enpls: R Package for Ensemble Partial Least Squares Regression

Nan Xiao, Dong-Sheng Cao, Qing-Song Xu

June 19, 2016

Computational Biology &
Drug Design Group
Central South Univ., China

1 Introduction

It is usually a difficult and time-consuming task to construct an accurate predictive model in QSAR (Quantitative Structure-Activity Relationship) studies. The process often involves feature selection, outlier detection, non-linearship, and model stability problems. Such modeling procedures is quite tedious for the non-expert users who do not have comprehensive knowledge or in-depth understanding of the related methods. Not to mention that there exists far too many customized algorithms that can solve such problems, which are often not easy to understand and implement.

For the most frequently used model in QSAR studies, i.e. the partial least squares, we present a simple, easy-to-understand unified framework to solve such problems, users can do feature selection, outlier detection, and ensemble prediction under our framework. Also, a "clean" dataset can be generated using our method before modeling. We present the R package `enpls` here as the implementation.

In theory, statistical distribution can provide abundant information about random variables. Most approaches of statistical inference are based on such a statistical distribution. In our previous studies (Cao *et al.*, 2010, 2011), we made use of such a strategy to construct the statistical distribution of model features, such as prediction errors and variable coefficients, and subsequently made statistical inference. The statistic of these distributions, namely mean value and standard deviation, are then used to quantitatively describe various model features. Monte-Carlo or bootstrap approaches are constantly employed to extract the information and used for statistical inference. In general, Monte-Carlo or bootstrap approaches can be used to generate a distribution of some statistic of interest by repeatedly calculating that statistic randomly selected portions of the data because of its good asymptotic properties. For each of the functions, two resampling methods were considered, i.e. Monte-Carlo resampling (default method) or bootstrap resampling. In general, the Monte-Carlo resampling method randomly samples from the original dataset for many times, each time by a tunable sampling ratio $\in (0, 1)$. The bootstrap resampling method, in the other hand, randomly samples the same size of the original dataset from the dataset with replacement.

In QSAR/QSPR study, if we model a given QSAR/QSPR dataset by a single training/validation set division, we can obtain predictive errors of this validation set and all variable coefficients, characterizing the behavior of model features (i.e., prediction errors and variable coefficients) within these two sets. However, these model features highly depend on the way in which we split the data into the training set and validation set. Different training/validation data division should yield different model features. Thus, by changing the training/validation data division by Monte-Carlo or bootstrap methods, we can obtain a large number of QSAR/QSPR models and corresponding model features so as to gain some insight of the data structure statistically.

What kind of information about these model features can be obtained from their distribution? Generally speaking, some parameters of interest can be acquired as a function of the probability density function or of the empirical cumulative distribution function of a random variable (e.g., model features), which will make statistical

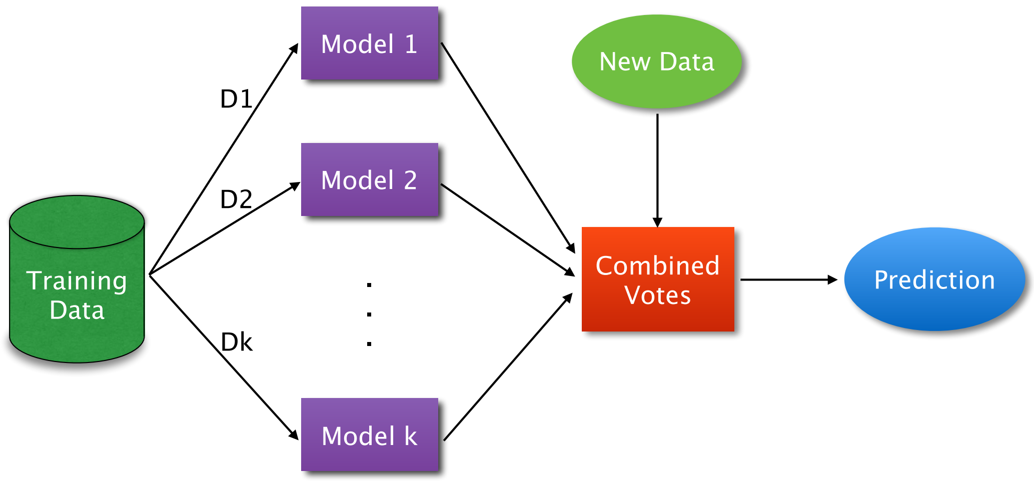

Figure 1: Ensemble methods for increasing prediction accuracy

inference about model features easier. Suppose that $z_1, \ldots, z_m$ will be used to estimate a population parameter θ. A function of a population distribution function, defining the parameter θ, can usually be expressed as:

$$\theta = \int g(z)\, dP_m(z)$$

Here $g(z)$ is the statistic used to estimate θ, whose expectations we might be interested in. $P_m(z)$ is the probability density of z. Thus, by constructing different $g(z)$, one can obtain different statistics θ describing specific information (e.g., mean value or standard deviation) of a population distribution.

In Cao *et al.* (2010, 2011), we addressed feature selection, outlier detection and model reliability problems simultaneously by constructing a unified framework, based on the idea of the statistical distribution. Our approach exploits the fact that the distribution of linear model coefficients provides a mechanism for ranking and interpreting the effects of variable, while the distribution of the prediction errors provides a mechanism for differentiating the outliers from normal samples. By combination of multiple models, we construct ensemble partial least squares model to improve prediction performance.

We will use the `alkanes` data throughout this manual for demonstration. The dataset has a predictor matrix `x` with 207 samples and 21 variables, with a continuous response `y`. The dataset is extracted from Liang *et al.* (2008). See `?alkanes` for details.

```
require(enpls)

## Loading required package:  enpls

data(alkanes)
x = alkanes$x
y = alkanes$y
```

2 Ensemble PLS for Feature Selection

Monte-Carlo uninformative variable elimination (Centner *et al.*, 1996) methods have been successfully employed in variable selection (Cai *et al.*, 2008; Han *et al.*, 2008). The important variables should be the ones that possess both large mean value and small standard deviation. We construct the following measure of variable importance:

$$c = \frac{\mathrm{mean}(s)}{\mathrm{sd}(s)}$$

where s is the coefficient vector for the i-th variable, generated by Monte-Carlo or bootstrap. $\mathrm{mean}(s)$ and $\mathrm{sd}(s)$ represent the mean value and standard deviation, respectively. Thus, the variable with the largest c_i value should be the most important one in the pool of variables. These variables with the smaller c_i value should be removed due to their small contribution to models.

The function `enpls.fs()` is made for ensemble PLS feature selection:

```
set.seed(42)
varimp = enpls.fs(x, y, MCtimes = 100)
print(varimp, nvar = 10L)

## Variable Importance by Ensemble Partial Least Squares
## ---
##           Importance
## Chi.P.4    2.2065244
## MEDV.33    1.9367545
## Chi.C.3    1.8545709
## MEDV.23    1.6883637
## Estate.1   1.3216136
## Chi.P.5    1.3067888
## MEDV.22    1.1424463
## Chi.P.3    0.9728668
## MEDV.12    0.8895649
## Estate.2   0.8722233

plot(varimp, nvar = 10L)
```

The top ten important varibles are printed, and plotted in Figure 3, by using `nvar = 10` in `print()` and `plot()`. See `?plot.enpls.fs` and `?print.enpls.fs` for more available options.

By changing the default parameters in `enpls.fs()` and other functions in the `enpls` package, we could control the maximum components included in each model, resampling method (Monte-Carlo or Bootstrap). By setting the `parallel` parameter to an (> 1) integer, the model fitting will be done in parallel, which will increase the computation speed significantly.

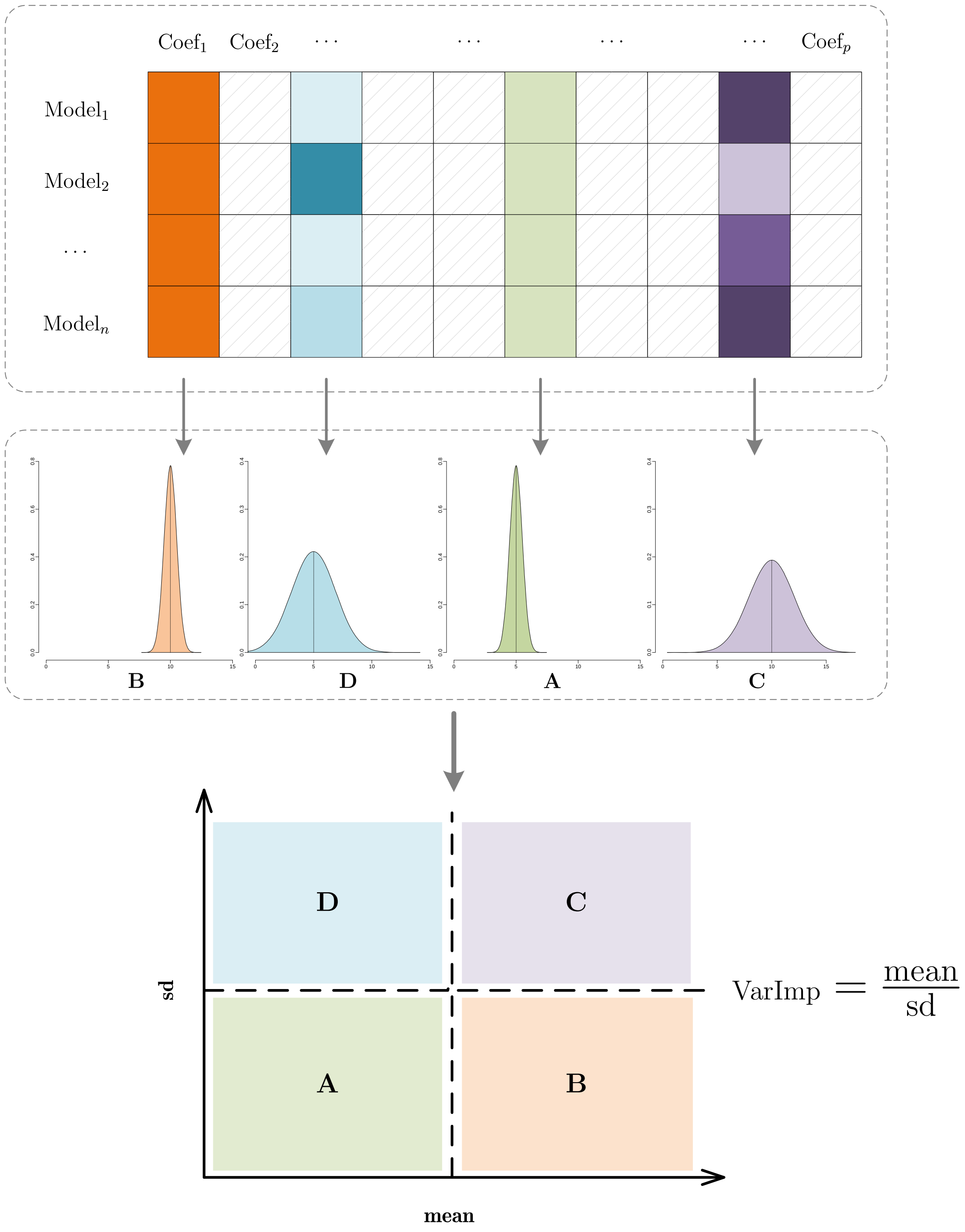

Figure 2: Feature selection in `enpls`

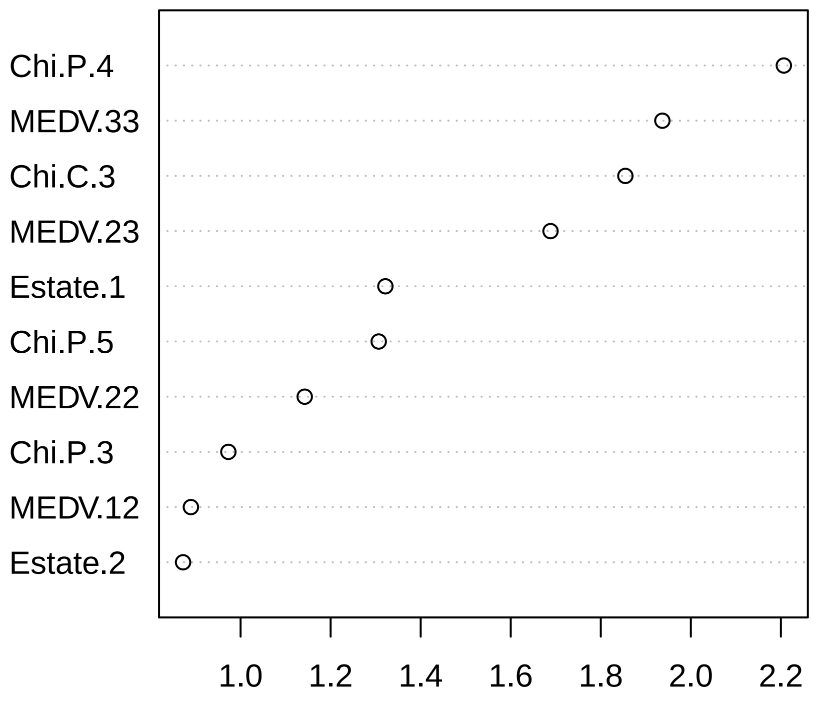

Figure 3: Top ten important variables of the `alkanes` dataset

3 Ensemble PLS for Outlier Detection

The distribution of prediction errors generated by a large number of models can contain more sample information (i.e., whether this sample is an outlier or not). Likewise, the mean value mean(j) and the standard deviation sd(j) of the prediction error distribution for the j-th sample are employed to describe this distribution.

$$\text{mean}(j) = \frac{1}{k}\sum_{i=1}^{k}\text{error}(i)$$

$$\text{sd}(j) = \left(\frac{1}{k-1}\sum_{i=1}^{k}(\text{error}(i) - \text{mean}(j))^2\right)^{\frac{1}{2}}$$

where k is the total times of which the j-th sample is found in the validation set. The error(i) is the prediction error of the j-th sample in the i-th cycle. Thus, a large mean value of prediction errors for some sample indicates that we can always obtain large prediction errors no matter how the training datasets are perturbated.

We can define two types of outliers, i.e. the y outlier and the X outlier. For the y outliers, the cross-prediction can provide information on potentially outliers. For example, if only one outlier molecule has many chlorine atoms and chlorine is an important variable, then the full dataset may be able to calibrate the effect of chlorine and make good predictions, but the dataset with the molecule excluded will likely lead to a large prediction residual on that molecule. So, the prediction errors obtained by cross-prediction allow us to easily detect such outliers compared to the fitted residuals.

In the other hand, in linear models, if an external data point x_i is being predicted and has a leverage of $h = x_i^t(X^tX)^{-1}x_i$, its prediction error has the variance $s^2\{e_i\} = \text{MSE}(1+h)$. We see that the variability of the sampling distribution of e_i is affected by how far x_i is from the centroid $\bar{X}$ through the term h. The further x_i is from $\bar{X}$, the greater the quantity is, and the larger the variance of e_i is. Thus, the variation of e_i obtained from different observations will be greater when x_i is far from the mean value than the ones near the mean value. We can therefore detect the X outliers by standard deviation of prediction errors.

The function `enpls.od()` is provided for outlier detection:

```
od = enpls.od(x, y, MCtimes = 100)
plot(od, criterion = 'sd')
```

Figure 5 reveals the outliers with `criterion = 'sd'`. This means samples that lie in n (default is 3) times out of the standard deviation of the mean *Error Mean* and mean *Error SD* are considered to be outliers. The black points are normal samples, the samples with red lables are y outliers (lower right), the blue ones are X outliers (upper left), the purple ones (may appear in the upper right part) will be the abnormal samples, as defined in Cao *et al.* (2011). Use `criterion = 'quantile'` to get the outliers by quantile information. See `?enpls.od` for details.

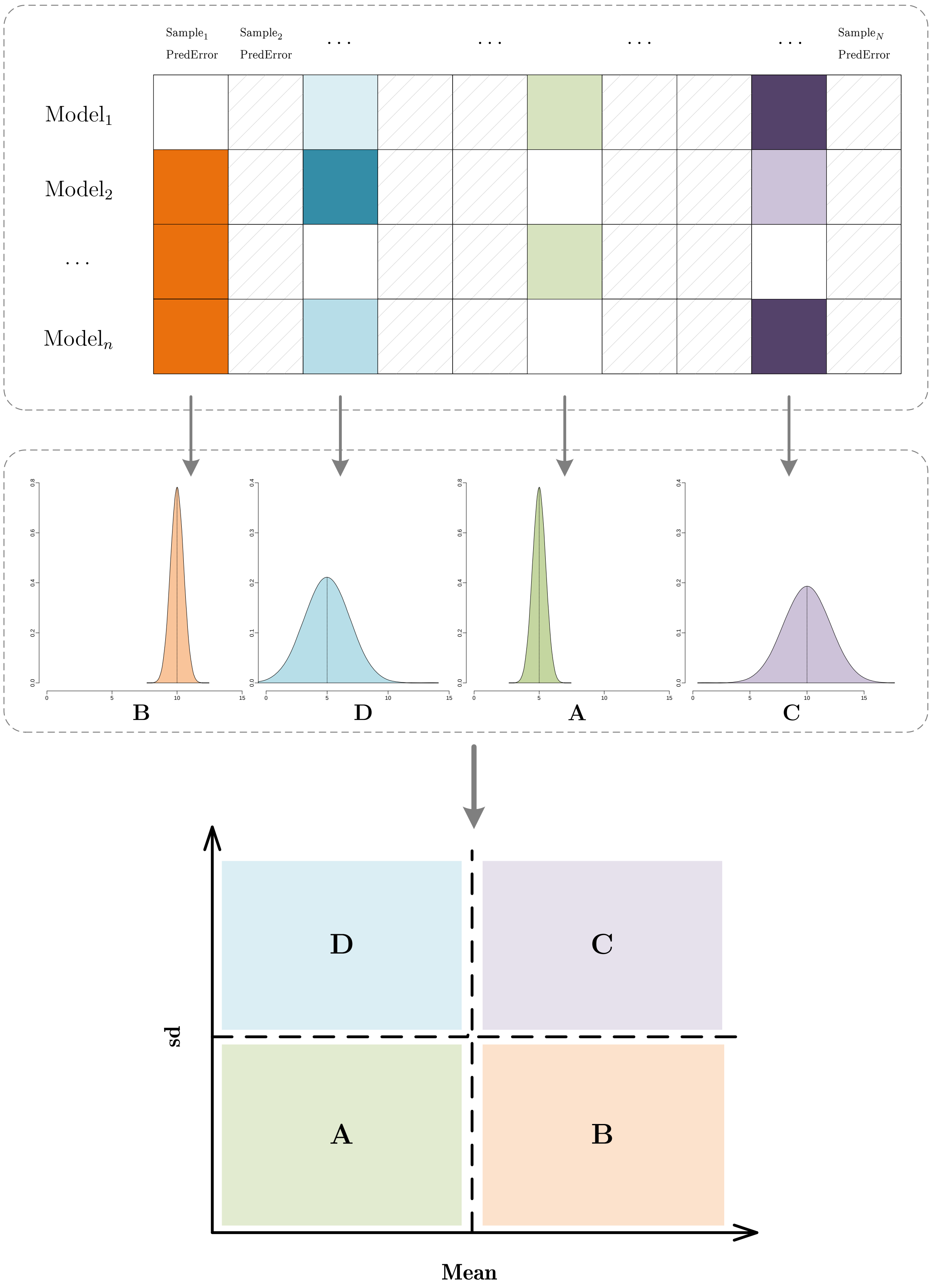

Figure 4: Outlier detection in `enpls`

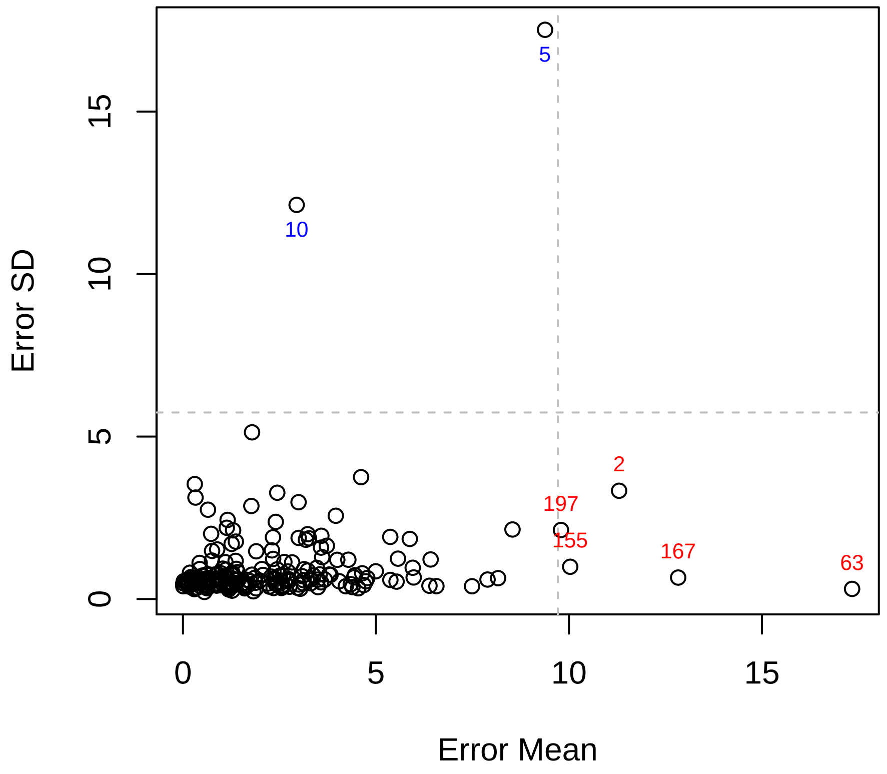

Figure 5: Outlier detection result of the `alkanes` dataset

4 Ensemble PLS Modeling and Prediction

Ensemble methods, like bagging (Breiman, 1996) and boosting (Friedman *et al.*, 2000), are usually used to improve model performance. Naturally, in `enpls`, we ensemble predictions from multiple PLS models generated by Monte-Carlo or bootstrap resampling methods to improve prediction performance.

For fitting ensemble partial least squares regression models, use `enpls.en()`:

```r
enpls.fit = enpls.en(x, y, MCtimes = 100)
```

With the fitted object `enpls.fit`, we could predict new X with `predict()`, and visualize the predicted result:

```r
y.pred = predict(enpls.fit, newx = x)
plot(y, y.pred, xlim = range(y), ylim = range(y),
     xlab = 'Experimental', ylab = 'Predicted')
abline(a = 0L, b = 1L)
```

Figure 7 shows the experimental values and predicted values.

5 Model Evaluation with k-fold Cross Validation

For ensemble partial least squares, `cv.enpls()` is used for k-fold cross validation:

```r
cv.enpls.fit = cv.enpls(x, y, MCtimes = 20)

## Beginning fold 1
## Beginning fold 2
## Beginning fold 3
## Beginning fold 4
## Beginning fold 5

print(cv.enpls.fit)

## Cross Validation Result for Ensemble Partial Least Squares
## ---
## RMSE = 3.2909, R2 = 0.999968

plot(cv.enpls.fit)
```

Then we printed the cross validation result: RMSE and R^2. The argument `nfolds` controls the fold number (default is 5). See `?cv.enpls` for details. Figure 8 shows the experimental values and the predicted values of the cross validation result.

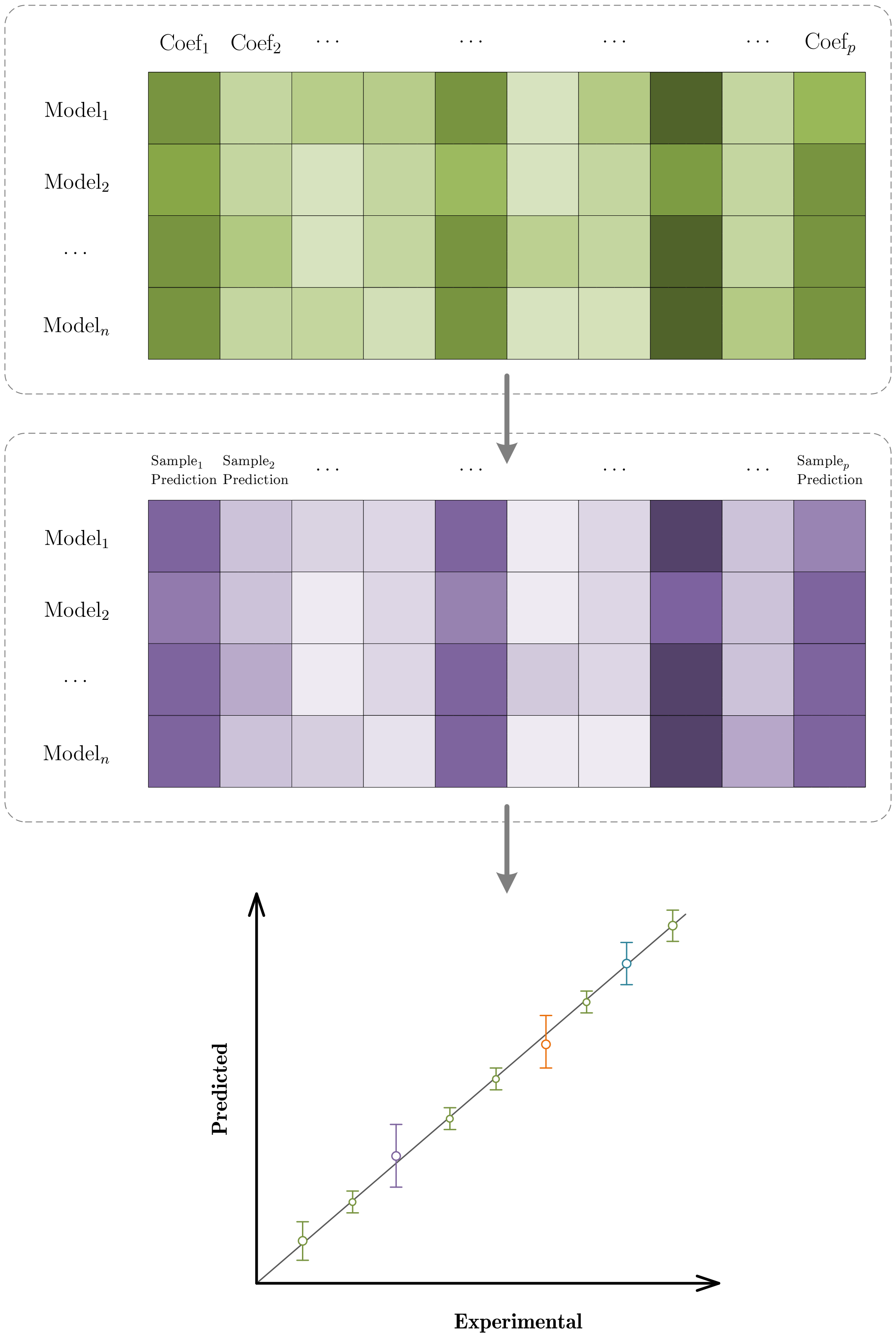

Figure 6: Ensemble modeling in `enpls`

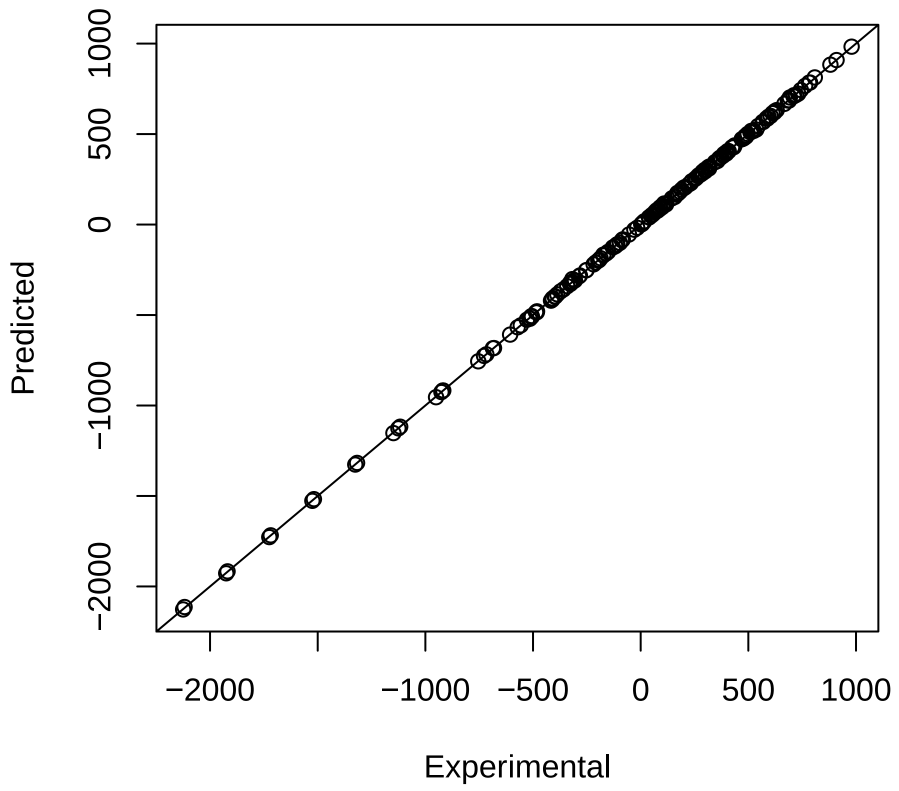

Figure 7: Experimental values vs. predicted values

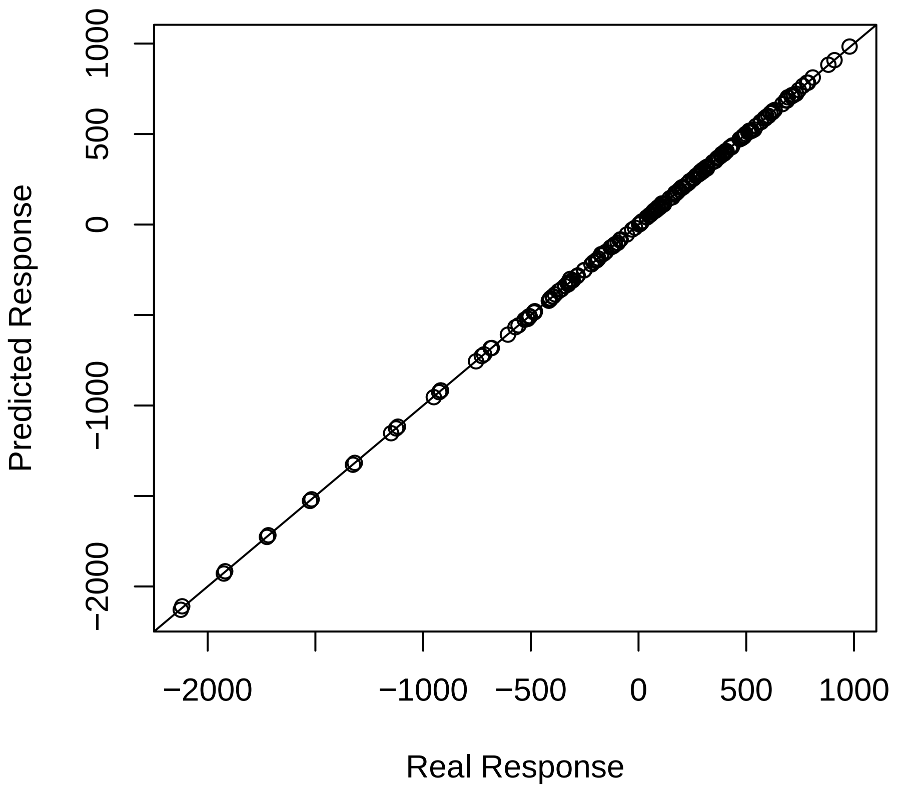

Figure 8: Cross validation result: experimental values vs. predicted values

References

Breiman L (1996). “Bagging predictors.” *Machine learning*, **24**(2), 123–140.

Cai W, Li Y, Shao X (2008). “A variable selection method based on uninformative variable elimination for multivariate calibration of near-infrared spectra.” *Chemometrics and intelligent laboratory systems*, **90**(2), 188–194.

Cao DS, Liang YZ, Xu QS, Li HD, Chen X (2010). “A new strategy of outlier detection for QSAR/QSPR.” *Journal of computational chemistry*, **31**(3), 592–602.

Cao DS, Liang YZ, Xu QS, Yun YF, Li HD (2011). “Toward better QSAR/QSPR modeling: simultaneous outlier detection and variable selection using distribution of model features.” *Journal of computer-aided molecular design*, **25**(1), 67–80.

Centner V, Massart DL, de Noord OE, de Jong S, Vandeginste BM, Sterna C (1996). “Elimination of uninformative variables for multivariate calibration.” *Analytical chemistry*, **68**(21), 3851–3858.

Friedman J, Hastie T, Tibshirani R (2000). “Special invited paper. additive logistic regression: A statistical view of boosting.” *Annals of statistics*, pp. 337–374.

Han QJ, Wu HL, Cai CB, Xu L, Yu RQ (2008). “An ensemble of Monte Carlo uninformative variable elimination for wavelength selection.” *Analytica chimica acta*, **612**(2), 121–125.

Liang YZ, Yuan DL, Xu QS, Kvalheim OM (2008). “Modeling based on subspace orthogonal projections for QSAR and QSPR research.” *Journal of Chemometrics*, **22**(1), 23–35.