

Script Review: '14_lidR-Processing_to_Git'

SRM-LQ

04/02/2022

Contents

Action

This following indcludes R Markdown documentation of the LiDAR processing steps taken to derive landscape metrics and raster covariates from a continuous point cloud using the lidR package. This report and literate programming, along with its virtual environment were are stored in the github repository here: 13_lidR_PointCloud_Processing.

1 Import LiDAR

LiDAR downloads for the Ahbau region were imported as zip files and unpacked from their top-directory and subdirectory folders using the unzip function. I could not find any published R packages that deal with .7z archive files. Instead custom-written function was adopted from the RAMP project. This was done with the following code chunk:

```
zip_file_ahbau = ("./14_LiDR-Processing_GitRepo/Data/Ahbau.zip")
zip_dir_ahbau_top = ("./14_LiDR-Processing_GitRepo/Data/")
unzip(zip_file_ahbau,
      exdir = zip_dir_ahbau_top,
      overwrite = TRUE)
zip_dir_ahbau_sub = ("./14_LiDR-Processing_GitRepo/Data/Ahbau/Las_v12_ASPRS")
zip_file_ahbau_sub = list.files(
  zip_dir_ahbau_sub,
  full.names = T,
  recursive = F,
  pattern = '.7z$')
Write RAMP function and extract
un7zip = function(archive, where) {
  archive <- normalizePath(archive)
  current_path <- setwd(where)
  system(paste("7zr x", archive, sep = " "))
  setwd(current_path) }
un7zip(zip_file_ahbau_sub, zip_dir_ahbau_top)
```

2 Read, Validate, and Assemble LiDAR Collection

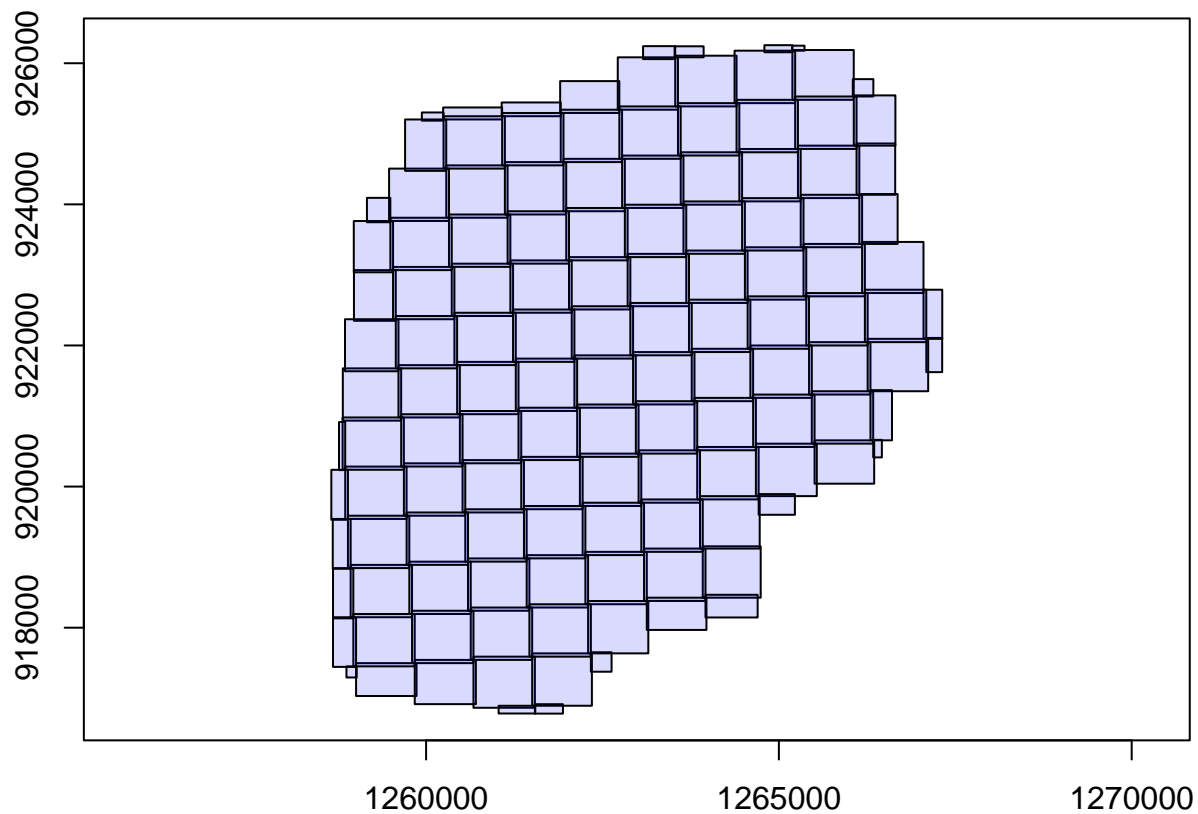
LiDAR data for the Ahbau region included 311 tiles. Tiles were assembled, imported and processed as a LAS collection object using the *LAScatalog* Engine. Initial validation of catalog tiling reported overlapping areas. As per PI recommendations, retiling and rebuffering was implemented to remove potential duplicates with new chunk sizing of 1km each, *readALSLAScatalog* function, which allowed settings for chunk processing, spatial indexing, buffering and duplicate cleaning. Parameters were calibrated to conduct retiling and remove duplicates generated by tiles. This was confirmed in validation reports below.

```
las_ctg_ahbau = readALSLAScatalog("./Data/Ahbau/Las_v12_ASPRS/", select = "xyzcr",)
```

```
## |
```

```
|=====
```

```
plot(las_ctg_ahbau)
```



```
#las_check(las_ctg_ahbau)
```

```
|=====
===== | 99%Be careful, some tiles seem to overlap each
other. lidR may return incorrect outputs with edge artifacts when processing this catalog.
```

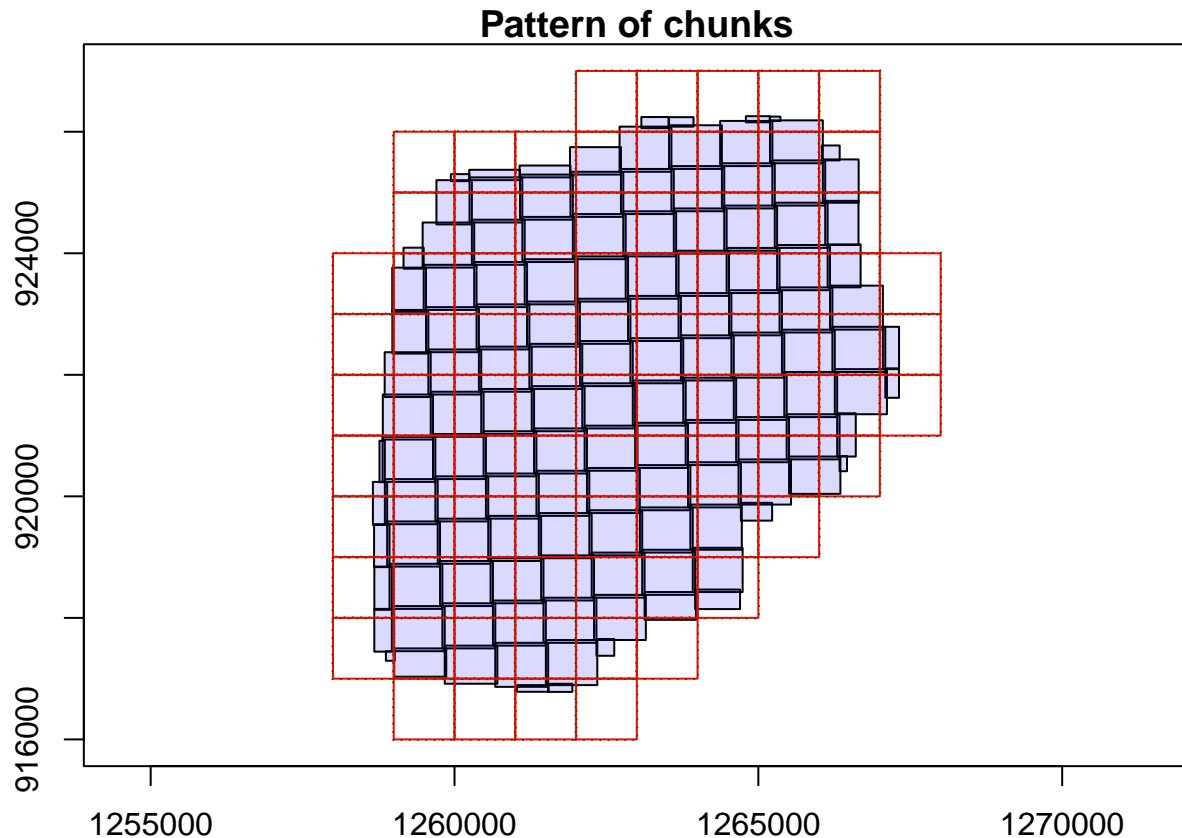
```
Checking headers consistency
- Checking file version consistency... ✓
- Checking scale consistency... ✓
- Checking offset consistency... ✓
- Checking point type consistency... ✓
- Checking VLR consistency... ✓
- Checking CRS consistency... ✓
Checking the headers
- Checking scale factor validity... ✓
- Checking Point Data Format ID validity... ✓
Checking preprocessing already done
- Checking negative outliers... ✓
- Checking normalization... no
Checking the geometry
- Checking overlapping tiles...
  △ Some tiles seem to overlap each other
- Checking point indexation... yes
```

Catalog

indexing and chunk processing was attempted first time around using the *catalog_laxindex* and the *catalog_retile* operations. This was a little more sluggish and buggy, and only relies on the internal lidR tools of *readALSLAScatalog* that was faster. The *readALSLAScatalog* function allows calibrating chunk processing, rebuffering, indexing, and filtering while reading in. According to lidR manual, ‘catalogs with files that overlap are not natively supported... the user should filter any overlaps’ (cite: pg #no.). To avoid edge artifacts, recommended buffering was left at 10m and the ‘-drop_class 19’ filter was applied. For LAS point classifications, see ESRI resource [here](#).

```
las_ctg_ahbau_indexed = readALSLAScatalog("./Data/Ahbau/Las_v12_ASPRS/")
opt_output_files(las_ctg_ahbau_indexed) = paste0(tempdir(), "./Data/las_ctg_ahbau_indexed")
opt_select(las_ctg_ahbau_indexed) = "xyzcr"
opt_filter(las_ctg_ahbau_indexed) = '-drop_class 19'
opt_chunk_size(las_ctg_ahbau_indexed) = 1000
opt_chunk_buffer(las_ctg_ahbau_indexed) = 10
filter_duplicates(las_ctg_ahbau_indexed)
is.indexed(las_ctg_ahbau_indexed)
#las_check(las_ctg_ahbau_indexed)
plot(las_ctg_ahbau_indexed, chunk = TRUE)
```

```
## [1] TRUE
```



===== | 99%Be careful, some tiles seem to overlap each other. lidR may return incorrect outputs with edge artifacts when processing this catalog.

Checking headers consistency

- Checking file version consistency... ✓
- Checking scale consistency... ✓
- Checking offset consistency... ✓
- Checking point type consistency... ✓
- Checking VLR consistency... ✓
- Checking CRS consistency... ✓

Checking the headers

- Checking scale factor validity... ✓
- Checking Point Data Format ID validity... ✓

Checking preprocessing already done

- Checking negative outliers... ✓
- Checking normalization... no

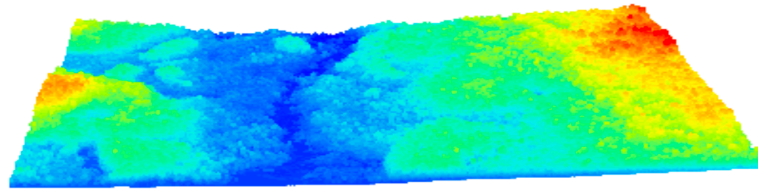
Checking the geometry

- Checking overlapping tiles...
△ Some tiles seem to overlap each other
- Checking point indexation... yes

For visualization purposes, a single las tile '093g030122ne' was loaded, processed and plotted. All subsequent

illustrations of processing operations were rendered using this tile chunk.

```
las_tile_ahbau = readLAS("./Data/Ahbau/Las_v12_ASPRS/093g030122ne.las", select = 'xyzcr', filter = '-dr')
plot(las_tile_ahbau, bg = "white")
```

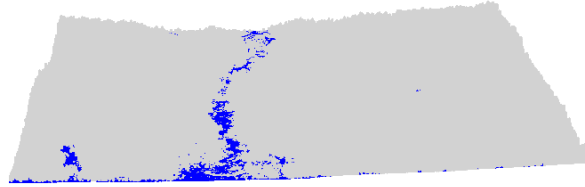


3 Classification of Ground Points

Classification of ground points was implemented twice comparing system running time and data voids between the ‘csf()’ Cloth Simulation Filter algorithm [zhang2016] and the ‘pmf()’ Progressive Morphological Filter algorithm [zhang2003]. To cover all potential eventualities, the RCSF package was installed from CRAN source file and loaded into the github repository ready for virtual environment cloning (see link in Action section). Though this package is not a package dependent, just seems to facilitate better with fewer bugs popping up.

The Cloth Simulation Filter algorithm was fitted with three parameters in order to account for variable topography across the study site and to reduce errors during any potential post-processing. This included the `sloop_smooth` that was applied over a cloth resolution of 10cm at a rigidity factor of 1.

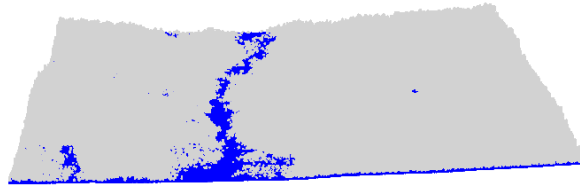
```
library(RCSF)
opt_output_files(las_ctg_ahbau_indexed) = paste0(tempdir(), "./Data/las_ctg_ahbau_csf")
las_ctg_ahbau_csf = classify_ground(las_ctg_ahbau_indexed, csf(sloop_smooth=TRUE, 0.5, 1))
las_tile_ahbau_csf = classify_ground(las_tile_ahbau, csf(sloop_smooth=TRUE, 0.5, 1))
plot(las_tile_ahbau_csf, color = "Classification", bg = "white")
```



3.1 Progressive Morphological Filter

The Progressive Morphological Filter algorithm was tuned using the ‘util_makeZhangParam()’ method, which generated a list of candidate parameters for testing window size and threshold. This internal function simply provides a sequence of parameters that match those in the original paper [Zhang2003].

```
opt_output_files(las_ctg_ahbau_indexed) = paste0(tempdir(), "./Data/las_ctg_ahbau_pmf")
util_makeZhangParam()
las_ctg_ahbau_pmf = classify_ground(las_ctg_ahbau_indexed, pmf(seq(5, 9, 13), seq(3, 3, 3)))
las_tile_ahbau_pmf = classify_ground(las_tile_ahbau, pmf(seq(5, 9, 13), seq(3, 3, 3)))
plot(las_tile_ahbau_pmf, color = "Classification", bg = "white")
```



4 Classification and Removal of Noise

Classification of photon noise was applied using the ‘sor’ Statistical Outlier Removal’ algorithm. This was fitted with default neighbourhood sample and multiplier of $k=10$ and $m=3$. Four possible methods for noise screening were found: 1) classification with internal algorithms and filtering with ‘filter_poi’ function and ‘LASNOISE’ string, 2) or filtering using the LAS classification codes with ‘opt_filter’ tool and ‘-drop class 19’ string, 3) screening by threshold such as ‘ $Z > 40$ & $Z < 0$ ’, 4) or screening by 95th percentiles. Seems a significant topic in forestry so leave this to your better judgement. The first approach seemed slower in the long run due to loss spatial indexing after assigning new catalog object to operate the filter_poi function, as seen below.

```
opt_output_files(las_ctg_ahbau_csf) = paste0(tempdir(), "./Data/las_ctg_ahbau_csf_so")
opt_select(las_ctg_ahbau_csf) = "xyzcr"
opt_filter(las_ctg_ahbau_csf) = "-drop_class 19"
opt_chunk_size(las_ctg_ahbau_csf) = 1000
opt_chunk_buffer(las_ctg_ahbau_csf) = 10
sensor(las_ctg_ahbau_csf) = "als"
index(las_ctg_ahbau_csf) = "quadtree"

las_ctg_ahbau_csf_so = classify_noise(las_ctg_ahbau_csf, sor(k=10, m=3))
opt_output_files(las_ctg_ahbau_csf_so) = paste0(tempdir(), "./Data/las_ctg_ahbau_csf_sor")
las_ctg_ahbau_csf_sor = filter_poi(las_ctg_ahbau_indexed, Classification != LASNOISE)

las_tile_ahbau_csf_so = classify_noise(las_tile_ahbau_csf, sor(k=10, m=3))
las_tile_ahbau_pmf_so = classify_noise(las_tile_ahbau_pmf, sor(k=10, m=3))
plot(las_tile_ahbau_csf_so, color = "Classification", bg = "white")
plot(las_tile_ahbau_pmf_so, color = "Classification", bg = "white")
las_tile_ahbau_csf_sor = filter_poi(las_tile_ahbau_csf_so, Classification != LASNOISE)
las_tile_ahbau_pmf_sor = filter_poi(las_tile_ahbau_pmf_so, Classification != LASNOISE)
```

5 Digital Terrain Model

A digital terrain model was derived from a continuous, cleaned point cloud by applying the ‘Inverse Distance Weighting’ algorithm [tu2020]. The LAScatalog was read in again for faster processing by selecting filter as is common practice in forestry analysis. As per PI’s guidance, the Inverse Distance Weighting algorithm was chosen for its improved running time and its sensitivity to lake anomalies. This was implemented twice using the sample tile to compare visually between those derived from ‘CSF’ and ‘PMF’ processing. Parameters were set to default maximum radius of 50m, neighbourhood of 10 and inverse distance weighting power of 2, with a resolution of 1m.

```
las_tile_ahbau_csf_sor_dtm = grid_terrain(las_tile_ahbau_csf_sor, 1, knnidw(10, 2, 50))
las_tile_ahbau_pmf_sor_dtm = grid_terrain(las_tile_ahbau_pmf_sor, 1, knnidw(10, 2, 50))
las_tile_ahbau_csf_sor_dtm_plot = plot_dtm3d(las_tile_ahbau_csf_sor_dtm, bg = "white")
las_tile_ahbau_pmf_sor_dtm_plot = plot_dtm3d(las_tile_ahbau_pmf_sor_dtm, bg = "white")
```

Visually, the cloth simulation filter produced better DEM result than the pmf that appears grainy. Rousseau (2021: 3.2) mentioned that the progressive morphological filter is an originally raster-based algorithm, while ‘csf’ may perform better with lidR operations that are point-orientated. The former was chosen to derive below a DEM raster for the Ahbau catalog. TODO: run individual las_check on two chunks showing warnings in processing image report of ‘grid_terrain...knnidw()’ operation bottom image:

```
opt_output_files(las_ctg_ahbau_csf_sor) = paste0(tempdir(), "./Data/las_ctg_ahbau_dtm")
opt_select(las_ctg_ahbau_csf_sor) = "xyzcr"
opt_filter(las_ctg_ahbau_csf_sor) = '-drop_class 19'
opt_chunk_size(las_ctg_ahbau_csf_sor) = 1000
opt_chunk_buffer(las_ctg_ahbau_csf_sor) = 10
sensor(las_ctg_ahbau_csf_sor) = 'als'
index(las_ctg_ahbau_csf_sor) = "quadtree"
las_ctg_ahbau_csf_dtm = grid_terrain(las_ctg_ahbau, 1, knnidw())
#opt_stop_early(las_ctg_ahbau_csf) <- FALSE #to skip voids
#las_ctg_ahbau_csf_dtm = grid_terrain(las_ctg_ahbau, 1, tin())
las_ctg_ahbau_csf_sor_dtm_crop = crop(
  las_ctg_ahbau_csf_dtm, extent(las_ctg_ahbau_csf_dtm) - 10)
crs(las_ctg_ahbau_csf_sor_dtm_crop) = 3005
las_ctg_ahbau_csf_sor_dtm_slope = terra::terrain(
  las_ctg_ahbau_csf_sor_dtm_crop, "slope", unit = "radians")
las_ctg_ahbau_csf_sor_dtm_aspect = terra::terrain(
  las_ctg_ahbau_csf_sor_dtm_crop, "aspect", unit = "radians")
las_ctg_ahbau_csf_sor_dtm_shade = hillShade(
  las_ctg_ahbau_csf_sor_dtm_slope, las_ctg_ahbau_csf_sor_dtm_aspect, 40, 270)
plot(las_ctg_ahbau_csf_sor_dtm_shade, col=grey(0:100/100), legend=FALSE)
```

6 Height Normalization of Point Cloud

```
opt_output_files(las_ctg_ahbau_csf_sor) = paste0(tempdir(), "./Data/las_ctg_ahbau_norm")
opt_select(las_ctg_ahbau_csf_sor) = "xyzr"
opt_filter(las_ctg_ahbau_csf_sor) = '-keep_first'
opt_chunk_size(las_ctg_ahbau_csf_sor) = 1000
opt_chunk_buffer(las_ctg_ahbau_csf_sor) = 10
sensor(las_ctg_ahbau_csf_sor) = 'als'
```