GranatumX: A community engaging, modularized and flexible software environment for single-cell analysis

David Garmire[1#] ,Xun Zhu[2#], Aravind Mantravadi[3], Qianhui Huang[3], Breck Yunits[2], Yu Liu[2] , Thomas Wolfgruber[1], Olivier Poirion[1], Tianying Zhao[4], Cédric Arisdakessian[1,4], Stefan Stanojevic[2], Lana X. Garmire[2 *]

[1]Department of Electrical & Computer Engineering, University of Michigan, Ann Arbor, 48105, USA.

[2]Epidemiology Program, University of Hawaii Cancer Center, Honolulu, HI 96813, USA.

[2]Department of Computational Medicine and Bioinformatics, University of Michigan, Ann Arbor, 48105, USA.

[3]Department of Biostatistics, University of Michigan, Ann Arbor, 48109, USA.

[4]Molecular Biosciences and Bioengineering Graduate Program, University of Hawaii at Manoa, Honolulu, HI 96822, USA.

[#] These authors contributed equally to the work.

[*] To whom correspondence should be addressed. Email address: lgarmire@med.umich.edu

Abstract

We present GranatumX, a next-generation software environment for single-cell data analysis. GranatumX is inspired by the interactive web tool Granatum. It enables biologists to access the latest single-cell bioinformatics methods in a web-based graphical environment. It also offers software developers the opportunity to rapidly promote their own tools with others in customizable pipelines. The architecture of GranatumX allows for easy inclusion of plugin modules, named Gboxes, that wrap around bioinformatics tools written in various programming languages and on various platforms. GranatumX can be run on the cloud or private servers, and generate reproducible results. It is a community-engaging, flexible, and evolving software ecosystem for scRNA-Seq analysis, connecting developers with bench scientists. GranatumX is freely accessible at http://garmiregroup.org/granatumx/app.

Keywords: single cell, RNA-Seq, analysis, software, pipeline, plugin

Introduction

Single-cell RNA sequencing (scRNA-seq) technologies have advanced our understanding of cell-level biology significantly [1]. Many exciting scientific discoveries are attributed to new experimental technologies and sophisticated computational methods [2,3]. Despite the progress in cultivating professionals with cross-discipline training, a gap continues to exist between the wet-lab biology and the bioinformatics community. Moreover, with the rapid development of many varieties of modules handling different parts of scRNA-seq analysis [4–6], it becomes increasingly challenging for bioinformaticians themselves to decide which method to choose. Although some analytical packages such as SINCERA [7], Seurat [8], and Scanpy [9] provide complete scRNA-seq pipelines, they require users to be familiar with their corresponding programming language (typically R or Python), installation platform, and command-line interface. This overhead hinders wide adoption by experimental biologists, especially those newly adopting scRNA-seq technologies. A few platforms, such as ASAP [10] and our own previous tool Granatum [11], provide intuitive graphical user interfaces and may be useful for a first-hand exploratory check. However, Granatum does not allow for modularity, while ASAP lacks flexibility and restricts the user to a set number of computational tools. Here we present GranatumX, the new generation of scRNA-seq analysis platforms that aims to solve these issues systematically. Its architecture facilitates the rapid incorporation of cutting-edge tools and enables the efficient handling of large datasets aided by virtualization [12].

Results

Overview of GranatumX

The objective of GranatumX is to provide scRNA-seq biologists better access to bioinformatics tools and the ability to conduct single-cell data analysis independently (**Figure 1**). Currently other scRNA-seq platforms usually only provide a fixed set of methods implemented by the authors themselves. It is difficult to add new methods developed by the community due to programming language lock-in as well as monolithic code architectures. If a pipeline is assembled between heterogeneous tools, it is manually crafted and inhibits a repeatable execution of data analysis tools by other wet-lab scientists. As a solution, GranatumX uses the plugin and virtualized framework that provides an easy and unified approach to add new methods in a data-analysis pipeline. The plugin system is agnostic to developer code and the choice of the original scripting language. It also eliminates inter-module incompatibilities, by isolating the dependencies of each module (**Figure 2A**). As a data portal, GranatumX provides a graphical user interface (GUI) that requires no programming experience.

Deployment of GranatumX

The web-based GUI can be accessed on various devices including desktop, tablets, and smartphones (**Figure 2A**). In addition to the web-based format, GranatumX is also deployable on a broad variety of computational environments, such as private PCs, cloud services, and High-Performance Computing (HPC) platforms with minimal effort by system administrators. The deployment process is unified on all platforms because all components of GranatumX are containerized in Docker [13] (also portable to Singularity [14]). GranatumX can handle larger-scale scRNA-seq datasets coming online, with an adequate cloud configuration setup and appropriate Gboxes. For example, after uploading data, it took GranatumX ~12 minutes to finish the recommended pipeline on an AMD 3950x with 16 cores and 128GB of DRAM memory running Ubuntu 20.04, using 10K cells downsampled from the dataset of “1.3 Million Brain Cells from E18 Mice" on the 10x Genomics website. The most time-consuming step is imputation using DeepImpute (~⅖ time), and the detailed breakdown of time consumption is shown in **Supplementary Table 1.**

Unique Gbox modules

Gbox is a unique concept of GrantumX. It represents a containerized version of a scientific package that handles its input and output by a format understood by the GranatumX core (**Figure 2B**). GranatumX has a set of pre-installable Gboxes that enable complete scRNA-seq analysis out of the box. Various Gboxes for data entry, preprocessing, and processing can be customized and organized together, to form a complete analysis pipeline (**Figure 2C)**. One highlight feature of the Gbox is that it stands alone and the user can assume any Gbox without the need to restart the full pipeline, in case one implemented by the user fails. Another highlight of the Gbox feature is that the entire GranatumX platform is fully interactive, with addition or removal of some Gboxes or parameter changes on the go, while some other Gboxes are being executed.

A comprehensive set of over 30 Gboxes are implemented in GranatumX to perform tasks all the way from data entry and processing to downstream functional analysis. The data processing tasks help to minimize the biases in the data and increase the signal-to-noise ratio. For each of these quality improvement categories, GranatumX provides multiple popular methods from which users can pick. To assist functional analysis, GranatumX provides a core list of methods for dimension reduction, visualization (including PCA, t-SNE, and UMAP), clustering, differential expression, marker gene identification, Gene Set Enrichment Analysis, network analysis and pseudo-time construction. Versioning for each of these Gboxes has been implemented so that users can use a specific tested version of a Gbox. Developers on the other hand can work on newer versions separately before the official upgrade. Gboxes can be stored on DockerHub for public use which maintains its own versioning system (https://hub.docker.com/u/granatumx). We provide templates and tutorials for writing and building Gboxes in **Supplementary File 1**.

Input files

The input files of GranatumX include expression matrices and optional sample metadata tables, acceptable in a variety of formats such as CSV, TSV, or Excel format. GranatumX even accepts zip files and gz files (GNU zip), and the user can choose that format for large expression matrices. Expression matrices are raw read counts for all genes (rows) in all cells (columns). The sample metadata tables annotate each cell with a pre-assigned cell type, state, or other quality information. The parsing step creates a sparse matrix using the coordinate list (COO) format, and this representation ensures swift upload onto the back end, even for large input datasets (>10K cells). Such information will either be used to generate computational results (such as Gene Set Analysis) or be mapped onto the PCA, t-SNE, or UMAP plot for visualization (see **Figure 2C** for the workflow). Once the user uploads the gene expression matrix, the data are read into a dataframe using *Pandas* and the step updates the user with a "preview", consisting of the first few rows and columns of the gene expression matrix, along with the number of genes and samples present.

User-centric design

As a user-friendly tool, GranatumX allows multiple users to be affiliated with the same project for data and result sharing, while restricting one user to run the pipeline at a time to avoid data conflicts. It allows dynamically adding, removing, and reordering pipeline steps on the go. It also allows users to reset the current step. All relevant data in the analysis pipeline and all results generated by each module are stored in a database, allowing users to access and download them. To ensure reproducibility, GranatumX can automatically generate a human-readable report detailing the inputs, running arguments, and the results of all steps (see examples in **Supplementary File 2** and **Supplementary File 3**). All of these features are designed with the mindset of “consumer reports" to facilitate research in experimental labs or genomics cores.

Case studies using GranatumX

In the following section, we demonstrate two case studies of GranatumX. The first data set was downloaded from GSE117988, including 7431 single cells generated by the 10x Genomics 3’ Chromium platform. It was obtained from a patient with metastatic Merkel cell carcinoma, treated using T cell immunotherapy as well as immune-checkpoint inhibitors (anti-PD1 and anti-CTLA4) but later developed resistance [15]. We used a customized pipeline to analyze the scRNA-seq data (**Figure 3A**). The pipeline comprises all common analysis steps, including 1) File upload, 2) imputation (based on DeepImpute [6]), 3) normalization, 4) gene filtering, 5) log transformation, 6) principal component analysis (PCA), 7) t-SNE/UMAP plot, 8) sample coloring, 9) clustering, 10) marker gene identification, 11) GSEA analysis, and 12) pseudo-time construction. The analysis report of the entire pipeline is included as **Supplementary File 2.** The clustering step identifies 7 clusters on the UMAP plot (**Figure 3B**). The exemplary GSEA analysis results (**Table 1)** show the significance in many important immune-related pathways, including the MAPK signaling pathway and antigen processing and presentation pathway (cluster_5 vs. rest), cell cycle genes (cluster_3 vs. rest), and ubiquitin-mediated proteolysis (cluster_3 vs. rest).

We also used GranatumX to analyze Tabula Muris dataset, which contains 54,865 cells from 20 organs and tissues of mouse [16]. Again we used the same pipeline as shown in **Figure 3A**. GranatumX offers multiple popular clustering algorithms, and for this dataset we used the Louvain algorithm. For illustration purposes, we focus on the viewing and clustering of this large graph-based clustering method implemented by Scanpy. A total of 44 clusters are assigned on the UMAP plot (**Figure 3C**). We also superimposed the metadata that contain tissue types for each cell on the same plot for visualization (**Supplementary Figure 1**). The complete analysis report of this dataset is included as **Supplementary File 3.**

Discussion

With the ever-increasing popularity of scRNA-seq, more and more experimental biologists will adopt this technology. At the same time, new bioinformatics tools are being developed rapidly. The development of GranatumX fills in a unique niche that enables both scientific and technical advancements. It is a "common ground" that connects scRNA-seq tool developers with the end-users, together for new discoveries. Domain experts can use GranatumX for the initial exploratory analysis. Additionally, with more Gboxes to be implemented on model performance metrics, GranatumX could be a vessel to enable benchmark studies to compare existing computational modules and pipelines, as well as assess the performance of a new method or pipeline relative to the existing ones. Moreover, it can also serve as the test engine to probe the source of variations in different modules, so as to optimize a pipeline for given datasets.

To demonstrate the uniqueness of GranatumX, we also compare it with other similar tools for comprehensive scRNA-seq analysis, such as SC1[17], ASAP [10] and Single Cell Explorer [18] in **Table 2.** While all these tools aim for simple report and interaction with biologists, GranatumX is the only framework that supports bioinformatics developers to contribute their own plug-ins (**Table 2**). This significantly enhances the adaptability of GranatumX among the developer community.

The web-tool that is closest to GranatumX is Single Cell Explorer. Both of them are Python-based web server applications, and they enable computational and experimental scientists to iteratively work together. However, they have significant differences in their functionalities. Single Cell Explorer begins from raw data processing including reading mapping alignment. GranatumX is a much lighter-weight tool that starts from a cell read count table. We chose this approach given the fact that the 10x genomics platform is very popular and the users can get the cell count table from CellRanger directly. Also, mapping alignment is a time-consuming process, and it is better to be separated from processes with much smaller time scales, such as the web-based GUI server. Single Cell Explorer pays a lot of attention to cell type exploration and annotation, but not for other downstream analyses, such as gene enrichment analysis, protein-protein interaction and pseudo-time construction; whereas GranatumX offers all these downstream functions (**Table 2**).

For ASAP, besides lacking modules to perform functions such as imputation, protein-protein interaction and pseudo-time construction, it also does not allow reconfiguration of the pipeline, a feature that is unique to Single Cell Explorer and GranatumX. For SC1, it lacks the flexibility and functionalities similar to ASAP; moreover, it is implemented in the R programming language, with an interactive web-based front-end built using the Shiny framework, which was restrictive to the R language. Stepping away from restrictions from R programming and its Shiny framework, as was done in Granatum as well, was one of the motivations for us to develop GranatumX. We envisioned a containerized system that can run R, Python, or other languages.

As an inclusive and open software environment that employs other third-party tools, GranatumX has some challenges. One of them is handling the upgrade of underlying 3rd party libraries and resources. Accompanying with updated 3rd party tools which may not be tested extensively by the original developers, errors from these packages may propagate into GranatumX. To deal with this issue, we implement versioning through the use of Docker which helps to maintain system-level dependencies as well as software dependencies in a complete package. The Gbox Docker containers for this release are listed in the **Supplementary Table 2** with a version number of 1.0.0. New versions can update the minor and major revision numbers so that users know exactly which code is being executed for a given pipeline.

The source code for the Docker containers which represent the Gboxes are stored in the corresponding GitHub repositories. For example, https://hub.docker.com/r/granatumx/gbox-differentialexpression is stored in https://github.com/granatumx/gbox-differentialexpression. Such an endeavour provides safety and reliability in maintaining the stability of the software not just in the source of the software but in the configuration of the system required to run the computational elements. Due to its openness, GranatumX can not prevent p-hacking or manipulating data analysis to improve the statistical significance of the desired result [19]. One way to discourage p-hacking is to suggest using standard pipeline and default parameters. If the user chooses values other than defaults, the reproducible design of GranatumX allows one to compare the outputs from the users (if they are recorded) with those from the default setting.

Commited to make GranatumX an open, shareable, and evolvable software environment, we hope to engage the community through social media (eg. a new GranatumX-Developer slack group) as well as Human Cell Atlas Data Portal Registry, to continuously improve this tool for the benefit of single-cell research.

Methods

Architectural overview

GranatumX consists of three independent components: Central Data Storage (CDS), User Interface (UI), and Task Runner (TR). CDS stores all data and metadata in GranatumX, including the uploaded files, processed intermediate data, and final results. The other two components of GranatumX both have controlled access to the central data storage, which allows them to communicate with each other. CDS is implemented using a PostgreSQL database and a secure file system based data warehouse. UI is the component with which wet-lab biologists interact. The layout is intuitive with Gbox settings while providing a flexible and customizable analysis pipeline. UI also allows for the asynchronous submission of tasks before they can be run by the back-end. UI is implemented using JavaScript, with the ReactJS framework. The submitted jobs queue up in the database and can be retrieved in real-time by TR. TR monitors the task queue in the CDS in real-time, actively retrieves the high-priority tasks (based on submission time), initializes the corresponding Gboxes, and prepares the input data by retrieving relevant data from CDS.

Deployment

GranatumX uses Docker to ensure that all Gboxes can be installed reproducibly with all their dependencies. As a result, GranatumX can be deployed in various environments including personal computers, dedicated servers, High-Performance Computing (HPC) platforms, and cloud services. The installation instructions are detailed in the README file of the source code.

Responsive UI

The web-based UI offers different device-specific layouts to suit a wider range of screen sizes. On Desktop computers, the UI takes advantage of the screen space and uses a panel-based layout, and maximizes the on-screen information. On small tablets and mobile devices with limited screen space, a collapsible sidebar-based layout is used to allow the most important information (the results of the current step) to show up on the screen.

Recipe system

Most studies can use similarly structured pipelines, which typically consist of data entry (upload and parsing), data pre-processing (imputation, filtering, normalization, etc.), and finally data analysis functionalities (clustering, differential expression, pseudo-time construction, network analysis, etc.). GranatumX allows users to save a given pipeline into a “recipe” for the future. GranatumX comes with a set of built-in recipes, which cover many of the most common experiment pipelines.

Software Development Kits (SDKs)

GranaumX SDKs are made for Python and R. These SDKs provide a set of Application Programming Interfaces (APIs) and helper functions that connect Gbox developer's own code with the core of GranatumX The detailed documentation can be found in the Github repository.

There are three steps to build a new Gbox from the existing code: 1) Write an entry point in the language of the developer's choice. The entry point uses the SDK to retrieve necessary input from the core of GranatumX and send back output to the core after the results are computed. 2) Package the entry point, the original package source code, and any dependencies into a docker image using a Dockerfile and the "docker build" command. 3) Write a UI specification for the Gbox. The specification is a simple YAML file that declares the data requirements of the Gbox.

Pipeline customization

GranatumX allows for full customization of the analysis pipeline. An analysis pipeline has a number of Gboxes organized in a series of steps. Note that two different steps can have the same underlying Gbox. For example, two PCA Gboxes can appear before and after imputation, to evaluate its effect. Because the data are usually processed in a streamlined fashion, later steps in the pipeline usually depend on data generated by the earlier steps. Steps can be added from the app-store into the current project and can be removed from the pipeline at any time. A newly added step can be inserted at any point in the pipeline and can be reordered in any way, as long as such re-arrangement does not violate the dependency relationships.

Current GranatumX cloud server setup

The current GranatumX web version is hosted on OVHcloud, with specs: Intel Haswell vCPU 128 GB RAM Xeon E5-1650 4GHz. Additionally the https protocol is verified with Let's Encrypt (https://letsencrypt.org) with an Apache 2 server (https://httpd.apache.org/) and a site registered with No-IP (https://www.noip.com/). This server uses a proxy implementation to pass a user to the Node.js web service. In this manner, Node does not have to manage the security or https connections which allows setup to occur efficiently in an enterprise system. Additionally, an optional fast compute system may be connected to the OVH cloud server through ssh tunneling which allows the local port to be mapped to the remote connection. In this manner, a high speed rig can be connected -- in this case, the AMD 3590x can be connected without having to procure a new cloud system.

Project management

The studies in GranatumX are organized as projects. Each user can manage multiple concurrent projects. The automatic customer's report can be generated per project using the parameters and results stored in the CDS.

Example data sets

Three datasets are used in this report. One data set is downloaded from GSE117988, a study on a patient with metastatic Merkel cell carcinoma, treated using T cell immunotherapy as well as immune-checkpoint inhibitors (anti-PD1 and anti-CTLA4) but later developed resistance [15]. A second dataset is Tabula Muris dataset, which contains 54,865 cells from 20 organs and tissues of the mouse [16]. Another dataset is the 1.3 Million Brain Cells from E18 Mice, downloaded from 10x genomics website: https://support.10xgenomics.com/single-cell-gene-expression/datasets/1.3.0/1M_neurons (accessed on date 05/09/2020). This dataset contains 1,308,421 cells from embryonic mice brains, done by Chromium™ Single Cell 3' Solution (v2 Chemistry).

GranatumX Plug-in development

The detailed tutorial for writing GranatumX plug-in is described in **Supplementary File 1**, as well as at the github page of the source code for GranatumX: https://github.com/granatumx. Additionally, we created a slack group named "GranatumX-Developer" to facilitate plug-in development from the 3rd party.

Code availability

The webtool of GranatumX can be found at http://garmiregroup.org/granatumx/app. The source code for GranatumX is available at https://github.com/granatumx under MIT license. All builds are deployed via Docker Hub at https://hub.docker.com/u/granatumx.

Copyright

Some of the cartoon icons in Figure 1 and Figure 2 are downloaded from https://www.flaticon.com/.

Acknowledgment

We thank Dr. Larry Reiter for providing valuable feedback while using GrantumX to test his single cell dataset. This research was supported by grants K01ES025434 awarded by NIEHS through funds provided by the trans-NIH Big Data to Knowledge (BD2K) initiative (www.bd2k.nih.gov), P20 COBRE GM103457 awarded by NIH/NIGMS, R01 LM012373 awarded by NLM, R01 HD084633 awarded by NICHD to L.X. Garmire.

Figure legends

Figure 1: Overview of the Granatum X platform. Granatum X aims to bridge the gap between the computational method developers (the bioinformaticians) and the experiment designers (the biologists). It achieves this by building end-to-end infrastructure including the packaging and containerization of the code (**Gbox Packaging**), organization and indexing of the Gboxes (**Apps**), customization of the analysis steps (**Pipeline building**), visualization and results downloading (**Interactive Analysis**), and finally the aggregation and summarization of the study (**Report Generation**).

Figure 2: A) Due to its heavy usage of dependency locking and containerization, Granatum X can be deployed on various computational environments, from personal computers, private servers, High-

Performance Computation systems, to cloud services. Granatum X's web UI is adaptable to devices with various screen sizes, which allows desktop and mobile access. B) Granatum X's data management. Each Gbox (labeled by a particular color to represent a certain functionality) with order dependency on the pipeline, may take some project data and some user-specified parameters as input and may generate results (interactive visualization, plots, tables, or even plain text) and new project data. All project data and results, as well as the specified parameters, are recorded and saved into the central data storage and can be used for reproducibility control. C) A scRNA-Seq computational study typically consists of three phases: the uploading and parsing of the expression matrices and metadata (Data Entry), the quality improvement and signal extraction of the data (Data Processing), and finally the assorted analyses on the processed data which offer biological insights (Data Analysis).

Figure 3: A) An exemplary workflow of a customized scRNA-seq pipeline set by the user. B) The UMAP plot clusters on Merkel cell carcinoma data from the 10x genomics platform. C) The UMAP plot clusters of Tabula Muris Consortium data.

Table 1. Gene Set Enrichment Analysis results on clusters from UMAP plot in Figure 3B.

Comparing pairs	**Gene Set Name**	**Gene Set Size**	**Normalized Enrichment Score**	**P Value**	**False Detection Rate**
Cluster 2 vs. Rest	KEGG Glycolysis Gluconeogenes is	13	4.23	0	0
	KEGG Pathogenic Escherichia Coli Infection	14	3.63	0	0
	KEGG Alzheimers Disease	20	3.97	0	0

	KEGG Tight Junction	16	3.03	0.004	0.0063
Cluster 3 vs. Rest	KEGG Oocyte Meiosis	15	3.75	0	0
	KEGG Pathogenic Escherichia Coli Infection	14	3.48	0.001	0.0315
	KEGG Cell Cycle	20	3.13	0.002	0.042
	KEGG Ubiquitin Mediated Proteolysis	12	3.29	0.005	0.0787
Cluster 4 vs. Rest	KEGG Spliceosome	13	3.84	0	0
	KEGG Viral Myocarditis	25	3.23	0.002	0.042
Cluster 5 vs. Rest	KEGG Alzheimers Disease	20	3.08	0	0
	KEGG Antigen Processing and Presentation	30	2.51	0.003	0.0472
	KEGG MAPK Signaling Pathway	45	2.91	0	0
	KEGG Glycolysis Gluconeogenesis	13	2.49	0.043	0.198
	KEGG Spliceosome	13	2.99	0.004	0.0504

Table 2. Comparison on multiple user-friendly web-tools.

Platform	**SC1**	**ASAP**	**Single-Cell Explorer**	**GranatumX**
Simple report and interactivity for biologists	Yes	Yes	Yes	Yes

Configurable* Pipeline	No	No	Yes	Yes
Supports computational developers to plug in their own containers	No	No	No	Yes
Programming languages allowed in plug ins	NA	NA	NA	Multiple languages (eg. Python, R)
Default pipeline supports imputation	No	No	No	Yes
Default pipeline supports pseudo-time analysis	No	No	No	Yes
Supports protein-protein interaction network	No	No	No	Yes

Note: *configurable refers to the ability to customize the analytical steps and orders.

Supplementary Materials

Supplementary Figure 1. UMAP plot with annotated tissue types in the Tabula Muris Consortium data.

Supplementary Table 1. The running time for different steps of the recommended pipeline in GranatumX over 10,000 cells, using an AMD 3950X processor, 16 cores, with 128 GB of RAM.

Supplementary Table 2. The list of currently implemented Gboxes.

Supplementary File 1: The instructions to create a Gbox.

Supplementary File 2: The analysis report using a dataset with metastatic Merkel cell carcinoma from the 10x genomics platform.

Supplementary File 3: The analysis report using a dataset with Tabula Muris Consortium data.

References

1. Saliba A-E, Westermann AJ, Gorski SA, Vogel J. Single-cell RNA-seq: advances and future challenges. Nucleic Acids Res. academic.oup.com; 2014;42:8845–60.

2. Zappia L, Phipson B, Oshlack A. Exploring the single-cell RNA-seq analysis landscape with the scRNA-tools database. PLoS Comput Biol. journals.plos.org; 2018;14:e1006245.

3. Svensson V, Vento-Tormo R, Teichmann SA. Exponential scaling of single-cell RNA-seq in the past decade. Nat Protoc. nature.com; 2018;13:599–604.

4. Huang Q, Liu Y, Du Y, Garmire L. Evaluation of Cell Type Deconvolution R Packages on Single Cell RNA-seq Data [Internet]. Available from: http://dx.doi.org/10.1101/827139

5. Poirion OB, Zhu X, Ching T, Garmire L. Single-cell transcriptomics bioinformatics and computational challenges. Front Genet. Frontiers; 2016;7:163.

6. Arisdakessian C, Poirion O, Yunits B, Zhu X, Garmire LX. DeepImpute: an accurate, fast and scalable deep neural network method to impute single-cell RNA-Seq data [Internet]. bioRxiv. 2018 [cited 2019 Sep 30]. p. 353607. Available from: https://www.biorxiv.org/content/10.1101/353607v1

7. Guo M, Wang H, Potter SS, Whitsett JA, Xu Y. SINCERA: a pipeline for single-cell RNA-Seq profiling analysis. PLoS Comput Biol. Public Library of Science; 2015;11:e1004575.

8. Butler A, Hoffman P, Smibert P, Papalexi E, Satija R. Integrating single-cell transcriptomic data across different conditions, technologies, and species. Nat Biotechnol. nature.com; 2018;36:411–20.

9. Wolf FA, Angerer P, Theis FJ. SCANPY: large-scale single-cell gene expression data analysis. Genome Biol. 2018;19:15.

10. David FPA, Litovchenko M, Deplancke B, Gardeux V. ASAP 2020 update: an open, scalable and interactive web-based portal for (Single-cell) omics analyses. *Nucleic Acids Research*. 2020;48(W1):W403-W414. doi:10.1093/nar/gkaa412

11. Zhu X, Wolfgruber TK, Tasato A, Arisdakessian C, Garmire DG, Garmire LX. Granatum: a graphical single-cell RNA-Seq analysis pipeline for genomics scientists. Genome Med. BioMed Central; 2017;9:108.

12. Felter W, Ferreira A, Rajamony R, Rubio J. An updated performance comparison of virtual machines and Linux containers [Internet]. 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS). 2015. Available from: http://dx.doi.org/10.1109/ispass.2015.7095802

13. Merkel D. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux J [Internet]. Houston, TX: Belltown Media; 2014;2014. Available from: http://dl.acm.org/citation.cfm?id=2600239.2600241

14. Kurtzer GM, Sochat V, Bauer MW. Singularity: Scientific containers for mobility of compute. PLoS One. journals.plos.org; 2017;12:e0177459.

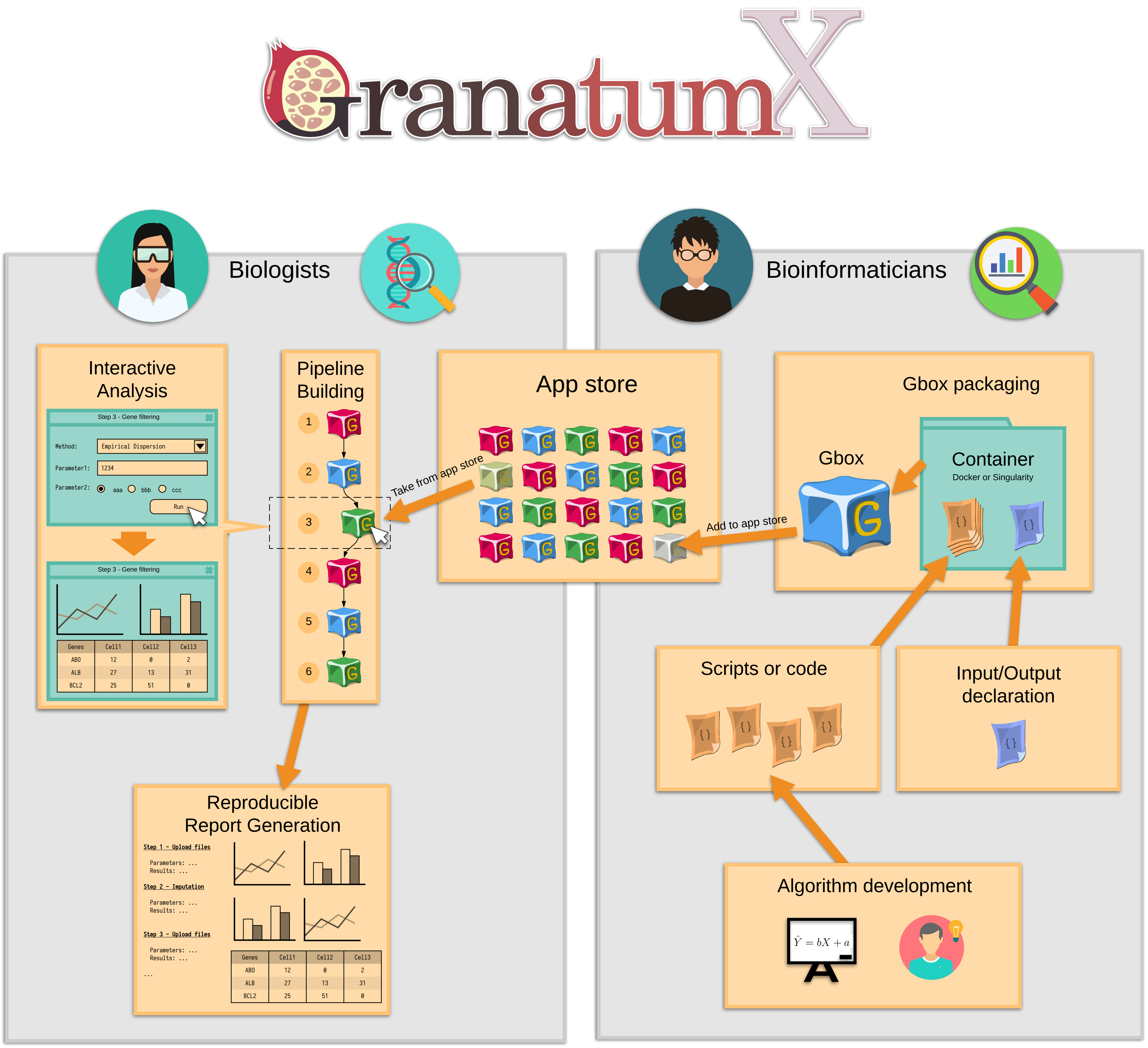
15. Paulson KG, Voillet V, McAfee MS, Hunter DS, Wagener FD, Perdicchio M, et al. Acquired cancer resistance to combination immunotherapy from transcriptional loss of class I HLA. Nat Commun. 2018;9:3868.

16. Tabula Muris Consortium, Overall coordination, Logistical coordination, Organ collection and processing, Library preparation and sequencing, Computational data analysis, et al. Single-cell transcriptomics of 20 mouse organs creates a Tabula Muris. Nature. 2018;562:367–72.

17. Moussa M, Mandoiu II. SC1: A web-based single-cell RNA-seq analysis pipeline [Internet]. 2018 IEEE 8th International Conference on Computational Advances in Bio and Medical Sciences (ICCABS). 2018. Available from: http://dx.doi.org/10.1109/iccabs.2018.8542088

18. Feng D, Whitehurst CE, Shan D, Hill JD, Yue YG. Single Cell Explorer, collaboration-driven tools to leverage large-scale single-cell RNA-seq data. BMC Genomics. 2019;20:676.m

19. P-hacking. (2020). In *Oxford Online Dictionary*. Retrieved from https://en.oxforddictionaries.com/definition/p-hacking



Step 3 - Gene filtering

Method: Empirical Dispersion

Parameter1: 1234

Parameter2: [✓ aaa] bbb ccc

Genes	Cell1	Cell2	Cell3
ABO	12	0	2
ALB	27	13	31
BCL2	25	51	0

Step 1 - Upload files

Parameters: ...
Results: ...

Step 2 - Imputation

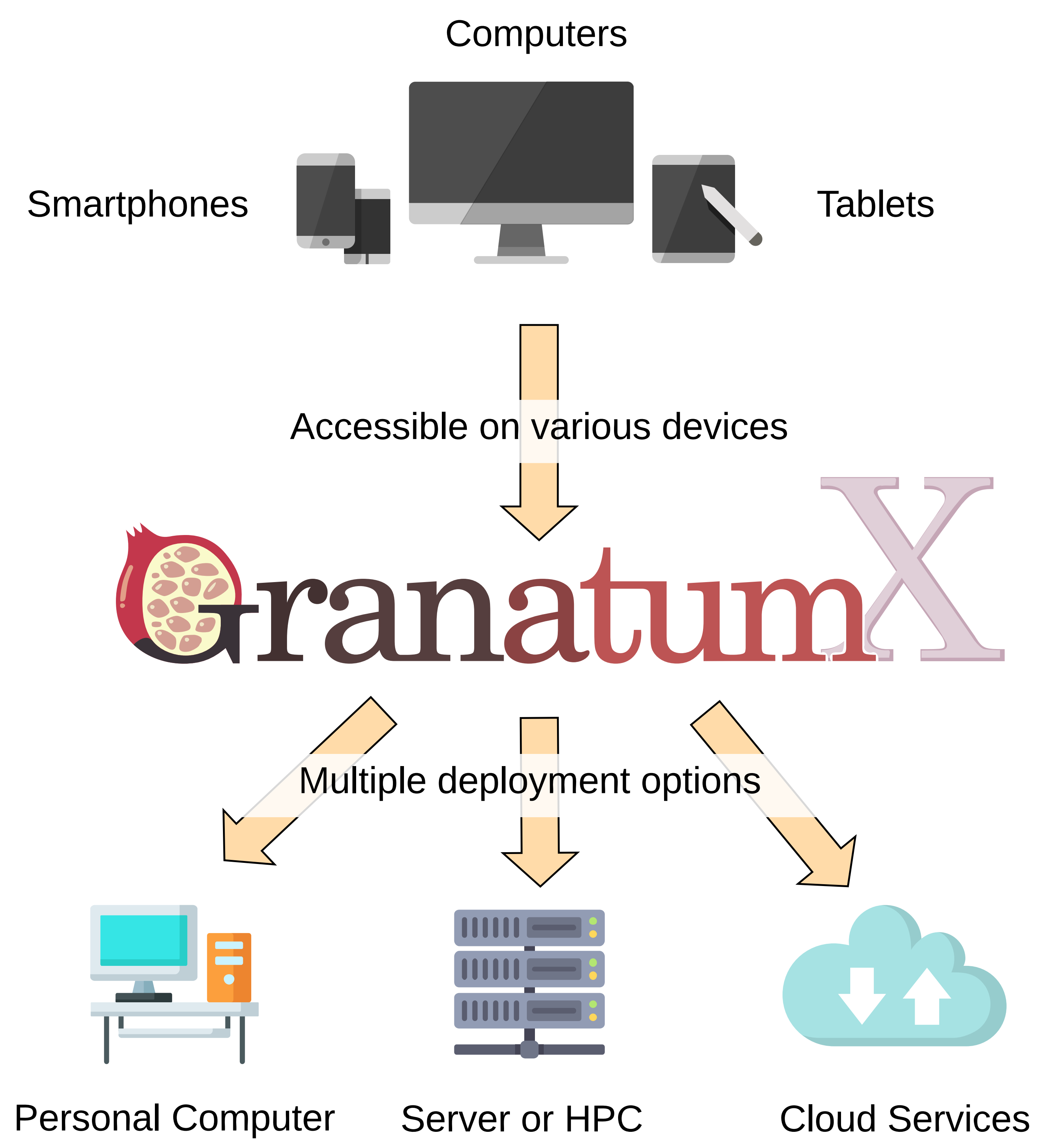
Parameters: ...
Results: ...

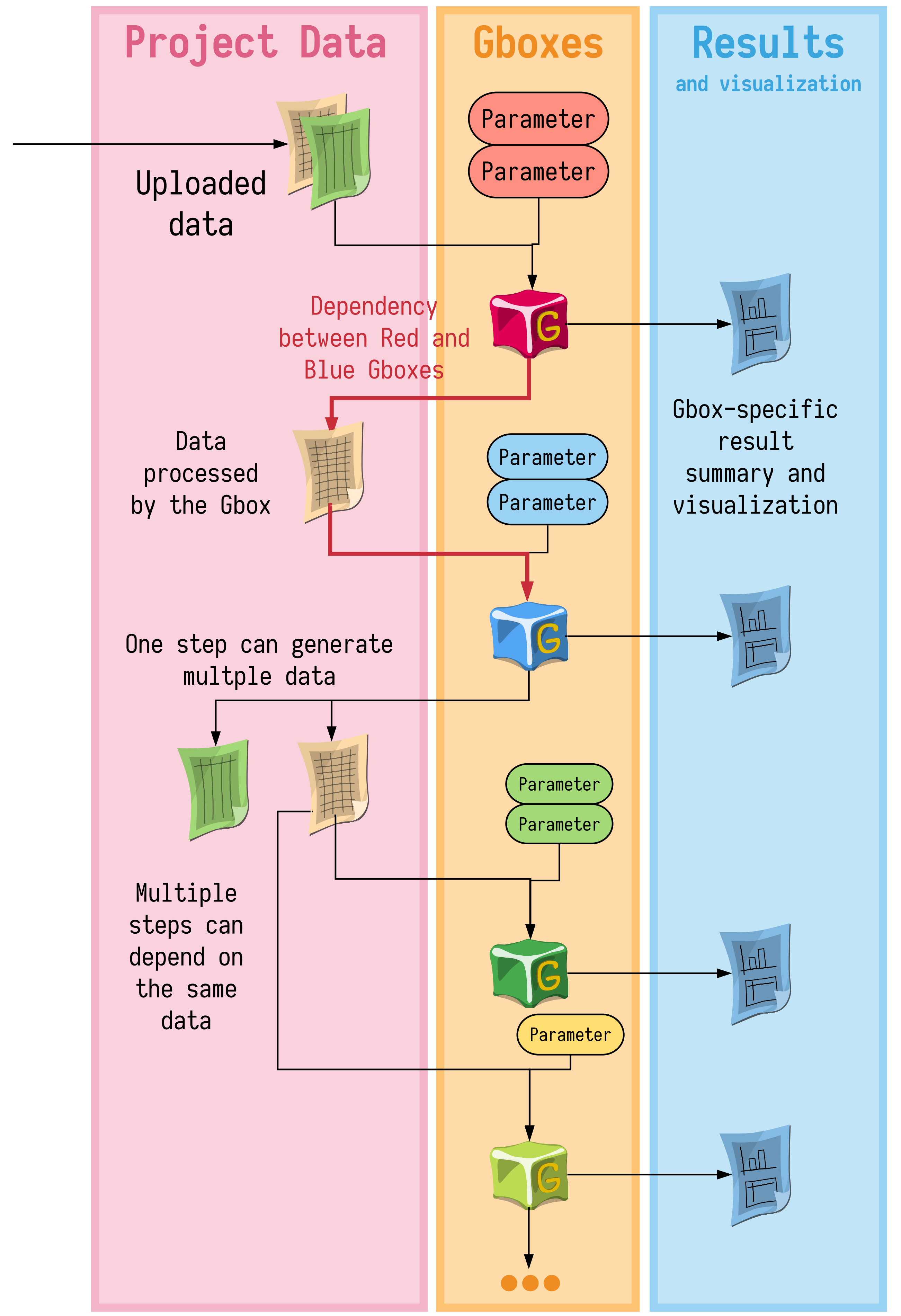
Step 3 - Upload files

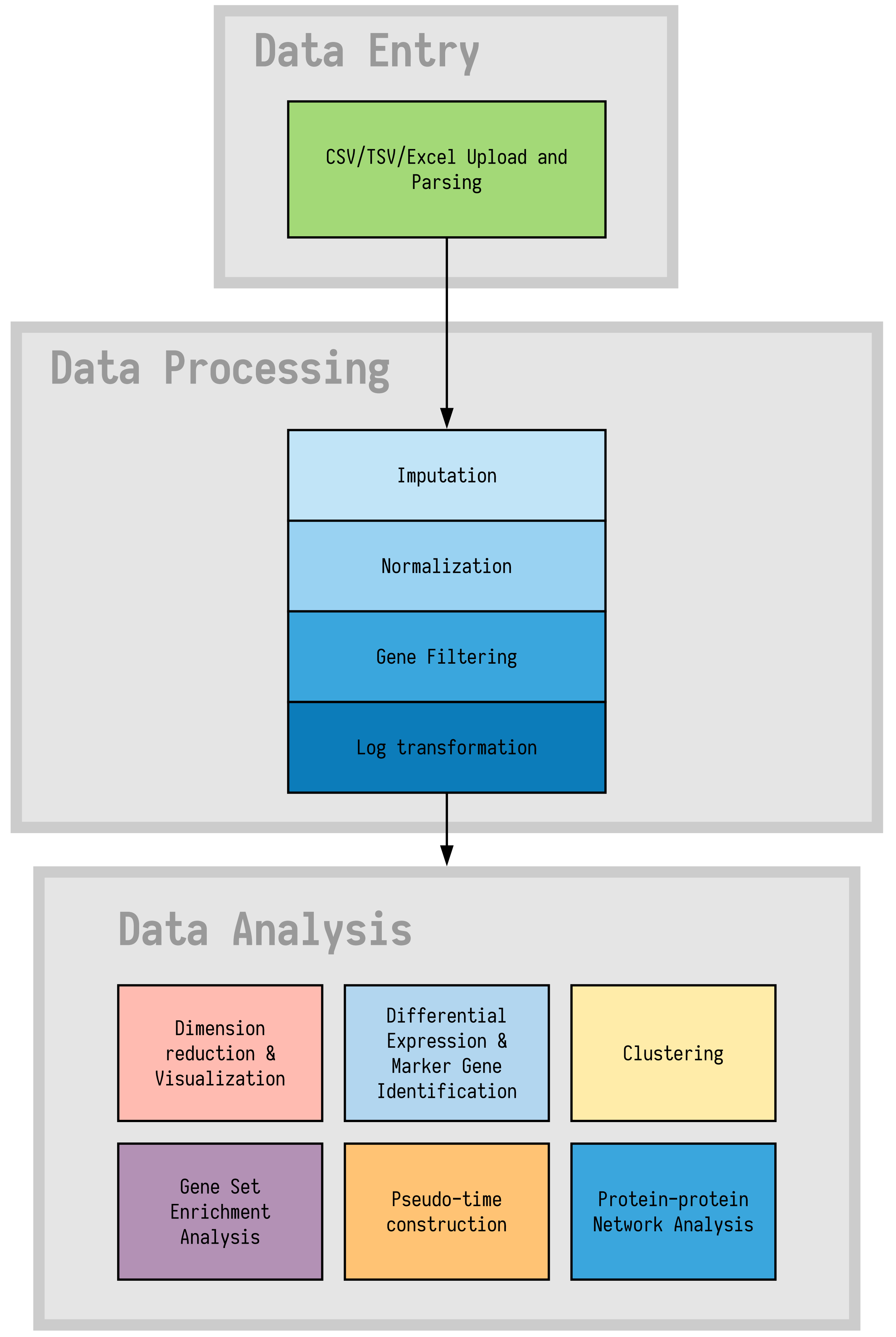
Parameters: ...
Results: ...

...

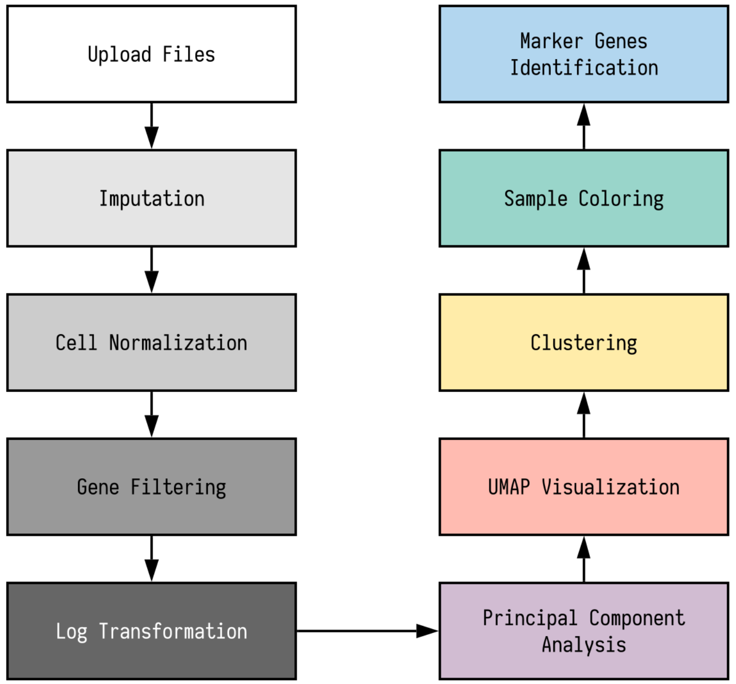
Genes	Cell1	Cell2	Cell3
ABO	12	0	2
ALB	27	13	31
BCL2	25	51	0

A

Computers
Smartphones
Tablets
Accessible on various devices
GranatumX
Multiple deployment options
Personal Computer
Server or HPC
Cloud Services

B

Project Data
Gboxes
Results
and visualization
Uploaded data
Parameter
Parameter
Dependency between Red and Blue Gboxes
Data processed by the Gbox
Parameter
Parameter
Gbox-specific result summary and visualization
One step can generate multple data
Parameter
Parameter
Multiple steps can depend on the same data
Parameter

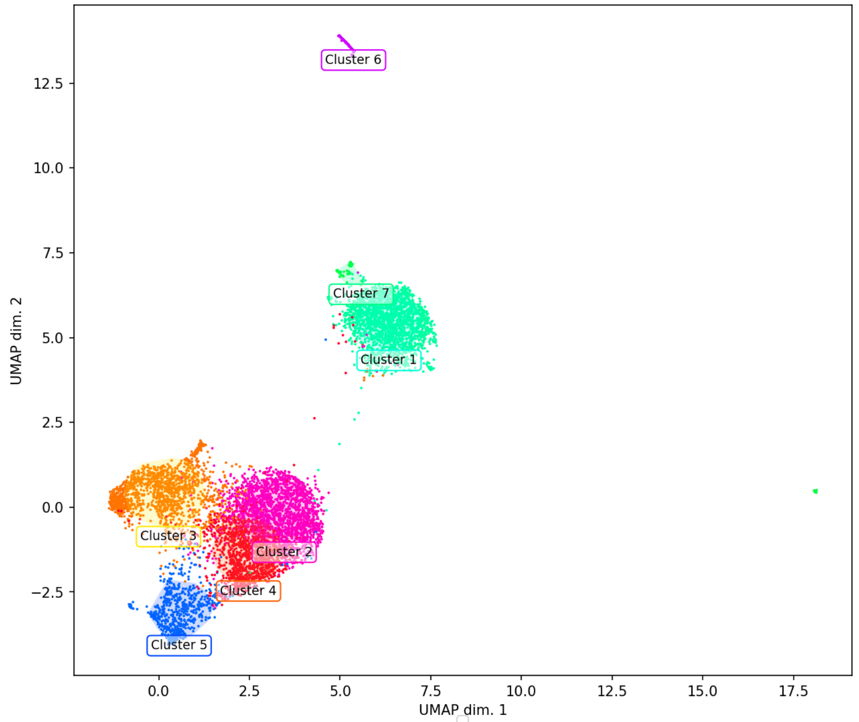
C

Data Entry
CSV/TSV/Excel Upload and Parsing
Data Processing
Imputation
Normalization
Gene Filtering
Log transformation
Data Analysis
Dimension reduction & Visualization
Differential Expression & Marker Gene Identification
Clustering
Gene Set Enrichment Analysis
Pseudo-time construction
Protein-protein Network Analysis

A



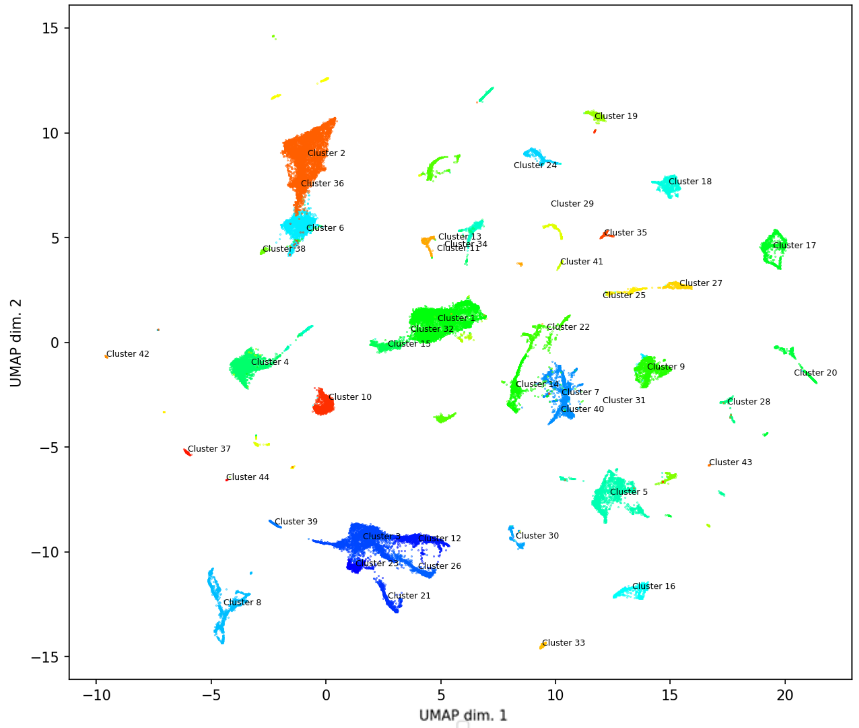
Upload Files
Imputation
Cell Normalization
Gene Filtering
Log Transformation
Principal Component Analysis
UMAP Visualization
Clustering
Sample Coloring
Marker Genes Identification

B



Cluster 6
Cluster 7
Cluster 1
Cluster 3
Cluster 2
Cluster 4
Cluster 5
UMAP dim. 2
UMAP dim. 1
12.5
10.0
7.5
5.0
2.5
0.0
−2.5
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5

C



UMAP dim. 2
UMAP dim. 1
15
10
5
0
−5
−10
−15
−10
−5
0
5
10
15
20
Cluster 19
Cluster 2
Cluster 36
Cluster 24
Cluster 18
Cluster 29
Cluster 6
Cluster 38
Cluster 13
Cluster 34
Cluster 11
Cluster 35
Cluster 17
Cluster 41
Cluster 27
Cluster 25
Cluster 1
Cluster 32
Cluster 22
Cluster 15
Cluster 42
Cluster 4
Cluster 9
Cluster 20
Cluster 14
Cluster 7
Cluster 10
Cluster 31
Cluster 28
Cluster 40
Cluster 37
Cluster 43
Cluster 44
Cluster 5
Cluster 39
Cluster 12
Cluster 30
Cluster 26
Cluster 8
Cluster 21
Cluster 16
Cluster 33

Steps to make a Gbox

1. **Download and run** GranatumX locally:
   a. Install docker and pull the images that you want to add to your pipeline. For example **docker pull granatumx/gbox-scanpyclustering:1.0.0**
   b. Run **source <(docker run --rm -it granatumx/scripts:1.0.0 gx.sh)** to pull scripts and aliases from the docker container
   c. If you are running for the first time, run **gx init.sh** and **gx initGXdatabase.sh,** otherwise run **gx run.sh** to run the webapp. It will tail the webapp, wait until it kicks off the server which takes 2 minutes to complete. You can "ctrl c" out of it without stopping the service, and you can run **gxtail** to monitor the progress at any time.
   d. **Go to** localhost:34567 to see the pipeline up and running
   e. If you would like to run as a webservice for others, you can install and run apache and use a Proxy to the port, but be sure to increase ProxyTimeout for large files and install to root "/" for the web location.

2. **Clone** the gbox-template repo from here, or pull it from docker using docker pull granatumx/gbox-template:1.0.0

3. **Extend the existing sdk with the Dockerfile**; either gbox-py-sdk:1.0.0 or gbox-r-sdk:1.0.0 depending on the language you write the Gbox in
   a. Have **FROM granatumx/gbox-x-sdk:1.0.0** as the first line of the Dockerfile
   b. Copy files from the container : **COPY . .**
   c. Edit the Dockerfile by adding any package installation scripts your Gbox requires
   d. At the end of the Dockerfile, you need to create the /gbox.tgz file so that it can be added to the gx database, and run the translation script.

      RUN ./GBOXtranslateVERinYAMLS.sh

      RUN tar zcvf /gbox.tgz package.yaml yamls/*.yaml # <- replace with your definition files defined in the next step

4. **Create the package.yaml file:** A key requirement of a GranatumX Gbox is the package.yaml file. In this file you'll add descriptive information about your Gbox, information about the backend scripts your Gbox uses to perform its operations, and definitions for the frontend user inferface of your Gbox. The file will look something like the below. Note that {VER} and {GBOX} are replaced

with the version and name of the gbox. You can specify those by supplying the files VERSION.txt and GBOX_BASE_NAME.txt, containing the version and name of the Gbox in their first lines.

```
id: TransposerPackage-{VER}
meta:
  maintainer:
    name: Your Name
    email: your.name@gmail.com
gboxes:
  - id: TransposerPackage-{VER}
    meta:
      title: TransposerPackage {VER}
      subtitle: Example RNA seq package that transposes a matrix
      description: |
        This is a template package. You can use this package as a starting point for your own package
    endpoints:
      backend:
        type: docker
        image: {GBOX}
        cmd: python ./main.py
    frontend:
      args:
        - type: seed
          injectInto: seed
          default: 12345
          label: Random seed
          description: >
            A random seed
        - type: number
          injectInto: someInput
          default: 2000
          label: Some number
          description: Example of a number input
      imports:
        - kind: assay
          label: Assay
          injectInto: assay
      exports:
        - kind: assay
          extractFrom: Transposed assay
```

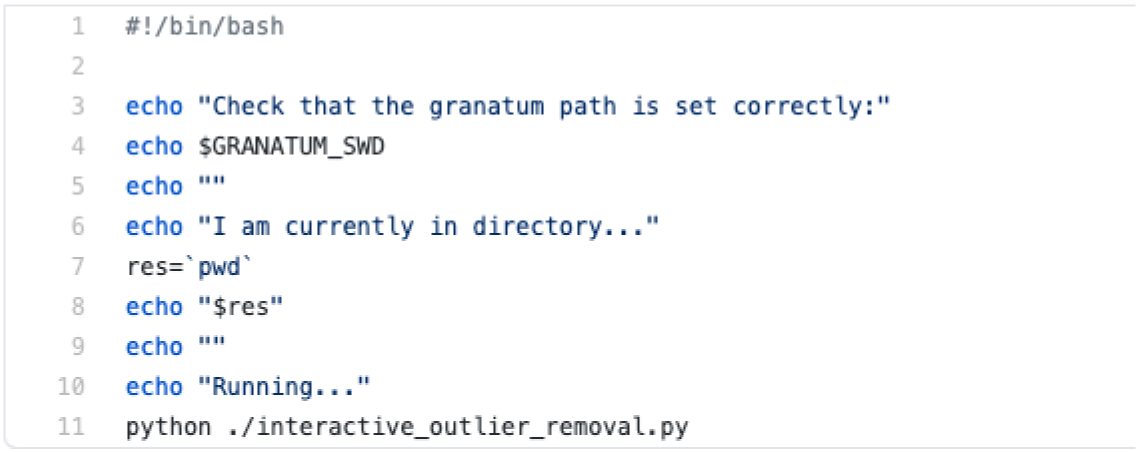
For a full description of all available fields, check out the index.d.ts file. Two key fields are briefly described below.

The "endpoints" field contains the logic which will connect a user's pipeline to your Gbox backend. In this example, the Gbox backend is a docker image, and the command to be executed when the user runs your Gbox is "python ./main.py". You can look at the g_packages folder for examples of other backend configuration options.

The "frontend" field contains the "args" section defining what HTML inputs to show the user (these serve as the arguments which will be passed to your Gbox), the "imports" section defining what types of inputs from other Gboxes your Gbox accepts, and the "exports" section defining the types your Gbox exports to other Gboxes.

5. Edit your main.py file with your application code (or whatever file you specified in package.yaml > gboxes > endpoints > backend > cmd). The Python package granatum_sdk contains helper methods to handle data transfer between the frontend and the backend.

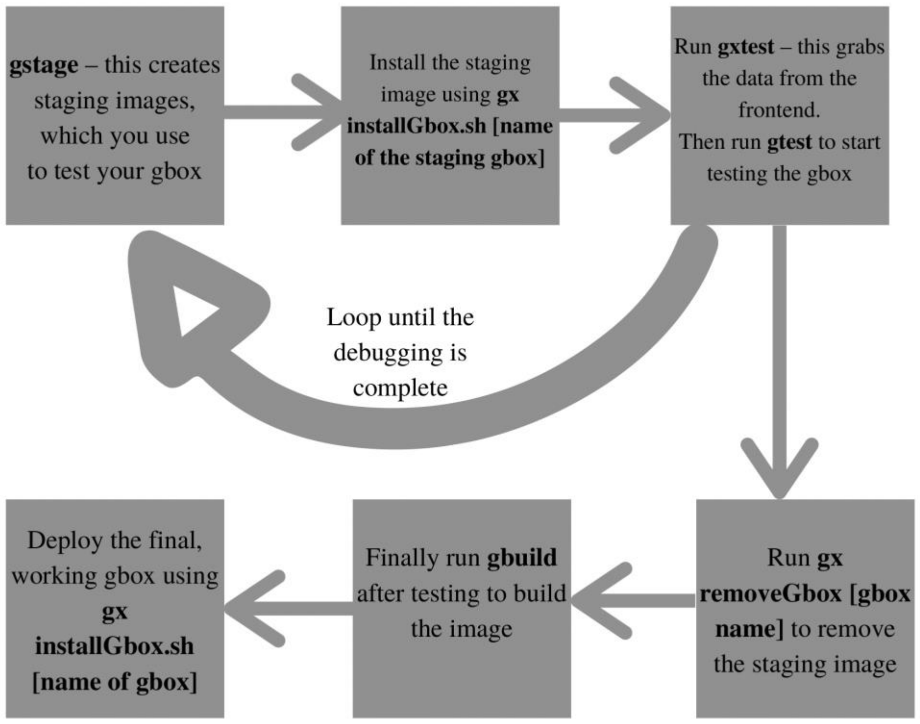
6. **Create the test/test.sh file:** This is what runs testing on the Gbox. An example of a test.sh file would be the one below, replacing the last line with the name of the file your Gbox is run by.



```
#!/bin/bash

echo "Check that the granatum path is set correctly:"
echo $GRANATUM_SWD
echo ""
echo "I am currently in directory..."
res=`pwd`
echo "$res"
echo ""
echo "Running..."
python ./interactive_outlier_removal.py
```

7. Run gstage to create staging images -- this creates images, grab the data on the frontend with gtest, then run gbuild to update the Gbox on the frontend.


gstage – this creates staging images, which you use to test your gbox
Install the staging image using **gx installGbox.sh [name of the staging gbox]**
Run **gxtest** – this grabs the data from the frontend. Then run **gtest** to start testing the gbox
Loop until the debugging is complete
Run **gx removeGbox [gbox name]** to remove the staging image
Finally run **gbuild** after testing to build the image
Deploy the final, working gbox using **gx installGbox.sh [name of gbox]**