ARPHA Preprints

PENSOFT

Software Description

Author-formatted document posted on 05/12/2023

Published in a RIO article collection by decision of the collection editors.

DOI: https://doi.org/10.3897/arphapreprints.e116669

BatchConvert: A command-line tool for parallelised conversion of image collections into the standard bioimage file formats OME-TIFF and OME-Zarr

Bugra Özdemir, Aastha Mathur, Johanna Bischof, John E. Eriksson, Jean-Karim Hériché, Josh Moore, Christian Tischer, Antje Keppler

ARPHA Preprints *Author-formatted document posted on 05/12/2023*. DOI: https://doi.org/10.3897/arphapreprints.e116669

BatchConvert: A command-line tool for parallelised conversion of image collections into the standard bioimage file formats OME-TIFF and OME-Zarr.

Authors: Özdemir, B., Mathur, A., Bischof, J.,Eriksson, J.,Hériché, J., Moore, J., Tischer, C., Keppler, A.

Abstract

File formats incompatibility has become a major obstacle in biological imaging, complicating downstream processes such as image processing and analysis. One way to address this challenge is to convert the acquired image data into standard image file formats. Here we introduce BatchConvert, a command line tool for parallelised conversion of image collections into OME-TIFF or OME-Zarr using the workflow management system Nextflow[1]. BatchConvert offers functionalities such as remote input-output support, optional execution on Slurm clusters and pattern-based filtering of input files. Conversion can be coupled to image concatenation, allowing selected images to be merged along specified dimensions. Support for remote locations includes an option to submit the output data to S3-compatible object stores or public archives such as BioImage Archive[2]. Overall, BatchConvert is a flexible tool for researchers who are routinely managing and analysing large multidimensional image data that is either locally or remotely stored.

Keywords: Bioimage, NGFF, OME-Zarr, OME-TIFF, workflow, Nextflow, conversion, file-format

1. Background

We are in an era of biological imaging that faces rapid proliferation of imaging technologies and modalities. Concurrent with the technological advances in the field, imaging data is increasing in size and complexity, creating major data storage and management issues. These issues are further complicated by the ever-expanding plethora of proprietary file formats (PFFs) that hamper data analysis, sharing and reuse within the biological imaging domain. This challenge has driven the community to work on standard file formats for biological imaging data.

To this end, significant efforts have been made by the Open Microscopy Environment (OME) consortium to develop a data model for biological imaging and implement it in an XML-based file format called OME-XML (Goldberg et al., 2005), capable of storing extensive structured metadata associated with biological imaging experiments. TIFF (Tagged Image File Format), a freely available and widely supported format, has been adopted to retain the pixel / voxel data for images following the OME model. When TIFF is used to store the binary data, the OME-XML metadata is typically embedded in the TIFF file header under the “Image Description Tag”, resulting in what is called OME-TIFF (Linkert et al., 2010). OME-TIFF has been a popular file format for exchanging image data between different software packages and submitting image data to centralised public repositories such as Image Data Resource (IDR) (Williams et al., 2017).

The adoption of OME-TIFF among the imaging community has also been facilitated by the powerful Bio-Formats library (Linkert et al., 2010). The Bio-Formats library is capable of reading from over 150 PFFs and writing to several common formats, including OME-XML and OME-TIFF. In addition to its use for stable conversion of image data to open formats, Bio-Formats has also become a part of many existing software packages for on-the-fly translation of the image data. Due to these capabilities, Bio-Formats has contributed substantially to the achievement of standards for data accessibility, sharing

[1] https://www.nextflow.io/
[2] https://www.ebi.ac.uk/bioimage-archive/

and interoperability in biological imaging.

Nevertheless, the plane-based data access mechanism offered by the TIFF format is inefficient with large, multidimensional data, such as data arising from high-content screening, lightsheet microscopy or volumeEM, where the aim is often to access a subset of the data at a time. For instance, access to arbitrary oblique slices or sub-volumes usually requires the expensive process of reading all corresponding planes from TIFF files. The aforementioned problems of increasing data size and complexity in the field have motivated researchers to seek formats that offer more flexible data storage and access mechanisms. While data in TIFF files is stored as a sequence of image planes, file formats such as Zarr (Miles et al., 2023) allow arbitrary chunking of the data along all dimensions. Thus, reading an arbitrary subset of voxels, e.g. an oblique plane or a sub-volume, can be achieved much more efficiently. The degree of efficiency can be tuned by choosing chunk sizes to optimise certain access patterns (Nguyen et al., 2023). The multi-file chunking strategy also supports direct parallel reading and writing of the data. This offers significant performance gains in general and is an essential feature when the data is remotely processed Moore et al., 2021, Moore et al., 2023).

To exploit the advantages offered by a modern binary format like Zarr in bioimaging, a “Next Generation File Format (NGFF)” specification is being developed, whose reference implementation is the OME-Zarr format, supported through efforts from multiple institutes and individuals across many communities. The current OME-Zarr format has specifications for storing multiresolution (pyramidal) images, their core metadata as well as derived data such as label images and image analysis results in a single hierarchical layout. In particular, the specifications allow for storing multiresolution (pyramidal) images along with spatial metadata allowing to encode physical voxel sizes. The label images derived from image segmentation can be stored together with certain display settings such as RGB and opacity values per label. The specifications also allow for storing image analysis results in the form of region properties (such as area, average intensity, etc) per label.

The combination of the performance gains it offers and the extensive, community-supported NGFF specifications have led OME-Zarr to gain wide adoption in the imaging community, which is reflected by the increasing number of tools in the form of libraries, independent software packages, plugins and web services being developed to serve different purposes (Moore et al., 2023).

Similar to OME-TIFF, OME-Zarr is typically generated from a PFF by performing a conversion process. A straightforward and safe strategy would be to use one of the dedicated conversion tools (Moore et al., 2023). One of these, namely bioformats2raw, is a command-line tool that is actively maintained to comply with the latest NGFF specifications. Bioformats2raw has all the readers that the Bio-Formats library offers, plus additional ones, and hence overcomes the major challenge in the format conversion - inability to capture essential image metadata from many of the PFFs. Bioformats2raw also provides extensive control over the conversion process, enabling the user to fine-tune the properties of the output OME-Zarr. Another conversion tool is the NGFF-Converter, which is a graphical user interface supporting scheduled conversions of multiple PFFs into OME-TIFF or OME-Zarr based on user’s selection.

While dedicated tools for data conversion from PFFs into OME-Zarr and OME-TIFF exist, tools focusing on the parallelisation of the conversion process over batches of remotely located images are scarce. In the context of NGFFs, one example is the Distributed-OMEZarrCreator (Weisbart & Cimini, 2022), which runs docker containers with bioformats2raw in Amazon Web Services infrastructure.

Strong community efforts are required for developing and maintaining data formats, standards and tools to help users adapt their data management and analysis practices with the latest technologies like cloud computing. In this context, the EOSC Future[3] project has been focusing on creating a centralised domain-agnostic resource for integrating research outputs like data, software and other digital assets, and incorporating them into the European Open Science Cloud[4] (EOSC), where all European research data can be discovered. Being part of the EOSC Future test science project “Open Imaging Data Sharing in EOSC / COVID-19 as Demonstrator”, one of our goals has been the development of a seamless data submission workflow to enable and encourage users to deposit their data in the centralised public repositories like BioImage Archive (Hartley et al., 2022, 2023), or Image Data

[3] https://eoscfuture.eu/
[4] https://www.eosc.eu/

Resource (Williams et al., 2017) (IDR[5] - repository for reference image datasets), in the cloud optimised OME-Zarr format.

In this work, we showcase BatchConvert, a Nextflow-based, simple-to-use command line tool that enables asynchronous parallel conversion of batches of image files into either of the standard formats OME-TIFF and OME-Zarr, and the transfer of the converted datasets to remote storage.

2. Method

2.1. Design

BatchConvert relies on the workflow management system Nextflow (Di Tommaso et al., 2017) for parallelisation of the data conversion and transfer as well as for automatic dependency management. Each conversion command internally triggers a Nextflow workflow that takes the input files from the input storage, which can be local or remote, converts them into a standard bioimage file format using one of the two conversion modes (see section 3.4.3 "**Execution Modes**" for details), and transfers the converted files to the output location, which can itself be remote or local (Fig-1). The user can parameterise the workflow either by directly specifying the arguments of the conversion command, or by modifying the default parameter values through a separate (optionally interactive) configuration step (see section 3.3 **"Configuration"** for details).

Conversion into OME-TIFF and OME-Zarr is achieved using the dedicated packages "bftools" and "bioformats2raw", respectively. BatchConvert supports remote input-output options for public and private s3 object storage and BioImage Archive, using the high-performance packages "go-mc" and "aspera-cli".

The current version of the tool has been developed for Unix systems and tested in Linux. Windows support will be provided in the upcoming versions.

[5] https://idr.openmicroscopy.org/

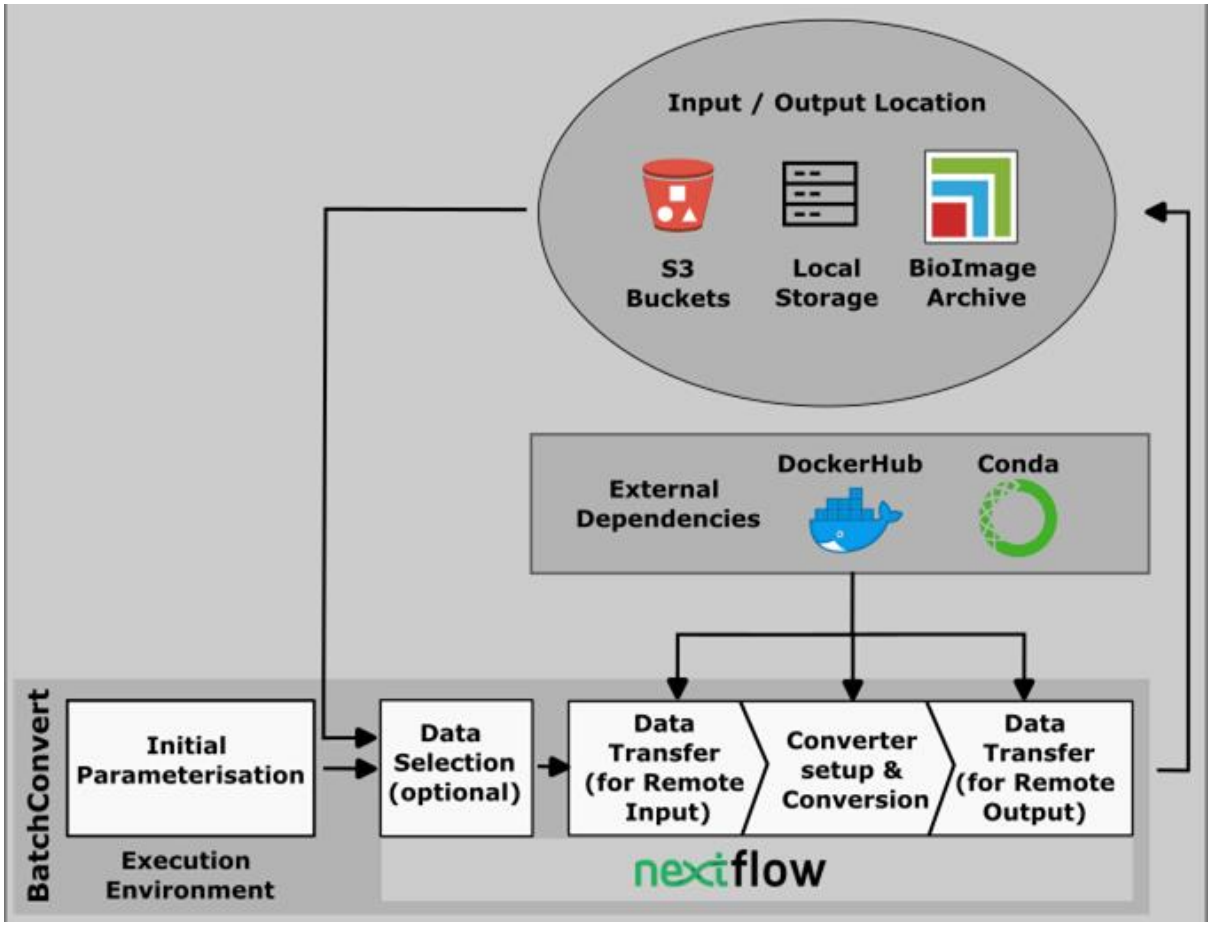

Figure-1. The design of BatchConvert: Initial parameterisation involves either passing arguments to the conversion command or using a separate interactive configuration step. Then the parameters are consumed by Nextflow to generate and execute the workflow. An optional early step is to select the input data by matching the filenames with certain patterns. If the input data is remotely located, BatchConvert will transfer it to the execution environment using the respective data transfer tool. The data that is available in the execution environment is converted into a standard format through either of the two conversion modes supported. If the output is specified as a remote storage, BatchConvert will transfer the converted data to this remote location using the respective client. The external dependencies for conversion and the data transfer are acquired and run either via a conda environment or docker / singularity container as specified by the `--profile` parameter that the user selects.

2.2. Dependencies and Installation

BatchConvert requires that the active shell is Bash and has been mainly tested with Bash 5.0.17(1)-release.

BatchConvert can be either installed using the package manager conda (recommended) or manually from the source.

3.2.1. Installation via conda

This option requires the package manager Anaconda (or its lighter version Miniconda) to be available (Anaconda Software Distribution, 2016). Anaconda / Miniconda can be downloaded and installed following the guidelines here:

https://docs.conda.io/projects/conda/en/latest/user-guide/install/index.html.

Once the installation is complete, the `conda` command will be available. This can be tested by checking the help message via `conda -h`.

The conda package for BatchConvert can be found in the following link:

https://anaconda.org/Euro-BioImaging/batchconvert.

For the installation of BatchConvert, a fresh conda environment is recommended. As an example, a new conda environment with the name of “batchconvert_env” can be created using the following command:

```
conda create -n batchconvert_env
```

The created environment can be activated via:

```
conda activate batchconvert_env
```

Then BatchConvert can be installed to the activated environment by running the following command:

```
conda install -c euro-bioimaging -c conda-forge -c bioconda batchconvert
```

After this step, the `batchconvert` command will be available as long as the environment “batchconvert_env” is active.

3.2.2. Installation from the source

This option could be useful if the user is willing to customise BatchConvert for their own needs and / or contribute to its further development.

Installation of BatchConvert from the source requires that the `python` command refers to Python-3, as BatchConvert has been mainly tested with Python 3.6 or higher. To ensure that the right Python version is used, the user can create a conda environment with a Python version 3.6 and higher and run BatchConvert inside this environment.

The first step to acquire BatchConvert is to clone the repository:

```
git clone https://github.com/Euro-BioImaging/BatchConvert.git
```

If Nextflow is already available on the system, the “install.sh” file should be run to install BatchConvert:

```
source BatchConvert/installation/install.sh
```

If Nextflow is not available, an alternative installation file can be run to install Nextflow together with

BatchConvert:

```
source BatchConvert/installation/install_with_nextflow.sh
```

Note that the installation together with Nextflow requires the package manager Anaconda (or Miniconda) to be available on the system.

After running the installation file, the `batchconvert` command will be globally available.

Note that the dependencies “bftools”, “bioformats2raw”, “go-mc” and “aspera-cli” will be automatically installed by Nextflow upon the first conversion execution.

2.3. Configuration

The default parameters for BatchConvert are stored in a specific file located in: “BatchConvert/params/params.json.default”. The content of the **params.json.default** file can be printed to the terminal window using the following command:

```
batchconvert show_default_params
```

Each time a conversion is run, **i)** these **default** parameters are first copied to a new file (located in “BatchConvert/params/params.json”), **ii)** the parameters in the **params.json** file are updated with any new values entered in the conversion command, **iii)** the **params.json** file is then read and parsed to execute the conversion workflow. In this context, configuration in this section refers to the process of setting **new default parameters** for BatchConvert, i.e., updating the **params.json.default** file.

There are multiple ways of configuring the default parameters: i) interactive configuration, i) configuration files, iii) single parameter update. Below each of these options is described.

2.3.1. Interactive Configuration

In this form of configuration, the user is iteratively presented with the parameter names in the terminal window and inquired to enter their preferred default values. This approach is useful when the user is new to BatchConvert as it allows the user to explore the tool and its various sets of configurable parameters.

Interactive configuration itself has three subcategories: i) configuration of the access credentials for the remote storage(s), ii) configuration of the default conversion parameters, iii) configuration of the Slurm computer cluster options.

3.3.1.1. Configuration of Access to the Remote Storage(s)

BatchConvert can communicate with two types of remote locations: The first one is the s3 object storage (private and public). The second one is the BioImage Archive (Hartley et al., 2022, 2023). The interactive configuration commands for these two remote locations are `batchconvert configure_s3_remote` and `batchconvert configure_bia_remote`, respectively.

Below is an example of how to configure an s3 bucket. The inquiries from the computer are shown in red and the input values supplied by the user are shown in white.

```
user@pc-institute:~$ batchconvert configure_s3_remote
enter remote name (for example s3)
Enter "skip" or "s" if you would like to keep the current value
s3
enter url:
Enter "skip" or "s" if you would like to keep the current value
https://s3.yourinstitute.de
enter access key:
Enter "skip" or "s" if you would like to keep the current value
KMAYNRDWJM9KDSK3ALWELPE
enter secret key:
Enter "skip" or "s" if you would like to keep the current value
zxtl8eLsLwlLmf5nsk9Llp3riyLspjrLaLl
enter bucket name:
Enter "skip" or "s" if you would like to keep the current value
MyProjectStorage
Configuration of the default s3 credentials is complete
```

This will configure BatchConvert for access to an imaginary s3 bucket named “MyProjectStorage”. Similarly, below is an example of configuration for submission to BioImage Archive. Note that submission to BioImage Archive is achieved via BioStudies (Sarkans et al., 2018), which serves as a temporary residence for the submitted image datasets prior to their release in the BioImage Archive. Consequently, the configuration process requests the secret directory for the BioStudies user space.

```
user@pc-institute:~$ batchconvert configure_bia_remote
enter the secret directory for your BioStudies user space:
<your-secret-directory>
configuration of the default bia credentials is complete
```

Once the configuration of the remote ends is complete, data can be read from and written to any paths in these locations.

3.3.1.2. Configuration of the Slurm Options

BatchConvert can run on Slurm-managed computer clusters. Here is an example of configuring BatchConvert for distributed execution using the command `batchconvert configure_slurm`:

```
user@pc-institute:~$ batchconvert configure_slurm
Please enter value for queue_size
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the current value ´50´
s
Please enter value for submit_rate_limit
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the current value ´10/2min´
s
Please enter value for cluster_options
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the current value ´--mem-per-cpu=3140 --cpus-per-
```

```
task=16´
--nodes=4 --ntasks-per-node=16 --cpus-per-task=1
Please enter value for time
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the current value ´6h´
s
configuration of the default slurm parameters is complete
```

Here we only updated the “cluster_options” parameter and kept the current values for the rest of the parameters. The parameter values supplied here will be used for creating Slurm jobs when the `--profile` option is specified as “cluster” on the command line.

3.3.1.3. Configuration of the Conversion Parameters

BatchConvert allows all conversion parameters to be passed as command line arguments to the conversion command. However, to avoid long command lines with many parameter specifications, the user can set their preferred default values once and then simply execute the conversion command without specifying any conversion parameter.

Here is an example of configuring BatchConvert for conversion into OME-TIFF using the command `batchconvert configure_ometiff`:

```
user@pc-institute:~$ batchconvert configure_ometiff
Please enter value for noflat
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the parameter´s current value, which is <bfconvert
defaults>
s
Please enter value for series
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the parameter´s current value, which is <bfconvert
defaults>
s
Please enter value for timepoint
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the parameter´s current value, which is <bfconvert
defaults>
s
...
...
...
...
...
Configuration of the default parameters for 'bfconvert' is complete
```

For a detailed overview of what each conversion parameter actually does when converting to OME-TIFF, we refer the reader to the official documentation of the Bio-Formats software: https://docs.openmicroscopy.org/bio-formats/6.7.0/users/comlinetools/conversion.html.

Similarly, here is an example of configuring BatchConvert for conversion into OME-Zarr using the command `batchconvert configure_omezarr`:

```
user@pc-institute:~$ batchconvert configure_omezarr
Please enter value for resolutions_zarr
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the parameter´s current value, which is <bioformats2raw defaults>
7
Please enter value for chunk_h
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the parameter´s current value, which is <bioformats2raw defaults>
256
Please enter value for chunk_w
Click enter if this parameter is not applicable
Enter "skip" or "s" if you would like to keep the parameter´s current value, which is <bioformats2raw defaults>
256
...
...
...
...
...
Configuration of the default parameters for 'bioformats2raw' is complete
```

When performing a conversion to OME-Zarr, arguably, the most important parameters are the number of pyramidal resolution layers (--resolutions_zarr) and the chunk sizes along depth (--chunk_d), width (--chunk_w) and height (--chunk_h), since these parameters strongly influence the efficiency of the access to the output OME-Zarr data. For an in-depth discussion of how chunk size can affect the read performance of the Zarr data, we refer the reader to (Nguyen et al., 2023).

Note that the parameters configured for the OME-TIFF and OME-Zarr conversion options are parsed to their equivalents in the relevant tools `bfconvert` and `bioformats2raw`, respectively, which are the entrypoints for each conversion process. In this regard, the initial defaults for these parameters in BatchConvert are the defaults of the `bfconvert` and `bioformats2raw` tools, as indicated in `<>` in the last two examples. Importantly, all parameters can be reset to these initial defaults by using the command `batchconvert reset_defaults`.

Also note that, like Nextflow, BatchConvert prioritises command line parameters over the configuration files. As an example, if the following line were executed, any value assigned to the `chunk_h` parameter during the configuration would be overridden with 20.

```
batchconvert omezarr --chunk_h 20 <path/to/input> <path/to/output>
```

2.3.2. Configuration Files

BatchConvert can also be configured by using configuration files, which can store specific sets of default values for the above-mentioned parameters. One advantage of this approach is that the user can keep multiple configuration files with different combinations of parameter values and quickly switch between

them whenever needed.

An easy way to create a configuration file is by exporting the current params.json.default file, which can be achieved by using the following command:

```
batchconvert export_default_params <path/to/config-file.json>
```

Once exported, the contents of the config-file.json can be manually updated by the user, who can then supply the modified file to BatchConvert using the following command:

```
batchconvert configure_from_json <path/to/config-file.json>
```

This will set the modified values in the config-file.json as the new defaults for the respective parameters of BatchConvert.

The initial export of the default parameters, as illustrated above, is recommended as it permits saving the current set of parameters to a backup file, which can then also be shared with other users, or can be used to reproduce a workflow.

On the other hand, the content of the `config-file.json` in the above command can be any set of parameters that BatchConvert accepts. For example, we can imagine that we have a file named `s3credentials.json` with the following content:

```
{
 "S3REMOTE": "s3",
 "S3ENDPOINT": "https://s3.myinstitute.de",
 "S3BUCKET": "MyProjectStorage",
 "S3ACCESS": "KMAYNRDWJM9KDSK3ALWELPE",
 "S3SECRET": "zxtl8eLsLwlLmf5nsk9Llp3riyLspjrLaLl"
}
```

As the name suggests, the file contains parameters for an imaginary s3 remote storage. We can use this file to configure BatchConvert for s3 access using the command:

```
batchconvert configure_from_json <path/to/s3credentials.json>
```

The same parameters in the params.json.default will be updated with the values given here. A practical approach would be to store multiple such credential files for multiple s3 endpoints. Similarly, Slurm parameters can be stored, for instance, in a configuration file named `slurm_params.json` with the following content:

```
{
 "queue_size": "50",
 "submit_rate_limit": "10/2min",
 "cluster_options": "--mem-per-cpu=3140 --cpus-per-task=16",
 "time": "6h"
}
```

We can similarly configure BatchConvert for Slurm using the following command:

```
batchconvert configure_from_json <path/to/slurm_params.json>
```

Using configuration files also permits the collective configuration of different process categories. For example, the illustrated parameter sets for the s3 storage and the Slurm scheduler can also be combined into a single configuration file named s3_and_slurm.json with the following content:

```
{
 "S3REMOTE": "s3",
 "S3ENDPOINT": "https://s3.myinstitute.de",
 "S3BUCKET": "MyProjectStorage",
 "S3ACCESS": "KMAYNRDWJM9KDSK3ALWELPE",
 "S3SECRET": "zxtl8eLsLwlLmf5nsk9Llp3riyLspjrLaLl",
 "queue_size": "50",
 "submit_rate_limit": "10/2min",
 "cluster_options": "--mem-per-cpu=3140 --cpus-per-task=16",
 "time": "6h"
}
```

Then the user can simply run the following command to configure BatchConvert for both s3 access and Slurm scheduling:

```
batchconvert configure_from_json <path/to/s3_and_slurm.json>
```

Depending on their needs, the user can prefer to keep a separate configuration file for each distinct process category (i.e., remote access, Slurm scheduling, conversion to OME-TIFF and conversion to OME-Zarr) or a single configuration file combining the parameters for all process categories.

2.3.3. Single Parameter Update

When the user intends to modify only a single default parameter, creating / keeping a configuration file or starting an interactive configuration session would be inconveniently time-consuming. Instead, the user can use the `set_default_param` subcommand to update the default value of individual parameters, as indicated in the following example:

```
batchconvert set_default_param cluster_options "--nodes=3 --mem-per-cpu=3140 --cpus-per-task=16"
```

2.4. Execution

Execution of BatchConvert triggers a Nextflow workflow consisting of multiple processes, which are asynchronously parallelised over the input data. These processes are generally related to format conversion and/or data transfer between the execution environment and a remote location. Different processes can be linked together in different arrangements, allowing the user to thoroughly control the workflow they wish to run.

The first argument to the batchconvert command is either ometiff or omezarr depending on the desired output format. This first mandatory argument is typically followed by a number of optional arguments related to the execution and a further two mandatory positional arguments, supplied at the second to last and last positions, specifying the input and output directories, respectively. So the simplest conversion command line for creating OME-Zarr data would look like:

```
batchconvert omezarr <path/to/input> <path/to/output>
```

This line will convert local data into OME-Zarr using the default conversion parameters, (which can be configured in different ways as demonstrated in the section 3.3 “Configuration”). From this point

onwards, we describe various command line options that can be added to this simple line to have more customised workflows.

2.4.1. Working Directory

BatchConvert is built upon Nextflow and shares its mechanism of temporary data management, in that the temporary data and the log files associated with a workflow are stored in their own directories, namely work and log directories, which can be specified by the user. The work directory is particularly important since this is where the actual computations of the workflow take place.

BatchConvert organises these two directories in a single “working directory”, which is named by default the “WorkDir”. The structure of the working directory is shown below:

```
./WorkDir
├── logs
└── work
```

BatchConvert allows the working directory to be specified in the command line using the `--workdir` (or `-wd`) argument. For example, we could modify our simple execution command above by specifying the working directory as shown below:

```
batchconvert omezarr -wd <path/to/WorkDir> <path/to/input> <path/to/output>
```

By default, the content of the working directory will be automatically deleted once the execution is complete in order to avoid unnecessary exhaustion of the disk space. To keep this data from being deleted, one can set the `--keep_workdir` flag. Then our command line would become:

```
batchconvert omezarr -wd <path/to/WorkDir> --keep_workdir <path/to/input> <path/to/output>
```

After this command is executed, all the temporary data and the log files created by Nextflow can be found at the `<path/to/WorkDir>`. This feature can be particularly useful for debugging purposes.

It is important to note that the default path for the working directory depends on the execution environment. If there is a `/scratch` directory, the default working directory becomes `/scratch/.batchconvert/WorkDir`. If `/scratch` does not exist, then the WorkDir is created in the base directory of BatchConvert. This default behaviour, however, can be very easily overridden by modifying the configuration file as discussed in the section 3.3 “Configuration”. Arguably, the easiest way to do so is by using the command `--set_default_param` as follows:

```
batchconvert set_default_param workdir <path/to/new/default/WorkDir>
```

After this command is executed, the new default working directory will be the `<path/to/new/default/WorkDir>`.

Note that the choice of the working directory determines where the actual computations are performed, and therefore can be important for the performance of the execution. In particular, HPC clusters often provide filesystems (usually under `/scratch`) with optimised read / write speed. The working directory can be selected under such a high-speed filesystem in order to achieve optimal workflow performance. In general, prior to running big jobs, it is advisable to check the `--workdir` parameter and update it accordingly if needed.

2.4.2. Execution Profiles

An execution profile refers to the environment in which the workflow is running. There are five different profiles: i) conda, ii) docker, iii) singularity, iv) cluster, and v) manual. One of these options can be supplied to the command line argument `--profile` or `-pf`. The specified execution profile determines how the dependencies are acquired.

Conda profile (--profile conda / -pf conda) requires that a conda package manager exists on the system. When this option is selected, BatchConvert checks if a cached conda environment (under the path "BatchConvert/.condaCache") exists. If so, the workflow is run in this environment. If the environment does not exist (which is the case upon the first conversion execution), it will be automatically created with the required dependencies. The created environment will be used to run the workflow, and retained as a cache to be used for any subsequent executions using the conda profile option. Conda profile is the default option in BatchConvert.

Docker profile (--profile docker / -pf docker) requires the presence of a docker engine on the system. Selection of the docker profile option triggers BatchConvert to search the local docker images for a particular image (namely bconv:minimised), which contains all the dependencies. If the image is not locally available, it will be pulled from our repository (https://hub.docker.com/r/bugraoezdemir/bconv) at the Docker Hub. The workflow will be run by creating a container from this image. Note that this Docker repository was created specifically for BatchConvert and will be maintained as part of it.

Singularity profile (--profile singularity / -pf singularity) requires the presence of a singularity runtime on the system. In a way similar to the conda profile option, BatchConvert first checks if a cache directory with the singularity image (under the path "BatchConvert/.singularityCache") exists. If not, it is created by pulling the image from our repository at the Docker Hub. The cached image is used to create the container and run the workflow.

Cluster profile (--profile cluster / -pf cluster) must be selected when BatchConvert is to be run on an HPC cluster using the Slurm scheduler. Singularity is automatically used to access the dependencies, using the caching approach as above. The jobs are, however, executed on the cluster based on the Slurm options specified during the configuration as described in the section 3.3.1.2 "Configuration of the Slurm Options".

Manual profile (--profile manual / -pf manual) is reserved for users who wish to manually acquire the dependencies and make them accessible to BatchConvert (e.g., because they do not have either of conda, docker or singularity on their system).

Here is an example command line specifying an execution profile as docker:

```
batchconvert omezarr -pf docker <path/to/input> <path/to/output>
```

Important to note here is that since the container images or the conda environment are not initially available, the start of the first execution will be delayed as BatchConvert needs to pull the image / create the environment upon the first execution of a conversion using a specific --profile option. All subsequent executions using the same --profile option, however, will use the cached resources acquired upon the first execution and thus will start faster.

2.4.3. Execution Modes

BatchConvert allows for two modes of conversion: i) one-to-one conversion and ii) grouped conversion. The former assumes that the input images are independent of each other, and, therefore, maps each input image to an output OME-TIFF/OME-Zarr (see Fig-2A). Very often, however, datasets contain multiple files that represent different channels, time frames and / or z-sections of the same image series. In such cases, the user may prefer the grouped conversion mode. When this option is selected, BatchConvert analyses the input filenames for patterns that may link the files that are part of a single series, and performs a grouped conversion - a conversion that merges multiple linked files into the same output (see Fig-2B and Fig-2C).

There are a number of conversion parameters that can be passed to each of the batchconvert omezarr and batchconvert ometiff prompts. Note that these parameters are generic to both conversion modes, i.e., all the parameters that can be passed to the one-to-one mode can also be passed to the grouped conversion mode. The parameters can be listed together with short explanations via batchconvert omezarr -h or batchconvert ometiff -h.

One-to-one conversion is the default mode and thus needs no specific command line options. Indeed, our simple example command will run a one-to-one conversion:

```
batchconvert omezarr <path/to/input> <path/to/output>
```

Some of the important parameters for the conversion into OME-Zarr include `--resolutions_zarr` (or `-rz`) `--chunk_h` (or `-ch`) `--chunk_w` (or `-cw`) `--chunk_d` (or `-cd`), which specify the number of pyramidal resolutions, chunk height, chunk width and chunk depth, respectively. With these arguments included, our command line would look like:

```
batchconvert omezarr -rz 5 -ch 8 -cw 256 -cd 256 <path/to/input> <path/to/output>
```

To prompt the grouped conversion mode, `--merge_files` flag must be added to the command line above. If the grouped conversion mode is active, BatchConvert attempts to group the input files based on the following rules:

1. Filenames within a group **must** be uniform except for one or more numerical characters - so called **numerical fields** (e.g., representing indices of z-planes for a 3D image).
2. A numerical field within a group must be of uniform length. This means that if the numerical field reaches multi-digit numbers, leading zeroes should be included where needed to make the length of the numerical field uniform within the group.
3. A numerical field within a group **must** show uniform incrementation (e.g., differences between sequential time stamps of a time-series dataset cannot be non-uniform).
4. Typically, the numerical fields within a group **should** be preceded by a **dimension specifier**, which can be “c” or “C” for channel, “z” or “Z” for z-slices and “t” or “T” for time.

If these conditions are fulfilled (see Fig-2B), the input dataset can be correctly split into groups and each group will be merged along the detected dimensions into a single series. Note that merging along multiple dimensions is also supported (i.e., files representing the single planes of a 5D image can be merged.). All the arguments that can be specified for a one-to-one conversion (such as chunk sizes, resolution levels, etc.) can also be specified for a grouped conversion.

In certain cases, automatic grouping of the input files may not be achievable. For example, if the 4th criterion is not fulfilled, BatchConvert will not know, along which dimension(s) the files must be concatenated (see Fig-2C). To allow the user to overcome such challenges, BatchConvert offers the command line option `--concatenation_order`, with which the user can manually specify the particular numerical field(s) and the concatenation axes in the command line for each existing group in the input dataset.

An example of this scenario is shown in Fig-2C, where the following command is run to perform a grouped conversion:

```
batchconvert omezarr --merge_files --concatenation_order ct,a Input_Folder Output_Folder
```

With this command line, the user provides BatchConvert with the following information:

1. There are two groups in the input dataset, corresponding to the two comma-separated values “ct” and “a”.
2. The first group (the group that comes first in alphanumerically ordered filenames) is assigned the dimension specifier “ct”, and the second group is assigned “a”.
3. The dimension specifier “ct”, having a character length of 2, tells BatchConvert to use the last two numerical fields in the filenames of the corresponding group as the concatenation axes (note that in Fig-2C, the filenames corresponding to this group lack any axis information). By providing the dimension specifier “ct”, the user dictates that the penultimate numerical field represents the channel dimension (as imposed by the specifier “c”) and the last numerical field represents the time dimension (as imposed by the specifier “t”). The conversion will, therefore, merge the files in this group along the channel and time dimensions.
4. The dimension specifier “a”, having a character length of 1, tells BatchConvert to use the last single numerical field in the filenames of the corresponding group as the concatenation axis. In

this case, contrary to point 3, the user specifies that BatchConvert should infer the identity of the concatenation axis from the filenames, automatically (as imposed by the specifier "a" for "automatic"). In the example in Fig-2C, the character preceding the last numerical field is "T" and therefore the concatenation will be along the time dimension. Note that the automatic detection only works when the filename contains the axis information.

As explained with this example, the --concatenation_order argument gives the user full control over the management of the merging axes. Of importance is that if the user uses this option to specify a concatenation axis for a specific numerical field, any existing dimension specifier for this numerical field in the filenames will be overridden with the value provided by the user.

The grouped conversion approaches discussed above are based on the assumption that the input files are, at the metadata level, independent of each other, and that BatchConvert can figure out how the files are related to each other based on the filename patterns. In this regard, behind the scenes, the --merge_files flag (together with the --concatenation_order if specified) triggers a Nextflow process to create a bioformats-compatible metadata file (so called a pattern file with the extension ".pattern"). A pattern file specifies the dimensional relationships between the files via regular expressions and gets consumed by the next Nextflow process where the respective converter, bfconvert or bioformats2raw, uses it to decide how the images should be grouped.

In many cases, however, such a pattern file, or even more complex types of metadata files specifying the context/hierarchy of the images, already exist. For example, some microscopes, while creating a collection of images, automatically yield a metadata file representing the entire image hierarchy. Alternatively, the user can create such a metadata file using a separate third-party software. In any case where such a **bioformats-compatible** metadata file exists, BatchConvert can be instructed to directly use it for the grouped conversion, instead of creating a new metadata file. This can be done by combining the arguments --merge_files and the --metafile as illustrated with the following example.

```
batchconvert omezarr --merge_files --metafile <Name_of_Metadata_File> <Input_Folder> <Output_Folder>
```

It is important to note that, for this option to work, the metadata file supplied to the --metafile argument must be recognisable by the Bio-Formats library. In addition to the pattern files described above, another prominent example of such metadata files is a companion OME-XML file (with the extension ".companion.ome"). The companion OME-XML is an xml-based metadata format that can also group together multiple images but offers additional features that are missing in the pattern files: i) In addition to capturing dimensional relationships between the image files, companion OME-XML can also specify more complex file hierarchies such as those of high-content screening datasets. ii) Companion OME-XML can also store extensive OME metadata (such as those related to the imaging instrument, modality, set-up, etc).

Here are a few other important considerations when using this option: the metadata file must be located in the same folder as the input images (i.e., in the <Input_Folder>), and as such, the value supplied to the --metafile (i.e., <Name_of_Metadata_File>) must be just the name of the file, **NOT** its full path, as opposed to the <Input_Folder> and <Output_Folder>, which must be the full paths to the input and output locations, as usual.

While there are further details concerning specifications of the grouped conversion, we do not cover them here for the sake of concision. Further documentation and examples are available at https://github.com/Euro-BioImaging/BatchConvert.

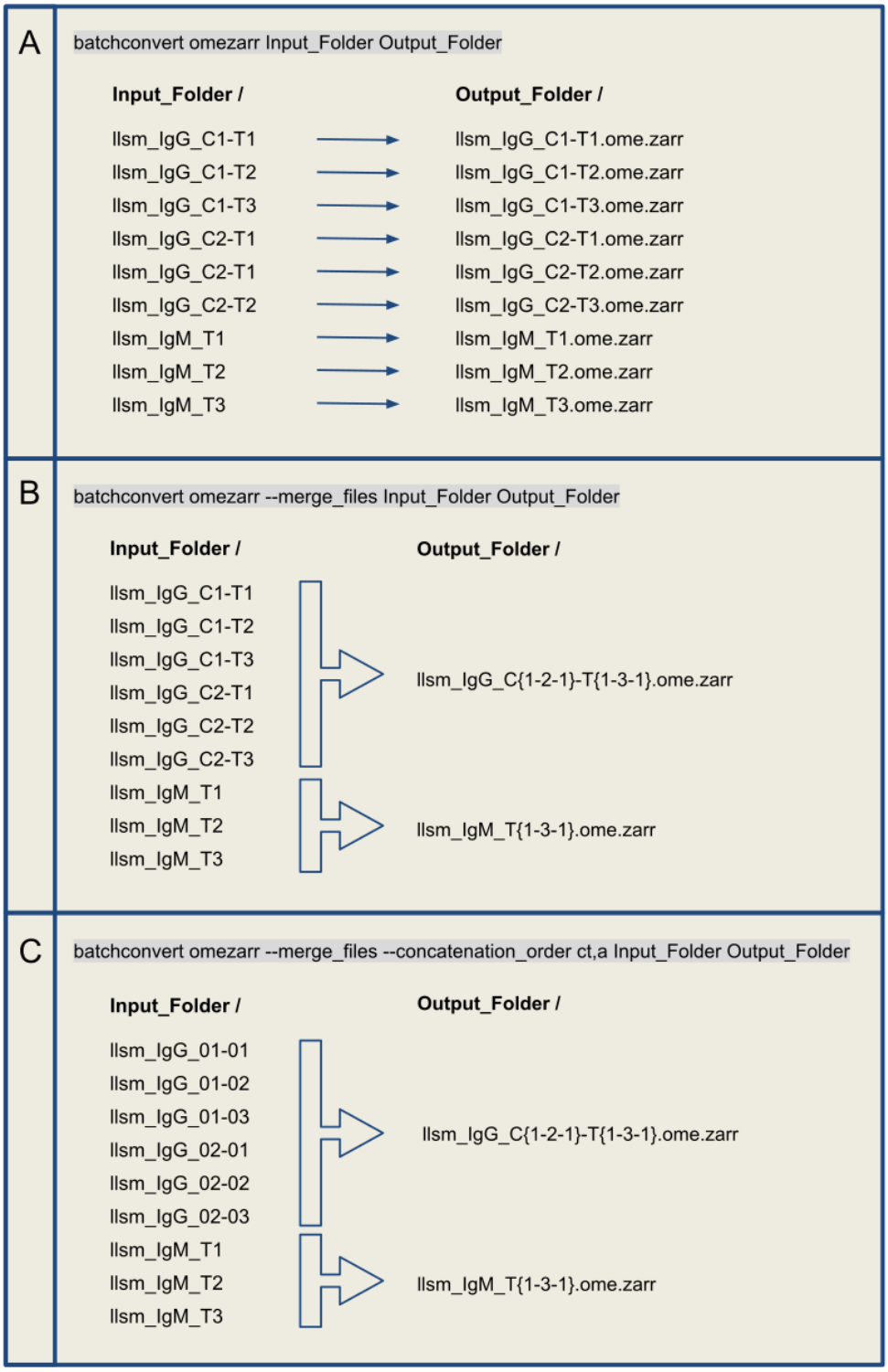

Figure-2. Different conversion modes. (A) A one-to-one conversion maps each input file to a single series. (B) A grouped conversion, activated via the `--merge_files` argument, merges multiple files into a single series based on the information in the filenames. A fully automatic detection of the concatenation axes requires the numeric fields in the filenames to be preceded by a dimension specifier, namely the characters “c” or “C” for channel, “t” or “T” for time and “z” or “Z” for z-slice. (C) It is possible for the user to manually enforce the axes of concatenation, using the `--concatenation_order` argument. The specified axes for the different groups are represented as comma-separated values that are passed to this argument.

2.4.4. Working with Remote Data

Both of the aforementioned modes of conversion support remote input-output options, which are currently s3 buckets and BioImage Archive. The first step to work with remote data is to establish the access as explained in the section 3.3 “Configuration”. BatchConvert will then be able to communicate with the relevant remote location based on the arguments `--source_type` (or `-st`) and `--destination_type` (or `-dt`). Both of these arguments default to `local`, which allows BatchConvert to read from, and write to, paths in the local filesystem. Specifying `--source_type` as “s3” or “bia” will prompt BatchConvert to search for the input path in the configured s3 or the BioImage Archive endpoints, respectively. Similarly, the output location, where the conversion results are transferred, can be specified as a remote location by passing either of “s3” or “bia” to the `--destination_type` argument. Using these two arguments, therefore, it is possible to specify various input-output combinations. As an example, one can execute the following command to convert datasets stored in an s3 bucket and submit the output data to BioImage Archive:

```
batchconvert omezarr -st s3 -dt bia <path/to/input> <path/to/output>
```

When this line is executed, a workflow consisting of three processes is triggered: i) transfer of the data to the execution environment, ii) conversion of the data into OME-Zarr, iii) submission of the resulting OME-Zarrs to BioImage Archive. In the first process, BatchConvert searches for the `<path/to/input>` in the configured s3 bucket. It is to be noted that the path is assumed to be relative to the bucket name provided in the configuration, and thus the bucket name should not be included in the `<path/to/input>`. The second process produces OME-Zarrs from the transferred data. The third process creates `<path/to/output>` at the configured BioStudies user space and transfers the produced OME-Zarrs to this path. The BioImage Archive team will then collect the data from the BioStudies user space and make it available at the BioImage Archive web page.

An important point here is that the described workflow is both parallelised and asynchronous, which enables fast and memory-efficient execution.

2.4.5. Filtering Input Files

BatchConvert allows for selection of input files by matching patterns to the filenames via the argument `--pattern` or `-p`, which accepts string values to be used as patterns. Only those files, whose names contain this string, are selected as inputs, on which to run the workflow. Negative selection is also possible, using the `--reject_pattern` or `-rp` argument. In contrast with the former case, here, only the files, whose names do **not** contain the specified string, will be selected as inputs. The following example shows a combined use of both of these arguments.

```
batchconvert omezarr -p "mutation" -rp "treatment" /path/to/input /path/to/output
```

This command will convert only the files that include “mutation” **and** exclude “treatment” in the filename.

As an alternative to the pattern-based selection, “globbing” is also supported for input-filtering and can be a better fit for more complex filtering purposes. For example, the following command will select images with the “.oir” extension within multiple directories, convert them and store conversion outputs in a single output directory (note that globbing **requires** double quotes around the input path).

```
batchconvert omezarr "/deep/path/to/*/*.oir" /path/to/output
```

Between these two approaches, the former approach, using `--pattern` and `--reject_pattern` can be readily applied to data in the local filesystem as well as data in an s3 bucket. For example, the following command will transfer only the files with “mutation” in their filename from the given s3 path and convert them.

```
batchconvert omezarr -st s3 -p "mutation" /path/to/input /path/to/output
```

On the other hand, globbing currently works only in the local filesystem and cannot be applied to

remotely stored inputs. As such, the following command would fail:

```
batchconvert omezarr -st s3 "/path/to/input/*.oir" /path/to/output
```

with the following error message:

```
Error: Globbing cannot be used with remote files. Try using '--pattern' or '-p' argument to filter input files with patterns.
```

3. Discussion

Considering the myriad proprietary file formats in biological imaging, format harmonisation has become a crucial goal in the field. Stable conversion of image files to a standard format is a common first step in image data inspection and analysis workflows. Here we have introduced BatchConvert, a command line tool for efficient conversion of bioimage collections to the standard file formats OME-TIFF and OME-Zarr.

An important functionality of BatchConvert is the support for multiple conversion modes. While one-to-one conversion mode maps each input file to an output series, grouped conversion mode allows for merging the input data along specific dimensions. This feature is useful in cases where data belonging to a single series is stored in multiple files. This scenario is not uncommon in biological imaging. For instance, different channels, or time frames of a large dataset are often saved to separate files to avoid large monolithic files. This representation, however, comes at the cost that any subsequent process that needs to operate over these multiple distinct files (processes such as multichannel / multiframe visualisation) becomes trickier and requires manual work. Grouped conversion into a single OME-Zarr is an effective solution for such issues as the chunk-based storage allows for loading arbitrary subsets of the data, without altering the layout or dimensionality.

Another important feature of BatchConvert is that it supports private remote locations (such as an s3 object store) for the input / output data, meaning that the conversion can be coupled to data transfer. This is especially useful for researchers who regularly work with remote data and maintain their data in a private object store. This option also facilitates data submission to BioImage Archive (Hartley et al., 2022, 2023) in the OME-Zarr format, encouraging researchers to publish their data in centralised repositories and enhance FAIRness of their data.

It is important to note that all of the above-mentioned functionalities of BatchConvert can be utilised in an HPC cluster with a Slurm job scheduler. Moving jobs from the local executor to Slurm only requires modifying a single parameter. This feature provides additional flexibility to the users who have access to a suitable HPC cluster and who are willing to run big jobs that are hard to handle locally.

BatchConvert has been developed within the purview of the EOSC Future test science project “Open Imaging Data Sharing in EOSC / COVID-19 as Demonstrator”, which sets interoperability as one of its main areas of focus. Accumulation of more biological data from different fields (omics, structural biology, imaging, etc.) in the cloud repositories creates new potential for integration of such data resources with each other, which, in turn, also enhances findability of the data, for instance via cross-referencing between different resources. BatchConvert helps populate open image data repositories with quality datasets (Subramanian et al., 2023)(eg: S-BIAD-628[6]). Increase in the number of such datasets will promote image data interoperability via tools like 3DBionotes[7], which supports interaction between imaging data and protein structures.

In summary, BatchConvert is a workflow generator that creates asynchronous parallel jobs for file format conversion with multiple conversion modes and support for remote input-output data. As the name suggests, BatchConvert is designed for batch processing and is optimal for input datasets with large numbers of files. Glueing together the excellent packages bioformats2raw, bftools, go-mc and aspera-cli, and providing several new features, BatchConvert generates OME-TIFF and OME–Zarr data

[6] https://www.ebi.ac.uk/biostudies/bioimages/studies/S-BIAD628
[7] https://3dbionotes.cnb.csic.es/ws

following the respective format specifications, and optionally allows for their deposition in remote storage, including BioImage Archive.

Though the current version of BatchConvert only supports Unix-based systems, future versions will be extended to include Windows systems as well. The future versions of BatchConvert will also facilitate data submission to IDR, the primary resource for reference image datasets. As NGFF specifications enable the storage of derived data in the OME-Zarr layout, the upcoming versions of BatchConvert will offer options to automatically integrate segmentation labels, connected component analyses and display metadata in the output OME-Zarr. This will ensure a standardised representation of the raw and derived data and thus contribute to their FAIRness.

4. References

Anaconda Software Distribution. Computer software. Vers. 2-2.4.0. Anaconda, Nov. 2016. Web. <https://anaconda.com>.

Di Tommaso, P., Chatzou, M., Floden, E. W., Barja, P. P., Palumbo, E., & Notredame, C. (2017). Nextflow enables reproducible computational workflows. *Nature Biotechnology*, *35*(4), 316–319. https://doi.org/10.1038/nbt.3820

Goldberg, I. G., Allan, C., Burel, J.-M., Creager, D., Falconi, A., Hochheiser, H., Johnston, J., Mellen, J., Sorger, P. K., & Swedlow, J. R. (2005). The Open Microscopy Environment (OME) Data Model and XML file: Open tools for informatics and quantitative analysis in biological imaging. *Genome Biology*, *6*(5), R47. https://doi.org/10.1186/gb-2005-6-5-r47

Hartley, M., Iudin, A., Padwardhan, A., Sarkans, U., Yoldaş, A. K., & Kleywegt, G. J. (2023). Providing open imaging data at scale: An EMBL-EBI perspective. *Histochemistry and Cell Biology*, *160*(3), 211–221. https://doi.org/10.1007/s00418-023-02216-2

Hartley, M., Kleywegt, G. J., Patwardhan, A., Sarkans, U., Swedlow, J. R., & Brazma, A. (2022). The BioImage Archive – Building a Home for Life-Sciences Microscopy Data. *Journal of Molecular Biology*, *434*(11), 167505. https://doi.org/10.1016/j.jmb.2022.167505

Linkert, M., Rueden, C. T., Allan, C., Burel, J.-M., Moore, W., Patterson, A., Loranger, B., Moore, J., Neves, C., MacDonald, D., Tarkowska, A., Sticco, C., Hill, E., Rossner, M., Eliceiri, K. W., & Swedlow, J. R. (2010). Metadata matters: Access to image data in the real world. *Journal of Cell Biology*, *189*(5), 777–782. https://doi.org/10.1083/jcb.201004104

Miles, A., Jakirkham, Bussonnier, M., Moore, J., Orfanos, D. P., Bourbeau, J., Fulton, A., Bennett, D., Lee, G., Zain Patel, Abernathey, R., Kristensen, M. R. B., Rocklin, M., AWA BRANDON AWA, Saransh Chopra, De Andrade, E. S., Hamman, J., Durant, M., Schut, V., … Noyes, C. (2023). *zarr-developers/zarr-python: V2.16.1* (v2.16.1) [Computer software]. Zenodo. https://doi.org/10.5281/ZENODO.8263439

Moore, J., Allan, C., Besson, S., Burel, J.-M., Diel, E., Gault, D., Kozlowski, K., Lindner, D., Linkert, M., Manz, T.,

Moore, W., Pape, C., Tischer, C., & Swedlow, J. R. (2021). OME-NGFF: A next-generation file format for expanding bioimaging data-access strategies. *Nature Methods*, *18*(12), 1496–1498. https://doi.org/10.1038/s41592-021-01326-w

Moore, J., Basurto-Lozada, D., Besson, S., Bogovic, J., Bragantini, J., Brown, E. M., Burel, J.-M., Casas Moreno, X., De Medeiros, G., Diel, E. E., Gault, D., Ghosh, S. S., Gold, I., Halchenko, Y. O., Hartley, M., Horsfall, D., Keller, M. S., Kittisopikul, M., Kovacs, G., … Swedlow, J. R. (2023). OME-Zarr: A cloud-optimized bioimaging file format with international community support. *Histochemistry and Cell Biology*. https://doi.org/10.1007/s00418-023-02209-1

Nguyen, D. M. T., Cortes, J. C., Dunn, M. M., & Shiklomanov, A. N. (2023). *Impact of Chunk Size on Read Performance of Zarr Data in Cloud-based Object Stores* [Preprint]. Preprints. https://doi.org/10.1002/essoar.10511054.2

Sarkans, U., Gostev, M., Athar, A., Behrangi, E., Melnichuk, O., Ali, A., Minguet, J., Rada, J. C., Snow, C., Tikhonov, A., Brazma, A., & McEntyre, J. (2018). The BioStudies database—One stop shop for all data supporting a life sciences study. *Nucleic Acids Research*, *46*(D1), D1266–D1270. https://doi.org/10.1093/nar/gkx965

Subramanian, K., Opstad, I., Singh, R., Ikmi, A., Prevedel, R., Kemmer, I., & Özdemir, B. (2023). *Volumetric light-sheet data of freely moving sea anemone* [dataset]. https://www.ebi.ac.uk/biostudies/BioImages/studies/S-BIAD628

Weisbart, E., & Cimini, B. A. (2022). *Distributed-Something: Scripts to leverage AWS storage and computing for distributed workflows at scale*. https://doi.org/10.48550/ARXIV.2210.01073

Williams, E., Moore, J., Li, S. W., Rustici, G., Tarkowska, A., Chessel, A., Leo, S., Antal, B., Ferguson, R. K., Sarkans, U., Brazma, A., Carazo Salas, R. E., & Swedlow, J. R. (2017). Image Data Resource: A bioimage data integration and publication platform. *Nature Methods*, *14*(8), 775–781. https://doi.org/10.1038/nmeth.4326

Statements and Declarations

Competing interests

The authors have no competing interests to declare that are relevant to the content of this article.

Funding

B.Ö. receives funding from EU Horizon 2020: grant agreement no. 101017536 (EOSC Future). Co-funded by the European Union. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union or the granting authority. Neither the European Union nor the granting authority can be held responsible for them.

Acknowledgements

The authors acknowledge EOSC Future Test Science Project partners Carolina Simón Guerrero and

Jose-Ramón Macías (CNB-CSIC and INSTRUCT-ERIC), and Reagon Karki and Phillip Gribbon (Fraunhofer Institute for Translational Medicine and Pharmacology and EU-OPENSCREEN ERIC). The authors acknowledge Yi Sun for valuable technical discussions. The authors acknowledge all members of the Euro-BioImaging Hub for discussions and support.

Data availability

Data sharing is not applicable to this article as no original datasets were generated or analysed as part of this study. The test dataset (S-BIAD628) used is available at the BioImage Archive: https://www.ebi.ac.uk/biostudies/bioimages/studies/S-BIAD628.

The code is available at: https://github.com/Euro-BioImaging/BatchConvert (Version: v0.0.2). The workflow is available at: https://workflowhub.eu/projects/167#workflows.