ForestGEO Database Handbook

Written by ForestGEO Systems Group, April 2014
Suzanne Lao, Shameema Esufali, Anudeep Singh, & Richard Condit

Updated and Revised, Summer 2020
Suzanne Lao, Shameema Esufali, Jessica Shue, Lauren Krizel, Caly McCarthy, & Anudeep Singh

ForestGEO
GLOBAL EARTH OBSERVATORY NETWORK

Smithsonian

Table of Contents

INTRODUCTION

Welcome to the *ForestGEO Database Handbook*! We hope that it will be a helpful reference as you learn how to navigate the ForestGEO database landscape.

Let's begin by introducing some important terms that you will encounter in the Handbook:

ForestGEO Systems Group: the group of people who work to maintain the ForestGEO database landscape and to help data managers interface with it. If you have questions about any content in the Handbook, please reach out to Suzanne Lao (laoz@si.edu) or Shameema Esufali (shameemaesufali@gmail.com), who will be glad to help you.

data manager: a person who takes responsibility for maintaining data from a ForestGEO plot. This person could be the Principal Investigator of the plot or they could be someone on the PI's team with a database background. Responsibilities of a data manager include:

- Creating forms and data entry screens that minimize collection and data entry errors.
- Screening data as soon as it is collected so that errors can be fixed in the field if necessary.
- Continuing to screen data until the only errors displayed are warnings, not fatal errors.
- Keeping records of all errors found and how they were fixed.
- Scanning field forms (if data is collected on paper), and backing up all temporary electronic work files in a timely fashion.
- Uploading all screened data into the production database.
- Backing up the final database after a census.
- Making changes to the database between censuses as necessary, keeping records, backing up all versions.
- Archiving the final database in a repository.

database: a series of tables (each holding only one type of information, *e.g.* one for species, one for site information, one for diameter, etc.) that are linked through defined relationships. There are two versions of the database: "stable" and "current." The stable database is the official database that the ForestGEO Systems Group has a copy of on the server. If a plot is affiliated with the ForestGEO Data Portal, the official copy will be shared when PIs approve data requests from other researchers. Throughout the year PIs may make updates and corrections to the current database. Every one to two years PIs may send the ForestGEO Systems Group the most recent current database, which will get frozen as the stable one.

tables: individual structures that together form a database. They are comprised of fields (represented by columns) that describe a specific type/category of information for every record, and records (represented by rows) that are the individual entries. There are two types of tables: "temporary tables," which are working tables that are deleted from the final database and "production tables," which are permanent tables that have been fully screened.

MySQL: an open source database management system. "SQL" stands for "Standard Query Language" and is a powerful tool that allows users to ask questions of the database, such as "How many trees died in our plot between the first and second census?" You can learn more about SQL here.

> *NOTE*: MariaDB is the current open source version of MySQL (which was purchased several years ago by Oracle and now requires a paid subscription). In nearly all regards MariaDB operates exactly as MySQL does. When we refer to MySQL throughout the Handbook, please know that you can do the same functions in MariaDB.

ctfsweb: a web application written in MySQL, PHP, and Javascript that has been tailored to ForestGEO's needs. With ctfsweb data managers can:

- enter data from field data sheets.
- upload data from text files into the database.
- screen for errors.

- create View tables of frequently requested data by joining several tables back into one big flat file.
- make user-defined reports in text format for downloading.

script: a series of commands that can be run against the database to perform specific functions (*e.g.* make corrections, create tables, upload a text file into a temporary table, etc.). Data managers will author scripts in either MySQL or PHP and are welcome to submit a script to Suzanne or Shameema to review. Suzanne and Shameema are always available to provide advice and examples of code.

Throughout the Handbook we will offer scripts to help you complete certain steps. To familiarize yourself with a LOAD command, for example, visit Appendix A to see what each part of the script means and how it must be formatted.

Another note on scripts: all changes to the database must be scripted. These scripts should be stored with the version of the database that they change (old database + application of scripts = new version of database). When writing scripts, we recommend using Notepad++, Geany, or gedit. They are much more powerful than the standard Wordpad and Notepad in that they can open large files significantly faster, they support smart syntax and code highlighting for many programming languages, they have line numbering, etc.

package: a combination of scripts, text files, data sheets, and README files pertaining to the same set of data. When data managers have completed the final screening of their data, they will send the relevant package to Suzanne or Shameema.

Before we move on with more information, let's pause to note two important things:

1. Data managers, **back up your progress** at every step of the way (in our appendix you can find instructions on how to use mysqldump to back up your MySQL database and tables)! Export your database every time you change it (or weekly if you are making changes often) and store it on a backup drive or in the cloud with Dropbox or Google drive – or even both. When the database is dormant there is no need to back it up on any regular basis. If the ForestGEO Systems Group is hosting your database, it will be backed up on the Smithsonian server.

2. You are about to encounter a lot of details. Why? Having a standardized format for ForestGEO data is an extension of having standardized protocols in the field; they allow us to compare data across sites and gain insight into global forest dynamics.

Below is a key to understanding how we'll convey instructions. Step-by-step instructions will be presented in a box with a bolded "**HOW TO**" header. When you need to issue a command in MySQL, we'll designate that by `MySQL >`. If you see anything else in `this font` it means that you should copy it precisely (unless it's `this font in blue`, in which case, customize that information to your site). If our instructions include "content in quotes," that means that you should look for that phrase on your screen (as a button, tab name, etc.).

HOW TO: Examples

1. Click on "Something Specific."

2. Open MySQL and follow a Command.

```
MySQL > SOURCE createTempTables.sql;
```

2. Type `precisely this`

3. Type `precisely this, 'but-customize-this-piece-to-your-plot'`

The last thing we'll do in this Introduction section is provide a roadmap of the steps involved to move data from the field to a stable database on the Smithsonian server. The table below is just a sketch of these steps; you'll find comprehensive explanations, instructions, and examples in the subsequent pages. We recognize that data managers will reference this handbook for different reasons and have tried to clearly identify where they can find the information they need.

> *NOTE*: The "electronic" designation applies to tablets and to paper data entered into electronic files via some other program (Excel, Filemaker, PostgreSQL, etc.). The "paper" designation is for records that only exist on paper data sheets and will be entered electronically via ctfsweb.

		Applicable?				
Chapter	**Step**	**1st Census: Paper**	**1st Census: Electronic**	**Re-census: Paper**	**Re-census: Electronic**	**Revisions**
1	install XAMPP	X	X			
1	create MySQL database	X	X	X	X	
1	install ctfsweb	X	X			
1	create empty Temp tables	X	X	X	X	
2	format fixed data files	X	X			
2	load fixed data files to ctfsweb	X	X			
3	set up configuration files	X		X		
3	double-enter data	X		X		
3	cross-check double-entered data	X		X		
3	insert location data	X		X		
4	format census data files in staging tables		X		X	
3 & 4	copy text files into temporary tables		X		X	
5	screen data in temporary tables	X	X	X	X	
5	run View Error Summary	X	X	X	X	
5	upload data to production tables	X	X	X	X	
5	run post screening	X	X	X	X	X
5	Run Checksum.sql	X	X	X	X	X
5	write scripts to correct errors in production tables					X
5	create report tables	X	X	X	X	X
6	compile package components	X	X	X	X	X
6	test package	X	X	X	X	X
6	compress & send package	X	X	X	X	X

CHAPTER 1: Creating Your ForestGEO MySQL Database

In order to create a MySQL database specifically for your plot, you will need to have Apache and MySQL installed on your computer. If you already do, proceed to the next section in this chapter, "ctfsweb, a Setup Data Entry and Reporting Tool." If you need to download these systems you can do so by installing XAMPP on your computer.

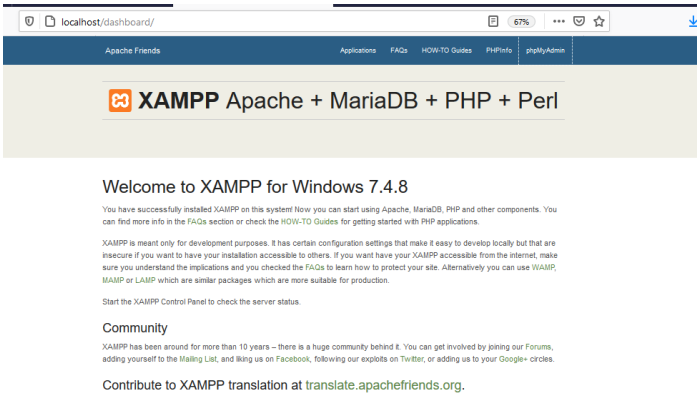
ctfsweb is a web application written in MySQL, PHP, and Javascript that has been tailored to ForestGEO's needs. Some data managers will use it to enter data directly from paper field sheets. All data managers will use it to upload data from text files into the database, to screen for errors, and to create View tables – that is, to populate their MySQL databases.

XAMPP

XAMPP is a free, easy to install Apache distribution containing MariaDB (MySQL Fork), PHP, and Perl.

HOW TO: Install XAMPP on Your Computer

1. Download the latest version of XAMPP: https://www.apachefriends.org/index.html
2. After the download, execute the setup file by following the prompts of the XAMPP Setup Wizard.
3. Select the components to install. Apache, MySQL, PHP, and phpMyAdmin are required. All others are optional.
4. Select the location for installation. If you don't have a preference, keep it as default.
5. Continue moving through the prompts, and you'll reach the installation.
6. After installation, Windows users will receive a security alert. Click "Allow access."
7. Uncheck "Do you want to start the Control Panel now?" and hit "Finish."
8. Search for XAMPP on your computer and right click to select "Run as administrator."
9. Click on the check boxes under "Service" for Apache and for MySQL, then click "Yes" in the confirmation pop-up windows. This will install services for both servers, which will allow both servers to start automatically at reboot.
10. After installing services, click "Start" action buttons to start Apache and MySQL.
11. Open a browser and type "localhost." If XAMPP was installed properly, you will see the following page:



NOTE: For Windows users only: To preserve the lettercase of database and table names, you need to do the following:

a. Open your MySQL configuration file in a text editor:
   `[drive]\xampp\mysql\bin\my.ini`

b. Look for:
   `# The MySQL server [mysqld]`

c. Directly beneath # The MySQL server, add:
   `lower_case_table_names = 2`

d. Save the file and restart MySQL service.

ctfsweb, a Setup Data Entry and Reporting Tool
As mentioned above, ctfsweb is a web application that has been tailored to ForestGEO's needs. Regardless of whether you're working from paper or from electronic records, you will be working with ctfsweb after you create your MySQL database.

HOW TO: Create Your MySQL Database and Install ctfsweb on Your Computer

NOTE: The instructions below are for PCs. Contact the ForestGEO team if you need instructions for installing MySQL/ctfsweb on a Mac.

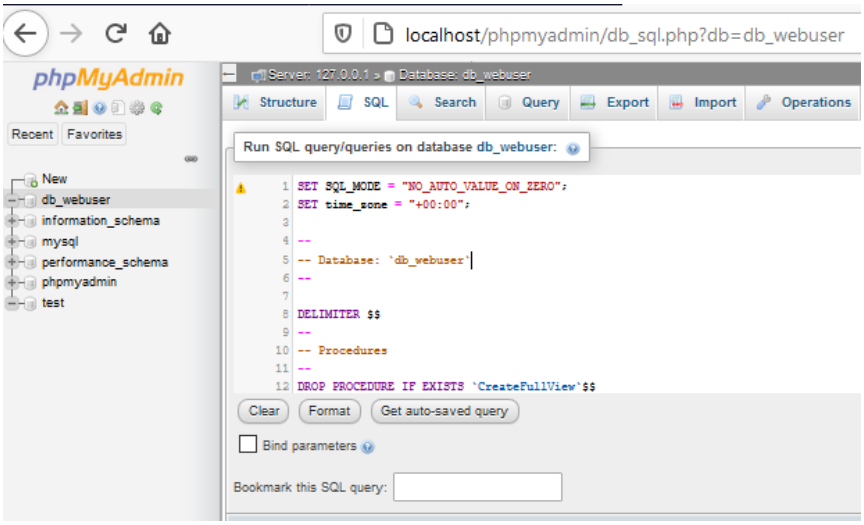
1. Download the ctfsweb package via this link.
   a. Copy the "ctfsweb" folder from the ctfsweb download package to "\xampp\htdocs\"

2. Create "db_webuser" database:

   a. Create a new database called "db_webuser."
      i. Open a MySQL console.
      ii. On your Desktop, click on "Start," then "Run." Or alternatively, go to the search box in the taskbar.
      iii. Type `cmd`, then click on "OK."
      iv. In the DOS window, navigate to the MySQL folder by typing `cd\xampp\bin\mysql`
      v. Go into MySQL by typing `mysql –u root` (and if it asks for a password, just press [ENTER] – no password is needed).
      vi. Create the database:
      ```
      MySQL > CREATE DATABASE db_webuser;
      MySQL > USE db_webuser;
      ```

   b. Use the "ctfsweb_webuser.sql" file included in the download package to create the required metadata tables. If you have a different user other than "root," please update the "DEFINER" information to the correct user. This may be done in a text editor or alternatively using phpMyAdmin. If using phpMyAdmin, open a browser window and type the following url:

      `http://localhost/phpmyadmin/`



This is the ctfsweb_webuser.sql file copied into the SQL tab of the db_webuser database created in step 1.

c. After the "DEFINER" information has been updated, run the "ctfsweb_webuser.sql" script by clicking on the "GO" button.

2. Create the site database:
   a. Open a MySQL console:
      i. On your Desktop, click on "Start," then "Run." Or alternatively, go to the search box in the taskbar.
      ii. Type **cmd**, then click on "OK."
      iii. In the DOS window, navigate to the MySQL folder by typing **cd\xampp\mysql\bin**
      iv. Go into MySQL by typing **mysql –u root** (and if it asks for a password, just press [ENTER] – no password is needed).

NOTE: If you are an administrator of your computer, if the XAMPP directory is not in your DOS path, and if you want to be able to call up MySQL anywhere, carefully do the following:

- Go to your System Settings.
- Type **env** in the Search box.
- Select: "Edit the system environment variables"
- Click on "Environmental Variables" (in the bottom).
- In "System Variables" in the second window, scroll to "Path"
- To the end of the existing path, add **;c:\xampp\mysql\bin**

   b. Type the following in your open MySQL terminal, remembering that:
      i. the names of the databases and the tables are case-sensitive.
      ii. you will replace **current_yoursitename** with the name of your database, e.g. **current_bci**

```
MySQL > CREATE DATABASE current_yoursitename;
MySQL > USE current_yoursitename;
```

   c. If you already have a database that you dumped (i.e. backed up), source that file, inserting your path-to-file and the name of your file, noting that your path cannot contain spaces:

```
MySQL > SOURCE/path-to-file/yoursitename.sql;
```

3. If you are not migrating data from an old database structure, insert empty tables into the

database using the script called "New_Database_Structure.sql:"

```
MySQL > SOURCE New_Database_Structure.sql;
```

Enter the complete taxonomic family and genus names according to the APG system into the Family and Genus tables (via their respective scripts, hyperlinked below):

```
MySQL > SOURCE Family.sql;
MySQL > SOURCE Genus.sql;
```

NOTE: For a list of all the tables in your database and a brief description of their columns, reference the Tree Database Data Dictionary

4. Create the temporary tables needed for ctfsweb by running the "createTempTables.sql" script:

```
MySQL> SOURCE createTempTables.sql;
```

5. Open the file "..\ctfsweb\application\config\database.php" in a text editor to begin updating the database configuration file.

6. Do NOT EDIT the first set (db1) except for the username and password, if different.

7. Use the following set of code for each site database that you need to link with the system, replacing 'db_bci' with the name of your database, and 'bci' with the name of your site. This database name will be required in the next step to add to the list of databases. Replace the username (and password) if different from "root" (if you added a password).

```
$db['db_bci'] = array(
        'dsn'    => '',
        'hostname' => 'localhost',
        'username' => 'root',
        'password' => '',
        'database' => 'bci',
        'dbdriver' => 'mysqli',
        'dbprefix' => '',
        'pconnect' => FALSE,
        'db_debug' => (ENVIRONMENT !== 'production'),
        'cache_on' => FALSE,
        'cachedir' => '',
        'char_set' => 'utf8',
        'dbcollat' => 'utf8_general_ci',
        'swap_pre' => '',
        'encrypt' => FALSE,
        'compress' => FALSE,
        'stricton' => FALSE,
        'failover' => array(),
        'save_queries' => TRUE
);
```

8. Copy the set of lines in "6" for each site database connection.

9. Open link: http://localhost/ctfsweb/index.php/admin

10. Use the password “local@admin” to login to the panel. *NOTE*: Password may change, please refer to the README file that comes with the ctfsweb package.

11. Add plot information:
    - Plot Name: This is the site/database name that will be displayed in the dropdown menu on the login panel.
    - Database Name: This is the name you used in step 7 in place of **'db_bci'**
    - Version of Database: Select “New.” The “Old” version is not used anymore.

12. Add user information:
    - Username: Username for the login system. Use different username for different users.
    - Password: Password for the login system. Use different passwords for different users.
    - Database Name: Select the database name from the list of databases you added in step 11.

CHAPTER 2: Adding Fixed Content to Your Database

In order to add fixed content to your database, you'll need to follow ForestGEO formatting specifications. Below, we outline general formatting considerations, identify specific formatting requirements, and provide step-by-step instructions on loading fixed content into your database.

Formatting Files: General

Before we get into the specifications for each file, let's start with a few general notes:

- All data files should be tab-delimited text files with a header using the column names outlined below. Avoid special characters and quotes (single or double). You can create them in a spreadsheet and save as tab-delimited or comma delimited files.

- There are six files that you will need to prepare before inserting them into tables in MySQL. In order to use the scripts that we provide (to import the tables), you must name the files as they are shown in parentheses below. Conversely, you can edit our scripts to reflect your file names.
    a. tree measurement codes (codes.txt)
    b. role reference (role.txt)
    c. personnel (personnel.txt)
    d. species (species.txt)
    e. quadrat (quadrat.txt)
    f. tree and stem measurements for each census, *i.e.* 1 file per census. This file is only necessary if you already entered your census data elsewhere (census.txt).

- We suggest you submit the above files for review to the ForestGEO Systems Group prior to upload, either by email or Dropbox.

Formatting Files: Specific

In this section we will reiterate the name of each text file (in parenthesis), introduce the scope of the table (with the indented description), specify the precise language of each column heading (underlined), and describe the content that is affiliated with said heading (following the colon).

Tree Measurement Codes (codes.txt)

These are the codes used by field personnel to describe the condition of a tree, stem or measurement. These codes are locally derived and can be in any language. These tree measurement codes will eventually be inserted into the TSMAttributes table, which is a permanent table.

-**code**: one or more letters that describe or explain the condition of a tree, stem, or measurement (*e.g.* "L")

- **description**: a free text description of the code (***e.g.*** *"*leaning")

- **status**: one of six standardized terms used as a summary category for the code and the condition of the stem which it describes:
    a. **alive**: the stem is alive
    b. **alive-not measured**: the stem is alive but was not measured
    c. **dead**: the ENTIRE TREE is dead
    d. **missing**: field crews missed this stem, and it was not measured during the census
    e. **broken below**: the stem was previously ≥ 1 cm dbh, but in this census was found alive but broken off, now with a dbh <1 cm
    f. **stem dead**: the stem is dead and/or not found

e.g. We may call a tree in the field "MS;R" – MS (multiple stems) could have an "alive" status on this table and R (description: resprout) would have "broken below."

NOTE: If a code can be used for more than one status (*e.g.* The code "L" for a leaning tree, could apply to either a dead or alive stem), or if a code does not indicate any of the above status options, the status column should be left blank.

Role Reference (role.txt)

This file should have a list of the job titles involved in the collection and management of ForestGEO plot data. It contains only a single column. The roles are site-specific, except for the ones listed in bold, which are required if using ctfsweb for data entry: **field technician**, **data entry technician**, **field supervisor**, student, volunteer, data manager, principal investigator, among others.

- **role**: the role that a person played in a census

Personnel (personnel.txt)

This file contains the names of the people who are or were involved with the plot, as well as the role that they played. If a person has played more than one role (for example she was a field technician in one census, then promoted to field supervisor in a later census), then that name should be entered twice. This file should have three columns, as designated below.

- **firstname**: the first (given) name of the person
- **lastname**: the last name (surname) of the person
- **role**: the role the person played in the census. This should match exactly one of the descriptions in the role.txt file.

Species (species.txt)

This file is integral to a key table in the system, so take time to review it for spelling errors, etc. Make sure the IDLevels are filled in. There should be at least one species code for unidentified species if your plot includes species not yet identified. There are four required columns ("spcode," "genus," "species," and "IDLevel"); the rest are optional.

- **spcode**: a code used in the field to identify the species of the tree
  - Most ForestGEO sites use six letter codes where the first four are from the genus name and the last two are from the species. If two species yield the same code, then an alternative letter or number as the last character may be used to differentiate them. For example, codes for *Shorea macroptera* subsp. baillonii and Shorea macrophylla, would both be SHORMA. The species codes ended up being SHORMB and SHORMC, respectively.
  - You should use a similar naming convention for each morphospecies incorporating details you know. For example, using LITSBL (Litsea "Big Leaf") or APORS1 (Aporosa sp. 1) are fine as long as each code applies to only one morphospecies. These can be changed once identification is more complete.
  - Other combinations are also acceptable. Some sites use 3 letters from the genus and 3 from the species, while others use 4 letters instead of 6 (2 letters from the genus and 2 from the species).
- **genus**: the taxonomic genus name according to the APG system. In case of an unknown genus, use "Unidentified."
- **species**: the species part of the Latin name; may be a morphospecies name.
- **IDLevel**: the deepest taxonomic level for which full identification is known. The IDLevel is limited to the values of: species, subspecies, genus, family, none, or multiple. "None" is used when the family is not known. "Multiple" is used when the name may include a mixture of more than one species.
- **family**: the taxonomic family name (optional)
- **authority**: author of the species (optional)

Quadrat (quadrat.txt)

This file contains a complete list of all quadrats used in your plot.

- **quadrat**: the name of the quadrat, *e.g.*0002
- **startx**: the *x* coordinate of the lower left corner of the quadrat, *e.g.* 0
- **starty**: the *y* coordinate of the lower left corner of the quadrat, *e.g.* 40

  NOTE: The *x* and *y* coordinates ("startx" and "starty") refer to the distance in meters between the quadrat under question and lowest, left-most corner of the entire plot (or wherever your plot origin, or 0,0 coordinates are).
- **dimx**: the *x* dimension of the quadrat (in meters), *e.g.* 20
- **dimy**: the *y* dimension of the quadrat (in meters), *e.g.* 20

Census (census.txt)

The tree data from each census must be in a separate file (*i.e.* one census, one file; three censuses, three files). All files must have columns listed below, but they can be in any order.

NOTE: Each of the multiple stems should be included in these files. You may indicate in the codes field which one is the *main* stem (if the tree has only one stem, you do not have to include the main stem code). The rest of the information should be repeated for each multiple stem. Make sure that the information (species code, date, etc.) is exactly the same for all multiple stems of the same tree.

NOTE: The dataset for each census should only contain trees and stems that were tagged and measured from that census. The dataset for subsequent censuses should contain all live stems from the previous census. Dead or lost stems should have the appropriate codes to indicate their absence in subsequent censuses.

- **tag**: the tag number for the tree (this should be unique for each tree)
- **stemtag**: the tag number of the stem.
- **spcode**: a code used in the field to identify the species of the tree. This MUST match the spcode that appears in the species.txt file.
- **quadrat**: the name of the quadrat (as designated in quadrat.txt) that the tree is located in
- **lx**: the *x* coordinate in meters of the stem within its quadrat
- **ly**: the *y* coordinate in meters of the stem within its quadrat
- **dbh**: the diameter of the tree.

  NOTE: If there is no diameter measurement because the tree is missing, dead, or a resprout, please put "NULL"
- **codes**: tree or measurement codes (as designated in codes.txt)

  NOTE: If there is more than one code, they should be delimited with semicolons. This allows for codes with more than one letter. The codes field may be left blank if there are no codes.
- **hom**: height (in meters) where the diameter was measured, if different from 1.3 meters.

  NOTE: If the height of measurement was 1.3 meters, you may leave this field blank.
- **date**: the date the stem was measured, expressed in the format YYYY-MM-DD, *e.g.* 2011-02-24 to indicate the 24th of February, 2011.

Loading Fixed Data

Before you insert census data, you will first populate several tables with fixed data in the database. This step is required for all users who are creating an initial dataset (it doesn't matter whether or not you use the ctfsweb data entry module). You can skip this step in subsequent recensuses if the fixed data has not changed.

NOTE: In the commands that follow, replace **'quoted, blue text'** with information corresponding to your plot. Quotes are needed for columns that are character fields, but not for numeric fields.

HOW TO: Load Fixed Data into Your Database

1. Open MySQL/MariaDB console window and log in. Open your database.

   MySQL> USE current_bci;

2. Write out the commands below (steps a-h) in a text editor, replacing the blue text with site specific info. Then copy and paste one command at a time into the MySQL console window.

   a. Enter the COUNTRY information:

      MySQL> INSERT INTO Country (CountryID, CountryName) VALUES (1, 'Country');

   b. Enter the SITE information with the following script, substituting 'placeholders' to reflect your site in the parentheses following "VALUES":

      MySQL> INSERT INTO Site (PlotID, PlotName, LocationName, CountryID, ShapeOfSite, DescriptionOfSite, Area, QDimX, QDimY, GUOM, GZUOM, PUOM, QUOM, GCoorCollected, PCoorCollected, QCoorCollected, IsStandardSize) VALUES (1, 'yoursitename', 'Your Site Location eg. South Western Province', 1, '1000x500', '*Any site description you want to insert here*', 500000, 20, 20,'','', 'm', 'm', 'N', 'Y', 'Y', 'Y');

      The above terms correlate with the following information:

      - **PlotID**: an integer automatically generated to uniquely identify a plot
      - **PlotName**: the plot name, preferably expressed without spaces. Capital letters are fine.
      - **LocationName**: regional descriptor (*e.g.* South Western Province)
      - **CountryID**: this is a primary key, and it is automatically generated
      - **ShapeOfSite**: length of each side for a rectangular plot OR identify the shape of an irregular plot
      - **DescriptionOfSite**: a free text description of the site, typically identifying the forest type
      - **Area**: area of plot; square meters is permissible, hectares is preferred (i.e. 50)
      - **QDimX** and **QdimY**: dimensions of each quadrat (in meters) along the X and Y axis, respectively
      - **GUOM**: unit of measure for global coordinates, *i.e.* latitude and longitude (e.g. degrees, decimal degrees, meters, etc.)
      - **GZOUM**: unit of measure for global elevation coordinates
      - **PUOM**: unit of measure for plot coordinates, *i.e.* plot dimensions (*e.g* meters)
      - **QUOM**: unit of measure for quadrat coordinates (*e.g*. meters)
      - **GcoorCollected**, **PcoorCollected**, and **QcoorCollected**: designation of whether the global, plot, and quadrat coordinates were collected, respectively; indicate with "Y" for yes, and "N" for no.
      - **IsStandardSize**: designation of whether or not the plot is a standard shape; indicate "Y" for rectangle, "N" for all other shapes

   c. Enter the CENSUS information (one census record for each census you plan to upload) substituting your site's information where we have 'placeholders':

MySQL> INSERT INTO Census (CensusID, PlotID, PlotCensusNumber, StartDate, EndDate, Description) VALUES (1, 1, 1, '2010-01-02', '2010-10-31', 'Any census description you want to insert here');

The above terms should convey the following information:

- **CensusID**: an integer automatically generated to uniquely identify this census
- **PlotID**: Foreign Key to Site table indicating the plot to which the census data belongs
- **PlotCensusNumber**: census number expressed as an integer (*e.g.* 1=1st census)
- **StartDate**: first day of data collection, expressed in YYYY-MM-DD
- **EndDate**: last day of data collection, expressed in YYYY-MM-DD
- **Description**: (optional) free text description or any relevant observations of census

d. Insert the MEASUREMENT CODES (from codes.txt) using the following command. Replace *path-to-file* with your path to the file, but otherwise keep the command exactly as it is. It will directly load your measurement codes file into the codes table (TSMAttributes):

MySQL> LOAD DATA LOCAL INFILE '/*path-to-file*/codes.txt' INTO TABLE TSMAttributes fields TERMINATED BY '\t' ENCLOSED BY '' IGNORE 1 LINES (TSMCode, Description, Status);

NOTE: For an explanation of the LOAD command, see Appendix A

e. Insert the ROLE REFERENCES (from role.txt) using the following command, replacing *path-to-file* with your path to the file:

LOAD DATA LOCAL INFILE '/*path-to-file*/role.txt' INTO TABLE RoleReference fields TERMINATED BY '\t' ENCLOSED BY '' IGNORE 1 LINES (Description);

f. Insert the Personnel data as follows, replacing *path-to-file* with your path to the file:

```
MySQL> DROP TABLE IF EXISTS stagePersonnel;
MySQL> CREATE TABLE stagePersonnel(
    PersonnelRoleID    INT UNSIGNED AUTO_INCREMENT,
    FirstName          CHAR(30),
    LastName           CHAR(32),
    Role               CHAR(40),
    RoleID             INT,
    PersonnelID        INT,
PRIMARY KEY (PersonnelRoleID)
);`
```

MySQL> LOAD DATA LOCAL INFILE '/*path-to-file*/personnel.txt' INTO TABLE stagePersonnel fields TERMINATED BY '\t' ENCLOSED BY '' IGNORE 1 LINES (FirstName, LastName, Role);

MySQL> UPDATE stagePersonnel A, RoleReference B SET A.RoleID=B.RoleID WHERE A.Role=B.Description;

MySQL> INSERT into Personnel (firstname, lastname) SELECT DISTINCT firstname, lastname FROM stagePersonnel;

MySQL> UPDATE stagePersonnel A, Personnel B SET A.PersonnelID=B.PersonnelID WHERE A.FirstName=B.FirstName and A.LastName=B.LastName;

MySQL> INSERT INTO PersonnelRole (PersonnelID,RoleID) SELECT PersonnelID, RoleID FROM stagePersonnel;

g. Insert the SPECIES data (species.txt) using the following command, replacing *path-to-file* with your path to the file:

MySQL> LOAD DATA LOCAL INFILE '/*path-to-file*/species.txt' INTO TABLE TempSpecies FIELDS TERMINATED BY '\t' ENCLOSED BY '' IGNORE 1 LINES (Mnemonic, Genus, SpeciesName, FieldFamily, Authority, IDLevel);

h. Insert the QUADRAT information (quadrat.txt) using the following command, replacing *path-to-file* with your path to the file. Before you do, note that:

- the below command is written for CensusID=1; if you are loading a recensus, change the CensusID accordingly.
- the below command assumes that startx and starty refer to the lower lefthand corner of the quadrat, in other words, to where QX=0 and QY=0

MySQL> LOAD DATA LOCAL INFILE '/*path-to-file*/quadrat.txt' INTO TABLE TempQuadrat FIELDS TERMINATED BY '\t' IGNORE 1 LINES (QuadratName, StartX, StartY, DimX, DimY);

MySQL> UPDATE TempQuadrat SET PlotID=1, CensusID=1;

MySQL> INSERT INTO Quadrat (PlotID, QuadratName, Area, IsStandardShape, QuadratID) SELECT PlotID, QuadratName, DimX*DimY, 'Y', QuadratID FROM TempQuadrat;

MySQL> INSERT INTO CensusQuadrat (CensusID,QuadratID) SELECT CensusID, QuadratID FROM TempQuadrat;

MySQL> INSERT INTO Coordinates (PlotID, QuadratID, PX, PY, QX, QY) SELECT PlotID, QuadratID, StartX, StartY, 0, 0 FROM TempQuadrat;

Woohoo! You've just added a bunch of tables with critical information for your plot's database. You're getting close to entering census data into your plot's database.

If you ARE going to use ctfsweb to enter your census data, please continue to Chapter 3.
If you are NOT going to use ctfsweb to enter your census data, please continue to Chapter 4.

CHAPTER 3: Adding Census Data: Working from Paper Records

Configuration Files for Entering Data via ctfsweb

Configuration files set the parameters and settings of a system or app. In the case of ctfsweb, we have to configure the field forms according to each plot, since not all field forms are exactly the same. Some plots use cm instead of mm for their dbhs, others have extra columns, etc.

HOW TO: Set-Up Configuration Files

1. Open the ctfsweb system using your username and password and call up your plot's database.

2. Click on "Data Entry Configuration" in the top menu and select your plot. In the case of a first census, you will only be working with the "New Plants Form." In the case of a re-census, you will work with the "Old Trees Form" (for stems previously censused), the "Multiple Stem Form" (for new secondary stems of trees previously censused), and the "New Plants Form" (for all stems on new recruits).

3. Set up a configuration file with columns that are in the same order as your field sheet. You can start from scratch and add new columns, or alternatively, you can use the default configuration and change it to reflect your field sheet. A few notes on column names:
   - The names that you enter in the "Column Name" parameter have to be written exactly the same as the names in "Column Name" in the "List of Column Names" (click on this list in the left panel to reference it).
   - Names are case sensitive.

4. Choose your column types. Certain types will fit better for certain features. The options for column types are:
   - *simple textbox with text, integer, or decimal values without limits*: This allows you to set up a column where you can enter any type of data but where no screening is done (e.g. descriptions, comments).
   - *simple textbox with numeric values with limits*: This allows you to set up a column to enter numeric values, specifying their minimum and maximum values, preventing the user from entering values outside this range (e.g. dbh)
   - *species dropdown*: This creates a column where the user can only choose valid codes from the Species table.
   - *subquadrat dropdown*: This allows the user to include the subquadrat where the current tree is located as a field. In most sites this refers to the 5x5 meter subquadrat within the 20x20 meter quadrat. The "Starts with" option allows you to enter "0" or "1," referring to which digit the site uses to start naming the subquadrats, *e.g.* 00, 01, 02, 03…33 OR 11, 12, 13, 14,…44.
   - *user dropdown*: This allows you to specify options for the user to select. You will need to separate the options with semicolons (;). Note that there should NOT be a space following the semicolon.
     - *e.g.* yes;no
     - *e.g.* L;B;M;D;B
   - *simple textbox with a prefix value*: This allows you to specify a field where the first few characters are set, while the rest changes from one record to the next.
     - *e.g.* For a site that uses tag numbers where the first four characters refer to the quadrat number: 1212-0001, 1212-0002, 1212-0003, and so on.
   - *tree tag increment with stem tag*: This option allows the user to enter incremental numbers for the stem, whereas the tree gets the tag that is the smallest stem tag of that tree. This option is for plots whose stem tags increase throughout the plot, instead of starting at one again for each new tree.

5. Configure the following three parameters for each of your columns, except for those with dropdown menus (Configuring these parameters will allow people doing data entry to enter data faster by not having to key in the tag numbers for each tree and stem, instead allowing the tag numbers to increment automatically from one record to the next):
   i. *Is Value Changed with Stem*: choose "incremented" OR "carried over"
      incremented: the current field is incremented by one in the next stem record
      carried over: the current field is repeated in the next stem record

   ii. *Is Value Changed with Tree*: choose "incremented" OR "carried over"
      incremented: the current field is incremented by one in the next tree record
      carried over: the current field is repeated in the next tree record

   iii. *Add Next Row Stroke*: choose "YES" OR "NO"
      YES: the cursor will go to the next record when you press [ENTER] OR [+] in this field.
         NOTE: You should not choose "YES" in the first fields of the form until all of the necessary fields have data entered. This is a safeguard to prevent blank variables from being erroneously recorded. We suggest that you choose the "YES" option only after the DBH variable.
      NO: the cursor will skip to the next field of the same record when you press [ENTER]

Configuring the parameters of your columns correctly is important. Let's look at a few examples:

A. You want your tree tag number to increase by one from one tree to the next, and your stem tags to start from one for each new tree:

For the tree tag:
- *Is Value Changed with Stem* = carried over
- *Is Value Changed with Tree* = incremented
- *Null Value Allowed* = no
- *Add Next Row Stroke* = no

For the stem tag:
- *Is Value Changed with Stem* = incremented
- *Is Value Changed with Tree* = no
- *Null Value Allowed* = no
- *Add Next Row Stroke* = no

B. If the stem tags are not repeated throughout the plot, in other words, all stems are tagged and the tags are unique across the plot, as below:

Tag	123412	123413	123414	123414	123414	123417	123417	123421	123422
Stemtag	123412	123413	123414	123415	123416	123417	123418	123421	123422

then choose the following in your configuration file:

For the tree tag, select "Tree Tag Increment with Stem Tag" and choose:
- *Is Value Changed with Stem* = carried over
- *Null Value Allowed* = no
- *Add Next Row Stroke* = no

For the stem tag, select "Simple text box with text, integer, or decimal values without limits" and choose:
- *Is Value Changed with Stem* = incremented

- *Is Value Changed with Tree* = incremented
- *Null Value Allowed* = no
- *Add Next Row Stroke* = no

HOW TO: Set Up Configuration Files (cont'd)

6. When you are finished configuring your data entry sheet, preview the form and try it out to make sure that you have set it up correctly, especially with respect to the tags and stem tag columns.

   If you need to make changes, you can delete any of the variables, add a new column, and then redo the configuration. This will insert your column in the bottom of the fields. To move this column to its correct position in the form, click on "Rearrange Columns" and drag the field to its correct place.

7. Once you are satisfied with the data entry form, make sure to save your changes by clicking on "Create Config File."

Entering Census Data from Paper Records into ctfsweb

Once you have created the configuration files, you are ready to use the data entry form to enter data from your field sheets. We strongly suggest that census data be entered twice by two different data entry technicians, if possible. Double data entry catches most of the errors caused by data entry. If you are unable to double-enter data, proceed to our instructions for single-entry data.

HOW TO: Enter Census Data from Paper Records into ctfsweb

1. Go to "Data Entry" in the top menu of ctfsweb and choose the data entry form.

2. Read the instructions on the first screen before you go on to the data entry screen. Some keys have been programmed to make data entry easier and faster:

   - The asterisk [*] key allows the user to go from one field to the next.
   - The slash [/] key allows the user to go to a previous field.
   - The plus [+] key from the numeric keypad allows your cursor to skip to the next stem record, carrying over specified data into this new record, in any field where you specified "yes" in the "Add Next Row Stoke" parameter.
   - The [Enter] key allows your cursor to skip to a new tree record, carrying over specified data into this new record, in any field where you specified "yes" in the "Add Next Row Stoke" parameter.

3. When you've finished reading the instructions, type in the name of the data entry person. Their name will get saved in the name of the finished text file. This allows two people to use the same laptop to enter data on the same quadrat.

4. Upon entering the data entry screen, select/type in the quadrat you are entering data from. Remember to use the programmed keys (step 2 of this box) to transfer data from your paper records into ctfsweb.

5. After you have finished entering a quadrat, press the "Submit" button. A warning will come up asking you whether you want to save the file. This prevents you from accidentally pressing "Submit" and taking you out of the form before you have entered all of the data.

The warning also asks that you check that the quadrat is correct. A common error in data entry is forgetting to change the quadrat number when starting a new quadrat, resulting in the wrong quadrat being entered for all the records of that quadrat.

6. Once saved, we strongly suggest that you click on the name of the resulting text file and quickly check that all the records have been saved (Some computers have run into memory problems when there are many stems in one quadrat, and not all records are saved. For a solution to this problem, see Appendix D).

The quadrat data files that the data entry technicians just typed in should be placed in their respective folders under "…/ctfsweb/files/datafiles." Notice that the names include the name of the form, the name of the plot, the census number, the quadrat name, the date of data entry, and the name of the person entering the data.

Notice that there is an option for you to add more data into a quadrat that you had already started entering previously. This allows you to enter data from trees that were missed in a first pass through the quadrat by the field technician. It also allows you to divide a quadrat with a lot of trees into two or more entry sessions. Be careful to pick the correct file to add data to. Data from later entry sessions will be appended to the end of the file.

Cross-Check Double-Entered Data with ctfsweb

Cross-checking is a safeguard to determine if the datasets that two different individuals entered into ctfsweb match each other. It should be done regularly throughout a census, not left until the end. One reason for this is that if there are any doubts or questions, the field technician can still remember what happened if the data was collected recently, or if necessary, the field technician can go back to the field to check an entry.

Once the census data from a quadrat is entered twice, cross-check both files, look at the errors, and pick the correct record to save.

NOTE: If the data enterers each worked from their own laptop then the text files have to be transferred over to one laptop to do the cross-checking.

HOW TO: Cross-Check Double-Entered Data

1. Click on "Data Screening" in the left panel of ctfsweb.
2. Click on "Double Data Entry."
3. Click on "Copy Text Files into Temporary Tables."
4. Select the two text files that you want to compare. This inserts the two files into MySQL temporary tables. Repeat this as many times as the number of quadrats that you have entered.
   NOTE: Make sure you don't pick the same two files – this will lead to no errors found when comparing them.
5. Start comparing the data in the temporary tables by choosing the quadrat that you want to compare. All records that are not exactly the same will show up in order of tag.
   - Choose the correct record to KEEP by clicking on the circle to the left (the default is the first record).
   - You can MAKE CHANGES to any of the fields of the record that you want to keep.

- You can also DELETE the record if you don't want to keep any of the two by pressing on "Delete Record" in the upper right-hand corner.
- You can also SKIP this record and go back to it later if you are not sure which is correct and need to check it later.

NOTE: Be careful when selecting the records to keep, mindful of the scenario where one of the files doesn't have a corresponding record in the other file. Because comparisons are made based on the tree tag, if the tree tag is entered incorrectly in one of the files, it will appear in one of the files in one instance while the other file has a missing record and vice versa in a later instance. This is not a discrepancy in the data, simply an error in the tag number.

As data entry progresses, there will be many temporary tables to scroll through when selecting the quadrats to compare. You can delete those quadrats that you have already cross-checked, and which are empty by clicking on "Delete all Screened Temporary Tables with no data".

There may be times when you want to compare two text files that you had already cross-checked before. Once the text files are selected, they are moved from their folders to their respective upload folder in …/ctfsweb/files/datafiles/uploaded/. You can move them back to their original folder and they will again appear in the dropdown list for you to select. You may have to delete the records of this quadrat from the corresponding Temp table also if they had already been uploaded, i.e. TempNewPlants, TempOldTrees, or TempMultiStems. Be careful when deleting – you will not be able to recover the records once deleted. Conversely, if you do re-crosscheck, ensure that you do not duplicate the records in the Temp table.

Single Data Entry
As mentioned above, we highly recommend that you use double-data entry when entering paper records into ctfsweb. In cases where that is not possible, follow the instructions below.

HOW TO: Transfer Data from Text Files into the Temporary Tables from Single-Entry

1. Press the "Single Data Entry" option under "Data Screening" to insert the text files into the corresponding Temp tables: "TempOldTrees," "TempNewPlants," and "TempMultiStems."

Inserting Location Data
If you use field maps to map the locations of the stems in the plot, you might want to scan them and digitize the stem locations. If you don't already have a digitizing software, we suggest that you use the open source image processing package ImageJ and the plugin PointPicker to digitize the maps (more details and instructions in Appendix E). The resulting coordinates are in pixels, and you can convert them to meters using the R function "fullplot.imageJ()" from the CTFSRPackage.

HOW TO: Insert Location Data into TempLocations Table

1. Digitize maps (see Appendix E).

2. Convert pixels to meters using the R function "fullplot.imageJ()" from the CTFSRPackage.

3. Upload the resulting coordinates into the TempLocations table:

```
MySQL> LOAD DATA LOCAL INFILE '/path-to-file-with-coordinates/plotcoordinates.txt' INTO TABLE TempLocations FIELDS TERMINATED BY '\t' ENCLOSED BY '' IGNORE 1 LINES (Tag,x,y,QuadratName);
```

You've successfully entered your census data, conducted a first-round comparison between your double-entered data sets (if applicable), and inserted location data into a TempLocations table. Proceed to Chapter 5.

CHAPTER 4: Adding Census Data: Working from Electronic Records*

**Electronic records" include not only field data collected into tablets, but also paper data that was entered into electronic files via some other program, such as Excel, Filemaker, PosGreSQL, etc.*

Formatting Your Electronic Census Records for Temporary Tables

If you are NOT going to use ctfsweb to enter your data because you already have it in some other electronic format, you still need to upload your data into temporary tables so that ctfsweb can upload them into production tables. We recommend, however, that you first upload your data into staging tables because the data will likely not be in the right format for uploading directly into temporary tables. Create staging tables that exactly reflect the structure of your text files, their columns and the type of data in them.

NOTE: Below is only a guide; you can change the structure of the staging tables to conform to the structure of your text files, but NOTE that there are four required variables: PlotID, PlotCensusNumber, CensusID, and tempID.

HOW TO: Format Your Electronic Records in Staging Tables

1. Open MySQL and run the following script:

```
MySQL> DROP TABLE IF EXISTS stageCensus;
MySQL> CREATE TABLE stageCensus(
        Tag                 CHAR(10),
        StemTag             CHAR(32),
        Mnemonic            CHAR(10),
        QuadratName         CHAR(12),
        x                   FLOAT(8) DEFAULT NULL,
        y                   FLOAT(8) DEFAULT NULL,
        DBH                 FLOAT(8) DEFAULT NULL,
        Codes               VARCHAR(50) DEFAULT NULL,
        HOM                 FLOAT(8) DEFAULT NULL,
        ExactDate           DATE,
        PlotID              INT UNSIGNED,
        PlotCensusNumber    INT UNSIGNED,
        CensusID            INT UNSIGNED,
        tempID              INT UNSIGNED AUTO_INCREMENT,
    PRIMARY KEY (tempID)
);
```

2. Now copy your data from the text files into the staging table. Run the following script, replacing *path-to-file* with the path to your tab-delimited data files:

```
MySQL> LOAD DATA LOCAL INFILE '/path-to-file/census.txt' INTO TABLE
stageCensus fields TERMINATED BY '\t' ENCLOSED BY '' IGNORE 1 LINES (Tag,
StemTag, Mnemonic, QuadratName, x, y, DBH, codes, HOM, Exactdate);
```

3. Run the following script, replacing the 1s that follow PlotID, PlotCensusNumber, and CensusID with the numbers that correspond with your plot and census (*e.g.* If this is the first census, then use "1," if it's the first re-census, then use "2").

```
MySQL> UPDATE stageCensus SET PlotID=1, PlotCensusNumber=1, CensusID=1;
```

4. Add an index to the table with the following script:

MySQL> ALTER TABLE stageCensus ADD INDEX(Tag,PlotID);

If your site uses stemtags, create the following index instead:

MySQL> ALTER TABLE stageCensus ADD INDEX(Tag, StemTag,PlotID);

Creating Temporary Tables

Create the temporary tables used by the ctfsweb system, if you have not already done so. Ctfsweb will screen the data in these temporary tables for errors before you upload it into production tables.

HOW TO: Create Temporary Tables

1. Run the "createTempTables.sql" script:

mysql> SOURCE createTempTables.sql;

Copying your data into Temporary Tables

Once you've created temporary tables, you will copy over the data from the staging table into the temporary tables.

HOW TO: Copy Your Data from the Staging Table into the Temporary Tables for a First Census

1. Use the following script, inserting additional columns into TempNewPlants and TempLocations when they are relevant to your site:

MySQL > INSERT INTO TempNewPlants (QuadratName, Tag, StemTag, Mnemonic, DBH, codes, HOM, ExactDate, x, y, PlotCensusNumber, PlotID, CensusID) SELECT QuadratName, Tag, StemTag, Mnemonic, DBH, codes, HOM, ExactDate, x, y, PlotCensusNumber, PlotID, CensusID FROM stageCensus;

MySQL > INSERT INTO TempLocations (QuadratName, Tag, StemTag, X, Y, PlotCensusNumber, PlotID, CensusID) SELECT DISTINCT QuadratName, Tag, StemTag, x, y, PlotCensusNumber, PlotID, CensusID FROM stageCensus;

Uploading Re-census Data from Electronic Records

You will need the MySQL database with the data from your previous censuses if you are adding new census data. To get the most current version, contact the ForestGEO database managers (Suzanne and Shameema) for the "stable_plotname.sql file." You can follow the teaching example "StepsCreateStagingTables_census2.sql" as a guide to create the staging tables. Be sure to edit the file to reflect the name of your database and the path to your new census data.

HOW TO: Copy Your Data from the Staging Table into the Temporary Tables for a Re-census

1. Add the new census data using the script below (modifying the 'plotid=1' if relevant).

```
INSERT INTO TempOldTrees
(QuadratName,Tag,StemTag,Mnemonic,DBH,codes,HOM,ExactDate,x,y,PlotID,Pl
otCensusNumber,CensusID)
  SELECT
QuadratName,Tag,StemTag,Mnemonic,DBH,codes,HOM,ExactDate,x,y,PlotID,Pl
otCensusNumber,CensusID
    FROM stageCensus WHERE tag IN (SELECT DISTINCT tag FROM
ViewFullTable WHERE plotid=1);

INSERT INTO TempNewPlants
(QuadratName,Tag,StemTag,Mnemonic,DBH,codes,HOM,ExactDate,x,y,PlotID,Pl
otCensusNumber,CensusID)
  SELECT
QuadratName,Tag,StemTag,Mnemonic,DBH,codes,HOM,ExactDate,x,y,PlotID,Pl
otCensusNumber,CensusID
    FROM stageCensus WHERE TAG NOT IN (SELECT DISTINCT tag FROM
ViewFullTable WHERE plotid=1);

INSERT INTO TempLocations (QuadratName,Tag, StemTag, X,
Y,PlotCensusNumber,PlotID,CensusID)
  SELECT DISTINCT QuadratName,Tag, StemTag, x, y, PlotCensusNumber,
PlotID, CensusID
    FROM stageCensus WHERE Tag NOT IN (SELECT DISTINCT tag FROM
ViewFullTable WHERE plotid=1);
```

If your site DOES TAG MULTIPLE STEMS, you have to do the following steps also:

```
INSERT INTO TempMultiStems
(QuadratName,Tag,StemTag,Mnemonic,DBH,codes,HOM,ExactDate,x,y,PlotID,Pl
otCensusNumber,CensusID)
  SELECT
QuadratName,Tag,StemTag,Mnemonic,DBH,codes,HOM,ExactDate,x,y,PlotID,Pl
otCensusNumber,CensusID
    FROM TempOldTrees WHERE (Tag,StemTag) NOT IN (SELECT
Tag,StemTag FROM ViewFullTable);

DELETE FROM TempOldTrees WHERE (Tag,StemTag) IN (SELECT Tag,StemTag
FROM TempMultiStems);
```

6. If there are new measurement codes in this census that are not already in the TSMAttributes table, they can be inserted as follows:

```
INSERT INTO TSMAttributes (TSMCode, Description, Status) VALUES
('newcode','a brief explanation of code','one of the statuses mentioned above
under Formatting Files: Specific');
```

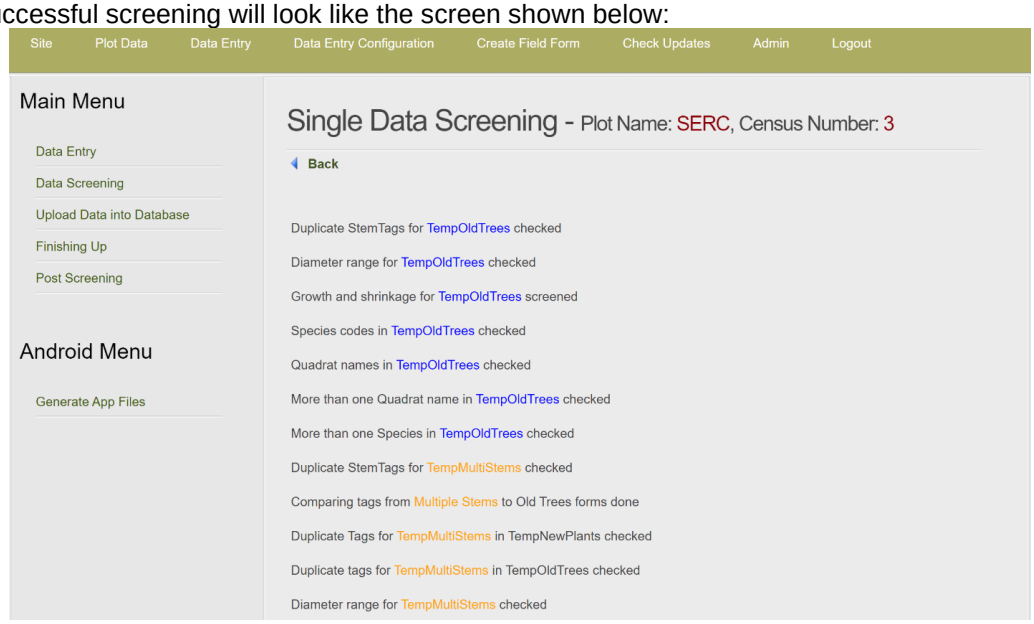
CHAPTER 5: Screening, Correcting, and Summarizing Content

Screening Data in Temporary Tables

Now that your data is in a temporary table, it's time to screen that data to see if it makes logical sense (*e.g.*, Does each tree have a valid species code? Do all of the stems on the same tree have the same species code? Is there more than one tree with the same tag number? Are the coordinates within the grid?).

HOW TO: Screen Data in Temporary Tables

1. Choose the "Data Entry" tab from the Main Menu of ctfsweb.
2. For a census (or re-census) that has not yet been screened, select the "Create New Census" option and provide the census number. If you have previously done this step and are returning to re-screen, simply select the plot and census number from the drop-down menus.
3. In the panel on the left side of the screen, select "Data Screening."
4. Select "Screen Data in Temporary Tables."
5. Enter the minimum and maximum DBHs allowed and submit. The screening may take a few minutes.
6. A successful screening will look like the screen shown below:



7. Return to the "Data Screening" page and select "View Error Summary." This will allow you to view a list of errors for each table (TempNewPlants, TempOldTrees, TempMultistems, and TempLocations).

Correcting Content in Temporary Tables

If there are *very few* corrections, it is okay to incorporate them into a script without listing them in a spreadsheet or text file. Otherwise, we highly suggest that you correct data errors that you identified with the View Error Summary tables by listing them in a spreadsheet or text file. A sample correction script can be found here. This is an important standard for making corrections for several reasons:

1. If you start with the most updated official database and run a script against it to load your data into the temporary tables and make corrections, and if you find that you have made a mistake, then you simply correct the script, and rerun the corrected script.
2. The same process can be duplicated on any server that holds your database to produce an exact duplicate of your current census. This allows another check of the steps in the script to be done by a second person, i.e. a database manager.
3. A permanent record of your corrections is available for inspection in case of later problems.

If there are many corrections to be made, you should keep track of your corrections in a spreadsheet (Excel) or tab or comma delimited text file (.txt or .csv). Each column in the spreadsheet should contain only one data item. The spreadsheet should have the following columns:

key: column or columns needed to completely identify the item to be changed, *e.g.* tag and stemtag to identify a stem to be changed

old value: value to be changed, *e.g.* old dbh, old species

new value: corrected value, *e.g.* corrected dbh, corrected species

If there are different kinds of changes, for instance, changes to dbh and changes to species, then report them in separate spreadsheets/.csv files.

In cases where the data was entered directly into the temporary files without using ctfsweb, database managers usually prefer to make the corrections directly in their text files and reupload them into the temporary files for rescreening. In these cases, there should be precise records kept of all changes made in these text files.

Save all scripts/records of your error corrections. Screening displays errors that may or may not be fatal, (i.e. prevents data to be uploaded into the production tables). See the table below for help distinguishing between fatal and non-fatal errors. Please note that this is not a comprehensive list. You should check all errors and try to correct them. After you have fixed as many of the errors as possible and made sure that the only errors displayed after screening are not fatal, then you can start uploading them from the temporary tables into production tables.

Fatal Errors	Non-Fatal Errors
• Duplicate tags or stemtags • Tree/stem tags in TempNewPlants (new recruits) that already exist in the database • Tree tags in the TempMultiStems table that are not found in the TempOldTrees table • Species codes that are not in the species table • Measurement codes that are not in the TSMAttributes table (beware of blanks before or after the code) *NOTE*: Measurement codes must be separated by semicolons. Other separators are not recognized as such and will generate code errors. • Quadrat names that are not in the Quadrat table	• Multiple stems with more than one species code (one of the species codes will be chosen) • Multiple stems with more than one quadrat name (this may or may not be an error, since some multiple stems of the same tree may have different locations. If you know that your plot does not have sprawling trees that require its stems to have different coordinates, then regard this as a fatal error). • X and Y coordinates that are greater than the limits of the quadrat (i.e. >=20m in quadrats that are 20x20m) • Dates that are not within the range of dates of the census • DBH measurements outside the dbh range

HOW TO: Load Data into Production Tables

1. Select "Upload Data into Database" from the main menu of ctfsweb and follow the prompts.

   - Be sure to upload the tables in order, *i.e.* the data from the old trees form before the new recruits, in the case that this is a re-census.

   - Uploading may take a while, especially in the case of plots with many trees.

Correcting Content in Production Tables
If you find errors after you have submitted a stable copy of the database to the ForestGEO Systems Group, collect them in batches and submit them (Excel sheets and scripts) every three months or so. You will implement the corrections by applying a script to a known version of the database. Each batch of corrections will result in a minor freeze which is a new version of the database.

We suggest that at the beginning of each calendar year a special effort be made to clean up all known database errors so that they can be corrected and uploaded by March for ForestGEO data staff to include in a major freeze, which is a version of the database that can be used by researchers.

HOW TO: Correct Content in Production Tables (You MUST use a script)

1. Login to the MySQL console window and create a blank test database in MySQL on your local computer:

   ```
   mysql> DROP DATABASE IF EXISTS test_sitename;
   mysql> CREATE DATABASE test_sitename;
   mysql> USE test sitename;
   ```

2. Install a known version (usually the last export) of your database. This is your starting point or base dataset. Every site should have an export file of their database in its current state (these can be downloaded from http://ctfs.si.edu/*sitename*/current/archive/ using the login given to the PI for their site). Download the most current database (by looking at the date in the file names) to your computer, then load it to your test database by sourcing it:

   ```
   mysql> SOURCE /path-to-file/sitename.sql;
   ```

3. Source your correction script against the imported database:

   ```
   mysql> SOURCE /path-to-file/correction_script.sql;
   ```

4. Check your results, debug the script if needed, and repeat from step 1.

5. When you are satisfied that the script is working correctly, run the correction script against the current database. You are now ready to proceed to the next step, creating the report tables.

* We have provided a set of sample scripts (teaching examples) for your use. Each script provides an example of how to correct one kind of error. Typically, a correction script will be composed of a series of these operations. Copy-and-paste them and modify a copy of whatever you need.

Summarizing: ViewFullTable & ViewTaxonomy
After all the data has been uploaded and corrections made to the production tables, it is time to create the report tables, ViewFullTable and ViewTaxonomy.

ViewFullTable makes querying the database easier and faster since the user does not have to join records from the various tables and instead can just query this table. The Plot Data Report queries ViewFullTable to get requests out. This is also the table that is used to make the R analytical tables.

These report tables are created by taking all the records from the relevant tables and merging them again into flat files. ViewFullTable will include a variable called "status" that defines whether the tree is dead or alive, and whether the stem is missing, dead, or broken below 1.3 m. This status variable is based on the status variable of the codes in the TSMAttributes table.

NOTE: Every time a change is made to the database, it is necessary to recreate ViewFullTable and ViewTaxonomy by running this step.

HOW TO: Create Report Tables (ViewFullTable & ViewTaxonomy)

1. Select "Finishing Up" in the left panel of the Main Menu.
2. Select "Create Report Tables." Create ViewTaxonomy first, then create ViewFullTable.

Post Screening

We suggest that you run a check on the data that was uploaded into the permanent tables and review the results to make sure there are no glaring problems. This may be done by running Post Screening. Some of the checks that are done in Post Screening are:

- Counting the number of records by quadrat
- Counting the number of dead trees and missing stems
- Checking whether there are any trees located outside the limits of the plot or any trees with missing coordinates
- Checking whether there are any duplicate tags
- Displaying the range of the dates of the censuses
- Displaying the largest dbh and height of measure for each species
- Checking for any trees that are both alive and dead in one census
- Checking if all trees from one census are accounted for in the next census
- Checking if any trees that were dead in one census were alive in the next census
- Checking if there are any trees with extreme growth rates

If post screening yields errors, correct them, keep a record of your scripts, and create updated report tables to reflect the corrected data.

Once the database has been reviewed and considered to be "stable" enough (until the next set of edits), back up the database by using mysqldump (see Appendix). The name of the resulting file should include the date the dump was made.

You can now erase all of the temporary and staging tables so that your database contains only the production tables. When you are finished, log out of ctfsweb.

Checksum.sql

The Checksum.sql script sums various numeric variables (i.e. DBHID, TreeID, StemID, etc.) to verify that you and the ForestGEO Systems Group both start off with the same database version before implementing changes. You should run it before you compile your package to share with the ForestGEO Systems Group.

HOW TO: Run Checksum.sql

1. Run Checksum.sql on your database (through the "SQL" tab on the database in phpMyAdmin) before you implement any changes. Save the results to a text file called "ResultsofChecksumbeforechanges.txt."

2. Implement your changes (via scripts) to your base data set.

3. Run Checksum.sql on your newly updated data set. Save the results to a text file called "ResultsofChecksumafterchanges.txt."

If the numbers that you send to the ForestGEO Systems Group correspond with the numbers that Suzanne/Shameema get from running your scripts against your base data, they run the scripts on the ForestGEO server or copy over the database from their laptops. If the numbers do not match, they'll have to find out why by looking over the scripts.

CHAPTER 6: Compiling, Testing, and Sending Your Data Package

To share a new version of your plot's database with the ForestGEO Systems Group you will compile and test a package. Uploading a new census and/or making changes to a previous version of the database both constitute a new version of the database.

Compiling Your Package Components

The package for a new version of the database should include the following files:

- base dataset prior to the application of scripts. This needs to be the dataset that is most recently lodged with ForestGEO.
- script or scripts (.sql or .php files)
- data (Excel or .csv/.txt files)
- README.txt containing:
  1. name of database manager, site, date, and a brief description of the corrections being made
  2. version number of the base dataset prior to the application of the script
  3. source command or commands to execute script(s)
  4. MySQL version
  5. ctfsweb version (if used)
- Checksum.sql script
- ResultsofChecksumbeforechanges.txt
- Results of Checksumafterchanges.txt

Testing Your Package

After you have assembled these components, test them for accuracy on your own computer by running the scripts with the appropriate text files in the specified folder (see below). This step will assure that everything runs smoothly.

You will also need to standardize the directory path of your files because programs that refer to files typically have a directory path that is only valid on your own computer. The package root directory should have the following structure:

yoursiteupgrade2020OCT (root folder for the upgrade -change to reflect your site and upgrade date)

databases – directory

1. initial database or scripts that make up a new database
2. final database

scripts – directory

correction scripts

data – directory

.csv or .txt data files referenced in the correction scripts

docs -directory

1. README.txt (file)
2. Checksum.sql (script)
3. ResultofChecksumbeforechanges.txt (file)
4. ResultofChecksumafterchanges.txt (file)

HOW TO: Standardize Directory Paths of Your Package

1. Go into the scripts directory and change the path of any referenced file so that it is relative to the root directory for the package.
   e.*g.* SOURCE scripts/correct_errors.sql;
   LOAD DATA LOCAL INFILE data/tag_corrections.csv …

2. Now start your MySQL console from the root folder and run your upgrade according to the instructions in your readme file. If this runs without error your package is ready to be compressed and sent.

Sending Your Package

Compress all of the package contents using 7-Zip or Tar before sharing them with Suzanne and/or Shameema. When you have done so, you may share the files via e-mail (LaoZ@si.edu, shameemaesufali@gmail.com) or Dropbox.

Appendix A

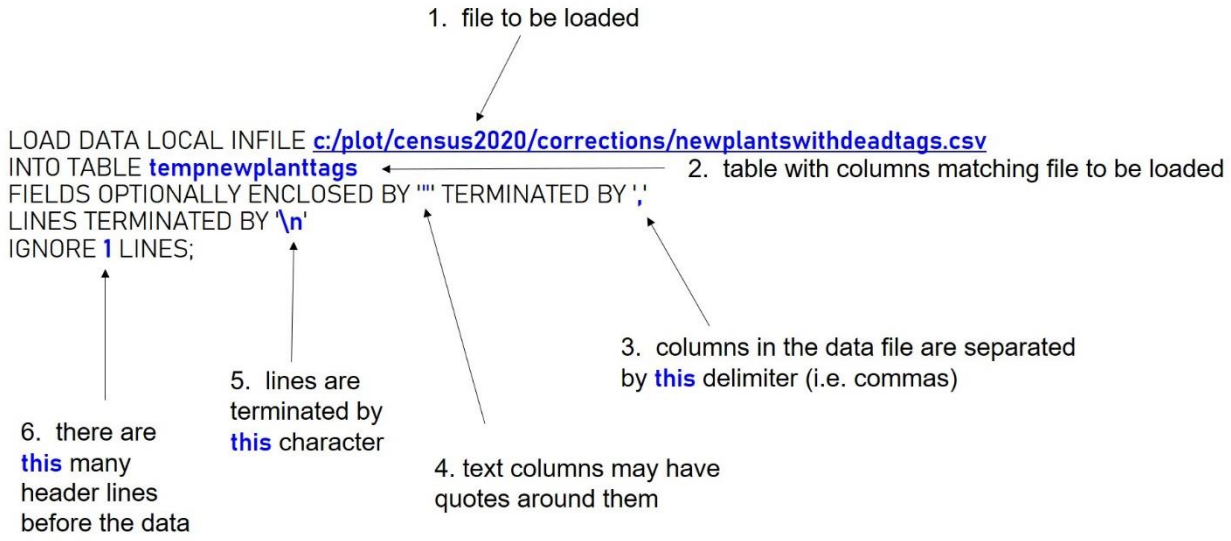
Deconstructing a Load Command

The LOAD DATA INFILE statement allows you to read data from a text file and import the file's data into a database table. Before importing the file, you need to prepare the following:

- a database table to which the data from the file will be imported.
- a CSV file with the same number of columns as the table and with the matching type of data in each column.

The following list identifies elements of a standard load command. Beneath it you will find more details about each of the components.

Below is an example of a standard load command, with the **text in blue** indicating site-specific customization:



1. The name of the delimited text file to be loaded. The file name must include either the full path on your local computer or the path relative to the folder from which the script is run. A script is a file containing pre-written code for manipulating the database.

   For example: if all of the files for the census are located in "c:/plot/census2020" with one folder containing scripts called "scripts" and another folder called "corrections" that contains the data files, then the relative path for "c:/plot/census2020/corrections/newplantswithdeadtags.csv" would be:

   '.../corrections/newplantswith deadtags.csv' when the script referencing the file is in c:/plot/census 2020/scripts/.

   The database table may have more columns than the text file; just make sure to allocate the columns in the same order as the columns in your text file:

   ```
   LOAD DATA LOCAL INFILE '/datafiles/census.txt' INTO TABLE stageCensus fields TERMINATED
   BY '\t' ENCLOSED BY '' IGNORE 1 LINES    (Tag,StemTag,Mnemonic,

   QuadratName,x,y,DBH,codes,HOM,Exactdate);
   ```

2. You need to create a table with at least as many columns as there are in the file to import (there can be extra columns in the table, such as CensusID, PlotID, PlotCensusNumber, etc.). The data types of the columns should match. So, if you only expect numeric values in a column then the field in the table will be of type INT, DECIMAL, or FLOAT. If it is a character field then use VARCHAR(30) where the number is an estimate of the maximum data length of the column to be imported. Starting the table name with "temp" is a convention that helps you to keep track of these tables. Delete the temp tables by using the DROP TABLE command when the database is finalized leaving only the standard set of tables in the database.

3. This indicates the separator used in the file. You may need to open the file in a text editor rather than in a spreadsheet (Excel) to see what separates the columns in your data file. A tab delimited file needs TERMINATED BY '\t' rather than the commonly used comma delimited TERMINATED BY ',' . You can convert any data file to comma delimited by opening them in Excel or Libre office and saving it as a .csv file. We recommend that you use the same delimiter for all data files to keep it simple.

   WARNING: If the delimiter (comma for example) is in the actual data then it will cause errors. You can either delete them, avoid using the delimiter in your data, or convert to tab-delimited text files.

4. This indicates that some text columns may have quotes around them which need to be stripped away before the data is stored. It is a standard inclusion in most Load Data commands.

5. Lines are typically terminated by '\r\n' on Windows but different operating systems and possibly even different text editors use different terminators. Unix uses '\n' and Mac uses '\r'.

6. IGNORE 1 LINES tells the load command that the first line is a header line and not a line of data. If you have multiple header lines or none, then alter the number accordingly. Note that the load command does not take its column names from your data file so the names do not need to match. It takes the columns in your data file and allocates them in order of the columns in your table. The data types should match.

For more information on load command examples, visit this link.

Appendix B

Backing-Up Your Data via mysqldump

Back up all your text files daily if possible. Back up everything from the …/ctfsweb/files/datafiles/ folder where the text files are located. If you have already started double data screening, then you should back up the MySQL database also.

HOW TO: Back-Up Your MySQL Database via mysqldump

1. Go to the DOS command prompt and type:

```
CD\XAMPP\mysql\bin\
mysqldump -u root --add-drop-table --extended-insert --result-file="/path-to-dump-file/yoursitename.sql" --no-create-db  current_yoursitename
```

** Alternatively, if you want to back up the Temp tables only:*

```
mysqldump -u root --add-drop-table --extended-insert --result-file="/path-to-dump-file/yoursitename_Temptables.sql" --no-create-db  current_yoursitename TempNewPlants TempOldTrees TempMultiStems TempLocations
```

where:

- yoursitename_Temptables.sql is the name that you want to call the "dumped" file
- current_yoursitename is the name of the database you want to back up

**If yoursitename.sql is a large file, you may want to compress it:*

```
zip -r /path-to-zip-file/yoursitename_22May2020.zip /path-to-dump-file/yoursitename.sql
```

We strongly suggest that you include the date in the name of the compressed file. Do not overwrite backups, instead save them all. This way you will always be able to go back to a previous version if necessary.

Error Correction and Backing-Up Data

Error corrections should be accumulated and updates to the database should be made according to a schedule, not every time an error is found. Any updates to the database should be recorded in a script. Updates include saving the original data from the record in the Log table before making the change to the record. A sample script and detailed instructions can be found here. The database should be backed up every time a set of corrections are done. The update scripts should be saved along with the database backup.

Updating and Archiving Stable Databases

The official database (the "stable" copy), or the database to be used for analyses and publications, should be updated no more than once a year. Official databases should be archived in a digital repository (such as GitHub). Repositories maintain your files and ensure that they remain usable in the future. Repositories also provide online access to papers and data for the research community and can serve as a method of publishing files and data, making them more easily citable. Most repositories are public access, but some may restrict access for a few years, or store some parts publicly and maintain other more proprietary parts inaccessible. It is especially important to save files in ASCII format as MySQL and R may not be maintained in the future. A detailed metadata is necessary to explain what data is collected, the quality of the data, units of measure, any divergent methods used, etc.

Appendix C

Directory of Links in Main Text

This is a directory of any links mentioned within the text. Please note that no links were mentioned in Chapters 3 & 6, therefore those chapters are not listed below.

Introduction

SQL Resources

https://www.w3schools.com/sql/sql_intro.asp

Chapter 1: Creating Your ForestGEO MySQL Database

XAMPP Download

https://www.apachefriends.org/index.html

ctfsweb Download

http://ctfs.si.edu/Public/software/ctfsweb/download.php

New_Database_Structure.sql

http://ctfs.si.edu/Public/DatabaseSetup/DatabaseTableScripts/New_Database_Structure.sql

Family.sql

http://ctfs.si.edu/Public/DatabaseSetup/DatabaseTableScripts/Family.sql

Genus.sql

http://ctfs.si.edu/Public/DatabaseSetup/DatabaseTableScripts/Genus.sql

Tree Database Data Dictionary

http://ctfs.si.edu/Public/DataDict/data_dict.php

Chapter 2: Adding Fixed Content to Your Database

createTempTables.sql

http://ctfs.si.edu/Public/DatabaseSetup/DatabaseTableScripts/createTempTables.sql

Chapter 4: Adding Census Data: Working from Electronic Records

createTempTables.sql

http://ctfs.si.edu/Public/DatabaseSetup/DatabaseTableScripts/createTempTables.sql

StepsCreateStagingTables_census2.sql

http://ctfs.si.edu/Public/DatabaseSetup/InstructionsExamples/TeachingExamples/StepsCreateStagingTables_census2.sql

Chapter 5: Screening, Correcting, and Summarizing Content

TempTablesCorrect.sql

http://ctfs.si.edu/Public/DatabaseSetup/InstructionsExamples/TeachingExamples/TempTablesCorrect.sql

Checksum.sql

http://ctfs.si.edu/Public/DatabaseSetup/DatabaseTableScripts/Checksum.sql

Appendix D

Troubleshooting Long Fieldsheets That Won't Save

Some computers have run into memory problems when entering data in ctfsweb, where long field sheets from a quadrat are not saved. When this happens, you may want to change the configuration php file that comes with XAMPP.

HOW TO: Change the Configuration PHP File that Comes XAMPP

1. Go to the "/xampp/php/" folder and make a copy of the original *php.ini* file (i.e. copy it to another file called php_orig.ini).

2. Open *php.ini* in a text editor, look for the following parameters, and make the corresponding changes. Leave a space in front of and after the equal sign:

   max_execution_time = 2400
   max_input_time = 600
   memory_limit = 768M
   post_max_size = 48M

3. Add the following line, maybe after line 460:

   max_input_vars = 20000

4. Save php.ini.

5. Stop XAMPP and restart Apache and MySQL.

*If you are still having problems, change the parameters again by increasing the numbers.

Appendix E

Digitizing and Uploading New Plant Coordinates

Digitizing and Uploading New Plant Coordinates

If your site uses paper maps, we suggest using the image processing and analysis software called ImageJ to digitize the points. This software is open source and available for Windows, Linux, and Mac OS platforms.

HOW TO: Use ImageJ to Digitize Plot Maps

1. Follow the instructions on this website to download ImageJ.

2. Download the modified version of Point Picker, accessible here. Extract the files from the zipped file "pointpicker_modified3.zip" into /ProgramFiles/ImageJ/plugins" (or wherever you installed ImageJ).

   The description and user manual for Point Picker can be downloaded from this site.

3. Scan all your paper maps and save them as JPEGS on your computer. Use the quadrat name (and subquadrat, if applicable) of the map as part of the file name. Be consistent when naming the map images. Examples of possible naming conventions for JPEG map files are: "Q0312.jpg" or "Map_0312_2.jpg" where "0312" denotes the quadrat number and (in the case of the second example) "2" specifies the subquadrat.

4. Open the ImageJ application, then open the specific JPEG map file that will be digitized and ultimately recorded as a text file.

   NOTE: It is strongly recommended that the original paper copy of the hand-drawn map be examined alongside the corresponding datasheet to make sure that all stems are actually mapped for the specific quadrat/subquadrat and that all tag numbers correspond between the map and the datasheet.

5. Once the JPEG image is open in ImageJ, select the "pointpicker" menu option:
   Plugins > pointpicker > PointPicker

6. Select the "Add crosses" button  (the very first option with the [+] sign) to add points to the four corners of the map. The four corners of the subquadrat map should be digitized first for calibration, and should be done in the following order and labeled as follows:
   p1 = lower left
   p2 = upper left
   p3 = upper right
   p4= lower right

7. Select the "Add crosses"  button to digitize the trees by putting the cursor over the center of each of the points. Label the points with the complete tag number.
   NOTE: In sites where the multiple stems from one tree may have different locations, each multiple stem should be digitized. If there is a case where only one dot is drawn in for a group of multiple stems, it is suggested that crosses be added to the main stem tag (the drawn dot) *and* the non-represented stems around the dot to account for the multiple stem tags.

One of the most common errors when digitizing maps is to leave out points. Another is to label the points incorrectly. There are several ways of preventing this:

a. One suggestion is to digitize the points in sequential tag order, with out-of-sequence tags done at the very end.
b. Another approach would be to finish one subquadrat at a time, making sure to digitize all the stems from that subquadrat.

You can use the following features to examine and edit the crosses/labels that you have added to your map:

a. The "Magnifying glass" feature  may be used to zoom in to areas of the map where stem dots or tag numbers appear small at the original magnification, or where many dots are clustered close together. Use the left-click button on the mouse to zoom in and the right click button to zoom out.

b. The "Edit label" button  (the second icon with a plus sign. *NOTE:* the first icon with a plus sign is the "Add crosses button") allows you to edit tag numbers.

c. The "Remove crosses" button  allows you to delete crosses.

d. The "Move crosses" button  allows you to adjust crosses that appear off-center.

HOW TO: Use ImageJ to Digitize Plot Maps (cont'd)

8. After all crosses have been added, check the map image to ensure that every drawn-in tagged stem has an added cross to it.

9. Review the Results list to ensure that all four corners and all the tagged stems are displayed in the list and labelled correctly. Access the Results list by clicking the "Export/Import list of points" button  and then clicking on the "Show" option.

10. When everything is verified as being correct for the now digitized quadrat or subquadrat map, save the list of points and corresponding x,y coordinates to a text file by clicking the "Export/Import list of points" button  and then clicking the "Save as" option.

    NOTE: Be consistent when naming the text files. The text files should be saved following the same naming convention that you used for the scanned maps, e.g. "Q0312.txt" or "Map_0312_2.txt" We suggest that the resulting text files be saved in directories denoting the column, e.g. "C:\Maps\Column03"

11. Following completion of the digitized quadrat/subquadrat map, the image must be closed before opening the next quadrat/subquadrat map image. This can be done in one of two ways:

    a. Click on the "Exit PointPicker" button  , then click the "Done" option to exit the "PointPicker" feature. This will remove all added crosses from the map image and return to the original ImageJ configuration. Then close the map image. Now the next image may be opened.
    b. Simply close the ImageJ application altogether and then reopen the application to completely reset it for the next map image.

* If a completed digitized map needs to be reopened to review the placement of crosses on the map image or the Results list, simply:
  - open the specific quadrat/subquadrat map image in ImageJ
  - select the "PointPicker" menu option

- click on the “Export/Import list of points” button 
- click the “Open” option to open the text file that corresponds with the matching quadrat/subquadrat map image.

All added crosses should appear in correct placement over the map image. The Results list can be accessed following the same procedures outlined in step 9.

After digitizing the maps, you’ll need to convert ImageJ text files from pixels to x,y coordinates in meters.

HOW TO: Convert ImageJ TextFiles from Pixels to Meters using R

1. Open R (The R statistical software)

2. Call up the CTFS R Package in R using the following command:

   `attach('/path/CTFSRPackage.rdata')`

   NOTE: If you haven’t already done so, download the CTFSRPackage following the instructions found here.

3. Use the fullplot.imageJ function. You can find a tutorial on the fullplot.imageJ function here. Before you begin, choose the correct parameters for fullplot.imageJ according to the path where the text files are located, the size of your quadrats/subquadrats, and how you named your map files. For an explanation of the parameters, visit this link.

   e.g. fullplot.imageJ(path='/maps/Column02', gridsize=c(10,10),
   outfile='ConvertedCoord1.txt', prefix='Map_', colrange=c(0,49),
   rowrange=c(0,24), corners=c('p1','p2','p3','p4'),
   subquadsuffix=c('_1','_2','_3','_4'))

   According to the above example:

   a. This will put all the converted coordinates with their tag numbers and corresponding quadrat names (from the name of the text files) into a file called “ConvertedCoord1.txt”
   b. This file will be saved in the path that you specified: /maps/Column02/
   c. All files in this path with “.txt” extension, “Map_” prefix, and subquadrat suffix specified above will be converted: Map_0201_1.txt, Map_0222_2.txt., etc.
   d. The subquadrat suffix should be named in a clockwise direction starting from the lower left corner.
   e. The calibrated corners should also be named in a clockwise direction from the lower left corner.

   In the case that there are no subquadrats, only 20x20m quadrats:

   e.g. fullplot.imageJ(path='/maps/Column02', gridsize=c(20,20), outfile='Converted.txt',
   prefix='Q', colrange=c(0,49), rowrange=c(0,24), corners=c('p1','p2','p3','p4'))

   All files in this path with “.txt” extension and “Q” prefix will be converted (Q0301.txt, Q0324.txt, etc.) and saved into a text file called “Converted.txt.”