Interoperable and scalable metabolomics data analysis with microservices

Payam Emami Khoonsari[1], Pablo Moreno[2], Sven Bergmann[3, 4], Joachim Burman[5], Marco Capuccini[6, 7], Matteo Carone[7], Marta Cascante[8, 9], Pedro de Atauri[8, 9], Carles Foguet[8, 9], Alejandra Gonzalez-Beltran[10], Thomas Hankemeier[11], Kenneth Haug[2], Sijin He[2], Stephanie Herman[1, 7], David Johnson[10], Namrata Kale[2], Anders Larsson[12, 7], Steffen Neumann[13, 14], Kristian Peters[13], Luca Pireddu[15], Philippe Rocca-Serra[10], Pierrick Roger[16], Rico Rueedi[3, 4], Christoph Ruttkies[13], Noureddin Sadawi[17], Reza M Salek[2], Susanna-Assunta Sansone[10], Daniel Schober[13], Vitaly Selivanov[8, 9], Etienne A. Thévenot[16], Michael van Vliet[11], Gianluigi Zanetti[15], Christoph Steinbeck[2, 18], Kim Kultima[1], Ola Spjuth[7*]

1. Department of Medical Sciences, Clinical Chemistry, Uppsala University, Uppsala, Sweden
2. European Molecular Biology Laboratory, European Bioinformatics Institute (EMBL-EBI), Cambridge, United Kingdom
3. Department of Computational Biology, University of Lausanne, Lausanne, Switzerland
4. Swiss Institute of Bioinformatics, Lausanne, Switzerland
5. Department of Neuroscience, Uppsala University, Uppsala, Sweden
6. Department of Information Technology, Uppsala, Sweden
7. Department of Pharmaceutical Biosciences, Uppsala University, Uppsala, Sweden
8. Department of Biochemistry and Molecular Biomedicine, Faculty of Biology, Universitat de Barcelona, Barcelona, Spain
9. Institute of Biomedicine of the Universitat de Barcelona (IBUB) and Associated Unit to CSIC, Barcelona, Spain
10. Oxford e-Research Centre, Department of Engineering Science, University of Oxford, Oxford, United Kingdom
11. Division of Analytical Biosciences, Leiden Academic Centre for Drug Research, Leiden University, Leiden, The Netherlands
12. National Bioinformatics Infrastructure Sweden, Uppsala University, Uppsala, Sweden
13. Department of Stress and Developmental Biology, Leibniz Institute of Plant Biochemistry, Halle, Germany
14. German Centre for Integrative Biodiversity Research (iDiv), Halle-Jena-Leipzig, Germany
15. CRS4: Center for Advanced Studies, Research and Development in Sardinia, Pula, Italy
16. CEA, LIST, Laboratory for Data Analysis and Systems' Intelligence, MetaboHUB, Gif-sur-Yvette, France
17. Faculty of Medicine, Department of Surgery & Cancer, Imperial College London, London, United Kingdom
18. Friedrich-Schiller-University, Jena, Germany

* Corresponding author

Abstract

Developing a robust and performant data analysis workflow that integrates all necessary components whilst still being able to scale over multiple compute nodes is a challenging task. We here present a generic method based on the microservice architecture, where software tools are encapsulated as Docker containers that can be connected into scientific workflows and executed in parallel using the Kubernetes container orchestrator. The method was developed within the PhenoMeNal consortium to support flexible metabolomics data analysis and was designed as a virtual research environment which can be launched on-demand on cloud resources and desktop computers. IT-expertise requirements on the user side are kept to a minimum, and established workflows can be re-used effortlessly by any novice user. We validate our method on two mass spectrometry studies, one nuclear magnetic resonance spectroscopy study and one fluxomics study, showing that the method scales dynamically with increasing availability of computational resources. We achieved a complete integration of the major software suites resulting in the first turn-key workflow encompassing all steps for mass-spectrometry-based metabolomics including preprocessing, multivariate statistics, and metabolite identification. Microservices is a generic methodology that can serve any scientific discipline and opens up for new types of large-scale integrative science.

Introduction

Metabolomics studies measure the occurrence, concentrations and changes of small molecules (metabolites) in organisms, organs, tissues, cells and cellular compartments. Metabolite abundances are assayed in the context of environmental or dietary changes, disease or other conditions[1]. Metabolomics experimental measurements are performed using a variety of spectroscopic methods: the two most common ones are Mass Spectrometry (MS) and Nuclear Magnetic Resonance (NMR). The use of metabolomics as a molecular phenotyping technique is growing across all biomedical domains, due to its ability to reflect the influence of external factors to which an organism is exposed, such as stress, nutrition and disease, subsumed under the term 'exposome'[2]. Metabolomics data analysis has matured over the years, but is still largely developed and performed at a laboratory level with the use of conventional computing solutions and little standardisation for reproducible research. The PhenoMeNal (**Pheno**me and **Me**tabolome a**Nal**ysis) project (http://phenomenal-h2020.eu/home/about) was conceived to ameliorate this situation by bringing advancements in computing architecture and technology into a modern and easily deployed e-infrastructure – i.e., a computing environment combining hardware and software technology as well as required protocols and data resources – tailored specifically for efficient processing and analysis of molecular phenotype data.

Metabolomics is, as most other omics technologies, characterized by the use of high-throughput experiments that produce large amounts of data[3]. With increasing data size and number of

samples, the analysis process becomes intractable for desktop computers due to requirements on compute cores, memory, storage etc. As a result, large-scale computing infrastructures have become important components in scientific projects[4]. Moreover, making use of such complex computing resources in an analysis workflow presents its own challenges, including achieving efficient job parallelism and scheduling as well as error handling[5]. In addition, configuring the necessary software tools and chaining them together into a complete re-runnable analysis workflow commonly requires substantial IT-expertise, while creating portable and fault-tolerant workflows with a robust audit trail is even more difficult.

Currently, the most common large-scale computational infrastructures in science are shared High-Performance Computing (HPC) systems. Such systems are usually designed primarily to support computationally intensive batch jobs – e.g., for the simulation of physical processes – and are managed by specialized system administrators. This model leads to rigid constraints on the way these resources can be used. For instance, the installation of software must undergo approval and may be restricted, which contrasts with the needs in omics analysis where a multitude of software components of various versions – and their dependencies – are needed, and where these need to be continuously updated.

Cloud computing offers a compelling alternative to shared HPC systems, with the possibility to instantiate and configure on-demand resources such as virtual computers, networks, and storage, together with operating systems and software tools. Users only pay for the time the virtual resources are used, and when they are no longer needed they can be released and incur no further

costs for usage or ownership. A few examples of cloud-based systems for metabolomics include XCMS ONLINE[6], Chorus (chorusproject.org) and The Metabolomics Workbench (www.metabolomicsworkbench.org), all of which provide virtual environments that scale with computational demands. However, these applications provide limited flexibility in terms of incorporating and maintaining tools as well as constructing and using customizable workflows.

Along with infrastructure provisioning, software provisioning – i.e., installing and configuring software for users – has also advanced. Consider, for instance, containerization[7], which allows entire applications with their dependencies to be packaged, shipped and run on a computer but isolated from one another in a way analogous to virtual machines, yet much more efficiently. Containers are more compact, and since they share the same operating system kernel, they are fast to start and stop and incur little overhead in execution. These traits make them an ideal solution to implement light-weight *microservices*, a software engineering methodology in which complex applications are divided into a collection of smaller, loosely coupled components that communicate over a network[8]. Microservices share many properties with traditional always-on web services found on the Internet, but microservices are generally smaller, portable and can be started on-demand within a separate computing environment. Another important feature of microservices is that they have a technology-agnostic communication protocol, and hence can serve as building blocks that can be combined and reused in multiple ways[9].

Microservices are highly suitable to run in elastic cloud environments that can dynamically grow or shrink on demand, enabling applications to be scaled-up by simply starting multiple parallel instances of the same service. However, to achieve effective scalability a system needs to be

appropriately sectioned into microservice components and the data to be exchanged between the microservices needs to be defined for maximum efficiency– both being challenging tasks.

In this manuscript, we present a method which uses components for metabolomics data analysis encapsulated as microservices and connected into computational workflows to provide complete, ready-to-run, reproducible data analysis solutions that can be easily deployed on desktop computers as well as public and private clouds. Our approach requires virtually no involvement in the setup of computational infrastructure and no special IT skills from the user. We validate the method on four metabolomics studies and show that it enables scalable and interoperable data analysis.

Results

Microservices

In order to construct a microservice architecture for metabolomics we used Docker[10] (https://www.docker.com/) containers to encapsulate a large suite of software tools (See Table S1). To automate the instantiation of this cloud-portable microservice-based system and its components for metabolomics analysis, we developed a Virtual Research Environment (VRE) which uses Kubernetes (https://kubernetes.io/) to orchestrate containers over multiple compute nodes. Scientists can interact with the microservices programmatically via an Application Programming Interface (API) or via a web-based graphical user interface (GUI), as illustrated in Figure 1. To connect microservices into computational workflows, the two frameworks Galaxy[11] and Luigi (https://github.com/spotify/luigi) were adapted to execute jobs on Kubernetes. Galaxy

is a web-based interface for individual tools and allows users to share workflows and analysis histories. Luigi on the other hand focuses on scheduled execution, monitoring, visualization and the implicit dependency resolution of tasks[12]. These basic infrastructure services, together with the Jupyter notebook[13] interactive programming environment, are deployed as long-running services in the VRE, whereas the other analysis tools are deployed as transient compute jobs to be used on-demand. System and client applications were developed for launching the VRE on desktop computers, public and private cloud providers, automating all steps required to instantiate the virtual infrastructures. The PhenoMeNal consortium maintains a web portal (https://portal.phenomenal-h2020.eu) providing a GUI for launching VREs on a selection of the largest public cloud providers, including Amazon Web Services, Microsoft Azure and Google Cloud Platform, or on private OpenStack-based installations. The containers provisioned by PhenoMeNal comprise tools built as open source software that are available in a public repository such as GitHub, and are subject to continuous integration testing. The containers that satisfy testing criteria are pushed to a public container repository, and containers that are included in stable VRE releases are also pushed to Biocontainers[9].

Demonstrator 1: Scalability of microservices in a cloud environment in the analysis of a human renal proximal tubule cells dataset

The objective of this analysis was to demonstrate the scalability of an existing workflow on a large dataset (MetaboLights[14] ID: MTBLS233, http://www.ebi.ac.uk/metabolights/MTBLS233[15]). The experiment includes 528 mass spectrometry samples from whole cell lysates of human renal proximal tubule cells that were

pre-processed through a five-step workflow (consisting of peak picking, feature finding, linking, file filtering and exporting) using the OpenMS software[16] as illustrated in Figure 2. This preprocessing workflow was reimplemented using Docker containers and run using the Luigi workflow engine. Scalability of concurrent running tools (on 40 Luigi workers, each worker receives tasks from the scheduler and executes them) was measured using weak scaling efficiency (WSE), where the workload assigned to each worker stays constant and additional workers are used to solve a larger total problem. The WSE was computed to reach 88% with an execution time of approximately four hours (online methods, Figure S2), compared with the ideal case of 100% where linear scaling is achieved if the run time stays constant while the workload is increased. In addition, the final result of the workflow (online methods, Figure S3) was identical to that presented by the original MTBLS233 study (Ranninger et al.[15]) in negative ionization mode. However, in the positive ionization mode, one m/z feature was found in a different group (m/z range 400-1000) than it was originally reported by Ranninger et al. (m/z range 200-400).

Demonstrator 2: Start-to-end LC-MS-analysis workflow on Multiple Sclerosis data

The objective of this analysis was to demonstrate interoperability as well as to present a real-world scenario in which patients' data are processed using a microservices-based platform. We used a dataset consisting of 37 clinical cerebrospinal fluid (CSF) samples including thirteen relapsing-remitting multiple sclerosis (RRMS) patients and 14 secondary progressive multiple sclerosis (SPMS) patients as well as ten non-multiple sclerosis controls. Twenty-six quality controls (19 blank and 7 dilution series samples) were also added to the experiment. In addition,

8 pooled CSF samples containing MS/MS data were included in the experiment for improving identification (MetaboLights ID: MTBLS558, http://www.ebi.ac.uk/metabolights/MTBLS558). The samples were processed and analysed on the Galaxy platform[11] applying the Liquid Chromatography-MS (LC-MS) workflow illustrated in Figure 3, running in a PhenoMeNal VRE behind the Uppsala University Hospital firewall to be compliant with local ELSI (Ethics, Legal, Social implications) regulations. The result of multivariate analysis showed a clear difference (Figure 4A) in the metabolic constitution between the three disease groups of RRMS, SPMS and non-multiple sclerosis controls. In addition, the univariate analysis resulted in a total of three metabolites being significantly altered ($p<0.05$) between multiple sclerosis subtypes and control samples, namely alanyltryptophan and indoleacetic acid with higher and linoleoyl ethanolamide with lower abundance in both RRMS and SPMS compared to controls (Figure 4B).

Demonstrator 3: 1D NMR-analysis workflow on human type 2 diabetes mellitus data

This NMR-based metabolomics study was originally performed by Salek et al.[17] on urine of type 2 diabetes mellitus (T2DM) patients and controls (MetaboLights ID: MTBLS1, http://www.ebi.ac.uk/metabolights/MTBLS1). In total, 132 samples (48 T2DM and 84 controls) were processed using the workflow shown in Figure 5. A total of 726 metabolites were quantified and used to perform Orthogonal Projections to Latent Structures Discriminant Analysis (OPLS-DA) which resulted in a clear separation between T2DM and controls (Figure 5), reproducing previous findings[17].

Demonstrator 4: Start-to-end fluxomics workflow on HUVEC cells under hypoxia

The purpose of this demonstrator was to show the integrated use of separately developed tools covering subsequent steps of the study of metabolic fluxes based on ^{13}C stable isotope-resolved metabolomics (*SIRM*)[18,19,20]. Here we implemented the analysis of flux distributions in HUVEC cells under hypoxia (MetaboLights ID: MTBLS412, http://www.ebi.ac.uk/metabolights/MTBLS412), from raw mass spectra contained in netCDF files, using the workflow illustrated in Figure 6. The result was a detailed description of the magnitudes of the fluxes through the reactions accounting for glycolysis and pentose phosphate pathway.

Discussion

Implementing the different tools and processing steps of a data analysis workflow as separate services that are made available over a network was in the spotlight in the early 2000's[21] as service-oriented architectures (SOA) in science. At that time, web services were commonly deployed on physical hardware and exposed and consumed publicly over the internet. However, it soon became evident that this architecture did not fulfill its promises as it did not scale well from a computational perspective. In addition, the web services were not portable and mirroring them was complicated (if at all possible). Furthermore, API changes and frequent services outage made it frustrating to connect them into functioning computational workflows. Ultimately, the ability to replicate an analysis on local and remote hardware (such as a computer cluster) was very difficult due to heterogeneity in the computing environments.

At first sight microservices might seem similar to abovementioned SOA web services, but microservices are generally executed in virtual environments (abstracting over OS and hardware architectures) in such a way that they are only instantiated and executed on-demand, and then terminated when they are no longer needed. This makes such virtual environments inherently portable and they can be launched on demand on different platforms (e.g., a laptop, a powerful physical server or an elastic cloud environment). A key aspect is that workflows are still executed identically, agnostic of the underlying hardware platform. Container-based microservices provide a wide flexibility in terms of versioning, allowing the execution of newer and older versions of each container as needed for reproducibility. Since all software dependencies are encompassed within the container, which is versioned, the risk of workflow failure due to API changes is minimized. An orchestration framework such as Kubernetes further allows for managing errors in execution and transparently handles the restarting of services. Hence, technology has caught up with service-oriented science, and microservices have taken the methodology to the next level, alleviating many of the previous problems related to scalability, portability and interoperability of software tools. This is advantageous in the context of omics analysis, which produces multidimensional data sets reaching beyond gigabytes, on into terabytes, leading to ever-increasing demand on processing performance[22,23].

In Demonstrator 1, we showed that microservices enable highly efficient and scalable data analyses by executing individual modules in parallel, and that they effectively harmonize with on-demand elasticity of the cloud computing paradigm. The reached scaling efficiency of ~88% indicates remarkable performance achieved on generic cloud providers. Furthermore, although

our results in positive ionization model was slightly different to that of Ranninger et al.[15], the results of our analysis were reproducible regardless of the platform used to perform the computations, indicating a level of replicability of study results and reusability of workflows in the analysis that - to the best of our knowledge - has never been reported before in metabolomics data analysis.

In addition to the fundamental demand for high performance, the increased throughput and complexity of omics experiments has led to a large number of sophisticated computational tools[24], which in turn necessitates integrative workflow engines[25]. In order to integrate new tools in such workflow engines, compatibility of the target environment, tools and APIs needs to be considered[25]. Containerization facilitates this by providing a platform-independent virtual environment for developing and running the individual tools. However, the problem of compatibility between tools/APIs and data formats remains and needs to be tackled by international consortia similarly to what PhenoMeNal addresses in metabolomics by promoting and strictly adhering to FAIR Data Principles[26]. PhenoMeNal also overcomes the currently non-trivial task of instantiating the complete microservice environments through a web portal that allows for convenient deployment of the VRE on public cloud providers. Moreover, using the PhenoMeNal web portal, microservices and VREs can be deployed on a trusted private cloud instance or a local physical server on an internal network, such as within a hospital network, allowing for levels of isolation and avoiding transfer of data across untrusted networks which often are requirements in the analysis of sensitive data. This was highlighted in Demonstrator 2, where a complete start-to-end workflow was run on the Galaxy platform on a secure server at

Uppsala University Hospital, Sweden, leading to the identification of novel disease fingerprints in the CSF metabolome of RRMS and SPMS patients. It is worth mentioning that the selected metabolites were part of tryptophan metabolism (alanyltryptophan and indoleacetic acid) and endocannabinoids (linoleoyl ethanolamide), both of which have been previously implicated in multiple sclerosis[27–32]. However, since the cross-validated predictive performance (Q2Y = 0.286) is not much higher than some of the models generated after random permutation of the response (Figure 4A), the quality of the model needs to be confirmed in a future study on an independent cohort of larger size.

In Demonstrator 3, we highlighted the fact that the microservice architecture is indeed domain-agnostic and is not limited to a particular assay technology, i.e. mass spectrometry. Using a fully automated 1D NMR workflow, we showed that the pattern of the metabolite expression is different between type 2 diabetic and healthy controls, and that a large number of metabolites contribute to such separation. The preprocessing of NMR-based experiments can be performed with minimal effort on other studies (i.e. simply by providing a MetaboLights accession number), leading to the capability to re-analyze data and compare the results with the original publication findings. Furthermore, it demonstrates the value of standardised dataset descriptions using nmrML[33] and ISA format[34,35] for representing NMR based studies, as well as the potential of the PhenoMenNal VRE to foster reproducibility.

A complete understanding of metabolic function implies a complete metabolic profile, but also knowledge of the associated distribution of metabolic fluxes in the metabolic network. In

Demonstrator 4, the microservices architecture is applied to deal with flux distributions derived from the application of stable isotope resolved metabolomics. Here we showed high rate of glycolysis in cell cultured in hypoxia which is consistent with the one expected for endothelial cells[36] and also further confirmation on how these cells maintain energy in low oxygen environments and without oxidative phosphorylation[37,38].

While microservices are not confined to metabolomics and generally applicable to a large variety of applications, there are some important implications and limitations of the method. Firstly, tools need to be containerized in order to operate in the environment. This is however not particularly complex, and an increasing number of developers provide containerized versions of their tools on public container repositories such as Dockerhub or Biocontainers[9]. Secondly, uploading data to a cloud-based system can take a considerable amount of time, and having to re-do this every time a VRE is instantiated can be time-consuming. This can be alleviated by using persistent storage on a cloud resource, but the availability of such storage varies between different cloud providers. Further, the storage system can become a bottleneck when many services try to access a shared storage. We observe that using a distributed storage system with multiple storage nodes can drastically increase performance, and the PhenoMeNal VRE comes with a distributed storage system by default. When using a workflow system to orchestrate the microservices, stability and scalability are inherently dependent on the workflow system's job runner. We observed that in the Galaxy workflow engine, executing a large number of jobs resulted in the VRE becoming unresponsive whereas the Luigi engine did not have these shortcomings. Although this problem can be resolved by defining the required resources in the

Galaxy job runner for each tool, the issue of knowing how much computational resources a specific tool needs remains. This can be partially addressed by tool/workflow developers to estimate the required resources for their tools and workflows. With cloud and microservices maturing, workflow systems will need to evolve and further embrace the new possibilities of these infrastructures. Also, not all research can be easily pipelined, for example exploratory research might be better carried out in an ad-hoc manner than with workflows and the overhead this implies. A Jupyter Notebook as used in in Demonstrator 1 or embedded in Galaxy[39] constitute promising ways to make use of microservices for interactive analysis.

In summary, we showed that microservices allow for efficiently scaling up analyses on multiple computational nodes, enabling the processing of large data sets. By applying a number of data (mzML[40] , nmrML) and metadata standards (ISA serialisations for study descriptions[34,35]), we also demonstrated a level of interoperability which has never been achieved in the context of metabolomics, by providing completely automated start-to-end analysis workflows for mass spectrometry and NMR data. The PhenoMeNal VRE realizes the notion of "bringing compute to the data" by enabling the instantiation of complete virtual infrastructures close to large datasets that could not be uploaded over the internet, and can also be launched close to ELSI sensitive data that is not allowed to leave a secure computing environment. While the current PhenoMeNal VRE implementation uses Docker for software containers and Kubernetes for container orchestration, the microservice methodology is general and not restricted to these frameworks. In addition, we emphasise that the presented methodology goes beyond

metabolomics and can be applied to virtually any field, lowering the barriers for taking advantage of cloud infrastructures and opening up for large-scale integrative science.

Main figures

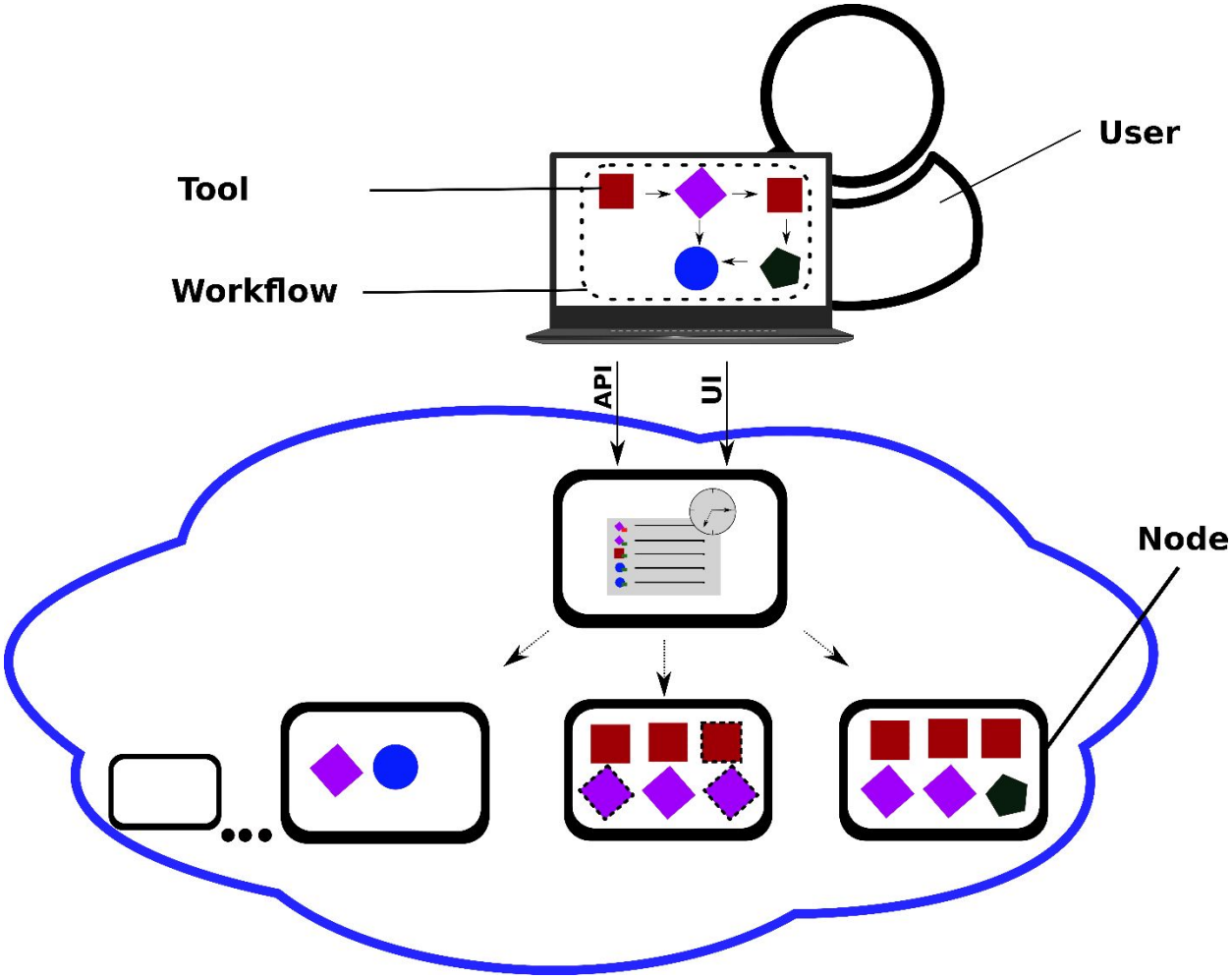

Figure 1: Overview of the components in a microservices-based framework. Complex applications are divided into smaller, focused and well-defined (micro-) services. These services are independently deployable and can communicate with each other, which allows to couple them into data processing workflows. The user can interact with the framework programmatically via an Application Program Interface (API) or via a graphical user interface (GUI) to construct or run workflows of different services, which are executed independently. Multiple instances of services can be launched to execute tasks in parallel, which effectively can be used to scale analysis over multiple compute nodes. When run in an

elastic cloud environment, virtual resources can be added or removed depending on the computational requirements.

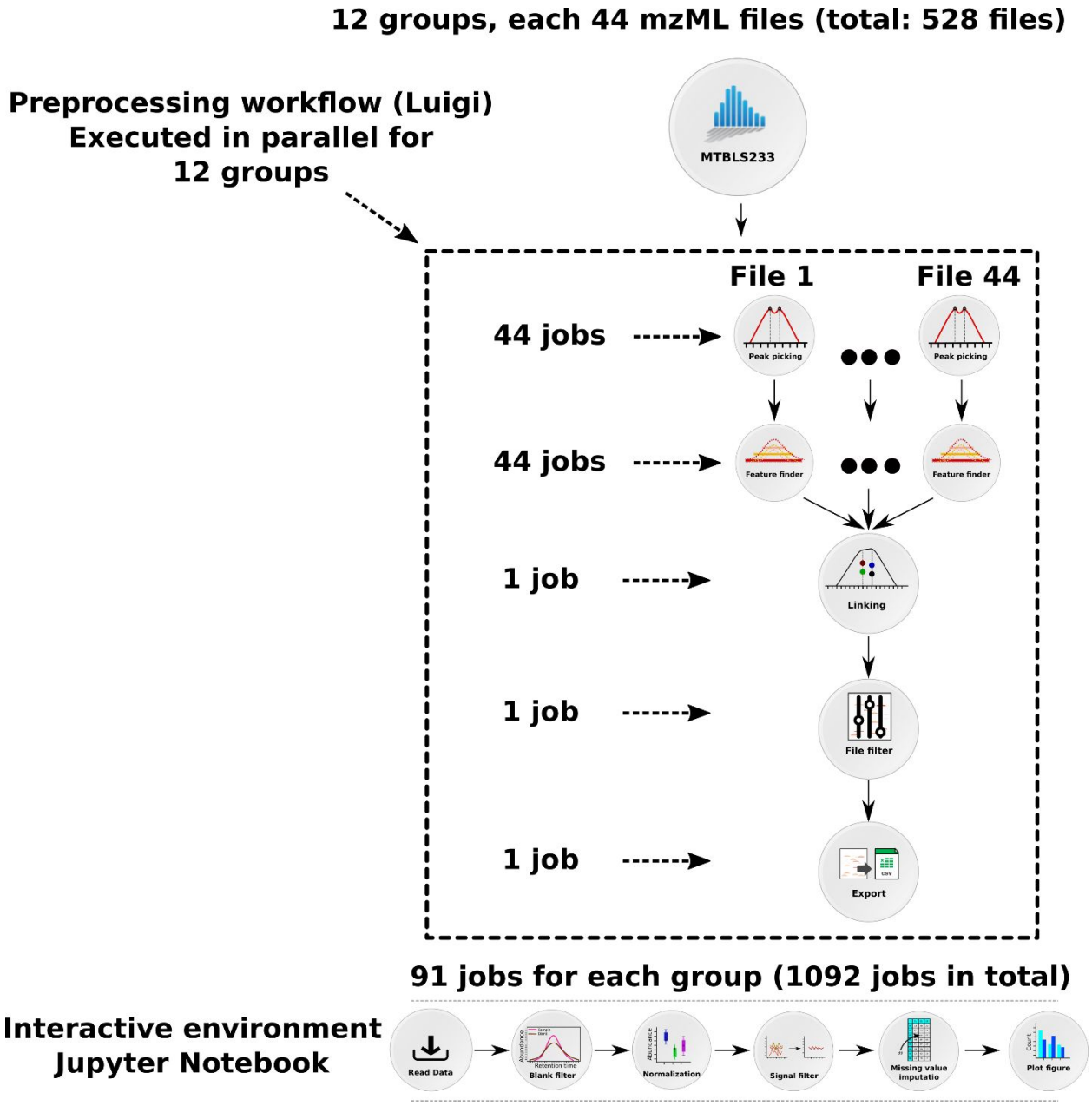

Figure 2. Diagram of scalability-testing on the metabolomics dataset (MetaboLights ID: MTBLS233) in Demonstrator 1 to illustrate the scalability of a microservice approach. The preprocessing workflow is

composed of 5 OpenMS tasks that were run in parallel over the 12 groups in the dataset using the Luigi workflow system. The first two tasks, peak picking (528 tasks) and feature finding (528 tasks), are trivially parallelizable, hence they were run concurrently for each sample. The subsequent feature linking task needs to process all of the samples in a group at the same time, therefore 12 of these tasks were run in parallel. In order to maximize the parallelism, each feature linker container (microservice) was run on 2 CPUs. Feature linking produces a single file for each group, that can be processed independently by the last two tasks: file filter (12 tasks) and text exporter (12 tasks), resulting in total of 1092 tasks. The downstream analysis consisted of 6 tasks that were carried out in a Jupyter Notebook. Briefly, the output of preprocessing steps was imported into R and the unstable signals were filtered out. The missing values were imputed and the resulting number of features were plotted.

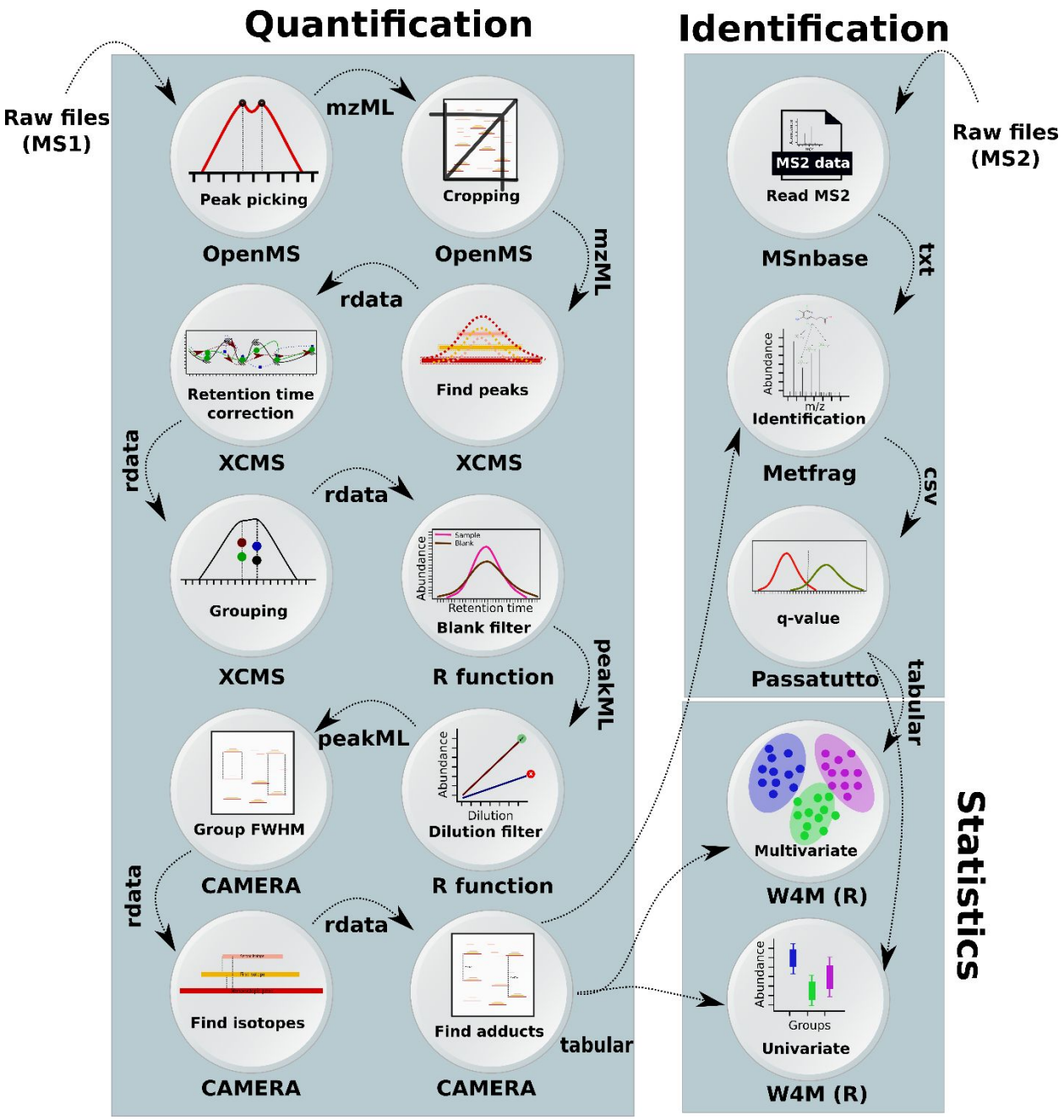

Figure 3. Overview of the workflow used to process multiple-sclerosis samples in Demonstrator 2, where a workflow was composed of the microservices using the Galaxy system. The data was centroided and limited to a specific mass over charge (m/z) range using OpenMS tools. The mass traces quantification and retention time correction was done via XCMS[41]. Unstable signals were filtered out based on the blank and dilution series samples using an in-house function (implemented in R). Annotation

of the peaks was performed using CAMERA[42]. To perform the metabolite identification, the tandem spectra from the MS/MS samples in mzML format were extracted using MSnbase and passed to MetFrag. The MetFrag scores were converted to q-values using Passatutto software. The result of identification and quantification were used in "Multivariate" and "Univariate" containers from Workflow4Metabolomics[43] to perform Partial Least Squares Discriminant Analysis (PLS-DA)[44].

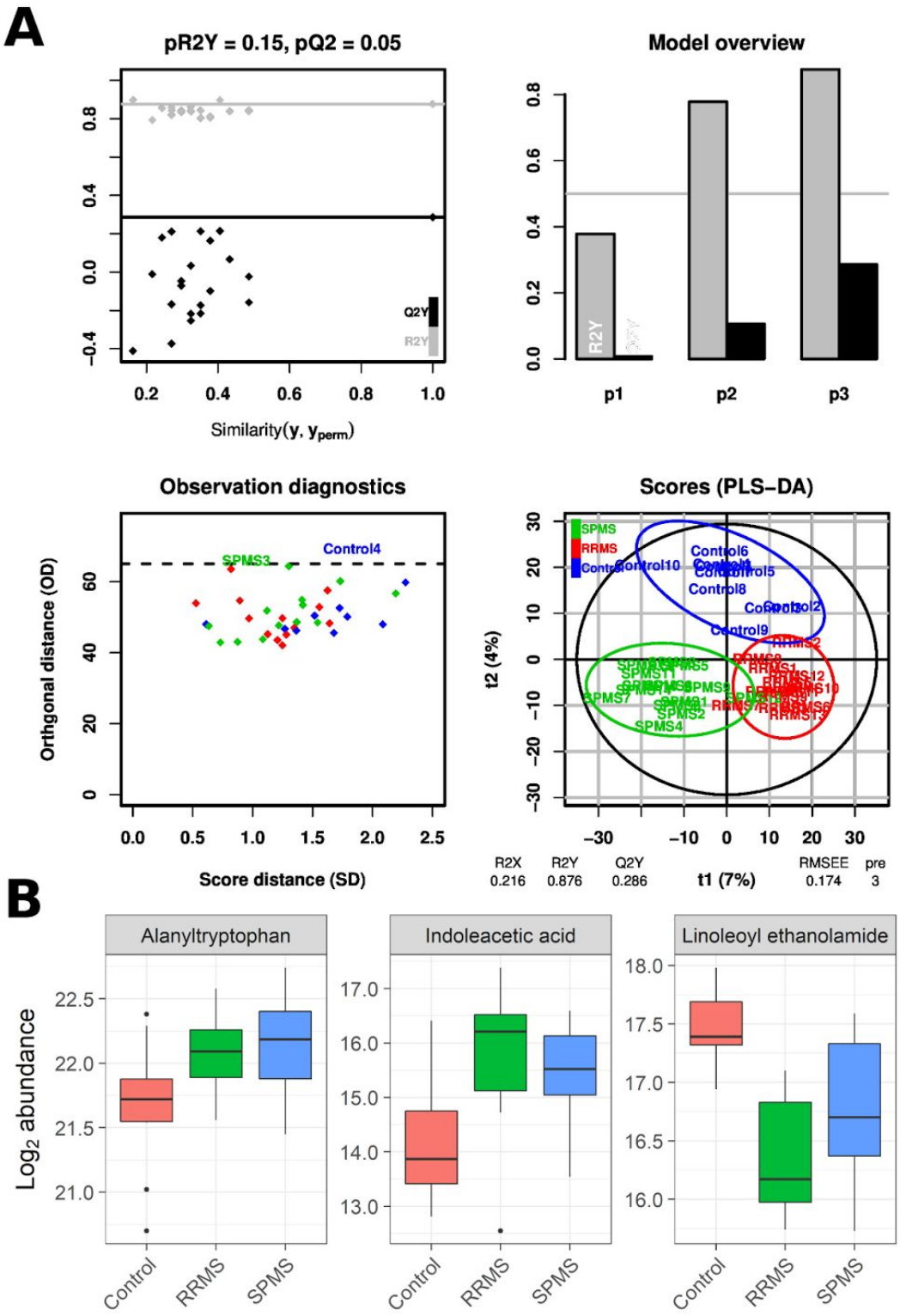
A
pR2Y = 0.15, pQ2 = 0.05
Q2Y
R2Y
Similarity(y, y_perm)
Model overview
R2Y
Q2Y
p1
p2
p3
Observation diagnostics
SPMS3
Control4
Orthgonal distance (OD)
Score distance (SD)
Scores (PLS−DA)
SPMS
RRMS
Control
t2 (4%)
t1 (7%)
R2X R2Y Q2Y RMSEE pre
0.216 0.876 0.286 0.174 3
B
Alanyltryptophan
Indoleacetic acid
Linoleoyl ethanolamide
Log2 abundance
Control
RRMS
SPMS

Figure 4. The results from analysis of multiple sclerosis data in Demonstrator 2, presenting new scientifically useful biomedical knowledge. A) The PLS-DA results suggest that the metabolite distribution in the RRMS and SPMS samples are different to controls. B) Three metabolites were identified as differentially regulated between multiple sclerosis subtypes and control samples, namely alanyltryptophan and indoleacetic acid with higher and Linoleoyl ethanolamide with lower abundance in both RRMS and SPMS compared to controls. Abbr., RRMS: relapsing-remitting multiple sclerosis, SPMS: secondary progressive multiple sclerosis.

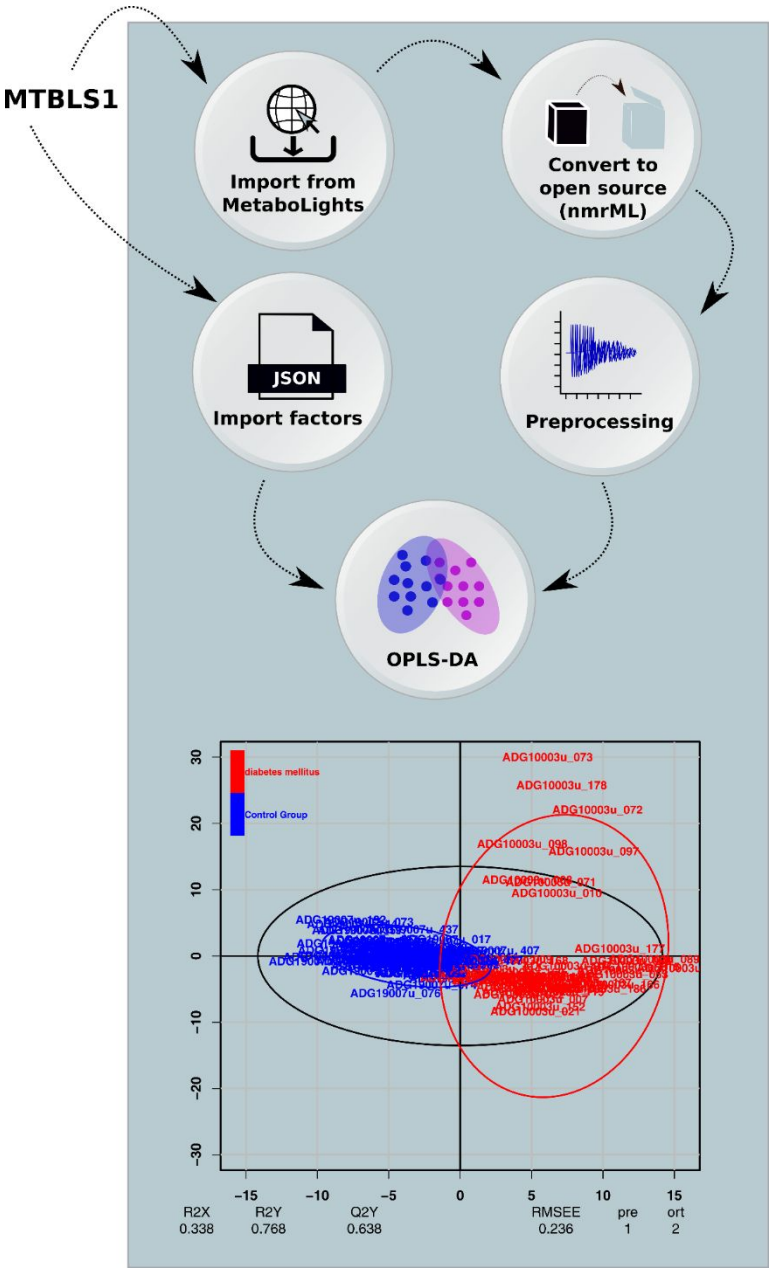

Figure 5. Overview of the NMR workflow in Demonstrator 3. The raw NMR data was automatically import from Metabolights (ISA-Tab) database and converted to open source nmrML format. The preprocessing was performed using the rnmr1d package part of nmrprocflow[45] tools. All study factors were imported from MetaboLights and were fed to the multivariate node to perform an OPLS-DA.

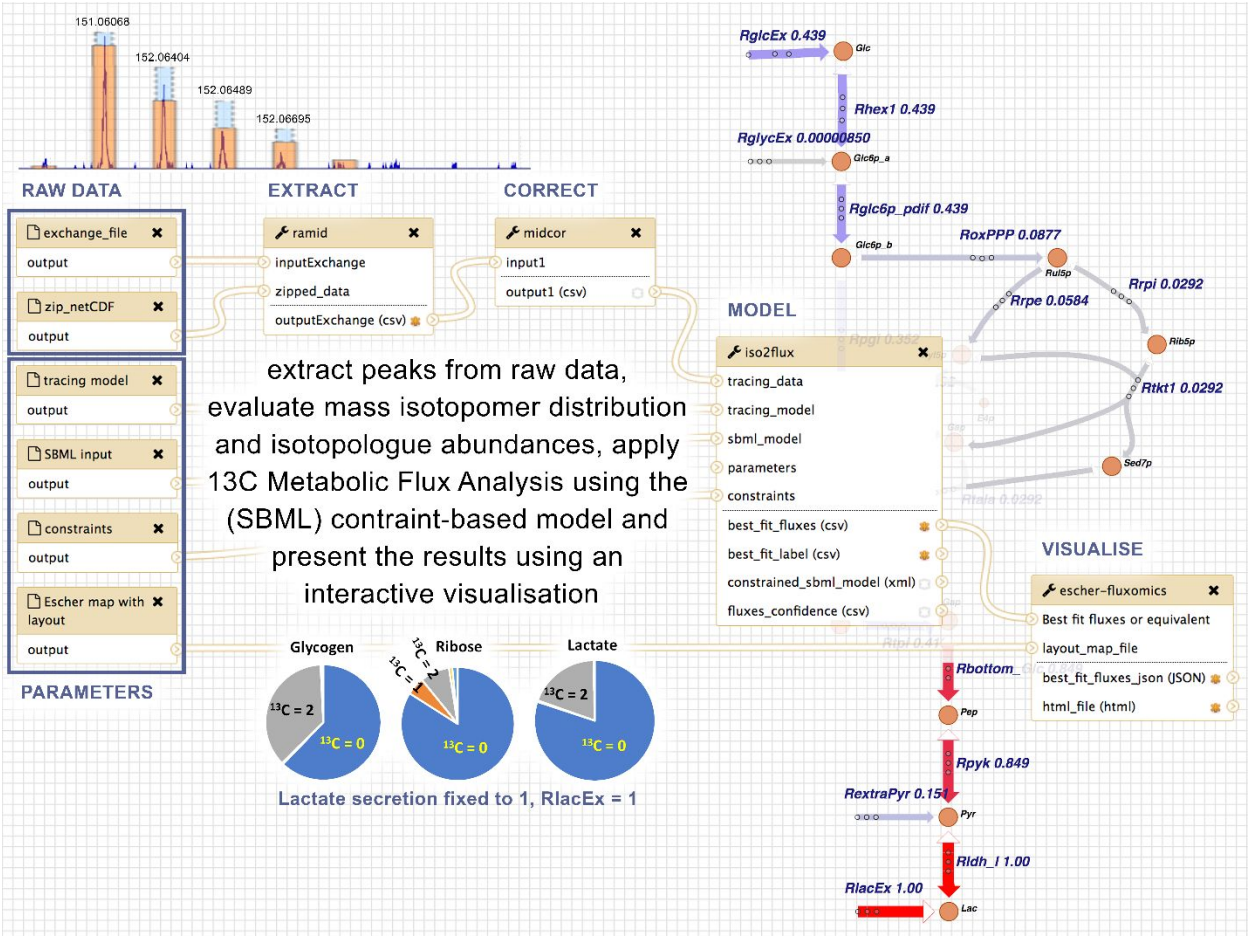

Figure 6: Overview of the workflow for fluxomics, with Ramid, Midcor, Iso2Flux and Escher-fluxomics tools supporting subsequent steps of the analysis. The example refers to HUVEC cells incubated in the presence of [1,2-$^{13}C_2$]glucose and label (^{13}C) propagation to glycogen, RNA ribose and lactate measured by mass spectrometry. Ramid reads the raw netCDF files, corrects baseline and extracts the peak intensities. The resulting peak intensities are corrected (natural abundance, overlapping peaks) by Midcor, which provides isotopologue abundances. Isotopologue abundances, together with a model description (SBML model, tracing data, constraints), are used by Iso2Flux to provide flux distributions through glycolysis and

pentose-phosphate pathways, which are shown as numerical values associated to a metabolic scheme of the model by the Escher-fluxomics tool.

Online methods

Containers

Our microservice-based architecture uses Docker (https://www.docker.com/) containers to encapsulate software tools. Tools are developed as open source and are available in a public repository such as GitHub (https://github.com/), and the PhenoMeNal project containers are built and tested on a Jenkins continuous integration (CI) server (http://phenomenal-h2020.eu/jenkins/). Containers are assembled in different branches using the git versioning system. Builds originating from the development branch of each container repository give rise to container images tagged as 'development'; builds coming from the master branches result in release images. PhenoMeNal produces guidelines on the release process to be followed by developers for each container (see the PhenoMeNal wiki; https://github.com/phnmnl/phenomenal-h2020/wiki). For a container image built to be pushed to the PhenoMeNal container registry (container-registry.phenomenal-h2020.eu/; see Figure S1), it must first pass simple testing criteria regarding the correct provision and executability of defined binaries inside the container. Containers tagged for release are also pushed to the BioContainers Docker Hub (https://hub.docker.com/r/biocontainers/). All published containers are thus available for download and can be used in any microservice architecture.

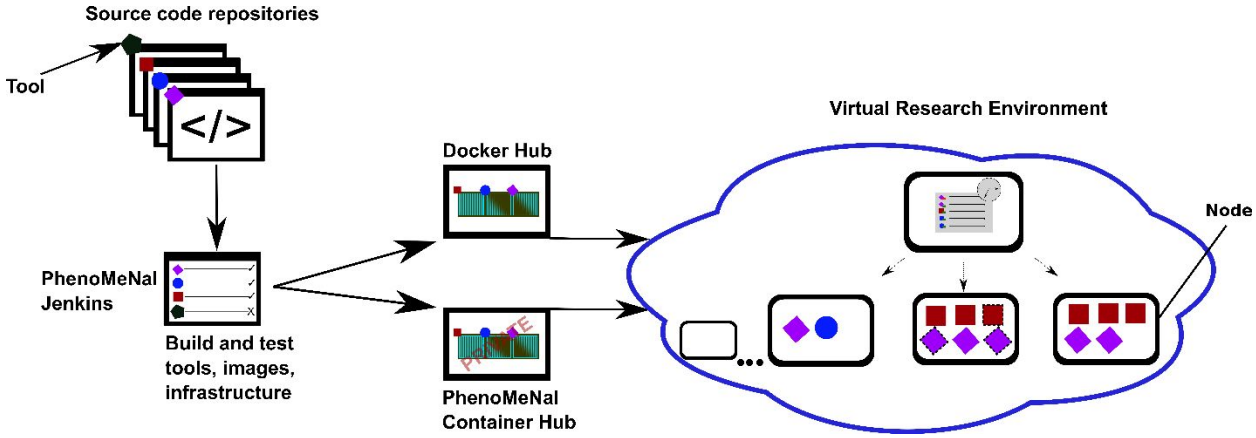

Figure S1: Overview of the continuous development and operation in PhenoMeNal. Source code for tools are pulled into a continuous integration system (Jenkins), where the containers are assembled, tested, and if passing the tests pushed to one or more container repositories from where they are made available to the VREs. While the depiction of the PhenoMeNal registry says private, this only means that it is private for depositing containers; the containers on the PhenoMeNal container registry are publicly available for anyone to use them.

PhenoMeNal Virtual Research Environment (VRE)

The PhenoMeNal VRE is a virtual environment based on the Kubernetes container orchestrator (https://kubernetes.io/). Container orchestration includes initialisation and scaling of jobs based on containers, abstractions to file system access for running containers, exposure of services within and outside the VRE, safe provisioning of secrets to running containers, resource usage administration, scheduling of new container based jobs as well as rescheduling of failed jobs and long running services (all based on containers) to ensure that a desired state is maintained (ie. n number of Galaxy instances must be running).

There are two main classes of services in a PhenoMeNal VRE: long-lasting services, and compute jobs. Long-lasting services are applications that need to be continuously up and running (e.g. a user interface, like the Galaxy workflow environment), while compute jobs are short-lived services that perform temporary functions in data processing and thus are only executed on demand and destroyed afterwards. Long-lasting services and jobs are implemented using both Kubernetes deployments and replication controllers, while jobs are implemented using Kubernetes jobs. The deployment of long term running services is normally more complex in terms of number of components (deployment/replication controller, service, secrets, configuration maps, front end container, database container, etc) compared to compute jobs (which rely only on a single container, a command and access to read data and write results). As such, PhenoMeNal uses the Helm package manager (https://github.com/kubernetes/helm) for Kubernetes, and has produced Helm charts for Galaxy, Jupyter, the PhenoMeNal Portal (towards automatic deployments behind firewalls) and Luigi. The use of Helm means that PhenoMeNal long-term services, and by extension the compute jobs triggered by them, can be executed on any Kubernetes cluster with access to a shared file system, and not only those provisioned by PhenoMeNal.

Kubernetes does not cover host cloud and virtual machine provisioning. For this reason, we developed KubeNow (https://github.com/kubenow/KubeNow), which is a solution based on Terraform (https://www.terraform.io/), Ansible (https://www.ansible.com/) and kubeadm (https://github.com/kubernetes/kubeadm). KubeNow makes it easy to instantiate a complete virtual infrastructure of compute nodes, shared file system storage, networks, configure DNS, operating system, container implementation and orchestration tools on a local computer or server

(via VirtualBox or KVM), a private cloud (e.g. OpenStack), or a public cloud provider (e.g. Google Cloud, Amazon Web Services, Microsoft Azure).

A key objective of the VRE is to relieve the user from the burden of setting up the necessary IT-infrastructure of hardware and software for analysis. The VRE allows users to take advantage of the microservices inside the VRE in several ways: a) directly starting a single application inside a container, b) wrapping containers in an analysis script, or the promoted way c) integrating it into a new or existing scientific workflow (a chain of containerized tools that serially process input data). In order to launch a PhenoMeNal VRE, users can either take advantage of a PhenoMeNal client for KubeNow (https://github.com/phnmnl/cloud-deploy-kubenow) or use the PhenoMeNal web portal (https://portal.phenomenal-h2020.eu) that allows users to launch VREs on the largest public cloud providers.

The main workflow interface and engine is Galaxy (https://galaxyproject.org/), as shown in Demonstrators 2, 3 and 4, but users can also use Luigi (https://github.com/spotify/luigi) and Jupyter (http://jupyter.org/) (if more interactive scripting/scientific computing is required) as presented in Demonstrator 1. Both workflow systems have been adapted to run jobs on top of Kubernetes, and these contributions are now integrated and maintained in the core Galaxy and Luigi projects. Apart from the VRE, PhenoMeNal makes available a suite of well-tested and interoperable containers for metabolomics (see Supplementary Figure S1).

In addition to GUIs, workflows can be accessed, executed and tested via APIs with wft4galaxy (https://github.com/phnmnl/wft4galaxy). Here, the user can send data and queries to Galaxy from

the command line either manually or with macro-files and test the workflow and expected output for reproducibility. In order to access the API, an API key for the desired account must be generated from within Galaxy. Requiring the API key ensures only authorized access to the API. In this way, demonstrators 2, 3 and 4 can be invoked from the command line, can be tested for reproducible output and executed with different parameters and data.

Demonstrator 1:

The original analysis workflow by C. Ranninger et al. [1], whose HPLC-MS data was deposited in ISA- Tab format [46] to MetaboLights (MetaboLights ID: MTBLS233), was containerized using Docker. In the study, the authors reported the effect of scan parameters in the number of detected features in an Orbitrap-based mass spectrometer. More specifically, the authors argued and proved that having independent parameters in different mass ranges can greatly improve the ability of MS to detect metabolites. The experiment was based on 27 samples from lysates of a human kidney cell line, six pooled samples and eleven blank samples (total of 44 unique samples). Each of the 44 samples was run using mass spectrometry with three different settings for both positive and negative ionization:

1) a **full scan** by which the analytes were quantified in a single m/z range 50–1000 (for each sample one MS data file was produced).

2) **alternating scan** approach in which the mass range was split into m/z 50–200 and m/z 200–1000 utilizing the same parameters as the first approach (for each sample three MS data files were produced).

3) in this **separate** approach, each sample was run two times, one run for low (m/z 50–200) and one for high range masses (m/z 200–1000) using two different settings (for each sample two MS data files were produced).

In addition to these three approaches, an optimized and conclusional run was made, generating one file per sample, based on the result from the three tested approaches. Considering both positive and negative ionization, 12 files were produced for each of the 44 samples, resulting in a total of 528 MS files. In other words, these 528 MS files can be placed into 12 groups, each corresponding to one of approaches mentioned above (note that the alternating scan produced 3 and the separate approach produced 2 groups). Each of these groups included 44 samples.

Original preprocessing workflow

The preprocessing workflow was based on the OpenMS platform[16] by which the raw data files were first centroided using PeakPickerHiRes, the mass traces were quantified using FeatureFinderMetabo and were matched across the data files using FeatureLinkerUnlabeledQT. The parameters for these tools were individually set for positive and negative ionization. In addition, FileFilter was used to filter out certain features having retention time lower than 39 seconds and being present in at least six samples and finally the list of features was exported to text format using TextExporter. The downstream analysis was performed using R[47] on the KNIME workflow engine[48].

Preprocessing workflow using microservices and Luigi

OpenMS version 1.11.1 was containerized using the build instructions provided in the OpenMS Github repository for Linux (https://github.com/OpenMS/OpenMS/wiki/Building-OpenMS). Briefly, the OpenMS dependencies were first installed followed by building "contrib" libraries as

well as OpenMS core. The docker file is publicly available in "https://github.com/phnmnl/container-openms/blob/mtbls233/Dockerfile" and can be pulled and run from "docker-registry.phenomenal-h2020.eu/phnmnl/openms". The file OpenMS tasks that were using Luigi, ran in parallel over the 12 groups, each consisting of 44 MS files corresponding to one specific setting.

The first two tasks, peak picking and feature finding, were perfectly parallelizable, hence we executed them concurrently for all the samples using 2 gigabytes of RAM and 1 CPU. However, feature linking matches the features across corresponding runs, meaning that the samples with identical MS settings needed to be linked together and separate from the samples with different settings. Therefore, we performed 12 of these tasks in parallel. In order to maximize parallelization, each feature linker was run using 4 gigabytes of RAM and 2 CPUs. The linking tool produced a single file for each the 12 groups which were then processed independently by the last two tasks: file filter and text exporter both of which were performed using 2 gigabytes of RAM and 1 CPU. All compute nodes were connected to a shared clustered Gluster filesystem (https://www.gluster.org).

Weak Scaling Efficiency (WSE)

The Weak Scaling Efficiency (WSE) is a measure of how good a concurrent program scales computationally, when the same workload (or problem size) is assigned to each processor and additional processors are added to solve a larger problem. Given the full MTBLS233 dataset we ran the preprocessing analysis on 40 Luigi workers. Then for 1/4, 2/4, 3/4 of MTBLS233 we ran again on 10, 20 and 30 workers respectively (Figure S2). For each run we measured the processing time T_{10}, T_{20}, T_{30} and T_{40}, and we computed the WSE as:

$$WSE_n = \frac{T_{10}}{T_n} \; for \; n = 10, \; 20, \; 30, \; 40$$

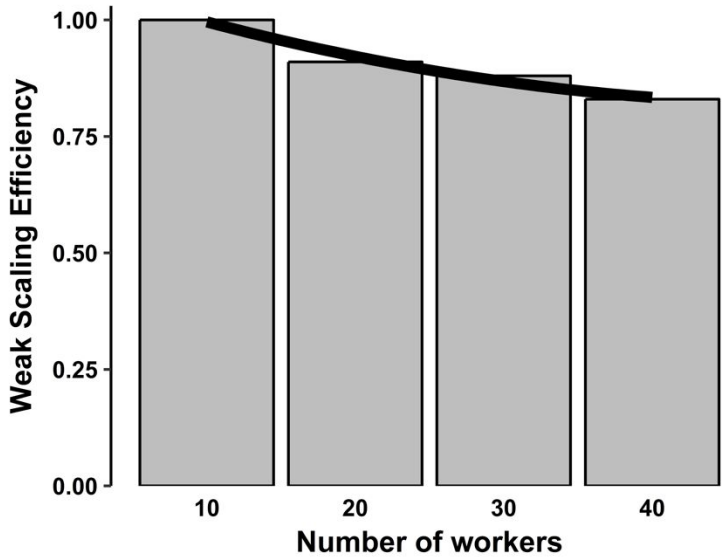

Figure S2. The weak scaling efficiency plot for Demonstrator 1. Given the full MTBLS233 dataset, the preprocessing was run on 40 Luigi workers. Then for 1/4, 2/4, 3/4 of MTBLS233, the analysis was run again on 10, 20 and 30 workers respectively. For each run, we measured the processing time T10, T20, T30 and T40, and we computed the WSEn = T10/Tn for n = 10, 20, 30, 40. The WSE plot shows that the loss in scalability levels off at 40 CPUs, where we achieved ~88% scaling efficiency. The running time for the full dataset (a total of 1092 tasks) on 40 workers was ~4 hours.

The downstream analysis, previously implemented in R as KNIME nodes, were extracted and moved to the interactive and responsive environment of Jupyter notebook using the provided R

kernel. The deployment and preprocessing as well as the downstream analysis can be accessed on https://github.com/phnmnl/MTBLS233-Jupyter

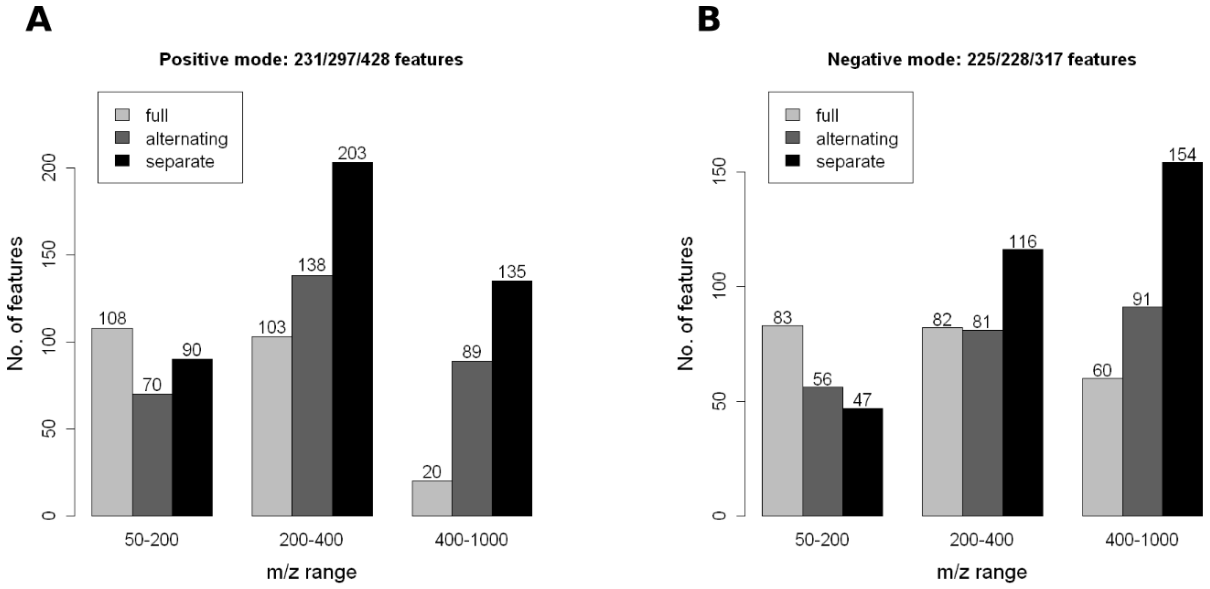

Figure S3. Result of downstream analysis on MTBLS233 dataset. A) Number of features quantified in positive ionization mode. B) Number of features quantified in negative ionization mode.

Demonstrator 2:

Mass spectrometry data processing is complex and normally requires a combination of many different tools e.g. to reduce the noise, perform quantification, alignment, matching, identification and statistics. This procedure becomes even more sophisticated considering that there is a large share of different algorithms and tools for each of the mentioned steps. This

indeed requires substantial technical knowledge of each tool including installation, format of input/output, and the parameters. The purpose of this demonstrator was to show the capability of the microservice architecture to eliminate the need for installation and converting data formats. Although, this demonstrator does not concern parameter optimization, a standard, well tested workflow can be reused for many type of instruments and experiments with minimal changes needed, given it has a standardized exchange data format.

The data used in this demonstrator includes 37 clinical CSF samples of thirteen relapsing-remitting multiple sclerosis (RRMS) and fourteen secondary progressive multiple sclerosis (SPMS) patients as well as ten controls (non-multiple sclerosis). 26 quality controls including 19 blank and seven dilution series samples were also added to the experiment. In addition, 8 MS/MS samples were included in the experiment for identification. The samples were processed and analyzed using the LC-MS workflow illustrated in Figure S4.

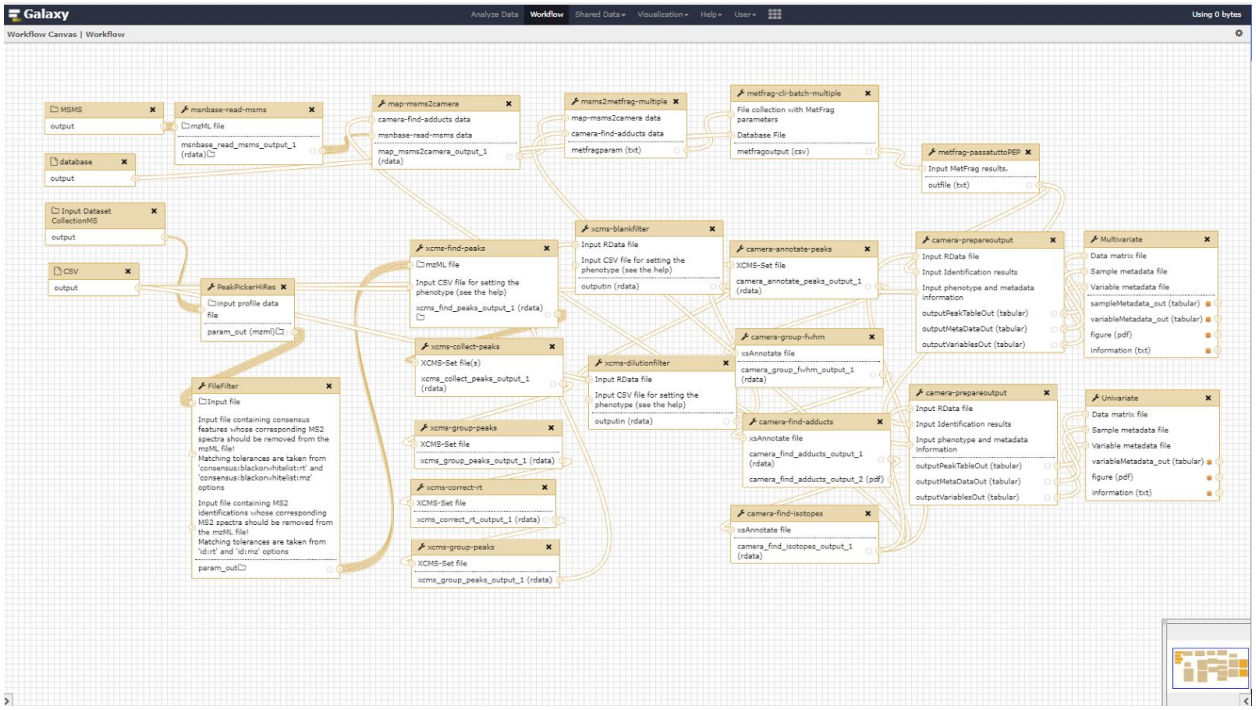

Figure S4. Overview of the mass spectrometry workflow in Galaxy used for demonstrator 2.

The LC-MS samples in vendor specific format (RAW) were first converted to the open access standard mzML using "msconvert" and then processed through the following workflow in Galaxy to perform the quantification:

Noise reduction and filtering (OpenMS)

The data was centroided using "PeakPickerHiRes" (default parameters) and intensity maps were cropped using "FileFilter" implemented in OpenMS Docker container (https://github.com/phnmnl/container-openms). All the OpenMS nodes in Galaxy were generated using CTDConverter (https://github.com/WorkflowConversion/CTDConverter).

Quantification, alignment and matching (XCMS)

The quantification was performed using XCMS container (https://github.com/phnmnl/container-xcms) such that the mass traces were identified and quantified with "findPeaks". In addition to the filtered mzML files, this node accepts a CSV (comma-separated value) file by which users can add phenotypic (class) information for each of the samples. Afterwards, the retention time drifts between samples were adjusted using "retcor" and the mass traces were grouped across the samples using "group" function in XCMS.

Further filtering of biological non-relevant signals (R)

Using the same approach as "PeakML.BlankFilter" and "PeakML.DilutionTrendFilter" from mzMatch[49], two containers were created to filter out biologically non-relevant and spurious signals based on the blank and dilution series samples. More specifically, "blankfilter" accepts the output of grouping (XCMS) and based on the user defined parameters, signals with higher abundance in the non-biological samples (blanks) will be removed from the dataset. Dilution series filtering (dilutionfilter) on the other hand, removes signals if they do not correlate with a user defined series (i.e. a series in which the amount of sample analyzed was sequentially increased). Both of these nodes can accept a table showing which files the algorithm must use as blank and biological samples (https://github.com/phnmnl/container-xcms).

Annotation of signals (CAMERA)

Annotation of the peaks was performed using CAMERA container (https://github.com/phnmnl/container-camera) by applying the following functions: "groupFWHM", "findAdducts" and "findIsotopes". This part of the workflow attempts to group the metabolites based on their isotopes as well as adduct information.

Identification (MetFrag)

To perform the metabolite identification, the tandem spectra from the MS2 samples in mzML format were extracted using MSnbase (https://github.com/phnmnl/container-MSnbase) container ("metfrag-read-msms") and parent ion m/z were converted to neutral mass based on the adducts and isotope information from CAMERA ("map-msms2camera" and "msms2metfrag"). The resulting neutral mass and the tandem information were used in MetFrag (https://github.com/phnmnl/container-metfrag-cli-batch) container ("metfrag-cli-multiple") to perform *in silico* fragmentation and scoring of the metabolite candidates retrieved from the human metabolome database (HMDB version 2017-07)[50]. The MetFrag scores were converted to q-values using Passatutto (https://github.com/phnmnl/container-passatutto) software ("metfrag-passatutoPEP"). Various parameters can be set in this part of the workflow including type of adducts, desired m/z and RT deviation as well as several options regarding MetFrag identification and score transformation.

Statistics (Workflow4Metabolomics)

The result of identification and quantification were used in "camera-prepareoutput". This node accepts the quantification (CAMERA results), identification (MetFrag results) and sample metadata (CSV file) and prepares the output for "Multivariate" and "Univariate" containers (https://github.com/phnmnl/container-multivariate and https://github.com/phnmnl/container-univariate) from Workflow4Metabolomics to perform a PLS-DA. In addition, analysis of variance (ANOVA) was performed to select the metabolites with altered levels. We used specific cutoff of 0.05 both on f-statistics and t-statistics (comparing RRMS and SPMS to controls) to select the metabolites. Finally, only the metabolites were limited to those that were identified and quantified in human based on HMDB information.

Demonstrator 3:

The purpose of this workflow was to demonstrate application of 1D NMR data analysis using microservices. The workflow illustrated in Figure S5 was implemented in Galaxy and applied to the MTBLS1 data set imported from the MetaboLights database.

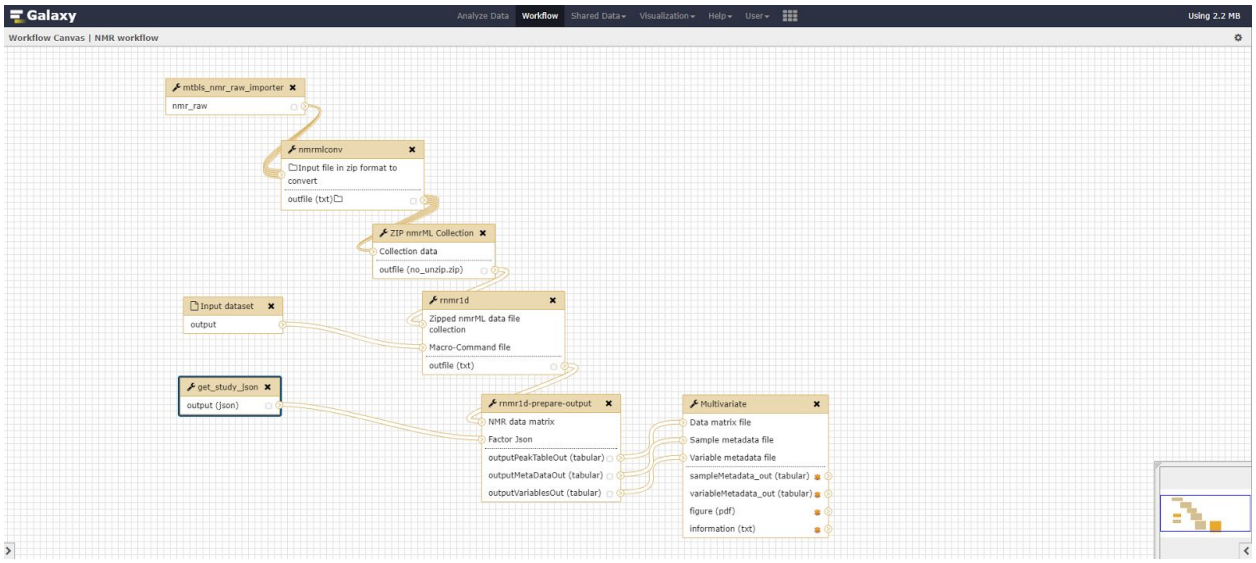

Figure S5. 1D NMR data analysis workflow, exemplifying secondary data usage through import from a dedicated metabolomics repository.

The NMR vendor data files are first downloaded using "mtbls_nmr_raw_import" and converted to nmrML format using "nmrmlconv". The resulting open access data are then zipped ("ZIP nmrML Collection") and fed into "rnmr1d" (https://github.com/phnmnl/container-rnmr1d). This specific node requires a macro settings file (which can be obtained from https://github.com/PayamEmami/galaxyWorkflows). Afterwards, the container performs the

fourier transformation (ftt), zero-filling, line broadening, phase-correction and baseline correction, resulting to a peak matrix. At the same time "get_study_json" must be used to extract all metadata information from a specified MetaboLights study to a JSON representation of the ISA formatted metadata (https://github.com/phnmnl/container-mtblisa). The peak matrix and the JSON file are used in "rnmr1d-prepare-output" which prepares the output consisting of all available metadata information for "Multivariate" or "Univariate" nodes for performing statistics on factor of interest.

Demonstrator 4:

In this GC-MS-based metabolomics study, HUVEC (Human Umbilical Vein Endothelial Cell) cells were incubated for 40 hours in the presence of [1,2-$^{13}C_2$]glucose and ^{13}C propagation to glycogen, RNA ribose and lactate was used to constrain the flux values through glycolysis and pentose -phosphate pathways. The workflow illustrated in Figure S6 was implemented in Galaxy. The fluxomics data analysis workflow corresponding to Figure 6 is available in the PhenoMeNal VRE as a shared workflow. HUVEC cell data are available on the MTBLS412 data set that can be imported from the MetaboLights database. ^{13}C Metabolic Flux Analysis (13C MFA), for estimation of metabolic reaction-fluxes through the integration of ^{13}C metabolomics data, is achieved by using Iso2Flux to maximize the consistency between flux-balanced model predictions and experimental mass spectrometry measurements. The flux values (numerical values associated with the metabolic scheme in Figure 6) refers to the output of lactate, as lactate was the only fixed value set to 1. The generated flux map showed that most of the secreted lactate derived from the consumed glucose, with a comparatively low pass through pentose-phosphate pathway.

The corresponding tools are:

- RaMID (https://github.com/seliv55/RaMID): An R-based tool for reading and extracting the necessary information from mass spectrometer generated netCDF files. It extracts mass spectra of the metabolites of interest corresponding to the m/z and retention time indicated in an input file, corrects baseline, and saves the resulting data.

- MIDcor (https://github.com/seliv55/midcor) an R-based tool that takes the output of RaMID and corrects the evaluated mass spectra for natural isotopes abundance and possible peak overlapping as described in[51] and computes isotopologue fractions.

- Iso2Flux (https://github.com/cfoguet/iso2flux): a python-based ^{13}C MFA tool. Iso2Flux uses constraint-based modelling and the EMU Elementary Metabolite Unit framework [52] to compute steady-state distribution of ^{13}C label as function of steady-state flux distribution. Fitting to experimental data provides the distribution of underlying metabolic fluxes. Iso2Flux, takes the output of MIDcor, an SBML file defining the metabolic network and a CSV file defining the ^{13}C propagation rules and generates the following outputs: a) the simulated fluxes that are most consistent with experimental ^{13}C data, b) the simulated label obtained with such fluxes, c) lower and upper limits of the 95% confidence intervals for fluxes e) an SBML model constrained with the results of the ^{13}C analysis.

- Escher Fluxomics (https://github.com/escher/escher-demo): a web-based tool for visualisation of reaction-fluxes in relation to pathways.

Containers have been compiled for these tools and galaxy wrappers have been developed for them. These tools, together with the ProteoWizard container, have been used for the creation of the workflow in Galaxy (figure S6).

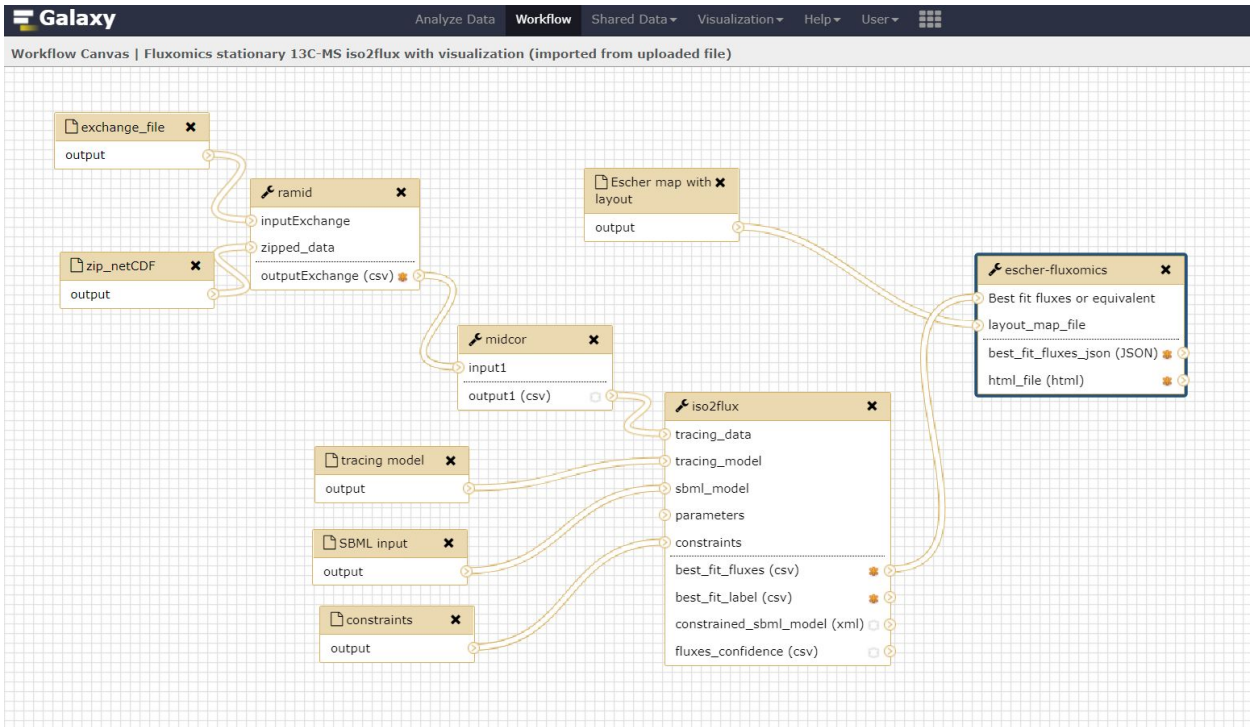

Figure S6: Fluxomics data analysis workflow from ^{13}C tracer-based data written by a mass spectrometer in netCDF files. The workflow connects the tools RaMID, MIDcor, Iso2flux and Escher Fluxomics.

Acknowledgements

This research was supported by The European Commission's Horizon 2020 programme funded under grant agreement number 654241 (PhenoMeNal), The Swedish Research Council FORMAS, Uppsala Berzelii Technology Centre for Neurodiagnostics and Åke Wiberg Foundation. We kindly acknowledge contributions to cloud resources by SNIC Science Cloud,

Embassy Cloud, and CityCloud. The funders had no role in study design, data collection and analysis, decision to publish, or preparation of the manuscript.

Author contributions

KK, MAC, MC, PEK, SH contributed to Demonstrator 1. CR, KK, KP, PEK, SH, SN contributed to Demonstrator 2. KK designed the study in Demonstrator 2. JB performed collection of samples and characterization of the multiple sclerosis cohort. SH performed mass spectrometry experiment in Demonstrator 2. DS, KP, PEK, PM, RMS, contributed to Demonstrator 3. AGB, CF, DJ, MCA, MVV, PDA, PM, PRS, SAS, TH and VS contributed to Demonstrator 4. GZ, LP, PEK and PM contributed to development of Galaxy in Kubernetes. AL and MC contributed to the development of Luigi in Kubernetes. AL, MAC, MC and NS developed KubeNow. PM contributed to Galaxy-Kubernetes. EAT and PR contributed to containerizing of Workflow4Metabolomics tools. AGB, DJ, PRS and SAS contributed to ISA-API. DJ, EAT, KP, MVV, NS, OS, PEK, PM, PR, PRS, RMS, RR and SB were involved in testing the containers and the VRE. PM, SIH and KH were involved in development and maintenance of the portal. MVV, PM and RMS contributed to the release. NK coordinated the PhenoMeNal project. CS conceived and managed PhenoMeNal project. OS conceived and coordinated the project and e-infrastructure. All authors contributed to manuscript writing.

Table S1. Supplement: List of nodes in Galaxy

Module name	Container name
BATMAN	container-batman
camera-annotate-peaks	container-camera
camera-find-adducts	container-camera
camera-find-isotopes	container-camera
camera-group-corr	container-camerar
camera-group-fwhm	container-camera
camera-prepareoutput	container-camera
cameraToFeatureXML	container-camera
featureXMLToCAMERA	container-camera
featureXMLToXcms	container-camera
Zip Data Collection	container-camera
escher-fluxomics	container-escher-fluxomics
isajson_validator	container-isajson-validator
isatab2json	container-isatab2json
isatab_validator	container-isatab-validator
json2isatab	container-json2isatab
Metabolite_ID_Converter	container-metaboliteidconverter
data_files	container-mtblisa
factors_summary	container-mtblisa
get_study	container-mtblisa
get_study_json	container-mtblisa
study_factor_values	container-mtblisa
study_factors	container-mtblisa

MetaboLights Factors Viz	container-mtbls-factors-viz
mw2isa	container-mw2isa
AdditiveSeries	container-openms
BaselineFilter	container-openms
CompNovo	container-openms
CompNovoCID	container-openms
ConsensusID	container-openms
ConsensusMapNormalizer	container-openms
Decharger	container-openms
DTAExtractor	container-openms
EICExtractor	container-openms
ExternalCalibration	container-openms
FalseDiscoveryRate	container-openms
FeatureFinderCentroided	container-openms
FeatureFinderIdentification	container-openms
FeatureFinderIsotopeWavelet	container-openms
FeatureFinderMetabo	container-openms
FeatureFinderMRM	container-openms
FeatureFinderMultiplex	container-openms
FeatureLinkerLabeled	container-openms
FeatureLinkerUnlabeled	container-openms
FeatureLinkerUnlabeledQT	container-openms
FidoAdapter	container-openms
FileConverter	container-openms
FileFilter	container-openms
FileInfo	container-openms

FileMerger	container-openms
HighResPrecursorMassCorrector	container-openms
IDConflictResolver	container-openms
IDFileConverter	container-openms
IDFilter	container-openms
IDMapper	container-openms
IDMerger	container-openms
IDPosteriorErrorProbability	container-openms
IDRipper	container-openms
IDRTCalibration	container-openms
InclusionExclusionListCreator	container-openms
InspectAdapter	container-openms
InternalCalibration	container-openms
IsobaricAnalyzer	container-openms
ITRAQAnalyzer	container-openms
LuciphorAdapter	container-openms
MapAlignerIdentification	container-openms
MapAlignerPoseClustering	container-openms
MapAlignerSpectrum	container-openms
MapNormalizer	container-openms
MapRTTransformer	container-openms
MapStatistics	container-openms
MascotAdapter	container-openms
MascotAdapterOnline	container-openms
MassTraceExtractor	container-openms
MRMMapper	container-openms

MSGFPlusAdapter	container-openms
MyriMatchAdapter	container-openms
MzTabExporter	container-openms
NoiseFilterGaussian	container-openms
NoiseFilterSGolay	container-openms
OMSSAAdapter	container-openms
OpenSwathAnalyzer	container-openms
OpenSwathAssayGenerator	container-openms
OpenSwathChromatogramExtractor	container-openms
OpenSwathConfidenceScoring	container-openms
OpenSwathDecoyGenerator	container-openms
OpenSwathFeatureXMLToTSV	container-openms
OpenSwathRTNormalizer	container-openms
PeakPickerHiRes	container-openms
PeakPickerWavelet	container-openms
PepNovoAdapter	container-openms
PeptideIndexer	container-openms
PhosphoScoring	container-openms
PrecursorIonSelector	container-openms
PrecursorMassCorrector	container-openms
ProteinInference	container-openms
ProteinQuantifier	container-openms
ProteinResolver	container-openms
PTModel	container-openms
PTPredict	container-openms
RTModel	container-openms

RTPredict	container-openms
SeedListGenerator	container-openms
SpecLibSearcher	container-openms
SpectraFilterBernNorm	container-openms
SpectraFilterMarkerMower	container-openms
SpectraFilterNLargest	container-openms
SpectraFilterNormalizer	container-openms
SpectraFilterParentPeakMower	container-openms
SpectraFilterScaler	container-openms
SpectraFilterSqrtMower	container-openms
SpectraFilterThresholdMower	container-openms
SpectraFilterWindowMower	container-openms
SpectraMerger	container-openms
TextExporter	container-openms
TMTAnalyzer	container-openms
TOFCalibration	container-openms
XTandemAdapter	container-openms
metfrag-passatuttoPEP	container-passatutto
iso2flux	container-iso2flux
isodyn	container-isodyn
ms-vfetc	container-ms-vfetc
metabomatching	container-metabomatching
metfrag-cli-batch	container-metfrag-cli-batch
metfrag-cli-multiple	container-metfrag-cli-batch
metfragcli	container-metfrag-cli

metfrag-vis	container-metfrag-vis
midcor	container-midcor
msconvert2	container-msconvert2
map-msms2camera	container-msnbase
metfrag-aggregatemetfrag	container-msnbase
msms2metfrag	container-msnbase
msnbase-read-msms	container-msnbase
MetaboLights Downloader	container-mtbls-downloader
MetaboLights Labs Uploader	container-mtbls-labs-uploader
mzml2isa	container-mzml2isa
mtbls2nmrglue	container-nmrglue
nmrml2isa	container-nmrml2isa
mtbls_nmr_raw_importer	container-nmrmlconv
nmrmlconv	container-nmrmlconv
ZIP nmrML Collection	container-nmrmlconv
PAPY	container-papy
ramid	container-ramid
rnmr1d-prepare-output	container-rnmr1d
rnmr1d-stacked-plot	container-rnmr1d
rnmr1d	container-rnmr1d
soap_opls	container-soap-nmr
soap_process	container-soap-nmr
LC/MS matching	container-w4m-lcmsmatching
MetaboLights downloader	container-w4m-mtbls-dwnld
Multivariate	container-w4m-multivariate

Univariate	container-w4m-univariate
save_chromatogram	container-xcms
show_chromatogram	container-xcms
xcms-blankfilter	container-xcms
xcms-collect-peaks	container-xcms
xcms-correct-rt	container-xcms
xcms-dilutionfilter	container-xcms
xcms-fill-peaks	container-xcms
xcms-find-peaks	container-xcms
xcms-group-peaks	container-xcms
xcmssplit	container-xcms

References

1. Nicholson, J. K. & Wilson, I. D. Opinion: understanding 'global' systems biology: metabonomics and the continuum of metabolism. *Nat. Rev. Drug Discov.* **2,** 668–676 (2003).
2. Manrai, A. K. *et al.* Informatics and Data Analytics to Support Exposome-Based Discovery for Public Health. *Annu. Rev. Public Health* **38,** 279–294 (2017).
3. Montenegro-Burke, J. R. *et al.* Data Streaming for Metabolomics: Accelerating Data Processing and Analysis from Days to Minutes. *Anal. Chem.* **89,** 1254–1259 (2017).
4. Liew, C. S. *et al.* Scientific Workflows. *ACM Computing Surveys* **49,** 1–39 (2016).
5. Suplatov, D., Popova, N., Zhumatiy, S., Voevodin, V. & Švedas, V. Parallel workflow manager for non-parallel bioinformatic applications to solve large-scale biological problems on a supercomputer. *J. Bioinform. Comput. Biol.* **14,** 1641008 (2016).
6. Warth, B. *et al.* Metabolizing Data in the Cloud. *Trends Biotechnol.* **35,** 481–483 (2017).
7. Silver, A. Software simplified. *Nature* **546,** 173–174 (2017).
8. Newman, S. *Building Microservices*. ('O'Reilly Media, Inc.', 2015).
9. da Veiga Leprevost, F. *et al.* BioContainers: an open-source and community-driven framework for software standardization. *Bioinformatics* **33,** 2580–2582 (2017).
10. Merkel, D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. *Linux J.* **2014,** (2014).
11. Goecks, J., Nekrutenko, A., Taylor, J. & Galaxy Team. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences. *Genome Biol.* **11,** R86 (2010).
12. Leipzig, J. A review of bioinformatic pipeline frameworks. *Brief. Bioinform.* **18,** 530–536 (2017).
13. Kluyver, T. *et al.* Jupyter Notebooks ? a publishing format for reproducible computational workflows. (2016).

14. Haug, K. *et al.* MetaboLights--an open-access general-purpose repository for metabolomics studies and associated meta-data. *Nucleic Acids Res.* **41,** D781–6 (2013).

15. Ranninger, C. *et al.* Improving global feature detectabilities through scan range splitting for untargeted metabolomics by high-performance liquid chromatography-Orbitrap mass spectrometry. *Anal. Chim. Acta* **930,** 13–22 (2016).

16. Sturm, M. *et al.* OpenMS - an open-source software framework for mass spectrometry. *BMC Bioinformatics* **9,** 163 (2008).

17. Salek, R. M. *et al.* A metabolomic comparison of urinary changes in type 2 diabetes in mouse, rat, and human. *Physiol. Genomics* **29,** 99–108 (2007).

18. Buescher, J. M. *et al.* A roadmap for interpreting (13)C metabolite labeling patterns from cells. *Curr. Opin. Biotechnol.* **34,** 189–201 (2015).

19. Niedenführ, S., Wiechert, W. & Nöh, K. How to measure metabolic fluxes: a taxonomic guide for 13 C fluxomics. *Curr. Opin. Biotechnol.* **34,** 82–90 (2015).

20. King, Z. A. *et al.* Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. *PLoS Comput. Biol.* **11,** e1004321 (2015).

21. Foster, I. Service-oriented science. *Science* **308,** 814–817 (2005).

22. Marx, V. Biology: The big challenges of big data. *Nature* **498,** 255–260 (2013).

23. Schadt, E. E., Linderman, M. D., Sorenson, J., Lee, L. & Nolan, G. P. Computational solutions to large-scale data management and analysis. *Nat. Rev. Genet.* **11,** 647–657 (2010).

24. Berger, B., Peng, J. & Singh, M. Computational solutions for omics data. *Nat. Rev. Genet.* **14,** 333–346 (2013).

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

26. Wilkinson, M. D. *et al.* The FAIR Guiding Principles for scientific data management and

stewardship. *Sci Data* **3,** 160018 (2016).

27. Lovelace, M. D. *et al.* Current Evidence for a Role of the Kynurenine Pathway of Tryptophan Metabolism in Multiple Sclerosis. *Front. Immunol.* **7,** 246 (2016).
28. Lim, C. K. *et al.* Kynurenine pathway metabolomics predicts and provides mechanistic insight into multiple sclerosis progression. *Sci. Rep.* **7,** 41473 (2017).
29. Amirkhani, A. *et al.* Interferon-beta affects the tryptophan metabolism in multiple sclerosis patients. *Eur. J. Neurol.* **12,** 625–631 (2005).
30. Centonze, D. *et al.* The endocannabinoid system is dysregulated in multiple sclerosis and in experimental autoimmune encephalomyelitis. *Brain* **130,** 2543–2553 (2007).
31. Zamberletti, E., Rubino, T. & Parolaro, D. The Endocannabinoid System and Schizophrenia: Integration of Evidence. *Curr. Pharm. Des.* **18,** 4980–4990 (2012).
32. Baker, D. & Pryce, G. The endocannabinoid system and multiple sclerosis. *Curr. Pharm. Des.* **14,** 2326–2336 (2008).
33. Schober, D. *et al.* nmrML: a community supported open data standard for the description, storage, and exchange of NMR data. *Anal. Chem.* (2017). doi:10.1021/acs.analchem.7b02795
34. Rocca-Serra, P. *et al.* Data standards can boost metabolomics research, and if there is a will, there is a way. *Metabolomics* **12,** 14 (2016).
35. Sansone, S.-A. *et al.* Toward interoperable bioscience data. *Nat. Genet.* **44,** 121–126 (2012).
36. Iyer, N. V. *et al.* Cellular and developmental control of O2 homeostasis by hypoxia-inducible factor 1alpha. *Genes Dev.* **12,** 149–162 (1998).
37. Eelen, G., de Zeeuw, P., Simons, M. & Carmeliet, P. Endothelial cell metabolism in normal and diseased vasculature. *Circ. Res.* **116,** 1231–1244 (2015).
38. Polet, F. & Feron, O. Endothelial cell metabolism and tumour angiogenesis: glucose and glutamine as essential fuels and lactate as the driving force. *J. Intern. Med.* **273,** 156–165 (2013).

39. Grüning, B. A. *et al.* Jupyter and Galaxy: Easing entry barriers into complex data analyses for biomedical researchers. *PLoS Comput. Biol.* **13,** e1005425 (2017).

40. Martens, L. *et al.* mzML--a community standard for mass spectrometry data. *Mol. Cell. Proteomics* **10,** R110.000133 (2011).

41. Smith, C. A., Want, E. J., O'Maille, G., Abagyan, R. & Siuzdak, G. XCMS: Processing Mass Spectrometry Data for Metabolite Profiling Using Nonlinear Peak Alignment, Matching, and Identification. *Anal. Chem.* **78,** 779–787 (2006).

42. Kuhl, C., Tautenhahn, R., Böttcher, C., Larson, T. R. & Neumann, S. CAMERA: an integrated strategy for compound spectra extraction and annotation of liquid chromatography/mass spectrometry data sets. *Anal. Chem.* **84,** 283–289 (2012).

43. Giacomoni, F. *et al.* Workflow4Metabolomics: a collaborative research infrastructure for computational metabolomics. *Bioinformatics* **31,** 1493–1495 (2015).

44. Thévenot, E. A., Roux, A., Xu, Y., Ezan, E. & Junot, C. Analysis of the Human Adult Urinary Metabolome Variations with Age, Body Mass Index, and Gender by Implementing a Comprehensive Workflow for Univariate and OPLS Statistical Analyses. *J. Proteome Res.* **14,** 3322–3335 (2015).

45. Jacob, D., Deborde, C., Lefebvre, M., Maucourt, M. & Moing, A. NMRProcFlow: a graphical and interactive tool dedicated to 1D spectra processing for NMR-based metabolomics. *Metabolomics* **13,** 36 (2017).

46. Rocca-Serra, P. *et al.* ISA software suite: supporting standards-compliant experimental annotation and enabling curation at the community level. *Bioinformatics* **26,** 2354–2356 (2010).

47. R Development Core Team. R: A Language and Environment for Statistical Computing. (2008).

48. Berthold, M. R. *et al.* KNIME - the Konstanz information miner. *ACM SIGKDD Explorations Newsletter* **11,** 26 (2009).

49. Scheltema, R. A., Jankevics, A., Jansen, R. C., Swertz, M. A. & Breitling, R. PeakML/mzMatch: a

file format, Java library, R library, and tool-chain for mass spectrometry data analysis. *Anal. Chem.* **83,** 2786–2793 (2011).

50. Wishart, D. S. *et al.* HMDB: the Human Metabolome Database. *Nucleic Acids Res.* **35,** D521–D526 (2007).

51. Selivanov, V. A. *et al.* MIDcor, an R-program for deciphering mass interferences in mass spectra of metabolites enriched in stable isotopes. *BMC Bioinformatics* **18,** 88 (2017).

52. Antoniewicz, M. R., Kelleher, J. K. & Stephanopoulos, G. Elementary metabolite units (EMU): a novel framework for modeling isotopic distributions. *Metab. Eng.* **9,** 68–86 (2007).