Volume 124, Article No. 124018 (2019) https://doi.org/10.6028/jres.124.018
Journal of Research of National Institute of Standards and Technology

# *pyMCR: A Python Library for MultivariateCurve Resolution Analysis with Alternating Regression (MCR-AR)*

**Charles H. Camp Jr.**

National Institute of Standards and Technology,
Gaithersburg, MD 20899 USA

charles.camp@nist.gov

**Software DOI:** https://doi.org/10.18434/M32064

**Key words:** chemometrics; endmember extraction; multivariate curve resolution; quantitative analysis; spectral unmixing.

**Accepted:** June 18, 2019

**Published:** June 24, 2019

https://doi.org/10.6028/jres.124.018

## 1. Summary

*pyMCR* is a new open-source software library for performing multivariate curve resolution (MCR) analysis with an alternating regression scheme (MCR-AR). MCR is a chemometric method for elucidating measurement signatures of analytes and their relative abundance from a series of mixture measurements, without any knowledge of these values *a priori*. This software library, written in Python, enables users to perform MCR analysis with their choice of error functions for minimization, constraints, and regressors. Further, users can apply different constraints and regressors for signature and abundance calculations. Finally, this library enables users to develop their own constraints, regressors, and error functions or import them from existing libraries.



**How to cite this article:**
Camp CH (2019) pyMCR: A Python Library for MultivariateCurve Resolution Analysis with Alternating Regression (MCR-AR). *J Res Natl Inst Stan* 124:124018. https://doi.org/10.6028/jres.124.018.

## 2. Software Specifications

| | |
|---|---|
| **NIST Operating Unit** | Material Measurement Laboratory, Biosystems and Biomaterials Division, Biomaterials Group |
| **Category** | Chemometrics |
| **Targeted Users** | Users who wish to separate signature (such as spectra) and relative concentration from multiple measurements of mixtures with unknown or known signatures and concentrations. |
| **Operating Systems** | Cross-Platform |
| **Programming Language** | Python 3.x with NumPy and SciPy. Scikit-learn is optional. Sphinx is necessary to build local versions of the documentation. |
| **Inputs/Outputs** | This software is a command-line tool; thus, users will need to write their own code/methods to interface input data with *pyMCR*. The output is two NumPy arrays: signature of components, and relative concentration of each component. |
| **Documentation** | https://pages.nist.gov/pyMCR |
| **Accessibility** | N/A |
| **Disclaimer** | https://www.nist.gov/director/licensing |

## 3. Background

Multivariate curve resolution (MCR), also known as self-modeling mixture analysis (SMMA), is a chemometric method for analyzing data collected from mixtures, extracting the relative abundances and signatures of the pure analytes [1] (a process known as "endmember extraction"). One of the most common uses of MCR is for spectroscopy (and spectroscopic imaging) [2, 3], in which MCR is utilized to extract spectra and [relative] concentrations from series of chemical mixture spectra. In this context, endmember extraction is often referred to as "spectral unmixing". MCR has been applied to a variety of spectroscopies such as spontaneous and coherent Raman [2–5], infrared [2], ultraviolet-visible (UV-Vis) [6], near-infrared (NIR) [7], mass spectrometry [8], and utilized for a myriad of applications from liquid chromatography [9] and injection flow analysis [6] to biological/biomedical imaging [3–5].

Mathematically, the series of mixture measurements is described by a matrix $\mathbf{D} \in \mathbb{U}^{m,n}$, where $m$ is the number of independent measurements, $n$ is the number elements for a single measurement (e.g., spectral frequency bins), and $\mathbb{U}$ indicates the universal set of numbers, which may be real, imaginary, or complex. MCR aims to find a solution to (i.e. builds a model for):

$$\mathbf{D} = \mathbf{C}\mathbf{S}^T + \boldsymbol{\varepsilon} \tag{1}$$

where $\mathbf{C} \in \mathbb{U}^{m,p}$ is the abundance matrix of $p$-analytes, $\mathbf{S} \in \mathbb{U}^{n,p}$ is the signatures of the $p$-analytes, and $\boldsymbol{\varepsilon} \in \mathbb{U}^{m,n}$ describes noise, error, or aspects of the mixture not captured by the model. One should note that, besides $\mathbf{D}$, the user provides the number of analytes, $p$. The *pyMCR* package currently does not provide rank estimation methods to solve for $p$, which is an active area of research in many fields [10].

This software library uses an iterative alternating regression (AR) scheme to find a solution to Eq. (1), fixing $\mathbf{C}$ and performing multivariate multiple regression for $\mathbf{S}$ and *vice versa*, described mathematically as:

$$\mathbf{C}^{[k+1]} = \underset{\mathbf{C}^{[k]}}{\operatorname{argmin}}\, Q_C(\mathbf{C}^{[k]}, \mathbf{S}^{[k]}) \tag{2}$$

$$\mathbf{S}^{[k+1]} = \underset{\mathbf{S}^{[k]}}{\operatorname{argmin}}\, Q_S(\mathbf{C}^{[k+1]}, \mathbf{S}^{[k]}) \tag{3}$$

where $k$ is the iteration number, and $Q_C$ and $Q_S$ are objective functions (for the **C**-solve and **S**-solve iteration, respectively) to be minimized as will be described below. The iterations continue until a breaking condition, $L_B$, is met: a) the number of iterations meets a preset maximum, b) an error function (e.g., mean-squared error [MSE]) exceeds a defined value (or a number of iterations exceeding such a value), c) the relative change of the error function after each iteration does not reach a certain value, or d) a new local minimum for the error function is not met within a certain number of iterations. Algorithm 1 provides pseudocode of the main *pyMCR* optimization routine, where in addition to the previously described parameters, $L_C$ and $L_S$ are the constraint functions applied to **C** and **S** matrices (respectively). It should be noted that the $Q$-objective functions are applied implicitly by selection of regression methods. For example, ordinary least-squares (OLS) regression imparts an $L_2$-norm (residual sum-of-squares) objective function, which is constrained to non-negative values when selecting a non-negativity constrained least-squares (NNLS) regressor. Other methods, such as ridge and Lasso regressions, also use an $L_2$-norm in conjunction with regularization terms.

**Algorithm 1** Multivariate Curve Resolution - Alternating Regression (MCR-AR)

**Inputs:** $\mathbf{C}^{[0]}$ or $\mathbf{S}^{[0]}$; **D**; $Q$; $L_C$; $L_S$; $L_B$

**for** $k \leftarrow 0$ to $k_{max}$ **do**
  **if** $k > 0$ **or** $\mathbf{S}^{[0]}$ inputted **then**
    $\mathbf{C}^{[k+1]} \leftarrow \underset{\mathbf{C}^{[k]}}{\text{argmin}}\, Q(\mathbf{C}^{[k]}, \mathbf{S}^{[k]})$
    $\mathbf{C}^{[k+1]} \leftarrow L_C\{\mathbf{C}^{[k+1]}\}$
    **if** $L_B\left(\mathbf{C}^{[\cdot]}, \mathbf{S}^{[\cdot]}\right)$ is TRUE **then**
      **break**
    **end if**
  **else**
    $\mathbf{C}^{[k+1]} \leftarrow \mathbf{C}^{[k]}$
  **end if**
  $\mathbf{S}^{[k+1]} \leftarrow \underset{\mathbf{S}^{[k]}}{\text{argmin}}\, Q(\mathbf{C}^{[k+1]}, \mathbf{S}^{[k]})$
  $\mathbf{S}^{[k+1]} \leftarrow L_S\{\mathbf{S}^{[k+1]}\}$
  **if** $L_B\left(\mathbf{C}^{[\cdot]}, \mathbf{S}^{[\cdot]}\right)$ is TRUE **then**
    **break**
  **end if**
**end for**

It should be noted that the most common implementation of MCR uses an alternating least-squares approach and is referred to as "multivariate curve resolution-alternating least squares" (MCR-ALS) [2, 11]. These implementations use either OLS or NNLS regressors. Other examples include MCR-LASSO, which uses Lasso regression [12], or a penalized least-squares method (P-ALS) [13]. The *pyMCR* library, though, is more general thus we will refer to its implementation as "multivariate curve resolution-alternating regression" (MCR-AR). Additionally, as previously described, different regressors may be applied to calculating **C** and **S** solutions. Of note, another popular analysis method, "non-negative matrix factorization" (NMF) [14], is fundamentally the same as MCR when MCR applies non-negativity constraints to all solutions.

## 4. Methods for Validation

All functions and methods in the *pyMCR* library are validated by test files (i.e., unit testing, available in the repository "pymcr/tests" directory). Currently, unit test coverage is $> 98\%$.

## 5. Example Results

The following MCR implementations are meant to highlight some of the utility and flexibility of *pyMCR*. The source code used to generate the following examples is available within a Jupyter notebook in the "pymcr/Examples" folder of the repository. All examples use the following, simple dataset with each spectrum being composed of a single Gaussian waveform with a constant offset as shown in Fig. 1 (a), and concentration maps shown in Figs. 1 (b-d). The total concentration at each pixel as summed across the 3 components is 1 [Fig. 1 (e)], the so-called sum-to-one constraint (STO). Additionally, additive white Gaussian noise is applied at a level of ≈2 % of maximum. The spectra and concentration maps are combined as in Eq. (1) to form a hyperspectral image (HSI). **D** is formed by unraveling the spatial axes of the HSI.

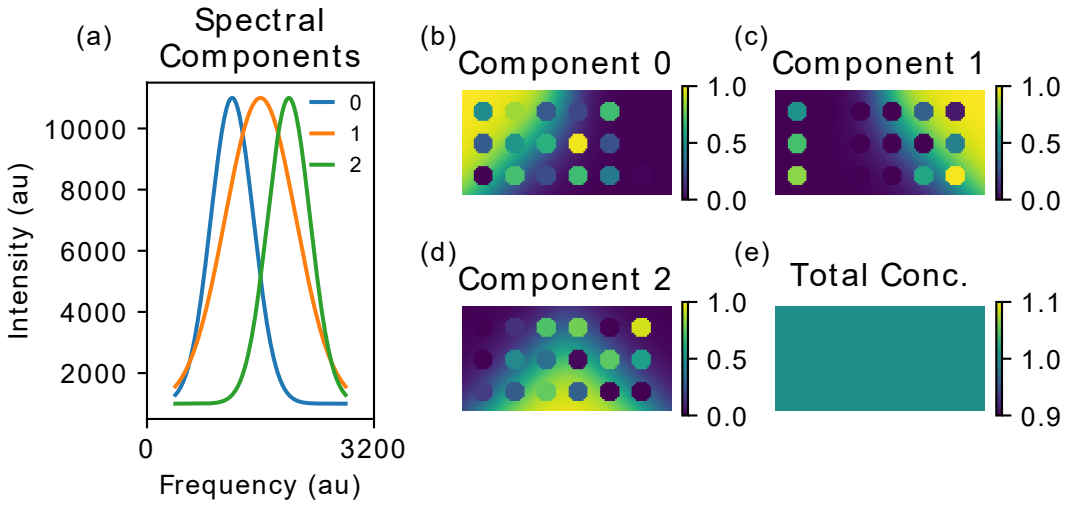


**Fig. 1.** Simulated dataset. au: arbitrary units. (a) Spectra of the 3 unique components. (b-d) Pseudocolor relative concentration maps. (e) Pseudocolor total relative concentration map (every pixel equals 1.0). All concentration maps are the same dimensions (50 pixels x 100 pixels) with the coloration depicting relative concentration (no units).

MCR methods require an initial guess for either **C** or **S**. Various strategies exist, such as random initialization, matrix decomposition [15], or expert input. In these examples, initial spectra ($\mathbf{S}^{[0]}$) are seeded from the absolute values of the first 3 right singular vectors from singular value decomposition (SVD), as shown in Fig. 2.

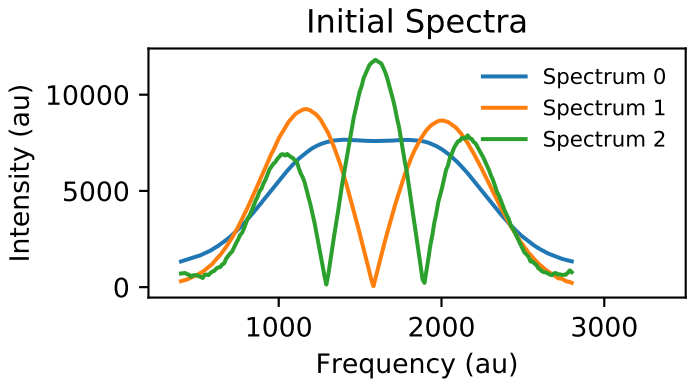


**Fig. 2.** All examples are spectrally seeded with the absolute values of the first 3 right singular vectors from SVD scaled by the maximum of the input dataset. au: arbitrary units.

In this manuscript, 5 different MCR-AR models were constructed, all with the STO constraint applied to **C** and non-negativity applied to **C** and **S**:

1. MCR-ALS: OLS regressors for **S** and **C**.
2. MCR-NNLS: NNLS regressors for **S** and **C**.
3. MCR-Gauss: MCR-NNLS with the additional constraint that spectra are Gaussian with a constant-valued baseline.
4. MCR-Ridge: **S** solved via ridge regression and **C** via OLS.
5. MCR-Lasso: **S** solved by OLS and **C** solved via positive Lasso regression.

Note that the use of "MCR-Lasso" above is just a short descriptor of the model and not the same implementation of MCR-Lasso provided in Ref. [12]. Each test serves to demonstrate different opportunities for using *pyMCR*. MCR-Gauss, for example, demonstrates the use of *ad hoc* constraints that can be constructed from an abstract "Constraint" class within *pyMCR*, in this case, using the *LMFIT* Python library [16] to fit a Gaussian lineshape with a constant baseline to each retrieved spectrum at each iteration. This is a potent method of enforcing *a priori* expert knowledge on the model as, in this case, one knows the spectra are each single Gaussians. MCR-Ridge and MCR-Lasso demonstrate native support for using regression methods directly imported from the *scikit-learn* Python library [17].

Under all tests, the error function was set to MSE and the breaking conditions were set to 1000 maximum iterations and a minimum change in MSE of $10^{-14}$. All other conditions, such as increases in MSE, were disabled.

Figure 3 presents the results of the MCR-AR fitting using the aforementioned designs. Figs. 3 (a-c) provide box plots calculated over all pixels and frequencies, which quantify the differences between the actual (known) and retrieved values. For each box plot, the whiskers detail the full span of values (no outliers), the filled box shows the interquartile range (IQR, $25^{th}$ to $75^{th}$ percentile), and the horizontal line identifies the median. In Figs. 3 (a,b), MCR-Gauss has the most optimal median and IQR; though, it has some larger extrema values than MCR-Lasso. The next most optimal methods are MCR-LASSO and MCR-NNLS. In Fig. 3 (c), however, MCR-NNLS is closest to 0 in all metrics save maximum extremum, with MCR-Lasso and MCR-Gauss being within 1 % for maximum extremum and IQR, and within 5 % for

minimum extremum. MCR-Lasso is within 5 % of MCR-NNLS for median value. Figure 3 (d) presents the MSE at each iteration for each implementation. MSE is calculated within the MCR-AR method at each iteration and reflects the ability of the model to capture the data, i.e. $\sum|\mathbf{D}-\mathbf{C}^{[\mathbf{k}]}(\mathbf{S}^{[k]})^T|^2$. Calculating the MSE of **D** with and without noise provides the "Ideal" level. MCR-ALS and MCR-Ridge terminated before 1000 iterations due to MSE changes of less than $10^{-14}$ per iteration. MCR-NNLS has the smallest MSE ≈1.1 % above ideal, followed by MCR-Lasso and MCR-Gauss at ≈1.8 % and ≈3.5 % above ideal, respectively. Figure 3 (e) describes the computational time to calculate each iteration and Fig. 3 (f) shows the number of iterations to reach the minimum MSE. To prevent terminating on local minima, there was no breaking condition set for increasing MSE; thus, the MSE at the final iteration could be larger than at previous iterations. *pyMCR* retains a copy of **C** and **S** matrices corresponding to the minimum error function even if continuing to iterate. For completeness, retrieval results for all methods are provided in the Appendix (Figs. 4-8).

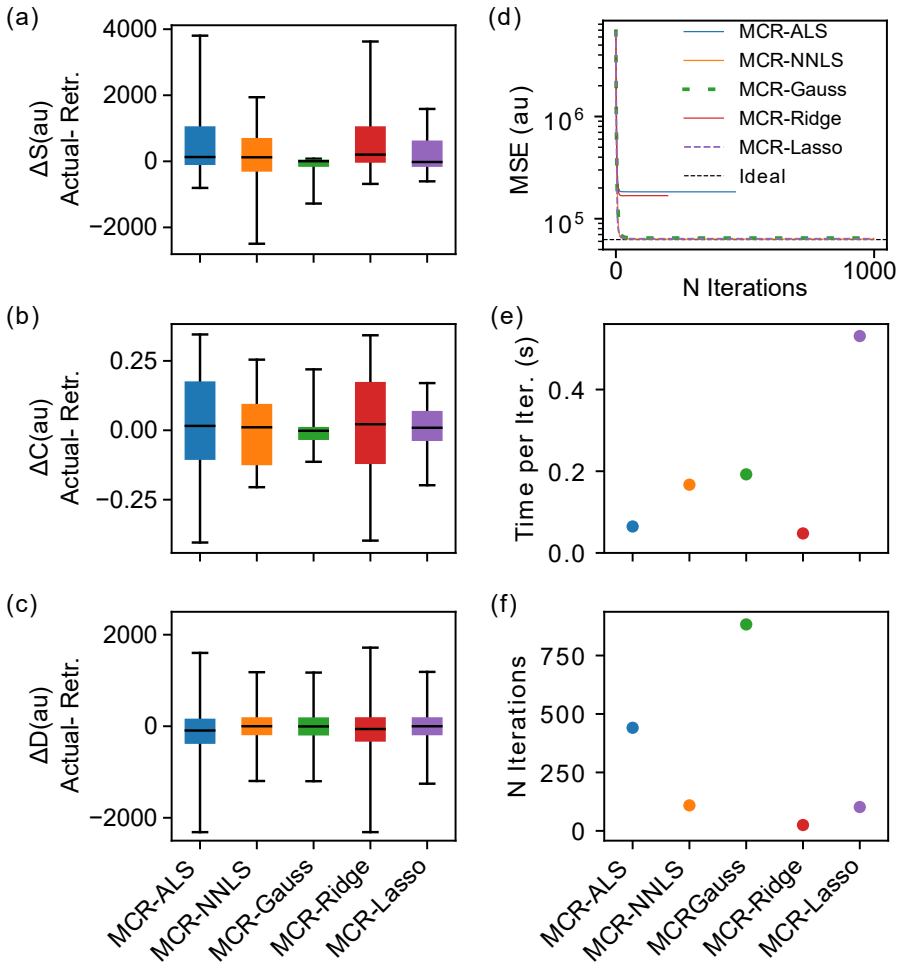


**Fig. 3.** Comparison of results across 5 MCR-AR models. Box plots comparing (a) per-analyte, per-frequency spectral differences ΔS, (b) per-pixel, per-analyte relative concentration differences ΔC, and (c) per-pixel, per-frequency spectral differences of the hyperspectral dataset ΔD. Box plot whiskers cover full range of values, box height spans IQR, and horizontal black line is the median. (d) MSE at each iteration for each method, which describes the ability of a model to capture the data. (e) The time per iteration (in seconds) for each method. (f) The number of iterations each method performed before reaching a local minimum or a breaking condition.

Fig. 3 indicate that MCR-Gauss, MCR-NNLS, or MCR-Lasso are likely optimal methods across the presented examples; though, MCR-NNLS and MCR-Gauss are significantly faster per iteration. Further improvement to MCR-Lasso could be attained, potentially, with optimization of the regularization weighting term (current value, $\alpha$=1). It should be emphasized that the optimal method(s) in these examples are rather particular to the dataset and the input spectral guesses. Improved initial spectra inputs could dramatically alter the outcomes as could different types of input data.

## 6. Appendix

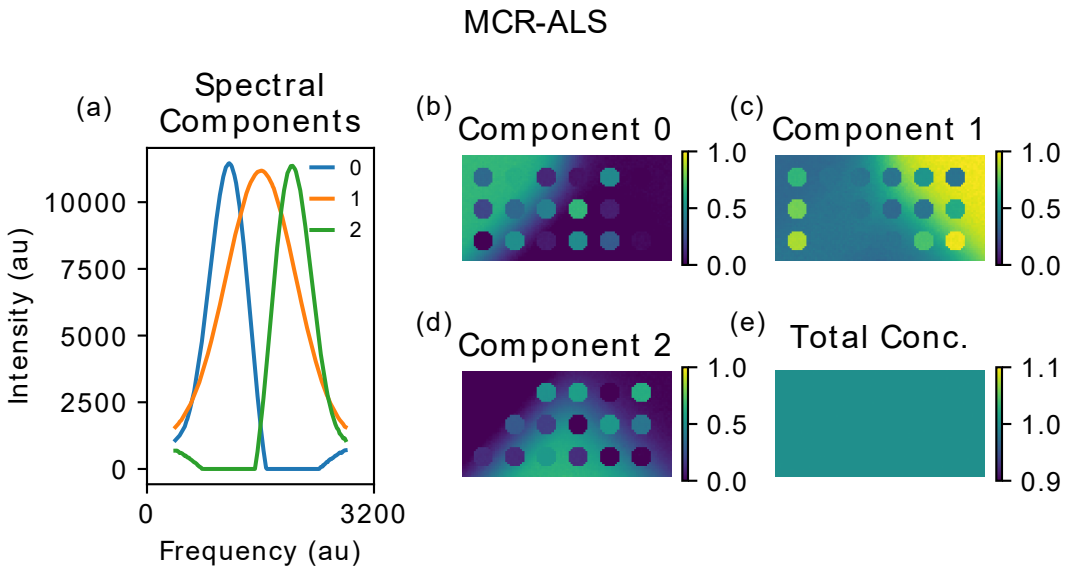


**Fig. 4.** Retrieved dataset using MCR-ALS. au: arbitrary units. (a) Retrieved spectra of the 3 unique components. (b-d) Pseudocolor relative concentration maps. (e) Pseudocolor total relative concentration map. All images are the same spatial dimensions (50 pixels x 100 pixels) with the coloration depicting concentration in arbitrary units.

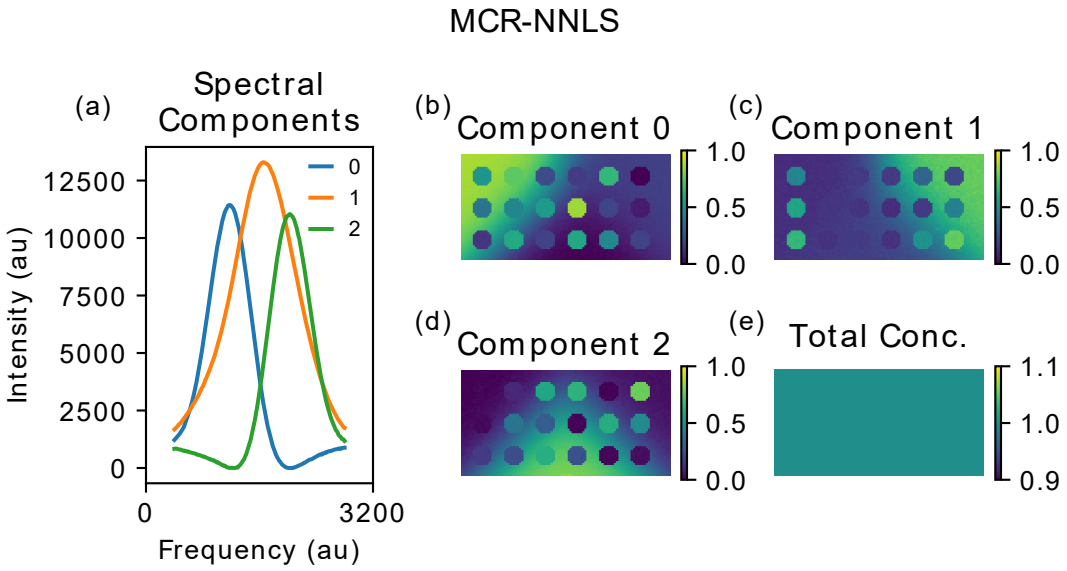


**Fig. 5.** Retrieved dataset using MCR-NNLS. au: arbitrary units. (a) Retrieved spectra of the 3 unique components. (b-d) Pseudocolor relative concentration maps. (e) Pseudocolor total relative concentration map. All images are the same spatial dimensions (50 pixels x 100 pixels) with the coloration depicting concentration in arbitrary units.

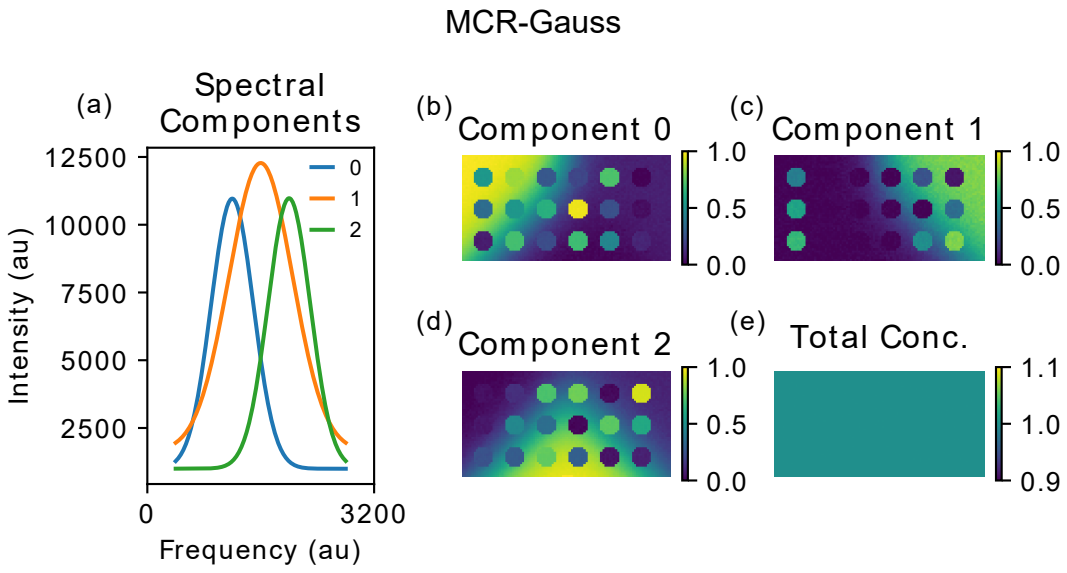


**Fig. 6.** Retrieved dataset using MCR-Gauss which enforces a single Gaussian fit with constant baseline per spectrum fit. au: arbitrary units. (a) Retrieved spectra of the 3 unique components. (b-d) Pseudocolor relative concentration maps. (e) Pseudocolor total relative concentration map. All images are the same spatial dimensions (50 pixels x 100 pixels) with the coloration depicting concentration in arbitrary units.

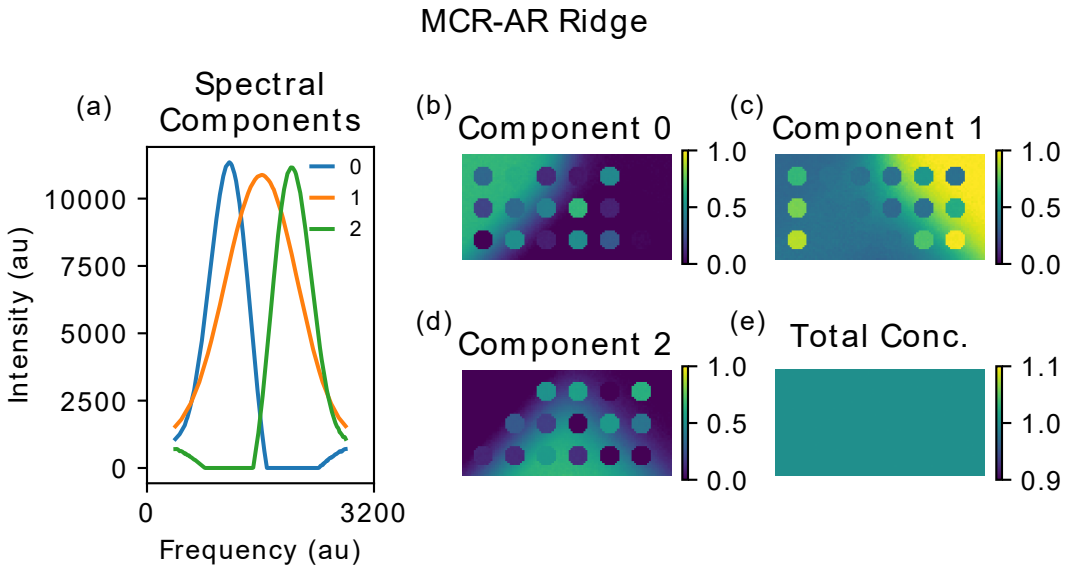


**Fig. 7.** Retrieved dataset using MCR-Ridge which performs ridge regression on spectra (**S**). au: arbitrary units. (a) Retrieved spectra of the 3 unique components. (b-d) Pseudocolor relative concentration maps. (e) Pseudocolor total relative concentration map. All images are the same spatial dimensions (50 pixels x 100 pixels) with the coloration depicting concentration in arbitrary units.

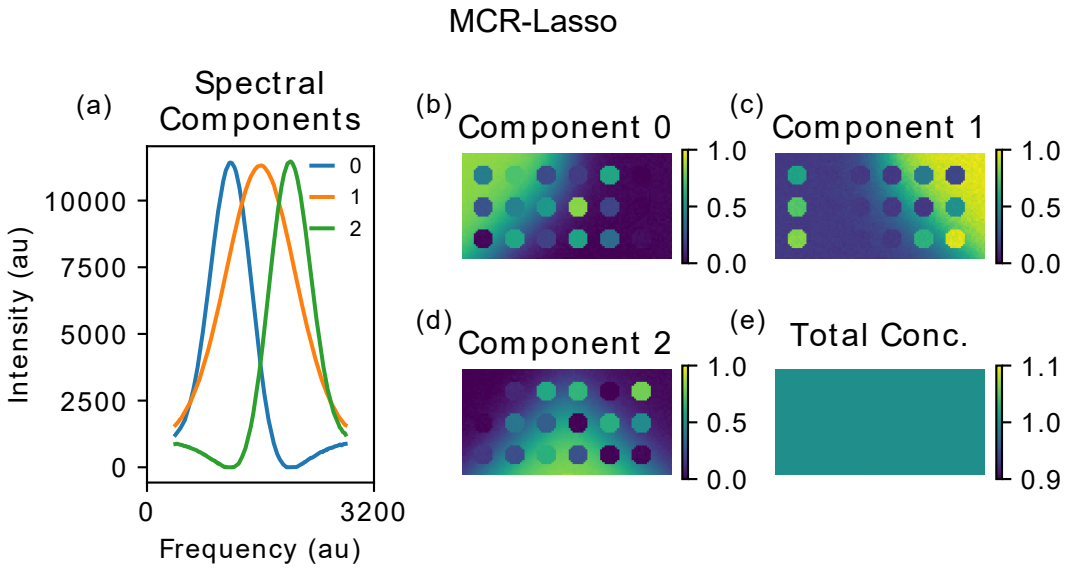


**Fig. 8.** Retrieved dataset using MCR-Lasso which performs Lasso regression on concentration (**C**). au: arbitrary units. (a) Retrieved spectra of the 3 unique components. (b-d) Pseudocolor relative concentration maps. (e) Pseudocolor total relative concentration map. All images are the same spatial dimensions (50 pixels x 100 pixels) with the coloration depicting concentration in arbitrary units.

**Acknowledgments**

The author would like to acknowledge the contributions of the open-source community for finding bugs,

improving documentation, and hastening the adoption of *pyMCR*. Additionally, the author wishes to thank Nancy Lin and Young Jong Lee of NIST for their helpful discussion and critiques of the manuscript.

## 7. References

[1] Lawton WH, Sylvestre EA (1971) Self Modeling Curve Resolution. *Technometrics* 13(3):617. https://doi.org/10.2307/1267173
[2] Felten J, Hall H, Jaumot J, Tauler R, De Juan A, Gorzsás A (2015) Vibrational spectroscopic image analysis of biological material using multivariate curve resolution-alternating least squares (MCR-ALS). *Nature Protocols* 10(2):217–240. https://doi.org/10.1038/nprot.2015.008
[3] Zhang D, Wang P, Slipchenko MN, Ben-Amotz D, Weiner AM, Cheng JX (2013) Quantitative vibrational imaging by hyperspectral stimulated raman scattering microscopy and multivariate curve resolution analysis. *Analytical Chemistry* 85(1):98–106. https://doi.org/10.1021/ac3019119
[4] Ando M, Hamaguchi Ho (2013) Molecular component distribution imaging of living cells by multivariate curve resolution analysis of space-resolved Raman spectra. *Journal of Biomedical Optics* 19(1):011016. https://doi.org/10.1117/1.JBO.19.1.011016
[5] Patel II, Trevisan J, Evans G, Llabjani V, Martin-Hirsch PL, Stringfellow HF, Martin FL (2011) High contrast images of uterine tissue derived using Raman microspectroscopy with the empty modelling approach of multivariate curve resolution-alternating least squares. *The Analyst* 136(23):4950–9. https://doi.org/10.1039/c1an15717e
[6] Saurina J, Hernández-Cassou S (2001) Quantitative determinations in conventional flow injection analysis based on different chemometric calibration statregies: a review. *Analytica Chimica Acta* 438(1):335–352. https://doi.org/10.1016/S0003-2670(01)00862-5
[7] Azzouz T, Tauler R (2008) Application of multivariate curve resolution alternating least squares (mcr-als) to the quantitative analysis of pharmaceutical and agricultural samples. *Talanta* 74(5):1201–1210. https://doi.org/10.1016/j.talanta.2007.08.024
[8] Peré-Trepat E, Tauler R (2006) Analysis of environmental samples by application of multivariate curve resolution on fused high-performance liquid chromatography–diode array detection mass spectrometry data. *Journal of Chromatography A* 1131(1):85–96. https://doi.org/10.1016/j.chroma.2006.07.047
[9] Tauler R, Barceló D (1993) Multivariate curve resolution applied to liquid chromatography—diode array detection. *TrAC Trends in Analytical Chemistry* 12(8):319–327. https://doi.org/10.1016/0165-9936(93)88015-W
[10] Bioucas-dias JM, Nascimento JJMP, Bioucas Dias JM, Nascimento JJMP (2008) Hyperspectral Subspace Identification. *IEEE Transactions on Geoscience and Remote Sensing* 46(8):2435–2445. https://doi.org/10.1109/TGRS.2008.918089
[11] Jaumot J, Gargallo R, De Juan A, Tauler R (2005) A graphical user-friendly interface for MCR-ALS: A new tool for multivariate curve resolution in MATLAB. *Chemometrics and Intelligent Laboratory Systems* 76(1):101–110. https://doi.org/10.1016/j.chemolab.2004.12.007
[12] Pomareda V, Calvo D, Pardo A, Marco S (2010) Hard modeling multivariate curve resolution using lasso: Application to ion mobility spectra. *Chemometrics and Intelligent Laboratory Systems* 104(2):318–332. https://doi.org/10.1016/j.chemolab.2010.09.010
[13] Gemperline PJ, Cash E (2003) Advantages of soft versus hard constraints in self-modeling curve resolution problems. Alternating least squares with penalty functions. *Analytical Chemistry* 75(16):4236–4243. https://doi.org/10.1021/ac034301d
[14] Lee DD, Seung HS (1999) Learning the parts of objects by non-negative matrix factorization. *Nature* 401(6755):788–91. https://doi.org/10.1038/44565
[15] Boutsidis C, Gallopoulos E (2008) SVD based initialization: A head start for nonnegative matrix factorization. *Pattern Recognition* 41(4):1350–1362. https://doi.org/10.1016/j.patcog.2007.09.010
[16] Newville M, Stensitzki T, Allen DB, Ingargiola A (2014) LMFIT: Non-Linear Least-Square Minimization and Curve-Fitting for Python. https://doi.org/10.5281/zenodo.2620617
[17] 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 (2011) Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12:2825–2830. Available at http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

***About the author:*** *Charles H. Camp Jr. is an electrical engineer within the Biomaterials Group of the Biosystems and Biomaterials Division at NIST. His primary research focuses on the development of biological and chemical imaging modalities using nonlinear optics, primarily broadband coherent anti-Stokes Raman scattering (BCARS) microspectroscopy. The National Institute of Standards and Technology is an agency of the U.S. Department of Commerce.*