Anduril 2: Upgraded large-scale data integration framework

Cervera, A. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Rantanen, V. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Ovaska, K. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Laakso, M. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Nuñez-Fontarnau, J. Institute for Molecular Medicine Finland, University of Helsinki, Helsinki 00014, Finland
Alkodsi, A. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Casado, J. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Facciotto, C. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Häkkinen, A. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Louhimo, R. Finnish Institute of Occupational Health, Helsinki 00032, Finland
Karinen, S. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Zhang, K. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Lavikka, K. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Lyly, L. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
Pal Singh, M. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
*Hautaniemi, S. Research Programs Unit, Genome-Scale Biology, Faculty of Medicine, University of Helsinki, Helsinki 00014, Finland
*To whom correspondence should be addressed

Abstract

Motivation: Anduril is an analysis and integration framework for biomedical and clinical data . We have updated Anduril to use Scala for pipeline construction instead of a custom scripting language. The shift to Scala has simplified maintenance work for Anduril developers, and allows the user much more freedom in their programs and facilitates design of complex pipelines. We have also extensively extended the component repository and expanded the documentation with example pipelines and code snippets.

Results: Anduril facilitates the parallelization and guarantees the reproducibility of bioinformatic pipelines. Anduril strength over other frameworks is the combination of extensive bionformatic resources, a component repository, which is constantly expanding, and being a robust framework for reproducible research that allows complex logic in its pipelines thanks to Scala.

Anduril is a tried and tested computational framework for large-scale biomedical data analysis and integration. Here, we report major upgrades of Anduril including the use of Scala for pipeline construction and the addition of multiple components for next-generation sequencing data analysis.
Availability: Freely available on the web at http://anduril.org
Contact: sampsa.hautaniemi@helsinki.fi

Introduction

Measurement technologies, such as next-generation sequencing, proteomics and automated imaging, are able to produce enormous amounts of data, which have transformed medical research into a data-rich field. While producing data from biological samples is cost efficient and easy, data analysis and interpretation has become a bottleneck. Computational frameworks that allow systematic, parallel and flexible pipeline design are indispensable for the reproducibility, maintenance and execution of large-scale analyses.

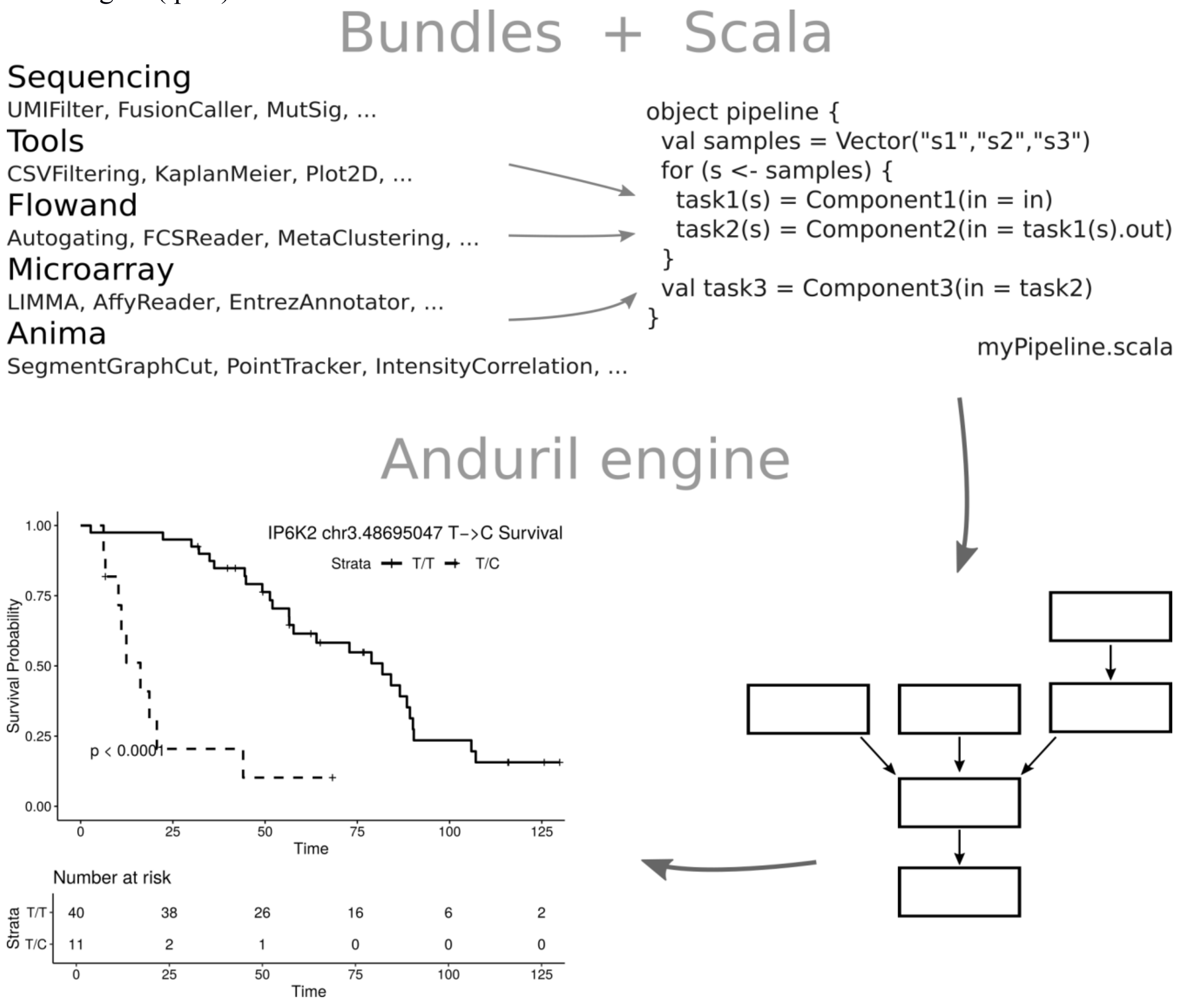
The design of current frameworks is heavily influenced by the expected end-user. Some frameworks like Galaxy (Goecks *et al.*, 2010)□, Taverna (Wolstencroft *et al.*, 2013)□, and GenePattern (Reich *et al.*, 2006)□, offer easy-to-run capabilities of existing pipelines with a graphical user interface (GUI), while frameworks like Anduril (Ovaska *et al.*, 2010)□, Snakemake (Koster and Rahmann, 2012)□, Ruffus (Goodstadt, 2010)□, and Nextflow (Di Tommaso *et al.*, 2017)□, offer more flexibility in pipeline construction and integration of tools for users with at least some level of programming skills. Currently no single framework caters to all users and the differing demands of all data analysis projects (Leipzig, 2017)□. Here we present an updated version of the Anduril data analysis and integration framework, designed for bioinformaticians and ideal for laboratories working with few in-house samples or considerably larger datasets, for example from The Cancer Genome Atlas (TCGA), which may require integration of several layers, such as clinical data and outputs of various high-throughput technologies.

The two major improvements in Anduril 2 are 1) the change from a custom-made scripting language to Scala () which grants more freedom and flexibility in pipeline construction, and 2) the expansion of Anduril's bioinformatics resource bundles. These resources confer built-in support to pipelines for analysis of central technologies in biomedicine, such as high-throughput imaging (Rantanen *et al.*, 2014)□, exome or whole genome, and micro- and mRNA data analys (Icay *et al.*, 2016)□. Other recent additions, to the Anduril framework, include both components for specific analysis such as gene fusions detection, clonal population inference in tumors, and methylation extraction and decomposition based on tumor purity (Häkkinen *et al.*, 2018)□ as well as components that facilitate general data analysis through a quick interface to R library dplyr (Wickham *et al.*, 2018)□ or Python Data Analysis library (pandas) (Mckinney, 2011)□. Anduril 2 comes with extensive documentation, which allows not only to learn how to get started, understand how to build pipelines and make use of parallelization, but also how to best exploit the available components, and learn how to start processing and analyzing high-throughput data sets. Examples on how to perform quality control, variant calling following the GATK gold standard, methylation analysis, single cell RNA-seq, or simple data transformation and/or annotation are available in Anduril's Bitbucket repository. Anduril 2 is distributed and licensed as open source software and is freely extendable. An overview of the framework is depicted in Figure 1.

Software description

A good framework for developing bioinformatic pipelines should handle both serial and parallel steps, complex dependencies, varied software and data file types, user-defined parameters and deliverables (Leipzig, 2017). Most popular bioinformatics frameworks, including Anduril 2, comply already with these aspects, below we describe other features and advantages of Anduril 2.

Automatic parallelization: Anduril models the component dependencies as a graph and parallelizes independent parts. Concurrency is limited by user-specified thread count and available resources. The generalized prefixing of the processes enables flexible use of SLURM (Jette *et al.*, 2002) and Sun Grid Engine (qsub).


Bundles + Scala
Sequencing
UMIFilter, FusionCaller, MutSig, ...
Tools
CSVFiltering, KaplanMeier, Plot2D, ...
Flowand
Autogating, FCSReader, MetaClustering, ...
Microarray
LIMMA, AffyReader, EntrezAnnotator, ...
Anima
SegmentGraphCut, PointTracker, IntensityCorrelation, ...
myPipeline.scala
Anduril engine
IP6K2 chr3.48695047 T->C Survival
Strata T/T T/C
Survival Probability
p < 0.0001
Time
Number at risk
Strata
T/T 40 38 26 16 6 2
T/C 11 2 1 0 0 0

```
object pipeline {
  val samples = Vector("s1","s2","s3")
  for (s <- samples) {
    task1(s) = Component1(in = in)
    task2(s) = Component2(in = task1(s).out)
  }
  val task3 = Component3(in = task2)
}
```

Figure 1: Many common pre-processing and downstream analysis steps have been encapsulated in components coded in a variety of supported languages (R, Matlab, Python, Java). Components are organized in bundles dedicated to specific datasets (anima - image processing, flowand - flow cytometry, sequencing - omics data, tools - general, microarray) and are combined into pipelines in a Scala program run by the Anduril engine. Anduril constructs a graph and handles execution of the pipeline tasks in parallel while keeping track of the changes and status (pass/fail) of different steps to ensure reentrancy. The Kaplan-Meier curve shown here is a direct output of the case study and shows

the survival difference in the TCGA ovarian patients with a T/C genotype in IP6K2 gene (chr3:48695047, p-value 0.0001).
Reentrancy: Resuming execution at the point of interruption is extremely useful when executing long-running complex pipelines on big datasets, as it spares the user from having to identify from which point onward to re-execute or which samples have been already analyzed. It is possible to update the component or their parameters and to add samples into the workflow without triggering execution of the completed independent steps. This results in significant improvements on both computing and programming time.

Dependency support: An update, such as a change in parameter, on a given step will cause re-execution of all dependent downstream processes. Components can be annotated to create synthetic dependencies between them when their input-outputs are not linked. For example, a component may not produce an output but can modify its environment, such as a database entry, and trigger downstream execution of a component marked as its dependent.

Bioinformatics resources: More than three hundred components for performing common tasks or running tools and methods for diverse bioinformatics analyses are available and fully documented. Components are organized in bundles for different types of analysis such as processing microarray, sequencing, or image data (see Figure 1). Installers for most third-party software supported by Anduril components are included, which simplifies the task of installing the myriad of software packages used in standard bioinformatic analysis, without adding the overhead to Anduril download and installation time. Anduril 2 can use its own optional installation or a user-defined one. Furthermore, any component can be run outside Anduril 2 with the same parameters and inputs derived from the pipeline since the effective configuration of each component is stored in a bash script facilitating testing and providing reproducibility.

Ease of integration of new tools or custom analysis: Integrating additional tools into a pipeline is extremely simple since own or third-party code can be embedded in eval-based components inside Anduril pipelines. Adding a new tool to the repository of components, for private use or for sharing with the community, requires only defining inputs, parameters and outputs through an XML file, ideally with appropriate test cases. Tools like Taverna require third-party software to implement plug-ins to be used in the pipelines. Both Galaxy and Anduril2 offer an easy way to build wrappers, but Anduril also supports immediate integration of custom analysis and software in any pipeline. For example, the following code shows how to use Kallisto (Bray *et al.*, 2016) in a pipeline without having a component for it. In this example the index is created separately so the quantification step can be run inside a for-loop with several samples without unnecessarily recomputing the index.

```
val index = QuickBash(in=reference,
                      script="kallisto index -i $out $in")

val quant = BashEvaluate(var1=index.out,
                         var2=reads,
                         script="kallisto quant -i @var1@ -o @folder1@ -b 100 --
single @var2@")
```

Anduril 2 is a tool intended for bioinformaticians who require a flexible and robust framework that allows reproducibility, parallelization, inclusion of custom analyses, and ease of re-execution.

Results and Conclusions

Case Study

To illustrate Anduril 2 in data analysis, we studied RNA-seq data from the good and poor responding high-grade serous ovarian cancer (HGSOC) patients. HGSOC is the most common and aggressive form of epithelial ovarian cancer with a 5-year survival rate of only 43% (Torre *et al.*, 2018)□. We hypothesized that comparing HGSOC patients who belong to the group of 10% of the longest response (n=26) to the patients who belong to the group of 10% of the shortest response (n=24) would reveal genes and genetic variants that are associated with treatment resistance and disease progression.

After downloading level 1 data from The Cancer Genome Atlas (TCGA) repository (Bell *et al.*, 2011)□, we performed quality screening and trimming of sequences, alignment, variant calling, and downstream analysis such as survival analysis, differential expression and pathway enrichment. Figures and reports produced for this case-study, as well as the pipelines, are available in http://csbi.ltdk.helsinki.fi/p/anduril2.

An interesting finding emerged by combining the variant calling analysis of RNA-seq data with survival analysis. The polymorphism (T->C in chr3 48695047) in *IP6K2* (Inositol Hexakisphosphate Kinase 2, a pro-apoptotic kinase) showed the most significant association to survival ($p<0.0001$) as shown in Figure 1. We confirmed that the mRNA-detected variant could be observed in whole genome sequencing as well, for which we used six available matching normal and tumor samples from both poor and good prognosis patients. The variant was found in the genome of one additional poor prognosis patient, which resulted in 11 samples with the TC genotype. *IP6K2* activity has been linked to therapy response in ovarian cancer (Morrison *et al.*, 2002)□, but the mechanism on how IP6K2 mediates apoptosis is still unclear (Nagata *et al.*, 2005)□.

Conclusions

With many frameworks for data analysis available, the choice of which to adopt needs to take into consideration the skills and backgrounds of the user, as well as the needs of the projects. Compared with Galaxy and Taverna, Anduril 2 offers ease of integration of new tools and custom analysis as well as batch processing. Frameworks like Luigi (https://github.com/spotify/luigi) and Snakemake handle efficient execution of pipelines, but do not provide any bioinformatic-related components, of which Anduril has more than 380, with installers when possible, that allows to construct complex pipelines easily. Tools such as WDL (https://software.broadinstitute.org/wdl/) are equipped with a scattering control flow that can mimic Anduril 2 dynamic for-loops although nested scattering is not supported; dependencies are containerized via Docker images and there is no parametric data typing. Furthermore, WDL does not include bundles specialized in bioinformatics.

Documentation is a key aspect of any software development project and it is crucial for successful community adoption and software maintenance (Karimzadeh and Hoffman, 2017)□. Anduril 2 can be downloaded from Bitbucket (https://bitbucket.org/anduril-dev/anduril) where links to a quick start guide, the reference manual, several examples on how to conduct basic sequence data analysis, snippets with tips and tricks and a public forum for addressing questions and issues for both users and developers are provided. Anduril 2 is also available as a docker image as standalone or with source code for running parts of the case study presented here.

Acknowledgements

We thank CSC – IT Center for Science Ltd. for computing resources as well as the current and past Hautaniemi lab members for their contributions to components and documentation. The results published here are in part based upon data generated by TCGA managed by the NCI and NHGRI. Information about TCGA can be found at http://cancergenome.nih.gov.

Funding

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 667403 for HERCULES, the Academy of Finland (project 305087), the Sigrid Jusélius Foundation, and Finnish Cancer Associations.

References

Bell,D. *et al.* (2011) Integrated genomic analyses of ovarian carcinoma. *Nature*, **474**, 609–615.

Bray,N.L. *et al.* (2016) Near-optimal probabilistic RNA-seq quantification. *Nat. Biotechnol.*, **34**, 525–527.

Goecks,J. *et al.* (2010) Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.*, **11**, R86.

Goodstadt,L. (2010) Ruffus: a lightweight Python library for computational pipelines. *Bioinformatics*, **26**, 2778–2779.

Häkkinen,A. *et al.* (2018) Identifying differentially methylated sites in samples with varying tumor purity. *Bioinformatics*, **34**, 3078–3085.

Icay,K. *et al.* (2016) SePIA: RNA and small RNA sequence processing, integration, and analysis. *BioData Min.*, **9**, 20.

Jette,M.A. *et al.* (2002) SLURM: Simple Linux Utility for Resource Management. *Lect. NOTES Comput. Sci. Proc. JOB Sched. Strateg. PARALLEL Process. 2003*, **2862**, 44--60.

Karimzadeh,M. and Hoffman,M.M. (2017) Top considerations for creating bioinformatics software documentation. *Brief. Bioinform.*, bbw134.

Koster,J. and Rahmann,S. (2012) Snakemake--a scalable bioinformatics workflow engine. *Bioinformatics*, **28**, 2520–2522.

Leipzig,J. (2017) A review of bioinformatic pipeline frameworks. *Brief. Bioinform.*, **18**, 530–536.

Mckinney,W. (2011) pandas: a Foundational Python Library for Data Analysis and Statistics.

Morrison,B.H. *et al.* (2002) Inositol hexakisphosphate kinase 2 sensitizes ovarian carcinoma cells to multiple cancer therapeutics. *Oncogene*, **21**, 1882–1889.

Nagata,E. *et al.* (2005) Inositol hexakisphosphate kinase-2, a physiologic mediator of cell death. *J. Biol. Chem.*, **280**, 1634–40.

Ovaska,K. *et al.* (2010) Large-scale data integration framework provides a comprehensive view on glioblastoma multiforme. *Genome Med.*, **2**, 65.

Rantanen,V. *et al.* (2014) Anima: modular workflow system for comprehensive image data analysis. *Front. Bioeng. Biotechnol.*, **2**, 25.

Reich,M. *et al.* (2006) GenePattern 2.0. *Nat. Genet.*, **38**, 500–501.

Di Tommaso,P. *et al.* (2017) Nextflow enables reproducible computational workflows. *Nat. Biotechnol.*, **35**, 316–319.

Torre,L.A. *et al.* (2018) Ovarian cancer statistics, 2018. *CA. Cancer J. Clin.*, **68**, 284–296.

Wickham,H. *et al.* (2018) A Grammar of Data Manipulation.

Wolstencroft,K. *et al.* (2013) The Taverna workflow suite: designing and executing workflows of Web Services on the desktop, web or in the cloud. *Nucleic Acids Res.*, **41**, W557-61.