Wissenschaftliche Softwareentwicklung

31. Finale furioso: Zusammenfassung und Ausblick

Till Biskup
Physikalische Chemie
Universität Rostock
26.01.2024

Zentrale Aspekte

- Programmierung ist eine Kernkompetenz für Naturwissenschaftler. Trotzdem hat sie einen viel zu geringen Stellenwert.
- Wissenschaft beruht auf Nachvollziehbarkeit und Reproduzierbarkeit. Das ist in der Praxis oft nicht gewährleistet.
- Die primäre Aufgabe von Code ist Kommunikation. Voraussetzung ist Ausdrucksstärke und Lesbarkeit.
- Kernaspekt aller Softwarearchitektur ist Modularität. Das führt zu Flexibilität und Wiederverwendbarkeit.
- Wissenschaftliche Datenauswertung erfordert ein durchdachtes Gesamtkonzept. Einzelaspekte sind (relativ) einfach umsetzbar.

Übersicht

Zusammenfassung: Themen der Vorlesung im Überblick

Fünf Thesen

Ausblick: Gesamtsystem zur wissenschaftlichen Datenverarbeitung

Feedback

Zusammenfassung

Die Themen der Vorlesung im Überblick

1. Motivation
   - Programmierung ist essentieller Bestandteil von Forschung.
2. Infrastruktur
   - notwendige Voraussetzungen für die Softwareentwicklung
3. Sauberer Code
   - Kommunikation erfordert Ausdrucksstärke und Lesbarkeit.
4. Softwarearchitektur
   - Modularität führt zu Flexibilität und Wiederverwendbarkeit.
5. Datenverarbeitung und -Analyse in den Naturwissenschaften
   - Gesamtkonzept von der Datenaufnahme bis zur Publikation

☛ Ziel: Software, die wissenschaftlichen Kriterien entspricht

Übersicht

Zusammenfassung: Themen der Vorlesung im Überblick

Fünf Thesen

Ausblick: Gesamtsystem zur wissenschaftlichen Datenverarbeitung

Feedback

Programmierung als Kernkompetenz

Das Problem: mangelnder Stellenwert

These

Programmierung ist eine Kernkompetenz für moderne Naturwissenschaftler. Ihr Stellenwert ist trotzdem viel zu gering.

- Datenauswertung ist (fast) immer rechnergestützt.
  - Programmierkenntnisse werden vorausgesetzt, sind aber selten in ausreichendem Maß vorhanden.
- Die Komplexität der Software entspricht der Fragestellung.
  - Beherrschung erfordert Kenntnis entsprechender Konzepte
- Softwareentwicklung wird nicht ausreichend gewürdigt.
  - Entsprechend gering ist die Motivation, Zeit zu investieren.

These

Wissenschaft beruht auf Nachvollziehbarkeit und Reproduzierbarkeit.
Das ist in der Praxis oft nicht gewährleistet.

- Voraussetzungen: Dokumentation und Archivierung
  - Zeitskala: tendenziell Jahrzehnte
  - Stichworte: Unversehrtheit, Versionierung
- Die Realität sieht in vielen Fällen anders aus.
  - Nachvollziehbarkeit der Auswertungen nicht gegeben
  - Reproduzierbarkeit von publizierten Ergebnissen unmöglich

Kommunikation: Hauptaufgabe von Code

Voraussetzungen: Ausdrucksstärke und Lesbarkeit

These

Die primäre Aufgabe von Code ist Kommunikation.
Voraussetzung ist Ausdrucksstärke und Lesbarkeit.

- Code wird viel häufiger gelesen als geschrieben.
  - „*Code for people, not computers.*“
  - (Wirklich) guter Code ist offensichtlich.
- Ausdrucksstärke erfordert Abstraktionsvermögen.
  - Konzepte durchdenken, verstehen und treffend benennen
  - Gute Namen sind das Ergebnis eines Prozesses.

These

Kernaspekt aller Softwarearchitektur ist Modularität.
Das führt zu Flexibilität und Wiederverwendbarkeit.

- Ansprüche an Software ändern sich ständig.
  - Das Verständnis der Problemstellung wächst.
  - Jede Fragestellung ist neu und leicht unterschiedlich.
- Gute Software ist wie ein Legokasten.
  - kleine, durchdachte, zueinander kompatible Bausteine
  - nahezu endlose Möglichkeiten, Neues zu schaffen

These

Wissenschaftliche Datenauswertung erfordert ein durchdachtes Gesamtkonzept. Einzelaspekte sind (relativ) einfach umsetzbar.

- Nachvollziehbarkeit erfordert lückenlose Dokumentation.
  - von der Datenaufnahme bis zur Veröffentlichung
  - weitestgehende Automatisierung einzelner Schritte
- plattform- und medienunabhängig und modular
  - nutzerfreundlich: Fokus auf einfacher Bedienbarkeit
  - Vorteile der Nutzung müssen offensichtlich sein.

Übersicht

Zusammenfassung: Themen der Vorlesung im Überblick

Fünf Thesen

Ausblick: Gesamtsystem zur wissenschaftlichen Datenverarbeitung

Feedback

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

Herausforderungen

- Vollständig nachvollziehbare Datenverarbeitung und -Analyse inkl. eines lückenlosen Protokolls jedes Schrittes
- Den meisten Wissenschaftlern wurde nie formal beigebracht, wie man programmiert oder Software entwickelt.

Anforderungen

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

Rezeptgetriebene Datenauswertung

Der Vorteil: Programmierkenntnisse sind nicht weiter notwendig.

Listing 1: Rezept

```
format:
  type: ASpecD recipe
  version: '0.2'

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

tasks:
  - kind: processing
    type: BaselineCorrection
    properties:
      parameters:
        kind: polynomial
        order: 0
  - kind: singleplot
    type: SinglePlotter1D
    properties:
      filename:
        - first-dataset.pdf
        - second-dataset.pdf
```

Listing 2: Rezept-Historie

```
system_info:
  python:
    version: "3.7.3 ..."
  packages:
    aspecd: 0.6.4
# ...
- kind: processing
  type: BaselineCorrection
  properties:
    parameters:
      kind: polynomial
      order: 0
      coefficients:
      - -0.04609818536259180
      fit_area:
      - 10
      - 10
      axis: 0
  apply_to:
  - /path/to/first/dataset
  # ...
```

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

Das ASpecD-Framework

Entscheidende Eigenschaften

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

Bereitgestellte Funktionalität

Viele Aspekte sind unabhängig von der Methode und bereits implementiert

Verarbeitungsschritte

- Normalisation
- Integration
- Differentiation
- ScalarAlgebra
- Projection
- SliceExtraction
- SliceRemoval
- RangeExtraction
- BaselineCorrection
- Averaging
- ScalarAxisAlgebra
- DatasetAlgebra
- Interpolation
- Filtering
- CommonRangeExtraction
- Noise
- ChangeAxesValues
- RelativeAxis

Analyseschritte

- BasicCharacteristics
- BasicStatistics
- BlindSNREstimation
- PeakFinding
- PowerDensitySpectrum
- PolynomialFit
- LinearRegressionWithFixedIntercept

Plotter

- SinglePlotter
- SinglePlotter1D
- SinglePlotter2D
- SinglePlotter2DStacked
- MultiPlotter
- MultiPlotter1D
- MultiPlotter1DStacked
- CompositePlotter
- SingleCompositePlotter

Ein ganzes Ökosystem

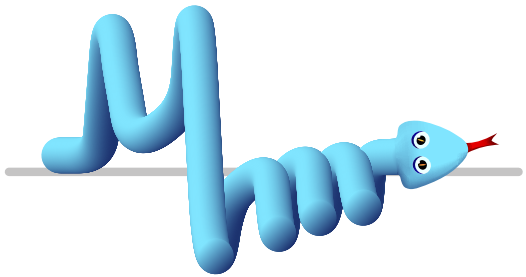
Beispiele für konkrete Pakete, die auf dem Framework aufbauen

ASpecD

`https://docs.aspecd.de/`

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

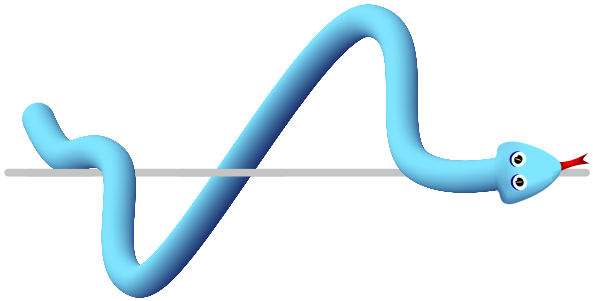
cwepr



`https://docs.cwepr.de/`

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

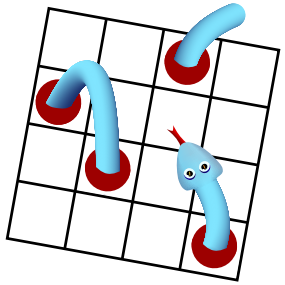
trEPR



`https://docs.trepr.de/`

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

FitPy



`https://docs.fitpy.de/`

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

Ein reales Beispiel

Komplexe Datenauswertung mit einem abgeleiteten Paket

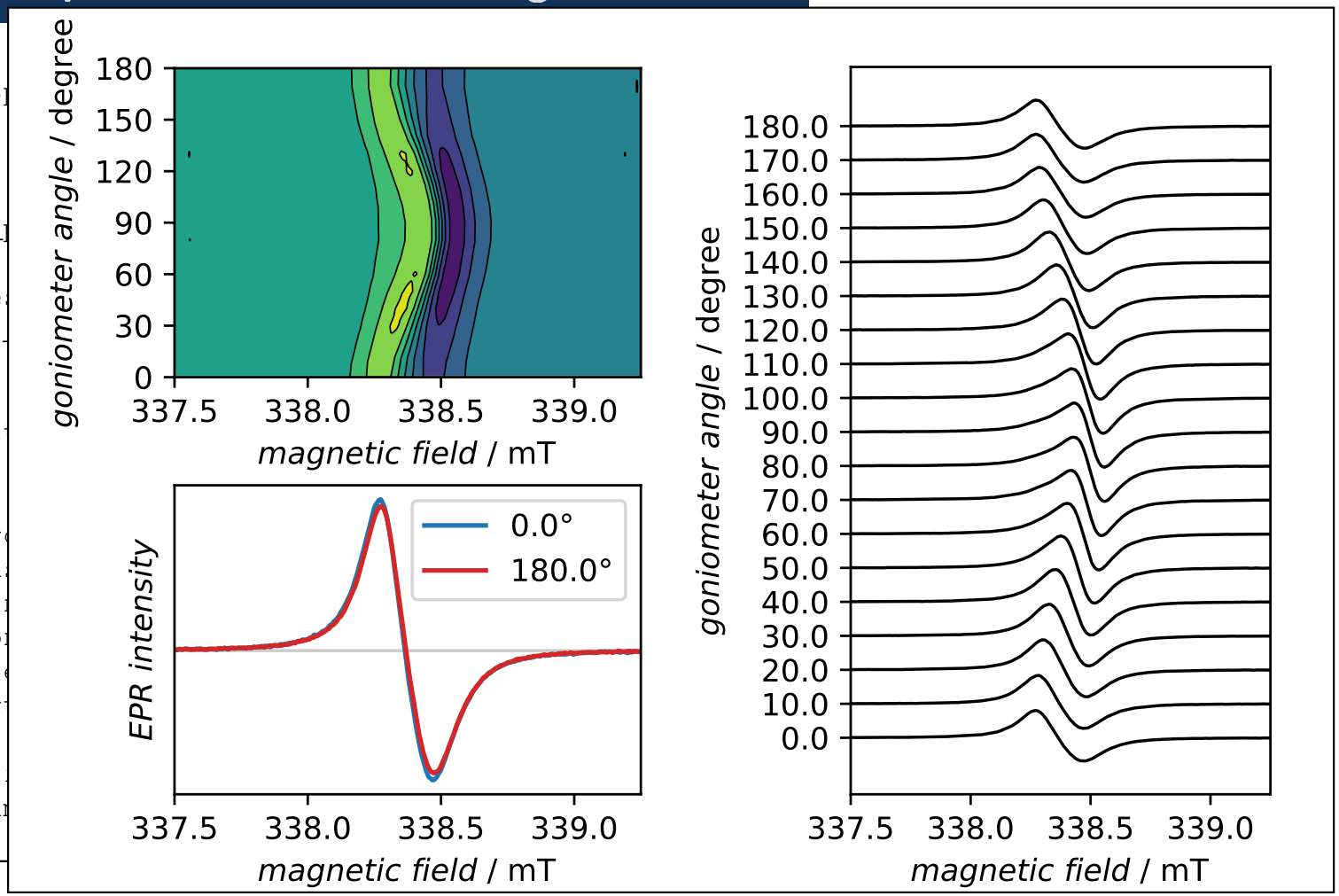
Listing 3: „Rezept“ zur Datenverarbeitung

```
1  format:
     type: AS
3    version:

5  settings:
     default_

7
   directorie
9    datasets

11 datasets:
     - magnet
13
   tasks:
15 - kind: pr
     type: Ba
17 - kind: si
     type: Go
19   propertie
       proper
21       axes
           xl
23     filena
```



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

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

☛ Jeder Python-Programmierer könnte das Projekt übernehmen.

Übersicht

Zusammenfassung: Themen der Vorlesung im Überblick

Fünf Thesen

Ausblick: Gesamtsystem zur wissenschaftlichen Datenverarbeitung

Feedback

Feedback

Eine ehrliche Aufforderung zu (konstruktiver) Kritik

Ein paar Fragen als Inspiration:

- Waren die Themen verständlich?
- Hat die Thematik für Sie eine Relevanz?
- Entsprach die Vorlesung Ihren Vorstellungen?
- Fühlen Sie sich den Ansprüchen an die Programmierung wissenschaftlicher Datenauswertung (mehr) gewachsen?
- ...

Feedback

Eine ehrliche Aufforderung zu (konstruktiver) Kritik

Und noch ein paar konkrete Fragen:

- Wären (fakultative) Übungszettel hilfreich?
- Konnten die Glossare mit den verwendeten (Fach-)Begriffen beim Verständnis helfen?
- Sind die Verständnisfragen zu jedem Kapitel hilfreich für die eigene Nachbereitung der Themen?
- Sollte es Grundlagenkurse zum Programmieren geben? Würden derlei Angebote auch wahrgenommen werden?
- Sollte die Vorlesung ausgedehnt werden, u.a. um im letzten Teil Bausteine eines Systems zur wissenschaftlichen Datenverarbeitung zu präsentieren?

Vorlesung:

Forschungsdatenmanagement

Notwendige, aber nicht hinreichende Voraussetzung
für den wissenschaftlichen Erkenntnisgewinn

https://www.till-biskup.de/de/lehre/forschungsdatenmanagement/