SUPPORTING INFORMATION

Atomic Property Weighted Radial Distribution Functions Descriptors for of Metal-Organic Frameworks for the Prediction of Gas Uptake Capacity

Michael Fernandez, Nicholas R. Trefiak, Tom K. Woo

Centre for Catalysis Research and Innovation, Department of Chemistry, University of Ottawa, Ottawa, Canada

*Corresponding author
email: tom.woo@uottawa.ca

Table of contents:

1. Hypothetical MOF Database

The construction of the database is discussed in Ref.[1] The entire list of SBUs used to construct the hypothetical MOFs is shown in Figure S1.

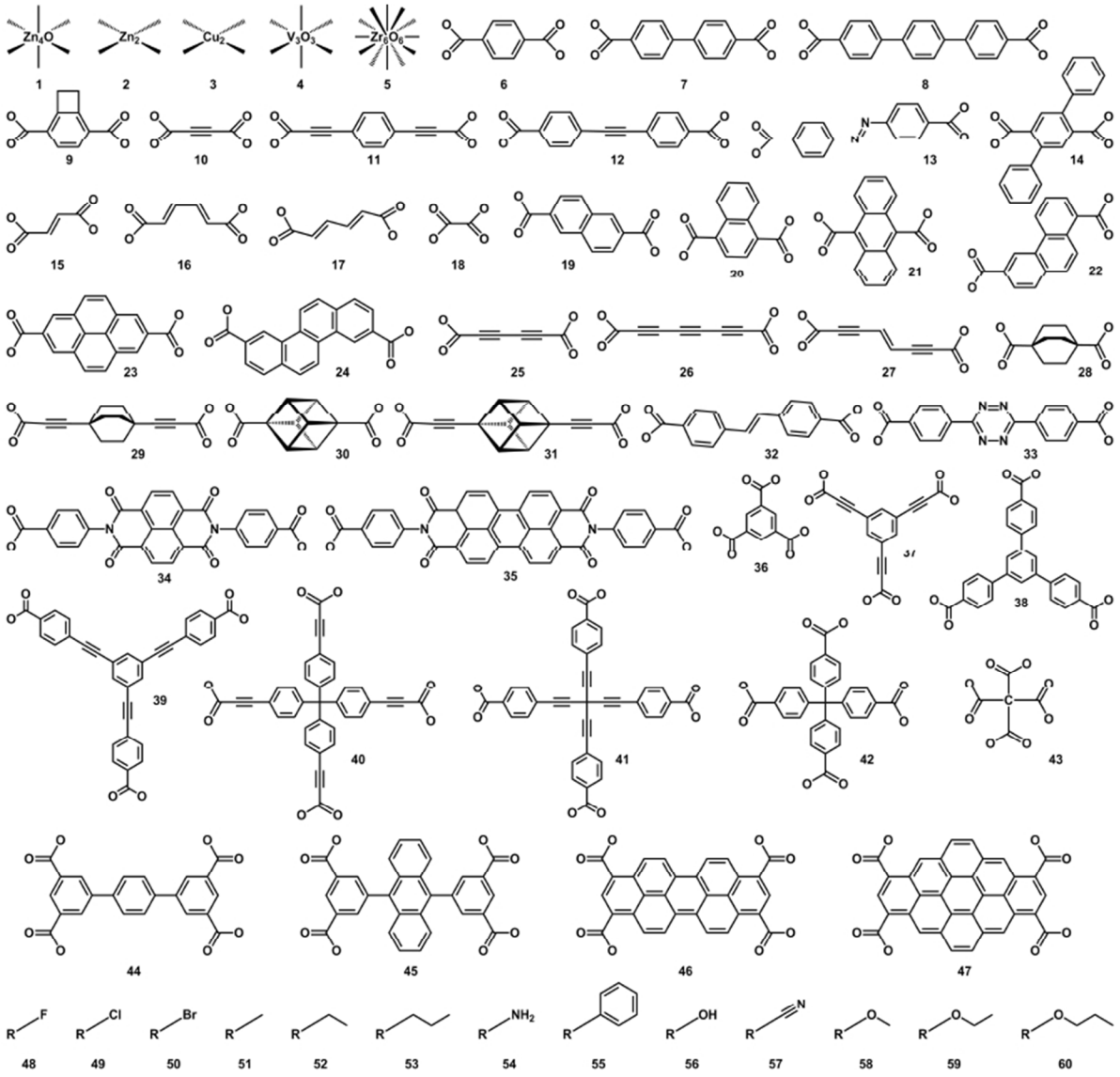

Figure S1. Structural Building Units (SBUs) used to generate the ~130,000 hypothetical MOFs from the Northwestern University database. The inorganic paddlewheels SBUs 2 and 3 are able to coordinate to nitrogen containing compounds (e.g.,pyrazine) but all of the analogous building blocks terminated by nitrogen atoms instead of carboxylic acid groups are not shown.

2. MOF Feature Calculation

A total of six geometrical features were taken the Northwestern University database namely, dominant pore diameter, maximum pore diameter, void fraction and gravimetric and volumetric surface areas. The dominant pore diameter corresponds to the tallest peak in the pore size distribution, which is calculated by inserting spheres at random locations within the framework and growing them until they collide with the surface of a framework atom. The maximum pore diameter is the largest sphere that could be grown inside the framework when calculating the pore size distribution. The void fraction is calculated by the helium insertion method that is described elsewhere.[2] The gravimetric and volumetric surface areas were determined by geometric calculations described previously.[3]

3. SVM Implementation

Here we briefly provide details of our implementation of support vector machines (SVMs) (an introduction to SVM methodology can be found in Ref. [4]). In our SVMs, the input vectors (i.e. MOF structural features) are first mapped onto one feature space (possibly with a higher dimension) by means of a kernel function. Then, a hyperplane is built to separate the positive and negative inputs in the data within this feature space. Only relatively low-dimensional vectors in the input space and dot products in the feature space will evolve by a mapping function. SVMs have been designed to minimize structural risk that prevents over-fitting by incorporating a regularization penalty into the optimization. The regularization penalty can be viewed as implementing a form of Occam's razor that prefers simpler functions over more complex ones controlling the bias/variance tradeoff. Other machine learning methods, such as Artificial Neural Networks, are based on empirical risk minimization, which minimizes the difference between the predictions of the independent variable of a hypothesis and the true outcome of the independent variable. Therefore, SVMs are less vulnerable to the over-fitting problem, and so they can typically deal with a large number of features. There are several important parameters in an SVM, including the kernel function (and its specific parameters) and the regularization parameters. Neither the kernel function nor the regularization parameters can be defined from the optimization problem but must be manually tuned. This can be done by applying Vapnik-

Chervonenkis bounds, cross-validation, an independent optimization set, or Bayesian learning.[5] We implemented a nonlinear radial-basis-function (RBF) Gaussian kernel that was optimized in a grid search through cross-validation. SVMs were implemented using the Python programming language and the SHOGUN toolbox.[6]

A radial-basis-function (RBF) kernel was implemented in the SVM framework to handle the nonlinearity in the data to yield higher accuracies. The optimum width of the RBF function and the regularization parameter of the SVM regressions were tuned by a grid search that maximized the cross-validation Q^2 of the training set (data not shown). The possibility of chance correlations was discarded by y-randomization analysis (data not shown).

4. Principal Components Analysis (PCA)

Principal Components Analysis (PCA) was applied to condense the variation in the *AP-RDF* scores into its principal properties and hence to obtain new information-rich orthogonal latent variables with reduced noise levels. PCA is a projection method in which systematic variation in a dataset is extracted into a few variables, so-called principal components.[7] The principal components are linear combinations of the original variables and are uncorrelated to each other as described by:

$$X = t^1 \cdot p'_1 + t^2 \cdot p'_2 + t^3 \cdot p'_3 + \ldots + t^A \cdot p'_A + E = TP' + E$$

where X is the original data matrix, A is the total number of extracted principal components, and E is the residual matrix. The new latent variables, t scores, show how the objects and targets relate to each other, while the p loadings reveal the importance of the original variables for the patterns seen in the scores. The two main Principal Components (PCs) of the *RDF* scores contained > 90% of the original variance. The PCA was implemented in python using the sklearn module for machine learning.[8]

5. Tabulated atomic properties

Table S1. Tabulated atomic properties used to weight the radial distribution function (*RDF*) descriptors. Each property value is normalized by the value for the carbon atom.

Element	electronegativity	polarizability	vdWaalsVolume
Al	0.6242	3.8636	1.6264
B	0.8285	1.7216	0.7962
Ba	0.2487	1.1252	1.7941
Br	1.1723	1.7330	1.3835
C	1.0000	1.0000	1.0000
Cd	0.7210	4.6600	1.3941
Cl	1.2655	1.2386	1.0347
Co	0.7283	4.2614	1.5609
Cr	0.7200	6.4444	1.6283
Cu	0.7403	3.4659	0.5120
F	1.4567	0.3165	0.4100
Fe	0.7283	4.7727	1.8287
Gd	0.7283	13.3523	3.2418
H	0.9439	0.3790	0.2989
I	1.0117	3.0398	1.7280
In	0.7786	5.3889	1.4633
Mg	0.4800	6.3600	1.0538
Mn	0.6045	5.2220	1.8287
N	1.1631	0.6250	0.6949
Ni	0.7283	3.8636	0.7643
O	1.3307	0.4557	0.5120
P	0.9159	2.0625	1.1814
S	1.0768	1.6477	1.0882
Si	0.7786	3.0568	1.4244
Sn	0.8369	4.3750	2.0415
V	0.6392	7.0455	1.6283
Zn	0.8095	4.0341	1.7084
Zr	0.5216	10.1705	1.6283

6. Coefficients of the RDF scores in the principal component 2

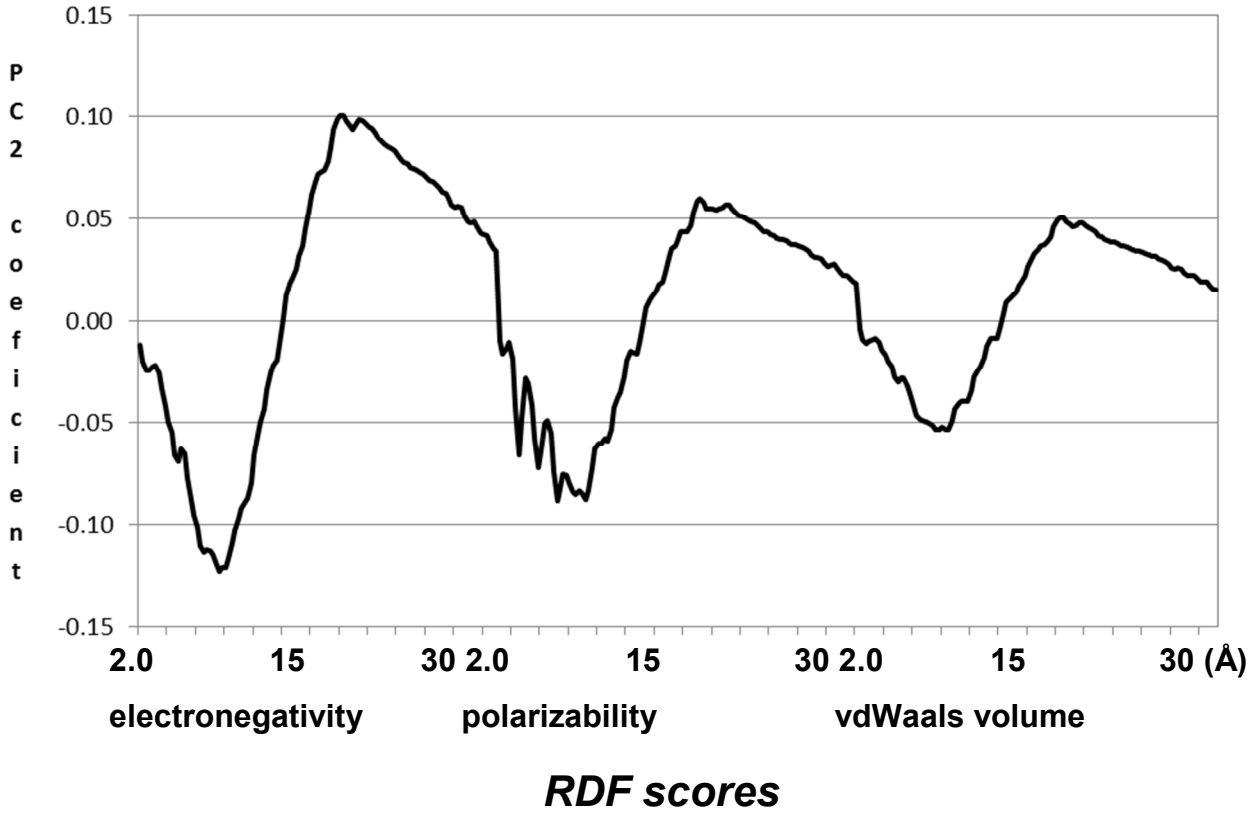

Figure S2. Coefficients (loadings) of the *AP-RDF* scores in the Principal Component 2 showing negative values for *RDF* scores in the short distance range of 2.5 to 12.5 Å.

7. Predictive Capability of the QSPR Model of CO_2 in Case 4 for fluorine-containing MOFs

We categorized the MOFs according to the functionalization of the organic SBUs to evaluate the prediction performance. We found that the QSPR model underestimated the CO_2 uptakes of fluorine-containing MOFs as depicted in Figures S3.

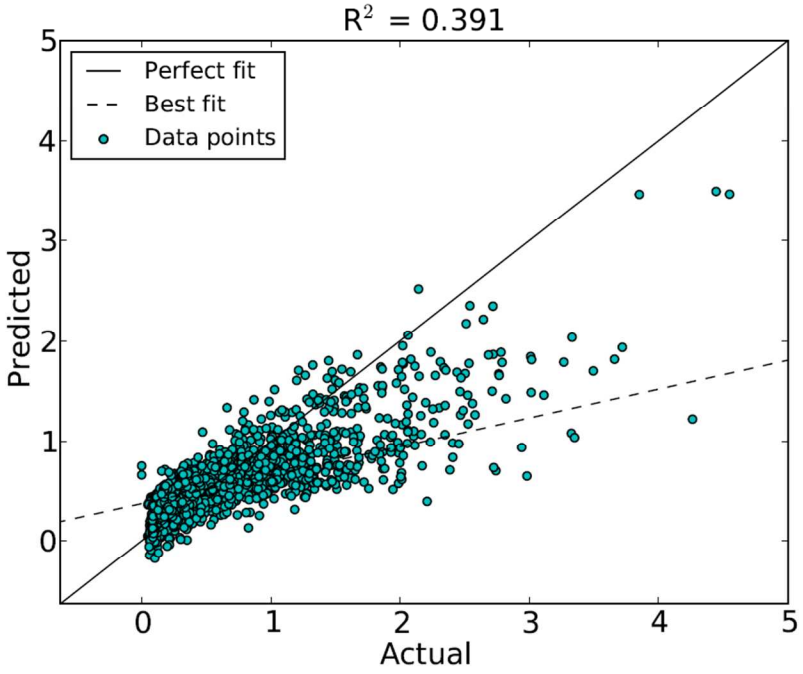

Figure S3. Scatter plots of QSPR predicted vs. actual (GCMC) uptake capacities of fluorine containing MOFs in the set using *AP-RDF* scores under VSA conditions relevant to flue gas separation (Case 4).

REFERENCES

(1) Wilmer, C. E.; Leaf, M.; Lee, C. Y.; Farha, O. K.; Hauser, B. G.; Hupp, J. T.; Snurr, R. Q. Large-scale screening of hypothetical metal-organic frameworks. *Nat. Chem.* **2012**, *4*, 83–89.

(2) Frost, H.; Düren, T.; Snurr, R. Q. Effects of surface area, free volume, and heat of adsorption on hydrogen uptake in metal-organic frameworks. *Journal of Physical Chemistry B* **2006**, *110*, 9565–9570.

(3) Duren, T.; Millange, F.; Ferey, G.; Walton, K. S.; Snurr, R. Q. Calculating Geometric Surface Areas as a Characterization Tool for Metal-Organic Frameworks. *J Phys Chem C* **2007**, *111*, 15350–15356.

(4) Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297.

(5) Frohlich, H.; Chapelle, O.; Scholkopf, B. Feature selection for support vector machines by means of genetic algorithm. In *Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence*; IEEE Comput. Soc, 2003; pages. 142–148.

(6) Henschel, S.; Zien, A.; Binder, A.; Gehl, C. The SHOGUN Machine Learning Toolbox. *JMLR* **2010**, *11*, 1799–1802.

(7) Jackson, J. E. A. *Users Guide to Principal Components*; Wiley: New York, 1991.

(8) Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; Vanderplas, J.; Passos, A.; Cournapeau, D.; Brucher, M.; Perrot, M.; Duchesnay, E. Scikit-learn: Machine Learning in Python . *JMLR* **2011**, *12*, 2825–2830.