

Course overview

– Part 2

Intro to Bioinfo & Compbio

- Lay of the bioinfo-compbio land
- Reading papers | Framing the problem
- Choosing a good problem
- Resources @ MSU
- Getting help

Computational Biology & Bioinformatics

Computational biology

- The study of biology using computational techniques.
- **Goal**: learn new biology, knowledge about living systems.
It is about science.

Bioinformatics

- The creation of tools (algorithms, databases) that solve problems.
- **Goal**: build useful tools that work on biological data.
It is about engineering.



Dr. Margaret
Dayhoff
1925 – 1983

The first bioinformatician

Applying math & computational techniques to study protein & nucleic acid sequences.

- **1965:** First biological sequence database:
 - *Atlas of Protein Sequence and Structure*
 - Single-letter code for amino acids.
- **1966:** 'Evolutionary trees'.
- **1978:** First AA similarity-scoring matrix.
- **1980:** Launched the Protein Information Resource, the first online database system that could be accessed by telephone line.

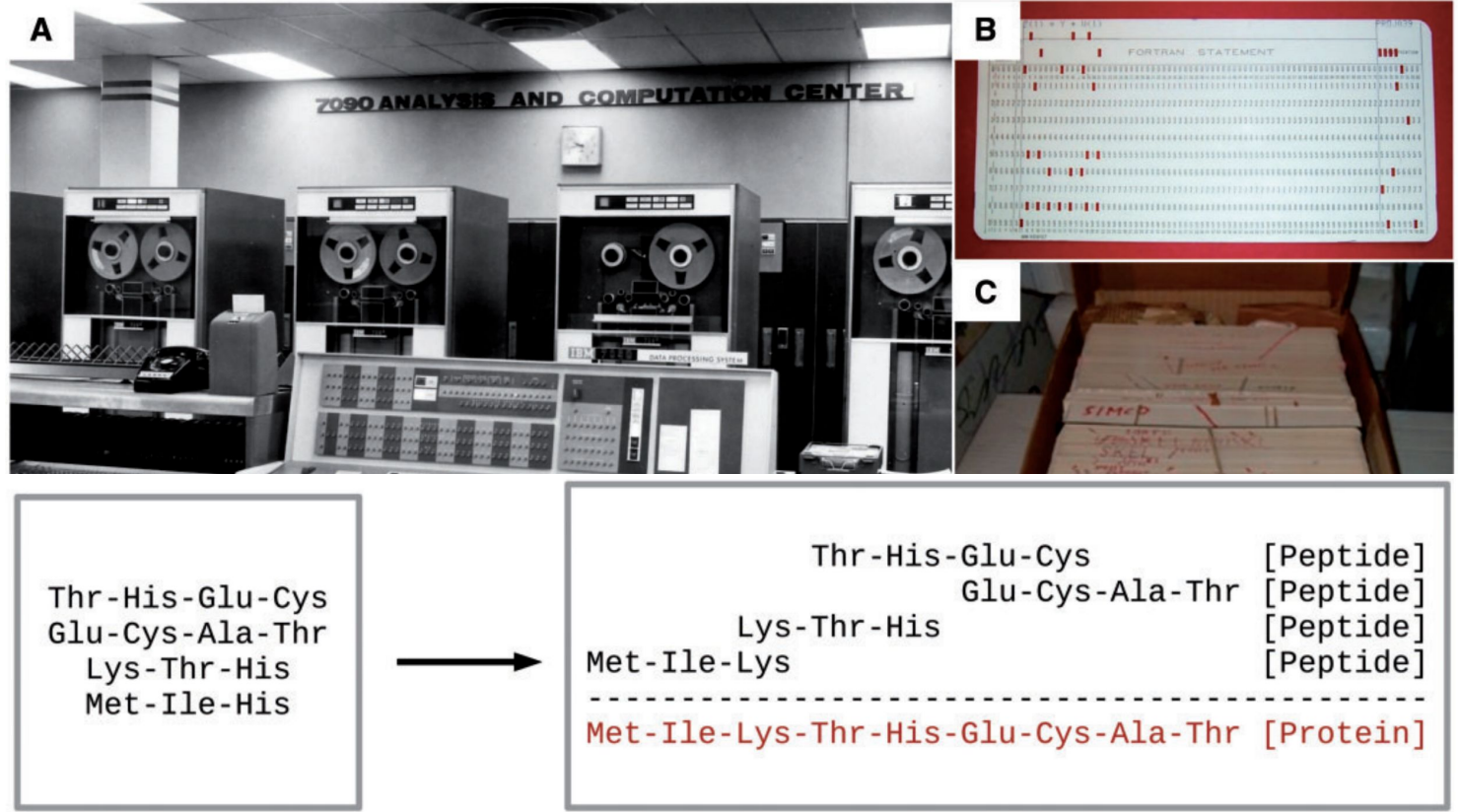
Origins



Dr. Margaret
Dayhoff
1925 – 1983

COMPROTEIN

– the first
de novo sequence
assembler



Origins

The IAS (von Neumann) machine built in 1951:

Binary computer

Memory: 1024 words (5.1 KB / 4 working).

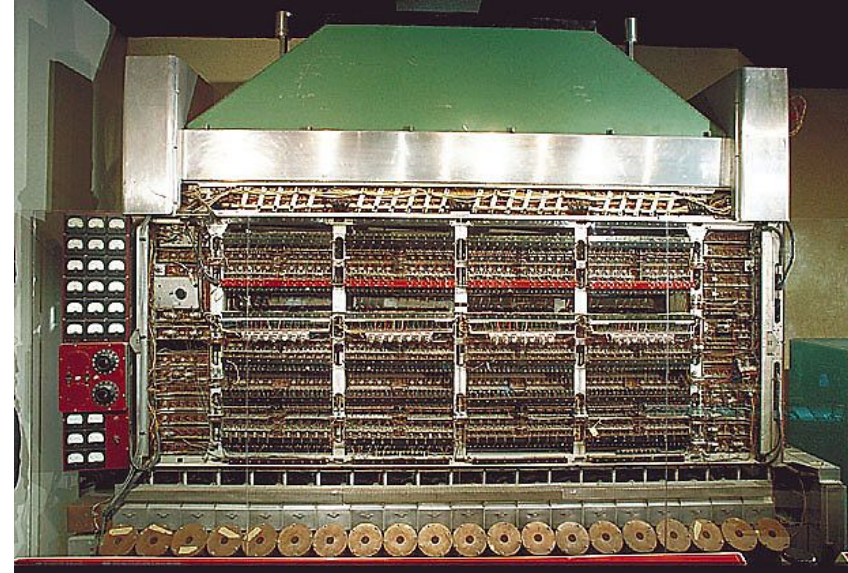
Addition time: 62 μ sec; multiplication time: 713 μ sec.



Dr. Nils
Barricelli

One of the first computational biologists

- **1953:** Simulated the evolution of populations of artificial organisms.
 - Each organism: genome consisting of a string of numbers.
 - Random mutations & sexual exchange of genes caused populations to evolve.



Past → Present

Data analysis **service**
providers



Leaders of cutting-edge
research programs

	Past	Current
Role in research	Supportive	Driver of research
A feeling for the biology	Computer science-centered	Biology- and computer science-centered
Environment	Isolated	Integrated
Data generation	Constrained	Resourceful
Data exploration	Largely limited to hypothesis testing	Both exploratory and hypothesis testing

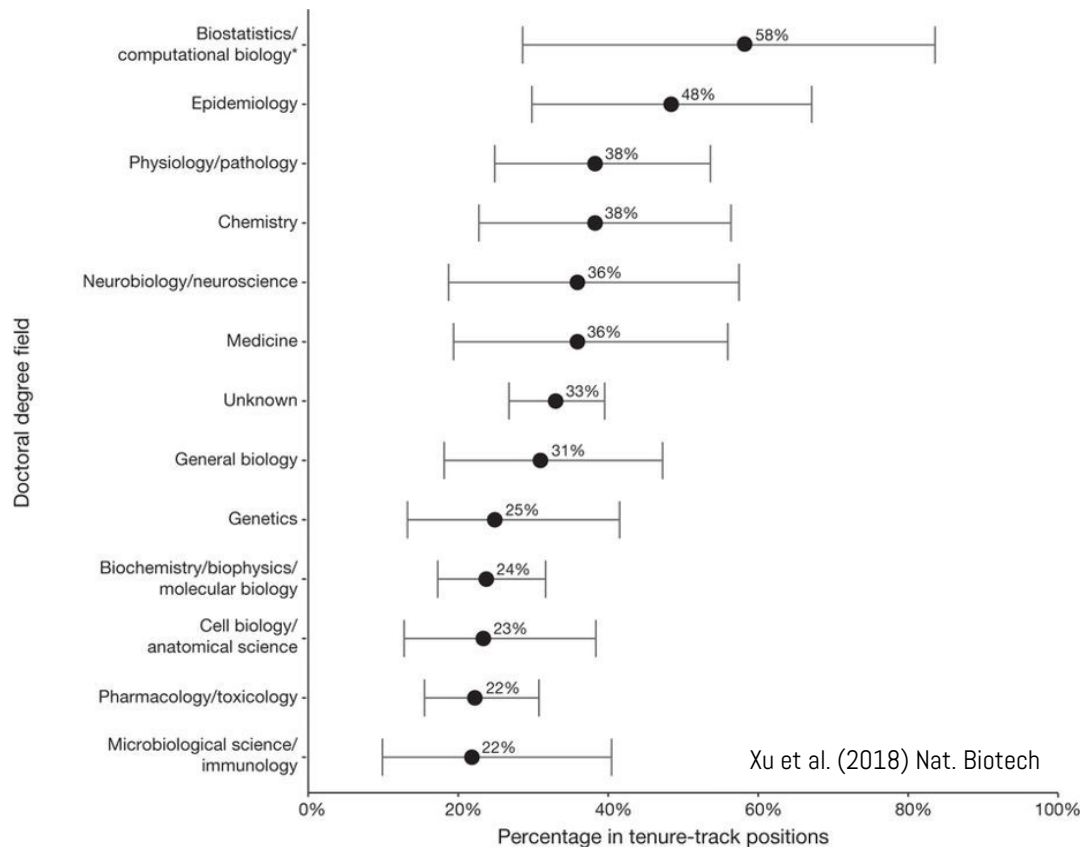
“Computational thinking and techniques are so central to the quest of understanding life that today **all biology is computational biology.**”

“The **next modern synthesis in biology** will be **driven by mathematical, statistical, and computational methods** being absorbed into mainstream biological training, turning biology into a quantitative science.”

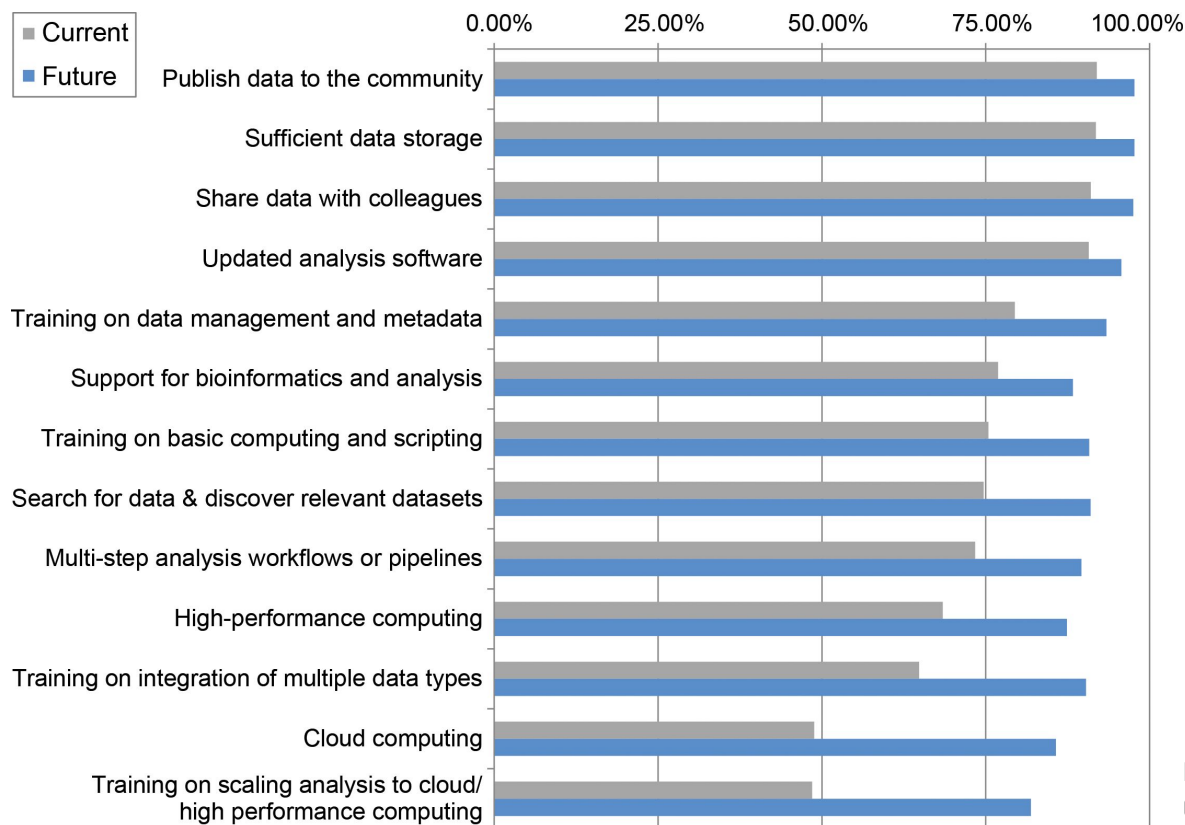
A melting pot of disciplines

1. Sequence alignment and pattern finding
 2. Genome assembly and annotation
 3. Molecular evolution and comparative genomics
 4. Quantitative genetics
 5. Regulatory genomics
 6. Functional genomics
 7. Single cell genomics
 8. Protein structure and dynamics
 9. Large-scale biological networks
 10. Modeling signaling, regulatory pathways
 11. Whole-cell models, artificial life, & digital evolution
- Dynamic programming
 - de Bruijn graphs, Hidden Markov Models
 - Tree construction, Suffix trees
 - Statistical inference, Multiple testing
 - Expectation maximization, Gibbs sampling
 - Clustering, Differential & enrichment analyses
 - Dimensionality reduction, Machine learning
 - Maximum entropy modeling, Atomic simulation
 - Graph theory, Label propagation
 - Dynamical simulation, State space, Bifurcations
 - Linear programming, Agent-based modeling

Opportunities & Unmet needs

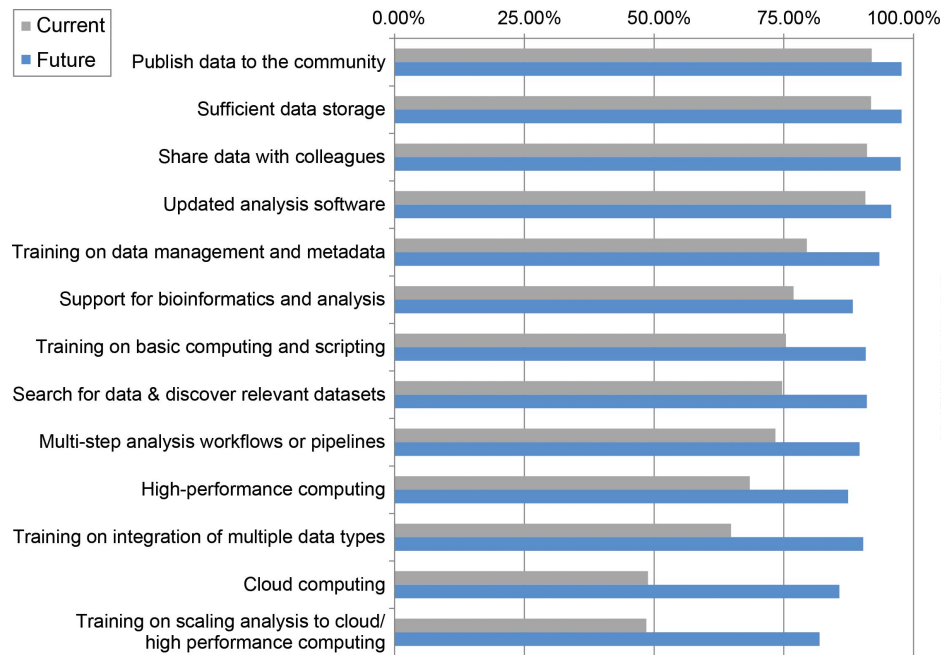


Opportunities & Unmet needs

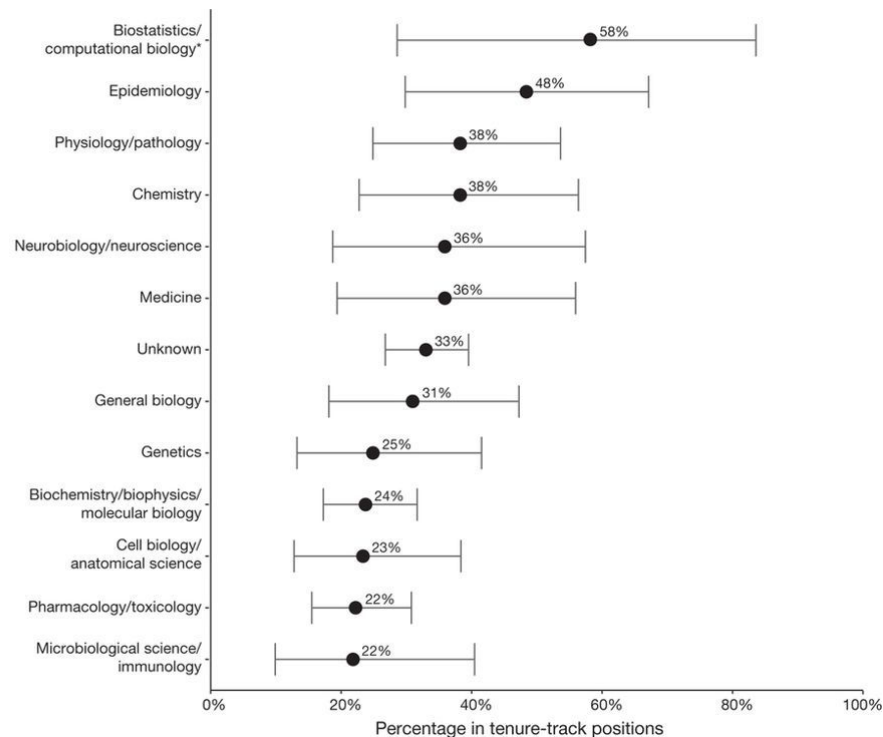


Barone et al. (2017) PLoS
Comp. Biol.

Opportunities & Unmet needs



Doctoral degree field



Community | Journals

- Bioinformatics
- bioRxiv Bioinformatics
- bioRxiv Genomics
- BMC Bioinformatics
- BMC Genomics
- Briefings in Bioinformatics
- BioData Mining
- Cell Systems
- Cell Patterns
- Database
- Genome Biology
- Genome Research
- IEEE/ACM CompBio & Bioinfo
- Molecular Systems Biology
- Nature Methods
- Nucleic Acids Research
- PLoS Computational Biology

Data types and repositories – some examples

Genomes & proteomes

all encompassing

Ensemble

comparative genomics

COGs | InParanoid | OrthoMCL

ref. gene/transcript sequences
& annotations

RefSeq | Entrez | GENCODE

sequences variation

1000 Genomes | dbSNP

everything protein

UniProt | InterPro | SCOP | CATH |
PDB

Functional annotations & relationships

biol. processes, mol. functions,
cellular components

Gene Ontology

pathways

Reactome, KEGG, WikiPathways

networks

BioGRID, TRANSFAC, STRING

Phenotype-, Disease-association

OMIM | GWAS Catalog | ClinVar |
COSMIC

Genome-Phenome

dbGaP | UK Biobank

Functional/regulatory genomics

data sets

NCBI GEO | EBI ArrayExpress

raw reads

NCBI SRA | EBI ENA

consortia

ENCODE | Roadmap | GTEx | TCGA

curated public data

Dryad | Repositive | Expression Atlas

Model organism databases

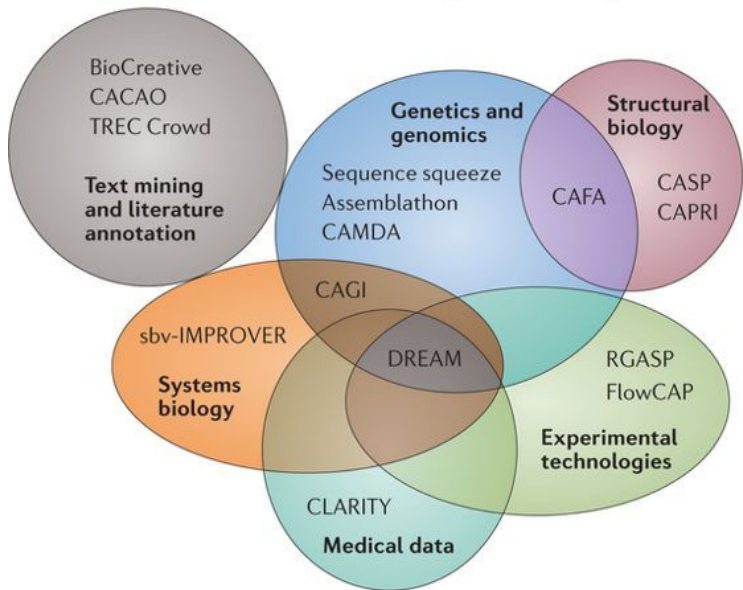
MGI | RGD | TAIR | FlyBase | WormBase
| ZFin | SGD

Some International Conferences

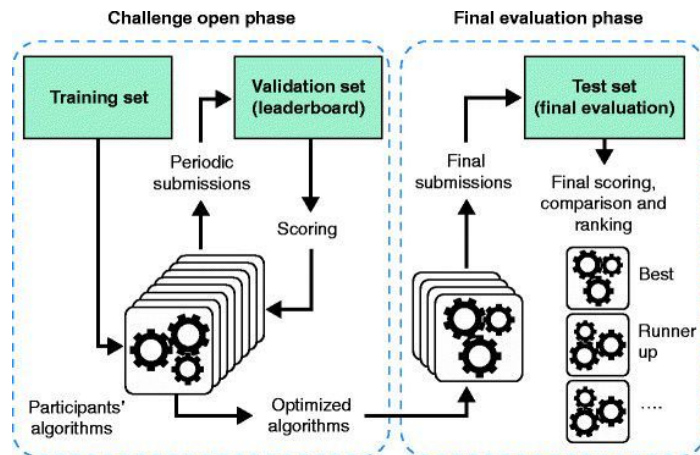
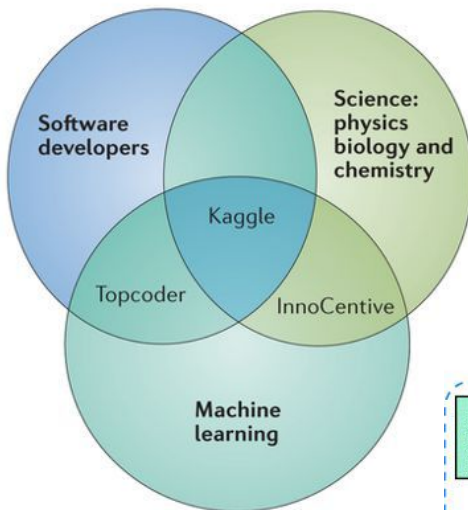
- Intelligent Systems for Molecular Biology
- Research in Computational Molecular Biology
- ACM Conference on Bioinformatics, Computational Biology, & Health Informatics
- European Conference on Computational Biology
- Cold Spring Harbor Laboratories Meetings
(Network Biology, Genome Informatics)

Community | Open Challenges

Researcher-driven domain-specific Challenges



Intermediary commercial Challenge platforms



The field has dramatically shifted in thinking on how to publish code

- Code used in research should be made available for research use free of charge.
- This is not just code for downloading & using. Original code must be made publicly available for others to use, review, and edit.
- Most common way to share code: GitHub.

Open Science | Preprints

Scientific publishing has been upended by preprints

- Rapid publication of new science + free access (e.g. bioRxiv).
- Major source of cutting-edge research.
- Can have multiple (progressively better) versions of each manuscript.
- Preprints have NOT been peer-reviewed for quality and soundness of science.
So, read/use with caution.

Types of bioinformatics & computational biology research studies

- New analytical/computational method
- Improvement of an existing method
- Evaluation of existing methods
- Development of (re-)usable software, web-service, or database
- New insights w/ new/existing methods

Reading papers | Learning to solve new problems

What can you learn from a paper?

- Learn how to frame a problem.
- Choose the methods/tools.
- Set up an analysis workflow.
- Establish groundwork.
- Generate a series of supportive results towards answering the central question.

How to read research papers?

- Make reading papers a habit.
- Critically analyze what you read/hear.
- Use a reference manager.
- Create and maintain a single source of knowledge.
- Contextualize what you read.

Reading papers

Title & Abstract

1. Use **Title, Abstract, & Figures** to select a paper. Read them again last!

Introduction

2. Read the **Introduction**:

Data & Methods

- a. Identify *the* question. What is the big challenge the authors are trying to solve?
- b. What are the then current approaches for solving that problem? What are their limitations that, according to the authors, need to be addressed?
- c. What are the *specific* questions this paper is setting out to answer?

Results

Discussion

References

Reading papers

Title & Abstract

3. Read **Data & Methods**: [Be critical!]

Introduction

- a. For each specific Q, note data (type & source) & method (algorithms/techniques, software, & approach).
 - i. ALWAYS read the **Supplementary Materials**. These days much of the good stuff is in here!
- b. Are the data & methods describes sufficient to answer the Qs raised in the Intro?
- c. Make detailed notes on: 1) what's unclear, 2) what you might do differently.

Data & Methods

Results

Discussion

References

Reading papers

Title & Abstract

Introduction

Data & Methods

Results

Discussion

References

4. Read the **Results**: [Be critical!]

- a. Go figure-by-figure, panel-by-panel. Based on your reading of Data & Methods, is there enough information to know/reproduce that analysis?
- b. Try to interpret each figure/panel, then read the figure legend and the part of the results that explains it. [**Supplementary Figures/Tables** abound!]
 - i. Do your interpretations match that of the authors'?
 - ii. Are the results answering the specific Qs?
- c. Make detailed notes on: 1) what's unclear, 2) what you might do differently.

Reading papers

Title & Abstract

5. Read the **Discussion & Conclusion, Title, & Abstract**:

- a. Step back to think about contributions, limitations, open Qs, & next steps.

Introduction

Data & Methods

6. Read what other researchers (**papers that cite this paper**) say about this work.

Results

Reading a paper can be *overwhelming*

Discussion

- Read the paper in phases & more than once.
- It is perfectly fine if things are not clear on first pass. Happens to everyone.
- Understanding will *a/ways* improve & the big picture will emerge with re-reading.

References

Picking up a new method or software

Read software/methods papers

- Read the Introduction & Discussion.
- Modern papers also have **graphical schematics of their methods/algorithms**.
- Use Google Scholar to find **recent application papers that use** the software/method & read those.
- Search and read **blogs** and watch YouTube **videos**.
- Together, these will not only help you understand the methods but also key **assumptions and parameters** that you need to think about for your project.

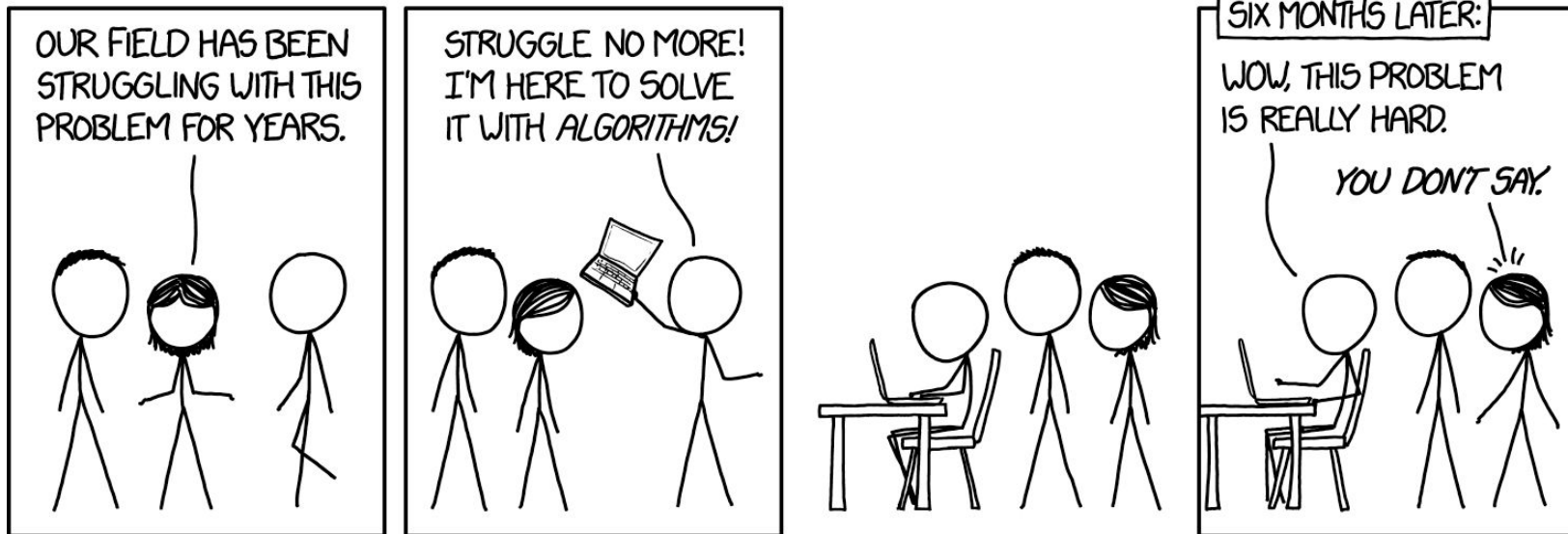
Picking up a new method or software

Explore the actual software/code

- Read the **documentation**: Overview and parts of it that correspond to the assumptions & parameters relevant to your project.
- Look into the exact data **input & output formats**.
- After installation, **replicate an example** run exactly as-is from the documentation/website or from an independent online tutorial.
 - If neither is available, email the (first & corresponding) authors asking for example data & detailed instructions on how to run their code.

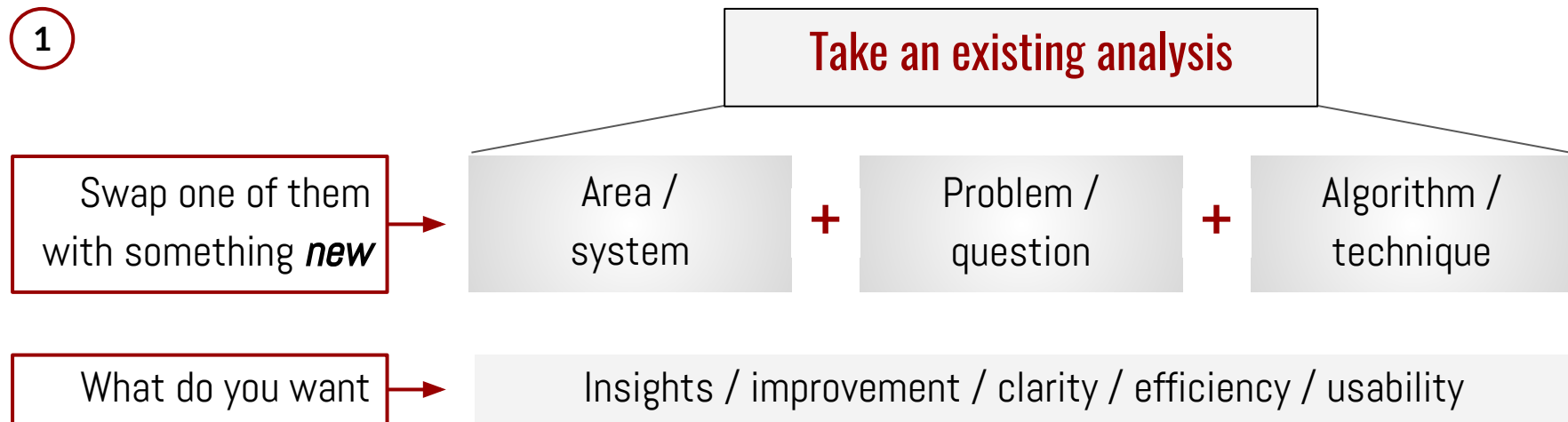
Choosing a good problem

xkcd.com/1831

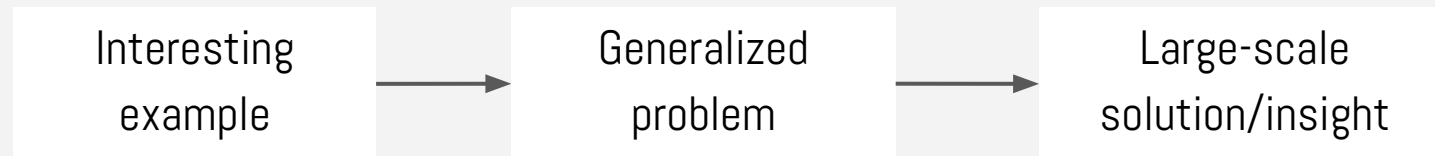


Choosing a good problem

1

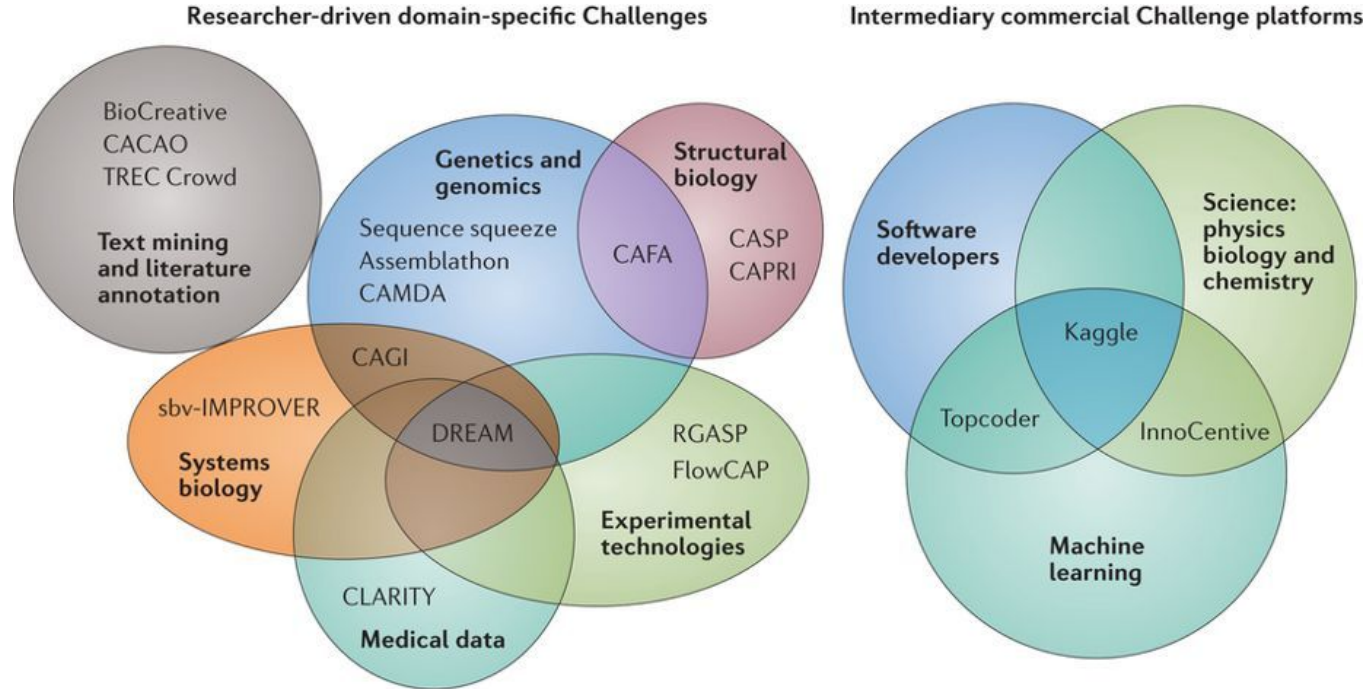


2



Choosing a good problem

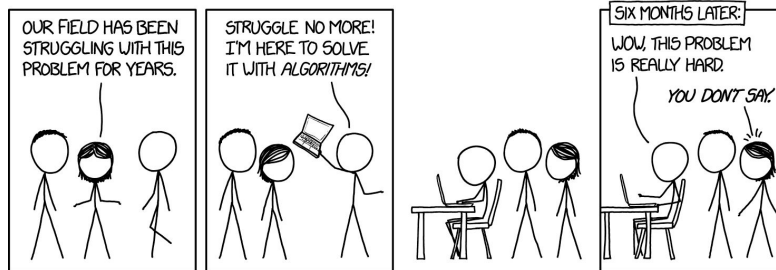
Open challenges are a great way to find good well-defined problems to work on.



Saez-Rodriguez et al. (2016) Nat. Rev. Genet.; Boutros et al. (2014) Genome Biol.

Choosing a good problem

xkcd.com/1831



Explore and prototype early to fail fast and learn

- Critical for determining if the problem is well-defined & tractable.
- Explore and understand your data using plots and summaries.
- Perform preliminary analysis with simple baselines, sample datasets, and toy examples.
- Don't speculate or make assumptions. Instead, implement something and check them.
- The value lies not in the code/plots you produce, but in the lessons you learn.

Programming languages & software ecosystems

Language, IDE, Notebook

Pre-built external packages

Scientific computing

Data wrangling, visualization

- R | RStudio | R Notebook
- CRAN, Bioconductor
- In-built + Hundreds of packages
- Tidyverse

- Python | Rodeo | Jupyter
- PyPI, Biopython
- NumPy, SciPy + Hundreds of packages
- Pandas, Seaborn

There are hundreds of software packages for bioinformatics & computational biology written in various languages (C, C++, R, & Python) that can be run from the command-line.

- Linux command-line
 - Navigating the file system
 - Running code
 - Manipulating data
 - Writing shell scripts

Resources @ MSU

Institute for Cyber-Enabled Research

- High-Performance Computing Cluster: wiki.hpcc.msu.edu
- Training resources: www.icer.msu.edu/education-events/training-resources
- Seminars and workshops: www.icer.msu.edu/upcoming-workshops
- Regular open office hours.

R-Ladies East Lansing


- >650 members from the larger MSU community
- <https://rladies-eastlansing.github.io/>

Getting help

- **Linux** | rik.smith-unna.com/command_line_bootcamp, commandline.guide, & swcarpentry.github.io/shell-novice
- **Python** | Introduction: learnpythonthehardway.org/book & developers.google.com/edu/python | Data analysis: jakevdp.github.io/WhirlwindTourOfPython | Visualization: www.r-graph-gallery.com
- **R** | Introduction: swcarpentry.github.io/r-novice-inflammation & swirlstats.com ('R Programming' & 'Data Analysis') | Data analysis: r4ds.had.co.nz | Visualization: python-graph-gallery.com
- **Git & GitHub** | swcarpentry.github.io/git-novice/, speakerdeck.com/alicebartlett/git-for-humans, & rogerdudler.github.io/git-guide/
- **Probability and Statistics** | Nature Collection (Statistics for Biologists | Practical Guides | Points of Significance): www.nature.com/collections/qghhqm
- **Genetics and Molecular Biology** | learn.genetics.utah.edu/ & www.genomicseducation.hee.nhs.uk



Getting help

 ... so many excellent blog posts!

 **stackoverflow**

 **Biostars**
— BIOINFORMATICS EXPLAINED —

Several video lessons/courses on YouTube

No shame!

StackOverflow Importer

Do you ever feel like all you're doing is copy/pasting from Stack Overflow?

Let's take it one step further.

from stackoverflow import quick_sort will go through the search results of [python] quick sort looking for the largest code block that doesn't syntax error in the highest voted answer from the highest voted question and return it as a module. If that answer doesn't have any valid python code, it checks the next highest voted answer for code blocks.

```
>>> from stackoverflow import quick_sort, split_into_chunks

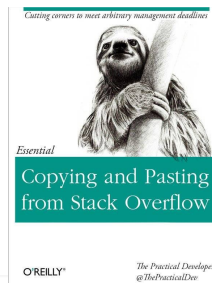
>>> print(quick_sort.sort([1, 3, 2, 5, 4]))
[1, 2, 3, 4, 5]

>>> print(list(split_into_chunks.chunk("very good chunk func")))
['very ', 'good ', 'chunk', ' func']

>>> print("I wonder who made split_into_chunks", split_into_chunks.__author__)
I wonder who made split_into_chunks https://stackoverflow.com/a/35107113

>>> print("but what's the license? Can I really use this?", quick_sort.__license__)
but what's the license? Can I really use this? CC BY-SA 3.0

>>> assert("nice, attribution!")
```



Getting help – Additional reading

- Checkout all the references cited in the slides.
- So you want to be a computational biologist? <https://www.nature.com/articles/nbt.2740>
- What Is the Key Best Practice for Collaborating with a Computational Biologist?
[https://www.cell.com/cell-systems/fulltext/S2405-4712\(16\)30223-X](https://www.cell.com/cell-systems/fulltext/S2405-4712(16)30223-X)
- A Quick Guide for Developing Effective Bioinformatics Programming Skills
<http://dx.plos.org/10.1371/journal.pcbi.1000589>
- Ten Simple Rules for Effective Computational Research
<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1003506>
- Good Enough Practices in Scientific Computing <http://arxiv.org/abs/1609.00037>
- Ten simple rules for documenting scientific software
<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1006561>

Getting help – Additional reading

- Fantastic resources on Reproducible code, Data management, Getting published, and Peer review
<http://www.britishecologicalsociety.org/publications/guides-to/>
- A Quick Guide to Organizing Computational Biology Projects
<https://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1000424>
- A Quick Introduction to Version Control with Git and GitHub
<http://dx.plos.org/10.1371/journal.pcbi.1004668>
- Ten Simple Rules for Taking Advantage of Git and GitHub
<http://journals.plos.org/ploscompbiol/article?id=10.1371/journal.pcbi.1004947>

What's next

- Those who have not met with me to discuss your research profile should do so today.
- I will post a short review video for next week's lecture by the end of the day.

Week 2: Sequence alignment & search

- Sequence alignment problem
- Dynamic programming
- Global alignment
 - Needleman-Wunsch algorithm
- Local alignment
 - Smith-Waterman algorithm
- Substitution matrix
 - Construction & properties
- BLAST
 - Statistics of similarity search