Contents

pandas.Series Cheat Sheet

The spectra of containers read by sensor will be stored in sensors_output in Pandas.Series structure, here is a quick reference for frequently used API. For more information, please refer to pandas documentation on https://pandas.pydata.org/docs/reference/api/pandas.Series.html

Retrieve spectra

```
>>> print(sensors_output)

    # A nested dictionary includes Sensor Type, Sensor Location, Spectra of c
ontainers
    {1: {'type': <SpectrumType.FTIR: 'FTIR'>, 'location': 500, 'spectrum': 36
00   -0.000053
    3598   -0.000049
    3596   -0.000015
    3594    0.000019
    3592    0.000051
            ...
    1252    0.103183
    1250    0.102646

>>> print(sensors_output[sensor_id]['spectrum'])

    # Spectrum is stored in pandas.Series structure. sensor_id should be set
to 1.
    3600    0.001952
    3598    0.001722
    3596    0.001528
    3594    0.001590
            ...
    1252    0.253636
    1250    0.285266
    Name: PC, Length: 1176, dtype: float64
```

Access spectra name

```
>>> print(sensors_output[sensor_id]['spectrum'].name)

    # Return the name of the Series. The name of the plastic is only passed t
o the sorting function in training mode. In testing mode, the name of the con
tainer will be 'unknown_plasitc'
    PC
```

Access all transmittance (y-value)

```
>>> print(sensors_output[sensor_id]['spectrum'].values)

    # Return Series as ndarray or ndarray-like depending on the dtype.
    [0.00195171 0.00172183 0.00152848 ... 0.2209626  0.2536359  0.2852665 ]
```

Access all wavenumbers (x-value)

```
>>> print(sensors_output[sensor_id]['spectrum'].keys())

    # Return alias for index.
    Int64Index([3600, 3598, 3596, 3594, 3592, 3590, 3588, 3586, 3584, 3582,
                ...
                1268, 1266, 1264, 1262, 1260, 1258, 1256, 1254, 1252, 1250],
               dtype='int64', length=1176)
OR
    # You could also convert it to a ndarry using .values
>>> print(sensors_output[sensor_id]['spectrum'].keys().values)
    [3600 3598 3596 ... 1254 1252 1250]
```

Access transmittance at a specific wavenumber

```
>>> print(sensors_output[sensor_id]['spectrum'].get(3600))

    # Get item from object for given key
    0.001952
OR
>>> print(sensors_output[sensor_id]['spectrum'].iloc[0])

    # Indexing for selection by position.
    0.001952
OR
>>> print(sensors_output[sensor_id]['spectrum'].loc[3600])

    # Access a group of rows and columns by label(s) or a boolean array.
    0.001952
```

Access transmittance based on a slice of wavenumbers

Refer to Python **slice notation**:
https://stackoverflow.com/questions/509211/understanding-slice-notation

```
>>> print(sensors_output[sensor_id]['spectrum'].iloc[0:5])

    3600    0.001952
    3598    0.001722
    3596    0.001528
    3594    0.001590
    3592    0.001453
    Name: PC, dtype: float64
OR
>>> print(sensors_output[sensor_id]['spectrum'].loc[3600:3592])

    3600    0.001952
    3598    0.001722
    3596    0.001528
    3594    0.001590
```

```
    3592    0.001453
    Name: PC, dtype: float64
```

Access the maximum transmittance and/or the corresponding wavenumber

```
>>> spectrum = sensors_output[sensor_id]['spectrum']
>>> print(spectrum.idxmax(), spectrum.max())

    # .max() Return the maximum of the values over the requested axis.
    # .idxmax() Return index of first occurrence of maximum over requested ax
is.
    1712 0.4544385
OR
>>> print(spectrum.sort_values(ascending=False))

    # Sort a Series in ascending or descending order by some criterion.
    1712    0.454439
    1710    0.447742
    1714    0.444016
            ...
    3574   -0.000589
    3576   -0.000607
    Name: Polyester, Length: 1176, dtype: float64
```

Access the minimum transmittance and/or the corresponding wavenumber

```
>>> spectrum = sensors_output[sensor_id]['spectrum']
>>> print(spectrum.idxmin(), spectrum.min())

    # .min() Return the minimum of the values over the requested axis.
    # .idxmin() Return index of first occurrence of minimum over requested ax
is.
    3576   -0.000607

OR
>>> print(spectrum.sort_values(ascending=True))
    # Sort a Series in ascending or descending order by some criterion.
      3576   -0.000607
      3574   -0.000589
             …
      1714    0.444016
      1710    0.447742
      1712    0.454439
```

Access local extrema of transmittance and the wavenumber

```
# You should import argrelextrema from scipy.signal module by using pip
install scipy command. After downloading it, you could copy and paste the
code to main.py to test argrelextrema function.

from scipy.signal import argrelextrema
import numpy as np
from matplotlib import pyplot as plt
```

```
def user_sorting_function(sensors_output):
     # random identification
     decision = {sensor_id: random.choice(list(Plastic)[0:-1]) for (sensor_id,
value) in sensors_output.items()}

    # retrieve spectrum
     sensor_id = 1
     spectrum = sensors_output[1]['spectrum']

    # exclude the blank spectra
     if spectrum.iloc[0] == 0:
        decision = {sensor_id: Plastic.Blank}
     else:
        # comparator = np.greater or np.less, stand for local maxima or
minima
         # order = n, means how many points on each side to use for the
comparison to consider
         # [0] at the end is to access the entire array of local extrema
wavenumbers, based on the structure of return value
         n = 10
         iloc_max_wavenumbers = argrelextrema(spectrum.values,
comparator=np.greater, order=n)[0]


        # Plot the spectrum
         spectrum.plot()

        # Plot the local maximum points on the spectrum you just plotted
         spectrum.iloc[iloc_max_wavenumbers].plot(title= spectrum.name,
style="v", color="red")

       # Press Ctrl+C or Ctrl+Z in terminal to interrupt the plot windows
         plt.show()
```

Do math on pandas.Series data

- Average Spectrum

```
    # First calculate the average value from raw_spectrum
 >>> average_value = raw_spectrum.mean()
     # Retrieve the list of wavenumbers from raw_spectrum
 >>> wavenumbers_list = list(raw_spectrum.keys())
     # Then create the corresponding list of average value (transmittance)
>>> average_value_list = [average_value for i in range(len(wavenumbers_list))]
     # Construct an average spectrum using the average value list and
wavenumber list into a pandas.Series
```

```
>>> average_spectrum = pd.Series(data=average_value_list,
index=wavenumbers_list)
```

- Addition

```
>>> spectrum_addition = raw_spectrum + average_spectrum
```

- Subtraction

```
>>> spectrum_subtraction = raw_spectrum - average_spectrum
```

- Sum Squared Error (Summation, square, subtraction)

```
>>> import numpy as np
>>> np.sum(np.square(raw_spectrum - average_spectrum))
```

- Comparison

```
    # Greater than
 >>> spectrum.gt(average_PP_spectrum))
    # Less than
 >>> spectrum.lt(average_PP_spectrum))
    # Greater than or equal to
 >>> spectrum.ge(average_PP_spectrum))
    # Less than or equal to
 >>> spectrum.le(average_PP_spectrum))

     # These four eturn a pandas.Series with bool datatype
```

Concatenate two series

```
>>> import pandas as pd
>>> pd.concat([average_PP_spectrum, spectrum])

     # Return a pandas.Series with average_PP_spectrum attached by spectrum
```

Import data from excel

```
# If you manipulated data in excel (e.g. set a threshold transmittance) and
would like to import it to Python for comparison, you could refer to the
following lines of codes

import random

from rcplant import *
import os
import pandas as pd
import numpy as np

def import_excel():
     # First need to put excel file 'demo_import.xlsx' under the same folder
as main.py
     data_file = os.path.join(os.path.dirname(__file__), 'demo_import.xlsx')
    # Use pandas to read the entire table
     data_table = pd.read_excel(data_file, sheet_name=0, index_col=0)
```

```
    # Use .loc[] the spectra you want to retrieve
    average_spectrum = data_table.loc['PP_AVERAGE']


 # OR
    # You could also use raw spectrum and calculate the average by .mean()
    raw_spectrum = data_table.loc['PP']
    # Calculate the average value
    average_value = raw_spectrum.mean()
    # Retrieve wavenumbers list from raw_spectrum
    wavenumbers_list = list(raw_spectrum.keys())
    # Then create the corresponding average value (transmittance) list
    average_value_list = [average_value for i in
range(len(wavenumbers_list))]
    # Construct the average value list and wavenumber list into a
pandas.Series
    average_spectrum_2 = pd.Series(data=average_value_list,
index=wavenumbers_list)
    return average_spectrum_2

def user_sorting_function(sensors_output):
    # random identification
    decision = {sensor_id: random.choice(list(Plastic)) for (sensor_id,
value) in sensors_output.items()}

    # If you have made some data in excel (e.g. average PP spectrum)
    # and want to import to sorting function
    average_PP = import_excel()
    print(f'\nImport excel successfully: average_PP is {average_PP}')

    return decision
```

Plot spectra

Refer to here for matlibplot module: https://pandas.pydata.org/pandas-docs/version/0.13.1/visualization.html. A plot of spectra was attached to the end of Simulator Guide on Avenue

```
>>> from matplotlib import pyplot as plt
# using matplotlib module to show the plot
>>> plt.figure()
>>> sensors_output[sensor_id]['spectrum'].plot()
# .plot() Make plots of Series or DataFrame.
>>> plt.show()
```

Export spectra to excel

```
>>> sensors_output[sensor_id]['spectrum'].to_excel()
# .to_excel()
```