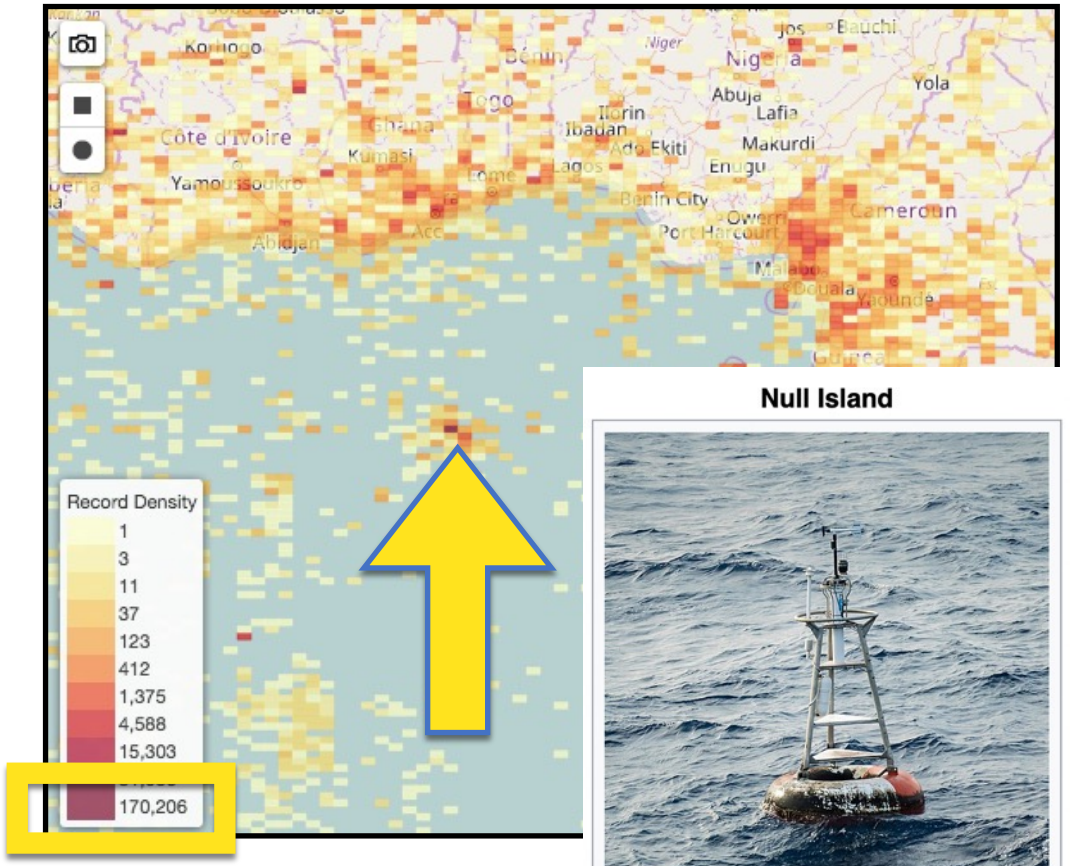
- Resolve taxon names
- Remove duplicates
- Clean localities
  - Round up the latitude/longitude
  - Remove coordinates at 0,0
  - Remove coordinates in cultivated zones, botanical gardens, etc.
  - Remove coordinates outside of the desired range
- Spatial correction to optimize data set for modeling



Exploring Concepts

Examples of Data Cleaning Issues:

- ✓ Resolve taxon names
- ✓ Remove duplicates
- Clean localities
  - Round up the latitude/longitude
  - Remove coordinates at 0,0
  - Remove coordinates in cultivated zones, botanical gardens, etc.
  - Remove coordinates outside of the desired range
- Spatial correction to optimize data set for modeling

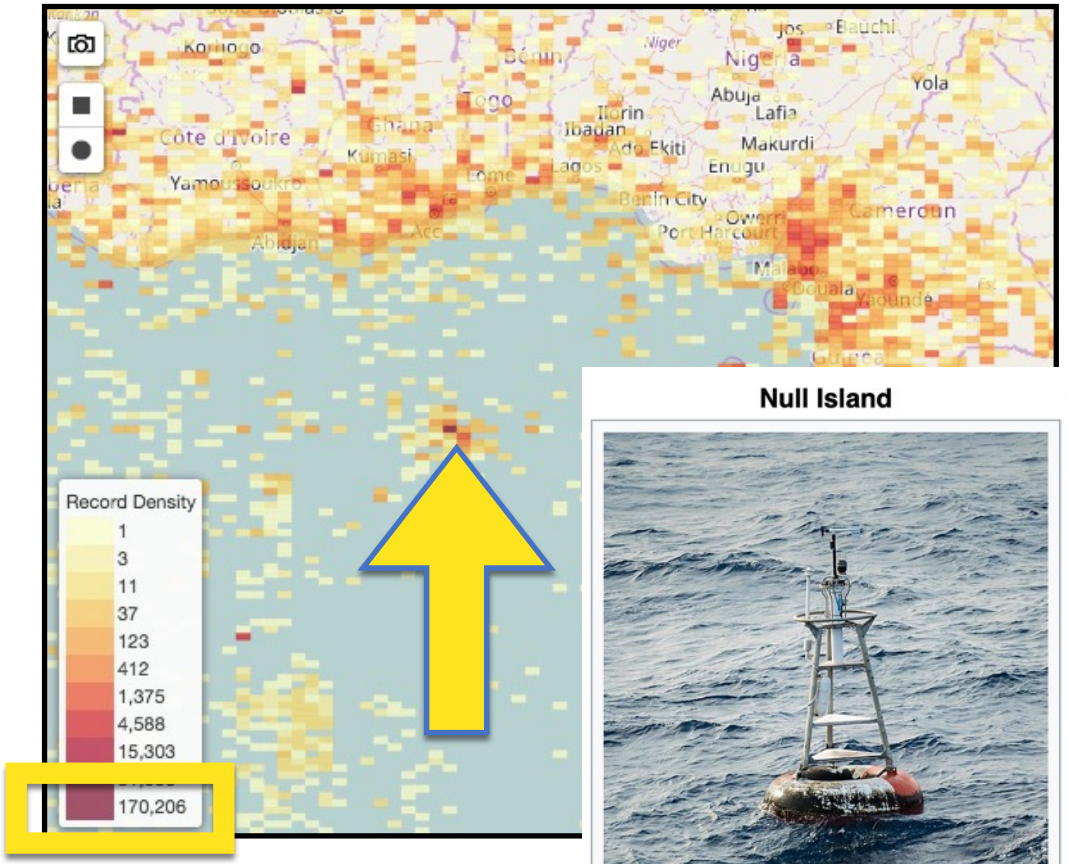


The weather buoy moored at the coordinates of Null Island, in the Gulf of Guinea at 0°N 0°E

Exploring Concepts

Examples of Data Cleaning Issues:

- ✓ Resolve taxon names
- ✓ Remove duplicates
- ✓ Clean localities
  - ✓ Round up the latitude/longitude
  - ✓ Remove coordinates at 0,0
  - ✓ Remove coordinates in cultivated zones, botanical gardens, etc.
  - ✓ Remove coordinates outside of the desired range
- ✓ Spatial correction to optimize data set for modeling



The weather buoy moored at the coordinates of Null Island, in the Gulf of Guinea at 0°N 0°E

Time to Exercise!

What happens when there is a large dataset?

Should we manually clean thousands to millions of records?

We use BiotaPhy tools to automate cleaning data!

Time to Exercise!

Let's put the automated framework developed by BiotaPhy to test!

How to clean your data:

3 steps:

- ✓ **Data Preparation**
- ✓ **Run Tutorial**
- ✓ **Inspect Output**

Input: Occurrence records

Input: Wrangler configuration files

Input: Script parameter file

[Link1]

[Link2]

Before we start ...

Download for the first time OR update the tutorials repository containing test data and configurations.

Initial download:

```
git clone https://github.com/biotaphy/tutorials
```

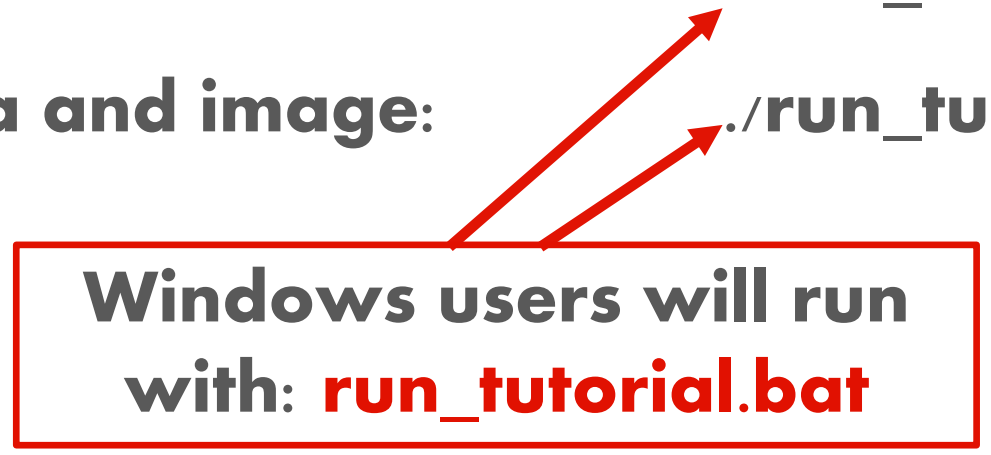
Update tutorial:

```
cd tutorials
git pull
```

Before we start …

Let's rebuild our Docker images to incorporate any updates. Move to the directory containing the tutorials repository that you downloaded or updated.

- ✓ Remove old docker elements: ./run_tutorial.sh cleanup_all
- ✓ Rebuild data and image: ./run_tutorial.sh build_all



Data Preparation: occurrence records

Data preparation

Input: occurrence records

The wrangle_occurrences tool accepts either a Darwin Core Archive (DwCA) file or a CSV file containing records for one or more taxa. More information is in the **Occurrence Data** section of Specimen Occurrences: Data and Wrangling.

	species_name	x	y
2	Bensoniella oregona	-123.751	42.802
3	Bensoniella oregona	-123.7903	42.802
4	Bensoniella oregona	-123.7903	42.802
5	Bensoniella oregona	-123.7707	42.7873
6	Bensoniella oregona	-123.751	42.9927
7	Bensoniella oregona	-123.9646	42.7788
8	Bensoniella oregona	-123.7117	42.9047
9	Bensoniella oregona	-123.7117	42.9047
10	Bensoniella oregona	-123.8266667	42.625

[Link3]
[Link4]

Data Preparation: occurrence records

Structure of a CSV file containing occurrence records

Species name | **Longitude** | **Latitude**

	species_name	x	y
2	Bensoniella oregona	-123.751	42.802
3	Bensoniella oregona	-123.7903	42.802
4	Bensoniella oregona	-123.7903	42.802
5	Bensoniella oregona	-123.7707	42.7873
6	Bensoniella oregona	-123.751	42.9927
7	Bensoniella oregona	-123.9646	42.7788
8	Bensoniella oregona	-123.7117	42.9047
9	Bensoniella oregona	-123.7117	42.9047
10	Bensoniella oregona	-123.8266667	42.625

Data Preparation: wrangler file

Input: Wrangler configuration file

The tool allows multiple operations, defined in a wrangler configuration file, to be applied to the data at the same time. A data wrangler configuration is a file containing a JSON list of zero or more wranglers - each performs a different operation, and each has its own parameters. The file is specified in the Script parameter file described above.

More information on file format, available wrangler types, and the required and/or optional parameters for each are in the **Occurrence Wrangler Types** section of data_wrangle_occurrence.

In this example, we will resolve occurrence data names with GBIF using the AcceptedNameOccurrenceWrangler and also the DecimalPrecisionWrangler, which filters out points without a certain number of digits past the decimal point. Our wrangler configuration file occ_wrangle_resolve.json contains these parameters.

[Link5]

Data Preparation: wrangler file

Input: Wrangler configuration file

In this example, we will resolve occurrence data names with GBIF using the AcceptedNameOccurrenceWrangler and also the DecimalPrecisionWrangler, which filters out points without a certain number of digits past the decimal point. Our wrangler configuration file occ_wrangle_resolve.json contains these parameters.

13 lines (13 sloc) | 345 Bytes

```
1  [
2      {
3          "wrangler_type": "AcceptedNameOccurrenceWrangler",
4          "name_resolver": "gbif",
5          "out_map_filename": "/volumes/output/occ_wrangle_resolve.namemap",
6          "map_write_interval": 100,
7          "out_map_format": "json"
8      },
9      {
10         "wrangler_type" : "DecimalPrecisionFilter",
11         "decimal_places" : 4
12     }
13 ]
```

Data Preparation: wrangler file

Different types of wranglers

- *AcceptedNameOccurrenceWrangler*
- *AttributeFilterWrangler*
- *AttributeModifierWrangler*
- *BoundingBoxFilter*
- *CommonFormatWrangler*
- *CoordinateConverterWrangler*
- *DecimalPrecisionFilter*
- *DisjointGeometriesFilter'*
- *IntersectGeometriesFilter*
- *MinimumPointsWrangler*
- *SpatialIndexFilter*
- *UniqueLocalitiesFilter*

[Link6]

Data Preparation: wrangler file

AcceptedNameOccurrenceWrangler

The AcceptedNameOccurrenceWrangler matches the value in the occurrence data identified as the "species" field with an "accepted name" as defined in a name-map or by a taxonomic service.

- optional
  - **name_map (str or dict): A dictionary or filename containing a dictionary of original**

    name to accepted name. Defaults to None, but either this or name_resolver **must be** provided.

  - name_resolver (str or Method): Use this method for getting new accepted names. If set to 'gbif' or 'otol', use GBIF or OTOL name resolution respectively. Defaults to None, but either this or name_map must be provided.
  - out_map_filename (str): Output for name-mapping between original and accepted names. This file is then acceptable for use as a **name-map** input for subsequent name wrangling. Defaults to None.
  - map_write_interval (int): Interval at which to write records to disk. Used to ensure that if something fails, all is not lost. Defaults to 100.
  - out_map_format (str): Type of file format for out_map_filename, defaults to "json".
  - store_original_attribute (str): A new attribute to store the original taxon name.

Data Preparation: wrangler file

AttributeFilterWrangler

The AttributeFilterWrangler filters out points based on whether the value in the given attribute passes the given function.

- required
  - attribute_name (str): The name of the attribute to modify.
  - filter_func (Method): A function to be used for the pass condition.

Data Preparation: wrangler file

AttributeModifierWrangler

The AttributeModifierWrangler modifies a newly added or existing attribute, computing the value with the given function.

- required
  - attribute_name (str): The name of the attribute to modify.
  - attribute_func (Method): A function to generate values for a point.

Data Preparation: wrangler file

BoundingBoxFilter

The BoundingBoxFilter filters out occurrence points if they do not fall within the given bounding box.

- required
  - min_x (numeric): The minimum 'x' value for the bounding box.
  - min_y (numeric): The minimum 'y' value for the bounding box.
  - max_x (numeric): The maximum 'x' value for the bounding box.
  - max_y (numeric): The maximum 'y' value for the bounding box.

Data Preparation: wrangler file

CommonFormatWrangler

The CommonFormatWrangler modifies points to a common format, using the given attribute-map between the original fields, and the desired fields in the common format.

- required
  - attribute_map (dict): A mapping of source key, target values.

Data Preparation: wrangler file

CoordinateConverterWrangler

The CoordinateConverterWrangler modifies occurrence points by transforming the x and y coordinates from one projection (coded as an EPSG number) into another projection. The new coordinates overwrite the x and y fields. If original_x_attribute and original_y_attribute are provided, these should be new fields in which to save the original x and y coordinates.

- required
  - target_epsg (int): Target map projection specified by EPSG code.
- optional
  - source_epsg (int): Source map projection specified by EPSG code. Either this or epsg_attribute MUST be provided.
  - epsg_attribute (str or None): A point attribute containing EPSG code. Either this or source_epsg MUST be provided.
  - original_x_attribute (str): An attribute to store the original x value.
  - original_y_attribute (str): An attribute to store the original y value.

Data Preparation: wrangler file

DecimalPrecisionFilter

The DecimalPrecisionFilter filters out occurrence points where one or both coordinates have values where the number of digits to the right of the decimal point is less than the given number.

- required:
  - decimal_places (int): Only keep points with at least this many decimal places of precision.

Data Preparation: wrangler file

DisjointGeometriesFilter'

The DisjointGeometriesFilter filters out points where the coordinates intersect with the given geometries.

- required:
  - geometry_wkts (list of str): A list of geometry WKTs to check against.

Data Preparation: wrangler file

IntersectGeometriesFilter

The IntersectGeometriesFilter filters out points where the coordinates do NOT intersect with the given geometries.

- required:
  - geometry_wkts (list of str): A list of WKT strings.

Data Preparation: wrangler file

MinimumPointsWrangler

See the Note above for important information on the use of this wrangler.

The MinimumPointsWrangler filters out groups of points where the number of points in a group does not meet the minimum.

- required:
  - minimum_count (int): The minimum number of points in order to keep all.

Data Preparation: wrangler file

SpatialIndexFilter

The SpatialIndexFilter filters out points that match some given condition (check_hit_function) on the given spatial index.

- required:
  - spatial_index (SpatialIndex): A SpatialIndex object that can be searched.
  - intersections_map (dict): A dictionary of species name keys and corresponding valid intersection values.
  - check_hit_func (Method): A function that takes two arguments (search hit, valid intersections for a species) and returns a boolean indication if the hit should be counted.

Data Preparation: wrangler file

UniqueLocalitiesFilter

See the Note above for important information on the use of this wrangler.

The UniqueLocalitiesFilter filters out points from a grouping that do not have unique coordinates. The filter can operate on one or more groups, and uniqueness is only checked within groups.

- optional parameters:
  - do_reset (bool): Reset the list of seen localities after each group.

Data Preparation: script file

Input: Script parameter file

A JSON parameter file is required for this command. The tutorial parameter file is wrangle_occurrences_resolve.json. These are the required and optional parameters:

- Required:
  - **reader_filename**: The input CSV file of occurrence records.
  - **writer_filename**: A file location to write cleaned points.
  - **wrangler_configuration_file**: occurrence wrangler configuration file, described in the next section. The tutorial example wrangler configuration contains several wranglers, the DecimalPrecisionFilter, the UniqueLocalitiesFilter, MinimumPointsWrangler, and the AcceptedNameOccurrenceWrangler, and is in occ_wrangle_resolve.json

Input CSV

Output for CSV cleaned points

Reminder

Reader_filename = Input
Writer_filename = Output

Choose one or more of the wranglers specified previously!

[Link7]

Data Preparation: script file

- Optional
  - **species_key**: The field name of the input file column containing species data. The default value is *species_name*, so if the data contains any other column name for the field to group on, this must be specified.
  - **x_key**: The field name of the input file column containing x/longitude coordinate. The default value is *x*, so if the data contains any other column name for the x coordinate, this must be specified.
  - **y_key**: The field name of the input file column containing y/latitude coordinate. The default value is *y*, so if the data contains any other column name for the y coordinate, this must be specified.
  - **geopoint**: The field name of the input file column containing a JSON encoded geopoint (iDigBio data uses this field), with sub-elements containing the x and y keys with their coordinates. The default value is None.
  - **log_filename**: Output filename to write logging data
  - **log_console**: 'true' to write log to console
  - **report_filename**: output filename with data modifications made by wranglers

Species names

Longitude

Latitude

Log filename

Logging appears on console

Modifications done by wrangler

Data Preparation: script file

Input: Script parameter file

A JSON parameter file is required for this command. The tutorial parameter file is wrangle_occurrences_resolve.json. These are the required and optional parameters:

11 lines (11 sloc) | 477 Bytes

Raw Blame

```
{
    "species_key": "species_name",
    "x_key": "x",
    "y_key": "y",
    "report_filename": "/volumes/output/wrangle_occurrences_w_resolve_report.json",
    "log_filename": "/volumes/output/wrangle_occurrences_w_resolve.log",
    "log_consoleq": true,
    "reader_filename": "/volumes/data/input/heuchera.csv",
    "writer_filename":  "/volumes/output/heuchera_clean_accepted.csv",
    "wrangler_config_filename":   "/volumes/data/wranglers/occ_wranglers_w_resolve.json"
}
```

Let's run this tutorial!

Run tutorial

Initiate the process with the following:

```
# Data cleaning and name resolution
./run_tutorial.sh wrangle_occurrences  data/config/wrangle_occurrences_resolve.json
```

Goal: produce a file containing edited occurrence data according to the wrangler(s) used!

Windows users will run with: run_tutorial.bat

Remember, you will RUN this code in the terminal or in the command prompt for Windows!

Let's look at the output!

Output

This process outputs files configured in the script parameter file:

2. If *report_filename* is specified in the script parameter file, a summary of name resolutions, like wrangle_occurrences.log

SECONDARY OUTPUT

zzeppozz tutorial ... Latest commit e696d1f 18 hours ago History

1 contributor

6224 lines (6224 sloc) | 698 KB Raw Blame

```
1  09 Sep 2022 15:14 INFO     AcceptedNameOccurrenceWrangler: Filtered 0, modified 0 of 0 assessed points.
2  09 Sep 2022 15:14 INFO     DecimalPrecisionFilter: Filtered 0, modified 0 of 0 assessed points.
3  09 Sep 2022 15:14 DEBUG    AcceptedNameOccurrenceWrangler: Resolution identical: Bensoniella oregona
4  09 Sep 2022 15:14 DEBUG    AcceptedNameOccurrenceWrangler: Return 51 of 51 points, total of 0 modified
5  09 Sep 2022 15:14 INFO     wrangle_occurrences: AcceptedNameOccurrenceWrangler wrangled 51 from /volumes/data/input/heuchera.csv
6  09 Sep 2022 15:14 INFO     DecimalPrecisionFilter: Bensoniella oregona -123.751, 42.802 fails precision test.
7  09 Sep 2022 15:14 INFO     DecimalPrecisionFilter: Bensoniella oregona -123.7903, 42.802 fails precision test.
8  09 Sep 2022 15:14 INFO     DecimalPrecisionFilter: Bensoniella oregona -123.7903, 42.802 fails precision test.
```

Let's look at the output!

3. If *log_filename* is specified in the script parameter file, a report like wrangle_occurrences.rpt containing a summary of the processing.

4. If *log_console* is specified in the script parameter file, logs will be written t e command prompt during execution.

SECONDARY OUTPUT

20 lines (20 sloc) | 445 Bytes — Raw | Blame

```
{
    "input_records": 11489,
    "output_records": 8203,
    "wranglers": [
        {
            "name": "AcceptedNameOccurrenceWrangler",
            "version": "1.0",
            "assessed": 0,
            "modified": 0,
            "filtered": 0
        },
        {
            "name": "DecimalPrecisionFilter",
            "version": "1.0",
            "assessed": 0,
            "modified": 0,
            "filtered": 0
        }
    ]
}
```

Let's look at the output!

4. an output file with occurrence records named in the writer_filename, like heuchera_wrangled.csv containing the occurrence records, one record per line. Note that the species_name field now contains the new taxonomic name resolved for each record. If the original records contain other attributes, those will be retained with their original values.

If the wrangler configuration file contains the AcceptedNameOccurrenceWrangler, as in the command above, using the wrangle_occurrences_resolve.json configuration file, the process produces one additional file as configured in that wrangler configuration:

PRIMARY OUTPUT

8204 lines (8204 sloc) | 328 KB

Raw | Blame

Search this file...

	species_name	x	y
2	Bensoniella oregona	-123.7707	42.7873
3	Bensoniella oregona	-123.9646	42.7788
4	Bensoniella oregona	-123.7117	42.9047
5	Bensoniella oregona	-123.7117	42.9047
6	Bensoniella oregona	-123.8038889	42.60861111

Let's look at the output!

If the wrangler configuration file contains the AcceptedNameOccurrenceWrangler, as in the command above, using the wrangle_occurrences_resolve.json configuration file, the process produces one additional file as configured in that wrangler configuration:

- An *out_map_filename* containing a name-map from the AcceptedNameOccurrenceWrangler. The name-map is a JSON file with pairs of names - the original name to the accepted name according to the specified authority. This name-map is suitable to use for input when resolving another dataset containing a subset of the same original names. A sample output name-map is occ_wrangle_resolve.namemap.

PRIMARY OUTPUT

66 lines (66 s…) | 3.04 KB | Raw | Blame


```
{
    "Bensoniella oregona": "Bensoniella oregona",
    "Conimitella williamsii": "Conimitella williamsii",
    "Elmera racemosa": "Elmera racemosa",
    "Heuchera abramsii": "Heuchera abramsii",
    "Heuchera acutifolia": "Heuchera acutifolia",
    "Heuchera alba": "Heuchera alba",
    "Heuchera americana": "Heuchera americana",
    "Heuchera bracteata": "Heuchera bracteata",
    "Heuchera brevistaminea": "Heuchera brevistaminea",
```

Session Summary, Q&A and Discussion

Multiple ways to obtain occurrence records:

- iDigBio, GBIF, other aggregators

Portals, APIs, R packages

Data can be very dirty!

- errors in names, spelling, locality descriptions, GPS coordinates, more!

Duplicates (physical & electronic)

May need to optimize (remove certain areas, reduce density of samples, etc.)

Save your clean data and consider making data set available online!

Any questions??

Please use the Chat to ask your question!