Codefest 2013 Project Ideas

This shared document provides project ideas for OpenBio Codefest 2013 (http://www.open-bio.org/wiki/Codefest_2013) in Berlin on July 17th and 18th.

Please brainstorm ideas and directions that we'll use to organize development for the two days:

- /Finding and implementing the best scheduling mechanism for ratatosk, a luigi-based bioinformatics pipeline.
- Bioinformatics tools run time estimation web service (e.g.how long should aligning 40 million reads against hg19 with BWA take if I use 8 cores?"). Potentially collaborate with "Genome Comparison of Analytic Testing": http://www.bioplanet.com/forum/§discussion/7916/runtimeswallclock-time-alongside-the-accuracy-metrics#Item_1 www.tmzilla.co.uk

- Port IPython Parallel to DRMAA connector rather than separate PBS/SGE/Torque connectors.
- Train fellows on how to give and take from EasyBuild & HPCBIOS etfforts: http://hpcbios.readthedocs.org/en/latest/HPCBIOS_2012-94.html # bold ones are ready. Integrate as build option from CloudBioLinux.
- How can we make our workflows more interactive? (improve interactions between analysis and visualization tools?)
- How can ontologies (e.g. EDAM) help in the use of bioinformatics services (resource discovery, workflow construction, format conversions, ...)? BioCatalogue? (Stian) [work document here]
- Anybody interested in discussing provenance metadata for transparency, reproducibility, and preservation of results|workflows|scripts, independent of a programming|scripting|workflow language and environment|system|workbench? What does need to be recorded? Is there any concise standard? (Stian: W3C PROV? http://www.w3.org/TR/prov-overview/ Wf4Ever research objects and wfprov? http://purl.org/wf4ever/model/) www.tmzilla.com
- Rubra/Ruffus pipelines - I have some work to do on these, if others are using Ruffus it would be interesting to talk.
- How about R especially bioconductor packages?
    - Common parameter descriptions for automated BioConductor usage within Galaxy and Taverna workflows?
- BioRuby ideas:
    - Integrate NGS-related BioRuby/Biogems into Illumina's BaseSpace application with newly developed BaseSpace Ruby SDK (we can provide hands-on tutorial).
    - Improve or add more RDF converters and ontologies in BioInterchange.
    - Biogem for Maximum Likelihood Phylogenetics (e.g. RAxML wrappers), part of the code can be extracted from https://github.com/fizquierdo/perpetually-updated-trees (just started with an hello-raxml at https://github.com/fizquierdo/bioruby-raxml)
- Biopython ideas:
    - Migrate converstion of GFF files to Biopython objects from proposed standalone library (https://github.com/chapmanb/bcbb/tree/master/gff) to gffutils (https://github.com/daler/gffutils).
    - Is there someone who can give a talk about biopython library? [Peter Cock?]
- CloudBioLinux ideas:
    - Provide CloudBioLinux Docker images (http://docs.docker.io/en/latest/) and test lightweight deployments on multiple environments.
        - Add Galaxy tool and tool shed support for using docker images (CBL or otherwise) as tool dependencies.
    - Improve versioning of tools. Provide a lightweight library that can be used from other tools that reports versions and checks if up to date. Identify tools in CloudBioLinux built from raw

GitHub/Bitbucket repositories and lock into specific commits.
    - Improve installation on local machines, avoiding the requirement for ssh when building locally. Basic infrastructure is in place but has not been ported to all builds.
    - Look at using x2go for remote desktop instead of FreeNX
    - Refactor Galaxy tool shed repository installs to not require a running Galaxy instance, and then extend CloudBioLinux to install Galaxy dependencies via the tool shed. Idea outlined here: http://dev.list.galaxyproject.org/Galaxy-less-Tool-Installing-td4660655.html
    - Use environment modules to give access to Toolshed-installed tools (this is probably nearly done). Useful as there are multiple versions and they already have environment info specified.
- Data management, pains and ideas: Progress report on Google Docs
    - Download, indexing, updates, reindexing
    - Sounds easy, but neither computational grids nor clouds nor linux distros have yet managed to come up with a community-wide solution. We do not really care, do we?
    - Common resources under package management (e.g. wouldn't it be nice to be able to add "GRCh37" to your project's dependency list and have everything taken care of?)
- Linux distribution outreach
    - Bio-Linux, Debian und Ubuntu developers long joint forces and are attending the codefest (Tim, Olivier, Steffen)
    - OpenSuSE or Fedora anyone ?
    - Packaging of biotools like Taverna (Stian, Steffen, Tim)
- Better structural biology support (Spencer)
    - Biopython/pymol integration
    - refactor biojava structural alignment datastructures
- Make free versions of some popular existing tools (Torsten)
    - RNAmmer (ribosomal RNA identification from DNA sequence)
    - SignalP (signal peptide [Sec,TaT,..] from AA sequence)

Possible beer-o'clock discussions?

- BioDART (http://biodart.org/). i.e. core bioinformatics libraries for http://dartlang.org/) (probably not a Codefest project, but might be a good opportunity for a half-hour discussion if anybody else is interested).

Summary paragraphs

Barrnap - *Bacterial ribosomal RNA predictor* - Torsten Seemann & Tim Booth
For the last 8 years RNAmmer has been the standard tool for predicting ribosomal RNA features in genomes, because it is reasonably fast, accurate, and works on bacteria and eukaryotes. Its drawbacks are that it relies on small, older databases; requires an older conflicting version of HMMER; and has restrictive licence terms. To resolve these issues we have implemented a new rRNA predictor which uses the new "nmmer" tool from HMMER 3.1 for searching DNA profiles against DNA sequence. We used the Silva and GreenGenes seed alignments for the 5S, 23S and 16S genes to build the profile models from. Barrnap is a small Perl script which takes FASTA as input, and outputs the rRNA feature predictions in GFF3 format. It will be packaged in Bio-Linux and replace RNAmmer in the Prokka bacterial annotation system.

Visualisation - *Integrate DGE-Vis & Dalliance, JS animation scheduler* - David Powell, Thomas Down & Skyler Brungardt

We worked on integrating the Tom's genome browser, Dalliance, with David's RNA-seq explorer, DGE-Vis. We now have a proof-of-concept tool that makes it possible to visualise RNA-seq analysis while browsing the genome. Proof-of-concept demo available here, source on a github branch.

This inspired another project. We have the need for a JavaScript scheduler that is able to schedule slow animation updates when the JavaScript engine is not busy. This allows smoother animations and a better user experience by performing cheap animations while the user is interacting then the slower, more accurate updates when the browser is idle. The scheduler itself is mostly complete, with some bugs being ironed it. It'll live here: http://strictlyskyler.github.io/timeywimey/.

In between chasing bugs, Thomas worked with Alex Kalderimis to added a JBrowse-compatible JSON backend for Dalliance, for integration with, *e.g.* Intermine, and worked on Tabix and GFF support.

BioJVM - *Coordinating and integrating BioJava and ScaBio* - Spencer Bliven, Andreas Prlic, Markus Gumbel
Both Java and Scala run on the Java Virtual Machine. As such, it makes sense to coordinate the various Bio* projects which run on the JVM and therefore can interoperate to some degree. We are able to successfully reference BioJava functions from Scala code and ScaBio functions from Java code. Some documentation for this process was generated on the BioJava Wiki. The ease of this process means that users can easily use both libraries from whichever language is more suited for their biological problem.

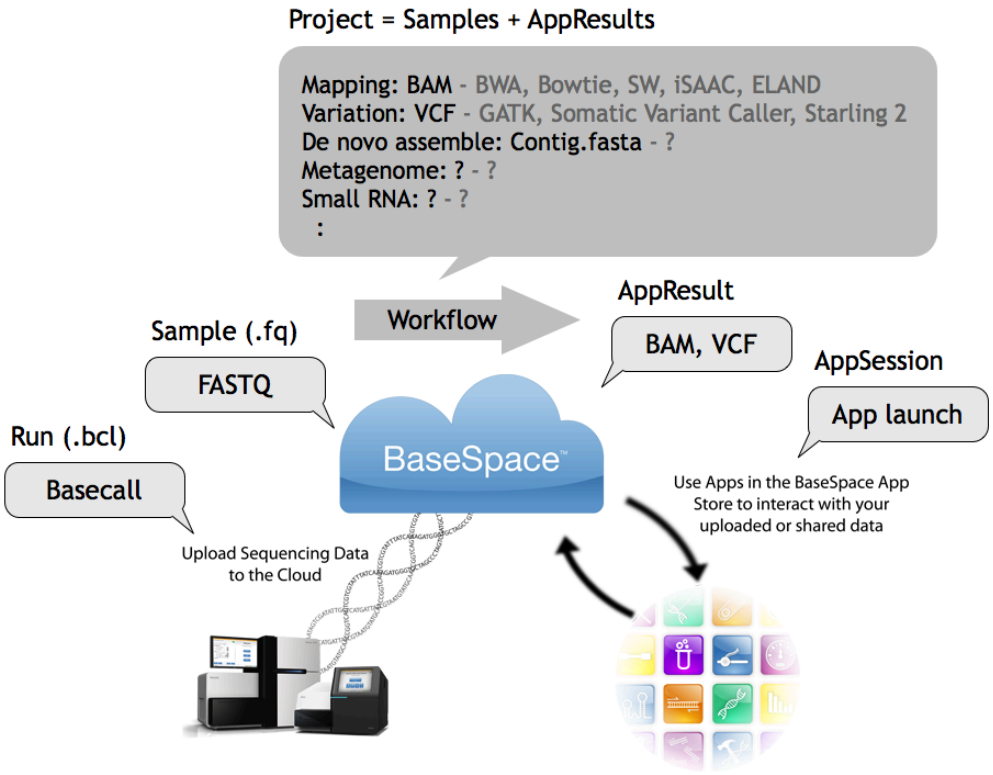
BioRuby/BaseSpace - *Develop SDK and apps for Illumina BaseSpace* - Toshiaki Katayama, Raoul Bonnal, Eri Kibukawa, Dan MacLean, Fernando Izquierdo-Carrasco, Spencer Bliven
We have been working on to port BaseSpace Python SDK to Ruby for past few months (available at https://github.com/joejimbo/basespace-ruby-sdk). During this codefest, we tested our Ruby SDK, created some mock apps, and finished documentations (Tutorial and RDoc).

Ruby/Biogem developers can create your own app
- You can easily utilize your NGS biogem w/ BaseSpace Ruby SDK
- You will easily obtain much more users

Users can use your app without coding
- Don't need to learn programming. Just a click!

BioBaseSpace for non-Ruby programmers

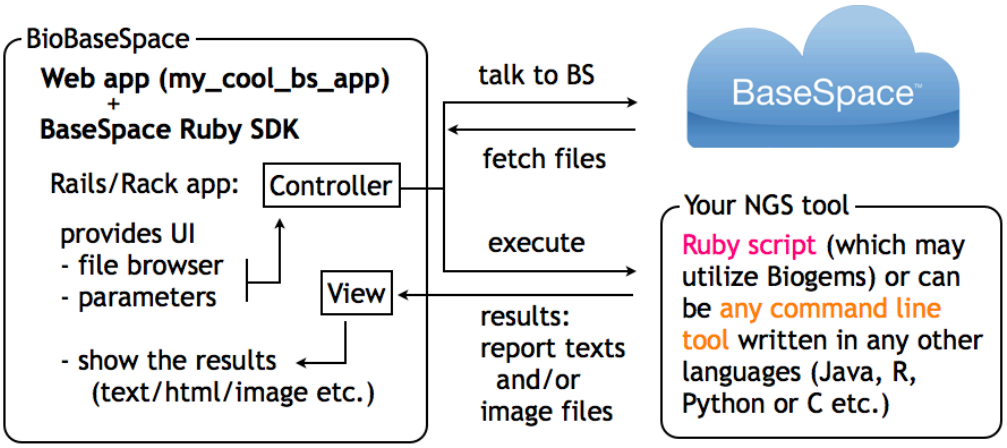
Then we found that it can be a burden to create new Web app from scratch on top of your NGS program. So we started new project to provide a Web-app scaffold for BS. We have already implemented the basic portion but will need some more time before releasing the app (BioBaseSpace).



BioBaseSpace

```
% biobasespace create my_cool_bs_app
% biobasespace deploy my_cool_bs_app --to (AWS|Heroku|others|localhost)
```

just configure the program and parameters to be executed in the app



ipython cluster runners - *Adding SLURM and DRMAA support to ipython and bcbio-nextgen* - Valentine Svensson, Guillermo Carrasco, Roman Valls

The motivation behind our specific task is to:

1. Implement basic SLURM support by understanding the already existing classes, which already

support SGE, LSF, Torque and Condor schedulers in ipython-cluster-helper.

2. Learning from that, introduce the use of the DRMAA connector, generalizing all the specific classes for the different job schedulers.
3. Ultimately port such generalization into ipython so that python scientific computations can be executed efficiently across different clusters around the world.

Please, have a look at the following posts to see what it turned out to be like:

http://mussolblog.wordpress.com/2013/07/17/setting-up-a-testing-slurm-cluster/
http://blogs.nopcode.org/brainstorm/2013/07/18/berlin-bosc-codefest-2013-day-1/

(Valentine will write a post in the future at some point)

The code to run SLURM jobs is availble here: https://github.com/roryk/ipython-cluster-helper/pull/6

One can select the number of workers one want, and specify how many compute-nodes these should be distributed over.

Here is the output from running the example with 10 workers over 2 machines (with 8 cores)

```
First check to see if we can talk to the engines.
This long computation that waits for 5 seconds before returning takes a while to run serially..
[('biologin.uppmax.uu.se',   60),   ('biologin.uppmax.uu.se',   63),   ('biologin.uppmax.uu.se',   66),
('biologin.uppmax.uu.se',   69),   ('biologin.uppmax.uu.se',   72),   ('biologin.uppmax.uu.se',   75),
('biologin.uppmax.uu.se',   78),   ('biologin.uppmax.uu.se',   81),   ('biologin.uppmax.uu.se',   84),
('biologin.uppmax.uu.se',   87),   ('biologin.uppmax.uu.se',   90),   ('biologin.uppmax.uu.se',   93),
('biologin.uppmax.uu.se',   96),   ('biologin.uppmax.uu.se',   99),   ('biologin.uppmax.uu.se',  102),
('biologin.uppmax.uu.se',  105),   ('biologin.uppmax.uu.se',  108),   ('biologin.uppmax.uu.se',  111),
('biologin.uppmax.uu.se', 114), ('biologin.uppmax.uu.se', 117)]
That took 20.0206918716 seconds.
Running it in parallel goes much faster...
[('q161.uppmax.uu.se',     60),     ('q166.uppmax.uu.se',     63),     ('q161.uppmax.uu.se',     66),
('q161.uppmax.uu.se',     69),     ('q166.uppmax.uu.se',     72),     ('q166.uppmax.uu.se',     75),
('q166.uppmax.uu.se',     78),     ('q161.uppmax.uu.se',     81),     ('q161.uppmax.uu.se',     84),
('q166.uppmax.uu.se',     87),     ('q161.uppmax.uu.se',     90),     ('q166.uppmax.uu.se',     93),
('q161.uppmax.uu.se',     96),     ('q161.uppmax.uu.se',     99),     ('q166.uppmax.uu.se',    102),
('q166.uppmax.uu.se',    105),     ('q166.uppmax.uu.se',    108),     ('q161.uppmax.uu.se',    111),
('q161.uppmax.uu.se', 114), ('q166.uppmax.uu.se', 117)]
That took 2.03281617165 seconds.
```

The example also demonstrates that both nodes are utilized.

Runtime metrics system - *Report runtime alongside system details for common bioinfo pipelines* - Per Unneberg

The aim of this project is to build a tool that works as the UNIX “time” command but augmented for bioinfo applications tools run time estimation web service (e.g. “how long should aligning 40 million reads against hg19 with BWA take if I use 8 cores?”). Potentially collaborate with “Genome Comparison of Analytic Testing”:

http://www.bioplanet.com/forum/§discussion/7916/runtimeswallclock-time-alongside-the-accuracy-metrics#Item_1

* Measure per organism

* Reference quality, number of reads, repetitiveness, application (RNAseq, variant calling, etc...)

Pipelines - rubra/ruffus-based variant calling pipeline - *GATK-based reusable pipeline based around Rubra/Ruffus* - Clare Sloggett, Bernie Pope (plus some not here)

This is a reusable pipeline for variant calling and annotation. It is based on BWA (alignment), GATK (alignment cleaning and variant calling) and ENSEMBL (annotation). It's up at
https://github.com/claresloggett/variant_calling_pipeline
It uses Rubra:
https://github.com/bjpop/rubra
which is based around the Ruffus library (not ours!): http://www.ruffus.org.uk/
At codefest we've been working on code cleanup, documentation and test data.
Also as part of the CloudBioLinux work (with Enis and John) I've been putting some things into the GVL flavour (https://github.com/afgane/gvl_flavor) of CBL to make these pipelines easier to run. These are

- rubra installed into CloudBioLinux
- environment modules installed into CloudBioLinux to give easy access to Galaxy-installed tools (not for non-Galaxy CBL tools)

Galaxy Debianisation - *Project description* - Tim Booth

I spent several hours revisiting previous work on the Galaxy package for Bio-Linux and made significant progress towards it being something that can go into Debian-proper. Results will be committed to Deb-Med public SVN and patches will be forwarded to the Galaxy dev mailing list.

Infrastructure management via CloudBioLinux (CBL) - *Project description*

Four topics were covered during the Codefest:

1. Integration of custom installation procedures present in CBL with the Galaxy-tools installation methodology. This means that tools can be installed in a versioned model so that Galaxy can readily use them.
2. General layout and an initial pass at new documentation for using CBL. Due to the increased interest by individuals to use and contribute to CBL, we've decided to invest some effort into creating purpose-driven documentation for CBL. This should help people use the endproduct of CBL, customize CBL their needs, as well as learn about the internals of CBL with the aim of contributing. The documentation will be fleshed out over the coming months and be hosted on ReadTheDocs.org.
3. In spirit of making CBL more accessible and easier to use, we've decided to tackle development of a lightweight webapp that helps with customizing and generating CBL configuration files. Initially, these will be downloadable as CBL favors, allowing a user to easily apply those.
4. Continuing on the simplification train, another aim is to create a simpler method for invoking CBL build framework. This will eliminate several manual steps and reduce complexity of typed commands.