Imputation Modelling for Growth and Yield Prediction of Trees at Plot and Individual Tree Level

-Pankaj Kumar
University of South Australia (UniSA), Adelaide.
Email: pankaj203kumar@gmail.com

The aim of this project is to develop imputation models for growth and yield prediction of trees, with specific objectives as below:

- Develop workflows to process 3D LiDAR point cloud data.
- Develop and implement imputation models for yield prediction.
- Analyze spatial-temporal transferability of imputation models at plot level using Airborne LiDAR data.
- Implement imputation models at individual tree level using Airborne and UAV LiDAR datasets.

The LiDAR data processing is done in Lastool and R-scripts, while the imputation models are developed and analyzed in R-scripts. The Lastool batch scripts can be used by installing Notepad++ while the R scripts can be used with the installation of R-studio, both of which are freely available. In these scripts, the LiDAR files are input in .laz format. The LiDAR data processing steps have been described in accordance with their respective batch scripts as below:

A. Data Processing Scripts:

1. **Step 1-LiDAR_Clipping:** In the Lastool batch script named "*Step_1(Lastool_Script)-LiDAR_Clipping.bat*", the LiDAR data is clipped with respect to the inventory boundaries to extract point cloud belonging to the region of interest. The LiDAR files are split in accordance with the "plot_ID" of inventory boundaries.

2. **Step_2(Lastool_Script)-LiDAR_Histogram:** The point density information along with height and intensity-based histograms are generated from the clipped LiDAR data in the Lastool batch script named "*Step_2(Lastool_Script)-LiDAR_Histogram.bat*". The height-based histogram information is required to setup height threshold during the noise removal and height normalization processing stages.

3. **Step_3(Lastool_Script)-Noise_Removal:** In the first part of Lastool batch script named "*Step_3(Lastool_Script)-Noise_Removal.bat*", it removes isolated LiDAR points in accordance with grid size of 4m and 1m along xy and z-axis respectively. In the second part, it removes noise points by setting a height threshold in between 0 and 50m. Any points with height below 0m and above 50m are considered as noise and are removed. These height threshold parameters are estimated from the height histogram information as generated in Step 2.

4. **Step 4-Ground Estimation:** The Lastool batch script named "*Step_4(Lastool_Script)-Ground_Estimation.bat*" classifies the noise-free LiDAR points into ground and non-ground. It estimates ground by finding the lowest point in each cell of a coarse grid, whose cell size is determined by the step size parameter. The option "-wilderness" uses a step size of 3m for estimating the ground points.

5. **Step 5-Height Normalization:** The Lastool batch script named "*Step_5(Lastool_Script)-Height_Normalization.bat*" normalizes the height of LiDAR point cloud with respect to estimated ground surface by setting a height threshold in between 0 and 50m. The height threshold parameters are estimated from the height histogram information as generated in Step 2.

6. **Step 6-Plot Standard Metrics:** The Lastool batch script named "*Step_6(Lastool_Script)-Plot_Standard_Metrics.bat*" generates plot based standard metrics from height normalized LiDAR point cloud. Around 66 standard metrics are derived from the height and intensity attributes of normalized LiDAR data for each plot, which have been described in Table 1 below:

No.	Standard Metric	Full Name	Description
1	all	Total number of points	
2	abv	Number of points	Estimated for heights about the cut-off (cut-off used = 2m)
3	min	Minimum	
4	max	Maximum	
5	avg	Average	
6	qav	Quadratic Average	
7	std	Standard Deviation	
8	ske	Skewness	
9	kur	Kurtosis	
10	hom	Height of Median	
11	p1	Percentiles	
12	p5		
13	p10		
14	p25		
15	p50		
16	p75		
17	p90		
18	p95		
19	p99		
20	b10	Bicentiles	
21	b20		
22	b30		
23	b40		

24	b50		
25	b60		
26	b70		
27	b80		
28	b90		
29	c10	Height Counts	Counts the points whose height values are falling into the intervals (e.g. 0-10-20-30-40)
30	c20		
31	c30		
32	c40		
33	c50		
34	d10	Density Counts	Counts are divided by the total number of points and scaled to percentage
35	d20		
36	d30		
37	d40		
38	d50		
39	int_min	Minimum	Estimated from the intensity attribute for the points above height cut-off (cut-off used = 2m)
40	int_max	Maximum	
41	int_avg	Average	
42	int_qav	Quadratic Average	
43	int_std	Standard Deviation	
44	int_ske	Skewness	
45	int_kur	Kurtosis	
46	int_p1	Percentiles	
47	int_p5		
48	int_p10		
49	int_p25		
50	int_p50		
51	int_p75		
52	int_p90		
53	int_p95		
54	int_p99		
55	int_c10	Intensity Counts	
56	int_c20		
57	int_c30		
58	int_c40		
59	int_c50		
60	int_d10	Intensity Density	
61	int_d20		
62	int_d30		
63	int_d40		
64	int_d50		
65	cov	Canopy Cover	Number of first returns above the cover cut-off divided by the number of all first returns and output

			as a percentage (cover cut-off used = 2m).
66	dns	Canopy Density	Number of all points above cover cut-off divided by the number of all returns.

Table 1: Statistical based Standard metrics derived from the height and intensity attributes of LiDAR data.

7. **Step 7-CHM Generation:** To extract individual trees, the Lastool batch script named "*Step_7(Lastool_Script)-CHM_Generation.bat*" generates Canopy Height Model (CHM). The CHM is a Digital Surface Model (DSM) that renders tree height as a continuous surface in raster dataset format with each pixel representing the tree height above the underlying ground topography. The CHM with 0.25m resolution is generated from elevation attribute of noise-free and normalized LiDAR data using spike-free method in which the user-defined value of 0.81 is provided.

8. **Step 8-Individual Tree Boundaries:** In the R-Script named "*Step_8(R_Script)-Individual_Tree_Boundaries.R*", the rLiDAR package is used to delineate individual tree boundaries from the CHM generated with the Lastool. In this approach, the CHM is first smoothed to eliminate spurious local maxima using mean filter and window size *(ws)* parameter. The smoothed CHM is then used to detect and compute the location and height of individual trees based on local maxima function within a fixed window size *(fws)* parameter. The local maxima function detects individual trees above specified height threshold *(minht)* parameter within fixed window. The detected locations of trees are then used to delineate individual tree boundaries from the smoothed CHM based on height variation *(exclusion)* and maximum crown radius *(maxcrown)* parameters. For example, the exclusion parameter of 0.2 will exclude all the pixels for a single tree that has a height value of less than 20% of the maximum height from the same tree. The input parameters in this approach are estimated empirically and then same values are applied for each LiDAR-derived CHM. The empirically estimated values of input parameters are shown in Table 2.

Parameter	**Value**
Window Size *(ws)*	7
Fixed Window Size *(fws)*	7
Height Threshold *(minht)*	2m
Height Variation *(exclusion)*	0.2
Maximum Crown Radius *(maxcrown)*	5m

Table 2: Empirically estimated parameters input in the individual tree detection approach.

9. **Step 9-Boundaries Editing:** This step is used to edit, and clean individual tree boundaries identified from the R-script. In the first part, the individual tree boundaries are edited in ESRI ArcGIS tool, to eliminate any spurious polygon parts that might appear in the dataset. In the ArcGIS, the "Eliminate Polygon Part" tool is used that eliminates any polygon parts which are less than 25 Percentage in size from its respective polygon. This is to ensure that the spurious polygon parts could not be used to clip individual trees from the LiDAR data in the next step. The ArcGIS model named "*Step_9(ESRI_ArcGIS)-Boundaries_Editing.svg*" has been generated in .svg format which can be directly imported onto a layout in the ESRI ArcGIS.

   In the second part, the individual tree boundaries are manually assigned with IDs in ArcGIS with respect to the marking of trees over the ground inside each plot. This is done with the ground measured GNSS positions taken for each tree.

10. **Step 10-Individual Tree Clipping:** In the Lastool batch script named "*Step_10(Lastool_Script)-Individual_Tree_Clipping.bat*", the height normalized LiDAR data is clipped with respect to the individual tree boundaries to extract point cloud belonging to the region of interest. The LiDAR files are split in accordance with the "Tree_ID" of each individual tree boundary.

11. **Step 11-Individual Tree Standard Metrics:** The Lastool batch script named "*Step_11(Lastool_Script)-Individual_Tree_Standard_Metrics.bat*" generates individual tree based standard metrics from height normalized LiDAR point cloud. Around 66 standard metrics are derived from the height and intensity attributes of normalized LiDAR data for each individual tree, which have been described in Table 1.

12. **Step 12 & Step 13-Plot Voxel Metrics & Individual Tree Voxel Metrics:** The plot and individual tree voxel metrics are generated from height normalized LiDAR data in R-scripts named "*Step_12(R_Script)-Plot_Voxel_Metrics.R*" and "*Step_13(R_Script)-Individual_Tree_Voxel_Metrics.R*" respectively. The developed voxel metrics have been described as below:

    - **Vertical Complexity Index (VCI):** The Vertical Complexity Index (VCI) is a normalization of the Shannon diversity index. The diversity index is a quantitative measure that identifies diversity in the data and is based on the number and frequency of species present in it. In the context of LiDAR data, the VCI measures elevational distribution – how even or uneven – of points through the canopy. It is estimated as

$$VCI = -\sum_{i=1}^{HB} \frac{(p_i * \ln(p_i)}{\ln(HB)}$$

      where, HB is the total number of height bins, and p_i is the proportional abundance of LiDAR returns in height bin *i*.

The points are divided into height bins and then VCI characterises the elevational distribution of points within different bin layers. The output value of VCI is in between 0 and 1 with value closer to 1 indicates that most height bins have equal number of LiDAR returns. The number of height bins are based on the maximum height of the LiDAR points, however, increase in the number of height bins improves the estimation.

In the R-script, the VCI metrics are generated from 0.3m, 0.5m, 1m, 1.5m, 2m, 2.5m and 3m resolution of voxels along XYZ axes.

- **Coefficient of Variation for Leaf Area Density (CV_{LAD}):** Leaf Area Index (LAI) is a dimensionless quantity that characterizes plant canopies and is measured as leaf area per unit ground surface area. LAI for the layer *i* between *z* and *(z + dz)* can be estimated using Beer-Lambert Law as

$$LAI_i = -\frac{In(P_i)}{k}$$

where *k* is an extinction coefficient with its values varying in between 0.28 to 0.67. The extinction coefficient is an indicator of interception of light through the forest canopy.

The value of *k* is well approximated as 0.5 for a canopy with spherical distribution and this value allows an adequate representation of many real canopy types. The P_i is gap fraction from the canopy top to a given height *z* and estimated as the ratio of number of returns below a specified height and the total number of returns as

$$P_i = \frac{N_{[0:z]}}{N_{total} - N_{[0:z+dz]}}$$

where $N_{[0:z]}$ is the number of returns below z, N_{total} is the total number of returns and $N_{[0:z+dz]}$ is the number of returns below *z+dz*.

The normalized profile of Leaf Area Density (LAD) is obtained by dividing the resulting values of LAI_i by *dz*. Finally, the Coefficient of Variation for Leaf Area Density (Cv_{LAD}) is estimated as ratio of the standard deviation to the mean of normalized LAD. Cv_{LAD} is a normalized measure of vertical dispersion of canopy density and provides information about vertical structure and heterogeneity of a stand. Its value varies in between 0 and 1 with values near to 1 indicating higher variation in canopy density.

In the R-script, Cv_{LAD} metrics are generated from 0.3m, 0.5m, 1m, 1.5m, 2m, 2.5m and 3m resolution of voxels along XYZ axes.

- **Biomass Voxel Metrics:** The biomass voxel-based metrics are estimated based on the LiDAR points falling within each sub-voxel (SV). For example, each SV that is divided into 10m interval is referred as SV_1 (0-10m), … SV_2 (10-20m), SV_7 (60-70m), SV_8 (70-80m). The point cloud is divided into height bins and then metrics are calculated based on number of points (P), frequency ratio (FR), median height (H_{med}) & median intensity (I_{med}) values falling within each sub-voxel. The frequency ratio is estimated as number of points existing within SV_i against the fraction of total LiDAR points. The biomass voxel metrics can be estimated as:

    - Variable Sub-Voxel (Svi): SV_i refers to the variables (i.e. P, FR, Hmed, Imed) derived from the points within corresponding sub-voxel SV. The SV_i is calculated based on number of points (P_SV), frequency ratio (FR_SV) and median intensity (Imed_SV) values. For example, P_SV_2 corresponds to the total LiDAR points within SV_2 ranging from 10-20m & I_SV_{3med} refers to the median intensity values of points within SV_3 ranging from 20-30m, as shown in Figure 1.

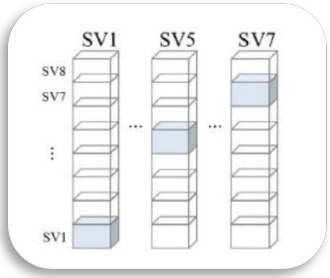

Figure 1: Variable Sub-Voxel (SV_i)

    - Variable Density (D_i): D_i provides the cumulative (total) number of points located in the sub-voxels at and above SV_i. The D_i is calculated based on number of points (P_D), frequency ratio (FR_D) and median intensity (I_D_{med}) values. For example, P_D_3 corresponds to the total number of LiDAR points within and above SV_3, ranging from 20m to highest level & I_D_{4med} refers to the median intensity values of points within and above SV_4, ranging from 30m to highest level, as shown in Figure 2.

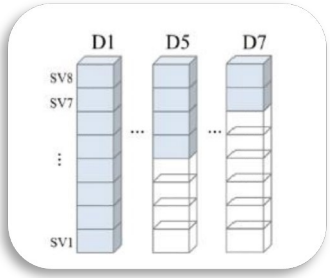

Figure 2: Variable Density (D_i)

- **Variable Sub-Voxel Maximum (SVM):** SVM is the location of sub-voxel with maximum point density found within the entire range of SV. The SVM is calculated based on median height value of points in identified sub-voxel (H_SVM_{med}) and distribution of points above the identified sub-voxel (F_SVM). These metrics are estimated with respect to single voxel identified with maximum density of points. For example, H_SVM_{med} corresponds to the median height of points within single voxel, identified with maximum point density and F_SVM refers to the distribution of points above and with respect to the identified voxel, as shown in Figure 3.

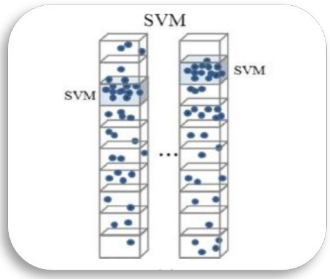

Figure 3: Variable Sub-Voxel Maximum (SVM)

In the R-script, these biomass voxel metrics are generated from 5m, 10m, 15m, 20m, 25m, 30m, 35m, 40m and 45m resolution of voxels along Z axis.

- **Canopy Closure (CC_Above) and Mean Percentage Canopy Closure (per_cc_above) at Height:** Canopy closure is the forest area covered by the vertical projection of tree leaves. It is estimated by dividing the point cloud into sub-voxels and after applying certain height thresholds to the LiDAR returns (e.g. canopy closure above 10m height), as shown in Figure 4. The canopy closure (CC_Above) is calculated as number of sub-pixels containing LiDAR returns at a chosen height with respect to the total number of sub-pixels. The canopy closure values near ground level are highest and reduces with height.

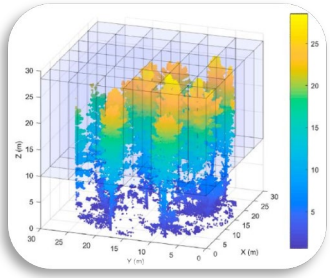

Figure 4: Voxelization at 10m height.

Mean Percentage Canopy Closure is defined as the percentage of forest area covered by canopy crown. It is estimated by subdividing the point cloud into sub-pixels and after applying certain height thresholds to the LiDAR returns (e.g. canopy closure above 10m

height), as shown in Figure 4. The mean percentage canopy closure at height is calculated as the percentage of LiDAR points at a chosen height with respect to the total number of points.

In the R-script, these voxel metrics are generated at height above 5m, 10m, 15m, 20m and 25m from 1m resolution of voxels along XYZ axes.

- **Effective Number of Layers (ENL):** Effective Number of Layers (ENL) is a measure to quantify the vertical structure of forest area. It describes the vertical structure based on the occupation of different vertical layers in relation to total space occupation of a stand. The point cloud is divided into different voxel layers and then each voxel layer is assessed based on the presence of points within them. hD_ENL index is then computed based on the proportion of filled voxels within each layer with respect to the total number of filled voxels, where h represents the Hill number (i.e. 0D_ENL, 1D_ENL, 2D_ENL).

  0D_ENL index provides an estimation of stand height, while 1D_ENL (Exponential Shannon Index) and 2D_ENL (inverse Simpson Index) indices provide the quantification of effective layers. The 0D_ENL and 1D_ENL indices include the weighted space occupation (i.e. 2D_ENL weights occupied layers more than 1D_ENL). Thus, their values increase with increasing stand height and more even distribution of tree components across the vertical profile. 0D_ENL, 1D_ENL, 2D_ENL can be computed as:

$$0D\_ENL = \sum_{i=1}^{N_{top}} p_i^0$$

$$1D\_ENL = \exp\left(-\sum_{i=1}^{N_{top}} p_i * \ln p_i\right)$$

$$2D\_ENL = 1/\sum_{i=1}^{N_{top}} p_i^2$$

  where p_i is the proportion of filled voxels in the ith vertical layer to the sum of filled voxel in the entire volume, while N_{top} refers to the top stand height. With more evenly occupied layers, the values of 1D_ENL and 2D_ENL tend to increase as layers are more weighted (weighing of 2D_ENL > weighing of 1D_ENL).

  In the R-script, these voxel metrics are generated from 1m and 0.5m resolution of voxels along XY and Z axis respectively.

13. **Step 14 & Step 15-Plot Integrated Metrics & Individual Tree Integrated Metrics:** The integration of ground inventory metrics with LiDAR derived standard and voxel metrics for plot and individual trees are done in R-scripts named "Step_14(R_Script)-Plot_Integrated_Metrics.R" and "Step_15(R_Script)-Individual_Tree_Integrated_Metrics.R" respectively.

B. Imputation Modelling Scripts:

Imputation modelling is the process of predicting ground metrics for target data with respect to reference data. In forest inventory, the imputation models are used to assign properties for plantation area (or individual tree) that have not been inventoried, based on ground inventory and statistical metrics available for sample area. The ground inventory metrics (i.e. DBH, Basal Area, Height etc.) are estimated using the field measurements, while the statistical metrics (i.e. statistical metrics) are derived from remote sensing data (i.e. LiDAR point cloud). The imputation model incorporates two set of datasets with a reference data containing both ground inventory and statistical metrics, while a target data containing only statistical metrics. The model is then used to impute ground metrics for target data by finding relationship between ground inventory and statistical metrics in the reference dataset. In the imputation models, the integrated metrics for plot and individual tree are input. In this project, kNN (k-Nearest Neighbor) and Random Forest models were analyzed and developed in R-script, which are being commonly used in Forest inventory study. The implementation of kNN and Random Forest models within traditional and spatial-temporal domain has been described in accordance with their respective R scripts as below:

kNN Model: The kNN model imputes the ground variables for target data from the nearest neighbor (i.e. proximity) found in the reference data. The kNN model is implemented with "yaImpute" package in R, using "yai" function which develops imputation model and "impute" function that imputes the response (i.e. ground inventory) variables for target observations. The syntax of "yai" and "impute" functions have been described as below:

yai(x, y, k, noRefs, method, mtry, ntree, rfMode)

x=statistical metrics for both reference and target data
y=ground metrics for reference data
k=number of nearest neighbors (1,3 or 5)
method=for computing distance and finding neighbors (randomForest)
mtry=number of x variables randomly selected for randomForest method (1/3 of variables)
ntree=number of regression trees when the method is randomForest (100)
rfMode=randomForest method mode (regression)

impute(model, ancillaryData, method, vars)

model= knn model developed using yai function
ancillaryData=combined ground inventory and statistical metrics for both reference and target data.
Method=to compute the imputed values (mean method is used for the k neighbors)
Vars=variables to impute (ground metrics for target data)

The accuracy assessment of imputed ground metrics for target data is done by estimating normalized RMSE and percent biasness with respect to their respective original values.

In the R-scripts named "*01-R_Script-Plot_Traditional_KNN_Model.R*" and "*05-R_Script-Individual_Tree_Traditional_KNN_Model.R*", the kNN model is implemented for plot and individual tree-based datasets respectively using Leave-One-Out-Cross-Validation (LOOCV) approach. In the LOOCV approach, one of the sample from the reference data is left-out during each iteration, the kNN model is developed from rest of the samples and then applied to impute the ground metrics for the left-out sample. Finally, the accuracy is assessed by comparing the imputing and ground metrics available for that sample.

In the R-scripts named "*03-R_Script-Plot_Spatial-Temporal_KNN_Model.R*", the KNN model is implemented for multiple plot-based datasets available for different spatial and temporal frames. In the spatial-temporal transferability, the kNN model is developed from one plot-based dataset and then applied to impute the ground metrics for another plot-based dataset belonging to another spatial and temporal frame.

Random Forest Model: The Random Forest model is a machine learning approach which randomly iterates through relationship established among reference data samples to impute ground metrics for target data. It works on the principal that many relatively uncorrelated models produce ensemble predictions that are more accurate than any of the individual predictions. The Random Forest uses two processes – bagging and feature randomness – to ensure that the behavior of each individual tree model is not too correlated with behavior of any of the other tree models. In bagging process, each individual tree model selects random sample from the training dataset, with replacement resulting in different model trees. In feature randomness process, each tree model picks random subset of features, which leads to more variation among the trees and eventually results in lower correlation across tree models and more diversification. For regression, the tree is fitted many times to bagged versions of the training data and then the average of prediction results is taken.

The Random Forest model is implemented with "randomForest" package in R, using "randomForest" function which develops imputation model and "predict" function that imputes the response (i.e. ground inventory) variables for target observations. The syntax of "randomForest" and "predict" functions have been described as below:

randomForest(n, data, ntree, mtry, importance, na.action)

n=ground metric to be imputed
data=ground inventory and statistical metrics for reference data
ntree= number of regression trees (100)
mtry=number of samples for splitting at each node (1/3 of variables)
importance=True (importance of variables is assessed)
na.action=True (a function to specify the action to be taken in NAs are found in data)

predict(model, data)

model=randomForest model developed using function
data=statistical metrics for target data

In the R-scripts named "*02-R_Script-Plot_Traditional_RandomForest_Model.R*" and "*06-R_Script-Individual_Tree_Traditional_RandomForest_Model.R*", the Random Forest model is implemented for plot and individual tree-based datasets respectively using Leave-One-Out-Cross-Validation (LOOCV) approach.

In the R-script named "*04-R_Script-Plot_Spatial-Temporal_RandomForest_Model.R*", the Random Forest is implemented for multiple plot-based datasets available for different spatial and temporal frames.