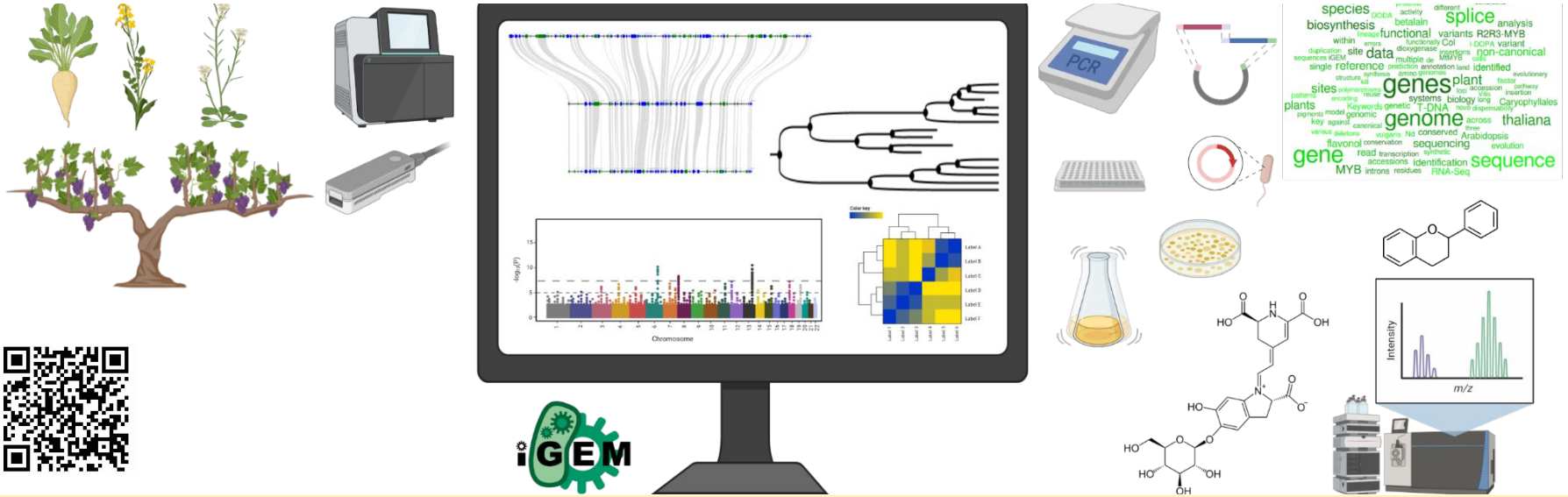
Technische Universität Braunschweig



Introduction to (plant) bioinformatics

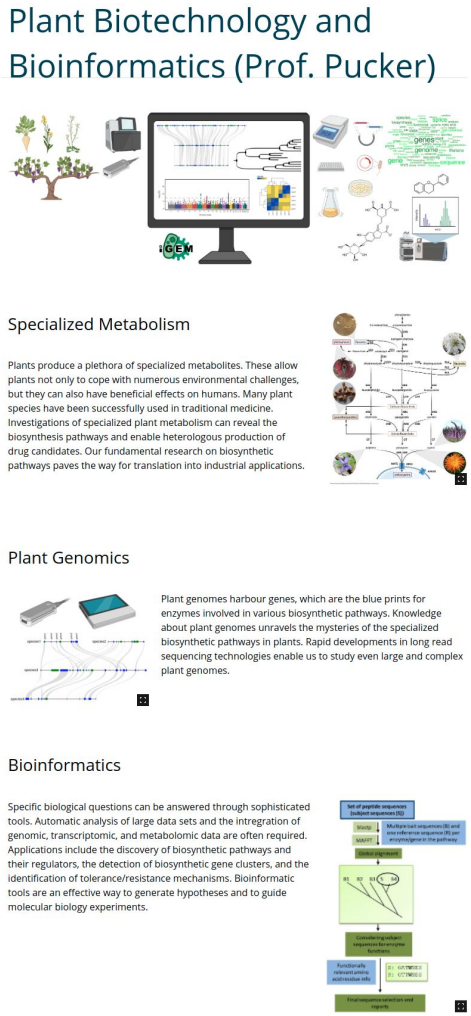
Prof. Dr. Boas Pucker (Plant Biotechnology and Bioinformatics)

Round of introduction

- Name?
- Study program?
- Semester?
- Previous experiences with biochemistry?
- Previous experiences with bioinformatics?
- Expectations?

Boas Pucker

- Biochemistry at HHU Düsseldorf
- (Systems) Biology at Bielefeld University
- Doctoral student (CeBiTec, Bielefeld University)
  - Genomics & Bioinformatics; synthetic biology (iGEM)
- Post doc (Ruhr-University Bochum)
- Post doc (Department of Plant Sciences, Cambridge, UK)
- Plant Biotechnology & Bioinformatics, TU Braunschweig (since 2021)
  - Specialized plant metabolites, applied bioinformatics



https://www.tu-braunschweig.de/en/ifp/pbb

Availability of slides

- All materials are freely available (CC BY) - after the lectures:
  - StudIP: Lecture: Grundlagen der Biochemie und Bioinformatik der Pflanzen (Bio-MB 09)
  - Skype: (link shared via email)
  - GitHub: https://github.com/bpucker/teaching

- Questions: Feel free to ask at any time

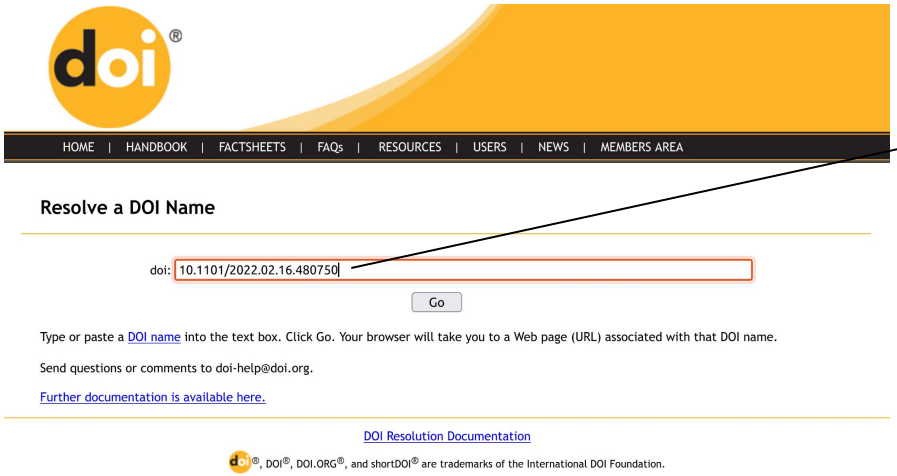
- Feedback, comments, or questions: b.pucker[a]tu-braunschweig.de

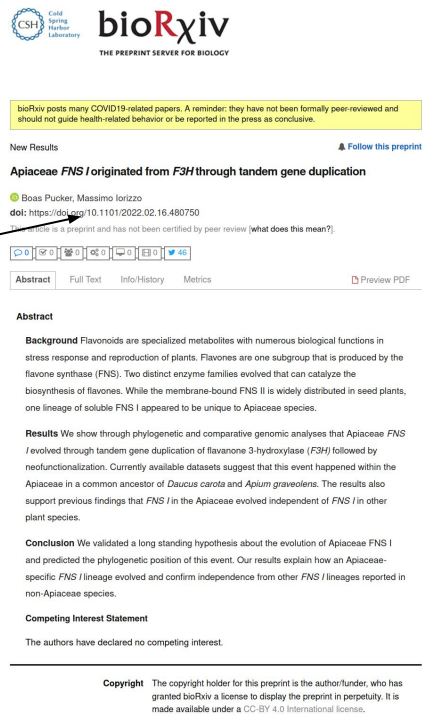
Overview

1) Introduction to bioinformatics (today)
2) Identification of biosynthesis pathways
3) Pathway databases
4) Gene expression & co-expression analyses
5) Phylogenetic analyses
6) Synteny & biosynthetic gene clusters
7) Metabolic flux & modeling
8) Repetition, questions & quiz

DOI - finding the literature

- Suggestions for detailed literature will be included on slides
- DOI = Digital Object Identifier
- Unique and short way to point to a publication
- How to resolve a DOI? https://dx.doi.org/

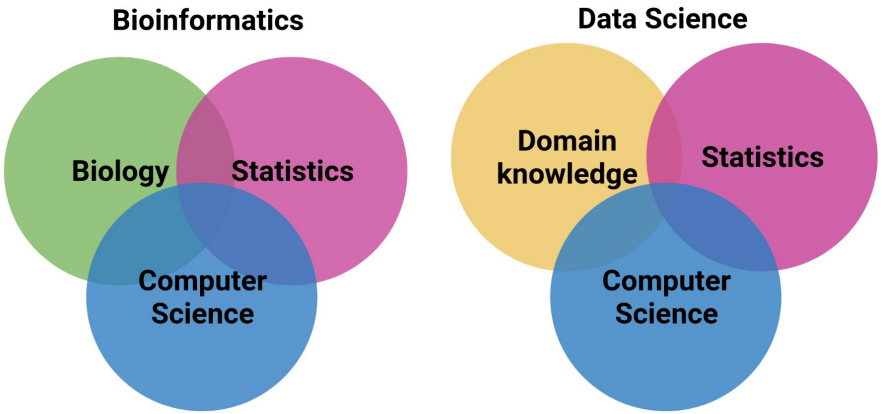




What is bioinformatics?

What is bioinformatics?

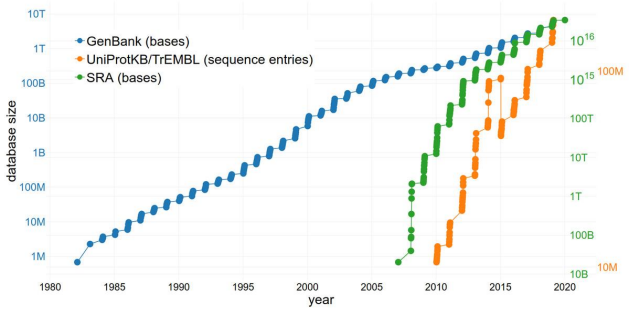
- Subdiscipline of biology, statistics, and computer science
- Acquisition, storage, analysis, and dissemination of biological data
- Bioinformatics is a biology-specific data science version

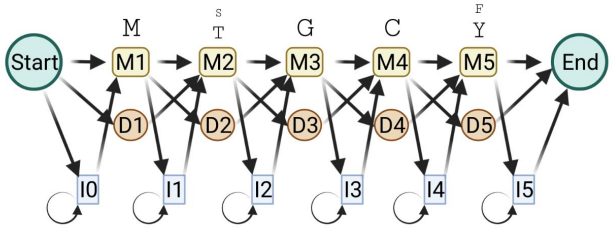


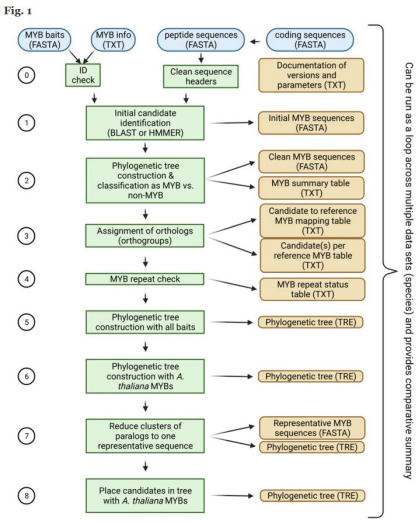
Why do we need bioinformatics?

Why do we need bioinformatics?

- Large data sets
- Complex models
- Automatic analyses

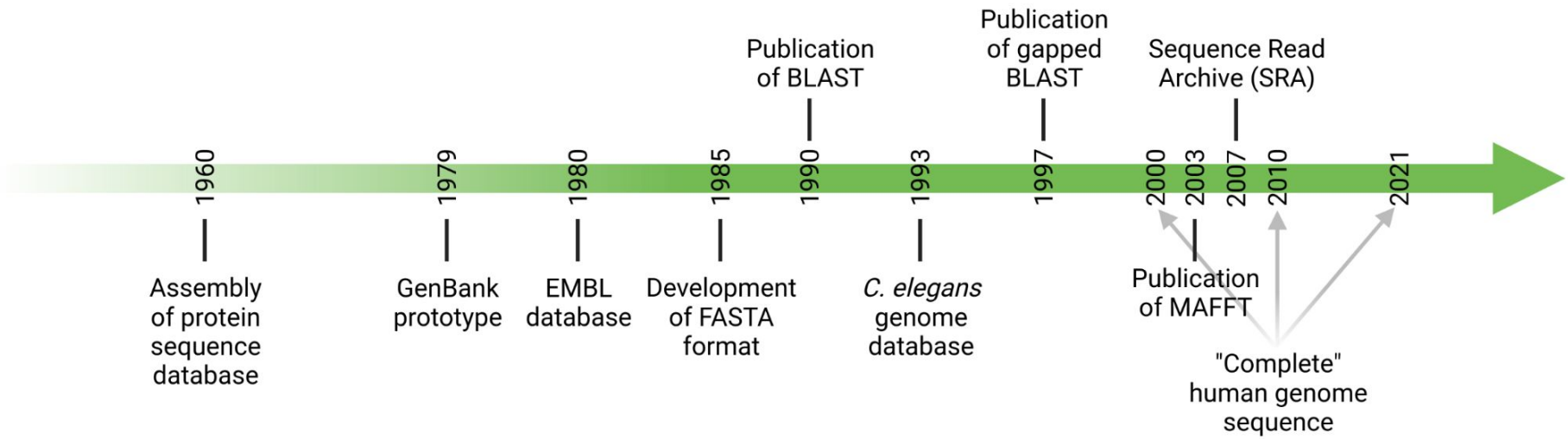




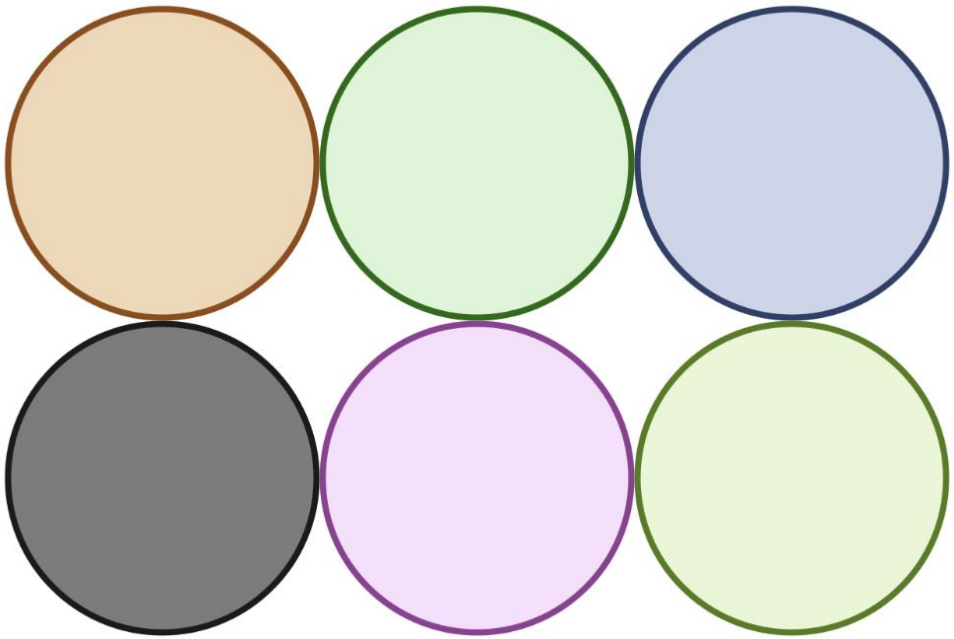


Sielemann & Hafner et al., 2020: 10.7717/peerj.9954
Pucker, 2022: 10.1186/s12864-022-08452-5

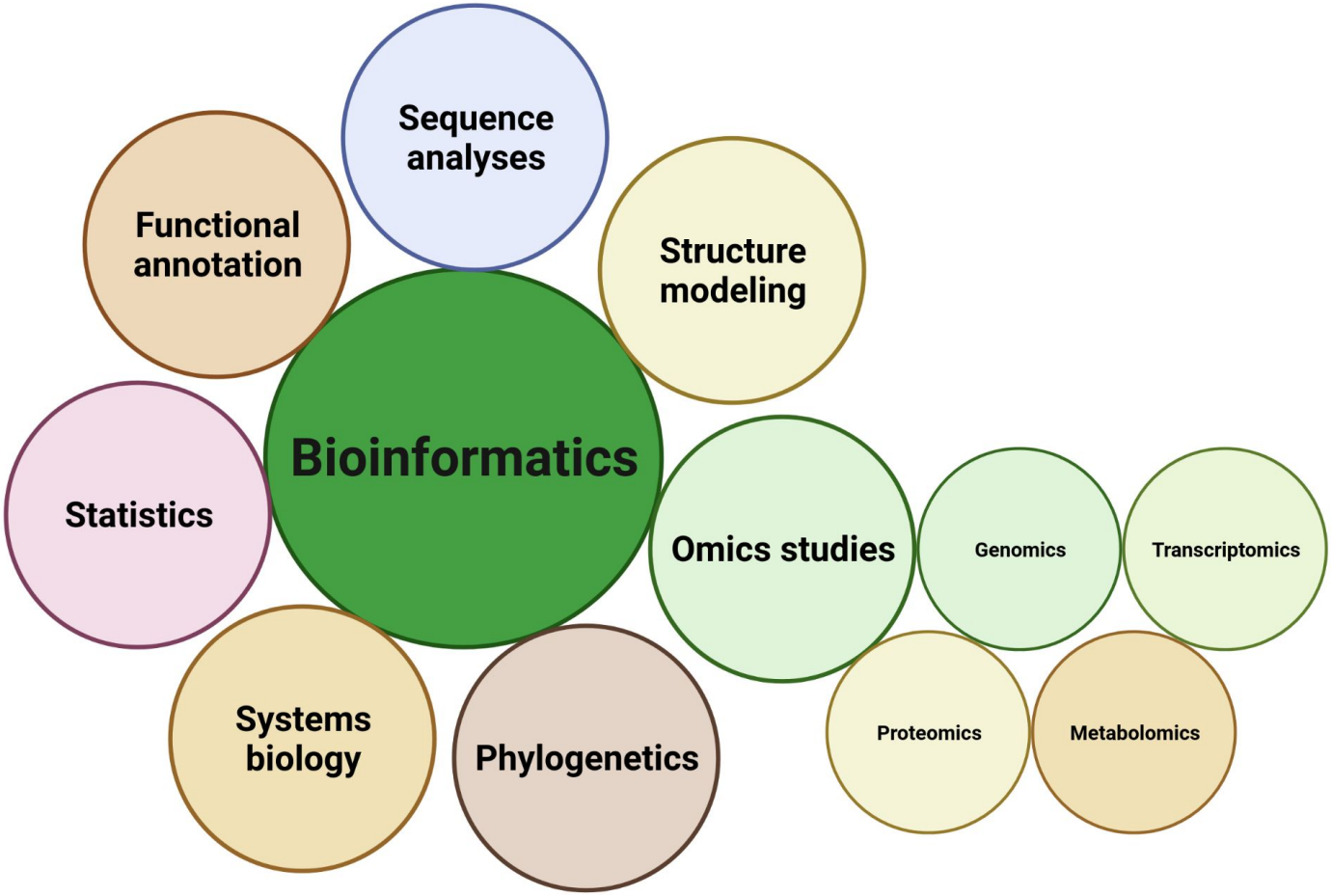
History of bioinformatics



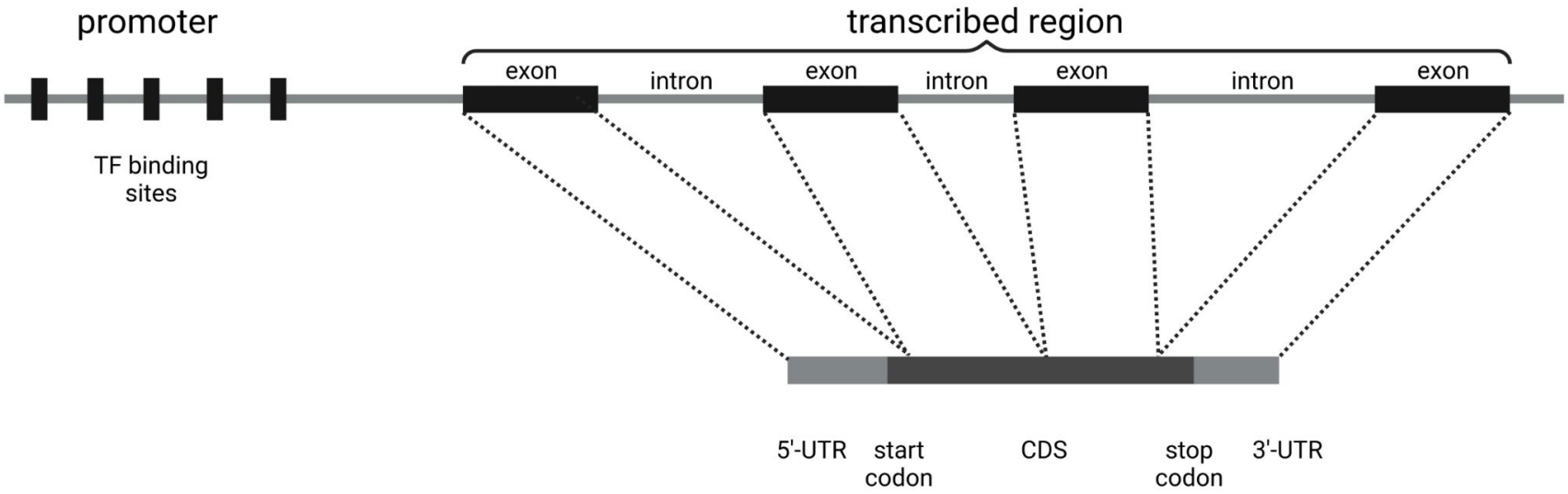
What are the different (sub)fields?



What are the different (sub)fields?



Sequence analysis



Sequence analysis II

- Comparison of sequences to find similarities
- Analysis of sequence composition
- Important methods:
  - BLAST = Basic Local Alignment Search Tool
  - MAFFT = Multiple Alignment using Fast Fourier Transform

chalcone synthase [Chenopodium quinoa]

Sequence ID: XP_021762431.1 Length: 392 Number of Matches: 1

Range 1: 2 to 389 GenPept Graphics ▼ Next Match ▲ Previous Match

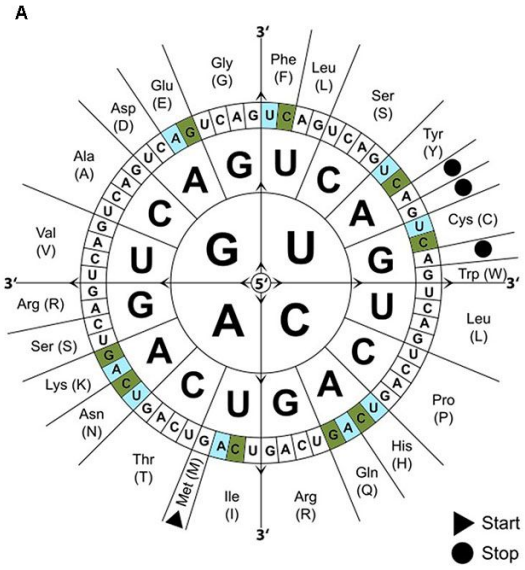
Score	Expect	Method	Identities	Positives	Gaps
666 bits(1719)	0.0	Compositional matrix adjust.	323/388(83%)	362/388(93%)	0/388(0%)

```
Query  3    TPSVQEIRDAQRSNGPATILAIGTANPANEMYQAEYPDFYFRVTKSEHMKELKQKFKRMC  62
            T S++EIR AQR++GPATILAIGTA P N +YQ+++PD+YFRVTKSEHM ELK+KFKRMC
Sbjct  2    TSSLEEIRKAQRADGPATILAIGTATPPNCVYQSDFPDYYFRVTKSEHMTELKEKFKRMC  61

Query  63   DNSMIKKRYMHVTEELLEENPHLCDFNASSLDTRQDILATEVPKLGKEAAVKAIKEWGQP  122
            D SMI KRYMH+TEE L+ENP++C +  SSLDTRQDI+ +EVP+LGKEAAVKAIKEWGQP
Sbjct  62   DKSMILKRYMHLTEEFLKENPNMCTYMGSSLDTRQDIVVSEVPRLGKEAAVKAIKEWGQP  121

Query  123  RSKITHVIFCTTSGVDMPGADYQLTKLLGLRPSVKRFMLYQQGCYAGGTVLRLAKDIAEN  182
            +SKITHVI CTTSGVDMPGADYQLTKLLGLRPSV+RFMLYQQGC+AGGTVLRLAKD+AEN
Sbjct  122  KSKITHVIMCTTSGVDMPGADYQLTKLLGLRPSVRRFMLYQQGCFAGGTVLRLAKDLAEN  181

Query  183  NRGARVLVVCAEITIICFRGPTQIHLDSMVGQALFgdgagavivgadPDESIERPIFQLV  242
            NRGARVLVVC+EIT ICFRGPT+ HLDSMVGQALFGDGAGA+IVGADPDESIERP+F++V
Sbjct  182  NRGARVLVVCSEITAICFRGPTETHLDSMVGQALFGDGAGALIVGADPDESIERPLFKMV  241

Query  243  WAAQTILPGSEGAIDGHLREVGLTFHLLKDVPGLISKNIEKALVEAFQPIGIDDWNSIFW  302
            WAAQTILP SEGAIDGHLREVGLTFHLLKDVPGLISKNI KAL EAF P+GI DWNSIFW
Sbjct  242  WAAQTILPDSEGAIDGHLREVGLTFHLLKDVPGLISKNINKALEEAFTPLGISDWNSIFW  301

Query  303  VAHPGGRAILDDVESKLGLKEDKLKTTRHVLSEYGNMSSACVLFILDEMRKRAMKEGMAT  362
            +AHPGG AILD VE+KLGLKE+KL  TR+VLSE+GNMSSACVLFILDEMRK++MKEG AT
Sbjct  302  IAHPGGPAILDQVEAKLGLKEEKLTATRNVLSEFGNMSSACVLFILDEMRKKSMKEGKAT  361

Query  363  TGEGLEWGVLFGFGPGLTVETVMLHSVP  390
            TG+GL+WGVLFGFGPGLTVETV+LHSVP
Sbjct  362  TGDGLDWGVLFGFGPGLTVETVVLHSVP  389
```

```
CLUSTAL format alignment by MAFFT FFT-NS-i (v7.487)


BvCHS          M---ATPSVQEIRDAQRSNGPATILAIGTANPANEMYQAEYPDFYFRVTKSEHMKELKQK
AtCHS          MVMAGASSLDEIRQAQRADGPAGILAIGTANPENHVLQAEYPDYYFRITNSEHMTDLKEK
               *   .:.*::***:***::*** ********* *.: ******:***:*:****.:**:*


BvCHS          FKRMCDNSMIKKRYMHVTEELLEENPHLCDFNASSLDTRQDILATEVPKLGKEAAVKAIK
AtCHS          FKRMCDKSTIRKRHMHLTEEFLKENPHMCAYMAPSLDTRQDIVVVEVPKLGKEAAVKAIK
               ******:* *:**:**:***:*:****:* : *.********:.:***************


BvCHS          EWGQPRSKITHVIFCTTSGVDMPGADYQLTKLLGLRPSVKRFMLYQQGCYAGGTVLRLAK
AtCHS          EWGQPKSKITHVVFCTTSGVDMPGADYQLTKLLGLRPSVKRLMMYQQGCFAGGTVLRIAK
               *****:******:*****************************:*:*****:*******:**


BvCHS          DIAENNRGARVLVVCAEITIICFRGPTQIHLDSMVGQALFGDGAGAVIVGADPDESI-ER
AtCHS          DLAENNRGARVLVVCSEITAVTFRGPSDTHLDSLVGQALFSDGAAALIVGSDPDTSVGEK
               *:**************:*** : ****:: ****:******.***.*:***:*** *: *:


BvCHS          PIFQLVWAAQTILPGSEGAIDGHLREVGLTFHLLKDVPGLISKNIEKALVEAFQPIGIDD
AtCHS          PIFEMVSAAQTILPDSDGAIDGHLREVGLTFHLLKDVPGLISKNIVKSLDEAFKPLGISD
               ***::* *******.*:*************************** *:* ***:*:**.*


BvCHS          WNSIFWVAHPGGRAILDDVESKLGLKEDKLKTTRHVLSEYGNMSSACVLFILDEMRKRAM
AtCHS          WNSLFWIAHPGGPAILDQVEIKLGLKEEKMRATRHVLSEYGNMSSACVLFILDEMRRKSA
               ***:**:***** ****:** ******:*:::************************:::


BvCHS          KEGMATTGEGLEWGVLFGFGPGLTVETVMLHSVPTAN
AtCHS          KDGVATTGEGLEWGVLFGFGPGLTVETVVLHSVPL--
               *:*:************************:*****
```

Sequence analysis III

- Codons define amino acids to integrate
- Genetic code is redundant i.e. multiple codons for each amino acid
- Organisms can have a preference for certain codons for a given amino acid
- Optimization of sequences for heterologous expression

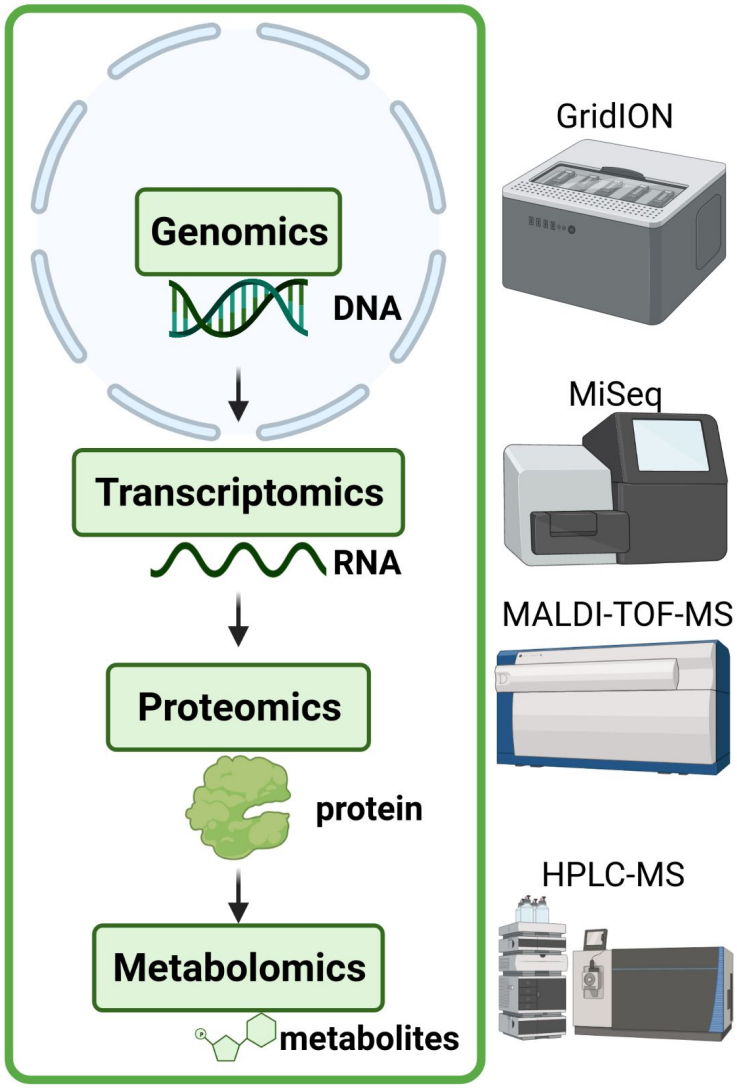


B

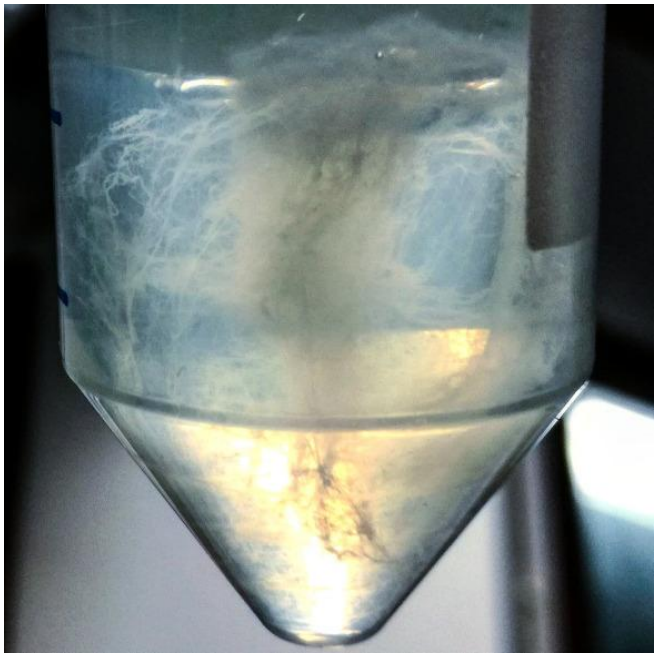
Amino acid	Codon	*P. patens*	*A. thaliana*	*S. cerevisiae*	*S. pombe*	*H. Sapiens*
Cys	UGC	+	+	-	+	+
	UGU	-	-	+	-	-
Glu	GAA	-	-	+	-	-
	GAG	+	+	-	+	+
Phe	UUC	+	+	+	+	+
	UUU	-	-	-	-	-
His	CAC	+	+	+	+	+
	CAU	-	-	-	-	-
Ile	AUA	-	-	/	-	-
	AUC	+	+	/	-	+
	AUU					
Lys	AAA	-	-	-	-	-
	AAG	+	+	+	+	+
Asn	AAC	+	+	+	+	+
	AAU	-	-	-	-	-
Gln	CAA	-	-	+	-	
	CAG	+	+	-	+	
Tyr	UAC	+	+	+	+	+
	UAU	-	-	-	-	-

Hiss et al., 2017: 10.3389/fpls.2017.01842

Omics levels



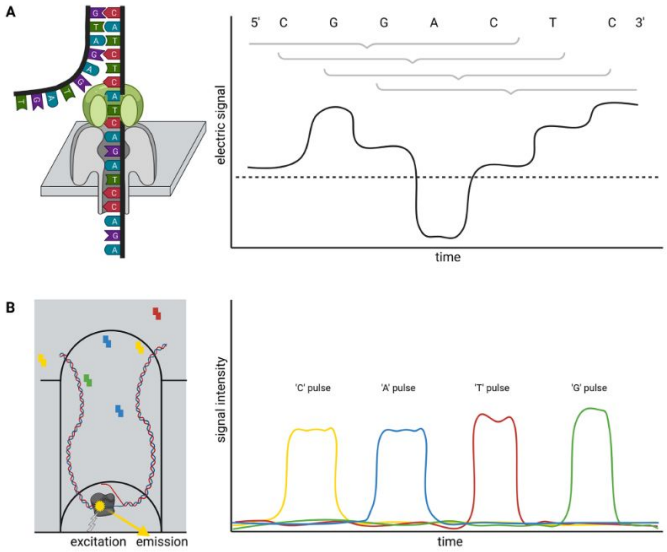
What is a genome?



Picture by Hanna Schilbert; Pucker, 2020: 10.17504/protocols.io.bcvyiw7w

Genomics

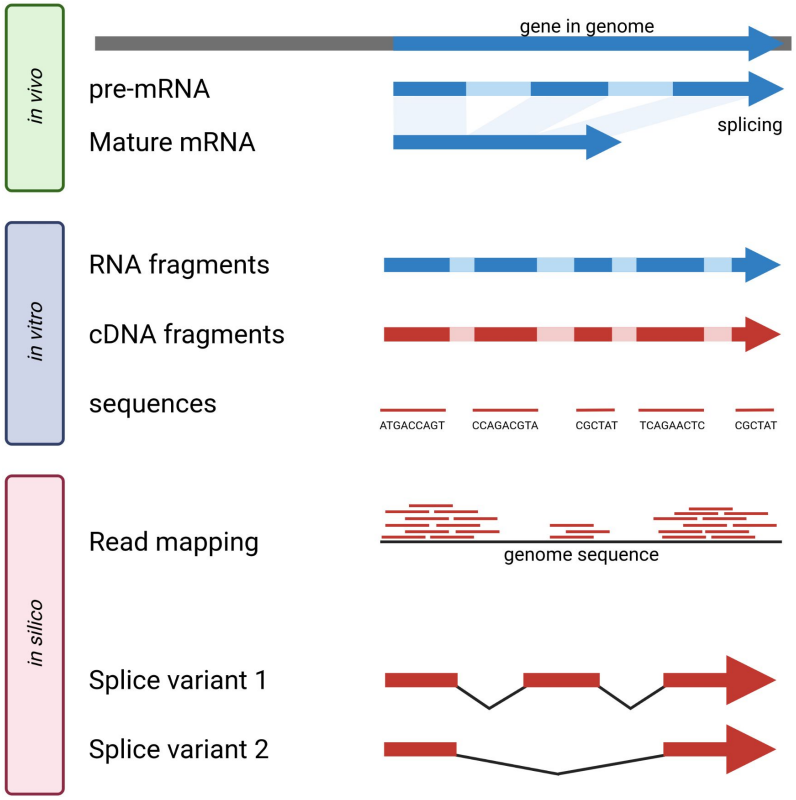
- Genome = sum all of genetic information in a cell
- Plants have DNA in nucleus (nucleome), chloroplasts (plastome), and mitochondria (chondromes)
- Sequencing plant genomes: Oxford Nanopore Technologies (ONT) and Pacific Biosciences (PacBio)
- Genome sequence assembly: HiCanu, FALCON, Shasta, …
- Comparison of genome sequences
- Handling of large data sets (>1TB) necessary for most projects



Pucker et al., 2022: 10.1017/qpb.2021.18

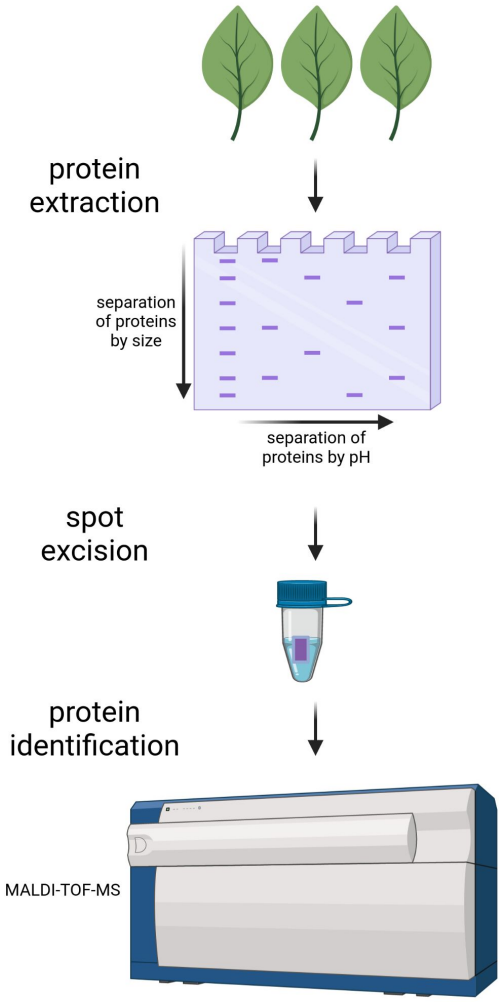
Transcriptomics

- Transcriptome = sum of transcripts in a cell/tissue under defined conditions and at a defined time point
- Systematic investigation of gene expression:
  - Microarrays: method of choice for many years
  - RNA-Seq: current method of choice to study gene expression systematically
- Comparison of different samples (tissues, conditions, genotypes, ….)



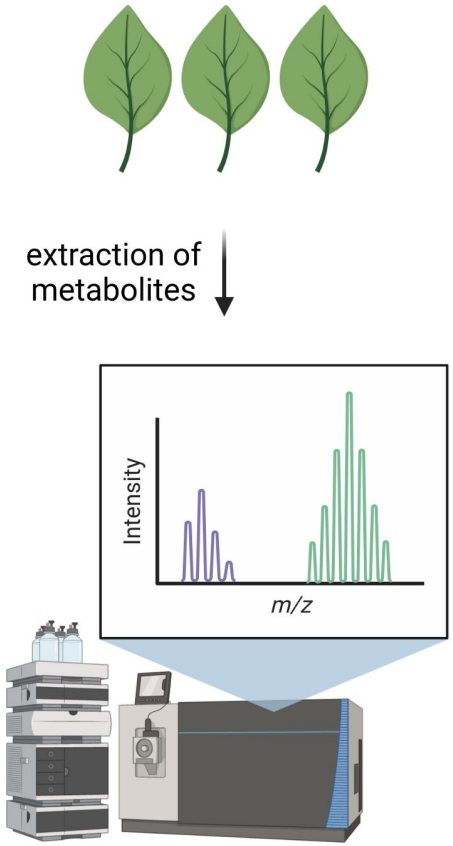
Proteomics

- Proteome = sum of all proteins in a cell/tissue under defined conditions at a defined time point
- Methods for systematic investigation:
  - High Performance Liquid Chromatography (HPLC)-Mass Spectrometry (MS)
  - Separation of proteins via 2D gels and investigation of spots by MS
- Heterogenous properties of proteins make systematic analysis challenging



Metabolomics

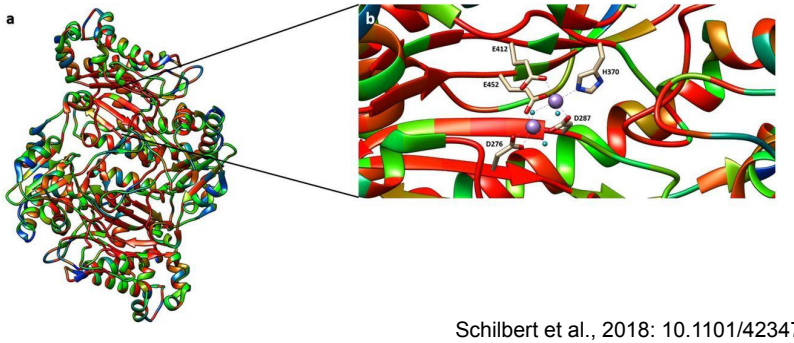
- Metabolom = sum all of metabolites in a cell/tissue under defined conditions at a defined time point
- Methods for systematic metabolite analysis:
  - HPLC-MS/MS
  - GC-MS/MS
- Chemical diversity of metabolites poses a challenge for the analysis
- Identification of metabolites remains a challenge



Structure modeling

- Modeling of RNA structures
  - Collection of tools: https://molbiol-tools.ca/RNA_analysis.htm
  - BiBiServ: https://bibiserv.cebitec.uni-bielefeld.de/rna
  - RNAfold: http://rna.tbi.univie.ac.at/cgi-bin/RNAWebSuite/RNAfold.cgi

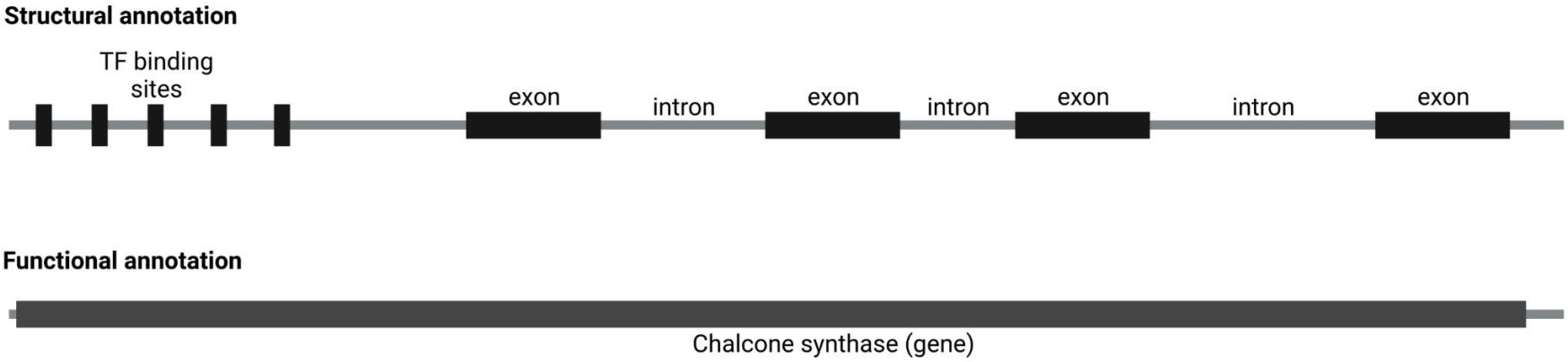
- Modeling of 3D protein structures
  - Collection of tools: https://molbiol-tools.ca/Protein_tertiary_structure.htm
  - PredictProtein: https://predictprotein.org/
  - AlphaFold2 (10.1021/acs.jcim.1c01114)



Schilbert et al., 2018: 10.1101/423475

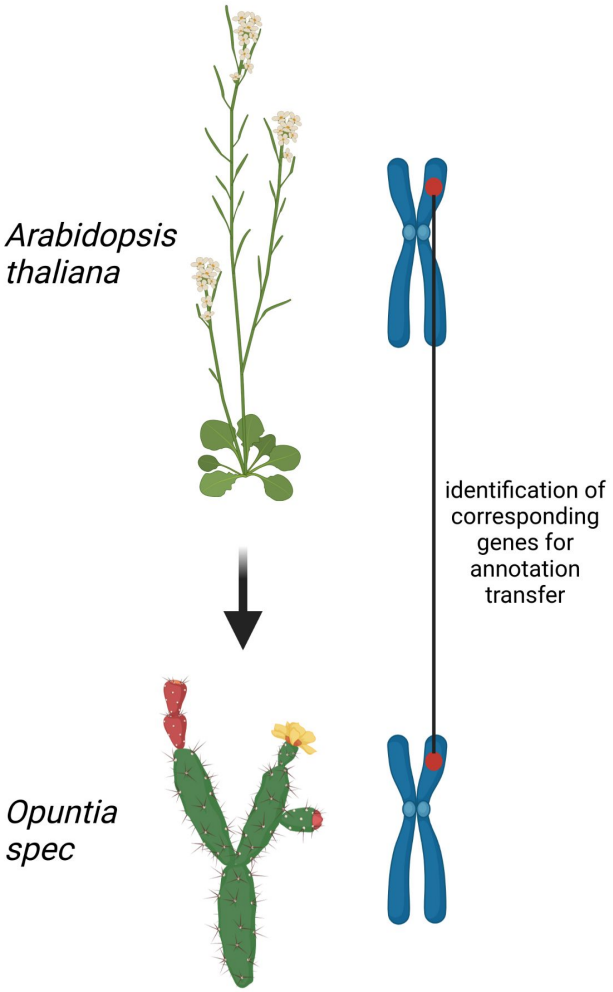
Structural vs. functional annotation

- Structural annotation = position of features (e.g. exons) in a genome sequence
- Functional annotation = biochemical function of a feature (e.g. gene)



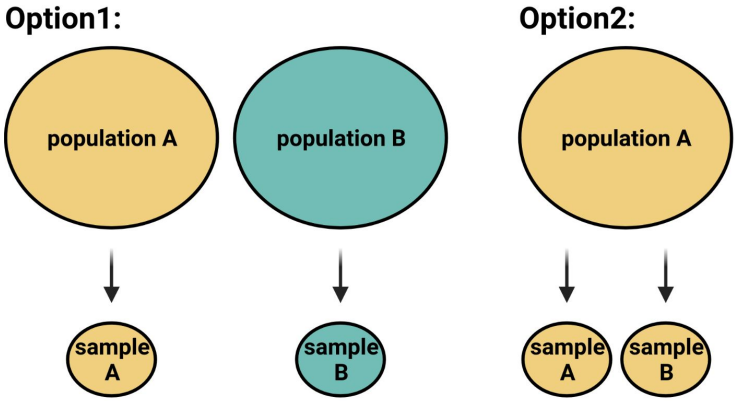
Functional annotation

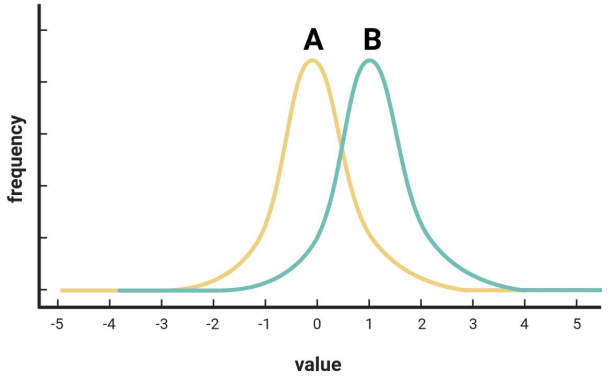
- Assumption: similar sequences (genes) have similar functions
- Sequence similarity indicates origin from a shared common ancestor
- Ancestor is likely to have inherited the same function to offspring
- Transfer of functional annotation based on sequence similarity
- *Arabidopsis thaliana* gene functions are well studied and usually serve as reference



Statistics

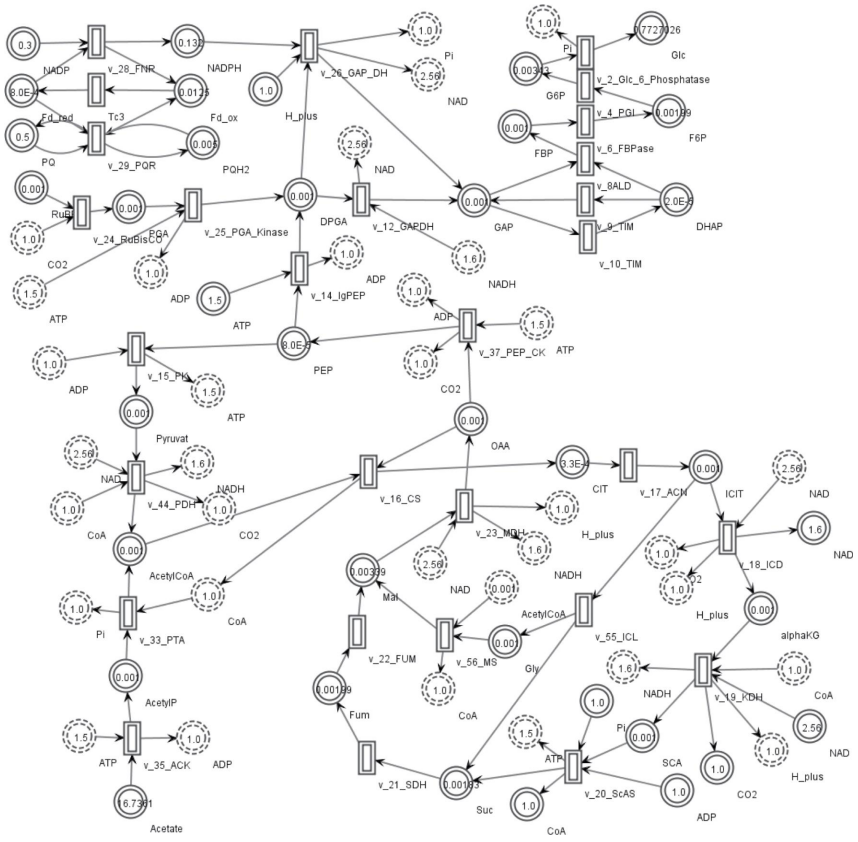
- Statistical tests to assess data sets
- Important for analysis of omics data sets
- Concept of a statistical test:
  - H0 = difference of two samples can be explained by random effects
  - H1 = certain factor(s) determine the difference of groups





Systems biology

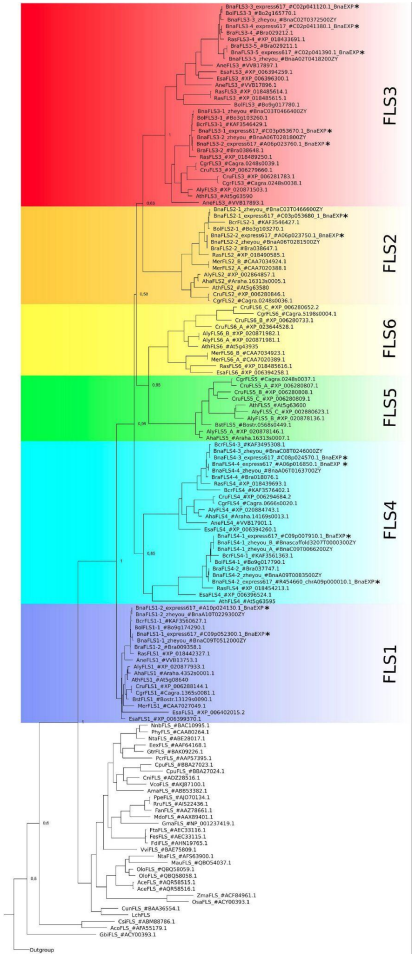
- Description of biological systems with mathematical methods
- Michaelis-Menten kinetics is basis of many metabolic models
- Integration of multiple omics data sets
- Modeling of metabolic flux through pathways
- Example: Modeling of aerobic carbon metabolism in *Chlamydomonas reinhardtii*



Brinkrolf et al., 2018: 10.1515/jib-2018-0018

Phylogenetics

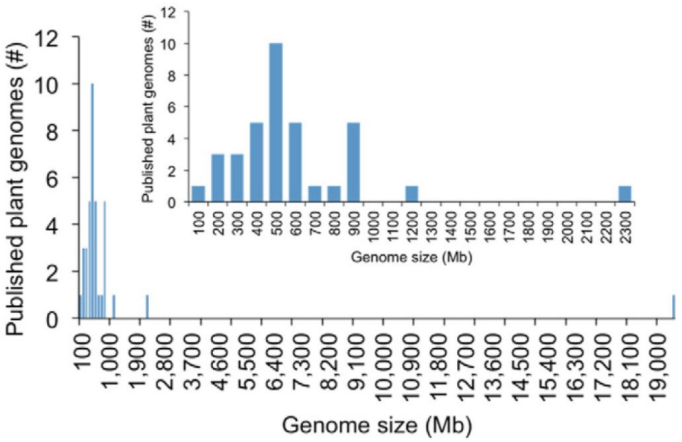
- Analysis of the evolutionary relationships between sequences/species
- Reconstruction of possible models to explain these relationships
- Not limited to sequence similarity, but considered (most likely) ancestral stages

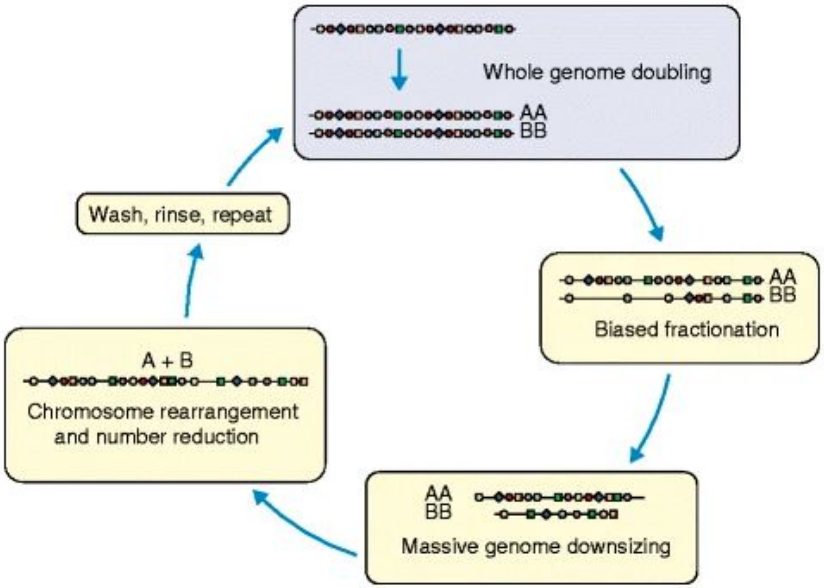


Schilbert et al., 2021: 10.3389/fpls.2021.733762

Phylogenomics

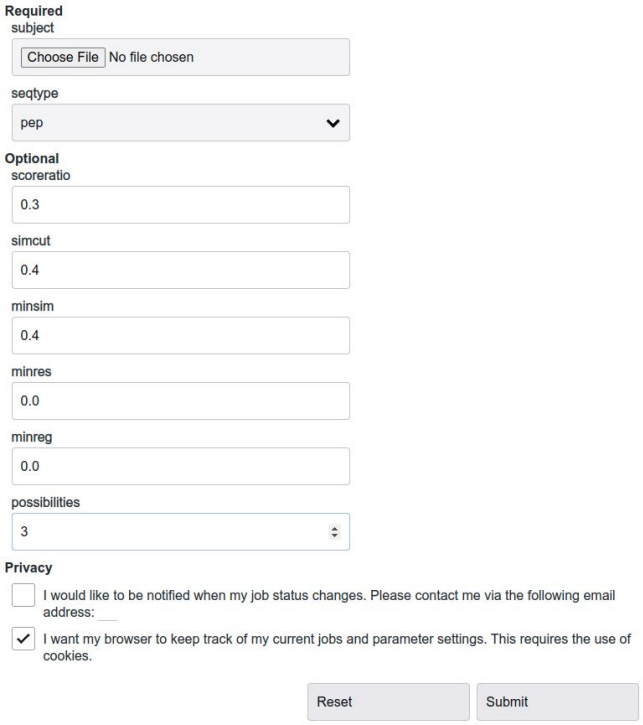
- Genomes evolve over time
- Whole genome duplication is crucial source for new sequences in plants
- Polyploidy = multiple copies of all chromosomes
- Transposable Elements (TEs) can increase genome size
- Huge differences in chromosome numbers and genome sizes





Wendel et al., 2016: 10.1186/s13059-016-0908-1
Michael & Jackson, 2013: 10.3835/plantgenome2013.03.0001in

Do you know any bioinformatics tools?

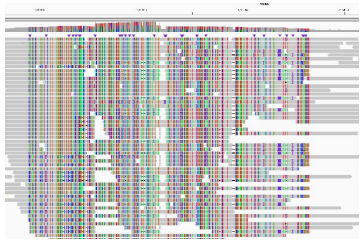


Types of bioinformatics tools

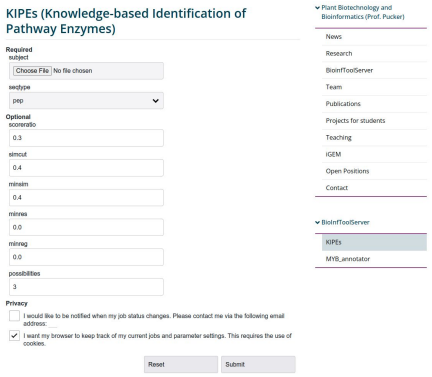
- Different types of tools:
  - Command line: require a basic understanding of Linux, but allow the processing of large data sets
    - Example: long read-based t-DNA analysis (loreta); https://doi.org/10.1186/s12864-021-07877-8

```
samtools view -S -b sample.sam > sample.bam
```

  - Graphical user interface (GUI): easier to use, but usually restricted to smaller data sets
    - Example: Integrated Genome Viewer (IGV), https://igv.org/



  - Web services: easy to use, but usually restricted to small data sets
    - Example: Knowledge-Based Identification of Pathway Enzymes (KIPEs); http://pbb.bot.nat.tu-bs.de/KIPEs



How to run command line tools

- Trimmomatic: Bolger, A. M., Lohse, M., & Usadel, B. (2014). Trimmomatic: A flexible trimmer for Illumina Sequence Data. *Bioinformatics*, btu170.
  http://www.usadellab.org/cms/?page=trimmomatic

```
java -jar <path to trimmomatic jar> SE [-threads <threads>] [-phred33 | -phred64] [-trimlog <logFile>] <input> <output> <step 1> ...
```

- Samtools: Heng Li, Bob Handsaker, Alec Wysoker, Tim Fennell, Jue Ruan, Nils Homer, Gabor Marth, Goncalo Abecasis, Richard Durbin, 1000 Genome Project Data Processing Subgroup, The Sequence Alignment/Map format and SAMtools, *Bioinformatics*, Volume 25, Issue 16, 15 August 2009, Pages 2078–2079, https://doi.org/10.1093/bioinformatics/btp352
  http://www.htslib.org/doc/samtools-view.html

```
samtools view -S -b sample.sam > sample.bam
```

Do you know any programming languages?

Script and programming languages

- Perl: one of the first script languages used in bioinformatics (sequences)
  - Examples: AUGUSTUS and helper scripts
  - https://www.perlfoundation.org/
- Python: established script language for bioinformatics (sequences)
  - Examples: BUSCO, KIPEs, MYB_annotator
  - https://www.python.org/psf/
- Julia: script language that is gaining attention (structures)
  - Examples: biojulia
  - https://julialang.org/
- R: established script language for bioinformatics (numbers/statistics)
  - Examples: DESeq2, ggplot2,
  - https://www.r-project.org/about.html
- Java: powerful for the analysis of large data sets
  - Integrated Genomics Viewer (IGV), SnpEff, samtools, Trimmomatic, …
  - https://www.java.com/de/ (commercial)

How to develop a tool?

1. Identify the problem
2. Precisely describe the problem and break into individual tasks
3. Identify a test data set (positive control)
4. Solve individual tasks
5. Test functionality using positive control
6. Define benchmarking dataset
7. Test functionality based on benchmarking data set
8. Release command line version
9. Add GUI (=Graphical User Interface)?
10. Establish web service?

Tool development - KIPEs (example): 1

- Identify the problem:
  - Annotate the flavonoid biosynthesis in a new plant species

- Precisely describe the problem and break into individual tasks:
  - Check and clean input data sets
  - Find genes (CHS, CHI, F3H, FLS, DFR, …)
  - Provide details about the candidates
  - Collect the CDS and peptide sequences of all players
  - Construct phylogenetic trees of all candidates
  - Analyse gene expression of the candidates

Tool development - KIPEs (example): 2

- Identify a test data set (positive control):
  - Flavonoid biosynthesis is well annotated in *Arabidopsis thaliana*

- Solve individual tasks:
  - Check whether user supplied valid sequences
  - Rename sequence names to remove “illegal” characters
  - Identify initial candidates based on BLAST
  - Check these initial candidates by global alignments
  - Screen for presence of functionally conserved amino acids
  - …

Tool development - KIPEs (example): 3

- Test functionality using positive control:
  - Run analysis on *A. thaliana* as technical check

- Define benchmarking dataset:
  - Good representation of actual data sets required (phylogenetic width)
  - Sufficient size to analyze computational requirements / run time
  - Manual validation of results must be possible

Tool development - KIPEs (example): 4

- Test functionality based on benchmarking data set:
  - Run analysis of benchmarking data set (repeat with each version)

- Release command line version:
  - Command line version allows quick fixing of bugs
  - Inclusion of additional features possible
  - Assessment of community interest in the tool
  - Tool deployment as part of collaborations
  - Large scale analyses are possible on compute clusters
  - Local data processing has security/safety benefits

Tool development - KIPEs (example): 5

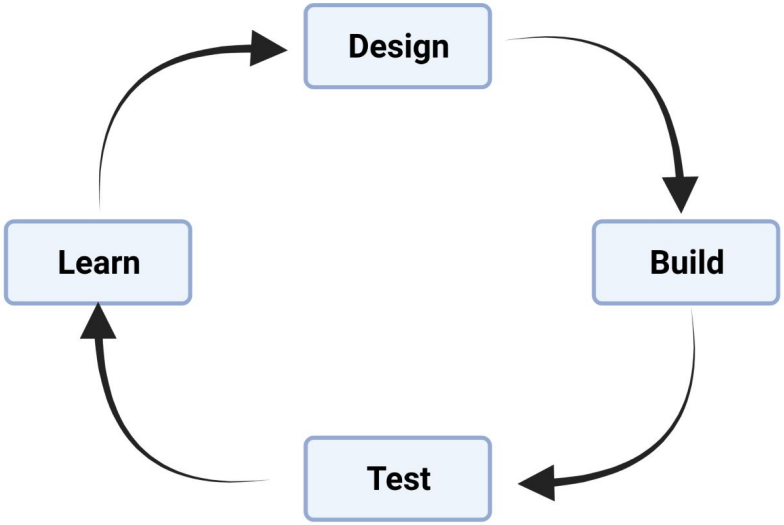
- Add GUI (=Graphical User Interface):
  - Addition of GUI is tedious work
  - Only successful tools justify the additional efforts
  - GUI is only helpful if additional user are expected
  - Still requires installation by (inexperienced) users

- Add web service:
  - Additional work that is only justified by additional benefits (more users/visibility)
  - Requires the availability of computational resources (provided by host)
  - Very convenient to use, but can raise data protection issues

How would you optimize a bioinformatics tool?

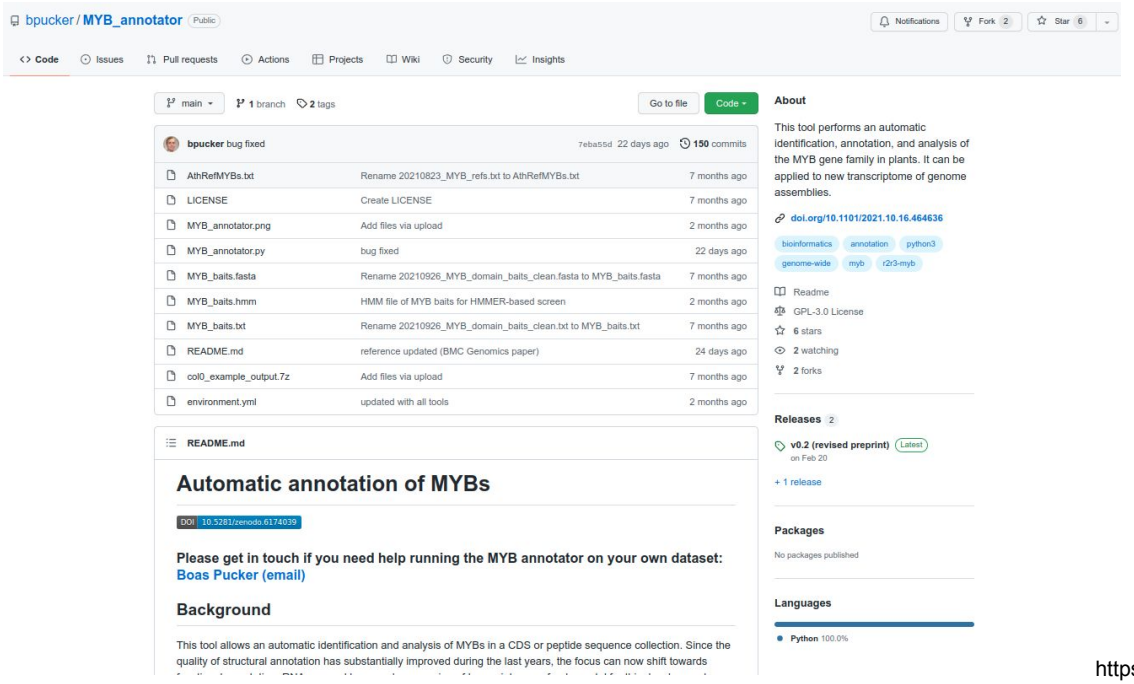
Design-Build-Test-Learn Cycle

- Software (any product) can be improved through the DBTL cycle:
  - Design how the software could work
  - Build the software according to the design
  - Test how well the software performs
  - Learn from the testing and start again with an improved design



Version control and repositories

- Version control is important when improving successively
- Different version control systems:
  - Git, CVS, SVN, …
- Online repositories that allow sharing of code:
  - GitHub, Bitbucket, GitLab, SourceForge, …



https://github.com/bpucker/MYB_annotator

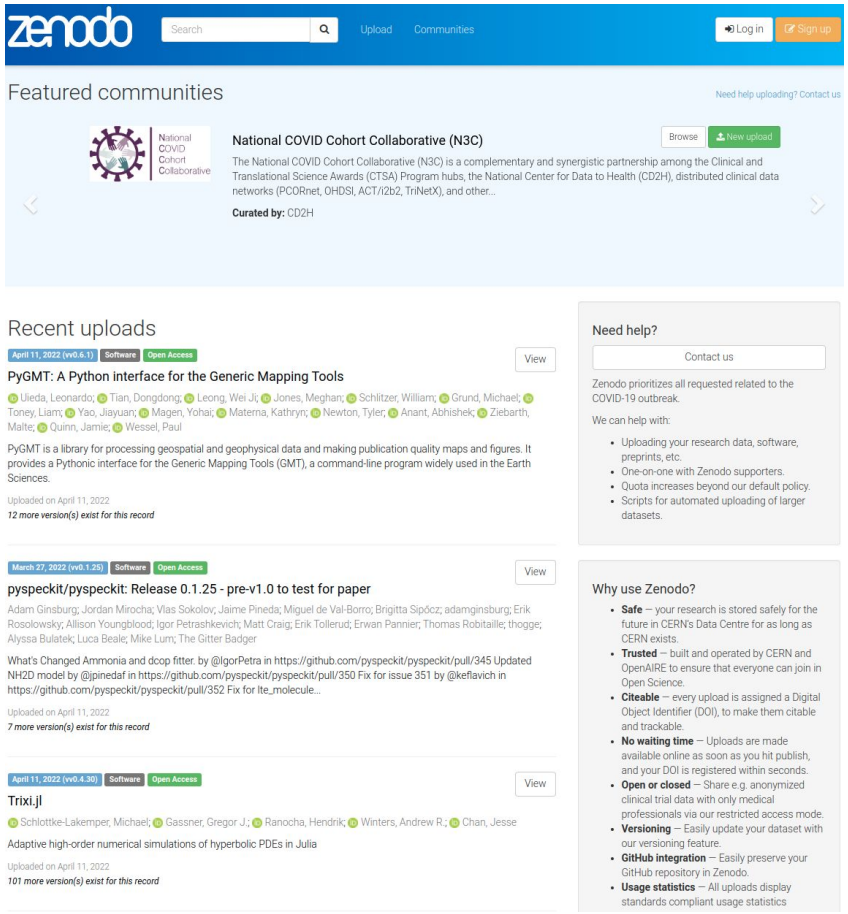
Licences

- MIT: leanest licence (everything is possible)
- Apache: similar to MIT, but lengthy
- GPL (General Public License): ensures that derived work remains open
- BSD (Berkeley Software Distribution): similar to MIT, but more cases specified

```
MIT License

Copyright (c) [year] [fullname]

Permission is hereby granted, free of charge, to any person obtaining a copy
of this software and associated documentation files (the "Software"), to deal
in the Software without restriction, including without limitation the rights
to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER
LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM,
OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
SOFTWARE.
```

Zenodo

- Data are permanently stored in CERN's data centre
- OpenAIRE allows everyone to engage in open science
  - socio-technical infrastructure for scholarly communication
- DOI makes entries citeable
- Uploads are published immediately
- Integration with github possible



Dryad

- International open-access repository
- Suitable for large data sets, but also an option for software
- CC0 licence makes everything completely re-useable by other
- DOIs make entries citeable

conda

- Bioinformatic tools often have dependencies
- Dependency = module/tool that needs to be installed already
- Conda enables installation without administrator privileges

Package, dependency and environment management for any language—Python, R, Ruby, Lua, Scala, Java, JavaScript, C/ C++, FORTRAN, and more.

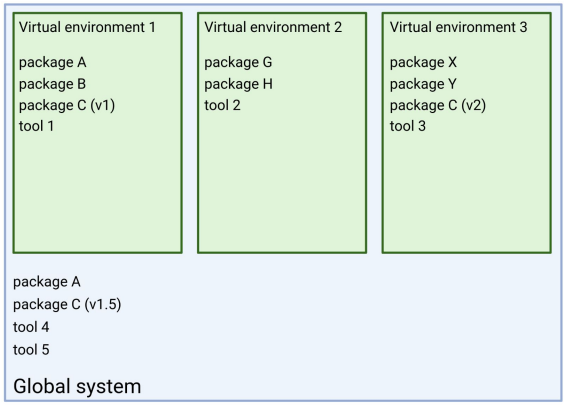
Conda is an open source package management system and environment management system that runs on Windows, macOS and Linux. Conda quickly installs, runs and updates packages and their dependencies. Conda easily creates, saves, loads and switches between environments on your local computer. It was created for Python programs, but it can package and distribute software for any language.

Conda as a package manager helps you find and install packages. If you need a package that requires a different version of Python, you do not need to switch to a different environment manager, because conda is also an environment manager. With just a few commands, you can set up a totally separate environment to run that different version of Python, while continuing to run your usual version of Python in your normal environment.

In its default configuration, conda can install and manage the thousand packages at repo.anaconda.com that are built, reviewed and maintained by Anaconda®.

Virtual environment

- Virtual environment allows controlled installation of tools without interfering with system
- Tool and all dependencies are installed in a box
- Different versions of packages/modules are required for different tools
- Different virtual environments can be used for different tools



12.2. Creating Virtual Environments

The module used to create and manage virtual environments is called `venv`. `venv` will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running `python3` or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the `venv` module as a script with the directory path:

```
python3 -m venv tutorial-env
```

This will create the `tutorial-env` directory if it doesn't exist, and also create directories inside it containing a copy of the Python interpreter and various supporting files.

A common directory location for a virtual environment is `.venv`. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with `.env` environment variable definition files that some tooling supports.

Once you've created a virtual environment, you may activate it.

On Windows, run:

```
tutorial-env\Scripts\activate.bat
```

On Unix or MacOS, run:

```
source tutorial-env/bin/activate
```

(This script is written for the bash shell. If you use the **csh** or **fish** shells, there are alternate `activate.csh` and `activate.fish` scripts you should use instead.)

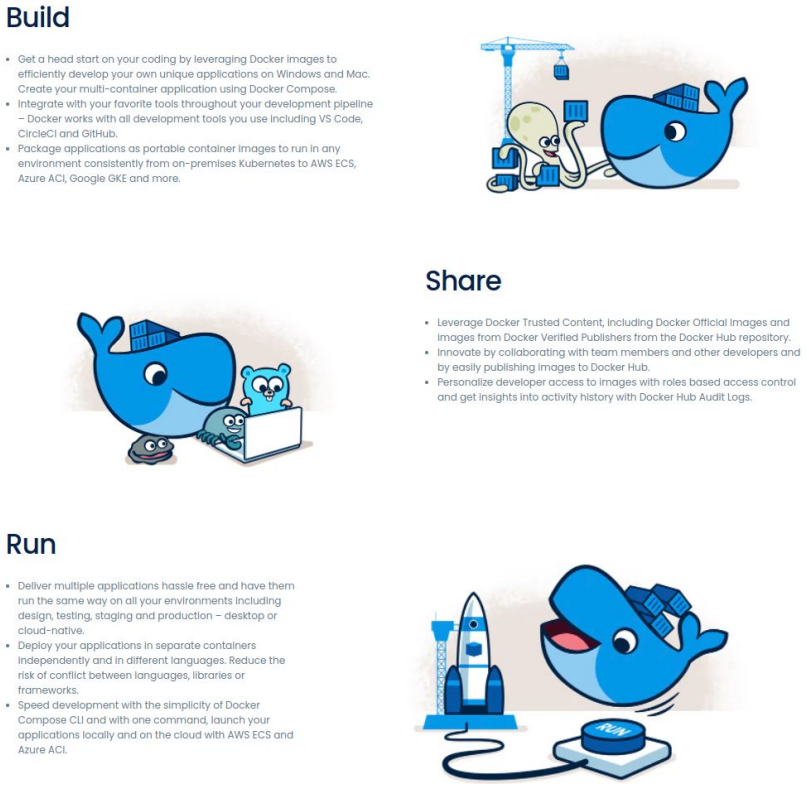
Activating the virtual environment will change your shell's prompt to show what virtual environment you're using, and modify the environment so that running `python` will get you that particular version and installation of Python. For example:

```
$ source ~/envs/tutorial-env/bin/activate
(tutorial-env) $ python
Python 3.5.1 (default, May  6 2016, 10:59:36)
  ...
>>> import sys
>>> sys.path
['', '/usr/local/lib/python35.zip', ...,
'~/envs/tutorial-env/lib/python3.5/site-packages']
>>>
```

https://docs.python.org/3/tutorial/venv.html

Docker

- Solution to prepare an environment for running tools
- Build: developer of tool generates an image of the required environment
- Share: image is shared with users of the tool
- Run: user can mount the image to run a tool in a defined environment
- Alternatives: podman, OpenVZ, VirtualBox, Kubernetes, LXC, ZeroVM, …



https://www.docker.com/

Galaxy

- Graphical user interface (web-based platform) for bioinformatic workflows
- Open source enables local installation (e.g. on compute cluster)
- Supported by de.NBI, elixir, and many others

https://usegalaxy.org/
Afgan et al., 2018: 10.1093/nar/gky379

de.NBI

- German Network for Bioinformatics Infrastructure Service, Training, Cooperations & Cloud Computing
- Centrally funded support for bioinformatic analyses in Germany
- Almost 100 tools provided and supported by de.NBI
- Training events for next generation scientists
- de.NBI cloud enables researchers in Germany to conduct bioinformatics research

https://www.denbi.de/

ELIXIR

- Intergovernmental organisation of life scientists and computer scientists in Europe
- Provides resources for European life scientists
- 23 ELIXIR nodes in different countries
- EMBL-EBI heads ELIXIR
- de.NBI is German ELIXIR node

https://elixir-europe.org/about-us/who-we-are

Time for questions!

Questions

1. Which fields are part of bioinformatics?
2. What is the definition of a genome, transcriptome, proteome, and metabolome?
3. What is the full name of ‘BLAST’?
4. Which (sequencing) method can be used to analyze a genome?
5. Which method can be used to analyze a transcriptome?
6. Why do biologists apply statistical methods?
7. Which languages are used for the development of bioinformatics tools?
8. Which steps are involved in software development/improvement?
9. Which platforms enable developers to share their software?
10. Which licence can be used to make software freely available to other researchers?
11. Where can you find (online) computational resources for bioinformatics?