Bioinformatics Resources

Lecture & Exercises
Prof. B. Rost, Dr. L. Richter, J. Reeb
Institut für Informatik I12
Bioinformatics Resources

- Organization
- Schedule
- Overview
Organization

- **Lecture:** Friday 9-12, i.e. 9.30-11.45 o’clock
  10 - 15 min break in between
  Room 00.13.009A

- **Exercise:** Monday 14-16 o’clock room
  00.08.038, starting Mon, May 2\textsuperscript{nd}
  Friday 13-15 o’clock room 01.09.014
  starting Fri, Apr. 29\textsuperscript{th}
Team Behind the Course
Putative Schedule

<table>
<thead>
<tr>
<th>Date</th>
<th>Topic</th>
<th>Date</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apr. 22nd</td>
<td>Intro, General Overview (1. sh.)</td>
<td>Jun 10th</td>
<td>No-SQL (7.sh.)</td>
</tr>
<tr>
<td>Apr. 29th</td>
<td>Sequence Databases (2. sh.)</td>
<td>Jun 17th</td>
<td>No-SQL (8.sh.)*</td>
</tr>
<tr>
<td>May 6th</td>
<td>No lecture</td>
<td>Jun 24th</td>
<td>JavaScript / UI (9.sh.)</td>
</tr>
<tr>
<td>May 13th</td>
<td>Sequence Databases (3. sh.)</td>
<td>Jul 1st</td>
<td>Web Services (10.sh.)</td>
</tr>
<tr>
<td>May 20th</td>
<td>Structure Databases (4. sh)*</td>
<td>Jul 8th</td>
<td>Bioinformatics Suites / Forums</td>
</tr>
<tr>
<td>May 27th</td>
<td>SQL (5. sh.)</td>
<td>Jul 15th</td>
<td>Wrap Up, Q&amp;A</td>
</tr>
<tr>
<td>Jun 3rd</td>
<td>SQL (6. sh)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* These exercises can earn you a bonus
Schedule Details

- No lecture on May 6th
- No exercise on Fri, May 13th and Mon, May, 16th
- Exercise sheets are published on Fridays and discussed Fri/Mon the week after
- Last sheet/exercise: Jul 4th, Fri/Mon 8th/11th
- Exam: (working date): August 5th, to be discussed with the audience
Overview

- lecture is new and considered beta 1
- second iteration
- no prior syllabus available and subject to change
- depending on the advancements in the lecture single topics could be added or dropped
- the sequence of topics might be shuffled
- hybrid nature: presentation of existing resources are blended with back- and front-end technology
Exercises

- Exercises help to convert knowledge into a skill
- practical application of topics covered in the lecture
- active exploration of bioinformatics resources
- implementing various parts of bioinformatics resource
- use Python/Biopython as common platform
Meaning

- What does “resource” actually means?
- A Google query about “Bioinformatics Resource” yields about 20 Mio hits
- Falls roughly into three categories:
  - Databases
  - Tools
  - Service centers
Working on a Definition

- a collection of information which is useful to do research in the area of life sciences/computational biology
- contains the information itself
- provides appropriate interfaces to access the information
- may provide tools for interactive data analysis
Genbank / NCBI

- NIH genetic sequence database
- annotated collection of all publicly available DNA sequences
- part of the International Nucleotide Database Collaboration together with DNA DataBank of Japan (DDBJ) and the European Molecular Biology Laboratory (EMBL)
Genbank II

- new release every 2 months
- retrievable via FTP from the NCBI website
- current release is 213.0, April 15, 2016
- 211,423,912,047 bases from 191,739,511 reported sequences
- (187,893,826,750 bases from 181,336,445 reported sequences Feb 2015)
- Genbank flat file format
Genbank III

- three main divisions: CoreNucleotide, dbEST, dbGSS
- Querying over Entrez Nucleotide
- interactive BLAST analysis with user sequences
- programmatic access via NCBI e-utils
Swissprot

- official name: UniProtKB/Swiss-Prot
- history
- current release: 2016_04
- 548208 sequence entries
- (550960 sequence entries, 195282524 amino acids abstracted from 235893 references last year)
- manually annotated

BioinfRes SoSe 16
Swissprot/Uniprot II

- manual annotation process
- standard operation procedure
- controlled vocabularies
- guidelines
- offered services: BLAST, Align, ID mapping
- associated services
Other Uniprot Services

- TrEMBL
- Proteomes
- UniRef
- UniParc
- programmatic access
PDB

- History
- 118,087 structures, incl. 115,169 proteins
  - (108124 structures, incl. 100450 proteins last year)
- PDB formats
- data upload/validation
- data dictionaries
PDB II

- retrieval
- programmatic access
- visualization with the different views
- file format transitions: pdb and mmcif
SCOP/e

- Structural Classification of Proteins
- history, current version is SCOPe 2.05
- changes in SCOPe
- access
- needed/recommended additional software
PFAM

- current version is 29.0, December 2015
- what is is about
- categories
- interactive use
- programmatic access
Prosite

- current version 20.125 Apr 5th, 2016
- UniRule format and ProRule
- access
- typical use and interfaces
PubMed and discussion forums

- What is it for
- Search opportunities
- Linking to other information sources
- Search strategies
- A tour through various discussion forums
File Formats*

- High Throughput data:
  - BAM, SAM
  - VCF
- Newick tree file format
- Genbank/EMBL
- PDB: mmCIF

* mostly integrated
File Formats

- Equivalence and transformations between different formats
- XML formats
- RDF formats
SQL

- SQL basics
- data types
- table creation and manipulation
- join
- select
SQL II

- keys
- indexes
- performance influence of indexes
- similarity search vs substrings
- permissions
SQL III

- transactions
- setup, administration, backup
- programmatic access
- mySQL, postgresQL
SQL IV

- general hints for database design
- do’s and don’ts
- normalization ultra light
No SQL

- definitions of NoSQL
- advantages / disadvantages
- underlying theory
- typical use cases
- types of No-SQL database
- query (languages)
No SQL Systems

- MongoDB
- CouchDB
- Neo4J
- programmatic access
(Storing Facts)*

- triple stores
- data model
- rdf refresher
- query language: sparql
- examples

* optional, might be dropped
Programming Libraries

- roadshow of programming libraries dedicated to bioinformatics:
  - bioperl
  - biopython
  - bioJS
  - visualization
Graphical User Interfaces

- principles
- interaction modes
- modelling
- interaction modes
Graphical User Interfaces*

- interactive user interfaces with JavaScript
- language basics
- programming model
- client/server communication with json

* to be confirmed
JavaScript

- libraries for data visualization/bioinformatics
- bioJS
- D3
Client/Server Models

- cgi
- Webservices
- Remote Procedure Calls / CORBA
- security considerations
Authentication/Encryption

- authentication models
- communication encryption
- data/result encryption
- legal privacy issues
- data access models
Web Services I

- types of web services
- web service components
- integration of web services in software
Web Services II

- client side interfaces to web services
- server side interfaces to web services
- Apache configuration for web services
- required modules
- configuration
- performance
Bioinformatics Suites

- where to find
- installation/configuration
- workflow systems: e.g. Taverna,....
- EMBOSS, STADEN
- bio-.....
- .....
Selected Bioinformatics Suites

- Aquaria
- PredictProtein
- ....
Summary I

- aim of this module:
  - shape the concept of a bioinformatics resource
  - become familiar with some of the most prominent examples out there
  - get in touch with the underlying technology
  - gather ideas and experience how to realize a new bioinformatics resource
Summary II

- hands on (interaction) experience with existing experience
- backend technology, i.e. various database models
- frontend technology to realize the UI/ design rationales
- communication models
Grading:

- graded by a written exam 90/100 min
- scheduled day xxx depends on:
  - available room
  - number of participants
- exam admission: no admission limit
- with sufficient performance in the two marked exercises you can earn a bonus
- the bonus applies only if you pass the exam
Exercises

- Exploration of available resources
- simple to intermediate programming tasks
- publication/presentation of the task in week x
- solutions x+1
Exercises II

- 10 exercise sheets
- work in groups of 2 for the bonus
- discussion with the audience
Exercises III

- groups fixed for the bonsu
- new sheets are published on Friday
- submission is due on Friday morning for all groups
- two slots for exercises
Questions & Answers
Programming Exercises

- we will use Python for our programming exercises
- scripting language
- basic understanding of Python should be sufficient to understand the presented code snippets
- vivid community for support and development
Programming Exercises II

- object oriented
- good integration with database systems and web access
- good integration with sophisticated data analysis tools like: numPy, sciPy, mathplotlib
- BioPython
Structure your research work

- computational biology is data driven
- results matter -> more results matter more
- other than e.g. software development there is no final release version and all prior bugs/version are abandoned
- appropriate documentation of the experiments to reconstruct the intermediate steps is important, otherwise you may with result01 - result1000 files
Our preferred Software Setup

- Anaconda
- iPython notebooks
Anaconda

- Python distribution (https://www.continuum.io)
- clever packet manager: conda
- allows a complete installation including various configuration next to each other in the user space
- no privileges needed
- your host system is not modified
- works with Windows, OS X, Linux

BioinfRes SoSe 16
Some snippets from the conda cheat sheet

- Use “conda create –n xxx biopython” to create a new environment xxx and install biopython
- Use “(source) activate xxx” to activate this environment in your shell
- Allows different versions of python to be installed at the same time
iPython/Jupyter

- http://jupyter.org
- supports many different languages, we use it for python
- use conda to install the package:
  conda install jupyter
- easy start of notebook:
  jupyter notebook

BioinfRes SoSe 16
Advantages of a Notebook

- allows you a seamless integration of:
  - (rich) text
  - (live) code
  - visualizations
- tie together your analysis script, the results and an interpretation/discussion
- you can archive and share the notebooks easily
Biopython

- [http://biopython.org](http://biopython.org)

- if installed: “import Bio” loads it in your scripts


```python
from Bio.Seq import Seq
# create a sequence object
my_seq = Seq('CATGTAGACTAG')

# print out some details about it
print 'seq %s is %i bases long' % (my_seq, len(my_seq))
print 'reverse complement is %s' % my_seq.reverse_complement()
print 'protein translation is %s' % my_seq.translate()
```
Biopython

seq CATGTAGACTAG is 12 bases long
reverse complement is CTAGTCTACATG
protein translation is HVD*

taken from http://biopython.org/wiki/SeqIO:
from Bio import SeqIO
handle = open("example.fasta", "rU")
for record in SeqIO.parse(handle, "fasta") :
    print record.id
handle.close()

from Bio import SeqIO
record = SeqIO.read(open("single.fasta"), "fasta")
Biopython

- advantage of easier / more clear syntax than Perl
- oriented to BioPerl
- supports a lot of common bioinformatics file formats
- supports access to online services like NCBI, Expasy ...
- more interfaces for bioinformatics software

[http://biopython.org/DIST/docs/tutorial/Tutorial.html](http://biopython.org/DIST/docs/tutorial/Tutorial.html)

BioinfRes SoSe 16
Dedicated Data Structures

- sequence (Seq): beside the sequence of residues it allows also to provide an Alphabet object -> kind of type safety for DNA and protein sequences
- typical functions like complement(), reverse_complement()
Dedicated Data Structures

- parsing functions for different sequence formats
- parsing functions alignment formats know about the different components
- as well as respective output functions
- different translation tables
- various predefined alphabets
Python Basics

- [https://docs.python.org/2/tutorial/index.html](https://docs.python.org/2/tutorial/index.html)
- good interactive handling, i.e. you can evolve and evaluate your code directly in python shell
- later you can include it in your script
- basic data types:
  - numerical types comparable to Perl, C, Java
  - strings
  - boolean
Sequence Types

- supports easy check for an element
- mutable types: List, Bytearray
- immutable: String, Tuple
- slicing: act on sub sets not only on single elements
Other Collection Types

- **Set**: every element exists only once
- **Dictionary**:
  - can store key/value pairs
  - key has to be immutable (hashable)
- all collection types support iterators
Important Syntax

- white space (tabs, spaces) and : are used to structure the code in blocks, similar to {} in other languages
- same indentation == same block
- usual control structures available

```python
for w in words:
    print w, len(w)

# if you want to iterate by numbers you
# have to use range()
for i in range(len(a)):
    print i, a[i]
```

BioinfRes SoSe 16
Important Syntax

- Definition of functions:

```python
def fib(n):    # write Fibonacci series up to n
    """Print a Fibonacci series up to n.""
    a, b = 0, 1
    while a < n:
        print a,
        a, b = b, a+b
```

- Arguments can be passed by:
  - name
  - position

- Arguments can have default values -> optional in the call

- Package are loaded with the import directive