Vespucci: A free, cross-platform tool for spectroscopic imaging

Daniel P. Foose*

Wright State University

E-mail: foose.3@wright.edu

*To whom correspondence should be addressed

Please cite this documentation as a book if you use Vespucci in a published work. A BibTeX citation follows:[1]

```
@Book{Vespucci,
Title = {Vespucci: a free, cross-platform tool for spectroscopic imaging},
Author = {Foose, D. P.},
Year = {2015},
Publisher = {Wright State
University}
Address = {Dayton, OH, USA}
Url ={http://wright.edu/~foose.3/Vespucci}
}
```

This citation is given in ACS format as citation 1.

Contents

1 About Vespucci

Vespucci is a software application developed for imaging and analysis of hyperspectral datasets. Vespucci offers several advantages over other software packages, including a simple, easily learned user interface, no cost, and less restrictive licensing. Vespucci expands several analysis techniques including univariate imaging, principal components analysis, partial-least-squares regression, and vertex components analysis with endmember extraction, and k-means clustering. Additionally, Vespucci can perform a number of useful data-processing operations, including filtering, normalization, baseline correction, and background subtraction. Datasets that consist of spatial or temporal data with a corresponding digital signal, including spectroscopic images, mass spectrometric images, and X-ray diffraction data can be processed in this software. Vespucci is written in C++ and makes use of the MLPACK, Armadillo , Qt, and QCustomPlot libraries. Vespucci is a graphically-driven package that is designed with ease-of-use in mind and is equally capable to other available tools. Vespucci's capabilities are extended by interfaces to Octave and R to allow existing research code to be run from a common environment. Additionally, Vespucci's C++ classes can be used to construct more specialized programs when an application programming interface (API) is desired. The source code and a Windows binary distribution can be accessed at https://github.com/dpfoose/Vespucci.

Vespucci is designed with several main goals in mind:

- Mathematical transparency. All operations are logged and reference internal function names. The code is entirely opensource so users may examine the internals of the program
- Ease of use. The Vespucci UI is designed to be as easy to use as possible to remove barriers between researchers and results.
- Extensibility. Vespucci uses the Armadillo and MLPACK libraries, which

have very intuitive APIs for novice C++ programmers familiar with MATLAB or Octave. This makes modification of the software and contribution to its development fairly easy. Vespucci also includes external code interfaces for R and Octave, allowing for the use of existing research code.

- Versatility. Vespucci can theoretically handle any dataset consisting of spatial data combined with a corresponding digital signal.

operating systems.

1.1 Windows

Portable packages for Windows are provided and updated regularly. These packages include all files necessary to run the program. This version is built as an x86-64 binary (executable on just about any computer with 64-bit Windows) and uses the OpenBLAS library (which is included in the package) for some matrix computations. To use other LAPACK/BLAS replacements in Windows, you must compile the program from source.

1.2 Mac OS X

Builds for Mac are periodically released. These consist of a .app file which can be installed by dragging the file to the ”Applications” folder. The Mac versioin relies on Apple’s Accelerate framework for BLAS and LAPACK routines. The Mac version of the software is

1.3 GNU/Linux

GNU/Linux binaries may be periodically released.

1.4 Compiling from Source

Vespucci uses Qt's QMake build system. These relases are updated to reflect the research needs of my group. Compiling Vespucci requires Qt, Armadillo, LAPACK, and BLAS (or a replacement for LAPACK and BLAS such as OpenBLAS). The MinGW_libs and Mac_libs branches, when checked-out with a source branch, should make compilation straightforward. If you are building Vespucci from source, you probably understand how this works. Vespucci consists of three separate subprojects which must be compiled. The Vespucci library is compiled first. The Vespucci GUI program links to this library. The Vespucci Octave interface is compiled separately by the mkoctfile system in Octave.

several formats: long text, wide text, and Vespucci binary (which is an Armadillo binary file format).

1.5 Dataset Format

Internally, Vespucci datasets are stored as `VespucciDataset` objects. The data is stored in this object as two spatial column vectors, `x_` and `y_`, a spectral abscissa vector `abscissa_`, and a spectral data matrix `spectra_`. Each column of `spectra_` represents a spectrum, where each element is the intensity (or absorbance, or transmittance, etc.) at the spectral abscissa value at the same index in `abscissa_`. The values of `x_` and `y_` at that row index correspond to the spatial position at which the spectrum was recorded. The spectral data matrix is an $m \times n$ matrix which can be viewed as n observations of m variables for the purpose of multivariate analysis (this is in contrast to the row-major nature of matrices in R).

1.6 Importing Datasets

To import a dataset, select ”Import Dataset from File” from the ”File” menu of the main window. Several options are provided. The axis labels for the spectra and their units may be changed in this screen. These values will be displayed in the spectra viewer.

1.6.1 From Text Files

Text files in two formats may be imported. In the ”wide” format each row represents a spectrum. The first two columns are the values for the two spatial coordinates. The first row represents the spectral abscissa (wavelength or wavenumber). In the ”long” format, the data is stored in four columns. The first two columns store the spatial variables. The third column stores a single spectral abscissa key value. The fourth column is the value (intensity, transmittance, absorbance, etc.). Some spectrometers utilizing the ”wide” format (such as some Horiba Jobin Yvon instruments) will store y in the first column and x in the second column. This can be accounted for by checking the box ”Swap x and y?” in the data import view.

1.6.2 From Previously Saved Vespucci Objects

Vespucci can export processed datasets using the Armadillo binary format. These are stored is `arma::field<arma::mat>` objects and take the extension *.*vds*. Previously saved datasets may be imported.

1.6.3 From Individual Point Spectra

Vespucci can create datasets from individual point spectra

1.7 Saving Dataset Elements

The following dataset elements can be saved from the main window (”Export Dataset Elements” from the ”Data” menu):

- Spectra
- Average Spectra
- Average Spectra (with abscissa)
- All Data

Elements may be saved as CSV or tab-delimited ASCII for viewing or processing in other software, or as Armadillo binary to be used by Vespucci for other purposes (such as background subtraction). Other dataset elements can be saved using the Data Viewer (”View Dataset Elements” in the Data menu). Larger dataset elements which may not be displayed in the Data Viewer can be saved by selecting ”Large Matrices...” from the

variety of basic data processing methods.

1.8 Background Subtraction

Background subtraction subtracts a vector of the same dimensions as the spectra from every spectra in the map. The input must be in the armadillo binary format, but use of text format will be available in a later release.

1.9 Baseline Correction

Baseline correction estimates the baseline of the spectra and subtracts this baseline from every spectra. Vespucci supports median filter background correction and improved modified polynomial fitting (IModPoly or the ”Vancouver Raman Algorithm”).

1.10 Filtering

Filtering reduces spectral noise. Several methods are available: Moving Average, Median, Savitzky-Golay Smoothing and SVD. Support for median filtering is currently experimental and seems to be dependent on which compiler is used. With the exception of SVD, all filtering methdos rely on taking some value of a ”window”, a subset of the spectral signal of a given size, centered on the value that will contain the filtered data.

1.10.1 Median Filtering

The value of a filtered data point is equal to the median of the window. Window size is the only adjustable parameter.

1.10.2 Moving Average Filtering

The value of the filtered data is equal to the average value of the window. Window size is the only adjustable parameter.

1.10.3 Savitzky-Golay Smoothing

The value of the filtered data is equal to the value of the fit of a polynomial function to the window. The derivative of this polynomial can also be taken. Adjustable parameters are the window size, the polynomial order and the derivative order.

1.10.4 Truncated Singular Value Decomposition

A *singular value decomposition (SVD)* is a factorization of a matrix M such that $M = U\Sigma V'$. The diagonal entries of Σ are referred to as *singular values*. A *truncated* SVD is an SVD where only the k highest magnitude singular

values are calculated. The highest magnitude singular values are responsible for the highest amount of variance in the matrix. The matrix can then be reconstructed from the truncated SVD, maintaining most of its variance while removing noise. This is especially useful for relatively discrete noise such as cosmic ray spikes. This procedure reduces the rank of the spectra matrix to the number of singular values selected.

1.10.5 QUIC-SVD

The QUIC-SVD method performs a SVD, but with the intent of maintaining a certain amount of variance rather than maintaining a certain number of singular values.

1.11 Normalization

Normalization manipulates the spectra so that the sum of the intensities of the spectra equal some value. This is achieved by dividing each value of each spectra by a weight:

$$y_{i,\tilde{\nu}} = \frac{x_{i,\tilde{\nu}}}{w_i}$$

1.11.1 Unit Area Normalization

Unit area normalization divides every point of each spectra by the sum of all the points of that spectrum, resulting in a plot where the area under the curve is 1 for every spectra. This removes information about the difference in total intensity between spatial points.

1.11.2 Min/Max Normalization

Min/Max Normalization subtracts the smallest value from every element of the `spectra_` matrix, then uses the overall maximum of each spectra as

weights. This results in a range of spectral intensities from 0 to 1. This retains all information about the spectra except for the absolute intensities.

1.12 Peak Intensity Normalization

Peak intensity normalization divides all points in each spectra by the maximum value in a particular wavelength range for that spectrum so that the value at that peak is 1.

1.12.1 Vector Normalization

Vector normalization uses the Euclidean norm of each spectra vector as weights.

1.12.2 Standard Normal Variate

Standard normal variate normalization scales the spectra by the standard deviation so that each spectrum has a unit standard deviation:

$$w_i = \sigma_i + \delta$$

This scaling may also be offset by a user-specified value (δ) which is reflective of instrument noise.

1.12.3 Z-score Normalization

Z-score normalization, similarly to standard normal variate normalization, scales each spectrum by its standard deviation. Additionally, Z-score nor-

malization centers each value by the mean of its spectrum. A Z-score:

$$Z_{i,\tilde{\nu}} = \frac{x_{i,\tilde{\nu}} - \bar{x}_i}{\sigma_i}$$

is calculated for each wavelength or wavenumber λ of each of the i spectra in the dataset, and replaces that point in the dataset with the Z-score of that point relative to the spectrum. If the instrument noise is estimated, an offset may be added to the standard deviation:

$$Z_{i,\tilde{\nu}} = \frac{x_{i,\tilde{\nu}} - \bar{x}_i}{\sigma_i + \delta}$$

1.12.4 Absolute Value Normalization

Absolute value normalization replaces every value with its absolute value.

1.12.5 Spectra Scaling

Scaling uses a user-specified value as the weight for all spectra. This number may be positive or negative. This can be used to reflect the spectra about the abscissa.

1.12.6 Mean Centering

Mean centering subtracts from each value the mean of all values in the spectra matrix at that value's row index (abscissa value):

$$x_{i,\tilde{\nu}} = x_{i,\tilde{\nu}} - \mu_{\tilde{\nu}}$$

This is commonly used before principal component analysis.

1.13 Univariate Peak Determination

Vespucci has four methods for univariate peak determination, intensity, bandwidth, area, and derivative.

1.13.1 Intensity

The intensity method calculates the local maximum within the region of interest.

1.13.2 Bandwidth

The bandwidth method calculates the full width at half maximum (FWHM) of the curve in the region of interest relative to the local baseline, defined as a straight line through the points on the left and right boundaries of the region of interest. The region of interest is used only to find a local maximum of the spectrum; bandwidths may be wider than the region of interest. The bandwidth of a peak can be viewed as a measure of its convolution. This may be useful in estimating the relative composition of a material where characteristic peaks of two different materials overlap.

1.13.3 Area

The area method determines the score for a region of interest by calculating the area between the curve in that region and the local baseline. The fastest method (Riemann sum) takes the sum of the differences between the points in that region and the local baseline, defined as a straight line through the points on the left and right boundaries of the region of interest. This sum is multiplied by the distance between points (in spectral abcissa units), to give the area under the curve. Peak fitting may be availible in later releases.

1.13.4 Derivative

The derivative method detects the edges of the peaks close to the peak region.

2 Multivariate Imaging and Analysis

Vespucci is capable of several multivariate analysis techniques. These methods reduce the variables in the data set from many (the multiple values of the spectral abcissa) to several (the principal components). Principal components analysis (PCA), vertex component analysis (VCA) and partial least-squares (PLS) regression reduce the total number of variables for each data point. Crisp clustering techniques order each spectra into one of a few discrete categories (clusters), essentially reducing the dataset to 3 dimensions (x, y, and cluster number).

These can be performed as imaging operations (Map->New Multivariate Map) or as analysis operations that do not produce maps (Data->Multivariate Analysis).

2.1 Principal Component Analysis

PCA is performed using a complete singular value decomposition. This involves finding the eigenvalues of a large sparse matrix formed by the matrix and its transpose, then a large matrix multiplcation to find principal components. For large matrices, this process is computationally expensive. For this reason, the program may appear to freeze during this process. On a 2 GHz Intel Core 2 Duo with 4 GB RAM on Windows 8.1, this process takes about a minute for a 3524×1694 matrix. Vespucci uses the Armadillo function `arma::princomp` to perform PCA.

2.1.1 Imaging

PCA images are formed from the component coefficients. The component number is selected. Areas where that component is most responsible for the result will be lightest in color.

2.1.2 Data Views

The projected data, principal component coefficients, covariance matrix eigenvalues and t^2 values may be viewed and saved from the Data Viewer (Data->View Dataset Elements). This data can be saved as tab-delimited text, which is easily imported in MATLAB or comma-delimited text (CSV), which can be opened in Excel. The ability to save HDF5 objects may be availible in later releases.

2.2 Vertex Component Analysis

VCA reduces the spectra matrix to a particular number of components referred to as *endmembers*.[2] Each of these endmembers represents a particular spectrum that is responsible for a certain amount of variance in the dataset, usually larger than 95Vespucci's implementation of this algorithm is found in `arma_ext::VCA()`.

2.2.1 Imaging

VCA images are based on the extent to which an endmember is responsible for the spectrum.

2.2.2 Data Views

Plots of endmembers may be viewed and saved from the Data Viewer (Data->View Dataset Elements). Statistics related to each endmember may also be saved.

2.3 Partial Least Squares Regression

PLS is another method for reducing the number of variables in the spectra matrix. PLS provides similar results to PCA, but with less computational expense. To maintain similarity to the MATLAB code used previously by my group, PLS is implemented using the SIMPLS algorithm.[3] Vespucci's implementation of this algorithm is found in `arma_ext::plsregres()`.

2.3.1 Imaging

PLS Images, like PCA images, are formed based on the coefficient.

2.3.2 Data Views

PLS statistics may be viewed and saved from the Data Viewer (Data->View Dataset Elements).

2.4 Crisp Clustering

2.4.1 *k*-Means Clustering

k-Means clustering orders every spectra into one of several categories based on the *k*-nearest neighbors algorithm. Vespucci perfoms this using `mlpack::kmeans::Km`

3 Viewing and Saving Images and Spectra

Images can be saved in a variety of formats (File->Save Image As from the map viewer). Double-clicking on a map name in the map list will open or close the map window.

3.1 The Map Viewer

3.1.1 The Display Menu

Images created by Vespucci are created to fairly arbitrary pre-defined size (designed to make a 50 by 50 map fit in the middle third of a 1280x720 screen. Normally, resizing the window is locked. Deselecting the ”Lock Size” option allows the window to be resized. To make the window size proportional to the spatial data, select the ”Reproportion” option.

Image interpolation will smooth the edges of the points, producing a more visually appealing image. However, this interpolation may create a false sense of increased resolution.

The axes and color scale can be enabled or disabled.

The map can be attached to the global color scale.

3.1.2 The Data Menu

A new dataset can be produced from a map by selecting (Data->New Dataset from Map...). The new dataset will contain only those spectra with a map value between the two bounds specified.

3.1.3 The Spectrum Viewer

When the Spectrum Viewer is open (Display->Spectrum Viewer), the spectrum at a particular point in the map may be viewed by clicking on the map. The bottom left corner displays the spatial data as an X-Y pair. The bottom center displays the value of the map at that point. The Export button can be used to save the image of the spectrum or the spectrum itself as CSV, text, or binary.

3.1.4 Saving Images

Images are saved using (File->Save Image As...). Everything visible in the Map Viewer will be saved.

References

(1) Foose, D. P. *Vespucci: a free, cross-platform tool for spectroscopic imaging*; Wright State University, 2014.

(2) Nascimento, J. M. P.; Bioucas Dias, J. M. *IEEE Transactions on Geoscience and Remote Sensing* **2005**, *43*, 898–910, cite this for VCA.

(3) de Jong, S. *Chemometrics and Intelligent Laboratory Systems* **1993**, *18*, 251–263, cite this for PLS.