LAZ is compressed (zipped) form, LAS is uncompressed form, LASD is dataset form for working in Arc.

Make sure lidar and other spatial data are in same projection.

Lastools run as Batch file scripts for command line. All files including tool.exe and lastoolslicense.txt must be in same folder, unless add D:\lastools\bin to the computer’s PATH (or use SET command) to be able to call the commands by name from any current directory. The current directory should be set to where the .bat file is located. Other input/output is called by their full path if they’re not in the current directory.

Tips:

Drag and drop folders and files to the shell to add the full path.

Get help for tool with e.g. >las2las /? Or las2las -h

Add -**v** option to output file with name “%*original\_filename*%\_1”.

Add -**odir** *folder name* -olaz to store the results as LAZ files in the folder (that must exist).

Use -**odix** *\_plot* to add “\_plot” to the end of all the output file names.

Use -**ocut** *2* to cut 2 characters from the output name (good for cutting out file extensions)

Extracting Lidar data for sites: **lasclip**

Determine which tiles are needed: Use the index in L:\lidar\indx\_q006kpy4.gdb\indx\_q006kpy4, DNR\_QQQ\_ID field has .laz tile name. Paul has organized laz files by county.

Use lastools to clip the laz file(s) to polygon(s): lasclip -i \*.laz -poly \_\_.shp –v -olas

lasclip -i L:\lidar\county\itasca\laz\\*.laz -poly C:\Users\nixon032\Desktop\blandin\Blandin2\_polygons.shp -odir C:\Users\nixon032\Desktop\blandin\lidar2 -v -olas

Both of the above clip all the laz (can do las also) files in the folder against the .shp and stores the results to a las file, depending on how many tiles covers the shapefile extent. It removes empty output. Can merge las files with lasmerge (see end of this doc).

To output separate files for each polygon in the .shp, use **-split**.

lasclip -i L:\lidar\county\itasca\laz\\*.laz -poly C:\Users\nixon032\Desktop\blandin\Blandin2\_polygons.shp -split -odir C:\Users\nixon032\Desktop\blandin\lidar3 -v -olas

Another way is to split the shapefile into separate polygons and then run:

lasclip -i L:\lidar\county\itasca\laz\1942-30-09.laz -poly C:\Users\paul\Desktop\blandin\separate\blandin\_1\_prj.shp -o C:\Users\paul\Desktop\blandin\separate\blandin\_1.las

etc for each shapefile

Cannot use “\*.shp” ☹ Need a way to split and name plots by shapefile: lasclip does this, but output files can’t be named. Someone on the Google user group suggested:

Once you have the split polygons based on your desired attribute, the following script (linux) is supposed to work fine:

for file in \*.shp; do

lasclip -i input.laz -poly $file -o ${file%%.\*}\_clipped.laz -v -odir

done

Here’s mine. When I used all laz files with \*.laz, it took 10 min and didn’t output anything. The reason is that the las files get overwritten even when there is empty output. I tested this with 2 laz files (specified two loop lines with laz file names as input). The solution: use -merged to merge the .laz files into one before clipping. Took two minutes to run.

for %%f in (C:\Users\nixon032\Desktop\blandin\separate\\*.shp) do lasclip -i L:\lidar\county\itasca\laz\\*.laz -merged -poly %%f -o %%f.las -ocut 4 -v -odir C:\Users\nixon032\Desktop\blandin\lidar3

View in ArcMap in order to observe whether ground classifications look good. Use tool **Create LAS Dataset** to create .lasd. Then use the LAS dataset toolbar and visualize by Class (dropdown) (this will create symbology for classes, which can then be edited in the layer properties) and Filter just Ground. From what I’ve observed, there are many points classified as 17 Reserved, including ground points, so, I think it would be useful to run lasground to determine additional ground points for the height normalization.

Compute normalized heights/elevation**:**

(CAN SKIP lasground) Identify bare earth points: **lasground\_new** (Can use this if ground points are not well-classified. Best to make *a single ground file* for easier script setup in the lasheight tool.)

Classifies points into ground (==2) and non-ground (==1). Can create separate “ground” las files for use in lasheight OR leave intact the original classification of any points that are not classified as ground with '-**non\_ground\_unchanged'** (this is a problem if ground points are classified as 17). By default the tool only considers the *last return*. Earlier returns are considered non-ground. You can turn this off by requesting '**-all\_returns'**. The default *step size* has been increased to 25 meters. There is the '-nature' option that uses a step size of 5 meters for terrains without buildings and the '-**wilderness'** option that uses 3 meters if you care for smaller features on the ground in high resolution LiDAR. User can fine-tune the algorithm further by specifying the threshold in meters at which *spikes get removed*. Setting **'-spike 0.5'** will remove up-spikes above 50 centimeter, and setting '-spike\_down 2.5' will remove down-spikes below 2.5 meters in the coarsest TIN. You can ask lasground\_new to *compute the height* above the ground for each point (so you can use lasclassify next without needing to run lasheight first) with **'-compute\_height'** or even ask to have the computed height *replace the elevation value* with option **'-replace\_z'**. Then you directly get a *height normalized* LAS/LAZ file that can be used, for example, with the lascanopy or lasgrid tools or the pit-free canopy height model (CHM) algorithm.

Eg:

lasground\_new -i C:\Users\nixon032\Desktop\blandin\lidarcopy\\*.las -all\_returns -wilderness -odir C:\Users\nixon032\Desktop\blandin\ground\ -v

When I used -keep\_class 2, it filtered it before performing the ground determination, so I ran a las2las filter next to only keep class 2:

las2las -i C:\Users\nixon032\Desktop\blandin\ground\\*.las -keep\_class 2 -odir C:\Users\nixon032\Desktop\blandin\groundonly -v

It ended up with greater density of points but gaps with missing points (??). I used it anyway in the next step, lasheight.

Compute elevation above ground: **lasheight.** Need to do because a ground/non-ground las file is created with lasground, and we want to keep original classifications. Good time to perform filters (see filtering step).

Ground classified points are used to construct a ground TIN and calculates elevation of other points with respect to the TIN. The ground points can be in a separate file '**-ground\_points** *ground.las*’. You can '-**replace\_z'** the elevation value of each point with the computed height. That means that all ground points will have an elevation of zero and all other points will have an elevation that equals their relative height above (or below) the ground TIN. You can also use the height to change the point classification with '-classify\_below -1.0 7' or '-classify\_above 100.0 10' and also '-classify\_between 0.5 2.0 3 -classify\_between 2.0 5.0 4'. Use the computed height to eliminate points with a particular ground height above or below a threshold with the options '-**drop\_below** 1.5' or '-**drop\_above** 6.8'.

lasheight -i C:\Users\nixon032\Desktop\blandin\lidarcopy\\*.las -drop\_scan\_angle\_above 15 -drop\_below 0 -replace\_z -odir C:\Users\nixon032\Desktop\blandin\height\_wo\_ground\ -v

lasheight -i C:\Users\nixon032\Desktop\blandin\lidarcopy\blandin\_1.las -ground\_points C:\Users\nixon032\Desktop\blandin\groundonly\blandin\_1.las -drop\_scan\_angle\_above 15 -drop\_below 0 -replace\_z -o C:\Users\nixon032\Desktop\blandin\height\_w\_groundonly\blandin\_1\_wground.las

etc…

NOTE: height\_w\_ground did not calculate for most points. When I ran it with the groundonly (which is only class 2 points extracted) (height\_w\_groundonly), it worked.

Compared lascanopy results of using groundonly and height\_wo\_ground: ***very similar*** results - probably because the method TINs the ground surface, and missing points are okay at this scale. Can SKIP lasground steps.

Optional Filtering: **las2las** (NOTE: filtering can be done within any tool, see **>lascanopy -h)**

Filter las data (e.g. drop certain heights, scan angles, classifications, returns, intensities), change classifications, specify projection, etc. Sometimes points are far outside the bounding box (corrupted files) and it is handy to remove them with '-**clip\_to\_bounding\_box'**.

e.g.:

las2las -i C:\Users\paul\Desktop\blandin\lidar2\blandin\_1\_1.las -drop\_scan\_angle\_above 15 -first\_only -drop\_class 3 4 5 -keep\_z 0 1000 -v

This created “blandin\_1\_1\_1.las”, dropping scan angle above 15 and below -15, keeping only first returns, dropping classes 3, 4, 5, and keeping heights of 0 to 1000. Beware that many of the points are classified as 17 - Reserved.

Calculating metrics: **lascanopy**

Computer popular forestry metrics for whole plots or by grid cells. Can grid them onto a raster or output a table.

Load polygonal plots from a shapefile with -**lop** *filename.shp*

Eg:

lascanopy -i C:\Users\paul\Desktop\blandin\lidar3\\*.las -files\_are\_plots -lop Blandin2\_polygons.shp -names -height\_cutoff 0 -dns -max -avg -std -ske -kur -p 10 25 50 75 90 95 -d 0 1.37 3.66 5.49 7.32 9.14 10.97 12.8 14.63 16.46 18.29 -centroids -o results.csv

Above script produces results with an “index” column, which, after comparing to an altered shapefile, does not seem to correspond to FID - not sure and probably not usable ☹

Below script produces identical results, but not using shapefile. It has names of the las files in a column.

lascanopy -i C:\Users\paul\Desktop\blandin\lidar3\\*.las -files\_are\_plots -names -height\_cutoff 0 -dns -max -avg -std -ske -kur -p 10 25 50 75 90 95 -d 0 1.37 3.66 5.49 7.32 9.14 10.97 12.8 14.63 16.46 18.29 -centroids -o results\_noshp.csv

To create rasters, exclude -**files\_are\_plots** and add -**step** # (default is 20x20 cell). It will create one raster for each metric calculated.

lascanopy -i 631702062.las -step 3 -height\_cutoff 0 -dns -max -avg -std -o 631702062.tif

In ArcMap:

Use **Create LAS Dataset** tool to view and create index.

Use **Point File Information** tool to create index tiles shapefile and calculate basic stats.

Use **LAS Dataset toolbar** to view in 3D.

Use **LAS Dataset to TIN** to create TIN surfaces to be viewed in 3D in ArcGlobe. First extract only relevant points (e.g. classes, return numbers), either using **las2las** or **LAS to Multipoint** tool. Using ground points (class 2) creates a DEM (digital elevation model), and using first returns creates a DSM (digital surface model).

# Other useful tools:

Merge las files: lasmerge

Grid into raster: lasgrid. Can also create raster with lascanopy.

Create polygon boundaries with **lasboundary**.

To open a quick viewer, use **lasview**. E.g. lasview C:\Users\nixon032\Desktop\blandin\lidar3\\*.las

Create DSM/CHM (digital surface/canopy height model of first returns): Use **lasinfo** to find average point spacing to determine approximate step (cell) size. *Since TINs aren’t created, I would opt for using Arc for creating a TIN. But see spike-free method below, and the viewer*.

See: <https://rapidlasso.com/2016/02/03/generating-spike-free-digital-surface-models-from-lidar/>

<https://rapidlasso.com/2014/11/04/rasterizing-perfect-canopy-height-models-from-lidar/>

Use lasthin -highest before las2dem to keep only highest points within each step/cell.

las2tin -i C:\Users\nixon032\Desktop\blandin\lidar3\\*.las -first\_only -v (this output a multipatch shapefile, not a TIN ☹. But there is a way to create a TIN from this: Convert multipatch to points using Feature Vertices to Points. (Output point features will have Z values stored in their geometry). Create TIN out of these roof points and use Shape.Z as Height Field.)

or to create raster (default output is .asc): las2dem -i C:\Users\nixon032\Desktop\blandin\lidar3\\*.las -first\_only -hillshade -step 1 -v

Spike free version: add -spike\_free to las2dem tool (triangulates all relevantLiDAR returns using Contrained Delaunay algorithm.) Convert the spike-free DEM to TIN using **Raster to TIN** tool (this didn’t work for .asc or .tif files, but I used ASCII to raster to convert to GRID (no extension), then it worked!).

las2dem -i C:\Users\nixon032\Desktop\blandin\lidar3\plot\_1.las -spike\_free 2.61 -odix \_spkfree

To generate on the fly visualization of spike free TIN with lasview: For -spike\_free option’s freeze constraint value, use **lasinfo** to find average pulse spacing, and use value that’s 3x point spacing (-cd to find spacing of last only points).

lasview -i C:\Users\nixon032\Desktop\blandin\lidar3\\*.las -spike\_free 2.61

In viewer, press SHIFT + y to create spike free TIN.