

# Wege aus der in-silico- Reproduzierbarkeitskrise

Johannes Köster

2016



Centrum Wiskunde & Informatica



HARVARD  
MEDICAL SCHOOL

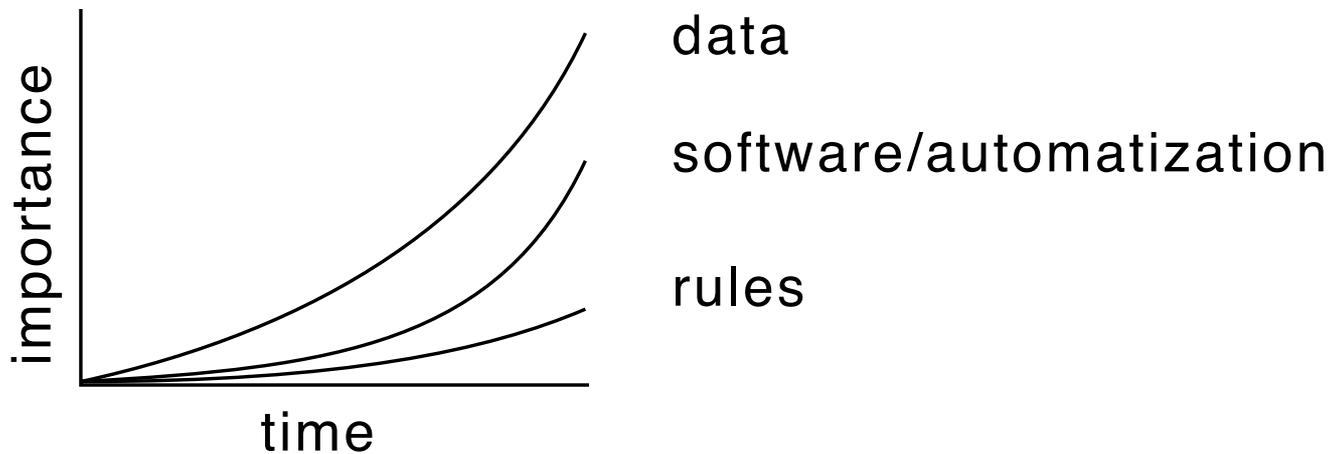
# Perspektive

- Promovierter Informatiker
- Bioinformatik seid 2008

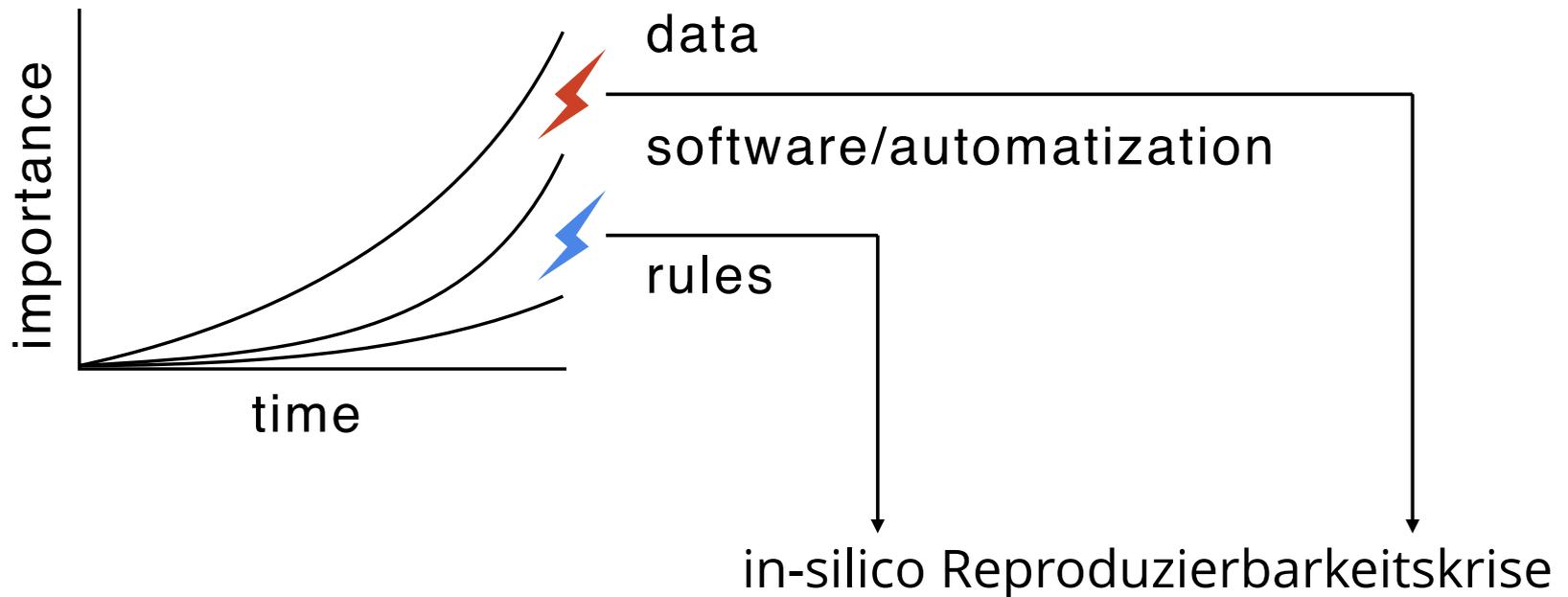
## **Projekte:**

- Algorithmische Bioinformatik
- Systembiologie (MPI Dortmund)
- Ökologie (Universität Duisburg-Essen)
- Medizin/Krebsforschung (Harvard Medical School, Dana-Farber Cancer Institute, Universitätsklinikum Essen)

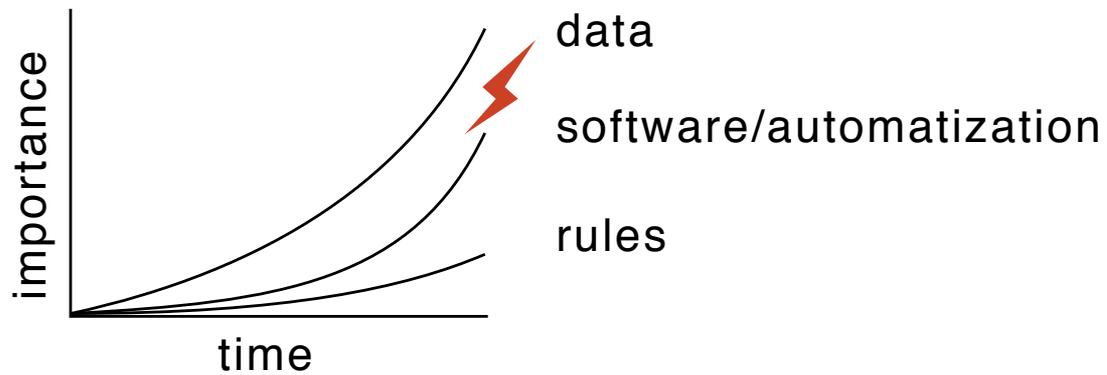
# Software und Daten(analyse) in der Wissenschaft



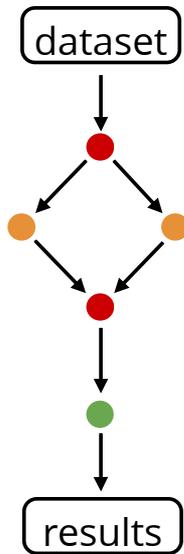
# Software und Daten(analyse) in der Wissenschaft



# Automatisierungskrise



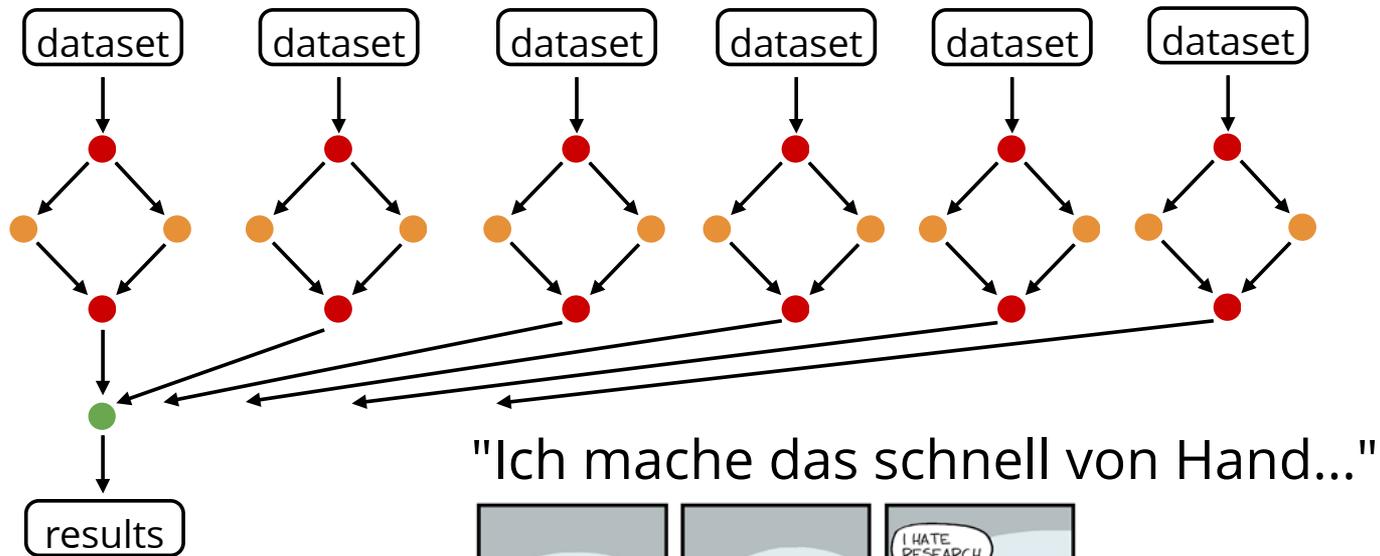
# Datenanalyse



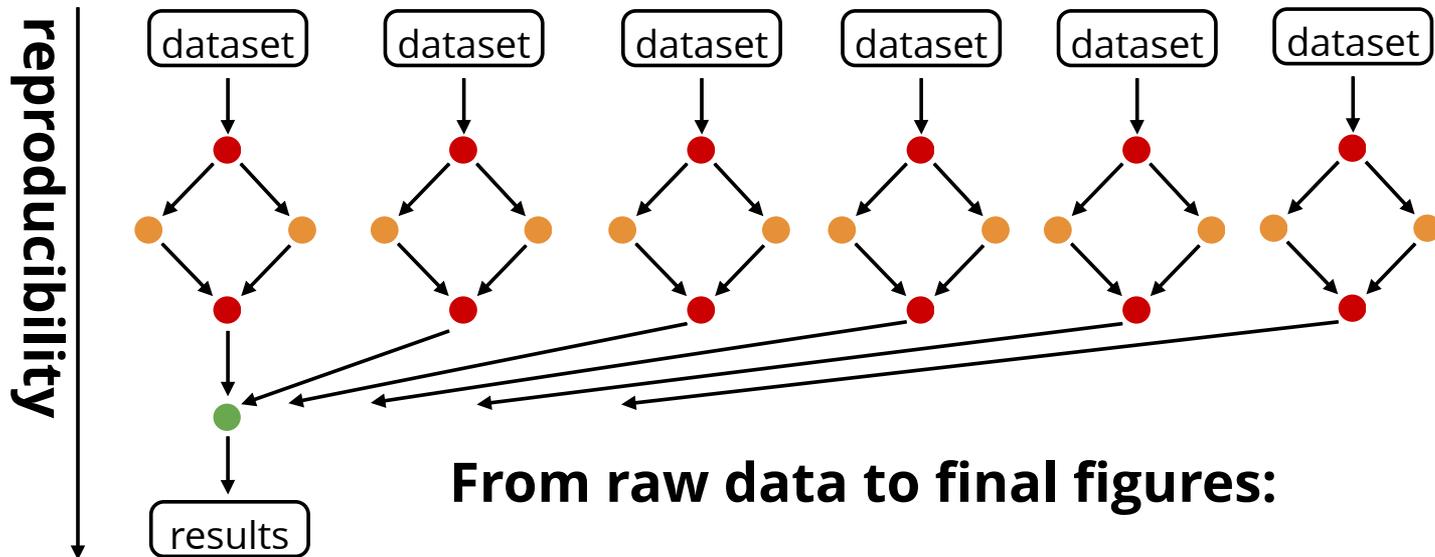
"Ich mache das schnell von Hand..."



# Datenanalyse



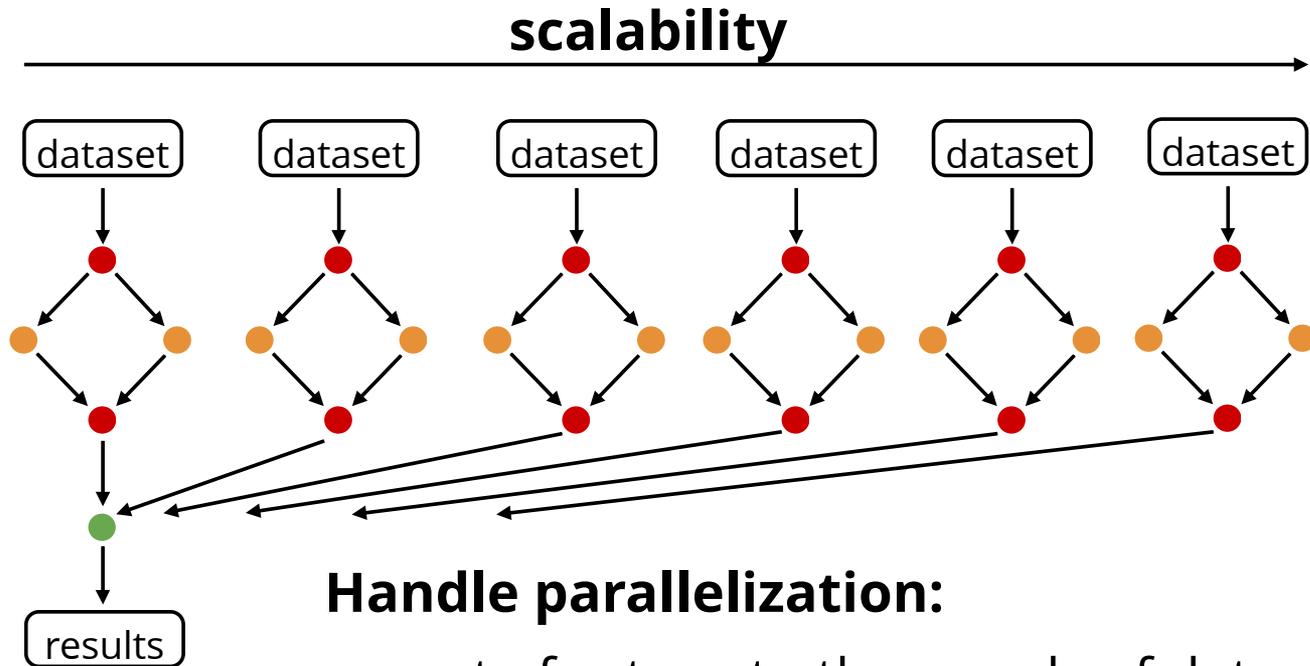
# Datenanalyse



**From raw data to final figures:**

- **document** parameters, tools, versions
- **execute** without manual intervention

# Datenanalyse



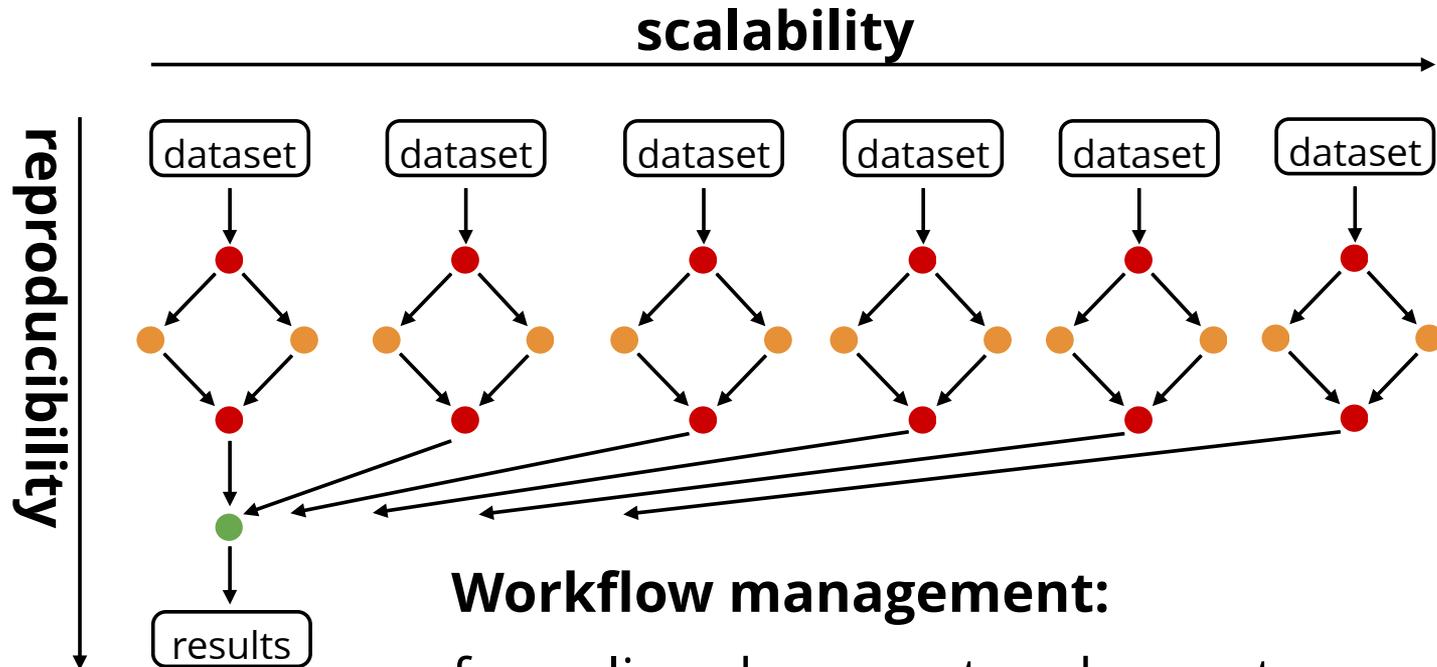
## Handle parallelization:

execute for tens to thousands of datasets

## Avoid redundancy:

- when adding datasets
- when resuming from failures

# Datenanalyse



## Workflow management:

formalize, document and execute data analyses

# Snakemake

**BIOINFORMATICS APPLICATION NOTE**

Vol. 28 no. 19 2012, pages 2520–2522  
doi:10.1093/bioinformatics/bts480

---

*Genome analysis*

Advance Access publication August 20, 2012

## **Snakemake—a scalable bioinformatics workflow engine**

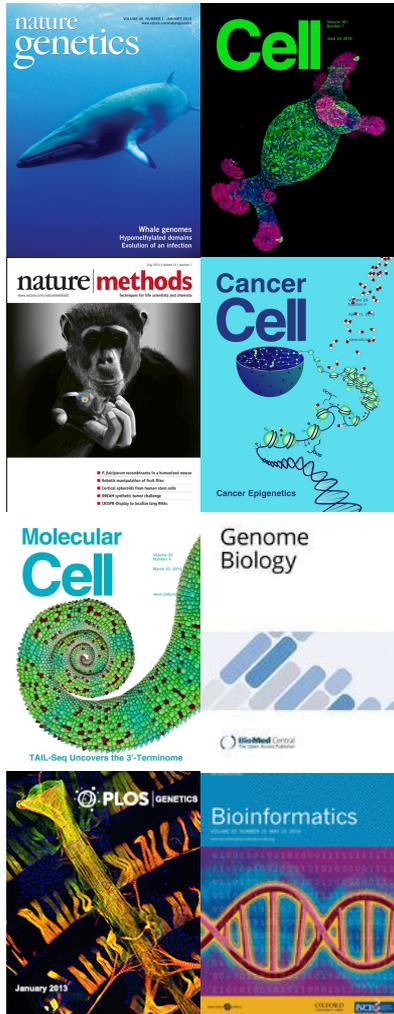
Johannes Köster<sup>1,2,\*</sup> and Sven Rahmann<sup>1</sup>

<sup>1</sup>Genome Informatics, Institute of Human Genetics, University of Duisburg-Essen and <sup>2</sup>Paediatric Oncology, University Childrens Hospital, 45147 Essen, Germany

Associate Editor: Alfonso Valencia

---

# Snakemake



## Genome of the Netherlands:

GoNL consortium. **Nature Genetics** 2014.

## Cancer:

Townsend et al. **Cancer Cell** 2016.

Schramm et al. **Nature Genetics** 2015.

Martin et al. **Nature Genetics** 2013.

## Ebola:

Park et al. **Cell** 2015

## iPSC:

Burrows et al. **PLOS Genetics** 2016.

## Computational methods:

Ziller et al. **Nature Methods** 2015.

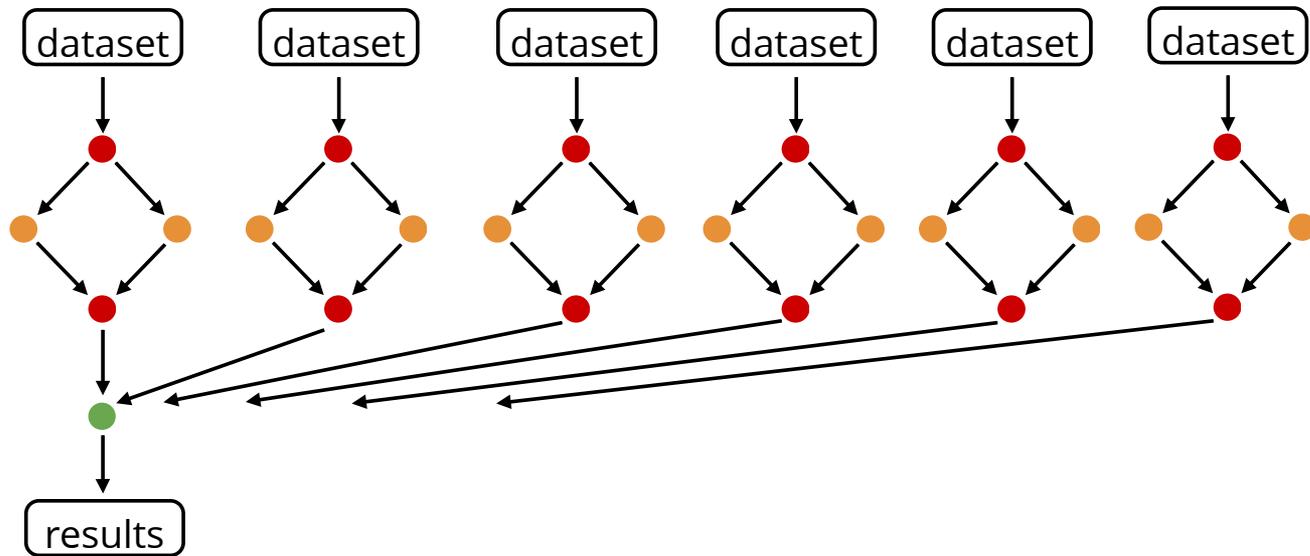
Schmied et al. **Bioinformatics** 2015.

Břinda et al. **Bioinformatics** 2015

Chang et al. **Molecular Cell** 2014.

Marschall et al. **Bioinformatics** 2012.

# Define workflows in terms of rules



# Define workflows in terms of rules

- 
- 
-

# Define workflows in terms of rules



```
rule mytask:  
  input:  
    "path/to/{dataset}.txt"  
  output:  
    "result/{dataset}.txt"  
  script:  
    "scripts/myscript.R"
```



```
rule myfiltration:  
  input:  
    "result/{dataset}.txt"  
  output:  
    "result/{dataset}.filtered.txt"  
  shell:  
    "mycommand {input} > {output}"
```



```
rule aggregate:  
  input:  
    "results/dataset1.filtered.txt",  
    "results/dataset2.filtered.txt"  
  output:  
    "plots/myplot.pdf"  
  script:  
    "scripts/myplot.R"
```

# Define workflows in terms of rules

rule name



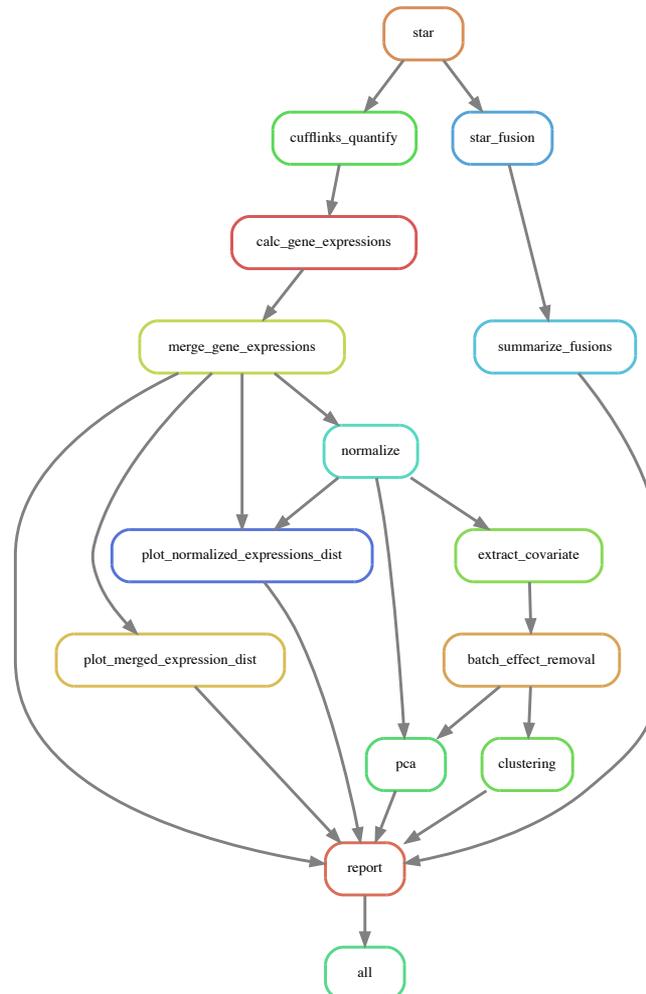
```
rule mytask:  
  input:  
    "data/{sample}.txt"  
  output:  
    "result/{sample}.txt"  
  conda:  
    "software-envs/some-tool.yaml"  
  shell:  
    "some-tool {input} > {output}"
```

how to create  
output from input  
(shell, Python, R)



refer to input and output  
from shell command

# Directed acyclic graph (DAG) of jobs



# Publikation

## Problem:

Keine festen Regeln in den Journal-Guidelines

## Beispiel Nature:

- the **exact sample size ( $n$ )** for each experimental group/condition, given as a number, not a range;
- a **description of the sample collection** allowing the reader to understand whether the samples represent **technical or biological replicates** (including how many animals, litters, culture, etc.);
- a **statement of how many times the experiment shown was replicated in the laboratory**;
- **definitions of statistical methods and measures**: (For small sample sizes ( $n < 5$ ) descriptive statistics are not appropriate, instead plot individual data points)
  - very common tests, such as  $t$ -test, simple  $\chi^2$  tests, Wilcoxon and Mann-Whitney tests, can be unambiguously identified by name only, but more complex techniques should be described in the methods section;
  - are tests one-sided or two-sided?
  - are there adjustments for multiple comparisons?
  - **statistical test results**, e.g.,  **$P$  values**;
  - definition of '**center values**' as **median** or **mean**;
  - definition of **error bars** as **s.d.** or **s.e.m.** or **c.i.**

# Vorschlag

## 1. Git repository:

```
├── config.yaml
├── scripts
│   ├── script1.py
│   └── script2.R
└── Snakefile
```

## 2. DOI zuweisen (figshare/Zenodo)

## 3. DOI im Manuskript zitieren

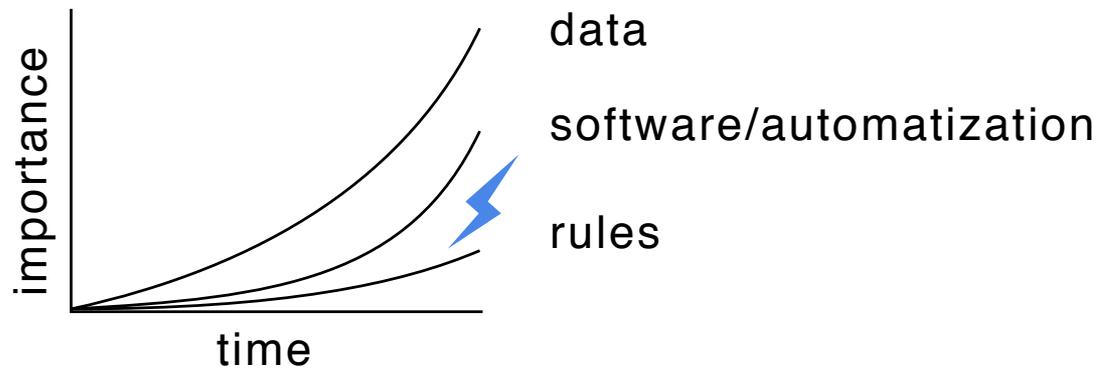
## 4. Resultate nachvollziehen und reproduzieren

```
# clone workflow into working directory
git clone https://bitbucket.org/user/myworkflow.git
cd myworkflow
```

```
# execute workflow locally
snakemake --cores 24
```

```
# execute workflow on a cluster
snakemake --jobs 1000 --drmaa
```

# Softwarekrise



# Softwarekrise

## **Wissenschaftliche Software wird häufig...**

- unzureichend dokumentiert
- ineffizient programmiert
- nicht gewartet

# Beispiel:

## ISMB 2016 "Wall of Shame"

Von 47 open-access Publikationen, ...

status	count
pubs without software implementation	7
based on the non-free MATLAB	4
either unclear, no, or erroneous installation instructions	3
invalid links	4
without any download link	1
collections of scripts without proper installer	12
R packages that are not yet in CRAN or Bioc	3
docker container	1
demo only although the README promises a release before the ISMB	1
available only upon request	2
already available in bioconda	2
web service	4
build error	1

# Ursache I

Zeitverträge, wechselnde Projekte und  
Mitarbeiter

## **Zu Lasten von:**

- Tests
- Wartung

# Ursache II

Programmierung als wissenschaftliche  
"Allgemeinbildung"

## **Problem:**

Programmierung != Informatik

## **Zu Lasten von:**

- algorithmischer Qualität (Effizienz)
- Code/Architekturqualität  
(Wiederverwendbarkeit)
- Distribution (Lizenz, Paketierung)

# Regeln

## Style-Guides beachten:

```
def make_complex(*args):  
    x, y = args  
    return dict(**locals())
```

VS

```
def make_complex(x, y):  
    return {'x': x, 'y': y}
```

# Regeln

## **Code-Duplikation vermeiden:**

Vererbung oder Delegation anstatt Copy-Paste

## **Testen:**

Unit- und Integration-Tests für möglichst 100%  
des Programms

# Regeln

## Das Rad nicht neu erfinden:

```
data = dict()
with open("somefile.tsv") as f:
    for line in f:
        line = line.split("\t")
        key, fields = line[0], line[1:]
        data[key] = [float(field) for field in fields]
```

VS

```
import pandas as pd

data = pd.read_table("sometable.tsv", index_col=0)
```

# Wahl der Programmiersprache

## **Ziel:**

Effiziente, ressourcenschonende, und  
fehlerfreie Software

# Wahl der Programmiersprache

## Ziel:

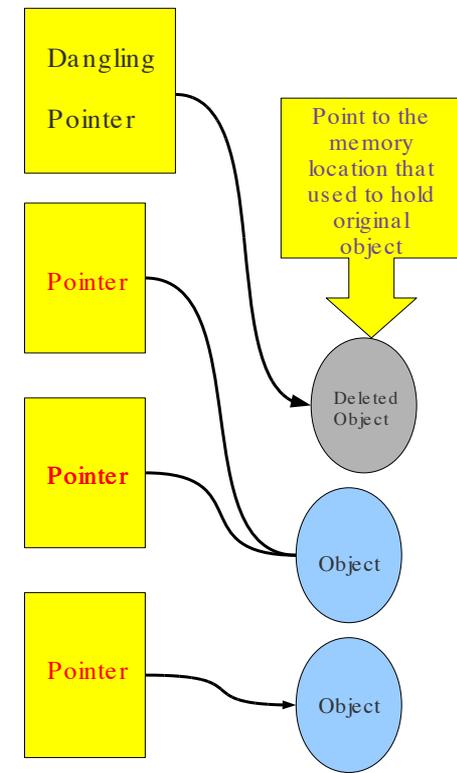
Effiziente, ressourcenschonende, und fehlerfreie Software

## Strategie I:

Klassische maschinennahe Sprachen  
(C, C++, ...)

## Problem:

**komplex** und **fehleranfällig**  
(Speichermanagement, Thread safety, ...)



Wikipedia

# Wahl der Programmiersprache

## Ziel:

Effiziente, ressourcenschonende, und fehlerfreie Software

## Strategie II:

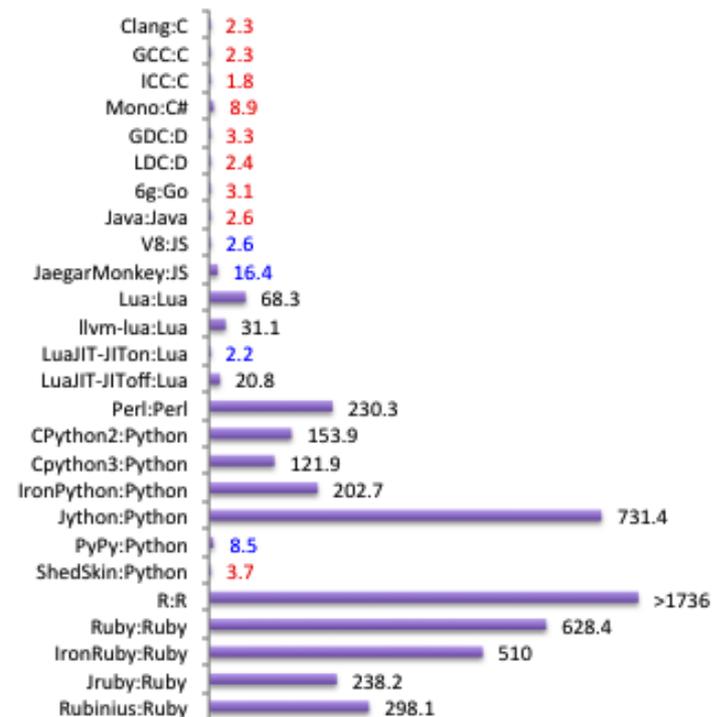
Skriptsprachen (Python, Ruby, ...)

## Problem:

**langsam** und **fehleranfällig**

(keine Typsicherheit,  
nur Laufzeitfehler, ...)

Matrix multiplication (CPU sec)



<https://attractivechaos.github.io/plb/>



- vereinfachte, C++-ähnliche Syntax
- Automatische Typ-Inferenz
- Elemente aus Skriptsprachen
- Sehr strikter Compiler der **Speicher-** und **Threadsicherheit** garantiert

## Ergebnis:

**Entwicklungszeit** verschiebt sich vom **Debugging** zum **Kompilieren**.

Software ist **leichter zu warten**.

# Rust-Bio

*Bioinformatics*, 32(3), 2016, 444–446

doi: 10.1093/bioinformatics/btv573

Advance Access Publication Date: 6 October 2015

Applications Note

OXFORD

---

Sequence analysis

## **Rust-Bio: a fast and safe bioinformatics library**

**Johannes Köster**

Center for Functional Cancer Epigenetics, Dana-Farber Cancer Institute, Department of Biostatistics and Computational Biology, Dana-Farber Cancer Institute, Harvard School of Public Health and Department of Medical Oncology, Dana-Farber Cancer Institute, Harvard Medical School, MA02215, Boston, USA.

Associate Editor: John Hancock

Received on June 20, 2015; revised on September 14, 2015; accepted on September 28, 2015

# Software installation is heterogeneous

```
source("https://bioconductor.org/biocLite.R")
biocLite("DESeq2")
```

```
cp lib/amd64/jli/*.so lib
cp lib/amd64/*.so lib
cp * $PREFIX
```

```
install.packages("matrixpls")
```

```
cmake ../../my_project \
  -DCMAKE_MODULE_PATH=~/.devel/seqan/util/cmake \
  -DSEQAN_INCLUDE_PATH=~/.devel/seqan/include
make
make install
```

```
easy_install snakemake
```

```
./configure --prefix=/usr/local
make
make install
```

```
cpan -i bioperl
```

```
apt-get install bwa
```

```
yum install python-h5py
```

# Der Conda-Paketmanager

Normalisierung von Installationsroutinen  
über "Recipes":

source or binary



```
package:  
  name: seqtk  
  version: 1.2  
  
source:  
  fn: v1.2.tar.gz  
  url: https://github.com/lh3/seqtk/archive/v1.2.tar.gz  
  
requirements:  
  build:  
    - gcc  
    - zlib  
  run:  
    - zlib  
  
about:  
  home: https://github.com/lh3/seqtk  
  license: MIT License  
  summary: Seqtk is a fast and lightweight tool for processing sequences  
  
test:  
  commands:  
    - seqtk seq
```



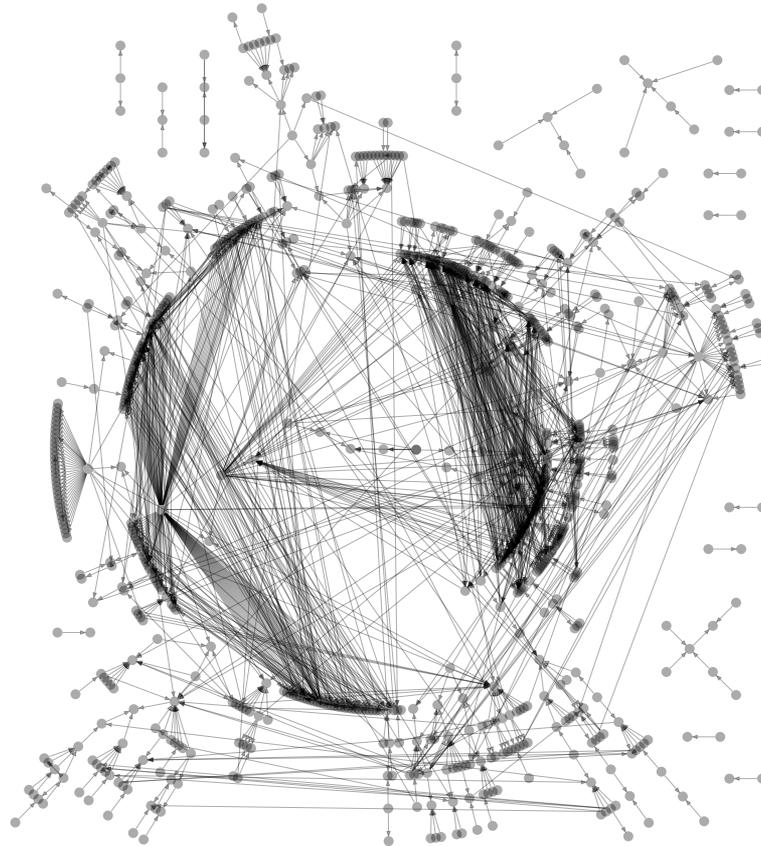
package

- Installationen **ohne Admin-Rechte**
- **Isolierte** Softwareumgebungen

# BIOCONDA®

Already over **1600** bioinformatics related  
conda packages

(C, C++, Python, R, Perl, ...)





# BIOCONDA<sup>®</sup>

Integrated with **3** popular  
workflow management systems:

Snakemake



bcbio

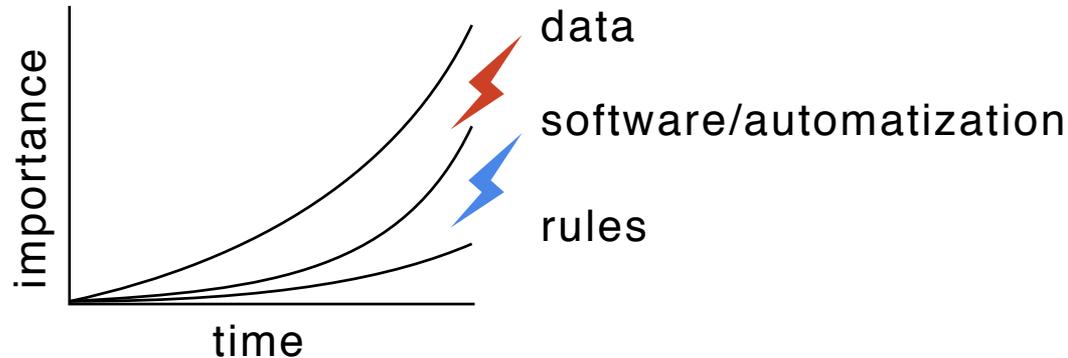
Partner project for general purpose software:



CONDA-FORGE

# Zusammenfassung

## In-silico-Reproduzierbarkeitskrise:



## Automatisierungskrise:

- Workflow Management mit Snakemake
- Regeln für die Publikation von Datenanalysen

## Softwarekrise:

- Besserer Code mit Rust
- Distribution mit Conda