This is a shared document to brainstorm project ideas for Codefest 2016 (http://www.open-bio.org/wiki/Codefest_2016). Feel free to also discuss ideas on the mailing list (https://groups.google.com/forum/#!forum/openbio-codefest-2016) or the chatroom (https://gitter.im/chapmanb/obf-codefest). Everyone is welcome to contribute ideas, and we'll use these to coordinate and organize ourselves at the start of Codefest.

Smaller project ideas for new attendees

This section is specifically targeted at new community members thinking of coming to Codefest. The ideas are aimed at being accomplishable tasks in a two day period that someone with minimal background in bioinformatics or the OpenBio community could work on. Ideas from any project or group are welcome.

Create a tutorial on Biostars

Biostars (https://www.biostars.org/) is a StackOverflow-like question and answer community. They have a Tutorial section (https://www.biostars.org/t/Tutorials/) with short descriptions of accomplishing specific tasks. These tutorials are incredibly useful and often found by Google searches when stuck. No task is too small to have a tutorial, so pick anything you're interested in learning about, ask at Codefest about how to get started doing it, and write up your results as you learn. Some example ideas: how do I get some human genome data to work with? How would I learn more about a gene I'm interested in? How do I find bioinformatics resources for the rice genome?

Add more quality control modules to MultiQC

MultiQC is a python package (http://multiqc.info and https://github.com/ewels/MultiQC/) that aggregates summary results from bioinformatics tools into a single report. For instance, it put together all FastQC output into a single and interactive HTML page. The idea of this project is to add any tools that are not already there, or improve the current ones. To add a new module that parses a new tool just need to add a python file like this multiqc/modules/featureCounts/feature_counts.py, modify search_patterns.py and config.py files inside multiqc/utils folder, and adapt accordingly (see the docs). For the event, you can work in any tool you would like to add to this package, or we will provide a list of tools, mainly the ones that are used by bcbio-nextgen pipeline framework. Or check this lis: https://github.com/ewels/MultiQC/milestone/6

Discussion of novel features for existing or future workflow authoring/execution tools

Several open-source projects develop workflow tools/execution engines (e.g., the Common Workflow Language, Nextflow, NextflowWorkbench [add to this list if you will represent a tool at

the meeting]). I for one (FC) would like to hear from users of workflow tools (i.e., pipeline programmers) what features they think would be useful and are missing from current workflow engine/tools. I think this discussion will be of interest to workflow tool developers as well as users (to help them shape the direction of future developments), and may help shape the direction where various tools are heading. It could also spur some collaborations among tool developers. We could author a short document summarizing what we learn. This is not per-se a coding project, but should still be of interest to some in attendance.

Scanning PubMed for Code

Development of a very simple prototype engine to scan PubMed abstracts for github, bitbucket or other opensource repos that are available.

General project ideas

This section is for project ideas that assume more experience or familiarity with projects. Everyone is welcome to contribute and all ideas, both ambitious and well-characterized, are encouraged.

Common Workflow Development in bcbio

Blue Collar Bioinformatics (bcbio: http://bcb.io/) is a collection of community build tools for biological analysis and includes cancer and germline variant calling, RNA-seq and small RNA analysis. We are actively moving bcbio infrastructure to use the Common Workflow Language (CWL: http://www.commonwl.org/) and have a working implementation (https://bcbio-nextgen.readthedocs.org/en/latest/contents/cwl.html). We'll be working on expanding this to work on more platforms and support all bcbio analysis. We welcome ideas for integration with a CWL-supporting platform, new pipelines to port to CWL, and active testing of the existing implementation.

Development of a Rating System for Usable and Accurate Documentation for (command line) Bioinformatics Software

+ Make it easier for people to provide really high quality documentation
+ Provide a rating system to rate documentation.
+ Incentivization of documentation and support for software in science & bioinformatics and
+ Easy to understand rating system.
+ Solicit participation of omicstools and/or Daniel Standage?

- Come up with a simple pipeline for error message -- documentation interaction.
    - Think about how to structure a publication identifying the need for new standard for “next-generation documentation”
        - Consider a model where a script throwing an error provides not just an error message but a link directly to the software wiki.

Enable interoperability between ADAM and Nextflow

ADAM (https://github.com/bigdatagenomics/adam) is a genomic analysis platform that takes advantage of distributed computing capabilities provided by Apache Spark framework and the HDFS file system.
Nextflow (http://www.nextflow.io) is a data-driven pipeline framework that simplifies the writing of complex parallel and distributed applications in a portable and reproducible manner.
Both ADAM and Nextflow are based on a functional-reactive programming model. However ADAM takes advantage of the scaling capability provided by the Apache Spark clustering technology, while Nextflow is more focused on fast prototyping of data analysis applications through easy integration of existing software tools and scripts.
The aim of this proposal is to enable interoperability between the two tools in such a way that would be possible to use Nextflow to compose ADAM tasks with other POSIX based genomic tools and run them in an optimal manner over the Spark platform.

Javascript-based viewers in BioJava

While BioJava is primarily intended for programmatic use, it is often useful to present data to users after an analysis. A strong example would be the structure comparison tool, which displays results of structural alignments using a Swing/Jmol interface.

With the obsolescence of applets, Java GUIs are no longer a viable way to interface with BioJava through a browser interface. Rather than requiring all client code to be written twice, it would make more sense to make HTML/Javascript the new default way to create GUIs. This could be enabled by embedding a lightweight HTTP server in a BioJava module, similar to how servers are shipped with many workflow engines, ipython notebooks, etc.

The goal of this project would be to create a new biojava module for launching the server, converting Java data structures to JSON, and perhaps some HTML/JS templates for common interfaces. The line between reusable code (belonging in BioJava) and application-specific code (belonging in client libraries) is likely to be blurry and would have to be discussed carefully.

See discussion at https://github.com/biojava/biojava/issues/529