**BIEN2 Validation Workflow**

**Contents**

1. Create & populate table `geoscrub` 1

2. Create & populate political division authority tables 1

3. Name resoltuion of political divisions (country, stateProvince, countyParish) 3

4. Geovalidation 4

4. Detection and flagging of cultivated specimens 5

5. Detection and flagging of FIA plots from plantations and logged areas 6

6. Transfer of results to bien2 database 6

All of the validations below are performed in database `geoscrub` on nimoy, unless otherwise noted.

## 1. Create & populate table `geoscrub`

* Scripts in: geoscrub/create\_geoscrub/
* Master script: create\_geoscrub.php
  + calls all others
* Creates table `geoscrub` & populates with verbatim locality information from two locations in bien2 core database:
  + bien2.IndividualObservation JOIN bien2.SpecimenObservation (specimen localities)
  + bien2.PlotMetaDataDimension (plot localities)
* Indentifies records by original PK + table name
* Table includes additional fields to hold results of standardization of political division names and validation of geocoordinates
* Transfers original identifiers (table + table PK), verbatim political division names, coordinates, elevations and locality descriptions. Currently, only political division names are standardized, and coordinates validated against the standardized political division names; verbatim elevation and locality\_description fields are not currently used, but were included for forward compatibility for georeferencing (not yet implemented)
* See Fig. 1

## 2. Create & populate political division authority tables

1. **Country**

* Scripts in: geoscrub/create\_country/
* Master script: create\_country.php
* See Fig. 1
* Creates the following tables:
  + country
  + countryName
  + countryStaging
  + countryNameStaging
* Populates with reference data in files country.txt, countryName.txt
* Includes 2-char and 3-char ISO county codes (stable) in addition to generating artificial auto\_increment PK.
* Staging tables are deleted after normalization

1. **StateProvince**

* Scripts in: geoscrub/create\_stateProvince/
* Master script: create\_stateProvince.php
* See Fig. 1
* Creates the following tables:
  + stateProvince
  + stateProvinceName
  + stateProvinceStaging
  + stateProvinceNameStaging
* Populates with reference data in files stateProvince.txt, stateProvinceName.txt.
* The name used as the "standard name" is converted to plain ascii (although I notice this is inconsistent for some reason…perhaps the function did not succeed in converting all extended ascii characters?)
* In addition to generating an artificial (auto\_increment) PK (stateProvinceID), includes 2-char stateProvince code (not universal, some missing), HASC\_1 codes (unique but some missing), a "standard" full name (chosen arbitrarily by me) and a unique code consisting of the country ISO code and the stateProvince name joined by an underscore.
* Provinces are joined to parent country initially by ISO codes, and later by the artificially generated countryID
* Determines and adds plain ascii equivalents of political division names
* Staging tables are deleted after normalization

1. **CountyParish**

* Scripts in: geoscrub/create\_ countyParish/
* Master script: create\_ countyParish.php
* See Fig. 1
* Creates the following tables:
  + countyParish
  + countyParishName
  + countyParishStaging
  + countyParishNameStaging
* Reference data in files countyParish.txt, countyParishName.txt are loaded first to the staging tables, then normalized to the primary reference tables (countyParish, countryParishName)
* The name used as the "standard name" is converted to plain ascii.
* Table `countyParish` consists of an artificial (auto\_increment) PK (countyParishID), a FK linking to the artificial PK (stateProvinceID) of the parent stateProvince, a standard plain ascii version of the name, to be used as the "standard" name, a unique code (made by me) consisting of the country ISO code and the countyParish name joined by an underscore, and the (mostly) stable HASC\_2 code.
* Table `countyParishName` contains synonyms of countyParish names, and consists of an artificial auto\_increment PK (countyParishNameID), the FK linking to the master entry for that political division in table `countyParish` (countyParishID), and countyParishName.
* Staging tables are deleted after normalization

## 3. Name resoltuion of political divisions (country, stateProvince, countyParish)

1. **Country**

* Scripts in: geoscrub/scrub\_country/
* Master script: country\_scrubbed.php
* Creates table `countryScrubbed` (see Fig. 1)
* Inserts into above table all unique verbatim country names from table `geoscrub`
* Attempts to link verbatim name to standard country in table `country` by joining by (i) 2-char ISO code and (ii) 3-char ISO code.
* Attempts to link verbatim name to standard name via synonyms, after converting any unconverted utf8, extended ascii and plain ascii codes in name to to plain ascii text
* Transfers results of scrubbing (countryID) to table `geoscrub`

1. **StateProvince**

* Scripts in: geoscrub/scrub\_stateProvince/
* Master script: poldiv\_scrubbed.php
* Creates table ` stateProvinceScrubbed` (see Fig. 1)
* Inserts into above table all unique verbatim country names from table `geoscrub`
* Attempts to link verbatim name to standard stateProvince in table `stateProvince` by joining by (i) independent code (HASC\_1) and (ii) dependent code (must be paired with country identifier).
* Attempts to link verbatim name to standard name via synonyms, after converting any unconverted utf8, extended ascii and plain ascii codes in name to to plain ascii text
* Transfers results of scrubbing (stateProvinceID) to table `geoscrub`

1. **CountyParish**

* Scripts in: geoscrub/scrub\_countyParish/
* Master script: poldiv\_scrubbed.php
* Creates table ` countyParishScrubbed ` (see Fig. 1)
* Inserts into above table all unique verbatim country names from table `geoscrub`
* Attempts to link verbatim name to standard countyParishin table `countyParish` by joining by (i) independent code (HASC\_2) and (ii) dependent code (must be paired with country identifier).
* Attempts to link verbatim name to standard name via synonyms, after converting any unconverted utf8, extended ascii and plain ascii codes in name to to plain ascii text
* Transfers results of scrubbing (countyParishID) to table `geoscrub`

## 4. Geovalidation

* Record identifiers (artificial PK geoscrubID), verbatim coordinates and standardized 1st, 2nd and 3rd political divisions (country, stateProvince, countyParish) are extracted from table `geoscrub`.
* Standardized political division names determined in step 3 above are used to validate or reject coordinates by performing point-in-polygon fits to political division polygons.
* Note that geovalidation **also** requires that names of political division shape files be standardized and linked to the political division authority tables populated in Step 2 above.
* Each point is coded as in or out of its declared political division, and distance from political division polygon in miles is estimated
* See separate description by John Donoghue of his point-in-polygon validation procedure
* Results of geovalidation are transferred back by joining results to table to `geoscrub` by geoscrubID.
* Results are stored in table `geoscrub` as the following fields:
  + **isInCountry**: 0=out, 1=in, NULL=not validated
  + **distErrCountry**: distance in km from country boundary. Equals 0 if isInCountry=1. NULL if not validated.
  + **isInStateProvince**: 0=out, 1=in, NULL=not validated
  + **distErrStateProvince**: distance in km from stateProvince boundary. Equals 0 if isInStateProvince =1. NULL if not validate.
  + **isInCountyParish**: 0=out, 1=in, NULL=not validated.
  + **distErrCountyParish**: distance in km from countyParish boundary. Equals 0 if isInCountyParish =1. NULL if not validated.
* NULL values for isInCountry, isInStateProvince and isInCountyParish occur when geovalidation status is unknown because a standard political division could not be assigned to that record.
* As of BIEN2, geovalidation was performed for all countries, and for stateProvince only for the United States, Mexico and Brazil. Geovalidation using countyParish was not performed
* Thus, selecting strictly "geovalid" records from the geoscrub table would require the following WHERE criteria:

WHERE isInCountry=0 AND (isInStateProvince=1 OR isInStateProvince IS NULL)

* Using a more relaxed error buffer of 10 km, the above condition becomes:

WHERE (isInCountry=0 OR distErrCountry<=10) AND (isInStateProvince=1 OR distErrStateProvince<=10 OR isInStateProvince IS NULL)

* An equivalent condition to the preceding for table bien2.viewFullOccurrence would be:

WHERE (countryError<=10) AND (ProvinceError<=10 OR ProvinceError IS NULL)

* And for table bien\_web.observation:

WHERE (isGeovalid=1)

Note that `bien\_web`.`observation`.`isGeovalid` incorporates the complex conditions of the preceding conditions.

## 4. Detection and flagging of cultivated specimens

* Specimens of cultivated plants can grossly distort range models
* These scripts attempt to detect and exclude cultivated specimen observations, and also flag plot observations from plots in human-altered forests such as plantations and logging treatments
* Scripts in: geoscrub/cultivated/
* Results tranferred to bien2.viewFullOccurrence are in geoscrub/cultivated/update\_view/ (not sure about geoscrub!)
* Updates the following two fields in `geoscrub` (transferred to both bien2.viewFullOccurrence and bien\_web.observation):
  + **isCultivated**: 1=yes, 2=no, NULL=not checked
  + **isCultivatedReason**: which of four methods below used as basis for setting isCultivated=1
* Four methods:

1. Original "isCultivated" flag. Some databases (e.g., ARIZ herbarium) flag cultivated specimens themselves. This information was not captured in the bien2 database, but could be retrieved in some cases from original data.
2. Keywords in locality and specimen descriptions (see geoscrub/cultivated/cult\_by\_locality/). Wildcard searches in English, Spanish, Portuguese and French for key words equivalent to "cultivated", also "plantation", "garden", "farm", etc.
3. Proximity to herbarium or botanical garden (see geoscrub/cultivated/cult\_by\_herbaria/). Specimens within 3 km radius of a herbarium are excluded. Coordinates of herbaria obtained from Index Herbariorum.
4. Proximity to city (see geoscrub/cultivated/cult\_by\_city/). NOT USED. We had originally intended to exclude specimens within a certain radius of cities (scaled by population) but decided not to use this method due to the high rate of false positives.

## 5. Detection and flagging of FIA plots from plantations and logged areas

* These scripts attempt to detect and exclude FIA plots human-altered forests such as plantations and logging treatments
* Based on fields in FIA COND table
* Scripts that build table `geoscrub`.`FIA\_COND` are in: geoscrub/fia/
* Results tranferred to bien2.viewFullOccurrence (not geoscrub!) are in geoscrub/cultivated/update\_view/
* Updates the following two fields in `geoscrub` (transferred to both bien2.viewFullOccurrence and bien\_web.observation):
  + **isCultivated**: 1=yes, 2=no
  + **isCultivatedReason**: if isCultivated=1, this field is set to 'FIA disturbed'

## 6. Transfer of results to bien2 database

* Validation results are NOT transferred to core bien2 database tables, nor are they transferred directly to analytical database tables bien2.viewFullOccurrence and bien\_web.observation
* Instead, they are transferred to two new tables in bien2:
  + geoIndividualObservation
  + geoPlotMetaDataDimension
* These tables are essentially copies of the table geoscrub, with verbatim and validation columns separated out into two separate tables corresponding to the original source tables.
* These validation results are then transferred separately to bien2.viewFullOccurrence by Rick Condit's SQL procedure that creates the table.
* Scripts in:
* geoscrub/create\_geovalidation\_tables/
  + Master script: create\_geovalidation\_tables.php
* geoscrub/update\_geovalidation\_tables/
  + Master script: update\_geovalidation\_tables.php

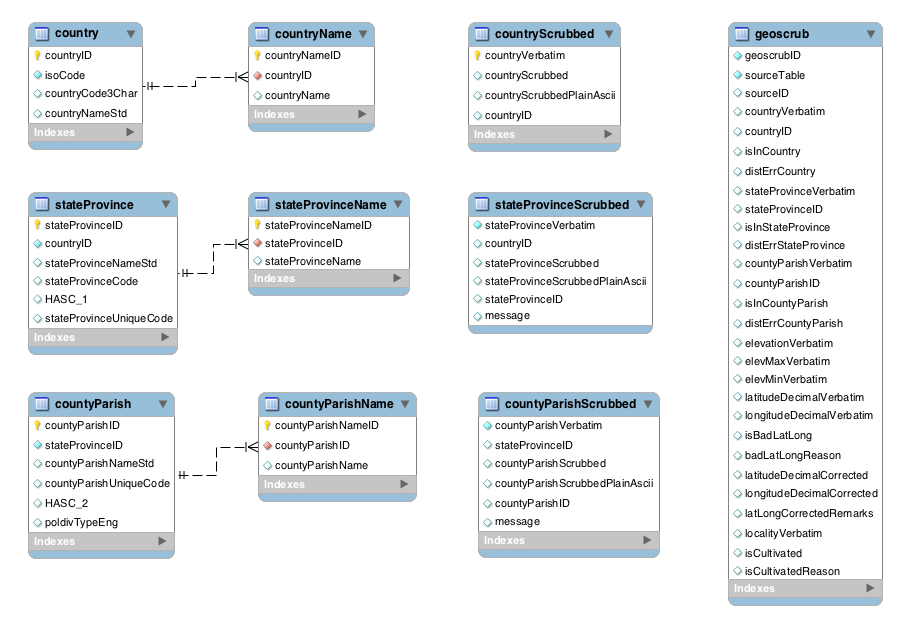


Fig. 1. Geoscrubbing database core schema. Table `geoscrub` contains locality information for all records in tables `IndividualObservation`,`SpecimenObservation` and `PlotMetaDataDimension` in database `bien2`.