Chapter 7

User friendly high-content imaging big-data analysis on a single desktop: R package H5CellProfiler.

This chapter is based on a manuscript in preparation:

Steven Wink, Joost Beltman and Bob van de Water

Division of Toxicology, Leiden Academic Centre for Drug Research, Leiden University, Leiden, The Netherlands

1. Abstract

Technological development has led to ever increasing amounts of data in high content screening. For utilizing such data in an efficient, thorough and user friendly manner we developed the R-package H5CellProfiler. H5CellProfiler is based on R-packages data.table, ggvis, ggplot2 and shiny. H5CellProfiler launches a browser which allows scientists to analyze large single cell datasets and make statistical summaries and graphs on their own local desktop in a fast and memory efficient manner. In addition, single cell track-labels are calculated and broken tracks are re-connected based on user-defined thresholds resulting in unique sets of annotated tracks.

2. Introduction

High-throughput high-content automated imaging (HT-HCI) includes image acquisition, image storage, image annotation, image analysis, data analysis and storage of the analysis results. Image acquisition is performed by high-end automated microscopes that are equipped with hardware and software for stage-movement or objective movement and often with an incubation chamber that has temperature and CO_2 regulation allowing live cell imaging. These conditions allow the acquisition of large imaging datasets from multi-well plates capturing the dynamics of cell biological events.

Image annotation can be divided in two categories: first, technical annotation which is automatically generated by the acquisition software and consists of technical acquisition parameters. Second, biological annotation consisting of parameters that the scientist needs to add himself. Biological annotation of images is closely related to data storage of images in a file server, which is managed by a database management system (DBMS). The DBMS allows annotation, organization, querying and often visualization of the acquired images. Well-known image DBM systems are OMERO from the Open Microscopy Environment (OME) [329] and its commercialized version Columbus.

Image analysis can be performed with the aid of commercial or open source software which differ in the offered functionality and flexibility. Several commercial software systems exist that provide easy to use image analysis solutions, sometimes directly combined with image acquisition. Well-known software systems include NIS-Elements (NIKON), Metamorph (Molecular Devices), AxioVision (Zeiss) and Volocity (Perkin-Elmer). Recently, commercial systems have been developed that even combine multiple HT-HCI stages. For example, Columbus from PerkinElmer is able to store, annotate and analyze images and analyze the resultant data with several statistical and graphical tools due to integration with TIBCO Spotfire.

A disadvantage of using commercial software for image analysis is that it is difficult or simply not possible for researchers to modify or extend it because the source code and even the applied image analysis algorithms are hidden in order to secure the commercial product. Commercial systems are therefore best suited for standard image analysis pipelines and mostly used by industry or by labs with relatively basic and fixed image analysis needs.

Labs which frequently change equipment or have specialized image analysis requirements often prefer open source solutions. The most well-known solution is ImageJ [330], which is an important platform for scientific image analysis development. However, it can be challenging to build workflows for automated image analysis because it requires some knowledge in the available ImageJ plugin libraries, a decent understanding of image analysis algorithms and Java

scripting skills. For this reason, some researchers prefer user friendly, GUI-based image analysis software tools. A well-known example is CellProfiler [49], which was developed initially by Carpenter and colleagues. In CellProfiler, parameter optimization can be performed with the help of the GUI and accompanying graphical displays of the segmentation results. Subsequently, image analysis with appropriate parameters can be performed in an automated fashion.

In parallel with recent technological advancements in microscopy, the interest in the analysis of single cell behavior has strongly increased. Indeed, research of the heterogeneity of cell populations [331, 332], of the morphology of subcellular structures [326], of transcription factor oscillatory dynamics [217] or of cell migration [333] all require single cell measurements. As a result, the amount of data has strongly increased with consequences for the handling of image analysis output. A typical overnight live single cell imaging session of an *in vitro* 2D 384-well plate leads to around 20 GB of raw imaging data (approximately 20,000 images) and depending on the features of interest up to 5 GB of analysis data. Memory limitations require data dumps during the analysis by temporary storage in relational databases or hierarchical file based formats. Hence, labs need to set up a dedicated database server which requires maintenance and technical expertise, which can be challenging for biological oriented labs.

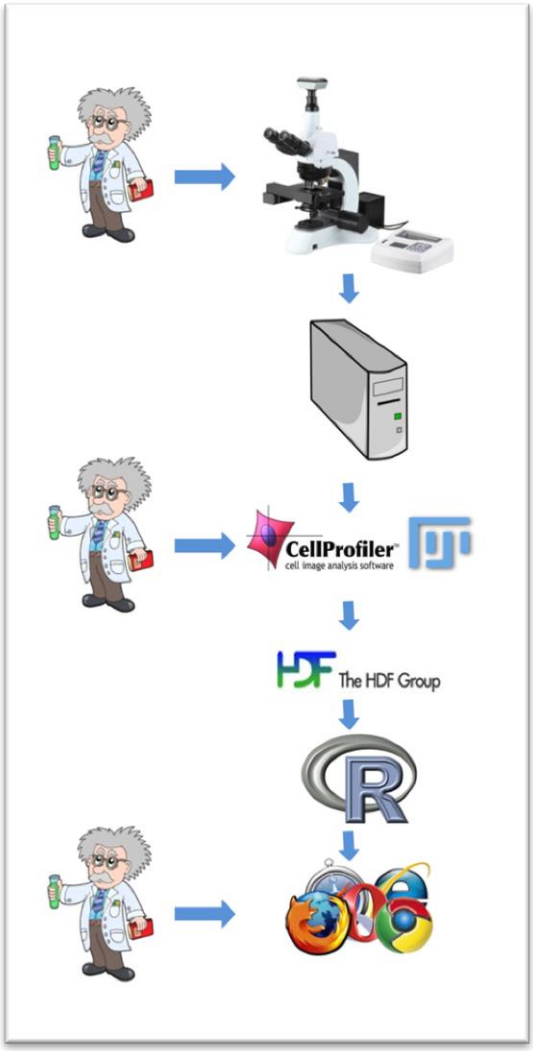


Figure 1: **User-friendly High Content image analysis workflow.** The scientist interacts with the imaging device, sets up the image analysis with GUI-based CellProfiler run on the local desktop, and finally performs the data analysis with a browser based GUI which communicates with the R server (usually on the local desktop).

Recently, CellProfiler included a feature for storing measurements during the analysis using the file format HDF5 [50], a portable data format without size limitations due to the hierarchical structure. A HDF5 file can be navigated through the use of file paths – strings separated by the forward slash '/' to define the hierarchical structure of the data, analogue to the Unix file system. In addition, the HDF5 format has been implemented specifically for cell-based assays in high-content microscopy in the form of CellH5 which can be used as a format for data exchange. CellH5 and several CellH5 interfaces have been developed by Huber and colleagues [194], allowing for visualization of the stored images and reading and writing of the quantitative data. However, for CellProfiler HDF5 output user friendly solutions do not yet exist. Therefore, we have developed the R-package H5CellProfiler.

H5CellProfiler requires a metadata file with biological annotations and a HDF5 file with the quantitative measurements from CellProfiler. The package loads the data into R-memory and

organizes user-requested features at the level of single cells. Subsequently, a browser is launched and functions as a GUI for the user to manipulate, summarize and display the single cell data. This enables biologists to perform their image analysis with CellProfiler followed by the semi-automated production of single cell data summaries and visualizations that would not be possible using spreadsheets. Thus, H5CellProfiler allows to perform all data operations on a single desktop without the need to set up database and client software and as such makes the entire image analysis workflow user friendly (Fig. 1).

3. Results and Discussion

3.1. Description of the H5CellProfiler package.

3.1.1. H5CellProfiler architecture.

H5CellProfiler reads the HDF5 data and uses the input arguments (which can be typed in the R-terminal or provided as a file) and annotation files provided by the user to reformat and summarize the data in a flexible, intelligent and computationally efficient manner (Figure 2). It is flexible because the user guides the reformatting with help of the GUI and has several options: extracting relevant data into a large text file containing a single row per parent object, making a summary for each treatment, writing a text file containing rows per tracked object and columns corresponding to the time points for each time-lapse or plotting graphical displays of the data The size of the data hardly affects the performance as long as it does not exceed the available memory capacity. This is because all data manipulations are very fast due to parallel computing and ordered indexing. As a result, up to 100GB of data re easily handled by H5CellProfiler.

3.1.2. Parallel computing.

*H5CellProfiler offers parallel processing to speed up the analysis of large datasets. This is achieved with the parallel computing package 'foreach' and parallel back-end registration for windows operating systems 'doSNOW'. The number of parallel sessions that can be initiated for reading and data analysis is limited by the user with a maximum of the number of separate HDF5 files. Parallel computing especially speeds up the summary calculations, the data reshaping for plotting and the re-labeling of tracking data (the number of cores for re-labeling the tracking data is independent of the number of hdf5 files and is set by the user-set argument). Chunks of the data in R memory are divided over the availa*ble cores without duplication of the data for each core, thus ensuring memory-efficient parallel computing.

3.1.3. Reading HDF5 and formatting data.

We employed the recently developed R package 'rhdf5'[194] for reading and writing HDF5 files. An advantage of using the HDF5 format to store and read data is its annotated hierarchical hyper-rectangular structure. Together with the annotation, this enables the selection of data subsets without the need to query through the entire dataset. As a consequence, very large datasets do not slow down reading significantly.

3.2. Application of R package H5CellProfiler.

Basic knowledge of CellProfiler is required to be able to provide the correct arguments for the H5CellProfiler function and are therefore briefly described here. A detailed manual of CellProfiler can be found on the website www.cellprofiler.org.

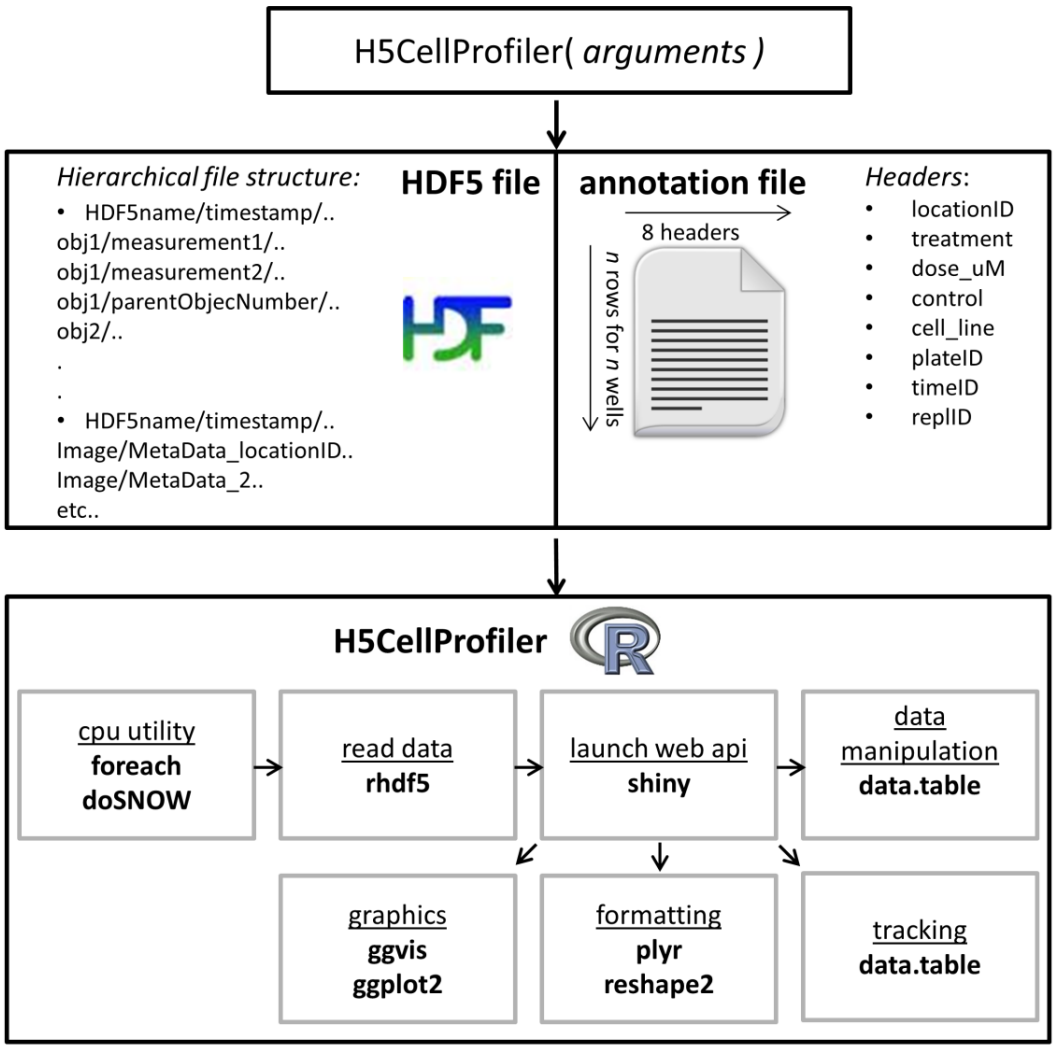


Figure 2: **Architecture of H5CellProfiler.** The main function is run by typing the H5CellProfiler (*arguments*) function in the terminal or by providing the arguments in a file. H5CellProfiler then uses the HDF5 file containing the image analysis data and the annotation file provided by the user containing the biological annotation. The architecture of the H5CellProfiler package is depicted in the lower panel: the order of the separate internal processes (underlined) using various R-packages (bold) are depicted by the arrows.

3.2.1. Regular expressions that enable linkage of biological annotation with image data.

In CellProfiler, variables are captured using a simple collection of regular expressions. For example, for an image named 'xy0013c1t01.tif' and located in a folder named '../C03' (according to the well number in the plate), two sets of regular expressions capture this image annotation. In this case, the first regular expression would be (?P<imageNumber>xy[0-9]4)c[1-4] (?P<tp>t[0-9]).tif to capture the image number 0013 in the CellProfiler variable 'imageNumber' and the time point 01 in the variable 'tp'. Note that the characters inside the square brackets are the characters to be searched for in the image file names, the numbers inside the curly brackets denote the number of such characters, and the construction with '(?P<variable_name>…)' determines in which CellProfiler variable the contents will be stored. Similarly, the folder name '../C03' can be captured by the regular expression *[\\/](?P<Well>.*)$ or by (?P<Well>[A-Z] [0-9])$ in the CellProfiler variable 'Well'. In the first expression the construction '*[\\/]' is a regular expression that searches

for a forward slash, the two backward slashes are escape characters for the forward slash. The asterisk '*' is a wildcard for any character(s). The dollar sigh '$' forces the matching to take place in reverse order (starting at the end). The first expression is flexible, the second is more specific.

The key feature in these examples is that images are automatically annotated by CellProfiler and that this information is stored together with the image analysis results in the HDF5 file. The annotation obtained by the regular expressions are stored in the HDF5 output and the HDF5 file path is defined as 'Metadata_*variablename*' Together with a user provided annotation text file, this enables H5CellProfiler to link biological annotation to the image measurements (Figure 2). A tab delimited text file containing the biological annotation is created by the scientist which consists of several columns/headers and a row for each well of all multi-well plates that are simultaneously analyzed with H5CellProfiler. The image analysis results of separate plates can be stored in a single HDF5 or in multiple HDF5 files, depending on how the images were analyzed. In addition to the well (column 'locationID') and plate id (column 'plateID') several optional columns can be entered: the column 'treatment' can be used to enter annotation regarding e.g. compound or siRNA treatment of the well. Moreover, if the column 'dose_µM' is used to annotate the applied concentrations, dose response curves can be automatically generated at a later stage using the GUI. Furthermore, it is possible to use columns to annotate replicate numbers ('replID') and time information ('timeID') which can be useful for imaging of multiple plates that represent different time points. Finally, the column 'control' can be used to enter control information, which can be useful for plate normalization purposes in the GUI.

3.2.2. H5CellProfiler arguments.

The H5CellProfiler package requires several of these metadata HDF5 file paths as arguments to the H5CellProfiler function. At least the 'locationID' and 'imageID' are required, and if multiple plates have been stored in a single HDF5 file also the 'plateID' variable must be provided as a HDF5 file-path. In addition, it is possible to provide a 'timeID' and 'replicateID' if required. The 'plateID', 'timeID' and 'replID' variables can be defined either as a HDF5 file-path or as a constant for a single or for multiple HDF5 files (the latter is only possible if these variables remain the same within a single HDF5 file.

Conversion of the integer values for the time (either by regular expression or manually defined per HDF5 file) to real experimental time is performed based on the two arguments 'exposureDelay' and 'timeBetweenFrames'. These represent the delay of the first image relative to the biological perturbation of interest and the time between two consecutive time points, respectively.

The CellProfiler nomenclature consists of objects and measurements on these objects. Objects can be cells, parts of cells (e.g. cytoplasm) but also the images themselves. Users will segment the objects of interest and define the name of these objects. For each object, several measurement categories can be chosen in CellProfiler, e.g., intensity, texture and morphological measurements can be passed on to H5CellProfiler. Up to ten of these object measurements can be passed on to H5CellProfiler by giving them as argument for the 'myFeaturePathsA' argument. The single entries of these arguments should have an identical format as the CellProfiler output namely '*object/MeasurementCategory_measurement*'.

Objects are often related to other objects. For example, when there are multiple objects for each cell, a parent object must be defined to link the secondary objects to. In this manner, every object defined by segmentation or object manipulations such as subtraction or shrinking has an object number and an associated parent object number. These object relations are provided to H5CellProfiler as the arguments 'parentObject', 'childObject1', 'childObject2',…..,'childObject5' and 'tertiaryObject', thus defining how the single cell data are organized. The 'tertiaryObject' is a CellProfiler-defined child object defined by other-object manipulations such as subtraction (often the cytosol as defined by cell subtracted by the nucleus). The image and parent object define the primary key of the tables in R-memory and secondary objects are associated to their corresponding parent object and thus on the same row in the table. Thus each row in the table corresponds to a unique parent-object. If multiple secondary objects exist for single parent objects (e.g. multiple organelles belonging to a single parent-nucleus), a summary statistic must be calculated. This summary statistic is by default the mean but can be added as an argument in the form of an r-function. Recommended statistics include the mean or a particular quantile, and is defined as a function, e.g. 'function(x) mean(x, na.rm = TRUE) ' to the argument 'multiplePerParentFunction'.

Finally the number of cpu cores the user wants to assign for all data processing steps can also be provided to H5CellProfiler as the argument 'numberCores'. However this is limited to maximally the number of HDF5 files provided, however for the tracking optimization algorithms this is only limited by the amount of individual locations that were imaged (number of time-lapses)

3.2.3. Track re-labeling and optimization.

CellProfiler also performs tracking in the context of time lapse imaging, which means that objects are linked to each other in time. The H5CellProfiler package recognizes tracking parameters automatically using regular expressions, so the user does not need to provide additional arguments. However, the 'fixTrackingFunction' utility from H5CellProfiler offers several options like re-labeling and combining 'broken' tracks based on user input. During tracking with CellProfiler, segmentation errors frequently leads to track interruption for one or several frames. These errors are quite common when cells tend to cluster together while migrating, resulting in an underestimation of the number of segmented objects. Alternatively, overestimation of segmented objects can also cause tracks to break because the tracking label can connect to the spurious cell which will disappear in the next few frames. Tracking errors are maintained in time, meaning that a small percentage of segmentation errors per time point results in accumulation of errors in track labeling. CellProfiler labels a tracked cell based on the object it is estimated to originate from at the previous time step and can assign the same label to multiple objects. Thus, a cell division or a single frame segmentation error results in the same label being assigned to multiple tracks. The 'fixTrackingFunction' re-labels these to new tracks starting at the last time point of imaging, using the track-parent object labels assigned by CellProfiler. Track-parents and track-children correspond to objects related in time, thus the same object tracked through time. When track-child objects lead to an identical track-parent (thus in the previous time point) the track label will be assigned to the closest parent using the x-y coordinates of the parent and child objects. After the re-labeling, the algorithm additionally attempts to reconnect broken tracks over 1-3 time frames if the argument 'reconnect_frames' is set by the user. In addition, the maximum pixel

distance to reconnect partial tracks can be set with the arguments 'max_pixel_reconnect1', 'max_pixel_reconnect2' and 'max_pixel_reconnect3'. If multiple tracks are within the distance threshold the closest tracks are connected. Finally, the minimal track length can be set with the argument 'minTrackedFrames' which means that only tracks with at least that length after relabeling and reconnection are retained. The tracks can be plotted using the GUI and it is up to the user to validate the reconnect-argument settings by comparing the track plots with the time-lapses.

3.2.4. Graphical User Interface.

After the selected data has been read into R-memory and is reorganized with the help of the parent-child object relations and the annotation file, a browser is launched (Figure 3). This utility is based on the package 'shiny', which is an R wrapper for HTML and JavaScript. Shiny requires a server.r and UI.r. The UI.r file defines the layout and functionality of the GUI and generates the tools required for user input and output. The server.r file functions as the R server that performs the operations.

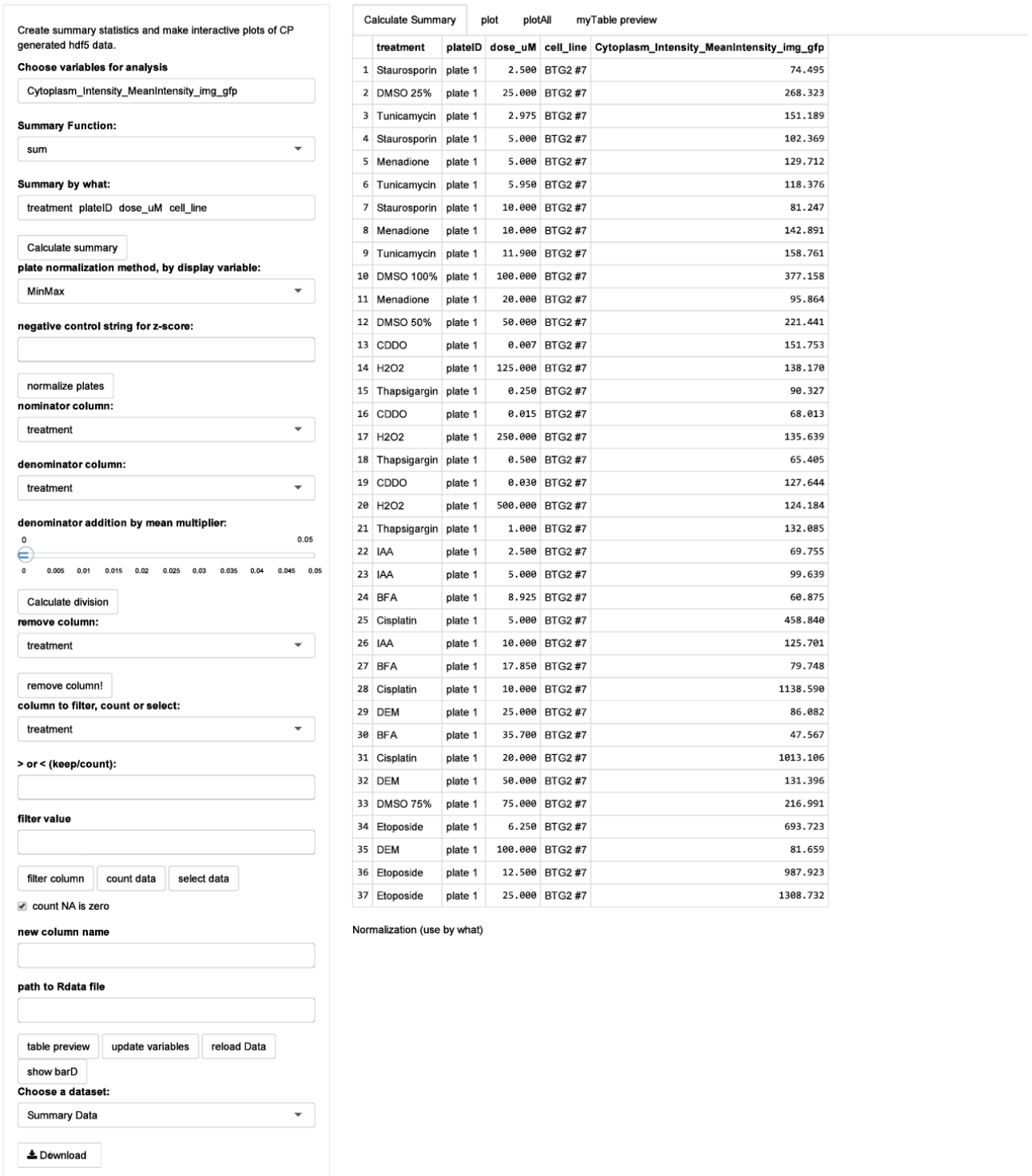
The web application provides a graphical user interface for frequently used data manipulation, summarization and graphical operations. It sends commands to an R-session so the user only indirectly employs R commands. This approach helps to ensure that the data manipulations are performed correctly.

The user can choose a single or multiple variables for summarization, and one or multiple factors over which the summarization takes place. The offered summary functions are the mean, standard deviation, median, sum, minimum, maximum and several quantiles. As soon as these options have been selected the table is calculated using the R-package 'data.table' and takes fractions of a second for tables up to 1GB. The raw measurement variables can also be normalized by z-score, by the plate median or by the min-max method $(x-x_{plate\_min}) / (x_{plate\_max}-x_{plate\_min})$. These normalizations are performed over the summarized results as defined by the 'by.what' factors because single cell data contain too many extreme outliers.

Other data manipulations provided by the browser include column division, deletions, filtering, counting and selection. Finally, the modified single cell table and summarized table can both be downloaded as tab-delimited text files. When the summarized table has been created the 'plot' tab (Figure 3, right panel) provides the option to explore the data graphically in an interactive manner (Figure 4). The user can select the treatment to be displayed and the variables to be plotted. Moreover, a third dimension in the data can be visualized with color mapping. and font size. The minimum and maximum values of the x- and y-axis can be set to 'fixed' (i.e. employing the minimum and maximum values of the entire dataset), to 'free' (i.e. specific ranges for each plot), or can be set manually. Finally, the dose values can be set to 'dose-levels' instead of their own specific dose ranges which are usually different for each treatment (see example in Figure 5). Dose level 1 is the lowest value in a dose-range and the highest dose level corresponds to the highest concentration employed for each treatment separately.

The tab 'myTable preview' (Figure 3, right panel) of the GUI displays 100 randomly selected rows of the single cell data table, which is useful to view the table along with the applied modifications.

The R package 'ggplot2' offers faceting of variables which means that multiple graphs can be



	treatment	plateID	dose_uM	cell_line	Cytoplasm_Intensity_MeanIntensity_img_gfp
1	Staurosporin	plate 1	2.500	BTG2 #7	74.495
2	DMSO 25%	plate 1	25.000	BTG2 #7	268.323
3	Tunicamycin	plate 1	2.975	BTG2 #7	151.189
4	Staurosporin	plate 1	5.000	BTG2 #7	102.369
5	Menadione	plate 1	5.000	BTG2 #7	129.712
6	Tunicamycin	plate 1	5.950	BTG2 #7	118.376
7	Staurosporin	plate 1	10.000	BTG2 #7	81.247
8	Menadione	plate 1	10.000	BTG2 #7	142.891
9	Tunicamycin	plate 1	11.900	BTG2 #7	158.761
10	DMSO 100%	plate 1	100.000	BTG2 #7	377.158
11	Menadione	plate 1	20.000	BTG2 #7	95.864
12	DMSO 50%	plate 1	50.000	BTG2 #7	221.441
13	CDDO	plate 1	0.007	BTG2 #7	151.753
14	H2O2	plate 1	125.000	BTG2 #7	138.170
15	Thapsigargin	plate 1	0.250	BTG2 #7	90.327
16	CDDO	plate 1	0.015	BTG2 #7	68.013
17	H2O2	plate 1	250.000	BTG2 #7	135.639
18	Thapsigargin	plate 1	0.500	BTG2 #7	65.405
19	CDDO	plate 1	0.030	BTG2 #7	127.644
20	H2O2	plate 1	500.000	BTG2 #7	124.184
21	Thapsigargin	plate 1	1.000	BTG2 #7	132.085
22	IAA	plate 1	2.500	BTG2 #7	69.755
23	IAA	plate 1	5.000	BTG2 #7	99.639
24	BFA	plate 1	8.925	BTG2 #7	60.875
25	Cisplatin	plate 1	5.000	BTG2 #7	458.840
26	IAA	plate 1	10.000	BTG2 #7	125.701
27	BFA	plate 1	17.850	BTG2 #7	79.748
28	Cisplatin	plate 1	10.000	BTG2 #7	1138.590
29	DEM	plate 1	25.000	BTG2 #7	86.082
30	BFA	plate 1	35.700	BTG2 #7	47.567
31	Cisplatin	plate 1	20.000	BTG2 #7	1013.106
32	DEM	plate 1	50.000	BTG2 #7	131.396
33	DMSO 75%	plate 1	75.000	BTG2 #7	216.991
34	Etoposide	plate 1	6.250	BTG2 #7	693.723
35	DEM	plate 1	100.000	BTG2 #7	81.659
36	Etoposide	plate 1	12.500	BTG2 #7	987.923
37	Etoposide	plate 1	25.000	BTG2 #7	1308.732

Figure 3: **GUI for HCI data.** Left panel shows an example of a GUI data manipulation and summarization utility page. Utilities include summarization statistics over the requested variables and factors. Further options include choice of plate normalization methods, column division, column deletion, filtering, selection, counting and downloading the modified single cell data or summarized data. Right panel shows an example of a summarized table preview within the browser.

automatically wrapped in a single figure. This option to plot large numbers of graphs at once is offered by the 'plotAll' tab (Figure 4, right panel). In figure 5, an example of graph faceting is shown, together with the input boxes. With this approach it is possible to plot hundreds of graphs in a single figure if the pdf size is sufficient large for the graph annotations to be displayed

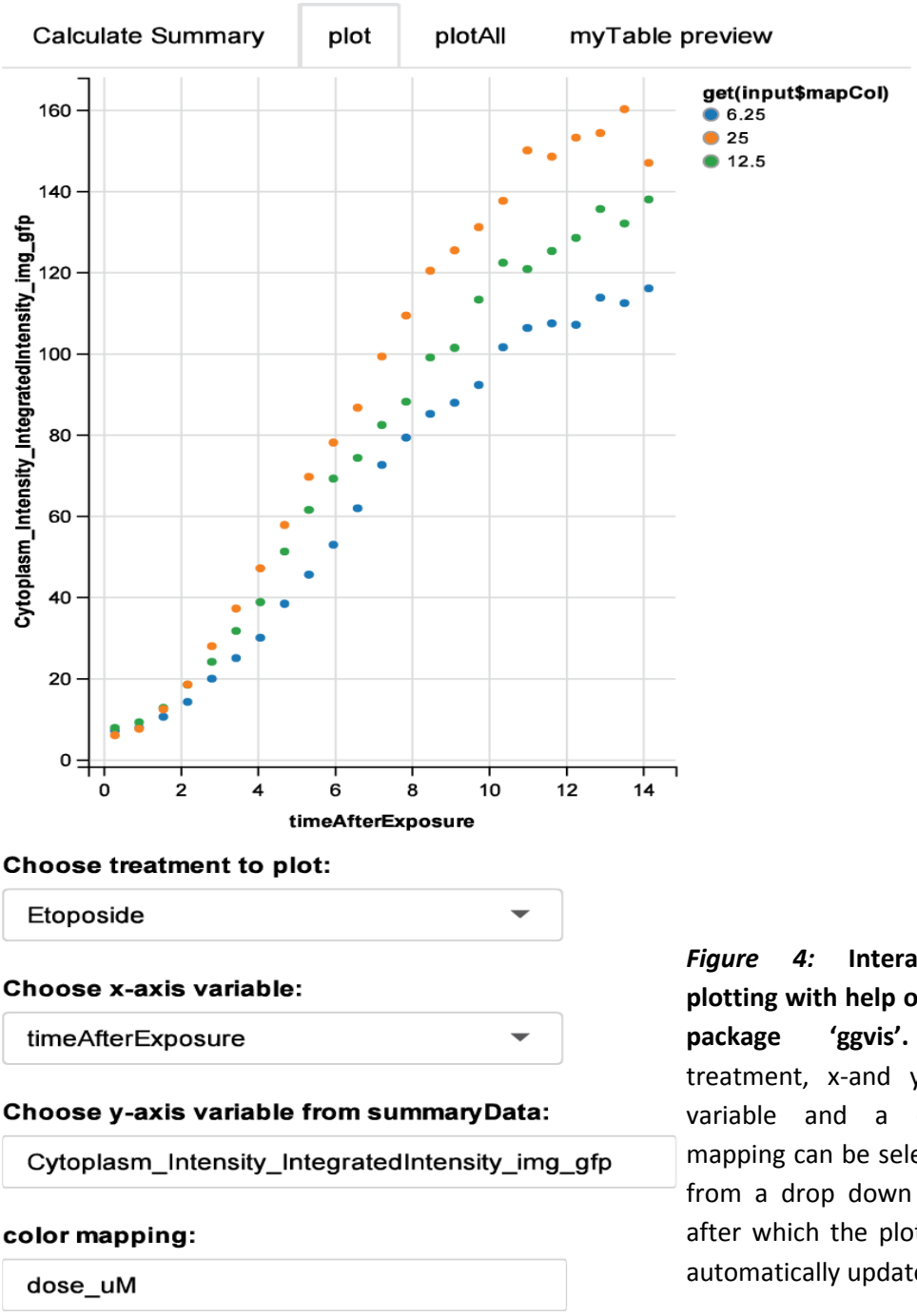


Figure 4: **Interactive plotting with help of the package 'ggvis'.** A treatment, x-and y-axis variable and a color mapping can be selected from a drop down box, after which the plot will automatically update.

correctly (with a small font). Options for multi-facetted plotting include the plot type (line plot or bar plots), variable collapse (i.e., calculation of the mean) and the choice of x and y variables. Moreover, the user can map up to six variables or specific values by employing different color, fill, shape and dot-size. Other options include adding lines between the dots and setting the pdf size

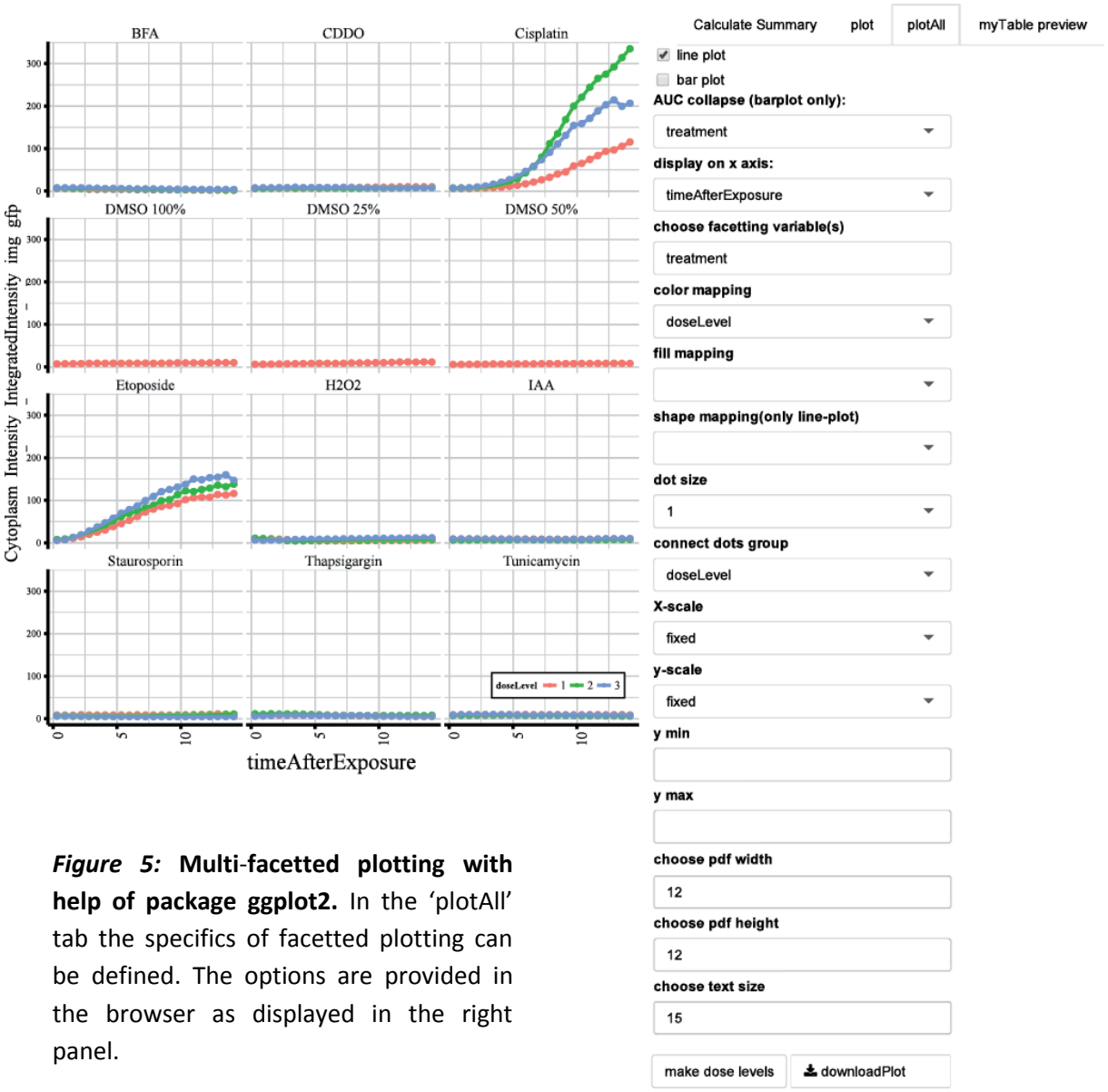


Figure 5: **Multi-facetted plotting with help of package ggplot2.** In the 'plotAll' tab the specifics of facetted plotting can be defined. The options are provided in the browser as displayed in the right panel.

4. Conclusions

High throughput high content image analysis of single cell data leads to large amounts of quantitative data. H5CellProfiler is a user-friendly solution to analyze such data without the requirement of a database. The current implementation of H5CellProfiler is based on the HDF5-output of the popular HCI image analysis tool CellProfiler. The benefit of such an implementation, in contrast to storing the data in a database, is that the entire pipeline can be performed on a single desktop without the need to set up a database and client software.

The current data workflow of CellProfiler is based on exporting the data to a database followed by analysis with CellProfiler Analyst [325] which is based on supervised single cell machine learning approaches. Additionally, the CellProfiler data can be exported in the form of spreadsheets, however this quickly becomes cumbersome in case of large datasets.

We propose the use of the image analysis and data analysis workflow with the use of CellProfiler and R-package CellProfiler for the biologist who prefers to handle their own data analysis and graphics. This will save time in the experimental design to final displayed results cycle, and avoids communication difficulties between the biologist and computational scientists. For the

more computational adherent scientists – H5CellProfiler has the additional benefit of being based on R which is a highly suited platform for data analysis, statistics and graphics.

5. Code description

5.1. mainFunction

'mainFunction' pulls the user-selected data from HDF5 together with the user provided metadata file and organizes the data into a data.table object with one row per parent object and all associated variables as columns. 'mainFunction' output is a outputLIst.Rdata file containing a 1 or several list object in the case of multiple HDF5 files. The first list object contains the data.table, metadata, user arguments and a default summary data.table. The outputList.Rdata file is used as the input for the *server.R* and *ui.R* code from the shiny application for the graphical user interface.

5.2. mainFunction code description

Argument class, string length, object definitions and relations using regular expressions matching to hdf5 paths (of h5ls of '$group', see rhdf5 vignette for details) is checked and error messages thrown to inform the user on the type of mistake.

The time stamp of the CellProfiler HDF5 output is extracted with a regular expression.

The tracked object and possibly track distance are extracted from the hdf5 paths and in addition the following track-related paths are defined: tracking label, track parent object numbers, x and y coordinates of tracked object and distance traveled. These track-related hdf5 paths are added to the user defined paths of 'myFeaturesPathsA'.

The 'hdf5IndexFun' pulls data from the hdf5 file and has input 'hdf5path', 'dataname' and 'rowIndName'. 'Hdf5path' points towards the location of the data in the hdf5 file, 'dataname' and 'rowIndName' represent the name of the data inside the object and the name of the object index data of that object, respectively. To understand how the 'hdf5IndexFun' operates, it is important to understand the high level data organization (how it is seen by e.g. the R-interface api).

Each object in the hdf5 file has information blocks; the measurements and metadata e.g. its object numbers and if applicable child or parent object numbers (Figure 6). When pulled from the hdf5 file by the h5read function of the rhdf5 package into R-memory, each information block of a certain object is represented as a list containing a 1 X $n_{objects}$ data array and an 3 X n_{images} index array with n being the number of unique entries of that particular object. The data array contains the measurements and object numbers or relation specific numbers, the index array contains the image number (first column) and the range of rows within the data array to which this image number belong (2^{nd} and 3^{rd} columns). To find out which object number a particular set of data points belongs to one must look at the data entries within the 'Number_Object_Number' information block. Since a cytoplasm object is a derived secondary object, the parent object number of each object must also be determined to link the measurement to the correct parent object. Hence, depending on the type of object different information about the object is required which explains why the user must provide this information. An alternative solution would be to use regular expressions to search for the 'Parent' and or 'Child' strings to automatically determine these relations.

Note that the order of the measurements and annotation information of the objects within an image is constant, so the data of the image-object, parent object and child objects can be

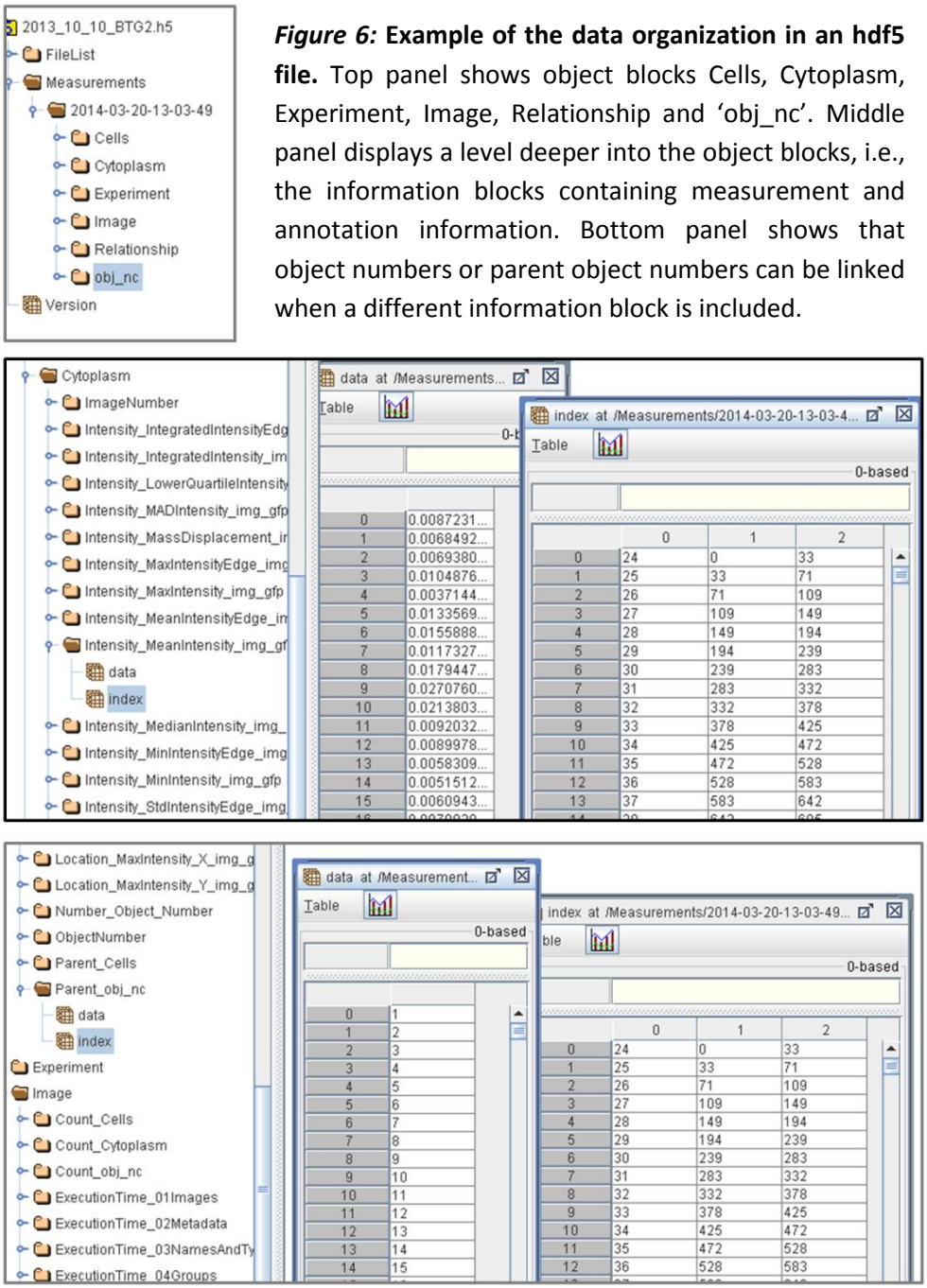


Figure 6: **Example of the data organization in an hdf5 file.** Top panel shows object blocks Cells, Cytoplasm, Experiment, Image, Relationship and 'obj_nc'. Middle panel displays a level deeper into the object blocks, i.e., the information blocks containing measurement and annotation information. Bottom panel shows that object numbers or parent object numbers can be linked when a different information block is included.

gathered per image at a time and then merged using the extracted image numbers, object numbers and parent object numbers. The 'hdf5IndFun' reformats the data (unstacks) the index information (index array has an entry for each image) by unstacking the row ranges (which point to the rows in the data array) and these are matched to the row number of the data array. The output is then a 'data.table' object for a single measurement/annotation from a single object containing the data array entries, the image numbers and the row index numbers. The row index numbers are not needed after the merge operation of the data and index arrays because the data.tables for each measurement of that same object will be identical. In addition, merging of the data.tables of the different objects will depend on the extracted data arrays containing the object numbers of the parent objects and the parent object numbers of the child-objects.

5.3. mainFunction arguments

hdf5FileNameL

list of hdf5 file names which serve as input for the function

locationID

hdf5 path as provided by user in CellProfiler with regular expressions, serves as linker column together with plateID to merge hdf5 data with metadata file. The locationID in the hdf5 file should be identical to the locaitonID in the metadata file.

plateID

list of hdf5 paths or per-hdf5 defined constant, serves as linker column together with locationID to merge hdf5 data with metadata file. The plateID in the hdf5 file or the manually defined plateIDs should be identical to the plateID in the metadata file.

timeID

list of hdf5 paths or a constant per hdf5 file containing the time point information. The timeID in the metadata file overrides the provided time argument unless no information in the metadata file is provided.

imageID

hdf5 path to image identifier as defined with regular expressions in CellProfiler.

replID

list of hdf5 paths or a constant per hdf5 file containing the replicate id constant. The replID in the metadata file overrides the provided time argument unless no information in the metadata file is provided.

myFeaturePathsA

List of hdf5 paths that define the measurements the user is interested in. CellProfiler measures everything per user-selected object measurement-class so a selection has to be made to avoid memory problems.

plateMDFileName

File name of the user defined metadata file. A tab delimited text file with headers 'locationID', 'treatment', 'control', 'dose_uM', 'plateID', 'timeID', 'replID'. Only the 'locationID' is obligatory, other columns can be left empty or defined as NA values.

parentObject

Parent object as defined by the user in CellProfiler. CellProfiler provides an object as parent by default in relation to other objects that are defined based on this parent. If this is not the case then the user must define the parent-child object using the 'relateObjects' module in CellProfiler. All objects of which measurements are selected for 'myFeaturePathsA', must be defined using the 'parentObject', 'childObject´ or 'tertiaryObject' arguments. Note that the 'parentObject' defines the way the data is formatted by 'mainFunction' as each 'parentObject' is assigned a single row in the data.table. The children of the 'parentObject' are summarized by the 'multiplePerParentFunction' argument, only if multiple children exist. This summary function is performed for each parent object even if only a single parent object has multiple children assigned. Therefore the user has to carefully consider the modules used in CellProfiler when assigning object relations. Using the parent object as seed is in general safe, relating multiple parent objects in a parent-child relation could lead to unnecessary computational overhead.

childObject1-5

Child of parent object as defined in CellProfiler 'identifySecondaryObject' module or 'relateObjects' module.

tertiaryObject

Child based on object derived from parent and/or child as defined in CellProfiler 'identifyTertiaryObject' module.

multiplePerParentFunction

User defined function that defines how multiple children per parent object are summarized.

Oscillation

Logical argument that enables analogues parameter calculation of in time oscillations (not implemented yet).

writeSingleCellDataPerWell

logical argument that enables writing of single cell data per well. Single cell text file data often becomes more manageable in spreadsheets when saved in chunks.

writeAllSingleCellData

logical argument that enables writing a single text file of the single cell data.

h5loop

numeric, default is the number of hdf5 files. Can be overwritten in case memory issues occur.

timeBetweenFrames

string that defines experimental time between consecutive frames. Format is 'hh:mm:ss' (hours, minutes, seconds). Please note that 60 minutes or 60 seconds don't exist in this format, increase the hours (no limit) or minutes by 1 instead.

exposureDelay

string that defines the time between the biological perturbation of interest and the first captured image. Note that at this time it is not possible to define time delays within a plate.