Application Note

**An R package stacking for ensemble learning**

Taichi Nukui, Akio Onogi

Department of Plant Life Science, Faculty of Agriculture, Ryukoku University, 1-5, Yokotani, Seta, Oe-cho, Otsu, Shiga, 520-2194, Japan

Corresponding author:

Akio Onogi

Postal address: 1-5, Yokotani, Seta, Oe-cho, Otsu, Shiga, 520-2194, Japan

Phone & Fax: +81-(0)77-599-5719

E-mail: [onogiakio@gmail.com](mailto:onogiakio@gmail.com)

**Abstract**

**Summary**

An R package that

**Availability and implementation**

The R package

**Contact**

onogiakio@gmail.com

**Supplementary information**

Supplementary Figures are

**Text (~2000 words with one figure)**

Ensemble learning is often used in prediction tasks in machine learning. Stacking is one of ensemble learning methods, and have various applicable issues in biology. For example, in genomic prediction, stacking using support vector regression (SVR), kernel ridge regression (KRR), and elastic nets (ENET) as base models had on average 7.70% higher prediction accuracy across the three data sets than single models (#Liang et al. 2021). However, stacking requires cumbersome scripting. It also requires longer computation time to train the models because multiple models should be trained. In this study, we developed the R package "stacking" to overcome these problems. This package is enable users to conduct stacking using supervised learnings implemented in the R package caret (#??) which allows many methods in different packages to be handled in the same manner. The package stacking would lower the hurdles for users to conduct stacking by mitigating scripting tasks, and reduces the computational time by implementing parallel computations using the R package snow (#??).

The strategy of stacking implemented in our package is illustrated below using an example in which the number of cross-validation (CV) folds is 5 and the number of learners (members of ensemble) is 3. First, CV is conducted with each learner using the training data. Because each learner is trained 5 times, total 15 () base models are obtained. Then, using the predicted values of each learner as the explanatory variables (i.e., three explanatory variables), the meta learner is trained. In prediction, first, the testing data is given to the base models resulting 15 predicted values. Then the predicted values are averaged for each learner resulting three predicted value. Giving these values as the explanatory variables to the meta model, the final predicted values are obtained.

The package stacking mainly consists of four functions: *stacking\_train*, *train\_basemodel*, *train\_metamode*l, and *stacking\_predict*. *stacking\_train* is the function for training and *stacking\_predict* is the function of prediction. The former internally calls *train\_basemodel* and *train\_metamodel*, but users can also call these functions themselves to optimize base learners and meta learners. Because the training process of base models can take long computational time, *train\_basemodel* implements a parallel computation using the package snow.

*stacking\_train* mainly takes six arguments, *Y* , *X* , *Nfold* , *Method* , *Metamodel*, and *core*. *Y* and *X* are the vector and matrix of objective and explanatory variables, respectively. *Nfold* is the scalar indicating the number of CV folds. *Method* is a list containing data frames as elements. Names of the list indicate the methods of base learners and are passed to caret functions. Each element of the list is a data frame including the hyperparameters of each learner. When the number of rows of the data frame is > 1, i.e., multiple hyperparameter values are given, all combinations of hyperparameter values are automatically created and treated as different base learners. *Metamodel* is a character indicating the meta learner. *core* is a scalar indicating the number of cores for parallel computing. *stacking\_train* pass the arguments *Y*, *X*, *Nfold*, *Method*, and *core* to *train\_basemodel*, then the output list of *train\_basemodel* and *Metamodel* are given to *train\_metamodel*. *train\_metamodel* also takes an additional argument *which\_to\_use* which indicating which base models are used for training the meta model. *train\_metamodel* then outputs the training results of the base and meta models as a list which is then output by *stacking\_training*. *stacking\_predict* mainly takes two arguments, *newX* and *stacking\_train\_result*. *newX* is a matrix containing explanatory variables of new data, and *stacking\_train\_result* is a list output by *stacking\_training* (*train\_metamodel*). *stacking\_predict* outputs a vector of predicted values.

Users can optimize base and meta learners by calling *train\_basemodel* and *train\_metamodel.* For example, by separating validation data from the training data, it is possible to optimize the base learners and/or meta learners by predicting the validation data with different base/meta learners.

First we demonstrate the usage of stacking using simulations. An explanatory variable matrix (*X*) with 100 samples and 200 explanatory variables was randomly generated from a standard normal distribution as training data, and a vector of objective variables (*Y*) was created by setting the regression coefficient of the first 10 explanatory variables to 1 and the remaining to 0, and adding random noise. Testing data of the same size was also generated with the same manner. As base learners, Elastic net, partial least squares regression, and principal component regression were used. The argument *Method* is specified as

where *glmnet*, *pls*, and *pcr* are the arguments of caret to specify the package loaded, and *alpha*, *lambda*, and *ncomp* are the hyperparameters of the methods. The hyperparameters to be specified can be confirmed by using the *modelLookup* function of caret. As the meta learner, linear regression (*lm*) was used as metamodel. Null in lambda of glmnet indicates that lambda is optimized automatically under the given alpha value. For comparison, we also used these base and meta learners alone. The prediction accuracy is presented in Figure 1A, suggesting the usefulness of stacking.

Next,

In conclusion, the ensemble learning stacking can be easily tried using the developed package. It was also found to be more accurate in predicting stacking compared to a single model.

**Funding information**

This study was financially supported by Ryukoku University.

**References**