

R workflow for ABA prediction model calibration

Jean-Matthieu Monnet

2021-07-06

The code below presents a workflow to calibrate prediction models for the estimation of forest parameters from ALS-derived metrics, using the area-based approach (ABA). The workflow is based on functions from R packages `lidaRtRee` and `lidR`.

Licence: GNU GPLv3 / Source page

Many thanks to Pascal Obst  tar for checking code and improvement suggestions.

Load data

The “Quatre Montagnes” dataset from France, prepared as described in the data preparation tutorial is loaded from the R archive files located in the folder “data/aba.model/output.”

Field data

The file “plots.rda” contains the field data, organized as a data.frame named `plots`. For subsequent use in the workflow, the data.frame should contain at least two fields: `plotId` (unique plot identifier) and a forest stand parameter. Each line in the data.frame corresponds to a field plot. A factor variable is required to calibrate stratified models. Plot coordinates are required for subsequent inference computations.

The provided data set includes one categorical variable: `stratum`, which corresponds to forest ownership, XY coordinates and three forest stand parameters :

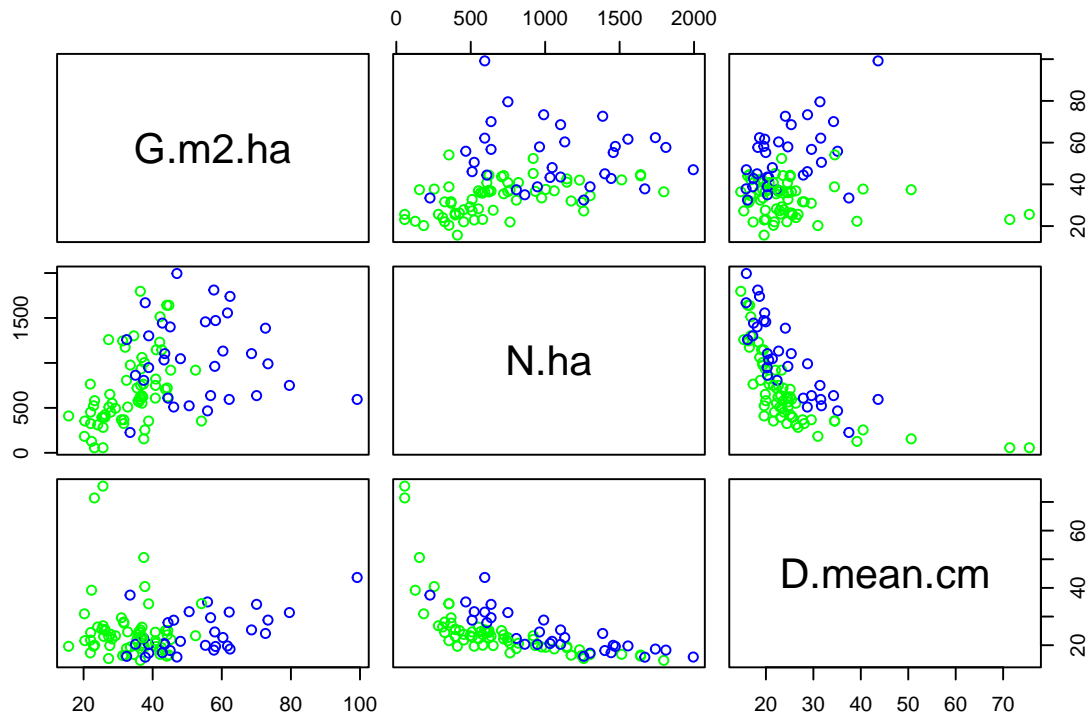
- basal area in m²/ha (`G.m2.ha`),
- stem density in /ha (`N.ha`),
- mean diameter at breast height in cm (`D.mean.cm`).

Scatterplots of stand parameters are presented below, colored by ownership (green for public forest, blue otherwise).

```
# load plot-level data
load(file = "../data/aba.model/output/plots.rda")
summary(plots)
```

##	plotId		X		Y		clusterId
##	Length:96		Min.	:895945	Min.	:6439570	Length:96
##	Class :character		1st Qu.:	896989	1st Qu.:	6443484	Class :character
##	Mode :character		Median :	899661	Median :	6451211	Mode :character
##			Mean :	899394	Mean :	6450228	
##			3rd Qu.:	900696	3rd Qu.:	6455042	
##			Max.	:903303	Max.	:6459820	
##	G.m2.ha		N.ha		D.mean.cm		stratum
##	Min.	:15.67	Min.	: 56.59	Min.	:14.70	private:32
##	1st Qu.:	31.22	1st Qu.:	488.08	1st Qu.:	19.55	public :64
##	Median :	37.35	Median :	714.43	Median :	22.65	
##	Mean :	40.17	Mean :	810.07	Mean :	24.78	
##	3rd Qu.:	44.78	3rd Qu.:	1110.55	3rd Qu.:	26.31	
##	Max.	:99.18	Max.	:1994.74	Max.	:75.53	

```
# display forest variables
plot(plots[, c("G.m2.ha", "N.ha", "D.mean.cm")], col = ifelse(plots$stratum == "public",
  "green", "blue"))
```



ALS data

Normalized ALS point clouds extracted over each plot, as well as terrain statistics previously computed from the ALS ground points can also be prepared according to the data preparation tutorial. Point clouds corresponding to each field plot are organized in a list of LAS objects. Meta data of one LAS object are displayed below.

```
# list of LAS objects: normalized point clouds inside plot extent
load("../data/aba.model/output/las.height.rda")
# display one point cloud # lidR::plot(lasn[[1]])
llas.height[[1]]
```

```
## class      : LAS (v1.2 format 1)
## memory     : 1.3 Mb
## extent     : 898185.8, 898225.7, 6455688, 6455728 (xmin, xmax, ymin, ymax)
## coord. ref. : RGF93 / Lambert-93
## area       : 1248.7 m2
## points     : 15.6 thousand points
## density    : 12.51 points/m2
```

The first lines of the terrain statistics are displayed hereafter.

```
##      altitude azimuth.gr slope.gr
## Verc-01-1 1096.9      99.1    20.1
## Verc-01-2 1083.4      94.0    16.1
## Verc-01-3 1120.7     104.0    22.9
```

The following lines ensure that the plots are ordered in the same way in the three data objects.

```
l1as.height <- l1as.height[plots$plotId]
metrics.terrain <- metrics.terrain[plots$plotId, ]
```

ALS metrics computation

Two types of metrics can be computed.

- Point cloud metrics are directly computed from the point cloud or from the derived surface model on the whole plot extent. These are the metrics generally used in the area-based approach.
- Tree metrics are computed from the characteristics of trees detected in the point cloud (or in the derived surface model). They are more CPU-intensive to compute and require ALS data with higher density, but in some cases they allow a slight improvement in models prediction accuracy.

Point cloud metrics

Point cloud metrics are computed with the function `lidaRtRee::cloudMetrics`, which applies the `lidR::cloud_metrics` to all point clouds in the list. Default computed metrics are those proposed by the function `lidR::stdmetrics`. Additional metrics are available with the function `lidaRtRee::ABamodelMetrics`.

```
# define function for later use
aba.pointMetricsFUN <- ~lidaRtRee::ABamodelMetrics(Z, Intensity, ReturnNumber, Classification,
2)
# apply function on each point cloud in list
metrics.points <- lidaRtRee::cloudMetrics(l1as.height, aba.pointMetricsFUN)
round(head(metrics.points[, 1:8], n = 3), 2)
```

```
##           zmax zmean  zsd zskew zkurt zentropy pzabovemean pzabov2
## Verc-01-1 33.87 16.99 8.21 -0.35 1.80    0.93    57.78    99.98
## Verc-01-2 30.54 16.79 5.66 -0.44 2.70    0.90    54.00    99.99
## Verc-01-3 29.89 18.64 5.15 -0.65 3.33    0.88    54.50    99.99
```

Tree metrics

Tree metrics rely on a preliminary detection of trees, which is performed with the `lidaRtRee::treeSegmentation` function. For more details, please refer to the tree detection tutorial. Tree segmentation requires point clouds or canopy height models with an additional buffer in order to avoid border effects when computing tree characteristics. Once trees are detected, metrics are derived with the function `lidaRtRee::stdTreeMetrics`. A user-specific function can be specified to compute other metrics from the features of detected trees. Plot radius has to be specified as it is required to exclude trees detected outside of the plot, and to compute the plot surface. Tree segmentation is not relevant when the point cloud density is too low, typically below five points per m². The function first computes a canopy height model which default resolution is 0.5 m, but this should be set to 1 m with low point densities.

```
# resolution of canopy height model (m)
aba.resCHM <- 0.5
# specify plot radius to exclude trees located outside plots
plot.radius <- 15
# compute tree metrics
metrics.tree <- lidaRtRee::cloudTreeMetrics(l1as.height, plots[, c("X", "Y")], plot.radius,
res = aba.resCHM, func = function(x) {
  lidaRtRee::stdTreeMetrics(x, area.ha = pi * plot.radius^2/10000)
})
round(head(metrics.tree[, 1:5], n = 3), 2)
```

```
##           Tree.meanH Tree.sdH Tree.giniH Tree.density TreeInf10.density
## Verc-01-1    27.11    6.39    0.11    226.35    14.15
## Verc-01-2    25.80    3.27    0.06    268.80    0.00
## Verc-01-3    25.74    3.88    0.08    268.80    0.00
```

Other metrics

In case terrain metrics have been computed from the cloud of ground points only, they can also be added as variables, and so do other environmental variables which might be relevant in modeling.

```
metrics <- cbind(metrics.points[plots$plotId, ], metrics.tree[plots$plotId, ], metrics.terrain[plots$plotId, 1:3])
```

Model calibration

Calibration for a single variable

Once a dependent variable (forest parameter of interest) has been chosen, the function `lidaRtRee::ABAmoDel` is used to select the linear regression model that yields the highest adjusted- R^2 with a defined number of independent variables, while checking linear model assumptions. A Box-Cox transformation of the dependent variable can be applied to normalize its distribution, or a log transformation of all variables (parameter `transform`). Model details and cross-validation statistics are available from the returned object.

```
variable <- "G.m2.ha"
# no subsample in this case
subsample <- 1:nrow(plots)
# model calibration
model.ABA <- lidaRtRee::ABAmoDel(plots[subsample, variable], metrics[subsample, ],
  transform = "boxcox", nmax = 4, xy = plots[subsample, c("X", "Y")])
```

```
## Reordering variables and trying again:
```

```
# renames outputs with variable name
row.names(model.ABA$stats) <- variable
# display selected linear regression model
model.ABA$model
```

```
##
```

```
## Call:
```

```
## stats::lm(formula = dep.var ~ Tree.density + TreeCanopy.meanH +
```

```
##   TreeCanopy.coverInPlot, data = df.transform)
```

```
##
```

```
## Coefficients:
```

```
##           (Intercept)           Tree.density           TreeCanopy.meanH
```

```
##           1.7867460             0.0005366             0.0240784
```

```
## TreeCanopy.coverInPlot
```

```
##           0.5459090
```

```
# display calibration and validation statistics
```

```
model.ABA$stats
```

```
##           n           formula      adjR2
```

```
## G.m2.ha 96 Tree.density + TreeCanopy.meanH + TreeCanopy.coverInPlot 0.6896275
```

```
##           transform      lambda      rmse      cvrmse      pwil      pttest      paov
```

```
## G.m2.ha   boxcox -0.1437872 8.303605 0.2066882 0.7134264 0.9925296 0.9966434
```

```
##           cor      looR2      var.res
```

```
## G.m2.ha 0.8193955 0.6708864 0.01283462
```

The function computes values predicted in leave-one-out cross-validation, by using the same combination of dependent variables and fitting the regression coefficients with all observations except one. Predicted values can be plotted against field values with the function `lidaRtRee::ABAmoDelPlot`. It is also informative to check the correlation of prediction errors with other forest or environmental variables.

In this example, only tree metrics are selected in the basal area prediction model. The model seems to fail to predict large values. The prediction errors are positively correlated with basal area because large values are under-estimated.

```

# check correlation between errors and other variables
round(cor(cbind(model.ABA$values$residual, plots[subsample, c("G.m2.ha", "N.ha",
"D.mean.cm")], metrics.terrain[subsample, 1:3])), 2)[1, ]

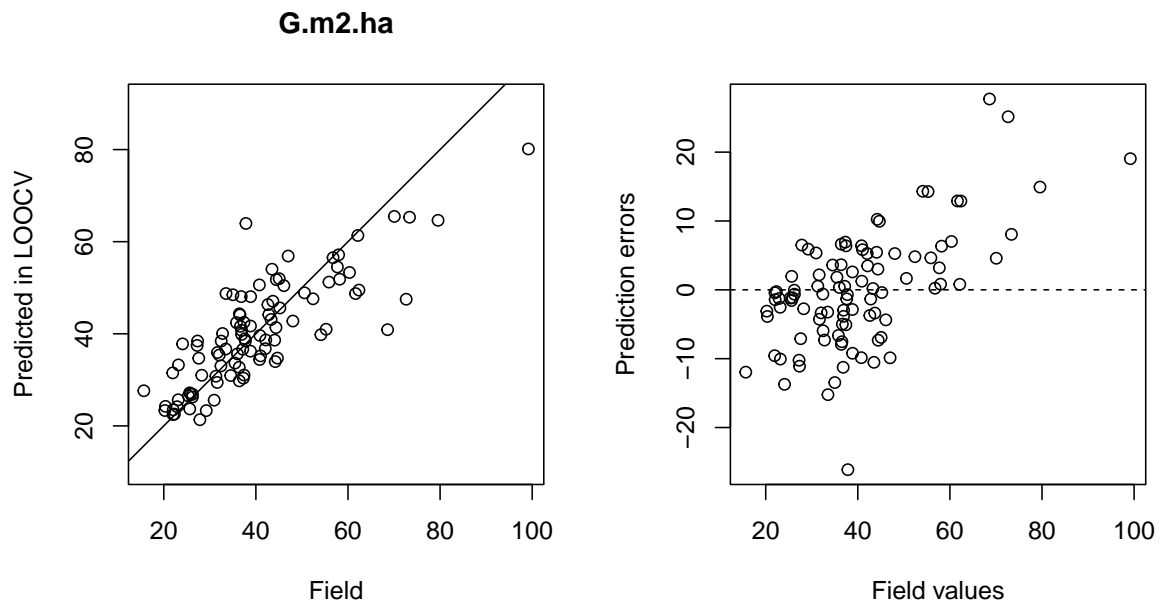
## model.ABA$values$residual      G.m2.ha      N.ha
##              1.00              0.61      0.09
##           D.mean.cm      altitude      azimuth.gr
##              0.16              0.10      -0.06
##           slope.gr
##           -0.01

# significance of correlation value
cor.test(model.ABA$values$residual, plots[subsample, variable])

##
## Pearson's product-moment correlation
##
## data: model.ABA$values$residual and plots[subsample, variable]
## t = 7.3748, df = 94, p-value = 6.396e-11
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.4608659 0.7186695
## sample estimates:
##      cor
## 0.6054134

# plot predicted VS field values
par(mfrow = c(1, 2))
lidaRtRee::ABAModelPlot(model.ABA, main = variable)
plot(plots[subsample, c("G.m2.ha")], model.ABA$values$residual, ylab = "Prediction errors",
     xlab = "Field values")
abline(h = 0, lty = 2)

```



In case only point cloud metrics are used as potential inputs, the errors are hardly better distributed. Coloring points by ownership shows that plots located in private forests have the largest basal area values which tend to be under-estimated.

```

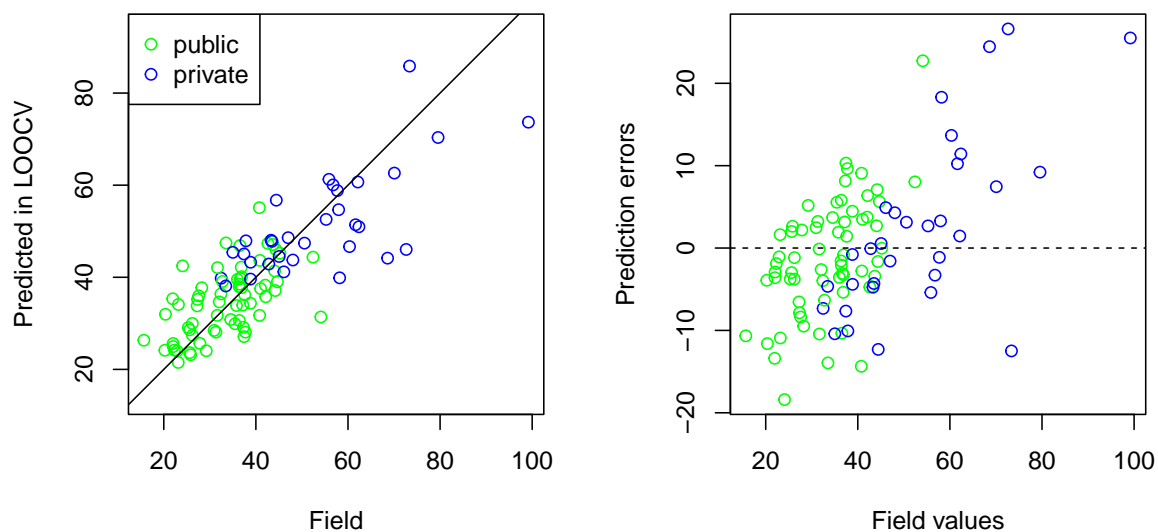
model.ABA.metrics.points <- lidaRtRee::ABAModel(plots[subsample, variable], metrics.points[subsample
], transform = "boxcox", nmax = 4, xy = plots[subsample, c("X", "Y")])
# renames outputs
row.names(model.ABA.metrics.points$stats) <- names(model.ABA.metrics.points$model) <- variable
# model.ABA.metrics.points$model[[variable]]
model.ABA.metrics.points$stats

##              n              formula      adjR2 transform      lambda
## G.m2.ha 96 pzabovemean + zpcum8 + p.1st.hmin 0.6519446 boxcox -0.1437872
##              rmse      cvrmse      pwil      pttest      paov      cor      looR2
## G.m2.ha 8.517813 0.2120202 0.5048285 0.9790591 0.9904006 0.808511 0.653687
##              var.res
## G.m2.ha 0.0143929

# cor.test(model.ABA.metrics.points$values$residual, plots[subsample,
# variable])
par(mfrow = c(1, 2))
# plot predicted VS field values
lidaRtRee::ABAModelPlot(model.ABA.metrics.points, main = variable, col = ifelse(plots$stratum ==
"public", "green", "blue"))
legend("topleft", c("public", "private"), col = c("green", "blue"), pch = 1)
plot(plots[subsample, c("G.m2.ha")], model.ABA.metrics.points$values$residual, ylab = "Prediction er
xlab = "Field values", col = ifelse(plots$stratum == "public", "green", "blue"))
abline(h = 0, lty = 2)

```

G.m2.ha



Calibration for several variables

The following code calibrates models for several forest parameters. In case different transformations have to be performed on the parameters, models have to be calibrated one by one.

```

models.ABA <- list()
for (i in c("G.m2.ha", "D.mean.cm", "N.ha")) {
  models.ABA[[i]] <- lidaRtRee::ABAModel(plots[, i], metrics, transform = "boxcox",
    nmax = 4, xy = plots[, c("X", "Y")])
}

```

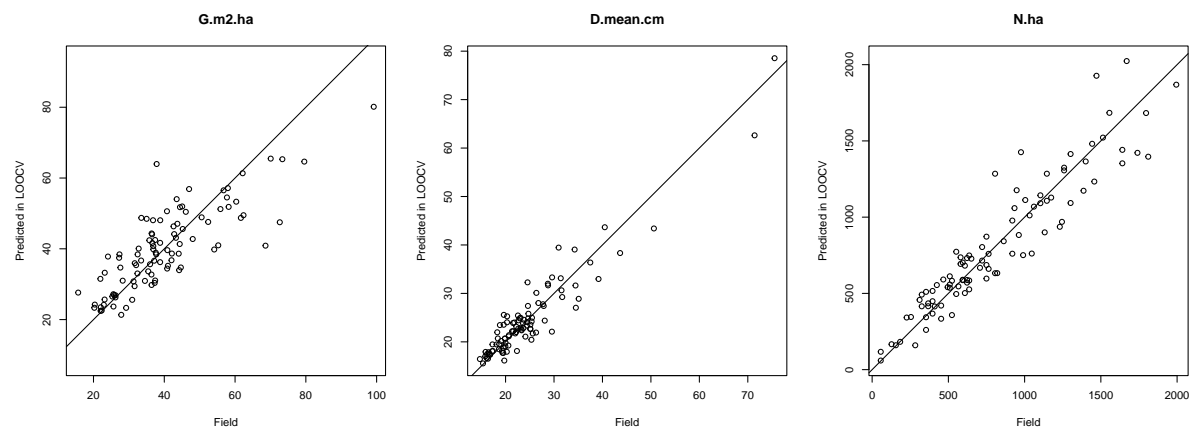
Reordering variables and trying again:

```
## Reordering variables and trying again:
## Reordering variables and trying again:
# bind model stats in a data.frame
model.stats <- do.call(rbind, lapply(models.ABA, function(x) {
  x[["stats"]]
})))
```

The obtained models are presented below. The table columns correspond to:

- **n** number of plots,
- **metrics** selected in the model,
- **adj-R2.%** adjusted R-squared of fitted model (%),
- **CV-R2.%** coefficient of determination of values predicted in cross-validation (CV) VS field values (%),
- **CV-RMSE.%** coefficient of variation of the Root Mean Square Errors of prediction in CV (%),
- **CV-RMSE** Root Mean Square Error of prediction in CV.

	n	metrics	adj-R2.%	CV-R2.%	CV-RMSE.%	CV-RMSE
G.m2.ha	96	Tree.density + TreeCanopy.meanH + TreeCanopy.coverInPlot	69.0	67.1	20.7	8.3
D.mean.cm	96	ipcunzq70 + Tree.density + TreeCanopy.meanH + altitude	82.0	89.5	12.6	3.1
N.ha	96	zentropy + mCH + p.1st.hmin + TreeSup10.density	90.5	87.5	19.6	158.4



Stratified models

Motivation

When calibrating a statistical relationship between forest stand parameters, which are usually derived from diameter measurements, and ALS metrics, one relies on the hypothesis that the interaction of laser pulses with the leaves and branches structure is constant on the whole area. However, differences can be expected either due to variations in acquisition settings (flight parameters, scanner model), in forests (stand structure and composition) or in topography (slope). Better models might be obtained when calibrating stratum-specific relationships, provided each stratum is more homogeneous regarding the laser / vegetation interaction. A trade-off has to be achieved between the within-strata homogeneity and the number of available plots for calibration in each stratum. A minimum number of plots is approximately 50, while 100 would be recommended. In this example we hypothesize that ownership reflects both structure and composition differences in forest stands.

Calibration of stratum-specific models

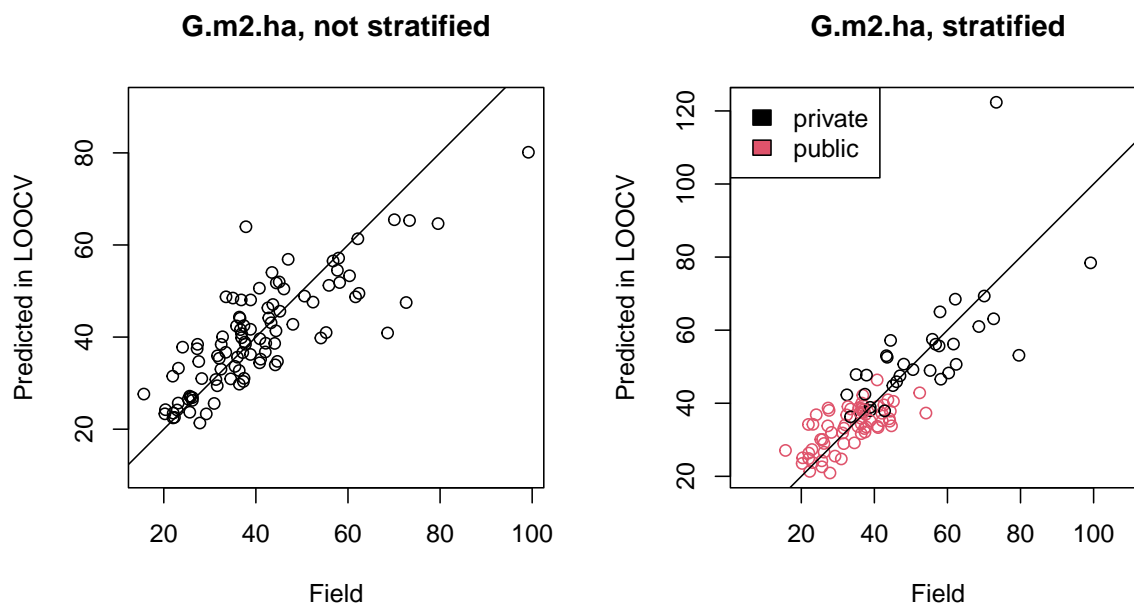
Stratum-specific models are computed and stored in a list during a `for` loop. The function `lidaRtRee::ABAModelCombineStrata` then combines the list of models corresponding to each stratum to compute aggregated statistics for all plots, making it easier to compare stratified with non-stratified models.

In this example, the model for “private” yields a large error on the plot “Verc-C5-1,” which considerably lowers the accuracy of the stratified approach.

```
# stratification variable
strat <- "stratum"
# create list of models
model.ABA.stratified <- list()
# calibrate each stratum model
for (i in levels(plots[, strat])) {
  subsample <- which(plots[, strat] == i)
  if (length(subsample) > 0) {
    model.ABA.stratified[[i]] <- lidaRtRee::ABAModel(plots[subsample, variable],
      metrics[subsample, ], transform = "boxcox", nmax = 4, xy = plots[subsample,
        c("X", "Y")])
  }
}

## Reordering variables and trying again:
# backup list of models for later use
model.ABA.stratified.boxcox <- model.ABA.stratified
# combine list of models into single object
model.ABA.stratified <- lidaRtRee::ABAModelCombineStrata(model.ABA.stratified, plots$plotId)
# model.ABA.stratified$stats
```

	n	metrics	adj- R2.%	CV- R2.%	CV- RMSE.%	CV- RMSE
NOT-STRATIFIED	96	Tree.density + TreeCanopy.meanH + TreeCanopy.coverInPlot	69.0	67.1	20.7	8.3
private	32	zpcum7 + ikurt + p.1st.hmin	62.8	29.2	23.4	12.5
public	64	Tree.meanH + TreeCanopy.coverInPlot	49.1	45.8	18.1	6.0
COMBINED	96	NA	NA	63.4	21.8	8.8



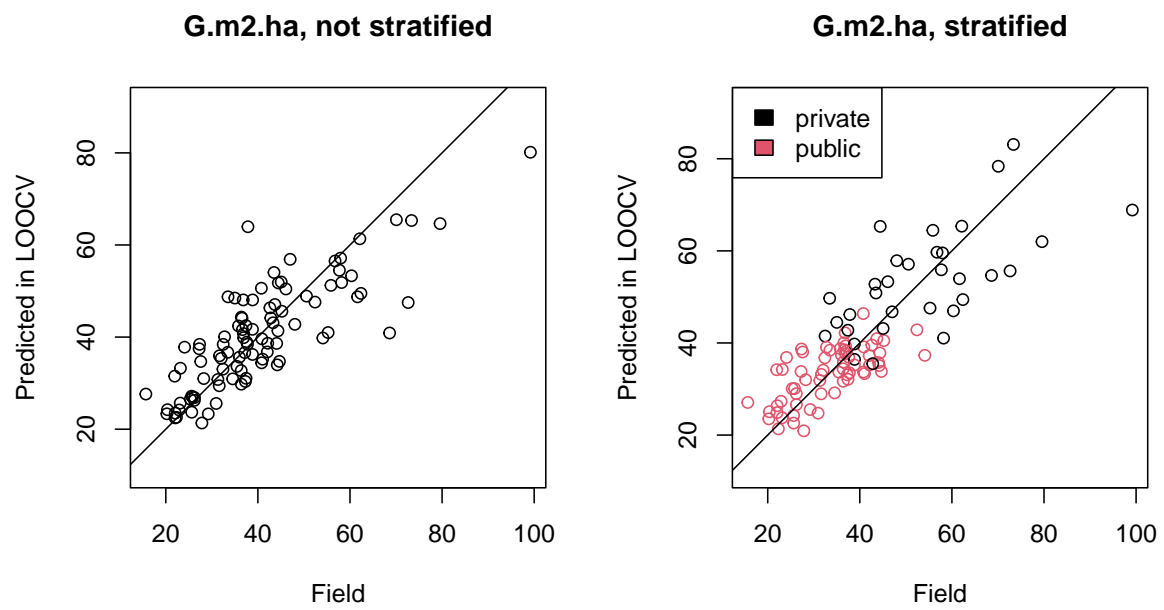
Stratified models with stratum-specific variable tranformations

In case one wants to apply different variable transformations, or use different subsets of ALS metrics depending on the strata, the following example can be used. First models using only the point cloud metrics are calibrated without transformation of the data. The statistics for all plots are then calculated by combining the following stratum-specific models :

- public ownership, all metrics, Box-Cox transformation of basal area values (calibrated in the previous paragraph),
- private ownership, only point cloud metrics, no data transformation.

```
# create list of models for no transformation
model.ABA.stratified.none <- list()
# calibrate each stratum model
for (i in levels(plots[, strat])) {
  subsample <- which(plots[, strat] == i)
  if (length(subsample) > 0) {
    model.ABA.stratified.none[[i]] <- lidaRtRee::ABAModel(plots[subsample, variable],
      metrics.points[subsample, ], transform = "none", xy = plots[subsample,
        c("X", "Y")])
  }
}
# combine list of models into single object
model.ABA.stratified.mixed <- lidaRtRee::ABAModelCombineStrata(list(private = model.ABA.stratified.none,
  public = model.ABA.stratified.boxcox[["public"]]), plots$plotId)
# bind model stats in a data.frame for comparison
model.stats <- rbind(model.ABA$stats, model.ABA.stratified.mixed$stats)
row.names(model.stats)[1] <- "NOT.STRATIFIED"
```

	n	metrics	transform	adj- R2.%	CV- R2.%	CV- RMSE.%	CV- RMSE
NOT.STRATIFIED	96	Dee.density + TreeCanopy.meanH + TreeCanopy.coverInPlot	boxcox	69.0	67.1	20.7	8.3
private	32	zpcum7 + ikurt	none	50.8	42.3	21.1	11.3
public	64	Tree.meanH + TreeCanopy.coverInPlot	boxcox	49.1	45.8	18.1	6.0
COMBINED	96	NA	NA	NA	68.0	20.4	8.2



Save data before next tutorial

The following lines save the data required for the area-based mapping step.

```
save(model.ABA.stratified.mixed, model.ABA, aba.pointMetricsFUN, aba.resCHM, file = "../data/aba.mod
```