

Research methods

Experimental CS research methods

Matteo Ceccarelli

Acknowledgements: thanks to Anton Dignös, Johann Gamper, and Francesco Ricci for the material on which part of these slides are based

About me

- RTD/a
 - Currently in the Database research group, studying temporal aspects of databases and time series
 - Background in theoretical computer science, parallel algorithms in particular
 - Lots of experiments...
 - ...and lots of errors!
-
- Most of what you will see is rooted in experimental algorithmics, but is applicable in the more general case

Assessment

- Homework
 - You should each read a paper about an empirical/experimental evaluation
 - Prepare a very short presentation (10 mins) where you illustrate the article, focusing on the experimental evaluation
- Critical presentation of the assigned article, showing that you have considered and evaluated all the dimensions illustrated in the lectures

How to structure your presentation

- State the problem addressed by the paper
 - Don't describe their method with too much detail, just give enough context to understand the experimental results
- Discuss the findings of the experimental section, focusing on pitfalls and improvements
- 10 minutes total
- If you use slides, no more than 10 slides!

Suggested papers

- Ding et al.: Querying and mining of time series data: experimental comparison of representations and distance measures, VLDB 2008
- Zhang et al.: Crowdsourced top-k algorithms: an experimental evaluation, VLDB 2016
- Lu et al.: Large-scale distributed graph computing systems: an experimental evaluation, VLDB 2016
- Papenbrock et al.: Functional dependency discovery: an experimental evaluation of seven algorithms, VLDB 2016
- Wu et al.: Shortest path and distance queries on road networks: an experimental evaluation, VLDB 2012
- Li et al.: An experimental study on hub labeling based shortest path algorithms, VLDB 2018.
- Jiang et al.: String similarity joins: an experimental evaluation, VLDB 2014
- Chen et al.: Spatial keyword query processing: an experimental evaluation, VLDB 2013
- Han et al.: An experimental comparison of Pregel-like graph processing systems, VLDB 2014
- Lu et al.: Large-scale distributed graph computing systems, VLDB 2014

Suggested papers/2

- Weber et al.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces, VLDB 1998
- Huang et al.: Experimental evaluation of real-time optimistic concurrency control schemes, VLDB 1991
- Zhang et al.: An experimental evaluation of simrank-based similarity search algorithms, PVLDB 2017.
- Memarzia et al.: A Six-dimensional Analysis of In-memory Aggregation, EDBT 2019
- Mann et al.: An Empirical Evaluation of Set Similarity Join Techniques, PVLDB 2016
- Blumenthal et al.: Comparing heuristics for graph edit distance computation, VLDB Journal 2019.
- Khayati et al.: Mind the Gap: An Experimental Evaluation of Imputation of Missing Values Techniques in Time Series, PVLDB 2020.
- Aumüller et al.: ANN-Benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Inf. Syst. 2020
- Graf and Lemire: Xor Filters: Faster and Smaller Than Bloom and Cuckoo Filters JEA 2020
- Kriegel et al: The (black) art of runtime evaluation: Are we comparing algorithms or implementations? Knowl. Inf. Syst. 2017
- McSherry et al. Scalability, but at what COST? HotOs 15

Starting points when reading an experimental evaluation

- Is the experimental setup well described?
- Are the goals of the experiments well stated?
- Is the choice of parameters sensible?
- Are there appropriate baselines?
- Is the data analysis sound?
- Is the code available?
- Are the datasets available?

Course structure

- Overview of research methods
- The experimental process in Computer Science
- Planning an experiment
- Reproducibility
- Running experiments

Research methods, techniques and methodology

- **Research method:** refers to the manner in which a particular research project is undertaken
- **Research technique:** refers to a specific means, approach, or tool-and-its-use, whereby data is gathered and analysed, and inferences are drawn
- **Research methodology:** refers to the study of research methods

Different research methods exist

Exploratory research

- Improve the basic knowledge on a concept and walk into the unknown realms of the subject
- Conducted on a project that has not been clearly defined
- It shouldn't draw definitive conclusions
- May conclude that a problem does not actually exist

Constructive research

- Find a new solution to a specific persisting problem
- Very common in computer science
- Construct: new theory, algorithm, software, model, framework
- Demands a form of validation: theoretical analysis or benchmark tests

Empirical research

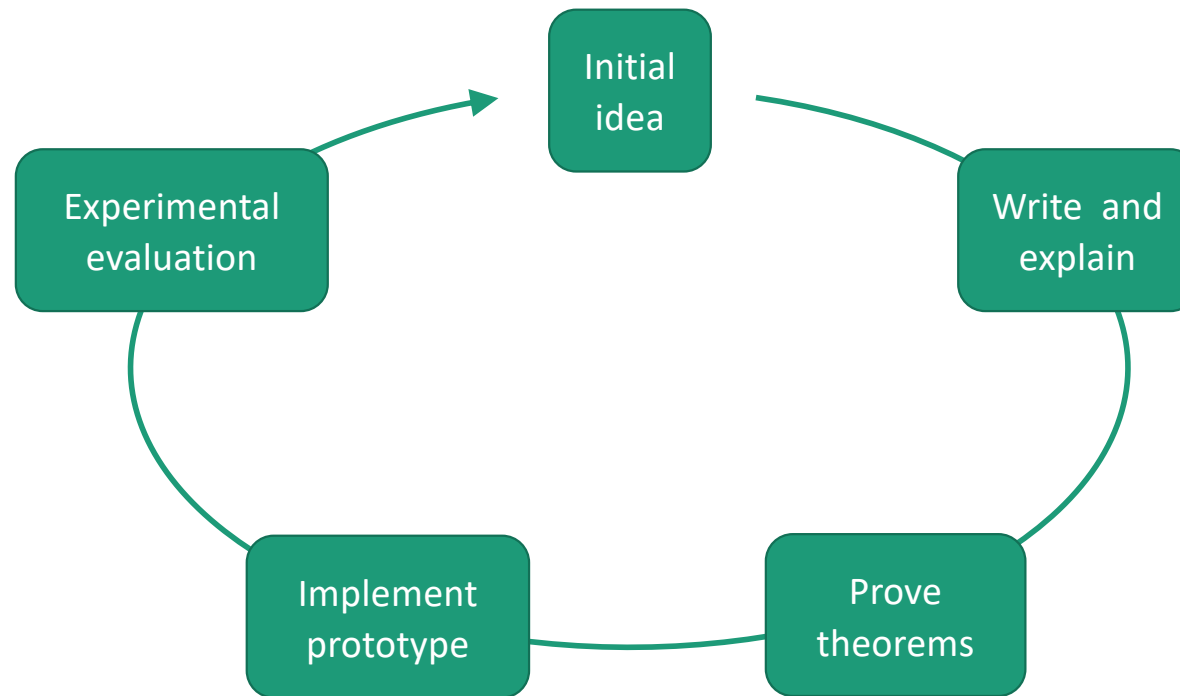
- Based on the observation of some phenomenon
- Empirical evidence can be analysed quantitatively or qualitatively
- Develop theories and models that are then validated or rejected by the observation

Why do experiments?

Why do experiments?

- Go beyond simply reporting observations or proving theorems
- Fill the gaps between theory and practice
- Help integrating research and practice
- Characterize the performance of worst-case, average-case, etc.. instances
- Force yourself to be very precise
- Provide insights into the theory
 - Can also suggest theoretical improvements for an algorithm

Experimental research improves the quality of research



Each step reveals **weaknesses** and **errors**

The experimental process

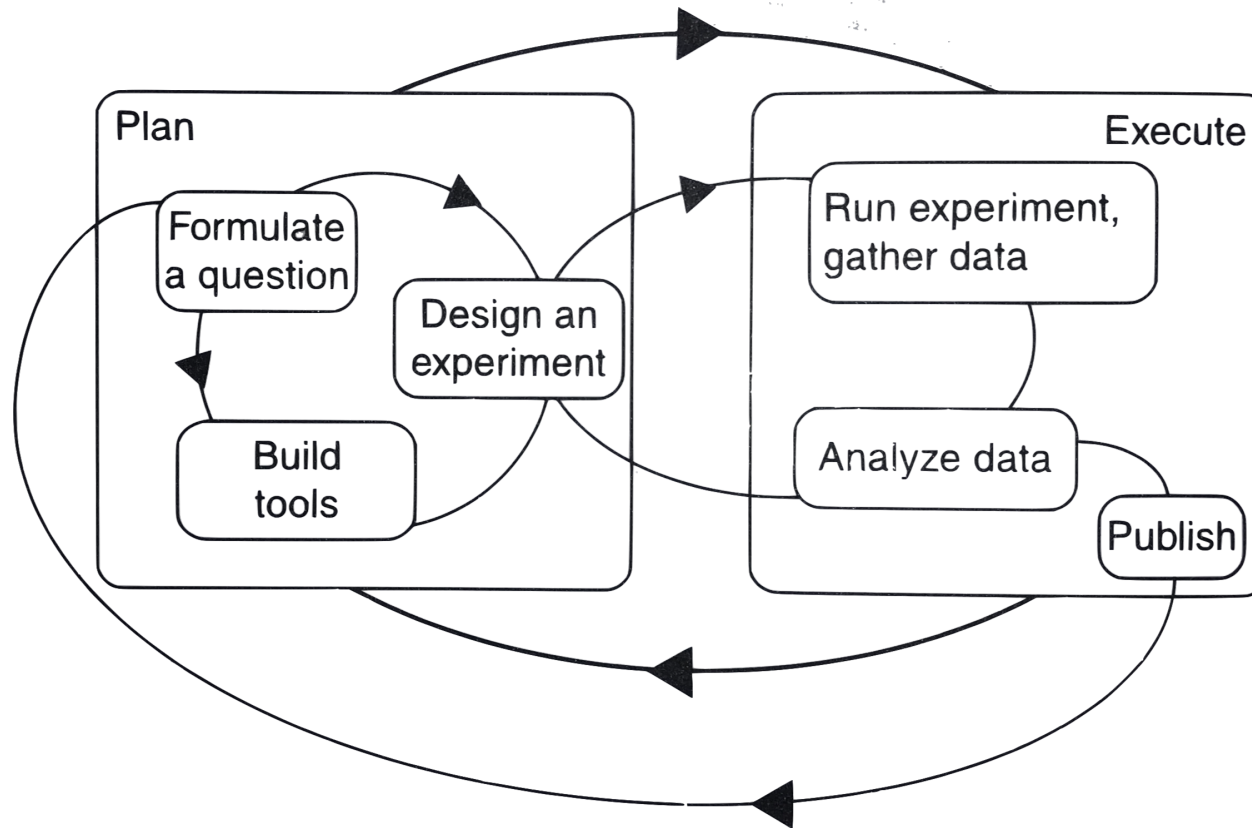


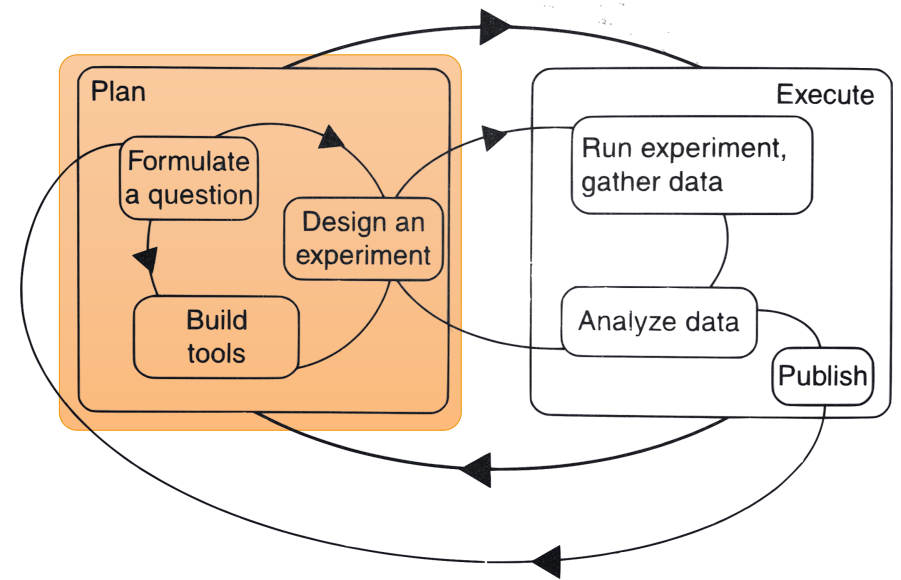
Fig 1.2: McGeoch *A guide to experimental algorithmics*

Experimental goals

Reproducibility and correctness

Generality and efficiency

Newsworthiness



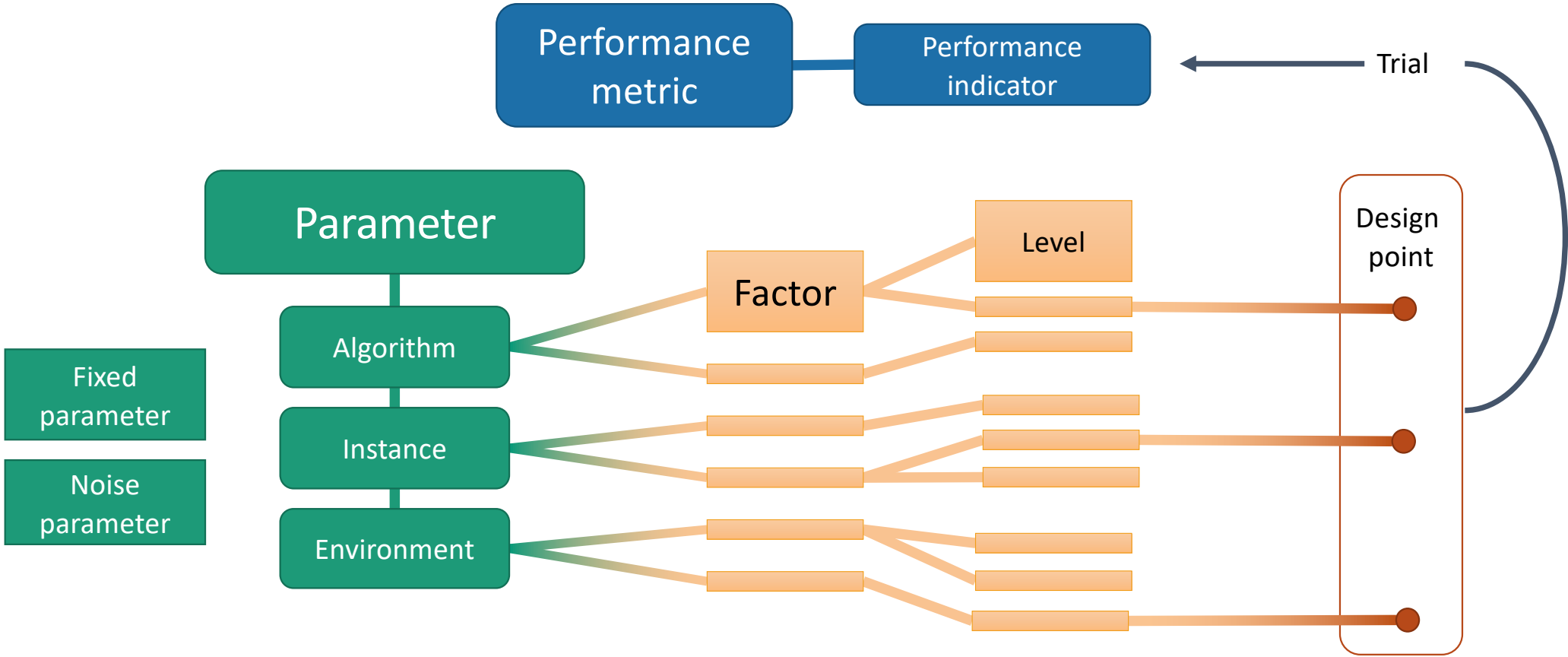
Planning an experiment

Material based on the book

“A guide to experimental algorithmics”

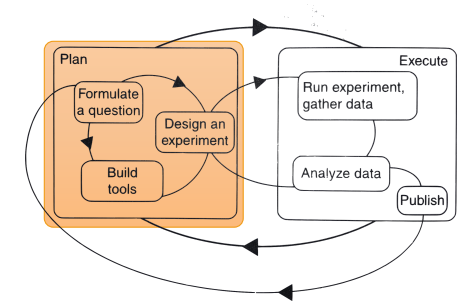
by Catherine McGeoch

Experimental design basics



How do we plan an experiment?

We might have a precise question, or we might not.
We might have a set of parameters to try, or we might not
We might have some assumptions about the algorithm/datasets

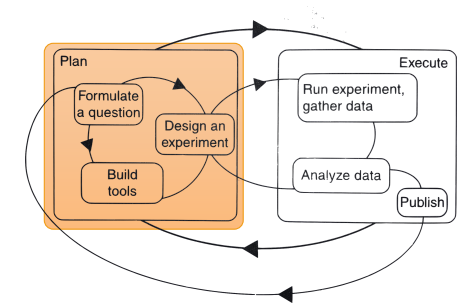


Run a *pilot study* first

A small set of quick
and dirty experiments
that don't cover too
many design points

- Check correctness/basic assumptions
- Learn the quirks of the datasets/algorithms/test environments
- Identify the most promising ideas and parameter configurations

How do we plan an experiment?



Workhorse experiments

With the knowledge acquired by means of the pilot study

- formulate precise questions
- select appropriate datasets to find answers
- select the design points to test

It's very important to run experiments with purpose: you should motivate every choice (of parameters, algorithms, answers) you make. Otherwise you end up with a pile of results you can't make a sense of

Datasets/inputs

Which types of datasets can you think of?

Stress test instances

Worst case instances

Random inputs

Structured random inputs

Real instances

Hybrid instances

Choose input classes to support goals of *correctness* and *generality*, and to *target the question* at hand

Choosing factors and design points

- Leverage the **pilot study** to find the factors that influence performance the most
- Similarly, choose the parameters and metrics that **highlight differences** the most
- To quickly explore factor ranges, use **doubling experiments**

Doubling experiments

- Suppose you are testing an algorithm with a parameter k .
- To quickly assess the influence of k on the performance, you can run a pilot study with $k \in \{1, 2, 4, 8, 16 \dots\}$ or $k \in \{1, 10, 100, 1000, \dots\}$
- The idea is to use a **geometric progression** to quickly cover the space of possible factor levels

Choosing factors and design points

- Leverage the **pilot study** to find the factors that influence performance the most
- Similarly, choose the parameters and metrics that **highlight differences** the most
- To quickly explore factor ranges, use **doubling experiments**
- Leverage what you **know**:
 - e.g. your algorithm has a phase transition at a particular value of a parameter
- Run a **full factorial** design

Full factorial designs

- For each factor, choose a **finite set of levels**
- Each combination defines a **design point**
- You **run a trial for each** design point
- **Exponential** in the number of factors! Can quickly go out of hand

Factor reduction techniques

- **Merge** similar factors
- **Trace** the behaviour of your experiments: enables **trial overloading**
- **Convert** some factors to **noise parameters**
Replace the explicit choice with a simple random distribution
- **Fix** some factors, thus **limiting the scope** of the experiment
- **Remove factors** that have little influence on the performance

Choosing performance metrics and indicators

- Running time:
 - Wall clock time
 - CPU time
 - Operation counters
 -
- Accuracy measures:
 - Precision/recall
 - F_1 score
 - RMSE
 -
- You can evaluate several metrics at once, to see if there are tradeoffs

The (black) art of runtime evaluation: Are we comparing algorithms or implementations?

Hans-Peter Kriegel¹ · Erich Schubert¹ ·
Arthur Zimek² 

Received: 6 August 2015 / Revised: 24 August 2016 / Accepted: 15 October 2016 /
Published online: 22 October 2016
© Springer-Verlag London 2016

Abstract Any paper proposing a new algorithm should come with an evaluation of efficiency and scalability (particularly when we are designing methods for “big data”). However, there are several (more or less serious) pitfalls in such evaluations. We would like to point the attention of the community to these pitfalls. We substantiate our points with extensive experiments, using clustering and outlier detection methods with and without index acceleration. We discuss what we can learn from evaluations, whether experiments are properly designed, and what kind of conclusions we should avoid. We close with some general recommendations but maintain that the design of fair and conclusive experiments will always remain a challenge for researchers and an integral part of the scientific endeavor.

Select appropriate baselines

- As we have seen, most CS research is **constructive**
- As a consequence, anyone can come up with a new solution: you need to show that yours is performing better according to some measure.
- However, it is crucial to consider appropriate baselines!

Select appropriate baselines/2

Armstrong et al.: *Improvements That Don't Add Up: Ad-Hoc Retrieval Results Since 1998*, CIKM 2009

- «Most researchers only **report results from their own experiments**, a practice that allows lack of overall improvement to go unnoticed.»
- «Our longitudinal analysis of published IR results in SIGIR and CIKM proceedings from 1998-2008 has uncovered the fact that **ad-hoc retrieval is not measurably improving**. »
- «A **central repository of effectiveness results** presents a solution to this problem: best known results could be quickly found by authors, and readers and reviewers could more effectively assess claims made in papers.»

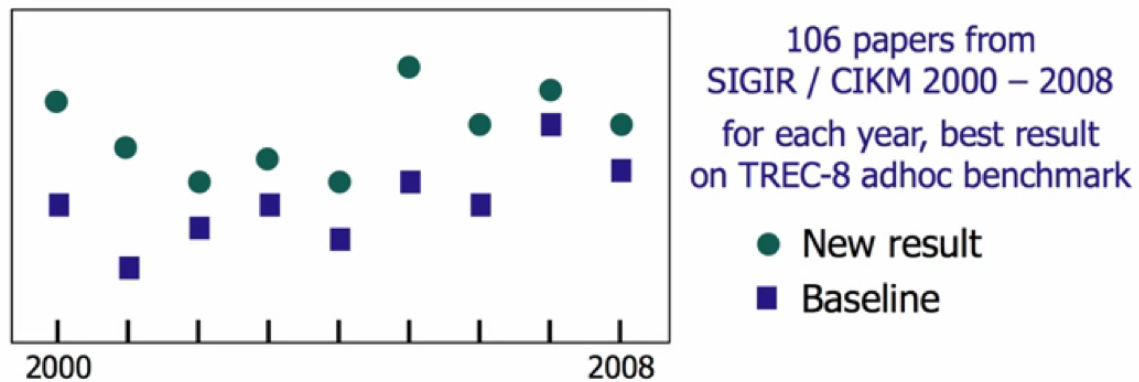
Select appropriate baselines/3

Jens Dittrich: The Case for Small Data Management

<https://youtu.be/O7Qgo6RSzmE?t=19m>

– Even worse: published papers all claim improvements

But they all compare to weak baselines

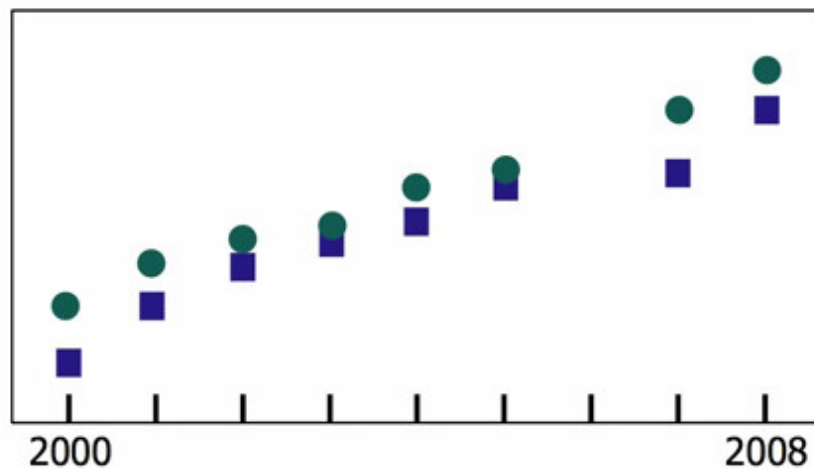


"Improvements that don't add up: adhoc results since 1998", CIKM 2009
(data points manually copied from Figure 4a in that paper)

Select appropriate baselines/3

– Even worse: published papers all claim improvements

But they all compare to weak baselines



**The picture
should rather
look like this**

● New result
■ Baseline

"Improvements that don't add up: adhoc results since 1998", CIKM 2009

(data points manually copied from Figure 4a in that paper)

Benchmarks

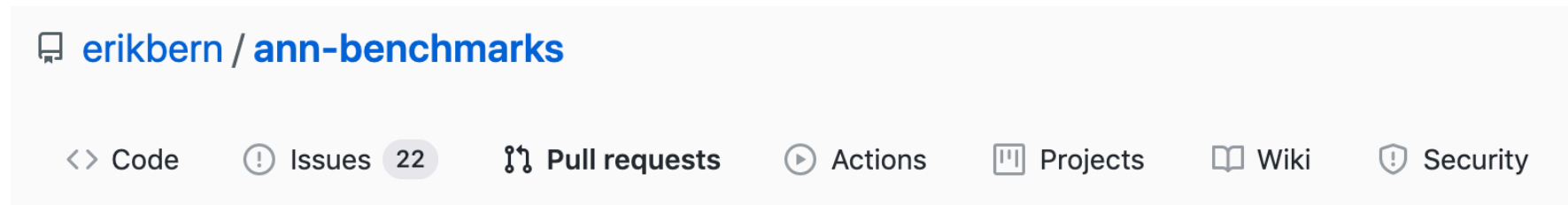
- A benchmark is a standard test or set of tests used to compare alternatives. It consists of the following components:
 - Motivating comparison
 - Task sample
 - Performance measures
- Becomes a standard through acceptance by a community
- What are the advantages of standardized benchmarks?

Benchmarks: the advantages

- Stronger consensus on the community's research goals
- Greater collaboration between laboratories
- More rigorous validation of research results
- Rapid dissemination of promising approaches
- Faster technical progress
- Benefits derive from process, rather than end product

ANN-benchmarks

- <http://ann-benchmarks.com/>
- Benchmarks for the k-nearest neighbours problem
- Tests approximation algorithms in terms of speed of execution, recall, index build time, etc...



erikbern / **ann-benchmarks**

<> Code ! Issues 22 **🔗 Pull requests** ▶ Actions 📁 Projects 📖 Wiki 🛡 Security

Add ScaNN to ann-benchmarks #165

Merged

erikbern merged 2 commits into `erikbern:master` from `unknown repository` on 26 Jun

Other benchmark suites

- TREC Ad Hoc Task (information retrieval)
- TPC-ATM (database)
- UCR Time Series Classification Archive
- SPEC CPU2017 (CPU performance)
- Calgary Corpus and Canterbury Corpus (text compression)
- Penn treebank (NLP)

Reproducibility

Finding a treasure is useless, if you can't trace back your steps



What is reproducibility for you?

When is reproducibility important?

- Someone questions your conclusions
- One year later, you want to re-run the analysis with new data
- One year later, you want to slightly modify the analysis
- You are collaborating with someone else

A L W A Y S

DOI:10.1145/3360311

Research replication only works if there is confidence built into the results.

BY ANDY COCKBURN, PIERRE DRAGICEVIC, LONNI BESANÇON, AND CARL GUTWIN

Threats of a Replication Crisis in Empirical Computer Science

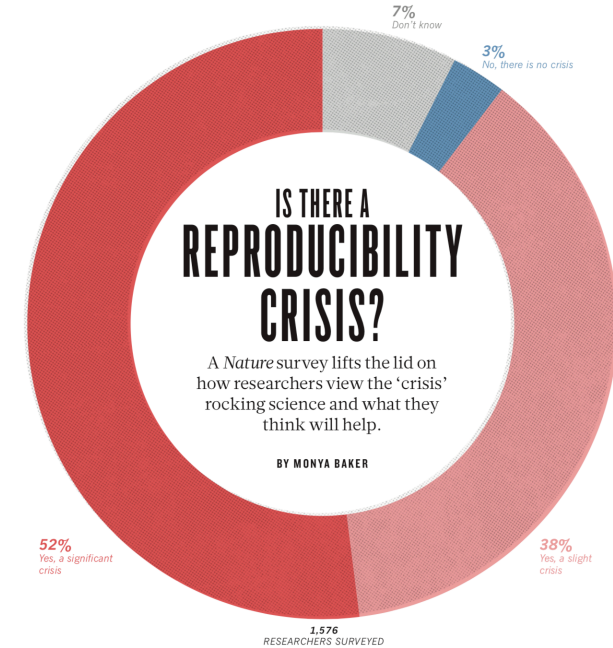
Cockburn et al. CACM 2020

COMPUTER SCIENCE

Artificial intelligence faces reproducibility crisis

Unpublished code and sensitivity to training conditions make many claims hard to verify

Hutson, Science 2018



Baker and Penny, Nature 2016

Scenario 1

- You install some libraries
- You develop a program using those libraries
- You send the program to someone else
- The program breaks in mysterious and subtle ways



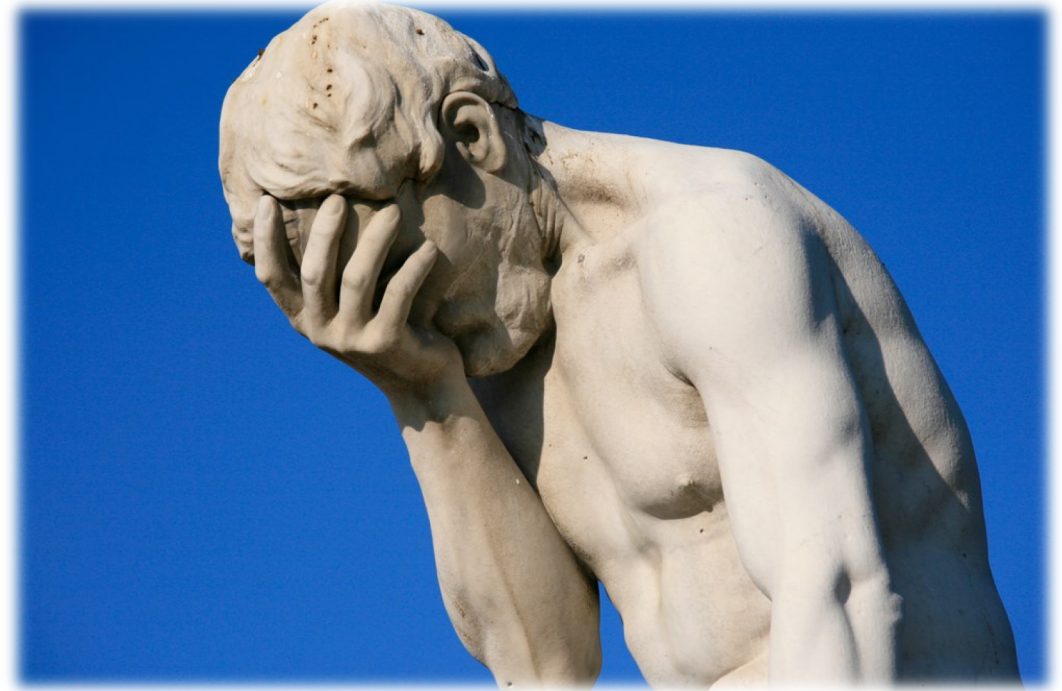
Scenario 2

- You install some libraries
- You develop a program using those libraries
- You start a new project, for which you need an updated version of the libraries
- After a while, you go back to your first project, and it's broken in mysterious and subtle ways!

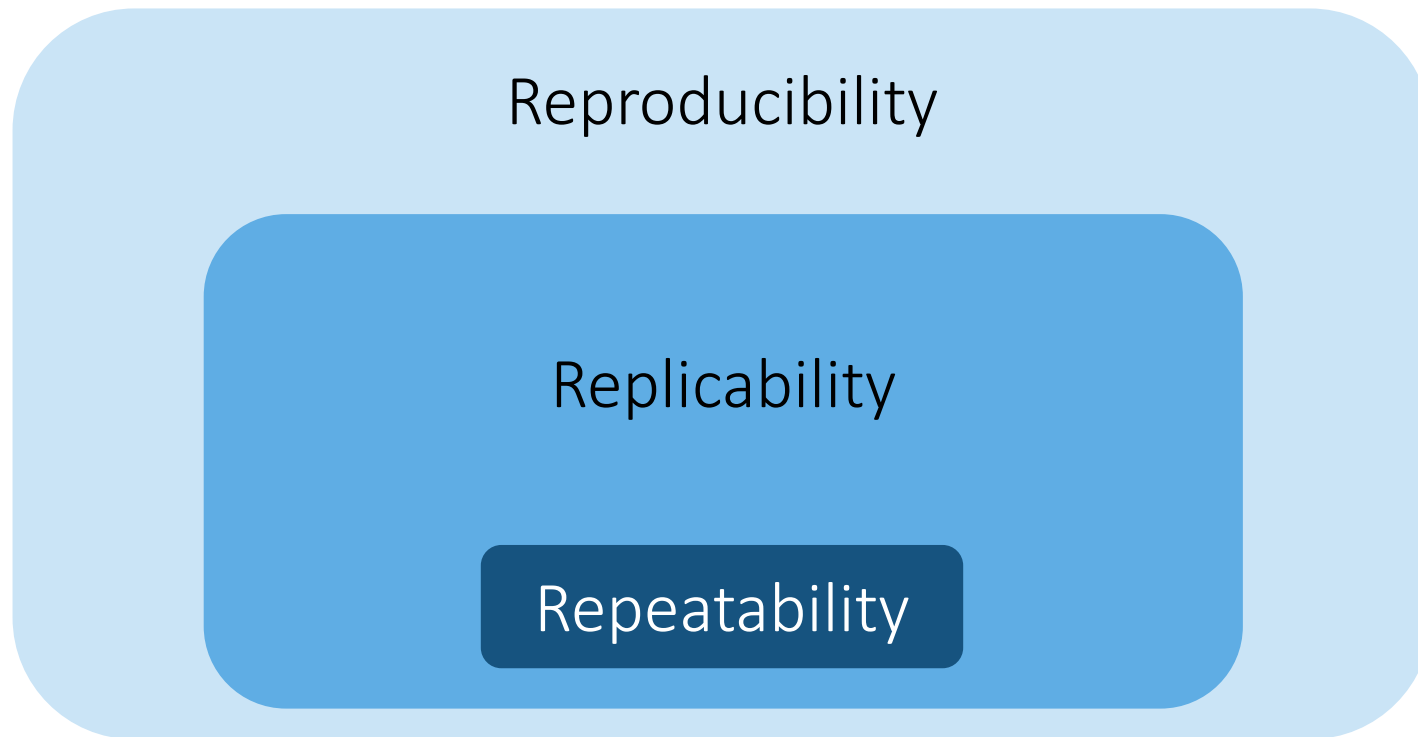


Scenario 3

- You download some data
- Preprocess it
- Use it in your experiments
- After one year, you want to write the journal version of the paper, but you can't recreate the same dataset



The three flavours of reproducibility



Repeatability

Same team, same experimental setup

The measurement can be obtained with stated precision by the same team using the same measurement procedure, the same measuring system, under the same operating conditions, in the same location on multiple trials.

For computational experiments, this means that a researcher can reliably repeat her own computation.

Definitions of the ACM (Association of Computing Machinery)
Source: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5778115/>

Replicability

Different team, same experimental setup

The measurement can be obtained with stated precision by a **different team** using the same measurement procedure, the same measuring system, under the same operating conditions, in the same or a different location on multiple trials.

For computational experiments, this means that an independent group can obtain the same result using the author's own artifacts.

Definitions of the ACM (Association of Computing Machinery)
Source: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5778115/>

Reproducibility

Different team, different experimental setup

The measurement can be obtained with stated precision by a **different team**, a **different measuring system**, in a **different location** on multiple trials.

For computational experiments, this means that an independent group can obtain the same result using artifacts which they develop completely independently.

Definitions of the ACM (Association of Computing Machinery)
Source: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5778115/>

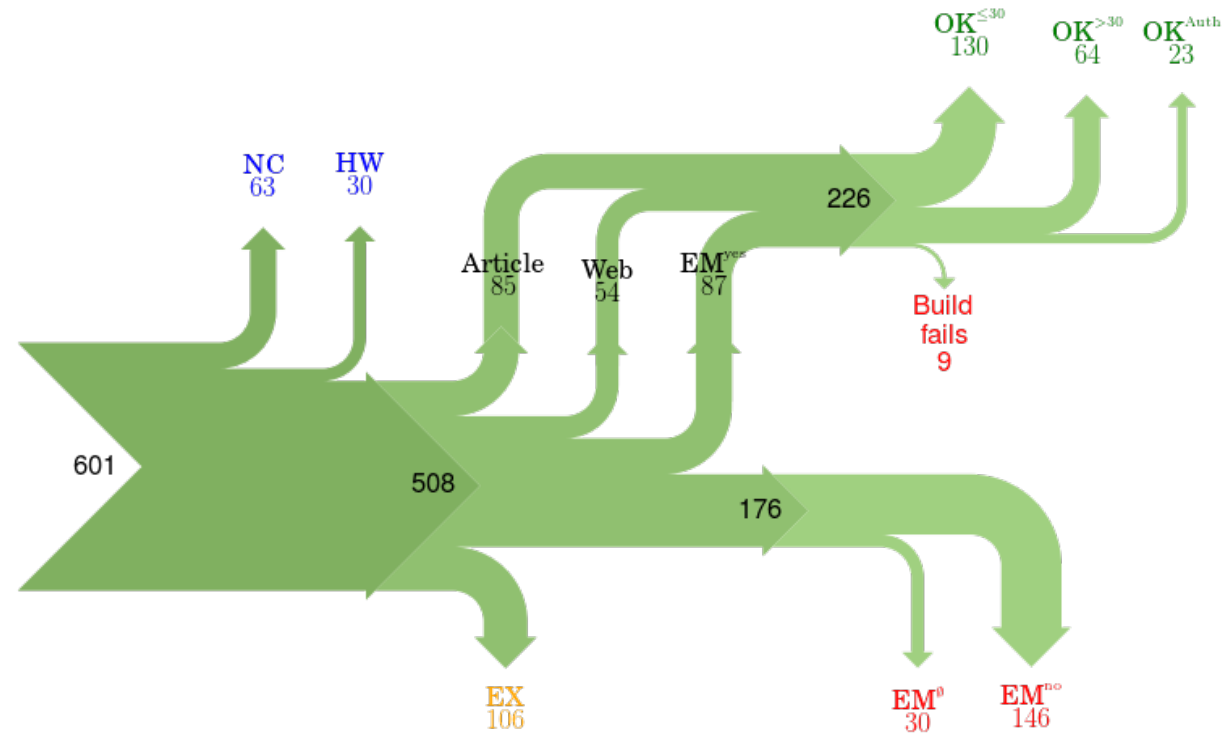
You want your results to be at least **replicable**

Attention: if you are not careful enough, you may have something that is not even repeatable!

It is very important to aim at reproducibility from the very beginning, it cannot be an afterthought.

A study of Replicability

<http://repeatability.cs.arizona.edu/>



Study 601 papers from ACM journals and conferences.
The task is just to **try to build the code**.

Reproducibility efforts

- Initiated by **SIGMOD 2012**

The goal of establishing reproducibility is to ensure your SIGMOD research paper stands as reliable work that can be referenced by future research. The premise is that experimental papers will be most useful when their results have been tested and generalized by objective third parties.

- Joined by **PVLDB** in 2018

What is SIGMOD Reproducibility?

SIGMOD Reproducibility has three goals:



- Highlight the impact of database research papers.
- Enable easy dissemination of research results.
- Enable easy sharing of code and experimentation set-ups.

In short, the goal is to assist in building a culture where sharing *results*, *code*, and *scripts* of database research is the norm rather than an exception. The challenge is to do this efficiently, which means building technical expertise on how to do better research via creating repeatable and sharable research. The [SIGMOD Reproducibility committee](#) is here to help you with this.

Why should I be part of this?

You will be making it easy for other researchers to compare with your work, to adopt and extend your research. This instantly means more recognition for your work and higher impact.



Taking part in the SIGMOD Reproducibility process enables your paper to take the **ACM Results Replicated** label. This is embedded in the PDF of your paper in the ACM digital library.

There is an option to also host your data, scripts and code in the ACM digital library as well to make them available to a broad audience, which will award the **ACM Artifacts Available** label.





PVLDB Reproducibility

Starting with PVLDB 2018, pVLDB joins SIGMOD in encouraging the database community to develop a culture of sharing and cross-validation. PVLDB's reproducibility effort is being developed in coordination with [SIGMOD's](#).

News

Reproducibility [submissions](#) are now open through [CMT](#).

Recent Reproducibility Highlights

[GPU Rasterization for Real-Time Spatial Aggregation over Arbitrary Polygons](#)

[Clustering Uncertain Graphs](#)

[Beyond Macrobenchmarks: Microbenchmark-based Graph Database Evaluation](#)

What is PVLDB Reproducibility?

PVLDB Reproducibility has three goals:

- Increase the impact of database research papers.
- Enable easy dissemination of research results.
- Enable easy sharing of code and experimentation set-ups.

In short, the goal is to assist in building a culture where sharing results, code, and scripts of database research is the norm rather than an exception. The challenge is to do this efficiently, which means building technical expertise on how to do better research via creating repeatable and sharable research. The pVLDB Reproducibility committee is here to help you with this.

Taking notes: the first step towards reproducibility

Always keep a **logbook** where you record:

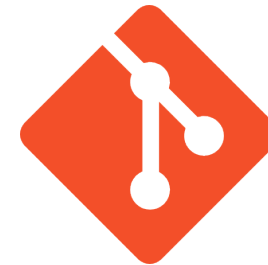
- the **experiments** you run
- **why** you run them
- the results and your **observations**

Doing so you will be able to reconstruct, even after one year, even if the entire codebase changed, what you did and why, and where those choices led you.

Use version control

- Source code
- Latex sources of papers
- Your notes
- Everything!

- <https://gitlab.inf.unibz.it/>
- <https://github.com/>

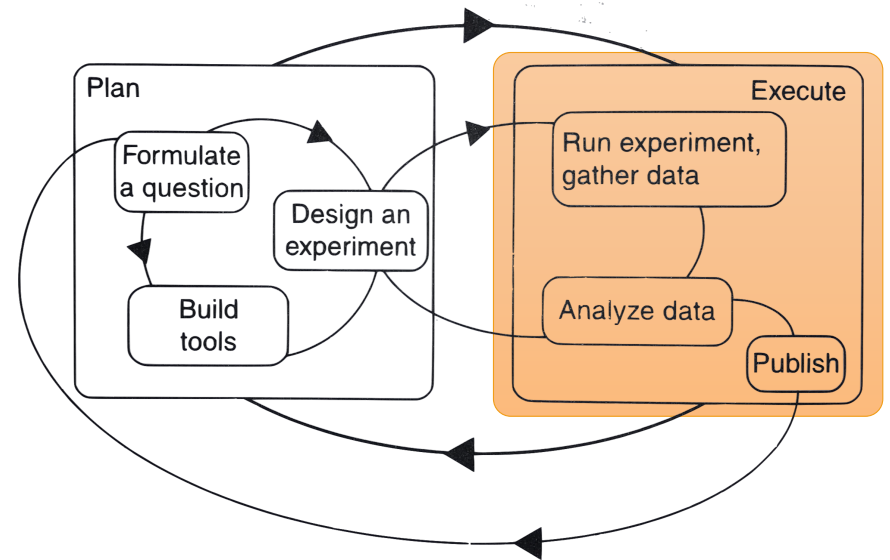


mercurial



Adopt a workflow that explicitly supports reproducibility

...more on this later



Running experiments

Material based on the SISAP 2020 paper and presentation

“Running experiments with confidence and sanity”

TEST your implementation!

Write unit tests

Check output against
a trivial implementation

What are the main challenges when running experimental evaluations?

Running experiments with confidence and sanity

Martin Aumüller¹ and Matteo Ceccarelo²

¹ IT University of Copenhagen, Denmark

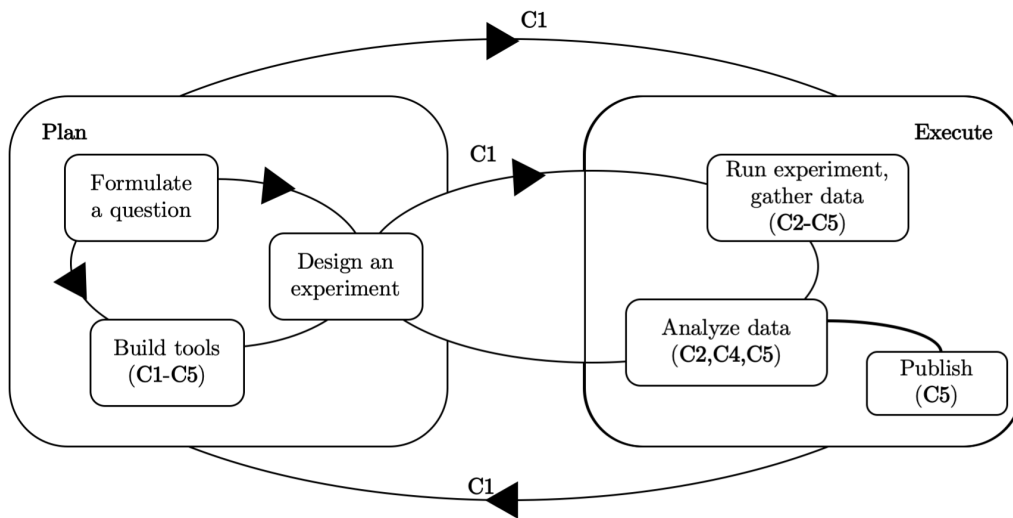
`maau@itu.dk`

² Free University of Bozen, Italy

`mceccarelo@unibz.it`

Abstract. Analyzing data from large experimental suites is a daily task for anyone doing experimental algorithmics. In this paper we report on several approaches we tried for this seemingly mundane task in a similarity search setting, reflecting on the many errors and consequent mishaps. We conclude by proposing a workflow, which can be implemented using several tools, that allows to analyze experimental data with confidence.

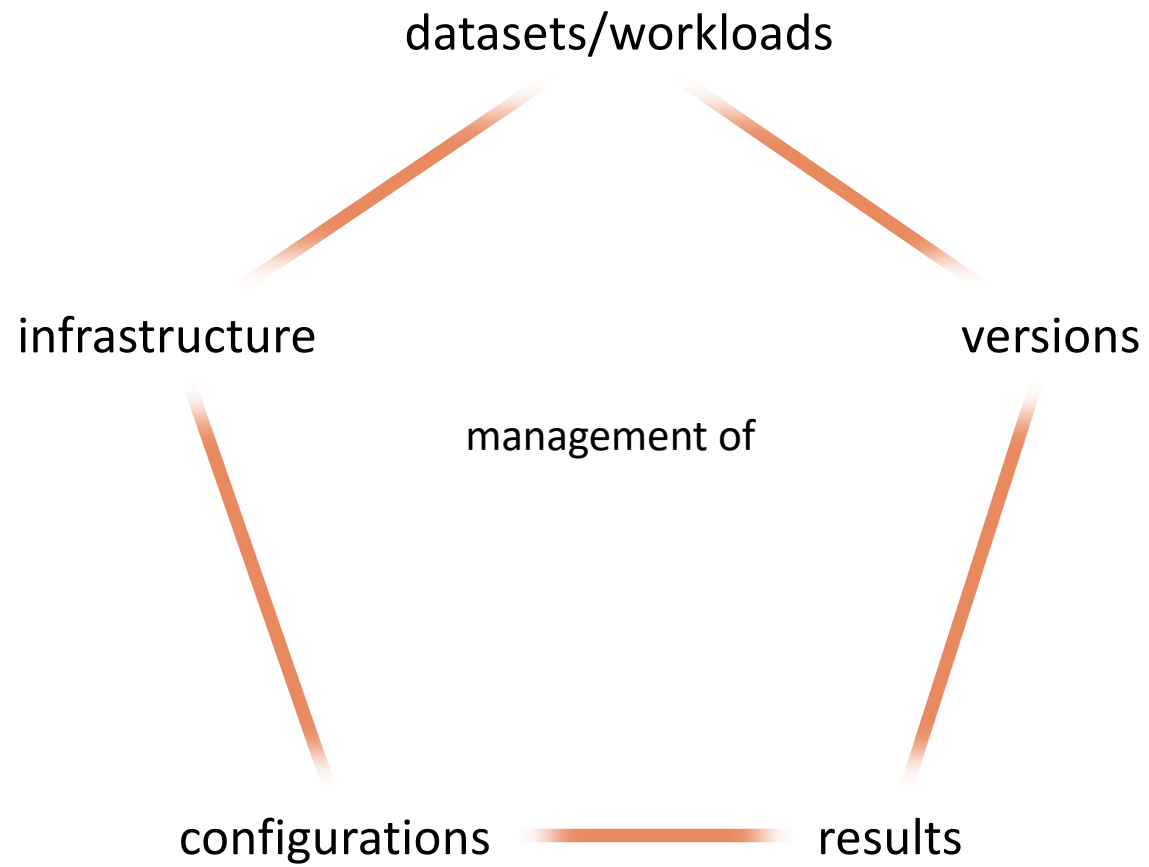
Keywords: Experimental algorithmics; Experimental analysis



All of these challenges are related to
managing information

Challenges

- (C1) Feedback loops by design
- (C2) Economic execution
- (C3) Versioning
- (C4) Machine independence
- (C5) Reproducibility by design



Dataset management

- Automate dataset download and preprocessing as much as possible
- Make preprocessed datasets publicly available
- Annotate datasets with meta-data (preprocessing parameters, ground truth values, ets...)
- For debugging, set up generators of small random testbeds

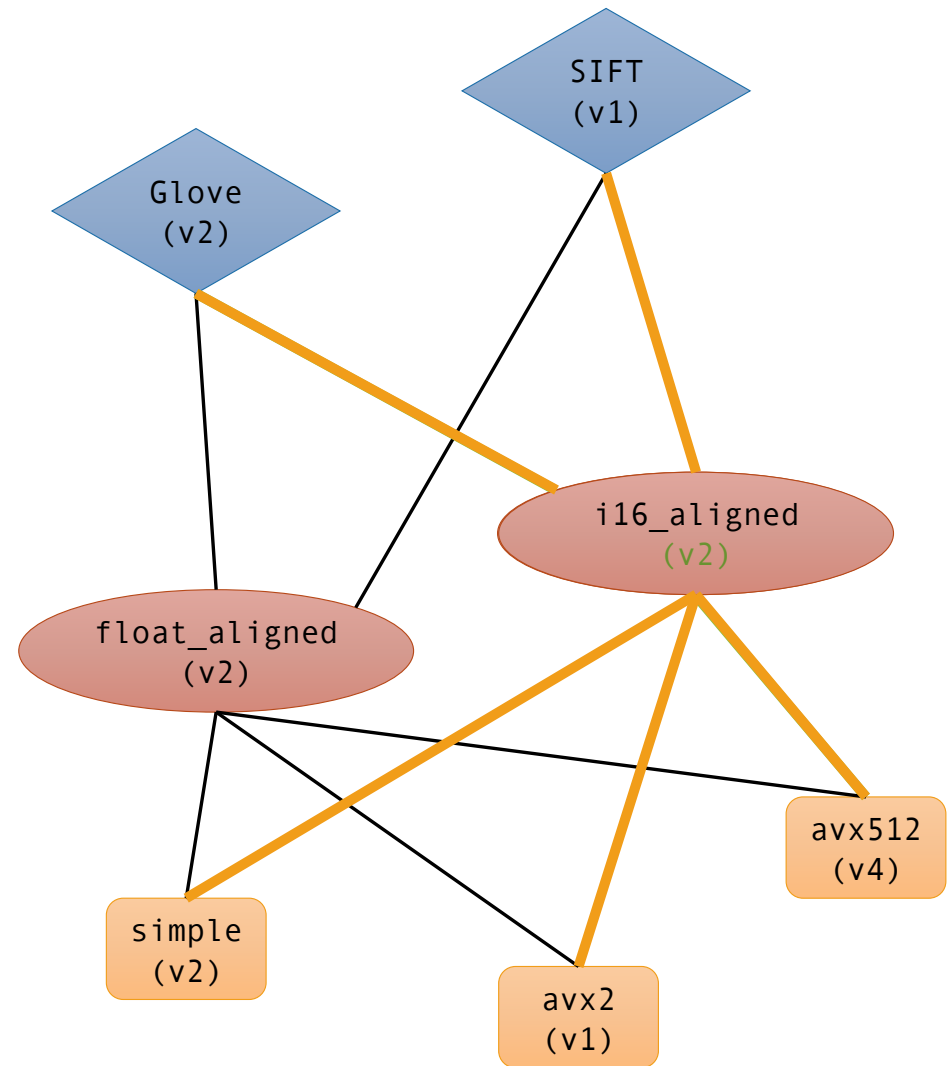
Configuration management

- Description of experiment in a separate file (**not** from the command line)
The file in the example configures a **full factorial** design
- Tracked in version control with no pending changes
- Mechanism to allow skipping already-run configurations
- Run whole pipeline in **Continuous Integration**

```
environment:  
  dataset: ['glove-100-angular']  
  seed: 4132  
  force: False  
experiments:  
  - name: baseline  
    storage: ['float_aligned']  
    method: ['simple', 'avx2', 'avx512']  
  - name: i16_alignment  
    storage: ['i16_aligned']  
    method: ['simple', 'avx2', 'avx512']  
  - name: sketches  
    storage: ['float_aligned', 'i16_aligned']  
    method: ['simple', 'avx2', 'avx512']  
    sketches: True  
    recall: [0.001, 0.1, 0.2, 0.5, 0.7, 0.9, 0.99]
```

Versioning

- Associate results to experiment/code versions
- Economic execution
- VCS IDs are insufficient




```
FROM ubuntu:20.10|
RUN apt-get update -qq
RUN DEBIAN_FRONTEND="noninteractive" apt-get -y install tzdata
RUN apt-get install -y \
  build-essential clang-format \
  sudo g++ g++-9 git linux-tools-generic \
  libssl-dev sqlite3 libsqlite3-dev \
  python3 python3-pip python3-yaml python3-h5py \
  python3-numpy python3-sklearn libhdf5-dev r-base \
  r-cran-tidyverse r-cran-dbi
RUN pip3 install jupyter pandas seaborn
RUN R -e "install.packages('ggiraph')"
```

Infrastructure Management

Containerized environment

- Current trend, e.g., VS Code
- Little setup cost, little execution cost

Goal:

- *Machine-independent pipeline*

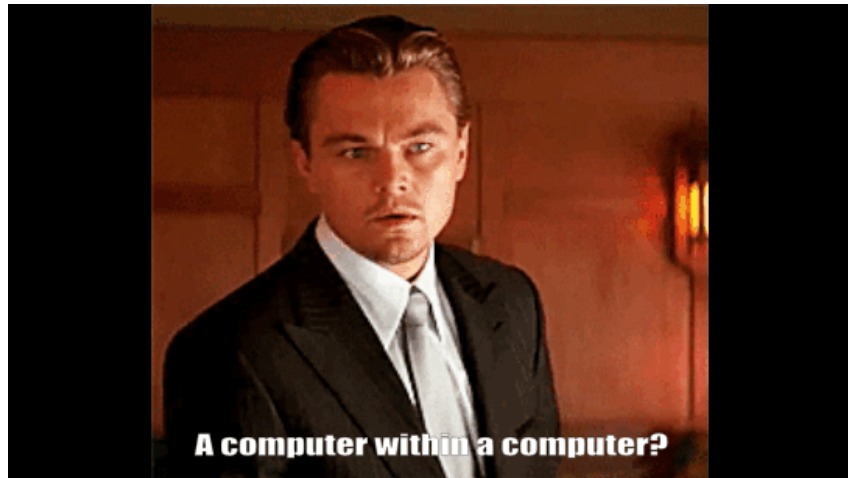
Bonus:

- Use CI to test all moving parts

Docker and containerized environments

A docker container is similar to a **virtual machine**:

- Has its own Unix operating system
- Has its own file system
- Has its own software applications
- Is much more **lightweight** than a virtual machine!



The idea is that you describe, in a **plain text file**, the configuration of the **entire software stack** you depend on.

This configuration can be instantiated on your machine, on someone else's, or in the cloud, guaranteeing that everywhere the **exact same code** is run!

A good tutorial is: *"An introduction to Docker for reproducible research, with examples from the R environment"* by Boettlinger [Arxiv 1410.0846]

Storing results



Custom file



JSON/CSV



Database

Evolving schemas

Feedback loops

Efficient access

Efficient analysis

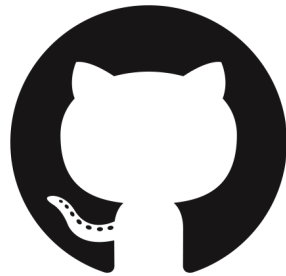
Enforce use of recent results

Confidence

Query from experiment's code

Economic execution

Demo implementation



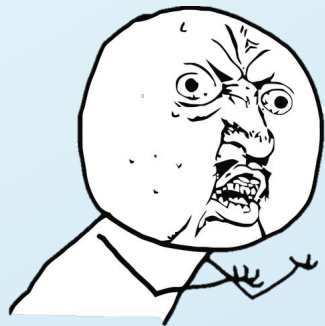
<https://github.com/Cecca/running-experiments>



<https://youtu.be/5Lgr2zp8LcM?t=582>

How does this workflow address reproducibility?

Scenario 1: you collaborate on a project but on different machines



Scenario 2: you use different versions of the same library in different projects



Scenario 3: you have to revisit your project, but no longer have the preprocessed dataset

