Producing Forest Variables from Point Cloud Data

February 19, 2024

Abstract

A terse and rather technical description of the forest variable production at SLU.

Introduction

The Remote Sensing section of the Department of Forest Resources at the Swedish Agricultural University (SLU) has developed a framework and tools to estimate forest variables wall-to-wall for Sweden[1]. The data are the ALS produced by the Swedish Land Survey and field data from the Swedish National Field Inventory, produced by SLU. The framework handles all parts of the process ensuring that all files that need to be processed are processed (in parallel) through all parts of the process. The process has the following principal parts:

1. Filtering and normalising raw point cloud data.
2. Calculating metrics for the area covered and for the individual field plots.
3. Modelling and prediction.
4. Post-processing: mosaics and evaluation.

The process unit (PU) is the area covered by an individual point data file. These files (and their offsprings) may be processed in parallel, as there are no algorithmic dependencies between them.

Implementation

Framework

To coordinate the processing of all files and processing steps, a `make` script is used. `make` is a tool to handle file dependency chains in complex processing. If a file is added or changed upstream the dependency chain, `make` ensures that all dependent files downstream are created or updated. This greatly simplifies updating the data set by only processing dependent files. A `make` script defines these dependencies and the tools with arguments to execute to produce or update the downstream files. It programmatically describes the complete process.

Once everything is set up properly, all you need to do to process all files (often in the hundreds of thousands of them) through all parts of the process is to execute one command in the command prompt:

```
make -j12 -k.
```

The argument `-j12` tells `make` to use up to 12 threads – use as many as your machine can muster, but using more threads than actually available does not make sense. On big machines with many available threads, a very large number of threads may saturate the file storage bus. Parameter `-k` tells `make` not to stop if it encounters an error, so if you have a corrupt file `make` will continue processing files that are independent of it. If you run the command when all is already up-to-date, nothing will happen except for a message that nothing needs doing.

1 where-to-find

Tools

The two ALS data processing tools used in the first parts of the process are based on PDAL, a general tool for point cloud processing. It uses modules for the actual processing. These may be combined into one command, so that the point cloud file is read once and run through the modules as specified before the resulting file is saved. The modules are used sequentially in the PDAL-based tools. Some of the modules used are included with PDAL, but other had to be developed.

Normalising and filtering

"raw" point data file + DEM → filtered point data file.

A PDAL tool with the following modules that are executed sequentially:

1. **Normalization** (PDAL module `filters.hag_dem`). Using a DEM, height above ground is calculated.
2. **Altitude and point class filtering** (proprietary module `filters.slu_lm`). All points with z-values below −2 m and above 50 m are removed. For the remaining points, all z-values below zero are set to zero. Only points with any of the these classes are retained:[2]
   - 0: created, but never classified,
   - 1: unspecified (also includes any incorrect classes), or
   - 2: point on ground.
3. **Flight overlap removal** (proprietary module `filters.remove_overlap`). The PU is divided into a raster of pixels of a specified size. In each pixel, only points from the flight with the smallest scan angle are retained.

Calculating metrics

Filtered point data file + plot data file → metric raster files and plot file.

A PDAL tool with the following modules that are executed sequentially:

1. **Raster metrics** (proprietary module `filters.raster_metrics`). Creates one raster file for each metric specified.
2. **Plot metrics** (proprietary module `filters.plot_metrics`). Creates a plot file with the original columns and added metric columns. Only contains the plots that are geographically contained by the point data file and is empty if none was contained.

After these modules are executed on all point data files, a tool is run to aggregate all individual plot metric files into one file. This file will be used in the model building.

Modelling and prediction

metric raster file + metrics plot file + metadata file → forest variable files.

To model and estimate an R-based tool is used. A number of plots are chosen from the metric plots file. Using the chosen plots, a local model is calculated for the PU and then applied on the raster metrics to produce forest variable estimations for the PU. The tool is available in the docker container: (`pax-regression --help`).

Post-processing

GDAL is used to produce the final wall-to-wall mosaics and accompanied pyramid files.

2 Swedish Land Survey point classes: `https://www.lantmateriet.se/globalassets/kartor-och-geografisk-information/hojddata/pb_laserdata_nedladdning_skog.pdf`.

Docker

All processing tools, the framework, and the processing environment are included in an Ubuntu-based Docker image available at `https://hub.docker.com/repository/docker/axensten/slu`. So if your computer supports Docker, you can install the image and run the tools individually or using the framework `make` script. Some of the proprietary modules might be of use by themselves. Execute `pdal --options filters.<name>` to get detailed information on their use.

- `filters.slu_lm`,
- `filters.remove_overlap`,
- `filters.raster_metrics`, and
- `filters.plot_metrics` are described above.
- `filters.plot_points` creates individual point cloud files of a specified radius, one for each plot contained by the input file.

Apart from PDAL, GDAL, and other command line tools there are also proprietary ones, *i. e.*:

- `pax-append-tables` aggregates all `.csv` files in a directory into one file.

Data

- All point cloud files produced use the .laz (compressed .las) format.
- All raster files produced use the geotif (.tif) format. The files contain only one band. They may use some non-destructive compression, *i. e.* deflate.
- All tabular data files use semicolon separated text files (.csv, but with semicolon as the column separator). Non-ascii characters are to be avoided, but the files are handled as utf8 if non-ascii characters are present. No BOM (byte order marker).
- All spatial references are in SWEREF99 TM.

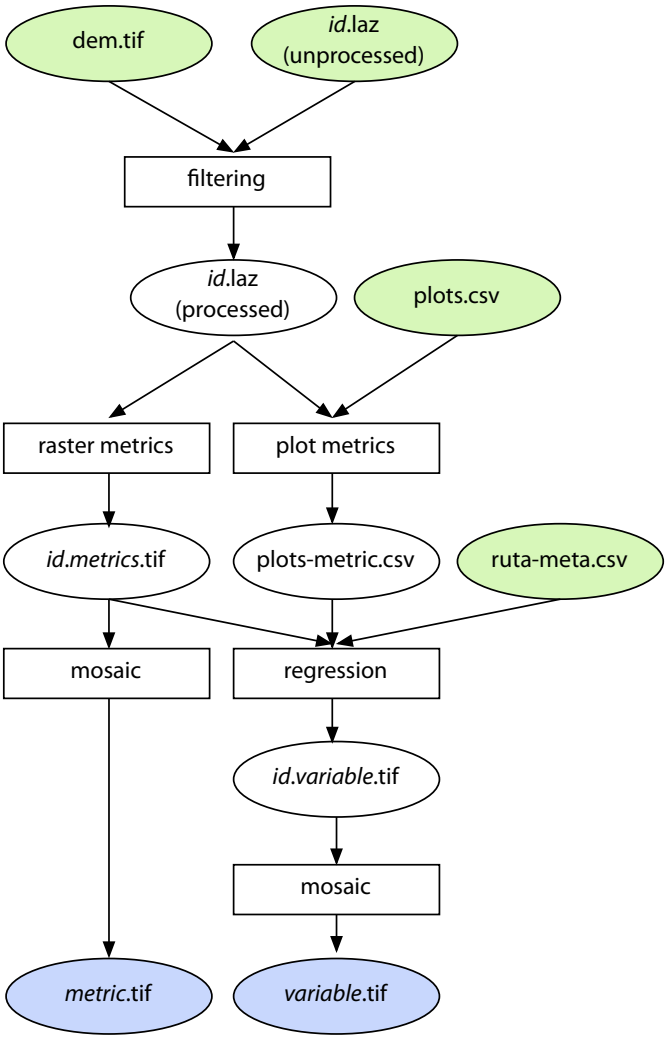

Figure 1. Graphic representation of the process. Ovals represent files, rectangles represent processing. Green ovals are in-data, white are temporary files, and blue are final files.

Metrics used

$\hat{p}_x$ is height percentile x of all values greater than 1.5 m;
$\hat{q}_x = \hat{p}_x \cdot q'_{1.5}$;
$q'_x = \frac{\text{number of first returns above } x \text{ m}}{\text{total number of first returns}}$; and
$\hat{s}$ is the standard deviation of all values greater than 1.5 m.

Choice of field data for modelling

Plots are chosen according to global and local criteria and then outliers are identified and removed. For this, metrics $\hat{p}_{30}$, $\hat{p}_{95}$, and $q'_{1.5}$ must be available for all plots.

- **Global criteria** (in `plot-sample-ruta.r`). Candidate plots must...
  i. have HGV > 3 m *and*
  ii. $-0.7 + 0.6 \cdot \hat{p}_{95} <$ HGV $< 2.6 + 1.1 \cdot \hat{p}_{95}$ *and*
  iii. have no overstorey (only if overstorey information is available) *and*
  iv. have metrics based on at least 20 points (in `plot-filter.hpp`).
- **Local criteria** (in `plot-sample-ruta.r`).
  i. Only plots with the same scanner type as the PU are eligible.
  ii. Only plots with the same scanning season as the PU are eligible (leaves on or off).
  iii. A "badness" value b_i is calculated and the 350 eligible plots with the lowest b_i are chosen. Presently, b_i is simply the distance in meters between the plot and the center of the PU.
- **Model criteria:** An area must have at least 50 plot candidates – or an error is issued (in `plot-sample-ruta.r`).
- **Outlier removal.** Two robust models (`rlm` in R) are used:

$$\begin{aligned} h^\star &\sim \hat{p}_{95} \quad \text{and} \\ v^\star &\sim \hat{p}_{30} + \hat{p}_{95} + \hat{q}_{95}. \end{aligned}$$

  i. For each plot i of the 350 plots, $h_i^\star$ and $v_i^\star$ are predicted using the other plots.
  ii. The distance $m_i = |h_i^\star - Hgv_i, (v_i^\star)^2 - (Hgv_i)^2|_{\text{Mahalanobis}}$ is calculated.
  iii. A threshold t_{80}, equal to percentile 80 of all m_i, is calculated.
  iv. Only plots with $m_i < t_{80}$ are used in the further modelling.

Modelling

Presently, only linear modelling is implemented.

Linear modelling

Hgv	$\sim$	$\hat{p}_{95}$	in dm
$\sqrt{Volume}$	$\sim$	$\hat{q}_{80} + \hat{p}_{95} + \hat{s}$	in m^3/ha
$\sqrt{Biomass_{above}}$	$\sim$	$\hat{q}_{80} + \hat{p}_{95} + q'_{1.5}$	in m^3/ha
$\sqrt{Biomass_{below}}$	$\sim$	$\hat{q}_{80} + \hat{p}_{95} + q'_{1.5}$	in m^3/ha
$Basal\ area$	$\sim$	$\hat{q}_{80} + \hat{s}$	in m^2
Dgv	$\sim$	$\hat{q}_{80} + \hat{p}_{95}$	in m

SIS is estimated for both pine and spruce and will be implemented early 2023.

Prediction

Forest variables for a PU are predicted using the local model. In case a transformation was used, such as $\sqrt{x_i}$ to predict a variable $\sqrt{X_i}$, the inverse function is multiplied by a bias correction factor to obtain the final predicted value, *i. e.*:

$$X_i = \left(\sqrt{X_i}\right)^2 \cdot \frac{\sum_j x_j}{\sum_j \left(\sqrt{X_j}\right)^2}.$$

Mosaics

One geo-tif file (with pyramids) is created for each variable as well as for $\hat{p}_{95}$ and $q'_{1.5}$. The file covers the whole estimated area. The tools used are all from the `gdal` library: `gdalbuildvrt`, `gdal_translate`, and `gdaladdo`.

Evaluation

Jörgen and/or Mats?

File structure

There is a file structure with fixed directories, filenames, and file extensions. Each project has its own directory with as follows:

- **0-project** Contains dem.tif, plots.csv, plots-metric.csv, and ruta-meta.csv.
- **1-rar-source** Presently not used.
- **2-pc-source** Contains the unprocessed *id*.laz files.
- **3-pc-filtered** Contains the processed (filtered and normalised) *id*.laz files.
- **4-raster-metrics** Contains the produced raster metric files (*id.metric*.tif) and their marker files (*id.variant*).
- **5-raster-variables** Contains the produced raster variable files (*id.variable*.tif) and their marker files (*id.variant*).
- **makefile** The `make` code that runs everything.

These directories might have an arbitrary file structure for their files. This structure will be maintained for the destination files.

Most processing steps generate metadata. It is either saved as a shadow file with the same name as the produced file, but with the extension .json or saved in a directory called *id*.metadata in the same directory.

Input files

dem.tif

A raster file with the ground's elevation above sea-level for the whole area to be processed. Located in 0-project.

id.laz

Unprocessed (in 2-pc-source) or processed (in 3-pc-filtered) point cloud files with *z*-values relative sea-level and ground, respectively. Located in 2-pc-source and 3-pc-filtered.

csv-files

These files are text files with the first line containing the headers and then one data item on each line. Line delimiters might be Windows or Linux/MacOS. Column delimiters must be ';'. All lines must have the same number of columns (column delimiters). What columns they are required to contain is defined below. Header names are case sensitive. The order of columns is not important and the file might have other columns, as long as they follow the formatting. Do not use "strange" characters, such as "*#\åäö" etc. Do not use quotes (") around text.

plots.csv

This file contains field measurements, ground data. It has a header line followed by data from one plot per line. Required columns (header names) are:

- `east` is the eastern coordinate in the geographical reference system used in rasters and point cloud files.
- `north` is the northern coordinate in the geographical reference system used in rasters and point cloud files.
- `radius` in the unit used by geographical reference system used in rasters and point cloud files. Most probably meters.
- `leaveson` was the plot measured during the summer season, values are 1. If not, values are 0.

- <variables> Measured values for the variables that are to be estimated. Typically they are: Hgv, Volume, Biomass_above, Biomass_below, Basal_area, and Dgv.

Located in o-project.

ruta-meta.csv

This file contains metadata on the areas covered by the point cloud files. It has a header line followed by data on one area per line. Required columns (header names) are:

- east is the eastern coordinate in the geographical reference system used in rasters and point cloud files.
- north is the northern coordinate in the geographical reference system used in rasters and point cloud files.
- id explanation.
- leaveson was the plot measured during the summer season, values are 1. If not, values are 0.

Located in o-project.

Produced files

plots-metric.csv

Same as plots.csv, but with the addition of required metrics. Located in o-project.

id.*variant*

An empty marker file.

- With the metric files: to mark when the metrics required by the regression variant *variant* were produced.
- With the variable files: to mark when the variables produced the regression variant *variant* were produced.

id.*metric*.tif

The metric raster file for ruta *id* and metric *metric*. Located in 4-raster-metrics.

id.*variable*.tif

The variable raster file for ruta *id* and metric *variable*. Located in 5-variable-metrics.