

Forschungsdatenmanagement

Notwendige, aber nicht hinreichende Voraussetzung
für den wissenschaftlichen Erkenntnisgewinn

19. Lückenloses Protokoll der Datenverarbeitung

Dr. habil. Till Biskup

Physikalische Chemie

Universität Rostock

10.07.2025





- ❏ Nachvollziehbarkeit ist konstituierend für Wissenschaft.
Voraussetzung: lückenloses Protokoll der Datenverarbeitung.
- ❏ Ein System zur lückenlosen Protokollierung ist möglich, aber erheblich komplex und nicht trivial umzusetzen.
- ❏ Den meisten Forschenden fehlen Werkzeuge und Erfahrung, um ein solches System selbst entwickeln zu können.
- ❏ Ein funktionierendes System muss den gesamten Weg von der Datenerhebung bis zur Präsentation abdecken.
- ❏ Nur konsequente Nutzung garantiert die Lückenlosigkeit.
Voraussetzung: Flexibilität und offensichtliche Vorteile.

Problemstellung, Herausforderungen und Anforderungen

Konzept und Umsetzung

Anwendung auf die eigene Praxis

Vorteile gegenüber Alternativen

Problemstellung

- ▶ Lückenloses Protokoll der Datenverarbeitung von den Rohdaten zur fertigen Datenpräsentation (Abbildung, Tabelle).
- ▶ Vollständige Nachvollziehbarkeit, ggf. Reproduzierbarkeit

Antwort auf fünf Fragen

- ❓ Wer hat
- ❓ was
- ❓ mit wem
- ❓ wann
- ❓ und wie gemacht?

Herausforderungen

- ▶ Die wenigsten Forschenden haben die (intellektuellen) Werkzeuge und die notwendige Erfahrung, um ein solches System selbst entwickeln zu können.
- ▶ Den meisten Wissenschaftlern wurde nie formal beigebracht, wie man programmiert, geschweige denn Software entwickelt.

Anforderungen

- ▶ Weitgehend automatisiertes „*scientific workflow system*“, das große Mengen von Daten verarbeiten kann
- ▶ Hinreichend einfach und elegant zu nutzen: sollte für eine(n) Bachelor-Studierende(n) funktionieren
- ▶ Passt sich an die (variablen) Anforderungen der Wissenschaft an: robust, modular, flexibel und (einfach) erweiterbar

Kontext

- ▶ wissenschaftlicher Kernaspekt
 - Nachvollziehbarkeit, ggf. Reproduzierbarkeit
- ▶ Station im Forschungsdatenlebenszyklus
 - Datenauswertung
- ▶ Bedeutung
 - Nachvollziehbarkeit der Datenauswertung
 - Ausgangspunkt für die Qualitätskontrolle und Qualitätssicherung

Beitrag

- ▶ intuitive Durchführung komplexer Datenauswertungen
 - für die Anwendung keine Notwendigkeit, Programmieren zu lernen
- ▶ automatisches lückenloses Protokoll aller Parameter

Problemstellung, Herausforderungen und Anforderungen

Konzept und Umsetzung

Anwendung auf die eigene Praxis

Vorteile gegenüber Alternativen

ASpecD: A modular framework for the analysis of spectroscopic data focussing on reproducibility and good scientific practice

*Jara Popp, Till Biskup**



Chemistry—Methods **2**:e202100097, 2022

Listing 1: Rezept

```
1 format:
  type: ASpecD recipe
3 version: '0.3'

5 datasets:
  - /path/to/first/dataset
7  - /path/to/second/dataset

9 tasks:
  - kind: processing
11   type: BaselineCorrection
   properties:
13     parameters:
       kind: polynomial
15     order: 0
  - kind: singleplot
17   type: SinglePlotter1D
   properties:
19     filename:
       - first-dataset.pdf
21     - second-dataset.pdf
```

Listing 2: Rezept-Historie

```
1 system_info:
  python:
3    version: "3.11.2 ..."
  packages:
5    aspecd: 0.12.0
  # ...
7  - kind: processing
   type: BaselineCorrection
9   properties:
     parameters:
11      kind: polynomial
       order: 0
13      coefficients:
         - -0.04609818536259180
15      fit_area:
         - 10
17      - 10
       axis: 0
19  apply_to:
     - /path/to/first/dataset
21  # ...
```

☞ Wir haben meist eine klare Idee, was mit den Daten passieren soll.

- ▶ Rezeptgetriebene Datenauswertung
 - Keine Programmierkenntnisse notwendig
- ▶ Datensatz als Einheit von Daten und Metadaten
 - Abstraktion von den Hersteller-Dateiformaten
- ▶ Nachvollziehbarkeit
 - Die Historie enthält explizite *und* implizite Parameter
- ▶ Reproduzierbarkeit
 - Die Historie ist ein vollständig funktionierendes Rezept
- ▶ Modular und erweiterbar
 - Fokussiert auf Anwendung, nicht auf die Infrastruktur
- ▶ Unterstützung für unterschiedliche (spektroskopische) Methoden
 - Python-Pakete für einige Methoden sind verfügbar

Verarbeitungsschritte

- ▶ Normalisation
- ▶ Integration
- ▶ Differentiation
- ▶ ScalarAlgebra
- ▶ Projection
- ▶ SliceExtraction
- ▶ RangeExtraction
- ▶ BaselineCorrection
- ▶ Averaging
- ▶ DatasetAlgebra
- ▶ Interpolation
- ▶ Filtering
- ▶ CommonRangeExtraction
- ▶ Noise
- ▶ ...

Berichte

Tabellen

Analyseschritte

- ▶ BasicCharacteristics
- ▶ PeakFinding
- ▶ PowerDensitySpectrum
- ▶ PolynomialFit
- ▶ LinearRegressionWithFixedIntercept
- ▶ ...

Plotter

- ▶ SinglePlotter1D
- ▶ SinglePlotter2D
- ▶ SinglePlotter2DStacked
- ▶ MultiPlotter1D
- ▶ MultiPlotter1DStacked
- ▶ CompositePlotter
- ▶ ...

Modelle

Importer/Exporter

Ein ganzes Ökosystem

Beispiele für konkrete Pakete, die auf dem Framework aufbauen



ASpecD



<https://docs.aspecd.de/>

J. Popp, T. Biskup. *Chem. Meth.* 2:e202100097, 2022

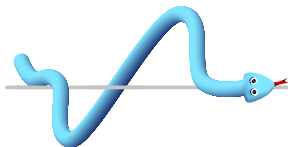
cwepr



<https://docs.cwepr.de/>

M. Schröder, T. Biskup. *J. Magn. Reson.* 335:107140, 2022

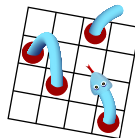
trEPR



<https://docs.trepr.de/>

J. Popp, M. Schröder, T. Biskup. *repr* (2022).
doi:10.5281/zenodo.4897112

FitPy

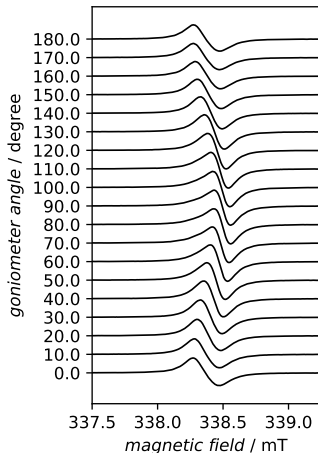
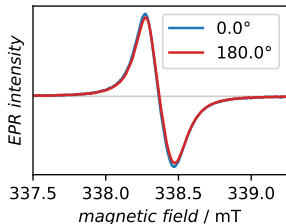
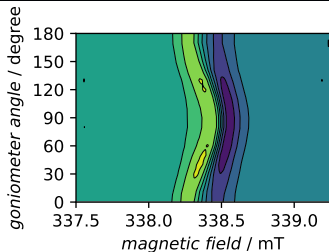


<https://docs.fitpy.de/>

T. Biskup. *fitpy* (2022).
doi:10.5281/zenodo.5920380

Listing 3: „Rezept“ zur Datenverarbeitung

```
1 format:
  type: AS
3 version:
5 settings:
  default
7
9 directories
11 datasets:
  - magnet
13
15 tasks:
  - kind: pr
    type: Ba
17 - kind: si
    type: Go
19 properties
  proper
21 axes
  xl
23 filename
```



Problemstellung, Herausforderungen und Anforderungen

Konzept und Umsetzung

Anwendung auf die eigene Praxis

Vorteile gegenüber Alternativen

unterschiedliche Szenarien

- ▶ direkte Nutzung vorhandener Pakete
 - Methoden: cwepr, trepr, NMR, UVvis (künftig Radiometrie)
 - werden aktiv weiterentwickelt
- ▶ Erweiterung vorhandener Pakete
 - setzt Python-Programmierkenntnisse voraus
 - ausführliche Dokumentation vorhanden
- ▶ Entwicklung neuer Pakete für andere Methoden
 - basierend auf dem ASpecD-Framework
 - ausführliche Dokumentation vorhanden
 - Hilfestellung durch Codegenerierung: pymetacode
- 👉 Fokus auf der eigentlichen wissenschaftlichen Aufgabe, nicht auf der notwendigen Infrastruktur

- ▶ inhärente Komplexität der Fragestellung
 - notwendigerweise relativ komplexes System
 - initial eine gewisse Lernkurve
 - setzt Bereitschaft voraus, sich auf ein neues System einzulassen
- ▶ Rezepte helfen, sich über die Auswertung klar zu werden.
 - Meist haben wir eine Idee, wie wir die Daten auswerten wollen...
 - Rezepte abstrahieren von der Programmierung
- ▶ „Abkürzungen“ korrumpieren das System und letztlich die Nachvollziehbarkeit (und Wissenschaftlichkeit)
 - lückenloses automatisches Protokoll nur in einem Gesamtsystem
 - umfassend: von den Rohdaten bis zur Abbildung/Tabelle
 - fehlende Bausteine sollten implementiert werden
- ▶ umfassende (und stetig erweiterte) Dokumentation
 - hilft sowohl bei der Anwendung als auch der Weiterentwicklung

Anwendung auf die eigene Praxis

Ausführliche Dokumentation, online verfügbar



Changelog

Roadmap

ASpecD dataset format (adf)

Dataset structure

API documentation

aspecd.analysis module

aspecd.annotation module

aspecd.dataset module

aspecd.exceptions module

aspecd.history module

aspecd.infofile module

aspecd.io module

aspecd.metadata module

aspecd.model module

aspecd.plotting module

aspecd.processing module

Concrete processing steps

Writing own processing steps

Module documentation

aspecd.report module

aspecd.system module

aspecd.table module

aspecd.tasks module

class aspecd.processing.BaselineCorrection

Examples

For convenience, a series of examples in recipe style (for details of the recipe-driven data analysis, see `aspecd.tasks`) is given below for how to make use of this class. The examples focus each on a single aspect.

In the simplest case, just invoke the baseline correction with default values:

```
- kind: processing
  type: BaselineCorrection
```

In this case, a zeroth-order polynomial baseline will be subtracted from your dataset using ten percent to the left and right, and in case of a 2D dataset, the baseline correction will be performed along the first axis (index zero) for all indices of the second axis (index 1).

Of course, often you want to control a little bit more how the baseline will be corrected. This can be done by explicitly setting some parameters.

Suppose you want to perform a baseline correction with a polynomial of first order:

```
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
      order: 1
```

If you want to change the (percental) area used for fitting the baseline, and even specify different ranges left and right:

```
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
```

<https://docs.aspecd.de/>

Anwendung auf die eigene Praxis

Vorschau: Entwicklung einer browserbasierten Oberfläche



ASpecD WebUI: Cook recipe

← → ↻ 🏠

🔍 127.0.0.1:5050

⋮ 📄 ⭐

⬇️ 📄 📄 ⓘ ? S 🌐 ☰

Cook recipe

Default package: cwepr Reset

Datasets

11 dataset(s) loaded Upload

- Sa754-01 ([Add](#) [Remove](#) [Delete](#))
- Sa765-01 ([Add](#) [Remove](#) [Delete](#))
- Sa766-01 ([Add](#) [Remove](#) [Delete](#))
- Sa738-01 ([Add](#) [Remove](#) [Delete](#))

Recipe

```
format:
  type: ASpecD recipe
  version: '0.2'
settings:
  default_package: cwepr
  autosave_plots: false
  write_history: true
directories:
  output: results
  datasets_source: datasets
datasets:
- Sa732-01
- Sa734-01
- Sa736-01
- Sa738-01
tasks:
- kind: processing
  type: BaselineCorrection
- kind: processing
  type: FrequencyCorrection
format: yaml
```

Cook&Serve

Results

3 files [Download](#)

- [comparison-unfiltered-filtered.pdf](#)
- [recipe-20211118T154733.yaml](#)
- [recipe.yaml](#)

Tasks

plotting CompositePlotter Add

```
- kind: compositeplot
  type: CompositePlotter
  properties:
    plotter:
      - unfiltered
      - filtered
    parameters:
      show_legend: false
      show_zero_lines: true
      tight_layout: false
    properties:
      figure:
        size:
          - 6.0
          - 8.0
        dpi: 100.0
      filename: comparison-unfiltered-filtered.pdf
      grid_dimensions:
        - 1
        - 2
      subplot_locations:
```

Messages

```
INFO - Import dataset "datasets/Sa732-01" as "Sa732-01"
INFO - Import dataset "datasets/Sa734-01" as "Sa734-01"
INFO - Import dataset "datasets/Sa736-01" as "Sa736-01"
INFO - Import dataset "datasets/Sa738-01" as "Sa738-01"
INFO - Perform "BaselineCorrection" on dataset "Sa732-01"
INFO - Perform "BaselineCorrection" on dataset "Sa734-01"
```

Problemstellung, Herausforderungen und Anforderungen

Konzept und Umsetzung

Anwendung auf die eigene Praxis

Vorteile gegenüber Alternativen

mögliche Alternativen

- ▶ ein Skript pro Datensatz
- ▶ grafische Werkzeuge wie Origin etc.

Vorteile des ASpecD-Frameworks und abgeleiteter Pakete

- ▶ keine Programmierkenntnisse für die Anwendung notwendig
 - funktioniert auch in einer Bachelorarbeit
 - ▶ Nachvollziehbarkeit gewährleistet
 - vollständige Dokumentation *aller* Parameter (implizit & explizit)
 - Liste verwendeter Bibliotheken und ihrer Versionen
 - Rezepte grundsätzlich auch reproduzierbar
- ☞ Dem Dozenten ist kein in der Spektroskopie anwendbares alternatives System bekannt, das auch nur ansatzweise die Nachvollziehbarkeit über ein lückenloses Protokoll gewährleistet.

Folgt bewährten Verfahren der Softwareentwicklung, z.B.:

- ▶ Testgetriebene Entwicklung
 - hohe Testabdeckung, bessere Verlässlichkeit
 - ▶ Sauberer Code
 - lesbar, ausdrucksstark, selbstdokumentierend
 - ▶ ausführlich dokumentiert
 - <https://docs.aspecd.de/>
 - ▶ Versionsverwaltungssystem
 - <https://github.com/tillbiskup/aspecd/>
 - ▶ quelloffen (open source)
 - BSD-Lizenz: jeder darf es verwenden und verändern
- 👉 Jede(r) Python-Programmierer(in) könnte das Projekt übernehmen.



- ❏ Nachvollziehbarkeit ist konstituierend für Wissenschaft.
Voraussetzung: lückenloses Protokoll der Datenverarbeitung.
- ❏ Ein System zur lückenlosen Protokollierung ist möglich, aber erheblich komplex und nicht trivial umzusetzen.
- ❏ Den meisten Forschenden fehlen Werkzeuge und Erfahrung, um ein solches System selbst entwickeln zu können.
- ❏ Ein funktionierendes System muss den gesamten Weg von der Datenerhebung bis zur Präsentation abdecken.
- ❏ Nur konsequente Nutzung garantiert die Lückenlosigkeit.
Voraussetzung: Flexibilität und offensichtliche Vorteile.