Supplementary Information

Non-analyte signals and supervised learning to evaluate matrix effects and predict analyte recoveries in inductively coupled plasma optical emission spectrometry

Jake A. Carter,[a] John T. Sloop,[a] Tina Harville,[b] Bradley T. Jones [a] and George L. Donati [a,*]

[a] Department of Chemistry, Wake Forest University

Salem Hall, Box 7486, Winston-Salem, NC 27109

[b] Agilent Technologies, 2500 Regency Parkway, Cary, NC 27518

∗ Corresponding author. Tel.: +1 336 758 4815; fax: +1 336 758 4656.

E-mail address: donatigl@wfu.edu (G. L. Donati)

Machine learning algorithms

Several different types of supervised[S1] machine learning algorithms were included in the present study. Examples include boosting, generalized linear models, nearest neighbors, neural networks, partial least squares regression, random forests and support vector machine.[S2] Table 2 in the main text lists each model used with the respective descriptions and abbreviations. The following is a brief summary of each modeling technique used in the present study. For further details, the reader is referred to some works in the reference list.[S3-S5]

Random forests

Random forests are ensembles of deep, unpruned decision trees that are trained on bootstrapped training samples.[S6] The trees are built uncorrelated to each other, thereby allowing for a reduction in variation when averaging the bagged predictions. To do this, a random sample of m predictors (p) is chosen each time a split in a tree is considered. This allows for individual trees to have high variance stemming from different parts of the training data.[S7] A rule of thumb is to choose $m \approx (p)^{1/2}$ and $m \approx p/3$ for classification and regression, respectively. In a comparison of many different types of machine learning classifiers, the random forest algorithm proved to be the most accurate.[S2]

Boosting

Boosting is an ensembling technique that is applied to individually weak learning algorithms. Here, we focus on boosted decision trees. To build an ensemble of boosted decision trees, the trees are grown sequentially where each tree is fit to a modified version of the original data set. Specifically, trees are fit to the residuals of the current model, not the response.[S4] Unlike random forests, which rely on bagging uncorrelated decision trees, boosted decision trees are highly dependent on each other. Typically, shallow trees (higher bias, lower variance) are sequentially added which slowly improve the fit function by lowering the residuals. The present work features gradient tree boosting via the extreme gradient tree boosting algorithm. The algorithm is a scalable end-to-end tree boosting system that has recently shown predictive success on various datasets.[S8]

Generalized linear models and regularized generalized linear models

Generalized linear models (GLM)s are extensions of standard linear regression models that account for non-normal response distributions and possibly nonlinear functions of the mean. They are composed of a response that is a member of the exponential family distribution, and a link function that describes how the mean of the response and a linear combination of the predictors are related.[S9] Traditionally, linear models are fit according to the least-squares method, in which model coefficients are selected to minimize the residual sum of squares (RSS). The RSS is defined by eqn. (S1), where β_0 and β_j are estimated model coefficients, y_i is the ith observation of the response variable (y), and x_{ij} is the ith observation of the jth feature of x. Other fitting methods, however, may lead to better model interpretability and prediction accuracy. Different from least-squares, the shrinkage method is a technique that constrains or regularizes the coefficient estimates to zero.[S4] This may have the effect of significantly reducing the variance in the model. Two popular techniques involving the shrinking method are ride regression and the lasso.[S10] Ridge regression and lasso regression select model parameters that minimize the quantities defined by those in eqns (S2) and (S3), respectively, where λ is a hyperparameter tuned empirically over training data.[S4]

$$RSS = \sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 \tag{S1}$$

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}\beta_j^2 \tag{S2}$$

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}\beta_j^2 = RSS + \lambda\sum_{j=1}^{p}|\beta_j| \tag{S3}$$

The ridge regression and lasso techniques select model coefficients that minimize quantities that include a penalty term in addition to the RSS. Ridge regression, with the penalty term, $\lambda\sum_{j=1}^{p}\beta_j^2$, achieves better prediction than traditional linear regression due to an optimized bias-variance tradeoff. As λ increases, the flexibility of the ridge regression fit decreases. This causes decreased variance but increased bias. The lasso regression, with the penalty term, $\sum_{j=1}^{p}|\beta_j|$, allows for variable selection due to the use of the *L1* penalty (*i.e.* $|\beta_j|$) instead of the *L2* penalty (*i.e.* β_j^2) featured in ridge regression.[S4,S11] As a result of the *L1* penalty, at larger values of λ, some of the model coefficients are forced to be exactly zero. Models generated from lasso regression may be more interpretable than ridge regression due to the reduced set of features. The elastic-net is another form of regularization of the generalized model that encompasses the bias-variance tradeoff of ridge regression and the feature selection capability of lasso regression.[S12] The elastic-net features the penalty term defined by eqn (S4), where α governs the contribution of each penalty term.

$$\lambda\sum_{j=1}^{p}(\alpha\beta_j^2 + (1 - \alpha)|\beta_j|) \tag{S4}$$

When $\alpha = 1$, the elastic-net penalty is simply ridge regression. When $\alpha = 0$, the penalty is lasso regression. The present work features results from a generalized linear model and a penalized generalized linear model using the elastic-net penalty.

k-nearest neighbor regression

The *k*-nearest neighbor algorithm for regression (KNN) is a simple strategy that takes a test prediction point and considers the value from the "*k*" nearest neighbors.[S4] The value of *k* is a hyperparameter determined empirically from training data. The response values of the *k* neighbors are averaged to give the predicted response. This process is defined by eqn (S5),[S4] where $\hat{f}(x_0)$ is the estimated predicted value for point x_0, N_0 are the *K* training observations that are closest to x_0, and y_i is the *i*th observation of the response.

$$\hat{f}(x_0) = \frac{1}{K}\sum_{x_i \in N_0} y_i \tag{S5}$$

Partial least-squares regression

Partial least-squares (PLS) regression is a dimension reduction method for linear regression, which is a popular modeling technique within the chemometrics community.[S13] Unlike shrinkage methods for linear regression, in which model coefficients are shrunk towards zero, PLS regression transforms the original predictor variables into a new set of features, *M*, which are linear combinations of the original features. A linear model is then fit via the least-squares method using the newly transformed variables. The linear combinations are determined according to eqn (S6), where $Z_1, Z_2, \ldots, Z_m$ represent $M < p$ linear combinations of the original *p* predictors.[S4]

$$Z_m = \sum_{j=1}^{p} \phi_{jm} X_j \tag{S6}$$

PLS regression determines Z_1 by setting each ϕ_{j1} equal to the coefficient from simple linear regression of Y onto X_j, where Y is the response matrix and X_j is the feature matrix for the jth feature. Therefore, the variables that are most strongly related to the response receive the highest weight according to PLS regression.

Neural network

A neural network is a form of nonlinear regression that was inspired by theories of how the brain works.[S14] Like PLS regression, the response is modeled by an intermediary set of unobserved variables (*i.e.* hidden units). These transformed variables are linear combinations of the original variables. A neural network model usually involves multiple hidden units to predict the response. Unlike PLS regression, the linear combinations are not determined in hierarchical fashion and there are no constraints that help define the linear combinations. As a result, there is little information in the coefficients from the linear combination of each hidden unit. After the optimal number of hidden units is defined, another linear combination connects the hidden units to the predicted response.[S3]

Optimal parameters are determined according to the minimization of the RSS, using an efficient algorithm such as back-propagation.[S5] Due to the highly flexible nature of neural networks (*i.e.* large number of regression coefficients), they are prone to overfitting. To prevent this, a penalization method known as weight decay, which is similar to the penalty used in ridge regression, may be used to regularize the model.[S5] The weight decay is a hyperparameter that is determined empirically from the training data. Therefore, the optimization procedure would seek to minimize the quantity listed by eqn (S7), where H is the number of hidden units, k is the kth hidden unit, λ is the weight decay, β_{jk} is the model coefficient of the linear combination connecting the *jth* feature to the kth hidden unit, and γ_k is the kth model coefficient of the linear combination connecting the hidden units to the response.[S3] This is the description for the simplest neural network architecture, known as a single-layer feed forward network. Our work features the use of a multilayer perceptron,[S5] which is a fully connected one-layer network.

$$\sum_{i=1}^{n}(y_i - f_i(x))^2 + \lambda\sum_{k=1}^{H}\sum_{j=0}^{p}\beta_{jk}^2 + \lambda\sum_{k=0}^{H}\gamma_k^2 \tag{S7}$$

Support vector regression

Support vector machines (SVM)s were first proposed to solve classification problems where response values lie on a discrete scale (*e.g.* 0 and 1 or "control" and "disease").[S15] To extend the applications to regression, in which values are on a continuous scale, an SVM model is fit to training data such that it has at most ϵ deviation from the response variable, while attempting to be as flat as possible.[S16] Considering a user set threshold value, ϵ, data with residuals within ϵ do not contribute to the regression fit, and data with an absolute difference greater than ϵ contribute a linear scale amount to the model fit. This process is referred to as *ϵ-insensitive* regression, which is described by the ϵ-insensitive function, L_ϵ. Model coefficients for SVMs are fit to minimize the quantity listed by eqn (S8).[S3] *Cost* penalizes large residuals and is a hyperparameter that is optimized over training data.

$$Cost\sum_{i=1}^{n}L_\epsilon(y_i - \hat{y}_i) + \sum_{j=1}^{p}\beta_j^2 \tag{S8}$$

Given a prediction point x_0, a trained SVM model makes predictions according to eqn (S9), where α_i is the set of unknown model parameters and $K(x_i, x_o)$ is a kernel function capable of describing non-linear relationships among the features.[S3] For a certain percentage of training data, the α_i parameters will be exactly zero.

$$\hat{f}(x_0) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(x_i, x_0) \tag{S9}$$

This set of training data represents the samples that are within $\pm\,\epsilon$ of the regression line. The regression line is determined using the training data points where $\alpha \neq 0$. Collectively, these are referred to as the support vectors since they support the regression line.[S3]

Supplementary figures

Figures associated to Co, Cr and Pb results, similar to those presented for Cd in the main manuscript, are presented here. Results for Co are shown in Figs. SI1, SI4 and SI7. For Cr, results are shown in Figs. SI2, SI5 and SI8. Lead results are shown in Figs. SI3, SI6 and SI9.

References

S1. D. Bzdok, M. Krzywinski, N. Altman, Machine learning: supervised methods, *Nat. Methods*, 2018, **15**, 5-6.

S2. M. Fernández-Delgado, E. Cernadas, S. Barro, D. Amorim, Do we need hundreds of classifiers to solve real world classification problems?, *The Journal of Machine Learning Research*, 2014, **15**, 3133–3181.

S3. 68. M. Kuhn, K. Johnson, *Applied Predictive Modeling*, Springer, New York, 2013.

S4. G. James, D. Witten, T. Hastie, R. Tibshirani, *An Introduction to Statistical Learning*, Vol. 103, Springer, New York, 2013.

S5. J. Friedman, T. Hastie, R. Tibshirani, *The Elements of Statistical Learning*, Springer series in statistics, Vol. 1, New York, 2001.

S6. L. Breiman, Random forests, *Machine Learning*, 2001, **45**, 5–32.

S7. N. Altman, M. Krzywinski, Ensemble methods: bagging and random forests, *Nat. Methods*, 2017, **14**, 933-934.

S8. T. Chen, C. Guestrin, XGBoost: A Scalable Tree Boosting System, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining - KDD ’16, ACM Press, San Francisco, California, USA, 2016: pp. 785–794. doi:10.1145/2939672.2939785.

S9. J. J. Faraway, Extending the Linear Model with R: Generalized Linear, Mixed Effects and Nonparameteric Regresison Models, second ed., CRC Press LLC, Florida, 2016.

S10. J. Friedman, T. Hastie, R. Tibshirani, Regularization paths for generalized linear models via coordinate descent, *Journal of Statistical Software*, 2010, **33**, 1-22.

S11. R. Tibshirani, Regression Shrinkage and Selection via the Lasso, *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 1996, **58**, 267-288.

S12. H. Zou, T. Hastie, Regularization and Variable Selection via the Elastic Net, *Journal of the Royal Statistical Society. Series B (Statistical Methodology)*, 2005, **67**, 301-320.

S13. S. Wold, M. Sjöström, L. Eriksson, PLS-regression: a basic tool of chemometrics, *Chemom. Intell. Lab. Syst.*, 2001, **58**, 109–130.

S14. F. Rosenblatt, The perceptron: a probabilistic model for information storage and organization in the brain, *Psychological Review*, 1958, **65**, 386-408.

S15. C. Cortes, V. Vapnik, Support-Vector Networks, *Machine Learning*, 1995, **20**, 273-297.

S16. A. Smola, B. Schölkopf, A tutorial on support vector regression, *Statistics and Computing*, 2004, **14**, 199-222.

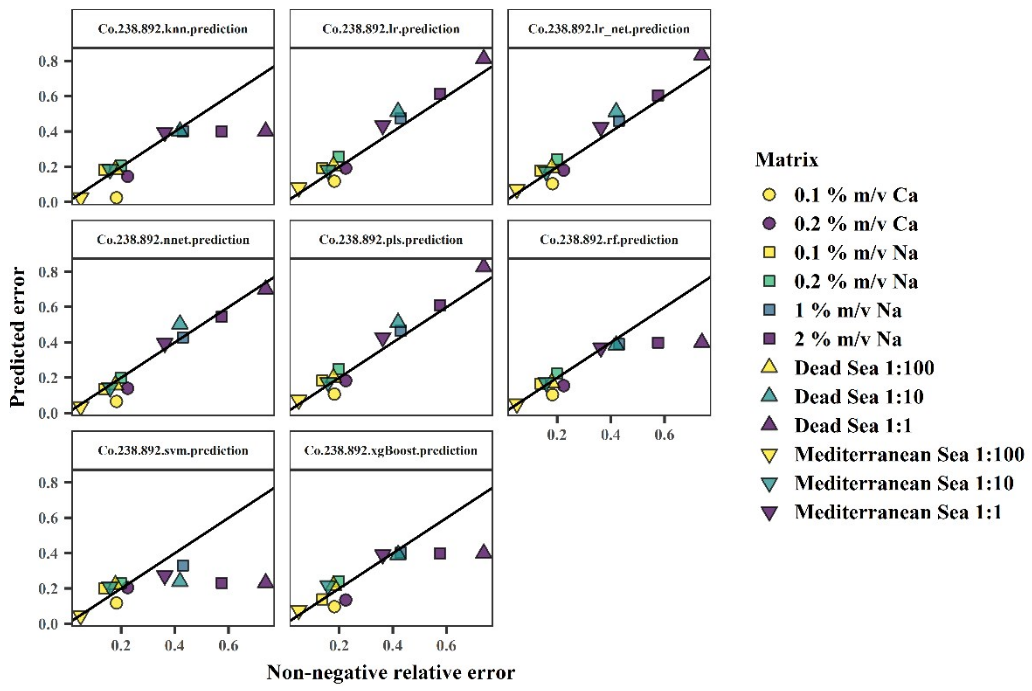

Fig. SI1. Predicted error *vs.* observed non-negative relative error for model predictions of matrix effects on Co for the external testing data. The line $y = x$ represents perfect prediction accuracy ($R^2 = 1$).

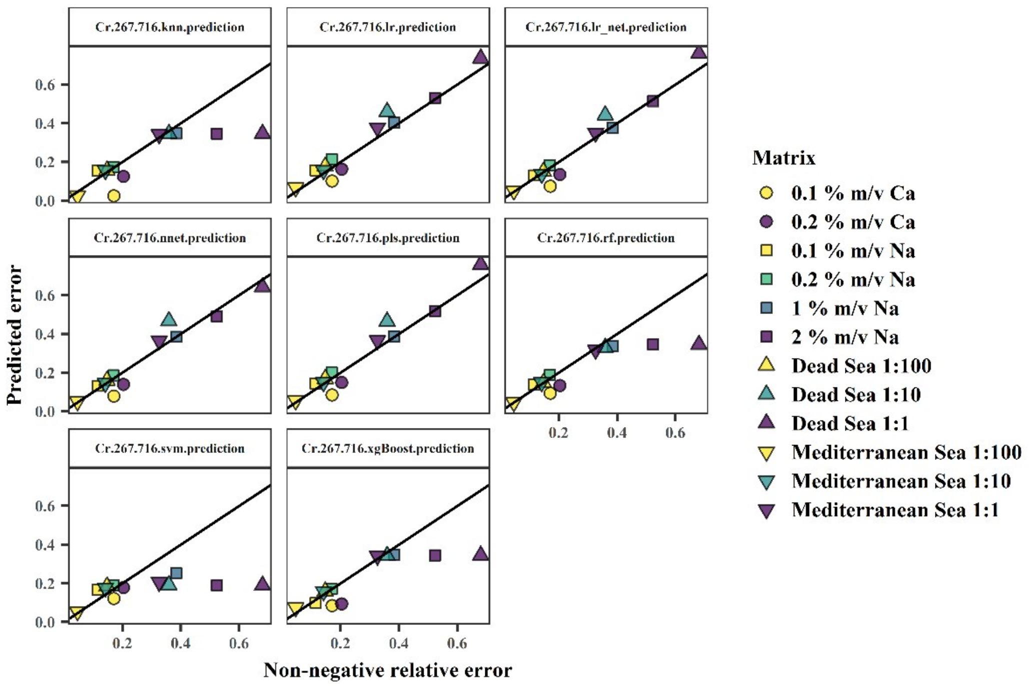

Fig. SI2. Predicted error *vs.* observed non-negative relative error for model predictions of matrix effects on Cr for the external testing data. The line $y = x$ represents perfect prediction accuracy ($R^2 = 1$).

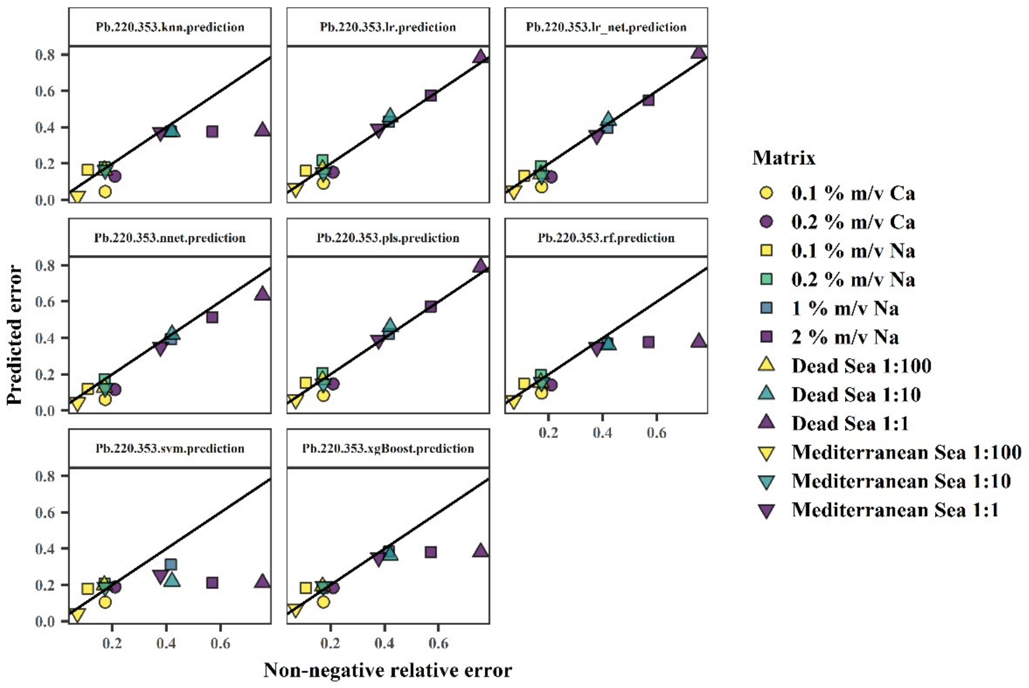

Fig. SI3. Predicted error *vs.* observed non-negative relative error for model predictions of matrix effects on Pb for the external testing data. The line $y = x$ represents perfect prediction accuracy ($R^2 = 1$).

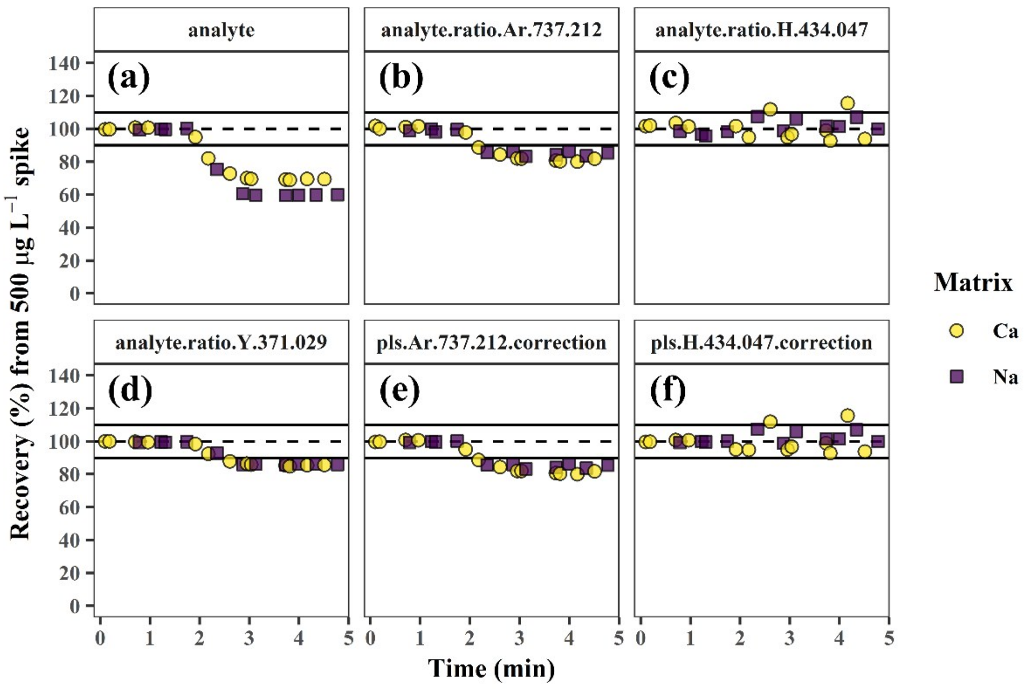

Fig. SI4. Analyte percent recoveries from a 500 µg L^{-1} spike of Co obtained for the withheld training data when using (a) the analytical signal alone, (b) the Ar signal at 737.212 nm as IS, (c) the H signal at 434.047 nm as IS, (d) the Y signal at 371.029 nm as IS, (e) the Ar signal at 737.212 nm as IS only when the trained *pls* model predicts a non-negative relative error > 0.1, and (f) the H signal at 434.047 nm as IS only when the trained *pls* model predicts a non-negative relative error ≥ 0.1. Dashed and solid lines represent 100% and between 90 - 110% recoveries, respectively.

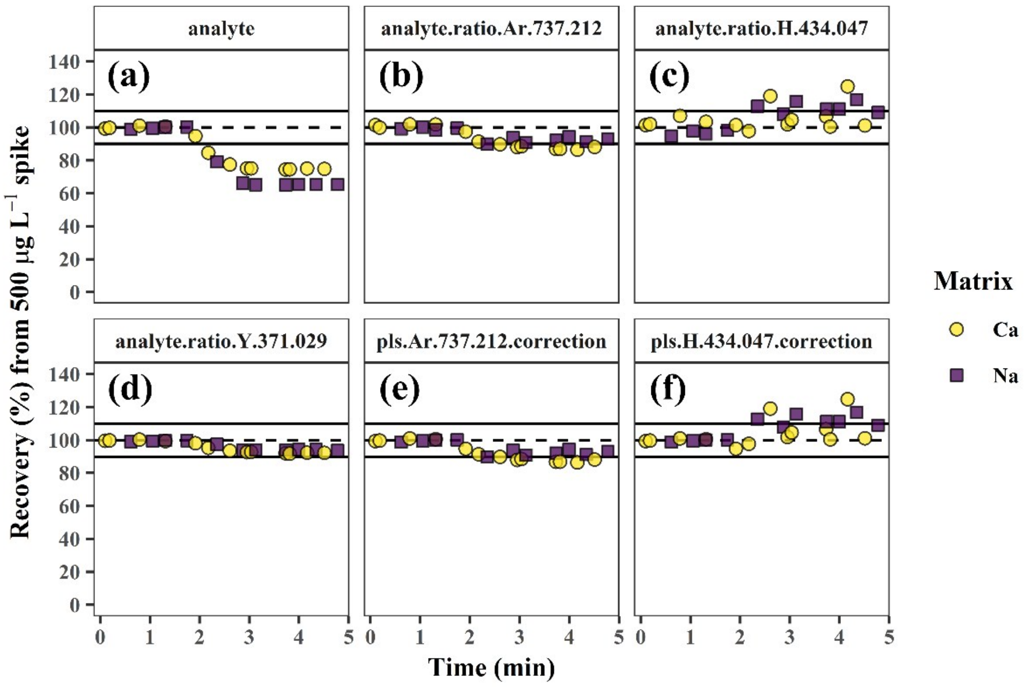

Fig. SI5. Analyte percent recoveries from a 500 µg L^{-1} spike of Cr obtained for the withheld training data when using (a) the analytical signal alone, (b) the Ar signal at 737.212 nm as IS, (c) the H signal at 434.047 nm as IS, (d) the Y signal at 371.029 nm as IS, (e) the Ar signal at 737.212 nm as IS only when the trained *pls* model predicts a non-negative relative error > 0.1, and (f) the H signal at 434.047 nm as IS only when the trained *pls* model predicts a non-negative relative error ≥ 0.1. Dashed and solid lines represent 100% and between 90 - 110% recoveries, respectively.

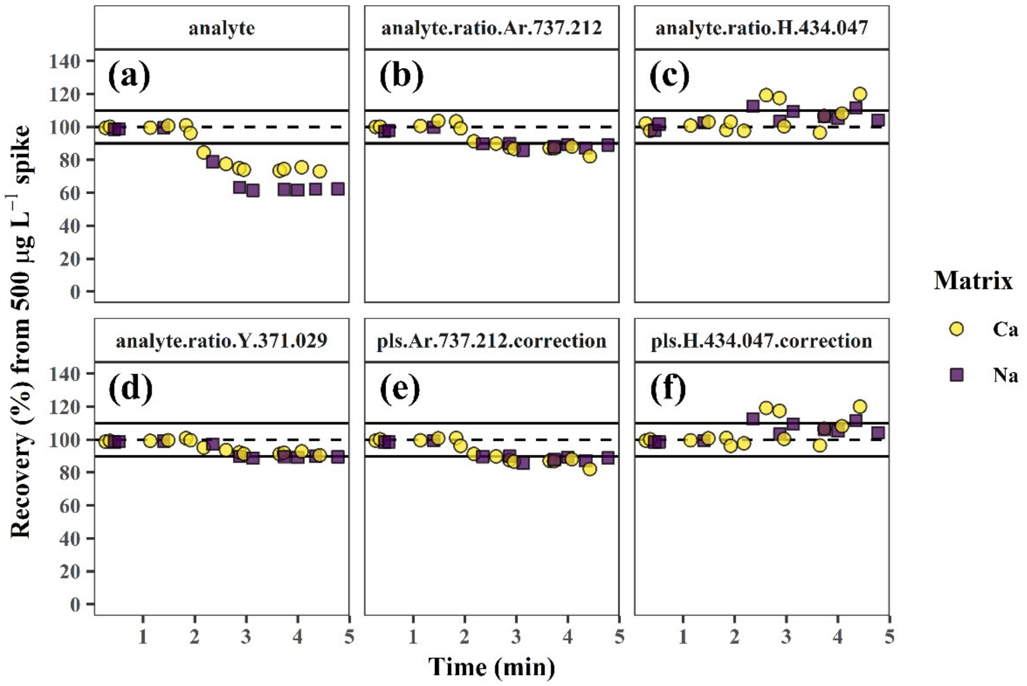

Fig. SI6. Analyte percent recoveries from a 500 µg L^{-1} spike of Pb obtained for the withheld training data when using (a) the analytical signal alone, (b) the Ar signal at 737.212 nm as IS, (c) the H signal at 434.047 nm as IS, (d) the Y signal at 371.029 nm as IS, (e) the Ar signal at 737.212 nm as IS only when the trained *pls* model predicts a non-negative relative error > 0.1, and (f) the H signal at 434.047 nm as IS only when the trained *pls* model predicts a non-negative relative error ≥ 0.1. Dashed and solid lines represent 100% and between 90 - 110% recoveries, respectively.

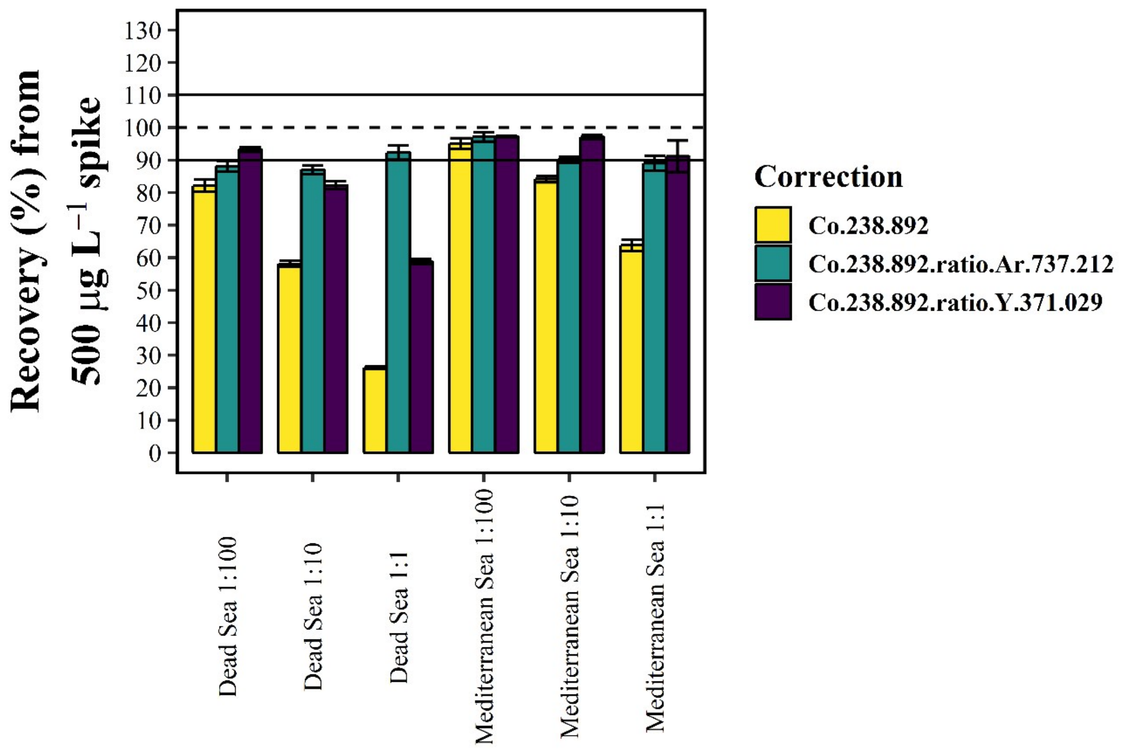

Figure SI7. Analyte percent recoveries from a 500 µg L^{-1} spike of Co in Dead Sea water and Mediterranean Sea water at different levels of dilution, *i.e.* 1:1, 1:10 and 1:100 v/v. Results are shown for calibration using the analytical signal alone (“Co.238.892”), the Ar signal at 737.212 nm as IS (“ratio.Ar.737.212”), and the Y signal at 371.029 nm as IS (“ratio.371.029”). Dashed and solid lines represent recoveries of 100% and between 90 - 110%, respectively.

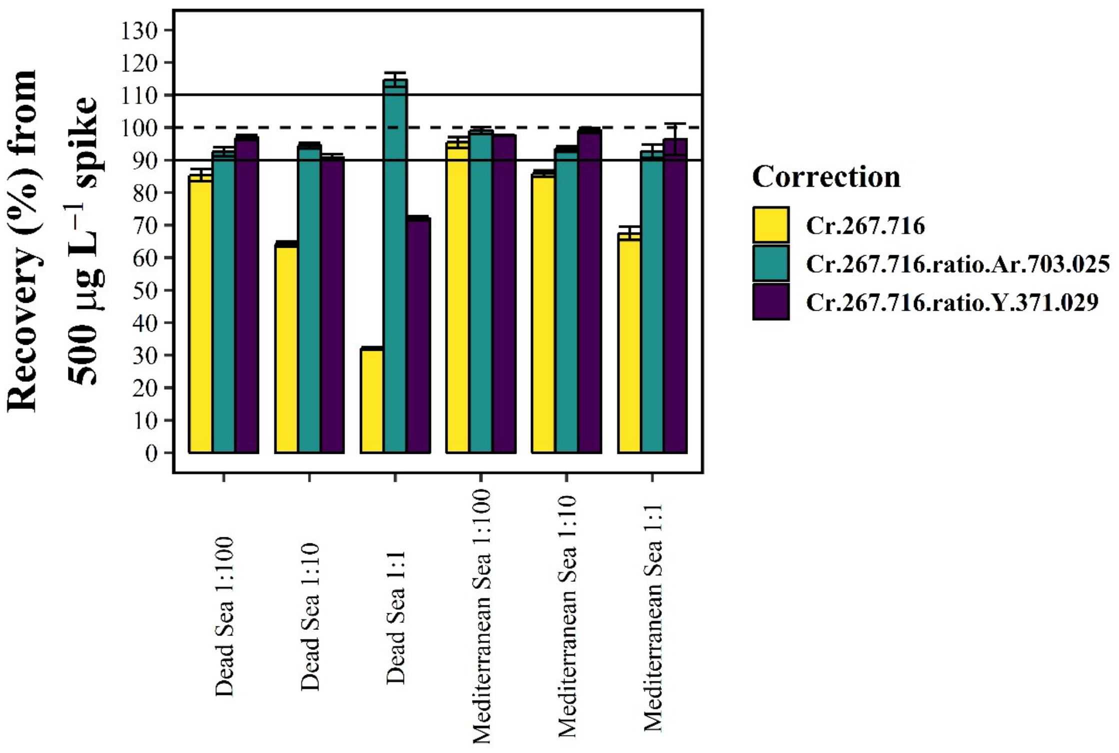

Figure SI8. Analyte percent recoveries from a 500 µg L^{-1} spike of Cr in Dead Sea water and Mediterranean Sea water at different levels of dilution, *i.e.* 1:1, 1:10 and 1:100 v/v. Results are shown for calibration using the analytical signal alone ("Cr.267.716"), the Ar signal at 737.212 nm as IS ("ratio.Ar.737.212"), and the Y signal at 371.029 nm as IS ("ratio.371.029"). Dashed and solid lines represent recoveries of 100% and between 90 - 110%, respectively.

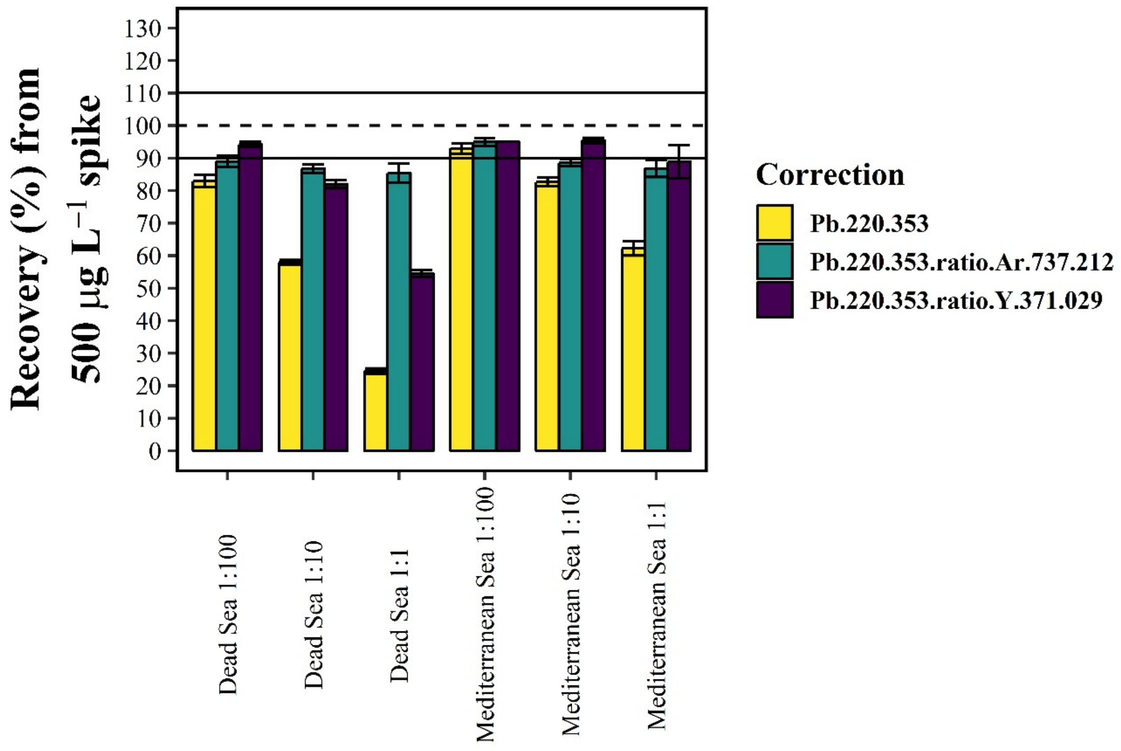

Figure SI9. Analyte percent recoveries from a 500 µg L^{-1} spike of Pb in Dead Sea water and Mediterranean Sea water at different levels of dilution, *i.e.* 1:1, 1:10 and 1:100 v/v. Results are shown for calibration using the analytical signal alone ("Pb.220.353"), the Ar signal at 737.212 nm as IS ("ratio.Ar.737.212"), and the Y signal at 371.029 nm as IS ("ratio.371.029"). Dashed and solid lines represent recoveries of 100% and between 90 - 110%, respectively.