Forest Data Analysis Report

Jakob Danel and Frederick Bruch

2023-12-07

Contents

1 Introduction

This report documents the analysis of forest data for different tree species.

2 Methods

2.1 Data acquisition

This file describing the data acquisition of this project.

Sources for gaining information about the tree species in NRW forests:

- Waldmonitor.de (Welle et al. 2022)
- Dominant tree species by Thuenen: https://atlas.thuenen.de/layers/geonode:Dominant_Species_Class

- Validating with Sentinel RGB imageries the shape and texture of the area:
  - leaf color
  - human made /natural shapes of the forest
  - Human made objects in the direct neighboorhoud of the area.

2.2 Preprocessing

In this research study, the management and processing of a large dataset are crucial considerations. The dataset's substantial size necessitates careful maintenance to ensure efficient handling. Furthermore, the data should be easily processable and editable to facilitate necessary corrections and precalculations within the context of our research objectives. To achieve our goals, we have implemented a framework that automatically derives data based on a shapefile, delineating areas of interest. The processed data and results of precalculations are stored in a straightforward manner to enhance accessibility. Additionally, we have designed functions that establish a user-friendly interface, enabling the execution of algorithms on subsets of the data, such as distinct species. These interfaces are not only directly callable by users but can also be integrated into other functions to automate processes. The overarching aim is to streamline the entire preprocessing workflow using a single script, leveraging only the shapefile as a basis. This subsection details the accomplishments of our R-package in realizing these goals, outlining the preprocessing steps undertaken and justifying their necessity in the context of our research.

The data are stored in a data subdirectory of the root directory in the format `species/location-name/tile-nam`
To automate the matching of areas of interest with the catalog from the Land NRW[1], we utilize the intersecting tool developed by Heisig[2]. This tool, allows for the automatic retrieval and placement of data downloaded from the Land NRW catalog. To enhance data accessibility, we have devised an object that incorporates species, location name, and tile name (the NRW internal identifier) for each area This object facilitates the specification of the area to be processed. Additionally, we have defined an initialization function that downloads all tiles, returning a list of tile location objects for subsequent processing. A pivotal component of the package's preprocessing functionality is the map function, which iterates over a list of tile locations (effectively the entire dataset) and accepts a processing function as an argument. The subsequent paragraph outlines the specific preprocessing steps employed, all of which are implemented within the mapping function.

To facilitate memory-handling capabilities, each of the tiles, where one area can span multiple tiles, has been split into manageable chunks. We employed a 50x50m size for each tile, resulting in the division of original 1km x 1km files into 400 tiles. These tiles are stored in our directory structure, with each tile housed in a directory named after its tile name and assigned an id as the filename. Implementation-wise, the `lidr::catalog_retile` function was instrumental in

[1]https://www.opengeodata.nrw.de/produkte/geobasis/hm/3dm_l_las/3dm_l_las/, last visited 7th Dec 2023
[2]https://github.com/joheisig/GEDIcalibratoR, last visited 7th Dec 2023

achieving this segmentation. The resulting smaller chunks allow for efficient iteration during subsequent preprocessing steps.

The next phase involves reducing our data to the actual size by intersecting the tiles with the defined area of interest. Using the `lidR::merge_spatial` function, we intersect the area derived from the shapefile, removing all point cloud items outside this region. Due to our tile-wise approach, empty tiles may arise, and in such cases, those tiles are simply deleted.

Following the size reduction to our dataset, the next step involves correcting the `z` values. The `z` values in the data are originally relative to the ellipsoid used for referencing, but we require them to be relative to the ground. To achieve this, we utilize the `lidR::tin` function, which extrapolates a convex hull between all ground points (classified by the data provider) and calculates the z value based on this structure.

Subsequently, we aim to perform segmentation for each distinct tree, marking each item of the point cloud with a tree ID. We employ the algorithm described by Li et al. (2012), using parameters `li2012(dt1 = 2, dt2 = 3, R = 2, Zu = 10, hmin = 5, speed_up = 12)`. The meanings of these parameters are elucidated in Li et al.'s work (Li et al. 2012).

Finally, the last preprocessing step involves individual tree detection, seeking a single `POINT` object for each tree. The `lidR::lmf` function, an implementation of the tree data using a local maximum approach, is utilized for this purpose (Popescu and Wynne 2004). The results are stored in GeoPackage files within our data structure.

See Section 5.1 for the implementation of the preprocessing.

3 Analysis

4 References

Li, Wenkai, Qinghua Guo, Marek Jakubowski, and Maggi Kelly. 2012. "A New Method for Segmenting Individual Trees from the Lidar Point Cloud." *Photogrammetric Engineering and Remote Sensing* 78 (January): 75–84. https://doi.org/10.14358/PERS.78.1.75.

Popescu, Sorin, and Randolph Wynne. 2004. "Seeing the Trees in the Forest: Using Lidar and Multispectral Data Fusion with Local Filtering and Variable Window Size for Estimating Tree Height." *Photogrammetric Engineering and Remote Sensing* 70 (May): 589–604. https://doi.org/10.14358/PERS.70.5.589.

Welle, Torsten, Lukas Aschenbrenner, Kevin Kuonath, Stefan Kirmaier, and Jonas Franke. 2022. "Mapping Dominant Tree Species of German Forests." *Remote Sensing* 14 (14). https://doi.org/10.3390/rs14143330.

5 Appendix

5.1 Script which can be used to do all preprocessing

Load the file with the research areas ::: {.cell}

```
sf <- sf::read_sf(here::here("research_areas.shp"))
print(sf)
```

```
Simple feature collection with 12 features and 3 fields
Geometry type: POLYGON
Dimension:     XY
Bounding box:  xmin: 7.071625 ymin: 51.08151 xmax: 8.539877 ymax: 52.25983
Geodetic CRS:  WGS 84
# A tibble: 12 x 4
      id species name                                                  geometry
   <dbl> <chr>   <chr>                                              <POLYGON [°]>
 1     1 oak     rinkerode           ((7.678922 51.85789, 7.675446 51.85752, 7.~
 2     2 oak     hamm                ((7.858955 51.66699, 7.866444 51.66462, 7.~
 3     3 oak     muenster            ((7.618908 51.9154, 7.617384 51.9172, 7.61~
 4     4 pine    greffen             ((8.168691 51.98965, 8.167178 51.99075, 8.~
 5     5 pine    telgte              ((7.779728 52.00662, 7.781616 52.00662, 7.~
 6     6 pine    mesum               ((7.534424 52.25499, 7.53378 52.25983, 7.5~
 7     7 beech   bielefeld_brackwede ((8.524749 51.9921, 8.528418 51.99079, 8.5~
 8     8 beech   wuelfenrath         ((7.071625 51.29256, 7.072311 51.29334, 7.~
 9     9 beech   billerbeck          ((7.324729 51.99783, 7.323548 51.99923, 7.~
10    10 spruce  marienheide         ((7.558102 51.08358, 7.558317 51.08527, 7.~
11    11 spruce  brilon              ((8.532195 51.41029, 8.535027 51.41064, 8.~
12    12 spruce  osterwald           ((8.369328 51.21693, 8.371238 51.21718, 8.~
```

:::

Init the project ::: {.cell}

```
library(lfa)
sf::sf_use_s2(FALSE)
locations <- lfa_init("research_areas.shp")
```

:::

Do all of the prprocessing steps ::: {.cell}

```
lfa_map_tile_locations(locations,retile,check_flag = "retile")
```

```
No further processing: flag retile is set!Function is already computed, no further computings
```

```
NULL
```

```
lfa_map_tile_locations(locations, lfa_intersect_areas, ctg = NULL, areas_sf = sf,check_fla
```

```
No further processing: flag intersect is set!Function is already computed, no further comput
```

```
NULL
```

```
lfa_map_tile_locations(locations, lfa_ground_correction, ctg = NULL,check_flag = "z_correc
```

```
No further processing: flag z_correction is set!Function is already computed, no further com
```

```
NULL
```

```
lfa_map_tile_locations(locations, lfa_segmentation, ctg = NULL,check_flag = "segmentation"
```

```
No further processing: flag segmentation is set!Function is already computed, no further com
```

```
NULL
```

```
lfa_map_tile_locations(locations, lfa_detection, catalog = NULL, write_to_file = TRUE,chec
```

```
No further processing: flag detection is set!Function is already computed, no further comput
```

```
NULL
```

:::