FOREST INVENTORY DATA MANAGEMENT AND PROCESSING WITH OPEN FORIS TOOLS

- PROJECT EXAMPLE -

Lauri Vesa

29th November, 2019

Contents

Abbreviations and acronyms

AG	Above-ground
AGB	Above-ground biomass
AGC	Above-ground carbon
AOI	Area of interest
BG	Below-ground
BGB	Below-ground biomass
BGC	Below-ground carbon
C	Carbon
CSV	Comma separated value (file)
dbh, DBH	Breast height diameter
DW	Dead wood
FAO	Food and Agriculture Organization of the United Nations
FREL/FRL	Forest Reference Emission Level / Forest Reference Level
GIS	Geographic Information System
IDMM	Inventory Data Metamodel
IPCC	Intergovernmental Panel on Climate Change
MS	Microsoft
NA	Not available (in data)
NFI	National Forest Inventory
OF	Open Foris
PoM	Point of Measurement
R	R - Statistical programming software and language
SQL	Structured Query Language
WD	(Dry) Wood density

PREFACE

This document contains description of the data management and processing for computing forest inventory results when using FAO Open Foris (OF) tools. These softwares are available at http://openforis.org/. The content of this paper is collected from technical reports prepared for various National Forest Inventories (NFIs).

The purpose of this document is to provide learning materials and examples for implementing forest inventory data management and processing chains. Secondly, this document aims to outline good practices for the data management processes that will ensure the good quality of information produced.

Typically, projects using OF tools apply OF Collect and Collect Mobile for data entry and validation and OF Calc for data analysis. OF Calc is a robust, modular browser-based tool for results calculation. Calc comes along with Saiku reporting tool that provides a flexible way to produce aggregated results from collected field data. The aggregated results shown with Saiku can be exported into Microsoft (MS) Excel or R for further analysis and visualization.

The data calculation scripts for Calc are written using R language. R is the leading tool for statistics, data analysis, and machine learning. It is more than a statistical package; it's a programming language. Expert users need to write custom R modules to perform country/inventory-specific calculations with Calc. One advantage using R is that scripts also work as a document of the data processing chain (see Chapter 4).

The NFI sampling design follows the next principles:

I. NFI adopts a stratified sampling approach. The country has been divided into three strata: Uplands, Wetlands and Mangrove forests.
II. The clusters were systematically selected by strata with 3 different sampling intensities..
III. The sample plots are grouped as clusters (Figure 1). There are 3 plots in a cluster in Uplands and Wetlands strata, and 4 plots in Mangrove stratum.
IV. Nested circular plot design is used for in the field assessment (Table 1).

Each nested sample plot is a either a homogenous land unit or it can be divided into land use/land cover sections. Trees are recorded according to their diameter applying 3 different radii/subplots. Bamboo, lying dead wood and stumps are recorded within a fixed subplot area. Regeneration and shrubs data are recorded in fixed-area smaller subplots too.

Sample plots are gpouped into clusters, and it is assumed that plots are statistically independent. However, cluster structure is taking into account in computing the realibitily estimates. In this forest inventory case a cluster consists of three sample plots and a nested circular plot design is used for in the field assessment.

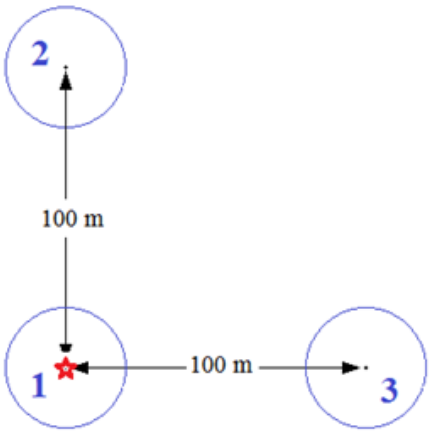


Figure 1. Cluster design

Table 1. Entities and plot sizes

Unit name	Radius	Condition
Cluster		
Plot	21.85 m	Trees: *dbh* ≥ 30 cm
Subplot 1	11.97 m	Trees: *dbh* ≥ 15 cm
Subplot 2	5.64 m	Trees: *dbh* ≥ 5 cm
Subplot 3 for saplings, shrubs, and climbers	2.82 m	Trees: 1cm≤ *dbh* <5 cm Shrubs, climbers: *dbh*≥ 5 cm
Subplot 4 for seedlings, small shrubs and climbers	1.13 m	Trees: *dbh* ≤ 1cm or no *dbh*; Shrubs, climbers: 1cm ≤ *dbh* < 5cm
Subplot (1) for dead wood (lying) and stumps	11.97 m	diameter ≥ 10 cm
Subplot (1) for bamboo	11.97 m	height > 1.3 m

For determination of the land cover classes, 27 classes are based on vegetative or land-use type characteristics to be recorded in the field (3rd column in Table 2). The land use/vegetation type classes with their aggregated classes (i.e. major classes, FRA and IPCC) can be used in the data processing chain and as reporting units in Open Foris Calc.

Table 2. Land cover classes

FAO-FRA *(for reporting)*	**Major LUVS Class** *(for reporting)*	**Land use/Vegetation type Class** *(recorded in the field)*	**IPCC 2006 Categorization** *(for reporting)*
Forest	Evergreen forest	Evergreen forest	Forest Land
	Semi-evergreen forest	Semi-evergreen forest	
	Deciduous forest	Deciduous forest	
	Flooded forest	Flooded forest	
	Mangrove forest	Mangrove forest	
		Rear mangrove forest	
	Forest plantation	Forest plantation, broadleaf	
		Forest plantation, coniferous	
		Rubber plantation	Cropland
		Oil palm	
	Bamboo	Bamboo	Forest Land
Other Wooded Land (OWL)	Shrubland	Shrubland	Grassland
		Flooded shrubland	
Other Land	Grassland, Abandoned field, Marsh	Orchard	Cropland
		Grassland	Grassland
		Flooded grassland	
		Abandoned field	
		Marsh and swamp	Wetlands
	Built-up/Barren Areas	Barren land	Other Land
		Infrastructure	Settlements
		Rocky outcrop	Other Land
		Sand bank	
		Settlement	Settlements
	Cropland/paddy	Cropland	Cropland
		Shifting cultivation	
Water	Water	Fish farming, Salt plant	Wetlands
		Water	

1. DATA MANAGEMENT

1.1. Required software

OF Collect and Calc run in MS Windows, Linux or MacOs operation system. OF Collect Mobile works only on Android gadgets. Open Foris Collect and Calc require preferably Google Chrome or MS Edge web-browsers. Adobe Flash Player needs to be installed and enabled (see the Collect User's Manual).

The list of the other required software is presented in Table 3. All other softwares except MS Excel are freeware. It should be noted that some software have installation packages for 32 and 64-bit operating systems.

Table 3. Required and recommended software

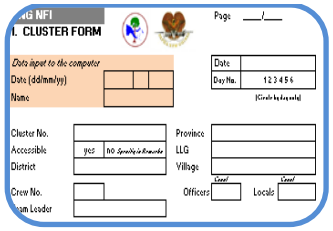
Name	Data entry & validation	Data management	Data analysis and reporting	Other useful tools	32/64 bit versions	WWW site
OF Collect	X	X				http://www.openforis.org/
OF Collect Mobile	X					Google Play store
PostgreSQL		X	X		YES	http://www.postgresql.org/
Java SE Development Kit (JDK 8)			X		YES	http://www.oracle.com/technetwork/java/javase/downloads/
OF Calc (comes with Saiku)			X			http://www.openforis.org/
MS Excel or Libre Office Calc	(X)		X			
R (v. 3.5.x or newer)			X		YES	https://www.r-project.org/
+ required R packages (see chapter 2.3)						
RStudio			X			https://www.rstudio.com/
NotePad++				X		https://notepad-plus-plus.org/

Note 1:
Java JDK, PostgreSQL, R, RStudio and Calc are all needed to run Calc. Please **read first** Calc User's Manual before installing these applications! Especially correct installation of R and required packages is essential in order to avoid problems.

Note 2:
If Saiku does not work and it just says "Running query..", in Chrome browser type URL chrome://extensions/
and try to disable all extensions, or change Web browser for running Saiku.

1.2. Overview of data management process

The next figure shows the flow of data from the field up to the data analysis phase.



Field form data

Digital data in tablet (OF Mobile)

* Keep photocopies or scanned copies of the field forms at a local office, if possible.
* Transfer field forms to the main office.
* Data is entered manually into the database.

* After a working day, data is exported from tablet and then copied to a computer.
* Keep a safety copy of the data.
* Data file is then sent to main office via email, Google Drive, DropBox, or carried by usb stick.
* Data file is imported into OF Collect database at PMU.
* *Data can also be sent to a Collect network server directly from tablet.*

OPEN FORIS COLLECT database

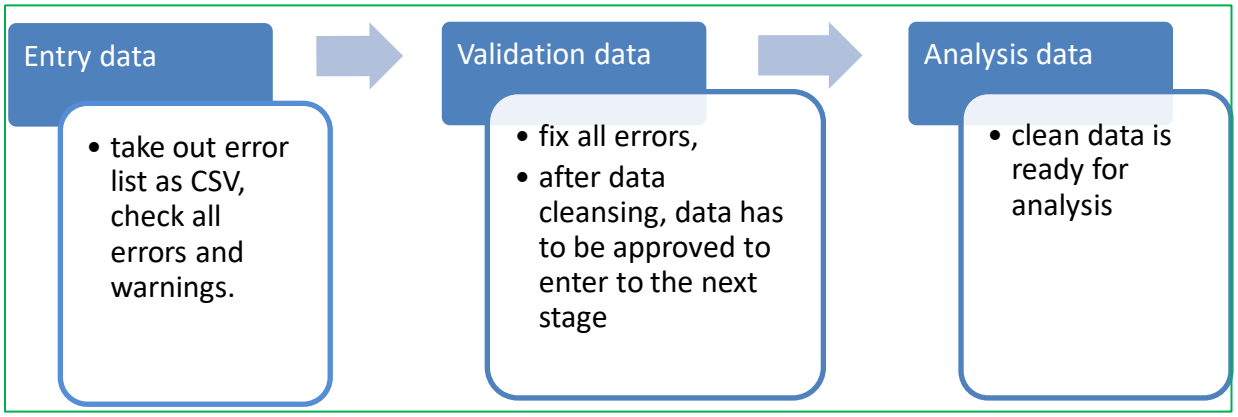


Figure 2. Raw data workflow

1.3. Data hosting

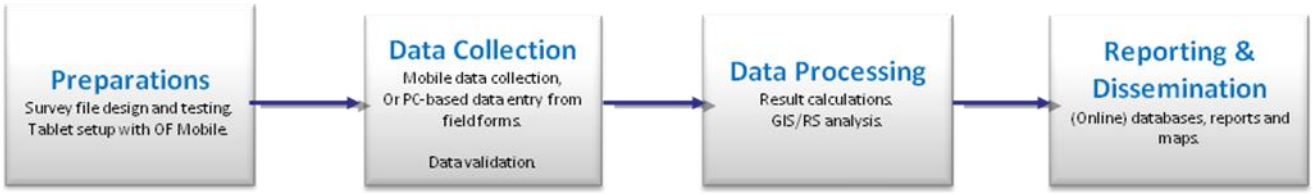
Open Foris Collect database works as the main entry point for collected data. OF Collect provides a solution for field data management, allowing full customization of inventory structure, variables and data validation rules. Data entry user interface (UI) is automatically generated and metadata driven. The system can be used in a standalone environment with no need for the Internet connection or installed into the server where in multi-user environment where users can work only on owned records. In a larger project as National Forest Inventory case, the server installation needs to be applied and the server provides the storage space for hosting the data.

By default, Collect uses SQLite database in the local machine for storing data[1]. However, PostgreSQL database should be used in the server installation. Collect Manual's annex 2 contains instructions for setting up Collect to use PostgreSQL[2].

There are three steps of data validation levels in OF Collect. Copies of data will be kept in the database for each level. The workflow using OF Collect is presented in Figure 3 and the overall data processing chain in Figure 4.

Figure 3. Integrated workflow in Collect

"Entry" level is for storing data from the field forms. After entering all data from a cluster plot, the cluster needs to be submitted to "Cleansing" level in OF Collect.
All data cleansing will be done for "Cleansing" level data. Data recorded with Collect Mobile goes automatically into "Cleansing" level when imported into Collect.



[1] In SQLite option, the data is located here: *C:\Users\YOUR_USERNAME\OpenForis\Collect\data\Collect.db*
[2] PostgreSQL, often simply Postgres, is an object-relational database management system with an emphasis on extensibility and standards-compliance. It can handle workloads ranging from small single-machine applications to large Internet-facing applications with many concurrent users. For setting up Collect for multi-user environment, read http://www.openforis.org/support/questions/1821/setup-collect-desktop-in-a-server-for-a-multi-user-environment

Figure 4. Overall data processing chain

OF Collect basic security is taken into account by giving different user-rights for its users. Each user can get a unique username and password. Possible user roles in Collect are shown in Table 4.

Table 4. User roles in Collect

	View	Entry Limited	Entry	Cleansing	Analysis	Design	Admin
View records	✓	✓	✓	✓	✓	✓	✓
Create new records			✓	✓	✓	✓	✓
Edit records in Entry phase		✓ (only owned records)	✓	✓	✓	✓	✓
Edit owned and not owned records from Entry to Analysis phase				✓	✓	✓	✓
Submit records from Entry to Cleansing phase		✓ (only owned records)	✓	✓	✓	✓	✓
Reject records from Cleansing to Entry phase				✓	✓	✓	✓
Reject records from Analysis to Cleansing phase				✓	✓	✓	✓
Delete records (only in Entry phase)			✓	✓	✓	✓	✓
Saiku analysis				✓	✓	✓	✓
Data Cleansing				✓	✓	✓	✓
Survey Designer						✓	✓
Backup/Restore							✓
Users/Roles management							✓

Typical roles of the personnel in the forest inventory case can be as follows:

- Data entry staff: users have access only to data entry phase, and they are allowed to enter new records, edit own records, submit them for cleansing and export records. Theses tasks can be performed by relevant members of the field team and/or office staff.
- Data cleansing/analysis experts: Same as Data entry + permission to edit records in cleansing phase and submit them to data analysis. Will be performed by the Forest Inventory Expert(s) and other nominated staff.
  Note: Analysis role gives ability to unlock records and re-submit them back to data cleansing phase.
- Administrator: Full access rights.

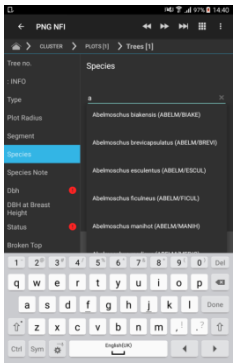
Only a few people in the organization should have “Admin” rights for the server. Suitable staffs with Administrator rights are the database expert, leading forest inventory or other technical experts.

1.4. Data entry

Data recording in the field is carried out with the help of tablets. An alternative method is to use field forms and carry out the data storing at the office. This will be performed by a well-trained member of the field team.

OF Collect Mobile is a data collection tool for field-based surveys using tablets.
This is a free Android app that allows the completion of complex data structures.
Its features include:

- On-the-fly validation to improve data quality;
- Handling of lists of species or other attributes, as defined in Collect Survey Designer;



- Integration with Collect for data management and analysis;
- Processes inputs and can calculate some new attributes, if needed.

Data transfer from the tablets can be organized in many ways. The first step is to take a data backup in Collect Mobile. The backup file will be stored into tablet's micro-SD card. If the tablet has the Internet connection, the backup can be sent by email to the Project Management Unit (PMU), or stored into the cloud server (as Google Drive, OneDrive). Data can be also exported first into Downloads folder in the tablet, and then copied to a laptop and transferred later to PMU. Data can be also sent directly to a Collect network server running in the organization.

Conventional use of paper field forms requires that the field officer delivers the forms to the office where they are inputted manually into the main Collect database.

NOTE 1.

In Collect and Collect Mobile there are two levels of data validation messages:

- **Errors** for impossible values;

- **Warnings** for values to be checked (being possibly errors).

Always correct errors and double-check warnings whether there is erroneous input data.

NOTE 2.

Collect and Collect Mobile: The data export/backup should be taken always before updating the software!

NOTE 3.

Collect Mobile: Updating of the application from Google Play Store will not delete data, **but updating of the Collect survey will always delete data in the tablet**!

Collect Mobile menu commands

Collect Mobile has a menu command on the right upper corner of the screen (Figure 5).

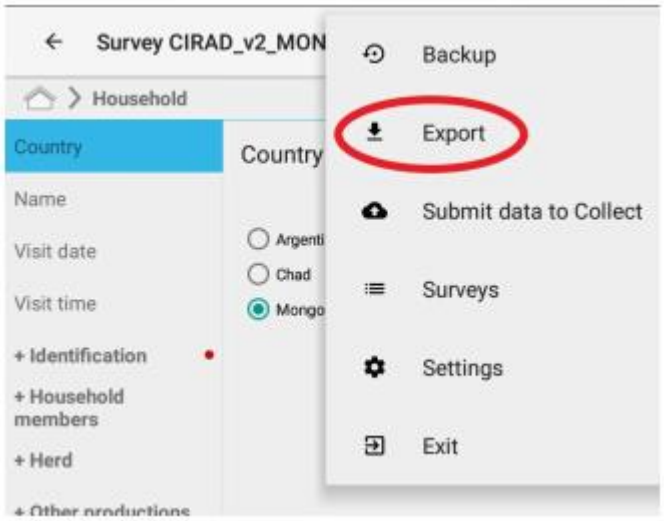


Figure 5. Mobile menu commands

The menu commands are as follows:

- **Backup**: data transfer to backup file. This is the main method of taking collected data out from the tablet. The backup file will be written into micro-SD card.
- **Export**: data export to *Downloads* folder, DropBox, Google Drive, email to other applications (in the tablet).

- **Submit data to Collect**: sent directly to the central server.
- **Surveys:** Selection of survey. Also Importing a survey is here.
- **Settings:** Mobile settings (Note: there is typically no reason to change the default settings).

1.5. Data cleansing

In Data Management view of Collect, you can run a *Validation Report* to investigate the nature of the errors shown in the list of records. Click on the *Validation Report* button (Figure 6). The result will be a CSV file with details on the error(s) found in the records.

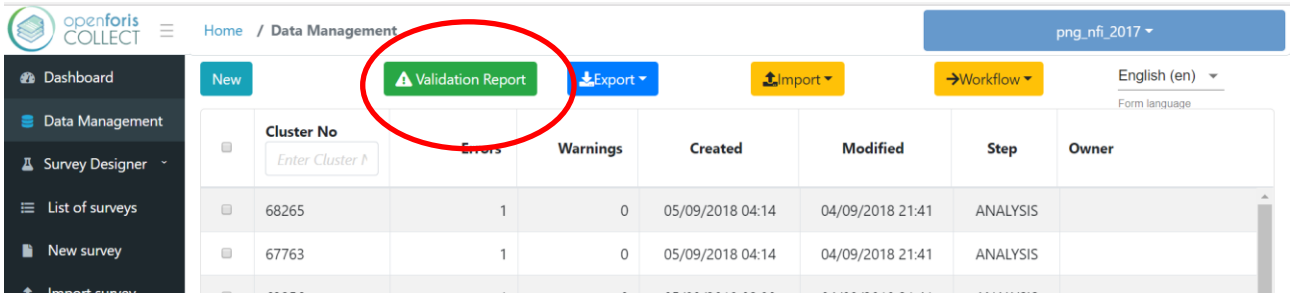
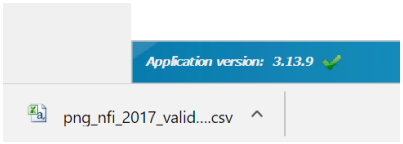
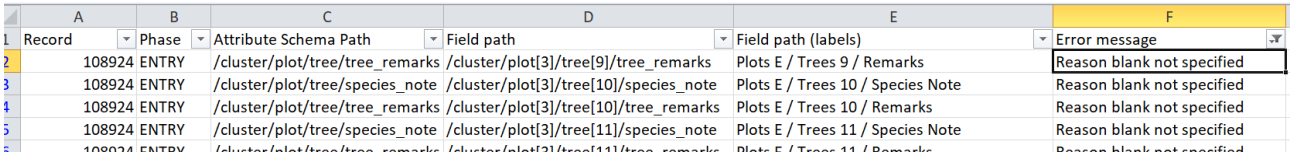


Figure 6. Validation Report command in Collect

The validation report will be written into your default *Downloads* folder.



Next, open this CSV file in Excel (or other spreadsheet software). In Excel, first enlarge the column widths for columns C – F. Then go into cell *A2*, and freeze panes (*View, Freeze panes*) so that you can always see the header row. Next, set the filter for the data (*Data, Filter*) (Figure 7).



	A	B	C	D	E	F
1	Record	Phase	Attribute Schema Path	Field path	Field path (labels)	Error message
2	108924	ENTRY	/cluster/plot/tree/tree_remarks	/cluster/plot[3]/tree[9]/tree_remarks	Plots E / Trees 9 / Remarks	Reason blank not specified
3	108924	ENTRY	/cluster/plot/tree/species_note	/cluster/plot[3]/tree[10]/species_note	Plots E / Trees 10 / Species Note	Reason blank not specified
4	108924	ENTRY	/cluster/plot/tree/tree_remarks	/cluster/plot[3]/tree[10]/tree_remarks	Plots E / Trees 10 / Remarks	Reason blank not specified
5	108924	ENTRY	/cluster/plot/tree/species_note	/cluster/plot[3]/tree[11]/species_note	Plots E / Trees 11 / Species Note	Reason blank not specified

Figure 7. Validation report in Excel

Now you can filter your report by header 'Error message' (Figure 8). In this list you will see error messages generated by your validation rules, and some checks that Collect will do.

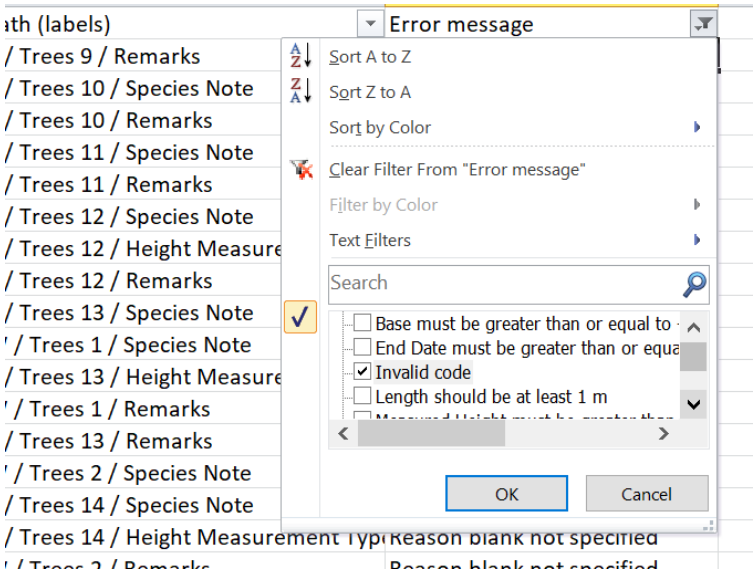


Figure 8. Filtering of error messages in Excel

Please notice that plot and tree numbers in this report refer to index (i.e. position) of the record in the database, and they may not refer to the actual attribute (as '*tree_no*') value(s) in the recorded data (Figure 13).

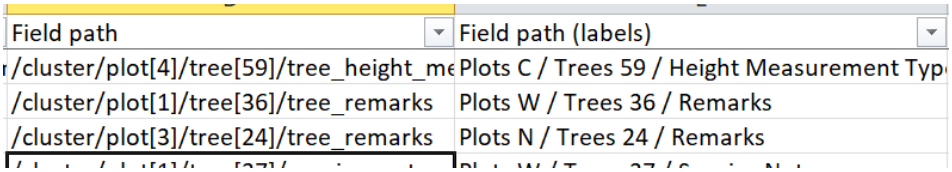


Figure 9. Plot and tree indexes in the validation report

If your data has been entered manually from paper forms into Collect database, then you may see several errors labeled as '*Reason blank not specified*'. See guidance at
http://www.openforis.org/support/questions/895/how-to-solve-error-reason-blank-not-specified

- *inserting a new record in Collect (Collect Desktop): you are supposed to fill all values, even the ones that are not required according to the survey definition; what you can do is to specify a "reason blank" for the fields that have no value in the paper form you are copying into Collect. You can do this by right clicking on a field a selecting one of the possible reason blank options or you can use one of the shortcuts available (*=Blank on Form, -=Dash on form, ?=Illegible).*

Once you have fixed some or all possible errors in your data, rerun *Validation report*. After viewing your validation report, you may also notice that some of your validation rules may be too strict or even wrong. In this case the best solution is to open *Survey Designer* in Collect, modify validation rule(s) of the current schema, save and publish the survey.

1.6. Collect Survey Designer

Collect introduces the concept of the **Inventory Data Metamodel** (IDMM) which is a formal description (i.e., metadata) of the types of variables, classifications and coding schemes used by the inventory. These are all stored into a "survey file" that can be used both in OF Collect and Collect Mobile using tablets. IDMM is created and maintained using Survey Designer tool. More specifically, the Survey Designer is used to maintain survey metadata, code lists, species lists, sampling point data, and inventory schema (Figure 10).

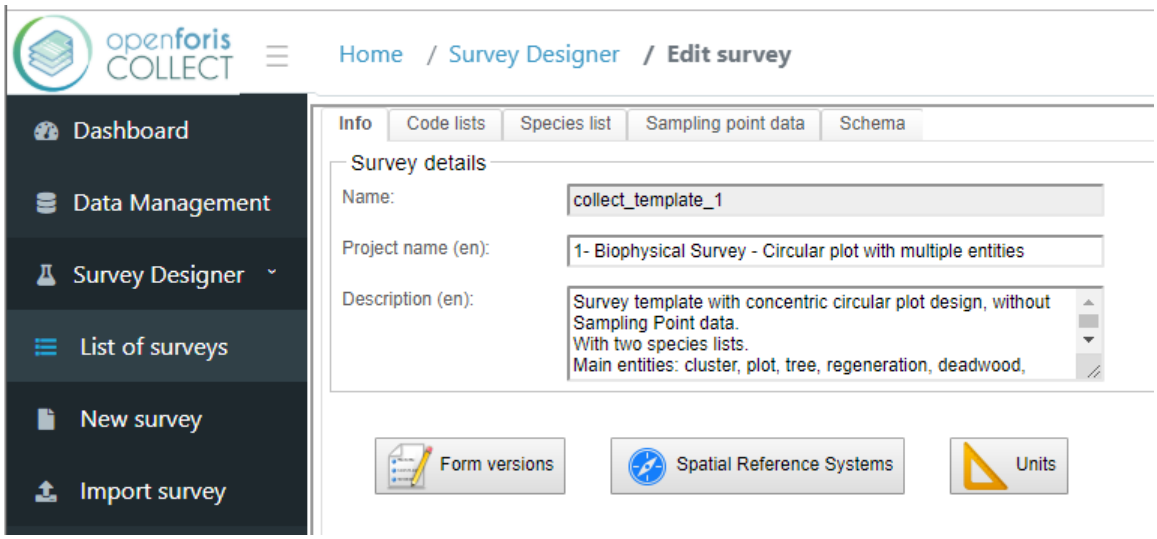


Figure 10. Collect Survey Designer view

The actual IDMM file is a XML type text file. This file can be exported by selecting *Export* command in the Survey Designer's list view (Figure 11).

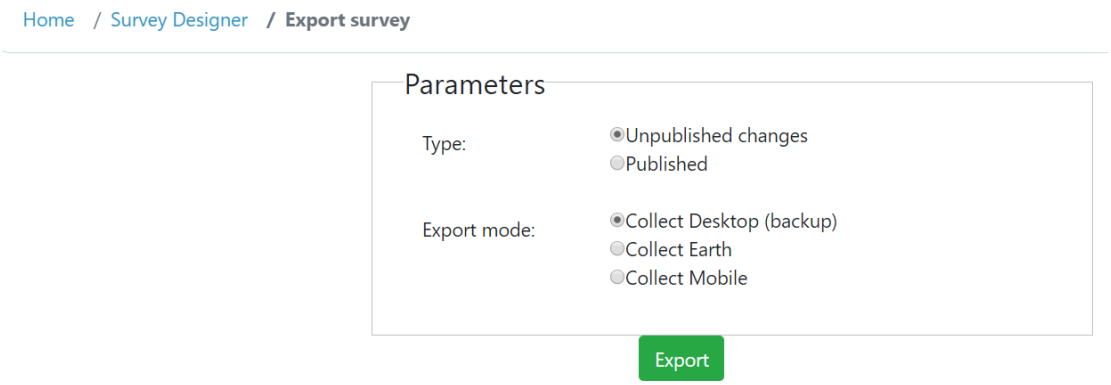


Figure 11. Collect survey export

Export to *Collect Desktop* creates a file with the name similar to
***surveyname**_2016-08-19T15-20-03.collect*
and export to *Collect Mobile* creates a file with the name similar to
***surveyname**_2016-08-19T15-20-32.collect-mobile*

The export survey file is actually a zip type file, see Figure 12. The file ***idml.xml*** contains the full inventory schema. However, do not start editing these files manually unless you are really confident what you are doing!

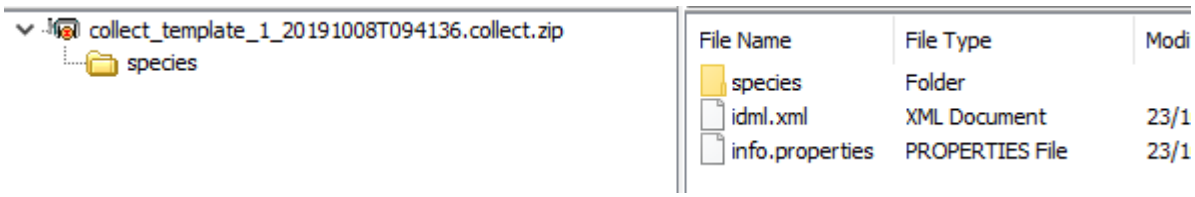


Figure 12. Content of *.collect* file (in zip)

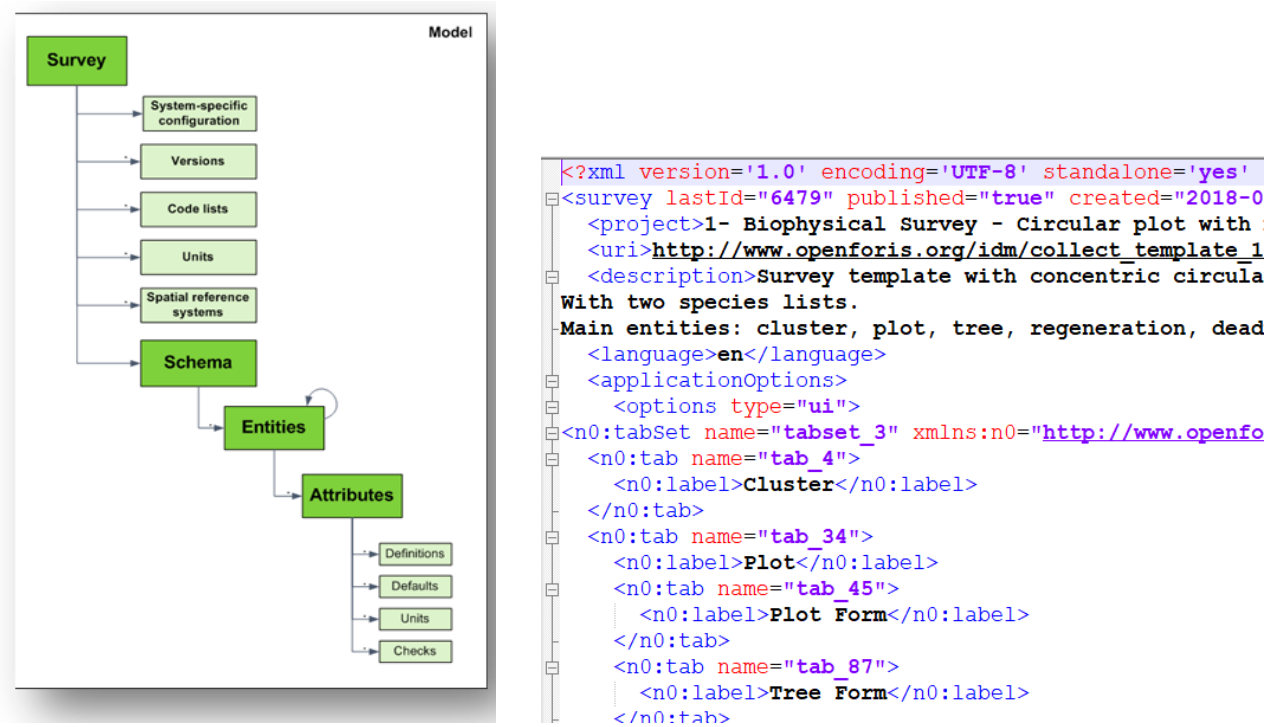


Figure 13. Survey structure and schema in XML file (opened in NotePad++ on the right side)

The schema can contain several entities that are hierarchically organized. The entity names of one survey example are presented in Figure 14. The list of clusters can be predefined and given for Collect via a 'Survey point data' table. This 'survey point data' can contain also the list of plots by clusters, plots' coordinates and other necessary information as administrative information (e.g. region or province, district).

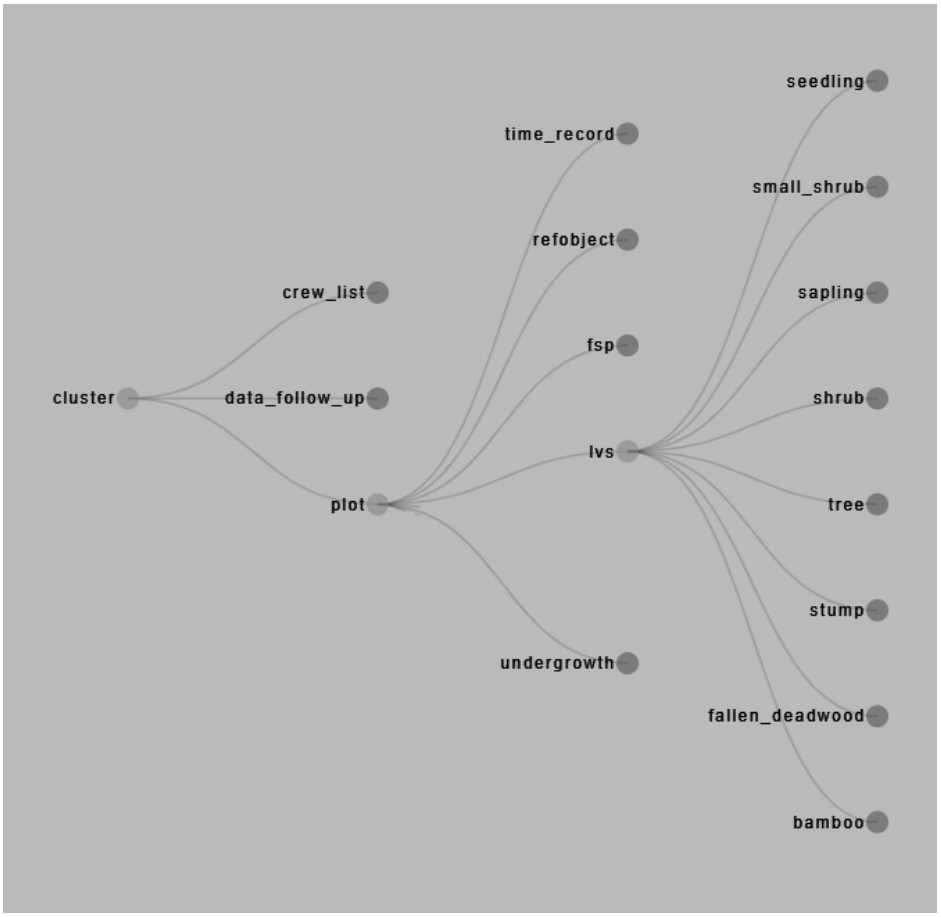


Figure 14. Example of main entities in a Collect schema

When some field data has been collected and recorded into the Collect database, doing changes to the Collect survey is limited. This means that the **current attributes, categories and codes cannot be deleted**. Only new attributes and new codes into the code lists can be added into the survey, and existing attribute labels (but not attribute names) can be edited and changed.

New species can be added and existing species renamed, but **species codes cannot not be changed nor deleted from the species list**.

Data validation rules can be changed, added or deleted.

2. DATA PROCESSING – PROJECT DEMONSTRATION CASE

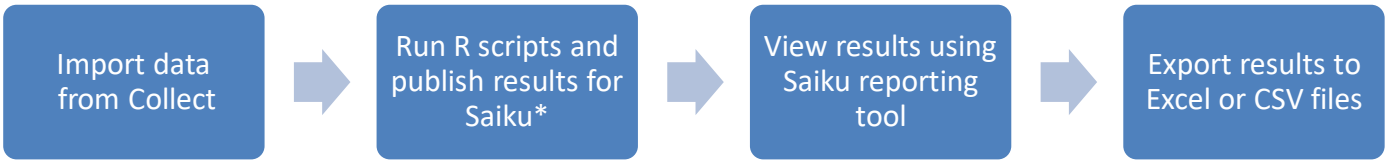
2.1. Overview

The project will use Open Foris Calc and R language for processing of the main biophysical results, and Saiku for reporting of these results. Excel or R can be used to produce graphs.

Calc is single-user software, and it is installed along Saiku reporting tool. Saiku reports can be accessed by multiple users via Local Area Network (LAN) or Internet. R is the script language for data processing and statistical analysis. R scripts are written and edited using RStudio, but when they are ready, these scripts can be run directly in Calc.

OF Calc is modular browser-based software for analysis and reporting of results of sampling based natural resource assessments, and it allows expert users to run custom R scripts to perform inventory-specific calculations. Still, part of the data analysis, visualization (e.g., graphs) and reporting will be done using Excel or R. There are manuals both for OF Calc and Saiku available via Open Foris web site, and plenty of materials about R and RStudio are available on-line.

The next figure shows the general working flow with Calc.



* Calc (via R) can also write results out directly into CSV, PDF or graphic files (as JPG).

Figure 15. Work phases with Calc

The input data for Calc is taken from OF Collect and its “analysis stage” data.

To make the calculation and reporting processes work **for the very first time** in Calc, the next main steps need to be completed:

1) [Calc:] Create workspace for the inventory,
2) [Calc:] Import back-up data file taken from OF Collect (from “analysis data stage”),
3) [Calc:] Set survey settings,
4) [Calc]: Create calculation and aggregation modules, export them into an R project,
5) [RStudio]: write R scripts,
6) [RStudio:] Test calculation process, Run scripts (by Source command),
7) [Calc:] Run calculation command,
8) [Saiku:] Conduct queries in Saiku, and save useful queries as templates for reporting.

When new data is available for analysis and Calc workspace is done, the list of work phases is brief:

1) [Calc:] Import *collect-data* file, got from OF Collect “analysis” stage data,
2) [Saiku:] Conduct queries in Saiku.

Saiku reporting tool provides a flexible way to produce aggregated results for any reporting area (or domain) of interest, as by forest types, FRA categories, provinces, etc.

2.2. Workspace

The concept ***workspace*** in Calc contains all R scripts, auxiliary tables, external equations and sampling design but no data. It can be exported from Calc and imported into another computer.

2.3. Required R packages and output data folder

Many useful R function come in packages, free code written by R's active user community. . Packages are collections of R functions, data, and compiled code in a well-defined format. The directory where packages are stored is called the library. R comes with a standard set of packages. Others are available for download and installation. Once installed, they have to be loaded into the session to be used.

Calc requires the following packages: sqldf, RPostgreSQL (see Calc User's Manual, part Installation).
The inventory-specific workspace can use several R packages (plus all their dependent packages) which must be installed in the computer. Some commonly applied packages are these ones:

- lmfor – functions for (tree) height modeling,
- ggplot2 – for plotting graphics,
- gridExtra – for organizing plotted (*ggplot2* type) graphs,
- dplyr – for (efficient) programming,
- tidyverse – for programming style,
- vegan – functions for computing biodiversity indexes,
- BIOMASS – functions to estimate AG biomass/carbon and its uncertainty in tropical forest,
- sp and rgdal – for KML plot data outputs, coordinate conversions, etc.

While running R scripts, some output tables (as CSV files) and graphs can be written out into a predefined folder set in variable '*FolderResult'* (see module *common.R*). Therefore if this output folder does not to exist in the computer, the script can create it if needed. The location of output files is in

C:\Users\USER_NAME\OpenForisCalc\testdata-output

2.4. Sampling design in Calc

The input data structure (i.e. metadata) and variable names come automatically from OF Collect database when Collect data is imported into Calc. Calc can only use input data that is coming from Collect[3].

Calc requires that the inventory design is defined in the section 'Settings'. In Calc in this example the reporting unit is '*stratum'* and the 'base unit' or the smallest homogenous unit (in terms of the land use/cover) is '*plot*' (Figure 16). The sampling design follows cluster sampling method.

[3] In Collect, all data for Calc analysis must be submitted into "Analysis phase".

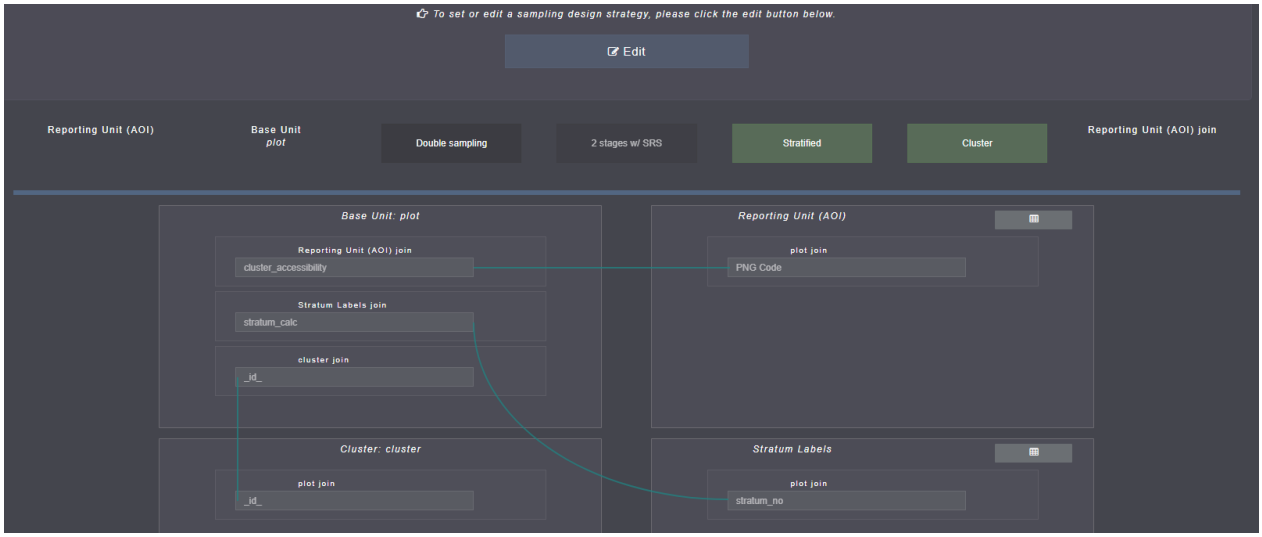


Figure 16. Sampling design view in OF Calc

In case of a stratifield inventory, the areas of strata are taken either from the inventory design (map) or from the 1st phase assessment conducted for example with the help of OF Collect Earth survey. The lookup table showing reporting levels and areas (in hectares) are read into Calc from a CSV file. The content of this file is shown in Table 5 for a case where a whole country is divided into 3 strata.

Table 5. CSV Lookup table showing reporting levels and areas (in hectares).

	A	B	C	D	E
1	level_1_code	level_1_label	stratum_code	stratum_label	stratum_area
2	0	My Country	1	Uplands	1230000
3	0	My Country	2	Wetlands	1300000
4	0	My Country	3	Mangrove	36000
5					

2.5. Result variables

The main result variables are listed in Table 6. In Calc, all results are computed for each record (as a single tally tree) first. The aggregation is done using Saiku, and per hectare results are taken out in Saiku.

Table 6. The main computed variables by entities and plant types.

Entity name *(in Collect DB)*		*tree*		*seed-ling*	*small_shrub*		*sapling*	*shrub*		*bamboo*	*stump*	*fallen_ dead wood*
		Tree (T)	Palm (P)	Seed-ling	Shrub (S)	Liana (L)	sapling	Shrub (S)	Liana (L)	Bamboo	Stump	Dead wood
Count (i.e. stocking)		X	X	X	X	X	X	X	X	X	X	
Height / Length	m	X	X					(X)	X			
Basal area	m^2	X	X				X	X				
Bole volume	m^3	X										
Volume	m^3	X	X				X				X	X
Above-ground biomass	tons	X	X				X	X		X	X	
Below-ground biomass	tons	X	X				X	X		X	X	
Total biomass	tons	X	X				X	X	X	X	X	X
Above-ground carbon	tons	X	X				X	X		X	X	
Below-ground carbon	tons	X	X				X	X		X	X	
Total carbon	tons	X	X				X	X	X	X	X	X
(Total) CO_2	tons	X	X				X	X	X	X	X	X
AGB before felling	tons										X	
AGB removal	tons										X	

Reporting of soil sample results will be done separately because the calculation of carbon in biomass sample plots requires laboratory analysis of collected samples.

In Calc, the calculation applies method where all accessible plots get weight one (1), and inaccessible plots get always weight zero (0). Hence only accessible plot are taken into the analysis. The plot means, as biomass per hectare, are multiplied by corresponding plot weight, and summed up for the category of interest. This sum is then divided by the sum of the weights to get mean estimate in this category.

In Calc and Saiku, area estimates for categories can be reported using the same method. The proportion of each category in the stratum is equal to the sum of weights of this category divided by the total sum of weights in the category.

In addition to Saiku reports, the R scripts documented in this paper will create several external output files as listed in Table 7.

Table 7. External output files created by Calc scripts.

File name	**Purpose**
MyCountry_dbh-height.pdf	Plot: Current top height curve and sample tree DBH-Height data
MyCountry_Plot_Results.csv	Plot level results, per hectare.
MyCountry_3_common_height_models_trees.pdf	3 fitted tree height curves in sample tree data
MyCountry_fitted_height_model_residual_trees.pdf	Residual and standard error for plots from *lmfor*
MyCountry_fitted_height_models_trees.pdf	*lmfor*: charts on fitting tree height estimates

2.6. Area estimates

The area estimates in in this NFI are taken from remote sensing data. Hence, Calc is primarily used to report results 'per hectare', as emission factors for Forest Reference (Emission) Level (FREL/FRL) reporting. However, areas of some land categories cannot be captured from satellite images, so they can be estimated using field sample data.

The method in computing area estimates in Calc follows in this example so-called point sampling method[4]. The points serve primarily as locators of sites where data will be collected. According to the inventory design, in each stratum there is a systematic dot grid, and that is a form of point sampling. The proportion of the number of dots (or points) judged to be "in" is multiplied by an appropriate expansion factor to determine the total area contained within the stratum or domain of interest. Each dot of the grid represents a sample point in a point sampling design.

In case of a (nested) circular sample plot, the geographic location of the field plot's center point defines the domain of the plot. According to the NFI Field Manual, plot center is always located in plot section 'A', and this plot section is showing the properties of the plot, as land cover type. In case of a rectangular sample plot, the geographic location of the field plot's starting point (south-west corner) defines the domain of the plot (as land cover class).

> Note: *In forestry literature, point sampling is also used to describe a means of selecting trees for measurement using variable radius plots by use of an angle gauge (relasscope). Point sampling and variable plot radius tree assessment are different!* (Lund 1982)

The point sampling method is applied in Calc with the help of base unit weight. In this script, plot center (or reference point) gets full weight of one (1), and other sections will get zero weight. The point sampling method is also applied in the error script calculation.

2.7. Data processing chain for trees

Tree data consists of the following life forms:

- Trees (code 'T')
- Palms ('P)

These two types are separated using species code as criteria. However, tree **(T)** is the default life form in the input data. In R scripts, a new result variable '*tree$tree_life_form'* is created, and it can be used when reporting results in Saiku.

Living and dead standing trees can be reported separately because dead standing trees are coded with health code '4' in the database, as follows:

```
tree$tree_live_dead <- ifelse( tree$tree_health=='4', '2', '1')
```

The calculation chain for computing tree biomass and carbon estimates are presented in Figure 17.

[4] Point sampling is simply a method of sampling a geographical area by selecting points in it, more specifically by choosing points at random or systematically on a map, aerial photograph, or in the field (Lund 1982). Point sampling is also applied in the National Forest Inventory in Finland (Tomppo 2006).

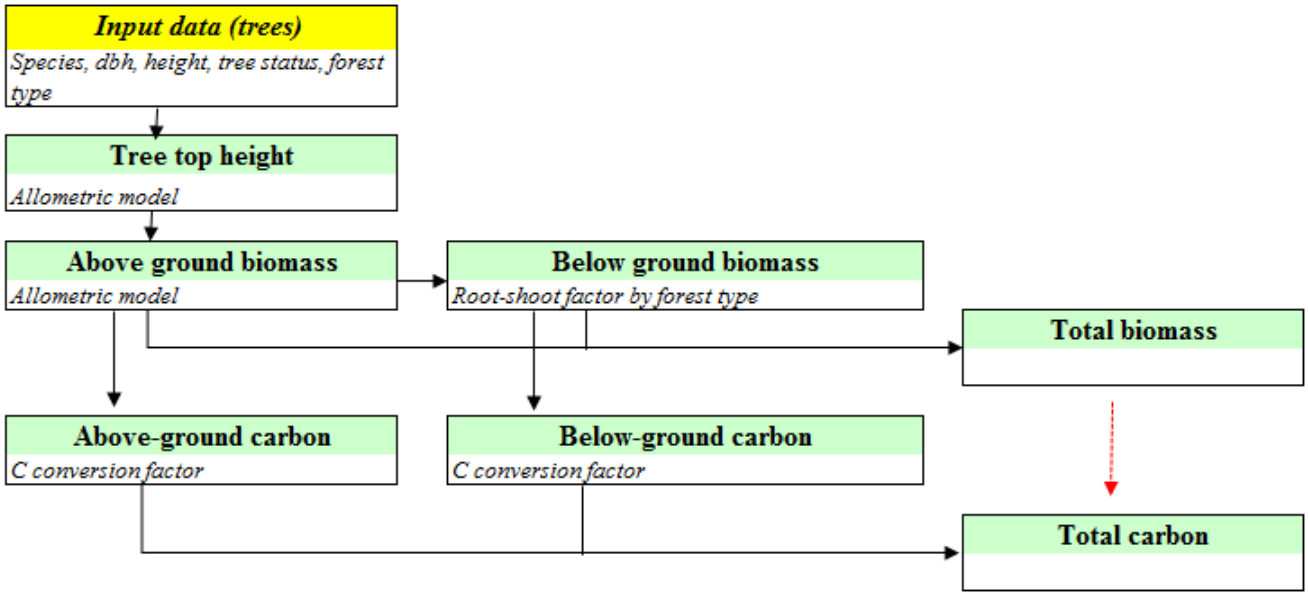


Figure 17. Tree biomass and carbon computing chain

2.8. Data processing chain for stumps

The chain for stump biomass and carbon estimates is presented in Figure 18.

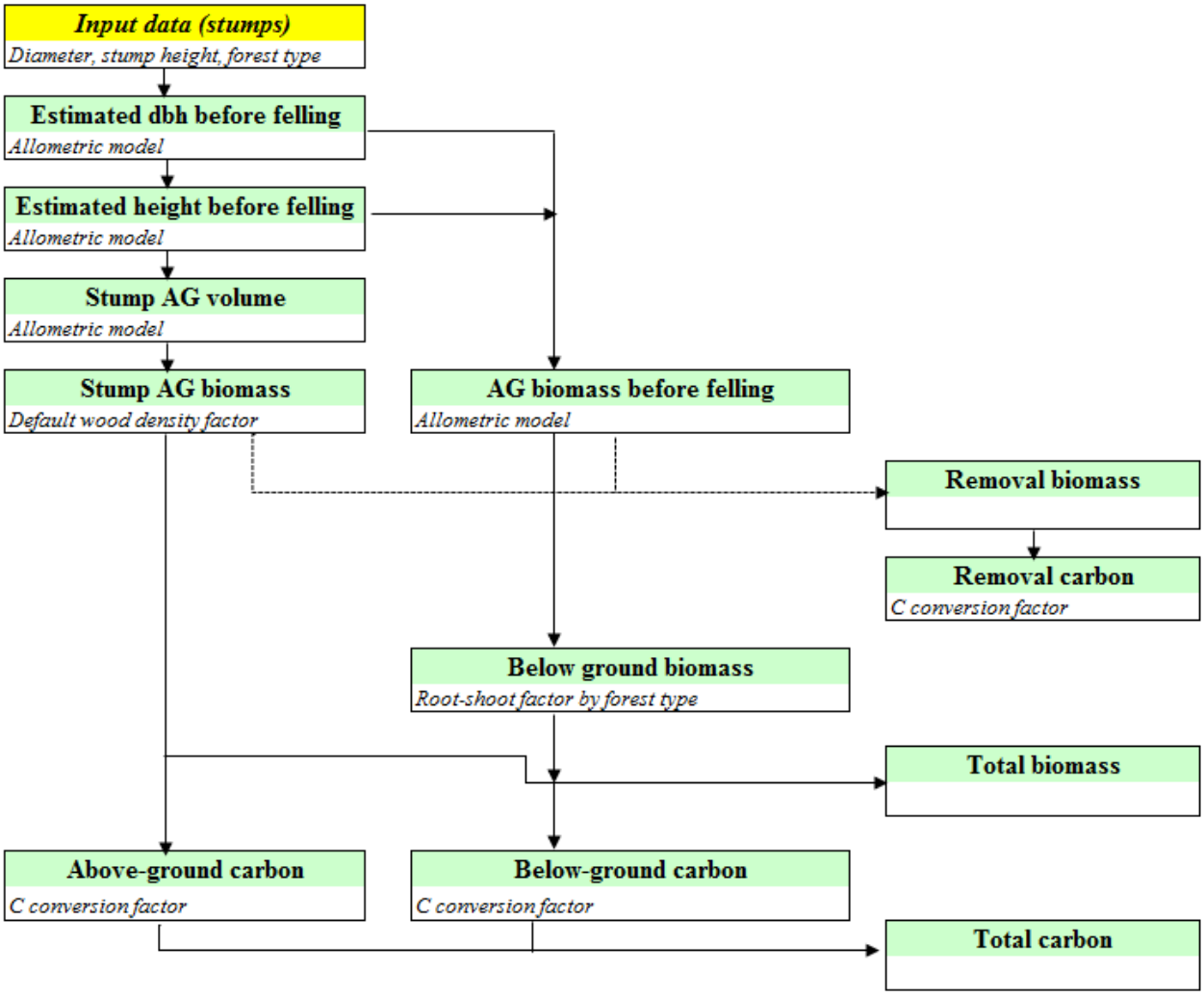
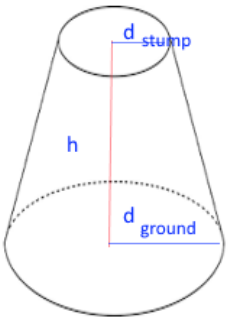


Figure 18. Stump biomass and carbon computing chain

In order to get estimate for the below-ground biomass of a stump, tree above-ground biomass **before felling** needs to be estimated. For this, an allometric model for estimating DBH is needed (see R script Module 3.4 in Chapter 4)

The stump above-ground volume (in m^3) is computed as conical frustum based on recorded stump's diameter, predicted diameter at the ground level, and stump height. Estimate for diameter at the ground level can be taken from stump DBH model by setting stump height to zero (0) (Module 3.2) as follows:

$$d_{ground} = dbh_{est} / (1 - 0.00173 * 130) = dbh_{est} / 0.77510 \quad \text{(Equation 1)}$$



$$Vol_{stump} = pi * (d_{stump}^2 + d_{ground}^2)/40000 * h \quad \text{(Equation 2)}$$

Stump's above-ground biomass (i.e. AG biomass remaining in the land) is computed with the help of default dry wood density and volume.

2.9. Data processing chain for fallen deadwood

For fallen deadwood (DW), volume using conical frustum model is computed first. The volume is then multiplied with the number of similar dead wood parts as recorded. Then dead wood biomass is computed with the help of default dry wood density.

The result calculation chain in OF Calc is presented in the following Figure.

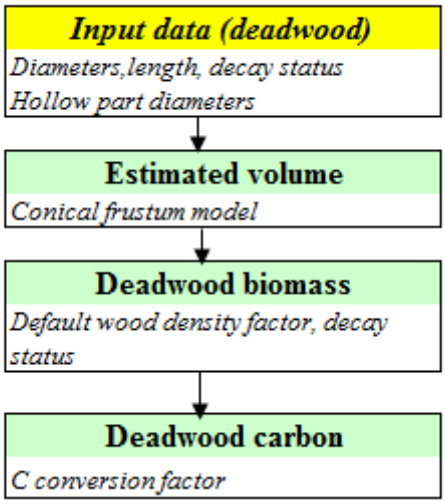


Figure 19. Deadwood biomass and carbon computing chain

Deadwood volume is computed first for the woody part, then separately for the possible hollow part of the deadwood. If there is one hollow part diameter > 0 cm, but another end's hollow diameter is zero (0), then the length of the hollow is assumed to be half of deadwood's length. The final deadwood volume is computed as follows:

Deadwood volume = StemVolume – HollowVolume

And as equation form deadwood volume is computed as follows:

$$Vol_{DW} = pi * (d_1^2 + d_2^2)/40000 * length_{DW} - pi * (d_{Hollow1}^2 + d_{Hollow2}^2)/40000 * length_{DW}$$

$$= \text{pi} /40000 * length_{DW} * ((d_1^2 + d_2^2) - (d_{Hollow1}^2 + d_{Hollow2}^2)) \quad \text{(Equation 3)}$$

And in case where $d_{Hollow2}$ =0, the equation is as follows

$$\text{Vol}_{DW} = \text{pi} * (d_1^2 + d_2^2)/40000 * length_{DW} - \text{pi} * d_{Hollow1}^2/40000 * 0.5 * length_{DW} \quad \text{(Equation 4)}$$

The deadwood volume is multiplied with the number of similar size DW parts. If the number of similar size parts is missing, it is assumed this is one (1).

Three decomposition classes are recorded for deadwood particles: solid, partially rotten and fully rotten. Because rotten wood is lighter than sound wood, the dry wood density of dead wood is scaled down using lower wood densities than for standing trees, as follows:

Solid: 90% * Default WD,
Partially rotten: 70% * Default WD,
Fully rotten: 50% * Default WD.

If the decomposition class is missing in the data, it is assumed that deadwood piece is solid.

2.10. Data processing chain for bamboo

The chain for bamboo biomass and carbon estimates is presented in Figure 15.

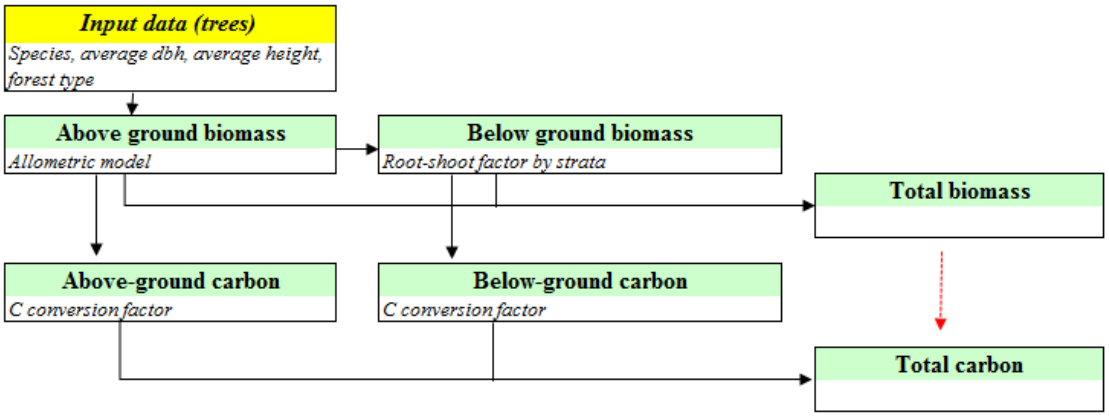


Figure 20. Bamboo biomass and carbon computing chain

The results for bamboo are computed first for a single (average) bamboo stem, and then by multiplied that result by the number of stems in the clump.

2.11. Parameters and allometric equations

2.11.1. About required parameters and equations

Before the actual data analysis stage there need to be an expert survey on available and applicable allometric models, parameters and conversion factors. The forest inventory project needs clear recommendations for the following cases:

- Default dry wood density factors for trees and palms, by genus or species. OF Calc workspace can use an imported auxiliary table which contains dry wood densities by species or/and genus.
- Tree bole volume and stem volume models;
- Tree height model(s), and rules for curve localization[5];
- Above-ground biomass model(s) for trees and palms;
- Root-to-shoot conversion factor(s);
- Carbon fraction conversion factor.

In addition, for calculation of stump below-ground biomass we need to estimate tree's dimension before felling. Therefore an equation estimating *dbh* at 1.3m as a function of stump diameter and stump height will be needed.

2.11.2. Tree height

Height models are needed for calculation of possibly missing tree heights and height for a tree before felling in case of a stump for its biomass estimation. Three different height models were tested here (see e.g. Mehtätalo *et al.* 2015):

1) Näslund (1937) model

$$h = 1.3 + \frac{dbh^2}{(a + b * dbh)^2} \quad \text{(Equation 5)}$$

2) Schumacher (1939) model[6]

$$h = 1.3 + a * e^{\frac{-b}{dbh}} \quad \text{(Equation 6)}$$

3) Curtis (1967) model

$$h = 1.3 + a * (\frac{dbh}{1 + dbh})^b \quad \text{(Equation 7)}$$

where h = estimated top height [m],
dbh = breast height diameter [cm],
a, b = parameters.

[5] R package named 'lmfor' is an applicable solution for tree diameter-height relationship modelling, see https://cran.r-project.org/web/packages/lmfor/index.html
[6] Schumacher model is also known as Michailoff model.

Based on sample tree data recorded from Uplands from 30 *Shorea* trees, the tree height models were fitted into the data (Figure 16).

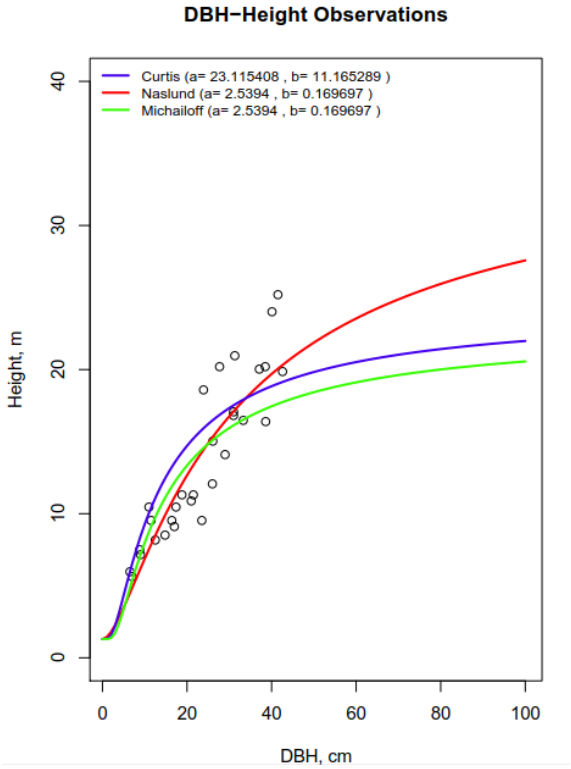


Figure 21. An example fitted height curves in a test data: Curtis (green), Näslund (red) and Schumacher (blue).

The model form presented by Naslund (Eq. 3) has been preliminary selected for predicting missing tree heights because it shows good fit especially with the big trees with DBH above 100 cm, when extrapolated. The parameters are estimated in R using library 'lmfor' and its nonlinear estimation techniques which can give unbiased estimators. The estimation is done by clusters, so that model parameters can vary between clusters.

In case of climbers (as liana), length of liana cannot be predicted. If liana has got a diameter but no length, then the average length of lianas in the same plot is computed and applied for missing cases. If the lianas' average length in the plot cannot be computed due to lack of observations, then lianas' average length in the cluster is applied.

In case of a missing palm height, the mean palm height in the cluster is applied.

2.11.3. Dbh before felling (of stump)

For stumps we need to estimate *dbh* before felling in order to estimate tree's above-ground biomass before felling. Because stump height varies in the data and there is no empirical data about relationship between *dbh* and stump diameter at different heights in the country yet, a model from Vietnam is applied[7]. The model is as follows:

$$dbh_{est} = d_{stump} + 0.00173 * (130 - h_{stump}) * d_{stump} \quad \text{(Equation 8)}$$

[7] The model is based on ~800 live tree data collected in FAO-Vietnam NFA Project in 2011-2015. The model developed by L. Vesa (2013). Unpublished.

where dbh_{est} = estimated *dbh* [cm],
d_{stump} = recorded stump diameter [cm],
h_{stump} = recorded stump height [m].

2.11.4. Above- and belowground biomass and carbon

The list of variables and parameters used in the biomass and carbon result calculations are presented in Table 6.

The default carbon fraction to convert biomass into carbon is 0.49 in the calculations (IPCC 2003, IPCC 2006)[8].

Table 8. Summary of key variables and parameters for volume, biomass and carbon calculations.

Variable	Belongs to entity level	Explanation	
stratum	cluster	Sampling intensity is equal inside stratum. Reporting area.	
land cover/use type	lvs (plot section)		
species	tree, saplings, bamboo, shrubs		
tree health	tree	Living or dead standing tree	
diameter *(dbh)*	tree	Recorded for all trees in the field	
bole (merchantable) height	tree		
tree top height (*H*)	tree	Recorded for all trees in the field. If missing computed with tree height model	
Parameter/Conversion factor	**Value**	**Source**	**Explanation**
Wood density (*WD*)	Default for all species: 0.5		Dry wood density
Root-to-shoot (*RS*)	Lookup table by strata	IPCC (2003, 2006)	Convert above-ground *(AG) biomass* into below-ground *(BG)* biomass
C fraction of dry matter	0.49	IPCC (2003, 2006)	Convert biomass into carbon

Calculation of tree above-ground biomass (AGB) for the final results is done using equation of Chave *et al.* (2014), as follows:

$$AGB = 0.0673 * (WD * dbh^2 * h)^{0.976} \quad \text{(Equation 9)}$$

where AGB = above-ground biomass [kg],
WD = dry wood density. The default value is 0.500 tons/m^3.

Calculation of palms' above-ground biomass is done using equation stem volume and default IPCC dry wood density factor (0.5), as follows:

$$AGB = WD * volume \quad \text{(Equation 10)}$$

The AGBs as well as other biomasses are converted into tons in the calculation.

[8] IPCC (2003), Chapter 3, Section 3.2.1.1.1.1., and in IPCC (2006), Chapter 6, Section 6.3.1.4. See *References*.

The below-ground biomass (BGB) is computed with the help of root-shoot (RS) factor which is different for the forest types taken from the IPCC guidelines:

Life form	Stratum	RS factor
Tree	Mangrove	0.49
Tree, palm, bamboo	Uplands, Wetlands	0.28
Shrub	all	0.40

The default carbon fraction to convert biomass into carbon is 0.49 (IPCC 2003, IPCC 2006)[9].

2.11.5. Biomass models for shrubs and climbers

Shrub biomass is computed with the following equation from China (Ali *et. al*, 2015):

$$AGB = \exp(-3.23 + 2.17 * \mathrm{LN}(dbh)) / 1000 \qquad \text{(Equation 11)}$$

Biomass for climbers (liana) is computed with the following equation from Malaysia (Addo-Fordjour & Rahmad, 2013):

$$AGB = 1.011 * 10^{(0.275 + 0.470*\mathrm{LOG10}(dbh) + 0.452 * \mathrm{LOG10}(length))} / 1000 \qquad \text{(Equation 12)}$$

Shrub below-ground biomass is computed similarly as for trees with the help of root-shoot factor, but climbers (lianas) do not get any BGB.

If a recorded climber has got diameter but its length is missing, the missing length is estimated as average of recorded climbers' length in the cluster.

2.11.6. Biomass model for bamboo

The applied above-ground biomass model for the bamboo is as follows[10]:

$$\mathrm{AGB} = 61.08613 * ((dbh_{avg} / 100)^{2} * \mathrm{height}_{avg})^{0.7126}) \ /1000 \qquad \text{(Equation 13)}$$

Where

AGB = above-ground biomass, in tons
dbh_{avg} = average diameter of the bamboo clump, cm
height_{avg} = average height of the bamboo clump, m

The below-ground-biomass and carbon variables are computed similarly as for trees.

Biomass and carbon estimates are converted into tons in the calculation.

[9] IPCC (2003), Chapter 3, Section 3.2.1.1.1.1., and in IPCC (2006), Chapter 6, Section 6.3.1.4. See *References*.
[10] Source: Nguyen Dinh Hung, FAO-Vietnam NFA Project 2014 and Forest Planning and Inventory Institute of Vietnam.

2.12. Variance

Variance, standard error (called as “absolute error” in Saiku), and relative standard error can be computed with the “Calc error script” (see Module 9) and reported using Saiku. Sampling error is (not yet) in the error scripts but it can be computed (in R or Excel) when standard error estimate is known.

Variance of the mean biomass (or other result variable of interest) on land estimate can be obtained with the following formula (Korhonen and Scott, 2016):

$$v(\bar{x}_d) = \frac{n}{n-1} \frac{\sum_{ij}^{n} (x_{id} - a_i \bar{x}_d)^2}{\left(\sum_{ij}^{n} a_i\right)^2} \quad \text{(Equation 14)}$$

Where x_{id}= mean of the biomass in the plot in domain of interest d.

$\bar{x}_d$= mean of the biomass in domain of interest d.

a_{ijk}= Because we assume that the entire plot area is the same as the condition as plot center, which means that a is always 1. If the condition is not measured due to being out of population or inaccessible, then then a is equal to 0.

n = number of clusters that fell (even partially) in the population

Variance for total biomass (of any category of interest):

$$v(X_d) = A^2 \cdot v(\bar{x}_d) \quad \text{(Equation 15)}$$

Where $\bar{x}_d$= mean of the biomass in domain of interest d

A= Area in domain of interest d

The standard error (s_d) of the mean biomass in domain is square root of the variance (Freese, 1962):

$$s_d = \sqrt{v(\bar{x}_d)} \quad \text{(Equation 16)}$$

The relative standard error (*rel.* s_d) is as follows:

$$relative \ \ s_d = \frac{100 * s_d}{\bar{x}_d} \quad \text{(Equation 17)}$$

In addition, when reliability estimates are exported from Saiku, the sampling error (SE_d) can be computed (e.g. in R or Excel) as follows:

$$SE_d = t * s_d \quad \text{(Equation 18)}$$

Where t = Student's t-value at 0.05 probability level

Note with Student's t-value: n = number of clusters in domain d

And the relative sampling error can be computed as follows:

$$relative \ SE_d = \frac{100 * SE_d}{\bar{x}_d} \quad \text{(Equation 19)}$$

3. Variable names, base unit and plot area scripts

3.1. About the R scripts and attribute names

All calculation modules for Calc are written using R language in RStudio editor. Every calculation module can contain just one new result variable that will be available later in Saiku reporting. The instructions for creating and editing calculation modules are given in the Calc User's Manual.

The entities created in Collect are imported within XML backup data into Calc database. Each entity used in R is read into a data frame by its name. For example, entity name 'plot' is called also as data frame 'plot' in R.

Attribute names are basically also remaining the same as given in Collect. However, if two entities have attribute with the same name, the latter (or lower level entity's) attribute name is changed in Calc. See two examples in the next table.

Entity	If attribute name in Collect is	Then attribute name in Calc is
cluster	accessibility	accessibility
plot	accessibility	plot_accessibility
tree	dbh	dbh
bamboo	dbh	bamboo_dbh

The situations described above can be avoided if the attributes have got unique names in OF Collect database.

3.2. Base unit weight

The base unit in this calculation is the plot section (i.e. entity '*lvs*'). This is the smallest homogeneous sampling unit in terms of land use/vegetation cover type. The weight script for base unit is as follows:

```
lvs$weight <- ifelse ( lvs$luvs_id == 'A', 1 , 0 );

lvs$weight[ lvs$plot_access != '0'] <- 0
```

3.3. Plot area

Plot area formulas represent the scripts to compute plot areas for each entity which need to be aggregated and reported in Saiku.

Entity	Plot area script

tree	`tree$plot_area <- with( tree,`
	`  ifelse( stratum=='3' & tree_dbh>=30, pi*20*20,`
	`  ifelse( stratum=='3' & tree_dbh>=10, pi*10*10,`
	`  ifelse( stratum=='3', pi*4*4,`
	`  ifelse( tree_dbh>=30, 30*50,`
	`  ifelse( tree_dbh>=15, 15*30, 10*10 ))))))`
	
	`# convert m2 -> ha`
	`tree$plot_area <- tree$plot_area / 10000`

sapling	`# in mangrove there are 2 subplots`
	`sapling$plot_area <- ifelse( sapling$stratum=='3', 2 * pi*1.4*1.4, 5*5 )`
	`sapling$plot_area <- sapling$plot_area / 10000`

seedling	`seedling$plot_area <- ifelse( seedling$stratum=='3', pi*1.4*1.4, 2*2 )` `seedling$plot_area <- seedling$plot_area / 10000`

shrub	`shrub$plot_area <- ifelse( shrub$stratum=='3', 2*pi*1.4*1.4, 5*5 )` `shrub$plot_area <- shrub$plot_area / 10000`
small_shrub	`small_shrub$plot_area <- ifelse(small_shrub$stratum=='3',pi*1.4*1.4,2*2)` `small_shrub$plot_area <- small_shrub$plot_area / 10000`

stump	`stump$plot_area <- with( stump,`
	`  ifelse(stratum=='3', pi*10*10, 15*30 ))`
	
	`stump$plot_area <- stump$plot_area / 10000`

fallen_deadwood	`fallen_deadwood$plot_area <- with( fallen_deadwood,`
	`  ifelse(stratum=='3', pi*10*10, 15*30 ))`
	
	`fallen_deadwood$plot_area <- fallen_deadwood$plot_area / 10000`

bamboo	`bamboo$plot_area <- 15*30/10000`

4. Calculation scripts

4.1. List of modules

The list of aggregating (green boxes in Fig. 15) and calculation modules (blue boxes) is presented in Figure 22 and more detailed in the following chapters.

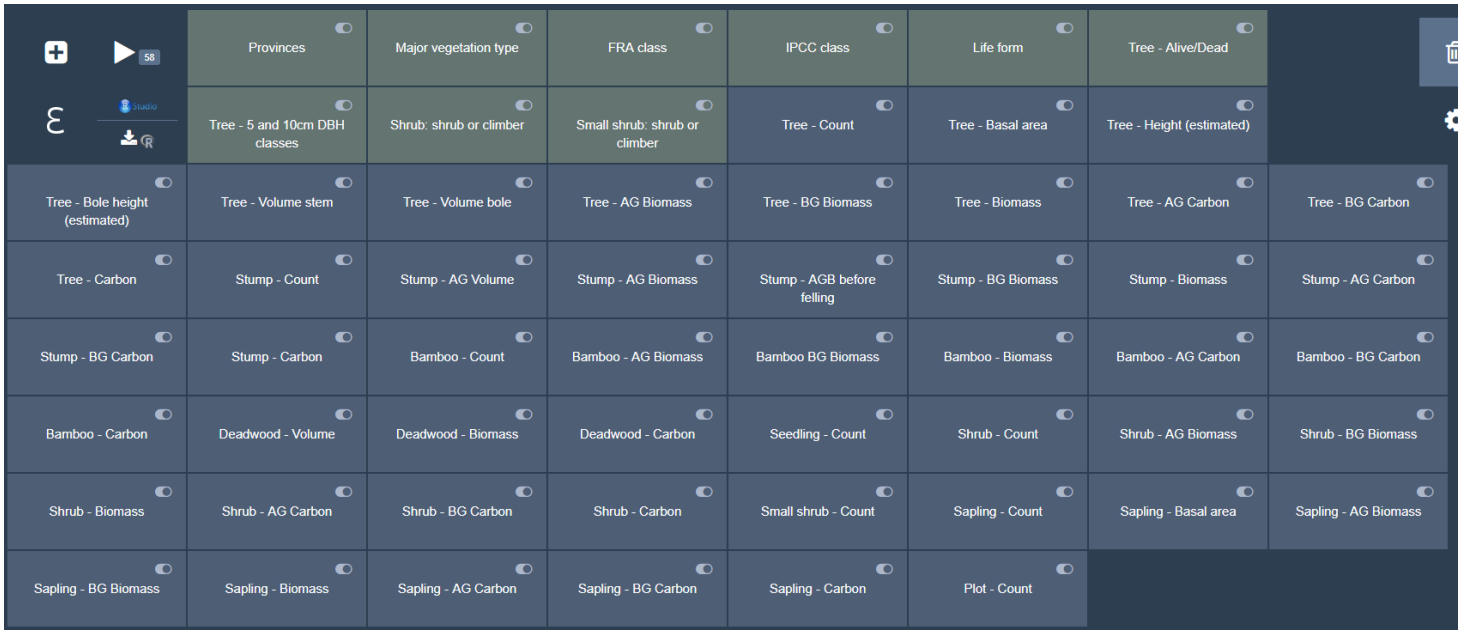


Figure 22. OF Calc calculation modules.

4.2. Common script

Common script in '004-common-R' are run just after reading the input data. Variables and functions written in this module can be called in any other modules.

#	**0.1**
Caption	
Type	Common scripts
Entity	
Purpose	
Code	

```
######################################################################
# Required R libraries
library('lmfor')
library('ggplot2')
library('gridExtra')
library('dplyr')

######################################################################
# Default values
# tree form factor
FF <- 0.65
# wood density
WD <- 0.500
# default root-shoot factor (see also function below)
RS <- 0.28
RS shrub <- 0.40
# carbon fraction, IPCC
CF <- 0.49

######################################################################
# Functions

# top height
paraA <- 2.5394
paraB <- 0.169697
h_model <- function(x) {1.3 + x^2/(paraA + paraB * x )^2}

# Root-shoot factor by strata
getRS_factor <- function(data) {
  data$RS <- with(data,
                  ifelse(stratum =='3', 0.49, # Mangrove
                         0.28)) # Uplands, Wetlands
  return(data)
}

############################################################
# Output folder for additional results and graphs
FolderResult <- "../testdata-output"

if (file.exists(FolderResult)){
  print("Output folder: Exists")
} else {
  print("Output folder: Created")
  dir.create(file.path('../', 'testdata-output'), showWarnings = FALSE)
}

cat( normalizePath(FolderResult) )
FolderResult <- paste(FolderResult, "/", sep="")
```

4.3. Categorical variables

Categorical variables are used to aggregate the data (see green module boxes in Fig. 7). These variables can be used in the R scripts, and they become visible in Saiku reporting window.

#	1.1
Caption	Province
Type	Category
Entity	lvs
Purpose	Province for Saiku
Code	`# '-1' NA, '01' Banteay Meanchey, '02' Battambang, '03' Kampong Cham, '04' Kampong Chhnang, '05' Kampong Speu, '06' Kampong Thom, '07' Kampot, '08' Kandal, '09' Koh Kong, '10' Kratie, '11' Mondul Kiri, '12' Phnom Penh, '13' Preah Vihear, '14' Prey Veng, '15' Pursat, '16' Ratanak Kiri, '17' Siem Reap, '18' Preah Sihanouk, '19' Stung Treng, '20' Svay Rieng, '21' Takeo, '22' Otdar Meanchey, '23' Kep, '24' Pailin`  `lvs$lvs province <- lvs$province`  `# if NA, set '-1'` `lvs$lvs_province[is.na(lvs$lvs_province)] <- '-1'`

#	1.2
Caption	Major LUVS Class
Type	Category
Entity	lvs
Purpose	

Code	`# '-1' NA, '1' Evergreen forest, '2' Semi-evergreen forest, '3' Deciduous forest,`
	`# '4' Flooded forest, '5' Mangrove forest, '6' Forest plantation, '7' Bamboo,`
	`# '8' Shrubland, '9' Grassland, Abandoned field, Marsh, '10' Built-up/Barren Areas,`
	`# '11' Cropland/paddy, '12' Water`
	
	`lvs$lvs_major_class <- with( lvs,`
	` ifelse( land_cover == '0' |  is.na(land_cover), '-1',`
	`  ifelse( land_cover == '1' , '1',`
	`   ifelse( land_cover == '3' , '2',`
	`    ifelse( land_cover == '4' , '3',`
	`     ifelse( land_cover == '5' , '4',`
	`      ifelse( land_cover == '6' | land_cover == '7', '5',`
	`       ifelse( as.integer(land_cover) >= 8 & as.integer(land_cover) <= 11, '6',`
	`        ifelse( land_cover == '12', '7',`
	`         ifelse( land_cover == '21' | land_cover == '22', '8',`
	`          ifelse( as.integer(land_cover) >= 31 & as.integer(land_cover) <= 34, '9',`
	`           ifelse( as.integer(land_cover) >= 35 & as.integer(land_cover) <= 39, '10',`
	`            ifelse( land_cover =='23' | land_cover == '40' | land_cover == '41','11',`
	`             ifelse( land_cover == '51' | land_cover == '52', '12', '-1'`
	`))))))))))))`

#	1.3
Caption	FAO-FRA Class
Type	Category
Entity	lvs
Purpose	

Code	`# '-1' NA, '1' Forest, '2' Other Wooded Land, '3' Other Land, '4' Water`
	
	`lvs$lvs_fra_class <- with( lvs,`
	` ifelse( lvs_major_class == '-1', '-1',`
	`  ifelse( as.integer(lvs_major_class) <= 7, '1',`
	`   ifelse( lvs_major_class == '8', '2',`
	`    ifelse( as.integer(lvs_major_class) <= 11, '3',`
	`     ifelse( lvs_major_class == '12', '4', '-1' ))))))`

#	1.4
Caption	IPCC Class
Type	Category
Entity	lvs
Purpose	
Code	`# '-1' NA, '1' Forest land, '2' Grassland, '3' Cropland, '4' Settlement,` `# '5' Other land, '6' Wetland` `lvs$lvs_ipcc_class <- with( lvs,` ` ifelse(land_cover == '0'	is.na(land_cover), '-1',` ` ifelse(as.integer(land_cover) <= 9	land_cover == '12', '1', # forest` ` ifelse(land cover == '10'	land cover == '11', '3', # cropland` ` ifelse(land_cover == '10'	land_cover == '12'	land_cover == '23', '3', #` `cropland` ` ifelse(land_cover == '40'	land_cover == '41', '3', # cropland` ` ifelse(land_cover == '21'	land_cover == '22', '2', # grassland` ` ifelse(land_cover == '31'	land_cover == '32'	land_cover == '33', '2', #` `grassland` ` ifelse(land_cover == '36'	land_cover == '39', '4', # settlement` ` ifelse(land cover == '35'	land cover == '37'	land cover == '38', '5', #` `other land` ` ifelse(land_cover == '34' | land_cover == '51' | land_cover == '52', '6',` `'-1' # wetland` `))))))))))`

#	1.5
Caption	Tree - Life form
Type	Category
Entity	tree
Purpose	Life form code

Code	`# '-1' NA, 'T' Tree, 'P' Palm`
	
	`# list of Palm species codes`
	`palm_list <-`
	`c('PASPR','PADPR','PCTST','PCTNS','PCSLN','PCTEN','PLPAA','PLCHR','PNCAK','POSTO',`
	`'PPDPE','PPCKT','PPDTS','PPSSN','PPSKH','PRSBR')`
	
	`tree$tree_life_form <-`
	`  ifelse(tree$tree_species_code %in% palm_list, 'P', 'T')`

#	1.6
Caption	Tree – Live / Dead
Type	Category
Entity	tree
Purpose	To group tree data into living and dead trees

Code	`# '-1' NA, '1' Living, '2' Dead`
	`tree$tree_live_dead <- ifelse( tree$tree_health=='4', '2', '1')`
	
	`tree$tree_live_dead[ is.na(tree$tree_live_dead) ] <- '1'`

#	1.7
Caption	Tree – DBH class (10cm)
Type	Category
Entity	tree
Purpose	Tree DBH classes in 10 cm interval

Code	`# '-1' NA, '0' 5-9.9 cm, '1' 10-19.9 cm, '2' 20-29.9 cm, '3' 30-39.9 cm, '4' 40-49.9 cm, '5' 50+ cm`
	`tree$dbh_class10 <- trunc((tree$tree_dbh)/10 ,0)`
	`tree$dbh class10 <- ifelse( tree$dbh class10 > 5, 5, tree$dbh class10)`
	
	`tree$dbh_10 <- as.character(tree$dbh_class10)`

#	1.8
Caption	Shrub - Shrub / Climber
Type	Category
Entity	shrub
Purpose	To group shrub data into shrub and climbers. Required by Saiku.
Code	`# '-1' NA, '1' Shrub, '2' Climber` `shrub$shrub_shrub_liana <- ifelse( shrub$shrub_is_liana, '2', '1')`

#	1.9
Caption	Small shrub - Shrub/Climber
Type	Category
Entity	small_shrub
Purpose	To group shrub data into shrub and climbers. Required by Saiku.

Code	`# '-1' NA, '1' Shrub, '2' Climber`
	`small_shrub$smallshrub_liana <-`
	`  ifelse( small_shrub$smallshrub_is_liana, '2', '1')`

4.4. R scripts for entities

4.4.1. Tree

Trees and palms are recorded in the same field form and same table, and they are treated as one entity 'tree' but often separated in calculation stages with the help of categorical variable ('*tree$tree_life_form*'), see Module 1.4.

#	2.1
Caption	Tree – Count
Type	R Script
Entity	Tree
Purpose	Number of trees

Code	`# select only trees with DBH given into the analysis`
	`tree <- subset(tree, !is.na(tree_dbh) & tree_dbh>0 )`
	
	`tree$tree_count <- 1`

#	2.2
Caption	Tree - Basal area
Type	R Script
Entity	tree
Purpose	Basal area of tree (m^2)
Code	`tree$tree_basal_area <- pi * (0.01 * tree$tree_dbh/2)^2`

#	2.3
Caption	Tree - Height
Type	R Script
Entity	tree
Purpose	Estimated height of tree/plant (m).
Code	

```
#################################################################
file_pdf_graphs <- paste(FolderResult, "MyCountry_",  sep="")
# create unique temporary plot ID as text variable
tree$cluster_plot_id <- paste( tree$cluster_no , tree$plot_no , sep = '' )

# if negative tree height for some reason, set to NA
tree$tree_top_height[ tree$tree_top_height < 0 ] <- NA

# temporary variable for analysis
tree$H <- tree$tree_top_height

# SELECT ONLY trees where total h >= 1.35 and <50
tree$H[tree$H < 1.35 | tree$H >= 50] <- NA

# colors for graph, grouped by clusters
cl <- data.frame(unique(tree$cluster_no))
names(cl)     <- 'cluster_no'
cl$color_id   <- row(cl)[,1]
tree <- sqldf("SELECT * FROM tree LEFT JOIN cl USING (cluster_no)")
rm(cl)

# select only trees into tree2
tree2      <- subset( tree, tree_life_form =='T' )
tree_palms <- subset( tree, tree_life_form !='T' )

# run analysis if at least 20 observations
  if (nrow(tree2) >= 20 )  {
    tree2$temp <- "1"

    # graphic output file

pdf(paste(file_pdf_graphs,"fitted_height_model_residual_trees.pdf",sep=""),width=5,he
ight=7,pointsize = 12)
    par(mfcol=c(3,1))

    # im1<- NULL;im2<-NULL; im3<- NULL;

    # three options for lmfor:
    # 1. models calibrated for plots
    im1 <- ImputeHeights(tree2$tree_dbh,tree2$H,tree2$cluster_plot_id,
                         modelName = "naslund",nranp = 2, varf = 1,
                         addResidual = FALSE, makeplot=TRUE, level = 1,
                         start=NA, bh=1.3, control=list(), random=NA)

    # 2. models calibrated for clusters
    im2 <- ImputeHeights(tree2$tree_dbh,tree2$H,tree2$cluster_no,
                         modelName = "naslund",nranp = 2, varf = 1,
                         addResidual = FALSE, makeplot=TRUE, level = 1,
                         start=NA, bh=1.3, control=list(), random=NA)

    # 3. fixed part of the model only for the whole data
    im3 <- ImputeHeights(tree2$tree_dbh,tree2$H,tree2$temp, modelName = "naslund",
level=0, makeplot=TRUE)

    dev.off()

    # Add imputed heights into the data (hpred) and a column indicating the type of
prediction (hpredType):
    # 0: tree has been measured, hpred includes the measured height
    # 1: tree height has been predicted either using a plot-level or cluster lever
random effect
    # 2: tree height has been predicted using the fixed part of the model
```

```
    # selected method 1 (plot level calibration)
    hpred <- im1$h
    hpred[im1$predType==2] <- im2$h[im1$predType==2]

    tree2 <- cbind(tree2, tree_height_calc = hpred) # , est_height_prediction_type =
im2$predType)
    tree2$tree_height_calc <- tree2$hpred


    heigth_model_fixed   <- c(im2$model$coefficients$fixed[[1]],
im2$model$coefficients$fixed[[2]])

    # get rounded next 10 cm/m limit for graphs
    maxD <- 10 * trunc(max(tree2$tree_dbh)/10) + 10
    maxH <- 50


pdf(paste(file_pdf_graphs,"fitted_height_models_trees.pdf",sep=""),width=5,height=7,p
ointsize = 11)
    par(mfcol=c(2,2))

    plot(tree2$tree_dbh[!is.na(tree2$H)],
         tree2$H[!is.na(tree2$H)],
         col=tree2$color_id[!is.na(tree2$H)],
         main="Observations", xlab="DBH, cm", ylab="Height, m",
         xlim=c(1,maxD),ylim=c(0,maxH))

    plot(tree2$tree_dbh[im1$imputed],
         im1$h[im1$imputed],
         col=tree2$color_id[im1$imputed],
         main="Naslund: Fixed + Plot", xlab="DBH, cm", ylab="Height, m",
         xlim=c(1,maxD),ylim=c(0,maxH))

    plot(tree2$tree_dbh[im2$imputed],
         im2$h[im2$imputed],
         col=tree2$color_id[im2$imputed],
         main="Naslund: Fixed + Cluster", xlab="DBH, cm", ylab="Height, m",
         xlim=c(1,maxD),ylim=c(0,maxH))

    plot(tree2$tree_dbh[im3$imputed],
         im3$h[im3$imputed],
         col=1,
         main="Naslund: Fixed", xlab="DBH, cm", ylab="Height, m",
         xlim=c(1,maxD),ylim=c(0,maxH))

    dev.off()

    pdf(paste(file_pdf_graphs,"3_common_height_models_trees.pdf",
sep=""),width=5,height=7,pointsize = 11)

    theta1 <- startHDnaslund( tree2$tree_dbh, tree2$H )
    theta2 <- startHDcurtis(  tree2$tree_dbh, tree2$H )
    theta3 <- startHDmichailoff( tree2$tree_dbh, tree2$H )

    plot(tree2$tree_dbh,tree2$H,

         main=paste("DBH-Height Observations -","trees"), xlab="DBH, cm",
ylab="Height, m",
         xlim=c(1, maxD),ylim=c(0, maxH))

    d<-seq(0,maxD)
    plot_colors <- c( "blue", "red", "green")

    lines(d, HDnaslund(d,    a=theta1[1], b=theta1[2]), col=plot_colors[1], lwd=2)
    lines(d, HDcurtis(d,     a=theta2[1], b=theta2[2]), col=plot_colors[2], lwd=2)
    lines(d, HDmichailoff(d, a=theta3[1], b=theta3[2]), col=plot_colors[3], lwd=2)

    tr <- array()
    tr[1] <- paste("Naslund (a=",   round(theta1[1],6),", b=",round(theta1[2],6),")")
    tr[2] <- paste("Curtis (a=",    round(theta2[1],6),", b=",round(theta2[2],6),")")
```

```
    tr[3] <- paste("Michailoff (a=",round(theta3[1],6),", b=",round(theta3[2],6),")")

    legend("topleft", tr, cex=0.8, col=plot_colors, lty=1, lwd=2, bty="n")

    dev.off()
    tree2$temp <- NULL

  }

# palms
tree_palms$tree_height_calc <- tree_palms$tree_top_height
# for missing cases, get cluster average palm height
plotH      <- sqldf(paste("SELECT cluster no, AVG(tree top height) AS meanH FROM
tree_palms WHERE tree_top_height>1.3 GROUP BY cluster_no ",sep="" ))
tree_palms <- sqldf("SELECT * FROM tree_palms LEFT JOIN plotH USING (cluster_no)")
tree_palms$tree_height_calc[ (is.na(tree_palms$tree_height_calc) |
tree_palms$tree_height_calc==0) & !is.na(tree_palms$meanH)] <- tree_palms$meanH[
(is.na(tree_palms$tree_est_height) | tree_palms$tree_height_calc==0) &
!is.na(tree_palms$meanH)]
tree_palms$meanH <- NULL
tree_palms$tree_height_calc[ is.na(tree_palms$tree_height_calc) ] <-
mean(tree_palms$tree_top_height, na.rm=TRUE)
rm(plotH)


tree <- rbind(tree2,tree_palms)
tree$H        <- NULL
tree$color_id <- NULL
tree$cluster_plot_id <- NULL

rm(tree2)

# use recorded height always when available
tree$tree_height_calc <- ifelse( !is.na(tree$tree_top_height) & tree$tree_top_height
> 1.3, tree$tree_top_height, tree$tree_height_calc)

tree$tree_height_calc[ is.na(tree$tree_height_calc)] <- 0

tree$H <- NULL
```

#	**2.3b**
Caption	Tree – Height bole (NOTE: Bole height should be always recorded, this module is temporary here)
Type	R Script
Entity	tree
Purpose	To estimate missing tree bole height (m)

Code	`tree$tree_bole_height <- ifelse( is.na(tree$tree_bole_height), -1,`
	`  ifelse(tree$tree_bole_height > tree$tree_height_calc, tree$tree_height_calc,`
	`tree$tree_bole_height))`
	
	`# get subset of data for model`
	`tree_model_data <- tree %>% filter(tree_bole_height>1.3 & tree_top_height>1.3) %>%`
	`select(tree_top_height, tree_bole_height)`
	`tree_model_data <- tree_model_data %>% filter(tree_bole_height/tree_top_height >0.5)`
	
	`# linear model through origo`
	`linearMod    <- lm(tree_bole_height ~ 0 + tree_top_height, data=tree_model_data)`
	`modelSummary <- summary(linearMod)  # capture model summary as an object`
	`modelCoeffs <- modelSummary$coefficients  # model coefficients`
	`rm(tree_model_data)`
	
	`tree$tree_bole_height_calc <- ifelse(tree$tree_bole_height == -1,`
	`      modelCoeffs[[1]] * tree$tree_height_calc, tree$tree_bole_height)`

#	2.4
Caption	Tree – Stem volume
Type	R Script
Entity	tree
Purpose	Tree stem volume (m^3)

Code	`tree$tree_volume_stem <- 0`
	`tree$tree_volume_stem <- with( tree,`
	`  ifelse( tree_life_form=='P', pi* tree_dbh * tree_dbh * tree_height_calc/40000,`
	`  FF * pi* tree_dbh * tree_dbh * tree_height_calc/40000))`

#	2.5
Caption	Tree - Bole volume
Type	R Script
Entity	tree
Purpose	Bole volume (m^3)

Code	`tree$tree volume bole <- 0.80 * tree$tree basal area * tree$tree bole height calc`
	
	`# if missing (NA), set to 0`
	`tree$tree_volume_bole[is.na(tree$tree_volume_bole)] <- 0`
	
	`# if more than stem volume, set to same as total volume`
	`tree$tree_volume_bole <- ifelse(tree$tree_volume_bole > tree$tree_volume_stem,`
	`tree$tree_volume_stem,tree$tree_volume_bole)`

#	2.6
Caption	Tree - AG biomass
Type	R Script
Entity	tree
Purpose	Above-ground biomass (tons)

Code	`# AG biomass computed for Trees and Palms`
	`# trees_ Model by Chave (2014)`
	
	`tree$tree_biomass_ag <- with( tree,`
	`  ifelse( tree_life_form=='P', tree_volume_stem * WD,`
	`  0.0673*(( WD * tree_dbh^2 * tree_height_calc)^0.976 /1000)`
	`  ))`

#	2.7
Caption	Tree - BG biomass
Type	R Script
Entity	tree
Purpose	Below-ground biomass (tons)
Code	`tree <- getRS_factor( tree )` `tree$tree_biomass_bg <- tree$RS * tree$tree_biomass_ag`

#	2.8
Caption	Tree - Biomass
Type	R Script
Entity	tree
Purpose	Total tree biomass (tons)
Code	`tree$tree_biomass_total <- tree$tree_biomass_bg + tree$tree_biomass_ag`

#	2.9
Caption	Tree - AG carbon
Type	R Script
Entity	tree
Purpose	Above-ground carbon (tons)
Code	`tree$tree_carbon_ag <- CF * tree$tree_biomass_ag`

#	2.10
Caption	Tree - BG carbon
Type	R Script
Entity	tree
Purpose	Below-ground carbon (tons)
Code	`tree$tree_carbon_bg <- CF * tree$tree_biomass_bg`

#	2.11
Caption	Tree – Carbon
Type	R Script
Entity	tree
Purpose	Total carbon (tons)
Code	`tree$tree_carbon_total <- tree$tree_carbon_bg + tree$tree_carbon_ag`

4.4.2. Stump

#	3.1
Caption	Stump– Count
Type	R Script
Entity	stump
Purpose	

Code	`# stumps without diameter are dropped out from analysis`
	`stump <- subset( stump, !is.na(stump_diameter) )`
	
	`stump$stump_count <- 1`

#	3.2
Caption	Stump – Volume (remaining)
Type	R Script
Entity	stump
Purpose	Estimated stump AG volume

Code	`# MODEL TO ESTIMATE DBH for tree before felling`
	`# Model using Vietnam (Bac Giang & Bac Kan combined) data, by L. Vesa 16.6.2014`
	
	`stump$stump_dbh <- stump$stump_diameter - 0.00173 *(130 -`
	`stump$stump_height)*stump$stump_diameter`
	
	`# diameter at the ground level using the as reversed, height is set to zero`
	`stump$stump_d0 <- stump$stump_dbh/( 1 - 0.00173 * 130 )`
	
	`# if missing stump height, use 30 cm`
	`stump$stump height[is.na(stump$stump height)] <- 30`
	
	`stump$stump_volume <- 0`
	`stump$stump_volume <- with( stump,`
	`  ( (pi*(stump_d0/200)^2 + pi*(stump_diameter/200)^2 ) / 2 ) * (stump_height/100))`

#	3.3
Caption	Stump – AG biomass (remaining)
Type	R Script
Entity	stump
Purpose	Estimated stump AG biomass
Code	`# computed in tons` `stump$stump_biomass_ag <-  stump$stump_volume * WD`

#	3.4
Caption	Stump - AGB before felling
Type	R Script
Entity	stump
Purpose	Above-ground biomass before felling (tons)

Code	`# MODEL TO ESTIMATE DBH for tree before felling`
	`# Model by L. Vesa 16.6.2014`
	
	`# if missing stump height, set 30 cm`
	`stump$stump height[is.na(stump$stump height)] <- 30`
	
	`stump$stump_dbh <- stump$stump_diameter - 0.00173 *(130 -`
	`stump$stump_height)*stump$stump_diameter`
	
	`# height before felling`
	`stump$stump_height_tree <- h_model( stump$stump_dbh )`
	
	`# using AG biomass model by Chave (2014)`
	`stump$stump_biomass_ag_prefelling <- 0.0673*( WD * stump$stump_dbh^2 *`

	stump$stump_height_tree)^0.976 # convert to tons stump$stump_biomass_ag_prefelling <- stump$stump_biomass_ag_prefelling / 1000

#	3.5
Caption	Stump – BG biomass
Type	R Script
Entity	Stump
Purpose	Below-ground biomass
Code	stump <- getRS_factor(stump) stump$stump_biomass_bg <- stump$RS * stump$stump_biomass_ag_prefelling

#	3.6
Caption	Stump – Biomass (remaining)
Type	R Script
Entity	Stump
Purpose	biomass tons)
Code	stump$stump_biomass_total <- stump$stump_biomass_bg + stump$stump_biomass_ag

#	3.7
Caption	Stump – AG carbon (remaining)
Type	R Script
Entity	Stump
Purpose	AG carbon (tons)
Code	stump$stump_carbon_ag <- CF * stump$stump_biomass_ag

#	3.8
Caption	Stump – BG carbon
Type	R Script
Entity	Stump
Purpose	BG carbon(tons)
Code	stump$stump_carbon_bg <- CF * stump$stump_biomass_bg

#	3.9
Caption	Stump – Carbon (remaining)
Type	R Script
Entity	stump
Purpose	Total carbon (tons)
Code	stump$stump_carbon_total <- stump$stump_carbon_bg + stump$stump_carbon_ag

#	3.10
Caption	Stump - AGB removal
Type	R Script
Entity	Stump
Purpose	Above-ground biomass removal due to felling (tons)
Code	stump$stump_biomass_removal <- stump$stump_biomass_ag_prefelling - stump$stump_biomass_ag

4.4.3. Bamboo

#	4.1
Caption	Bamboo – Count
Type	R Script
Entity	bamboo
Purpose	

Code	`# select only bamboos with average diameter given`
	`bamboo <- subset(bamboo, !is.na(bamboo$bamboo_avg_diameter))`
	
	`# if missing number of bamboo, set to 1`
	`bamboo$bamboo_stems_total[is.na(bamboo$bamboo_stems_total)] <- 1`
	`bamboo$bamboo_stems_total[bamboo$bamboo_stems_total==0 &`
	`bamboo$bamboo_avg_diameter>0] <- 1`
	
	`bamboo$bamboo_count <- bamboo$bamboo_stems_total`

#	4.2
Caption	Bamboo – AG biomass
Type	R Script
Entity	bamboo
Purpose	

Code	`# Vietnamese bamboo biomass model`
	`bamboo$bamboo_biomass_ag <- 61.08613*(((bamboo$bamboo_avg_diameter / 100)^2 *`
	`bamboo$bamboo_avg_height)^0.7126)`
	`  # convert to tons`
	`bamboo$bamboo_biomass_ag <- bamboo$bamboo_biomass_ag /1000`
	
	`bamboo$bamboo_biomass_ag <-bamboo$bamboo_biomass_ag * bamboo$bamboo_count`

#	4.3
Caption	Bamboo – BG biomass
Type	R Script
Entity	bamboo
Purpose	

Code	`bamboo <- getRS_factor( bamboo )`
	
	`bamboo$bamboo_biomass_bg <- bamboo$RS * bamboo$bamboo_biomass_ag`

#	4.4
Caption	Bamboo – Biomass
Type	R Script
Entity	bamboo
Purpose	
Code	`bamboo$bamboo_biomass_total <- bamboo$bamboo_biomass_bg + bamboo$bamboo_biomass_ag`

#	4.5
Caption	Bamboo – AG carbon
Type	R Script
Entity	bamboo
Purpose	
Code	`bamboo$bamboo_carbon_ag <- CF * bamboo$bamboo_biomass_ag`

#	4.6
Caption	Bamboo – BG carbon
Type	R Script
Entity	bamboo
Purpose	
Code	`bamboo$bamboo_carbon_bg <- CF * bamboo$bamboo_biomass_bg`

#	4.7
Caption	Bamboo – Carbon
Type	R Script
Entity	bamboo
Purpose	
Code	`bamboo$bamboo_carbon_total <- bamboo$bamboo_bg_carbon + bamboo$bamboo_carbon_ag`

4.4.4. Lying dead wood

#	5.1
Caption	Dead wood- Volume
Type	R Script
Entity	fallen_deadwood
Purpose	

Code	`fallen deadwood$dw volume <- with( deadwood,((pi*(dw diameter1/200)^2 + pi*(dw_diameter2/200)^2 ) /2 )* dw_length )`
	
	`# length of hollow part`
	`fallen_deadwood$dw_length_hollow <- with( deadwood,`
	`  ifelse( dw hollow1==0 & dw hollow2>0 & dw diameter1>0 & dw diameter2>0, 0.5*dw_length,`
	`    ifelse( dw_hollow1>0 & dw_hollow2==0 & dw_diameter1>0 & dw_diameter2>0, 0.5*dw_length, dw_length )))`
	
	`fallen_deadwood$dw_volume_hollow <- with( deadwood, ((pi*(dw_hollow1/200)^2 + pi*(dw_hollow2/200)^2 ) /2 )* dw_length_hollow )`
	
	`fallen_deadwood$dw_volume <- fallen_deadwood$dw_volume - fallen_deadwood$dw_volume_hollow`
	`fallen_deadwood$dw_volume_hollow <- NULL`
	`fallen_deadwood$dw_length_hollow <- NULL`
	
	`# result must be multiplied with number of similar dw particles`
	`fallen_deadwood$dw_volume <- fallen_deadwood$dw_volume * fallen_deadwood$dw_parts`

#	5.2
Caption	Dead wood- Biomass
Type	R Script
Entity	fallen_deadwood
Purpose	

Code	`# if missing decay class, it is assumed to be solid (1)`
	
	`fallen deadwood$decay factor <- ifelse(fallen deadwood$dw decay=='3', 0.5,`
	`  ifelse( fallen_deadwood$dw_decay=='2', 0.7, 0.9))`
	
	`fallen_deadwood$dw_biomass <- WD * fallen_deadwood$dw_volume * fallen_deadwood$decay_factor`

#	5.3
Caption	Dead wood- Carbon
Type	R Script
Entity	fallen_deadwood
Purpose	
Code	`fallen_deadwood$dw_carbon <- CF * fallen_deadwood$dw_biomass`

4.4.5. Seedlings

#	6.1
Caption	Seedlings – Count
Type	R Script
Entity	seedling
Purpose	Total number of seedlings

Code	`# remove blank entries with no data. Must at least species code or count given`
	`seedling <- subset( seedling, !is.na(seedling_species_code) | seedling_count>0 )`
	
	`# if missing count, add 1`
	`seedling$seedlings_count <- ifelse( seedling$seedling_count>0,`
	`  seedling$seedling_count, 1)`

4.4.6. Sapling

#	7.1
Caption	Sapling – Count
Type	R Script
Entity	sapling
Purpose	Total number of saplings

Code	`# convert missing values to 0`
	`sapling$sapling_dbh1[ is.na(sapling$sapling_dbh1)] <- 0`
	`sapling$sapling_dbh2[ is.na(sapling$sapling_dbh2)] <- 0`
	`sapling$sapling_dbh3[ is.na(sapling$sapling_dbh3)] <- 0`
	`sapling$sapling_dbh4[ is.na(sapling$sapling_dbh4)] <- 0`
	
	`sapling$sapling_count <- with( sapling,`
	`  sapling_dbh1 + sapling_dbh2 + sapling_dbh3 + sapling_dbh4 )`

#	7.2
Caption	Sapling – Basal area
Type	R Script
Entity	sapling
Purpose	Basal area of saplings

Code	`# convert missing values to 0`
	`sapling$sapling_dbh1[ is.na(sapling$sapling_dbh1)] <- 0`
	`sapling$sapling_dbh2[ is.na(sapling$sapling_dbh2)] <- 0`
	`sapling$sapling_dbh3[ is.na(sapling$sapling_dbh3)] <- 0`
	`sapling$sapling_dbh4[ is.na(sapling$sapling_dbh4)] <- 0`
	
	`sapling$sapling basal area <- with( sapling,`
	`  0.01 * pi * (sapling_dbh1*(1.5/2)^2 + sapling_dbh2*(2.5/2)^2 +`
	`sapling_dbh3*(3.5/2)^2 + sapling_dbh4*(4.5/2)^2 ))`

#	7.3
Caption	Sapling – AG Biomass
Type	R Script
Entity	sapling
Purpose	AGB of saplings

Code	`# convert missing values to 0`
	`sapling$sapling_dbh1[ is.na(sapling$sapling_dbh1)] <- 0`
	`sapling$sapling_dbh2[ is.na(sapling$sapling_dbh2)] <- 0`
	`sapling$sapling_dbh3[ is.na(sapling$sapling_dbh3)] <- 0`
	`sapling$sapling_dbh4[ is.na(sapling$sapling_dbh4)] <- 0`
	
	`height1 <- h_model( 1.5 )`
	`height2 <- h model( 2.5 )`
	`height3 <- h_model( 3.5 )`
	`height4 <- h_model( 4.5 )`
	
	`# Chave et al. (2014)`
	`sapling$sapling_biomass_ag <- with( sapling,`
	`  0.0673*( (sapling_dbh1 *(WD *1.5^2 * height1)^0.976) + (sapling_dbh2 *(WD *2.5^2`
	`* height2)^0.976) + (sapling_dbh3 *(WD *3.5^2 * height3)^0.976) + (sapling_dbh4 *(WD`
	`*4.5^2 * height4)^0.976)))`
	
	`sapling$sapling_biomass_ag <- sapling$sapling_biomass_ag / 1000`

#	7.4
Caption	Sapling – BG Biomass
Type	R Script
Entity	Saplings – BG Biomass
Purpose	BGB of sapling

Code	`sapling <- getRS_factor( sapling )`
	
	`sapling$sapling_biomass_bg <- sapling$RS * sapling$sapling_biomass_ag`

#	7.5
Caption	Sapling – Biomass
Type	R Script
Entity	sapling
Purpose	Biomass of sapling
Code	`sapling$sapling_biomass_total <- sapling$sapling_biomass_bg + sapling$sapling_biomass_ag`

#	7.6
Caption	Sapling – AG Carbon
Type	R Script
Entity	sapling
Purpose	AGC of sapling
Code	`sapling$sapling_carbon_ag <- CF * sapling$sapling_biomass_ag`

#	7.7
Caption	Sapling – BG Carbon
Type	R Script
Entity	sapling
Purpose	BGC of sapling
Code	`sapling$sapling_bg_carbon_total <- CF * sapling$sapling_biomass_bg`

#	7.8
Caption	Sapling – Carbon
Type	R Script
Entity	sapling
Purpose	Carbon of sapling
Code	`sapling$sapling_carbon_total <- sapling$sapling_carbon_bg + sapling$sapling_carbon_ag`

4.4.7. Shrubs and climbers

#	8.1
Caption	Shrub – Count
Type	R Script
Entity	shrub
Purpose	Total number of shrubs and climbers

Code	`# if missing number of similar parts, set 1`
	`shrub$shrub parts[ is.na(shrub$shrub parts) & shrub$shrub dbh>0 ] <- 1`
	`shrub$shrub_parts[ shrub$shrub_parts==0 & shrub$shrub_dbh>0 ] <- 1`
	
	`shrub$shrub_count <- shrub$shrub_parts`

#	8.2
Caption	Shrub – AG Biomass
Type	R Script
Entity	shrub
Purpose	AGB of shrubs and climbers

Code	`# Liana: model for total above-ground biomass, Malaysia. Table 5 & 6, model 14 in article:`
	`# http://www.hindawi.com/journals/ijecol/2013/658140/`
	
	`# Shrub: Ali A., Xu M.S., Zhao Y.T., Zhang Q.Q., Zhou L.L., Yang X.D., Yan E.R. (2015).`
	`# Allometric biomass equations for shrub and small tree species in subtropical China. Silva Fennica vol. 49 no. 4 article id 1275. 10 p.`
	`# https://www.silvafennica.fi/article/1275`
	
	`# for  missing (height/liana length) cases, get cluster average`
	`clusterH_climber <- sqldf(paste("SELECT cluster_no, AVG(shrub_height) AS meanH_climber FROM shrub WHERE shrub_is_liana AND shrub_height>1.3 GROUP BY cluster no ",sep="" ))`
	`clusterH_shrub   <- sqldf(paste("SELECT cluster_no, AVG(shrub_height) AS meanH_shrub FROM shrub WHERE NOT shrub_is_liana AND shrub_height>1.3 GROUP BY cluster_no ",sep="" ))`
	
	`shrub <- sqldf("SELECT * FROM shrub LEFT JOIN clusterH_climber USING (cluster_no)")`
	`shrub <- sqldf("SELECT * FROM shrub LEFT JOIN clusterH_shrub   USING (cluster_no)")`
	`clusterH_climber <- NULL`
	`clusterH_shrub <- NULL`
	
	`# if no data, set 2m. FIX THIS`
	`shrub$meanH_climber[ is.na(shrub$meanH_climber)] <-2`
	`shrub$meanH_shrub[ is.na(shrub$meanH_shrub)] <-2`
	
	`shrub$meanH_climber <- as.numeric(shrub$meanH_climber)`
	`shrub$meanH_shrub <- as.numeric(shrub$meanH_shrub)`
	
	`shrub$shrub_est_height <- ifelse( !is.na(shrub$shrub_height) & shrub$shrub_height>0, shrub$shrub_height,`
	`  ifelse( shrub$shrub_is_liana, shrub$meanH_climber, shrub$meanH_shrub ))`
	
	`# Biomass model is here`
	`shrub$shrub_biomass_ag <- ifelse( shrub$shrub_is_liana,`
	` 10^(0.275 + 0.470*log(shrub$shrub_dbh, 10) + 0.452 * log(shrub$shrub_est_height, 10)) * 1.011,`
	`  exp( -3.23 + 2.17 * log(shrub$shrub_dbh)))`
	
	`shrub$shrub_biomass_ag <- shrub$shrub_count * shrub$shrub_biomass_ag / 1000`
	`shrub$shrub_biomass_ag [ is.na(shrub$shrub_biomass_ag)] <- 0`

#	8.3
Caption	Shrub – BG Biomass
Type	R Script
Entity	shrub
Purpose	BGB of shrubs

Code	`shrub$shrub_biomass_bg <- RS_shrub * shrub$shrub_biomass_ag`
	
	`# no BGB for climbers`
	`shrub$shrub_biomass_bg[shrub$shrub_is_liana] <- 0`

#	8.4
Caption	Shrub – Biomass
Type	R Script
Entity	shrub
Purpose	Biomass of shrubs
Code	`shrub$shrub_biomass_total <- shrub$shrub_biomass_bg + shrub$shrub_biomass_ag`

#	8.5
Caption	Shrub – AG Carbon
Type	R Script
Entity	shrub
Purpose	AGC of shrubs
Code	`shrub$shrub_carbon_ag <- CF * shrub$shrub_biomass_ag`

#	8.6
Caption	Shrub – BG Carbon
Type	R Script
Entity	shrub
Purpose	BGC of shrubs
Code	`shrub$shrub_carbon_bg <- CF * shrub$shrub_biomass_bg`

#	8.7
Caption	Shrub – Carbon
Type	R Script
Entity	shrub
Purpose	Carbon of shrubs
Code	`shrub$shrub_carbon_total <- shrub$shrub_carbon_bg + shrub$shrub_carbon_ag`

4.4.8. Small shrubs and climbers

#	9.1
Caption	Small shrubs – Count
Type	R Script
Entity	small_shrub
Purpose	Total number of small shrubs and climbers

Code	`# missing values set to 0`
	`small_shrub$ss_dbh1[ is.na(small_shrub$ss_dbh1)] <- 0`
	`small_shrub$ss_dbh2[ is.na(small_shrub$ss_dbh2)] <- 0`
	
	`small_shrub$smallshrub_count <- small_shrub$ss_dbh1 + small_shrub$ss_dbh2`

4.4.9. Plot

#	10.1
Caption	Plot - Count
Type	R Script
Entity	plot
Purpose	Number of plots – and scripts for plot level results (/ha) into CSV file
Code	

```
plot$plot_count <- 1

##########################################
# PLOT RESULTS into CSV -------------------
##########################################

## CANOPY CLOSURE --------------
x  <- cbind(plot$cc_east, plot$cc_south, plot$cc_center, plot$cc_north, plot$cc_west)
# average, without NAs
plot$Canopy_closure <- as.integer(rowMeans(x, na.rm = TRUE))
plot$Canopy_closure[is.na(plot$Canopy_closure)] <- 0
# densitometer readings (0-24) converted into percentages (0-100)
plot$Canopy_closure <- 100/24 * plot$Canopy_closure
rm(x)

plot2 <- sqldf("SELECT
plot_id_,cluster_no,plot_no,stratum,Plot_type,plot_access,Canopy_closure FROM plot")

# Land cover is section A (i.e. at plot center point)
lvs2 <- sqldf("SELECT plot_id ,land_cover FROM lvs WHERE luvs_id='A' ")
# read labels for land cover classes
vegType <- dbGetQuery(conn=connection, statement="SELECT veg_type AS land_cover,
veg_type_label
             FROM veg_type_code ");
lvs2 <- sqldf("SELECT * FROM lvs2 LEFT JOIN vegType USING (land_cover)")
plot2 <- sqldf("SELECT * FROM plot2 LEFT JOIN lvs2 USING (plot_id_)")
rm(vegType)
rm(lvs2)

## TREE DATA  --------------
tree2 <- sqldf("SELECT * FROM tree")
tree2$Number_trees_ha <- tree2$tree_count / tree2$plot_area
tree2$Basal_area_ha <- tree2$tree_basal_area / tree2$plot_area
tree2$Volume_bole_ha <- tree2$tree_volume_bole / tree2$plot_area
tree2$AGB_ha <- tree2$tree_biomass_ag / tree2$plot_area
tree2$BGB_ha <- tree2$tree_biomass_bg / tree2$plot_area
tree2$Biomass_ha <- tree2$tree_biomass_total / tree2$plot_area
tree2$AGC_ha <- tree2$tree_carbon_ag / tree2$plot_area
tree2$BGC_ha <- tree2$tree_carbon_bg / tree2$plot_area
tree2$Carbon_ha <- tree2$tree_carbon_total / tree2$plot_area

queryTree <- sqldf("SELECT plot_id_, SUM(Number_trees_ha) AS
Number_trees_ha,SUM(Basal_area_ha) AS Basal_area_ha,
 SUM(Volume_bole_ha) AS Volume_bole_ha,
 SUM(AGB_ha) AS AGB_ha, SUM(BGB_ha) AS BGB_ha, SUM(Biomass_ha) AS Biomass_ha,
SUM(AGC_ha) AS AGC_ha,
 SUM(BGC_ha) AS BGC_ha, SUM(Carbon_ha) AS Carbon_ha
 FROM tree2
 GROUP BY plot_id_ ")

plot2 <- sqldf("SELECT * FROM plot2 LEFT JOIN queryTree USING (plot_id_)")
rm(tree2)

## SAPLING DATA  --------------
saplings2 <- sqldf("SELECT * FROM sapling")
saplings2$Number_trees_ha <- saplings2$sapling_count / saplings2$plot_area
saplings2$Basal_area_ha <- saplings2$sapling_basal_area / saplings2$plot_area
saplings2$AGB_ha <- saplings2$sapling_biomass_ag / saplings2$plot_area
```

```
saplings2$BGB_ha <- saplings2$sapling_biomass_bg / saplings2$plot_area
saplings2$Biomass_ha <- saplings2$sapling_biomass_total / saplings2$plot_area
saplings2$AGC_ha <- saplings2$sapling_carbon_ag / saplings2$plot_area
saplings2$BGC_ha <- saplings2$sapling_carbon_bg / saplings2$plot_area
saplings2$Carbon_ha <- saplings2$sapling_carbon_total / saplings2$plot_area

querySapling <- sqldf("SELECT plot_id_, SUM(Number_trees_ha) AS
Sapling_Number_ha,SUM(Basal_area_ha) AS Sapling_Basal_area_ha,
 SUM(AGB_ha) AS Sapling_AGB_ha, SUM(BGB_ha) AS Sapling_BGB_ha, SUM(Biomass_ha) AS
Sapling_Biomass_ha, SUM(AGC_ha) AS Sapling_AGC_ha,
 SUM(BGC_ha) AS Sapling_BGC_ha, SUM(Carbon_ha) AS Sapling_Carbon_ha
 FROM saplings2
 GROUP BY plot_id_ ")

plot2 <- sqldf("SELECT * FROM plot2 LEFT JOIN querySapling USING (plot_id_)")
rm(saplings2)

## STUMP DATA ##
stump2 <- sqldf("SELECT * FROM stump")
stump2$Number_trees_ha <- stump2$stump_count / stump2$plot_area
stump2$Biomass_ha <- stump2$stump_biomass_total / stump2$plot_area
stump2$Carbon_ha <- stump2$stump_carbon_total / stump2$plot_area

queryStump <- sqldf("SELECT plot_id_, SUM(Number_trees_ha) AS Stump_Number_ha,
 SUM(Biomass_ha) AS Stump_Biomass_ha, SUM(Carbon_ha) AS Stump_Carbon_ha
 FROM stump2
 GROUP BY plot_id_ ")

plot2 <- sqldf("SELECT * FROM plot2 LEFT JOIN queryStump USING(plot_id_)")
rm(stump2)

## DEADWOOD DATA  --------------
dw2 <- sqldf("SELECT * FROM fallen_deadwood")
dw2$Volume_ha  <- dw2$dw_volume / dw2$plot_area
dw2$Biomass_ha <- dw2$dw_biomass/ dw2$plot_area
dw2$Carbon_ha  <- dw2$dw_carbon / dw2$plot_area

queryDW <- sqldf("SELECT plot_id_, SUM(Volume_ha) AS DW_Volume_ha,
 SUM(Biomass_ha) AS DW_Biomass_ha, SUM(Carbon_ha) AS DW_Carbon_ha
 FROM dw2
 GROUP BY plot_id_ ")

plot2 <- sqldf("SELECT * FROM plot2 LEFT JOIN queryDW USING(plot_id_)")
rm(dw2)

# Remove unneeded dataframes
rm(queryTree)
rm(querySapling)
rm(queryStump)
rm(queryDW)

plot2$plot_id_ <- NULL
plot2[is.na(plot2)] <- 0

# sort data
plot2 <- sqldf("SELECT * FROM plot2 ORDER BY cluster_no,plot_no")
write.csv(plot2, paste(FolderResult, "MyCountry_Plot_Results.csv",sep=""))

rm(plot2)
```

4.5. Error script

Error scripts are written into module *nnn-error-functions.R.*

In Calc, they can be seen by clicking this button: 

#	9
Caption	
Type	R Script
Entity	
Purpose	Error script for computing variance, standard error and relative standard error in cluster sampling
Code	

```
### ============================================================
# Error calculation script based on
# "Formulas for estimators and their variances in NFI 28.2.2014 K.T. Korhonen & Olli
Salmensuu, point estimators"
#
# @author Mino Togna, FAO
### ============================================================
# **
# Calculate area error
# **
calculateAreaError <- function( plots , strata ){
  clusters <- getClusters( plots );
  strata <- addStratumCounts( strata , clusters , plots );
  # == (1)
  strata$propInClass <- strata$noCenterPlotsInClass / strata$noPlots;
  # == (2)
  strata$areaInClass <- strata$propInClass * strata$area;
  # == (3)
  clusters <- sqldf("select c.*, s.propInClass
                    from clusters c
                    join strata s
                    on s.stratum = c.stratum");
  clusters$x <- (clusters$noCenterPlotsInClass - clusters$propInClass *
clusters$noPlots ) ^ 2;
  strata <- sqldf( "select
                   s.*,
                   sum(c.x) as x
                   from strata s
                   left outer join clusters c
                   on s.stratum = c.stratum
                   group by s.stratum");
  strata$var <- 1 / (strata$noPlots^2) * strata$noClusters / (strata$noClusters -1 )
* strata$x ;
  # == (4)
  strata$areaVariance <- strata$area^2 * strata$var;
  #absolute error
  strata$areaAbsoluteError <- sqrt( strata$areaVariance );
  # add se%(A(f)) - relative error
  strata$areaRelativeError <- 100 * strata$areaAbsoluteError / strata$areaInClass;
  return (strata);
};
# **
# Calculate quantity error
# **
calculateQuantityError <- function( data , plots, strata, quantitative_variable ) {
  # add plot weight to data
  data <- sqldf( "select d.*, p.weight from data d inner join plots p on d.plot_id =
p.plot id" );
  results <- calculateAreaError( plots=plots , strata=strata );
  strata <-sqldf( "select s.* ,
                  r.areaInClass ,
```

```
                  r.areaVariance,
                  r.areaRelativeError
                  from
                  strata s
                  join
                  results as r
                  on
                  r.stratum = s.stratum" );
  clusters <- getClusters( plots );
  strata <- addStratumCounts( strata , clusters , plots );
  #== add sum of quantity per ha to strata and clusters
  strata <- sqldf(" select s.*, d.quantity from strata s join ( select stratum,
sum(quantity * class) as quantity from data group by stratum ) as d on s.stratum =
d.stratum");
  clusters <- sqldf(" select c.*, d.quantity from clusters c left outer join ( select
cluster, sum(quantity * class) as quantity from data group by cluster ) as d on
c.cluster = d.cluster");
  clusters[is.na(clusters)]<-0;

  #== (5) = (7) / (6)
  strata$meanQuantity <- strata$quantity / strata$noCenterPlotsInClass;
  clusters <- sqldf("select c.*, s.meanQuantity
                     from clusters c
                     join strata s
                     on s.stratum = c.stratum");
  clusters$x <- (clusters$quantity - clusters$meanQuantity *
clusters$noCenterPlotsInClass ) ^ 2;
  strata <- sqldf( "select
                   s.*,
                   sum(c.x) as x
                   from strata s
                   left outer join clusters c
                   on s.stratum = c.stratum
                   group by s.stratum" );
  #== (8)
  strata$meanQuantityVariance <- 1 / (strata$noCenterPlotsInClass^2) *
strata$noClusters / (strata$noClusters - 1 ) * strata$x;
  # == (9)
  strata$totalQuantity <- strata$areaInClass * strata$meanQuantity ;
  # == (10)
  strata$totalQuantityVariance <- strata$areaInClass^2 * strata$meanQuantityVariance
+ strata$meanQuantity^2 * strata$areaVariance;
  # add se for mean quantity se%(x(f))
  strata$meanQuantityAbsolute <- sqrt( strata$meanQuantityVariance );
  strata$meanQuantityRelative <- 100 * strata$meanQuantityAbsolute /
strata$meanQuantity;
  strata$totalQuantityAbsolute <- sqrt( strata$totalQuantityVariance );
  strata$totalQuantityRelative <- sqrt( strata$meanQuantityRelative^2 +
strata$areaRelativeError^2 );
  return (strata);
};
# ====
# extract a dataframe of unique clusters included in the data argument
# ====
getClusters <- function( data ) {
  # clusters
  clusters <- sqldf( "select distinct
                     stratum ,
                     cluster ,
                     sum(weight) as noPlots ,
                     sum(class * weight) as noCenterPlotsInClass ,
                     sum(class) as noPlotsInClass
                     from
                     data
                     group by
                     stratum ,
                     cluster" );
  return ( clusters );
};
# ====
```

```
# add no. plots and no. clusters to all strata
# ====
addStratumCounts <- function( strata , clusters , plots ) {
  # add no of plots to strata
  strata <- sqldf( "select
                   s.*,
                   p.noPlots,
                   p.noPlotsInClass,
                   p.noCenterPlotsInClass
                   from
                   strata s
                   left outer join
                   (select stratum,
                   sum(weight) as noPlots,
                   sum(class * weight) as noCenterPlotsInClass ,
                   sum(class) as noPlotsInClass
                   from plots group by stratum
                   ) as p
                   on
                   p.stratum = s.stratum");
  # add no of clusters to strata
  strata <- sqldf( "select
                   s.*,
                   c.noClusters
                   from
                   strata s
                   left outer join
                   (select stratum, count(*) as noClusters from clusters group by
stratum) as c
                   on
                   c.stratum = s.stratum");
  return ( strata );
}
```

References

Addo-Fordjour, P. and Rahmad, Z.B. (2013). **Development of Allometric Equations for Estimating Above-Ground Liana Biomass in Tropical Primary and Secondary Forests, Malaysia**. International Journal of Ecology, Article ID 658140, 8 p. doi:10.1155/2013/658140

Ali A., Xu M.S., Zhao Y.T., Zhang Q.Q., Zhou L.L., Yang X.D., Yan E.R. (2015). **Allometric biomass equations for shrub and small tree species in subtropical China**. Silva Fennica vol. 49 no. 4 article id 1275. 10 p.

Chave J., Andalo C., Brown S, Cairns M.A., Chambers J.Q., Eamus D., Fölster H., Fromard F., Higuchi N., Kira T., Lescure J.P., Nelson BW, Ogawa H, Puig H, Riéra B, Yamakura T. (2005). **Tree allometry and improved estimation of carbon stocks and balance in tropical forests**. Oecologia 145(1): 87–99.

Chave, Jerome; Rejou-Mechain, Maxime; Burquez, Alberto; Chidumayo, Emmanuel; Colgan, Matthew S.; Delitti, Welington B. C.; Duque, Alvaro; Eid, Tron; Fearnside, Philip M.; Goodman, Rosa C.; Henry, Matieu; Martinez-Yrizar, Angelina; Mugasha, Wilson A.; Muller-Landau, Helene C.; Mencuccini, Maurizio; Nelson, Bruce W.; Ngomanda, Alfred; Nogueira, Euler M.; Ortiz-Malavassi, Edgar; Pelissier, Raphael; Ploton, Pierre; Ryan, Casey M.; Saldarriaga, Juan G.; Vieilledent, Ghislain. (2014). **Improved allometric models to estimate the aboveground biomass of tropical trees**. Global Change Biology, Vol. 20, No. 10, 26.06.2014, p. 3177-3190.

Curtis, R.O. (1967). **Height-diameter and height-diameter age equations for second-growth Douglas fir.** Forest Science 13(4): 365–375.

Freese, F. (1962). Elementary Forest Sampling. Handbook No. 232, Forest Service, U.S. Department of Agriculture. USA. 91 p.

IPCC (2003). **Good Practice Guidance for Land Use, Land-Use Change and Forestry** (GPG-LULUCF) 1.

IPCC (2006). **AFOLU Guidelines 2**. Volume 4.

Korhonen, K. T. & Scott, C. (2016). Formulas for estimators and their variances in NFI. Manuscript for FAO.

Lund, H. Gyde. (1982). **Point sampling--the role in in-place resource inventories**. In: Brann, Thomas B; House, Louis O.; Lund, H. Gyde, tech. coords. In-place resource inventories: principles and practices. Proceedings of a national workshop; 9-14 August 1981; Orono, ME. SAF 82-02. Bethesda, MD: Society of American Foresters; 79-84.

Näslund, M. (1937). **Skogsförsöksanstaltens gallringsförsök i tallskog** (Forest research institute's thinning experiments in Scots pine forests). Meddelanden från statens skogsförsöksanstalt Häfte 29. In Swedish.

Mehtätalo L.; de-Miguel S.; Gregoire T.G. (2015). **Modeling height-diameter curves for prediction**. Can. J. For. Res. 45: 826–837.

Schumacher, F.X. (1939). **A new growth curve and its application to timber yield studies**. Journal of Forestry 37:819–820.

Tomppo, E. (2006). **The Finnish National Forest Inventory**. In: Kangas, A. & Maltamo, M. (eds.). Forest inventory. Methodology and Applications. Managing Forest Ecosystems. Vol 10. Springer, Dordrecht. p. 179-194.