Systems Medicine
Volume 3.1, 2020
DOI: 10.1089/sysm.2019.0007

SYSTEMS MEDICINE
Journal of Medical Systems Biology and Network Medicine

Mary Ann Liebert, Inc. publishers

ORIGINAL RESEARCH: INTEGRATIVE DATA ANALYSIS IN SYSTEMS MEDICINE SPECIAL ISSUE Open Access

PurifyR: An R Package for Highly Automated, Reproducible Variable Extraction and Standardization

Wienand A. Omta,[1–3,*] Roy G. van Heesbeen,[4] Ian Shen,[2] Ad J. Feelders,[2] M.J.S. Brinkhuis,[2] David A. Egan,[3] and Marco R. Spruit[2]

Abstract

Life science experiments that employ automated technologies, such as high-content screens, frequently produce large datasets that require substantial amounts of preprocessing before analysis can be carried out. Standardization of this preprocessing becomes impossible as the dataset size increases if there are manual steps involved. Virtually no standards for preprocessing currently exist and few user-friendly tools are available that allow the cleaning of data files in a simple and transparent manner while also allowing for reproducibility.
We demonstrate in a publicly available R package, PurifyR, how preprocessing steps can be streamlined and automated. PurifyR supports multithreading and the standardization of large-matrix preprocessing. These steps provide transparent and reproducible preprocessing for matrix-oriented datasets. The PurifyR package is open source and can be downloaded from github.

Keywords: variable extraction, preprocessing standardization, knowledge discovery, data preparation, R package

Introduction

Machine-generated datasets, such as those from high-content screens (HCSs), are increasingly unavoidable in life science and bioinformatic experiments[1,2] and, given their size and complexity, are challenging to both maintain and process without automated preprocessing tool kits.[3] Big data analysis using machine-generated (sensor, image, and robotics) datasets involves ever-increasing time to clean, maintain, and summarize.[4] Time spent cleaning datasets continues to increase, consuming valuable time that could be better used for interpretation in later analyses.[3] Looking forward, manual approaches to big data analysis are unsustainable given how big data and the number of separate tools continue to increase in size and complexity.[5] Currently, thousands of variables and millions of observations are generated for every high-content screening experiment, requiring big data analysis approaches.[6] Continuous improvements to sensor software and broad advancements in hardware provide increased opportunities to capture big data in nearly every experiment,[7] hinting that big datasets will become more common and orders of magnitude larger in the near future.[8] For example, data collected during high-content experiments capture consistently higher resolutions, dimensions (3D),[9] and detail as the technology continues to develop.[10] Big data analyses require automation beyond that of pipelines and profiling tools.[11]

Cell screening experiments, such as Luminex bead technology,[12] are assisted by robots for automating repetitive and tedious cell screening experimental tasks that generate large amounts of information at the cell or even subcellular object level. Although groundbreaking, these experimental methods can have unintended consequences for researchers.[13] Up to 80% of the analysis time of large experimental datasets can be consumed[14] by repetitive and nonvalue adding tasks[15] such as

[1]Department of Cell Biology, Centre for Molecular Medicine, UMC Utrecht, Utrecht, The Netherlands.
[2]Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands.
[3]Core Life Analytics B.V., Utrecht, The Netherlands.
[4]Department of Cell Biology, NKI-AVL, Amsterdam, The Netherlands.

This article belongs to a special issue on Integrative Data Analysis in Systems Medicine.

*Address correspondence to: Wienand A. Omta, MSc, Core Life Analytics B.V., Padualaan 8, 3584 CH, Utrecht, The Netherlands, E-mail: wienand@corelifeanalytics.com

removing undesirable experiment-specific artifacts, identifying context-specific outliers, excluding systematic machine-based errors from results, interpreting manufacturer-specific machine codes, correcting missing data, and repairing inconsistent classification methods. Increases in the complexity and the amount of data create ever-increasing work for experimenters to aggregate experimental results and conclusions. Each observation and every value must be screened to validate methodological standards, to be useful for later statistical analyses, and ultimately publication.[16] Often, researchers could benefit from the ability to identify these types of data problems early in experimentation to validate the research procedures that are being followed and to ensure that data quality is sufficient for significant testing and publication.[17] These types of problems go undetected for the duration of experimentation or during the piloting stage, which could easily have been addressed and avoided if detected earlier. Ideally, experimenters could check these statistics repeatedly during the research process to identify issues and correct them before the experiment is complete and it is too late to address any collection issues. For instance, datasets must be clean for analysis and prepared before the use of any later machine learning methods.[18] Time-consuming cleansing processes require focus, which can become a distraction from the original purpose of the experiment. When these steps are not documented, the results can be nearly impossible to reproduce and can lead to experimental results that cannot be reused or compared with subsequent projects and findings.[19]

Many packages currently exist for preprocessing datasets[20] to meet the assumptions of statistical testing methods and machine learning models. However, few methods exist that can comprehensively automate the majority of preprocessing steps in both documentation and reproducibility in a manner sufficient to meet the necessary fundamental assumptions conceivable for most statistical methods such as outlier handling, missing data imputation, and feature selection.[21]

Researchers are therefore required to repeat these tedious and standardizable preprocessing steps manually in every case, which include modeling of datasets, building data pipelines, and testing the code. There is a need for robust and automated preprocessing frameworks for detecting inconsistencies in data and repairing or removing and reporting them in a transparent and autonomous manner. Furthermore, these repetitive steps can be difficult to log and can be impossible to replicate by future researchers. The exact steps followed can easily be forgotten or repeated in different sequences for various reasons, making the cleaning operations impossible to repeat during later research. Often, these issues create irreproducible datasets, which leave later researchers confused and unable to verify values or update the results with recent findings.

There is a need to automate the preprocessing of research datasets.

Previously published packages have demonstrated the ability to reduce the time necessary to automate preprocessing and produce generally usable datasets for analysis[22] while ensuring that assumptions for various machine learning and statistical methods are met without user intervention or extensive programming expertise.[23] When experimental data are processed using an automated and documented approach, results can be consistently reproduced in large workflows and used to validate datasets during experimentation to prevent using bad data (records and features) and to compare results with later studies.[24] Preprocessing pipelines themselves can also be published and reused for later research and improve over time.[25] Furthermore, automation of data preprocessing could result in a substantial reduction of the necessary effort required to make use of big datasets, creating a need to automate the preprocessing of big data.[26]

PurifyR Package Components

In this study, we present PurifyR, an R package for big data preprocessing, specifically for preparing high-dimensional datasets in a dynamic, repeatable, and autonomous manner to avoid reinventing the wheel. Experimental data sources such as automated cell screening data are primarily machine-generated datasets with thousands of columns and millions of records. Often, tens or hundreds of these matrices are generated during the course of experiments, requiring an almost impossible amount of work if aggregated and processed manually using spreadsheet software, such as MS Excel, or statistical software such as SPSS.[27] The PurifyR package facilitates three preprocessing steps; ScanR, ScrubR, and SmashR (Fig. 1). Writing preprocessing scripts are tedious, time-consuming, and error-prone.[14] Reused scripts mainly contain hard-coded column names and references to values, lists, and file names. These scripts cannot be reused in follow-up experiments and require manual rewriting for subsequent analysis, resulting in inadvertent errors or mistranslations and leading to irreproducible results.

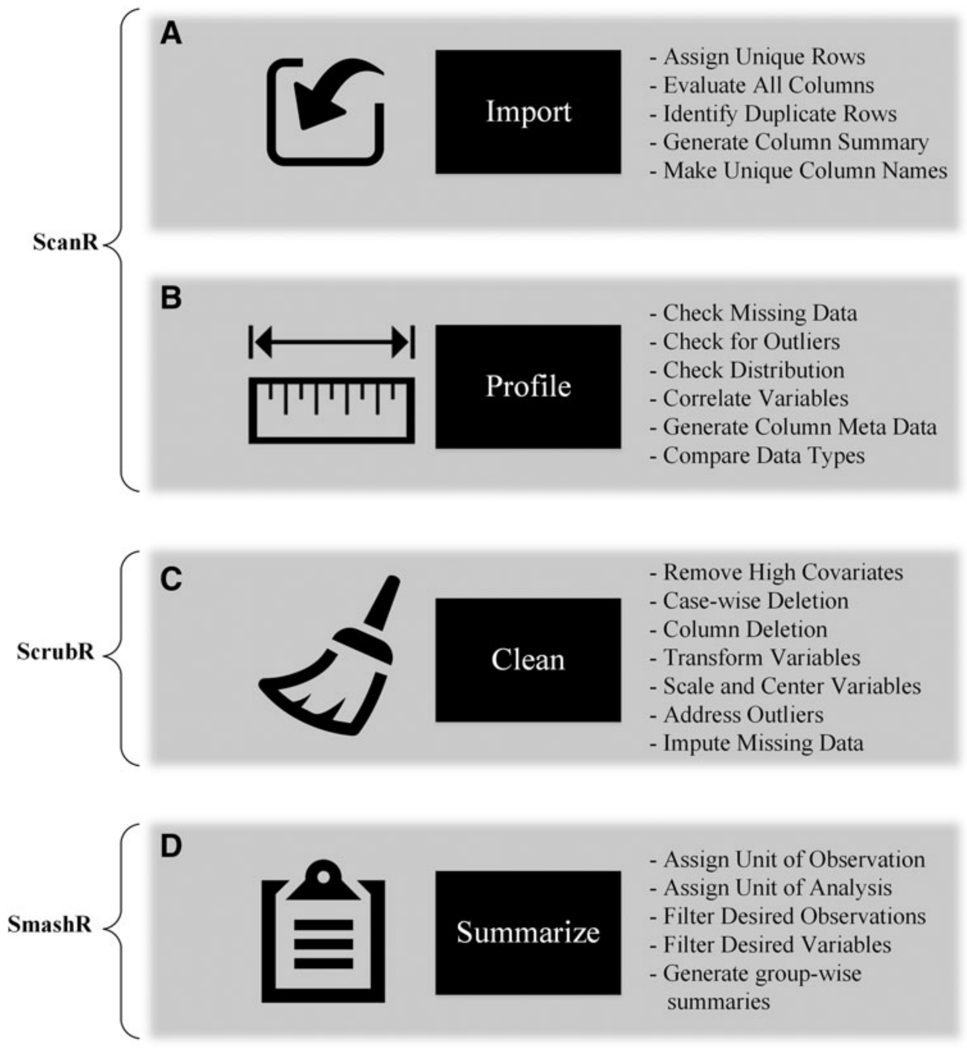

FIG. 1. The workflow of PurifyR. **(A)** Makes sure all variables have unique column names without special characters that can be handled in R. **(B)** Carries out a data scan, including missing data, distributions, and data types. **(C)** Allows for removal of outliers and handling transformation, scaling, and missing data. **(D)** Allows for aggregation of data based on a defined unit of analysis.

ScanR

ScanR generates common meta-data[28] for each column such as uniqueness and percentage missingness, variance, outliers, and distribution type. The meta-data generated in this step have been built specifically to determine which columns are considered useful and in a healthy state based on well-defined and analysis-specific assumptions of data quality.[29] Users are able to review the meta-data of the original dataset, if desired, before producing a cleaned dataset.

ScanR requires a data source for input and is a data frame or data table. This data table is sampled based on the sampling percentage to ensure enough records for summary statistics, but not an excessive number if the data table is very large. Each variable in the input dataset is summarized for simple statistics such as mean, median, and mode (Supplementary Data S1). This is completed in parallel, according to the number of processors available on the computer.

Variable names are cleaned of spaces and special characters and assigned a unique ID to ensure that duplicate values are not confused and can be used in later steps. Variable meta-data are generated and provide the ability to compare each column with others by providing a simple list of example values found in each column as well as its highest correlated column and the maximum Pearson correlation coefficient. Variables are tested against rules to identify which contain unique information and certain types of data (categorical and continuous, etc.). Each variable receives a test against other similar variables to ensure that a sufficient amount of unique information exists to warrant inclusion in later machine learning applications.

Finally, a Mahalanobis distance and covariance matrix is calculated using the resulting healthy variables. If these fail to be calculated, highly correlated variables are iteratively removed one by one until a final list of healthy predictors allows for the calculation of a nonsingular matrix. The list provides information about why certain features are excluded, for example, because of high covariation. A final list of healthy predictors is created and assigned to the most clean and unique variables. This list of healthy variables can be passed onto the next step, ScrubR.

ScrubR

The ScrubR step applies rules to each variable and then analyzes each included variable row-wise. It automatically produces appropriate and method-specific transformations, standardization, and imputes outliers and missing values given the results of the meta-data generated above. These configurations have default values and can be configured to meet the specific requirements and assumptions of later analysis methods (Supplementary Data S2).[30] Examples of subsequent analyses are principle component analysis (PCA), linear regression, and neural networks.[11] Output datasets of the ScanR function can be used without further manipulation for later analysis, feature engineering, and prediction steps.

The ScrubR function requires a dataset and the list of healthy predictors calculated in the ScanR function. It will output a cleaned version of the original data. The user can select from a few options, including the row-wise missing percent allowed in the final dataset, the threshold standard deviation allowed for outliers, the proper transformation method for variables, the intended scaling method for variables, and the desired imputation method for addressing missing data.

The function begins by removing records that exceed the missingness threshold, that is, the percentage of columns per record containing missing data. It then selects any outlying points that based on the settings define an outlying data point. Variables that have failed the skewness tests during the ScanR function will receive a recommended transformation to adjust skewness toward normality. Each variable is then scaled and replaced with a *z*-value or other desired scaling calculation. Missing values are finally case-wise deleted or imputed using a standardized approach or using a package for imputation, such as MICE[31] for random value replacement, multivariate imputation by regression, or random forest (Fig. 2).

SmashR

The SmashR function calculates estimators representing the unit of analysis. It requires a clean dataset, such as the dataset output by the previous step, ScrubR, as well as a list of healthy predictors, calculated by ScanR or provided manually. This can be used to easily represent the original identity of the data (Fig. 3). The unit of analysis input can be provided by one or more variables in the data, usually a categorical variable. The SmashR function aggregates and groups the originally observed data by the analysis variable. Any missing or N/A values within the unit of analysis are excluded from the analysis. For every value of the unit of analysis variable, a list of summary statistics is generated for every variable in the dataset, such as mean, median, maximum, and minimum values. These summarized data are extremely useful for analyzing the original dataset and creating visualizations given that the summary calculations have been preprepared and are quickly available to slice and compare the dataset. This step is useful for interpretation, comparison, and exploration of the data at a high level.

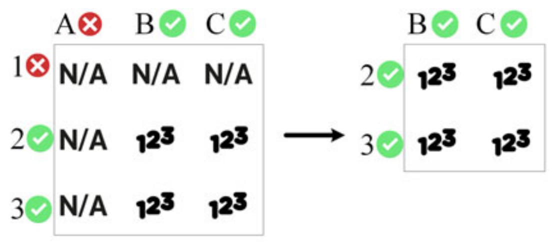

FIG. 2. Visual representation of the PurifyR input and output that checks for columns that contain missing data, that is, column A, or checks missing data on the records, for example, case-wise deletion, that is, Record No. 3.

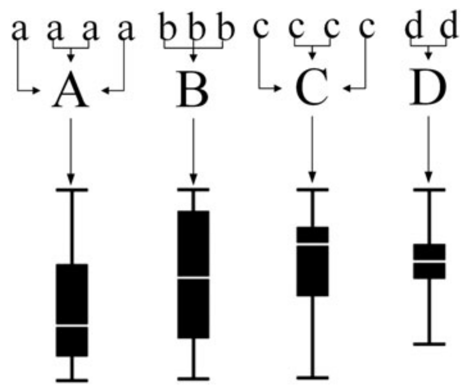

FIG. 3. This figure visually represents the function of SmashR. The lowercases represent the unit of observation, which is input for calculation of the summary statistics representing the unit of analysis, the capital letters in this figure. Let the lowercase letters be measured cases and the capital letters a set of estimators, for example, a minimum, Q1, median, Q3, and maximum estimator for input for visualization, interpretation, and understanding the data.

Results

Rule-based preprocessing frameworks, such as PurifyR, as well as additional scripts can be assembled together to standardize and automate a great deal of work normally left for bioinformaticians, which is also very error-prone and time-consuming to understand. Once automated, bioinformaticians are able to focus on more substantial work such as interpreting the semantics of the data, improving used methods, and interpreting the outcome. A suitable preprocessing layer can be reused and datasets reprocessed repeatedly in a consistent manner. Bioinformaticians can create a data preprocessing pipeline to first begin exploring data without a great deal of exploratory effort initially in removing data, which can be described and completed by a rule-based standardized tool. PurifyR focuses on providing this standardized and reproducible context for preprocessing workflows.

The PurifyR package can be installed from GitHub, see Supplementary Data S6, or a live Shiny implementation can be seen at https://purifyr.stratominer.com/ Shiny Users can call three specific functions to automate preprocessing steps. Predefined configuration values are prepared to ensure that datasets meet the assumptions of the following machine learning methods for use by other packages, PCA, regression, and others. Input data are an existing R data frame or data.table object from a file or other source. The package can be used to profile data and perform column health checks to recommend only useful features for the use of downstream analysis steps, for example, machine learning. Second, the package scrubs the healthy columns, from the previous step, and performs row-specific processing to ensure only high-quality records are included and missing or out-of range values are repaired. Finally, data are transformed and scaled to meet the requirements of matrix-based methods, such as PCA. Finally, the package performs postanalysis profiling to display per-column statistics, such as intravariable variance and correlation metrics, useful for evaluation before performing additional machine learning steps.

We tested this on five public datasets (Table 1 and Supplementary Data S4) and three HCS datasets (Supplementary Data S3).[32] The dataset with over 400K records and >200 features completes in ~0.5 min, generated on an AWS EC2 R5 instance with 4 cores and 32 GB RAM. PurifyR operates completely with the data.table package for optimized computation and minimized space required in memory and is using the package, parallel, for multicore usage. The data.table approach requires significant additional development effort, but demonstrates huge performance improvements, as shown in Figure 4.

Table 1. This Table Represents the Results of ScanR for the Following Datasets; Mtcars, Iris, Diamonds (ggplot2 Package), Baseball (plyr Package), and Flights (nycflights13 Package)

Dataset	Rows	Columns	Calculation time, sec
Mtcars	32	11	0.285
Iris	150	5	0.292
Diamonds	~53K	10	0.144
Baseball	~21K	22	0.339
Flights	~336K	19	0.522
HCS dataset I	~400K	49	2.647
HCS dataset II	~3.5mln	233	21.174
HCS dataset III	~251K	1787	771.472

In addition, three HCS datasets were used, one of them available through the Supplementary Data. The column Dataset describes the dataset, the column Rows describes the number of rows of the dataset, and the column Columns describes the number of columns in the dataset. The column Calculation Time describes the amount of time required to process the dataset with ScanR in PurifyR. Results are generated using an AWS R5 xlarge EC2 instance (4 cores and 32 GB RAM).

HCS, high-content screen.

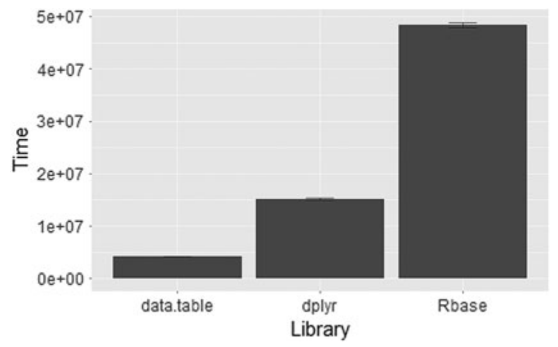

FIG. 4. This Figure demonstrates the speed of a standard procedure, calculating a column-wise mean, on the same public dataset *Baseball.* The calculation is carried out by the packages data.table, dplyr, and Rbase, respectively (see *x*-axis). The *y*-axis represents the time in nanoseconds. The calculation is measured a 100 times using the package microbenchmark where the bars visualize the mean and standard error of the measurements. The performance of data.table shows a 20-time speedup compared with Rbase and a 3.5-time speedup compared with dplyr. See Supplementary Data S5 for implementation.

It finally enables the calculation of statistics per unit of analysis provided by the function SmashR. This allows to calculate and visualize a mean and standard error or minimum, Q1, median, Q3, and maximum to visualize a boxplot or error bar for each unit of analysis in the dataset.

Discussion

The PurifyR package demonstrates the ability to reliably carry out on-demand preprocessing of large datasets without creation of multiple copies thanks to the package data.table. This allows researchers to confidently produce statistics and analyze imperfect datasets directly by running data through PurifyR. Researchers can build data processing pipelines during research and during quantification. PurifyR will clean datasets and highlight records and columns with missing data or outliers, which are not able to be used in downstream statistics and reports. A few public datasets from very small to moderate size such as *Mtcars*, *Iris*, *Baseball*, and *Flights* plus three large datasets were subjected to ScanR for feature selection. Supplementary Data S4 provides the code and processing outcome of processing the datasets using PurifyR. Table 1 reports the size and speed of processing them using the PurifyR package.

Complex datasets can require substantial time to compare results from previous experiments and represent potential roadblocks for further experimentation due to the excessive amount of time required for aggregating simple statistics on large and disparate datasets. Ideally, experimenters could check these statistics repeatedly during the research process to identify issues and correct them before the experiment is complete or it becomes too late to address any collection issues.

Large datasets ideally require a rule-based approach to review the large number of automatically generated records and columns. Without automation, the results are difficult to reproduce and frequently prone to reporting errors. Additionally, manual curation of these types of datasets often requires a great deal of time for cleansing and to standardize the values,[14] for example, to standardize data to common ranges in preparation for later statistical processing and machine learning steps. Missing data and outlier handling often prove to be complicated and subject to interpretation. This can consume up to 80% of the total time for analysis and reporting of results.[15] The PurifyR package aims to automate a great part of this time by applying feature synthesis and engineering methods to automate data preparation steps. The PurifyR package automates common preprocessing steps to ensure that high-quality features are used and each row is prepared to meet the assumptions of later machine learning processing steps. Not only does the package reduce the manual effort required and time to process data but it will also improve transparency and reproducibility of the final results.

Ultimately, a framework for automating data preprocessing steps would remove the need for repetitive efforts and complex code for each analysis. Unfortunately, there is no golden standard, but there are a few statistical rules of thumb and recommendations in specific domains and methods.[33,34] It would simplify analysis and comparison with previous research and ensure a simple explanation that can be seen by all researchers. Moreover, it would help reproducibility move a step forward.[35,36] Many researchers do not need or desire to be involved in the intricate details of cleaning data and would prefer a more autonomous approach, where best practice standards are applied without a great deal of intervention. Ideally, researchers could quickly compute and reanalyze datasets without manual cleaning effort between iterations.

Acknowledgment

The authors would like to thank Aaron Carmack for the useful contribution and discussions in this work.

Author Disclosure Statement

D.A.E. and W.A.O. are both cofounders of Core Life Analytics B.V., a company in data analytics software.

Funding Information

No funding was received for this article.

Supplementary Material

Supplementary Data S1
Supplementary Data S2
Supplementary Data S3
Supplementary Data S4
Supplementary Data S5
Supplementary Data S6

References

1. Macarron R, Banks MN, Bojanic D, et al. Impact of high-throughput screening in biomedical research. Nat Rev Drug Discov. 2011;10:188.
2. Sundberg SA. High-throughput and ultra-high-throughput screening: solution-and cell-based approaches. Curr Opin Biotechnol. 2000;11:47–53.
3. Bergström S, Ivarsson O. Automation of a Data Analysis Pipeline for High-Content Screening Data. Master thesis. Linköping University, Linköping, Sweden. 2015.
4. Billingsley KJ, Bandres-Ciga S, Saez-Atienzar S, et al. Genetic risk factors in Parkinson's disease. Cell Tissue Res. 2018;373:9–20.
5. Bajcsy P, Cardone A, Chalfoun J, et al. Survey statistics of automated segmentations applied to optical imaging of mammalian cells. BMC Bioinformatics. 2015;16:330.
6. Sullivan DP, Faeder JR, Rhode GK, et al. Image-derived generative modeling of complex cellular organization in both space and time. PhD thesis CMU-CB-15-102, Carnegie Mellon University, Pittsburgh, PA. 2015.
7. Banks P, Appledorn D, Shumate C, et al. Roundup: lights! Camera! Live-Cell Imaging!. Genet Eng Biotechnol N (GEN). 2017;37.
8. Esner M, Meyenhofer F, Bickle M. Live-cell high content screening in drug development. In: *High Content Screening*. (Johnston PA, Trask OJ; eds). Humana Press, New York, NY. 2018; pp. 149–164.
9. Kriston-Vizi J, Flotow H. Getting the whole picture: high content screening using three-dimensional cellular model systems and whole animal assays. Cytometry Part A. 2017;91:152–159.
10. Buchberger AR, DeLaney K, Johnson J, et al. Mass spectrometry imaging: a review of emerging advancements and future insights. Anal Chem. 2017;90:240–265.
11. Kraus OZ, Grys BT, Ba J, et al. Automated analysis of high-content microscopy data with deep learning. Mol Syst Biol. 2017;13:924.
12. Labuda D, Krajinovic M, Richer C, et al. Rapid detection of CYP1A1, CYP2D6, and NAT variants by multiplex polymerase chain reaction and allele-specific oligonucleotide assay. Anal Biochem. 1999;275:84–92.
13. Nickischer D, Elkin L, Cloutier N, et al. Challenges and opportunities in enabling high-throughput, miniaturized high content screening. In: *High Content Screening*. (Johnston PA, Trask OJ; eds). Humana Press, New York, NY. 2018; pp. 165–191.
14. Wickham H. Tidy data. J Stat Softw. 2014;59:1–23.
15. Stodder D. Why data preparation matters. In: *Data preparation challenges facing every enterprise*. (Dooley B; ed). Tdwi. 2017; pp. 5–11. Retrieved from TDWI e-book database.
16. Li S, Besson S, Blackburn C, et al. Metadata management for high content screening in OMERO. Methods. 2016;96:27–32.
17. Hoffman AF, Nolan J, Gebhard DF, et al. Society of biomolecular imaging and informatics high-content screening/high-content analysis emerging technologies in biological models, when and why?. Assay Drug Dev Technol. 2018;16:1–6.
18. Omta WA, Nobel JD, Klumperman J, et al. Improving comprehension efficiency of high content screening data through interactive visualizations. Assay Drug Dev Technol. 2017;15:247–256.
19. Thomas N. High-content screening: a decade of evolution. J Biomol Screen. 2010;15:1–9.
20. Bellomo F, Medina DL, De Leo E, et al. High-content drug screening for rare diseases. J Inherit Metab Dis. 2017;40:601–607.
21. Dinkla K, Strobelt H, Genest B, et al. Screenit: visual analysis of cellular screens. IEEE Trans Vis Comput Graph. 2017;23:591–600.
22. Piccinini F, Balassa T, Szkalisity A, et al. Advanced cell classifier: user-friendly machine-learning-based software for discovering phenotypes in high-content imaging data. Cell Syst. 2017;4:651–655.
23. Echeverri CJ, Perrimon N. High-throughput RNAi screening in cultured cells: a user's guide. Nat Rev Genet. 2006;7:373.
24. Joslin J, Gilligan J, Anderson P, et al. A fully automated high-throughput flow cytometry screening system enabling phenotypic drug discovery. SLAS Discov. 2018;23:697–707.
25. Sommer C, Hoefler R, Samwer M, et al. A deep learning and novelty detection framework for rapid phenotyping in high-content screening. Mol Biol Cell. 2017;28:3428–3436.
26. Bray MA, Carpenter A. Advanced assay development guidelines for image-based high content screening and analysis. In: *Assay Guidance Manual*. Eli Lilly & Company and the National Center for Advancing Translational Sciences, Bethesda, MD. 2017.
27. Li S. (2017). Evaluation and Improvement of Current Computational Tools for Metabolomics Data Analysis (Doctoral dissertation, Auckland University of Technology).
28. Danovi D, Folarin AA, Baranowski B, et al. High content screening of defined chemical libraries using normal and glioma-derived neural stem cell lines. In: *Methods in Enzymology,* vol. 506. (Conn PM; ed). Academic Press, San Diego, CA. 2012; pp. 311–329.
29. Bougen-Zhukov N, Loh SY, Lee HK, et al. Large-scale image-based screening and profiling of cellular phenotypes. Cytometry Part A. 2017; 91:115–125.
30. Boutros M, Heigwer F, Laufer C. Microscopy-based high-content screening. Cell. 2015;163:1314–1325.
31. Van Buuren S. MICE: Multiple Imputation by Chained Equations. R Package Version 2.21. 2014.
32. van Heesbeen RG, Raaijmakers JA, Tanenbaum ME, et al. Aurora A, MCAK, and Kif18b promote Eg5-independent spindle formation. Chromosoma. 2017;126:473–486.
33. Birmingham A, Selfors LM, Forster T, et al. Statistical methods for analysis of high-throughput RNA interference screens. Nat Methods. 2009;6:569.
34. Costello AB, Osborne JW. Best practices in exploratory factor analysis: four recommendations for getting the most from your analysis. Pract Assess Res Eval. 2005;10:1–9.
35. Weigt D, Sammour DA, Ulrich T, et al. Automated analysis of lipid drug-response markers by combined fast and high-resolution whole cell MALDI mass spectrometry biotyping. Sci Rep. 2018;8: 11260.
36. Szymańska E, Brown PA, Ziere A, et al. Comprehensive data scientific procedure for enhanced analysis and interpretation of real-time breath measurements in in vivo aroma-release studies. Anal Chem. 2015;87: 10338–10345.

Cite this article as: Omta WA, van Heesbeen RG, Shen I, Feelders AJ, Brinkhuis MJS, Egan DA, Spruit MR (2020) PurifyR: an R package for highly automated, reproducible variable extraction and standardization, *Systems Medicine* 3:1, 1–7, DOI: 10.1089/sysm.2010.0007.

Abbreviations Used

HCS = high-content screen
PCA = principle component analysis