Random forests for time series

Benjamin Goehry, Hui Yan, Yannig Goude, Pascal Massart, Jean-Michel Poggi

EDF Lab & Univ. Paris-Sud

Table of contents

Standard random forests

Context

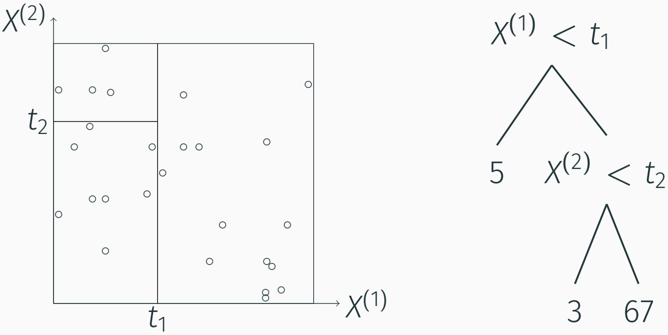
We have some stationary data set $\mathcal{D}_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$, $(X_i, Y_i) \in \mathbb{R}^p \times \mathbb{R}$ and

$$Y = f(X) + \epsilon$$

Goal: estimate the regression function f.

How: random forests

Random forest? Regression tree



A partitioning of $[0,1]^2$ and the associated binary tree.

Breiman's random forest

Parameters: number of trees M, number of observations per tree α_n, size of the random set of variables m_{try}

Repeat for each tree:

- Draw randomly $\alpha_n \leq n$ points among the n points with or without replacement.
- Repeat recursively at each node:
  - choose a random set of m_{try} variables among the p variables and apply the CART criterion on this subset.
  - Cut on the best split.

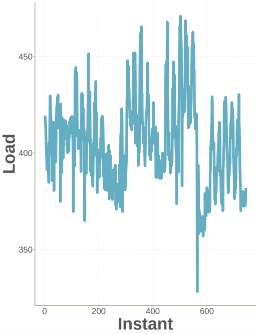
Breiman L. Random forests. 2001.

Key step: bootstrapping

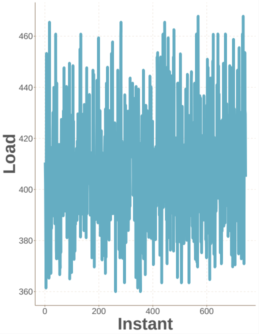
Randomly drawing $\alpha_n \leq n$ observations with replacement.

> Pros: adapted to i.i.d observations.
> Cons: destroys the underlying structure.

Example:



Original load.



Bootstraped load.

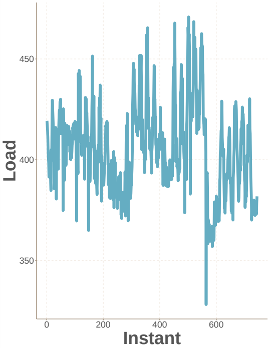
Adaptation to time series

Solution: Block bootstrap

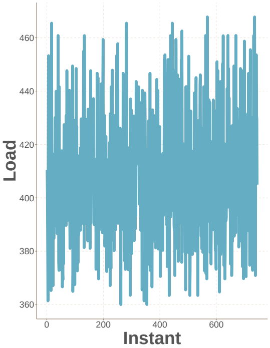
Replace the standard bootstrap with a block bootstrap variant to subsample time series during the tree construction phase

➔ Dependence structure preserved.

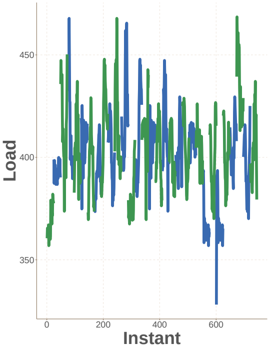
Example:



Original load.



Bootstrapped load.



24h block bootstrapped

New algorithm

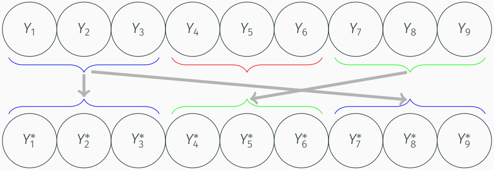
Parameters: number of trees M, number of observations per tree α_n, size of the random set of variables m_{try}, block size l_n

Repeat for each tree:

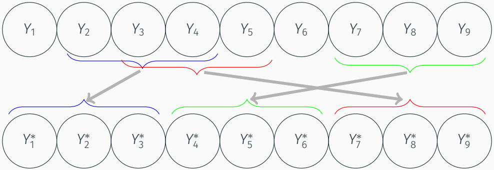
- Draw $\alpha_n \leq n$ observations using a block bootstrap variant with parameter l_n.
- Repeat recursively at each node:
  - chose a random set of m_{try} variables among the p variables and apply the CART criterion on this subset.
  - Cut on the best split.

Block bootstrap variants

Non-overlapping block bootstrap[1]



Moving block bootstrap[2]



[1] E. Carlstein. *The use of subseries values for estimating the variance of a general statistic from a stationary sequence.* 1986.

[2] H.R. Kunsch, *The jackknife and the bootstrap for general stationary observations.* 1989.
R.Y. Liu, et al. *Moving blocks jackknife and bootstrap capture weak dependence.* 1992.

Software aspect

Existing packages for trees/RF: party, rpart, randomForest, ranger[3], etc.

We propose an extension of *ranger* called rangerts.

New code parameters:

- bootstrap.ts: "circular", "moving", "non-overlap" (and others)
- block.size: number of consecutive observations per block
- *by.end:* build blocks by the end of the series or not
- *period:* seasonality period (only for seasonal variant)

Code example:

```
forest_ts ← ranger(Y ∼ ., data, bootstrap.ts = "moving", block.size = l_n)
forecast_ts ← predict(forest_ts, data_test)$prediction
```

[3] Wright, M. N., Ziegler, A. *ranger: A fast implementation of random forests for high dimensional data in C++ and R.* 2017.

Application to load forecasting & Conclusion

Dataset, goal & model

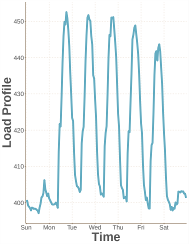
Dataset: load of a building called *UnivLab Patrick*[4]. One observation per hour over one year. Access to the temperature and schedule.

Training January-October, validation November, test December.

Goal: Load forecasting at a 24 hour horizon Y_t.

Predictor variables:

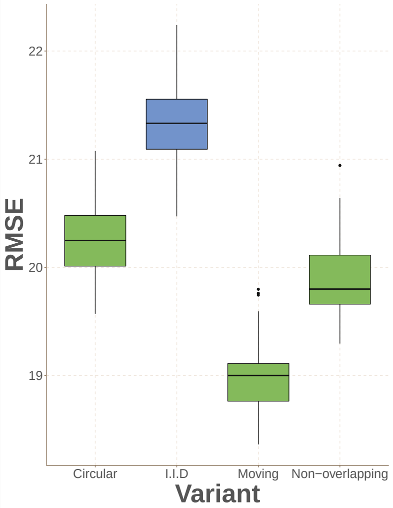
- Y_{t-24} & $Y_{t-7\times24}$;
- Temp_t;
- Schedule_t;
- Hour_t, InstantWeek_t, DayType_t, Toy_t.



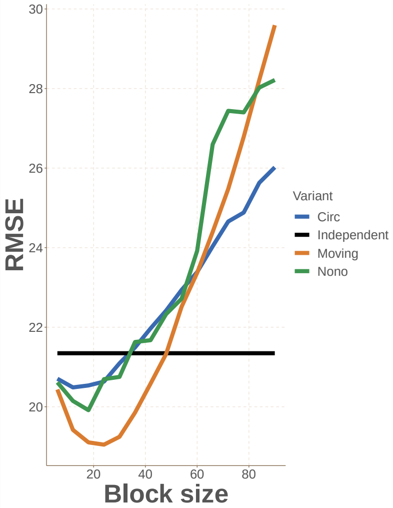
Weekly load profile

[4] C. Miller, F. Meggers. The building data genome project: An open, public data set from non-residential building electrical meters. 2017

Comparison to the standard random forest



Performance of the variants for $m_{try} = 2$.



Evolution of the performance for each variant according the block length l_n.

Take-home message

- ☞ Introduced a new way to incorporate the dependence structure in random forests.
- ☞ Improve the performance over the standard random forests.
- ☞ Variable importance can also be redefined.

References

- L. Breiman. Random forests. *Machine learning*, 2001.
- E. Carlstein. The use of subseries values for estimating the variance of a general statistic from a stationary sequence. *Ann. Statist.*, 1986.
- H.R. Kunsch. The jackknife and the bootstrap for general stationary observations. *Ann. Statist.*, 1989.
- R.Y. Liu, et al. Moving blocks jackknife and bootstrap capture weak dependence. *Wiley, New York*, 1992.
- C. Miller, F. Meggers, The building data genome project: An open, public data set from non-residential building electrical meters. *Energy Procedia*, 2017.
- D.N Politis, J. Romano, A circular block-resampling procedure for stationary data. *Wiley, New York*, 1992.
- Wright, M. N., Ziegler, A. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *J Stat Softw*, 2017.

Variable importance, a new way to define it

Random forests can be used to compute the variable importance.

Mean Decrease Accuracy: if a variable is not important, then permuting its value should not change prediction accuracy. The importance of the variable $X^{(j)}$ is defined by

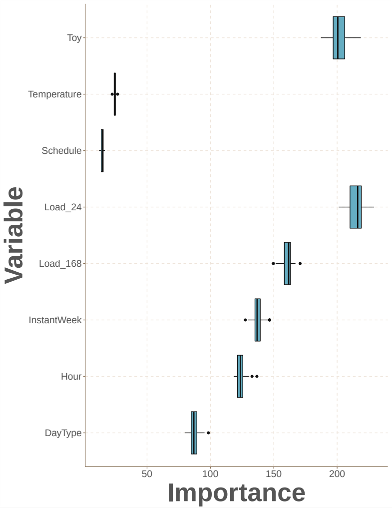
$$VI\left(X^{(j)}\right) = \frac{1}{M}\sum_{m=1}^{M}\left(\underbrace{\widetilde{errOOB_m^j}}_{\text{OOB error after } X^{(j)} \text{ permuted}} - \underbrace{errOOB_m}_{\text{OOB error for } m\text{th tree}}\right).$$

New definition:

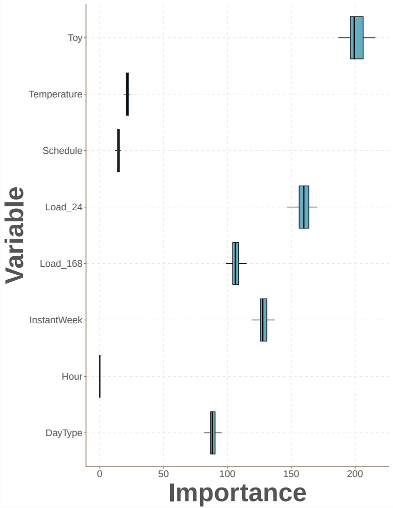
$$VI\left(X^{(j)}\right) = \frac{1}{M}\sum_{m=1}^{M}\left(\underbrace{\overline{errOOB_m^j}}_{\text{OOB error after blocks of length } l_n \text{ are permuted for } X^{(j)}} - errOOB_m\right).$$

Breiman L. *Random forests.* 2001.

Standard variable importance *vs* Block variable importance



Non-overlapping standard variable importance



Non-overlapping block variable importance with $l_n = 24$.

Software aspect in R (help)

```
ranger(formula = NULL, data = NULL, num.trees = 500, mtry = NULL,
  importance = "none", write.forest = TRUE, probability = FALSE,
  min.node.size = NULL, max.depth = NULL, replace = TRUE,
  sample.fraction = ifelse(replace, 1, 0.632), case.weights = NULL,
  class.weights = NULL, splitrule = NULL, num.random.splits = 1,
  alpha = 0.5, minprop = 0.1, split.select.weights = NULL,
  always.split.variables = NULL, respect.unordered.factors = NULL,
  scale.permutation.importance = FALSE, keep.inbag = FALSE,
  inbag = NULL, holdout = FALSE, quantreg = FALSE,
  oob.error = TRUE, num.threads = NULL, save.memory = FALSE,
  verbose = TRUE, seed = NULL, dependent.variable.name = NULL,
  status.variable.name = NULL, classification = NULL,
  bootstrap.ts = NULL, by.end = TRUE, block.size = 10, period = 1)
```

New ranger function with all the parameters

`bootstrap.ts`	Bootstrapping mode : empty for iid observations, "nonoverlapping" is default, "moving" for moving blocks, "circular" for circular blocks, "stationary" for stationary blocks, and "seasonal" for seasonal blocks.
`by.end`	Logical. Build block by the end of time series or not. Default = TRUE.
`block.size`	Number of observations in one block only if bootstrap by block is activated (bootstrap.ts has non null value).
`period`	Number of steps of one period. Only for the 'seasonal' block bootstrap.

The new parameters