Read this first

Please add project ideas by Thursday Feb 18 2021 if possible

Instructions: Copy/paste the template below and add your project's details. Remember:

1. This year, students will be given **175 hour projects** (about half as long as previous years) - take this into account with your expectations and project sizes.
2. **Projects must be *code* projects.** Not research projects or tech projects that aren't code (e.g. no pure design, docs, or ux projects)
3. **Questions?** Please email obf-gsoc-mentors@googlegroups.com (all mentors and admins), obf-gsoc-admins@googlegroups.com (admins only), or book a chat.

TEMPLATE: (please copy/paste this template to add your project below)

PROJECT TITLE + ORGANISATION

- **Project description**
  - *<tell us about your project>*
- **Languages and skills needed**
  - *(Shorter lists are more likely to attract a broader range of candidates)*
  - Language / skill
  - Language / skill
- **Difficulty**
  - <Easy/Medium/Hard> - *if possible use qualifiers, e.g. "easier if you know react, medium if not"*
- **Mentors**
  - <name>, contact info (email, github id, etc. as desired)
- **Student Benefits**
  - <details if relevant / or delete>
- **How to Apply**
  - <instructions if relevant - maybe add a chat room, application template, or other requirements>

2021 OBF projects

PROJECT TITLE + ORGANISATION

- **Project description**
  - *<tell us about your project>*

- **Languages and skills needed**
  - *(Shorter lists are more likely to attract a broader range of candidates)*
  - Language / skill
  - Language / skill
- **Difficulty**
  - <Easy/Medium/Hard> - *if possible use qualifiers, e.g. "easier if you know react, medium if not"*
- **Mentors**
  - <name>, contact info or github id
- **Student Benefits**
  - <details if relevant / or delete>
- **How to Apply**
  - <instructions if relevant - maybe add a chat room, template, or other requirements>

Distributed workflow execution with data streaming (Common Workflow Language project / Toil)

- **Project description**

Command line scientific analysis tools often support streaming data into or out of the tool. (At the command line we use the unix pipe "|" or named pipes to implement this). This speeds up the analysis by avoiding slow disk/storage IO.

While the CWL standard supports this approach, no CWL-aware workflow system makes use of this optimization.

You would implement this feature (automatic streaming data in and out of scientific computing tools) to one of the CWL workflow engines, such as Toil (which is Python based).

The first iteration would stream in and out of object stores (Amazon S3, Google Cloud Storage, etc..). More advanced implementations may feature direct streaming between the tools, but this requires refactoring the job scheduling engine.

Notes:

All work must be done openly and under the Apache 2.0 license.

- **Languages and skills needed**
  - Python
- **Difficulty**
  - Medium to Hard

- **Mentors**
    - Michael R. Crusoe
- **Student Benefits**
    - If successful, you will have contributed a major feature to a popular workflow engine!
- **How to Apply**
    - Get in touch with us at https://github.com/DataBiosphere/toil/issues/3469

Enabling and prototyping JavaScript visualizations in the Qt-based viewer TOPPView (OpenMS)

1. **Project description**

    OpenMS is an open-source library and toolset for mass-spectrometry data analysis. One of its key capabilities is the visualization of mass-spectrometry data via its viewer "TOPPView". TOPPView is written in C++ using the Qt (currently 5.x) library. However, TOPPView is lacking some often requested high-level summary views of results that are produced by OpenMS' other tools. In the typical open-source spirit we would like to integrate other open-source libraries by the community that already solved this problem sufficiently. Unfortunately, most of the successful interactive visualization libraries in this research area are developed in JavaScript (Nightingale, Lorikeet) and reimplementation in C++ would be tedious. Also, general JavaScript-based plotting libraries like Plotly.js would be a great addition for the Viewer. With this addition, we hope to attract both new users and (frontend) developers.

    Therefore we are looking for a student that enables the usage of JavaScript libraries within TOPPView, probably using the QtWebEngine AddOn. The aim of the project would be to develop an easy-to-use interface between QtWebEngine to the existing C++-based QWidgets and showcase this interface by integrating one or more visualization prototypes that we developed for external JavaScript supporting workflow systems (KNIME) that read the data from a table instead of our C++ data structures. If time permits, additional library integrations or own visualizations could be developed.

    An example of a similar project tackling general data visualizations might be https://github.com/YimingYAN/qvisualisation.

2. **Languages and skills needed**
    - *C++ (intermediate; mostly for interfacing with Qt)*
    - *Basic knowledge about Qt slots and signals helps*
    - *CMake (basic knowledge to build OpenMS/TOPPView and add a dependency to the new addons needed)*
    - *Git (basic; our Git workflow can be learned quickly but it should be done during the application period)*

    - *JavaScript (intermediate; react to signals from Qt using e.g. qtwebchannels.js library)*
    - *Abstract knowledge about the data to be visualized in mass-spectrometry can be acquired in a few days or when the need arises*
3. **Difficulty**
    - Medium (easy if you are an expert in Qt)
4. **Mentors**
    - Julianus Pfeuffer (@jpfeuffer)
5. **Student Benefits**
    - Practical experience in proteomics and metabolomics, using an open-source software project that is used around the world
    - Gain insight into the development process of a medium-size open-source project (including working with continuous integration systems and pull request reviews)
    - Improving your oral and written communication skills in a team environment
6. **How to Apply**
    - For the short period of this year's GSOC we require a first minor contribution to our GitHub project to see that the student's coding environment is correctly set up and that they are familiar with a basic Git workflow.
    - Provide a cover letter that explains why your skills would be a good fit. If you don't have the skills, explain why you would like to learn those skills (2 pages maximum)
    - Provide a resume with a list of skills and experience (2 pages maximum)
    - Provide a breakdown of how you'd run this project – i.e. Features A, B delivered in the first two weeks, Features C, D delivered in later weeks. Show your proposal to mentors for feedback as they may be able to suggest improvements!
    - Provide links to any other code you might have contributed to eg. Github, bitbucket repos/commits

Deploying deep learning models (OpenMS)

7. Project description

    OpenMS is an open-source library and toolset for mass-spectrometry data analysis. Similar to other fields, many state-of-the-art methods nowadays use deep learning, are python based and use one of the prominent deep learning frameworks.

    To make these methods available to a larger audience and to integrate with larger OpenMS workflows, we are looking for a student that carefully evaluates how existing deep learning methods and models can be natively used and deployed in OpenMS. The outcome of the project would be an OpenMS tool written in C++ that uses a published model, potentially performs some retraining to adapt current data and performs the predictions. E.g. one application could be predicting when peptides elute from a chromatographic column (see e.g., https://github.com/compomics/DeepLC). Ideally, all supported operating systems (Mac, Win, Linux) will be covered.

8. Languages and skills needed
    - C++ (intermediate)
    - Basic knowledge of deep learning frameworks
    - CMake (basic knowledge to build OpenMS/TOPPView and to add new dependencies to machine learning framework)
    - Git (basic; our Git workflow can be learned quickly, but it should be done during the application period)
    - Abstract knowledge about mass spectrometry data, can be acquired in a few days or when the need arises
9. Difficulty
    - Medium
10. Mentors
    - Timo Sachsenberg (@timosachsenberg)
    - Oliver Alka (@oliveralka)
11. Student Benefits
    - Practical experience in the field of proteomics and metabolomics, using an open-source software project that is used around the world
    - Gain insight into the development process of a medium-size open-source project (including working with continuous integration systems and pull request reviews)
    - Improving your oral and written communication skills in a team environment
12. How to Apply
    - For the short period of this year's GSOC we require a first minor contribution to our GitHub project to see that the student's coding environment is correctly set up and that they are familiar with a basic Git workflow.
    - Provide a cover letter that explains why your skills would be a good fit. If you don't have the skills, explain why you would like to learn those skills (2 pages maximum)
    - Provide a resume with a list of skills and experience (2 pages maximum)
    - Provide a breakdown of how you'd run this project – i.e. Features A, B delivered in the first two weeks, Features C, D delivered in later weeks. Show your proposal to mentors for feedback as they may be able to suggest improvements!
    - Provide links to any other code you might have contributed to eg. Github, bitbucket repos/commits

Implementations of NEON functions (SIMDe)

- **Project description**

    SIMD Everywhere (SIMDe) contains portable implementations of (traditionally) architecture-specific SIMD functionality such as SSE (x86) and NEON (Arm). This allows code written to target a specific architecture or architecture extension to be run on any architecture (e.g., running SSE code on an Arm CPU) with minimal performance penalties.

For this project, you would be implementing NEON functions which are not yet supported, including implementing entire families of functions SIMDe currently doesn't support. For more information, see issue #10 in our issue tracker.

A portable implementation is required for all functions, but you'll also be using other ISA extensions (e.g., SSE/AVX, AltiVec, WASM SIMD128, etc.) to create accelerated implementations for some platforms, as well as tests to ensure correctness.

- **Languages and skills needed**
  - *C*
- **Difficulty**
  - Medium. Easier if you already know C and/or are familiar with SIMD programming.
- **Mentors**
  - Evan Nemerson
  - Ng Zhi An (@ngzhian)
- **Student Benefits**

  You will likely walk away with a fairly deep understanding of how SIMD programming works and the differences in functionality between various ISA extensions (including SSE and NEON), which should help you better optimize software.

  You will also have helped make it much easier to port existing software to and from the Arm architecture, and to develop new software for Arm from non-Arm machines.

  Furthermore, it should provide you with valuable experience with developing portable software and working with an extensive test suite and CI to verify correctness.

- **How to Apply**
  - You can use the SIMDe issue tracker to get in touch (via issue #10 or #702), visit our chat room on Gitter, or e-mail Evan Nemerson directly.

Implementations of AVX-512 functions (SIMDe)

- **Project description**

  SIMD Everywhere (SIMDe) contains portable implementations of (traditionally) architecture-specific SIMD functionality such as SSE (x86) and NEON (Arm).

This allows code written to target a specific architecture or architecture extension to be run on any architecture (e.g., running SSE code on an Arm CPU) with minimal performance penalties.

For this project, you would be implementing AVX-512 functions which are not yet supported, including implementing entire families of functions SIMDe currently doesn't support. For more information, see the various AVX-512 issues in our issue tracker.

A portable implementation is required for all functions, but you'll also be using other ISA extensions (e.g., SSE/AVX, AltiVec, WASM SIMD128, etc.) to create accelerated implementations for some platforms (including x86 machines which don't support AVX-512), plus tests to ensure correctness.

- **Languages and skills needed**
    - *C*
- **Difficulty**
    - Medium. Easier if you already know C and/or are familiar with SIMD programming.
- **Mentors**
    - Evan Nemerson (@nemequ)
- **Student Benefits**

    You will likely walk away with a fairly deep understanding of how SIMD programming works and the differences in functionality between various ISA extensions (including SSE and NEON), which should help you better optimize software.

    You will also have helped make it easier to port existing x86 software to other architectures such as Arm, RISC-V, PPC, etc., and made it easier for software to take advantage of new ISA extensions which are not yet widely available.

    Furthermore, it should provide you with valuable experience with developing portable software and working with an extensive test suite and CI to verify correctness.

- **How to Apply**
    - You can use the SIMDe issue tracker to get in touch (via one of the AVX-512 issues or #702), visit our chat room on Gitter, or e-mail Evan Nemerson directly.

Development of a user interface for the Ensembl Variant Effect Predictor neXtProt plugin as one of the community tools hosted on the neXtProt portal (Swiss Institute of Bioinformatics/CALIPHO)

- **Project description**

In order to be able to interpret human genomic variation data, several open source tools such as the Ensembl Variant Effect Predictor (VEP) have been developed to predict the structural and functional effects of variants (SNPs, insertions, deletions, CNVs or structural variants) on genes, transcripts, and protein sequences, as well as regulatory regions. The VEP tool takes input variants in different formats such as VCF, HGVS and variant identifier formats such as SNPs, and produces the output with the predicted effect on the selected biological entities. The VEP tool has numerous plugins including a neXtProt plugin, which was made public last november as a command line. This plugin improves the accuracy of predictions about coding single nucleotide polymorphisms by integrating manually curated information from neXtProt about domains, PTMs, interacting regions etc. associated with the affected amino-acid positions.

neXtProt is an open source discovery platform for human genes and proteins developed at the Swiss Institute of Bioinformatics (ELIXIR-CH). The neXtProt team would like to develop a web based user interface for the VEP neXtProt plugin and include it as one of the community tools hosted on the neXtProt portal, in order to improve its accessibility.

The proposed user interface visualizes the predicted variant effect output for all variants within a neXtProt entry. It has to handle the possibly large number of variants in the neXtProt entry and handle the corresponding large VEP output in an optimal manner. The student is expected to have knowledge in REST APIs and to utilize the neXtProt API to get the variants given a neXtProt accession. The student has the choice of the web technology to use to implement the UI (Javascript based technologies recommended). The UI has to handle large amounts of prediction data and visualize them in an efficient and user friendly manner.

- **Languages and skills needed**
  - *HTML/Javascript/CSS*
  - *React/Angular*
  - *Knowledge of REST API*
- **Difficulty**
  - Medium/Hard
- **Mentors**
  - Kasun Samarasinghe, kasun.wijesiriwardana@sib.swiss

    - Lydie Lane, lydie.lane@sib.swiss
- **Student Benefits**
    - To gain experience in implementing efficient UIs for big data visualization
- **How to Apply**
    - Email the CV and a short description on the skills relevant for the project `

gtfbase - A curated resource of multispecies genomic regions

- **Project description**
    - Each genome has some common features: exons that make the mRNA, coding domain sequence (CDS), and untranslated regions (UTRs) that are located both towards the 5' and 3' ends of the transcripts. ENSEMBL (http://ensembl.org/), Gencode (https://www.gencodegenes.org), and NCBI (https://www.ncbi.nlm.nih.gov/) are some of the key available resources that provide access to these features in the form of General Transfer Format (GTF) files (https://uswest.ensembl.org/info/website/upload/gff.html). While GTF files are by themselves comprehensive, a lot of analysis is focused on individual features. For example, any analysis focused on transcriptional regulation would focus more on exons and possibly introns and non-coding RNA than UTRs while translational regulation analysis would focus on only the CDS and possibly the UTRs. These analyses often require a BED file (https://uswest.ensembl.org/info/website/upload/bed.html). Though it is trivial to obtain a BED file from GTF, currently there are no resources that provide ready access to BED files. Though the GTF is supposed to be a standard format, there are differences in the annotation features for different species.
    - We have a collection of scripts currently available as part of gencode_regions repository: https://github.com/saketkc/gencode_regions that provides ready access to BED files of 5'UTR/exons/CDS/3'UTRs across multiple species. We plan to generalize these scripts into a usable tool that can be used to generate BED files for a variety of use cases and serve as a readily updated database of BED files that will keep in sync with ENSEMBL's GTF releases.
    - Goals: The current codebase is in Python and makes use of gffutils library for processing GTFs. The GTFs themselves cannot be assumed to be free of errors and hence while processing we need to be able to handle issues such as overlapping regions or infer missing feature annotations from known features. The student will execute the following:
        - Convert the existing scripts to a library with an extensible API that can be exposed to command line

        - Create a modular pipeline that will use the above library to create BEDs for GTFs of all organisms hosted on ENSEMBL: https://uswest.ensembl.org/info/data/ftp/index.html
        - The following bed files should be supported at the minimum:
            a. 5' UTR
            b. CDS
            c. Exons
            d. Introns
            e. Start codons
            f. Stop codons
            g. Non-coding RNA
            h. 3' UTR
            i. First exons
            j. Last Exons

- **Languages and skills needed**
    - Requires Python programming and some knowledge of Biology/genomics.
- **Difficulty**
    - Medium, easier if you are familiar with Python and the GTF file format
- **Mentors**
    - Saket Choudhary- saketkc@gmail.com
    - Amal Thomas- amalthomas111@gmail.com
- **Student Benefits**
    - Implementing and extending a bioinformatics software project that is a requirement of every bioinformatics researcher in one way or another
    - Gaining practical experience in writing a scientific manuscript
    - Improving your oral and written communication skills in a team environment
    - Authorship on a scientific manuscript (conditional on if we decide to write one)
- **How to Apply**
    - Get acquainted with gencode_regions, GTF, BED file formats (before GSoC!)
    - Provide a cover letter that explains why your skills would be a good fit. If you don't have the skills, explain why you would like to learn those skills. 2 pages maximum.
    - Provide a resume with a list of skills and experience. 2 pages maximum.
    - Provide a breakdown of how you'd run this project – i.e. Features A, B delivered in the first two weeks, Features C, D delivered in later weeks.
    - Show your proposal to mentors for feedback as they may be able to suggest improvements!

- Provide links to any code you might have contributed to eg. github, bitbucket repos/commits

Genomic Context Visualization modules - ETEToolkit/eggNOG

- **Project description**

This proposal involves **two** bioinformatic resources that enable phylogenomic analysis at the large scale: *ETE Toolkit* and *eggNOG*.

- **ETE** (Environment for Tree Exploration) is a Python computational framework that assists in the programmatic reconstruction, analysis and visualization of phylogenetic trees and multiple sequence alignments. It provides both a comprehensive API to interact with phylogenomic data, and a collection of general-purpose command-line tools.

- **eggNOG** (evolutionary genealogy of genes - Non-supervised Orthologous Groups) consists of a public database of phylogenomic data, and a set of open-source tools (e.g. eggNOG-mapper) for fast functional annotation of newly sequenced genomes and metagenomes. eggNOG resources use ETE libraries extensively for **i)** building the evolutionary histories of all gene families (4.4M phylogenetic trees in current eggNOG), **ii)** computing fine-grained orthology assignments (identifying *the same genes* over multiple genomes) across 5,000 reference species and **iii)** improving functional annotation of novel genes.

The purpose of this proposal is to develop an ETE Toolkit extension that allows for the visualization of genomic context (i.e. synteny) of the orthologous groups provided by eggNOG.

- **Languages and skills needed**
  - *Python (intermediate)*
  - Javascript (intermediate)
  - Visualization libraries, including low level drawing with SVG, Qt, etc.(intermediate/advance)
- **Difficulty**
  - Easy if you know ETEToolkit and have experience with general visualization libraries.
- **Mentors**
  - Jaime Huerta Cepas (jhcepas@gmail.com, @jhcepas)
  - Joaquín Giner Lamia (ginerorama@gmail.com)
  - Carlos Perez Cantalapiedra (cpcantalapiedra@gmail.com)
- **Student Benefits**
  - Full integration into a genomics research lab
  - Implementations would have a direct impact on well established resources
  - Training in bioinformatics

- **How to Apply**
    - Send (by email) resume, links to previous work and a short motivation letter.

Developing WellcomeML further for visualisation of academic research data

WellcomeML (https://github.com/wellcometrust/WellcomeML) currently contains a good set of utils for reading, processing, embedding, extracting entities, and classifying academic text data (publications, grants, and other documents) using machine learning. However the visualisation modules of that library are a bit thin. Some code has been developed in this space, including (BertViz) https://github.com/jessevig/bertviz, and other internal/external repositories, but they are very ad-hoc. There has been some discussions about standardising the way we visualise the results of that library, currently reflected on the following issues:

- #220: Meta-issue about all visualisation discussions
- #221: Visualising academic topics/clustering results
- #222: Visualising predictions of text classifiers

In this project, the student will be involved in the proposal of a new feature regarding one of the visualisation themes above. The project will be deemed successful if we end up with one (or more) RFC pull-request(s), and a set of examples using open academic research data, provided by the core maintainers.

Languages and skills needed

Python
Some interactive visualisation experience is desirable (e.g. plotl.ly/ dash, bokeh. https://docs.bokeh.org/en/latest/, hv https://holoviews.org)

Difficulty

Medium/Hard, it depends on how far into the visualisations the student wants to go.

Mentors

Antonio Campello (@aCampello)
Liz Gallagher (@lizgzil)
Nick Sorros (@nsorros)

Student benefits

The student will work with open-research data, be involved with a very active team on the topic through discussions/code reviews, RFCs PRs, and will be able to do hands-on work-on with python visualisation frameworks.

How to apply
Comment directly on one of the issues above (https://github.com/wellcometrust/WellcomeML) demonstrating interest in pursuing it. Alternatively, you can e mail one of the maintainers directly: (a.campello, e.gallagher, n.sorros)@wellcome.org.

Development of the Journal Code Policy Backend operations and routes (Code Is Science)

- **Project description**
    - In CodeIsScience, a journal code review policy database is under development, with its front/backend split in two different repositories (https://github.com/codeisscience/codecompliance-backend/ and https://github.com/codeisscience/codecompliance-frontend/). The code-compliance will be used to retrieve information about scientific journals and check which of them do or not have policies in respect to open-source code and how much they enforce it (for example, if it must be fully peer-reviewed or just partially, or even if there's any code peer-review).
- **Languages and skills needed**
    - Python (basics)
- **Difficulty**
    - Medium, easier if familiar with Flask
- **Mentors**
    - João Paulo Taylor Ienczak Zanette, (jpaulotiz@gmail.com, @jptiz)
    - Yo Yehudi, yochannah@gmail.com
- **Student Benefits**
    - Gain experience implementing and deploying hands-on flask applications based on a predefined specification, with small
- **How to Apply**
    - Please visit our GitHub repos above to comment on issues (primarily the backend one) or join the Slack to chat.

Mentor list

1. Michael R. Crusoe

2. Timo Sachsenberg (@timosachsenberg)
3. Julianus Pfeuffer (@jpfeuffer)
4. Oliver Alka (@oliveralka)
5. Jaime Huerta Cepas (jhcepas@gmail.com, @jhcepas)
6. Joaquín Giner Lamia (ginerorama@gmail.com)
7. Carlos Perez Cantalapiedra (cpcantalapiedra@gmail.com)
8. Saket Choudhary- saketkc@gmail.com
9. Amal Thomas- amalthomas111@gmail.com
10. Kasun Samarasinghe, kasun.wijesiriwardana@sib.swiss
11. Lydie Lane, lydie.lane@sib.swiss
12. Evan Nemerson <evan@nemerson.com>
13. Ng Zhi An (@ngzhian)
14. Yo Yehudi yochannah@gmail.com
15. Antonio Campello (@aCampello)
16. Liz Gallagher (@lizgzil)
17. Nick Sorros (@nsorros)
18. João Paulo Taylor Ienczak Zanette (jpaulotiz@gmail.com, jptiz)