Bioconda: A sustainable and comprehensive software distribution for the life sciences

Björn Grüning

Ryan Dale[[1]](#footnote-21),

Andreas Sjödin

Brad A. Chapman

Jillian Rowe

Christopher H. Tomkins-Tinch

Renan Valieris

Adam Caprez

Bérénice Batut

Mathias Haudgaard

Thomas Cokelaer

Kyle A. Beauchamp

Brent S Pedersen

Youri Hoogstrate

Anthony Bretaudeau

Devon Ryan

Gildas Le Corguillé

Dilmurat Yusuf

Sebastian Luna-Valero

Rory Kirchner

Karel Brinda

Thomas Wollmann

Martin Raden

Lorena Pantano

Nicola Soranzo

Simon J. van Heeringen

Zachary Charlop-Powers

Per Unneberg

Matthias De Smet

Marcel Martin

Greg Von Kuster

Tiago Antao

Milad Miladi

Christian Brueffer

Kevin Thornton

Marius van den Beek

Daniel Maticzka

Clemens Blank

Sebastian Will

Kévin Gravouil

Joachim Wolff

Manuel Holtgrewe

Jörg Fallmann

Vitor C. Piro

Ilya Shlyakhter

Ayman Yousif

Philip Mabon

Xiao-Ou Zhang

Jennifer Cabral

Wei Shen

Cristel Thomas

Joseph Brown

Eric Enns

Mattias de Hollander

Jorrit Boekel

Jerome Kelleher

Dave Bouvier

Nitesh Turaga

Simon Gladman

Julian R. de Ruiter

Saket Choudhary

Nicholas Harding

Florian Eggenhofer

Arne Kratz

Zhuoqing Fang

Robert Kleinkauf

Henning Timm

Aziz Khan

Peter J. A. Cock

Enrico Seiler

Hai Nguyen

Colin Brislawn

Endre Bakken Stovner

James E. Johnson

Emil Hägglund

Philip Ewels

Matt Chambers

Elmar Pruesse

Roman Valls Guimera

Simon Ye

W. Augustine Dunn

Lance Parsons

MacIntosh Cornwell

Alex Reynolds

Marcel Bargull

Elena Grassi

David Koppstein

Rob Patro

Inken Wohlers

Nicholas Stoler

Daniel Blankenberg

Guowei He

Mallory Freeberg

Rick Farouni

Alexander Junge

Daniel R. Bogema

Sourav Singh

Fabio Cumbo

David E Larson

Liang-Bo Wang

Matthew L. Workentine

Upendra Kumar Devisetty

Sacha Laurent

Xavier Garnier

Rasmus Agren

Pierrick Roger

John M Eppley

Wei Li

Tobias Rausch

Aroon T. Chande

James Taylor

Bianca Katharina Stöcker

Iain Milne

Patrick R. Wright

Adam P. Taranto

Davide Chicco

Nezar Abdennur

Bengt Sennblad

Jasmijn A. Baaijens

Matthew Gopez

Oana M. Enache

Luca Pinello

Avi Srivastava

Ino de Bruijn

Philip Reiner Kensche

Kai Blin

Yuri Pirola

Jens Preussner

Michael Knudsen

Matthew L. Bendall

Vivek Rai

Giorgio Gonnella

Christoph Stahl

Alistair Miles

Nicholas R. Waters

Saskia Hiltemann

Yannick Boursin

Yasset Perez-Riverol

Sebastian Schmeier

Erik Clarke

Kevin Arvai

Matthieu Jung

Gordon Stephen

Tomás Di Domenico

Julien Seiler

Eric Rasche

Etienne Kornobis

Daniela Beisser

Sven Rahmann

Alexander S Mikheyev

Camy Tran

Jordi Capellades

Oleksandr Moskalenko

Christopher Schröder

Simon Dirmeier

Timothy H. Webster

Adrian Emanuel Salatino

Gianvito Urgese

Johannes Köster[[2]](#footnote-22),

We present Bioconda (<https://bioconda.github.io>), a distribution of bioinformatics software for the lightweight, multi-platform and language-agnostic package manager Conda. Currently, Bioconda offers a collection of over 3000 software packages, which is continuously maintained, updated, and extended by a growing global community of more than 200 contributors. Bioconda improves analysis reproducibility by allowing users to define isolated environments with defined software versions, all of which are easily installed and managed without administrative privileges.

Thousands of new software tools have been released for bioinformatics in recent years, in a variety of programming languages. Accompanying this diversity of construction is an array of installation methods. Often, software has to be compiled manually for different hardware architectures and operating systems, with management left to the user or system administrator. Scripting languages usually deliver their own package management tools for installing, updating, and removing packages, though these are often limited in scope to packages written in the same scripting language and cannot handle external dependencies. Published scientific software often consists of simple collections of custom scripts distributed with textual descriptions of the manual steps required to install the software. New analyses often require novel combinations of multiple tools, and the heterogeneity of scientific software makes management of a software stack complicated and error-prone while consuming inordinate amounts of time. Moreover, it inhibits reproducible science  (Mesirov 2010; Baker 2016; Munafò et al. 2017), because it is hard to reproduce a software stack on different machines. System-wide deployment of software has traditionally been handled by administrators, but reproducibility requires that the researcher (who is often not an expert in administration) is able to maintain full control of the software environment and rapidly modify it without administrative privileges.

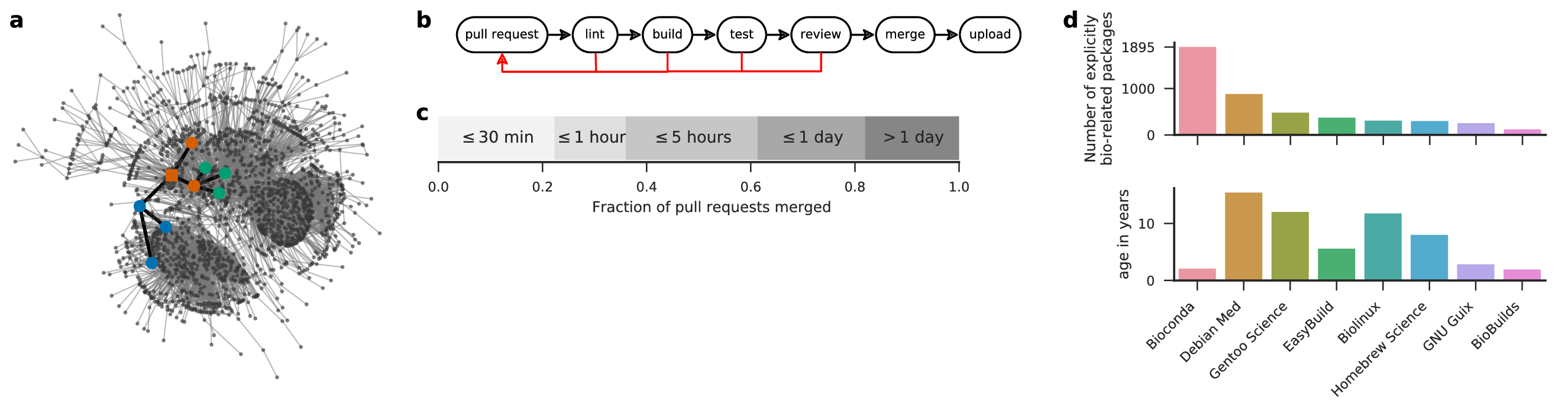
The Conda package manager (<https://conda.io>) has become an increasingly popular approach to overcome these challenges. Conda normalizes software installations across language ecosystems by describing each software package with a *recipe* that defines meta-information and dependencies, as well as a *build script* that performs the steps necessary to build and install the software. Conda prepares and builds software packages within an isolated environment, transforming them into relocatable binaries. Conda packages can be built for all three major operating systems: Linux, macOS, and Windows. Importantly, installation and management of packages requires no administrative privileges, such that a researcher can control the available software tools regardless of the underlying infrastructure. Moreover, Conda obviates reliance on system-wide installation by allowing users to generate isolated software environments, within which versions and tools can be managed per-project, without generating conflicts or incompatibilities (see online methods). These environments support reproducibility, as they can can be rapidly exchanged via files that describe their installation state. Conda is tightly integrated into popular solutions for reproducible scientific data analysis like Galaxy (Afgan et al. 2016), bcbio-nextgen (<https://github.com/chapmanb/bcbio-nextgen>), and Snakemake (Köster and Rahmann 2012). Finally, while Conda provides many commonly-used packages by default, it also allows users to optionally include additional repositories (termed *channels)* of packages that can be installed.

In order to unlock the benefits of Conda for the life sciences, the Bioconda project was founded in 2015. The mission of Bioconda is to make bioinformatics software easily installable and manageable via the Conda package manager. Via its Conda channel, Bioconda currently provides over 3000 software packages for Linux and macOS. Development is driven by an open community of over 200 international scientists. In the prior two years, package count and the number of contributors have increased linearly, on average, with no sign of saturation (Fig. ???a,b). The barrier to entry is low, requiring a willingness to participate and adherence to community guidelines. Many software developers contribute recipes for their own tools, and many Bioconda contributors are invested in the project as they are also users of Conda and Bioconda. Bioconda provides packages from various language ecosystems like Python, R (CRAN and Bioconductor), Perl, Haskell, as well as a plethora of C/C++ programs (Fig. ???c). Many of these packages have complex dependency structures that require various manual steps to install when not relying on a package manager like Conda (Fig. ???a, Online Methods). With over 6.3 million downloads, the service has become a backbone of bioinformatics infrastructure (Fig. ???d). Bioconda is complemented by the conda-forge project (<https://conda-forge.github.io>), which hosts software not specifically related to the biological sciences. The two projects collaborate closely, and the Bioconda team maintains over 500 packages hosted by conda-forge. Among all currently available distributions of bioinformatics software, Bioconda is by far the most comprehensive, while being among the youngest (Fig. ???d).

Bioconda development and usage since the beginning of the project. (a) contributing authors and added recipes over time. (b) code line additions and deletions per week. (c) package count per language ecosystem (saturated colors on bottom represent explicitly life science related packages). (d) total downloads per language ecosystem. The term other entails all recipes that do not fall into one of the specific categories. Note that a subset of packages that started in Bioconda have since been migrated to the more appropriate, general-purpose conda-forge channel. Older versions of such packages still reside in the Bioconda channel, and as such are included in the recipe count (a) and download count (d). Statistics obtained Oct. 25, 2017. 

Bioconda development and usage since the beginning of the project. (a) contributing authors and added recipes over time. (b) code line additions and deletions per week. (c) package count per language ecosystem (saturated colors on bottom represent explicitly life science related packages). (d) total downloads per language ecosystem. The term “other” entails all recipes that do not fall into one of the specific categories. Note that a subset of packages that started in Bioconda have since been migrated to the more appropriate, general-purpose conda-forge channel. Older versions of such packages still reside in the Bioconda channel, and as such are included in the recipe count (a) and download count (d). Statistics obtained Oct. 25, 2017.

To ensure reliable maintenance of such numbers of packages, we use a semi-automatic, agent-assisted development workflow (Fig. ???b), orchestrated by a suite of tools we authored and maintain (<https://github.com/bioconda/bioconda-utils>). All Bioconda recipes are hosted in a GitHub repository (<https://github.com/bioconda/bioconda-recipes>). Both the addition of new recipes and the update of existing recipes in Bioconda is handled via *pull requests*. A contributor opens a pull request on the GitHub repository with a modified version of one or more recipes, and these changes are automatically compared against the current state of Bioconda. Once a pull request arrives, our infrastructure performs several automatic checks. Problems discovered in any step are reported to the contributor and further progress is blocked until they are resolved. First, the modified recipes are checked for syntactic anti-patterns, i.e., formulations that are syntactically correct but bad style (termed *linting*). This process ensures consistency across all submitted recipes, serving as an initial quality-control step and easing the automated maintenance of recipes. Second, the modified recipes are built on Linux and macOS, via a cloud based, free-of-charge service (<https://travis-ci.org>). Successfully built recipes are tested (e.g., by running the generated executable). Since Bioconda packages must be able to run on any supported system, it is important to check that the built packages do not rely on particular elements from the build environment. Therefore, testing happens in two stages: (a) test cases are executed in the full build environment (b) test cases are executed in a minimal Docker (<https://docker.com>) container which purposefully lacks all non-common system libraries. Hence, a dependency that is not explicitly defined will lead to a failure in the latter, more stringent test. Once the *build* and *test* steps have succeeded, a member of the Bioconda team reviews the proposed changes and, if acceptable, merges the modifications into the official repository. Upon merging, packages are uploaded to the hosted Bioconda channel (<https://anaconda.org/bioconda>), where they become available via the Conda package manager. When a Bioconda package is updated to a new version, older builds are generally preserved, and recipes for multiple older versions may be maintained in the Bioconda repository. The usual turnaround time of above workflow is short (Fig. ???d): 61% of the pull requests are merged within 5 hours. Of those, 36% are even merged within 1 hour. Only 18% of the pull requests need more than a day. Hence, publishing software in Bioconda or updating already existing packages can be accomplished typically within minutes to a few hours.



Dependency structure, workflow, comparison with other resources, and turnaround time. (a) largest connected component of directed acyclic graph of Bioconda packages (nodes) and dependencies (edges). Highlighted is the induced subgraph of the CNVkit (Talevich et al. 2016) package and its dependencies spanning Python, C/C++, and R ecosystems (node coloring as defined in Fig. ???c, squared node represents CNVkit). (b) GitHub based development workflow: a contributor provides a pull request that undergoes several build and test steps, followed by a human review. If any of these checks does not succeed, the contributor can update the pull request accordingly. Once all steps have passed, the changes can be merged. (c) Turnaround time from submission to merge of pull requests in Bioconda. (d) Comparison of explicitly life science related packages in Bioconda with Debian Med ([https://www.debian.org/devel/debian-med](https://www.debian.org/devel/debian-med/)), Gentoo Science Overlay (category sci-biology,  <https://github.com/gentoo/sci>), EasyBuild (module bio, [https://easybuilders.github.io/easybuild](https://easybuilders.github.io/easybuild/)), Biolinux (Field et al. 2006), Homebrew Science (tag bioinformatics, [https://brew.sh](https://brew.sh/)), GNU Guix (category bioinformatics, [https://www.gnu.org/s/guix)](https://www.gnu.org/s/guix/), and BioBuilds (<https://biobuilds.org>). The lower panel shows the project age since the first release or commit. Statistics obtained Oct. 23, 2017.

Reproducible software management and distribution is enhanced by other current technologies. Conda integrates well with environment modules (<http://modules.sourceforge.net/>), a technology used nearly universally across HPC systems. An administrator can use Conda to easily define software stacks for multiple labs and project-specific configurations. Another approach to reproducibility is to use containers, popularized by Docker, which provide a way to publish an entire software stack down to the operating system. They provide greater isolation and control over the environment that software is executed in, but at the expense of some customizability. Conda complements container-based approaches. Where flexibility is needed, Conda packages can be installed directly on the system. Where the uniformity of containers is required, Conda can be used to build images, avoiding the nuanced installation steps that would ordinarily be required to build and install software within an image. In fact, for each Bioconda package, our build system automatically builds a minimal Docker image containing that package and its dependencies, which is subsequently uploaded and made available via the Biocontainers project (da Veiga Leprevost et al. 2017). As a consequence, every built Bioconda package is available not only for installation via Conda, but also as a container via Docker, Rkt ([https://coreos.com/rkt](https://coreos.com/rkt/)), and Singularity (Kurtzer, Sochat, and Bauer 2017), such that the desired level of reproducibility can be chosen freely (Grüning et al. 2017).

By turning the arduous and error-prone process of installing bioinformatics software, previously repeated endlessly by scientists around the globe, into a concerted community effort, Bioconda frees significant resources to instead be invested into productive research. The new simplicity of deploying even complex software stacks with strictly controlled software versions enables software authors to safely rely on existing methods. Previously, the cost of depending on a third party tool - requiring its installation and maintaining compatibility with new versions - was often higher than the effort to re-implement its methods. Now, using Bioconda, authors can simply specify the tool and version required, incurring only negligible costs even for a large number of dependencies.

For reproducible data science, it is crucial that software libraries and tools are provided via an easy to use, unified interface, such that they can be easily deployed and sustainably managed. With its ability to maintain isolated software environments, the integration into major workflow management systems and the fact that no administration privileges are needed, the Conda package manager is the ideal tool to ensure sustainable and reproducible software management. With Bioconda, we unlock Conda for the life sciences while coordinating closely with other related projects such as conda-forge and Biocontainers. Bioconda offers a comprehensive resource of thousands of software libraries and tools that is maintained by hundreds of international contributors. Although it is among the youngest, it outperforms all competing projects by far in the number of available packages. With almost six million downloads so far, Bioconda packages have been well received by the bioinformatics and reproducible research community. We invite everybody to join the Bioconda community, participate in maintaining or publishing new software, and reach the goal of a central, comprehensive, and language-agnostic collection of easily installable software for the life sciences.

# Funding

The Bioconda project has received support from Anaconda, Inc., Austin, TX, USA, in the form of expanded storage for Bioconda packages on their hosting service (<https://anaconda.org>). Further, the project has been granted extended build times from Travis CI, GmbH ([https://travis-ci.com](https://travis-ci.com/)). The Bioconda community would like to thank ELIXIR ([https://www.elixir-europe.org](https://www.elixir-europe.org/)) for their constant support and donating staff. Ryan Dale was supported by the Intramural Program of the National Institute of Diabetes and Digestive and Kidney Diseases, National Institutes of Health, USA. Johannes Köster was supported by the The Netherlands Organisation for Scientific Research (NWO) via a VENI grant.

# Acknowledgements

We thank the participants of various hackathons (e.g., the GalaxyP and IUC contribution fest, ELIXIR BioContainers and NETTAB hackathon) for porting numerous packages to Bioconda.

# Contributions

Kyle Beauchamp, Christian Brueffer, Brad Chapman, Ryan Dale, Florian Eggenhofer, Björn Grüning, Johannes Köster, Elmar Pruesse, Martin Raden, Jillian Rowe, Devon Ryan, Ilya Shlyakter, Andreas Sjödin, Christopher Tomkins-Tinch, and Renan Valieris (in alphabetical order) have written the manuscript. Johannes Köster has drafted the manuscript. Johannes Köster and Ryan Dale have conducted the data analysis. Dan Ariel Søndergaard contributed by supervising student programmers on writing recipes and maintaining the connection with ELIXIR. All other authors have contributed or maintained recipes.

# Online Methods

## Software management with Conda

Via the Conda package manager, installing software from Bioconda becomes very simple. In the following, we describe the basic functionality assuming that the user has access to a Linux or macOS terminal. After installing Conda (<https://conda.io/miniconda.html>), the first step is to set up the Bioconda channel via:

$ conda config --add channels conda-forge  
$ conda config --add channels bioconda

Now all Bioconda packages are visible to the Conda package manager. For example, the software CNVkit (Talevich et al. 2016), can be searched for with

$ conda search cnvkit

in order to check if and in which versions it is available. Alternatively,  packages can be searched interactively at <http://bioconda.github.io>.  CNVkit can be installed with:

$ conda install cnvkit

CNVkit needs various dependencies from Python and R, which would otherwise have to be installed in separate manual steps (Fig. ???a). The above command installs all dependencies needed by CNVkit, including Python and R themselves. Conda enables updating and removing all these dependencies via one unified interface.

A key benefit of Conda is the ability to define isolated, shareable software environments. This can happen ad-hoc, or via YAML (<https://yaml.org>) files. For example, the following file defines an environment consisting of Salmon (Patro et al. 2017) and DESeq2 (Love, Huber, and Anders 2014). Here we specify the exact versions to install, though this is not strictly necessary as the latest available version will be installed by default:

channels:  
 - bioconda  
 - conda-forge  
 - defaults  
dependencies:  
 - bioconductor-deseq2 =1.16.1  
 - salmon =0.8.2  
 - r-base =3.4.1

Assuming that the above environment specification is stored in the file env.yaml, an environment my-env containing the specified requirements and all of their dependencies – including R itself – can be created via the command:

$ conda env create --name my-env --file env.yaml

To use the commands installed in this new environment, it must first be “activated” by issuing the following command. Multiple environments can exist on the same system and only the packages in the currently-activated environment will be accessible:

$ source activate my-env

Within the activated environment, R, Salmon, and DESeq2 are available in exactly the defined versions. For example, Salmon can be executed with:

$ salmon --help

It is possible to modify an existing environment by using conda update, conda install and conda remove. For example, we could add a particular version of Kallisto (Bray et al. 2016) and update Salmon to the latest available version with:

$ conda install kallisto=0.43.1  
$ conda update salmon

Finally, the environment can be deactivated again with the following command:

$ source deactivate

## How isolated software environments enable reproducible research

With isolated software environments as shown above, it is possible to define an exact version for each package. This increases reproducibility by eliminating differences due to implementation changes. Note that above we also pin an R version, although the latest compatible one would also be automatically installed without mentioning it. To further increase reproducibility, this pattern can be extended to all dependencies of DESeq2 and Salmon and recursively down to basic system libraries like zlib and boost (<https://www.boost.org>). Environments are isolated from the rest of the system, while still allowing interaction with it: e.g., tools inside the environment are preferred over system tools, while system tools that are not available from within the environment can still be used. Conda also supports the automatic creation of environment definitions from already existing environments. This allows to rapidly explore the needed combination of packages before it is finalized into an environment definition:

$ conda env export --name my-env > full-env.yaml

This file can then be shared with other researchers, who, with the following two commands, can exactly reproduce the same environment on their system and then activate that environment to use it:

$ conda env create --name from-collaborator --file full-env.yaml  
$ source activate from-collaborator

In combination with workflow management systems like Galaxy (Afgan et al. 2016), bcbio-nextgen (<https://github.com/chapmanb/bcbio-nextgen>), and Snakemake (Köster and Rahmann 2012) that interact directly with Conda, a data analysis can be shipped and deployed in a fully reproducible way, from description and automatic execution of every analysis step down to the description and automatic installation of any required software.

## Security Considerations

Using Bioconda as a service to obtain packages for local installation entails trusting that (a) the provided software itself is not harmful and (b) it has not been modified in a harmful way. Ensuring (a) is up to the user. In contrast, (b) is handled by our workflow. First, source code or binary files defined in recipes are checked for integrity via MD5 or SHA256 hash values. Second, all review and testing steps are enforced via the GitHub interface. This guarantees that all packages have been tested automatically and reviewed by a human being. Third, all changes to the repository of recipes are publicly tracked, and all build and test steps are transparently visible to the user. Finally, the automatic parts of the development workflow are implemented in the open-source software *bioconda-utils* (<https://github.com/bioconda/bioconda-utils>). In the future, we will further explore the possibility to sign packages cryptographically.

## Data analysis

The presented figures and numbers have been generated via a fully automated, reproducible Snakemake (Köster and Rahmann 2012) workflow that is freely available under <https://github.com/bioconda/bioconda-paper>.

# References

Mesirov, J. P. 2010. “Accessible Reproducible Research”. *Science* 327 (5964). American Association for the Advancement of Science (AAAS): 415–16. doi:10.1126/science.1179653.

Baker, Monya. 2016. “1,500 Scientists Lift the Lid on Reproducibility”. *Nature* 533 (7604). Springer Nature: 452–54. doi:10.1038/533452a.

Munafò, Marcus R., Brian A. Nosek, Dorothy V. M. Bishop, Katherine S. Button, Christopher D. Chambers, Nathalie Percie du Sert, Uri Simonsohn, Eric-Jan Wagenmakers, Jennifer J. Ware, and John P. A. Ioannidis. 2017. “A Manifesto for Reproducible Science”. *Nature Human Behaviour* 1 (1). Springer Nature: 0021. doi:10.1038/s41562-016-0021.

Afgan, E, D Baker, den Beek M van, D Blankenberg, D Bouvier, M Čech, J Chilton, et al. 2016. “The Galaxy Platform for Accessible, Reproducible and Collaborative Biomedical Analyses: 2016 Update”. *Nucleic Acids Res* 44: W3–W10. doi:10.1093/nar/gkw343.

Köster, J, and S Rahmann. 2012. “Snakemake–a Scalable Bioinformatics Workflow Engine”. *Bioinformatics* 28: 2520–22. doi:10.1093/bioinformatics/bts480.

Talevich, E, AH Shain, T Botton, and BC Bastian. 2016. “CNVkit: Genome-Wide Copy Number Detection and Visualization from Targeted DNA Sequencing”. *PLoS Comput Biol* 12: e1004873. doi:10.1371/journal.pcbi.1004873.

Field, Dawn, Bela Tiwari, Tim Booth, Stewart Houten, Dan Swan, Nicolas Bertrand, and Milo Thurston. 2006. “Open Software for Biologists: from Famine to Feast”. *Nature Biotechnology* 24 (7). Springer Nature: 801–3. doi:10.1038/nbt0706-801.

da Veiga Leprevost, F, BA Grüning, Aflitos S Alves, HL Röst, J Uszkoreit, H Barsnes, M Vaudel, et al. 2017. “BioContainers: an Open-Source and Community-Driven Framework for Software Standardization”. *Bioinformatics* 33: 2580–82. doi:10.1093/bioinformatics/btx192.

Kurtzer, GM, V Sochat, and MW Bauer. 2017. “Singularity: Scientific Containers for Mobility of Compute”. *PLoS One* 12: e0177459. doi:10.1371/journal.pone.0177459.

Grüning, Björn, John Chilton, Johannes Köster, Ryan Dale, Jeremy Goecks, Rolf Backofen, Anton Nekrutenko, and James Taylor. 2017. “Practical Computational Reproducibility in the Life Sciences”, October. Cold Spring Harbor Laboratory. doi:10.1101/200683.

Patro, R, G Duggal, MI Love, RA Irizarry, and C Kingsford. 2017. “Salmon Provides Fast and Bias-Aware Quantification of Transcript Expression”. *Nat Methods* 14: 417–19. doi:10.1038/nmeth.4197.

Love, MI, W Huber, and S Anders. 2014. “Moderated Estimation of Fold Change and Dispersion for RNA-Seq Data with DESeq2”. *Genome Biol* 15: 550. doi:10.1186/s13059-014-0550-8.

Bray, NL, H Pimentel, P Melsted, and L Pachter. 2016. “Near-Optimal Probabilistic RNA-Seq Quantification”. *Nat Biotechnol* 34: 525–27. doi:10.1038/nbt.3519.

1. Co-first author [↑](#footnote-ref-21)
2. To whom correspondence should be addressed. [↑](#footnote-ref-22)