Two scatter correction techniques for NIR spectroscopy in Python

07/21/2018

Ever wanted to master the art of NIR calibration but got lost in the acronyms? Struggling with getting simple explanations of the basics? Well then, this post is for you and today we'll work through two scatter correction techniques for NIR spectroscopy in Python. We'll discuss functions to perform **Multiplicative Scatter Correction (MSC)** and

Standard Normal Variate (SNV) on near-infrared (NIR) data. Three acronyms in one sentence, that's a good start.

Data pre-processing is an essential step to build most (ahem, all) types of calibration models in NIR and most spectroscopy analysis. With a well-designed pre-processing step, the performance of the model can be greatly improved.

Before we move on, if you want to take a look at some of our posts on related topics, here you go.

- NIR classification with LDA
- Partial Least Square
- Principal Component Regression

Pre-processing techniques can be generally divided into **1) Scatter Corrections** techniques and **2) Derivative** techniques. Both MSC and SNV belong to the first category.

In this post we are going to introduce the problem of scatter correction, describe the details and write some code to perform both MSC and SNV, then apply both method on some real data. A **Jupyter notebook** containing the code described in this post is available at our Github repository.

Correcting scattering effects in NIR

NIR spectra contain a mix of diffuse and specular reflectance (or straight transmittance). Three main factors affecting the shape of each spectrum are:

1. Different wavelengths of the incident light experience different **absorption** by the sample, due to the chemical nature of the sample itself. In most cases this is the signal we want to measure, and it relates to the analyte of interest.
2. Differences in particle size in the material will cause light to be deviated at different angles depending on its wavelength. **Scattering effects** (particle size), along with possible differences in path length constitute the major causes of variations in NIR spectra.
3. Path length differences from sample to sample due to variations in positioning and/or irregularities in the sample surface.

Scattering effects can be both additive and multiplicative. **Additive effects** (such as path length differences) produce a baseline displacement of the spectrum along the vertical axis, while **multiplicative effects** (particle size for instance) modify the local slope of the spectrum.

The idea behind scattering corrections is to get rid of all effects that are unrelated to the chemical nature of the sample, but just depend on the sample morphology and the measurement geometry. So, the idea goes,

if we are able to remove these undesirable effects beforehand, we should get a better model for the quantity of interest.

As always, that is easier said than done. In practice it may be extremely difficult to separate scattering from absorbance effects, and the methods developed by the community tend to be approximations that are valid under specific assumptions.

Having said that however, years of practice showed that both MSC and SNV often do a good job in improving the quality of the calibration model and indeed are two of the most common, yet simple, pre-processing techniques for NIR data.

Let's dive into it!

Multiplicative Scatter Correction in Python

MSC requires a reference spectrum to start with. This is the most important difference between MSC and SNV. The reference spectrum is ideally a spectrum free of scattering effects.

Now, as you can gather, getting our hands on a spectrum that is free of unwanted scattering effects is not easy, definitely not across all wavelengths we are interested in. For this reason, if the data is reasonably well behaved, we can take the average spectrum to be a close approximation to the ideal spectrum we are after. Particle size and path length effects should vary randomly from sample to sample, and therefore the average should reasonably reduce these effects, at least in

the approximations that these effects are genuinely random. This is the main assumption behind MSC.

Mathematically, if we call X_m the mean spectrum, the multiplicative scatter correction is done in two steps.

1. We first regress each spectrum X_i against the mean spectrum. This is done by ordinary least squares: $X_i \approx a_i + b_i X_m$.
2. We calculate the corrected spectrum $X_i^{\mathrm{msc}} = (X_i - a_i)/b_i$.

In general all these spectra have a non-zero mean, and therefore we can optionally mean-centre the spectra beforehand.

Here's a Python function to perform MSC.

```
def msc(input_data, reference=None):
    ''' Perform Multiplicative scatter correction'''

    # mean centre correction
    for i in range(input_data.shape[0]):
        input_data[i,:] -= input_data[i,:].mean()

    # Get the reference spectrum. If not given, estimate it from the mean
    if reference is None:
        # Calculate mean
        ref = np.mean(input_data, axis=0)
    else:
        ref = reference

    # Define a new array and populate it with the corrected data

```

```
    data_msc = np.zeros_like(input_data)
    for i in range(input_data.shape[0]):
        # Run regression
        fit = np.polyfit(ref, input_data[i,:], 1, full=True)
        # Apply correction
        data_msc[i,:] = (input_data[i,:] - fit[0][1]) / fit[0][0]

    return (data_msc, ref)
```

Note that, in addition to the corrected spectra, it is good practice to return the mean spectrum too. That can be used on a second data set (for instance validation data) as a reference correction that is consistent with the first set of spectra.

Standard Normal Variate in Python

Unlike MSC, SNV correction is done on each individual spectrum, and a reference spectrum is not required. The SNV correction can be divided in two conceptual steps as well.

1. Mean centre each spectrum X_i by taking away its mean $\bar{X}_i$
2. Divide each mean centred spectrum by its own standard deviation: $X_i^{\text{snv}} = (X_i - \bar{X}_i)/\sigma_i$.

Translated into Python the previous algorithm looks something like this

```
def snv(input_data):

    # Define a new array and populate it with the corrected data
    output_data = np.zeros_like(input_data)
    for i in range(input_data.shape[0]):

```

```
        # Apply correction
        output_data[i,:] = (input_data[i,:] - np.mean(input_data[i,:])) / np.std(
    return output_data
```

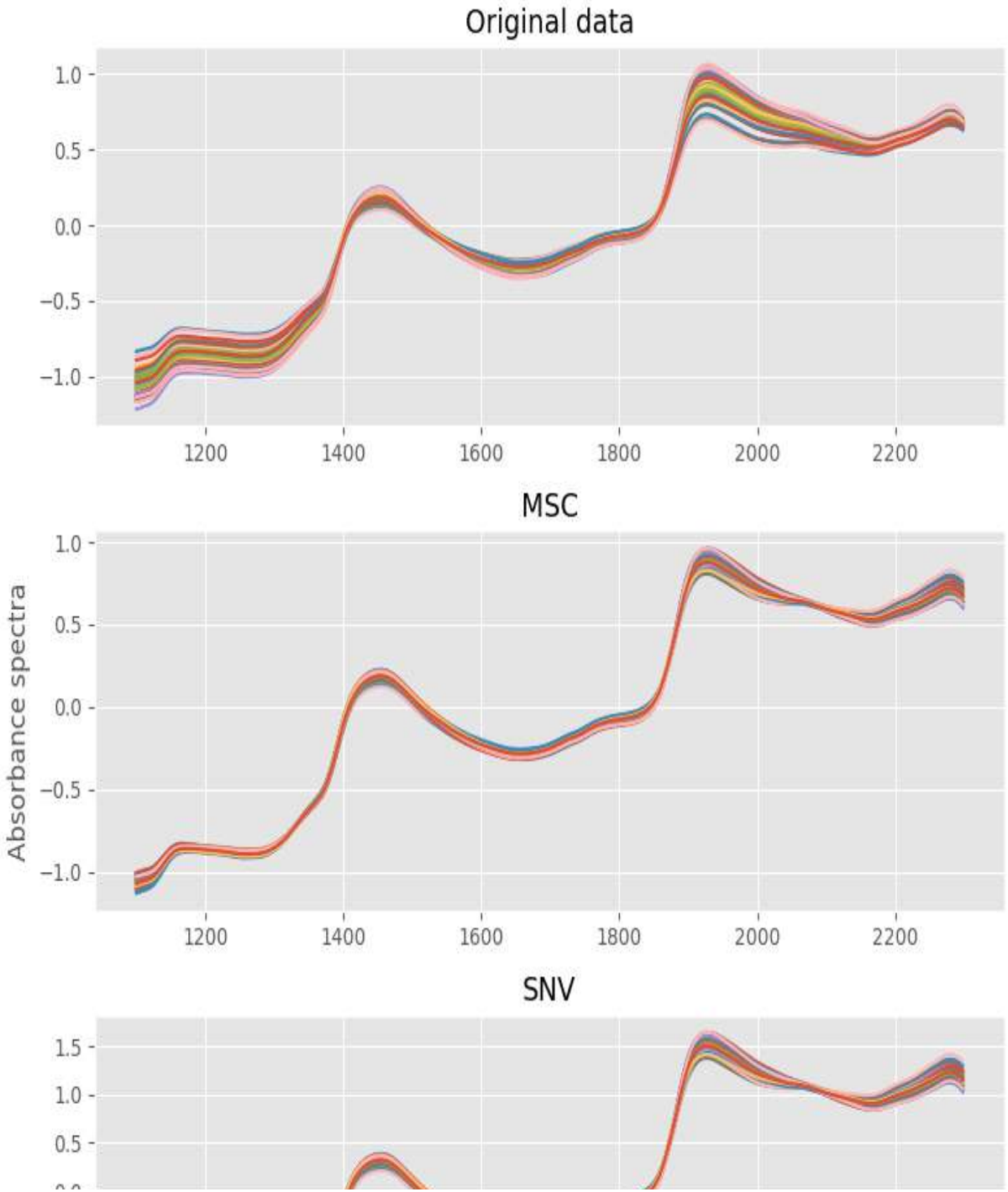
Putting it all together

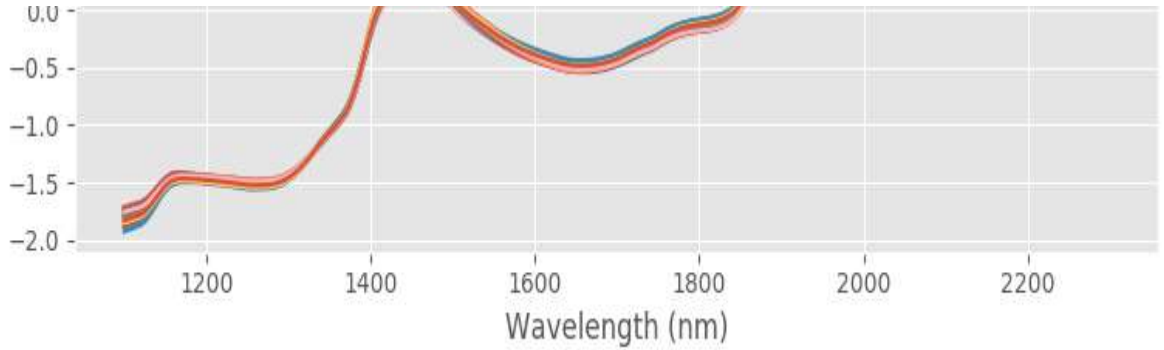
To test these functions, run the following code. The data is available for download at our Github repository.

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

# import data and define wavelengths
data = pd.read_csv('./data/peach_spectra+brixvalues.csv')
X = data.values[:,1:]
wl = np.arange(1100,2300,2)

Xmsc = msc(X)[0] # Take the first element of the output tuple
Xsnv = snv(X)

## Plot spectra
plt.figure(figsize=(8,9))
with plt.style.context(('ggplot')):
    ax1 = plt.subplot(311)
    plt.plot(wl, X.T)
    plt.title('Original data')

    ax2 = plt.subplot(312)
    plt.plot(wl, Xmsc.T)
    plt.ylabel('Absorbance spectra')
    plt.title('MSC')

    ax2 = plt.subplot(313)
    plt.plot(wl, Xsnv.T)
    plt.xlabel('Wavelength (nm)')
    plt.title('SNV')

```

```
        plt.show()
```

The data set is the same that I've used in past articles. It comes from a NIR reflectance experiment on fresh peaches (50 samples). The output is


Original data
MSC
SNV
Absorbance spectra
1.0
0.5
0.0
−0.5
−1.0
1.5
1200
1400
1600
1800
2000
2200



Indeed the two methods give an almost identical result, at least in this case. As it has been first shown by Danoha *et al.* in this paper (if you can't get a copy, feel free to contact me), the two methods are indeed related by a linear transformation.

In closing, one of the advantage of MSC is that can relate all spectra to a common reference. If the reference spectrum is, to a good approximation, close to a spectrum free of unwanted scattering, this is a pretty good choice. Conversely, if outliers are present then the mean

spectrum may not be a good reference and the MSC is not a good choice. In these cases SNV can be opted for instead.

Thanks for reading and until next time!

About The Author

Daniel Pelliccia
Physicist and entrepreneur. Founder of Instruments & Data Tools, specialising in custom sensors and analytics. Founder of Rubens Technologies, the crop intelligence system.

Related Posts