International Journal of Geo-Information

MDPI

Article

Leveraging Container Technologies in a GIScience Project: A Perspective from Open Reproducible Research

Benito M. Zaragozí[1,†], Sergio Trilles[2,†*], José T. Navarro-Carrión[3]

[1] Departament de Geografia, Universitat Rovira i Virgili, C/ Joanot Martorell, 43480, Vilaseca, Spain
[2] Institute of New Imaging Technologies (INIT), Universitat Jaume I, Av. Vicente Sos Baynat s/n, 12071, Castelló de la Plana, Spain
[3] Instituto Interuniversitario de Geografía (IIG), Universidad de Alicante, C/ San Vicente s/n, 03690, Alicante, Spain
* strilles@uji.es, Tel. (+34) 964 38 76 86.
† The authors contributed equally to this work.

Received:17 November 2019; Accepted: 20 February 2020; Published: 25 February 2020

Abstract: Scientific reproducibility is essential for the advancement of science. It allows the results of previous studies to be reproduced, validates their conclusions and develops new contributions based on previous research. Nowadays, more and more authors consider that the ultimate product of academic research is the scientific manuscript, together with all the necessary elements (i.e., code and data) so that others can reproduce the results. However, there are numerous difficulties for some studies to be reproduced easily (i.e., biased results, the pressure to publish, and proprietary data). In this context, we explain our experience in an attempt to improve the reproducibility of a GIScience project. According to our project needs, we evaluated a list of practices, standards and tools that may facilitate open and reproducible research in the geospatial domain, contextualising them on Peng's reproducibility spectrum. Among these resources, we focused on containerisation technologies and performed a shallow review to reflect on the level of adoption of these technologies in combination with OSGeo software. Finally, containerisation technologies proved to enhance the reproducibility and we used UML diagrams to describe representative work-flows deployed in our GIScience project.

Keywords: reproducibility; open science; open reproducible research; GIScience; containerisation; dockers; UML

1. Introduction

Replicability and reproducibility are essential aspects of the scientific method that add consistency to scientific contributions in any discipline. The reproduction of any scientific work seeks to verify and validate the results obtained, which will lead to a better understanding of the realm that has been studied. This approach increases the chances of reusing or extending previous research works [1].

The statement "Replicability is not reproducibility" has already been discussed in various works. In [2], the author distinguishes between reproducibility and replicability, arguing that a replication is a study that arrives at the same scientific findings as another study, collecting new data and completing new analyses. As long as research can provide all the necessary data and the computer codes to rerun the analysis to re-create the results, it can be considered reproducible research. In other words, reproducibility refers to a phenomenon that can be predicted to be repeated under the same experimental conditions. At the same time, replicability describes the ability to obtain an identical result when an experiment is performed under new conditions (data and methods). This distinction makes more sense in the computer sciences field, more specifically in computational research, because

if data and code are available, the results should be identical. It must be taken into account that replicability does not guarantee the validity of the scientific contribution [3].

A full degree of reproducibility or replicability will not always be possible, as it dramatically increases the cost of studies and the researchers' workload [4]. This statement is also becoming increasingly evident in studies related to computer science. In the last two decades, a vast amount of data has been collected and will continue to grow exponentially. The term "Big Data" refers to an enormous volume of data that exceeds the capabilities of conventional database systems because the data are too bulky, recorded too fast or is not structured according to the existing database architectures [5]. The phenomenon of Big Data makes it challenging to reproduce certain studies—usually for reasons related to time, resources and labour—and so only a well founded suspicion about the results would make it necessary to repeat the computational experiences [6].

Over the last decade, the concern to determine to what extent studies published in scientific journals are reproducible has been widely discussed. Some data can be alarming and in other studies, authors can talk of a "reproducibility crisis" [7,8] to the point that 90% of participants in a recent survey of the journal Nature agreed that the aforementioned crisis exists [8]. Some more striking figures estimate that 85% biomedical research efforts are wasted [7], while [9] have calculated that—only in the USA—28,000 million dollars are spent every year on research that is not reproducible.

The risks of the lack of reproducibility are also manifested in the social sciences. An example of this is the article by Reinhart and Rogoff in 2010 [10], which was published in a special issue (without peer review) of the prestigious journal *The American Economic Review*. Its results were taken as essential arguments to support pro-austerity policies applied after the economic crisis that began in 2007–2008. Later, it was shown that the results could not be reproduced due to errors in the coding of the data of a spreadsheet, selective exclusion of available data and a low weighting of variables. As a result, the phenomenon studied would be much less significant than the authors of the first article indicated [10,11]. If political decisions are to be based on scientific studies from the social sciences, it is logical to expect such studies to be fully reproducible [5].

Although some authors use the term "crisis" to describe the widespread lack of reproducibility, it must be remembered that science is a cumulative process and that many studies are needed to generate new knowledge. To a greater or lesser extent, the lack of reproducibility is part of the essential functioning of science, which continuously corrects itself [12]. In any case, there is much room for the improvement of research practices and thus, being able to make better use of the resources allocated for science.

Researchers from different disciplines have begun to promote reproducible research as a minimum standard to defend the results of research, especially in studies in which obtaining complete replicability is not viable [13]. In this way, reproducibility is defined as a spectrum of possibilities between an investigation that cannot be validated and another that is fully replicable (see Figure 1).

Figure 1. The spectrum of scientific reproducibility, between a scientific publication and full replication (based on [13]).

Obviously, to achieve a high degree of reproducibility, it is necessary to maintain detailed documentation and to describe the methods applied meticulously. Traditionally, scientific publications have been used for this purpose. However, as certain studies increase in complexity, it becomes clear that just papers—text, tables and static figures —are no longer enough to reproduce a scientific study [14]. Researchers should include textual and numerical descriptions of the research, as well as other

essential elements, such as software, data, environment variables, platform dependencies and other resources involved in the methodology used [15].

This research work focuses on reproducibility in the field of Geographic Information Science [16] (also GIScience or GISc), which comprises a wide range of research areas, such as climate modelling [13] and studies on the applicability of Voluntary Geographic Information (VGI) [17]. In addition, special attention is also paid to the Open Reproducible Research (ORR) perspective, which is exemplified as a research work in which all the elements (i.e., data, code, etc.) are based on open and publicly accessible resources that can produce the same results as those that have been published [18]. The concept of reproducibility is addressed here in a broader sense, including works with different levels of configuration and development needs (e.g., from geostatistical experiments to new software development). It is considered that something cannot be understood as being reproducible without it being open [19]. In this sense, ORR must be Open Science (OS) and should meet the following requirements: 1) open access, which refers to the possibility of accessing research results (publications) for free without restrictions [20]; 2) open data, which is based on the publication of research data collected during the research process and made available to the community for study and reuse [21] using a license; and 3) open-source, which refers to software whose source code is publicly available along with a licence that allows its use, modification and distribution [22].

1.1. The SIOSE-INNOVA project

As explained above, the concern for scientific reproducibility is shared by many researchers and is taken into account in more and more projects. Accordingly, in 2017, when we started the SIOSE-INNOVA project, we evaluated how we would ensure that our project did not suffer from the problems mentioned in the previous section. To understand the rest of this work and our decisions, in this subsection, we introduce the main details and requirements of the project.

The Information System on Land Occupation of Spain (SIOSE; https://www.siose.es/) mainly involves managing the creation and maintenance of a geodatabase that contains land occupation data (land use/land cover; LU/LC) from Spain. These data are necessary for researchers studying different environmental and socioeconomic problems. The SIOSE geodatabase contains a big volume of vector LU/LC data at a 1:25,000 scale, comprising about 10 GB in each of the published database versions (2005, 2009, 2014 and 2019). These volumes are going to be much more important in the future high-resolution versions that will reach the cadastral parcel scale. This database follows a complex (object-oriented) database model that is cumbersome for typical GIScience projects. These limitations are closely related to the previously commented reproducibility problems. As explained previously, working with such big databases and complex data models makes reproducing any workflow much more difficult unless proper practices are applied.

The SIOSE-INNOVA project (http://siose-innova.es/) proposes technical and methodological innovations for the SIOSE database. These innovations are considered necessary to make these data more accessible for its application in geographical studies. The SIOSE-INNOVA project is organised into two complementary parts: 1) a technical innovation part that consists in verifying which open source technologies provide the best solutions to exploit certain facets of the SIOSE database, and 2) an applied part that involves the implementation of these new technologies in real case studies. In this project, experts with very different technological profiles—GIS, geodatabases, programming, and data mining, among other tools—need to collaborate and overcome the difficulties in using the SIOSE database. All these specialists must be able to work with the same tools, and the only possibility is that some intermediary provides a stable and transparent working environment.

SIOSE data are public and distributed under a CC-BY 4.0 licence. This licence allows free access and free use for any legitimate purpose, the only strict obligation being to acknowledge and mention the origin and ownership of licensed geographic information products and services, such as the Instituto Geográfico Nacional (IGN). SIOSE data can be retrieved from the downloads page of the

Centro Nacional de Información Geográfica (CNIG) [23] or the websites of Spanish regional mapping agencies.

As mentioned above, the SIOSE database is designed according to an object-oriented model, but due to practical reasons, it is adopted and implemented within an object-relational database, which is a clear example of the *object-relational impedance mismatch* [24]. The usability problem investigated by the SIOSE-INNOVA project has to deal with the fact that regular users of this type of information layers usually work with *ESRI Shapefiles* and these are generally not used to work with such information in any of the original distributions of SIOSE (corporate databases). These problems could be solved with the development of new tools for querying the database, which implies the need to define a working environment in which the development becomes manageable and everyone involved can collaborate in the realisation of computational experiments. This could be understood as "intra-project reproducibility", but that would have positive effects on the higher reproducibility and sustainability also by researchers not belonging to this project.

1.2. Objectives

The requirements of the SIOSE-INNOVA project led us to investigate how GIScientists use open source tools and techniques to produce ORR outcomes. Only then would it be possible to make good use of those resources in specific GISc projects. In this manuscript, we explain this process of discovery by providing some contextual information and describing how we finally addressed reproducibility in our project.

The target audience of this work is researchers and practitioners—with or without much experience in ORR—who wish to explore how to produce reproducible research in the GIScience domain. Considering this audience and the SIOSE-INNOVA requirements, the specific objectives of this work are: 1) to define the most essential facilitating technologies, platforms and good practices that enhance ORR; 2) to provide an overview on how containerisation technologies can boost reproducibility in GIScience projects; 3) to analyse the level of adoption of containerisation technologies in the geospatial domain; and 4) to describe the challenges that must be overcome to integrate reproducible workflows into GISc projects with requirements similar to those of the SIOSE-INNOVA project.

The rest of the paper is organised as follows. Section 2 presents how ORR can be achieved within the scope of GIScience. Section 3 shows the impact of reproducibility on GIScience using the *Docker Hub* repository as a proxy. Section 4 describes how ORR was achieved in different computational experiences involving a SIOSE database. The paper finishes with Section 5, which presents the conclusions and proposes lines for future work.

2. Open Reproducible Research in the geospatial domain

GIScience shares the concerns mentioned above about reproducibility. However, upon analysing the literature published in the field of geospatial sciences, it can be observed that their presence is much lower than in other disciplines [25].

Due to the importance of this topic, several studies assess the application of ORR in GISc. For example, [26] aims to analyse the reproducibility of the work of geoscientists who carry out computational research and to identify the obstacles hindering the achievement of reproducible works. In Ostermann and Granell [27], the authors researched reproducibility in VGI, where they obtained non-promising results as regards the degree of reproducibility. Reference [28] proposed a short-term strategy to achieve reproducible research within the Association of Geographic Information Laboratories in Europe (AGILE) community [29]. In [30], the same authors performed an analysis of 32 conference papers presented in the AGILE conferences, between 2010 and 2017, concluding that there is a low average level of reproducibility in GIScience and no positive trend is perceived. In [31], the authors proposed a programme for submitting reproducible publications—including data, source code and metadata—in an open geosciences journal. Reference [32] put forward the Executable

Research Compendium (ERC), which uses *Docker* containers to encapsulate in an environment with all components.

Many works in GISc build statistical models (computational research) from geospatial data. Of course, when they share their algorithms and data, they can be considered works that are reproducible without much difficulty [26]. However, this type of work is not the most predominant in computationally intensive sciences, and on many occasions, achieving reproducibility is not trivial. Most of the scientific work carried out in the computer science field requires complex environments—with specific workflows—for each of the problems to be investigated [33]. Moreover, geospatial datasets are not always easy to share, as they tend to be huge in terms of size or volume. As explained in Subsection 1.1, this would be the case of the SIOSE-INNOVA project.

Nowadays, there are many available resources for addressing ORR, and we even knew many of these resources before starting the SIOSE-INNOVA project. In order to contextualise them, we propose a diagram based on Peng's [3] reproducibility spectrum (see Figure 1). Our proposal is just a juxtaposition of the reproducibility spectrum proposed by Peng with a partially ordered space showing important *practices*, *platforms* and types of *assets* that could be used in a GISc project. Although with certain precautions, we will refer to this diagram as the spectrum of reproducibility in GIScience (see Figure 2). As in the original reproducibility spectrum, our proposal is divided into three main zones: 1) publication only (not reproducible in most cases); 2) publication with some elements that promote reproducibility (i.e., code, data, environments, *assets*, *platforms*, *practices*, etc.); and 3) full replicable research.The elements required to ensure a good level of reproducibility will vary between projects and research objectives, so many combinations are possible. Of course, there is no universal solution that solves all cases. In this ORR spectrum, the key element is the level of commitment to achieve reproducibility. A higher level of commitment means developing all those elements that promote reproducibility in each particular case (i.e., *assets*, *platforms* and *practices*).

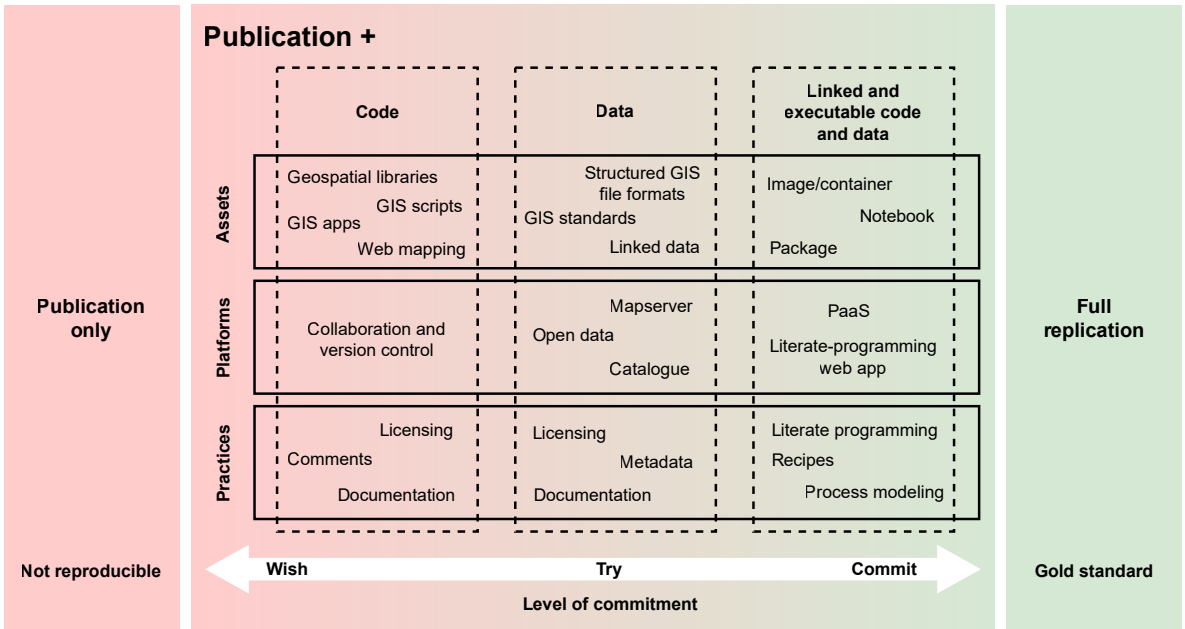

Figure 2. An Open Reproducible Research spectrum in the field of GISciences. Note: the relationships between the elements in the core of the diagram are not defined, so these will vary according to the needs of each project.

Another important particularity of this ORR spectrum is that many of the elements that promote reproducibility (e.g., *assets*, *platforms* or *practices*) can accomplish very different purposes. For example, any of the development (collaboration and version control) *platforms*—*GitHub*, *Bitbucket* [34] and *Gitlab* [35] —can be used for different functions in a reproducible project. They could be used for code sharing

and versioning, project documentation and management, publishing small datasets, informing about the chosen licensing or even linking data and code in some web applications, among other possibilities.

The inherent complexity of this spectrum means that in this work, we can only present a general context on specific *practices* and tools that generally solve the most typical reproducibility problems. The following subsections detail how the reproducibility spectrum can be interpreted within the context of GIScience projects—from the most elementary *assets* (e.g., programming languages, data formats, etc) to their integration into the most common good *practices* (those introduced in Figure 2). In this work, we include *assets* in the list of heterogeneous resources that may be used in the geospatial domain, such as programming libraries, desktop applications or implemented standards.

2.1. Collaborative development and code sharing

Reading the central zone of our spectrum—from left to right and from top to bottom—we first find what we have called code *assets*, including free geospatial projects of different categories, such as programming libraries or desktop GIS applications, and products created for their usage. In the SIOSE-INNOVA project, this would be referred to as spatial Data Base Management Systems (DBMS), database clients and scripts written in a database query language.

The decision of which tools are adequate for a particular GISc project depends on many factors, but it is essential to know the available options and their main characteristics. As a convention, geospatial projects can be grouped by software category or by programming language [36,37]. Grouping projects by a programming language is interesting because it should be easier to use projects and reuse code developed with the same programming language. Depending on the particular objectives of a GISc project, many different programming languages can be used, but here, we only focus on the most relevant ones. In decreasing order of usage [38], the following are some of the most commonly used programming languages and projects under the Open Source Geospatial Foundation (OSGeo) [39] umbrella:

- *Python*. This is the language with the greatest presence in the GIS realm. This is due to its use as a scripting language in the best-known GIS desktop applications, such as *QGIS* [40].
- *Javascript*. Widely used for web programming, it is the language with the most widespread presence in web mapping. Two well-known web mapping APIs are *OpenLayers* [41] or *Leaflet* [42].
- *R*. This language is used for statistical computing, data visualisation, data science and geospatial analysis. It can also be used to access other GIS or inside GIS tools such as *QGIS* via connectors. The most relevant and useful R libraries are collected into the *Spatial Task View* (https://cran.r-project.org/view=Spatial).
- *SQL*. For many years, it has been used to query geospatial databases. *SQL* is at the heart of many GIS operations in spatial queries. It is used in software such as *PostGIS* [43], *Spatialite* and *QGIS*.
- *Java*. This is one of the most widely used languages in open source GIS programs, such as *GeoServer* [44], *GeoNetwork* [45] or *JTS* [46].
- *C/C++*. These languages are two of the most commonly used for creating GIS desktop applications and other platforms. In the GIS field, we can find open source applications or libraries in this language, such as *QGIS*, *GRASS GIS* [47], *MapServer* [48], *GDAL* [49] or *Orfeo Toolbox* [50].

Among this list of programming languages and projects, the objectives of the SIOSE-INNOVA project can be better achieved using *SQL* on a DBMS like *PostgreSQL/PostGIS*. This solution can manage volumes of geospatial data similar to those in the SIOSE database without any problem. *PostgreSQL* allows to store relational data or hierarchically structured data thanks to the *JSONB* support, which is very convenient considering the object-oriented data model used by SIOSE.

Following with our spectrum of reproducibility, we consider the use of collaborative platforms for sharing code and keep it accessible at any version as essential. Code must be linked to research publications using permanent and stable public URIs. There are different collaboration and version

control platforms that help to achieve these requirements (i.e., *GitHub*, *BitBucket* or *GitLab*). These *platforms* enable several people to work collaboratively on a single project and track all the contributions using unique identifiers for each contribution (e.g., git commit hashes or released versions). This last feature allows us to know exactly which version of the code was used to generate certain research results, which thus enables others to test modifications or to suggest improvements without affecting the normal development of the main project. Perhaps the essential feature of these development platforms is to generate a community that can continue development around the research questions [51]. Even if the platforms mentioned above are not enough, there are more indicated solutions such as *Archivematica* [52] and *RODA* [53] to preserve long-term access to trustworthy, authentic and reliable digital contents.

At the end of the first column of our spectrum, we locate the *practices* that can contribute to sharing the code of GISc projects in the most appropriate way. Within the movement towards open science, the concept of open methods is defined [54]. This is based on the idea that researchers should document their methods as completely as possible [30], including the provision of textual instructions or other material (photos, videos, etc.). Documenting the code, both at the level of general documentation and comments within the code itself, is a necessary practice to achieve reproducible research. Apart from the *README* files that can describe each project in a general way, it is necessary to generate documentation that describes each component that has been developed.

Another critical aspect of the code to keep in mind is the type of licence attributed, which is also a notable metadata element. There is a general lack of knowledge about the meaning of each type of licence, which is one of the main barriers to reusing code (and data) [55]. As far as we develop open research, open licences for reuse should be adopted after evaluating the most appropriate choices [56].

2.2. Geospatial data availability

Making a dataset accessible does not guarantee the reproducibility of research work but reduces the costs for reproducibility. For this reason, we are going to review the essential elements for publishing open geospatial data. In this subsection, we focus on the central part of our spectrum of reproducibility (Figure 2).

In any discipline applied through computer science, there is a set of structured data formats used by its community. When these formats are widely known and based on open standards, they become a useful and interoperable means of sharing data. In the case of GIScience, there are mainly two essential data models with many file-types based on them: 1) vector files usually contain geometries defined by coordinate lists, which represent points, lines or polygons, and 2) raster files are bitmap images that can be used to store satellite images, elevation models, heat maps or other topographic representations. The most important file formats for both types are outlined below.

As mentioned in Subsection 1.1, the SIOSE requirements for the data are the following: 1) to be stored in a vector format, 2) to hold a considerable volume of data—that could be difficult to manage within specific data formats, and 3) to manage a hierarchical structure of land uses. The most appropriate data formats for a SIOSE database should fulfil these requirements. In this way, we present a selection of the vector data formats that are commonly used to publish data by the CNIG.

- *ESRI Shapefile* [57]. This is a traditional and widely used vector data format. Normally, a *Shapefile* encloses at least three or four different files (*.shp*, *.dbf*, *.shx* and *.prj*) and defines a single type of geometry, which implies that each *Shapefile* can only store points, lines or polygons. This format, although widely used, was not intended as an open format when it was conceived. However, any GIScientist should know this format given the large amount of legacy data stored in this format and users that still use it.
- *GeoCSV* [58]. Comma-Separated Values (*CSV*) is a vector file format and is extensively used in any discipline. The *CSV* format allows work to be carried out using the most usual programs, such as *Excel* or *Notepad*. *CSV* is a plain text file where columns are separated using commas and

rows by lines. More specifically, an optional geometry extension (*GeoCSV*) has been created that can store points (latitude and longitude) and Well-Known Text (*WKT*) standard geometries.
- *GeoJSON* [59]. This is an extension of JavaScript Object Notation (*JSON*) adding geospatial data. It offers the advantage of being a lightweight, easy-to-read notation for web applications. It is usually used as output for APIs and is a vector format.
- *Spatialite/SQLite* [60]. *SpatiaLite* is an open source library that extends the *SQLite* core with Spatial *SQL* capabilities. This is conceptually similar to the combination of *PostgreSQL/PostGIS*, but in this case, *Spatialite* is meant for embedded and portable geodatabases.
- *Geopackage* [61]. This format is a universal file format for sharing and transferring vector and raster spatial data. It is open and supported by Open Geospatial Consortium (OGC) [62]. *Geopackage* is backed by and extends *SQLite* and is highly influenced by *Spatialite*.
- *NetCDF* [63]. This is a data model for array-oriented scientific data that can also store both, vector and raster data. It is a recommended format to encode and distribute multi-dimensional and gridded geospatial data.

Before 2019, SIOSE databases were distributed in ESRI Shapefiles and other proprietary formats, but now data is internally managed using *PostGIS* and can be downloaded as *Geopackages*—one for each of the 17 Spanish regions. As mentioned above, these solutions match the requirements and facilitate the introduction of new improvements, such as the storage of hierarchical data structures. Thus, these formats would also be the most adequate for the SIOSE-INNOVA project.

The choice of an appropriate data format is important for reproducibility to some degree but, any of the mentioned GIS formats come along with a variety of software libraries and tools that add more options to work with these data. Moreover, sharing GIS data files is not the only way to make data accessible. According to the *5-star* open data classification (http://5stardata.info/en/) proposed by [64], there are different levels of openness for sharing (GIS) data on the web. This scheme defines protocols and guidelines that endorse open data format and linkages among datasets, facilitating data interoperability within the context of a Web of Data [65]. In the GIScience community, the OGC standards have a goal to define an interoperable way to build data services on the web. These standards imply a considerable number of conventions (interfaces and encodings), but adhering to these standards guarantees interoperability [66] and reuse, increasing the reproducibility of a work [67].

In the recent years, the OGC has performed an important task in order to publish different standards, proposing technical documents for each of them, such as *Web Map Service* (WMS) [68], *Web Feature Service* (WFS) [69], *Sensor Observation Service* (SOS) [70], *Catalogue Services for the Web* (CSW) [71] or *Web Processing Service* (WPS) [72]. In the proposed spectrum, we have included GIS standards as *assets* that should be taken into account when possible and, also based on several standards, "linked data" is used for publishing structured data so that they can be interlinked with other data, becoming more useful through semantic queries. This way, others can reuse the data to be applied in other scenarios or reproduce a previous experiment [55].

Most of the standards described above are defined and developed as web services. The use of these pieces of software, in general, has strengths and weaknesses for reproducibility. As favourable points, web standards facilitate the execution of experiments since they do not require additional software installation or data management and supersedes the need for local file copies [73]. On the contrary, a web service can be considered a "black box" from which to obtain a result—but not a version number or a copy of the current source code. The results obtained should be identical if there was not a service update or data had been modified; in that case, the outputs could be different. Another problem presented by web services is the possible lack of availability. For example, it may happen due to server maintenance or abandonment due to the end of research funding.

Publishing geospatial datasets in well known open formats is important, but data must be shared permanently to ensure accessibility and guarantee the reproducibility of research. The data used during research must be available for all implications (use and publication) without any copyright

restrictions, patents or other control mechanisms. Currently, there are several *platforms* (open data) for storing data permanently and open, including *Zenodo* [74], *Figshare* [75], *DSpace* [76] or *CKAN* [77]. In addition to storage management, these platforms facilitate the acquisition of a Digital Object Identifier (DOI), which is a persistent interoperable identifier that can be used for unambiguously to identify the outputs of research. The authors of [78] review the current storage platforms to determine whether these platforms are robust and useful for data management. The authors concluded that after analysing 32 similar platforms, extensive work still needs to be done in order to ensure that the data is online and meets the minimum standards.

The above-mentioned possibilities are generic and not designed specifically for the geospatial domain. *GeoNetwork* [45]—which corresponds to a catalogue in the proposed GISc spectrum—or *GeoServer* [44]—a well known map server—are explicitly designed for sharing geospatial data. However, these specific alternatives do not offer the possibility of obtaining a permanent DOI, only different ways of accessing the data.

As can be seen in the reproducibility spectrum, it is also important to apply some type of license to the data, thus facilitating its accessibility and use. The licences that fulfil the Open data concept are: *Creative Commons Attribution*, *Creative Commons Attribution (Share-alike and Zero)* [79], *Open Data Commons (Public Domain Dedication and License, Attribution License and Open Database License)* [80].

Finally, the shared data, regardless of the formats chosen or the publishing *platforms*, must be described so that they can be adequately understood and used. In this sense, another best practice is the promotion of metadata to improve the discovery and cataloguing phases, as metadata, in particular, can provide a context to guide users on how to use what is published [81]. One solution to improve the generation of metadata is to automate the process of creating them after the publication of the data (indexing) in a way that is unattended by the user [82]. In any case, these documentation practices are similar to those applied in the case of software.

2.3. Linked solutions in the geospatial context

Even if the code and data are published, they are only parts of the whole computational environment required to conduct reproducible research. In this subsection, we focus on the right column of the central part of our spectrum of reproducibility (Figure 2).

Generally speaking, researchers are free to define their working environments. This implies that the number of unique environments and workflows depends only on the research requirements and limitations. From the reproducibility perspective, it is crucial to specify the execution environment correctly. Some steps are paramount concerning workflows and must be documented to allow understanding, execution and reproduction. One possible solution in Unix-like systems is to use *Makefiles* [83], which are used to compile outputs by executing commands according to the instructions given (targets, dependencies and receipts).

In GIScience, these workflows may vary and, as explained in Section 2.1, we can use different programming languages and development platforms. Workflows can mix software and data for different purposes, so complex workflows usually are expected. For example, a workflow can use a desktop GIS application, such as *QGIS*, to process data stored in a *GeoJSON* file and then the result will be published in a map server and consumed by a web mapping application. Although data, code and documentation may be available to recreate the entire computational environment, many researchers in non-computational disciplines (e.g., hydrology) will not necessarily have the time, experience or resources to replicate and maintain applications or libraries required by a software to work correctly [84], these are considered as *dependencies* in the workflow. Finding a solution capable of packaging and sharing execution environments that contain these workflows is, therefore, one of the primary purposes of ORR.

Packaging data, code and execution environments have already been described under the term of research compendiums [85,86]. The technology capable of making this possible is called "virtualisation", whereby it is conceivable to package all the workflows that are used to perform a

given research study. For instance, [87] lists the improvements that virtualisation and cloud computing can provide to reproducibility. Among them, we can highlight fewer constraints on research methods, virtual machines as publications, automatic upgrades, or sharing big data using cloud computing features.

Virtual Machines (VMs) can be used as a way of sharing scientific production [88]. VMs can run packaged code regardless of the researcher's computer, configurations and dependencies using a hypervisor [89]. They can host any workflow, from desktop software to web services or databases. However, most of the VM technologies have some drawbacks, but three of them seem more important: 1) a high level of system management knowledge is needed, 2) they usually take up a lot of storage space, and 3) depending of the chosen technology, they may be challenging to manage reliably in version control.

More recently, the concept of "containerisation" has appeared [90]. Containerisation is a lightweight and fast to launch alternative to complete machine virtualisation that involves an application in a container with its operating environment. Containers use the same kernel of the host Operating System (OS), instead of installing an OS for each VM. In this way, containers have an OS limitation marked by the host's kernel. Containers offer a way to isolate applications and provide a virtual platform to run applications. One of the advantages of containerisation over virtualisation is the better management of computer resources [91].

One of the best known containerization solutions is *Docker* [92]. *Docker* uses images to encapsulate already compiled and configured software (code *assets*) without users having to take the trouble of going through the so-called *dependency hell*. It does not depend on a hypervisor layer (distributes the resources to virtual machines) used in virtualisation. *Docker* images are shared using a central registry called *Docker Hub*, which makes the process of reusing these images even more practical. *Docker Hub* (hub.docker.com) provides a prebuilt image distribution service, adding metadata to allow discovery and subsequent downloading by others. Furthermore, *Docker* images are built from layers that are also reusable, thus reducing the number of building steps. Base *Docker* images are produced using *Dockerfiles*, which is a kind of *Makefile* or recipe with inheritance support for creating *Docker* containers.

Docker images can be built and run on a single machine or integrated application development and deployment solution called Platform as a Service (PaaS) using a *Docker* host. These images can be distributed cleanly to repositories, such as *Docker Hub* (see Figure 3). As we will see in Section 2, *Docker* can be used to share research scenarios and increase their transparency, portability, and reproducibility in the GISc world. Examples of these scenarios or use-cases are packaging algorithms, or object-based analysis of remote sensing images [93].

Apart from Docker, there are also compelling language-specific containerisation solutions. These are out the scope of this work because they have not been designed to handle general workflows involving diverse software components, such as the SIOSE-INNOVA project requires. However, even if these solutions were not going to be used, it is interesting to know that in different domains we can find specific solutions for these purposes. As in more general environments, an *R* environment can be reproduced if the correct versions of the interpreter and all packages, libraries, or extension modules are joined together. These dependencies quickly become complicated (the so-called *dependency hell*) so manual documentation is not feasible [32]. Several solutions have also emerged to solve these problems in the *R* language [32]. Some examples of them are *Packrat*, Renv or *containerit*. *Packrat* is an *R* extension [94] which facilitates the process of managing project dependencies. *Renv* [95] is an *R* package for managing reproducible environments that provides isolation, portability and pinned versions of packages. Finally, *containerit* [96] automates the generation of *Dockerfiles* for *R* workflows, based on images from the Rocker project [97].

The last practice included in our spectrum is literate programming [98]. This is not used regularly in the SIOSE-INNOVA project, where we focus more on conducting experiments with databases, but it can be useful for compiling always up-to-date versions of most documents derived from the project. One viral initiative used in the *Python* language is *Jupyter* notebooks [99]. *Jupyter* combines

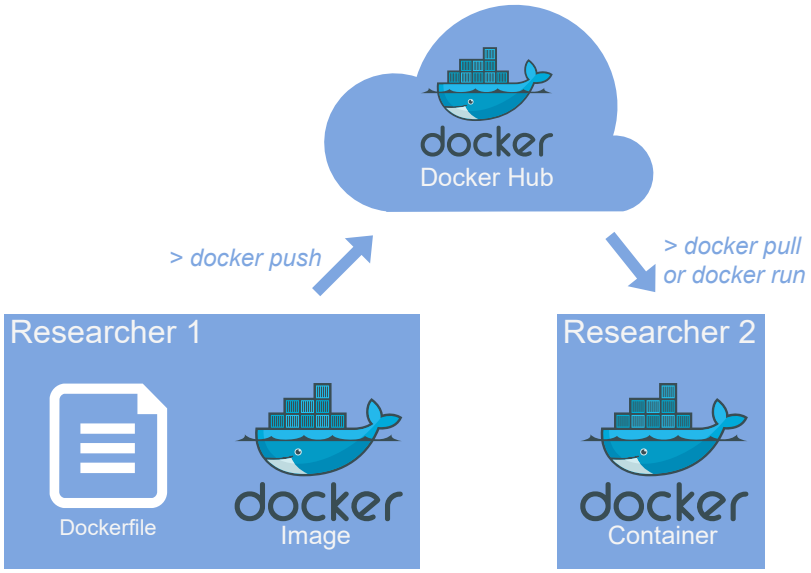

Figure 3. *Docker* Hub ecosystem and packaging of compiled containers.

comprehensive workflows with narrative text and visualisation of results in a transparent, collaborative, reproducible and reusable way. Similarly, in the *R* field, there are notebooks [100] to combine executable code, rendered visualisations and descriptive text into a single interactive, portable document. Another initiative is the interactive guidelines presented in [101], which aim to bring data and code together in a single web document (*Javascript* or *p5.js* [102]) and to perform the executions online.

In order to meet the requirements of SIOSE-INNOVA, we decided to use *Docker* containers to link data and code in a ready to use execution environment. The reason for this selection is due to the broad and general support, adding the possibility to package any piece of software and the workflows between them. As best practices in containerisation, it is necessary to define and share what kind of configuration, connections, dependencies and other details are present in the published containers.

3. A brief look at the adoption of containerisation in GISc projects

Before incorporating docker containers into the project, it was interesting to know how containers are being used in combination with some geospatial technologies. We were specially interested in how to containerise *PostgreSQL/PostGIS* for storing SIOSE data, database clients for performing queries and other tools for processing spatial data. A larger user community would provide a greater number of examples and more available resources. For example, a greater number of geospatial docker images ready to be used or extended. In order to evaluate the level of adoption of these technologies, we performed a systematic review to assess the presence of OSGeo software in the *Docker Hub registry*—the main repository of docker images.

OSGeo is a community of researchers focused on geospatial software development. Since its inception in 2006, OSGeo has been providing organisational support to build an active international community to advance geospatial technologies and solutions using free and open-source software (FOSS), open data and open standards. Under the umbrella of OSGeo, we can find a significant number of free and open-source tools in the field of geoinformatics. From the perspective of reproducibility, OSGeo guarantees free access to software (code). OSGeo projects are published under open licences that grant freedom for their use, adaptation and the redistribution of possible modifications. Moreover, software included in OSGeo can be installed and used without restrictions in any environment—commercial or non-commercial—without any associated cost.

Grouping by software categories, OSGeo promotes desktop GIS applications, libraries for handling map projections, geospatial data conversion tools, geodatabases, geostatistics libraries or web mapping

tools, among others. In the OSGeo website, this software is classified into seven different categories, which are: 1) *content management systems*, 2) *desktop applications*, 3) *geospatial libraries*, 4) *metadata catalogues*, 5) *spatial databases*, and 6) *web services*.

In order to perform the systematic review, we extracted the download statistics of *Docker* images (*pulls*) that included OSGeo tools in the *Docker Hub registry*. This review has been carried out by querying the *Docker Registry HTTP API v2* (https://docs.docker.com/registry/spec/api/).

The semiautomatic search consisted of a series of queries by image name. Thus, we only took into account images containing the name of any of a few selected OSGeo tools. This selection was made depending on whether the tool appeared as featured on the OSGeo website. Query results were sorted by the number of downloads (*pulls*) and limited to a maximum of 1000 results per search. A *Powershell* script and a *Docker* image to reproduce these queries are available in https://github.com/sergitrilles/OSGEO_docker and https://hub.docker.com/repository/docker/strilles/osgeo_docker respectively. After running the queries, we revised each returned row to be sure that it was an OSGeo tool. Finally, the results were grouped using the previously mentioned OSGeo software categories, and the total number of downloads by each tool was obtained. This process is described in detail below:

1. *Selection of OSGeo tools*. There are currently sixty-three different projects indexed on the OSGeo website, including OSGeo projects and community projects. Following the naming convention used on this website in determining featured tools that appears in the main menu, twenty-one tools have been selected from the seven categories defined in Table 1.

Category	**Tools**
Content Management Systems	*GeoNode*
Desktop Applications	*Marble*, *gvSIG Desktop*, *QGIS Desktop* and *GRASS GIS*
Geospatial Libraries	*GeoTools*, *Orfeo ToolBox*, *GDAL/OGR* and *GEOS*
Metadata Catalogues	*GeoNetwork* and *pycsw*
Spatial Databases	*PostGIS*
Web Mapping	*MapServer*, *MapFish*, *Deegree*, *OpenLayers*, *GeoMoose*, *Mapbender*, *PyWPS* and *GeoServer*

Table 1. List of the selected OSGeo tools considered in our review.

2. *Definition of searches*. Searches were defined using the names of the OSGeo tools as query terms. Each search returned the list of images stored in *Docker Hub*, ordered by descending number of downloads (pulls). A filter was applied to set a maximum of 1000 results in order to ignore residual *Docker* images and build the analysis on those that are widely used.
3. *Execution of searches*. Searches were executed by using the above-mentioned *Docker Registry API*. More specifically, we used a *PowerShell* module called *PSDockerHub* (https://github.com/beatcracker/PSDockerHub). The results were saved in a *CSV* file for each OSGeo project. From all the images, we obtained the name, description and number of downloads (pulls).
4. *Manual selection of results*. After the search step, a manual phase is performed to detect possible *Docker* images not related to OSGeo tools. To reach this goal, the returned image results have been verified using the description field to check whether an OSGeo tool is used.
5. *Aggregation and final results*. In the last step, an aggregation is performed on each tool to obtain a total number of downloads.

The results obtained show that the OSGeo category with the highest number of downloads is *spatial databases*, and *PostGIS* is the most *pulled* software with 35,267,083 pulls. The next software with the highest number of downloads is *GeoNetwork* with 2,983,271 within the category of metadata catalogues, followed by *GeoServer* (*web mapping*) and *GDAL/OGR* (*spatial libraries*) with 1,731,538 and 1,653,481, respectively. The *desktop applications* category has the lowest number of downloads among its software options, which is quite normal because *Docker* containers are not initially intended for

Category	Tool	Downloads
Content Management Systems	*GeoNode*	873,869
Desktop Applications	*GRASS GIS*	317
	gvSIG	0
	Marble	0
	QGIS	392,882
Geospatial Libraries	*GDAL/OGR*	1,653,481
	GEOS	101,165
	GeoTools	2,437
	Orfeo Toolbox	1,736
Metadata Catalogues	*GeoNetwork*	2,983,271
	pycsw	482,912
Spatial Databases	*PostGIS*	35,267,083
Web Mapping	*geomoose*	0
	Degree	67
	Geoserver	1,731,538
	MapBender	986
	MapFish	0
	MapServer	1,170,538
	OpenLayers	320,234
	pywps	1,162

Table 2. Number of downloads for each OSGeo tool in *Docker Hub*. All results are published in *Zenodo* [103].

this purpose. These results show that *PostGIS* is a widely used spatial DBMS in different contexts, while other projects of interest for the SIOSE-INNOVA project have a lower but also important level of adoption (e.g., *GDAL*, *QGIS*).

4. Enabling research reproducibility in the SIOSE-INNOVA project

As a response to the ideas argued in this paper, the SIOSE-INNOVA project made use of several open tools and platforms that were previously mentioned in this manuscript and in the GISc reproducibility spectrum proposed in Figure 2. In sum, different versions and configurations of *PostgreSQL* and *PostGIS* were used for database management, *Geopackages* and *GeoJSON* were our preferred interchange geospatial formats, *GitHub* was used as collaborative development platform and GNU Make was used to compile different database extensions and deliverables (see Section 2). Finally, containerisation tools (*Docker* and *Docker Compose*) have been used to deploy different workflows, from the simplest to relatively complex ones. In all cases, the necessary tasks have been automated and a second researcher has been able to reproduce the results on a different computer.

In the following subsections, we explain two different cases where using *Docker* containers made a difference, and we propose the adoption of Unified Modeling Language (UML) diagrams for documenting and visually explaining how *Docker* helped in making some computational experiences more reproducible. These two experiences were developed on a personal computer running Linux *Ubuntu 18.04.3 LTS* (Linux 4.15.0-66-generic) with *Docker* (17.05.0-ce, build 89658be), *docker-compose* (1.11.2, build dfed245), *GNU Make* (v.4.1) and *Bash* (4.4.20(1)-release (x86_64-pc-linux-gnu)). All the produced code repositories can be found in the project's *GitHub* profile (https://github.com/siose-innova) and a few container images were stored in *Docker Hub* (https://hub.docker.com/orgs/sioseinnova/repositories).

4.1. Benchmarking SQL and NoSQL database models

A sophisticated computational experience using SIOSE data was described in detail in [24]. In this case, developed in 2016, an experiment was carried out in which it was verified that specific queries to the SIOSE database could be expedited if the data were stored in a document-oriented database

(document store). The experiment consisted in downloading the SIOSE-2005 regional databases, joining them in a single instance, cleaning and renaming tables and columns, duplicating and converting the data to *JSON* format (*JSONB*) and running a battery of thousands of queries on two databases (one relational and the other document-oriented). Despite the complexity of this workflow, a user with minimal instructions would be able to reproduce the experiment (see Figure 4).

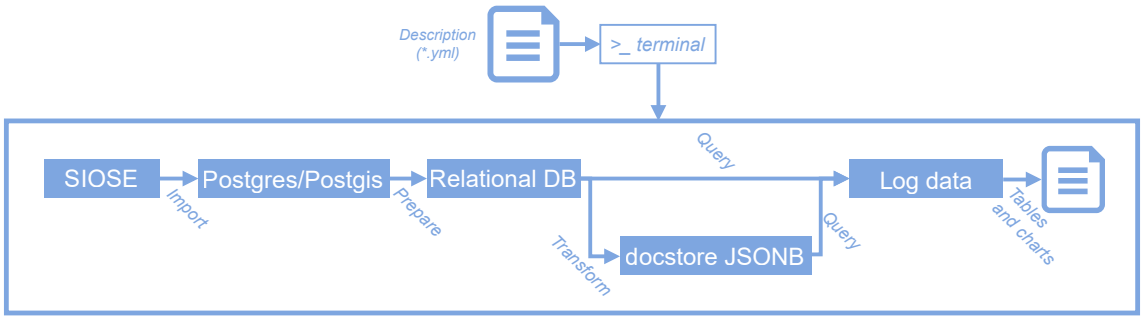

Figure 4. Computational experience in the SIOSE-INNOVA project [24].

It should be mentioned that since the experiment was published, we have been able to reproduce it several times without finding any drawbacks related to the reproducibility issues mentioned above (*dependency hell*). In this way, it can be considered that this research is reproducible, according to the concepts proposed in the previous sections (see Figure 2).

The base *Docker* image used in this experiment was built in a personal computer, and then it was uploaded to the *Docker Hub registry* (https://hub.docker.com/repository/docker/labgeo/siose_2005). This image can be used as a base image for developing any computational experience over the SIOSE database in a *PostgreSQL/PostGIS* environment. The complete image—containing code, data and environment configuration—weights 10.5GB, which may not seem very impressive but, when trying to reproduce the experiment, this intermediate step can save up to 5 hours of computing time on a personal computer. This was the time required for downloading and building the SIOSE database from *ESRI personal Geodatabase* files and load that data into a *PostgreSQL/PostGIS* database. This image is the base for a *Docker Hub* automated build. All these resources are available in the same repository, together with all the scripts of this experiment (see Figure 4) [104].

4.2. An ETL process using containerisation

In this use-case, we used *Docker* containers and other elements presented in Section 2 to prepare a reproducible ETL workflow (Extract-Transform-Load). In order to split the SIOSE database into small portable GIS files, we wrote all the necessary scripts (*Bash*, *Makefile* or spatial-*SQL*) and designed them to deploy all the necessary tools [104]. This workflow is shown in more detail in the UML activity diagram in Figure 5. The proposed diagram separates those tasks that need to be performed by the researcher and those that were completely automated. The steps can be deployed by using a Makefile and are enumerated as follows:

1. The user launches the workflow using several *docker run* commands or one single execution of the deployment software (*docker-compose up*).
2. *Docker* starts all the necessary services sequentially (see Figure 6).
3. A *PostgreSQL/PostGIS* server is started which already contains a preloaded SIOSE database.
4. *SQL* scripts are executed from the *Makefile*, creating spatial grids and statistics that will be used to split the database into GIS portable files. *GNU Make* will ensure that all files are created, and it will prevent any unnecessary overwrite operations from being carried out on any file.
5. Then, the process loops over the cells of several spatial grids at different mapping scales.
6. In every iteration, a spatial query is executed and intersects the SIOSE database with the corresponding area of interest (cell grid).

7. Using the *GDAL/OGR* library, the selected SIOSE polygons together with a selection of LU/LC attributes are written into a new *Geopackage* file.
8. This new *Geopackage* is compressed and stored in the working directory.
9. Then the user can inspect the database or check the produced outputs using a containerised version of *PGadmin4* or *QGIS*.
10. When the user stops the process, *Docker* stops all services and frees all the unused computing resources.

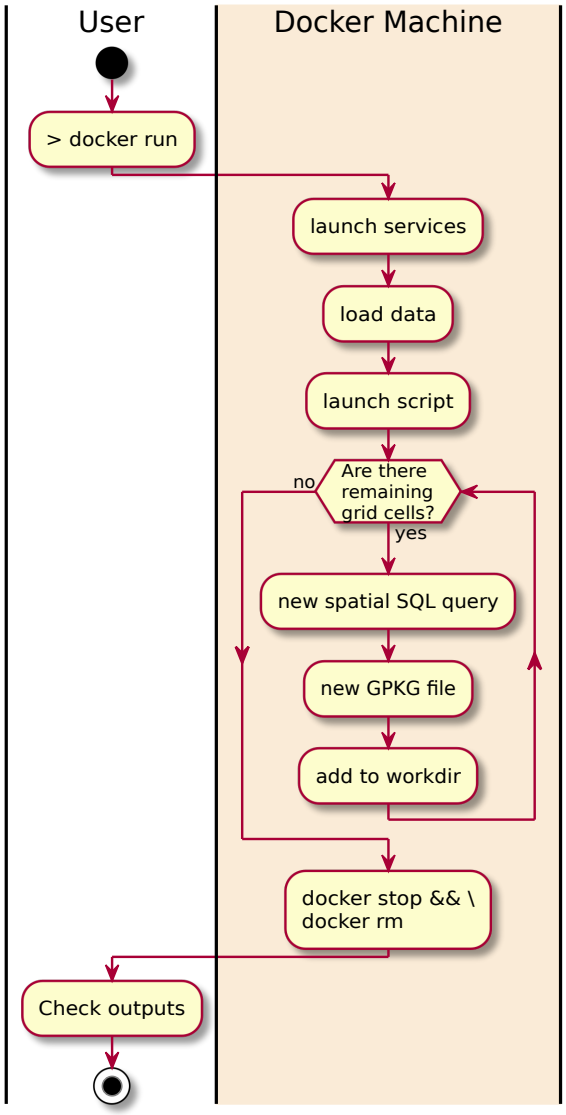

Figure 5. UML activity diagram showing all the tasks automated in the SIOSE-splitter solution.

Launching this workflow should be straightforward for researchers familiar with these technologies. However, when it comes to adapting it for other purposes, the process may become far more complicated (e.g., using other database versions or different output formats, or testing new software configurations, among other possibilities). There are several scripts, software extensions, *Makefiles*, queries and *Dockerfiles* that have been developed specifically for this workflow. Moreover, due to the *Docker* inheritance mechanisms mentioned in Section 2.3, many configuration options or resources could be hidden in the development process. That being said, this workflow is more complicated at this level of detail. For this reason, we propose using UML diagrams to document such complex deployments.

The UML components diagram in Figure 6 shows the static part of the deployment required for this workflow to be run. The correct use of these diagrams for documentation can facilitate the understanding of essential parts of the working environment without the need to revise all the dependencies systematically. In this diagram, we show what happens inside and outside the *Docker* environment. As an example of hidden configurations, in this diagram, we are showing ports and software dependencies. This aspect is essential because these details are not available in our *Dockerfiles*, since they are in other base images from which they inherit. One example of these details is the port 5433 added to map a *PostgreSQL* server in Figure 6. This application usually uses 5432 as a default TCP port. However, using this port in a container could become in conflict with a *PostgreSQL* server running on the host machine. In this case, the diagram (Figure 6) shows that the container port is mapped to avoid such type of problem.

According to the diagram of the components, the user has many possibilities to inspect the workflow and its outputs without knowing anything about the operating systems and configurations running under the surface: 1) use the system console to execute new *Docker* commands; 2) check the produced geopackages transparently in a folder mounted in the host machine; 3) connect a *Postgres* client using port 5433; 4) make use of a *PgAdmin4* web client running on port 5050; and 5) use a dockerised version of *QGIS* both for exploring new GIS files or for connecting to the SIOSE database running in the *Docker* container. As can be seen from the number of *QGIS* pulls in *Docker Hub* (see Table 2), using a containerised version of *QGIS* is a practice that may have some interest. Moreover, it is interesting to mention that, as can be seen from the component diagram, it would be easy to launch more than one instance of *QGIS* in its different versions. This approach can be interesting both in development tasks and in more routine tasks in which having a specific *QGIS* version may be necessary. The *QGIS* API documentation can be useful for checking major changes that could affect reproducibility (https://qgis.org/api/api_break.html).

5. Concluding remarks

The challenges of reproducible science are systemic and generally costly to address. However, there is a consensus among researchers on the need to improve collaborative work and scientific rigour to overcome these obstacles. The latest experiences in different fields recommend the use of open data, open software and reproducible workflows. In recent years, many new resources have become popular and have reached a sufficient level of maturity to be applied in GISc projects.

Once the need for higher reproducibility in research is accepted, more and more researchers are encouraged to incorporate new tools and good practices into their projects. In this paper, we have described the necessary decision-making to make the SIOSE-INNOVA project more reproducible, selecting OSGeo tools, choosing standards-based files and trying to explain our research in the best possible way. This includes open-source code and data with a good description of workflows, among other desirable features. This concern for reproducibility began internally within the project—all researchers in the SIOSE-INNOVA project should be able to reproduce the work of other colleagues—but it has led to features that can be beneficial even after the project ends.

In the process, we have learned that there are currently many resources that can be used to increase the reproducibility of a GIScience investigation. Therefore, it is essential to know the different options available and know what they contribute to the research. Given the complexity of this scenario, we have tried to organise these elements in a diagram based on the spectrum of reproducibility proposed by Peng [3]. This schema has been described throughout Section 2. The relationships between tools and practices of such a diverse nature are complex to be described in a single diagram, so there is still room for improvement to develop a more comprehensive and precise representation, maybe a GIScience ORR map. However, even with all the qualifications that can be made, our spectrum of reproducibility in GISc has allowed us to realise that, to a greater or lesser extent, it is possible to improve the reproducibility of some computational experiences without significantly deviating from the original objectives of the project. Even before considering reproducibility as a cross-cutting objective of the

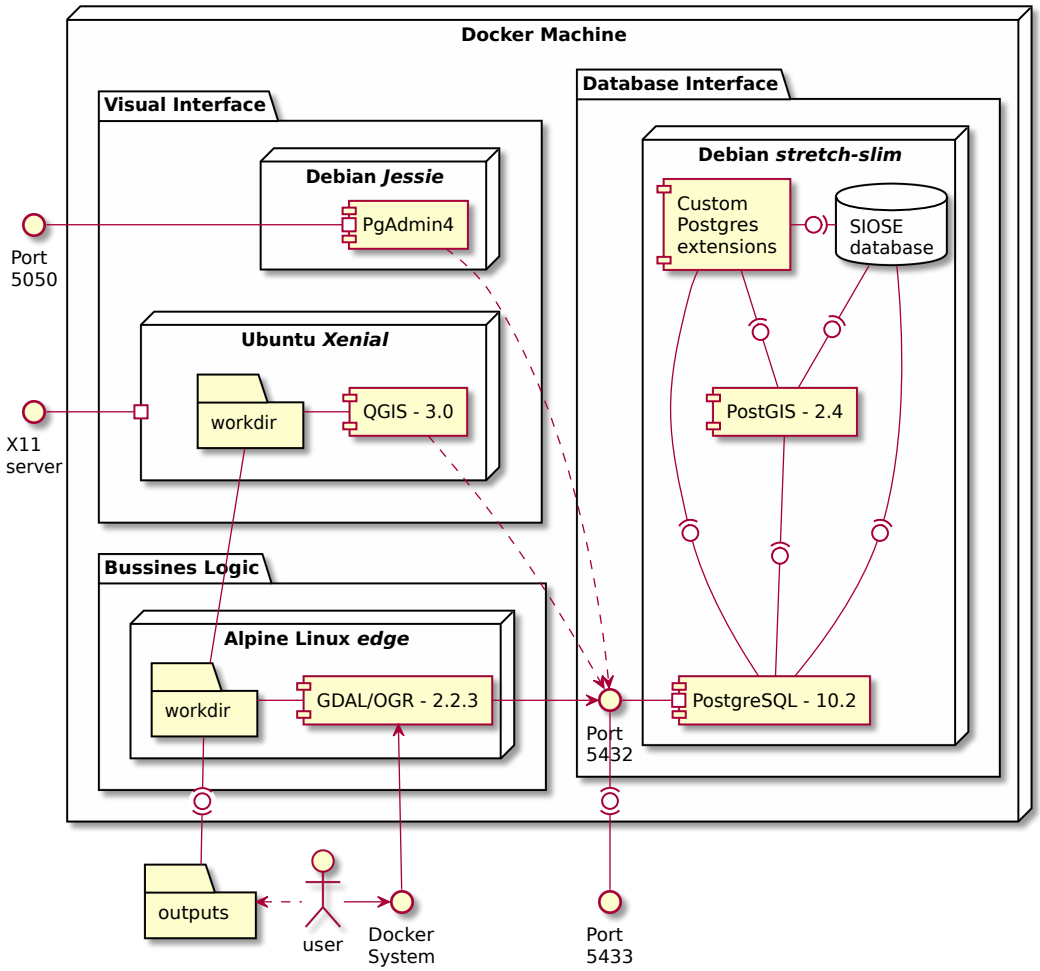

Figure 6. UML components diagram showing the containers and configurations used in the SIOSE-splitter solution.

SIOSE-INNOVA project, some techniques, standards and practices that favour reproducibility were already being applied. For example, the use of FOSS GIS for economic reasons, the publication of SIOSE data with an open license due to legal constraints, the distribution of data in standard and well-known formats or the need for collaborative work among members of the same research team. All these issues also have their advantages in terms of reproducibility of research. Other elements were new to the project researchers or involved an additional effort to be used in some computational experiences. The most prominent case has been the use of *Dockers* to package workflows of a certain level of complexity and be able to distribute them to the other members of the project (or external researchers).

Before applying Dockers to our project and to better understand the level of adoption of containerisation technologies in GIScience, we carried out a systematic review of the official *Docker registry (Docker Hub)*. In section 3, we measured the presence of the most popular geospatial solutions supported by the OSGeo community in *Docker Hub*. Our results show that the most frequently attended software category is spatial databases using *PostGIS* software, followed by the metadata catalogue category, where *GeoNetwork* is the most recurrent option. Other containerised tools like *QGIS* were of interest in achieving our objectives. The level of adoption of these resources—containerised OSGeo projects—is considerable, which means that there are many examples and use cases available, making it easier to start using them. Of course, this is a shallow review that should be better addressed in the near future. There are many questions to answer about the level of adoption of containers in GISc

(e.g., user census and profiles, pull trends, the purpose of the images stored in *Docker Hub*, diversity of environments, among other interesting topics).

Once we defined our working environment for the SIOSE-INNOVA project, we performed some computational experiences that are described in Section 4. During these experiences, we used different technology stacks to achieve a high level of reproducibility (e.g., version control, open geospatial libraries, containerisation platforms). The tools used and the proposed workflows make up one solution among many other possibilities. As mentioned above, one of the main problems that may arise is that the tools mentioned and other aspects of reproducible research are not part of the processes currently used by researchers. Although simple in general, these tools entail additional training for everyone involved and imply coordinated tasks that can eventually become more complex. This extra burden for researchers is not directly rewarded and, as noted, the pressure to publish research may be so high that there is no room to try to introduce these improvements [8]. However, attempting to conduct reproducible research, as shown in the GISc reproducibility spectrum, is not only related to sophisticated technologies, but also to paying attention to certain decisions, such as choosing an appropriate code and data licence, or trying to make the necessary effort in documenting the research. For example, some first steps could be to write *README* files or to create concise diagrams to explain the purpose of the research in a better way. As a minor contribution of this work, we propose how to describe a dockerised workflow using UML diagrams.

Derived from our previous considerations, it must be emphasised that, in this work, only the technological approach has been evaluated for the problem of low scientific reproducibility. However, the issue as a whole needs to be addressed more comprehensively, as solutions depend on a change in the attitude of the scientific community. As can be seen from our GISc spectrum of reproducibility, there are many tools and mechanisms to promote research reproducibility – regardless of whether they are specific to the geospatial domain or inherited from other disciplines – and so trying to conduct reproducible research or achieving is highly related to the level of commitment [30]. For future work, we intend to keep applying best practices required to achieve a better level of reproducible research.

Author Contributions: Conceptualisation, Sergio Trilles and Benito Zaragozí; Formal analysis, Sergio Trilles; Funding acquisition, Sergio Trilles; Methodology, Benito Zaragozí; Software, José Tomás Navarro-Carrión and Benito Zaragozí; Supervision, Benito Zaragozí; Writing – original draft, Sergio Trilles and Benito Zaragozí; Writing – review & editing, José Tomás Navarro-Carrión.

Funding: This work has been funded by the Generalitat Valenciana through the "Subvenciones para la realización de proyectos de I+D+i desarrollados por grupos de investigación emergentes" programme (GV/2019/016) and by the Spanish Ministry of Economy and Competitiveness under the subprogrammes Challenges-Collaboration 2014 (RTC-2014-1863-8) and Challenges R+D+I 2016 (CSO2016-79420-R AEI/FEDER, EU). Sergio Trilles has been funded by the postdoctoral programme PINV2018 - Universitat Jaume I (POSDOC-B/2018/12) and stays programme PINV2018 - Universitat Jaume I (E/2019/031).

1. Meng, H.; Kommineni, R.; Pham, Q.; Gardner, R.; Malik, T.; Thain, D. An invariant framework for conducting reproducible computational science. *Journal of Computational Science* **2015**, *9*, 137–142. doi:10.1016/j.jocs.2015.04.012.
2. Barba, L.A. Terminologies for reproducible research. *arXiv preprint arXiv:1802.03311* **2018**.
3. Peng, R.D. Reproducible research in computational science. *Science* **2011**, *334*, 1226–1227.
4. Stodden, V. Reproducible research for scientific computing: Tools and strategies for changing the culture. *Computing in Science & Engineering* **2012**, *14*, 13.
5. Brunsdon, C. Spatial science - Looking outward. *Dialogues in Human Geography* **2014**, *4*, 45–49. doi:10.1177/2043820614525709.
6. Gil, Y.; David, C.H.; Demir, I.; Essawy, B.T.; Fulweiler, R.W.; Goodall, J.L.; Karlstrom, L.; Lee, H.; Mills, H.J.; Oh, J.H.; Pierce, S.A.; Pope, A.; Tzeng, M.W.; Villamizar, S.R.; Yu, X. Toward the Geoscience Paper of the Future: Best practices for documenting and sharing research from data to software to provenance. *Earth and Space Science* **2016**, *3*, 388–415. doi:10.1002/2015EA000136.

7. Begley, C.G.; Ioannidis, J.P.A. Reproducibility in science: Improving the standard for basic and preclinical research. *Circulation Research* **2015**, *116*, 116–126. doi:10.1161/CIRCRESAHA.114.303819.
8. Baker, M. Is there a reproducibility crisis? *Nature* **2016**, *533*, 452–454. doi:10.1038/533452a.
9. Freedman, L.P.; Cockburn, I.M.; Simcoe, T.S. The economics of reproducibility in preclinical research. *PLoS Biology* **2015**, *13*, 1–9. doi:10.1371/journal.pbio.1002165.
10. Reinhart, C.M.; Rogoff, K.S. Growth in a time of debt. *American Economic Review* **2010**, *100*, 573–578. doi:10.1257/aer.100.2.573.
11. Herndon, T.; Ash, M.; Pollin, R. Does high public debt consistently stifle economic growth? A critique of Reinhart and Rogoff. *Cambridge Journal of Economics* **2014**, *38*, 257–279. doi:10.1093/cje/bet075.
12. Open Science Collaboration. Estimating the reproducibility of psychological science. *Science* **2015**, *349*, aac4716–aac4716, [arXiv:1011.1669v3]. doi:10.1126/science.aac4716.
13. Peng, R.D. Reproducible Research in Computational Science. *Science* **2011**, *334*, 1226–1227, [0901.4552]. doi:10.1126/science.1213847.
14. Stodden, V. *Implementing Reproducible Research*; Chapman and Hall/CRC, 2014; p. 440. doi:10.1201/b16868.
15. Singleton, A.D.; Spielman, S.; Brunsdon, C. Establishing a framework for Open Geographic Information science. *International Journal of Geographical Information Science* **2016**, *8816*, 1–15. doi:10.1080/13658816.2015.1137579.
16. Goodchild, M.F. Geographical information science. *International journal of geographical information systems* **1992**, *6*, 31–45.
17. Ostermann, F.O.; Granell, C. Advancing Science with VGI: Reproducibility and Replicability of Recent Studies using VGI. *Transactions in GIS* **2017**, *21*, 224–237. doi:10.1111/tgis.12195.
18. Goodman, S.N.; Fanelli, D.; Ioannidis, J.P. What does research reproducibility mean? *Science translational medicine* **2016**, *8*, 341ps12–341ps12.
19. Pontika, N.; Knoth, P.; Cancellieri, M.; Pearce, S. Fostering open science to research using a taxonomy and an eLearning portal. Proceedings of the 15th international conference on knowledge technologies and data-driven business. ACM, 2015, p. 11.
20. Antelman, K. Do Open Access Articles Have a Greater Research Impact? *College & Research Libraries* **2004**, *65*. doi:10.5860/crl.65.5.372.
21. Murray-Rust, P. Open data in science. *Serials Review* **2008**, *34*, 52–64.
22. Perens, B.; others. The open source definition. *Open sources: voices from the open source revolution* **1999**, *1*, 171–188.
23. CNIG download center. http://centrodedescargas.cnig.es. Accessed: 2019-12-15.
24. Navarro-Carrión, J.T.; Zaragozí, B.; Ramón-Morte, A.; Valcárcel-Sanz, N. Should EU land use and land cover data be managed with a NOSQL document store? *International Journal of Design & Nature and Ecodynamics* **2016**, *11*, 438–446. doi:10.2495/DNE-V11-N3-438-446.
25. Giraud, T.; Lambert, N. Reproducible cartography. International Cartographic Conference. Springer, 2017, pp. 173–183.
26. Konkol, M.; Kray, C.; Pfeiffer, M. Computational reproducibility in geoscientific papers: Insights from a series of studies with geoscientists and a reproduction study. *International Journal of Geographical Information Science* **2019**, *33*, 408–429.
27. Ostermann, F.O.; Granell, C. Advancing science with VGI: Reproducibility and replicability of recent studies using VGI. *Transactions in GIS* **2017**, *21*, 224–237.
28. Granell, C.; Nüst, D.; Ostermann, F.O.; Sileryte, R. Reproducible Research is like riding a bike. Technical report, PeerJ Preprints, 2018.
29. Association of Geographic Information Laboratories in Europe (AGILE) website. https://agile-online.org/. Accessed: 2019-12-15.
30. Nüst, D.; Granell, C.; Hofer, B.; Konkol, M.; Ostermann, F.O.; Sileryte, R.; Cerutti, V. Reproducible research and GIScience: an evaluation using AGILE conference papers. *PeerJ* **2018**, *6*, e5072.
31. Skaggs, T.; Young, M.H.; Vrugt, J. Reproducible research in vadose zone sciences. *Vadose Zone Journal* **2015**, *14*.
32. Nüst, D.; Konkol, M.; Pebesma, E.; Kray, C.; Schutzeichel, M.; Przibytzin, H.; Lorenz, J. Opening the publication process with executable research compendia. *D-Lib Magazine* **2017**, *23*.

33. Barga, R.S.; Simmhan, Y.L.; Chinthaka, E.; Sahoo, S.S.; Jackson, J.; Araujo, N. Provenance for Scientific Workflows Towards Reproducible Research. *IEEE Data Eng. Bull.* **2010**, *33*, 50–58.
34. BitBucket. https://www.bitbucket.com. Accessed: 2019-12-15.
35. GitLab. https://www.gitlab.com. Accessed: 2019-12-15.
36. Steiniger, S.; Hunter, A.J. The 2012 free and open source GIS software map - A guide to facilitate research, development, and adoption. *Computers, Environment and Urban Systems* **2013**, *39*, 136–150. doi:10.1016/j.compenvurbsys.2012.10.003.
37. Steiniger, S.; Bocher, E. An overview on current free and open source desktop GIS developments. *International Journal of Geographical Information Science* **2009**, *23*, 1345–1370. doi:10.1080/13658810802634956.
38. Index, T. TIOBE-The Software Quality Company. *TIOBE Index| TIOBE–The Software Quality Company [Electronic resource]. Mode of access: https://www. tiobe. com/tiobe-index/-Date of access* **2018**, *1*.
39. OSGeo. http://www.osgeo.org. Accessed: 2019-12-15.
40. Team, Q.D.; others. QGIS geographic information system. *Open Source Geospatial Foundation Project, Versão* **2015**, *2*.
41. Hazzard, E. *Openlayers 2.10 beginner's guide*; Packt Publishing Ltd, 2011.
42. Leaflet JS. https://leafletjs.com. Accessed: 2019-12-15.
43. PostGIS. https://postgis.net. Accessed: 2019-12-15.
44. Geoserver. http://geoserver.org. Accessed: 2019-12-15.
45. Geonetwork. http://geonetwork.org. Accessed: 2019-12-15.
46. Solutions, V. JTS Topology Suite: Technical Specifications. *Version* **2003**, *1*, 36.
47. Neteler, M.; Bowman, M.H.; Landa, M.; Metz, M. GRASS GIS: A multi-purpose open source GIS. *Environmental Modelling & Software* **2012**, *31*, 124–130.
48. MapServer. http://www.saga-gis.org. Accessed: 2019-12-15.
49. GDAL. https://gdal.org. Accessed: 2019-12-15.
50. Orfeo ToolBox. https://www.orfeo-toolbox.org. Accessed: 2019-12-15.
51. Ram, K. Git can facilitate greater reproducibility and increased transparency in science. *Source code for biology and medicine* **2013**, *8*, 7.
52. Van Garderen, P. Archivematica: Using Micro-Services And Open-Source Software To Deliver A Comprehensive Digital Curation Solution. iPRES. Citeseer, 2010.
53. Faria, L.; Ferreira, M.; Castro, R.; Barbedo, F.; Henriques, C.; Corujo, L.; Ramalho, J.C. RODA: a service-oriented repository to preserve authentic digital objects. 4th International Conference on Open Repositories, 2009.
54. Whyte, A.; Pryor, G. Open Science in Practice: Researcher Perspectives and Participation. *IJDC* **2011**, *6*, 199–213.
55. Benitez-Paez, F.; Comber, A.; Trilles, S.; Huerta, J. Creating a conceptual framework to improve the re-usability of open geographic data in cities. *Transactions in GIS* **2018**, *22*, 806–822.
56. Stodden, V. The legal framework for reproducible scientific research: Licensing and copyright. *Computing in Science & Engineering* **2008**, *11*, 35–40.
57. ESRI Shapefile Technical Description. https://www.esri.com/library/whitepapers/pdfs/shapefile.pdf. Accessed: 2019-12-15.
58. Stults, M.; Arko, R.; Davis, E.; Ertz, D.; Turner, M.; Trabant, C.; Valentine Jr, D.; Ahern, T.; Carbotte, S.; Gurnis, M.; others. GeoCSV: tabular text formatting for geoscience data. AGU Fall Meeting Abstracts, 2015.
59. Butler, H.; Daly, M.; Doyle, A.; Gillies, S.; Hagen, S.; Schaub, T.; others. The geojson format. *Internet Engineering Task Force (IETF)* **2016**.
60. SpatiaLite webpage. https://www.gaia-gis.it/fossil/libspatialite/index. Accessed: 2019-12-15.
61. OGC GeoPackage. https://www.geopackage.org. Accessed: 2019-12-15.
62. Open Geospatial Consortium. https://www.opengeospatial.org/standards. Accessed: 2019-12-15.
63. netCDF. https://cdn.earthdata.nasa.gov/conduit/upload/497/ESDS-RFC-022v1.pdf. Accessed: 2019-12-15.
64. Berners-Lee, T. Linked data, 2006.
65. Bizer, C.; Lehmann, J.; Kobilarov, G.; Auer, S.; Becker, C.; Cyganiak, R.; Hellmann, S. DBpedia-A crystallization point for the Web of Data. *Journal of web semantics* **2009**, *7*, 154–165.

66. Percivall, G. Progress in OGC web services interoperability development. In *Standard-based data and information systems for earth observation*; Springer, 2010; pp. 37–61.
67. Giuliani, G.; Camara, G.; Killough, B.; Minchin, S. Earth Observation Open Science: Enhancing Reproducible Science Using Data Cubes, 2019.
68. de la Beaujardiere, J., Ed. *OpenGIS Web Map Service Implementation Specification, Version 1.3.0*; Open Geospatial Consortium Inc., 2006; pp. 85+.
69. Vretanos, P.A. Web feature service implementation specification. *Open Geospatial Consortium Specification* **2005**, pp. 04–094.
70. Bröring, A.; Stasch, C.; Echterhoff, J. OGC Interface Standard 10-037: SOS 2.0 Interface Standard. *Open Geospatial Consortium* **2010**.
71. Nebert D., W.A. OpenGIS catalogue services specification (version 2.0). *OpenGIS Project Document 04-021r2. Open GIS Consortium Inc* **2004**.
72. Schut, P. Opengis web processing service version 1.0.0. Technical report, Open Geospatial Consortium (OGC), 2008.
73. Nüst, D.; Stasch, C.; Pebesma, E. Connecting R to the sensor web. In *Advancing Geoinformation Science for a Changing World*; Springer, 2011; pp. 227–246.
74. zenodo. https://www.zenodo.org. Accessed: 2019-12-15.
75. figshare. https://www.figshare.com. Accessed: 2019-12-15.
76. dspace. https://www.duraspace.org. Accessed: 2019-12-15.
77. ckan. https://ckan.org. Accessed: 2019-12-15.
78. Austin, C.C.; Brown, S.; Fong, N.; Humphrey, C.; Leahey, A.; Webster, P. Research data repositories: review of current features, gap analysis, and recommendations for minimum requirements. *IASSIST Quarterly* **2016**, *39*, 24–24.
79. Creative Commons. https://creativecommons.org/licenses. Accessed: 2019-12-15.
80. Open Data Commons. https://opendatacommons.org/licenses. Accessed: 2019-12-15.
81. Benitez-Paez, F.; Degbelo, A.; Trilles, S.; Huerta, J. Roadblocks hindering the reuse of open geodata in Colombia and Spain: A data user's perspective. *ISPRS International Journal of Geo-Information* **2018**, *7*, 6.
82. Trilles, S.; Díaz, L.; Huerta, J. Approach to facilitating geospatial data and metadata publication using a standard geoservice. *ISPRS International Journal of Geo-Information* **2017**, *6*, 126.
83. Mecklenburg, R. *Managing Projects with GNU Make: The Power of GNU Make for Building Anything*; " O'Reilly Media, Inc.", 2004.
84. Hutton, C.; Wagener, T.; Freer, J.; Han, D.; Duffy, C.; Arheimer, B. Most computational hydrology is not reproducible, so is it really science? *Water Resources Research* **2016**, *52*, 7548–7555.
85. Chirigati, F.; Rampin, R.; Shasha, D.; Freire, J. Reprozip: Computational reproducibility with ease. Proceedings of the 2016 International Conference on Management of Data. ACM, 2016, pp. 2085–2088.
86. Stodden, V.; Miguez, S.; Seiler, J. Researchcompendia. org: Cyberinfrastructure for reproducibility and collaboration in computational science. *Computing in Science & Engineering* **2015**, *17*, 12.
87. Howe, B. Virtual appliances, cloud computing, and reproducible research. *Computing in Science & Engineering* **2012**, *14*, 36–41.
88. Dua, R.; Raja, A.R.; Kakadia, D. Virtualization vs containerization to support paas. 2014 IEEE International Conference on Cloud Engineering. IEEE, 2014, pp. 610–614.
89. Soltesz, S.; Pötzl, H.; Fiuczynski, M.E.; Bavier, A.; Peterson, L. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors. ACM SIGOPS Operating Systems Review. ACM, 2007, Vol. 41, pp. 275–287.
90. Turnbull, J. *The Docker Book: Containerization is the new virtualization*; James Turnbull, 2014.
91. Meadusani, S.R. Virtualization Using Docker Containers: For Reproducible Environments and Containerized Applications. *Culminating Projects in Information Assurance* **2018**.
92. Merkel, D. Docker: lightweight linux containers for consistent development and deployment. *Linux Journal* **2014**, *2014*, 2.
93. Knoth, C.; Nüst, D. Reproducibility and practical adoption of geobia with open-source software in docker containers. *Remote Sensing* **2017**, *9*, 290.
94. Packrat - Introduction to renv.
95. Ushey, K. Renv - Introduction to renv.

96. Nüst, D.; Hinz, M. containerit: Generating Dockerfiles for reproducible research with R. *The Journal of Open Source Software* **2019**, *4*.
97. Boettiger, C.; Eddelbuettel, D. An introduction to rocker: Docker containers for R. *arXiv preprint arXiv:1710.03675* **2017**.
98. Knuth, D.E. Literate programming. *The Computer Journal* **1984**, *27*, 97–111.
99. Kluyver, T.; Ragan-Kelley, B.; Pérez, F.; Granger, B.E.; Bussonnier, M.; Frederic, J.; Kelley, K.; Hamrick, J.B.; Grout, J.; Corlay, S.; others. Jupyter Notebooks-a publishing format for reproducible computational workflows. ELPUB, 2016, pp. 87–90.
100. Hillebrand, J.; Nierhoff, M.H. *Mastering RStudio–Develop, Communicate, and Collaborate with R*; Packt Publishing Ltd, 2015.
101. Trilles, S.; Granell, C.; Degbelo, A.; Bhattacharya, D. Interactive Guidelines: Public Communication of Data-based Research in Cities. *Plos One (in press)* **2020**. doi:http://10.1371/journal.pone.0228008.
102. p5js. https://p5js.org. Accessed: 2019-12-15.
103. Trilles, S. OSGeo tools in Docker Hub, 2019. doi:10.5281/zenodo.3615855.
104. Zaragozí, B.M.; Carrión, J.T.N. siose-innova/pg_siose_bench, 2020. doi:10.5281/zenodo.3631364.