JASON BeautifulJASONによるPython活用の紹介

関連製品：核磁気共鳴装置(NMR)

現在、JASON[1]ではHDF5ファイルの階層構造とBeautifulJASONの活用により、Python[2]によるNMRスペクトル解析が効率的に行えます。
本資料では、以下の2つの内容について紹介します：

1. BeautifulJASONの概要とPythonによるNMRスペクトル解析の例
2. JASON×BeautifulJASON×Pythonによるバッチ処理の使用例

1. BeautifulJASONの概要とPythonによるNMRスペクトル解析の例

BeautifulJASONの概要

BeautifulJASONは、JASONおよびそのファイル形式（.jjh5）に対応した、Python用のインターフェースライブラリです。これにより、JASONで作成・保存されたスペクトルデータを、Pythonから直接操作・解析することが可能になります。
このライブラリは、以下のような用途に適しています：

- スペクトルの自動処理・解析
- レイアウトやレポートのカスタマイズ
- バッチ処理やラボの自動化
- データ解析パイプラインへの統合

特に、JASONを日常的に使用しているユーザーが、プログラムから柔軟にスペクトルデータを扱いたい場合に有効です。Pythonスクリプトを通じてJASONの機能を呼び出すことで、GUI操作とは異なる一括処理や統計解析、可視化などが可能になります。

BeautifulJASONを用いたNMRデータの抽出と解析例（1ファイル）

ここでは、BeautifulJASONを活用した、Pythonによる簡単なNMRスペクトル解析の例を紹介します。解析の流れは、Table 1のワークフローで構成されています。各ステップの内容と使用ライブラリはTable 1に、計算結果はTable 2に示しています。この例は、BeautifulJASONの基本的な使い方を理解し、スペクトル解析を試していただく際の参考になります。

Table 1. 1ファイルNMRデータの解析ワークフロー例

ステップ	内容	使用ライブラリ	代表コード例
データ読み込み	.jjh5ファイルの読み込み	beautifuljason	doc = bjason.Document("file.jjh5", mode='r')
情報抽出	スペクトル情報、ピーク、積分値の取得	beautifuljason, numpy	for m in spectrum.multiplets: print(m.pos[0], m.value_hz)
統計計算	平均化学シフト、積分値の総和など	numpy	np.mean(peak_positions), np.sum(integral_values)
純度計算	基準物質との比較による純度算出	numpy	purity = (I / Istd) * (Nsstd / Ns) * ...
可視化・表形式	ピークごとの純度を表で表示	pandas, matplotlib	plt.bar(peak_positions, peak_purities) pd.DataFrame(...).to_string()

Table 2. 解析例

Chemical Shift (ppm)	Purity (%)
7.6	99.4170
6.7	99.4340
4.0	99.3399
1.5	99.3364
1.2	99.8782
0.7	99.6659
平均純度	**99.5119**

BeautifulJASONによるNMRデータの読み込みと情報抽出

Figure 1に示すコードは、.jjh5[3]形式のNMR測定データから、スペクトル情報やピーク情報を抽出する処理の一例です。BeautifulJASONの基本的な使い方を理解する上で、参考になる内容です。このコードでは、以下のような情報を取得しています：

- スペクトルのタイトルや核種情報
- 多重線（multiplet[4]）の化学シフト位置と積分値
- スペクトル構造（nmr_data や nmr_items）

このような処理は、スペクトル解析の前処理やデータ整理に不可欠であり、BeautifulJASONの強力な抽出機能を活用することで、Pythonベースの解析がスムーズに行えます。

BeautifulJASONの公式ドキュメント

より詳細な情報やAPIリファレンスは、以下の公式サイトをご参照ください：
https://www.jeoljason.com/beautifuljason/docs/

日本電子株式会社
本社・昭島製作所
〒196-8558 東京都昭島市武蔵野3−1−2 TEL：(042) 543−1111(大代表) FAX：(042) 546−3353
www.jeol.co.jp ISO 9001・ISO 14001 認証取得

```python
import beautifuljason as bjason

# JASON形式のNMRデータファイルを読み込みます（読み取りモード）
with bjason.Document(input_1H_file, mode='r') as doc:

    # すべてのスペクトルデータ（nmr_data）についてタイトルを表示 → スペクトルに付けられたラベルや説明を確認できます
    for spec in doc.nmr_data:
        title = bjason.utils.ensure_str(spec.spec_info.get_param("Title"))  # タイトル情報を文字列として取得（バイト列の可能性があるため整形）
        print(title)

    # NMRアイテム（nmr_items）ごとにヘッダーとスペクトル情報を表示 → 複数のスペクトルが含まれる場合それぞれの核種とタイトルを確認できます
    for nmr_item in doc.nmr_items:
        print(nmr_item.header)  # アイテムのヘッダー情報
        for spec in nmr_item.spec_data_list:
            nuclide = spec.spec_info.nuclides[0]  # 例: '1H' や '13C'
            title = bjason.utils.ensure_str(spec.spec_info.get_param("Title"))
            print("  ", nuclide, title)

    # スペクトル内のマルチプレット情報を表示 → 化学シフト（ppm）とその強度（Hz）を確認できます
    for spectrum in doc.nmr_data:
        for multiplet in spectrum.multiplets:  # マルチプレット（ピーク）の位置（ppm）と強度（Hz）を取得
            print(f"ppm: {multiplet.pos[0]}, value: {multiplet.value_hz}")
```

Figure 1. BeautifulJASONによるNMRデータ抽出コード例

BeautifulJASONを用いた複数NMRデータの統計解析例

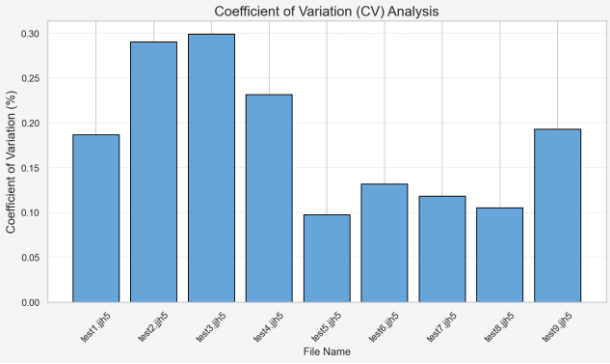
ここでは、複数の.jjh5形式のNMRデータを対象に、定量値の統計解析を行った例を紹介します。BeautifulJASONを用いて各ファイルからピーク情報を抽出し、定量計算を行った後、変動係数（CV）によるばらつき評価（Figure 2）、KDEプロットによる分布の可視化（Figure 3）、QQプロットによる正規性の確認（Figure 4）、さらに相関分析による要因解析（Figure 5）を行いました。これらの手法により、測定のばらつきや分布特性、変数間の関係性を定量的に評価できます。本解析は、測定の再現性評価やサンプル間の比較に有効です。解析ステップとそのポイントについてはTable 4にまとめています。

Table 4.複数NMRデータ解析ステップと考察ポイント

ステップ	処理内容	使用ライブラリ・モジュール	代表コード例	考察ポイント
① 定量計算	ピーク積分値から純度を算出	beautifuljason, numpy	purity = (I / Istd) * …	測定値から定量的な純度を導出
② CV評価	平均純度と標準偏差から変動係数を算出	numpy, matplotlib	cv = std / mean * 100	測定のばらつきや再現性を評価
③ KDEプロット	純度分布の可視化	seaborn, matplotlib	sns.kdeplot(purities, …)	分布の偏りや広がりを視覚的に把握
④ QQプロット	純度分布の正規性を確認	scipy.stats, matplotlib	stats.probplot(purities, …)	統計解析の前提条件（正規性）を検証
⑤ 相関分析	純度と他変数の関係性を評価	scipy.stats	pearsonr(x, y) spearmanr(x, y)	変数間の影響や傾向を探索

変動係数（CV）によるばらつき評価

複数のNMRデータファイルに対して、ピーク純度の平均値と標準偏差をもとに変動係数（CV：Coefficient of Variation）を算出しました。CVは、平均値に対する標準偏差の割合を示す指標で、測定値のばらつきや再現性を評価する際に用いられます。Figure 2に示すように、test3.jjh5 や test2.jjh5 ではやや高めのCVが見られ、test5.jjh5 や test8.jjh5 ではばらつきが小さい傾向が確認されました。




```python
import beautifuljason as bjason
# ファイルを読み込み
with bjason.Document(input_1H_file, mode='r') as doc:
    # multiplets プロパティからピーク情報を取得
    peak_positions = [multiplet.pos[0] for spectrum in doc.nmr_data
                      for multiplet in spectrum.multiplets]
    integral_values = [multiplet.value_hz for spectrum in doc.nmr_data
                       for multiplet in spectrum.multiplets]
```

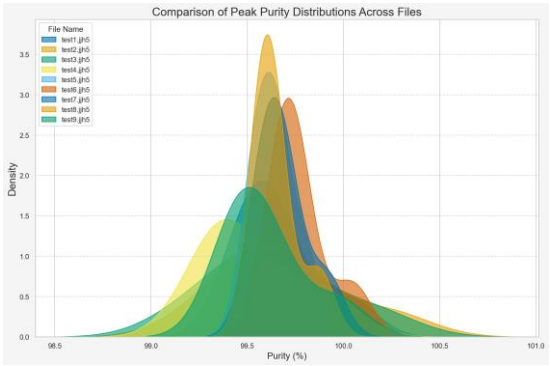
```python
    import numpy as np
    # 平均純度と変動係数を計算
    average_purity = np.mean(peak_purities)
    std_purity = np.std(peak_purities)
    cv_purity = std_purity / average_purity * 100  # 変動係数(%)
```

Figure 2. 変動係数による比較、変動係数計算コード例の抜粋

KDEプロットによるデータ分布の可視化

Figure 3は各NMRファイルのピーク純度をもとに、カーネル密度推定（KDE：Kernel Density Estimation）を用いて分布を可視化した結果です。KDEプロットは、データの分布を滑らかな曲線で可視化する手法で、ヒストグラムよりも詳細な分布傾向を把握できます。純度のばらつきや偏りを視覚的に把握するのに有効です。例えば、分布の幅が広いデータは、純度のばらつきが大きい可能性があり、分布が狭いデータは測定が安定していることを示します。複数ファイルの分布を重ねることで、測定条件やサンプル間の違いを比較することができます。

JEOL 日本電子株式会社
本社・昭島製作所
〒196-8558 東京都昭島市武蔵野3－1－2 TEL：(042) 543－1111(大代表) FAX：(042) 546－3353
www.jeol.co.jp ISO 9001・ISO 14001 認証取得

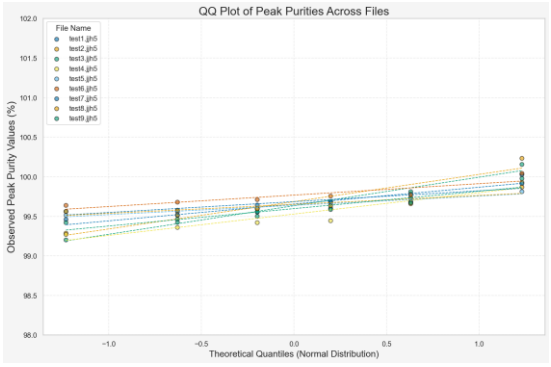



```python
import seaborn as sns
# KDEプロットの作成
for idx, (file_name, purities) in enumerate(file_peak_purities.items()):
    sns.kdeplot(
        purities,
        fill=True,
        label=file_name,
        alpha=0.6,
        color=color_palette[idx % len(color_palette)]  # 対応色の選択
    )
```

Figure 3.ピーク純度分布の比較（KDEプロット）、KDEプロット生成コード例の抜粋

QQプロットによる正規性の確認

Figure 4は各NMRファイルのピーク純度について、正規分布との一致度をQQプロットで評価した結果です。QQ（Quantile-Quantile）プロットは、データの分布が理論的な正規分布に従っているかを視覚的に確認するための手法です。理論的な正規分布の分位点と実測値の分位点を比較します。プロットが直線に近いほど、データは正規分布に近い性質を持ちます。外れ値がある場合、プロットの端が大きく逸れる傾向があります。今回のデータでは、いずれも直線性が高く、今後の統計解析において正規性の前提が妥当であることが確認されました。




```python
import scipy.stats as stats
# 各ファイルでQQプロットを作成
for file_name, purities in file_peak_purities.items():
    plt.figure(figsize=(10, 6))
    stats.probplot(purities, dist="norm", plot=plt)  # 正規分布と比較
    plt.title(f"QQ Plot of Peak Purities for {file_name}")  # ファイル名をタイトルに追加
    plt.xlabel("Theoretical Quantiles (Normal Distribution)")  # 理論分布の説明
    plt.ylabel("Observed Peak Purity Values (%)")  # 実測値
    plt.grid(linestyle="--", alpha=0.7)
    plt.tight_layout()
    plt.show()
```

Figure 4. QQプロットによる正規性の可視化、QQプロット生成コード例の抜粋

相関分析による関係性の評価

Figure 5は複数の変数（化学シフト、積分値、CV）とピーク純度との関係性を、ピアソン相関係数およびスピアマン相関係数で評価した結果です。相関係数（r/ρ）は、それぞれPearsonおよびSpearmanの相関係数を示し、線形・非線形の関係性を包括的に評価できます。いずれの相関係数も絶対値が小さく、p値も高いため、統計的に有意な相関は確認されませんでした。定量結果はこれらの変数に対して安定しており、測定条件による影響が少ないことが示されています。なおCVとの負の相関についても、相関係数が小さく統計的な有意性は確認されていないため、因果関係の特定にはさらなる検証が必要です。

比較項目	相関係数（r/ρ）	P値	傾向
化学シフト vs 純度	0.1051 / 0.0950	0.4870 / 0.5298	弱い正の相関（有意性なし）
積分値 vs 純度	0.0493 / -0.1095	0.7451 / 0.4690	非常に弱い相関（有意性なし）
CV vs 純度	-0.1272 / -0.1139	0.3996 / 0.4511	弱い負の相関（有意性なし）

```python
from scipy.stats import pearsonr, spearmanr

pearson_r, p_val = pearsonr(x, y)
spearman_rho, p_val_s = spearmanr(x, y)
```

Figure 5. 相関分析による関係性の評価、相関分析コード例の抜粋

2. JASON × BeautifulJASON × Pythonによるバッチ処理の使用例

ここでは、Pythonスクリプトを用いたNMRデータの自動解析の例を紹介します。JASON公式サイトからダウンロード可能な「Batch Processing with BeautifulJASON」のスクリプトファイル使用し、複数NMRデータに対してJASONのルール機能[5]を適用した自動解析と、結果のCSVファイル出力について紹介します。

Batch Processing with BeautifulJASONのスクリプトファイルについて

以下2つのスクリプトファイルが提供されています：

1. jason_batch_convert.py：JASONのルール機能を用いてデータを一括変換
2. batch_extract_integrals.py：変換後のデータから積分値を抽出し、CSV形式で出力

いずれも、JASON公式サイトからダウンロード可能です：

https://www.jeoljason.com/resources-external-nmr-processing-scripts/

JEOL 日本電子株式会社
本社・昭島製作所
〒196-8558 東京都昭島市武蔵野3－1－2 TEL：(042) 543－1111(大代表) FAX：(042) 546－3353
www.jeol.co.jp ISO 9001・ISO 14001 認証取得

処理ステップについて

Table 5にJASONによるNMRバッチ処理の3ステップとその内容を示します：

Table 5. NMRバッチ処理のステップについて

ステップ	説明	代表コード
① 入力準備	入力フォルダに複数のNMRデータ（.jdf）を配置します。	—
② .jjh5形式に変換	jason_batch_convert.py を使用して .jdf ファイルを .jjh5 形式に変換し、JASONルールを適用します。	jason_batch_convert.py <input> <output> --formats jjh5 --extensions jdf --rules "ルールファイル" --execute
③ CSVに出力	batch_extract_integrals.py を使用して .jjh5 ファイルから積分値やパラメータを抽出し、CSVに保存します。	batch_extract_integrals.py <output> <csv_path> -p parameters/ACTUAL_START_TIME jason_parameters/SpectrometerFrequencies[0]

変換スクリプト(jason_batch_convert.py)について

Figure 6に、jason_batch_convert.pyのコードの一部を示します。このスクリプトでは、JASONのルール機能を適用して、.jjh5などの形式に変換する処理が行われます。特に、main()関数の引数処理部分では、ユーザーがコマンドラインで指定するオプションの意味が明確に記述されており、実行方法の理解に役立ちます。

```python
def process_file(
    execute: bool, file: str, in_dir: str, out_dir: str,
    jason: bjason.JASON, formats: list[str], rules: str,
    counter: ValueProxy, total_files: int
):
    """1つのファイルを指定された形式に変換する処理を行います。"""
    # 出力ファイル名を形式ごとに生成（重複しないようにユニークな名前に）
    out_fnames = [unique_output_filename(file, format) for format in formats]

    if execute:
        global processed_files  # 実行フラグがTrueの場合のみ変換処理を実行
        # 入力ファイルを読み込み指定ルールに従ってドキュメントを作成
        with jason.create_document(os.path.join(in_dir, file), rules=rules) as doc:
            doc.close()  # 処理が終わったら明示的にクローズ
            # 指定された形式で出力ディレクトリに保存
            jason.save(doc, [os.path.join(out_dir, fname) for fname in out_fnames])

    # 処理済みファイル数をカウントアップ
    counter.value += 1
    print(f"{counter.value}/{total_files}: {file} => {', '.join(out_fnames)}")
```

```python
def main():
    init(autoreset=True)  # coloramaの初期化（コンソール出力の色付け用）

    # コマンドライン引数の定義
    parser = argparse.ArgumentParser(
        description='指定されたディレクトリ内のファイルを拡張子やパターンに基づいて変換します。',
        usage='jason_batch_convert [-h] in_dir out_dir --formats FORMATS [FORMATS...

    # 入力ディレクトリ（変換対象のファイルがある場所）
    parser.add_argument('in_dir', help='変換対象のファイルが含まれるルートディレクトリ')

    # 出力ディレクトリ（変換後のファイルを保存する場所）
    parser.add_argument('out_dir', help='変換後のファイルを保存するディレクトリ')

    # 出力形式（複数指定可能）
    parser.add_argument(
        '--formats', nargs='+', required=True,
        choices=['jjh5', 'jjj', 'jdx', 'jdf', 'pdf', 'png', 'jpg', 'svg'],
        help='出力したいファイル形式を指定（複数指定可能）')

    # 変換ルール（ルールファイルのパスまたはライブラリ名）
    parser.add_argument(
        '--rules', type=str, default="off",
        help='使用する変換ルールのファイルパスまたはライブラリ名')

    # 実行フラグ（指定しない場合はドライラン）
    parser.add_argument(
        '--execute', action='store_true',
        help='変換処理を実行します。指定しない場合はドライラン（変換対象の一覧表示）のみ行います。
        # まずはドライランで確認するのがおすすめです。
```

Figure 6. .jdfファイル変換およびJASONのルール機能の適用に関するコードの抜粋

出力スクリプト(batch_extract_integrals.py)について

Figure 7に、batch_extract_integrals.pyのコードの一部を示します。このスクリプトでは、.jjh5 ファイルから積分値やパラメータを抽出し、CSVに書き出す処理が行われます。

```python
def extract_integrals(jjh5_file_path: str, csv_writer: csv.writer, params: list[str]):
    """
    JJH5ファイルから積分値（integrals）と指定されたパラメータを抽出し、
    csv.writer を使ってCSVファイルに書き出します。
    """
    try:
        # ファイル名から拡張子を除いた名前を取得（出力用の行の先頭に使う）
        base_name = os.path.basename(jjh5_file_path)
        file_name_without_extension, _ = os.path.splitext(base_name)

        # JASON形式のNMRデータファイルを読み込み（読み取りモード）
        with bjason.Document(jjh5_file_path, mode="r") as doc:
            spec = doc.nmr_data[0]  # 最初のスペクトルデータを取得
            row = [file_name_without_extension]  # 出力行の先頭にファイル名を追加

            # 指定されたパラメータを順に処理
            for param in params:
                # 例: "jason_parameters/SpectrometerFrequencies[0]" のような形式を分解
                param_parts = param.split('/')
                param_group = param_parts[0]
                param_name = param_parts[1]
                param_index = None

                # インデックス指定がある場合（例: "ParamName[0]"）を処理
                if '[' in param_name:
                    param_name, param_index = param_name.split('[')
                    param_index = int(param_index[:-1])  # "]" を除いて整数に変換
```

```python
                # パラメータの取得（jason_parameters か raw_data かで分岐）
                if param_group == "jason_parameters":
                    param_value = spec.spec_info.get_param(param_name)
                else:
                    param_value = spec.raw_data.spec_info.get_orig_param
                    (param_group, param_name)

                # インデックス指定がある場合はその要素を取り出す
                if param_index is not None and param_value is not None:
                    if hasattr(param_value, '__getitem__'):
                        param_value = param_value[param_index]

                # 改行などをエスケープしてCSVに書き出せるように整形
                row.append(escape_newlines_for_csv(param_value))

            # スペクトル内のマルチプレット（ピーク）の積分値（Hz単位）を追加
            row.extend(integral.value_hz for integral in spec.multiplets)

            # 最終的な行をCSVに書き出し
            csv_writer.writerow(row)

    except Exception as e:
        # エラーが発生した場合はファイル名とともに表示
        print(f"Error processing file '{jjh5_file_path}': {e}")
```

Figure 7. 積分値抽出およびcsv出力に関するコードの抜粋

JEOL 日本電子株式会社
本社・昭島製作所
〒196-8558 東京都昭島市武蔵野3－1－2 TEL：(042) 543－1111(大代表) FAX：(042) 546－3353
www.jeol.co.jp ISO 9001・ISO 14001 認証取得

```
@echo off
REM バッチファイルの開始。コマンドの実行内容を表示しないように設定

REM ▼ CSVファイルからディレクトリ情報を読み込む
REM dir_setting.csv の2列目～4列目 (input, output, csv) を取得
set "csv_file=C:¥Path¥To¥Your¥Project¥tool¥dir_setting.csv"

for /f "skip=1 delims=, tokens=2-4" %%i in (%csv_file%) do (
    set "input=%%i"
    set "output=%%j"
    set "csv=%%k"
)

REM 読み込んだディレクトリ情報を表示 (確認用)
echo 入力ディレクトリ: %input%
echo 出力ディレクトリ: %output%
echo CSV出力先: %csv%

REM ▼ Python仮想環境のアクティベート (必要に応じて)
call C:¥Path¥To¥PythonEnv¥Scripts¥activate.bat

echo --- 変換処理開始 ---

REM 作業ディレクトリに移動 (Pythonスクリプトがある場所)
cd C:¥Path¥To¥Your¥Project
```

```
REM ▼ JASONデータの一括変換処理
REM - input: 入力ディレクトリ
REM - output: 出力ディレクトリ
REM - formats: 出力形式 (例: jih5)
REM - extensions: 対象ファイルの拡張子 (例: jdf)
REM - rules: 変換ルールファイルのパス
REM - execute: 実行フラグ (指定しないとドライラン)
python jason_batch_convert.py %input% %output% --formats jih5 --
extensions jdf --rules "C:¥Path¥To¥Rules¥test.jjr" --execute

echo --- ・変換処理終了 ---

echo --- 積分値のCSV出力開始 ---

REM ▼ 積分値の抽出とCSV出力処理
REM - output: 変換後のファイルがあるディレクトリ
REM - csv: 出力先CSVファイルのパス
python batch_extract_integrals.py %output% %csv%

echo --- 処理完了 ---

pause
REM ウィンドウを閉じないように一時停止
```

バッチファイルによる自動化について

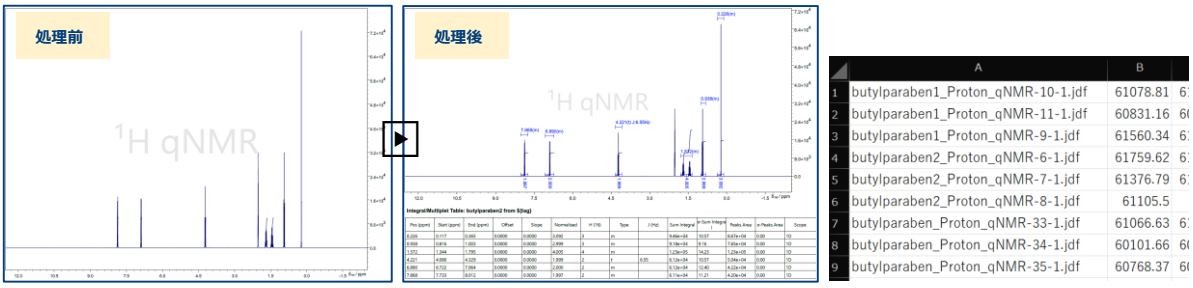
Figure 8では、簡単なバッチファイルの例を紹介します。このバッチファイルは、上記の公式スクリプトを使ったNMRデータのバッチ処理を自動化するためのものです。ダブルクリックで処理を開始できるため、初心者でも実行しやすく、他のデータへの応用が可能です。また、Pythonスクリプトで統合することでより柔軟な処理も可能になります。

Figure 8. バッチファイルによる自動化例とコードの抜粋

出力結果について

Figure 9に出力例を示します。ここでは、9個のNMRデータに対してJASONのルール機能を適用し、自動解析を行った結果をCSVファイルにまとめて出力しています。出力されたデータおよびCSVファイルの内容はFigure 10に示しています。

Figure 9. Batch Processing with BeautifulJASON 自動解析例



	A	B
1	butylparaben1_Proton_qNMR-10-1.jdf	61078.81
2	butylparaben1_Proton_qNMR-11-1.jdf	60831.16
3	butylparaben1_Proton_qNMR-9-1.jdf	61560.34
4	butylparaben2_Proton_qNMR-6-1.jdf	61759.62
5	butylparaben2_Proton_qNMR-7-1.jdf	61376.79
6	butylparaben2_Proton_qNMR-8-1.jdf	61105.5
7	butylparaben_Proton_qNMR-33-1.jdf	61066.63
8	butylparaben_Proton_qNMR-34-1.jdf	60101.66
9	butylparaben_Proton_qNMR-35-1.jdf	60768.37

Figure 10. Batch Processing with BeautifulJASON 自動解析 出力例

補足と注意点

- スクリプトはPython 3環境で実行してください。
- JASONのルール機能ファイル(.jjr)はJASONで作成しておく必要があります。
- パラメータ名はJASON GUIで確認した正確な名称を使用してください。

まとめ

- 本資料では、JASONとBeautifulJASONを活用したNMRスペクトル解析の自動化手法について紹介しました。
- BeautifulJASONにより、PythonからJASONデータを直接操作・解析できる環境が整い、柔軟な処理や統計解析が可能になります。
- 特に、公式スクリプト(jason_batch_convert.py、batch_extract_integrals.py)を活用することで、複数データの一括処理やCSV出力が容易になります。

参照

JASON公式サイト: https://www.jeoljason.com/
BeautifulJASONドキュメント: https://www.jeoljason.com/beautifuljason/docs/
Batch Processing スクリプトファイル: https://www.jeoljason.com/resources-external-nmr-processing-scripts/
Python公式ドキュメント: https://docs.python.org/ja/3/

[1] JEOL Analytical Software Network
[2] Pythonは、Python Software Foundationの商標または登録商標です。
[3] .jjh5はJASONにおける.hdf5の形式です。
[4] multiplet(多重線)とは、NMRスペクトルにおいてスピン結合により分裂したピーク群を指します。
[5] JASONのルール機能とは、スペクトル処理の手順やレポートレイアウトを定義した設定ファイル(.jjr)を用いて、複数データに対して一括処理を行う仕組みです。

JEOL 日本電子株式会社
本社・昭島製作所
〒196-8558 東京都昭島市武蔵野3－1－2 TEL：(042) 543－1111(大代表) FAX：(042) 546－3353
www.jeol.co.jp ISO 9001・ISO 14001 認証取得