This talk is accompanied by poster #1.

GenomeSpace: Open source interoperability platform with crowd-sourced analysis recipes

Ted Liefeld[1], Sara Garamszegi[2], Felix Wu[2], Marco Ocana[1], Barbara Hill[2], Helga Thorvaldsdottir[2], Michael Reich[1,], Jill Mesirov[1]

[1] University of California San Diego, La Jolla, CA, USA. Email: liefeld@ucsd.edu

[2] The Broad Institute of MIT and Harvard, Cambridge, MA, USA.

Project Website: http://www.genomespace.org
Source Code: https://bitbucket.org/GenomeSpace/combined/
License: GNU Lesser General Public License (LGPL).

Main Text of Abstract

GenomeSpace is a cloud-based environment that provides interoperability between best-of-breed computational tools, enabling scientists to easily combine tool capabilities without needing to program. It offers a common space to utilize, contribute, and share an ever-growing range of genomic analysis tools. GenomeSpace provides support for cloud-based data storage and analysis, multi-tool analysis scenarios, automatic conversion of data between tools, and ease of connecting new tools to the environment. GenomeSpace is free for all users and is open source under the GNU Lesser General Public License (LGPL).

The GenomeSpace architecture consists of (1) a server running in the Amazon cloud that manages communications and data transfer between tools, and data storage; (2) the collection of computational tools, updated to communicate with the GenomeSpace server while retaining their original user interfaces; and (3) connections to Amazon, Dropbox, and Google Drive cloud-based data services. The connection between server, tools, and data is provided by a RESTful API that can be easily adopted by new tools. A web-based user interface provides an integrated tool bar and data view from which users can launch tools, perform analyses, and transfer data. GenomeSpace also provides single-sign-on across compatible tools via OpenID.

We will describe the open source GenomeSpace platform and how it can be used to support genomic analyses that utilize multiple independent tools including GenePattern, Galaxy, Cytoscape, and IGV. We will also show how developers can add their own tools to the GenomeSpace ecosystem. We will also describe the newly released GenomeSpace Recipe Resource, a repository of short, standalone guides for performing integrative bioinformatic analyses. Each recipe walks users step-by-step through the process of obtaining and analyzing data across multiple GenomeSpace-enabled tools. The Recipe Resource provides the GenomeSpace community with the ability to create, share, and collaborate on analytic recipes of common value.

This talk is accompanied by poster #2.

Title	This is Why We Can Have Nice Things: Getting to 1.0 of the Common Workflow Language
Authors	Michael R. Crusoe
Contact	michael.crusoe@gmail.com
URLs	http://www.commonwl.org/
License	Apache License, Version 2.0

Common infrastructure that is usable by diverse participants does not come for free: it requires cooperation, patience, time, and care. When a community decides to invest its resources into creating and maintaining a common good, like F/OSS scientific software or interoperability standards, they can reap significant rewards: both from an academic/research perspective, and from a commercial market perspective.

How do we build communal goods that are 1) made in an open manner 2) not heavy-handed top-down projects 3) attentive to actual needs of others and 4) still useful?

As a follow-up to the debut at BOSC 2015 of a project that began at the BOSC 2014 Codefest, we are proud to present version 1.0 of the Common Workflow Language and the community of practitioners, implementors, vendors, academics, and businesses involved in its creation and maintenance.

In this talk we present an update on our activities this last year and how we (and the standards) changed along the way. The challenges outlined above will be reviewed alongside what we have done, or plane to do, to address them.

This talk is accompanied by poster #3.

Title	CWL in Practice: Experiences, challenges, and results from adopting Common Workflow Language
Authors	*Dan Leehr*, Alejandro Barrera, Tim Reddy, Hilmar Lapp
Affiliation	Duke University, Center for Genomic and Computational Biology
Contact	dan.leehr@duke.edu
URL	https://github.com/Duke-GCB/GGR-cwl
License	MIT

Assembling individual bioinformatics software packages to run together as a pipeline is often done by scripting them together in an implementation language like Bash, Python, or Perl. While simple to build and understand, this often yields pipelines that are difficult to repeat across computing environments, are not generalizable across data sets, and require nontrivial effort for tracking and archiving the exact versions of the tools that make up pipeline components. As a result, reusing, adapting, or even only repeating a bioinformatics analysis pipeline later in an environment other than its original one has become notorious for the difficulties involved. To address this problem, the Common Workflow Language (CWL) endeavors to provide a blueprint for architecting workflows declaratively. With open source implementations and a comprehensive specification, CWL provides a basis for building reusable components and reproducible, modular bioinformatics pipelines that facilitate swapping different analytic methods or algorithm implementations in and out. Because CWL is agnostic of workflow engine implementation, pipeline definitions do not need to be tied to a particular execution environment. CWL also encourages the use of Docker images for pipeline components, which not only isolate tools and their myriad of dependencies from each other, but also allows entire compute environments to be faithfully archived, shared, and reused.

Here we report on using CWL in support of the Genomics of Gene Regulation (GGR) project, a multi-institutional collaboration which aims to comprehensively characterize and better understand the first 12 hours of the glucocorticoid response (GCR). The lab of one of the project PIs at Duke, Tim Reddy, uses gene editing techniques to turn on and off genes, with the goal of elucidating the network of gene regulation involved in the GCR. This involves comparing the results of numerous genomic experiments in each condition, which requires developing data processing pipelines that can be confidently repeated, reused and customized, both by collaborators within the project, and by scientists at large wishing to reproduce and build on the team's findings.

We will highlight our experiences, challenges, and results from transforming the team's existing scripted workflows to workflows defined in the CWL standard. In comparison, GGR workflows migrated to CWL became more flexible, easier to modify, and amenable to replacing entire parts with alternative methods, which improved the project's analytic capabilities. Another benefit was the ability to reuse pipeline components shared between the analysis of ChIP-seq, RNA-seq, and DNase-seq experiments. Finally, publishing alongside a paper the CWL definitions of the underlying computational pipelines and the data sets allows others to reproduce the results, even though our specific high-performance computing environment, like most others, is not a public resource.

During the development of these CWL pipelines we have also faced two major challenges, and we will detail our approaches to them. One of them was translating the control flow and imperative scheduling logic found in the existing workflows expressed in a Turing-complete scripting language, to CWL's declarative language that is evaluated at compile time. The second challenge is the practical aspects of adopting CWL workflows in a shared academic HPC cluster environment. Our research center operates a computational genomics workload-tailored HPC cluster running Slurm as the schedular, and as a shared HPC environment users do not have the elevated privileges needed to run Docker containers or virtual machines, as CWL would favor. To address this, we have started to create pipeline component alternatives as loadable Environment Modules using helmod. In addition, we are extending toil, a CWL-supporting workflow engine that interfaces with job schedulers, to integrate with Slurm, which will allow us to better scale up CWL workflows to the available HPC resources.

Using the Common Workflow Language (CWL) to run portable workflows with Arvados and Toil

Peter Amstutz[1], Brian O'Connor[2], Benedict Paten[2], Alexander Wait Zaranek[1]

[1] Curoverse, Boston. peter.amstutz@curoverse.com, awz@curoverse.com
[2] University of California, Santa Cruz. broconno@ucsc.edu, benedict@soe.ucsc.edu

With special thanks to all contributors to Common Workflow Language, Toil, and Arvados.

Project Websites:
http://commonwl.org, http://arvados.org, https://toil.readthedocs.org/en/latest/
Source Code:
https://github.com/common-workflow-language/common-workflow-language,
https://github.com/curoverse/arvados, https://github.com/BD2KGenomics/toil
License:
https://github.com/common-workflow-language/common-workflow-language/blob/master/LICENSE.txt
https://github.com/curoverse/arvados/blob/master/COPYING
https://github.com/BD2KGenomics/toil/blob/master/LICENSE.txt

The Common Workflow Language (CWL) is a community effort that started at the BOSC Codefest 2014 to create a common specification for describing analysis tools and workflows that is portable and scalable across a variety of hardware and software platforms. Arvados is a cluster and cloud compute platform developed by Curoverse consisting of a content-addressed storage system "Keep", a compute management system "Crunch". Arvados can run in a variety of cloud and cluster computing configurations. Toil is a workflow management engine developed by the University of California Santa Cruz Genomics Institute. Toil also supports a number of cluster and cloud computing environments.

Some of the benefits of a common, community-developed language for computational workflows include the ability of scientists to collaborate and leverage the work of others, enhanced development of tooling around a common format, code that can "travel to the data" avoiding the need for large downloads and addressing legal issues around data migration, and ease of benchmarking methods across tools, techniques and platforms.

This talk will briefly introduce CWL and provide a project and community update since BOSC 2015. The talk will discuss the implementation of CWL on the Arvados platform and in the Toil workflow engine and how compatibility with CWL was achieved differences in the design of each system. The talk will discuss how to run workflows written in CWL on each system. The talk will then present results from running a representative genomics workflow written using CWL on both Arvados and Toil, using different cloud providers, with no workflow customization or porting required.

Planemo – A Scientific Workflow SDK

John Chilton[1], Aysam Guerler[2], and The Galaxy Team
[1]Department of Biochemistry and Molecular Biology, Penn State University, University Park, PA 16801, US
[2]Department of Biology, Johns Hopkins University, Baltimore, MD 21218, US
jmchilton@gmail.com
Project: http://galaxyproject.org/
Code: http://github.com/galaxyproject/planemo
License: Academic Free License version 3.0

Galaxy is a data analysis platform capable of integrating diverse command-line utilities into a consistent and intuitive web-based interface and API. A salient feature of Galaxy is the ability it provides to compose analysis steps together into workflows. A novel approach to building, refining, and running scientific workflows leveraging Galaxy through the command-line toolkit Planemo will be presented.

Traditionally there have been two methods to build Galaxy workflows - a graphical workflow editor and a workflow extraction interface. Both of these methods are great end-user facing tools that allows users with development experience to build workflows. However, sophisticated bioinformaticians and Galaxy plugin developers (e.g. tool developers) may prefer driving workflow development through their existing tool chains and methodologies such as programming text editors, command-line invocation, test-driven development, and revision control. The approach presented leverages YAML-based workflow descriptions as plain files allowing exactly this.

The approach will be used as a lens to highlight these workflows formats (Format 2 Galaxy workflows and Common Workflow Language (CWL) workflows) as well as important updates from the myriad of recent Galaxy workflow enhancements that have made them dramatically more usable, powerful, and performant.

Format 2 Galaxy workflows map directly to existing Galaxy tool and workflow concepts and are described in a very concise and readable YAML format. These will work without modifications to Galaxy today. CWL specifications for tools and workflows are developed in an open fashion by many organizations with the aim of creating truly portable descriptions. The execution of CWL workflows in Galaxy is being actively worked on and progress will be discussed.

These innovations are enabled in part by many core Galaxy enhancements. The most important of these enhancements will be highlighted. Including:

- The user interface for workflows has been overhauled and improved.
- Workflows now allow nesting, non-data inputs, implicit connections, and many new operations over collections - Galaxy workflows are now vastly more expressive.
- Recent performance enhancements allow Galaxy workflows to scale to thousands of datasets.

Sample Size Does Matter: Scaling Up Analysis in Galaxy with Metagenomics

Daniel Blankenberg[1, 2], Sarah Carnahan-Craig[3], and the Galaxy Team[2]
[1] Department of Biochemistry and Molecular Biology, Penn State University, University Park, PA 16802, USA. Email: dan@bx.psu.edu
[2] https://galaxyproject.org.
[3] Department of Biology, Penn State University, University Park, PA 16802, USA.

Project Website: galaxyproject.org **License**: Academic Free License version 3.0
Source Code: github.com/galaxyproject/galaxy

Galaxy (http://galaxyproject.org) is an open, web-based platform for accessible, reproducible, and transparent computational biomedical research. Galaxy makes bioinformatics analyses accessible to users lacking programming experience by enabling them to easily specify parameters for running tools and workflows. Analyses are made transparent by allowing users simple access to share and publish analyses via the web and create Pages, interactive, web-based documents that describe a complete analysis.

Metagenomics provides an exciting opportunity to begin to explore large-scale multiple sample analysis with Galaxy. As part of an obesity study, we have obtained over 400 buccal and stool samples from mother-child pairs. These samples have been subjected to 16S RNA extraction and sequencing on a MiSeq instrument. While sequencing 400 samples is no small feat, once generated, the data analysis reveals itself as crippling bottleneck.

Galaxy provides researchers with a vast quantity of tools and methods to analyze a wide-array of data, and makes connecting any number of tools together easy via Workflows. Although running a workflow individually over a handful of samples is approachable, how does one deal with 10, 20, or even 100 samples without becoming frustrated, introducing errors, breaking their mouse, or falling back to writing an API script? While Dataset Collection functionality provides a significant portion of a solution to this problem, there are still major hurdles that need to be overcome before Galaxy is usable for large multiple sample analysis.

Here we describe a generalizable metagenomic pipeline as implemented within Galaxy that is able to handle the simultaneous analysis of over 5,000 Human Microbiome Project samples. In addition to integrating a number of third-party algorithms and toolsets, some requiring the creation of upstream fixes and enhancements, we have developed new tools and approaches for dealing with large collections of data. Furthermore, we discuss the problems encountered using Galaxy at a large-scale, what has been done to overcome these issues, as well as initial results.

This talk is accompanied by poster #4.

NextflowWorkbench: Reproducible and Reusable Workflows for Beginners and Experts

Jason P. Kurs[1], Manuele Simi[1,2], Fabien Campagne[1,2,3,*]
[1]The HRH Prince Alwaleed Bin Talal Bin Abdulaziz Alsaud Institute for Computational Biomedicine, Weill Cornell Medicine, New York, NY, United States of America; [2]Clinical Translational Science Center, Weill Cornell Medicine, New York, NY, United States of America; [3]Department of Physiology and Biophysics, Weill Cornell Medicine, New York, NY, United States of America. *To whom correspondence should be addressed: fac2003@campagnelab.org

Keywords: Workflows, Pipelines, Reproducibility, Docker, Language Workbench Technology

Computational workflows and pipelines are often created to automate series of processing steps. For instance, workflows enable one to standardize analyses for large projects or core facilities, but are also useful for individual biologists who need to perform repetitive data processing.

Some workflow systems are designed for beginners: they offer a graphical user interface and have been very popular with biologists. In practice, these tools are infrequently used by more experienced bioinformaticians, who may require more flexibility or performance than afforded by the user interfaces, and seem to prefer developing workflows with scripting or command line tools.

The talk will introduce the NextflowWorkbench, a workflow system designed for both beginners and experts that blends the distinction between user interface and scripting language. This system extends and reuses the popular Nextflow workflow description language (http://nextflow.io) and shares its advantages. It is built using Language Workbench Technology, also used to develop the MetaR platform [1].

In contrast to Nextflow, NextflowWorkbench offers an integrated development environment that helps complete beginners get started with workflow development. Auto-completion helps users who do not know the syntax of the Nextflow language, and typesystem checks make it possible to validate a workflow as it is being developed, in order to provide immediate feedback to the developer. Expert bioinformaticians will also benefit from unique interactive features that help them work more productively with docker containers. Finally, reusable processes provide modular workflows, a feature useful to both beginners and experts.

We illustrate this tool with a workflow to estimate RNA-Seq counts using Kallisto. We found that beginners can be taught how to assemble this workflow in a two hours training session. The workflow can transparently run either on a laptop with docker, on a lab cluster, or in the cloud.

NextflowWorkbench simplifies the development of reproducible, implicitly parallel workflows [2]. Software is distributed under the Apache 2.0 license and available at https://github.com/CampagneLaboratory/NextflowWorkbench, http://workflow.campagnelab.org.

References: [1] Fabien Campagne, William ER Digan, Manuele Simi. MetaR: simple, high-level languages for data analysis with the R ecosystem. bioRxiv doi: http://dx.doi.org/10.1101/030254
[2] Jason P Kurs, Manuele Simi, Fabien Campagne. NextflowWorkbench: Reproducible and Reusable Workflows for Beginners and Experts. bioRxiv doi: http://dx.doi.org/10.1101/041236

Title	Promoting platform interoperability with portable bcbio workflows
Authors	*Brad Chapman*, Rory Kirchner, Lorena Pantano, Peter Amstutz, Alexander Zaranek, Shannan Ho Sui, Oliver Hofmann
Affiliations	Harvard Chan School Bioinformatics Core (http://bioinformatics.sph.harvard.edu/), Curoverse (https://curoverse.com/), Wolfson Wohl Cancer Research Centre (http://www.gla.ac.uk/researchinstitutes/cancersciences/ics/facilities/wwcrc/)
Contact	bchapman@hsph.harvard.edu
Availability	https://github.com/chapmanb/bcbio-nextgen
License	MIT

Running multi-step bioinformatics analyses requires coordinating software and data across a wide variety of heterogeneous computational resources. We've actively developed bcbio (https://github.com/chapmanb/bcbio-nextgen) for the past six years as a open, community built approach to developing variant calling, RNA-seq and small RNA analyses. The complexity of supporting scalable parallel workflows has become a barrier to allowing bcbio to interact with other open source platforms.

bcbio previously used a parallelization framework build on IPython parallel (https://ipyparallel.readthedocs.org) that runs on both local compute infrastructure (https://bcbio-nextgen.readthedocs.org/en/latest/contents/parallel.html) and on cloud resources (https://bcbio-nextgen.readthedocs.org/en/latest/contents/cloud.html). This approach unintentionally isolated bcbio development. For example, we could not easily deploy bcbio on community developed systems like Galaxy (https://galaxyproject.org/) due to different approaches to running compute jobs. This incompatibility results in duplication of effort as bcbio develops and tests system specific parallel code, while communities like Galaxy need to re-implement validated and tested analyses available in bcbio.

We re-engineered bcbio's internal workflow representation to use the Common Workflow Language (CWL: http://www.commonwl.org/). By using this community standard, users choose an infrastructure that matches their usage requirements. First build a bcbio parallel workflow directly from existing sample description files (https://bcbio-nextgen.readthedocs.org/en/latest/contents/cwl.html), then choose the appropriate run environment. A clinical lab requiring full data provenance could run the generated bcbio CWL using Arvados (https://arvados.org/). Research teams with local compute could use an alternative engine like Toil (https://github.com/BD2KGenomics/toil). By interoperating with other workflows, bcbio supplements existing infrastructure and analysis development within each system.

We'll discuss the challenges of migrating to CWL versus the benefits of being able to integrate within multiple platforms. bcbio is now a better architected, more portable set of validated tools and workflows to help scientists answer biological questions. We focus on developing analysis methods and validations while CWL supporting tools focus on other essential functionality like run tracking, multi-architecture support, resource usage assessment, provenance and data management. We hope to promote the continued exploration of ways to re-use and cooperate more effectively as an open source community.

Enhancements to MISO: An open-source community-driven LIMS

Heather Armstrong[1], Dillan Cooke[1], Tony DeBat[1], Andre Masella[1], Christopher Salt[2], Robert Davey[2], Morgan Taschuk[†1]

1 Ontario Institute for Cancer Research, Toronto, Canada

2 The Genome Analysis Centre, Norwich, UK

[†] Corresponding author

Project Website: http://tgac.github.io/miso-lims/
Source Code: https://github.com/TGAC/miso-lims/
License: GNU Public License v3

Laboratory Information Management Systems (LIMS) are most beneficial when they mirror lab procedures closely. Sequencing LIMS need to track sample receipt and preparation, record results from quality control, submit libraries for sequencing, and monitor results. Commercial LIMS require license fees and ongoing support contracts to keep software up-to-date and bug-free. New features are only considered once there is demand from a sufficient number of customers. Purchasing a commercial LIMS is often infeasible or impractical for smaller sequencing centres or those who focus on technical developments in sequencing practice. Using an open source LIMS allows the cost of updates to be distributed amongst its users. Developers can directly fix bugs and add new features as desired, and these benefits can be shared with the community without incurring further costs.

MISO is an open-source LIMS used and developed by several institutes in the UK, the Netherlands, and Canada. Since last presented at BOSC, we have expanded its functionality. Now, users can create and propagate detailed samples through customizable laboratory workflows. Web-based bulk entry is available in many places, avoiding the need to use external spreadsheets. Using the new order system, users can request the number and type of sequencing runs for each pool of samples and track outstanding requests. We have added support for barcoded storage tube boxes and integrated MISO with the Thermo Scientific VisionMate plate scanner. Sequencer information pages have detailed support history. Change logs now record user modifications on almost every entity. The bundled MISO analysis server can launch primary analysis processes both on LSF and Slurm. We have also made the codebase much more robust through intense bug fixing and increased test coverage, thus providing stability enhancements and simplified deployment.

MISO is available on Github at https://github.com/TGAC/miso-lims/ under GPL v3.

This talk is accompanied by poster #5.

Biothings APIs: high-performance bioentity-centric web services

Cyrus Afrasiabi[1], Jiwen Xin[1], Ginger Tsueng[1], Andrew I. Su[1*], Chunlei Wu[1*]

[1] The Scripps Research Institute, 10550 N Torrey Pines Rd, La Jolla, CA 92037

*Email: cwu@scripps.edu, asu@scripps.edu

Project Website: http://biothings.io, also see http://mygene.info and http://myvariant.info
Source Code: https://github.com/sulab/biothings.api
License: Apache License v2

The accumulation of biological knowledge and the advance of web and cloud technology are growing in parallel. The latest computation technology allows us to modernize the way we collect, organize and disseminate the growing biological knowledge. Just like what has been happening in the software-engineering field, biological data providers start to provide web-based APIs (Application Programming Interfaces) for accessing data in a simple and reliable manner, in addition to the traditional raw flat-file downloads. Web APIs provide many benefits over traditional file downloads. For instance, users can request specific data such as list of genes of interest without having to download the entire dataset, thereby providing the latest data on demand and reducing compute and data transfer times. Web APIs are also more likely to return data that conforms to common standards (e.g. JSON or XML). This means that programmers can spend less time on wrangling data, and more time on analysis and discovery.

Building and deploying scalable and high-performance web APIs requires sophisticated software engineering techniques that may not be known to many bioinformatics developers. We previously developed high-performance and scalable web APIs for gene and genetic variant annotations, accessible at MyGene.info and MyVariant.info. These two services are a tangible implementation of our expertise and collectively serve over 6 million requests every month from thousands of unique users. Crucially, the underlying design and implementation of these systems are in fact not specific to genes or variants, but rather can be easily adapted to other biomedical data types such drugs, diseases, pathways, species, genomes, domains and interactions, collectively, we refer them as "**BioThings**".

Based on the existing MyGene.info and MyVariant.info APIs, we now provide a generic framework (or called **BioThings SDK**) for building the same high-performance APIs for other BioThings data types. This SDK enables other developers to build their own BioThings API for their specific data types. Users can take advantage of the abstracted technical layers we built into the SDK, and produce a high-performance API, which follows the best practice and community standards. We also adopted JSON-LD technology to form the connections between different data types, so that the set of BioThings APIs will form a network of linked programmatic-accessible biological knowledge. As these BioThings APIs are preferred to be built under the cloud environment, the BioThings SDK is essentially serving as a component of "Software as the Service".

This talk is accompanied by poster #6.

The Noctua Modeling Tool

Seth Carbon*, Heiko Dietze*, Chris Mungall*

17th Bioinformatics Open Source Conference (BOSC) 2016, Orlando, FL, USA

Website: http://noctua.berkeleybop.org
Repository: https://github.com/geneontology/noctua
License: BSD 3-Clause

We present Noctua, a modern web application and stack for modeling complex biology. Noctua directly models information as a graph, escaping many of the pitfalls of more "tabular" modeling. Noctua also presents a rich, interactive, and collaborative user interface, as well as a complete set of tooling for data extraction and integration.

The Gene Ontology [1] project aims to create a comprehensive and up-to-date model of biological systems based on annotating knowledge graphs (ontologies) with curated and generated information. Over its existence, the project has aimed to capture this information with ever increasing richness and specificity. The traditional storage format for GO annotations, the tabular GAF, has undergone several iterations, but can no longer support the types of annotations required by current use cases. To move forward, the project has adopted LEGO, a graph-base abstraction for modeling biology. The Noctua application and stack was created to annotate information with the LEGO abstraction, but has the built-in flexibility to be used in any number of pathway or workflow models.

The current main end user client application for Noctua presents a real-time collaborative graph editing environment, allowing users to assemble graphs representing biological knowledge, including aspects such as references and evidence. The user environment uses typed inputs combined with a click, drag, and connect interface for easy and intuitive graph editing. As multiple users work on the same model, no matter the client, it updates the common environment in real time, allowing for easy discussion, presentation, and collaboration.

The Noctua stack is composed of three layers: the client, written in JavaScript using the jsPlumb and AmiGO/BBOP libraries; the communication layer, written in JavaScript and providing client-to-client and client-to-server communication; and the graph engine, a Java server that uses OWL to model and OWL universe tools for reasoning. This stack is strongly separated, respecting protocol and common patterns. For example, while the main user interface is a graph editor, it could be easily replaced by a different client that could speak the same wire protocol–the flexibility of the framework allows for the easy creation of alternate clients, such as the form or REPL based ones.

The Noctua stack is being actively used by the Gene Ontology, with the produced annotations finding their way back into the pipelines of several model organism databases. As well, Noctua data produced for the GO is loaded into AmiGO [2], where it is available for exploration and made available to be consumed in clients via our JavaScript API. The data modeling and retrieval systems for the client are available as separate packages, allowing third parties to create their own clients or embed annotation widgets in their own resource pages.

References

[1] Gene Ontology Consortium: going forward. *Nucl. Acids Res.* 43, D1049D1056.

[2] AmiGO: online access to ontology and annotation data. *Bioinformatics* 25, 288289.

*Berkeley Bioinformatics Open-source Projects, Lawrence Berkeley National Lab, Berkeley, CA, USA.

Processing phenotype data using Phenopackets-API and PXFTools

Christopher J Mungall[1*], Jules Jacobsen[2*], James Balhoff[3], Jeremy Nguyen-Xuan[1], Kent Shefchek[4], Dan Keith[4], Harry Hochheiser[5], Suzanna E Lewis[1], Sebastian Köhler[6], Peter Robinson[6], Julie McMurray[4], Tudor Groza[7], Melissa Haendel[4]

[1] Lawrence Berkeley National Laboratory, Berkeley, CA, USA. Email: cjmungall@lbl.gov

[2] Sanger Institute, Hinxton, UK.

[3] RTI International, Durham, NC, USA.

[4] Oregon Health and Sciences University, Portland, OR, USA.

[5] University of Pittsburgh, Pittsburgh, PA, USA.

[6] Charité – Universitätsmedizin Berlin, Germany.

[7] Garvan Institute of Medical Research, Sydney, Australia

*These authors contributed equally.

Project Website: http://phenopackets.org
Source Code: https://github.com/phenopackets
License: Code: BSD-3. Format specification and documentation: CC-BY-3

While great strides have been made in exchange formats for genomic sequence and variation data (e.g. Variant Call Format; VCF), the same is not true for phenotypic features. Similarly, bioinformatics software libraries such as BioPython and BioPerl have rich object models for genomic or phylogenetic datatypes, but lack a uniform phenotype representation. This is due in part to the diversity of phenotypic descriptions, from clinical observations through QTLs and newer high-throughput phenopype measurements. As a result, phenotypes are represented differently in different databases, making it harder to exchange, aggregate and operate over phenotypic data.

We have designed a datamodel and exchange format standard for flexible, extensible and expressive representation of a broad range of phenotypes in humans or any other species. The Phenotype eXchange Format (PXF) works hand-in-hand with phenotype ontology such as the Human Phenotype Ontology or the Ontology of Biological Attributes, but allows for representation of other fields such as quantitative measurements, environments and evidence. PXF can be serialized as either JSON or YAML, and we provide JSON schema for validation. We also provide a JSON-LD context for use within semantic web and OWL stacks.

For software developers we provide the Phenopacket Application Programmer Interfaces (APIs) in Java, Python and Javascript. We have implemented bindings for Neo4J via SciGraph (https://github.com/SciGraph/SciGraph), but the model is storage-layer independent and can potentially be used in conjunction with GMOD databases such as Chado. We are also prototyping a language-independent ProtoBuf-API to facilitate interoperability with the Global Alliance for Genomes and Health (GA4GH) stack.

As a demonstration of purpose, we have implemented a command line tool library called pxftools, analogous to the popular vcftools used for operating over variant files.

This talk is accompanied by poster #7.

The EDAM Ontology

Jon Ison[1], Hervé Ménager[2], Matúš Kalaš[3], EDAM contributors

[1] Department of Systems Biology, Technical University of Denmark, Kongens Lyngby, Greater Copenhagen, Denmark. Email: jison@cbs.dtu.dk

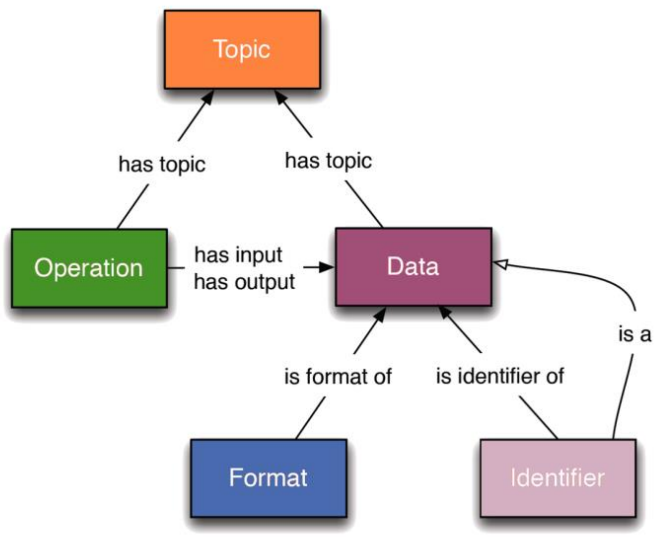
[2] Institut Pasteur, Paris, France.

[3] Computational Biology Unit, Department of Informatics, University of Bergen, Bergen, Norway.

Project Website: http://edamontology.org
Source Code: https://github.com/edamontology/edamontology
License: CC BY-SA 4.0

Bioinformaticians handle an increasingly large and diverse set of tools and data. Meanwhile, researchers demand ever more powerful and convenient means to organise, find, understand, compare, select, use and connect the available resources. These tasks often rely on consistent, machine-understandable descriptions of the underlying components, but these have been generally lacking in *ad hoc* resource descriptions. The urgent need - filled by EDAM - is for an ontology that unifies semantically the bioinformatics concepts in common use, provides the curator with a comprehensive nomenclature that is broadly applicable, and enables new and powerful search, browse and query functions.

EDAM is an ontology of well established, familiar concepts that are prevalent within bioinformatics, including types of data and data identifiers, data formats, operations and topics. EDAM is a simple ontology – essentially a set of concepts with terms, synonyms, definitions, and relations. EDAM is organised into an intuitive hierarchy for convenient use by curators, software developers, and end-users.



5 consecutive stable versions of EDAM have been released since July 2015, with version 1.14 being the current one at the time of submission. EDAM is developed in a participatory and transparent fashion, with a growing community of contributors, connected *e.g.* to the **Bio.Tools** registry (http://bio.tools) of bioinformatics tools and data services, or to the development of various bioinformatics workbenches.

This talk is accompanied by poster #8.

Title	Towards traceable, scriptable, and efficient data distribution for next-generation genomics
Author	*John Bradley*, Dan Leehr, Erich S. Huang, Jonathan Turner, Hilmar Lapp
Affiliation	Duke University, Center for Genomic and Computational Biology
Contact	john.bradley@duke.edu
URLs	https://github.com/Duke-GCB/DukeDSClient https://github.com/Duke-GCB/DukeDSHandoverService
License	MIT

At many research institutions, next-generation genomics data starts their lifecycle at a core facility. Depending on the type of core this may be as primary data generated by instruments, or as secondary data generated by analysis or other processing. From there, data will be handed over to a principal investigator, who then derives scientific conclusions underpinned by the data and publishes them in scholarly journals. To be consistent with open and reproducible science principles, this lifecycle creates a number of challenges, many of which can be traced back to the shortcomings of distributing and sharing data via traditional, yet still very common, networked block storage. In particular, (1) lifecycle progression, including key steps such as handing over data from core to investigator, and depositing data for permanent archival in a repository, is difficult or impossible to formalize as a documented and programmable transaction; (2) the metadata, which importantly includes tracing a data object's entire provenance chain, are difficult to track and tie to the data; (3) versioning of data, including identifying in a globally unique way the data used for each analysis step, is highly cumbersome at best. These challenges are further exacerbated by the large volume of data generated by ever-evolving next-generation genomics technologies.

To enable these challenges to be addressed in a principled way, an interdisciplinary team at Duke University is building an open-source informatics infrastructure for managing and tracking data at scale through its lifecycle, called the Duke Data Service (DDS). DDS is inspired by years of experience developing and using SAGE Bionetworks' Synapse[1], comes with an extensive API deployed publicly on the Heroku cloud, and for storing, versioning, and identifying data federates across cloud-based and on-premise object store service APIs, including AWS S3 and OpenStack Swift.

Here we report on our work using this infrastrucure to build scalable command line and user-interface tools for digital genomics data during the lifecycle period from generation in a core facility to hand-off to investigator. We highlight how the tools efficiently and traceably ingest and register data and their provenance; how investigators can receive the data on high-performance computing environments; and how the object store API enables parallelization of data streaming to better saturate available I/O bandwidth for high-volume data. Finally, we will show how a web-based user-interface implemented on top of this infrastructrure can formalize the hand-over of data from core to investigator as a documented transaction that signifies the passing of data stewardship from one party to another.

References:

1. Omberg, Larsson, Kyle Ellrott, Yuan Yuan, Cyriac Kandoth, Chris Wong, Michael R. Kellen, Stephen H. Friend, Josh Stuart, Han Liang, and Adam A. Margolin. 2013. "Enabling Transparent and Collaborative Computational Analysis of 12 Tumor Types within The Cancer Genome Atlas." Nature Genetics 45 (10): 1121–26.

This talk is accompanied by poster #9.

Mango: data exploration on large genomic datasets

Alyssa Morrow[1], Eric Tu[2], Frank Austin Nothaft[3], Anthony Joseph[4], David Patterson[5]

[1] University of California-Berkeley, United States. Email: akmorrow@berkeley.edu

[2] University of California-Berkeley, United States. Email: erictu@berkeley.edu

[3] University of California-Berkeley, United States. Email: fnothaft@berkeley.edu

[4] University of California-Berkeley, United States. Email: adj@berkeley.edu

[5] University of California-Berkeley, United States. Email: pattrsn@cs.berkeley.edu

Project Website: http://bdgenomics.org/

Source Code: https://github.com/bigdatagenomics/mango

License: Apache License 2.0 (see http://www.apache.org/licenses/LICENSE-2.0.txt)

Current genomics visualization tools are intended for a single node environment and lack the scalability required to visualize multiple whole genome samples. Data from the 1000 Genomes Project provides 1.6 terabytes of variant data and over 14 terabytes of alignment data. However, typical genomic visualizations materialize less than 10 kbp, only 3.3e-7% of the genome. Mango is a visualization browser that selectively materializes and organizes genomic data to provide fast in-memory queries. Mango materializes data from persistent storage as the user requests different regions of the genome. This data is efficiently partitioned and organized in memory using interval trees. This interval based organizational structure supports ad hoc queries, filters, and joins across multiple samples at a time, enabling exploratory interaction with genomic data.

Mango is built on top of Spark and ADAM, both open source projects under the Apache license. Leveraging Spark as Mango's cluster computing framework enables scalable, distributed computations on terabytes of genomic data. Mango leverages ADAM's genomic file formats which can be stored in persistent storage and accessed by Spark. Both ADAM and Mango are part of the Big Data Genomics project at University of California-Berkeley. Mango is published under the Apache 2 license.

ADAM Enables Distributed Analyses Across Large Scale Genomic Datasets

Frank Austin Nothaft[1,2], Arun Ahuja[3], Timothy Danford[1,4], Michael Heuer[1], Jey Kottalam[1], Matt Massie[1], Audrey Musselman-Brown[5], Beau Norgeot[5,6], Ravi Pandya[7], Justin Paschall[1], Jacob Pfeil[5], Hannes Schmidt[5], Eric Tu[1], John Vivian[5], Ryan Williams[3], Carl Yeksigian[8], Michael Linderman[3], Jeff Hammerbacher[3], Uri Laserson[3,9], Gaddy Getz[10], David Haussler[5], Benedict Paten[5], Anthony D. Joseph[1], David A. Patterson[1,2]

Website: http://bdgenomics.org
Repositories: https://github.com/bigdatagenomics/adam and https://github.com/BD2KGenomics/toil-scripts
License: Apache 2 License

The detection and analysis of rare genomic events requires integrative analysis across large cohorts with terabytes to petabytes of genomic data. Contemporary genomic analysis tools have not been designed for this scale of data-intensive computing. This abstract presents recent updates to ADAM, an Apache 2 licensed library built on top of the popular Apache Spark distributed computing framework. We are using ADAM and the Toil workflow management system (Apache 2 licensed) to recall the Simons Genome Diversity project dataset against the GRCh38 build of the human reference genome. Because ADAM is designed to allow genomic analyses to be seamlessly distributed across large clusters, we achieve a $3.5\times$ improvement in end-to-end variant calling latency and a 66% cost improvement over current toolkits, without sacrificing accuracy.

On top of our previous results [1], we have achieved an additional 2–$3\times$ improvement in performance by modifying the schemas that ADAM uses. In addition to improving performance, these modifications address the problem of storing and processing metadata (e.g., information about a reference genome build) inline with an analysis. To run our workflow at large scale, we use the Toil workflow system. Toil is a system for running arbitrary computation that can be structured as a directed acyclic graph across various different schedulers. Toil provides fault tolerance, and supports computational reproducibility and portability. We build upon Toil to enable dynamic scaling of Spark clusters using the AWS spot market. Our end-to-end variant calling pipeline allocates machines on an as-needed basis to improve cost effectiveness, and allows the use of Apache Spark in a workflow with traditional, single node bioinformatics tools.

References

[1] F. A. Nothaft, M. Massie, T. Danford, Z. Zhang, U. Laserson, C. Yeksigian, J. Kottalam, A. Ahuja, J. Hammerbacher, M. Linderman, M. Franklin, A. D. Joseph, and D. A. Patterson. Rethinking data-intensive science using scalable analytics systems. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data (SIGMOD '15)*. ACM, 2015.

[1]AMPLab, University of California, Berkeley, CA
[2]ASPIRE Lab, University of California, Berkeley, CA
[3]Icahn School of Medicine at Mount Sinai, New York, NY
[4]Tamr, Cambridge, MA
[5]Genome Informatics Lab, University of California, Santa Cruz, CA
[6]University of California, San Francisco, CA
[7]Microsoft Research, Redmond, WA
[8]GenomeBridge, Cambridge, MA
[9]Cloudera, Inc., San Francisco, CA
[10]The Broad Institute of MIT and Harvard, Cambridge, MA
Correspondence should be addressed to fnothaft@berkeley.edu.

SUPERSMART - A Self-Updating platform for Estimating Rates of Speciation and Migration, Ages, and Relationships of Taxa

Hannes Hettling [*] Alexandre Antonelli [†, ‡] Rutger A. Vos [*]

17th Bioinformatics Open Source Conference (BOSC) 2016, Orlando, FL, USA

Website: http://www.supersmart-project.org
Repository: https://github.com/naturalis/supersmart
License: MIT license (see https://opensource.org/licenses/MIT)

Background

Massive volumes of biological data, including molecular sequences and fossils, are accumulating quickly and are frequently made available in public repositories. This wealth of data facilitates large-scale quantitative analyses reveiling the evolutionary mechanisms that generate biodiversity. However, exploiting the data to generate time-calibrated phylogenies that represent the evolutionary relationships and history among species is generally a time-consuming and error-prone procedure involving a variety of different analysis tools.

Results

Here we present the SUPERSMART *virtual research environment* which provides an integrative solution for automated mining, cleaning and assembly of molecular sequencing data and the inference of large, time-calibrated species phylogenies. Phylogenetic inference for large numbers of taxa and sequence data is accomplished by a recursive divide-and-conquer approach implemented in our platform, engineered for massive parallelization. Our software is therefore scalable to build phylogenies for tens of thousands of taxa. SUPERSMART comprises state-of-the-art analysis tools for taxonomic name resolution, sequence alignment (e.g. *muscle, mafft*), maximum likelihood and Bayesian tree inference (*RaXML, ExaML, ExaBayes, *BEAST*) and molecular dating (*TreePL, *BEAST*). Deployment of the platform including the above tools is accomplished using the virtualization frameworks *Docker* and *Vagrant*, rendering the installation of any dependencies unnecessary. A graphical user interface to SUPERSMART is provided via *Galaxy* web services. The core of SUPERSMART is implemented in Perl/BioPerl in a highly modular fashion. In the future, we aim to integrate single SUPERSMART modules, such as wrappers for the above analysis tools, into the OBF BioPerl library.

Conclusions

SUPERSMART is an integrated platform for data mining and phylogenetic inference workflows. From simply a list of taxa of interest, users can generate dated species phylogenies in merely a few steps. Our software is easily deployable, platform independent and can run on personal computers and high-performance cluster infrastructures. SUPERSMART is originally a command-line application but can alternatively be accessed through *Galaxy*. Thereby, we give low-threshold access to phylogenetic inference workflows to researchers with little computer expertise.

[*]Naturalis Biodiversity Center, Darwinweg 4, 2333 CR Leiden, the Netherlands. Email: hannes.hettling@naturalis.nl

Title: Characterization of the small RNA transcriptome using the bcbio-nextgen python framework

Authors: Lorena Pantano, Brad Chapman, Rory Kirchner, John Hutchinson, Oliver Hofmann, Shannan Ho Sui
Affiliations Harvard T.H. Chan School of Public Health, Boston. Wolfson Wohl Cancer Research Centre
Contact: lpantano@hsph.harvard.edu
Project: https://github.com/chapmanb/bcbio-nextgen
License: MIT

The study of small RNA helps us understand some of the complexity of gene regulation of a cell. Of the different types of small RNAs, the most important in mammals are miRNA, tRNA fragments and piRNAs. The advantage of small RNA-seq analysis is that we can study all small RNA types simultaneously, with the potential to detect novel small RNAs. bcbio-nextgen is a community-developed Python framework that implements best practices for next-generation sequence data analysis and uses gold standard data for validation. We have extended bcbio to include a small RNA-seq analysis pipeline that performs quality control, removal of adapter contamination, annotation of miRNA, isomiRs and tRNAs, novel miRNA discovery, and genome-wide characterization of other types of small RNAs. The pipeline integrates tools such as miRDeep2[1], seqbuster[2], seqcluster[3] and tdrMapper[4] to facilitate annotation to small RNA categories. It produces a R Markdown template that helps with downstream statistical analyses in R, including quality control metrics and best practices for differential expression and clustering analyses. Finally, the pipeline generates an interactive HTML-based browser for visualization purposes. This is useful for characterizing novel small RNA types, working with non-model organisms, or providing a general profiling description. This browser shows the small RNA regions along with their genomic annotation, expression profile over the precursor, secondary structure, and the top expressed sequences. Here, we show the capabilities of the pipeline and validation using data from the miRQC project. We show that the quantification accuracy is around 95% for miRNAs. We obtained similar results for other types of small RNA molecules, demonstrating that we can reliably detect small RNAs without a dependency on specific databases.

[1] Friedlander, M. R., MacKowiak, S. D., Li, N., Chen, W., & Rajewsky, N. (2012). MiRDeep2 accurately identifies known and hundreds of novel microRNA genes in seven animal clades. Nucleic Acids Research, 40(1), 37–52.

[2] Pantano, L., Estivill, X., & Martí, E. (2010). SeqBuster, a bioinformatic tool for the processing and analysis of small RNAs datasets, reveals ubiquitous miRNA modifications in human embryonic cells. Nucleic Acids Research, 38(5), e34

[3] Pantano, L., Friedlander, M. R., Escaramis, G., Lizano, E., Pallares-Albanell,

This talk is accompanied by poster #10.

MetaR: simple, high-level languages for data analysis with the R ecosystem

Fabien Campagne[1,2,3,*], William ER Digan[1], Manuele Simi[1,2]
[1]The HRH Prince Alwaleed Bin Talal Bin Abdulaziz Alsaud Institute for Computational Biomedicine, Weill Cornell Medicine, New York, NY, United States of America; [2]Clinical Translational Science Center, Weill Cornell Medicine, New York, NY, United States of America; [3]Department of Physiology and Biophysics, Weill Cornell Medicine, New York, NY, United States of America. *To whom correspondence should be addressed: fac2003@campagnelab.org

Keywords: Data Analysis, Reproducibility, Docker, Language Workbench Technology

Data analysis tools have become essential to the study of biology. Here, we applied language workbench technology (LWT) to create data analysis languages tailored for biologists with a diverse range of experience: from beginners with no programming experience to expert bioinformaticians and statisticians.

A key novelty of our approach is its ability to blend user interface with scripting in a single platform. This feature helps beginners and experts alike analyze data more productively.

This new approach has several advantages over state of the art approaches currently popular for data analysis: experts can design simplified data analysis languages that require no programming experience, and behave like graphical user interfaces, yet have the advantages of scripting. We report on such a simple language, called MetaR [1], which we have used to teach complete beginners how to call differentially expressed genes and build heatmaps. We found that beginners can complete this task in less than 2 hours with MetaR, when more traditional teaching with R and its packages would require several training sessions (6-24hrs). Furthermore, MetaR seamlessly integrates with docker to enable reproducibility of analyses and simplified R package installations during training sessions.

We used the same approach to develop the first composable R language. A composable language is a language that can be extended with micro-languages. We illustrate this capability with a Biomart micro-language designed to compose with R and help R programmers query Biomart interactively to assemble specific queries to retrieve data, (The same micro-language also composes with MetaR to help beginners query Biomart.)

Our teaching experience suggests that language design with LWT can be a compelling approach for developing intelligent data analysis tools and can accelerate training for common data analysis task. LWT offers an interactive environment with the potential to promote exchanges between beginner and expert data analysts. This talk will provide an introduction to LWT and describe our results with the MetaR platform [1,2].

Software is distributed under the Apache 2.0 license and available at https://github.com/CampagneLaboratory/MetaR and http://metaR.campagnelab.org.

References: [1] Fabien Campagne, William ER Digan, Manuele Simi. MetaR: simple, high-level languages for data analysis with the R ecosystem. BioRxiv doi: http://dx.doi.org/10.1101/030254 [2] Documentation http://tinyurl.com/zx9wvjw

This talk is accompanied by poster #11.

Development of NGSEP as an open-source comprehensive solution for analysis of high throughput sequencing data

Juan Fernando de la Hoz, Juan David Lobaton, Claudia Perea, Daniel Felipe Cruz, Juan Camilo Quintero, Paulo Izquierdo, Bodo Raatz and Jorge Duitama

[1] International Center for Tropical Agriculture (CIAT), Cali, Colombia. Email: j.duitama@cgiar.org

Project Website: https://sourceforge.net/p/ngsep/wiki/Home/
Source Code: https://sourceforge.net/projects/ngsep/files/SourceCode/
License: GNU General Public License, version 3 (GPL-3.0) (https://opensource.org/licenses/GPL-3.0)

The development and availability of high throughput sequencing (HTS) technologies revolutionized the research on genomics allowing to obtain genome-wide data on entire populations of nearly every form of life. A key step to extract relevant information from HTS data was the development of open-source software tools to perform different bioinformatic analyses. However, although even small labs are now able to efficiently produce large amounts of HTS data, comprehensive analysis of these data integrating different solutions remains a challenging task. We initially developed NGSEP as an open-source package that tightly integrates novel java implementations of algorithms for discovery of single nucleotide variants (SNVs), indels, and copy number variants (CNVs), called from a rich interface implemented in an Eclipse Plugin as well as basic command line usage. We built several functions to facilitate users processing HTS reads and genotype calls, including a one step wizard for parallel automated processing of entire populations, genotype filters and statistics, imputation for inbred populations and format conversion for integration with tools for assessment of population structure, GWAS, genomic prediction, among others. Benchmark using first Whole Genome Sequencing (WGS) data from human, yeast and rice samples and later Genotype by Sequencing (GBS) data from cassava and bean populations showed that NGSEP provides similar or better accuracy for SNP detection and genotyping compared to other tools such as GATK, Samtools and Tassel. NGSEP is now a useful software package for different research groups, as demonstrated by download statistics and recent scientific publications.

Continuing our efforts, we implemented in NGSEP three new algorithms to find structural variants (SVs) from WGS data: CNV-seq, for read-depth comparison between two samples; EWT to detect small CNVs; and a novel implementation of the read-pair and split-read strategies for detection of large indels with breakpoint resolution. Experiments with simulated data show that these functions provide similar accuracies compared to other tools such as Delly or Pindel. We now re-branded NGSEP as Next Generation Sequencing Experience Platform because we expanded the interfaces of NGSEP improving its documentation for command line usage and its integration with the Galaxy environment. We also made NGSEP available within the CyVerse (former iPlant) platform and we integrated the variants detector within the DNAnexus platform for SV discovery in the 3000 rice genomes project. We expect that all these development efforts facilitate the use of NGSEP for a growing number of researchers in different fields of basic and applied genomics.

This talk is accompanied by poster #12.

GRNmap and GRNsight: open source software for dynamical systems modeling and visualization of medium-scale gene regulatory networks

Kam D. Dahlquist[1], Ben G. Fitzpatrick[2], John David N. Dionisio[3], Nicole A. Anguiano[1,3], Juan S. Carrillo[2,3], Trixie Anne M. Roque[2], Anindita Varshneya[1,3], Mihir Samdarshi[1], and Chukwuemeka Azinge[3]

[1]Department of Biology, [2]Department of Mathematics, [3]Department of Electrical Engineering & Computer Science, Loyola Marymount University, 1 LMU Drive, Los Angeles, CA 90045 USA

Email: kdahlquist@lmu.edu
Project Websites: http://kdahlquist.github.io/GRNmap/ and http://dondi.github.io/GRNsight/
Source Code: https://github.com/kdahlquist/GRNmap/ and https://github.com/dondi/GRNsight
License: BSD license, https://github.com/kdahlquist/GRNmap/blob/master/LICENSE and https://github.com/dondi/GRNsight/blob/master/LICENSE.txt

A gene regulatory network (GRN) consists of genes, transcription factors, and the regulatory connections between them that govern the level of expression of mRNA and proteins from those genes. Over a period of several years, our group has developed a MATLAB software package, called GRNmap, that uses ordinary differential equations to model the dynamics of medium-scale GRNs. The program uses a penalized least squares approach (Dahlquist et al. 2015, https://doi.org/10.1007/s11538-015-0092-6) to estimate production rates, expression thresholds, and regulatory weights for each transcription factor in the network based on gene expression data, and then performs a forward simulation of the dynamics of the network. GRNmap has options for using a sigmoidal or Michaelis-Menten production function. Although originally developed for yeast data, the model is broadly applicable to any species. The large number of developers and time span of development led to a code base that was difficult to revise and adjust. We therefore brought the code under version control in a GitHub repository and refactored the script-based software with global variables into a function-based package that uses an object to carry relevant information from function to function. This modular approach allows for cleaner, less ambiguous code and increased maintainability. We standardized the format of the input and output Excel workbooks, adding an optimization diagnostics output worksheet which includes both the actual and theoretical minimum least squared error overall, and the mean squared errors for the individual genes. The MATLAB compiler was used to create an executable that can be run on any Windows machine without the need of a MATLAB license. Finally, we have implemented test-driven development for new features, and are improving the test coverage of previous code.

GRNsight is an open source web application for visualizing such models of gene regulatory networks (Dahlquist et al. 2016, https://doi.org/10.7287/peerj.preprints.2068v1). GRNsight accepts GRNmap- or user-generated spreadsheets containing an adjacency matrix representation of the GRN and automatically lays out the graph of the GRN model. It is written in JavaScript, with diagrams facilitated by D3.js. Node.js and the Express framework handle server-side functions. GRNsight's diagrams are based on D3.js's force graph layout algorithm, which was then extensively customized. GRNsight uses pointed and blunt arrowheads, and colors the edges and adjusts their thicknesses based on the sign (activation or repression) and magnitude of the GRNmap weight parameter. Visualizations can be modified through manual node dragging and sliders that adjust the force graph parameters. From the early stages, GRNsight has had a unit testing framework using Mocha and the Chai assertion library to perform test-driven development where unit tests are written before new functionality is coded. This framework consists of over 160 automated unit tests that examine over 450 test files to ensure that the program is running as expected. Error and warning messages inform the user what happened, the source of the problem, and possible solutions. Together, the life cycle of these two programs illustrates the differences

Biopython Project Update 2016

Christian Brueffer[a], Tiago Antão[b], Peter Cock[c], Eric Talevich[d], Michiel de Hoon[e], Wibowo Arindrarto[f], Leighton Pritchard[c], Anuj Sharma[g], Eric Rasche[h], Aaron Rosenfeld[i], Connor T. Skennerton[j], Marco Galardini[k], Markus Piotrowski[l], and the Biopython Contributors

17th Bioinformatics Open Source Conference (BOSC) 2016, Orlando, USA

Website: http://biopython.org
Repository: https://github.com/biopython/biopython
License: Biopython License Agreement (MIT style, see http://www.biopython.org/DIST/LICENSE)

The Biopython Project is a long-running distributed collaborative effort, supported by the Open Bioinformatics Foundation, which develops a freely available Python library for biological computation [1].

We present here details of the latest Biopython release - version 1.66. New features include: extended Bio.KEGG and Bio.Graphics modules to support drawing KEGG pathways with transparency; extended "abi" Bio.SeqIO parser to decode almost all documented fields used by ABIF instruments; a QCPSuperimposer module using the Quaternion Characteristic Polynomial algorithm for superimposing structures to Bio.PDB; and an extended Bio.Entrez module to implement the NCBI Entrez Citation Matching function and to support NCBI XML files with XSD schemas. Additionally we fixed miscellaneous bugs, enhanced our test suite and continued our efforts to abide by the PEP8 coding style guidelines.

We are currently preparing a new release – version 1.67 – that will deprecate the ability to compare SeqRecord objects with "==", which sometimes lead to surprising results. In addition it will feature a new experimental Bio.phenotype module for working with Phenotype Microarray data; updates to Bio.Data to include NCBI genetic code table 25, covering Candidate Division SR1 and Gracilibacteria; an update to Bio.Restriction to include the REBASE May 2016 restriction enzyme list; updates to BioSQL to use foreign keys with SQLite3 databases; as well as corrections to the Bio.Entrez module and the MMCIF structure parser.

Our website has been migrated from MediaWiki to GitHub Pages and is now under version control. The continuous integration process on GitHub has been enhanced by including external services like Landscape, Quantified Code and Codecov to perform quality review, test coverage analysis and generation of quality metrics.

Finally, our range of Docker containers has been greatly enhanced. In addition to a basic container that includes Python 2 and 3 with Biopython and all its dependencies, as well as a BioSQL container, we now also provide two versions of Jupyter notebook containers: a basic one, and a version including the Biopython tutorial as notebooks.

References

[1] Cock, P.J.A., Antao, T., Chang, J.T., Chapman, B.A., Cox, C.J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., de Hoon, M.J. (2009) Biopython: freely available Python tools for computational molecular biology and bioinformatics. *Bioinformatics* **25**(11) 1422-3. doi:10.1093/bioinformatics/btp163

[a]Department of Clinical Sciences, Lund University, Lund, SE. Email: christian.brueffer@med.lu.se
[b]Division of Biological Sciences, University of Montana, Montana, USA
[c]Information and Computational Sciences, James Hutton Institute (formerly SCRI), Invergowrie, Dundee, UK
[d]Department of Dermatology, University of California San Francisco, San Francisco, CA, USA
[e]Division of Genomic Technologies, RIKEN Center for Life Science Technologies, Yokohama, JP
[f]Sequencing Analysis Support Core, Leiden University Medical Center, Leiden, NL
[g]Department of Informatics and Telecommunications, University of Athens, Athens, Greece
[h]Center for Phage Technology, Department of Biochemistry and Biophysics, Texas A&M University, USA
[i]Department of Biomedical Engineering, Science & Health Systems, Drexel University, USA
[j]Division of Geological and Planetary Sciences, California Institute of Technology, Pasadena, CA, USA
[k]EMBL-EBI, Wellcome Trust Genome Campus, Cambridge, UK
[l]Department of Plant Physiology, Ruhr-Universität Bochum, DE

This talk is accompanied by poster #13.

Sequenceserver: a modern graphical user interface for custom BLAST databases

Anurag Priyam[1,2,3*], Ben J. Woodcroft [3], Vivek Rai [3], Alekhya Munagala [3], Ismail Moghul [3], Filip Ter [3], Mark Anthony Gibbins [3], HongKee Moon [3], Guy Leonard [3], Wolfgang Rumpf [3], Yannick Wurm[1,3]

[1] School of Biological and Chemical Sciences, Queen Mary University of London, UK.
[2] Current: William Harvey Research Institute, Queen Mary University of London, UK.
[3] Affiliation omitted due to lack of space.

[*] Email: anurag08priyam@gmail.com

Project Website: http://sequenceserver.com
Source Code: https://github.com/wurmlab/sequenceserver
License: AGPL v3

Main Text of Abstract

The dramatic drop in DNA sequencing costs has created many opportunities for novel research that require comparing newly obtained and previously known sequences. This is commonly done with BLAST, yet using BLAST directly on new datasets requires substantial technical skills or helpful colleagues. Furthermore, graphical interfaces for BLAST are challenging to install and largely mimic underlying computational processes rather than work patterns of researchers.

We combined a user-centric design philosophy with sustainable software development approaches to create Sequenceserver. The innovations in Sequenceserver over other BLAST servers are at three levels. First, our software can be rapidly installed and used on custom datasets for individual use or sharing with a community. Second, by analysing user input and using simple algorithms, Sequenceserver reduces the amount of decisions the user must make, provides interactive visual feedback, and prevents common potential errors that would otherwise cause erroneous results. Finally, Sequenceserver provides multiple highly visual and text-based output options that mirror the requirements and work patterns of researchers.

The result of our approach has been quite successful as demonstrated by >27 mentions of the software in peer-reviewed publications. We believe the software, our user-centric design philosophy, community building and sustainable development approach, and the customisable codebase will be of significant interest to the OBF community.

This talk is accompanied by poster #14.

Title	Linuxbrew and Homebrew-Science to Navigate the Software Dependency Labyrinth
Author	*Shaun Jackman*, Inanc Birol
Affiliation	Genome Sciences Centre, British Columbia Cancer Agency
Contact	sjackman@bcgsc.ca
URL	http://linuxbrew.sh/, https://github.com/Homebrew/homebrew-science
License	BSD 2-Clause
Keywords	software installation, package manager, open source, reproducible research, high performance computing

Installing software is at best a tedious experience, and is often a distressing experience. Each operating system (OS) provides its own package manager to install software and manage dependencies: for example, apt-get and yum for the various distributions of Linux, and Homebrew is popular for Mac OS. Using the package manager provided by the system is not however without its own set of challenges. The system's package manager may require administrator access to the machine, which is typically not available on high performance computing clusters. It installs versions of software from the era of the OS, and clusters are notorious for running ancient distributions of Linux. Current bioinformatics software is often not yet packaged and provided only as source. Compiling software from source can vary from difficult to impossible, for example when the compiler and libraries provided by the operating system are a decade old. Manually navigating the recursive dependency chain of the tool and its dependencies, and their dependencies, can feel like a labyrinth with no end, and can even result in conflicting dependencies that are mutually exclusive and impossible to satisfy.

Linuxbrew is a package manager for Linux derived from Homebrew, the Mac OS package manager. It is a cross-platform utility, compatible with any distribution of Linux and version of Mac OS released in the last decade, allowing you to use the same package manager on both your Linux server and your Mac laptop. It can be installed in your home directory, and does not require administrator access. Using Linuxbrew, challenging tasks are made easy; for example installing a modern compiler in your home directory takes a few minutes, even on an ancient distribution of Linux.

Homebrew-Science is a collection of scientific software packages installable by either Linuxbrew or Homebrew. A third of the 600 software packages available on Homebrew-science are bioinformatics tools. Software packages are maintained up-to-date by a fervent community of over 400 contributors, and the scripts to install historical versions of software are retained in version control.

Repeating an analysis of data starts with obtaining the original data and installing the software used for that analysis. Linuxbrew streamlines the installation of software in a repeatable fashion. Having the data and software is not itself sufficient, but is certainly a necessary component of reproducible science.

This talk is accompanied by poster #15.

SnoVault and encodeD: A novel object-based storage system and applications to ENCODE metadata

Benjamin C. Hitz[1] (hitz@stanford.edu)
Laurence D. Rowe[1], Nikhil R. Podduturi[1], David I. Glick[1], Ulugbek K. Baymuradov[1], Venkat S. Malladi[3], Esther T. Chan[1], Jean M. Davidson[1], Idan Gabdank[1], Aditi K. Narayana[1], Kathrina C. Onate[1], Marcus C. Ho[1], Brian T. Lee[2], Stuart R. Miyasato[1], Timothy R. Dreszer[1], Cricket A. Sloan[1], J. Seth Strattan[1], Forrest Y. Tanaka[1], Eurie L. Hong[4], and J. Michael Cherry[1*]

[1]Department of Genetics, Stanford University School of Medicine, Stanford, CA 94305, USA, and [2]Center for Biomolecular Science and Engineering, School of Engineering, University of California Santa Cruz, Santa Cruz, CA 95064, USA

Project Website: https://www.encodeproject.org
Source Code: https://github.com/ENCODE-DCC/encoded, https://github.com/ENCODE-DCC/snovault
License: MIT

Main Text of Abstract

The Encyclopedia of DNA elements (ENCODE) project is an ongoing collaborative effort[1–6] to create a comprehensive catalog of functional elements initiated shortly after the completion of the Human Genome Project[7][1]. The current database exceeds 5500 experiments across more than 350 cell lines and tissues using a wide array of experimental techniques to study the chromatin structure, regulatory and transcriptional landscape of the *H. sapiens* and *M. musculus* genomes. All ENCODE experimental data, metadata, and associated computational analyses are submitted to the ENCODE Data Coordination Center (DCC) for validation, tracking, storage, and distribution to community resources and the scientific community. As the volume of data increases, the identification and organization of experimental details becomes increasingly intricate and demands careful curation. The ENCODE DCC[8–10] has created a general purpose software system, known as SnoVault, that supports metadata and file submission, a database used for metadata storage, web pages for displaying the metadata and a robust API for querying the metadata. The software is fully open-source, code and installation instructions can be found at: http://github.com/ENCODE-DCC/snovault/. The core database engine, SnoVault (which is completely independent of ENCODE, genomic data, or bioinformatic data) has been released as a separate Python package.

This talk is accompanied by poster #16.

Title	State of the openSNP.org Union: Dockerizing, Crowdfunding & Opening for Contributors
Author	*Bastian Greshake*, Philipp E. Bayer, Helge Rausch, Lore Mroz, Julia Reda
Affiliation	openSNP.org
Contact	info@opensnp.org
URL	the project and the code
License	Code: MIT License, Data: CC-Zero

openSNP is a central open-data repository for people who got their genomes analyzed through Direct-To-Consumer (DTC) genetic testing channels. The project allows people to publish their genetic data, along with phenotypic annotations and quantified self data, into the public domain using a Creative Commons Zero license. *openSNP* offers annotations for the tested genetic variants, by mining different public and open data sets, such as the Public Library of Science, *SNPedia*, *Mendeley* and more.

Since our presentation last year, at BOSC2015, our database has grown even further, from nearly 1,700 data sets to now over 2,500 published genotypes. Additionally there are now over 370 different phenotypes listed for which over 38,000 annotations were entered. This makes *openSNP* one of the largest data resources of this type. With this data the project actively contributes to the discussion on open human genetic data, such as bioethical implications and privacy research, genealogy, teaching, pharmacogenomics and even art.

Unlike many other projects of its kind, *openSNP* is not hosted or supported by any university or other academic institution. Instead it is completely driven and maintained by the community of people sharing and using it. This means that the project is facing some unique problems. Due to this, we had to make 2015 a year of consolidation.

With the increasing growth of the database, we had to optimize our infrastructure. On the technical side this meant moving from a monolithic server-based installation to running in multiple Docker containers, hosted on 3 small virtual servers, optimized to specific tasks. But upgrading our server infrastructure also meant that the project could not be funded solely by our core team's day jobs any longer.

Instead of relying only on official funding bodies or corporate sponsors we chose to turn to crowd-funding via the platforms of Patreon and Gratipay, which both allow for recurring donations by individuals. Since the beginning of the campaign in autumn 2015 we could win around 30 people to contribute financially. Thanks to this, we can now pay our infrastructure costs for running the project through the cumulative efforts of small donations from the crowd. Furthermore we got one corporate contributor on board, but decided to limit its contributions in order to stay independent.

In this talk we will give a small summary of how we performed our infrastructural updates, managed to run a successful crowdfunding campaign, the political implications of corporate sponsorship and how we prepared for new contributors.

This talk is accompanied by poster #17.

The GenePattern Notebook Environment

Michael Reich[1], Thorin Tabor[2], Ted Liefeld[1], Barbara Hill[2], Helga Thorvaldsdottir[2], Jill P. Mesirov[1]

[1] University of California, San Diego, La Jolla CA 92093 Email: mmreich@ucsd.edu
[2] Broad Institute, Cambridge MA 02142

Project Website:	http://www.genepattern.org/genepattern-notebooks
Source Code:	https://github.com/genepattern/genepattern-notebook
License:	BSD-style, see https://github.com/genepattern/genepattern-notebook/blob/master/LICENSE.txt

Interactive notebook systems have made significant strides toward realizing the potential of reproducible research, providing environments where users can interleave descriptive text, mathematics, images, and executable code into a "live" sharable, publishable "research narrative." However, many of these systems require knowledge of Python, R, or another programming language and are therefore out of the reach of non-programming genomic researchers.

To address this gap, we have developed the GenePattern Notebook environment, which connects the popular Jupyter Notebook system to the GenePattern platform for integrative genomics, making the hundreds of bioinformatics analysis methods in GenePattern available within the powerful and flexible Jupyter Notebook interface. GenePattern provides domain-specific methods for the analysis of gene expression (RNA-seq and microarray), sequence variation, proteomics, and genomic networks, as well as general machine learning methods for clustering, classification, and dimensionality reduction, and utility methods for data import, formatting and preprocessing. GenePattern includes methods adapted from many popular open source libraries, including Bioconductor and scikit-learn. Through its module-based architecture, GenePattern also allows users to easily add their own tools, automatically making them available within the notebook environment.

The GenePattern Notebook environment consists of a plugin that provides access to GenePattern from within Jupyter Notebook, a collection of notebook documents that demonstrate the use of GenePattern Notebook in various analysis scenarios and which can be adapted for a researcher's individual use, and a web site (in development) where users can create, share, run, and publish their own notebooks. The plugin is implemented as an extension to the Jupyter Notebook system that provides a new "GenePattern" cell type, which allows users to log in to any available GenePattern server, select any available analysis, and configure and execute it. Results are available to downstream analyses, which can be either additional GenePattern method cells or standard Jupyter Notebook code cells. GenePattern Notebook is currently available as a Docker image and as an installation from the Python Package Index (PyPI).

This talk will describe how the GenePattern Notebook environment extends the analytical and reproducible research capabilities of Jupyter Notebook and GenePattern, how using GenePattern Notebook obviates the need for coding in many analysis cases, and will demonstrate genomic analyses within GenePattern notebooks.

Reproducible Computational Workflows with Continuous Analysis

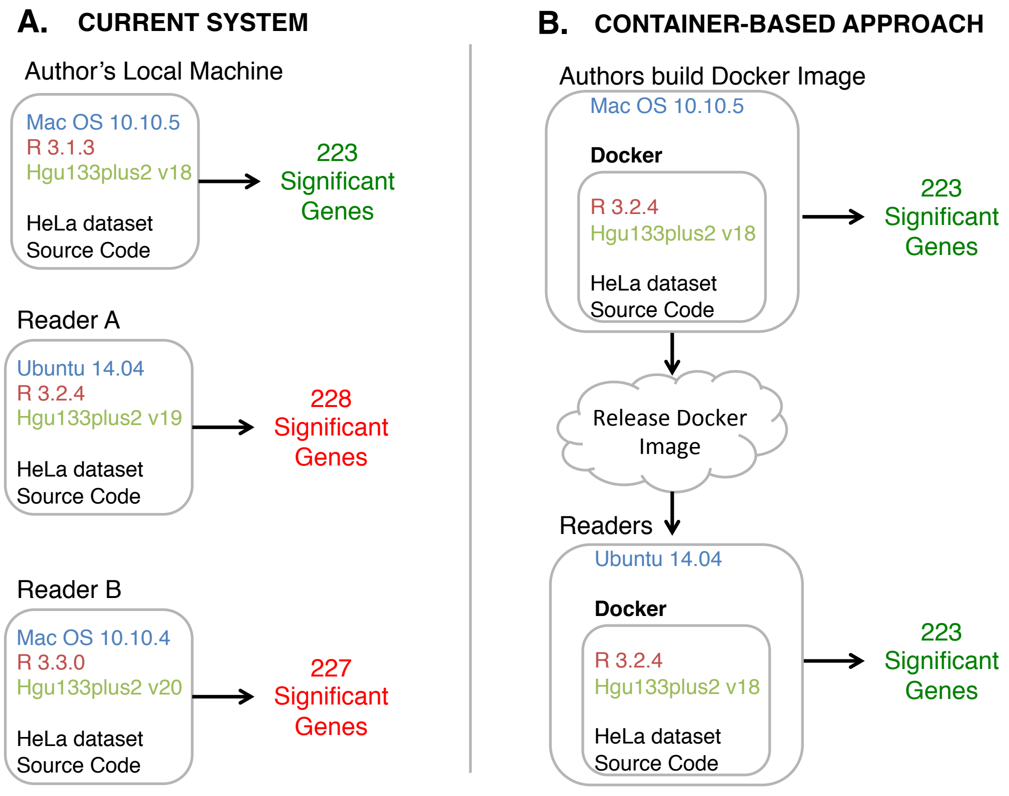
Brett K Beaulieu-Jones[1], Casey S Greene[2]

[1] Graduate Group in Genomics and Computational Biology, Computational Genetics Lab, Institute for Biomedical Informatics, University of Pennsylvania. Email: brettbe@med.upenn.edu

[2] Department of Systems Pharmacology and Translational Therapeutics, Institute for Biomedical Informatics, Institute for Translational Medicine and Therapeutics, University of Pennsylvania

Source Code: https://github.com/greenelab/continuous_analysis
Example Project: https://github.com/greenelab/DAPS/ (Example Project)
License: BSD3

Reproducing experiments is vital to science. Being able to replicate, validate and extend previous work also speeds new research projects. Reproducing computational biology experiments, which are scripted, should be straightforward. But reproducing such work remains challenging and time consuming. In the ideal world we would be able to quickly and easily rewind to the precise computing environment where results were generated. We would then be able to reproduce the original analysis or perform new analyses. We introduce a process termed "continuous analysis" which provides inherent reproducibility to computational research at a minimal cost to the researcher. Continuous analysis combines Docker, a container service similar to virtual machines, with continuous integration, a popular software development technique, to automatically re-run computational analysis whenever relevant changes are made to the source code. This allows results to be reproduced quickly, accurately and without needing to contact the original authors. Continuous analysis also provides an audit trail for analyses that use data with sharing restrictions. This allows reviewers, editors, and readers to verify reproducibility without manually downloading and rerunning any code.



This talk is accompanied by poster #18.

Reproducible Research in the Cloud with the Refinery Platform

***Nils Gehlenborg**[1], Shannan J Ho Sui[2], Ilya Sytchev[2], Stefan Luger[3], Fritz Lekschas[1], Richard W Park[1], Jennifer Marx[1], Scott Ouellette[1], David R Jones[4], Anton Xue[1], Psalm Haseley[1], Marc Streit[3], Winston Hide[2,4], Peter J Park[1]*

[1] Department of Biomedical Informatics, Harvard Medical School, Boston, MA, USA
[2] Department of Biostatistics, Harvard School of Public Health, Boston, MA, USA
[3] Department of Computer Science, Johannes Kepler University Linz, Linz, Austria
[4] Sheffield Institute for Translational Neuroscience, University of Sheffield, Sheffield, UK

Website: *http://refinery-platform.org*
Repository: *https://www.github.com/parklab/refinery-platform*
License: *MIT (+ additional clause) - https://github.com/parklab/refinery-platform/blob/develop/LICENSE*
Correspondence: *nils@hms.harvard.edu*

The Refinery Platform is a data analysis environment for reproducible research that links a data repository with analysis pipelines and visualization tools within a single user interface. The goal of Refinery is to facilitate analysis and interpretation of genomic and epigenomic data in a reproducible fashion. To support this, the data repository is built around the ISA-Tab data model (http://isa-tools.org) and analyses are executed in Galaxy (http://usegalaxy.org). Workflows are configured, launched, and monitored through the Refinery user interface, which offers a sophisticated file browser that operates on data set sample annotations. Among other efforts, we have created an instance of the Stem Cell Commons based on Refinery for the Harvard Stem Cell Institute with over 200 stem cell related data sets.
Originally designed to run on institutional or lab compute clusters, we have recently extended Refinery to be able to execute analyses on the Amazon Web Services (AWS) platform and to overcome some of the limitations of typical research computing environments. Refinery relies on Galaxy Cloudman (http://cloudman.irb.hr) to provide a Galaxy cluster on AWS. To reduce the effort required to deploy Refinery instances on AWS, we are creating custom Cloudman machine images that include the tools that we need for specific instances. By deploying our own compute cluster on AWS rather than relying on the availability of local infrastructure, the platform will also be more attractive to outside developers.
A second challenge that we recently addressed to make analyses more reproducible is the visualization of the provenance graphs that result from the execution of workflows on data sets with dozens of files. While node-link diagrams are useful to convey the flow of data from the input through the tools of workflows to the outputs, this representation does not scale to more than a handful of typical workflow executions. We have designed a new approach that employs dynamic graph aggregation and expansion based on user interest. Users can expand a subset of highly compressed graph to view details of a particular analysis without losing the overall context. Additionally, filters can be applied to hide part of the provenance graph—for example analyses conducted before a particular date—on demand.
The combination of cloud-based workflow execution, automated tracking of provenance information, and tools to visualize data provenance allows Refinery users to conduct reproducible computational research without any additional effort.

The Refinery Platform project is funded by NIH (R00 HG007583) and the Harvard Stem Cell Institute. Additional support is provided by an AWS in Education Research Grant.

This talk is accompanied by poster #19.

Title	ReportMD: Writing complex scientific reports in R
Author	*Peter Humburg*
Affiliation	Garvan Institute of Medical Research
Contact	p.humburg@garvan.org.au
URL	https://github.com/humburg/reportmd
License	MIT

R is commonly used for complex analyses of large datasets. To improve the reproducibility of such analyses and provide the necessary documentation it is increasingly common to use literate programming techniques. Although R has had the ability to embed R code in long-form documents describing the analysis and presenting the results for many years via *Sweave*, it has been the introduction of *RMarkdown* that has driven the increasing popularity of this approach. The relative simplicity of authoring *Markdown* documents and the ability to convert these into a large variety of output formats via *pandoc* combined with the tight integration with *RStudio* make this approach easy to use and powerful at the same time.

While this is an essential technique for reproducible data analysis and much progress has been made over recent years to facilitate this literate programming approach, some challenges remain. One such issue is that complex analyses are best split into smaller units to retain some modularity but at the same time are likely to depend on the results of earlier stages of the analysis, i.e. modules may have complex interdependencies.

The solution to this problem provided by *knitr* is the use of child documents. These can be referenced in a main document and will then be evaluated separately and concatenated into one, potentially very large, document. While this has the advantage that multiple analysis modules can be combined into a larger report and all parts of the analysis are collated in a single document, ensuring that they are not inadvertently separated, it also has its downsides. In particular, it does not distinguish well between the data analysis and the presentation of the results obtained through this analysis. While both of these are intimately related and should not be entirely separated the natural flow of an analysis and the order imposed by internal dependencies is not always well suited to the effective presentation of results.

At the other extreme a commonly employed alternative largely separates the presentation layer from the analysis documents. Although this approach incorporates selected outputs from the analysis, such as figures and tables, into the presentation layer these connections are often fragile and a disconnect between the presentation and analysis layers can easily occur.

An approach that falls between these two extremes may be preferable. The aim is to produce a document that is designed to facilitate the presentation of results without being constraint by the structure of the analysis documents, forming a presentation layer on top of the analysis layer, while maintaining tight links between the two. To accomplish this such a document should

- have direct access to the R objects produced during the analysis;
- respond to changes in the analysis documents with corresponding changes in the presentation layer; and
- facilitate cross-references between individual documents.

The R package *ReportMD* presented here aims to provide the infrastructure required to achieve this. Build on top of *Rmarkdown* it is easy to use in existing work flows. *ReportMD* extends *knitr*'s chunk dependencies to extend across document boundaries and provides facilities to automatically load cached results from chunks in other documents. This makes it straightforward to include figures, tables and other summaries in the presentation layer while the required computations are documented in detail in an analysis document.

This talk is accompanied by poster #20.

Apollo Genome Annotation Editor: Latest Updates, Including New Galaxy Integration

Monica C Munoz-Torres[1], Nathan A Dunn[1], Deepak Unni[2], Eric Yao[3], Eric Rasche[4], Colin Diesh[2], Christine E Elsik[2], Ian Holmes[3], and Suzanna E Lewis[1]

[1] Lawrence Berkeley National Laboratory, Genomics Division, Berkeley, CA. Email: MCMunozT@lbl.gov
[2] University of Missouri, Divisions of Plant and Animal Sciences, Columbia, MO.
[3] University of California Berkeley, Bioengineering, Berkeley, CA.

Project Website: http://genomearchitect.org/
Source Code: https://github.com/GMOD/Apollo
License: Berkeley Software Distribution (BSD) License. See https://github.com/GMOD/Apollo/blob/master/LICENSE.md

Manual curation is crucial to improving the quality of the annotations for a genome sequencing project. During this portion of the genome sequencing workflow, curators use a variety of experimental evidence to improve on automated predictions to more accurately represent the underlying biology.

Apollo is a web-based genome annotation editor that allows curators to manually revise and edit genomic elements. It provides a reporting structure for annotated genomic elements and an '*Annotator Panel*' that allows users to quickly browse the genome and all available annotations. Users can manually edit the structure of a genomic element as well as add metadata, including references to other databases, adding functional assignments to genes and gene products with specific lookup support for Gene Ontology (GO) terms, as well as including references to published literature in support of these annotations.

Apollo is currently used in more than one hundred genome annotation projects around the world, ranging from the annotation of a single species to lineage-specific efforts supporting annotation for dozens of organisms at a time. Apollo enables collaborative, real-time curation (akin to Google Docs); researchers may restrict access to certain annotations depending on the role of users and groups within the community, as well as share tracks of evidence data with the public. Users are able to export their manual annotations via FASTA and GFF3 files, the Chado database schema, and web services. The news hot of the presses is that Apollo is now available for integration with Galaxy via Docker! This allows users to run analyses on their genome of interest, including a step of manual curation, all from the comfort of their installation of the versatile Galaxy platform.

This talk is accompanied by poster #21.

An Invitation to the bioinformatics community to participate in the HUBzero® open source release

Michael Zentner[1], Ishwar Chandramouliswaran[2], Richard Zink[3]

[1] Purdue University, West Lafayette, IN. Email: mzentner@purdue.edu

[2] National Cancer Institute, Bethesda, MD.

[3] Purdue University, West Lafayette, IN.

Project Website: http://www.hubzero.org
Source Code: https://github.com/hubzero
License: GNU General Public License

The HUBzero® platform enables online scientific communities to collaborate and explore science by sharing information and computational resources. HUBzero HUBs facilitate collaboration using shared projects, groups, and resource collections. HUBzero HUBs also allow easy deployment of simulation and modeling tools. Members can put a simple user interface on their tools in a matter of weeks, and deploy those tools over the web, enabling others to run them in their browser without downloading or installing any code. Today over 60 scientific communities use HUBzero, with 2 million visitors annually. HUBzero is sustainable as a funded component of more than 20 grants, service contracts, and its own foundation from a variety of agencies / entities including the NSF, NIH, AFRL, non-profit societies, and corporations, and lately has been receiving interest from the bioinformatics community. HUBzero has been adopted by the National Cancer Institute (NCI), U.S. Army Medical and Materiel Command (MRMC), the Regenstrief Foundation (RF), and the Regenstrief Center for Healthcare Engineering (RCHE) to form the HUBs NCIP Hub, cceHUB, CitSciBio hub, and CatalyzeCare. A goal of the National Cancer Informatics Program's NCIP Hub is to create community driven, adaptive, and collaborative environments to promote exchange of research ideas and resources, such as software tools, data, standards, or other relevant digital assets in this open access resource that includes nearly 400 public resources about imaging, pathology, informatics, and more. MRMC and RF sponsor cceHUB, where many special interest communities share data and use data search and exploration interfaces to apply systems engineering principles to the treatment of cancer, for both prospective and retrospective clinical research. NCI sponsors CitSciBio hub as an online space for the growing and virtually dispersed biomedical citizen science resources, projects, references, methods, and communities to be discovered. RCHE has formed a community on their CatalyzeCare hub where hospitals voluntarily contribute the alert streams emanating from smart IV pumps to the community, benchmark themselves against each other, and collectively improve overall patient safety by creating a set of best practices regarding infusions. HUBzero provides common infrastructural components to these communities, including compute environments like Purdue's community cluster, Open Science Grid, and parallel Matlab computing from Cornell's Red Cloud environment. It also provides support for R tool development, assembly of complex workflows with Pegasus, and collaborative development capability for bioinformatics tools using linux workspaces available in a web browser interface. As these communities interact with their HUBs, we expect to contribute to the creation of a 'community impact score' based on data sharing, software sharing, discoverability, annotation, and use and reuse. Individuals engaged in cancer research or medical device informatics can become members by visiting www.nciphub.org, www.ccehub.org, www.citscibio.org, and www.catalyzecare.org. We invite members of the biomedical informatics field to join the HUBzero community and contribute to their open source development efforts as applications or core capabilities in the HUBzero open source release to better meet the needs and use cases of the field.

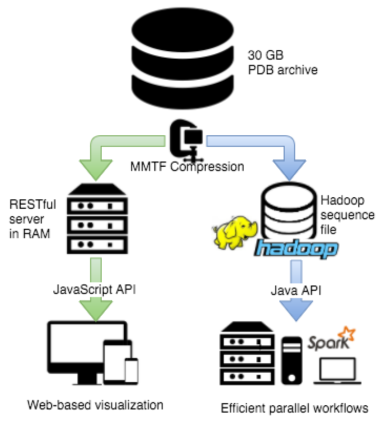
This talk is accompanied by poster #22.

PDB on steroids – compressive structural bioinformatics

Peter W. Rose[1,2], Anthony R. Bradley[1,2], Alexander S. Rose[2], Yana Valasatava[2], Jose M. Duarte[1,2], Andreas Prlić[1,2]

[1] RCSB Protein Data Bank, San Diego Supercomputer Center, UC San Diego, peter.rose@rcsb.org
[2] Structural Bioinformatics Laboratory, San Diego Supercomputer Center, UC San Diego

Project Website: http://mmtf.rcsb.org/
Source Code: https://github.com/rcsb/mmtf
License: Apache 2 (Java, Python API), MIT (JavaScript API)



Abstract

We are developing compressed 3D molecular data representations and workflows ("Compressive Structural Bioinformatics") to speed up mining and visualization of 3D structural data by one or more orders of magnitude. Our data representations allow scanning and analyzing the entire PDB archive in minutes or visualizing structures with millions of atoms in a web browser on a smart phone.

Compact and self-contained data representation - Existing text-based file formats for macromolecular data are slow to parse, are not easily extensible, and do not contain certain key data (e.g., all bonding information). For these reasons we have developed the **M**acro**m**olecular **T**ransmission **F**ormat (MMTF) (http://mmtf.rcsb.org/). MMTF has three core benefits over existing file formats. First, through custom compression methods, the entire Protein Data Bank (PDB) archive can be stored in 7GB. This enables fast network transfer for visualization and in-memory processing of the entire PDB. Second, MMTF data are serialized into MessagePack (http://msgpack.org), a compact, extensible and efficient format, similar to JSON, but binary for faster parsing. Third, MMTF is user friendly, extensible and contains information not found in current formats. In this work we show that MMTF enables high-performance visualization and scalable structural analysis of the PDB archive.

High-performance web-based visualization - The MMTF files are served directly from RAM using a RESTful service. This low latency service, combined with the reduced individual file size and the increased parsing speed of the binary format facilitates high performance web-based visualizations. Specifically we have seen a greater than 20x speedup over mmCIF in loading of PDB entries from sites across the USA, Europe, and Asia. Using the MMTF JavaScript API and NGL, a highly memory-efficient WebGL-based viewer (https://github.com/arose/ngl), even the largest structures in PDB can be visualized on a smart phone.

High-performance distributed parallel workflows - The order of magnitude increase in parsing speed enables scalable Big Data analysis of 3D macromolecular structures. A Hadoop sequence file (binary flat file of key value pairs, optimized for parallel sequential access) of MMTF data is released and updated weekly for the entire PDB archive. Distributed, parallel processing is then possible from this file, using Big Data frameworks such as Apache Spark (http:spark.apache.org/). As an example, we have used this file for ligand extraction. We extracted all ligands from the PDB using the MMTF Hadoop file with Apache Spark in about 3 minutes. In contrast, using mmCIF files as input, the same task took several hours.

The MMTF file format enables a paradigm change for structural bioinformatics applications. It is now possible to store the entire PDB in memory to eliminate I/O bottlenecks, to rapidly visualize large structures over the web, and to trivially perform distributed parallel processing on laptops, desktops, and compute clusters.

This project was supported by the National Cancer Institute of the NIH's Big Data to Knowledge initiative (BD2K) under award number U01 CA198942.

This talk is accompanied by poster #23.

Puzzle: VCF/GEMINI interface for genetic disease analysis

Authors: Robin Andeer, Måns Magnusson, Henrik Stranneheim
Affiliations: Department of Molecular Medicine and Surgery, Science for Life Laboratory, Center for Molecular Medicine, Karolinska Institutet, Stockholm, Sweden
Email: robin.andeer@scilifelab.se
Project website: https://robinandeer.gitbooks.io/puzzle/content/
Source code: https://github.com/robinandeer/puzzle
License: MIT

VCF and BCF files are the ubiquitous storage formats for variant calls. However, at their core it's no more than tab-delimited text files. They lack a powerful and standardized data model to enable sophisticated genetic analysis. Converting VCFs into a GEMINI database goes a long way to enable this type of complex processing. However, they are not human friendly when it comes to inspecting their contents. You need knowledge of SQL and the internal schema. This isn't something you can expect from your average clinical or research users.

Puzzle is a web interface for variant calling resources (VCF files and GEMINI databases). It lets you organize, inspect, filter, annotate, explore and analyze variant calls in a web browser. It provides a unified and intuitive interface for all loaded resources and can be used by anyone without computer programming literacy. It's already used extensively by clinicians and researchers at Karolinska Hospital working with a wide range of genetic disorders.

These are the main highlights of Puzzle:

- Simple installation (PyPI or Bioconda) and minimal setup
- Automatic discovery of resources, just point to a folder and Puzzle will do the rest
- Support for single nucleotide variant (SNV) and structural variant (SV) analysis
- Gene panels and human phenotype ontology (HPO) support
- Automatic annotations on gene and transcript level
- Support for grouping samples into logical groups like families
- Support for comments and highlighting of interesting variants

We set out to make Puzzle simple to install and intuitive to use. It's implemented in Python as a stand alone web server with a light weight SQLite backend. You can visualize your variant calls in minutes with minimal prerequisites. Simply point it to a directory with some VCF files to get started:

```
$ puzzle view tests/fixtures/
```

This talk is accompanied by poster #24.

Modernization of the Cytoscape ecosystem

Keiichiro Ono[1], Eric Sage[2], Barry Demchak[3]

[1] University of California, San Diego. La Jolla, CA USA. Email: kono@ucsd.edu

[2] University of California, San Diego. La Jolla, CA USA. Email: eric.david.sage@gmail.com

[3] University of California, San Diego. La Jolla, CA USA. Email: idekerlab.bdemchak@gmail.com

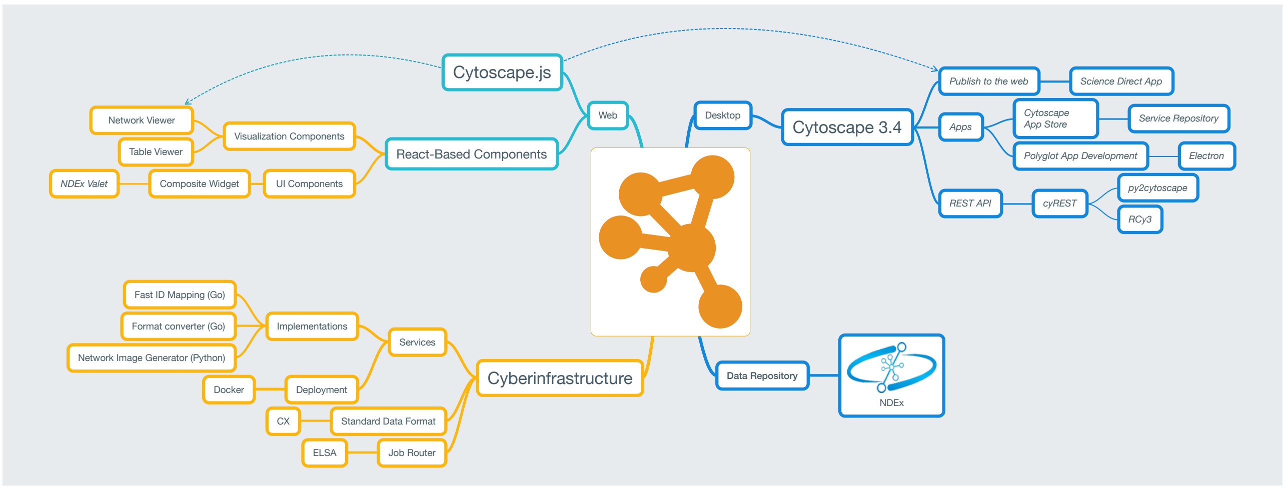
Project Website: http://www.cytoscape.org/
Source Code: https://github.com/cytoscape/
License: LGPL/MIT License

Abstract

Cytoscape is the de-facto standard biological network analysis and visualization platform – its first version was released in 2002. Compared to other systems biology tools, it is one of the oldest applications and it has one of the largest user and developer communities. Maintaining sustainability of such platform is a challenge due to its desktop-only deployment, evolving user needs and changing funding scenarios. Cytoscape's extensibility mechanism is a plugin architecture, called Apps, and has been a great way to satisfy diverse community needs. However, it is insufficient for modern bioinformatics workflows requiring massive memory and parallelization, such as is available on emerging web- and cloud-based application platforms.

Today, data analysis/visualization tools are in transition from desktop applications to loosely coupled, large-scale distributed systems with support for different types of clients, including web browsers and command line applications. The desktop version of Cytoscape is not designed to support such use cases. To provision for these modern bioinformatics workflows, we are modernizing Cytoscape ecosystem using mainstream technologies and design techniques including a service-oriented architecture with microservices, containers, browser-based visualization components, and web/native hybrid applications. The new Cytoscape ecosystem, called Cytoscape Cyberinfrastructure (CI) is the main project for modernizing the Cytoscape platform.

In this presentation, we will discuss the technical challenges of modernizing long-lived, widely-used bioinformatics applications and will introduce the new Cytoscape CI architecture and technologies we used for the process.



Collaborative Software Development: Lessons from Open Source

Abigail Cabunoc Mayes[1], Mozilla Science Lab team and contributors

[1]Mozilla Science Lab, Mozilla Foundation, Toronto, Ontario, Canada. Email: abigail@mozillafoundation.org

Since the Mozilla Science Lab launched three years ago, we've tested ways to help the research community work together to build new tools for science. Through our open source projects and code sprints, we've seen that despite the interest in scientific software development, that sustaining development and doing so collaboratively remain a challenge.

Pulling from the open source community, we've developed trainings and mentorship programs to bring some of those lessons and values to the scientific software community. The materials in our Working Open Workshop take the norms and protocols for open source community organization and development and unpack tips to facilitate collaborative learning and contributorship. We are investing in leaders and facilitators who can help the research community learn to build software together.

The Working Open Workshop followed by one-on-one mentorship focuses on a few practical ways to get started in open source, such as roadmapping, identifying pathways for new participants, engaging with contributors, open licensing and software citation.

By working openly, a project lead demonstrates and encourages learning and participation. While mentorship and community engagement are vital throughout the life of a project, a few key steps can help jumpstart community involvement. This talk will outline a few ways to help elevate contributors to work more collaboratively and sustainably in the open.

Poster #25.

Lightweight sample labeling, barcoding and tracking systems for the academic laboratory

Dimitra Sarantopoulou[1], Anand Srinivasan[4], Steve Vitale[1], Katherine Theken[3], Emanuela Riciotti[3], Faith Coldren[1], Tilo Grosser[1,3], Gregory R. Grant[1,2]

[1]Institute for Translational Medicine and Therapeutics, University of Pennsylvania, Philadelphia, PA. Email: dimitras@sas.upenn.edu
[2]Department of Genetics, University of Pennsylvania, Philadelphia, PA
[3]Department of Systems Pharmacology and Translational Therapeutics, University of Pennsylvania, Philadelphia, PA
[4]Penn Medicine Academic Computing Services, University of Pennsylvania, Philadelphia, PA

Source Code: https://github.com/dimitras/collos
License: MIT License

Abstract:
In the typical lab many bench investigators will come and go over the years and labs will accumulate a large collection of samples. Traditionally in most laboratories sample labeling and tracking of these samples is a problem that is left up to the individual bench investigators and it often amounts to no more than a few handwritten cryptic indications on the side of a tube and some notes in a notebook, or at best in an Excel spreadsheet. When people leave labs it can become very difficult for those remaining to pick up the pieces of their research, often because of an idiosyncratic and confusing documentation of their samples. The value of overcoming this problem is therefore enormous. However achieving a more sophisticated system, with barcoding and full sample tracking, has traditionally been too expensive for all but the most highly funded labs because of the significant overhead and expert systems support required. To overcome this limitation, we have developed a lightweight system called CollOS (**Coll**ection **O**f **S**amples), together with detailed documentation that will allow any lab to achieve relatively sophisticated sample labeling and tracking without excessive overhead. In particular we give specific guidance on hardware and software setup which reflects our experience testing numerous labels under harsh conditions. The entire system can be installed for under $5000. This is not a commercial endorsement and we have no commercial interest in the recommended hardware it is simply an unbiased review.

Poster #26.

Kronos: a workflow assembler for genome analytics and informatics

M Jafar Taghiyar[1,2], Jamie Rosner[1], Diljot Grewal[1,2], Bruno Grande[3], Radhouane Aniba[1,2], Jasleen Grewal[3], Paul C Boutros[4,5], Ryan D Morin[3], Ali Bashashati[1,2], and Sohrab P Shah[1,2]

[1]Department of Molecular Oncology, British Columbia Cancer Agency. Email: jtaghiyar@bccrc.ca
[2]Department of Pathology and Laboratory Medicine, University of British Columbia.
[3]Department of Molecular Biology and Biochemistry, Simon Fraser University.
[4]Ontario Institute for Cancer Research.
[5]Department of Medical Biophysics, University of Toronto.

Project URL: https://github.com/jtaghiyar/kronos
https://github.com/MO-BCCRC
License: MIT

Background
The field of next generation sequencing informatics has matured to a point where algorithmic advances in sequence alignment and individual feature detection methods have stabilized. However, practical and robust implementation of complex analytical workflows still requires significant programming investment and expertise. For example, analysis of cancer genomes may invoke sequence alignment followed by mutation, copy number, and rearrangement detection, followed by annotation of predicted functional effects of these alterations. In both research and clinical settings it is vital to encode specific analytical protocols for transparency of derivative results and clinical audits.

Results
We present Kronos, a software platform for automating the development and execution of reproducible, auditable and distributable bioinformatics workflows. Kronos obviates the need for explicit coding of workflows by transforming a text configuration file into an executable Python script. Given a set of existing components (analysis modules), a new workflow can be created in three simple steps:

Step 1. Create a configuration file template by running *kronos make_config* command.
Step 2. In the template, specify the order by which the components in the workflow should be run.
Step 3. Create the workflow by running *kronos init* command with the template as input.

The result is an executable Python script that uses the built-in run manager of Kronos. The run manager provides scalability by enabling users to run the workflow on a single machine or a cluster of computing nodes. Moreover, each component in the workflow can individually be run either locally, or on a cluster. Workflows created by Kronos are highly modular and configurable through configuration file editing. The workflows are fully encoded for ease of distribution and can be instantiated on external systems, promoting and facilitating reproducible research and comparative analyses. Kronos also has a framework for making components. Each component is an analysis module wrapped according to the framework. A component template can be made using *kronos make_component* command. Developers can use the framework to implement custom tools, reuse existing tools, and contribute to the community at large. Kronos is shipped with Docker and Amazon AWS machine images. It is free, open source and available through Python Package Index.

Conclusion
This work provides a framework towards rapid integration of new (and optimal) genomic analysis advances in high-throughput studies. The flexibility, customization, and modularity of the resulting pipelines make it an attractive system to use in any high-throughput genomics analysis endeavour. We expect Kronos will be a small but foundational step towards standardization and distribution of NGS workflows in both clinical and research applications.

Poster #27.

PhyPipe: an automated pipeline for phylogenetic reconstruction from multilocus sequences

Nicolás D. Franco-Sierra[1], Mateo Gómez-Zuluaga[2], Juan F. Díaz-Nieto[1], Javier C. Alvarez

[1]Grupo CIBIOP, Departamento de Ciencias Biológicas, Universidad EAFIT. Medellín, Colombia. Email: nfranco@eafit.edu.c
[2]Centro de Computación Científica APOLO, Dirección de Informática, Universidad EAFIT. Medellín, Colombia.

Website: `https://gitlab.com/cibiop/phypipe/wikis/home`
Repository: `https://gitlab.com/cibiop/phypipe`
License: BSD 3-Clause License, see `https://gitlab.com/cibiop/phypipe/blob/master/LICENSE`

Phylogenetic reconstruction based on multiple loci of DNA sequences is nowadays a common task in many fields of biological research such as: ecology, epidemiology, evolution, and taxonomy, among many others. This process involves several methodological steps for going from a DNA sequence to a final output tree (i.e., sequence alignment, testing substitution models, concatenation of loci, phylogenetic reconstruction). Phylogenetic analyses require the use of different bioinformatic tools that demand human supervision in intermediate steps, in particular, it is necessary to edit input files with the output of previous analyses. A disconnected approach between those steps could lead to a time consuming and error-prone task, specially when multiple runs are required (e.g., same dataset and different run parameters or different datasets with or without varying the parameters). Thus, this seems to be a common problem in bioinformatic routines. Previous efforts have developed workflow management tools that can handle routines using different phylogenetic systematic software; nonetheless, none of such routines are particularly developed to tackle the problem of developing partitioned phylogenetic analyses. Our approach is particularly useful for reducing both error in intermediate steps and overall time in the phylogenetic reconstruction, in addition to facilitate the multilocus reconstruction when major or minor changes are needed to the initial dataset or running parameters.

In this work we present Phypipe, an automated pipeline written in a programming language for bioinformatic pipelines named bpipe [1], that aims to facilitate and speed-up phylogenetic reconstructions using multilocus concatenated analyses. Phypipe is intended to be used with multiple-locus data. The program receives as input multiple FASTA files (one per locus) containing DNA sequence data and it also supports previously aligned sequences. DNA data can be either from coding or non-coding regions from organelle or nuclear genomes.

The workflow consists of four major stages using the following dependencies:

- An alignment step (MAFFT and RevTrans for coding sequences)
- Concatenation of all alignment files in a single file (homescript in Python)
- Best partition scheme search (partitionfinder)
- Phylogenetic reconstruction using Bayesian Inference (BI) or Maximun Likelihood (ML). (MrBayes for BI, and Garli or RAxML for ML)

The routine implements wrapper scripts in Python for reading each stage's output file and creating the corresponding configuration file of subsequent steps. Often, these kind of analyses are performed by wet lab biologists or researchers with little computational experience, consequently an automated approach would be helpful to obtain results in a short time, avoiding the software manipulation and learning curve. This software is also useful for researchers in molecular systematics or bioinformaticians with previous experience in the included tools. Although detailed knowledge of the included software is not mandatory to run Phypipe, the potential of this pipeline would be maximized if the researcher is familiar with each independent program. Thus, researchers can modify software parameters before running the complete pipeline and obtain results while saving time.

[1]Sadedin, S. P., Pope, B., Oshlack, A. (2012) Bpipe: A tool for running and managing bioinformatics pipelines. *Bioinformatics* 28(11) 1525-6. doi:10.1093/ bioinformatics/bts167

Poster #28.

Skip the line and balance your work with vQ

Gregory Zynda[*], Mehmet Dalkilic[†] and Matthew Vaughn[‡]

17th Annual Bioinformatics Open Source Conference (BOSC) 2016, Orlando, FL, USA

Repository: https://github.com/zyndagj/vQ
License: BSD 3-Clause

Processor core-counts are climbing in servers, computational accelerators, and even the phones in our pockets. The bioinformatics community takes advantage of this trend with threaded applications to utilize all available processor cores. While threading improves workflows at the workstation-level, they provide little benefit on high-performance supercomputers. Supercomputers, like Stampede at The Texas Advanced Computing Center and Comet at The San Diego Supercomputing Center, are not monolithic mainframes with immense resources that are shared between programs and users like on a traditional desktop. Instead, they are comprised of thousands of separate computers that can be reserved in groups to cooperatively work on computing tasks over a network. Sequence assembly is among the most arduous of bioinformatics tasks due to the substantial computational resources and large, complex datasets it requires. To reduce the time it takes to assemble a genome, developers have begun to develop multi-process workflows that align to using cluster architecture by submitting concurrent tasks as separate jobs to a centralized scheduler that manages resource usage across the system. Job schedulers usually implement a queue of some sort. This means that while concurrent job submissions are possible, the total run time for the task is limited by resource availability in that queue. And, resources are often in short supply. Most shared computing clusters, from department-level systems, up to national class systems such as those provided by the National Science Foundation, are oversubscribed with long queues. This renders any assumption that hundreds or thousands of job submissions can run to accomplish a given task totally impractical.

To improve the run times for such tasks, we developed the vQ, a virtual queue for multi-node jobs that will dynamically balance the execution of tasks issued with normal SLURM, SGE, TORQUE, and OpenLava (LSF) commands across a pre-allocated set of computing nodes on a shared resource. A distributed workflow system that previously relied on submitting and running thousands of jobs and/or sitting resident on a master node can now be structured as a single job which waits in a shared queue only once before commencing execution and does not unfairly consume shared resources on a login node. We show that this drop-in solution removes queue overhead and improves the experience of running complicated workflows like Trinity, SMRT, Falcon, Canu, Celera, and Cluster Flow. Most importantly, unlike Makeflow [3] or TACC's Launcher utilities which assume the tasks in a workflow can be described in a simple text file [2, 1], vQ requires no modifications to the original tools and works out-of-the-box with a minimal, user-level installation. Thus, vQ provides a potential solution to queue bottlenecks in current distributed bioinformatics tools.

References

[1] Victor Eijkhout. pylauncher: The python launcher. https://github.com/TACC/pylauncher, 2012.

[2] Lucas A Wilson and John M Fonner. Launcher: A shell-based framework for rapid development of parallel parametric studies. In *Proceedings of the 2014 Annual Conference on Extreme Science and Engineering Discovery Environment*, page 40. ACM, 2014.

[3] Li Yu, Christopher Moretti, Andrew Thrasher, Scott Emrich, Kenneth Judd, and Douglas Thain. Harnessing parallelism in multicore clusters with the all-pairs, wavefront, and makeflow abstractions. *Cluster Computing*, 13(3):243–256, 2010.

Poster #29.

Using the Nextflow framework for reproducible in-silico omics analyses across clouds and clusters

Paolo Di Tommaso[1], Evan Floden[1,2], Maria Chatzou[1,2] , Cedric Notredame[1]

[1] Centre for Genomic Regulation (CRG), Dr. Aiguader 88, 08003 Barcelona, Spain.
[2] Universitat Pompeu Fabra (UPF), 08003 Barcelona, Spain.
Email: paolo.ditommaso@crg.eu.

Project Website: http://www.nextflow.io
Source Code: https://github.com/nextflow-io/nextflow
License: GPLv3

Reproducibility has become one of biology's most pressing issues. This impasse has been fueled by the combined reliance on increasingly complex data analysis methods and the exponential growth of biological datasets. When considering the installation, deployment and maintenance of bioinformatic pipelines, an even more challenging picture emerges due to the lack of community standards. The effect of limited standards on reproducibility is amplified by the very diverse range of computational platforms and configurations on which these applications are expected to be applied (workstations, clusters, HPC, clouds, etc.). With no established standard at any level, diversity cannot be taken for granted.

Nextflow is a pipeline orchestration tool that has been designed to ease deployment and guarantee reproducibility across platforms. It is a computational environment, which provides a domain specific language (DSL) to simplify the writing of complex distributed computational workflows in a portable and replicable manner. It allows the seamless parallelization and deployment of any existing application with minimal development and maintenance overhead, irrespective of the original programming language.

The built-in support for container technologies such as Docker and Shifter, along with the native integration with the Git tool and popular code-sharing platforms like GitHub, make it possible to precisely prototype self-contained computational workflows, maintain all variations over time and rapidly reproduce any former configuration one may need to re-use. These capabilities guarantee consistent results over time and across different computing platforms.

Using a simple RNA-Seq based analysis we show how two seemingly irreproducible analyzes can be made stable across platforms when ported into Nextflow. Applying the transcript quantification tool Kallisto and the companion differential expression package Sleuth, we highlight how the same pipeline produces different results on Mac OSX and Linux. With a well studied dataset, 67 genes are only reported as significant on the Mac and 72 only reported in the Linux analysis out of a total of 6,138 differentially expressed genes. When the same pipeline was factored into Nextflow with Docker, the results were identical across platforms.