Franklin for students

The MBG jupyter exercise tool

Table of contents

Welcome

Download the HTML course website for offline viewing at the exam (unzip once downloaded and double-click index.html).

Download the HTML Python documentation for offline viewing at the exam (unzip once downloaded and double-click index.html).

This is the homepage for the AU course Bioinformatics and programming (Bioinformatik og programmering). You will find all course content here. The Brightspace course page is only used for communication, and assignments.

Course description

After the course, the participants will have basic knowledge of computer methods and applications for analyzing biological sequence data and insight into principles and techniques for constructing simple programs. Participants will acquire practical experience with analyzing problems in bioinformatics and related fields and implementing programs to solve such problems using the Python programming language.

The participants must, at the end of the course, be able to:

- Apply fundamental constructs of a programming language.
- Analyse data and construct data structures for the representation of data.
- Analyse simple computational problems and construct programs for their solution.
- Describe and relate essential methods in bioinformatics analysis.
- Apply bioinformatics software to biological data.
- Judge the reliability of results obtained using Bioinformatics software.

Course contents

The course introduces programming and its practical applications in bioinformatics. The course also outlines and discusses bioinformatics algorithms, and the most common tools for bioinformatics analysis of sequence data are presented and demo nstrated. The participant will acquire and train basic programming skills during the first seven

weeks. The last seven weeks introduce key topics in bioinformatics, focusing on applying bioinformatical software and developing programming skills. Subjects for lectures and exercises include bioinformatics databases, sequence alignment, genome annotation, sequence evolution, and phylogenetic analysis.

Part I

Jupyter

1 Jupyter

This pages are under construction

Jupyter Notebooks are an interactive computing environment that allow users to create and share documents containing live code, equations, visualizations, and narrative text. Originally developed as part of the IPython project, Jupyter (short for Julia, Python, R) now supports over 100 programming languages and has become a standard tool in data science, scientific computing, and education.

At the core of a Jupyter Notebook is a web-based interface that organizes content into "cells." These cells can contain code (typically in Python, but also in other languages via kernels), formatted text using Markdown, LaTeX for equations, and embedded multimedia elements. Users execute code in-place, and outputs such as plots or tables appear directly below the corresponding cells. This structure enables exploratory data analysis and facilitates reproducibility by interleaving code and its results with documentation.

Notebooks are stored in .ipynb files (JSON format), which preserve the code, outputs, and formatting. They can be run locally using the Jupyter server or hosted in cloud environments such as Google Colab or Binder. For scientific workflows, notebooks can integrate with tools for version control, containerization, and workflow management, making them a flexible instrument for open and reproducible research.

Despite their strengths, Jupyter Notebooks are not without limitations. Version control can be challenging due to the JSON-based format, and improper use (e.g., out-of-order execution) can compromise reproducibility. Nevertheless, their advantages in accessibility, interactivity, and communication have made them central to modern computational work.

Part II

Docker

2 Jupyter

This pages are under construction

Jupyter Notebooks are an interactive computing environment that allow users to create and share documents containing live code, equations, visualizations, and narrative text. Originally developed as part of the IPython project, Jupyter (short for Julia, Python, R) now supports over 100 programming languages and has become a standard tool in data science, scientific computing, and education.

At the core of a Jupyter Notebook is a web-based interface that organizes content into "cells." These cells can contain code (typically in Python, but also in other languages via kernels), formatted text using Markdown, LaTeX for equations, and embedded multimedia elements. Users execute code in-place, and outputs such as plots or tables appear directly below the corresponding cells. This structure enables exploratory data analysis and facilitates reproducibility by interleaving code and its results with documentation.

Notebooks are stored in .ipynb files (JSON format), which preserve the code, outputs, and formatting. They can be run locally using the Jupyter server or hosted in cloud environments such as Google Colab or Binder. For scientific workflows, notebooks can integrate with tools for version control, containerization, and workflow management, making them a flexible instrument for open and reproducible research.

Despite their strengths, Jupyter Notebooks are not without limitations. Version control can be challenging due to the JSON-based format, and improper use (e.g., out-of-order execution) can compromise reproducibility. Nevertheless, their advantages in accessibility, interactivity, and communication have made them central to modern computational work.