Integrating R in a distributed scientific workflow via a Jupyter-based Environment

Mariantonietta La Marra[1], Diederik Blanson Henkemans[2], Jessica Titocci[3], Spiros Koulouzis[2,4]
Ilaria Rosati[3], Zhiming Zhao[2,4]
[1]Department of Biological and Environmental Sciences and Technologies University of Salento, Lecce, Italy
mariantonietta.lamarra@studenti.unisalento.it
[2]Multiscale Networked System, University of Amsterdam, Amsterdam, the Netherlands
diederik.blanson.henkemans@student.uva.nl, {s.koulouzis | z.zhao}@uva.nl
[3]Institute of Research on Terrestrial Ecosystems, National Research Council, Lecce, Italy
{Jessica.titocci | ilaria.rosati}@cnr.it
[4]LifeWatch ERIC, Virtual Lab and Innovation Center (VLIC), Amsterdam, the Netherlands

***Abstract*—The Research Infrastructure Lifewatch Italy has developed a Virtual Research Environment for studies on phytoplankton ecology that includes computational services based on R, a programming language widely used for data science and ecology. Here we have verified the feasibility of a Jupyter-based research environment, the NaaVRE, which has so far been tested only with Python, for running R code in a workflow on the Cloud. The successful execution demonstrated the potentialities of R in Cloud-based research environments. However, further investigation is needed, in particular, to overcome the issue of the lack of dependencies declaration in R. The possibility of performing analyses in a workflow, combined with the computational resources of remote infrastructures, will support scientists in carrying out FAIR and innovative research in a more efficient, integrated and collaborative way.**

***Index Terms*—Virtual Research Environments, R language, phytoplankton, Jupyter Notebook, workflow**

I. Introduction

High-level programming languages like R, Python, and Julia have emerged as powerful tools for data science and are widely used in ecological studies [1]. However, executing large-scale data problems on local machines present additional challenges, particularly as many specialists program on personal computers or laptops that have limited CPU specifications and memory capacity. To enable open sciences, AI and data specialists often desire to share their results and make their components available to others, allowing integration into their workflows with different data sets. Unfortunately, web-based notebook environments typically lack out-of-the-box solutions for this purpose. Addressing this challenge requires an efficient method to package and distribute coding components, ensuring seamless integration into various workflows. However, packaging reusable and efficient components from source code remains a complex task requiring individual attention, as each high-level programming language presents unique challenges. While Python has been researched and implemented in previous work [2], effectively containerizing R code remains an open question that requires further investigation.

LifeWatch Italy uses R to design the computational web services of the Phytoplankton Virtual Research Environment (Phyto VRE), which facilitate phytoplankton data analysis using automated procedures and algorithms and help researchers address basic and applied studies on phytoplankton ecology [3].

The aim of this research was to test the feasibility of executing the R-based computational web services of the Phyto VRE in a workflow on the Cloud via a Jupyter-based environment.

II. Background, Related Work and Methodology

Notebook-as-a-VRE (NaaVRE), a Cloud research environment, expands Jupyter Notebook [2] [4] with extra features for searching relevant notebooks, containerizing and composing notebook fragments (called cells) as workflows, and automating the execution on cloud environments. While the NaaVRE has been mainly tested on Python-based notebooks, it has not been tested for the R language. There are several existing tools for containerizing R code, as shown in Figure 1.

Tool	Binder	holepunch	WholeTale	containerit	liftr
Container Technology	docker	docker	docker	docker	docker
Base Image	buildpack-deps	rocker/binder	rocker/geospatial	rocker/verse	rocker/r-base
Tag	bionic	4.3.0	latest	3.5.2	latest
Base Size (comp.)	211MB	1.67GB	1.56GB	1.06GB	336MB
Base Size (uncomp.)	621MB	4.82GB	4.44GB	3.05GB	854MB
us-500 size	N.A.	N.A.	5.25GB	3.05GB	N.A.
us-500 contain. time	4m05s	5m12s	N.A.	0m08s	1h30m
Functional	Only Old Repos	No	Yes	Yes	No
In./Out. Detection	No	No	No	No	No
Dependency Detection	No	Yes	No	No	No
Repo CRAN	N.A.	Yes	N.A.	Yes	N.A.
Repo BioC	N.A.	No	N.A.	Yes	N.A.
Repo GitHub	N.A.	Some	N.A.	Yes	N.A.
Minimal Dependencies	No	No	No	Some	No

Fig. 1. Comparison of existing containerization tools for R.

Some of these tools have demonstrated the ability to create functional containers for R scripts; however, no solution meets all of our requirements for integrating into the NaaVRE environment, e.g., effectively detecting input/output of the program, minimising the size of containers, and optimizing the packages to be included in the containers.

We will develop an effective R containerization extension for Jupyter to tackle those issues. Using the extension, we can load the Phytoplankton R scripts in the NaaVRE as a notebook to interactively containerize it using the R-Containerizer, and compose and execute the workflow of containers via the NaaVRE engine.

III. R-Containerizer

Fig. 2 shows a screen snapshot of the R-Containerizer. It is integrated as a Jupyter extension as part of the NaaVRE environment. In the left panel of the Jupyter environment, the R-Containerizer will automatically parse the R code selected by the user from the notebook on the right side. To determine the input/output interface of the selected R code, R-Containerizer needs to analyze embedded function parameters, building identifiers and iterator variables. While packages in the R cell can be detected through static analysis, shown in the figure under "Dependencies", determining the package repository is a challenge. For hidden dependencies, a Generative AI approach is used to analyze error messages where a set of dependencies might be suggested to be included. The system maintains an internal knowledge base of all repositories and their packages.

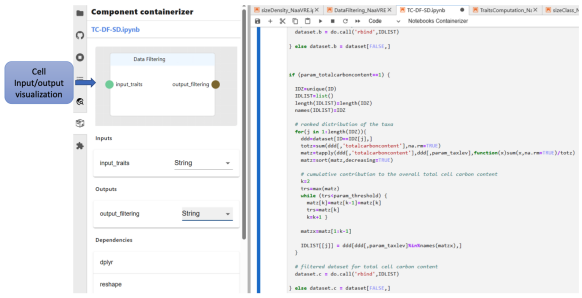

Fig. 2. Component containerizer for one of the ecological scripts.

A user can select a suitable base image from a number of options based on the packages used in the code cell to generate a container (currently Docker) image.

IV. Cloud workflow using containerized R code

We first loaded all the R scripts of the Phyto VRE into NaaVRE environment and verified if they can all be executed in Jupyter using the R kernel. Using the R-Containerizer, we containerize them as Docker images on different granularity. The container size and containerization time predominantly depend on the number of libraries a script requires. Using NaaVRE, we can compose the workflow among those Dockers, as shown in Fig. 3.

The workflow among containerized R scripts can be executed on cloud infrastructure via the NaaVRE engine. The workflow creates correct results as the legacy R code but with better automation among different scripts.

V. Summary

The integration of the Phyto VRE demonstrated that the R scripts can be containerized and composed as workflow in a Jupyter-based environment like NaaVRE. The metrics

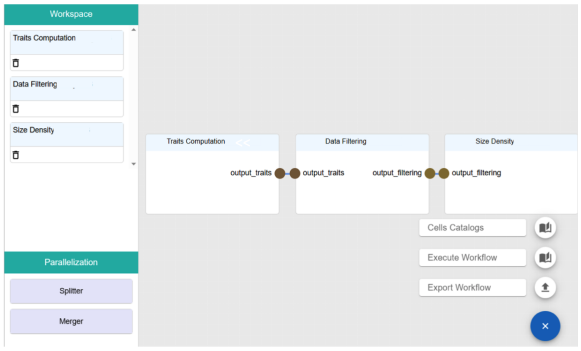

Fig. 3. Workflow for the ecological scripts.

used to evaluate the process showed that the R-Containerizer developed for the NaaVRE, compared with already existing tools, shows advantages in terms of lightweight, efficacy and efficiency, considering in particular the rate of success of containerization and the container size. The analysis of the container size also highlighted an important difference between Python and R. Python has a more formal way of declaring dependencies that makes it easier to automatically determine which library a particular function depends on. R lacks this form of dependency declaration. For this reason, when containerizing individual cells, often more libraries than actually needed are included in the container, to assure that all the functions in the cell can be executed. In fact, missing dependencies are the main reason why notebooks do not work properly [5].

From the study, we demonstrated the feasibility of integrating the R-based Phytoplankton VRE in the NaaVRE, which allowed us to identify areas for near-future developments. It will enable R-Studio users to share their code across R-Studio and Jupyter and scale out the workflow on the dynamic infrastructure services of the cloud via containerized code cells.

Acknowledgment

This work has been partially funded by the European Union project CLARIFY (860627), ENVRI-FAIR (824068), BlueCloud-2026 (101094227) and LifeWatch ERIC.

References

[1] F.M. Giorgi, C. Ceraolo and D. Mercatelli, "The R Language: An Engine for Bioinformatics and Data Science", Life, 12, pp. 1-25, 2022.

[2] Z. Zhao, S. Koulouzis, R. Bianchi, S. Farshidi, Z. Shi, R. Xin, Y. Wang, N. Li, Y. Shi, J. Timmermans et al., "Notebook-as-a-vre (naavre): from private notebooks to a collaborative cloud virtual research environment", arXiv preprint arXiv:2111.12785, 2021.

[3] E. Stanca, N. Fiore, I. Rosati, L. Vaira, F. Cozzoli and A. Basset, "Case Study: LifeWatch Italy Phytoplankton VRE", In Zhao, Z. and Hellström, M. (eds). Springer International Publishing, Cham, pp. 324– 341, 202.

[4] J. M. Perkel, "Why jupyter is data scientists' computational notebook of choice," Nature, vol. 563, no. 7732, pp. 145–147, 2018.

[5] Y. Wang, S. Koulouzis, R. Bianchi, N. Li, Y. Shi, J. Timmermans et al., "Scaling Notebooks as Re-configurable Cloud Workflows", Data Intell., 4, pp. 409-4023, 2022.