open sharing of protocols. With a precise ontology to describe standardized protocols, it may be possible to share methods widely and create community standards.

We envisage that in future individual research laboratories, or clusters of co-located laboratories, will have in-house, low-cost automation work cells but will access DNA foundries via the cloud to carry out complex experimental workflows. Technologies enabling this from companies such as Emerald Cloud Lab (S. San Francisco, CA, USA), Synthace (London) and Transcriptic (Menlo Park, CA, USA) could, for example, send experimental designs to foundries and return output data to a researcher. This 'mixed economy' should accelerate the development and sharing of standardized protocols and metrology standards and shift a growing proportion of molecular, cellular and synthetic biology into a fully quantitative and reproducible era.

COMPETING FINANCIAL INTERESTS
The authors declare no competing financial interests.

David W McClymont[1] & *Paul S Freemont*[1,2]

[1]*The London DNA Foundry, UK Synthetic Biology Innovation, Commercialisation and Industrial Translation Centre, London, UK.* [2]*Centre for Synthetic Biology and Innovation, Department of Medicine, South Kensington Campus, Imperial College London, London, UK.* *e-mail: d.mcclymont@imperial.ac.uk or p.freemont@imperial.ac.uk.*

1. Baker, M. *Nature* **533**, 452–454 (2016).
2. Yachie, N. *et al. Nat. Biotechnol.* **35**, 310–312 (2017).
3. Hadimioglu, B., Stearns, R. & Ellson, R. *J. Lab. Autom.* **21**, 4–18 (2016).
4. ANSI SLAS 1–2004: Footprint dimensions; ANSI SLAS 2–2004: Height dimensions; ANSI SLAS 3–2004: Bottom outside flange dimensions; ANSI SLAS 4–2004: Well positions; (ANSI SLAS, 2004).
5. Mckernan, K. & Gustafson, E. in *DNA Sequencing II: Optimizing Preparation and Cleanup* (ed. Kieleczawa, J.) 9.128 (Jones and Bartlett Publishers, 2006).
6. Storch, M. *et al.* BASIC: a new biopart assembly standard for idempotent cloning provides accurate, single-tier DNA assembly for synthetic biology. *ACS Synth. Biol.* **4**, 781–787 (2015).

Toil enables reproducible, open source, big biomedical data analyses

To the Editor:

Contemporary genomic data sets contain tens of thousands of samples and petabytes of sequencing data[1–3]. Pipelines to process genomic data sets often comprise dozens of individual steps, each with their own set of parameters[4,5]. Processing data at this scale and complexity is expensive, can take an unacceptably long time, and requires significant engineering effort. Furthermore, biomedical data sets are often siloed, both for organizational and security considerations and because they are physically difficult to transfer between systems, owing to bandwidth limitations. The solution to better handling these big data problems is twofold: first, we need robust software capable of running analyses quickly and efficiently, and second, we need the software and pipelines to be portable, so that they can be reproduced in any suitable compute environment.

Here, we present Toil, a portable, open-source workflow software that can be used to run scientific workflows on a large scale in cloud or high-performance computing (HPC) environments. Toil was created to include a complete set of features necessary for rapid large-scale analyses across multiple environments. While several other scientific workflow software packages[6–8] offer some subset of fault tolerance, cloud support and HPC support, none offers these with the scale and efficiency to process petabyte and larger-scale data sets efficiently. This sets Toil apart in its capacity to produce results faster and for less cost across diverse environments.

We demonstrate Toil by processing >20,000 RNA-seq samples (**Fig. 1**). The resulting meta-analysis of five data sets is available to readers[9]. The large majority (99%) of these samples were analyzed in under 4 days using a commercial cloud cluster of 32,000 preemptable cores.

To support the sharing of scientific workflows, we designed Toil to execute common workflow language (CWL; **Supplementary Note 1**) and provide draft support for workflow description language (WDL). Both CWL and WDL are standards for scientific workflows[10,11]. A workflow comprises a set of tasks, or 'jobs', that are orchestrated by specification of a set of dependencies that map the inputs and outputs between jobs. In addition to CWL and draft WDL support, Toil provides a Python application program interface (API) that allows workflows to be declared statically, or generated dynamically, so that jobs can define further jobs during execution and therefore as needed (**Supplementary Note 2** and **Supplementary Toil Documentation**). The jobs defined in either CWL or Python can consist of Docker containers, which permit sharing of a program without requiring individual tool installation or configuration within a specific environment. Open-source workflows that use containers can be run regardless of environment. We provide a repository of genomic workflows as examples[12]. Toil supports services, such as databases or servers, that are defined and

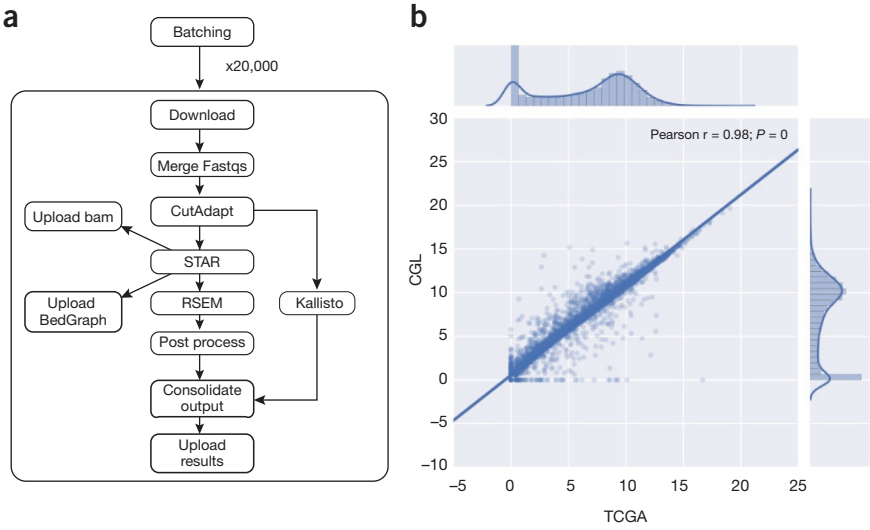

Figure 1 RNA-seq pipeline and expression concordance. (**a**) A dependency graph of the RNA-seq pipeline we developed (named CGL). CutAdapt was used to remove extraneous adapters, STAR was used for alignment and read coverage, and RSEM and Kallisto were used to produce quantification data. (**b**) Scatter plot showing the Pearson correlation between the results of the TCGA best-practices pipeline and the CGL pipeline. 10,000 randomly selected sample and/or gene pairs were subset from the entire TCGA cohort and the normalized counts were plot against each other; this process was repeated five times with no change in Pearson correlation. The unit for counts is: $\log_2$(norm_counts+1).

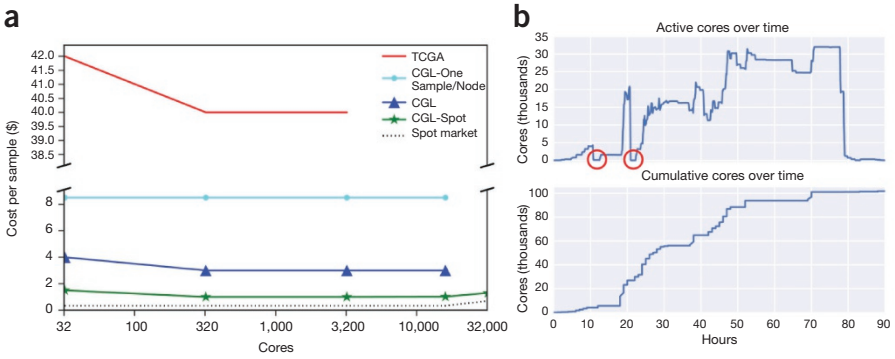

Figure 2 Costs and core usage. (**a**) Scaling tests were run to ascertain the price per sample at varying cluster sizes for the different analysis methods. TCGA (red) shows the cost of running the TCGA best-practices pipeline as re-implemented as a Toil workflow (for comparison). CGL-One-Sample/Node (cyan) shows the cost of running the revised Toil pipeline, one sample per node. CGL (blue) denotes the pipeline running samples across many nodes. CGL-Spot (green) is the same as CGL, but denotes the pipeline run on the Amazon spot market. The slight rise in cost per sample at 32,000 cores was due to a couple of factors: aggressive instance provisioning directly affected the spot price (dotted line), and saving *bam* and *bedGraph* files for each sample. (**b**) Tracking number of cores during the recompute. The two red circles indicate where all worker nodes were terminated and subsequently restarted shortly thereafter.

managed within a workflow. Through this mechanism it integrates with Apache Spark[13] (**Supplementary Fig. 4**), and can be used to rapidly create containerized Spark clusters[14] (**Supplementary Note 3**).

Toil runs in multiple cloud environments including those of Amazon Web Services (AWS; Seattle, WA, USA), Microsoft Azure (Seattle, WA, USA), Google Cloud (Mountain View, CA, USA), OpenStack, and in HPC environments running GridEngine or Slurm and distributed systems running Apache Mesos[15–17] (Forest Hill, MD, USA). Toil can run on a single machine, such as a laptop or workstation, to allow for interactive development, and can be installed with a single command. This portability stems from pluggable backend APIs for machine provisioning, job scheduling and file management (**Supplementary Note 4**). Implementation of these APIs facilitates straightforward extension of Toil to new compute environments. Toil manages intermediate files and checkpointing through a 'job store', which can be an object store like AWS's S3 or a network file-system. The flexibility of the backend APIs allow a single script to be run on any supported compute environment, paired with any job store, without requiring any modifications to the source code.

Toil includes numerous performance optimizations to maximize time and cost efficiencies (**Supplementary Note 5**). Toil implements a leader/worker pattern for job scheduling, in which the leader delegates jobs to workers. To reduce pressure on the leader, workers can decide whether they are capable of running jobs immediately downstream to their assigned task (in terms of resource requirements and workflow dependencies). Frequently, next-generation sequencing workflows are I/O bound, owing to the large volume of data analyzed. To mitigate this, Toil uses file caching and data streaming. Where possible, successive jobs that share files are scheduled on a single node, and caching prevents the need for repeated transfers from the job store. Toil is robust to job failure because workflows can be resumed after any combination of leader and worker failures. This robustness enables workflows to use low-cost machines that can be terminated by the provider at short notice and are currently available at a significant discount on AWS and Google Cloud. We estimate the use of such preemptable machines on AWS lowered the cost of our RNA-seq compute job 2.5-fold, despite encountering over 2,000 premature terminations (**Fig. 2**). Toil also supports fine-grained resource requirements, enabling each job to specify its core, memory and local storage needs for scheduling efficiency.

Controlled-access data requires appropriate precautions to ensure data privacy and protection. Cloud environments offer measures that ensure stringent standards for protected data. Input files can be securely stored on object stores, using encryption, either transparently or with customer managed keys. Compute nodes can be protected by SSH key pairs. When running Toil, all intermediate data transferred to and from the job store can be optionally encrypted during network transmission and on the compute nodes' drives using Toil's cloud-based job store encryption. These and other security measures help ensure protection of the input data, and as part of a broader security plan, can be used to ensure compliance with strict data security requirements.

To demonstrate Toil, we used a single script to compute gene- and isoform-level expression values for 19,952 samples from four studies: The Cancer Genome Atlas (TCGA)[1], Therapeutically Applicable Research To Generate Effective Treatments (TARGET; https://ocg.cancer.gov/programs/target), Pacific Pediatric Neuro-Oncology Consortium (PNOC; http://www.pnoc.us/), and the Genotype Tissue Expression Project (GTEx)[18]. The data set comprised 108 terabytes. The Toil pipeline uses STAR[19] to generate alignments and read coverage graphs, and performs quantification using RSEM[20] and Kallisto[21] (**Fig. 1** and **Supplementary Note 6**). Processing the samples in a single batch on ~32,000 cores on AWS took 90 h of wall time, 368,000 jobs and 1,325,936 core hours. The cost per sample was $1.30, which is an estimated 30-fold reduction in cost, and a similar reduction in time, compared with the TCGA best-practices workflow[5]. We achieved a 98% gene-level concordance with the previous pipeline's expression predictions (**Figs. 1**,**2** and **Supplementary Fig. 1**). Notably, we estimate that the pipeline, without STAR and RSEM, could be used to generate quantifications for $0.19/sample with Kallisto. To illustrate portability, the same pipeline was run on the I-SPY2 data set[22] (156 samples) using a private HPC cluster, achieving similar per sample performance (**Supplementary Table 1**). Expression-level signal graphs (read coverage) of the GTEx data (7,304 samples from 53 tissues, 570 donors) are available from a UCSC Genome Browser[23] public track hub (**Supplementary Fig. 2**). Gene and isoform quantifications for this consistent, union data set are publicly hosted on UCSC Xena[9] and are available for direct access through a public AWS bucket (**Supplementary Fig. 3** and **Supplementary Note 7**).

Although there is an extensive history of open-source workflow-execution software[6–8], the shift to cloud platforms and the advent of standard workflow languages is changing the scale of analyses. Toil is a portable workflow software that supports open community standards for workflow specification and enables researchers to move their computation according to cost, time and data location. For example, in our analysis the sample data were intentionally co-located in the same region as the compute servers in order to provide optimal bandwidth when scaling to thousands of simultaneous jobs (**Supplementary Note 8**). This type of flexibility enables larger, more

comprehensive analyses. Further, it means that results can be reproduced using the original computation's set of tools and parameters. If we had run the original TCGA best-practices RNA-seq pipeline with one sample per node, it would have cost ~$800,000. Through the use of efficient algorithms (STAR and Kallisto) and Toil, we were able to reduce the final cost to $26,071 (**Supplementary Note 9**).

We have demonstrated the utility of Toil by creating one of the single largest, consistently analyzed, public human RNA-seq expression repositories, which we hope the community will find useful.

Editor's note: This article has been peer-reviewed.

Note: Any Supplementary Information and Source Data files are available in the online version of the paper.

ACKNOWLEDGMENTS

This work was supported by (BD2K) the National Human Genome Research Institute of the National Institutes of Health award no. 5U54HG007990 and (Cloud Pilot) the National Cancer Institute of the National Institutes of Health under the Broad Institute subaward no. 5417071-5500000716. The UCSC Genome Browser work was supported by the NHGRI award 5U41HG002371 (Corporate Sponsors). The content is solely the responsibility of the authors and does not necessarily represent the official views of the National Institutes of Health or our corporate sponsors.

AUTHOR CONTRIBUTIONS

J.V., A.A.R. and B.P wrote the manuscript. J.V., A.A.R., A.N., J.A., C.K., J.N., H.S., P.A., J.P., A.D.D., B.O. and B.P. contributed to Toil development. F.A.N. and A.M. contributed to Toil-Spark integration. J.V. wrote the RNA-seq pipeline and automation software. M.H. and C.B. contributed WDL and cloud support. P.A. and S.Z. contributed CWL support. J.Z., B.C. and M.G. hosted quantification results on UCSC Xena. K.R. hosted GTEx results in UCSC Genome Browser. W.J.K., J.Z., S.Z., G.G., D.A.P., A.D.J., M.C., D.H. and B.P. provided scientific leadership and project oversight.

Data availability. Data are available from this project at the Toil xena hub (https://genome-cancer.soe.ucsc.edu/proj/site/xena/datapages/?host=https://toil.xenahubs.net).

COMPETING FINANCIAL INTERESTS

The authors declare competing financial interests: details are available in the online version of the paper.

John Vivian[1], Arjun Arkal Rao[1], Frank Austin Nothaft[2,3], Christopher Ketchum[1], Joel Armstrong[1], Adam Novak[1], Jacob Pfeil[1], Jake Narkizian[1], Alden D Deran[1], Audrey Musselman-Brown[1], Hannes Schmidt[1], Peter Amstutz[4], Brian Craft[1], Mary Goldman[1], Kate Rosenbloom[1], Melissa Cline[1], Brian O'Connor[1], Megan Hanna[5], Chet Birger[5], W James Kent[1], David A Patterson[2,3], Anthony D Joseph[2,3], Jingchun Zhu[1], Sasha Zaranek[4], Gad Getz[5], David Haussler[1] & Benedict Paten[1]

[1]*Computational Genomics Lab, UC Santa Cruz Genomics Institute, University of California Santa Cruz, Santa Cruz, California, USA.* [2]*AMP Lab, University of California Berkeley, Berkeley, California, USA.* [3]*UC Berkeley ASPIRE Lab, Berkeley, California, USA.* [4]*Curoverse, Somerville, Massachusetts, USA.* [5]*Broad Institute of Harvard and Massachusetts Institute of Technology (MIT), Cambridge, Massachusetts, USA.*
e-mail: benedict@soe.ucsc.edu

1. Weinstein, J.N. *et al. Nat. Genet.* **45**, 1113–1120 (2013).
2. Zhang, J. et al. Database. http://dx.doi.org/10.1093/database/bar026 (2011)
3. Siva, N. *Lancet* **385**, 103–104 (2015).
4. McKenna, A. *et al. Genome Res.* **20**, 1297–1303 (2010).
5. UNC Bioinformatics. TCGA mRNA-seq pipeline for UNC data. https://webshare.bioinf.unc.edu/public/mRNAseq_TCGA/UNC_mRNAseq_summary.pdf (2013).
6. Albrecht, M., Michael, A., Patrick, D., Peter, B. & Douglas, T. in *Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies* (SWEET '12) 1. ACM (Association of Computing Machinery. http://dx.doi.org/10.1145/2443416.2443417 (2012).
7. Bernhardsson, E. & Frieder, E. Luigi. *Github* https://github.com/spotify/luigi (2016).
8. Goecks, J., Nekrutenko, A. & Taylor, J. *Genome Biol.* **11**, R86 (2010).
9. UCSC. Xena http://xena.ucsc.edu (2016).
10. Amstutz, P. Common workflow language. *Github* https://github.com/common-workflow-language/common-workflow-language (2016).
11. Frazer, S. Workflow description language. *Github* https://github.com/broadinstitute/wdl (2014).
12. Vivian, J. Toil scripts. *Github* https://github.com/BD2KGenomics/toil-scripts/tree/master/src/toil_scripts (2016).
13. Apache Software Foundation. Apache Spark http://spark.apache.org/ (2017).
14. Massie, M. *et al.* ADAM: genomics formats and processing patterns for cloud scale computing. University of California, Berkeley, Technical Report No. UCB/EECS-2013-207 (2013).
15. Gentzsch, W. in *Proceedings First IEEE/ACM International Symposium on Cluster Computing and the Grid* 35–36 http://dx.doi.org/10.1109/ccgrid.2001.923173 (IEEE, 2001).
16. Yoo, A.B., Jette, M.A. & Mark, G. in *Lecture Notes in Computer Science* 44–60 (2003) Springer, Berlin, Heidelberg.
17. Apache Software Foundation. Apache Mesos http://mesos.apache.org/
18. GTEx Consortium. *Science* **348**, 648–660 (2015).
19. Dobin, A. *et al. Bioinformatics* **29**, 15–21 (2013).
20. Li, B. & Dewey, C.N. *BMC Bioinformatics* **12**, 323 (2011).
21. Bray, N.L., Pimentel, H., Melsted, P. & Pachter, L. *Nat. Biotechnol.* **34**, 525–527 (2016).
22. Barker, A.D. *et al. Clin. Pharmacol. Ther.* **86**, 97–100 (2009).
23. Kent, W.J. *et al. Genome Res.* **12**, 996–1006 (2002).

Nextflow enables reproducible computational workflows

To the Editor:

The increasing complexity of readouts for omics analyses goes hand-in-hand with concerns about the reproducibility of experiments that analyze 'big data'[1–3]. When analyzing very large data sets, the main source of computational irreproducibility arises from a lack of good practice pertaining to software and database usage[4–6]. Small variations across computational platforms also contribute to computational irreproducibility by producing numerical instability[7], which is especially relevant to high-performance computational (HPC) environments that are routinely used for omics analyses[8]. We present a solution to this instability named Nextflow, a workflow management system that uses Docker technology for the multi-scale handling of containerized computation.

In silico workflow management systems are an integral part of large-scale biological analyses. These systems enable the rapid prototyping and deployment of pipelines that combine complementary software packages. In genomics the simplest pipelines, such as Kallisto and Sleuth[9], combine an RNA-seq quantification method with a differential expression module (**Supplementary Fig. 1**). Complexity rapidly increases when all aspects of a given analysis are included. For example, the Sanger Companion pipeline[10] bundles 39 independent software tools and libraries into a genome annotation suite. Handling such a large number of software packages, some of which may be incompatible, is a challenge. The conflicting requirements of frequent software updates and maintaining the reproducibility of original results provide another unwelcome wrinkle. Together with these problems, high-throughput usage of complex pipelines can also be burdened by the hundreds of intermediate files often produced by individual tools. Hardware fluctuations in these types of pipelines, combined with poor error handling, could result in considerable readout instability.

Nextflow (http://nextflow.io; **Supplementary Methods**, **Supplementary Note** and **Supplementary Code 1**) is designed to address numerical instability, efficient parallel execution, error tolerance, execution provenance and traceability. It is a domain-specific language that enables rapid pipeline development through the adaptation of existing pipelines written in any scripting language.

We present a qualitative comparison between Nextflow and other similar tools in **Table 1** (ref. 11). We found that multi-scale containerization, which makes it possible to