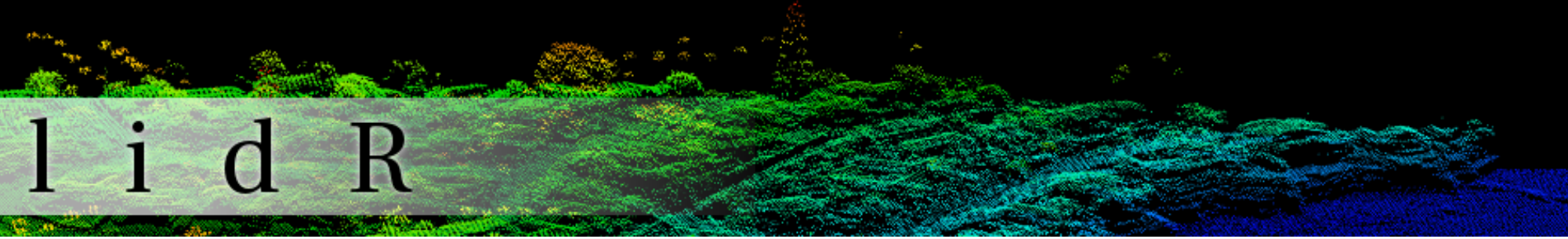


R in the Z-Dimension

Jean-Romain Roussel[a], Tina A. Cormier[b]

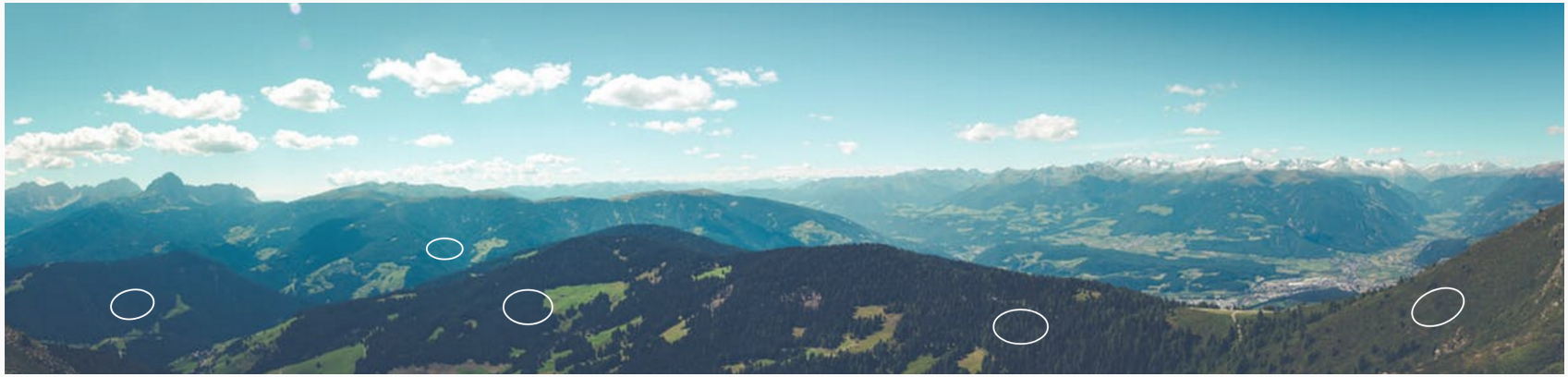
[a]Université Laval – jean-romain.roussel.1@ulaval.ca
[b]telluslabs – tina@telluslabs.com – @TinaACormier

Why use R for lidar?

Simple! Almost everything else we do is in R, so moving toward a workflow in a single environment is awesome!

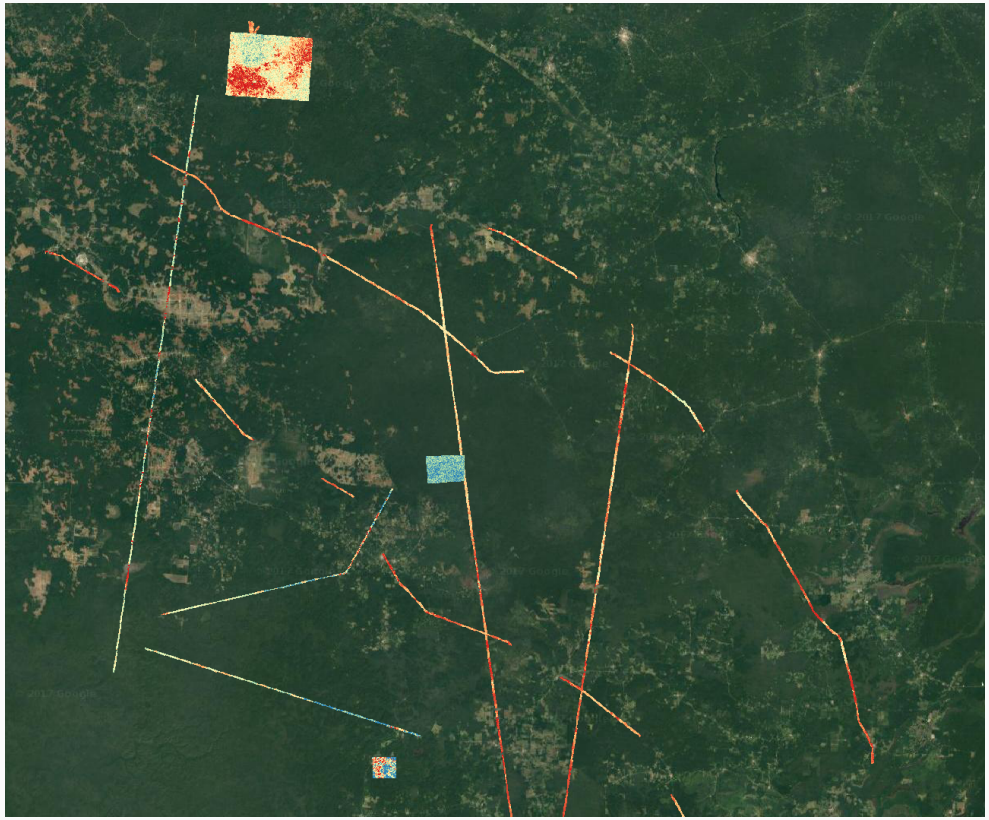
General workflow

Connect field plots $\longrightarrow$ lidar transects $\longrightarrow$ landsat images



The Goal: lidar-based biomass map

1. Acquire lidar in areas where we have field data.
2. Tile lidar to ultimate raster res of biomass map (e.g., 30 m).
3. Calculate metrics for each tile.
4. Use plot-level lidar metrics (where we know the biomass) to train a model.
5. Apply model to the rest of the tiles = map!



Process *before* R

1. 1 km las files from vendor $\longrightarrow$ 30 m las files (LAStools).
2. `las2txt` to read the lidar files into R as text files
   $\longrightarrow$ duplicated our entire data set!
3. Calculate custom metrics on the text files (R).
4. Calculate traditional metrics on the las files (system call from R to FUSION software).
5. Modeling (R).
6. Apply model to map (R) = YAY!

New R packages to directly read las files!

`rLiDAR` – 04/20/2015

- a lidar processing and visualization package.
- emphasis on identification and metrics surrounding individual trees.
- limited number of functions.
- for use on small las files.

New R packages to directly read las files!

`lidR` – 12/31/2016

- a lidar processing and (2D & 3D) visualization package.
- build and apply functions to lidar catalogs.
- build terrain models.
- normalization.
- clip lidar with various geometries.
- compute predefined and custom metrics.
- point filtering
- individual tree segmentation
- ...

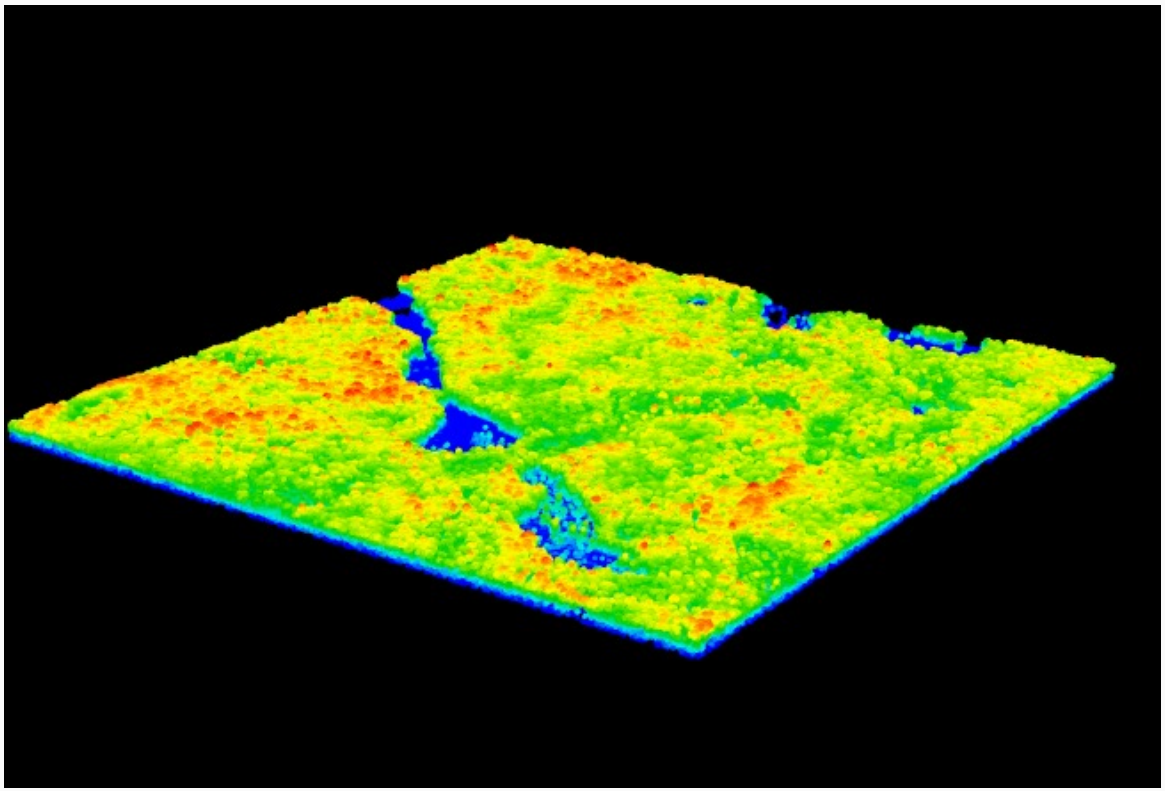
Now

I can quickly open and view each las or laz tile:

```
1 library(lidR)
2
3 las <- readLAS(lasfile)
```

Quickly inspect my tile

```
1  plot(las)
```



Check out the las header information

```
1  print(las)
```

lidar collection info

```
memory        : 8 Mb
extent        : 278200,278300,602200,602300
area          : 9996.2 m^2 (convex hull)
points        : 150395 points
pulses        : 107991 pulses
point density: 15.05 points/m^2
pulse density: 10.8 pulses/m^2
field names   : X Y Z gpstime Intensity
       ReturnNumber NumberOfReturns
       Classification ScanAngle
```

+ projection info (epsg code)

```
Variable length records:
  User ID     :  LASF_Projection
  record ID   :  34735
  Description:  LAS Georeferencing
  Tags:
     Key 1024 tiff_tag 0 count 1 offset 1
     Key 3072 tiff_tag 0 count 1 offset 32767
     Key 3075 tiff_tag 0 count 1 offset 1
     Key 3076 tiff_tag 0 count 1 offset 9001
     Key 3088 tiff_tag 34736 count 1 offset 0
     Key 3081 tiff_tag 34736 count 1 offset 1
     ...
```

+ las header info

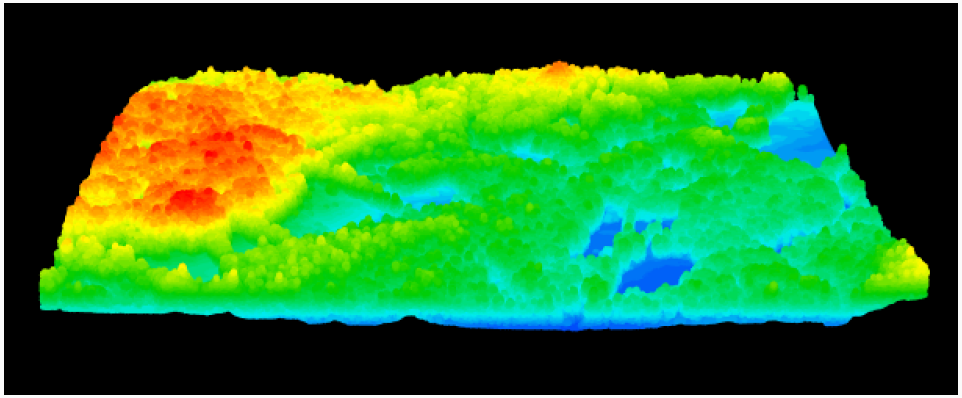
```
File signature:             LASF
File source ID:             0
Global encoding:            1
Project ID - GUID:          0
Version:                    1.2
System identifier:          LAStools (c)
Generating software:        lasview (141021)
File creation d/y:          220/2017
header size:                227
Offset to point data:       473
Num. var. length record:    2
Point data format:          1
Point data record length:   28
Num. of point records:      150395
Num. of points by return:   106930 35816 6928 690
Scale factor X Y Z:         0.01 0.01 0.01
Offset X Y Z:               0 0 0
min X Y Z:                  278200 602200 93.34
max X Y Z:                  278300 602300 123.1
```

Preview lidar values

```
1 print(las@data)
```

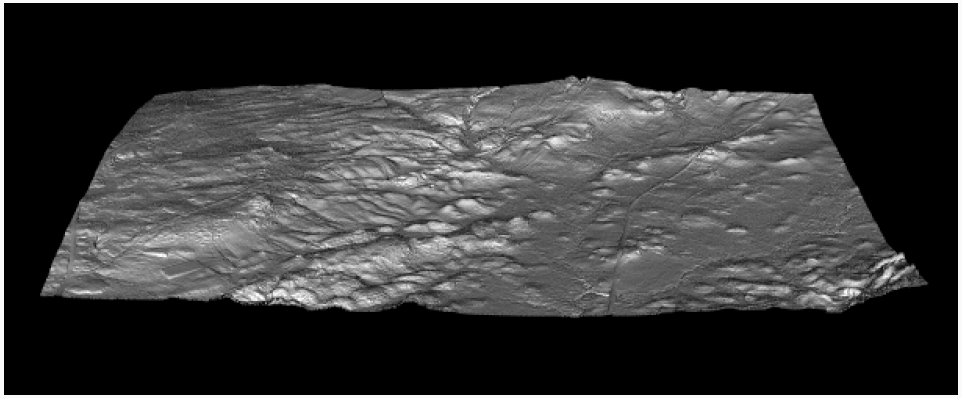
```
      X        Y      Z  gpstime Intensity ReturnNumber NumberOfReturns Classification ScanAngle
78234.3 602200.2  94.54  87558.4        12            1               1              2        21
78234.0 602200.5  94.48  87558.4        14            1               1              2        21
78232.5 602200.7  94.57  87558.4        12            1               1              1        21
78230.6 602200.2  98.38  87558.5        11            1               1              1        21
78230.7 602200.1 106.54  87558.5        11            1               1              1        22
---
78201.0 602299.9  94.21  89903.9        29            1               1              1       -24
78200.6 602299.4  94.70  89903.9        33            1               1              1       -24
78200.4 602299.4  94.01  89903.9        34            1               1              2       -24
78200.4 602299.9  94.72  89903.9        13            1               2              1       -24
78200.2 602299.7  93.98  89903.9        46            2               2              2       -24
```

Create DTM and normalize (1/2)



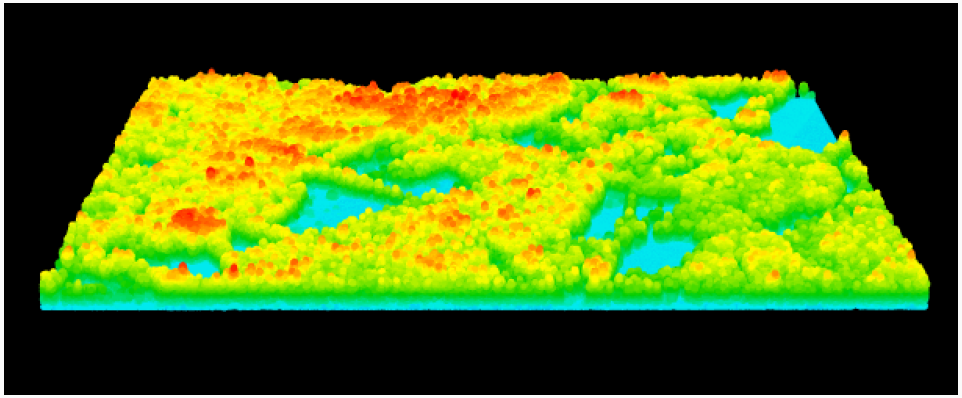
raw lidar elevation

-



ground elevation

=



height above ground

Create DTM and normalize (2/2)

Seriously? Look how easy this is.

```
# Create a terrain grid at 1 m and use it
dtm <- grid_terrain(las, res = 1, "knnidw")

# Normalize points
las.norm <- lasnormalize(las, dtm)
```

Create DTM and normalize (2/2)

Seriously? Look how easy this is.

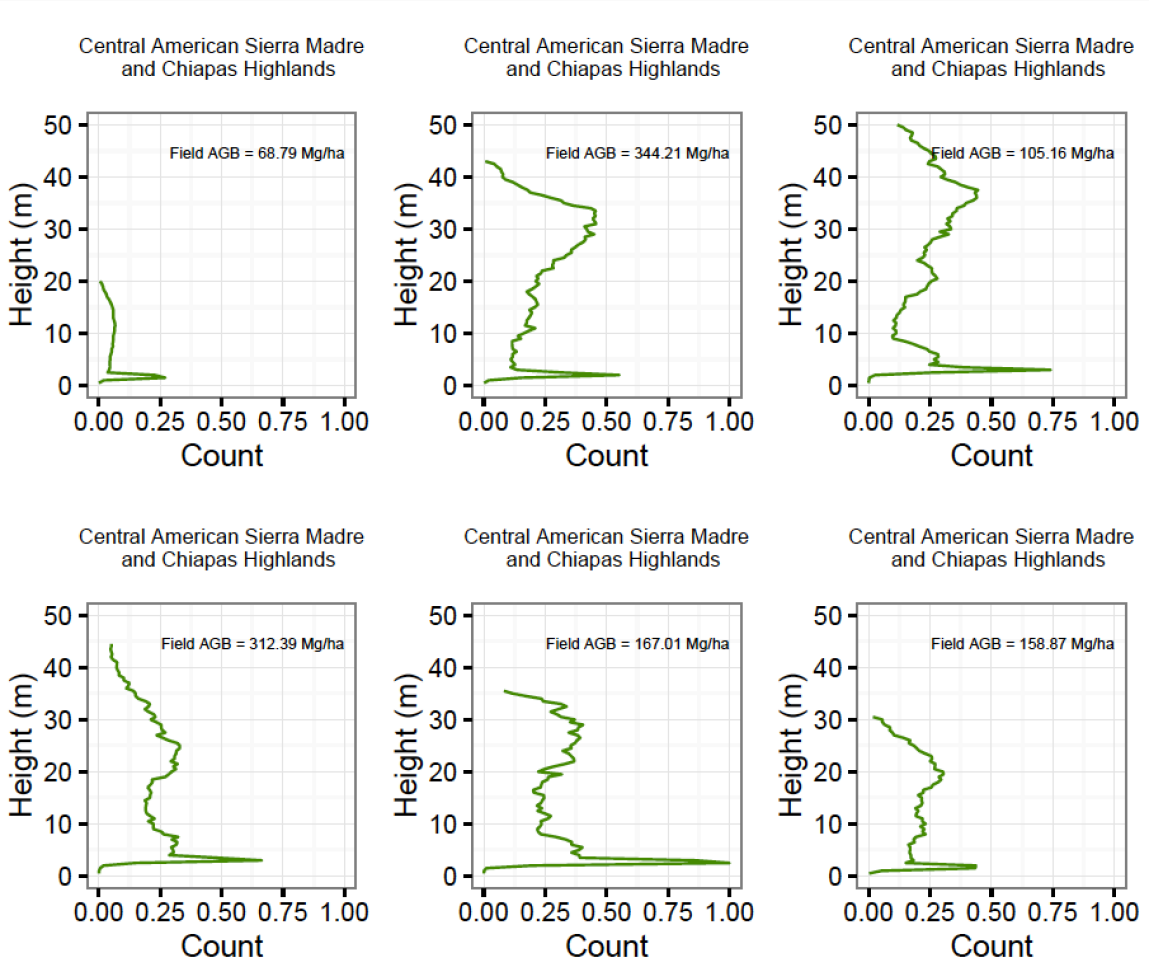
```
# Create a terrain grid at 1 m and use it
dtm <- grid_terrain(las, res = 1, "knnidw")

# Normalize points
las.norm <- lasnormalize(las, dtm)
```

I used to normalize like this – first had to build terrain model using LAStools

```
# the old way, which was slower and remenber
# requiered .txt version of all my files:
library(raster)
lasnames <- c("x", "y", "z", "i", "a", "n", "r", "c")
las <- read.csv(lasfiles, header=F, col.names=lasnames)
ptground <- extract(dtm, las[, 1:2])
las$z <- las$z - ptground
```

Now I can do fun things. . .



Like `lasclip` our field sites from the larger lidar acquisition.

Build height profiles and look at the vertical structure of our field plots.

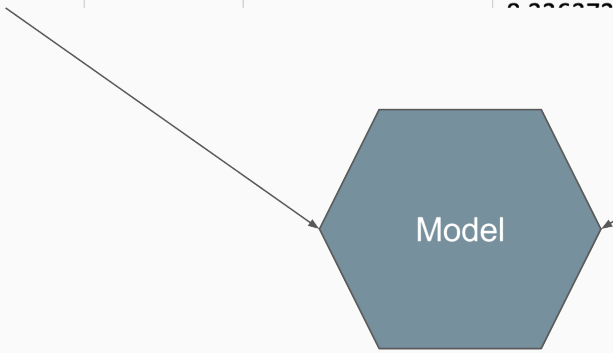
Metrics!

Field information now linked with **lidar** information

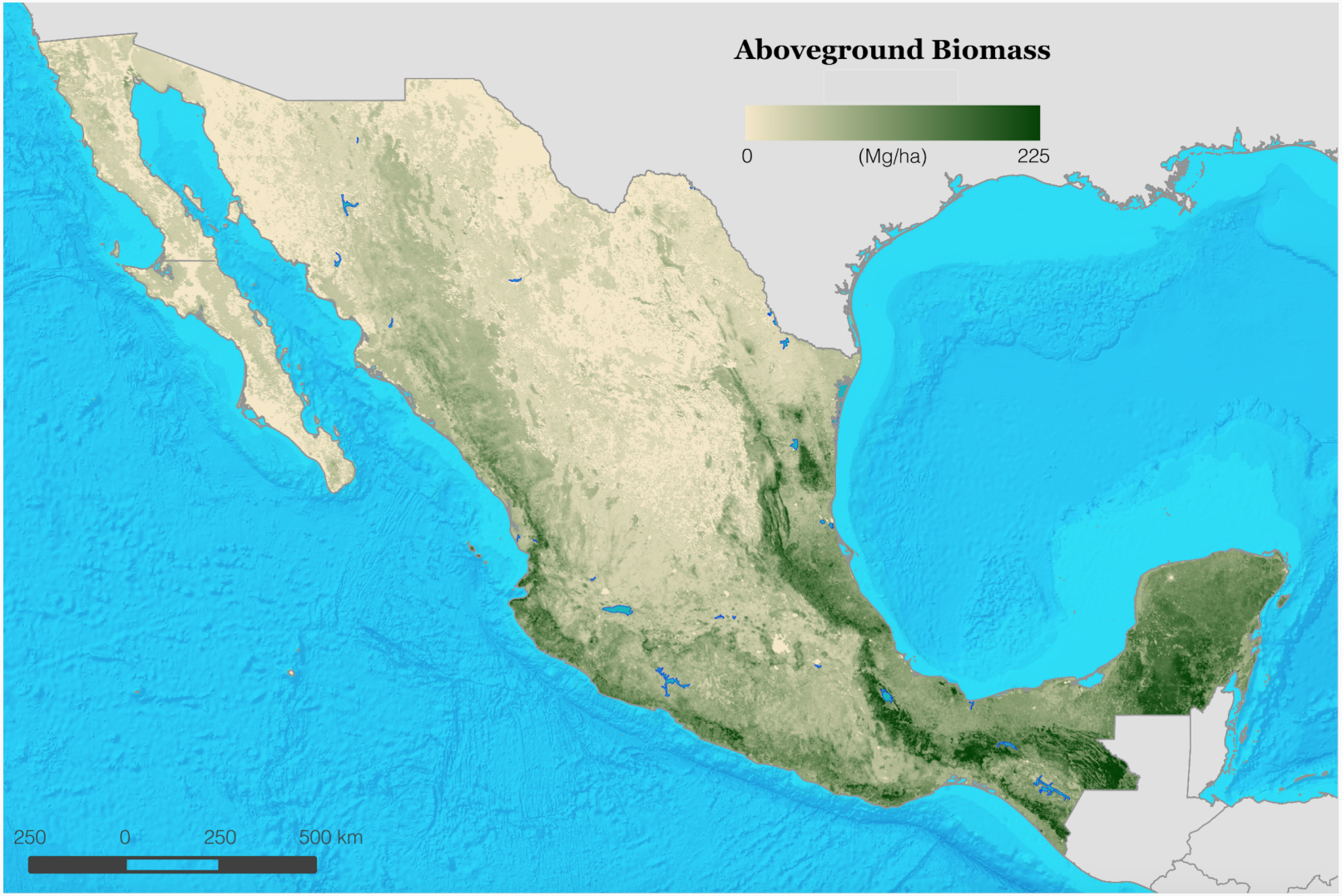
$\longrightarrow$ predict **biomass** in other locations

Plot_ID	Bio_MgHa	Meas_Year
1	136.4789	2013
2	119.9345	2013
3	86.0092	2013
4	12.2841	2013
5	9.0807	2013
6	118.0097	2013
7	152.5887	2013
8	129.1928	2013
9	33.3991	2013
10	142.8811	2013

meanh	stdevh	h25	h50	h75	mch
12.0470662	8.64147835	3.5	11	20.5	25.5
7.60447924	7.65992679	0	6	15	24.5
7.77523874	6.22875335	0	9	13	17.5
1.84864856	2.77684247	0	0	3.5	23.5
0.83739584	3.06322831	0	0	0	20.5
8.19146735	8.38690704	0	3.5	17	22.5
11.883473	7.03019035	6.5	12.5	17	32.5
9.03023903	5.79064162	3	10	14	19
7.66126516	9.88745201	0	1	14.5	31.5
8.72951288	6.83443752	1	10	15	20



Results



Wall-to-wall map of biomass!

More about the lidR package

Presented by its maintainer

Why an R package?

1. An integrated framework: no switching environments

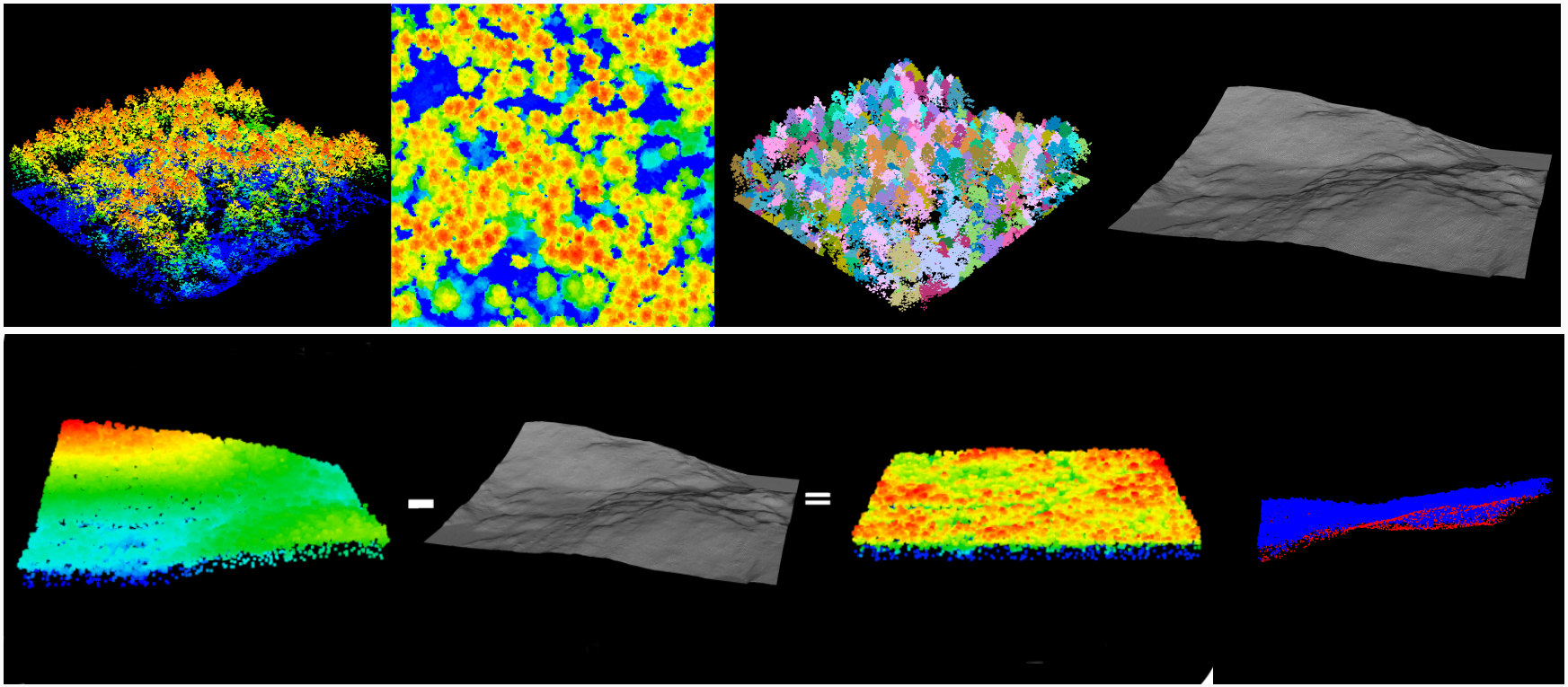
Why an R package?

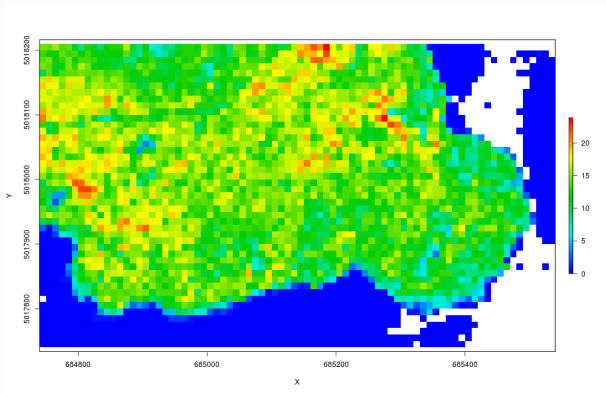
1. An integrated framework: no switching environments
2. As researchers we must use open-source tools

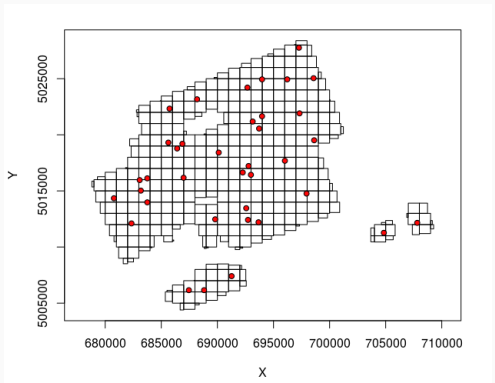
Why an R package?

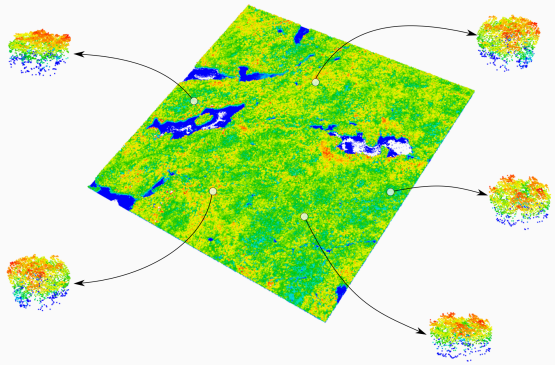
1. An integrated framework: no switching environments
2. As researchers we must use open-source tools
3. More flexible, more interactive, more tunable

Tools overview









Other tools

- Merge geographic data (raster, shapefile) at the point cloud level
- Retrieve individual flightlines
- Filters
- Colourize a point cloud
- Leaf area density, Rumple index (surface roughness), ...
- Grid point density
- Point decimation and homogenization
- Voxelization and metrics
- Streaming filters
- ...

lidR philosophy

`lidR` does not aim to provide predefined processes, it is designed to enable users to build their own.

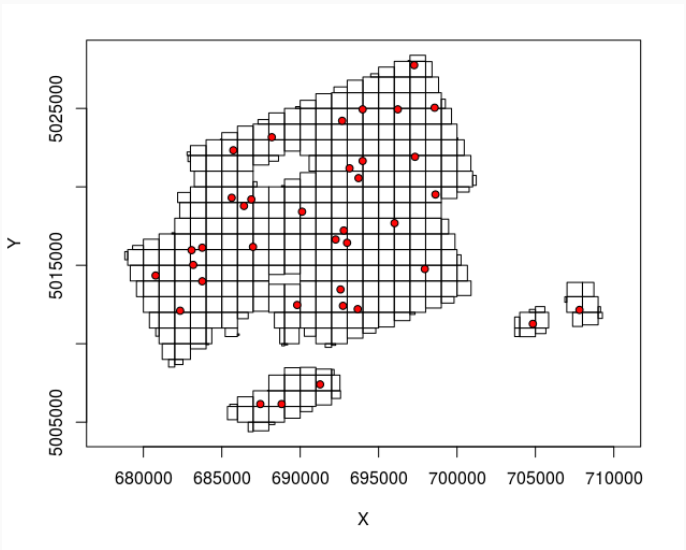
lidR philosophy (1/3) – straightforwardness

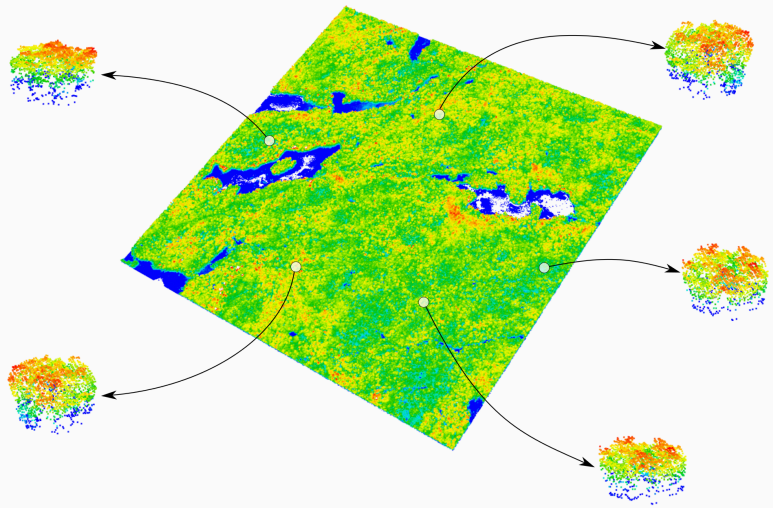
R environement is straightforward

lidR philosophy (1/3) – straightforwardness

R environement is straightforward

```
1  mycatalog = catalog("path/to/catalog")
2  coords    = read.table("inventory_coordinates.txt")
3  inventory = catalog_queries(mycatalog, coords, r = 15)
```





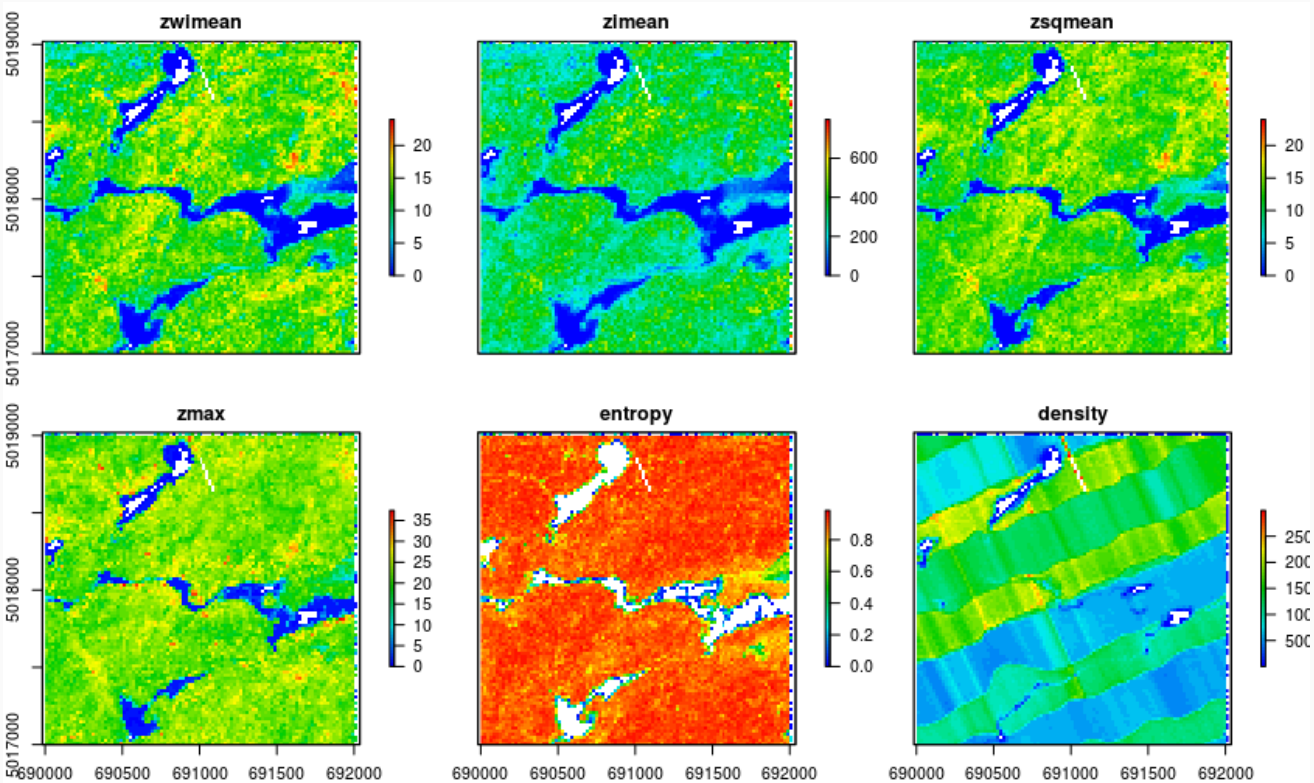
lidR philosophy (2/3) – flexibility

Flexiblility of the analyses

lidR philosophy (2/3) – flexibility

Flexiblility of the analyses

```
1  cloud  = readLAS("path/to/file.laz")
2  metric = grid_metrics(cloud, myMetric(X, Y, Z, Intensity))
3  plot(metric)
```



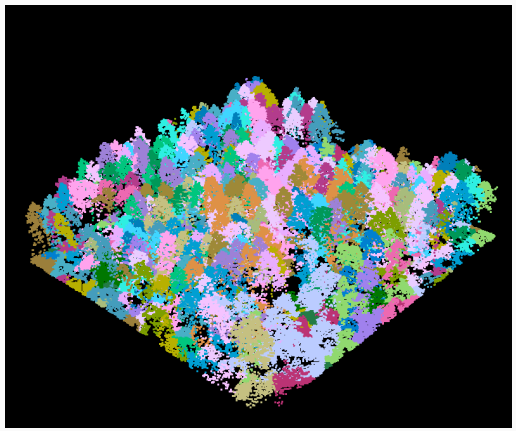
lidR philosophy (3/3) – literature

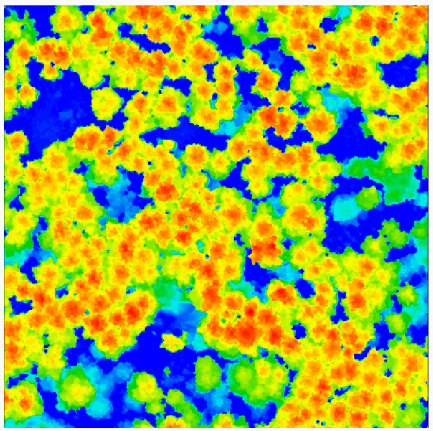
Implementation of algorithms from the literature

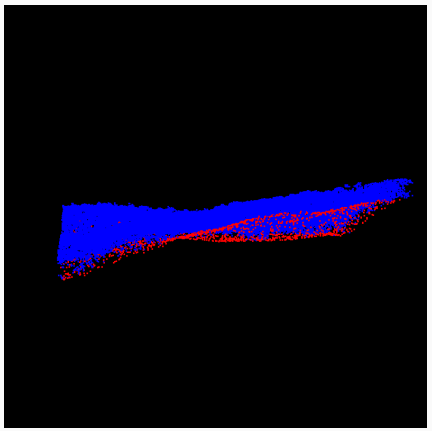
lidR philosophy (3/3) – literature

Implementation of algorithms from the literature

- Li et al. (2012) tree segmentation
- Dalponte et al. (2016) tree segmentation
- Zhang et al. (2003) groud segmentation
- Kane et al. (2010) rumple index
- Bouvier et al. (2015) leaf area density profile
- Krosvavipour et al. (2014) pit-free canopy height model
- Van Ewik et al. (2011) vertical complexity index







Cost of such ease of use

The design of such a straightforward framework is done with irreducible costs

- **speed:** many low level optimizations but cannot be as fast as dedicated software.
- **memory usage:** is high due to R itself (weak typing, weak garbage collector, ...).

Conclusions

- R is an amazing straightforward language to design tools & analyse data
  $\longrightarrow$ I want to deal with lidar data in R

- R is not the best tool to manipulate lidar data
  $\longrightarrow$ but optimizations at the C/C++ level
  $\longrightarrow$ rewrote several existing features

- An R framework to deal with lidar data
  $\longrightarrow$ With a convenient computation time.
  $\longrightarrow$ At the cost of memory usage.

- Designed to try and explore, not to process country-wide dataset (but you can).

Resources, bug reports, feature requests, colaborations

- https://github.com/Jean-Romain/lidR/
- https://github.com/Jean-Romain/lidR/wiki
- https://CRAN.R-project.org/package=lidR

Thank you for your attention.