Published in final edited form as:
Genet Epidemiol. 2019 March ; 43(2): 227–237. doi:10.1002/gepi.22178.

System for Quality-Assured Data Analysis: Flexible, reproducible scientific workflows

Jerry Fowler[1], **Francis Anthony San Lucas**[2], and **Paul Scheet**[1]

[1]Department of Epidemiology, The University of Texas MD Anderson Cancer Center, Houston, Texas

[2]Molecular Diagnostic, The University of Texas MD Anderson Cancer Center, Houston, Texas

Abstract

The reproducibility of scientific processes is one of the paramount problems of bioinformatics, an engineering problem that must be addressed to perform good research. The System for Quality-Assured Data Analysis (SyQADA), described here, seeks to address reproducibility by managing many of the details of procedural bookkeeping in bioinformatics in as simple and transparent a manner as possible. SyQADA has been used by persons with backgrounds ranging from expert programmer to Unix novice, to perform and repeat dozens of diverse bioinformatics workflows on tens of thousands of samples, consuming over 80 CPU-months of computing on over 300,000 individual tasks of scores of projects on laptops, computer servers, and computing clusters. SyQADA is especially well-suited for paired-sample analyses found in cancer tumor-normal studies. SyQADA executable source code, documentation, tutorial examples, and workflows used in our lab is available from http://scheet.org/software.html.

Keywords

bioinformatics; cancer genomics; computer software; reproducibility; workflow

1 | INTRODUCTION

Biologists and geneticists are forming a consensus toward what scientists in the bioinformatics community have long appreciated, namely that the pedigree of an analysis result should be rigorously maintained, and managing the passage of data and analyses through a reproducible scientific workflow is essential to that goal (Leipzig, 2017; Nekrutenko & Taylor, 2012). While proper preprocessing of data and consistent method application are not hard problems in computer science, biology, genomics, or biostatistics, they are nontrivial engineering problems that must be addressed to perform reproducible

Correspondence Paul Scheet, Department of Epidemiology, 1515 Holcombe Blvd., Unit 1340, The University of Texas MD Anderson Cancer Center, Houston, Texas. pscheet@alum.wustl.edu.

CONFLICTS OF INTEREST
The authors declare that there are no conflicts of interest.

SUPPORTING INFORMATION
Additional supporting information may be found online in the Supporting Information section at the end of the article.

research and derive high-quality conclusions from experimental data. Our System for Quality-Assured Data Analysis (SyQADA; pronounced like the insect "cicada") seeks to address emerging standards for reproducible computational research (Sandve, Nekrutenko, Taylor, & Hovig, 2013) in as simple a framework as possible. SyQADA provides a bioinformatics framework that allows the specification of protocols that marshal and manage input data, allowing analyses to be run repeatably over large volumes of next-generation sequencing data for many samples and for different projects. One of SyQADA's chief attributes is the ability to trivially replicate computations on hundreds of samples at once and automatically confirm the completeness and consistency of all results. Additionally, SyQADA can run equally well on a laptop or computer server and can recognize the presence of cluster management software and make use of it. SyQADA attempts good-citizen use of resources whether run on a laptop, computer server, or computing cluster. On a laptop or computer server, this means running tasks with long compute time estimates at lower priority via the nice command. On a computing cluster, it means automatically identifying the right queue based on compute time and memory usage estimates and feeding tasks with large numbers of jobs into the cluster manager in parcels. Queueing constraints for load sharing facility (LSF) and portable batch system (PBS) clusters are configurable via local configuration files (standard files created for our clusters are provided as examples).

There are several related systems for managing bioinformatics workflows; from an analyst's perspective, they fall into the following four categories based on human interface:

Command-line interface (CLI) Bpipe (Sadedin, Pope & Oshlack, 2012), NGSANE (Buske, French, Smith, Clark & Bauer, 2014), Omics Pipe (Fisch et al., 2015);

Standalone application with graphic interface (app) NEAT (Schorderet, 2016), Chipster (Kallio et al., 2011), GenePattern (Kuehn, Liberzon, Reich & Mesirov, 2008);

Web application (webapp) Galaxy (Goecks et al., 2010), IMP (Narayanasamy et al., 2016);

Application program interface (API) GATK (McKenna et al., 2010), Queue (Shakir, 2011), Omics Pipe (Fisch et al., 2015), Ruffus (Goodstadt, 2010).

SyQADA is a CLI system whose only dependencies are a Unix operating system and a standard installation of Python 3.5 (or higher). This simplicity by design is shared by Bpipe, NGSANE and to some extent by Omics Pipe.

As elegant and useful as APIs, apps, and web apps may be, they often demand large learning curves of their users because of dependencies on hierarchical configuration languages (such as business process execution language [BPEL], extensible markup language [XML], or JavaScript object notation [JSON]), database management software (such as structured query language [SQL] or not only SQL [NoSQL]), or the implicit requirement of an API to write a program. In addition, the ergonomic demands of manipulating data through an app or web app are often great enough that a Unix user comfortable with ls, grep and sort may prefer those tools to an app or web app for identifying inputs, outputs, or errors. Furthermore, the use of a graphic interface introduces an additional source of user error and requires constant user focus to avoid introducing variation in the treatment of different samples. On the other hand, the cited command language interpreter systems are excellent examples of the advantages of the Keep It Simple Stupid (KISS) approach (ascribed to aircraft engineer Kelly Johnson [Dalzell, 2008]) also used by SyQADA. These KISS systems agree on the predicate that any serious computation requires knowledge of and interaction with a Unix shell (usually bash) and try to minimize their learning curves around that constraint.

The authors of Bpipe and NGSANE enumerate many attributes that are arguably true for all these KISS systems: transactional task management, automatic connection of stages, easy job restarting, audit trails, data security, reusability, adaptability, robust execution and monitoring, customization, repeated calls, and knowledge transfer.

Compared to related systems, there are two design principles of SyQADA that make it unique: (a) the explicit support of standards for reproducible analyses (see Supporting Information: SyQADA and Reproducibility) and (b) the goal of making adoption of these standards as simple as possible through a transparent and easy-to-use framework. A tabular comparison of the related CLI systems is found in Table 1. The Supporting Information contains a more detailed description of behaviors unique to SyQADA.

2 | METHODS

2.1 | Design and implementation of SyQADA

Consistent with the design goal of simplicity, the installation procedure for SyQADA requires merely downloading and extracting the executable source code zip file, whereas running it on a Unix system requires no other software besides a text editor and the analysis programs used in the workflow. SyQADA does not need or provide a programming interface. SyQADA maintains system state in the file system instead of in a database, software framework, or web application. Configuration files are simple property lists, and workflows are expressed as a list of tasks in a protocol file. Importantly, these features reduce development complexity and require no special skills of the user.

There are two main components of SyQADA: an Automator that generates and monitors workflows and a BatchRunner that creates and executes in parallel all the jobs for a given task (unless serial execution is explicitly requested). Defining a SyQADA workflow requires three files (see Supporting Information Example files): (a) a protocol file that identifies the tasks to be performed and designates a script template for each step; (b) a configuration file that defines the versions of the tool suite and reference files used as well as designating the source data directory, in which to find the named sample data; and (c) a sample file listing sample names to be processed. As illustrated in Figure 1, running a workflow consists of a single syqada auto command (the optional -project parameter identifies the required files if more than one protocol is present). When syqada auto is called, the Automator generates a numbered directory for each task and sequentially manages the execution of the workflow. For each task, a BatchRunner is created. The BatchRunner generates job scripts (usually one script per sample) using the specified bash script template. The template uses a simple syntax for specifying parameters that will be replaced with input and output filenames, task-dependent definitions, and sample names that vary with each invocation. BatchRunners also manage the good-citizen execution of those job scripts, either directly on the machine where SyQADA runs, or, if on a cluster, through submission to the cluster manager (LSF and PBS currently supported). Although SyQADA runs execute automatically, generated job scripts can easily be found and run manually if desired.

Upon task completion, the Automator checks the results of each job to ensure that all have completed successfully and produced the desired output files, before proceeding to the next task. If any of the generated scripts results in an error, SyQADA stops the workflow at the current task and reports back to the user simplified error messages, which can be interrogated in greater detail if necessary in audit logs. An example of an error might be a failure to write an output file due to lack of disk space. In such a scenario, a user could then remedy the problem by adding disk space or freeing up existing disk space. When the error has been resolved, running syqada repend [batchname] requests the BatchRunner to clean up individual jobs that failed to complete correctly and repend them to the pending submission queue. Then simply running syqada auto resumes the workflow, which is extremely convenient when running large, complex workflows.

SyQADA also provides features that facilitate execution of cancer analyses, which often demand “tumor-normal” comparisons. In a mode implemented for this specific purpose, SyQADA generates jobs using sample pairings that are defined in a “.tumor_normal” file, tracking those pairings for analysis. This can be generalized to other analysis models that require multiple samples per participant. SyQADA also supports small-scale parametric task replication. Given a file specifying a set of replication parameters and their respective values, SyQADA can create one task replicate (executable in parallel if desired) for each permutation of the value set and compare the results collectively in a summary step. Each replicate is its own task and must meet SyQADA’s normal completion criteria. Task replicates have been used for testing alternative program parameters, for varying random seeds, and for differential pre-processing of initial data to be fed to a pipeline, all of which can be helpful when evaluating new bioinformatic tools to incorporate into a workflow. A portion of the syqada tutorial introduction to replication that shows a replicated directory structure is found in Figure 2.

3 | RESULTS

3.1 | SyQADA in practice

Numerous researchers have used SyQADA workflows for a range of projects (e.g., Table 2). Perhaps among the most ambitious is benchmarking the allelic imbalance detection of hapLOH (Vattathil & Scheet, 2013) against the copy number analyses of The Cancer Genome Atlas (TCGA) on data from over 22,000 samples for the 31 TCGA diseases. We have also analyzed somatic variation in 452 lung cancer tumor-normal pairs, comprising nine workflows with a total of 53 distinct steps and 8714 distinct jobs. For this project, we constructed workflows to compare single nucleotide variants (SNVs) called by four different methods. SyQADA allowed us to run CPU-intensive sequence alignment and variant calling workflows in our research computing cluster, while running subsequent analysis workflows on our team compute servers, desktops or laptops, demonstrating the high degree of portability that SyQADA analyses and workflows have. The only difference between these environments was the source path to a team repository of executables and reference files, which was parameterized by a single shell variable.

We also frequently reuse SyQADA workflows created for one experimental project to perform the same analyses on other similar datasets. This is easily exemplified across consortial data sets, of which the TCGA is our largest example thus far. Any workflows developed and applied to one TCGA cancer type, can be applied almost trivially to dozens of other cancer types. In another example, a biologist with knowledge of a limited set of Unix commands (e.g., mkdir, cp, cd, and syqada auto) has performed several CPU-months of computation in a half-dozen projects by reusing the same SyQADA workflow while changing only the configuration file to use different microbiome source data. Of note, SyQADA has been used in the analysis of data for publications including analysis of somatic variation in sporadic MEN1 (Romero Arenas et al., 2014), Familial Adenomatous Polyposis (Borras et al., 2016), field of cancerization of lung adenocarcinoma (Jakubek et al., 2016), diet and oral microbiome (Hoffman et al., 2018), as well as in a comparison of somatic variant calling on Ion Torrent sequencing data (Deshpande et al., 2018).

In preparation for the 2017 AACR conference, SyQADA provided upstream analysis support for five presentations, among eight analysts, executing well over 100 pipeline instances comprising about 350 tasks, processing nearly 89,000 jobs that consumed over 70 CPU-months from two dedicated laboratory servers as well as one of the institutional computing clusters. Together, the projects analyzed well over 2700 samples from about 800 individuals, including more than 1100 tumor-normal pairs; data types included whole exome and 409-gene panel DNA sequence, RNA-seq, 16s ribosomal DNA, and whole genome microbiome sequence, as well as SNP microarrays, produced from Affymetrix, Illumina, and Ion Torrent technologies. Analyses included the Affymetrix birdsuite (Korn et al., 2008), the GATK best practices variant calling workflow, hapLOH, hapLOHseq, Ion Reporter (Thermo-Fisher Scientific, 2015), JLOH (Xia, Vattathil & Scheet, 2014), MaCH (Li, Willer, Ding, Scheet & Abecasis, 2010, MuTect (Cibulskis et al., 2013), the Tuxedo suite (Ghosh & Chan, 2016), VarScan2 (Koboldt et al., 2012), and an in-house pairwise phasing utility, as well as numerous programs for pre- and post-processing data.

For the ongoing hapLOH TCGA benchmarking study, we analyzed 22,674 Affymetrix SNP 6.0 (SNP6) samples from 11,128 individuals, using a workflow of 31 steps (including Affymetrix birdsuite, MACH, and hapLOH) and consuming over 40 CPU-weeks of our institutional computing cluster while processing more than 200,000 jobs. A tabular summary of some of the significant uses to which SyQADA has been put may be found in Tables 2 and 3.

4 | DISCUSSION

Bioinformatic workflow systems have been devised and propagated because of the demand induced by the explosion of molecular data from enhanced and scalable biotechnologies, such as next-generation sequencing and high-throughput “omics.” With an abundance of free-to-use software for specific components of analysis or quality control for such data come various preprocessing steps, version changes in software and often myriad parameter choices, all of which impact the final analytical product (e.g., inference, conclusions). Thus, to manage the options and support reproducible work we created SyQADA.

4.1 | Comparison of SyQADA with CLI systems

The desire to facilitate a scientist with modest or generic computing skills to safely conduct or fix bioinformatics analyses drove the design principles of simplicity (KISS) and transparency ("Avoid Magic," see below) used in SyQADA. The CLI systems cited (Bpipe, NGSANE, and Omics Pipe) were most likely motivated by this as well, they all tend to avoid the additional complexities of API, standalone graphic interface, or web application inherent in other workflow management systems for bioinformatics.

A second precept of SyQADA development, based on experiences with many different software packages, is Avoid Magic, by which we mean that the tool should not mystify its user or require complex knowledge merely to use the tool. In effect, the tool should not do anything a competent user could not do for herself so that the tool's behavior remains understandable. This is largely true of all these KISS tools, although SyQADA's ability to identify errors might seem magical if one is not aware of how simple the filesystem layout of SyQADA is. All the tools discussed require the user to learn a bit of tool-specific vocabulary and syntax. We believe SyQADA to be among the simplest in this regard.

Three of the systems require a user who wishes to write a new workflow to know how to write a simple bash script; Omics Pipe (Fisch et al., 2015 whose diverse set of packaged workflows deserves mention, requires construction of simple python modules. NGSANE requires the greatest bash knowledge, and also requires that the user know how to read and write a Unix makefile. This relatively straightforward skill is common knowledge among C programmers, but less common among bioinformaticists in general. In contrast, Bpipe and SyQADA each have simple home-grown syntax for specifying linear workflow. Where NGSANE uses pure bash for its workflow management logic, Bpipe uses groovy, whereas SyQADA and Omics Pipe use python.

SyQADA differs substantially from the other systems in the structure of its task management. In a design decision to Avoid Magic, the entire state of a SyQADA workflow is visible in the filesystem, so that the analyst can always complete his work manually if software development has not yet provided a feature or a bug halts progress.

4.2 | Comparison of SyQADA with web and stand-alone systems

Our design philosophy emphasizes simplicity at the expense of flexibility or power. This greatly reduces the time required to learn how to use the system and also seems to increase its adaptability to new analytical demands and settings.

The absence of a graph library in SyQADA stems from the observation that conditional logic is rarely appropriate to replicated task execution. SyQADA's only conditional branching is termination upon task failure. None of our users has seen this as a deficiency; without conditional branching it is easier to guarantee that all data are processed identically, a cornerstone of reproducible analysis.

In contrast to workflow systems (e.g., NGSANE and Omics Pipe) that rely on dependency graphs for execution or treat samples individually, SyQADA cannot optimize throughput by initiating next steps for completed jobs before all jobs for a given step have completed. On the other hand, SyQADA's execution dependency model is trivial, and there is no question in SyQADA about which step has produced a job failure. The time spent troubleshooting which specific task is not robustly executed might otherwise substantially outweigh the total processing time.

Also for reasons of simplicity, SyQADA makes no attempt to pipeline output from one step into input of the next step. This pays a price in file system I/O and short-term intermediate storage requirements. However, common usages such as piping the output of SAMtools into VarScan2, or pipelining BEDtools commands, can be defined in bash templates that are reproducibly executed as a single SyQADA task.

As mentioned, SyQADA has no web interface (although it will report progress through updates to a static web page if desired). Whether the sole use of CLI is a strength or a weakness is a matter of taste, but adding a web interface would raise the height of the software dependency stack greatly, adding to system complexity, not only increasing the chance of bugs but also introducing security concerns that are inherent in web applications.[1]

SyQADA has made no direct attempt to address "cloud" computing in contrast to Galaxy, IMP, and Omics Pipe (Fisch et al., 2015; Goecks et al., 2010; Narayanasamy et al., 2016), among others. Those with institutional access to large-scale computing resources may not yet have encountered a need for this. However, nested SyQADA protocols for data transfer and subsequent computation have been written and used, so the use of SyQADA in a cloud environment is easily conceivable.

4.3 | Special attributes of SyQADA

SyQADA began as a skunkworks project to speed up and ensure the quality and reproducibility of large-scale genomic data analyses in as simple a manner as possible. On-demand feature addition led to a number of useful features that may distinguish SyQADA from its KISS cousins. Of importance is guaranteeing that every sample completed each step correctly, a predicate of SyQADA development. Other distinctions include generic tumor-normal comparison, error classification, splitting tasks by chromosome, simple parametric replication of tasks, automatic use of cluster submission when present and local execution otherwise, the ability to nest protocols, use of samples' phenotypic attributes as script parameters, a protocol validation tool and the presence of an expanding workflow repository that includes support for computational cancer genomics. These existing workflows give users the ability quickly to implement many high-throughput data analyses. A chart describing the scope of several of these is found in Supporting Information Table 1. The workflow library provided in the software bundle contains all the configuration specifications necessary to run several protocols (users must modify the configuration file to

[1]Interpretation of the seven-level Open Systems Interconnect (OSI) stack may vary, but from one perspective, the OSI footprint of Bpipe, NGSANE, and SyQADA would be only the Physical (1), Data-link(2), and Application(7) layers, with the host operating system collapsing the other four; the addition of a web interface necessarily implies the existence of the Network(3), Transport(4), Session(5) and Presentation(6) layers.

specify their own executable paths). The SyQADA manual and tutorials are bundled with SyQADA and can be viewed by invoking syqada manual or syqada tutorial. The manual is online at scheetlabsoftware.org/syqada/docs/index.html. Small portions of each of the tutorials are reproduced in Figures 3, 4, and 5.

In summary, SyQADA performs many common activities necessary to the reliable and repeatable management and execution of sequential tasks on large volumes of data, imposing simple conventions on results and handling common failure modes gracefully.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

ACKNOWLEDGMENTS

This study has been supported in part by National Institutes of Health (NIH) Grants R01HG005859 and P30CA016672 as well as Cancer Prevention and Research Institute of Texas (CPRIT) grants RP150079 and RP160668. The authors Thank Addy Deshpande, Lili Huang, Kyle Chang, Sasha Jakubek, Yihua Liu, Smruthy Sivakumar, Tina McDowell, Ryan Bohlender, Kristi Hoffman, and Humam Kadara for their enthusiasm for SyQADA and their helpful comments on software and article, and to Matthew Giordano for his engineering contributions. The authors acknowledge the High Performance Research Computing Center at the University of Texas, MD Anderson Cancer Center for providing HPC and computational resources (including consulting support) that have contributed to results reported in this article.

Funding information

National Human Genome Research Institute, Grant/Award Number: R01HG005859; Cancer Prevention and Research Institute of Texas, Grant/Award Numbers: RP150079, RP160668; National Cancer Institute, Grant/Award Number: P30CA016672

REFERENCES

Borras E, San Lucas FA, Chang K, Zhou R, Masand G, Fowler J, & Vilar E (2016). Genomic landscape of colorectal mucosa and adenomas. Cancer Prevention Research (Philadelphia, Pa.), 9(6), 417–427.

Buske FA, French HJ, Smith MA, Clark SJ, & Bauer DC (2014). NGSANE: A lightweight production informatics frame-work for high-throughput data analysis. Bioinformatics, 30(10), 1471–1472. [PubMed: 24470576]

Cibulskis K, McKenna A, Fennell T, Banks E, DePristo M, & Getz G (2011). ContEst: Estimating cross-contamination of human samples in next-generation sequencing data. Bioinformatics, 27(18), 2601–2602. [PubMed: 21803805]

Cibulskis K, Lawrence MS, Carter SL, Sivachenko A, Jaffe D, Sougnez C, & Getz G (2013). Sensitive detection of somatic point mutations in impure and heterogeneous cancer samples. Nature Biotechnology, 31(3), 213–219.

Dalzell T (2008). The Routledge Dictionary of Modern American Slang and Unconventional English. New York, NY: Routledge.

Deshpande A, Lang W, McDowell T, Sivakumar S, Zhang J, Wang J, & Scheet P (2018). Strategies for identification of somatic variants using the Ion Torrent deep targeted sequencing platform. BMC Bioinformatics, 19(1), 5. [PubMed: 29301485]

Fisch KM, Meissner T, Gioia L, Ducom JC, Carland TM, Loguercio S, & Su AI (2015). Omics Pipe: A community-based framework for reproducible multi-omics data analysis. Bioinformatics, 31(11), 1724–1728. [PubMed: 25637560]

Friedman J, & Alm EJ (2012). Inferring correlation networks from genomic survey data. PLoS Computational Biology, 8(9), e1002687.

Ghosh S, & Chan CK (2016). Analysis of RNA-Seq data using TopHat and Cufflinks. Methods in Molecular Biology, 1374, 339–361. [PubMed: 26519415]

Goecks J, Nekrutenko A, Taylor J, Afgan E, Ananda G, Baker D, & Vincent K (2010). Galaxy: A comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. Genome Biology, 11(8), R86. [PubMed: 20738864]

Goodstadt L (2010). Ruffus: A lightweight Python library for computational pipelines. Bioinformatics, 26(21), 2778–2779. [PubMed: 20847218]

Hoffman K, Hutchinson DS, Fowler J, Smith DP, Ajami NJ, Zhao H, & Daniel CR (2018). Oral microbiota reveals signs of acculturation in Mexican American women. PLoS One, 13(4) 10.1371/journal.pone.0194100

Institute TB (2013). Best practice variant detection with the gatk v4, for release 2.6. https://www.broadinstitute.org/partnerships/education/broade/best-practices-variant-calling-gatk-0

Jakubek Y, Lang W, Vattathil S, Garcia M, Xu L, Huang L, & Kadara H (2016). Genomic landscape established by allelic imbalance in the cancerization field of a normal appearing airway. Cancer Research, 76(13), 3676–3683. [PubMed: 27216194]

Kallio MA, Tuimala JT, Hupponen T, Klemela P, Gentile M, Scheinin I, & Korpelainen EI (2011). Chipster: User-friendly analysis software for microarray and other high- throughput data. BMC Genomics, 12, 507. [PubMed: 21999641]

Koboldt D, Zhang Q, Larson D, Shen D, McLellan M, Lin L, & Wilson R (2012). Varscan 2: Somatic mutation and copy number alteration discovery in cancer by exome sequencing. Genome Research, 22, 568–576. [PubMed: 22300766]

Korn JM, Kuruvilla FG, McCarroll SA, Wysoker A, Nemesh J, Cawley S, & Altshuler D (2008). Integrated genotype calling and association analysis of SNPs, common copy number polymorphisms and rare CNVs. Nature Genetics, 40(10), 1253–1260. [PubMed: 18776909]

Kuehn H, Liberzon A, Reich M, & Mesirov JP (2008). Using GenePattern for gene expression analysis. Current Protocols in Bioinformatics, 10.1002/0471250953.bi0712s22

Leipzig J (2017). A review of bioinformatic pipeline frameworks. Briefings in Bioinformatics, 18(3), 530–536. [PubMed: 27013646]

Li Y, Willer CJ, Ding J, Scheet P, & Abecasis GR (2010). MaCH: Using sequence and genotype data to estimate haplotypes and unobserved genotypes. Genetic Epidemiology, 34(8), 816–834. [PubMed: 21058334]

Lucas FS, Wang G, Scheet P, & Peng B (2012). Integrated annotation and analysis of genetic variants from next-generation sequencing studies with variant tools. Bioinformatics, 28(3), 421–422. [PubMed: 22138362]

McKenna A, Hanna M, Banks E, Sivachenko A, Cibulskis K, Kernytsky A, & DePristo MA (2010). The genome analysis toolkit: A MapReduce framework for analyzing next-generation DNA sequencing data. Genome Research, 20(9), 1297–1303. [PubMed: 20644199]

Narayanasamy S, Jarosz Y, Muller EE, Heintz-Buschart A, Herold M, Kaysen A, & Wilmes P (2016). IMP: A pipeline for reproducible reference-independent integrated metagenomic and metatranscriptomic analyses. Genome Biology, 17(1), 260. [PubMed: 27986083]

Nekrutenko A, & Taylor J (2012). Next-generation sequencing data interpretation: Enhancing reproducibility and accessibility. Nature Reviews Genetics, 13(9), 667–672.

Purcell S, Neale B, Todd-Brown K, Thomas L, Ferreira MA, Bender D, & Sham PC (2007). PLINK: A toolset for whole- genome association and population-based linkage analyses. The American Journal of Human Genetics, 81(3), 559–575. [PubMed: 17701901]

Romero Arenas MA, Fowler RG, San Lucas FA, Shen J, Rich TA, Grubbs EG, & Zhao H (2014). Preliminary whole-exome sequencing reveals mutations that imply common tumorigenicity pathways in multiple endocrine neoplasia type 1 patients. Surgery, 156(6), 1351–1357. [PubMed: 25456907]

Sadedin SP, Pope B, & Oshlack A (2012). Bpipe: A tool for running and managing bioinformatics pipelines. Bioinformatics, 28(11), 1525–1526. [PubMed: 22500002]

San Lucas FA, Sivakumar S, Vattathil S, Fowler J, Vilar E, & Scheet P (2016). Rapid and powerful detection of subtle allelic imbalance from exome sequencing data with hapLOHseq. Bioinformatics.

Sandve GK, Nekrutenko A, Taylor J, & Hovig E (2013). Ten simple rules for reproducible computational research. PLoS Comput. Biol., 9(10), e1003285.

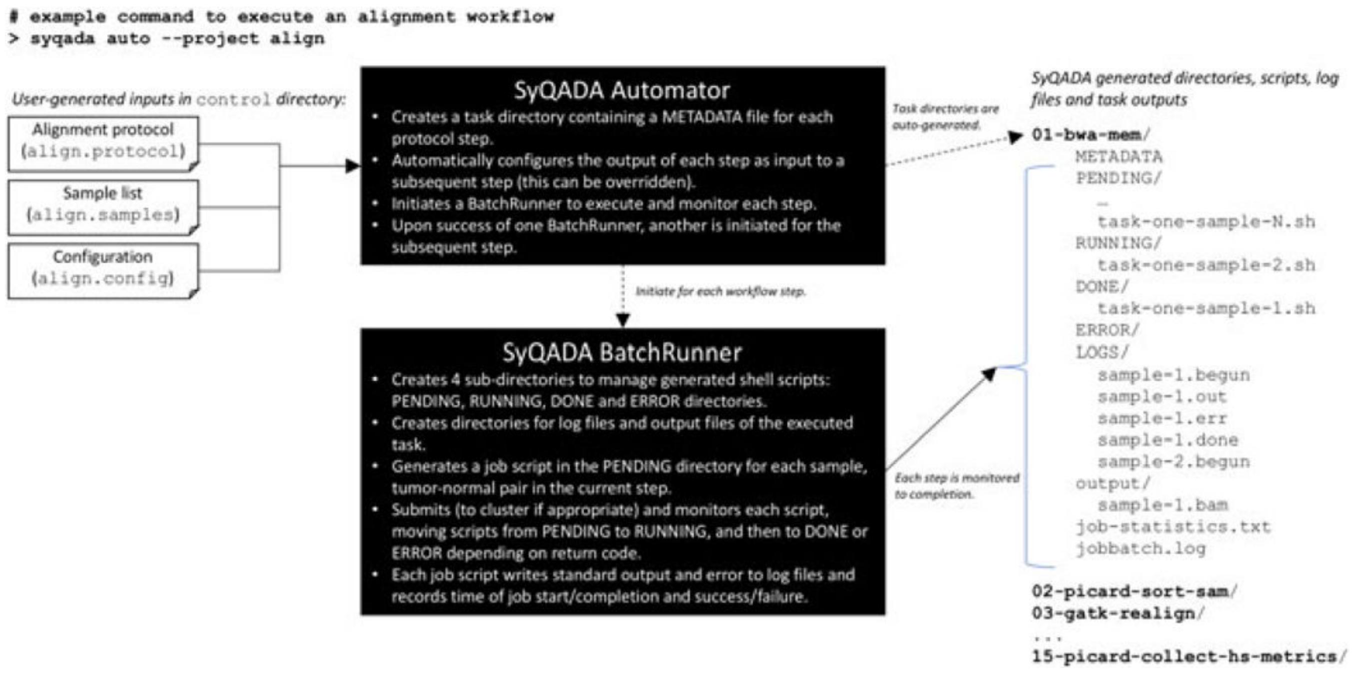
Schorderet P (2016). NEAT: A framework for building fully automated NGS pipelines and analyses. BMC Bioinformatics, 17, 53. [PubMed: 26830846]

Scientific ThermoFisher. (2015). About ion Reporter Software.

Shakir K (2011). GATK-Queue: Command-line job manager and scripting framework for multi-stage genomic analysis. https://www.broadinstitute.org/files/shared/mpg/nextgen2011/nextgen2011_shakir.pdf

Vattathil S, & Scheet P (2013). Haplotype-based profiling of subtle allelic imbalance with SNP arrays. Genome Research, 23(1), 152–158. [PubMed: 23028187]

Xia R, Vattathil S, & Scheet P (2014). Identification of allelic imbalance with a statistical model for subtle genomic mosaicism. PLoS Computational Biology, 10(8), e1003765.


```
# example command to execute an alignment workflow
> syqada auto --project align
```

SyQADA Automator

- Creates a task directory containing a METADATA file for each protocol step.
- Automatically configures the output of each step as input to a subsequent step (this can be overridden).
- Initiates a BatchRunner to execute and monitor each step.
- Upon success of one BatchRunner, another is initiated for the subsequent step.

SyQADA BatchRunner

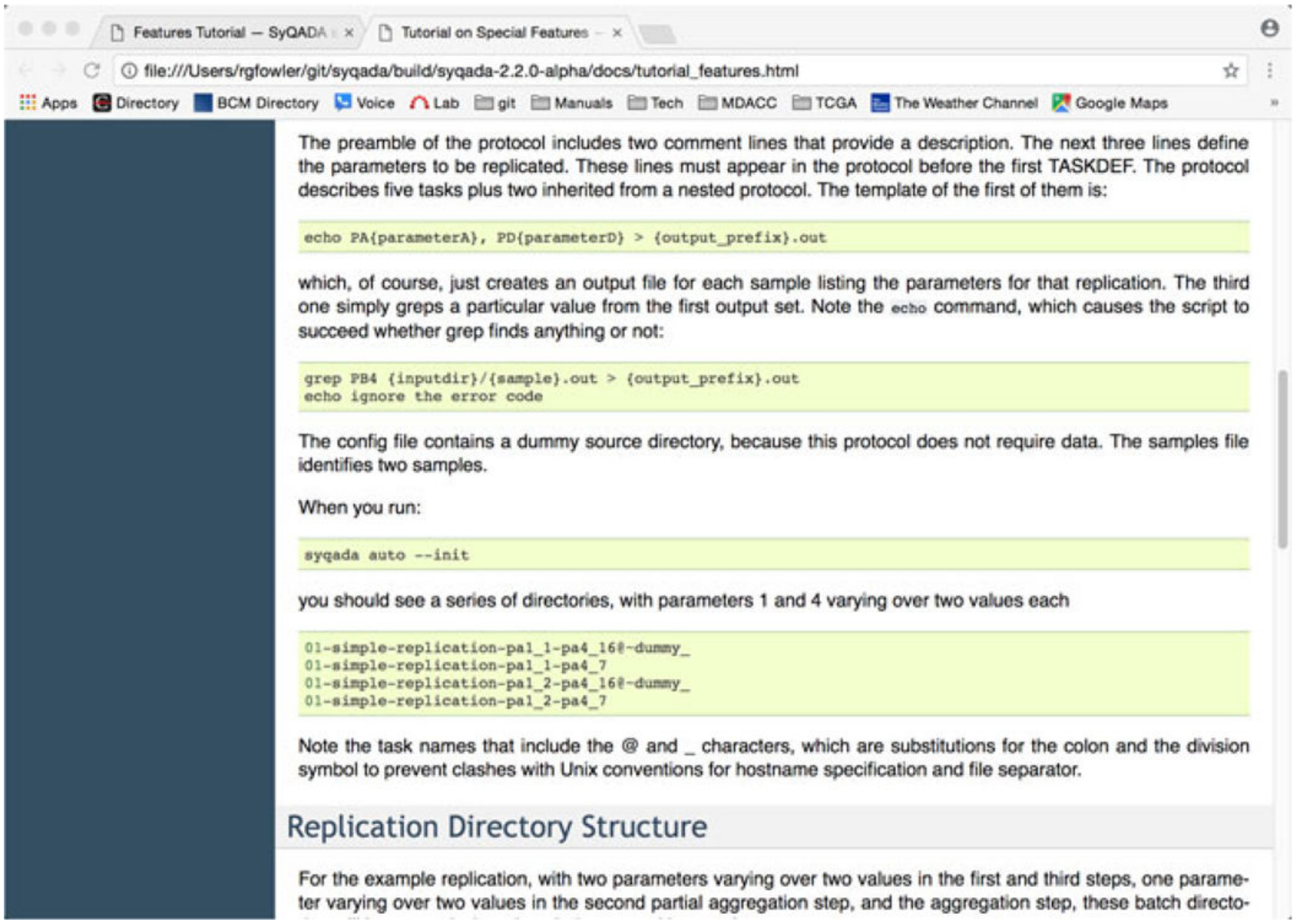
- Creates 4 sub-directories to manage generated shell scripts: PENDING, RUNNING, DONE and ERROR directories.
- Creates directories for log files and output files of the executed task.
- Generates a job script in the PENDING directory for each sample, tumor-normal pair in the current step.
- Submits (to cluster if appropriate) and monitors each script, moving scripts from PENDING to RUNNING, and then to DONE or ERROR depending on return code.
- Each job script writes standard output and error to log files and records time of job start/completion and success/failure.

```
01-bwa-mem/
    METADATA
    PENDING/
      …
      task-one-sample-N.sh
    RUNNING/
      task-one-sample-2.sh
    DONE/
      task-one-sample-1.sh
    ERROR/
    LOGS/
      sample-1.begun
      sample-1.out
      sample-1.err
      sample-1.done
      sample-2.begun
    output/
      sample-1.bam
    job-statistics.txt
    jobbatch.log

02-picard-sort-sam/
03-gatk-realign/
...
15-picard-collect-hs-metrics/
```

User-generated inputs in control directory:
Alignment protocol (align.protocol)
Sample list (align.samples)
Configuration (align.config)
Task directories are auto-generated.
Initiate for each workflow step.
Each step is monitored to completion.
SyQADA generated directories, scripts, log files and task outputs

FIGURE 1.
Example SyQADA workflow for NGS sequence alignment



Features Tutorial — SyQADA | Tutorial on Special Features

file:///Users/rgfowler/git/syqada/build/syqada-2.2.0-alpha/docs/tutorial_features.html

The preamble of the protocol includes two comment lines that provide a description. The next three lines define the parameters to be replicated. These lines must appear in the protocol before the first TASKDEF. The protocol describes five tasks plus two inherited from a nested protocol. The template of the first of them is:

```
echo PA{parameterA}, PD{parameterD} > {output_prefix}.out
```

which, of course, just creates an output file for each sample listing the parameters for that replication. The third one simply greps a particular value from the first output set. Note the `echo` command, which causes the script to succeed whether grep finds anything or not:

```
grep PB4 {inputdir}/{sample}.out > {output_prefix}.out
echo ignore the error code
```

The config file contains a dummy source directory, because this protocol does not require data. The samples file identifies two samples.

When you run:

```
syqada auto --init
```

you should see a series of directories, with parameters 1 and 4 varying over two values each

```
01-simple-replication-pa1_1-pa4_16@-dummy_
01-simple-replication-pa1_1-pa4_7
01-simple-replication-pa1_2-pa4_16@-dummy_
01-simple-replication-pa1_2-pa4_7
```

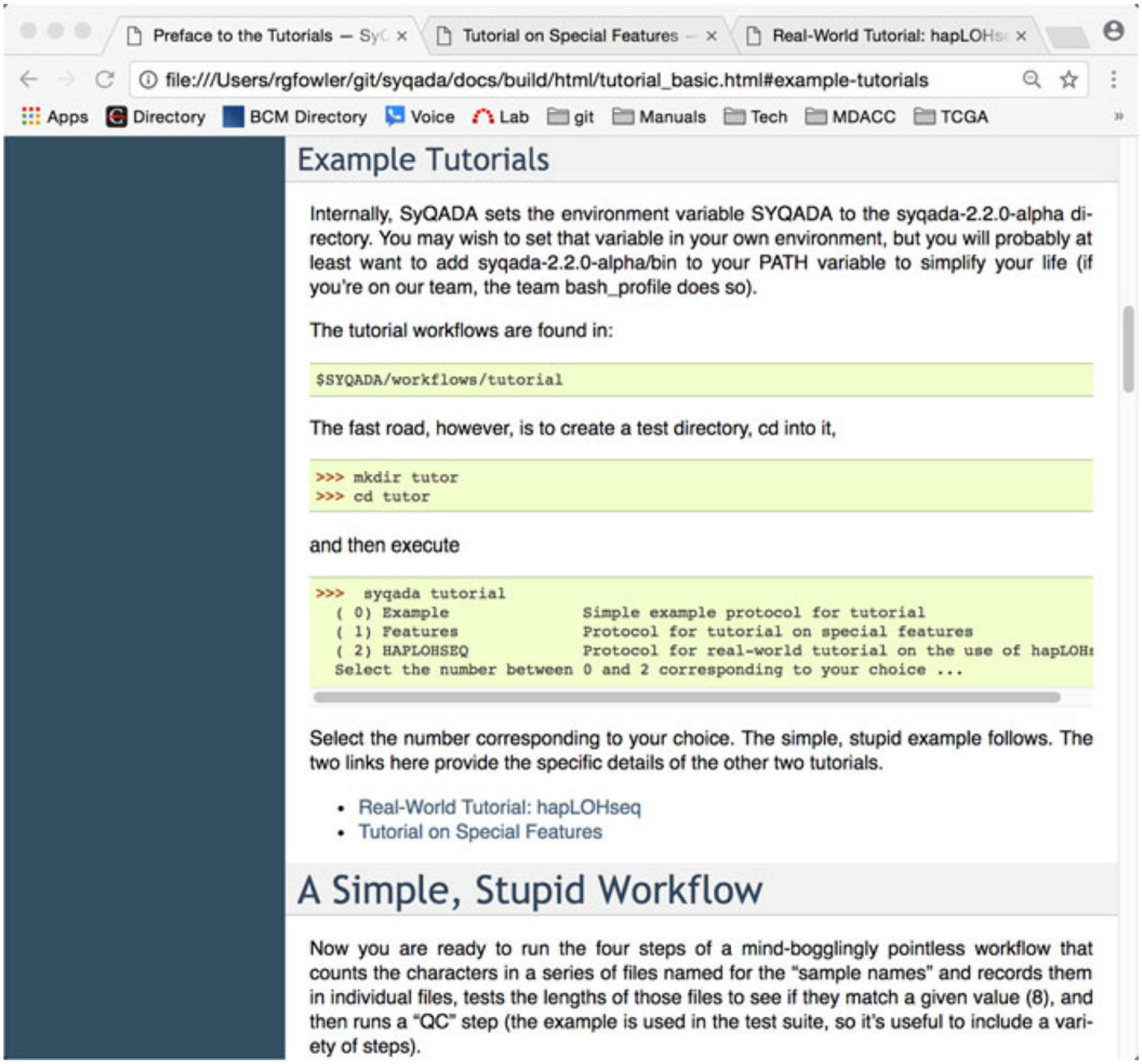
Note the task names that include the @ and _ characters, which are substitutions for the colon and the division symbol to prevent clashes with Unix conventions for hostname specification and file separator.

Replication Directory Structure

For the example replication, with two parameters varying over two values in the first and third steps, one parameter varying over two values in the second partial aggregation step, and the aggregation step, these batch directo-

FIGURE 2.

Initial file structure created by replication tutorial


Preface to the Tutorials — SyQ
Tutorial on Special Features —
Real-World Tutorial: hapLOHs
file:///Users/rgfowler/git/syqada/docs/build/html/tutorial_basic.html#example-tutorials
Apps
Directory
BCM Directory
Voice
Lab
git
Manuals
Tech
MDACC
TCGA

Example Tutorials

Internally, SyQADA sets the environment variable SYQADA to the syqada-2.2.0-alpha directory. You may wish to set that variable in your own environment, but you will probably at least want to add syqada-2.2.0-alpha/bin to your PATH variable to simplify your life (if you're on our team, the team bash_profile does so).

The tutorial workflows are found in:

```
$SYQADA/workflows/tutorial
```

The fast road, however, is to create a test directory, cd into it,

```
>>> mkdir tutor
>>> cd tutor
```

and then execute

```
>>>  syqada tutorial
  ( 0) Example                 Simple example protocol for tutorial
  ( 1) Features                Protocol for tutorial on special features
  ( 2) HAPLOHSEQ               Protocol for real-world tutorial on the use of hapLOHs
  Select the number between 0 and 2 corresponding to your choice ...
```

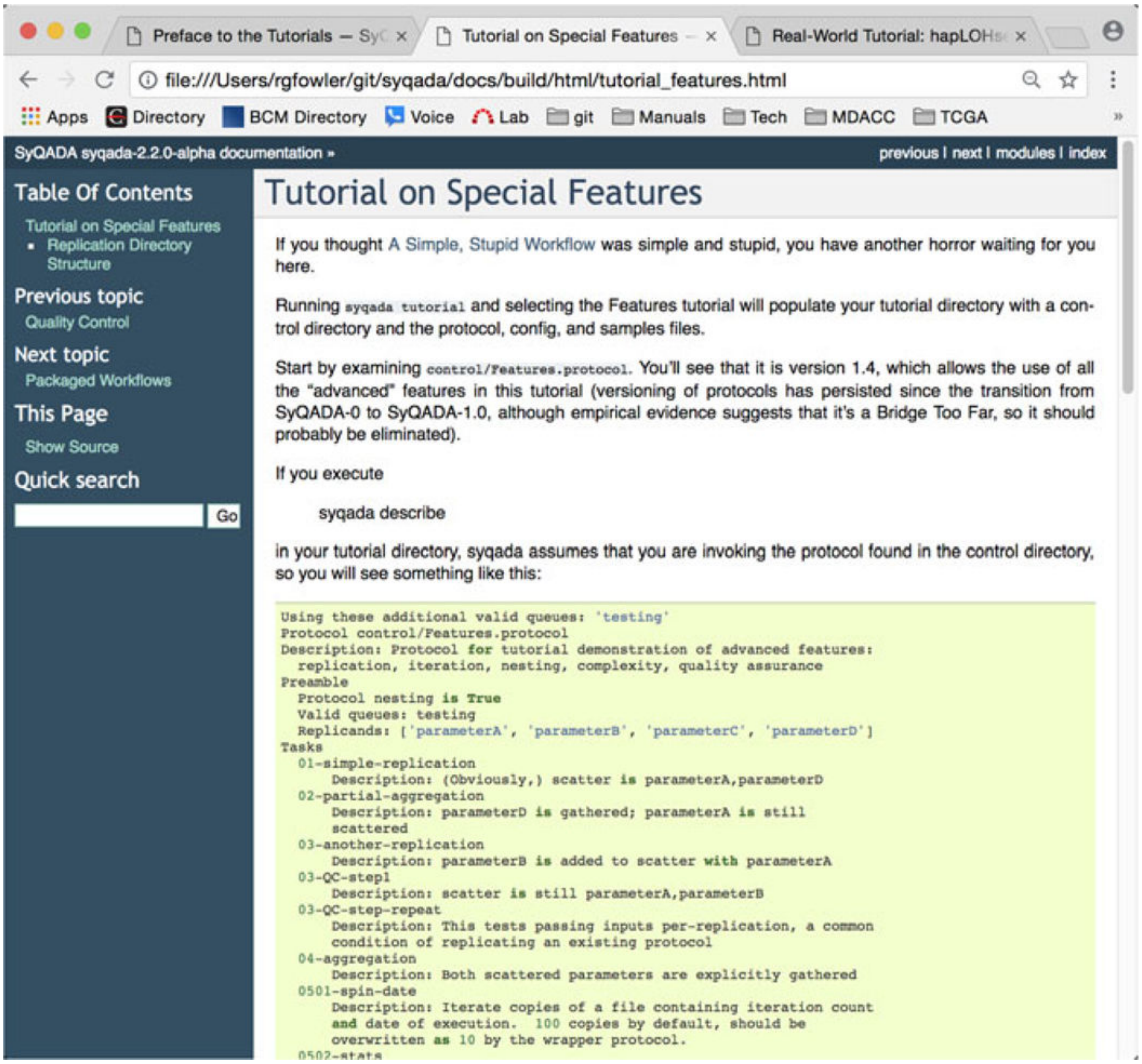
Select the number corresponding to your choice. The simple, stupid example follows. The two links here provide the specific details of the other two tutorials.

- Real-World Tutorial: hapLOHseq
- Tutorial on Special Features

A Simple, Stupid Workflow

Now you are ready to run the four steps of a mind-bogglingly pointless workflow that counts the characters in a series of files named for the "sample names" and records them in individual files, tests the lengths of those files to see if they match a given value (8), and then runs a "QC" step (the example is used in the test suite, so it's useful to include a variety of steps).

FIGURE 3.
Introduction to the basic tutorial


Preface to the Tutorials — SyQ
Tutorial on Special Features —
Real-World Tutorial: hapLOHs
file:///Users/rgfowler/git/syqada/docs/build/html/tutorial_features.html
Apps
Directory
BCM Directory
Voice
Lab
git
Manuals
Tech
MDACC
TCGA
SyQADA syqada-2.2.0-alpha documentation »
previous | next | modules | index
Table Of Contents
Tutorial on Special Features
Replication Directory Structure
Previous topic
Quality Control
Next topic
Packaged Workflows
This Page
Show Source
Quick search
Go

Tutorial on Special Features

If you thought A Simple, Stupid Workflow was simple and stupid, you have another horror waiting for you here.

Running `syqada tutorial` and selecting the Features tutorial will populate your tutorial directory with a control directory and the protocol, config, and samples files.

Start by examining `control/Features.protocol`. You'll see that it is version 1.4, which allows the use of all the "advanced" features in this tutorial (versioning of protocols has persisted since the transition from SyQADA-0 to SyQADA-1.0, although empirical evidence suggests that it's a Bridge Too Far, so it should probably be eliminated).

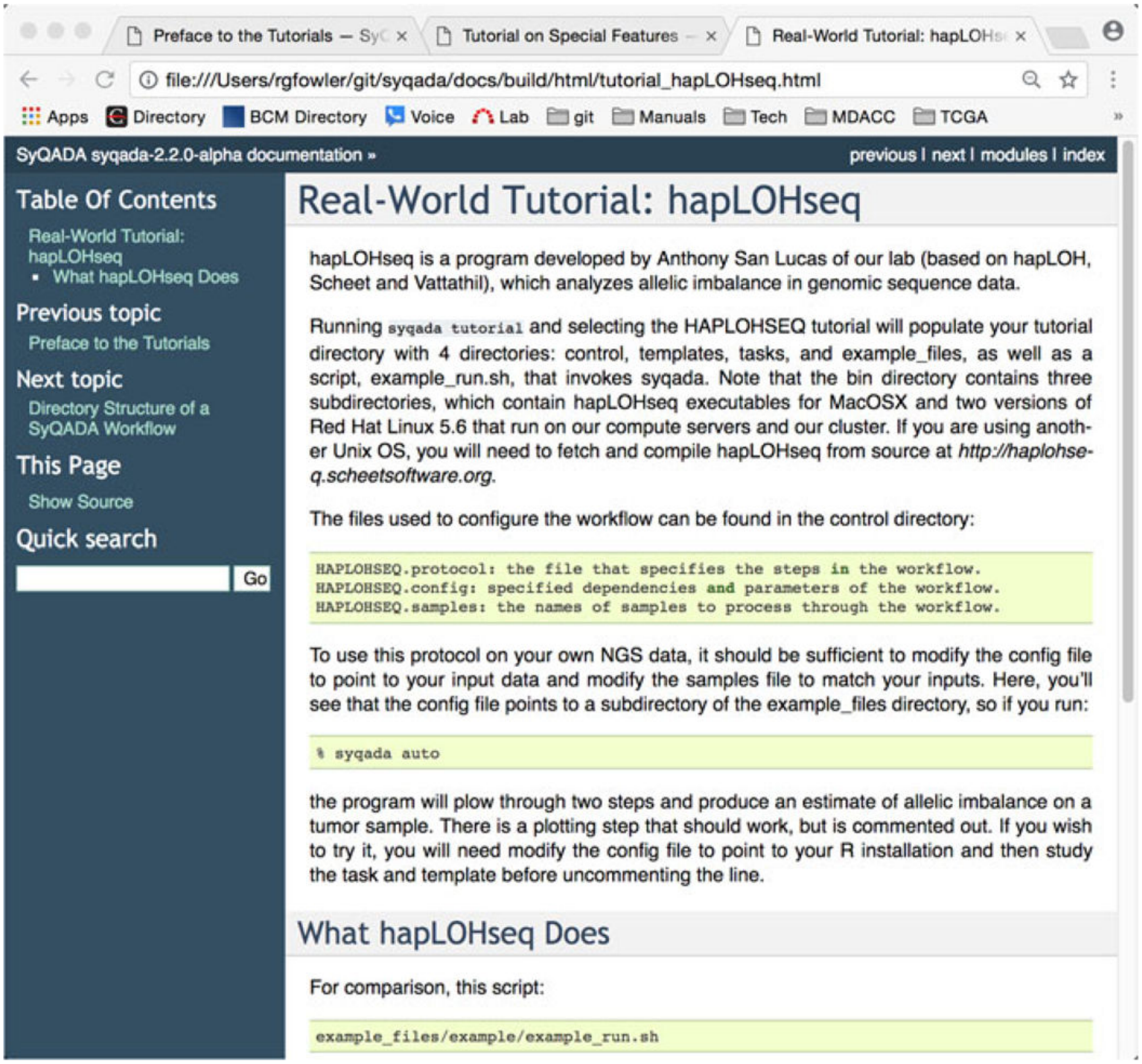
If you execute

```
syqada describe
```

in your tutorial directory, syqada assumes that you are invoking the protocol found in the control directory, so you will see something like this:

```
Using these additional valid queues: 'testing'
Protocol control/Features.protocol
Description: Protocol for tutorial demonstration of advanced features:
  replication, iteration, nesting, complexity, quality assurance
Preamble
  Protocol nesting is True
  Valid queues: testing
  Replicands: ['parameterA', 'parameterB', 'parameterC', 'parameterD']
Tasks
  01-simple-replication
      Description: (Obviously,) scatter is parameterA,parameterD
  02-partial-aggregation
      Description: parameterD is gathered; parameterA is still
      scattered
  03-another-replication
      Description: parameterB is added to scatter with parameterA
  03-QC-step1
      Description: scatter is still parameterA,parameterB
  03-QC-step-repeat
      Description: This tests passing inputs per-replication, a common
      condition of replicating an existing protocol
  04-aggregation
      Description: Both scattered parameters are explicitly gathered
  0501-spin-date
      Description: Iterate copies of a file containing iteration count
      and date of execution.  100 copies by default, should be
      overwritten as 10 by the wrapper protocol.
  0502-stats
```

FIGURE 4.
Introduction to the tutorial on special features


Preface to the Tutorials — SyC
Tutorial on Special Features —
Real-World Tutorial: hapLOHs
file:///Users/rgfowler/git/syqada/docs/build/html/tutorial_hapLOHseq.html
Apps
Directory
BCM Directory
Voice
Lab
git
Manuals
Tech
MDACC
TCGA
SyQADA syqada-2.2.0-alpha documentation »
previous | next | modules | index
Table Of Contents
Real-World Tutorial: hapLOHseq
What hapLOHseq Does
Previous topic
Preface to the Tutorials
Next topic
Directory Structure of a SyQADA Workflow
This Page
Show Source
Quick search
Go

Real-World Tutorial: hapLOHseq

hapLOHseq is a program developed by Anthony San Lucas of our lab (based on hapLOH, Scheet and Vattathil), which analyzes allelic imbalance in genomic sequence data.

Running `syqada tutorial` and selecting the HAPLOHSEQ tutorial will populate your tutorial directory with 4 directories: control, templates, tasks, and example_files, as well as a script, example_run.sh, that invokes syqada. Note that the bin directory contains three subdirectories, which contain hapLOHseq executables for MacOSX and two versions of Red Hat Linux 5.6 that run on our compute servers and our cluster. If you are using another Unix OS, you will need to fetch and compile hapLOHseq from source at *http://haplohseq.scheetsoftware.org*.

The files used to configure the workflow can be found in the control directory:

```
HAPLOHSEQ.protocol: the file that specifies the steps in the workflow.
HAPLOHSEQ.config: specified dependencies and parameters of the workflow.
HAPLOHSEQ.samples: the names of samples to process through the workflow.
```

To use this protocol on your own NGS data, it should be sufficient to modify the config file to point to your input data and modify the samples file to match your inputs. Here, you'll see that the config file points to a subdirectory of the example_files directory, so if you run:

```
% syqada auto
```

the program will plow through two steps and produce an estimate of allelic imbalance on a tumor sample. There is a plotting step that should work, but is commented out. If you wish to try it, you will need modify the config file to point to your R installation and then study the task and template before uncommenting the line.

What hapLOHseq Does

For comparison, this script:

```
example_files/example/example_run.sh
```

FIGURE 5.
Introduction to the real-world tutorial

TABLE 1

Comparison of command-line workflow systems

	SyQADA	**Bpipe**	**NGSANE**	**Omics Pipe**
General				
Flow Model	Linear	Simple graph	Unix make	Ruffus graph
Tracking	structured	flat	flat	flat
Dependencies	bash,python3.5	python	bash, make	Ruffus, Sumatra...
Bpipe traits				
bash script	almost as-is	almost as-is	as-is	python module
Other attributes	Supported	Supported	Supported	Supported
NGSANE traits				
Reproducibility	Yes	?	Yes	Yes
Checkpointing	Yes	No	Yes	No
Auto summary	Status only	No	Detailed	No
Other attributes	Supported	Supported	Supported	Supported
Omics traits				
Docker	Under test	No	No	Yes
Version control	Partial	No	No	Yes
SyQADA traits				
Cluster agnostic	Yes	No	No	No
Consistency check	Yes	No	No	No
Sample pairing	Explicit	No	No	No
Error handling	Assisted	No	No	No
Nested workflow	Yes	?	No	No
Replication	Yes	No	No	No
Validation aids	Yes	No	No	No
Sample handling	1000s	10s	10s	10s

TABLE 2

Partial catalog of projects and SyQADA workflows used in them

	Used in project (number of samples/tumor-normal pairs)								
Workflow steps	**Misc**	**MEN1**	**FAP**	**TCGA**	**Tech**	**LUAD**	**SKCM**	**PDAC**	**Diet**
GATK-Var 16	143/	32/							
Annotation 7		32/16			34/30	500/452	48/33		
MuTect 2		32/16	53/38		34/30	500/452	48/33		
varscan2 2		32/16	53/38		34/30	500/452	48/33		
IR-somatic 1						305/278			
mm-MuTect 11						39/16	15/11		
Somatic4 16					34/30	500/452	48/33		
hapLOHseq 4		32/16						/56	
SamQC-1 5								1452	
NestContest 10								1452	
SamQC-2 3								1452	
SparCC 10									7×66
SparCC-R 10				LUAD 1119/591					1×66×8
				BRCA 1646/815					
AI-Bench 10				OV 1798/898					
				SKCM 864/433					
				(All) 23K/12K					

Note. Projects and workflows are described in Table 3. Shaded cells in this table are associated with team members and colleagues who were responsible for using the SyQADA workflow on the sample set of a given project. Those in light gray are two bioinformaticists experienced in Unix; in medium gray are two bioinformaticists with little previous Unix experience; and in the dark gray are two students, a post-doctoral fellow, and a principal investigator with no prior exposure to Unix. Unshaded workflows were performed by the authors.

TABLE 3

Project and workflow descriptions for labels in Table 2

Project descriptions including publications	
Misc	Three small germline variation studies
MEN1	Somatic variation in sporadic MEN1 (Romero Arenas et al., 2014)
FAP	Somatic variation in Familial Adenopolyposis (Borras et al., 2016).
TCGA-AI	Benchmarking of hapLOH in TCGA
Tech	Variant caller comparison for Ion Torrent sequencing (Deshpande et al., 2018)
LUAD	Somatic variation of Lung adenocarcinoma (Jakubek et al., 2016)
SKCM	Somatic variation in the field of cancerization of melanoma
PDAC	Genetic variation in pancreatic adenocarcinoma
Diet	Epidemiological studies of oral and gut microbiome (Hoffman et al., 2018)
Workflow descriptions	
GATK-Var	GATK Best Practices workflow for sequence alignment and variant calling (Institute, 2013)
Annotation	variant_tools workflow for annotation of project-wide single-nucleotide variants (Lucas et al., 2012)
MuTect	GATK and MuTecttasks for pairwise detection of somatic mutation, optionally parallelized by splitting input by chromosome
varscan2	VarScan2 (Koboldt et al., 2012) tasks for pairwise detection of somatic mutation
IR-somatic	Generation of project files for the Ion Reporter somatic mutation detector (ThermoFisher Scientific, 2015)
mm-MuTect	GATK Best Practices workflow for Mus musculus sequence alignment against build mm10 plus pairwise somatic variant calling of tumor-normal pairs
Somatic4	Common filtration and summary processing for four somatic variant callers
AI-Bench	Birdsuite processing of Affymetrix SNP6 chip data to prepare for and run Mach, with post-processing to prepare for and run hapLOH (Vattathil & Scheet, 2013), followed by event detection and plotting of comparisons with TCGA copy number variation calls
hapLOHseq	GATK variant calling of somatic and germline sequence against 1K genomes variant sites, followed by Mach phasing and hapLOHseq allelic imbalance detection (San Lucas et al., 2016)
SparCC	Extraction of genus or species-level microbiome prevalence data, execution of SparCC (Friedman & Alm, 2012), generation of 500 permutations of the data, and execution of SparCC to obtain p-values, filtration and plotting of results
SparCC-R	SparCC protocol replicated by genus and species selection across 4 filtration criteria
SamQC-1	Run plink (Purcell et al., 2007) to extract samples from two pedfiles from different chips, render them into vcfs, and then create vcfs reflecting only the common loci
NestContest	Nested protocol to transfer bams from tier two storage to tier one, use GATK to call variants on common loci from SamQC-1 data, transfer those vcfs from tier two storage and run ConTest (Cibulskis et al., 2011), removing transferred files upon completion
SamQC-2	Run cross-correlations of sequence vcfs from the NestContest workflow with corresponding chip vcfs from SamQC-1